IBM

SC21-7731-5

File No. S38-36

IBM System/38

IBM System/38 Control Language Reference Manual

Program Number 5714-SS1

SC21-7731-5

File No. S38-36

IBM System/38

IBM System/38 Control Language Reference Manual

Program Number 5714-SS1

Sixth Edition (September 1982)

This is a major revision of, and obsoletes, SC21-7731-4 and Technical Newsletter SN21-8235. This edition applies to release 4, modification 1 of the IBM System/38 Control Program Facility (Program 5714-SS1), and to all subsequent releases until otherwise indicated in technical newsletters or in new editions.

Changes or additions to the text, syntax diagrams, and illustrations are indicated by a vertical line to the left of the change or addition.

Changes are periodically made to the information herein; changes will be reported in technical newsletters or in new editions of this publication.

Contents

About This Manual

PURPOSE OF THIS MANUAL

This document is intended for use as a reference manual to assist the System/38 programmer, data processing manager, and system operator in using the control language commands. The System/38 user uses the control language commands to request functions of the system's Control Program Facility (CPF) and of the various languages and utilities.

This manual does *not* describe all of the functions of CPF or of the languages and utilities. That information can be found in the manuals listed under the section *If You Need More Information.*

ORGANIZATION OF THIS MANUAL

This publication is divided into three parts, consisting of five chapters and six appendixes.

Part 1 contains the following:

- Chapter 1 identifies (in chart form) the functions performed by the control language commands, and introduces the types of CPF objects used by the commands.
- Chapter 2 describes the control language syntax.

Part 2 contains the following:

- Chapter 3 explains the format used to describe control language commands.
- Chapter 4 describes every control language command, including commands for CPF and commands for languages and utilities.
- Chapter 5 describes the statements used for defining commands.

Part 3 contains the following:

- Appendix A describes in further detail a number of control language parameters.
- Appendix B describes the expressions and built-in functions used in control language programs.
- Appendix C identifies which IBM-supplied user profiles are authorized to use each command.
- Appendix D provides a cross-reference between commands and IBM-supplied data base and device files used by those commands.
- Appendix E lists information about command-generated error messages that can be monitored.
- Appendix F lists abbreviations used in control language command names and parameter keywords and values.

The following changes have been made to this manual for System/38 for release 4.1:

- The following are new commands supported on System/38:

APYJRNCHG	DSPJRN
CHGJRN	DSPJRNRCVA
CHGLF	DSPJRNA
CHGLFM	DSPOBJLCK
CHGPF	DSPPGMCHG
CHGPFM	ENDJRNPF
CHGSRCPF	GRTUSRAUT
CRTJRN	JRNPF
CRTJRNRCV	LSTCNPDTA
DLTJRN	LSTCNPHST
DLTJRNRCV	RMVJRNCHG
DMPCLPGM	RTVDTAARA
DMPTAP	SAVCHGOBJ
DSPACTJOB	SNDJRNE

The Journal facility provides improved function for data base backup and recovery, as well as enabling you to use an audit trail. The change file and change file member commands allow you to change the attributes specified when the file or member was created. The Dump CL Program (DMPCLPGM) command can be used in a CL program to dump program variables and messages, should an unmonitored escape message occur. The Dump Tape (DMPTAP) command allows you to display or list the labels and/or data of a tape. The Display Active Jobs (DSPACTJOB) command provides a summary display of jobs active in the system. The Display Object Locks (DSPOBJLCK) command displays all locks and lock requests for a particular object. The Display Programming Change (DSPPGMCHG) command provides a display of the status of programming changes. The Grant User Authority (GRTUSRAUT) command allows a named user to be given the same authority as another named user. The List CSNAP History and List CSNAP Data (LSTCNPHST and LSTCNPDTA) commands allow you to display current and past communications line statistics. The Retrieve Data Area (RTVDTAARA) command is used in CL programs to retrieve the contents of a data area and place it into a variable. The Save Changed Object (SAVCHGOBJ) command saves only those objects that have been changed since a specified date/time.

- Miscellaneous new command parameters and technical changes support the following enhancements:
  - Maximum objects allowed per save/restore have been increased
  - Data saved by save/restore is now given an expiration date
  - CLEAR option is now supported for tape
  - Saved data can now be displayed at the member level
  - DSPJOB menu now provides an option to show job locks
  - DSPSYSSTS now shows percentage of machine addresses used
  - File displays now support scanning and windowing
  - Program size limitation has been increased from 8 K to 32 K bytes
  - New concatenation operators *BCAT and *TCAT are supported
  - Ten line descriptions per communications line are now supported
  - Generic names may be specified for CHGPRTF operations
  - Spanned records, undefined format records, and improved error recovery are now supported for tape
  - Tape may now be used as PID media and for system backup
  - I-Format is now supported for diskette interchange
  - Long-running machine instructions can now be canceled
  - Debug support has been enhanced
  - The power warning feature has been enhanced
  - Functional enhancements have been added to simplify conversion from the System/34

Also, various examples have been updated and improved.

Technical changes are noted with a vertical change bar to the left of the changed material.

Note: This manual follows the convention that *he* means *he* or *she*.

WHAT YOU SHOULD KNOW

To use this manual, you should understand the concepts of the IBM System/38 Control Program Facility. Information about those concepts can be found in the *IBM System/38 Control Program Facility Concepts Manual,* SC21-7729.

Also, you should know how to use the 5251 and 5252 Display Stations as System/38 work stations. That information can be found in the *IBM System/38 Programmer's/User's Work Station Guide,* SC21-7744.

IF YOU NEED MORE INFORMATION

The following list describes other System/38 publications that explain in further detail topics related to the information presented in this reference manual.

System/38 Overview Information

- *IBM System/38 System Introduction,* GC21-7728
  - Summarizes the System/38 design and highlights its major functions
  - Describes System/38 licensed programs
  - Describes possible System/38 configurations
  - Describes hardware device characteristics

- *IBM System/38 Application Example 1,* SC21-7881
  - Uses a basic application to illustrate the use of CPF, RPG III, and the Interactive Data Base Utilities (IDU) on System/38

Control Language Commands

- *IBM System/38 Programming Reference Summary,* SC21-7734
  - Contains syntax diagrams for all CL commands
  - Describes object authority required for commands and objects
  - Lists the names of IBM-supplied objects
  - Contains a brief description of system values

Data Description Specifications

- *IBM System/38 Control Program Facility Reference Manual—Data Description Specifications,* SC21-7806
  - Describes in detail how to describe files using DDS
  - Provides a list of valid DDS keywords for each file type

- *IBM System/38 Screen Design Aid Reference Manual and User's Guide,* SC21-7755
  - Describes how to design, create, and maintain display record formats and menus using SDA

- *IBM System/38 Programming Reference Summary,* SC21-7734
  - Provides a list of valid DDS keywords for each file type

Messages

- *IBM System/38 Messages Guide: CPF, RPG III, and IDU,* SC21-7736
  - Describes each message, including the first- and second-level text, the substitution variables, the severity, and the system action

- *IBM System/38 Messages Guide: COBOL,* SC21-7823
  - Describes each message, including the first- and second-level text, the substitution variables, the severity, and the system action

- *IBM System/38 Programmer's/User's Work Station Guide,* SC21-7744
  - Describes how to send and receive messages at a display station

Languages

- *IBM System/38 RPG III Reference Manual and Programmer's Guide,* SC21-7725
  - Describes RPG III specifications
  - Provides information on writing, testing, and maintaining RPG III programs

- *IBM System/38 COBOL Reference Manual and Programmer's Guide,* SC21-7718
  - Describes the System/38 COBOL compiler and language
  - Provides information on writing, testing, and maintaining COBOL programs

Communications

- *IBM System/38 Data Communications Programmer's Guide,* SC21-7825
  - Describes the System/38 data communications devices
  - Describes how to use the communications functions

- *IBM System/38 Control Program Facility Reference Manual—Data Description Specifications,* SC21-7806
  - Describes the DDS for a communications file and a BSC file

Utilities

- *IBM System/38 Source Entry Utility Reference Manual and User's Guide,* SC21-7722
  - Describes how to use SEU to enter and maintain control language statements, data description specifications, and command definition statements

- *IBM System/38 Data File Utility Reference Manual and User's Guide,* SC21-7714
  - Describes how to use DFU to create and maintain data files

- *IBM System/38 Query Utility Reference Manual and User's Guide,* SC21-7724
  - Describes how to use query to create reports from information in data base files

- *IBM System/38 Screen Design Aid Reference Manual and User's Guide,* SC21-7755
  - Describes how to design, create, and maintain display record formats and menus using SDA

- *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914
  - Describes how to use RJEF to install, start, control, and terminate a remote job entry system

- *IBM System/38 Remote Job Entry Facility Installation Planning Guide,* GC21-7924
  - Describes RJEF functions
  - Describes how to install and configure an RJEF network

System Operation

- *IBM System/38 Operator's Guide,* SC21-7735
  - Describes system operator and system request menus
  - Describes job and system status displays
  - Describes how to submit and control jobs through job and spooling commands
  - Describes how to vary or power devices on and off
  - Describes how to save and restore objects, libraries, and the system
  - Describes diskette handling
  - Describes message handling for the system operator

Installation and Device Configuration

- *IBM System/38 Guide to Program Product Installation and Device Configuration,* GC21-7775
  - Describes how to install and configure System/38

- *IBM System/38 Remote Job Entry Facility Installation Planning Guide,* GC21-7924
  - Describes how to install and configure a RJEF network

Problem Determination

- *IBM System/38 Problem Determination Guide,* SC21-7876
  - Describes the procedures for resolving system problems that are indicated by error messages, operator/service panel lights, interactive/batch jobs or spooling functions that do not work as expected, or devices that do not work as expected

Content and Use of System/38 Publications

- *IBM System/38 Guide to Publications,* GC21-7726
  - Describes the contents of System/38 publications

- *IBM System/38 Glossary and Master Index,* GC21-7727
  - Defines terms used in System/38 publications
  - Contains index entries from frequently used System/38 publications

Part 1. Control Language Functions and Syntax

Part 1 provides an overview of the control language commands and describes the syntax coding rules needed to code them. Over 250 commands are provided in the control language, permitting the users of a System/38 to request a broad range of functions from the system.

Control language (CL) commands can be entered into the system in several forms, and they can be entered in the interactive and batch environments. The commands can be coded in a fixed positional form that omits trailing optional parameters, or in a free form that omits all unneeded parameters. The commands can be entered interactively at a work station, submitted in batch input streams, or compiled in CL programs. When entering a command interactively, you can directly enter the complete command; or you can be prompted by the system for each parameter value so you can change the displayed default values and fill in the blanks. Some commands can be used only in certain forms (such as interactively or in CL programs). These restrictions are included in the description of the command.

Because CPF is object oriented, many of the commands are designed to create or operate on these objects. Also, varying degrees of security can be applied to the objects, to the commands, and to the system's users. If you have the proper authority for a command to be entered and for the objects to be operated on (in the manner specified by the command), you can request that function of the system.

The charts in Chapter 1 introduce the CPF object types and the CL command set. Chapter 2 describes the syntax coding rules that the user must follow to properly enter the commands for execution. Refer to the *CPF Programmer's Guide* for additional information.

Chapter 1. Summary of CPF Functions and Object Types

The purpose of the charts in this chapter is to help you become familiar with the functions of the CPF control language and the names of the commands. These charts can also be used to quickly retrieve various kinds of command-related information.

The first group of charts summarizes the CPF object types and shows the commands that operate on the object types. The second group of charts provides an overview of the broad range of functions that can be performed by the control language. Finally, all of the commands are shown together in one master matrix chart.

CPF OBJECT TYPES

CPF objects provide the means through which all of your data processing information is stored and processed by the system. A *CPF object* is a named unit that exists in (occupies space in) storage and upon which operations can be performed by the CPF. Each object has a set of attributes that describe the object; these attributes are defined when the object is created. For the object to be used by the system to perform a specific function, the name of the object must be specified in the CL command that performs that function.

Twenty-three types of CPF objects can be created and used in the control language. They are identified in the following chart, which gives the object type, the system-recognized identifier for the object type, and a brief description of its purpose in CPF.

Type	Identifier	Description
File	*FILE	Contains, or provides access to, a group of related data records in the system. Includes: data base, card, diskette, tape, printer, and display files.
Program	*PGM	Contains the executable code needed to perform the user's task. For example: CL and high-level language programs.
Library	*LIB	Contains one or more objects of the other object types. Serves as a directory to find objects by name when they are to be used. Subtypes: production and test libraries.

Type	Identifier	Description
Command	*CMD	Contains the description of a CL command.
Data area	*DTAARA	Contains a data value that can be used and changed by multiple jobs.
User profile	*USRPRF	Identifies a user to the system and specifies what system resources and objects he can use.
Message file	*MSGF	Contains descriptions of predefined messages.
Message queue	*MSGQ	Contains messages being sent and received by the system and its users.
Job queue	*JOBQ	Contains entries for jobs that are to be executed by the system.
Output queue	*OUTQ	Contains entries for spooled output files to be written to an output device.
Job description	*JOBD	Contains a set of attributes that are used to control job execution.
Subsystem description	*SBSD	Describes a subsystem and its operating environment in the system.
Class	*CLS	Describes the processing environment and attributes of routing steps.
Table	*TBL	Contains a set of values used to define a byte-by-byte translation of data, or to define a collating sequence.
Edit description	*EDTD	Describes an edit code mask used for formatting output fields.
Print image	*PRTIMG	Contains an image of a printable character set on a print belt.
Device description	*DEVD	Describes a device on the system, and its features.
Control unit description	*CUD	Describes a control unit on the system, and its features.
Line description	*LIND	Describes a communication line on the system, and its features.

Type	Identifier	Description
Forms control table	*FCT	Describes, for the Remote Job Entry Facility, special processing requirements for data received from the host system.
Session description	*SSND	Identifies, for the Remote Job Entry Facility, all objects and devices associated with an RJE operating environment.
Journal	*JRN	Contains information about journaled data base files and journal receivers and provides access to journal receivers.
Journal receiver	*JRNRCV	Contains journal entries that are generated when changes are made to data base files.

All CPF objects have the following characteristics in common: Each object has a set of attributes that describe the object, and specific values assigned for those attributes. Most of the objects are stored in libraries. Five types of CPF objects (*LIB, *DEVD, *CUD, *LIND, and *USRPRF) are actually stored in the machine context, which is part of the internal system. However, these types appear as if they exist in the QSYS (system) library. They can be displayed if QSYS is specified in the DSPLIB or DSPOBJD commands.

Generally, each object exists independently of all other objects. However, some objects must be created before other objects can be created; for example, a logical file cannot be created if its based-on physical file does not exist. Each object must be created before other CPF operations can be performed using the object.

For more information on each of the object types, refer to Part 2 for the description of each create command that creates one of the object types. Additional information can also be found in the appropriate sections of the *CPF Programmer's Guide.*

CPF-Provided Libraries

Several libraries are defined in CPF when the system is shipped. The IBM-supplied libraries are:

- QGPL (general purpose library): Contains user-created objects, such as programs and files, and IBM-supplied versions of objects that a user might create. When a user creates an object without specifying the name of the library in which it is to be placed, the created object is placed in the QGPL library by default.

- QSYS (system library): Contains IBM-supplied system support objects.

- QSPL (spooling library): Contains IBM-supplied objects used for spooling data.

- QTEMP (temporary library): Automatically created for each job to contain temporary objects that are created by that job. Each job has its own temporary library; the library and its objects exist only as long as the job is active in the system.

- QSRV (service library): Used for loading IBM-supplied programming changes and assembling data for APAR submission.

- QRECOVERY (recovery library): Contains information that is used for recovery after a system failure.

More information about the use of libraries can be found in the *CPF Concepts Manual* and the *CPF Programmer's Guide.*

COMMANDS OPERATING ON CPF OBJECTS

Each of the CPF object types has a set of commands that operates on that type. Most objects have commands that do the following:

- Create. Creates the object and specifies its attributes.
- Delete. Deletes the object from the system.
- Change. Changes the attributes and/or contents of the object.
- Display. Displays the contents of the object.

The following matrix chart (*Commands Operating on Specific Object Types*) shows all of the CPF object types (in alphabetic order) and the actions that can be performed upon them by CL commands. Both the descriptive name and the command abbreviations for each object type are listed vertically on the left side of the chart, and the verbs (actions) are listed across the top of the chart. When an action can be performed on a particular object, the command abbreviation for that verb is given on the same line as the object's name.

The functions that can be performed on CPF objects, then, are the combination of the verbs and the objects upon which the action is to be performed: (CPF function = verb + object acted upon). For example, you can create, delete, or display a class; so the verb abbreviations CRT, DLT, and DSP are printed opposite the abbreviation for class, CLS. The result is the three commands that can operate on a class: CRTCLS, DLTCLS, and DSPCLS.

The IBM-supplied commands are all named in a consistent manner. Generally, three letters from each word in the descriptive command name are used to form the abbreviated command name that is recognized by the system. For examples of how commands and other objects supplied by IBM are named, see *Control Language* in the *CPF Programmer's Guide*.

Included in the chart are the subtypes that are identified by name in CL commands. These subtypes are shown in logical sublevels under their primary object types, file and program. The subtypes for files are logically grouped as spooled files, data base files (physical and logical), and device files (card, diskette, display, and print). The chart shows, for example, that you create a file according to its *subtype* (CRTCRDF, for example) and you delete it by the object *type* (DLTF).

The chart also identifies (under *Other Associated Commands*) other commands that are indirectly related to an object type:

- Subsystem commands associated with the subsystem description
- File-related commands associated with various file subtypes
- Device and line-related commands associated with their descriptions

Commands Operating on Specific Object Types

CPF Object Types			Actions						Other Associated Commands
			Create	Delete	Change	Override	Display	Other Verbs	
1.	Class	CLS	CRT	DLT			DSP		
2.	Command	CMD	CRT	DLT	CHG				
3.	Control unit description	CUD	CRT	DLT	CHG		DSP		PWRCTLU, VRYCTLU, DSPCTLSTS
4.	Data area	DTAARA	CRT	DLT	CHG		DSP	DCL RCV SND	
5.	Device description	DEVD	CRT	DLT	CHG		DSP		PWRDEV, VRYDEV, DSPDEVSTS
6.	Edit description	EDTD	CRT	DLT			DSP		
7.	File	F		DLT				CPY DCL SND RCV SNDRCV	CPYFI, DSPFD, DSPFFD
	BSC file	BSCF	CRT		CHG	OVR			
	Spooled file	SPLF					DSP	CNL HLD RLS	CHGSPLFA, DSPSPLFA
	Data base file	DBF				OVR		LOG	DSPDBR, RMVM, ENDLOG
	Logical file	LF	CRT		CHG				ADDLFM, CHGLFM
	Physical file	PF	CRT		CHG			JRN	ADDPFM, CHGPFM, CLRPFM,
	Source physical file	SRCPF	CRT		CHG				INZPFM, RGZPFM
	Card file	CRDF	CRT		CHG	OVR			
	Communications file	CMNF	CRT		CHG	OVR			
	Diskette file	DKTF	CRT		CHG	OVR			
	Display file	DSPF	CRT		CHG	OVR			
	Printer file	PRTF	CRT		CHG	OVR			
	Tape file	TAPF	CRT		CHG	OVR			
8.	Forms control table	FCT	CRT	DLT	CHG		DSP		
9.	Job description	JOBD	CRT	DLT	CHG		DSP		
10.	Job queue	JOBQ	CRT	DLT			DSP	CLR HLD RLS	
11.	Journal	JRN	CRT	DLT	CHG		DSP		DSPJRNA
12.	Journal receiver	JRNRCV	CRT	DLT					DSPJRNRCVA
13.	Library	LIB	CRT	DLT			DSP	CLR SAV RST	
14.	Line description	LIND	CRT	DLT	CHG		DSP		VRYLIN, DSPLINSTS
15.	Message file	MSGF	CRT	DLT		OVR	DSP		RTVMSG, ADDMSGD, CHGMSGD, RMVMSGD, DSPMSGD
16.	Message queue	MSGQ	CRT	DLT	CHG				DSPMSG, RCVMSG, RMVMSG, SNDMSG, SNDBRKMSG, SNDPGMMSG, SNDRPY
17.	Output queue	OUTQ	CRT	DLT	CHG		DSP	CLR HLD RLS	
18.	Print image	PRTIMG	CRT	DLT					
19.	Program	PGM		DLT				END	CALL, TFRCTL
	CL program	CLPGM	CRT					DMP	
20.	Session description	SSND	CRT	DLT	CHG		DSP		
21.	Subsystem description	SBSD	CRT	DLT	CHG		DSP		DSPSBS, STRSBS, TRMSBS
22.	Table	TBL	CRT	DLT					
23.	User profile	USRPRF	CRT	DLT	CHG		DSP	RST	GRTUSRAUT

In addition to the commands that operate on single object types, there are commands that operate on multiple object types; for example:

- Display object description: Displays the common attributes of an object.
- Move object: Moves an object from one library to another.
- Rename object: Specifies the new name of an object.
- Save object: Saves an object and its contents on diskette or tape.
- Restore object: Restores a saved version of the object from diskette or tape.

The following chart shows the commands, in matrix form, that can perform an action on many of the object types. Some of the commands, such as the MOVOBJ command, can operate on only one object at a time, but that object can be any one of several CPF object types; for example, the MOVOBJ command can move a file or a job description.

Other commands, such as the DSPOBJD command, can operate on several objects of different types at the same time. By specifying multiple objects in a single DSPOBJD command, you can display the object descriptions of a group of objects.

Commands Operating on Multiple Object Types

Item		**Actions**
Object	OBJ	ALC, DLC, SAV, RST, CHK, MOV, RNM, DMP
Object Authority	OBJAUT	DSP, GRT, RVK
Object Description	OBJD	DSP
Object Lock	OBJLCK	DSP
Object Owner	OBJOWN	CHG

For more information on these commands and the object types that each one can operate on, see the command description of each command in Part 2.

COMMAND GROUPS (BY FUNCTION)

The following sets of commands contain all of the CL commands in functional groups and subgroups. The commands are grouped by common functions in various ways to help you identify which commands are associated with the major functional areas in CPF.

These groups are organized in the same manner as the groups are displayed when the *command grouping menus* are requested at a work station.

If you press the prompt (CF4) key without entering a command name, the command grouping menu is presented. From the menu, you can specify an option number to view any of the various groups of commands that are shown on the following pages.

OBJECT AND LIBRARY COMMANDS

Object

ALCOBJ	(Allocate Object)
CHKOBJ	(Check Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Library

CLRLIB	(Clear Library)
CRTLIB	(Create Library)
DLTLIB	(Delete Library)
DSPLIB	(Display Library)
RSTLIB	(Restore Library)
SAVLIB	(Save Library)

Common Functions for Library

ALCOBJ	(Allocate Object)
CHKOBJ	(Check Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
RNMOBJ	(Rename Object)

Library List

DSPLIBL	(Display Library List)
RPLLIBL	(Replace Library List)

DATA BASE COMMANDS

Valid for Both Physical and Logical Files

CPYF	(Copy File)
DLTF	(Delete File)
DLTOVR	(Delete Override)
DSPDBR	(Display Data Base Relations)
DSPFD	(Display File Description)
DSPFFD	(Display File Field Description)
DSPPGMREF	(Display Program Reference)
ENDLOG	(End Logging)
LOGDBR	(Log Data Base File)
OVRDBF	(Override with Data Base File)
RMVM	(Remove Member)

Common Functions for Files

ALCOBJ	(Allocate Object)
CHKOBJ	(Check Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Physical File

ADDPFM	(Add Physical File Member)
CHGPF	(Change Physical File)
CHGPFM	(Change Physical File Member)
CHGSRCPF	(Change Source Physical File)
CLRPFM	(Clear Physical File Member)
CRTPF	(Create Physical File)
CRTSRCPF	(Create Source Physical File)
INZPFM	(Initialize Physical File Member)
RGZPFM	(Reorganize Physical File Member)

Logical File

ADDLFM	(Add Logical File Member)
CHGLF	(Change Logical File)
CHGLFM	(Change Logical File Member)
CRTLF	(Create Logical File)

Journal

APYJRNCHG	(Apply Journaled Changes)
CHGJRN	(Change Journal)
CRTJRN	(Create Journal)
CRTJRNRCV	(Create Journal Receiver)
DLTJRN	(Delete Journal)
DLTJRNRCV	(Delete Journal Receiver)
DSPJRN	(Display Journal)
DSPJRNA	(Display Journal Attributes)
DSPJRNRCVA	(Display Journal Receiver Attributes)
ENDJRNPF	(End Journaling Physical File Changes)
JRNPF	(Journal Physical File)
RMVJRNCHG	(Remove Journaled Changes)
SNDJRNE	(Send Journal Entry)

DEVICE FILE COMMANDS

Valid for All Device Files

CPYF	(Copy File)
DLTF	(Delete File)
DLTOVR	(Delete Override)
DSPFD	(Display File Description)
DSPFFD	(Display File Field Description)
DSPOVR	(Display Override)
DSPPGMREF	(Display Program References)

Common Functions for Device Files

ALCOBJ	(Allocate Object)
CHKOBJ	(Check Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

BSC File

CHGBSCF	(Change BSC File)
CRTBSCF	(Create BSC File)
OVRBSCF	(Override with BSC File)

Card File

CHGCRDF	(Change Card File)
CRTCRDF	(Create Card File)
OVRCRDF	(Override with Card File)

Communications File

CHGCMNF	(Change Communications File)
CRTCMNF	(Create Communications File)
OVRCMNF	(Override with Communications File)

Diskette File

CHGDKTF	(Change Diskette File)
CRTDKTF	(Create Diskette File)
OVRDKTF	(Override with Diskette File)

Display File

CHGDSPF	(Change Display File)
CRTDSPF	(Create Display File)
OVRDSPF	(Override with Display File)

Display File (In CL Program)

CNLRCV	(Cancel Receive)
DCLF	(Declare File)
RCVF	(Receive File)
SNDF	(Send File)
SNDRCVF	(Send/Receive File)
WAIT	(Wait)

Printer File

CHGPRTF	(Change Printer File)
CRTPRTF	(Create Printer File)
OVRPRTF	(Override with Printer File)

Tape File

CHGTAPF	(Change Tape File)
CRTTAPF	(Create Tape File)
OVRTAPF	(Override with Tape File)

DEVICE MANAGEMENT COMMANDS

Device

DSPDEVSTS	(Display Device Status)
PWRDEV	(Power Device)
VRYDEV	(Vary Device)

Control Unit

DSPCTLSTS	(Display Control Unit Status)
PWRCTLU	(Power Control Unit)
VRYCTLU	(Vary Control Unit)

Line

ANSLIN	(Answer Line)
DSPLINSTS	(Display Line Status)
VRYLIN	(Vary Line)

Diskette Volume

CLRDKT	(Clear Diskette)
DLTDKTLBL	(Delete Diskette Label)
DSPDKT	(Display Diskette)
DUPDKT	(Duplicate Diskette)
INZDKT	(Initialize Diskette)
RNMDKT	(Rename Diskette)

Printer

CLNPRT	(Clean Printer)
VFYPRT	(Verify Printer)

Tape Volume

DMPTAP	(Dump Tape)
DSPTAP	(Display Tape)
INZTAP	(Initialize Tape)

PROGRAMMING COMMANDS

Valid for All Programs

DLTPGM	(Delete Program)
RCLRSC	(Reclaim Resources)

Common Functions for Programs

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

CL Program

CRTCLPGM	(Create Control Language Program)
DMPCLPGM	(Dump CL Program)
RTVCLSRC	(Retrieve CL Source)

CL Program Limits

ENDPGM	(End Program)
PGM	(Program)

CL Program Variable

CHGVAR	(Change Variable)
CVTDAT	(Convert Date)
DCL	(Declare Control Language Variable)

CL Program Logic

DO	(Do)
ELSE	(Else)
ENDDO	(End Do)
GOTO	(Go To)
IF	(If)

Changing Program Control

CALL	(Call Program)
RETURN	(Return)

Program Control (In CL Program)

TFRCTL	(Transfer Control)

RPG III Program (If Installed)

CRTRPGPGM	(Create RPG Program)
CRTRPTPGM	(Create Report Program)

COBOL Program (If Installed)

CRTCBLPGM	(Create COBOL Program)

Data Area

CHGDTAARA	(Change Data Area)
CRTDTAARA	(Create Data Area)
DLTDTAARA	(Delete Data Area)
DSPDTAARA	(Display Data Area)

Data Area (In CL Program)

DCLDTAARA	(Declare Data Area)
RCVDTAARA	(Receive Data Area)
RTVDTAARA	(Retrieve Data Area)
SNDDTAARA	(Send Data Area)

Common Functions for Data Area

ALCOBJ	(Allocate Object)
CHKOBJ	(Check Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

PROGRAM DEBUG COMMANDS

Debug Mode

ADDPGM	(Add Program)
CHGDBG	(Change Debug)
DSPDBG	(Display Debug)
ENDDBG	(End Debug)
ENTDBG	(Enter Debug)
RMVPGM	(Remove Program)

Program Variable

CHGPGMVAR	(Change Program Variable)
DSPPGMVAR	(Display Program Variable)

Program Pointer

CHGPTR	(Change Pointer)

Breakpoint

ADDBKP	(Add Breakpoint)
CNLRQS	(Cancel Request)
DSPBKP	(Display Breakpoints)
RMVBKP	(Remove Breakpoint)
RSMBKP	(Resume Breakpoint)

Trace

ADDTRC	(Add Trace)
CLRTRCDTA	(Clear Trace Data)
DSPTRC	(Display Trace)
DSPTRCDTA	(Display Trace Data)
RMVTRC	(Remove Trace)

COBOL Debug Mode

ENDCBLDBG	(End COBOL Debug)
ENTCBLDBG	(Enter COBOL Debug)

MESSAGE HANDLING COMMANDS

Message

DSPMSG	(Display Messages)
SNDBRKMSG	(Send Break Message)
SNDMSG	(Send Message)

Message (In CL Program)

MONMSG	(Monitor Message)
RCVMSG	(Receive Message)
RMVMSG	(Remove Message)
RTVMSG	(Retrieve Message)
SNDPGMMSG	(Send Program Message)
SNDRPY	(Send Reply)

Message Queue

CHGMSGQ	(Change Message Queue)
CRTMSGQ	(Create Message Queue)
DLTMSGQ	(Delete Message Queue)

Common Functions for Message Queue

ALCOBJ	(Allocate Object)
CHKOBJ	(Check Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)

Message File

CRTMSGF	(Create Message File)
DLTMSGF	(Delete Message File)
DSPMSGF	(Display Message File)
OVRMSGF	(Override with Message File)

Common Functions for Message File

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
| DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
| SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Message Description

ADDMSGD	(Add Message Description)
CHGMSGD	(Change Message Description)
DSPMSGD	(Display Message Description)
RMVMSGD	(Remove Message Description)

INPUT/OUTPUT SPOOLING COMMANDS

Job Queue

CLRJOBQ	(Clear Job Queue)
CRTJOBQ	(Create Job Queue)
DLTJOBQ	(Delete Job Queue)
DSPJOBQ	(Display Job Queue)
HLDJOBQ	(Hold Job Queue)
RLSJOBQ	(Release Job Queue)

Common Functions for Job Queue

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)

Output Queue

CHGOUTQ	(Change Output Queue)
CLROUTQ	(Clear Output Queue)
CRTOUTQ	(Create Output Queue)
DLTOUTQ	(Delete Output Queue)
DSPOUTQ	(Display Output Queue)
HLDOUTQ	(Hold Output Queue)
RLSOUTQ	(Release Output Queue)

Common Functions for Output Queue

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)

Spooled File

CHGSPLFA	(Change Spooled File Attributes)
CNLSPLF	(Cancel Spooled File)
CPYSPLF	(Copy Spooled File)
DSPSPLF	(Display Spooled File)
DSPSPLFA	(Display Spooled File Attributes)
HLDSPLF	(Hold Spooled File)
RLSSPLF	(Release Spooled File)

Job

DSPSBMJOB	(Display Submitted Jobs)
SBMCRDJOB	(Submit Card Jobs)
SBMDBJOB	(Submit Data Base Jobs)
SBMDKTJOB	(Submit Diskette Jobs)

Reader

CNLRDR	(Cancel Reader)
DSPRDR	(Display Reader)
HLDRDR	(Hold Reader)
RLSRDR	(Release Reader)
STRCRDRDR	(Start Card Reader)
STRDBRDR	(Start Data Base Reader)
STRDKTRDR	(Start Diskette Reader)

Writer

CNLWTR	(Cancel Writer)
DSPWTR	(Display Writer)
HLDWTR	(Hold Writer)
RLSWTR	(Release Writer)
STRCRDWTR	(Start Card Writer)
STRDKTWTR	(Start Diskette Writer)
STRPRTWTR	(Start Printer Writer)

Job Stream Statements

DATA	(Data)
ENDINP	(End Input)

SYSTEM AND JOB CONTROL COMMANDS

System

DSPSYS	(Display System)
DSPSYSSTS	(Display System Status)
PWRDWNSYS	(Power Down System)
TRMCPF	(Terminate Control Program Facility)

Subsystem

DSPSBS	(Display Subsystem)
STRSBS	(Start Subsystem)
TRMSBS	(Terminate Subsystem)

Job

CHGJOB	(Change Job)
CNLJOB	(Cancel Job)
DSPACTJOB	(Display Active Jobs)
DSPJOB	(Display Job)
DSPSBMJOB	(Display Submitted Jobs)
HLDJOB	(Hold Job)
RLSJOB	(Release Job)
RRTJOB	(Reroute Job)
SBMCRDJOB	(Submit Card Jobs)
SBMDBJOB	(Submit Data Base Jobs)
SBMDKTJOB	(Submit Diskette Jobs)
SBMJOB	(Submit Job)
SIGNOFF	(Sign Off)
TFRJOB	(Transfer Job)

Job (In CL Program)

RTVJOBA	(Retrieve Job Attributes)

Job Stream Statements

JOB	(Job)
ENDJOB	(End Job)

Log

DSPLOG	(Display Log)

System Value

CHGSYSVAL	(Change System Value)
DSPSYSVAL	(Display System Value)

System Value (In CL Program)

RTVSYSVAL	(Retrieve System Value)

Storage

RCLSTG	(Reclaim Storage)

SUBSYSTEM DESCRIPTION, JOB DESCRIPTION, AND CLASS COMMANDS

Subsystem Description

CHGSBSD	(Change Subsystem Description)
CRTSBSD	(Create Subsystem Description)
DLTSBSD	(Delete Subsystem Description)
DSPSBSD	(Display Subsystem Description)

Common Functions for Subsystem Description

ALCOBJ	(Allocate Object)
CHKOBJ	(Check Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Subsystem Autostart Job Entry

ADDAJE	(Add Autostart Job Entry)
CHGAJE	(Change Autostart Job Entry)
RMVAJE	(Remove Autostart Job Entry)

Subsystem Work Station Entry

ADDWSE	(Add Work Station Entry)
CHGWSE	(Change Work Station Entry)
RMVWSE	(Remove Work Station Entry)

Subsystem Job Queue Entry

ADDJOBQE	(Add Job Queue Entry)
CHGJOBQE	(Change Job Queue Entry)
RMVJOBQE	(Remove Job Queue Entry)

Subsystem Routing Entry

ADDRTGE	(Add Routing Entry)
CHGRTGE	(Change Routing Entry)
RMVRTGE	(Remove Routing Entry)

Job Description

CHGJOBD	(Change Job Description)
CRTJOBD	(Create Job Description)
DLTJOBD	(Delete Job Description)
DSPJOBD	(Display Job Description)

Common Functions for Job Description

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Class

CRTCLS	(Create Class)
DLTCLS	(Delete Class)
DSPCLS	(Display Class)

Common Functions for Class

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

CONFIGURATION COMMANDS

Device Configuration

DSPDEVCFG	(Display Device Configuration)

Device Description

CHGDEVD	(Change Device Description)
CRTDEVD	(Create Device Description)
DLTDEVD	(Delete Device Description)
DSPDEVD	(Display Device Description)

Common Functions for Device Description

ALCOBJ	(Allocate Object)
DLCOBJ	(Deallocate Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)

Control Unit Description

CHGCUD	(Change Control Unit Description)
CRTCUD	(Create Control Unit Description)
DLTCUD	(Delete Control Unit Description)
DSPCUD	(Display Control Unit Description)

Common Functions for Control Unit Description

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Line Description

CHGLIND	(Change Line Description)
CRTLIND	(Create Line Description)
DLTLIND	(Delete Line Description)
DSPLIND	(Display Line Description)

Common Functions for Line Description

DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)

Edit Code

CRTEDTD	(Create Edit Description)
DLTEDTD	(Delete Edit Description)
DSPEDTD	(Display Edit Description)

Common Functions for Edit Code

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Print Image

CRTPRTIMG	(Create Print Image)
DLTPRTIMG	(Delete Print Image)

Common Functions for Print Image

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Translate Table

CRTTBL	(Create Table)
DLTTBL	(Delete Table)

Common Functions for Translate Table

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

UTILITY COMMANDS[1]

Data

CHGDTA	(Change Data)
DSPDTA	(Display Data)
FMTDTA	(Format Data)
QRYDTA	(Query Data)

Source

EDTSRC	(Edit Source)
RTVDFUSRC	(Retrieve DFU Source)
RTVQRYSRC	(Retrieve Query Source)

DFU

CHGDFUDEF	(Change DFU Definition)
CHGDTA	(Change Data)
CRTDFUAPP	(Create DFU Application)
CRTDFUDEF	(Create DFU Definition)
DLTDFUAPP	(Delete DFU Application)
DSNDFUAPP	(Design DFU Application)
DSPDTA	(Display Data)

Query

CHGQRYDEF	(Change Query Definition)
CRTQRYAPP	(Create Query Application)
CRTQRYDEF	(Create Query Definition)
DLTQRYAPP	(Delete Query Application)
DSNQRYAPP	(Design Query Application)
QRYDTA	(Query Data)

Display Formats

DSNFMT	(Design Format)

Conversion Reformat Utility[2]

FMTDTA	(Format Data)

SECURITY COMMANDS

General

CHGOBJOWN	(Change Object Owner)
DSPAUTUSR	(Display Authorized Users)

User Profile

CHGUSRPRF	(Change User Profile)
CRTUSRPRF	(Create User Profile)
DLTUSRPRF	(Delete User Profile)
DSPUSRPRF	(Display User Profile)

Common Functions for User Profile

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)

Object Authorization

DSPOBJAUT	(Display Object Authority)
GRTOBJAUT	(Grant Object Authority)
GRTUSRAUT	(Grant User Authority)
RVKOBJAUT	(Revoke Object Authority)

SAVE/RESTORE COMMANDS

Object

RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Library

RSTLIB	(Restore Library)
SAVLIB	(Save Library)

System

RSTAUT	(Restore Authority)
RSTUSRPRF	(Restore User Profiles)
SAVSYS	(Save System)

[1]These commands are part of the IBM System/38 Interactive Data Base Utilities Program.
[2]This command is part of the IBM System/38 Conversion Reformat Utility Licensed Program.

COMMAND DEFINITION COMMANDS

Command

CHGCMD	(Change Command)
CRTCMD	(Create Command)
DLTCMD	(Delete Command)
DSPCMD	(Display Command)
LSTCMDUSG	(List Command Usage)

Common Functions for Command

CHKOBJ	(Check Object)
DSPOBJD	(Display Object Description)
DSPOBJLCK	(Display Object Locks)
MOVOBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

SERVICE COMMANDS

Job

DMPJOB	(Dump Job)
DSPACTJOB	(Display Active Jobs)
DSPSRVSTS	(Display Service Status)
ENDSRV	(End Service)
SRVJOB	(Service Job)
TRCJOB	(Trace Job)

Object

DMPOBJ	(Dump Object)
DMPSYSOBJ	(Dump System Object)

Device

CHGCNPA	(Change CSNAP Attributes)
DSPCNPA	(Display CSNAP Attributes)
LSTCNPDTA	(List CSNAP Data)
LSTCNPHST	(List CSNAP History)
LSTERRLOG	(List Error Log)
STRCNFCHK	(Start Confidence Check)
STRPDP	(Start Problem Determination Procedure)

Printer

CLNPRT	(Clean Printer)
VFYPRT	(Verify Printer)

Tape Volume

DMPTAP	(Dump Tape)

Internal Machine

DMPJOBINT	(Dump Job Internal)
LSTERRLOG	(List Error Log)
LSTINTDTA	(List Internal Data)
TRCINT	(Trace Internal)

Problem Reporting

PRPAPAR	(Prepare APAR)

Programming Change

APYPGMCHG	(Apply Programming Change)
DSPPGMCHG	(Display Programming Change)
LODPGMCHG	(Load Programming Change)
PCHPGM	(Patch Program)
RMVPGMCHG	(Remove Programming Change)

REMOTE JOB ENTRY FACILITY COMMANDS

Session Description

CHGSSND	(Change Session Description)
CRTSSND	(Create Session Description)
DLTSSND	(Delete Session Description)
DSPSSND	(Display Session Description)

Reader Entry

ADDRJERDRE	(Add RJE Reader Entry)
CHGRJERDRE	(Change RJE Reader Entry)
RMVRJERDRE	(Remove RJE Reader Entry)

Writer Entry

ADDRJEWTRE	(Add RJE Writer Entry)
CHGRJEWTRE	(Change RJE Writer Entry)
RMVRJEWTRE	(Remove RJE Writer Entry)

Communications Entry

ADDRJECMNE	(Add RJE Communications Entry)
CHGRJECMNE	(Change RJE Communications Entry)
RMVRJECMNE	(Remove RJE Communications Entry)

Forms Control Table

CHGFCT	(Change Forms Control Table)
CRTFCT	(Create Forms Control Table)
DLTFCT	(Delete Forms Control Table)
DSPFCT	(Display Forms Control Table)

Forms Control Table Entry

ADDFCTE	(Add Forms Control Table Entry)
CHGFCTE	(Change Forms Control Table Entry)
RMVFCTE	(Remove Forms Control Table Entry)

Reader

CNLRJERDR	(Cancel RJE Reader)
STRRJERDR	(Start RJE Reader)

Writer

CNLRJEWTR	(Cancel RJE Writer)
STRRJEWTR	(Start RJE Writer)

Session Control

DSPRJESSN	(Display RJE Session)
STRRJESSN	(Start RJE Session)
TRMRJESSN	(Terminate RJE Session)

Console

STRRJECSL	(Start RJE Console)

Job

SBMRJEJOB	(Submit RJE Job)

Data

FMTRJEDTA	(Format RJE Data)

MASTER COMMAND MATRIX CHART

The following chart contains *all* of the CL commands that have more than one word in their descriptive names. All of the items (CPF objects and other entities) are listed vertically on the left side in alphabetic order (that is, each entry contains the descriptive name of the command minus the verb that precedes the item upon which it acts). The verbs that define the actions performed on each item are listed across the top; the verbs used on many items are listed in separate columns, and the verbs used on a few items are grouped together in the rightmost column.

This chart enables you to find in one place all of the functions that CL can perform, and gives the command names that are entered to perform the desired functions. The chart, therefore, can be used as an index that enables you to go directly to the command descriptions in Part 2, because they are described in alphabetic order there.

Items		**Actions**					
Items Affected	**Item Abbrev.**	**Create/ Delete**	**Add/ Remove**	**Change/ Override**	**Display**	**Hold/ Release**	**Other Actions**
Active jobs	ACTJOB				DSP		
APAR	APAR						PRP
Authority	AUT						RST
Authorized users	AUTUSR				DSP		
Auto report program	RPTPGM	CRT					DLT (see DLTPGM)
Autostart job entry	AJE		ADD RMV	CHG			
Break message	BRKMSG						SND
Breakpoint(s)	BKP		ADD RMV		DSP		RSM
BSC File	BSCF	CRT		CHG OVR			DLT (see DLTF) DSP (see DSPFD)
Card file	CRDF	CRT		CHG OVR			DLT (see DLTF) DSP (see DSPFD)
Card jobs	CRDJOB						SBM
Card reader	CRDRDR						STR (see also Reader)
Card writer	CRDWTR						STR (see also Writer)
Changed object	CHGOBJ						SAV
Class	CLS	CRT DLT			DSP		
COBOL debug (mode)	CBLDBG						END ENT
COBOL program	CBLPGM	CRT					DLT (see DLTPGM)
Command	CMD	CRT DLT		CHG	DSP		
Command usage	CMDUSG						LST
Communications file	CMNF	CRT		CHG OVR			DLT (see DLTF) DSP (see DSPFD)
Communications statistics network analysis procedure attributes	CNPA			CHG	DSP		LST (see LSTCNPDTA, LSTCNPHST)
Confidence check	CNFCHK						STR
Control	CTL						TFR
Control language program	CLPGM	CRT					DLT (see DLTPGM, DMP)

Items		Actions					
Items Affected	Item Abbrev.	Create/ Delete	Add/ Remove	Change/ Override	Display	Hold/ Release	Other Actions
Control language source	CLSRC						RTV
Control Program Facility	CPF						TRM
Control unit	CTLU						PWR VRY
Control unit description	CUD	CRT DLT		CHG	DSP		
Control unit status	CTLSTS				DSP		
Data	DTA			CHG	DSP		FMT QRY
Data area	DTAARA	CRT DLT		CHG	DSP		DCL SND RCV RTV
Data base file	DBF			OVR			CRT (see CRTLF/CRTPF) DLT (see DLTF) LOG
Data base jobs	DBJOB						SBM
Data base reader	DBRDR						STR
Data base relations	DBR				DSP		
Data File Utility application	DFUAPP	CRT DLT					DSN
Data File Utility definition	DFUDEF	CRT		CHG			
Data File Utility source	DFUSRC						RTV
Date	DAT						CVT
Debug (mode)	DBG			CHG	DSP		ENT END
Device	DEV						PWR VRY
Device configuration	DEVCFG				DSP		
Device description	DEVD	CRT DLT		CHG	DSP		
Device status	DEVSTS				DSP		
Diskette	DKT				DSP		CLR DUP INZ RNM
Diskette file	DKTF	CRT		CHG OVR			DLT (see DLTF) DSP (see DSPFD)
Diskette jobs	DKTJOB						SBM
Diskette label	DKTLBL	DLT					DSP (see DSPDKT) INZ (see INZDKT)
Diskette reader	DKTRDR						STR (see also Reader)
Diskette writer	DKTWTR						STR (see also Writer)
Display file	DSPF	CRT		CHG OVR			DLT (see DLTF) DSP (see DSPFD)
Do	DO						END
Edit description	EDTD	CRT DLT			DSP		
Error log	ERRLOG						LST
File	F	DLT					CPY DCL SND RCV SNDRCV
File description	FD				DSP		
File field description	FFD				DSP		
File interactive	FI						CPY
Format	FMT						DSN
Forms control table	FCT	CRT DLT		CHG	DSP		
Forms control table entry	FCTE		ADD RMV	CHG			
Input	INP						END
Internal	INT						TRC
Internal data	INTDTA						LST

Items		Actions					
Items Affected	Item Abbrev.	Create/ Delete	Add/ Remove	Change/ Override	Display	Hold/ Release	Other Actions
Job	JOB			CHG	DSP	HLD RLS	CNL DMP END RRT SBM SRV TFR TRC
Job attributes	JOBA						RTV
Job description	JOBD	CRT DLT		CHG	DSP		
Job internal	JOBINT						DMP
Job queue	JOBQ	CRT DLT			DSP	HLD RLS	CLR
Job queue entry	JOBQE		ADD RMV	CHG			
Journal	JRN	CRT DLT		CHG	DSP		
Journal attributes	JRNA				DSP		
Journal entry	JRNE						SND
Journal receiver	JRNRCV	CRT DLT					DSP (see DSPJRNRCVA)
Journal physical file	JRNPF						END (see ENDJRNPF)
Journaled changes	JRNCHG		RMV				APY
Library	LIB	CRT DLT			DSP		CLR SAV RST
Library list	LIBL				DSP		RPL
Line	LIN						ANS VRY
Line description	LIND	CRT DLT		CHG	DSP		
Line status	LINSTS				DSP		
Log(ging)	LOG				DSP		END
Logical file	LF	CRT[1]		CHG			DLT (see DLTF) DSP (see DSPFD) OVR (see OVRDBF)
Logical file member	LFM		ADD	CHG			
Member	M		RMV				
Message(s)	MSG		RMV		DSP		MON SND RCV RTV
Message description	MSGD		ADD RMV	CHG	DSP		
Message file	MSGF	CRT DLT		OVR	DSP		DLT (see DLTF)
Message queue	MSGQ	CRT DLT		CHG			
Object	OBJ						ALC CHK DLC DMP MOV RNM SAV SAVCHG RST
Object authority	OBJAUT				DSP		GRT RVK
Object description	OBJD				DSP		
Object lock	OBJLCK				DSP		
Object owner	OBJOWN			CHG			
Output queue	OUTQ	CRT DLT		CHG	DSP	HLD RLS	CLR
Override	OVR	DLT			DSP		
Physical file	PF	CRT		CHG			DLT (see DLTF) DSP (see DSPFD), JRN OVR (see OVRDBF)
Physical file member	PFM		ADD	CHG			CLR INZ RGZ
Pointer	PTR			CHG			
Print image	PRTIMG	CRT DLT					

[1]See also Data Base File.

Items		Actions					
Items Affected	**Item Abbrev.**	**Create/ Delete**	**Add/ Remove**	**Change/ Override**	**Display**	**Hold/ Release**	**Other Actions**
Printer	PRT						CLN VFY
Printer file	PRTF	CRT		CHG OVR			DLT (see DLTF) DSP (see DSPFD)
Printer writer	PRTWTR						STR (see also Writer)
Problem determination procedure	PDP						STR
Program	PGM	DLT	ADD RMV				END PCH
Program message	PGMMSG						SND
Program references	PGMREF				DSP		
Program variable	PGMVAR			CHG	DSP		
Programming change	PGMCHG		RMV		DSP		APY LOD
Query application	QRYAPP	CRT DLT					DSN
Query definition	QRYDEF	CRT		CHG			
Query source	QRYSRC						RTV
Reader	RDR				DSP	HLD RLS	CNL
Receive	RCV						CNL
Reply	RPY						SND
Request	RQS						CNL
Resources	RSC						RCL
RJE communications entry	RJECMNE		ADD RMV	CHG			
RJE console	RJECSL						STR
RJE data	RJEDTA						FMT
RJE job	RJEJOB						SBM
RJE reader	RJERDR						STR CNL
RJE reader entry	RJERDRE		ADD RMV	CHG			
RJE session	RJESSN						STR TRM
RJE writer	RJEWTR						STR CNL
RJE writer entry	RJEWTRE		ADD RMV	CHG			
Routing entry	RTGE		ADD RMV	CHG			
RPG program	RPGPGM	CRT					DLT (see DLTPGM)
RPT program	RPTPGM	CRT					DLT (see DLTPGM)
Service	SRV						END
Service status	SRVSTS				DSP		
Session description	SSND	CRT DLT		CHG	DSP		
Source	SRC						EDT
Source physical file	SRCPF	CRT		CHG			DLT (see DLTF) DSP (see DSPFD)
Spooled file	SPLF				DSP	HLD RLS	CNL CPY DLT (see DLTF)
Spooled file attributes	SPLFA			CHG	DSP		
Storage	STG						RCL
Submitted jobs	SBMJOB				DSP		
Subsystem	SBS				DSP		STR TRM
Subsystem description	SBSD	CRT DLT		CHG	DSP		

Items		Actions					
Items Affected	**Item Abbrev.**	**Create/ Delete**	**Add/ Remove**	**Change/ Override**	**Display**	**Hold/ Release**	**Other Actions**
System	SYS				DSP		SAV PWRDWN
System object	SYSOBJ						DMP
System status	SYSSTS				DSP		
System value	SYSVAL			CHG	DSP		RTV
Table	TBL	CRT DLT					
Tape	TAP				DSP		DMP INZ
Tape file	TAPF	CRT		CHG OVR			DLT (see DLTF) DSP (see DSPFD)
Trace	TRC		ADD RMV		DSP		
Trace data	TRCDTA				DSP		CLR
User authority	USRAUT						GRT
User profile(s)	USRPRF	CRT DLT		CHG	DSP		RST
Variable	VAR			CHG			
Work station entry	WSE		ADD RMV	CHG			
Writer	WTR				DSP	HLD RLS	CNL

Note: The following commands are all one-word commands that are also part of CL:

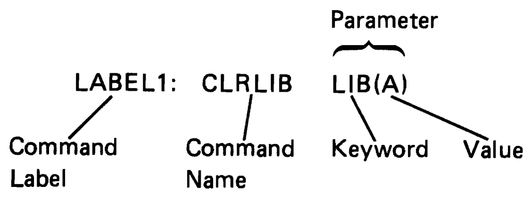
CALL	ELSE	PGM
DATA	GOTO	RETURN
DCL	IF	SIGNOFF
DO	JOB	WAIT

Chapter 2. Control Language Syntax

This chapter describes the control language syntax that you use to code and enter control language commands. Each CL command is processed by the CPF to perform the specified command function upon the CPF objects named in the command.

PARTS OF A COMMAND

A CL command is made up of the following parts: command label (optional), command name (mnemonic), and parameters.



Command Label

Command labels identify particular commands in a CL program for branching purposes. Labels can also be used to identify statements in CL programs that are being debugged: they can identify statements used (a) as breakpoints, and (b) as starting and ending statements for tracing purposes.

A command label is entered just before the command name of the command that is to be branched to. The label can contain as many as 10 characters and follows the standard rule for specifying names (see *Rules for Specifying Names*). The label must be immediately followed by a colon, and blanks (though not required) can occur between the colon and the command name. (START: and TESTLOOP: are examples of command labels.)

All commands can have labels. If a label is placed on a nonexecutable command (such as the DCL command) and that label is branched to, the next executable command following the label is executed as a result of the branch. Only one label can be specified on a line (or in a record); if no command is on that line, the next command is executed.

To specify multiple labels, each additional label must be on a line preceding the command as shown:

```
LABEL1:
LABEL2: CMDX
```

No continuation character (+ or -) is allowed on the preceding label lines.

Command Name

The command name identifies the function to be performed by the program that is invoked when the command is executed. The command name (mnemonic) is an abbreviation of the description of what the command does; for example, the mnemonic MOVOBJ identifies the CL command (Move Object) that moves an object from one library to another. (Like other CPF objects, a command name can be optionally qualified by a library name. See *Simple and Qualified Object Names* discussed later in this chapter.)

The IBM-supplied commands are all named in a consistent manner. Generally, three letters from each word in the descriptive command name are used to form the abbreviated command name which is recognized by the system. For examples of how commands and other objects supplied by IBM are named, see *Control Language* in the *CPF Programmer's Guide*.

Command Parameters

Most CL commands have one or more parameters that specify the objects and values to be used in the execution of the commands. The user who enters the command supplies the object names and the values to be used by the command. The number of parameters specified depends upon the command. Some commands (like DO and ENDJOB) have no parameters, and others have one or more.

A *parameter* identifies an individual value or group of values to be used by the command. The specification of a group of values on one parameter is described later under *Lists of Values*.

Most uses of the word *parameter* in this reference manual refer to the combination of the parameter keyword and its value. For example, the MOVOBJ command has a parameter called OBJ that requires an object name to be specified. OBJ is the parameter keyword, and the name of the object is the value to be entered for the OBJ parameter.

A command can have parameters that must be coded (required parameters) and parameters that do not have to be coded (optional parameters). Optional parameters usually have a default value assigned to them by the system if a value is not specified for the parameter when the command is entered.

Parameters in CL can be specified in keyword or positional forms, or in a combination of the two.

Parameters in Keyword Form

A parameter in *keyword form* consists of a keyword immediately followed by a value (or a list of values separated by blanks) that is enclosed in parentheses. No blanks can occur between the keyword and the left parenthesis preceding the value. (Blanks can occur between the parentheses and the value.) For example, LIB(MYLIB) is a keyword parameter specifying that MYLIB is the name of the library that is to be used in some way, depending upon which command this LIB parameter is used in.

When the parameters in a command are specified in keyword form only, they can be specified in any order. For example, in the CRTLIB (Create Library) command, three of its four parameters can be specified in a number of ways, two of which are:

```
CRTLIB  LIB(MYLIB) TYPE(*TEST) PUBAUT(*NONE)
CRTLIB  TYPE(*TEST) LIB(MYLIB) PUBAUT(*NONE)
```

Parameters in Positional Form

A parameter in *positional form* does not have its keyword coded; it contains only the value (or values, if it is a list) whose function when executed is determined by its position in the parameter set for that command. The parameter values are separated from each other and from the command name by one or more blanks. Because there is only one sequence in which parameters can be coded positionally, the positional form of the previous CRTLIB example is:

```
CRTLIB  MYLIB *TEST *NONE
```

Each command having more than one parameter has a specific positional order for its parameters. The correct order is shown in the syntax diagram for each command (in Part 2). However, in the few cases where dependent (or mutually exclusive) parameters occur in the syntax diagram and the positional order is not readily apparent, the correct order can be easily determined from the text, because the parameters are always described in positional order. When parameters are entered positionally, they must be entered in the specified order (or positions), or the parameter values will be associated with the wrong parameters.

If you do not want to enter a value for one of the parameters, the predefined value *N can be entered in that parameter's position. The system recognizes *N as an omitted parameter, and either assigns a default value or leaves it null. In the previous CRTLIB command example, if you coded *N instead of *TEST for the TYPE parameter, the default value *PROD is used when the command is executed, and a production library named MYLIB is created with no public authority. (Refer to the description of the CRTLIB command in Part 2 for the explanation of each parameter.)

Note: Parameters may not be coded positionally beyond the positional coding limit, designated in the syntax diagrams with the symbol . If you do attempt to code positionally beyond that point, the system will respond with an error message. When all parameters of a command can be coded positionally, no positional limit symbol appears in the syntax diagram.

Entering Parameters in Both Forms

A command can also have its parameters coded in both forms. The following examples show three ways to code the Declare CL Variable (DCL) command.

Keyword form:

```
DCL  VAR(&QTY)  TYPE(*DEC)  LEN(5)  VALUE(0)
```

Positional form:

```
DCL  &QTY  *DEC  5  0
```

Positional and keyword forms together:

```
DCL  &QTY  *DEC  VALUE(0)
```

In the last example, because the optional LEN parameter was not coded, the VALUE parameter *must* be coded in keyword form. There are certain restrictions that apply when parameters are entered in both forms. Refer to the *CPF Programmer's Guide* for details.

COMMAND SYNTAX

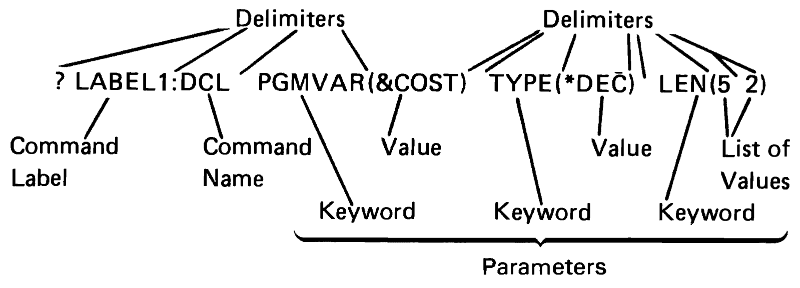
A command has the following general syntax. The brackets indicate that the item within them is optional; however, the parameter set may or may not be optional, depending upon the command.

```
[//] [?] [label-name:] command-name[.library-name]  [parameter-set]
```

The // is valid only for a few batch job control commands, such as the DATA command. The // identifies these commands to the spooling reader that reads the batch job input stream.

Command Delimiters

Delimiters are special characters that mark the beginning or end of a group of characters. Delimiters are used to separate a character string into its individual parts that together form a command: command label, command name, parameter keywords, and parameter values (which can be constants, variable names, lists, or expressions).



The following delimiters are used in the CPF control language.

- The colon (:) separates the command label from the command name. (For example, LABEL1:DCL and LABEL2: DCL are both valid.)

- Blanks separate the command name from parameters and separate parameters from each other. They also separate values in a list. Multiple blanks are treated as a single blank except in a quoted string or comment. A blank *cannot* separate a keyword and the left parenthesis for the value.

- Parentheses ( ) separate parameter values from their keywords, group lists of values, and group lists within lists.

- Periods connect the parts of a qualified name. For a qualified object name, the two parts are the object name and the library qualifier (OBJA.LIBX). Qualified object names are described in *Identifying CPF Objects* later in this chapter.

- Either a period or a comma can be used as a decimal point in a decimal value (3.14 or 3,14); only one per decimal value is allowed.

- Apostrophes specify the beginning and end of a quoted character string, which is a combination of any of the 256 EBCDIC characters that are used as a constant. For example, 'YOU CAN USE $99@123.45 ()*></ and lowercase letters' is a valid quoted string. An apostrophe used within a quoted string must be specified as two apostrophes.

- One of four special characters can be used as date separators to separate a date into three parts: month, day, and year (two parts for Julian dates: year and day). The four date separators are the slash (/), hyphen (-), period (.), and comma (,). The special character coded in a command must be the same as the special character specified in the QDATSEP system value.

- The colon (:) is the only special character that can be used as a time separator. It can be used to separate a time value into two or three parts (hours, minutes, and seconds).

- The characters /* and */ indicate the beginning and end of a comment.

- A question mark (?) preceding the command name indicates that the command is to be prompted. If the command is specified with a label, the question mark can precede the label, or follow the label and precede the command name.

  Within a CL program, when a question mark precedes a command name, a prompt display is presented to the user who called the program in which the command is encountered. The user can enter values for parameters for which values were not specified on the command in the program.

Command Continuation

Commands can be entered in free format. That is, a command does not have to begin in a specific location on a coding sheet, on the display, or in cards. A command can be contained entirely in one record, or it can be continued on several lines or records. (Whether continued or not, the total command length cannot exceed 3000 characters.) Either of two special characters is entered as the last nonblank character on the line to indicate that a command is to be continued: the plus sign (+) or the minus sign (-). Any blanks immediately *preceding* a + or - sign are always included; any blanks immediately following a + or - in the *same record* are ignored. Any blanks in the *next record* that precede the first nonblank character in the record are ignored when + is specified and included when - is specified.

The + is generally of use between parameters or values. (At least one blank must precede the + sign when it is used between separate parameters or values.) The difference between the plus and minus sign usage is particularly important when continuation occurs within a quoted character string. The following example shows the difference:

```
CRTLIB  LIB(XYZ)  TEXT('This is CONT-         The minus sign causes the
ƀƀƀINUED')                        or +        leading blanks on the next
                                              line to be entered.
```

```
For -: CRTLIB  LIB(XYZ)  TEXT('This is CONTƀƀƀINUED')

For +: CRTLIB  LIB(XYZ)  TEXT('This is CONTINUED')
```

Entering Comments

Comments can appear outside of a command, or within a command wherever a blank is permitted; that is, both outside and *inside* the character string that makes up a command. However, because a continuation character defines the end of a line (or record), comments *cannot* follow a continuation character on the same line.

For readability, it is recommended that each comment be specified on a separate line preceding or following the command it describes, as shown here:

```
MOVOBJ  OBJA TOLIB(LIBY)
           /* Object OBJA is moved to library LIBY. */
DLTLIB  LIBX
           /* Library LIBX is deleted. */
```

Comments can include any of the 256 EBCDIC characters. However, the character combination */ should not appear within a comment because these characters terminate the comment.

Note: The characters /* in positions 1 and 2 of an input record from the MFCU (multi-function card unit) is recognized as an end-of-file terminator. The delimiter for comments should not begin in columns 1 and 2 in commands entered via the MFCU.

CONTROL LANGUAGE CHARACTER SET

The CPF control language uses the extended binary coded decimal interchange code (EBCDIC) character set. For convenience in describing the relationship between characters used in the control language and the EBCDIC character set, the following CPF control language categories contain the EBCDIC characters shown:

Category	**Characters Included**
Alphabetic[1]	26 letters (A-Z), a-z, and $, #, and @
Numeric	10 digits (0-9)
Alphameric[2]	A-Z, a-z, 0-9, and $, #, @, and __
Special characters	All other EBCDIC characters (for those having special uses in CL, see *Summary of Special Character Usage*)

[1]Lowercase letters (a-z) are accepted, but they are translated into the corresponding uppercase letters by the system, except when included within a quoted character string or a comment. In the Katakana EBCDIC character set, the character positions corresponding to a-z in the US character set contain Katakana characters that can be used as data in quoted strings or comments; if those same characters are used outside quoted strings or comments, they are translated to A-Z.
[2]The underscore (__) is an alphameric connector that can be used to connect words or alphameric characters to form a name (for example, PAYLIB__01). This use of the underscore might not be valid in other high-level languages.

The first three categories contain the characters that are allowed in quoted and unquoted character strings, in comments, and in CL names, such as in names of commands, labels, keywords, variables, and CPF objects. All the special characters, in the last category, can only be used in quoted character strings and comments; they cannot be used in unquoted strings. However, some have special syntactical uses when coded in the proper place in CL commands. These uses are given in the chart under *Summary of Special Character Usage*.

SPECIAL CHARACTERS AND PREDEFINED VALUES

This section summarizes in chart form all of the special characters and their uses in the CPF control language. A description of predefined values and how they are used is also given.

Summary of Special Character Usage

The following special EBCDIC characters are used by the control language in various ways. They are most frequently used as delimiters (which were covered previously) and as symbolic operators in expressions (see Appendix B). Special characters can only be used in these special ways or within quoted character strings or comments. The special characters have the following assigned meanings when coded in CPF control language commands:

Delimiters

Name	Symbol	Meanings
Blank	ƀ[1]	Basic delimiter for separating parts of a command (label, command name, and its parameters), and for separating values within lists.
Left and right parentheses	( )	Grouping delimiter for lists and keyword values, and for evaluating the order of expressions.
Colon	:	Ending delimiter for command labels. Separates parts of time values.
Comma	,	In many countries, used as decimal point in numeric values. Separates parts of date values.[2]
Period	.	Decimal point; also connects parts of qualified names. Separates parts of date values.[2]
Apostrophes	' '	Quoted character string (a constant) delimiter; apostrophes must be paired.
Slashes	//	Identifying characters used in positions 1 and 2 of JOB, ENDJOB, and DATA commands in job stream. Also, a default delimiter on inline data files.
End of file	/*	Indicates the end of a file on MFCU, when in card columns 1 and 2.
Begin and end comment	/* */	Indicates the beginning and end of a comment. The comment (/*) must not begin in column 1 of cards because the /* in columns 1 and 2 is recognized by the MFCU as the end-of-file delimiter.

[1]In this manual, ƀ is used when necessary to represent a blank space.
[2]Valid only when the QDATSEP system value specifies the same character.

Symbolic Operators

Name	Symbol	Meanings
Plus	+	Addition operator, command continuation character, and positive signed value indicator.
Minus (hyphen)	-	Subtraction operator, command continuation character, and negative signed value indicator. Separates parts of date values.[3]
Slash	/	Division operator. Separates parts of date values.[3]
Asterisk	*	Multiplication operator. Indicates a generic name when it is the last character in the name. Indicates CPF reserved values (predefined parameter values and expression operators) when it is the first character in a string.
Not	¬[1]	Symbolic *not* relational operator.
Equal	=	Symbolic *equal* relational operator.
Less than	<	Symbolic *less than* relational operator.
Greater than	>	Symbolic *greater than* relational operator.
And	&	Symbolic logical operator for AND.
Or	|[2]	Symbolic logical operator for OR.
Concatenation	|> |<	Character string operator (indicates both values are to be joined). See Appendix B for more information on the differences in the concatenation operators.

[1]In some character sets, including the multinational character set, the character ^ replaces the ¬ character. Either ^ or *NOT can be used as the logical NOT operator in those character sets.
[2]In some character sets, including the multinational character set, the character ! replaces the | character. Either ! or *OR can be used as the OR operator, and either !! or *CAT can be used as the concatenation operator in those character sets.
[3]Valid only when the QDATSEP system value specifies the same character.

Note: The symbolic operators can also be used in combinations as listed in the chart under *Operators in Expressions* in Appendix B.

Other Uses

Name	Symbol	Meanings
Ampersand	&	Identifies a CL variable name when it is the first character in the string.
Percent	%	Identifies a built-in function when it is the first character in the string.
Question mark	?	Specifies a prompt request when it precedes a command name.

Predefined Values

Predefined values are IBM-defined fixed values that have predefined uses in the control language and are considered to be reserved in CPF. Predefined values have an asterisk (*) as the first character in the value followed by a word or abbreviation, such as *ALL or *PGM. The purpose of the * in predefined values is to prevent possible conflicts with user-specified values, such as object names. Each predefined value has a specific use in one or more command parameters; each is described in detail in Part 2, under the commands in which it is allowed.

Some predefined values are used as operators in expressions, such as *EQ and *AND. The predefined value *N is used to specify a null value and can be used for any optional parameter. A *null value* (*N) indicates a parameter position for which no value is being specified; it allows other parameters that follow it to be entered in positional form. To specify the characters *N as a character value (not as a null), the string must be enclosed in apostrophes ('*N') to be passed. Also, when the value *N appears in a CL program variable at execution time, it is always treated as a null value.

RULES FOR SPECIFYING NAMES

The standard rule for specifying names used by the control language is:

> Every name must begin with an alphabetic character (A-Z, $, #, or @) and can be followed by no more than 9 alphameric characters (A-Z, 0-9, $, #, @, or __). No name can exceed 10 characters. Blanks are never allowed in a name.

The standard rule applies to CPF object names, CL variable names, command labels, system values, built-in functions, and job names. It also applies to both parts of a qualified object name, which is described in the following section. When you create a new command using command definition (see Chapter 5), the names of the command and its parameter keywords must follow the same standard rule.

Additional rules involving special characters that apply to the following types of names (as an extra character) are:

- A *command label* must be immediately followed by a colon (:). Blanks can follow the colon, but none can precede it.
- A *CL variable name* must be preceded by an ampersand (&) to indicate that it is a CL variable used in a CL program.
- A *built-in function name* must be preceded by a percent sign (%) to indicate that it is an IBM-supplied built-in function, which can be used in an expression.

These special characters are not part of the name; each is an additional character attached to a name (making a maximum of 11 characters) indicating to the system what the name identifies.

The names of CPF objects, CL program variables, system values, and built-in functions can be specified in the parameters of individual commands as indicated in the syntax diagram for each command. (Instead of specifying a constant value, a CL variable name can be used on most parameters in CL programs to specify a value that may change during program execution.) The names, then, identify which objects and values are to be used when the command is executed.

IDENTIFYING CPF OBJECTS

Each of the CPF objects used by the control language has a name. The object name specified in a CL command identifies which object is to be used by the CPF to perform the function of the command.

Simple and Qualified Object Names

The name of a specific object can be specified in two ways: as a simple name or as a qualified name. A *simple object name* is the name of the object only. A *qualified object name* is the name of the object followed by the name of the library in which the object is stored in the system. In a qualified object name, the object name is connected to the library name by a period.

Name Type	Name Syntax	Example
Simple object name	object-name	OBJA
Qualified object name	object-name.library-name	OBJB.LIB1

Either the simple name or the qualified name of an object can be specified if the object exists in one of the libraries named in the job's library list; the library qualifier is optional in this case. A qualified name *must* be specified if the named object is not in a library named in the library list.

Note: A job name also has a qualified form, but it is not a qualified object name because a job is not a CPF object. A job name is qualified by a *user* name and a job number, *not* by a library name. (Refer to the expanded description of the JOB parameter in Appendix A for a complete description of job names.)

Generic Object Names

Another type of object name is the *generic object name*. This type may refer to more than one object. That is, a generic name contains one or more characters that are the first group of characters in the names of several objects; the system then searches for all the objects that have those characters at the beginning of their names and that are in the libraries named in the library list. A generic name is identified by an asterisk (*) as the last character in the name.

A generic name can also be qualified by a library name. If the generic name is qualified, the system searches only the specified library for objects whose names begin with that generic name.

Name Type	Name Syntax	Example
Simple generic name	generic-name*	OBJ*
Qualified generic name	generic-name*.library-name	OBJ*.LIB1

CPF Object Naming Rules

The following rules are used to name all CPF objects used in control language commands. Use these rules, in addition to the standard rule given for all names, to specify the object names indicated in the CL command descriptions in Part 2. (The syntax diagram for each CL command shows whether a simple object name, a qualified name, or a generic name can be specified.)

- *Specifying a Single Object:* In the name of a single object, each part (the simple name and the library qualifier name) can have a maximum of 10 characters. The first character in each part must be alphabetic (A-Z, $, #, or @), and the rest must be alphameric (alphabetic, 0-9, and __). When a library qualifier is used, a period (.) connects the object name to the library name.

- *Naming User-Created Objects:* To be able to distinguish user-created objects from IBM-supplied objects, you should not name your objects with names beginning with Q because the names of all IBM-supplied objects (except commands) begin with Q. Although you can use as many as 10 characters in CL object names, you may need to use fewer to be consistent with the naming rules of the HLL (high-level language) that you are also using. Also, the HLL might not allow underscores in the naming rules. For example, RPG limits file names to 8 characters and does not allow underscores.

- *Specifying a Generic Object Name:* In a generic name, a maximum of 9 alphameric characters can be used, not including the asterisk (*) that must immediately follow the last character. The first character must be alphabetic. Generic names are not valid in some commands. In commands where a generic name is accepted, a regular name is also accepted (that is, without the *).

Name Type	Name Syntax		Examples
	Object Name	Library Name	
Simple object name	object-name		INVENPGM1
Qualified object name	$\underbrace{\text{object-name}}_{\text{10 characters maximum}}\text{.library-name}$ Connector		INVENPGM2.QGPL
Generic name	generic-name* Asterisk Connector		INV*
Qualified generic name	$\underbrace{\text{generic-name}}_{\text{9 characters maximum}}\text{*.}\underbrace{\text{library-name}}_{\text{10 characters maximum}}$		INV*.QGPL

Valid values where a generic name is accepted are INV and INV*. When the name INV is specified, only the object INV is referenced. When the generic name INV* is specified, objects that begin with INV are referenced, such as INV, INVOICE, INVENTORY, and INVENPGM1.

- *Object Library Qualifier Limitations:* When the object being created is a library, user profile, device description, control unit description, or line description, no library qualifier can be specified with the name. A library name can never be qualified because a library cannot be placed in a library. The other object types (*USRPRF, *DEVD, *CUD, and *LIND) appear as if they exist only in the QSYS library. When only the name of an object of these four object types is accepted, a library qualifier cannot be specified with the object name. On the DSPOBJD command, where any object name is accepted, QSYS can be specified.

- *Library List Qualifiers:* The predefined value *LIBL (and others, such as *USRLIBL and *ALLUSR) can be used in place of a library name in most commands. *LIBL indicates that the libraries named in the job's library list are to be used to find the object named in the first part of the qualified name.

- *Duplicate Object Names:* Duplicate names of objects that are of the same type and in the same library are not allowed.

Two objects having the same name cannot be stored in the same library unless their object types are different. Two objects named OBJA can be stored in the library LIBX only if, for example, one of the objects is a program and the other is a file. The following combinations of names and object types could all exist on the system at the same time.

$$\left.\begin{array}{l}\text{OBJA.LIB1}\\ \text{OBJA.LIB2}\\ \text{OBJA.LIB3}\end{array}\right\}\text{three programs}\qquad \begin{array}{l}\left.\begin{array}{l}\text{OBJA.LIB1}\\ \text{OBJA.LIB2}\end{array}\right\}\text{two files}\\ \left.\text{OBJA.LIB1}\right\}\text{one command}\end{array}$$

If more than one library contains an object by the same name (and both libraries are in the same library list) and a library qualifier is not specified with the object name, the first object found by that name is used. Therefore, when you have multiple objects of the same name, you should specify the library name with the object name or ensure that the appropriate library occurs first in the library list. For example, if you are testing and debugging and choose not to qualify the names, ensure that your test library precedes your production library in the library list.

Default Libraries

In a qualified object name, the library name is always optional. If a library name is not specified, the default given in the command's description is used (usually either QGPL or *LIBL). If the named object is being created, QGPL is the default; when the object is created, it is placed in the QGPL library (the general purpose library). For objects that already exist, *LIBL is the default for most commands; the job's library list is used to find the named object. The system will search all of the libraries currently in the library list until it finds the object name specified. (Of course, the library in which the desired object is contained must be a part of the job's library list.)

PARAMETER VALUES

Parameter values are user-supplied information to be used during command execution. An individual value can be specified in any one of these forms:

- Constant (its actual value): The types of constants are: character string (includes names), decimal, and logical.

- CL variable name (the name of the variable containing the value): The types of variables are: character string (includes names), decimal, and logical. The type of variable must match the type of value expected for the parameter, except that any type of value can be specified by a character variable. For example, if a decimal value is expected, it can be specified by a character variable as well as by a decimal variable.

- Expression (the value used is the result of evaluating an expression): The types of expressions are arithmetic, character string, relational, and logical. Expressions can be used as a value for parameters in commands in CL programs only.

A parameter can specify one or a group of such values, depending on the parameter's definition in a command. If a group of values is allowed, the parameter is called a *list parameter* because it can contain a list of values.

All values can be specified in the command parameters in keyword form, positional form, or a combination of both forms. Parameter values must be enclosed in parentheses if:

- A keyword precedes the value.

- The value is an expression.

- A list of values is specified. If only one value is specified for a list, no parentheses are required.

A description of each type of parameter value is given in the following paragraphs.

Constant Values

A constant is an actual numeric value or a specific character string whose value does not change. Three types of constants can be used by the control language: character (quoted and unquoted character strings), decimal, and logical.

Character Strings

A *character string* is a string of any EBCDIC characters (alphameric and special) that are used as a value. A character string can have two forms: quoted string or unquoted string. Either form of character string can contain as many as 2000 characters.

A *quoted* character string is a string of alphameric and special characters that are enclosed in apostrophes. For example, 'Credit limit has been exceeded.' is a quoted character string.

The quoted string is used for character data that is not valid in an unquoted character string. For example, user-specified text can be entered in several commands to describe the functions of the commands; the text must be enclosed in apostrophes if more than one word is used in the description because blanks are not allowed in an unquoted string.

An *unquoted* character string is a string consisting of only alphameric characters and the special characters that are shown in the *Unquoted String* column of the table on the following page. The special characters allow the following to be unquoted character string values:

- Predefined values (* at the beginning)
- Qualified object names (.)
- Generic names (* on end)
- Decimal constants (+, -, ., and ,)

Any of these unquoted strings can be specified for parameters that are defined to accept character strings. In addition, some parameters are defined to accept only predefined values, names, or decimal values, or a combination of the three.

The following table summarizes the characters valid in unquoted and quoted character string values. An X in the column indicates the character on the left is valid; a superscript number next to the X indicates the character is valid in the way described in the corresponding note listed following the table:

Name of Character	Character	Unquoted String	Quoted String
Blank	ƀ		X
Comma	,	Note 1	X
Dollar sign	$	X	X
Number sign	#	X	X
At sign	@	X	X
Letters (uppercase)	A-Z	X	X
Letters (lowercase)	a-z	Note 2	X
Digits	0-9	Note 1	X
Period	.	Notes 1 and 3	X
Left parenthesis	(	Note 4	X
Right parenthesis	)	Note 4	X
Ampersand	&	Note 5	X
Asterisk	*	Notes 5 and 6	X
Semicolon	;		X
Minus	-	Notes 1 and 5	X
Slash	/	Note 5	X
Apostrophe	'		Note 7
Equal	=	Notes 5 and 8	X
Less than	<	Notes 5 and 8	X
Greater than	>	Notes 5 and 8	X
Plus	+	Notes 1 and 5	X
Vertical bar	|	Notes 5 and 8	X
Not	¬	Notes 5 and 8	X
Percent	%		X
Question mark	?		X
Colon	:		X
Underscore	_	Note 9	X
Other EBCDIC characters			X

Notes:

1. An unquoted string of all numeric characters, an optional single decimal point (. or ,), and an optional leading sign (+ or -) is a valid unquoted string. Depending on the parameter attributes in the command definition, this unquoted string is treated as a numeric or character value. On the CALL command or in an expression, this unquoted string is treated as a numeric value; a quoted string is required if the character representation is desired. Numeric characters used in any combination with alphameric characters is also valid in an unquoted string.
2. In an unquoted string, lowercase letters are translated into uppercase letters.
3. A period can be used as a connector in qualified names.
4. In an unquoted string, parentheses are valid when used to delimit keyword values and lists or in expressions to indicate the order of evaluation.
5. In an unquoted string, the characters +, -, *, /, &, |, ¬, <, >, and = are valid by themselves. If they are specified on a parameter that is defined in the command definition with the EXPR(*NO) attribute, they are treated as character values. If they are specified on a parameter that is defined in the command definition with the EXPR(*YES) attribute, they are treated as expression operators.
6. In an unquoted string, the asterisk is valid when followed immediately by a name (such as in a predefined value) and when preceded immediately by a name (such as in a generic name).

7. Because an apostrophe within a quoted string is paired with the opening apostrophe (delimiter) and is interpreted as the terminating delimiter, an adjacent pair of apostrophes must be used within a quoted string to represent an apostrophe that is not a delimiter. When characters are counted in a quoted string, such a pair of adjacent apostrophes is counted as a single character.
8. In an unquoted string, the characters <, >, =, ¬, and | are valid in some combinations with another character in the same set. Valid combinations are: <=, >=, ¬=, ¬>, ¬<, ||, |<, and |>. If the combination is specified on a parameter that is defined in the command definition with the EXPR(*NO) attribute, then it is treated as a character value. If it is specified on a parameter that is defined in the command definition with the EXPR(*YES) attribute, then it is treated as an expression operator.
9. In an unquoted string, the underscore is not valid as the first character or when used by itself.

The following are examples of quoted string constants:

Constant	**Value**
'1,2,'	1,2,
'DON''T'	DON'T
'24 12 20'	24 12 20

The following are examples of unquoted strings:

Constant	**Meaning**
CHICAGO	CHICAGO
FILE1	FILE1
*LIBL	Library list
PGMA.LIBX	Program PGMA in library LIBX
1.2	1.2

Decimal Values

A decimal value is a numeric string of one or more digits, optionally preceded by a plus (+) or minus (-) sign. A decimal value can contain a maximum of 15 digits, of which no more than nine can follow the decimal point (which can be a comma or a period). Therefore, a decimal value can have no more than 17 character positions, including the sign and decimal point. The following are examples of decimal values.

```
123.                                +.017
  1.23 } Equivalent            6278,954374
  1,23 } Values             -123456.987654321
 -1,23                      87654321.123
```

Logical Values

A logical value is a single character 1 or 0 enclosed in apostrophes. It is often used as a switch to represent a condition such as on or off, yes or no, and true or false. When used in expressions, it can be optionally preceded by *NOT or ¬. The following are examples of logical values:

Constant	Value	Meaning
'0'	0	Off, no, or false
'1'	1	On, yes, or true

Hexadecimal Values

A hexadecimal value is a constant that is made up of a combination of the hexadecimal digits A through F and 0 through 9. All character strings except names, dates, and times can be specified in hexadecimal form. To specify a hexadecimal value, the digits must be specified in multiples of two, be enclosed within apostrophes, and be preceded by an X. Examples are: X'F6' and X'A3FE'.

Note: Care should be used when hexadecimal values in the range of 00 through 3F, or the value FF, are entered. If data containing these characters is displayed or printed, undesirable results on the device may occur, because they may be treated as device control characters.

Variables

A *variable* contains a data value that can be changed during program execution. The variable is used in a command to pass the value that it contains at the time the command is executed. The change in value can be the result of: receiving the value from a data area, a display device file field, or a message; being passed as a parameter; executing a CHGVAR command within the program; or calling another program that returns a value.

The variable name identifies a value to be used; the name points to where the actual data value is. Because CL variables are valid only in CL programs, they are often called *CL program variables* or, simply, CL variables. CL variable names must begin with an &.

CL variables can be used to specify values for almost all parameters of CL commands. When a CL variable is specified as a parameter value and the command containing it is executed, the current value of the variable is used as the parameter value. That is, the variable value is passed as if the user had specified the value as a constant.

Because it is generally true that CL variables can be used for most parameters of commands in CL programs, the command descriptions in Part 2 of this manual usually do not mention CL variables. For those parameters that are restricted to constants only (such as in the DCL command), to CL variables only (such as all of the parameters of the RTVJOBA command), or to specific types of variables (such as on the RTVJOBA or RTVMSG command), the individual parameter descriptions specify those limitations. Otherwise, if the command is allowed in a CL program, CL variables can be used in place of a value, including parameters with only predefined values. For example, a SAVE parameter having only predefined values of *YES and *NO can have a CL variable specified instead; its value can then be *YES or *NO, depending on its value at the time the command is executed.

A CL variable must contain only one value; it may not contain a list of values separated by blanks.

The value of any CL program variable can be defined as one of the following types:

- Character: A character string that can contain a maximum of 2000 characters. The character string can be coded in quoted or unquoted form, but only the characters in the string itself are stored in the variable.

- Decimal: A packed decimal value that can contain a maximum of 15 digits, of which no more than nine can be decimal positions.

- Logical: A logical value of '1' or '0' that represents on/off, true/false, or yes/no.

If value is:	CL variable can be declared as:
Name Date or time Character string	Character
Numeric	Decimal or character
Logical	Logical or character

Expressions

An expression is a group of constants or variables separated by operators that results in a single value. The operators specify how the values are to be combined to produce the single value or result. The operators can be arithmetic, character string, relational, or logical. The constants or variables can be character, decimal, or logical. For example, the expression (&A + 1) specifies that the result of adding 1 to the value in the variable &A is to be used in place of the expression.

Character string expressions can be used in certain command parameters defined with EXPR(*YES) within CL programs. An expression can contain the built-in functions %SUBSTRING (or %SST) and %SWITCH, which are covered in detail in Appendix B. The types of expressions and examples of each are described there.

Lists of Values

A list of values is a series of one or more values that can be specified for a parameter. Not all parameters can accept a list of values. A *list parameter* can be defined to accept a specific set of multiple values that can be of one or more types. Values in the list must be separated by one or more blanks. Each list of values is enclosed by parentheses, indicating that the list is to be treated as a single parameter. (Parentheses are used even when a parameter is specified in positional form.) To determine whether a list can be specified for a parameter, and what kind of list it can be, refer to the description of the parameter under the appropriate command.

A list parameter can be defined to accept a list of multiple like values (a simple list) or a list of multiple unlike values (a mixed list). Each value in either kind of list is called a *list element*. List elements can be constants, variables, or other lists; expressions are not allowed.

- A *simple list* parameter accepts one or more values of the type allowed by a parameter. For example, (RSMITH BJONES TBROWN) is a simple list of three user names.

- A *mixed list* parameter accepts a fixed set of separately defined values that are in a specific order. Each value can be defined with specific characteristics such as type and range. For example, LEN(5 2) is a mixed list where the first element (5) gives the length of a field and the second element gives the number of decimal positions in the same field.

  LOC(*M1 4 6) is a mixed list of three elements: the first element is a predefined character value (*M1) that indicates a magazine location in the diskette magazine drive; the second and third elements (4 and 6) are numeric values that identify the starting and ending diskette positions within the magazine identified by the first element. This example indicates that diskettes 4, 5, and 6 in magazine 1 are to be used.

- For many parameters defined to accept lists, predefined single values can be specified in place of a list of values. One of these single values can be the default value, which can be specified or assumed if no list is specified for a simple or mixed list. To determine what defaults are accepted for a given list parameter, refer to the description of the parameter in the command description for which the parameter is defined and used.

  Note: *N cannot be specified in a simple list, but it can be specified in a mixed list. Also, individual parameters passed on the CALL command cannot be lists.

- The maximum level of nesting within lists is three levels, including the first (three nested levels of parentheses).

The following are examples of lists:

```
( )        }
KWD( )     } Null lists
(A)
(A B C)
KWD(A B C)
(1  B  &C)
(A B *N C) ←—(assuming a list of unlike values)
((A B) (1 2)) }
((A B)(1 2))  } Nested lists
```

The last two examples contain two nested lists within a list: the first list has values of A and B; the second has values of 1 and 2. The space between the two nested lists is not required. Blanks are the separators between the values within each list, and the sets of parentheses group the values into lists.

SYNTAX CODING RULES (SUMMARY)

This section contains a summary of general information needed to properly code control language commands.

Delimiters

- Blanks are the basic separators between the parts of a command:
  - Between command label and command name (not required, because the colon (:) is the delimiter).
  - Between command name and first parameter, and between parameters.
  - Between values in a list of values (not required between ending and beginning parentheses of lists within a list).
  - Between the slashes and the name or label of some job control commands, like // ENDJOB (not required).

- Blanks cannot separate a parameter's keyword from the left parenthesis preceding its value(s). When a keyword is used, parentheses must be used to enclose the values; blanks *can* occur between the parentheses and the values. For example, KWD(A) is valid.

- Multiple blanks are treated as a single blank, unless they occur within a quoted string or a comment.

- A colon must immediately follow a command label. Only one label can be used on any command (LABEL1: DCLF).

- Apostrophes must be used to specify the beginning and end of a quoted character string. (If a character string contains special characters, such as blanks, apostrophes are required.) If an apostrophe must be used within the quoted string, two apostrophes must be entered side by side to indicate that it is an apostrophe and not the end of the quoted string.

- Parentheses must be used:
  - On parameters that are specified (coded) in keyword form
  - To group multiple values in a single list, in a positional parameter, or around expressions
  - To indicate a list (of none, one, or several elements) within *another* list

- Sets of parentheses within parentheses can be entered as long as they are paired, up to the maximum of five nested levels in logical expressions or three nested levels in lists of values.

- Comments can appear wherever blanks are permitted, except after a continuation character on the same line or record.

- A plus or minus sign at the end of a line indicates that the command is continued on a following line. Blanks following a + or - sign in the same record are ignored; any blanks in the next record that precede the first nonblank character in the record are ignored when + is specified and included when - is specified. One blank must precede the + sign when it is used between separate parameters or values.

Parameters

- All required parameters must be coded.

- If an optional parameter is not coded, the system uses its default value, if the parameter has one. In the syntax diagram of each command, all default values are indicated by the heavy branch lines that lead to them. If no default value is indicated, then the default varies (depending on other parameter values) and is described in the text, or the action taken does not require that parameter.

- Words or abbreviations specified in capital letters in the command and parameter descriptions must be coded as shown. This is true of all command names (mnemonics), keywords of parameters (if used), and many parameter values. If lowercase letters are coded that are not in quoted strings or comments, they are translated to uppercase.

- Parameters may not be coded positionally past the positional coding limit symbol ⟨P⟩ found in the syntax diagrams (if applicable). If no positional coding limit symbol appears, all parameters in the command may be coded positionally. The order of positional coding is the order in which the parameters are presented in the syntax diagram.

Values

- The first character in all names must be an alphabetic character (A-Z, $, #, @). Names must not exceed 10 characters. (CL variable names and built-in function names can have 11 characters maximum, including the preceding & or % characters.) In some commands, the names of objects can be specified in qualified form (object-name.library-name).

- Predefined values that begin with an asterisk can be used only for the purposes intended, unless included in comments or quoted strings. They include predefined parameter values (*ALL, for example), symbolic operators (*EQ, for example), and the null value (*N).

- Within a CL program, a variable can be specified for all parameters, except where explicitly restricted. The contents of the variable are passed as if the value were specified on the command.

- Within a CL program, a character string expression can be specified for any parameter defined with EXPR(*YES). The resulting value of the expression is passed as if the value were specified on the command.

- Null (omitted) values are specified with the characters *N, which mean that no value was specified and the default value, if one exists, should be used. *N is needed only when another value following the omitted value is being specified as a positional parameter or an element in a list.

- Either a comma or a period can be used to indicate a decimal point in a numeric value. The decimal point is the only special character allowed between digits in the numeric string; there is no delimiter for indicating thousands, for example.

- When repetition is indicated for a parameter:
  - A predefined value is not to be coded more than once in a series of values.
  - As many user-defined values (like names or numeric limits) can be entered as there are different values or names, up to the maximum number of repetitions allowed.

  Note: When you are using parameters that have the same name in different commands, the meaning of (and the values for) that parameter in each command may be somewhat different. Refer to the correct command description for the explanation of the parameter you are using. For some parameters, you can also refer to the *Common Parameter Descriptions* in Appendix A for both general information about a parameter and an expanded description of its values coded in commands.

Part 2. Control Language Command Descriptions

All of the System/38 control language commands are described in detail in Part 2. Generally, each command is described independently of all the other commands; they are not described in functional groupings. The commands are in alphabetic order by their command names.

The command definition statements used for creating and changing commands are grouped separately at the end of Part 2, in Chapter 5, *Command Definition Statements*. These five statements perform a function completely independent of the rest of the commands, namely, defining or changing the parameter attributes of IBM-supplied or user-defined commands.

To aid you in quickly locating commands and their parameters, marginal references (similar to that used in a dictionary) are used in the top outer corner of each page in Part 2. Each marginal reference shows the command name and parameter keyword of the first command and first new parameter described on that page. More than one command can appear on one page, but if the command from the previous page is continued, the continued command is the one identified when a new parameter or a new section (such as *Examples*) starts on the page.

Before the first command is described, an explanation of the format used to describe each command is given. Following that, an explanation of how to interpret the syntax diagrams is given; the diagrams graphically show the syntax of each command.

Chapter 3. Format of Command Descriptions

HOW COMMANDS ARE DESCRIBED

Each command description follows the same format. First, the function of the command and restrictions on its use are described. Next, a syntax diagram presents all parameters and values that can be coded on the command. Next, each parameter and its choice of values are described. Finally, coded examples of the command are given. Some commands have additional information that is supplied after the examples.

Command Description

The general description of the command briefly explains the function of the command and any relationships that it has with a program or with other commands. If there are restrictions on the use of the command, they are described under *Restrictions*.

It should be noted that, because a command is a CPF object, each command can be authorized for specific users or authorized for use by the public (all users authorized in some way to use the system). Because this is true for nearly every command, it is not stated in each command description. (Refer to the user profile chart in Appendix C for the IBM-supplied user profiles and the commands initially authorized for each one.)

Command Syntax

The command syntax is presented in the syntax diagram for the command. The syntax diagram shows all parameters and values that are valid for the command. The parameters are divided into two groups: those that must be coded (required), and those that need not be coded (optional). Heavy branch lines are used to indicate default values, which are used by the system for uncoded parameters.

A complete description of the syntax diagram is provided later in this chapter under *How to Interpret Syntax Diagrams*.

Parameter Descriptions

Each parameter is described in the text in the same order as shown in the syntax diagram. The syntax diagram shows the order in which the parameters must be specified if the values are specified positionally (that is, without keywords). If a parameter has more than one value, the values are described in the same order as shown. The default value, if there is one, is always first and is shown as an underlined heading at the beginning of the text that describes the value.

The description of each parameter contains what the parameter means, what it specifies, and the dependent relationships it has with other parameters in the command. When the parameter has more than one value, the information that applies to the parameter as a whole is covered first, then the specific information for each of the values is described after the name of each value.

Command Coding Examples

Each command description shows at least one coded example if the command has at least one parameter. Where necessary, several examples are provided for commands that have many parameters and several logical combinations.

For clarity, each example is coded in keyword form only. The same examples could, of course, be coded in positional form or in a combination of both forms.

Additional Command Considerations

A section called *Additional Considerations* follows the coded examples of some commands when there is additional useful information to be presented about the command. For example, some of the display commands result in displays of information that are in tabular form. This section is used to clarify the meanings of the information displayed.

The displays shown in this manual are only representative of the format of the items that could be displayed for a given command. That is, the manual shows and explains all of the fields that can appear on the displays, and explains the sequence in which the various groups of fields are presented. Xs are shown in the areas where variable information would actually appear, and the length of each field is determined by the number of Xs shown for that field. An *actual* display, in many cases, may contain only a few of the fields that could be displayed, but only the applicable fields are displayed.

HOW TO INTERPRET SYNTAX DIAGRAMS

Syntax diagrams show all the parameters and values used by each CL command. Each syntax diagram specifies, for one command, the parameters that can be coded in the command and the choice of values for each parameter.

All parameters are shown in each diagram in the order that the system requires them to be when the parameters are coded in positional form.

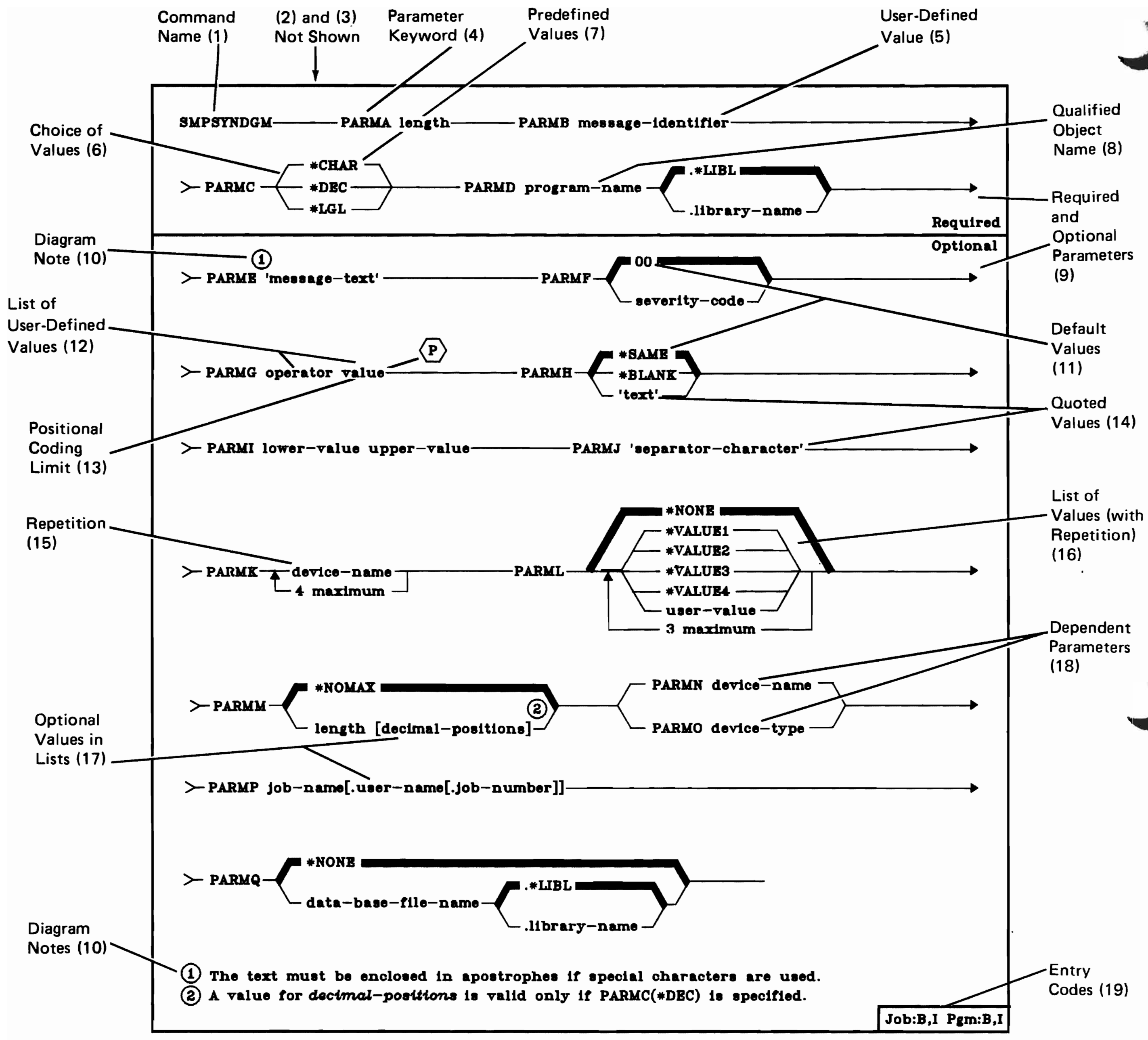
All required parameters precede all optional parameters. The required parameters (if any) are boxed with the command name at the beginning of the diagram. All the other parameters are optional and do not have to be coded; a default value (indicated by heavy branch lines) is assumed for each uncoded parameter, for most commands.

For each parameter that can have a repetition of values, the maximum number of repetitions that can be entered is shown in the diagram with the parameter's values. The syntax diagrams also show (by flow lines and by notes) which parameters are mutually dependent or mutually exclusive.

Entry codes are shown in the bottom right corner of the diagram; they tell you where the command can be entered. Notes are also included that give information that is needed to properly interpret the syntax.

Sample Syntax Diagram

Illustrated on the following page is a sample syntax diagram. It shows the parameter syntax of a fictitious command named SMPSYNDGM (Sample Syntax Diagram). This command shows a number of representative parameters that are used in the set of rules that follow the sample syntax diagram. These parameters are used to illustrate how each kind of parameter syntax that exists in the CL commands is to be interpreted. Included with the rules are coded examples of these parameters.



① The text must be enclosed in apostrophes if special characters are used.
② A value for *decimal-positions* is valid only if PARMC(*DEC) is specified.

Syntax Diagram Rules

The syntax diagrams for the CL commands are interpreted according to the following rules.

1. *Command Name*

The command name appears first in the diagram. In the sample syntax diagram, the name of the fictitious command is SMPSYNDGM.

2. *Command Labels*

All coded commands can be preceded by labels; therefore, they are not shown in the syntax diagrams. If used, a label must have a colon (:) immediately after the last character in the label name.

3. *Parameter Order*

All parameters of the command are shown in the correct positional sequence. The order goes left to right on each line and continues on the following line. (To show the positional order of the parameters in the sample diagram, the representative parameters have keyword names that are named in alphabetic order, such as PARMA and PARMB; they are named in the same order as they would have to be coded positionally if this command actually existed.)

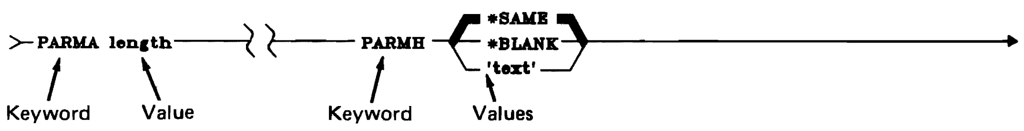
Note: In the few cases where dependent parameter relationships are shown (see rule 18), the positional order of those parameters may not be readily apparent. The order may be specified in a note in the diagram, or the order can be easily determined in the text, because all parameters in the command are described in positional order in the text.

When coding parameters in the positional form, you must enter them in the order shown in the diagram. If you choose not to code a parameter and another positional parameter is to be coded after it, then you must enter *N to represent the uncoded parameter in the positional sequence.

No parentheses are shown in the diagrams, but parentheses must be coded in each parameter that either has the keyword coded with its value (see rule 4) or has multiple values in one parameter (see rule 12).

4. *Parameter Keywords*

For each parameter, the keyword is always shown first, followed by the parameter values. Parameter keywords use uppercase letters; if you code them in lowercase, they will be changed to uppercase.

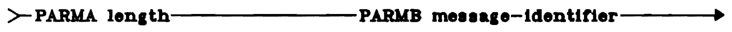


When a parameter is coded in keyword form, its associated values must be enclosed in parentheses. Although parentheses are not shown in the diagram, they must be used when you are coding in keyword form. Examples of PARMA and PARMH are:

PARMA(15) PARMH(*BLANK)

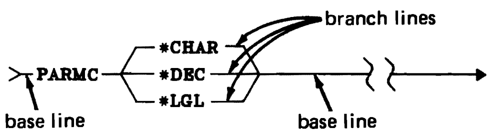
5. *User-Defined Values*

User-defined values are shown with lowercase characters that describe the kind of value to be coded by the user. If more than one word is used to describe a single value, the words are connected by hyphens:



6. *Choice of Values*

A parameter having a choice of values of which only one can be specified is shown with the values on different *branch* lines that occur after the parameter keyword (which is on the base line), as follows:



If the second value is to be coded, it can be coded as:

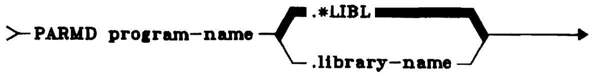
PARMC(*DEC) (keyword form)	or as	*DEC (positional form)

7. *Predefined Values*

Predefined values are shown exactly as they must be coded. *CHAR and *DEC are examples of predefined values.

8. *Qualified Object Names*

Qualified names of CPF objects are shown as:



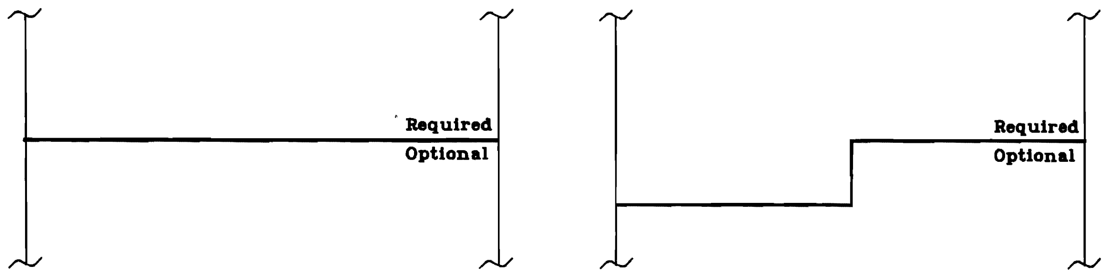
Qualified object names have the object name followed by the optional library qualifier. If the qualifier is not specified, the default shown by the heavy line is used. Usually, *LIBL is the default value for a qualified object name; it means that the library list associated with the job is used to find the object. The syntax for PARMD shows that a qualified program name can be specified.

PARMD(PGMX.LIBA)   PARMD(PGMX)

The first example shows PGMX coded in its qualified form; the parameter specifies that the program named PGMX in library LIBA is to be used. The second example shows the program name only; the library list must be used to find a program by the name of PGMX.

9. *Required and Optional Parameters*

All required parameters, if any, occur before the optional parameters. The required parameters, with their keywords and values, are separated from the optional parameters by a heavy dividing line. The required parameter area is identified by *Required* above the dividing line. The optional parameter area is identified by *Optional* below the dividing line.



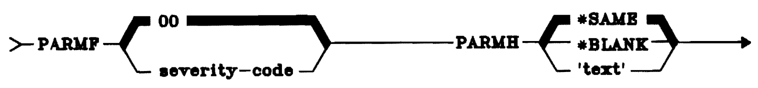
If there are no required parameters, no dividing line is used. If there are required parameters but no optional parameters, *Required* is entered at the top of the diagram.

10. *Diagram Notes*

Any necessary notes about a parameter or value in the diagram are identified by a circled number and explained at the bottom of the diagram.

11. *Default Values*

If parameters have default values, those values always lie within a heavy branch line, except when the values are shown as a list in a box. The default is always the top value in the group shown. When the values are boxed (such as in the LOC parameter of most diskette commands) for a parameter, its default value is underlined and is the first value in the list. Also, the default value is underlined in the text heading that describes the value. For an example of a boxed default value, see the LOC parameter for the Change Diskette File (CHGDKTF) command.



In PARMF, the default value is 00. In PARMH, it is *SAME. The default values are assumed by the system if you do not enter other values for parameters PARMF and PARMH.

Default values occur for most optional parameters, and for the library qualifier portion of both required and optional parameters. The indicated default value is assumed by the system if:

- A value is not specified for an optional parameter.
- A list element (value), in an optional parameter that allows a mixed list of values, is not specified.
- A library name is not specified in the library portion of a qualified object name.

12. *List of User-Defined Values*

If a list of user-defined values can be coded, spaces (blanks) are used to separate the values in the list. A list of values is shown as:

```
>─PARMG operator value───┤ ├───PARMI lower─value upper─value───►
```

Both PARMG and PARMI show a list of two values that must be coded if the parameter is coded. Parentheses are required around the list if multiple values are coded even if no keyword is used. PARMG and PARMI could be coded as:

```
PARMG(*EQ 16)        PARMI(1 9999)
```

13. *Positional Coding Limit*

The point to which the command's parameters can be coded positionally is indicated with this symbol. An attempt to code positionally beyond this point will result in a syntax error. If this symbol does not appear in the syntax diagram, all parameters of the command may be coded positionally.

14. *Quoted Values*

User-defined values that may require that the value be enclosed in apostrophes are shown in the diagram with apostrophes. Apostrophes are shown where special characters are normally expected.

```
>─PARME 'message─text'───┤ ├───PARMJ 'separator─character───►
```

The value specified for PARME requires apostrophes if more than one word is entered (blanks, such as between words, are not allowed in an unquoted character string) or if special characters are used. PARMJ requires apostrophes if a character other than an alphameric character is specified.

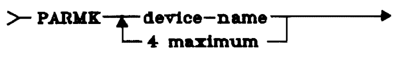
```
PARME('This is a quoted string')
PARME('10-24-78')
PARME(102478)
```

The first and second values require apostrophes because they have either blanks (spaces) or special characters (-). The third value is a date with no separator characters, and therefore does not require apostrophes.

15. *Repetition*

An arrow going back to the left ↑└ (n maximum) ┘ is used to show how many values (shown on several branch lines following the keyword) can be specified, or how many repetitions of one value (shown on one line) can be specified.

If a value of one kind can be specified more than once, it is shown as:



As many as four device names can be specified for PARMK. The following values could be coded:

PARMK(DSPSTN1 DSPSTN2)
PARMK(MFCU DKT1 PTR1 WSPTR1)

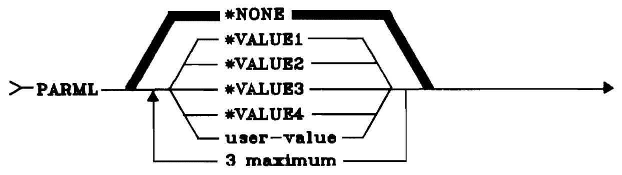
The first example specifies two device names, and the second example specifies the maximum of four device names.

When repetition is indicated for a parameter:

- A predefined value should not be coded more than once in a series of values.
- As many user-defined values (like names or numeric limits) can be entered as there are different values or names, up to the maximum number of repetitions allowed.

16. *List of Values (with Repetition)*

A parameter that can have several values specified (a list of like values) is shown as:



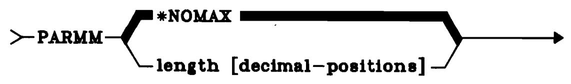
The parameter PARML can specify one, two, or three of the values shown. Any combination of the four predefined values and user-defined values can be specified. The user-defined value could be any value allowed for that parameter. Any of the following could be coded:

PARML(*VALUE1 *VALUE3)
PARML(*VALUE3 16)
PARML(16 3 12)

If PARML is not specified, the single value *NONE is the default used by the system. Note that if *NONE is specified, none of the other values can be specified because the heavy branch line begins on the base line *before* the return point of the repetition arrow, and it returns to the base line *after* the starting point of the arrow. Also, because the arrow does return to the base line after the heavy branch line begins, *NONE cannot be specified if any of the values within the repetition loop are specified.

17. *Optional Values in Lists*

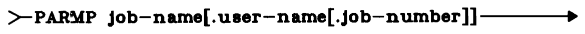
A list of values in which one or more of the elements are optional is shown with brackets ([ ]). The value within the brackets cannot be coded unless the value outside the brackets is also coded. The brackets themselves are never coded.



PARMM has one optional element in a list of two values and can be coded as:

PARMM(15 5)   or   PARMM(7)

The first example specifies a length of 15 digits of which 5 are decimal positions. The second example specifies a length of 7 digits with no decimal positions. If PARMM is not specified, the single value *NOMAX is used as the default.

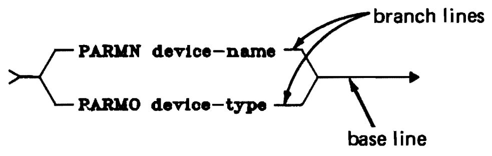


PARMP requires a job name as its value; the name can be optionally qualified. The job name can be specified with only the first part, the first two parts, or with all three parts. A job named JOBX owned by the user RANDY having the job number 210742 can be coded as:

PARMP(JOBX)
PARMP(JOBX.RANDY)
PARMP(JOBX.RANDY.210742)

18. *Dependent Parameter Relationships*

Some parameters or values have dependent relationships with other parameters or values. Some relationships are shown by the placement of whole parameters on one or more *branch* lines; others are indicated by notes that specify the relationships. Also, where the positional order of dependent parameters may not be readily apparent in the diagram, a note is included that specifies the correct positional order of those parameters. The dependent relationships must be considered in the coding process.



PARMN and PARMO are dependent parameters; if a device name (PARMN) is coded, the device type (PARMO) cannot be coded, and vice versa.

Dependencies exist between parameters when they follow one another on the same branch line or when they are on different branch lines that have split from the same base line. (Parameters on the same *base* line, which is usually the case, do *not* indicate dependencies.)

An example of three parameters being dependent on one another is shown in the syntax diagram for the Create Physical File (CRTPF) command. There, the following relationships are shown between the SRCFILE, SRCMBR, and RCDLEN parameters:

- If none of the three are coded, the default values on the top branch line are assumed: SRCFILE(QDDSSRC.*LIBL) and SRCMBR(*FILE).

- If either SRCFILE or SRCMBR is specified (or both of them), RCDLEN cannot be specified; if RCDLEN is specified, SRCFILE and SRCMBR cannot be specified and their defaults do not apply. The two branch lines make them *mutually exclusive*.

- If SRCFILE is specified, SRCMBR may or may not be specified, and vice versa. (In some cases, where parameters are on the same branch line, if one parameter is specified, the following parameter may also *have* to be specified.)

If the CRTPF command is used to create, for example, a physical file named FILEX that is to have records 120 positions long, and if the first few parameters are coded *positionally* (refer to Note 1 at the bottom of the CRTPF syntax diagram), the following would be coded:

```
CRTPF  FILEX *N *N  120
```

19. *Entry Codes (Batch and Interactive)*

The box insert in the lower right corner of each syntax diagram contains the entry codes that specify where the command can be entered. The codes indicate whether the command can be:

- Used within a job (outside a compiled program; Job:B and/or I). When used in this manner, the command is considered a separate entity within the job, and is executed by itself as a separate function (in what is called interpretive mode). That is, commands within batch and/or interactive jobs that are not in compiled programs are interpreted and executed one at a time, one after the other. The function of one interpreted command in the job is performed and completed before the next command is interpreted.

- Used within a compiled program (Pgm:B and/or I). In this case, the command is part of the program: the command is in compiled form with the rest of the program, and the command's execution is dependent on when the program is called and on the program's logic preceding the command. That is, a compiled command cannot be executed unless the program is executed.

The explanations of the combinations of entry codes are shown in the following chart:

Code	**Representing**	**Meaning**
Job:B	Batch job	Valid in batch input stream, external to compiled CL program
Job:I	Interactive job	Valid for interactive entry, external to compiled CL program
Job:B,I	Batch and interactive jobs	Valid for batch and interactive entry, external to compiled CL program
Pgm:B	Program, batch	Valid in compiled CL program that is called from batch entry
Pgm:I	Program, interactive	Valid in compiled CL program that is called from interactive entry
Pgm:B,I	Program, batch and interactive	Valid in compiled CL program that is called from batch or interactive entry

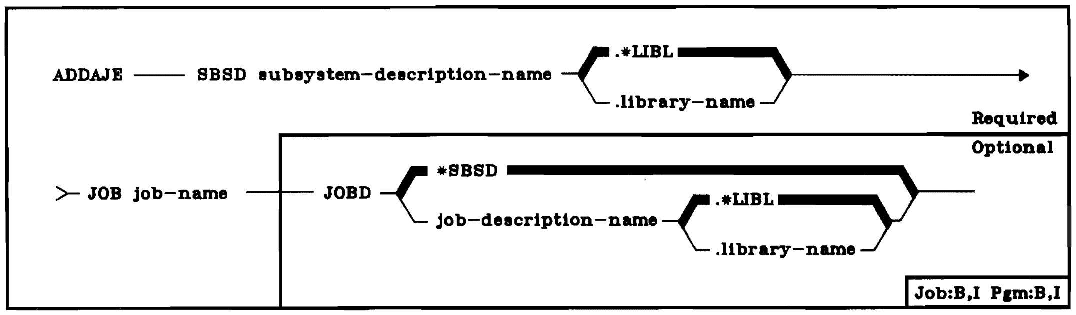
By looking at the entry codes at the bottom of each syntax diagram, you can tell whether the command can be used: only within CL programs (Pgm:B,I), only outside CL programs (Job:B,I), only within interactive jobs (Job:I), or inside or outside a CL program within any batch or interactive job (Job:B,I Pgm:B,I).

Chapter 4. Command Descriptions

ADDAJE (Add Autostart Job Entry) Command

The Add Autostart Job Entry (ADDAJE) command adds an autostart job entry to the specified subsystem description; (the associated subsystem must be inactive at the time). The job entry identifies the job and its associated job description to the subsystem. Autostart jobs are jobs that are automatically initiated when the subsystem is started.

Restriction: To use this command, you must have operational and object management rights for the specified subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description to which the autostart job entry is to be added. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

JOB Parameter: Specifies the simple name of the job that is to be automatically initiated when a subsystem is started using the subsystem description specified in the SBSD parameter.

JOBD Parameter: Specifies the qualified name of the job description to be used for the job that is initiated by this autostart job entry. If the job description does not exist when the entry is added, a library qualifier must be specified because the qualified job description name is retained in the subsystem description.

*SBSD: The job description having the same qualified name as the subsystem description, specified by the SBSD parameter, is to be used for the initiated job.

qualified-job-description-name: Enter the qualified name of the job description that is to be used for the job initiated by this autostart job entry. If no library qualifier is specified, the library list (*LIBL) of the job in which this ADDAJE command is executed is used to find the job description.

Example

```
ADDAJE  SBSD(ACCTINT.ACCTLIB)  JOB(ACCTINIT)  +
      JOBD(INITSBS.ACCTLIB)
```

This command adds the job ACCTINIT as an autostart job entry to the subsystem description ACCTINT in the library ACCTLIB. In this case, the autostart job might be used to perform certain housekeeping functions whenever the subsystem ACCTINT is started. When the subsystem is started, the job description INITSBS in ACCTLIB is used to obtain the attributes for this job and a job named ACCTINIT is automatically started in the subsystem.

ADDBKP (Add Breakpoint) Command

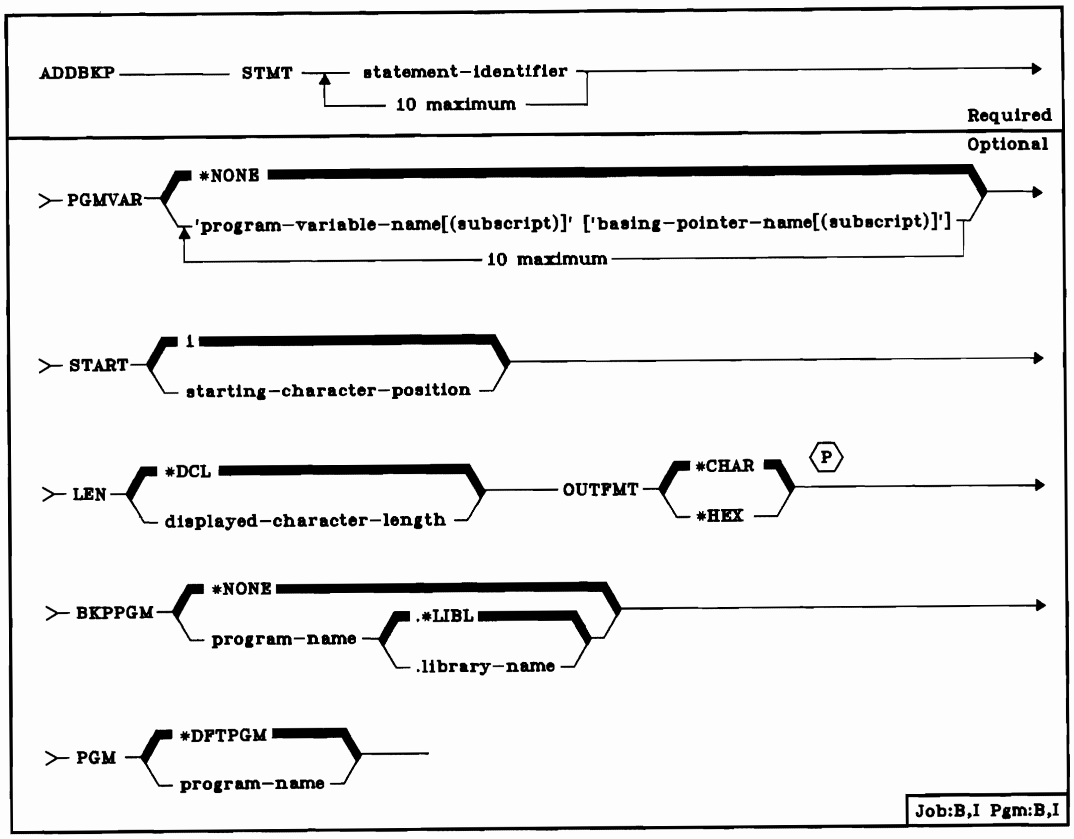
The Add Breakpoint (ADDBKP) command sets one or more breakpoints in a program. A breakpoint is a location in a program where execution of the program stops and control is given to the user. The breakpoint is set when a statement number or label of an HLL command or System/38 instruction is specified. The program is stopped just before executing the statement (or System/38 instruction) on which the breakpoint was set.

The ADDBKP command can also specify that the values of certain program variables are to be displayed or printed when any breakpoint in the command is reached. As many as 10 variables per breakpoint can be specified, and as many as 10 breakpoints per command can be set. However, the same program variables apply to every breakpoint specified in the command. To specify different sets of variables for each breakpoint, you must use different ADDBKP commands.

When a breakpoint is reached in the interactive debugging environment, a display is shown to the user that identifies which breakpoint has been reached and (optionally) the values of the specified program variables when the program is halted. Also, information about the breakpoint only is written to the job log. Control is given to the user so that he can enter another CL command. The RSMBKP command or a command function key may then be used to resume execution of the program.

When a breakpoint is reached in the batch debugging environment, the breakpoint information is written to the job's output queue for printing and, optionally, another program can be called to take action on the breakpoint condition. The name of the called program is specified in the BKPPGM parameter.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*



STMT Parameter: Specifies the statement identifiers of one or more statements or commands (or System/38 instructions) in the program at which breakpoints are to be set. The list can contain a maximum of 10 identifiers (statement numbers, program labels, or System/38 instruction numbers) that are valid for the program specified by the PGM parameter. At least one identifier is needed. If a System/38 instruction number is specified, the number must be preceded by a slash and enclosed in apostrophes: STMT('/21') for example.

PGMVAR Parameter: Specifies the names of one or more program variables (if any) to be displayed that are in an HLL or MI program. The name and the value of each program variable is displayed when any of the breakpoints specified in the STMT parameter are reached.

*NONE: No program variables are to be displayed for any of the breakpoints specified.

'program-variable-name': Enter the names of one or more program variables (no more than 10) to be displayed when a breakpoint is reached. If the variable name contains special characters (such as the & in a CL variable name or the hyphen (-) in a COBOL name), it must be enclosed in apostrophes. An example is:

PGMVAR('&VAR2')

An RPG indicator or an MI ODV number can be specified instead of a program variable name. An example of an RPG indicator is: PGMVAR('*IN22'). The ODV number must be preceded by a slash: PGMVAR('/264') for example.

COBOL qualified program variable names may be specified in this parameter. These names have the following syntax:

```
var-name-1 OF/IN var-name-2 OF/IN varname-3...varname-N
```

where varname-N is the last possible variable name that will fit into the input field of the PGMVAR parameter. The input field length for each variable in the PGMVAR parameter is 98 characters. The subscript specified for a qualified variable name may also be a qualified variable name. A qualified variable name (or one with a subscript), including blanks and parentheses, must be contained within the 98-character limit. The 98-character limit includes the necessary keywords (OF/IN) and blanks, but does not include the enclosing apostrophes.

'program-variable-name(subscript)': For variables in an array, enter the name of the variable and (optionally) the subscript representing the positional element in the array that is to be displayed. If a subscript is not specified, all elements in the array are displayed. The subscript, if specified, must be enclosed in parentheses, and the variable name and subscript number must be enclosed in apostrophes. No more than 10 sets can be specified, and blanks must separate each set. An example is:

```
PGMVAR('A(5)' 'B(5)' 'C(5)')
```

Either an integer or another variable name can be specified for each subscript.

For COBOL variable names, any combination of variable name length and subscript length that will fit into the 98-character limit is valid. For example, one qualified variable name 98 characters in length (including the keywords OF or IN) can be used with no subscript, or a one-character variable name may be used with a qualified variable name (used as a subscript that uses the other 97 spaces, including parentheses).

For COBOL, the following apply:

- Variable names used in qualifying strings must be high-level language variable names (qualification with ODVs is not allowed).
- Either keyword (OF or IN) is allowed.
- Each OF/IN keyword must be separated from adjacent variable names by at least one blank.
- A qualified variable name can be used as a variable subscript.
- The order the variable names are specified must be from the lowest to the highest levels in the structure.

- Structure levels may be skipped; enough levels must be specified, however, to uniquely identify the variable.

- Qualified variable names must be enclosed in apostrophes, since they contain blank characters.

['basing-pointer-name[(subscript)]']: This set of values in the PGMVAR parameter applies only to MI or HLL programs that support based-on variables. The values can optionally be used with either of the previous two choices to also specify the value in an array that is based on a pointer. The same description of the coding syntax applies here. An example is:

```
PGMVAR(('VAR1(5)' 'PTR1(9)') ('VAR2(8)' 'PTR2(11)'))
```

This example shows that one (different) element in each of two program variables is to be displayed. The fifth element in the array named VAR1, which is based on the ninth element in the pointer array named PTR1, and the eighth element in the VAR2 array, based on the eleventh element in the PTR2 pointer array, are to be displayed.

The field length for the basing pointer name is 24 characters.

START Parameter: Specifies, for character variables only, the beginning position in the variable from which its value is to be displayed when the breakpoint is reached. If more than one character variable is specified in the PGMVAR parameter, the same starting position is used for each one.

1: The variable is to be displayed from the first position on through the length specified in the LEN parameter.

starting-character-position: Enter the position number from which the variable is to be displayed. The position number (as well as the *combination* of START and LEN) must be no greater than the length of the shortest variable specified in the PGMVAR parameter.

LEN Parameter: Specifies the number of bytes to be displayed from the character variable specified in the PGMVAR parameter, starting at the position specified in the START parameter. If more than one character variable is specified in the PGMVAR parameter, the same length is used for each one.

*DCL: The character variable is to be displayed to the end of the variable or for 200 bytes, whichever is less.

displayed-character-length: Enter the number of characters that are to be displayed. The length (as well as the *combination* of START and LEN) must be no longer than the length of the shortest variable specified in the PGMVAR parameter.

OUTFMT Parameter: Specifies the format to be used for displaying the variables.

<u>*CHAR:</u> Variables are to be displayed in character form.

**HEX:* Variables are to be displayed in hexadecimal form.

BKPPGM Parameter: Specifies, for batch environment debugging only, the name of the user-supplied program (if any) to be called when a breakpoint is reached in this program. When the program is called, it is passed a character string containing four parameters that identify the program, invocation level, HLL statement identifier, and System/38 instruction number at which the breakpoint occurred. The character string has the following format:

1. Program name (10 bytes). The name of the program in which the breakpoint was reached.
2. Invocation level (5 bytes). The invocation level number of the program in which the breakpoint was reached.
3. Statement identifier (10 bytes). The high-level language program statement identifier that was reached. This statement identifier is the statement identifier specified in the Add Breakpoint (ADDBKP) command that defined the breakpoint. If a System/38 instruction number was used to specify the breakpoint, this parameter contains the System/38 instruction number preceded by a slash (/).
4. Instruction number (5 bytes). The System/38 instruction number that corresponds to the high-level language statement at which the breakpoint was reached. (No slash precedes this System/38 instruction number.) If a System/38 instruction number is passed in the third parameter, the numbers in the third and fourth parameters are the same.

All the parameter values are left-adjusted and padded with blanks. When the called program returns, the program being debugged resumes execution, starting with the statement that has the breakpoint on it.

<u>*NONE:</u> No breakpoint-handling program is to be called when any breakpoint specified in this ADDBKP command is reached in the batch environment. Then the stopped program resumes execution.

qualified-program-name: Enter the qualified name of the user-supplied program to be called when a breakpoint is reached during debug mode in a batch environment. (If no library qualifier is given, *LIBL is used to find the program.) The program specified here should not be the same as the program specified in the PGM parameter. If they are the same, the results are unpredictable. After the called program executes, it returns control to the stopped program, which resumes execution.

PGM Parameter: Specifies the name of the program in which the breakpoints are to be added.

__*DFTPGM:__ The breakpoints are to be added to the program currently specified as the default program in the debugging mode.

program-name: Enter the name of the program to which the breakpoints are to be added. The specified program must already be in debug mode.

Examples

```
ADDBKP  STMT(150 RTN1 205) PGMVAR('&TEMP' '&INREC')
```

This command establishes breakpoints at CL statement numbers 150 and 205 and the label RTN1 in the default program in the debug mode. When any of these breakpoints is reached, the CL variables &TEMP and &INREC are automatically displayed.

```
ADDBKP  STMT(100) PGMVAR('AMOUNT(200)') +
        PGM(MYPROG)
```

Assume in this example that MYPROG is an HLL program being debugged in an interactive environment and that the program variable AMOUNT is a 250-element array in MYPROG. This command adds a breakpoint to statement 100 in MYPROG. When MYPROG is executed, the program halts execution at statement 100 and the value of the 200th element of the AMOUNT array is displayed. (If AMOUNT had been entered without a subscript, the entire array would have been displayed.) The work station user can then enter another command. For MYPROG to resume execution, the RSMBKP can be entered.

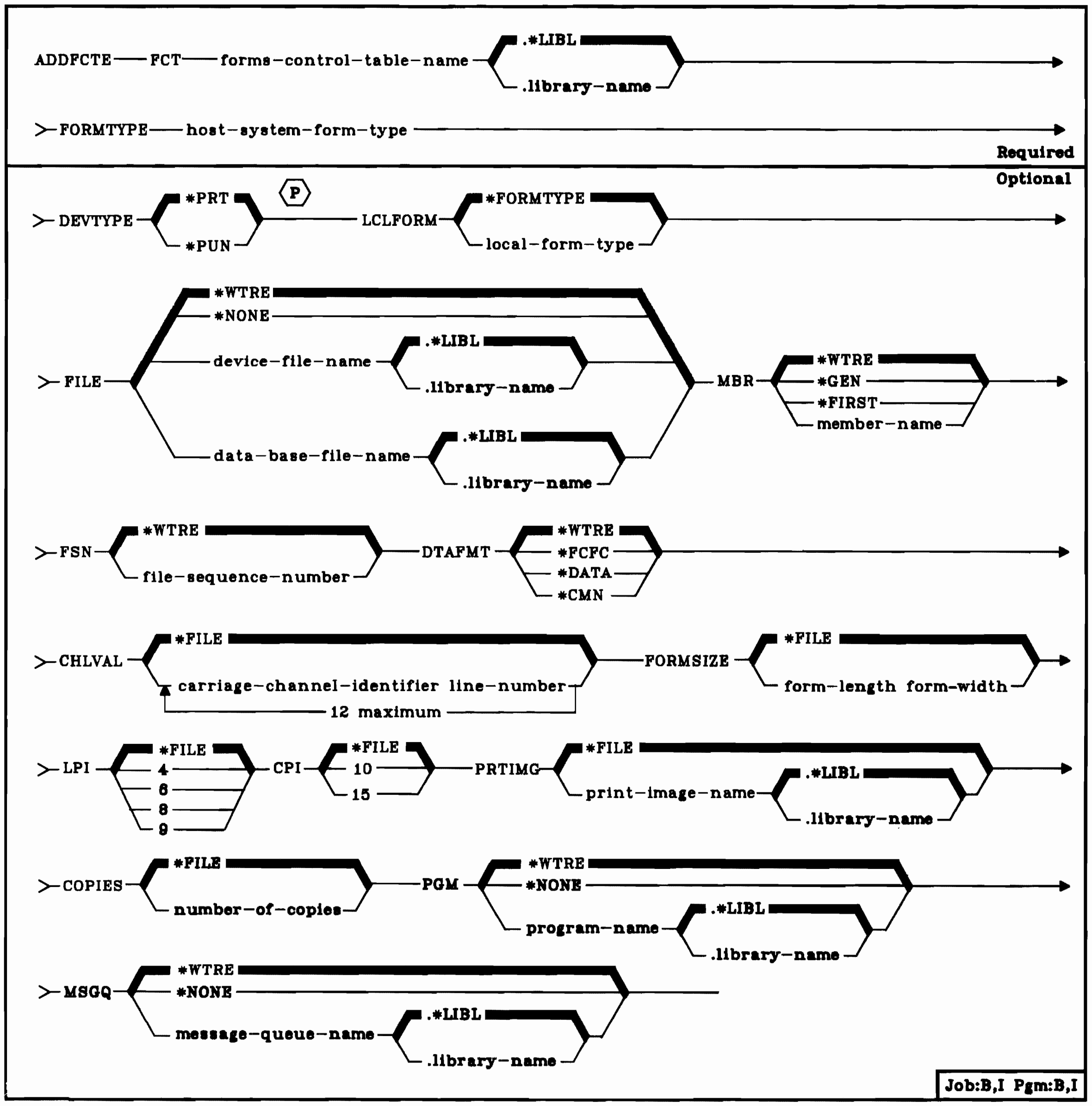
ADDFCTE (Add Forms Control Table Entry) Command

The Add Forms Control Table Entry (ADDFCTE) command adds a new forms entry to an existing forms control table (FCT). The FCT can contain up to 999 entries. Each FCT entry includes such forms-control attributes as:

- Host system form type
- Host system writer type
- Local form type
- Data base file member creation information
- First-character forms-control channel and line number associations
- Form size
- Lines and characters per inch
- Print image
- Number of copies
- User program name

Restriction: To use this command, you must have operational rights for the FCT and read rights for the library in which the FCT is stored.

The Add Forms Control Table Entry (ADDFCTE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.


ADDFCTE FCT forms-control-table-name .*LIBL .library-name
FORMTYPE host-system-form-type
Required
Optional
DEVTYPE *PRT *PUN
P
LCLFORM *FORMTYPE local-form-type
FILE *WTRE *NONE device-file-name .*LIBL .library-name data-base-file-name .*LIBL .library-name
MBR *WTRE *GEN *FIRST member-name
FSN *WTRE file-sequence-number
DTAFMT *WTRE *FCFC *DATA *CMN
CHLVAL *FILE carriage-channel-identifier line-number 12 maximum
FORMSIZE *FILE form-length form-width
LPI *FILE 4 6 8 9
CPI *FILE 10 15
PRTIMG *FILE print-image-name .*LIBL .library-name
COPIES *FILE number-of-copies
PGM *WTRE *NONE program-name .*LIBL .library-name
MSGQ *WTRE *NONE message-queue-name .*LIBL .library-name
Job:B,I Pgm:B,I

FCT Parameter: Specifies the qualified name of the forms control table (FCT) to which the entry is to be added. (If no library qualifier is given, *LIBL is used to find the FCT.)

FORMTYPE Parameter: Specifies the host system form type that is to be associated with the FCT entry. This value (one through eight alphameric characters in length) will be returned by the host system in a forms mount message. A host system form type of blanks can be entered as FORMTYPE('   '). The LCLFORM parameter can be used to change this value to one more understandable to the System/38 user.

DEVTYPE Parameter: Specifies the device type with which the FCT entry is to be associated.

<u>*PRT:</u> This FCT entry can be used only when processing printer output streams.

**PUN:* This FCT entry can be used only when processing punch output streams.

LCLFORM Parameter: Specifies the local form type. This value is to be substituted for the FORMTYPE value used by the host system, to make the forms mount message more understandable to the System/38 user.

<u>*FORMTYPE:</u> No local form type is to be substituted for the host system form type (therefore, the host system form type is to be used).

local-form-type: Enter the name of the local form type to be substituted for the host system form type when the output from the job is actually received. Valid values can be one through ten alphameric characters in length.

FILE Parameter: Specifies the qualified name of the file that is to receive data from the host system.

<u>*WTRE:</u> The file specified in the session description writer entry is to be associated with the FCT entry.

**NONE:* No file is to be associated with the FCT entry. The session description writer entry must be used to determine where the data is to be sent. None of the information in the FCT entry is to be used.

device-file-name: Enter the qualified name of the program-described printer file that is to receive the data. (If no library qualifier is given, *LIBL is used to find the device file.)

data-base-file-name: Enter the qualified name of the System/38 physical file to receive the data. (If no library qualifier is given, *LIBL is used to find the data base file.)

MBR Parameter: Specifies the data base file member to which the output is to be directed (if a data base file was specified either in the FILE parameter of this command or the associated session description writer entry).

**WTRE:* The data base file member is to be generated according to the method specified in the associated session description writer entry.

**GEN:* RJEF creates a member name as follows:

Affffffccc or Bffffffccc

Where:

A	=	file member names beginning with the character A contain print data.
B	=	file member names beginning with the character B contain punch data.
ffffff	=	first six characters of the forms name specified in the FCT or received from the host system. **Note:** Only characters that are valid in a System/38 name are valid in the forms type used to generate data base file member names.
ccc	=	three-digit sequence value controlled by the RJEF session to maintain member uniqueness (refer also to the FSN parameter description of this command).

If a member with this name already exists in the data base file, the three-digit sequence value is incremented by one and another attempt is made to create a member. Incrementing of the sequence value continues until a unique name is generated and a member is created or until all 1000 possibilities have been exhausted without creating a member. If no member is created, the RJEF operator receives a message indicating the failure and a request to retry or cancel this file.

**FIRST:* The output is to be directed to the first member of the data base file (if a data base file is specified in the FILE parameter of this command or in the associated session description writer entry).

member-name: Enter the name of the data base file member to which output is to be directed (if a data base file is specified in the FILE parameter of this command or the associated session description writer entry). If the member does not exist when it is needed, an inquiry message is sent to the RJEF message queue.

FSN Parameter: Specifies the initial three-digit file sequence number to be used when creating data base file member names. This parameter is ignored unless MBR(*GEN) is specified for this command or in the associated session description writer entry.

<u>*WTRE:</u> The initial file sequence number to be used is the same as the number specified in the session description writer entry.

file-sequence-number: Enter the initial three-digit file sequence number to be used. Leading zeros are not required for sequence numbers less than 100.

DTAFMT Parameter: Specifies the format of the output data.

<u>*WTRE:</u> The output data is to be in the format specified in the session description writer entry.

**FCFC:* The output data is to be in the FCFC data format, with the first character of every record being the ANSI forms control code. Specify *FCFC if the data is to be printed. If DEVTYPE(*PUN) is specified, *FCFC is not valid.

The data can be written to a data base file in the FCFC data format and then printed later by issuing the Copy File (CPYF) command and specifying an FCFC printer file on the TOFILE parameter.

**DATA:* The output data is to be in the normal data format (that is, no FCFC characters are embedded in the data). Specify *DATA if the data is to go to a data base file and be processed by a program. If the data is directed to a printer file, a single space ANSI control character is the first character in each record.

**CMN:* The output data is to be in the communications data format (that is, still compressed or truncated). *CMN can be used to decrease communications time. However, before the data can be used, the Format RJE Data (FMTRJEDTA) command must be used to change the data to *FCFC or *DATA. If *CMN is specified, the output file must be a data base file with a length of 256.

CHLVAL Parameter: Specifies the printer carriage channel information.

<u>*FILE:</u> The carriage information specified in the device file is to be used.

carriage-channel-identifier line-number: Enter the channel identifiers and line numbers to be used.

Each identifier can be specified only once per command invocation. The identifiers are 1 through 12, corresponding to printer channels 1 through 12. Single spacing is used for any channel not associated with a line number.

The maximum valid line number is 255.

The CHLVAL parameter associates the channel identifier with a page line number; for example, CHLVAL((1 5)(10 55)) means to associate channel 1 with line 5 and channel 10 with line 55.

FORMSIZE Parameter: Specifies the form size to be used on the System/38 printer.

<u>*FILE:</u> The form size specified in the device file is to be used.

form-length form-width: Enter the form length and width to be used for the FCT entry. The maximum valid form length is 255 and the maximum valid form width is 132.

LPI Parameter: Specifies the number of lines of print per inch to be used on the System/38 printer.

<u>*FILE:</u> The number of lines of print per inch specified in the device file is to be used.

4: The number of lines of print per inch is 4.

6: The number of lines of print per inch is 6.

8: The number of lines of print per inch is 8.

9: The number of lines of print per inch is 9.

CPI Parameter: Specifies the number of characters per inch to be used on the System/38 printer.

<u>*FILE:</u> The number of characters per inch specified in the device file is to be used.

10: The number of characters per inch is 10.

15: The number of characters per inch is 15.

PRTIMG Parameter: Specifies the qualified print image name to be used on the System/38 printer.

<u>*FILE:</u> The print image specified in the device file is to be used.

print-image-name: Enter the qualified name of the print image to be used. If no library qualifier is given, *LIBL is used to find the print image.

COPIES Parameter: Specifies the number of copies to be printed. This parameter applies only for spooled files.

<u>*FILE:</u> The number of copies specified in the device file is to be used.

number-of-copies: Enter the number of copies to be printed.

PGM Parameter: Specifies the qualified name of a user-supplied program to be used for processing data received from the host system.

<u>*WTRE:</u> The associated session description writer entry is to be used.

**NONE:* No user-supplied program is to be used.

program-name: Enter the qualified name of the user-supplied program to be used. (If no library qualifier is given, *LIBL is used to find the user-supplied program.)

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF writer are to be recorded.

Note: Messages for RJEF writers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands. If inquiry messages are issued by RJEF, they are sent to the user message queue (if specified) where they must receive a response.

<u>*WTRE:</u> The message queue specified in the session description writer entry is to be used.

**NONE:* No user message queue exists on which the messages for the FCT entry are to be recorded.

message-queue-name: Enter the qualified name of the user message queue on which the messages for the RJEF writer job's messages are to be recorded. (If no library qualifier is given, *LIBL is used to find the message queue.)

Examples

```
ADDFCTE  FCT(FORMCTRL.USERLIB) +
      FORMTYPE(MEDICAL) +
      DEVTYPE(*PRT) +
      LCLFORM(BIOCHEM) +
      FILE(MEDICAL44.MEDLIB) +
      DTAFMT(*FCFC) +
      MSGQ(BROWN.MEDLIB)
```

This command adds a forms control entry named MEDICAL to an FCT (forms control table) called FORMCTRL in library USERLIB. The forms control table entry is to be used with print files from the host. The local forms used when the data is printed are called BIOCHEM. The data from the host is written to a printer device file MEDICAL44 in library MEDLIB. The data format is first character forms control (*FCFC). Messages produced by RJEF while referencing this forms control entry are written to the user message queue named BROWN in library MEDLIB.

```
ADDFCTE  FCT(FORMCTRL.USERLIB) +
      FORMTYPE(MEDICAL) +
      DEVTYPE(*PUN) +
      FILE(MEDHISTORY.MEDLIB) +
      MBR(*GEN) +
      FSN(100) +
      DTAFMT(*DATA) +
      MSGQ(BROWN.MEDLIB)
```

This command adds a forms control entry named MEDICAL to an FCT (forms control table) called FORMCTRL in library USERLIB. The forms control table entry is to be used with punch files from the host. The data is written to a data base file named MEDHISTORY in library MEDLIB. RJEF will generate a new member for each host file received referencing this entry. The file sequence number of 100 will be used to generate the first data base member name. The first member generated by RJEF is named BMEDICA100. Messages produced by RJEF while referencing this forms control entry are written to the user message queue named BROWN in library MEDLIB.

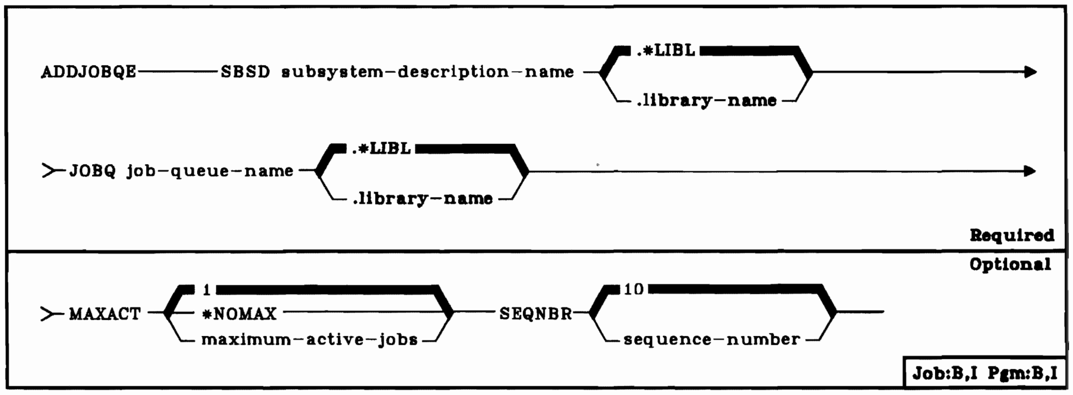
ADDJOBQE (Add Job Queue Entry) Command

The Add Job Queue Entry (ADDJOBQE) command adds a job queue entry to the specified subsystem description (the associated subsystem must be inactive at the time). A job queue entry identifies the job queue from which jobs are to be selected for execution within the subsystem. Jobs can be placed on a job queue by spooling readers or by using the following commands:

- Submit Job (SBMJOB)
- Submit Card Jobs (SBMCRDJOB)
- Submit Data Base Jobs (SBMDBJOB)
- Submit Diskette Jobs (SBMDKTJOB)
- Transfer Job (TFRJOB)

Within a subsystem, job queues with lower sequence numbers are processed first. For more information, refer to the SEQNBR parameter.

Restrictions: To use this command, you must have operational and object management rights for the specified subsystem description. The specified job queue must already exist in the system if the library qualifier is not given. A job queue is created by the CRTJOBQ command.



SBSD Parameter: Specifies the qualified name of the subsystem description to which the job queue entry is to be added. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

JOBQ Parameter: Specifies the unique qualified name of the job queue that is to be a source of batch jobs that are to be initiated by the subsystem. (If no library qualifier is given, *LIBL is used to find the job queue.) If the job queue does not exist when the entry is added, a library qualifier must be specified because the qualified job queue name is retained in the subsystem description.

MAXACT Parameter: Specifies the maximum number of jobs that can be concurrently active from this job queue. (For an expanded description of the MAXACT parameter, see Appendix A.)

1: Only one job from the job queue can be active at any time.

**NOMAX:* There is no maximum for the number of jobs that can be concurrently initiated through this job queue entry. However, the maximum activity level of the routing entries might prevent routing steps from being initiated. If *NOMAX is specified, all the jobs on the job queue will be initiated (within the limit specified by the MAXJOBS parameter in the subsystem description), even though the activity level of the storage pool used might prohibit them from executing concurrently.

maximum-active-jobs: Enter a value that specifies the maximum number of jobs that can be concurrently active from this job queue.

SEQNBR Parameter: Specifies a sequence number for this job queue, to be used by the subsystem to determine the order in which the job queues are to be processed.

10: A sequence number of 10 is to be assigned to this job queue.

sequence-number: Enter the sequence number to be assigned to this job queue. The sequence number must be unique within the subsystem description. Valid values are 1 through 9999.

The subsystem first selects jobs from the job queue with the lowest sequence number. Once all jobs on that queue have been processed or the number of jobs specified on the MAXACT parameter has been reached, the subsystem processes jobs on the queue with the next higher sequence number. This sequence continues until all job queue entries have been processed or until the subsystem has reached its limit for overall maximum jobs (as specified by the MAXJOBS parameter in the subsystem description). In some cases, this sequence is interrupted and the subsystem processes a queue with a lower sequence number. This occurs for this subsystem when:

- A held job or job queue is released
- A job is placed on or transferred to a queue
- A new queue is allocated
- A job terminates

Example

```
ADDJOBQE  SBSD(NIGHTSBS.QGPL)  JOBQ(NIGHT.QGPL)  +
     MAXACT(3)
```

This command adds a job queue entry for the NIGHT job queue (in the QGPL library) into the NIGHTSBS subsystem description, contained in the QGPL library. The entry specifies that a maximum of three batch jobs from the NIGHT job queue can be concurrently active within the subsystem. The default sequence number of 10 is assumed.

ADDLFM (Add Logical File Member) Command

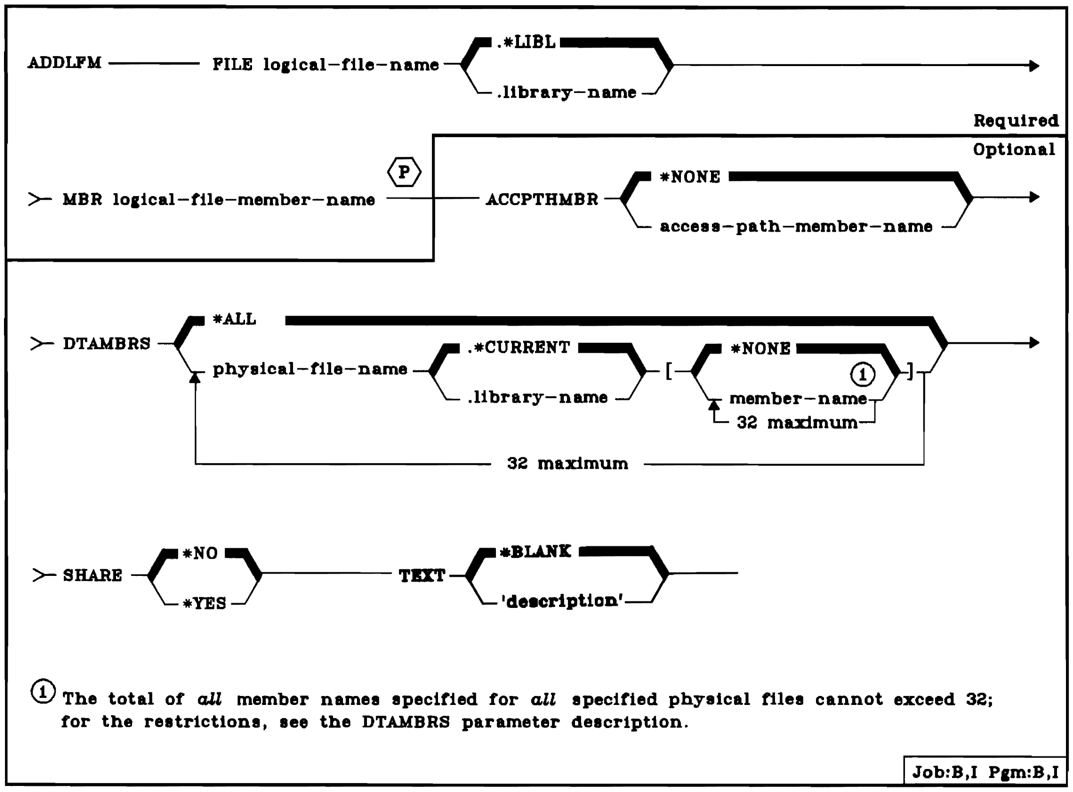
The Add Logical File Member (ADDLFM) command adds a named member to the specified logical file, which must have already been created. A member must be added in the logical file before the file can have access to data stored in any physical file member. (You can add the first member of a file by entering an ADDLFM command or by specifying a member name in the MBR parameter of the CRTLF command. To add other members to the file, use the ADDLFM command to specify each one.)

A logical file member can use the data from all, or a subset of, the physical files included in the scope of the logical file. Each member has its own set of data and can have its own access path (or share an access path) that provides an organization to that data.

The number of members that can be added for the logical file is limited to the maximum specified in the MAXMBRS parameter of the associated CRTLF command. Each member added has the same attributes as those defined in the logical file. However, each member can have its own access path or a shared access path, as specified in the DDS access path specifications. The access path determines the order in which the records in the based-on physical file(s) are processed.

Restrictions: To add a member to a logical file, you must have object management rights and operational rights for each of the physical file members (specified explicitly by the DTAMBRS parameter or implicitly by the PFILE keyword specified in DDS) upon which the logical file member is based. And, if the logical file member is to share the *keyed* sequence access path of another file member (specified by the ACCPTHMBR parameter), you must have operational rights for that member.

Note: Because this command adds a member to a file in a library, the library must not be locked (*SHRNUP or *EXCLRD in the Allocate Object command) for another job.



① The total of *all* member names specified for *all* specified physical files cannot exceed 32; for the restrictions, see the DTAMBRS parameter description.

FILE Parameter: Specifies the qualified name of the logical file in which this added member is to be stored. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name by which the logical file member being added is to be known. The member name must be unique within the file to which it is being added.

ACCPTHMBR Parameter: Specifies whether the added member is to share an access path with another file member, and, if so, specifies the name of that member. For information about access path sharing, refer to the description of the ACCPTH keyword in the *CPF Reference Manual–DDS*. The ACCPTH keyword can be specified in the logical file source description.

Note: If the logical file specified in FILE *is* sharing an access path, this parameter *must* specify a member name (identifying the member whose access path is to be shared with this added member). If the file is *not* sharing an access path, this parameter *cannot* specify a member name.

*NONE: This member is not to share the access path of another file member.

access-path-member-name: Enter the name of the member of the file that contains the access path to be shared with this member. The name of the file is specified in the logical file source description.

DTAMBRS Parameter: Specifies the names of the physical files and members that contain the data to be associated with the logical file member being added by this command. The scope of the logical file member can contain all of the physical files and members that the logical file itself contains, specified by DTAMBRS(*ALL); or the member can contain a subset of the total files and members, specified by DTAMBRS(qualified-file-name(s) [member-name(s)]).

Note: For additional information about coding this parameter and displaying access path information, refer to the *Additional Considerations* section at the end of the CRTLF command description.

<u>*ALL</u>: If no access path is shared, the scope of the logical file member being added is to be the same as that for the entire logical file. That is, the data to be associated with the member is in all the physical files and members (that exist at the time this ADDLFM command is entered) used by the logical file. The physical file names are specified by the PFILE keyword in the DDS source file named in the SRCFILE and SRCMBR parameters in the CRTLF command.

If *ALL is specified (or is the default) and the logical file is to share an access path with an existing physical or logical file, the data for the logical file member is the same as the data associated with the member specified by the ACCPTHMBR parameter; that is, the same based-on physical file(s) and member(s) are used.

qualified-physical-file-name [member-name]: Enter the names of the physical files and their members that contain the data to be accessed by the logical file member being added. Each entry for a physical file in the PFILE keyword in DDS should have a corresponding entry in the DTAMBRS parameter. Also, each physical file specified in the DTAMBRS parameter must correspond to one of the physical files specified by the PFILE parameter when the logical file was created. If no member name is specified for a physical file that is specified, <u>*NONE</u> is assumed and the logical file scope list or the based-on member's scope list is bypassed. (Refer to *Additional Considerations* in the CRTLF command for more details.)

A maximum of 32 qualified physical file names and physical file member names can be specified. Also, the total of *all* member names cannot exceed 32; that is, all of the member names specified for all of the files specified cannot be greater than 32. For example, one file can specify 32 members, two files can each have 16 members, or 32 files can each have one member specified.

When the file is created, the DDS PFILE keyword is used to specify physical file names and, optionally, the library qualifiers of the physical files being associated with the logical file. If a library qualifier is not specified, *LIBL is used to find the physical file when the logical file is created. (The physical file and the library in which it is stored are saved in the description of the logical file when the logical file is created.) When members are added to the file, each physical file name specified in the DTAMBRS parameter can be optionally qualified by the name of the library; however, the library name must be specified only if the logical file is based on more than one physical file of the same name, as defined in the PFILE keyword. If a library name is not specified for a physical file, the current library name (*CURRENT) for the specified file is determined from the qualified file name saved in the description of the logical file (not the current *LIBL library list).

The following examples show the syntax for specifying single and multiple members for single and multiple physical files. In the examples, the abbreviation PF represents a physical file name, LIB represents a library qualifier, and M represents a member name. Physical file names need only be qualified if the PFILE keyword in the DDS specifies multiple physical files of the same name.

```
Single physical file and member:
   DTAMBRS((PFA M1)) or DTAMBRS((PFA M1))
Single file with multiple members:
   DTAMBRS(PFA (M1 M2 M3))
Multiple files with single members and no members:
   DTAMBRS((PFA M1) (PFB M4) (PFE.NONE))
Multiple files with multiple members:
   DTAMBRS((PFA (M1 M3 M4)) (PFB (M1 M2 M4)))
Multiple files with the same name in different libraries:
   DTAMBRS((PFA.LIBX M1) (PFA.LIBY (M1 M2)))
Multiple files with the same name in the same library:
   DTAMBRS((PFA.LIBX M1) (PFA.LIBX M1))
```

As shown in the preceding example, each physical file specified in the PFILE keyword in the DDS should have a corresponding entry in the DTAMBRS parameter, even though it may mean specifying the same qualified physical file and member many times.

When more than one physical file member is specified for a physical file, the member names are specified in the order in which records are retrieved when duplicate composite key values occur across those members.

SHARE Parameter: Specifies whether or not an ODP (open data path) to the logical file member is to be shared with other programs in the same job. When an ODP is shared, the programs accessing the file share such things as the position being accessed in the file, the file status, and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

NO: An ODP created by the program when the file member is opened is not to be shared with other programs in the job. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* The same ODP is to be shared with each program in the job that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Lets the user enter text that briefly describes the logical file member. (For an expanded description of the TEXT parameter, see Appendix A.)

BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
ADDLFM  FILE(STOCKTXS.INVENLIB)  MBR(JANUARY) +
        DTAMBRS((INVENTXS JANUARY))  TEXT('JANUARY +
        STOCK ACTIVITY BY LOCATION')
```

This command adds a member named JANUARY to the logical file named STOCKTXS, which is stored in the INVENLIB library. The logical file has access to the data stored in the JANUARY member of the INVENTXS physical file.

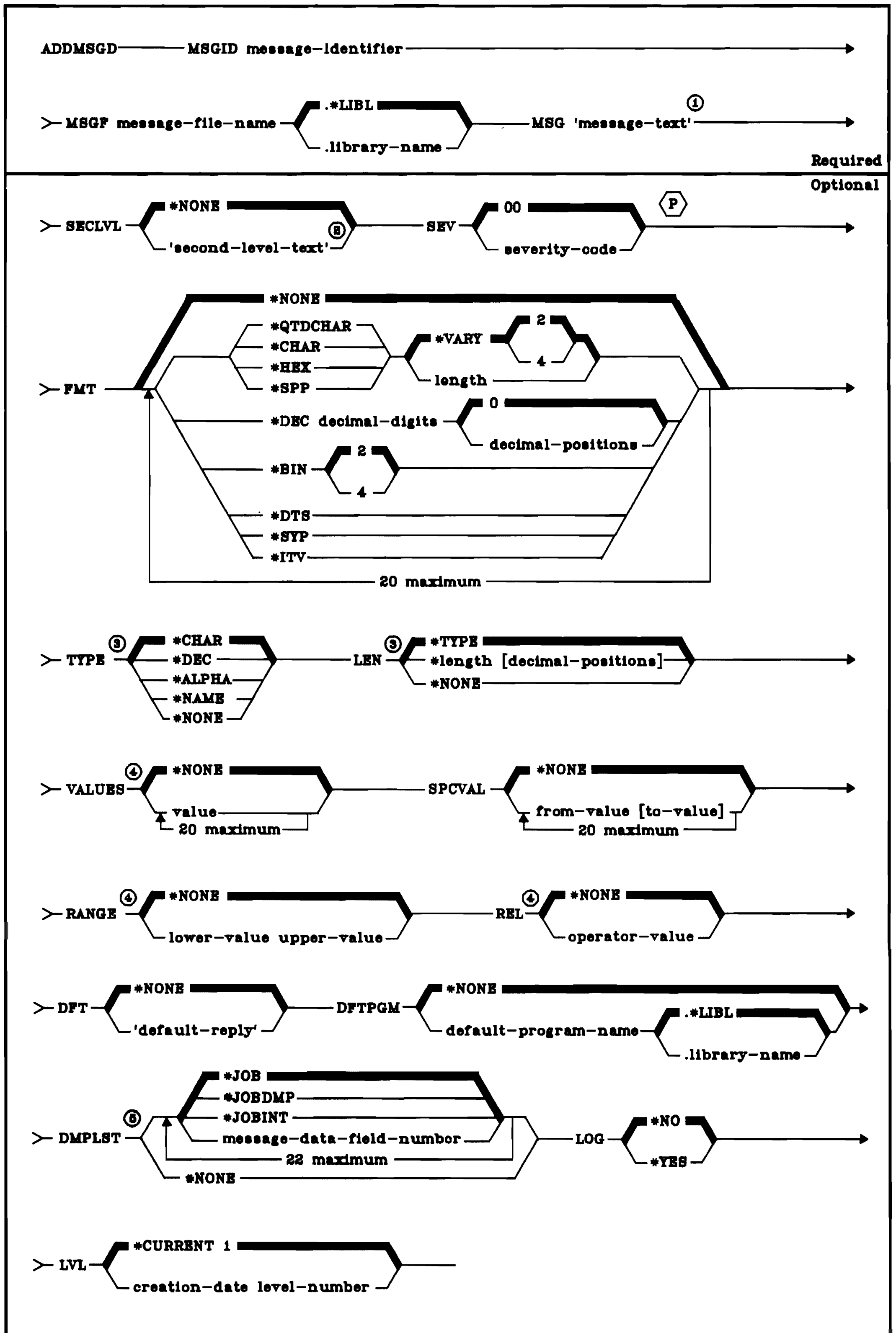
ADDMSGD (Add Message Description) Command

The Add Message Description (ADDMSGD) command describes a message and stores it in a message file for later use. The message description remains in the message file until the file is deleted or until the RMVMSGD command is used to remove it from the file. To change any of the attributes of the message description, such as its message text or severity code, you must execute the Change Message Description (CHGMSGD) command.

Substitution variables can be embedded in both the first-level and second-level text that can be replaced later by message data fields specified in the RTVMSG and SNDPGMMSG commands.

Note: The *type* of message being defined is *not* specified in the ADDMSGD command. The type is specified in the command that actually sends the message, in either the SNDMSG or SNDPGMMSG command.

Restriction: To add a message description to a message file, you must have operational rights for the message file and the library in which the file is stored.



① No more than 132 characters can be specified.
② No more than 1435 characters can be specified.
③ If either TYPE or LEN is specified as *NONE, the other of the two parameters must also be specified as *NONE.
④ VALUES, RANGE, and REL are mutually exclusive; only one of them can be specified.
⑤ If any of the parameter values are specified for DMPLST, *JOB is assumed to be part of the value(s).

Job:B,I Pgm:B,I

MSGID Parameter: Specifies the message identifier under which the message is stored in the message file. Every message must have an identifier, and every identifier in the message file must be unique.

The message identifier must be 7 characters long and in the following format:

pppnnnn

The first 3 characters must be a code consisting of an alphabetic character followed by two alphameric (alphabetic or decimal) characters, and the last 4 characters must be a decimal. The following codes are used to identify the messages in the IBM-supplied message files:

CBE	COBOL execution time
CBL	COBOL compiler
CBX	COBOL titles and texts
CSC	COBOL syntax checker
CPF	Control Program Facility (CPF)
CPI	CPF informational messages
CPX	CPF titles and texts
EDT	Edit source (SEU II)
EDX	Edit source titles and texts
FMT	Reformat Utility
FMX	Reformat Utility titles and texts
IDU	Interactive Data Base Utilities (IDU)
IDX	IDU titles and text
KBD	Keyboard
MCH	System/38 machine instruction interface
QRG	RPG language compiler
RPG	RPG execution time
RPT	RPG auto report
RSC	RPG syntax checker
RTX	RPG auto report titles and texts
RXT	RPG relational diagnostic texts
SDA	Screen Design Aid (SDA)
SDX	Screen Design Aid titles and texts

The same format must be used for user-defined messages; the 3-character code must begin with a U to distinguish user-defined messages from IBM-supplied messages. For example, the message identifier of a message in a payroll application message file could be UPY0027.

MSGF Parameter: Specifies the qualified name of the message file in which the message is to be stored. (If no library qualifier is given, *LIBL is used to find the file.) The IBM-supplied message files (QCPFMSG and QRPGMSG, for example) cannot be specified unless the user entering the command is explicitly authorized to update those files. (When the system is installed, only the system security officer has that authority.) Any message file overrides in effect for the job are ignored by this command; the file specified here is the one in which the message is stored.

MSG Parameter: Specifies the first level of message text of the message being defined. This text is the message that is initially displayed or printed, or sent to a program or log. A maximum of 132 characters (enclosed in apostrophes) can be specified, but the limitations of the display stations (their screen size) should be considered. The entire message must be enclosed in apostrophes if any blanks are to be included in the message. To code an apostrophe for use in the message, enter a double apostrophe.

One or more substitution variables can be embedded in the message text string to indicate positional replacement fields that allow variable data to be substituted in the message by the program before the message is sent. The variables must be specified in the form &n, where n is a one-digit number identifying the data field to be substituted. Each variable can be immediately followed by any nonnumeric character (such as &2M or &9?), but *not* by another digit (such as &99). (The variables in the text do *not* have to be in ascending sequence by these numbers. Also, blanks do not have to precede or follow each variable. The variables can be enclosed in apostrophes if only the variables themselves make up the message. For example, to show a two-part decimal value, the message '&1. &2' can be specified.) The data fields are described positionally in the FMT parameter and are specified positionally in the MSGDTA parameter of the SNDPGMMSG command. Refer to the *CPF Programmer's Guide* for details on substituting data fields in message text.

SECLVL Parameter: Specifies the second-level text, if any, that is to be displayed to a work station user to further explain the message specified in the MSG parameter. The user presses the Help key to request the second-level text. Second-level text can also be written to the job log if *SECLVL is specified on the LOG parameter of the job commands.

<u>*NONE:</u> There is to be no second-level text for this message description.

'second-level-text': Enter the text to be displayed as second-level text when it is requested by the user. No more than 1435 characters (enclosed in apostrophes) can be specified, but display (up to a maximum of nine) limitations must be considered. One or more substitution variables can be embedded in the second-level text, as described in the MSG parameter.

SEV Parameter: Specifies the severity code of the message being defined. The severity code indicates the severity level of the condition that causes the message to be sent.

00: The severity code assigned to this message is 00. The message is an information only message.

severity-code: Enter a value, 00 through 99, that is to be the severity level associated with this message. The assigned code for the message should correspond in importance to the IBM-predefined severity codes. (These codes and their meanings are given in the chart under the SEV parameter, in Appendix A.) Any two-digit value can be entered, even if no severity code has been defined for it (either predefined or user-defined).

FMT Parameter: Specifies the formats of from one to 20 message data fields. Each field is described in this parameter by a list of attributes. The first nine message data fields can be used as substitution values in the first-level and second-level text messages defined in this message description. All 20 of the fields can be specified in the DMPLST parameter of this command. When specified in the MSGDTA parameter of the SNDPGMMSG command, the data fields must be concatenated in one character string and must match the format and sequence specified here. The length of the entire character string of concatenated message data fields cannot exceed 132 characters.

*NONE: No format is being described for message fields. If *NONE is specified, or if this parameter is omitted, no references can be made to message data fields in the MSG, SECLVL, or DMPLST parameters.

type [length [decimal-positions]]: The format of each message data field (up to a maximum of 20 fields) to be substituted in the message in this message description is defined by a list of attributes. These attributes specify the type of data in the field, the total length of the field, and, optionally, the number of decimal digits to the right of the decimal point. Certain data types do not require a length field. Boundary alignment requirements must be considered (for example, pointers are always aligned on 16-byte boundaries). While 20 fields may be defined, &1 through &9 can appear in the message text; the others can appear only in the dump list.

Type of Message Data: The first value, type, specifies the type of data the substitution field contains and how the data is to be formatted when substituted in the message text. The contents of the second and third values vary depending on the type specified. One of the following types can be specified for each field described by this parameter:

*QTDCHAR: A character string to be formatted (by CPF) with enclosing apostrophes ('Monday, the 1st').

*CHAR: A character string to be formatted without enclosing apostrophes. It is an alphameric string that can be used, for example, to specify a name (BOB). Trailing blanks are truncated.

*HEX: A string of bytes to be formatted as a hexadecimal value (X'C0F4').

*DEC: A packed decimal number that is formatted in the message as a signed decimal value with a decimal point. Values for length (required) and decimal positions (optional) are specified for this type (*DEC) to indicate the number of decimal digits and the number of digits to the right of the decimal point. Zeros to the left of the first significant digit are suppressed, and leading blanks are truncated (removed). If a decimal position other than zero is specified, a decimal point is shown in the result even if the decimal precision in the result is zeros; examples are 128.00 and 128.01 if FMT(*DEC 5 2) is specified. If the number of decimal positions is not specified, zero is assumed. The following gives two examples:

- If FMT(*DEC 2) is specified for a substitution field and the message data is a packed decimal value of X'058C', the message text will contain a positive value of 58 with no decimal point indicated.
- If FMT(*DEC 4 2) is specified and the packed value is specified as X'05810C' (3 bytes long), then the text will contain the formatted decimal value of 58.10.

*BIN: A binary value that is either 2 or 4 bytes long (B'0000 0000 0011 1010') and is formatted in the message as a signed decimal value (58).

The following formats are valid only in IBM-provided message descriptions and should not be used for other messages.

*DTS: An 8-byte field that contains a system date time stamp. The date time stamp contains the date followed by one blank separator and the time. The date is formatted in the output message in the format specified by the system values QDATFMT and QDATSEP. The time is formatted as hh:mm:ss.

*SPP: A 16-byte space pointer to data in a space object. When referenced in the DMPLST parameter, the data in the space object (from the offset indicated by the pointer) for the length specified, is to be dumped. *SPP is not valid as a replacement field in message text.

*SYP: A 16-byte system pointer to a system object. When referenced in message text, the simple name of the system object is formatted as described in the name type, *CHAR. When referenced by the DMPLST parameter, the object itself is to be dumped.

*ITV: An 8-byte binary field that contains the time interval (in seconds) for wait time-out conditions. The time interval is formatted in the message as a zero-suppressed zoned decimal value (15 0) representing the number of seconds to wait.

Length of Message Data: Following the type specification, a second value (length) can be specified to indicate the number of characters or digits that are passed in the message data. How the second value is used depends upon the type specified in the first value.

1. If a length is not specified for *QTDCHAR, *CHAR, *HEX, or *SPP, then <u>*VARY</u> is assumed for the length. If *VARY is specified or assumed, the message data field passed by the SNDPGMMSG command must be preceded by a 2-byte or 4-byte binary field that indicates the actual number of bytes of data being passed. However, when *SPP is specified, the length field is contained in the first bytes pointed to by the space pointer. Therefore, the 2- or 4-byte field must precede the data pointed to by the space pointer, and *not* precede the space pointer that is passed as part of the message data.

2. If the type *DEC is specified, the total number of decimal digits (including the fraction) *must* be specified as the second value; the number of digits in the fraction can be specified (optional) as the third value.

3. If the type *BIN is specified, the message data field can be only 2 or 4 bytes long; the default is <u>2</u> bytes.

Length Field Size/Decimal Positions: The third value is used in one of two ways, depending upon the type specified in the first value. (1) If *QTDCHAR, *CHAR, *HEX, or *SPP is specified, and if *VARY is specified or assumed for the second value, the third value is used with *VARY to indicate the size of the length field actually passed. The third value can be either a <u>2</u> or a 4, which is the number of bytes to be used to specify the length (in binary) of the passed value. (2) If *DEC is specified, the third value indicates the number of decimal positions in the decimal value. If not specified for a decimal substitution value, the default is <u>0</u> decimal positions.

Note: If an object has been damaged or deleted, the substitution variable will not be replaced by the object name when it is displayed; instead, the variable will appear as &n, where n = number. Also, if the length of the message data that is passed to the substitution variable is shorter than the length specified for FMT, the substitution value becomes a null field.

Reply Validity Specification Parameters

If the message is to be sent as an inquiry message (specified by *INQ in one of the send message commands) or as a notify message (specified by *NOTIFY in the SNDPGMMSG command only) and a reply is expected, seven parameters can be used to specify some requirements that relate to the reply received. The seven validity checking parameters are: TYPE, LEN, VALUES, SPCVAL, RANGE, REL, and DFT.

These parameters are not necessary for a message to allow a reply, but they can be used to define valid replies that can be made to the message. Also note that the VALUES, RANGE, and REL are mutually exclusive—only one of them can be specified in this command.

TYPE Parameter: Specifies, only if the message is sent as an inquiry or notify message, the type of reply that is valid to respond to this message.

CHAR: Any character string. If it is a quoted character string, the apostrophes are passed as part of the character string.

**NONE:* No reply type is specified. No reply validity checking is to be performed if this message is sent as an inquiry or notify message. LEN(*NONE) must also be specified.

**DEC:* Only a decimal number is a valid reply.

**ALPHA:* Only an alphabetic (A through Z, $, #, and @) character string is valid. Blanks are not allowed.

**NAME:* Only a simple name is a valid reply. The name does not have to be a CPF object name, but it must begin with an alphabetic character; the rest must be alphameric.

LEN Parameter: Specifies, only if the message is sent as an inquiry or notify message, the length that cannot be exceeded by a reply to this message. The values specified under *TYPE apply *only* if one or more of the other validity checking parameters are specified. If, however, *none* of the validity checking parameters are specified, the reply (of type *CHAR) can contain as many as 132 characters.

TYPE: The maximum length is determined by the type of reply specified in the TYPE parameter. The maximum length for each type of reply is:

- 132 characters for types *CHAR and *ALPHA. If any further validity checking is to be perfomed (VALUES, RANGE, REL, SPCVAL, or DFT are specified), the maximum length allowed for *CHAR and *ALPHA is 32 characters.
- 15 digits for *DEC, of which a maximum of 9 digits can be to the right of the decimal point.
- 10 alphameric characters for *NAME.

**NONE:* No reply type is specified. No reply validity checking is to be performed if this message is sent as an inquiry or notify message. TYPE(*NONE) must also be specified.

length [decimal-positions]: Enter the maximum length to be allowed for the message reply. The length specified here cannot exceed the maximums shown above. If the reply type is a decimal value, the number of decimal positions can be optionally specified; if it is not specified, zero decimal positions are assumed.

VALUES Parameter: Specifies, only if the message is sent as an inquiry or notify message, a list of values of which one can be received as a valid reply to the message. No more than 20 values can be specified in the list. Each value in the list must meet the requirements specified for message replies by the TYPE and LEN parameters.

If VALUES is specified, the RANGE and REL parameters cannot be specified. A reply, to be valid, must match one of the values in this list.

For the reply value to match the compare value, both must be of the same keyboard shift. For example, if your program requires a reply containing uppercase characters, one of the following methods ensures a response in uppercase characters:

- Requiring a response in uppercase characters.
- Entering the compare values for the VALUES parameter in lowercase, but using the SPCVAL parameter to convert the characters to uppercase.
- Using the TYPE(*NAME) keyboard value to convert the characters to uppercase. To use this method, all reply characters must be alphabetic (A-Z).

*NONE: No list of reply values is specified. The reply can have any value that is consistent with the other validity specification parameters.

value: Enter one or more values, up to a maximum of 20, that are to be compared with a reply value that is sent in response to the message defined in this message description; the reply value must match one of these values to be a valid reply to this message. The maximum length of each value is 32 characters.

SPCVAL Parameter: Specifies, only if the message is sent as an inquiry or notify message, a list of up to 20 sets of special values of which one set (if the from-value is matched by the sent reply) is used as the reply. These values are special in that they may not meet all the validity checking specifications given in the other reply-oriented parameters. The reply sent is compared to the from-value in each set; if a match is found, and a to-value was specified in that set, the to-value is sent as the reply. If no to-value was specified, the from-value is sent as the reply. The to-value must meet the requirements specified in the TYPE and LEN parameters. If the reply sent does not match any from-value, then the reply is validity checked by the specifications in the other reply-oriented parameters.

*NONE: No special values are specified for the replies to this message.

from-value [to-value]: Enter one or more sets of values, up to a maximum of 20 sets, that are used to determine the reply sent to the sender of the message. Each set must have a from-value, which the reply is compared with, and an optional to-value to be sent as the reply if its from-value matches the reply.

RANGE Parameter: Specifies, only if the message is sent as an inquiry or notify message, the upper and lower value limits for valid replies to this message. These values must meet the requirements specified for replies by the TYPE and LEN parameters, and both values must be of the same type. If both values are not of the same length, the shorter value is padded on the right with blanks. For the types *CHAR and *ALPHA replies, the reply is padded to the right with blanks, or truncated on the right, to the length of the specified values, before the value range is validity checked. If RANGE is specified, the VALUES and REL parameters cannot be specified.

*NONE: No range values are specified for the replies to this message.

REL Parameter: Specifies, only if the message is sent as an inquiry or notify message, the relation that must be met for a reply to be valid. The value specified must meet the requirements specified for replies by the TYPE and LEN parameters. For replies of the types *CHAR and *ALPHA, the reply is padded to the right with blanks, or truncated on the right, to match the length of the value specified, before the system performs the relational test on the reply value sent.

*NONE: No range values are specified for the replies to this message.

operator-value: Enter one of the relational operators and the value against which the message reply is to be validity checked. If the reply is valid in the relational test, it is sent to the message sender. If REL is specified, the VALUES and RANGE parameters cannot be specified. The relational operators that can be entered are:

*LT	Less than
*LE	Less than or equal to
*GT	Greater than
*GE	Greater than or equal to
*EQ	Equal to
*NL	Not less than
*NG	Not greater than
*NE	Not equal to

DFT Parameter: Specifies, only if the message is sent as an inquiry or notify message, the default reply (enclosed in apostrophes, if it contains special characters) that is to be used when the receiver of the message has indicated that all messages to him are to use default replies, or when a message is deleted from a message queue and no reply was specified. The default reply can also be used to answer unmonitored notify messages. The default reply must meet the requirements specified for replies by the validity specification parameters, TYPE and LEN.

<u>*NONE:</u> No default reply is specified for the replies to this message.

DFTPGM Parameter: Specifies the name of the default program (if any) to be called to take default action when this message is sent as an escape message to a program that is not monitoring for it. This parameter is ignored if the message is *not* sent as an escape message. If it is sent as an escape message, the following parameters are passed to the program:

- Program message queue name (10 characters). The name of the program message queue to which the message was sent. (This is the same name as that of the program.)

- Message reference key (4 characters). The message reference key of the escape message on the program message queue.

<u>*NONE:</u> No default program is specified for this message.

qualified-default-program-name: Enter the qualified name of the default program to be called when an escape message is sent. (If no library qualifier is given, *LIBL is used to find the default program.)

DMPLST Parameter: Specifies the data to be dumped when this message is sent as an escape message to a program that is not monitoring for it. This parameter can specify that data related to the job be dumped, that data from message data fields be dumped, or that a combination of these be dumped. When data from message data fields is to be dumped, this parameter specifies one or more numbers that positionally identify the data fields to be dumped.

The system objects indicated by system pointers are to be completely dumped. The data in a space object, indicated by a space pointer, is to be dumped starting from the offset indicated by the space pointer for the length indicated in the field description. The standard job dump can also be requested. Dumps are taken as part of system default actions when escape messages are not monitored by the program to which they were sent.

JOB: This value is the equivalent of specifying DSPJOB JOB(*) OUTPUT(*LIST); refer to *DSPJOB (Display Job) Command* for more information.

**JOBDMP:* The data areas of the job are to be dumped as specified by the DMPJOB command. *JOB can be specified by itself, along with *JOBINT, or along with a list of message data field numbers.

**JOBINT:* The internal machine data structures related to the machine process in which the job is executing are to be dumped to the machine error log as specified by the DMPJOBINT command. *JOBINT can be specified by itself, along with *JOBDMP, *JOB, or along with a list of message data field numbers.

message-data-field-number: Enter the numbers of the message data fields that identify the data to be dumped when this escape message is sent but not monitored. As many as 20 data field numbers can be specified in the list; additionally, the list can contain the values *JOB and *JOBINT.

**NONE:* There is no dump list for this message. No dump is to be taken.

Note: If any of these values are specified for DMPLST, *JOB is assumed to be part of the values. For example, DMPLST(1 2 *JOBDMP) results in the equivalent of DMPLST(*JOB 1 2 *JOBDMP).

LOG Parameter: Specifies, when it is sent as an escape message that is not monitored, whether the message is to be logged in the system service log.

NO: The unmonitored escape message is not to be logged in the system service log when it is used.

**YES:* Every occurrence of the unmonitored escape message's use is to be logged in the system service log.

LVL Parameter: Specifies the level identifier of the message description being defined. The level identifier is made up of the date on which the message is defined and a two-digit number that makes the identifier unique.

*CURRENT _1:_ The current date and a 1 are to be used as the first and second parts of the message description level identifier.

creation-date level-number: Enter the date on which the message is being defined, and enter a two-digit value (1 through 99) that makes the level identifier of the message description unique. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP.

Examples

```
ADDMSGD  MSGID(UIN0115) MSGF(INV) +
      MSG('Enter the name of your department') +
      SECLVL('A department''s name is a +
      3-character code such as X12') +
      TYPE(*CHAR) LEN(3) DFT('ZZZ')
```

This command defines a message and stores it in a file named INV under the identifier UIN0115. The message supplies second-level text, and the reply requires validity checking so that a valid reply can only be a 3-character identifier. A default reply of ZZZ is also provided.

```
ADDMSGD  MSGID(UPY0047) MSGF(TIMECARD.PAYLIB) +
      MSG('For the week of &1, &2 time +
      cards were processed.  Do you have more?') +
      FMT((*CHAR 8) (*CHAR 3)) +
      TYPE(*ALPHA) LEN(1) VALUES(N Y) +
      SPCVAL((YES Y)(NO N)) DFT(N)
```

This command defines a message description that is stored in the TIMECARD message file in the PAYLIB library. The program that processes the time cards can send a message (as an inquiry type message) telling how many time cards (in &2) have been processed for the week (specified in &1). To send this message to a user (via a message queue), the program must use the SNDPGMMSG command; the command specifies (in this example):

- The message identifier of this message (UPY0047).
- The file (TIMECARD) that contains this message.
- The time card date in 8 characters (such as 9/15/78). This must be the first value in the MSGDTA parameter.
- The number of time cards in no more than three digits (such as 125).

If a reply of YES is sent, it is accepted as a Y (SPCVAL parameter). If NO is sent, it is accepted as an N. If neither YES nor NO is sent, the reply is validity checked according to the TYPE, LEN, and VALUES parameters. If the user chooses, no reply can be sent and the default reply (N) is assumed.

```
ADDMSGD  MSGID(UPY1234) MSGF(TIMECARD.PAYLIB) +
      MSG('Tax for employee &1 exceeds +
      gross salary.') SEV(75) +
      FMT((*CHAR 6)(*DEC 9 2)(*CHAR 8)) +
      DFTPGM(BADTAX.PAYLIB) +
      DMPLST(1 2 3 *JOB)
```

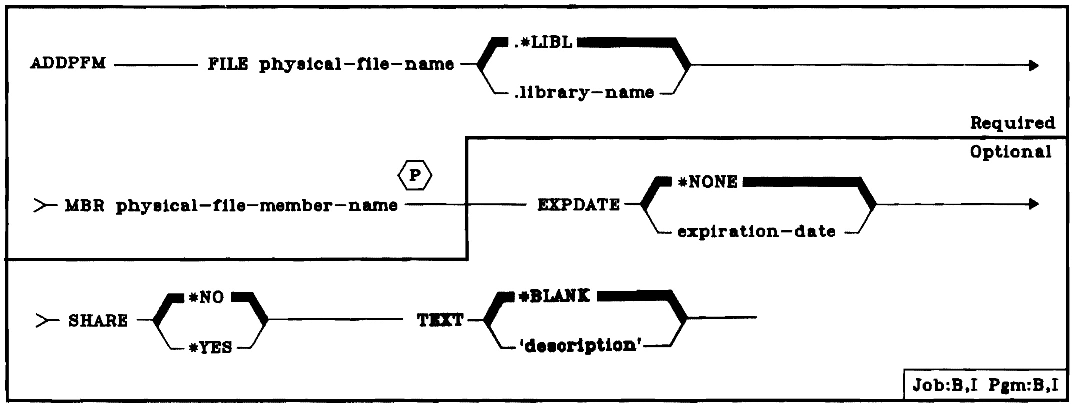
This command defines a message to be sent as an escape message. The sender of the message passes three data values, the first of which (employee serial number) is used as replacement text in the message. If the program to which this message is sent does not monitor for message UPY1234, default system action is to be taken. This includes dumping the three data values passed and the job structure. After the dump is taken, program BADTAX is to be invoked.

ADDPFM (Add Physical File Member) Command

The Add Physical File Member (ADDPFM) command adds a named member in the specified physical file, which must have already been created. A member must be added in the physical file before the file can have data stored in it. (You can add the *first* member of a file by entering an ADDPFM command or by specifying a member name in the MBR parameter of the CRTPF command. To add other members to the file, use the ADDPFM command to specify each one). Each member has its own set of data and its own access path that provides an organization for that data.

The number of members that can be added for the physical file is limited to the maximum specified in the MAXMBRS parameter of the associated CRTPF command. Each member added has the same attributes as those defined in the physical file. However, each member has its own access path as specified in the DDS access path specifications. The access path determines the order in which the records within that member are processed.

Note: Because this command adds a member to a file in a library, the library must not be locked (*SHRNUP or *EXCLRD in the Allocate Object command) for another job.



FILE Parameter: Specifies the qualified name of the physical file in which this added member is to be stored. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name by which the physical file member being added is to be known. The member name must be unique within the file to which it is being added.

EXPDATE Parameter: Specifies the expiration date of the member. Any attempt to open a file that uses a member that has expired causes an error message to be sent to the user. (The RMVM command is used to remove the member.)

<u>*NONE:</u> The member has no expiration date.

expiration-date: Enter the date after which the member should not be used. The date must be specified in the format defined by the system values, QDATFMT and QDATSEP. The date must be enclosed in apostrophes if special characters are used in the format.

SHARE Parameter: Specifies whether or not an ODP (open data path) to the physical file member is to be shared with other programs in the same job. When an ODP is shared, the programs accessing the file share such things as the position being accessed in the file, the file status, and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

<u>*NO:</u> An ODP created by the program when the file member is opened is not to be shared with other programs in the job. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* The same ODP is to be shared with each program in the job that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Lets the user enter text that briefly describes the physical file member. (For an expanded description of the TEXT parameter, see Appendix A.)

<u>*BLANK:</u> No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

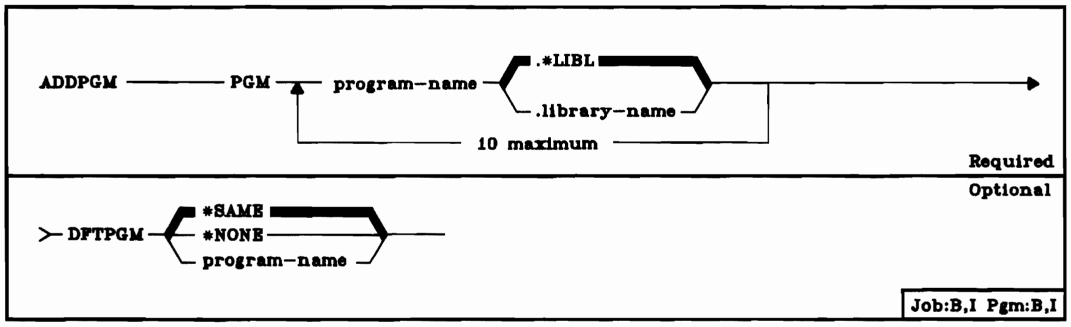
```
ADDPFM  FILE(INVENTX) MBR(MONDAYTX) +
     TEXT('Monday''s Inventory Transactions')
```

This command adds a member named MONDAYTX in the physical file named INVENTX. The library list (*LIBL) is used to find the file because the FILE value was not qualified by a library name. The size of the member and the storage allocation values assigned to this member were specified in the CRTPF command that created the physical file in which MONDAYTX will be stored. The text, *Monday's Inventory Transactions*, describes this member of the INVENTX file.

ADDPGM (Add Program) Command

The Add Program (ADDPGM) command adds one or more programs to the group of programs currently being debugged. The specified programs can have breakpoints and traces added to them for controlling and tracing their execution. The values of the programs' variables can also be displayed and changed.

Restrictions: No more than 10 programs can be in debug mode at the same time. Two or more programs with the same simple name cannot be debugged simultaneously. This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command*.



PGM Parameter: Specifies the qualified names of one or more programs to be debugged. (If no library qualifier is given for a program, *LIBL is used to find the program.) The number of programs specified here depends on how many programs are already in debug mode; 10 is the maximum at any time.

DFTPGM Parameter: Specifies the name of the program that is to be the default program during debug mode. The program specified here is used as the default program for any of the other debug commands that specify *DFTPGM on their PGM parameter. (That is, if a default program was previously specified, this parameter can change it.)

*SAME: The same program, if any, currently specified as the default program is to be used.

**NONE:* No program is to be specified as the default program; if a program was specified as a default program, it is no longer. Therefore, *DFTPGM cannot be specified on the PGM parameter of any other debug commands.

program-name: Enter the simple name of the program that is to be the default program during debug mode. The same name (in qualified form) must also be specified in the PGM parameter of this command or have been specified on the Enter Debug (ENTDBG) command or on a previous Add Program (ADDPGM) command.

Example

```
ADDPGM  PGM(MYPROG.QGPL)
```

This command adds the program MYPROG, located in the QGPL library, to the current debug mode. Breakpoints and traces can be put in MYPROG, and its variables can be displayed and changed by other debug commands. Because DFTPGM was not specified, the same default program is used.

ADDRJECMNE (Add RJE Communications Entry) Command

The Add RJE Communications Entry (ADDRJECMNE) command adds a new communications device file entry to an existing RJEF session description.

Two communications entries are required to start an RJEF session:

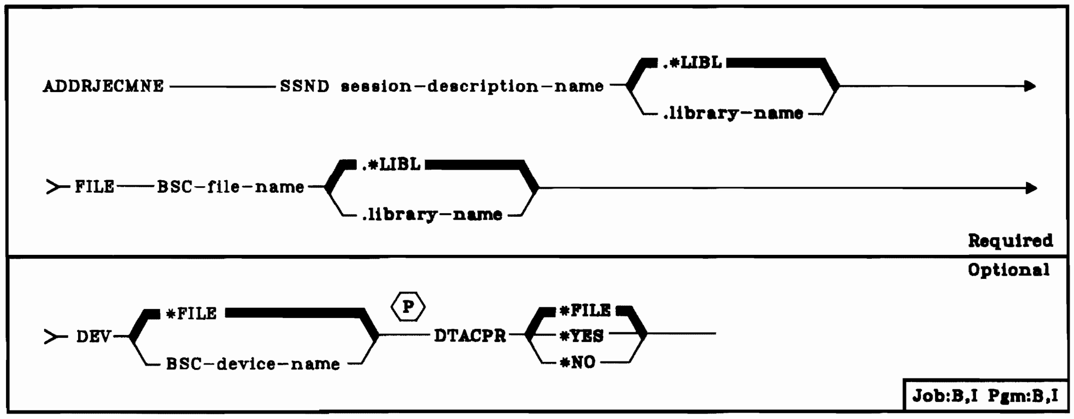
- One entry for an RJEF console input job
- One entry for an RJEF console output job

Additionally, one communications entry is required for each active RJEF reader or RJEF writer in the RJEF session.

Each communications entry must reference a unique BSC device file. All BSC device files must reference devices attached to the same BSC control unit.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Add RJE Communications Entry (ADDRJECMNE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSND Parameter: Specifies the qualified name of the session description to which the communications entry is to be added. (If no library qualifier is given, *LIBL is used to find the session description.)

FILE Parameter: Specifies the qualified name of the BSC device file to be added to the session description. (If no library qualifier is given, *LIBL is used to find the communications device file.)

DEV Parameter: Specifies the communications device to be used with the specified communications device file for sending and receiving data.

<u>*FILE:</u> The device name specified in the communications device file is to be used.

BSC-device-name: Enter the name of the BSC device to be used. This device name overrides the device that was specified when the communications device file was created.

DTACPR Parameter: Specifies whether data compression is to be performed for the communications file entry.

<u>*FILE:</u> Data compression is to be performed, based on the specification in the communications device file.

**YES:* Data compression is to be performed for the communications file entry.

**NO:* Data compression is not to be performed for the communications file entry.

Example

```
ADDRJECMNE  SSND(RJE.USERLIB) +
     FILE(DEVPRT1.USERLIB) +
     DTACPR(*YES)
```

This command adds a communication entry to the session description called RJE in library USERLIB. The BSC device file associated with this communication entry is named DEVPRT1 in library USERLIB. Data compression is to be performed for this BSC file.

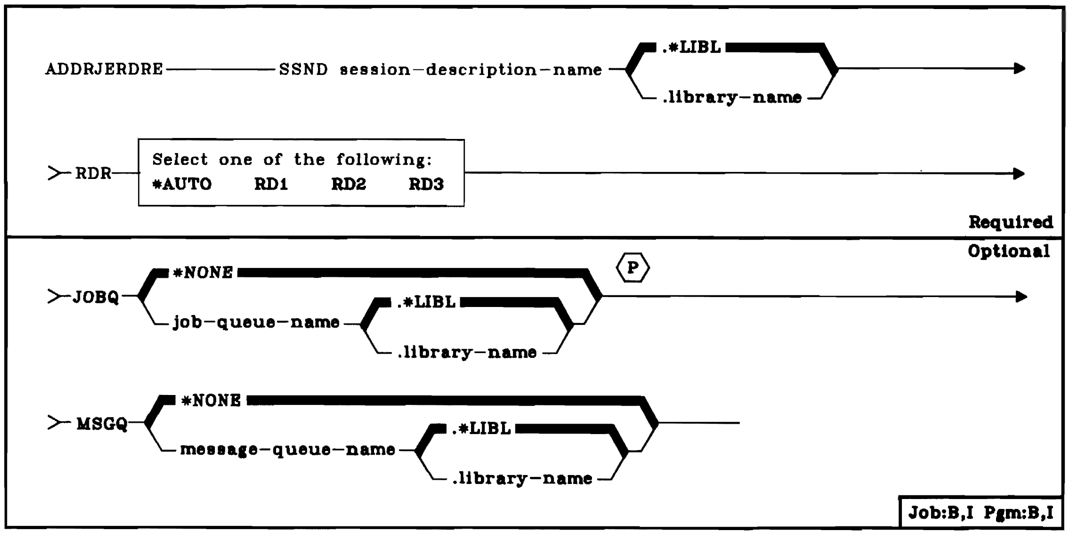
ADDRJERDRE (Add RJE Reader Entry) Command

The Add RJE Reader Entry (ADDRJERDRE) command adds a new RJEF reader entry to an existing RJEF session description.

Each RJEF reader entry (except *AUTO) requires a corresponding communications entry (refer to the Add RJE Communications Entry (ADDRJECMNE) command). A maximum of four RJEF reader entries can be added (including *AUTO).

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Add RJE Reader Entry (ADDRJERDRE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description to which the RJEF reader entry is to be added. (If no library qualifier is given, *LIBL is used to find the session description.)

RDR Parameter: Identifies the RJEF reader that is to be associated with this reader entry.

**AUTO:* Any RJEF reader input stream that is available at the time the Submit RJE Job (SBMRJEJOB) command executes is to be used.

RD1: RJEF Reader 1 input stream is to be used.

RD2: RJEF Reader 2 input stream is to be used.

RD3: RJEF Reader 3 input stream is to be used.

JOBQ Parameter: Specifies the job queue on which the reader jobs for this reader are to be placed.

__*NONE:__ No reader job queue is to be associated with this reader. RJEF reader input streams can be reserved for the interactive user issuing the SBMRJEJOB command and specifying OPTION(*IMMED). Therefore, the interactive user does not have to compete with the batch RJEF reader jobs that are started from the RJEF reader job queue.

job-queue-name: Enter the qualified name of the job queue on which reader jobs for this reader are to be placed for transmission to the host system. (If no library qualifier is given, *LIBL is used to find the job queue.)

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF reader are to be recorded.

Note: Messages for RJEF readers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands. If inquiry messages are issued by RJEF, they are sent to the user message queue (if specified) where they must receive a response.

__*NONE:__ No user message queue exists on which the messages for these RJEF reader jobs are to be recorded.

message-queue-name: Enter the qualified name of the user message queue on which this RJEF reader job's messages are to be recorded. (If no library qualifier is given, *LIBL is used to find the message queue.)

Example

```
ADDRJERDRE  SSND(RJE.USERLIB) +
      RDR(RD1) +
      JOBQ(READQ1.USERLIB) +
      MSGQ(BAKER.USERLIB)
```

This command adds an RJEF reader entry to the session description named RJE in library USERLIB. The reader added is RD1 (reader 1). RJEF jobs submitted to this reader will be submitted to RJEF reader job queue READQ1 in library USERLIB. Messages associated with jobs submitted to RD1 are to be written to the user message queue named BAKER in library USERLIB.

ADDRJEWTRE (Add RJE Writer Entry) Command

The Add RJE Writer Entry (ADDRJEWTRE) command adds a new RJEF writer entry to an existing RJEF session description.

Each RJEF writer entry requires a corresponding communications entry (refer to the Add RJE Communications Entry (ADDRJECMNE) command). A maximum of six RJEF writer entries can be added (three printers and three punches).

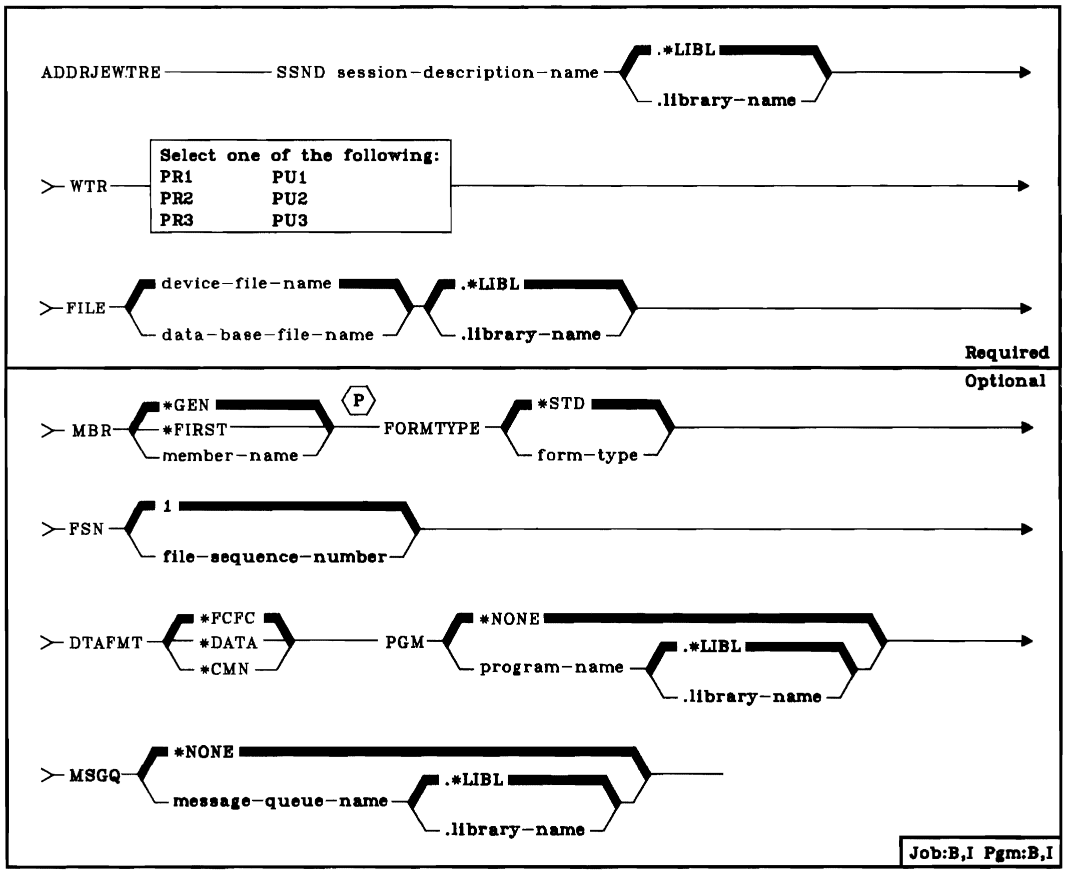
Except for the SSND parameter, all the parameters of this command are used to direct the output data only if any of the following conditions are true:

- There is no FCT associated with this session description.
- The forms mount message from the host system specifies a form type that does not exist as an entry in the FCT.
- *NONE is specified on the FILE parameter in the Add Forms Control Table Entry (ADDFCTE) command.
- A writer entry is used for each parameter in the FCT.
- A writer entry is used for each parameter on the Start RJE Writer (STRRJEWTR) command.

The parameter values specified for this RJE writer entry can be overridden by parameter values specified for the Start RJE Writer (STRRJEWTR) command.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Add RJE Writer Entry (ADDRJEWTRE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSND Parameter: Specifies the qualified name of the session description to which the RJEF writer entry is to be added. (If no library qualifier is given, *LIBL is used to find the session description.)

WTR Parameter: Identifies the RJEF writer that is to be associated with this writer entry.

PR1: RJEF Printer 1 output stream is to be used.

PR2: RJEF Printer 2 output stream is to be used.

PR3: RJEF Printer 3 output stream is to be used.

PU1: RJEF Punch 1 output stream is to be used.

PU2: RJEF Punch 2 output stream is to be used.

PU3: RJEF Punch 3 output stream is to be used.

FILE Parameter: Specifies the qualified name of the file that is to receive data from the host system.

device-file-name: Enter the qualified name of the program-described device file to receive data. (If no library qualifier is given, *LIBL is used to find the device file.)

data-base-file-name: Enter the qualified name of the System/38 physical data base file to be used. (If no library qualifier is given, *LIBL is used to find the data base file.)

MBR Parameter: Specifies the data base file member to which the output is to be directed (if a data base file was specified in the FILE parameter of this command). If a device file was specified in the FILE parameter of this command, this parameter is ignored.

<u>*GEN:</u> RJEF creates a member name as follows:

Afffffccc or Bffffffccc

Where:

A	=	file member names beginning with the character A contain print data.
B	=	file member names beginning with the character B contain punch data.
ffffff	=	first six characters of the forms name specified in the FCT or received from the host system. **Note:** Only characters that are valid in a System/38 name are valid in the forms type used to generate data base file member names.
ccc	=	three-digit sequence value controlled by the RJEF session to maintain member uniqueness (refer also to the FSN parameter description of this command).

If a member with this name already exists in the data base file, the three-digit sequence value is incremented by one and another attempt is made to create a member. Incrementing of the sequence value continues until a unique name is generated and a member is created or until all 1000 possibilities have been exhausted without creating a member. If no member is created, the RJEF operator receives a message indicating the failure and a request to retry or cancel this file.

**FIRST:* The output is to be directed to the first member of the data base file (if a data base file is specified in the FILE parameter of this command).

member-name: Enter the name of the data base file member to which output is to be directed (if a data base file is specified in the FILE parameter of this command).

FORMTYPE Parameter: Specifies the initial form type to be used if no forms mount message is received from the host system.

*STD: The initial form type to be used is *STD.

form-type: Enter the initial form type. Valid values can be one through eight alphameric characters in length.

FSN Parameter: Specifies the initial three-digit file sequence number to be used when creating data base file member names (when the MBR parameter default of *GEN is taken).

1: The initial three-digit file sequence number to be used is 001.

file-sequence-number: Enter the initial three-digit file sequence number to be used. Leading zeros are not required for sequence numbers less than 100.

DTAFMT Parameter: Specifies the format of the output data.

*FCFC: The output data is to be in the FCFC data format, with the first character of every record being the ANSI forms control character. Parameter value WTR(PUn) is invalid with parameter value DTAFMT(*FCFC). Specify *FCFC if the data is to be printed.

The data can be written to a data base file in the FCFC data format and be printed later by using the Copy File (CPYF) command and specifying an FCFC printer file on the TOFILE parameter.

**DATA:* The output data is to be in the normal data format (that is, no FCFC characters are embedded in the data). Specify *DATA if the data is to go to a data base file and be processed by a program. If the data is directed to a printer file, a single space ANSI control character is the first character in each record.

**CMN:* The output data is to be in the communications data format (that is, still compressed or truncated). *CMN can be used to decrease communications time. However, before the data can be used, the Format RJE Data (FMTRJEDTA) command must be used to change the data to *FCFC or *DATA. If *CMN is specified, the output file must be a data base file with a length of 256.

PGM Parameter: Specifies the qualified name of a user-supplied program to be used with this session description.

*NONE: No user-supplied program is to be used for the RJEF writer.

program-name: Enter the qualified name of the user-supplied program to be used. (If no library qualifier is given, *LIBL is used to find the user-supplied program.)

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF writer are to be recorded.

Note: Messages for RJEF writers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands. If inquiry messages are issued by RJEF, they are sent to the user message queue (if specified) where they must receive a response.

*NONE: No user message queue exists on which the messages for these RJEF writer jobs are to be recorded.

message-queue-name: Enter the qualified name of the user message queue on which this RJEF writer job's messages are to be recorded. If no library qualifier is given, *LIBL is used to find the message queue.

Example

```
ADDRJEWTRE  SSND(RJE.USERLIB) +
      WTR(PR1) +
      FILE(COMPILES.LISTINGS) +
      MBR(*GEN) +
      DTAFMT(*CMN) +
      MSGQ(COMPL.USERLIB)
```

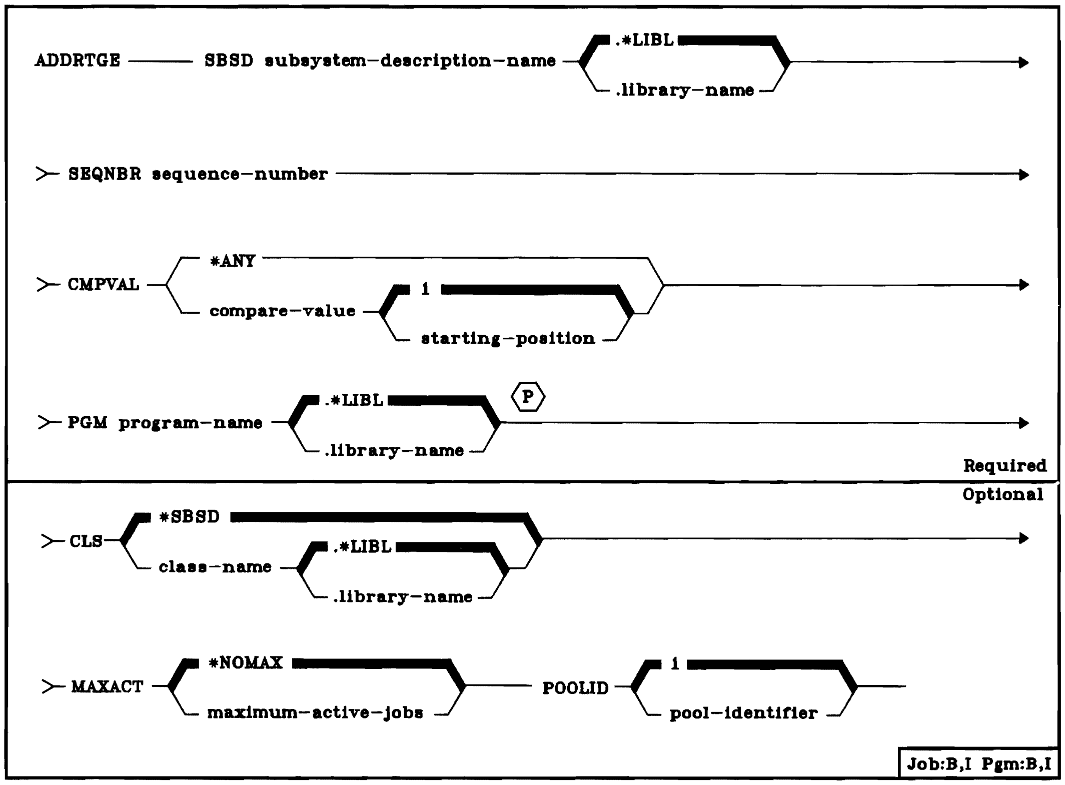
This command adds an RJEF writer entry to the session description named RJE in library USERLIB. The writer added is PR1 (printer 1). The FILE parameter specifies that host data written to this printer should go to a data base file named COMPILES in library LISTINGS. For each file received from the host, RJEF will generate a new data base member. The data is to be written in the communication format (still compressed or truncated).

Messages associated with this writer will be written to the user message queue named COMPL in library USERLIB.

ADDRTGE (Add Routing Entry) Command

The Add Routing Entry (ADDRTGE) command adds a routing entry to the specified subsystem description; the associated subsystem must be inactive at the time. Each routing entry specifies the parameters used to initiate a routing step; for example, the routing entry specifies the name of the program to be executed when the routing data that matches the compare value in this routing entry is received.

Restriction: To use this command, you must have operational and object management rights for the subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description to which the routing entry is to be added. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

SEQNBR Parameter: Specifies the sequence number of the routing entry to be added. Routing data is matched against the routing entry compare values in ascending sequence number order. Searching ends when a match occurs or the last routing entry is encountered. Therefore, if more than one match possibility exists, only the first match is processed. Enter a unique sequence number (1 through 9999) that identifies the routing entry.

CMPVAL Parameter: Specifies a value that is to be compared with the routing data to determine whether this is the routing entry to be used for initiating a routing step. If the routing data matches the routing entry compare value, that routing entry is used. Optionally, a starting position within the routing data character string can be specified for the comparison.

**ANY:* Any routing data is considered to be a match. To specify *ANY, the routing entry must have the highest SEQNBR value of any routing entry in the subsystem description.

compare-value: Enter a value (any character string not exceeding 80 characters) that is to be compared with routing data for a match. When a match occurs, this routing entry is used to initiate a routing step. A starting position within the routing data character string can be specified for the comparison; if no position is specified, 1 is assumed.

<u>1</u>: The comparison between the compare value and the routing data begins with the first position in the routing data character string.

starting-position: Enter a value, 1 through 80, that indicates which position in the routing data character string is the starting position for the comparison. The last character position compared must be less than or equal to the length of the routing data used in the comparison.

PGM Parameter: Specifies the qualified name of the program to be invoked as the (first) program to be executed in the routing step. (No parameters can be passed to the specified program.) (If no library qualifier is given, *LIBL is used to find the program.) If the program does not exist when the routing entry is added, a library qualifier must be specified because the qualified program name is retained in the subsystem description. If QCL.QSYS is specified, the IBM-supplied control language processor, QCL, is invoked in the routing step.

CLS Parameter: Specifies the qualified name of the class to be used for the routing steps initiated through this routing entry. The class defines the attributes of the routing step's execution environment. (For an expanded description of the CLS parameter, see Appendix A.) If the class does not exist when the routing entry is added, a library qualifier must be specified because the qualified class name is retained in the subsystem description.

*SBSD: The class having the same qualified name as the subsystem description, specified by the SBSD parameter, is to be used for routing steps initiated through this entry.

qualified-class-name: Enter the qualified name of the class that is to be used for routing steps initiated through this routing entry. (If no library qualifier is specified, the library list (*LIBL) of the job in which this ADDRTGE command is executed is used to find the class.)

MAXACT Parameter: Specifies the maximum number of routing steps (jobs) that can be concurrently active through this routing entry. (Within a job, only one routing step is active at a time.) When a subsystem is active and the maximum number of routing steps is reached, any subsequent attempts to initiate a routing step through this routing entry will fail. If the routing data was entered interactively, an error message is sent to the user. Otherwise, the job is terminated and a message is sent by the subsystem to the job's log. (For an expanded description of the MAXACT parameter, see Appendix A.)

*NOMAX: There is no maximum number of routing steps that can be concurrently active and processed through this routing entry. (This value is normally used when there is no reason to control the number of routing steps.)

maximum-active-jobs: Enter the maximum number of routing steps that can be concurrently active through this routing entry. If a routing step would exceed this number if it were started, the job is implicitly terminated.

POOLID Parameter: Specifies the pool identifier of the storage pool in which the program is to run.

1: Storage pool 1 of this subsystem is the pool in which the program is to run.

pool-identifier: Enter the identifier of the storage pool defined for this subsystem in which the program is to run. Valid values are 1 through 10.

Examples

```
ADDRTGE  SBSD(PERT.ORDLIB) SEQNBR(46) +
     CMPVAL(WRKSTN2) PGM(GRAPHIT.ORDLIB) +
     CLS(AZERO.MYLIB) +
     MAXACT(*NOMAX) POOLID(2)
```

This command adds routing entry 46 to the routing portion of the subsystem description PERT in the ORDLIB library. To use routing entry 46, the routing data must start with the character string 'WRKSTN2' beginning in position 1. Any number of routing steps can be active through this entry at any one time. The program GRAPHIT in the library ORDLIB is to run in storage pool 2 using class AZERO in library MYLIB.

```
ADDRTGE  SBSD(ABLE.QGPL) SEQNBR(5) +
     CMPVAL(XYZ) PGM(REORD.QGPL) +
     CLS(MYCLASS.LIBX) MAXACT(*NOMAX)
```

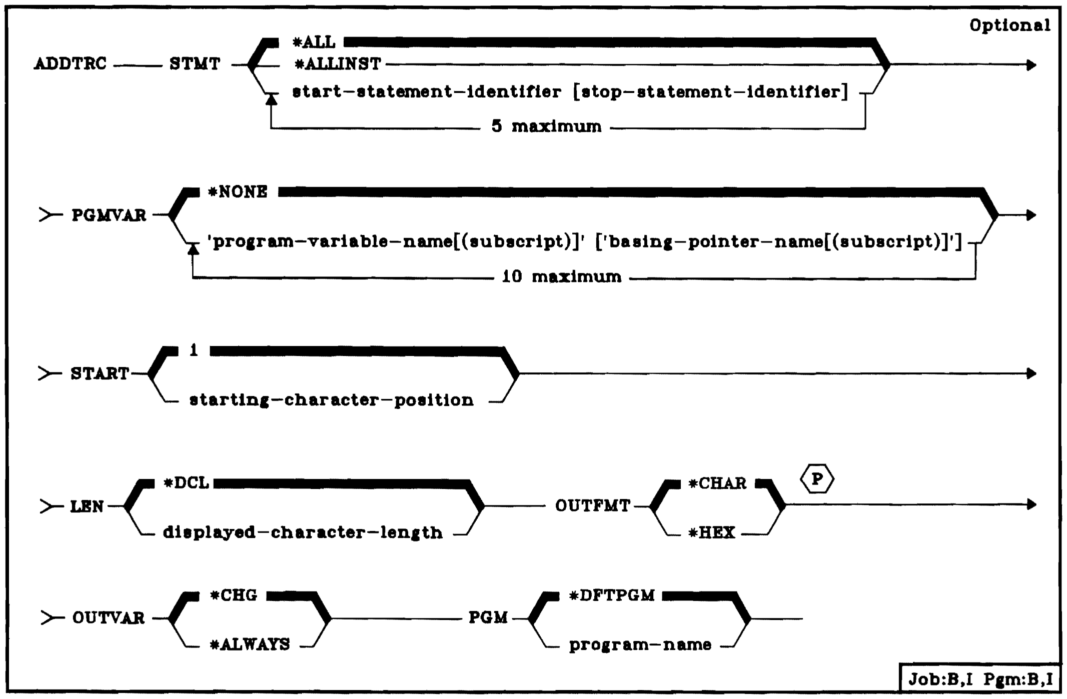
This command adds routing entry 5 to the subsystem description ABLE in the QGPL library. The program REORD in the general purpose library is initiated and uses the class MYCLASS in LIBX when a compare value of XYZ (beginning in position 1) is matched in the routing data. The program runs in storage pool 1, and there is no maximum on the number of active routing steps allowed.

ADDTRC (Add Trace) Command

The Add Trace (ADDTRC) command specifies which program statements in a program are to be traced in debug mode. Up to five ranges of HLL statements or System/38 instructions can be traced during the execution of a program through one or more ADDTRC commands, and up to 10 program variables can be monitored for change within each specified statement range. A separate ADDTRC command is required for each unique variable associated with a statement range. When the specified program being traced is executed, the system records the sequence in which the traced statements were executed and optionally records the value of the variables associated with the trace each time a traced statement is executed. After a trace has been completed, you can display this information using the DSPTRCDTA command.

All of the trace ranges specified in a program are simultaneously active. If both an HLL statement identifier and a System/38 instruction number are used to specify a given trace range, the trace range is treated as an HLL trace range. That is, in addition to tracing the System/38 instruction number specified, the system traces the HLL statement identifiers between that System/38 instruction number and the specified HLL statement identifier.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*


Optional
ADDTRC
STMT
*ALL
*ALLINST
start-statement-identifier [stop-statement-identifier]
5 maximum
PGMVAR
*NONE
'program-variable-name[(subscript)]' ['basing-pointer-name[(subscript)]']
10 maximum
START
1
starting-character-position
LEN
*DCL
displayed-character-length
OUTFMT
*CHAR
*HEX
P
OUTVAR
*CHG
*ALWAYS
PGM
*DFTPGM
program-name
Job:B,I Pgm:B,I

STMT Parameter: Specifies which program statements (or System/38 instructions) are to be traced in one or more statement ranges in the program where tracing is to occur.

**ALL:* All statements in the specified HLL program are to be traced.

**ALLINST:* All System/38 instructions in the specified program are to be traced.

start-statement-identifier [stop-statement-identifier]: Enter the HLL statement identifier (or System/38 instruction numbers) at which tracing is to start and, optionally, the identifier at which tracing is to end. As many as five trace ranges can be specified in the program for each use of this command. Each trace range begins with the specified starting statement, and all following statements are traced until the ending statement is reached. If only a starting statement identifier is specified for a range, the single statement specified is the only statement traced for that range. If System/38 instruction numbers are specified, each number must be preceded by a slash and enclosed in apostrophes: STMT(('/21' '/43')('/62' '/98')) for example.

PGMVAR Parameter: Specifies whether the values of one or more program variables are to be recorded every time a traced statement in an HLL or MI program is executed, and if so, specifies the names of the variables whose values are to be recorded. Depending upon the OUTVAR parameter, the values can be recorded for every trace statement executed, or only when any variable changes value. The program variables can be specified either by their HLL names or by their MI ODV numbers. No more than 10 program variables can be specified.

**NONE:* No program variables are to have their values recorded while tracing is being performed.

'program-variable-name': Enter the names of one or more program variables (no more than 10) whose values are to be recorded while tracing is being performed. If the variable name contains special characters (such as the & in a CL variable name), it must be enclosed in apostrophes. An example is: PGMVAR('&VAR2').

An RPG indicator or an MI ODV number can be specified instead of a program variable name. An example of an RPG indicator is PGMVAR('*IN22'). The ODV number must be preceded by a slash: PGMVAR('/264') for example.

COBOL qualified program variable names may be specified in this parameter. These names have the following syntax:

```
var-name-1 OF/IN var-name-2 OF/IN varname-3...varname-N
```

where varname-N is the last possible variable name that will fit into the input field of the PGMVAR parameter. The input field length for each variable in the PGMVAR parameter is 98 characters. The subscript specified for a qualified variable name may also be a qualified variable name. A qualified variable name (or one with a subscript), including blanks and parentheses, must be contained within the 98-character limit. The 98-character limit includes the necessary keywords (OF/IN) and blanks, but does not include the enclosing apostrophes.

'program-variable-name[(subscript)]': For variables in an array, enter the name of the variable and (optionally) the subscript representing the positional element in the array that is to be displayed. If a subscript is not specified, all elements in the array are displayed. The subscript, if specified, must be enclosed in parentheses, and the variable name and subscript number must be enclosed in apostrophes. No more than 10 sets can be specified, and blanks must separate each set. An example is:

```
PGMVAR('A(5)' 'B(5)' 'C(5)')
```

Either an integer or another variable name can be specified for each subscript.

For COBOL variable names, any combination of variable name length and subscript length that will fit into the 98-character limit is valid. For example, one qualified variable name 98 characters in length (including the keywords OF or IN) can be used with no subscript, or a one-character variable name may be used with a qualified variable name (used as a subscript that uses the other 97 spaces, including parentheses).

For COBOL, the following apply:

- Variable names used in qualifying strings must be high-level language variable names (qualification with ODVs is not allowed).
- Either keyword (OF or IN) is allowed.
- Each OF/IN keyword must be separated from adjacent variable names by at least one blank.

- A qualified variable name can be used as a variable subscript.

- The order the variable names are specified must be from the lowest to the highest levels in the structure.

- Structure levels may be skipped; enough levels must be specified, however, to uniquely identify the variable.

- Qualified variable names must be enclosed in apostrophes, since they contain blank characters.

['basing-pointer-name[(subscript)]']: This set of values in the PGMVAR parameter applies only to MI or HLL programs that support based-on variables. The values can optionally be used with either of the previous two choices to also specify the value in an array that is based on a pointer. The same description of the coding syntax applies here. An example is:

```
PGMVAR(('VAR1(5)' 'PTR1(5)') ('VAR2(8)' 'PTR2(8)'))
```

This example shows that one (different) element in each of two program variables is to be displayed. The fifth element in the array named VAR1, based on the fifth element in the pointer array named PTR1, and the eighth element in the VAR2 array, based on the eighth element in the PTR2 pointer array, are to be displayed.

The field length for the basing pointer name is 24 characters.

START Parameter: Specifies, for character variables only, the beginning position in the variable from which its value is to be recorded when the trace is performed. If more than one character variable is specified in the PGMVAR parameter, the same starting position is used for each one.

1: Recording of the variable is to start with the first position and continue for the length specified in the LEN parameter.

starting-character-position: Enter the starting position number at which the variable is to be recorded. The position number (as well as the *combination* of START and LEN) must be no greater than the length of the shortest variable specified in the PGMVAR parameter.

LEN Parameter: Specifies the number of bytes to be recorded from the character variable specified in the PGMVAR parameter, starting at the position specified in the START parameter. If more than one character variable is specified in the PGMVAR parameter, the same length is used for each one.

<u>*DCL:</u> The character variable is to be recorded to the end of the variable or for 200 bytes, whichever is less.

displayed-character-length: Enter the number of characters that are to be recorded. The length (as well as the *combination* of START and LEN) must be no longer than the length of the shortest variable specified in the PGMVAR parameter.

OUTFMT Parameter: Specifies the format to be used for recording the variables.

<u>*CHAR:</u> Variables are to be recorded in character form.

**HEX:* Variables are to be recorded in hexadecimal form.

OUTVAR Parameter: Specifies whether the values of the program variables are to be recorded only when their values change, or whether they are to be recorded regardless of any of their values being changed. This parameter does not apply if PGMVAR(*NONE) is specified or assumed.

Note: Within each range, the values of all the traced variables are always recorded the first time a statement in the range is executed. For all other statements in the range executed after the first one, the OUTVAR parameter determines when the variables are to be recorded.

<u>*CHG:</u> The system should record the values of all the program variables when one or more of the values are changed by a traced statement being executed.

**ALWAYS:* The system should record the values of the specified variables every time any of the specified trace statements are executed, whether or not any variable had its value changed.

PGM Parameter: Specifies the name of the program that contains the specified statement identifiers or the System/38 instruction numbers that are to be traced. This program name must also be specified in the Enter Debug (ENTDBG) command.

*DFTPGM: The program previously specified as the default program contains the statements to be traced.

program-name: Enter the name of the program that contains the statements to be traced. The specified program must already be in debug mode.

Example

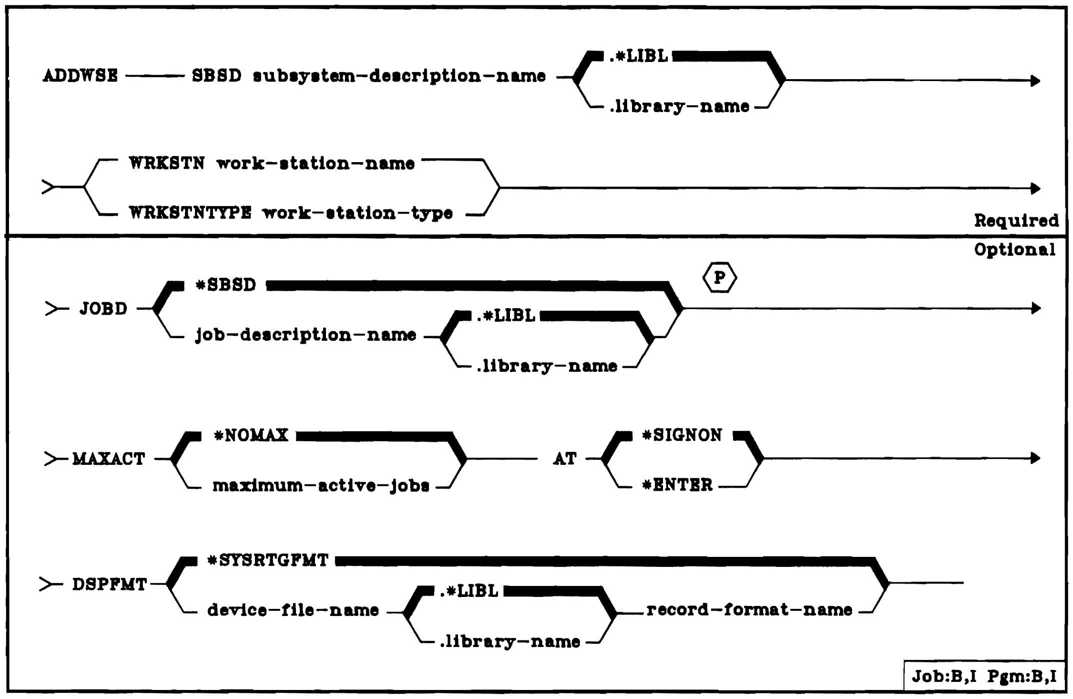
```
ADDTRC  STMT((100 120) (150 200)) +
     PGMVAR('&CTR' '&BRCTR' '&SAM')
```

This command traces program statements in the default program between the ranges of statements 100 through 120 and 150 through 200. Also, whenever the values of any of the program variables &CTR, &BRCTR, and &SAM are changed by one of the traced statements within those ranges, the values of all three are recorded. When all of the traced statements have been executed, or when a breakpoint is reached, the DSPTRCDTA command can be used to display the trace data collected.

ADDWSE (Add Work Station Entry) Command

The Add Work Station Entry (ADDWSE) command adds a work station job entry to the specified subsystem description; the associated subsystem must be inactive at the time. Each entry describes one or more work stations that are to be controlled by the subsystem. The work stations identified in the work station entries are allowed to sign on to or enter the subsystem and execute jobs.

Restriction: To use this command, you must have operational and object management rights for the subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description to which the work station job entry is to be added. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

WRKSTN Parameter: Specifies the name of the work station to be used by the subsystem. The name that was specified in the CRTDEVD command associated with the work station is the name to be used. The work station must have a type code of *CONS, 5251, or 5252 specified in its device description (by the DEVTYPE parameter of the CRTDEVD command).

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

WRKSTNTYPE Parameter: Specifies the type of work station associated with the entry being added. This entry applies to all work stations of this type that do not have specific entries for an individual work station. The following type codes are valid:

Type Code	Device
CONS	System console display
5251	5251 Display Station
5252	5252 Dual Display Station

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

JOBD Parameter: Specifies the qualified name of the job description to be used for jobs that are created and processed through this entry. If the job description does not exist when the entry is added, a library qualifier must be specified because the qualified job description name is retained in the subsystem description.

*SBSD: The job description having the same qualified name as the subsystem description, specified by the SBSD parameter, is to be used for jobs created through this entry.

qualified-job-description-name: Enter the qualified name of the job description that is to be used for jobs created through this entry. If no library qualifier is specified, the library list (*LIBL) of the job in which this ADDWSE command is executed is used to find the job description.

MAXACT Parameter: Specifies, for work stations that use this work station job entry, the maximum number of work station jobs that can be concurrently active. (For an expanded description of the MAXACT parameter, see Appendix A.)

*NOMAX: There is no maximum number of jobs (work stations) that can be concurrently active through this work entry.

maximum-active-jobs: Enter the maximum number of jobs that can be concurrently active through this work entry.

AT Parameter: Specifies when the work stations associated with this job entry are to be allocated. For more information on how work stations are allocated to subsystems, see *Start Subsystem (STRSBS) Command.*

Note: The following should be considered if two or more work station entries specify AT(*SIGNON), they apply to the same work station, and they are in more than one subsystem description: If the work station is varied on while more than one of the subsystems are active, you cannot predict to which subsystem the work station will be assigned.

*SIGNON: The work stations are to be allocated when the subsystem is started if the work station is not already in use (signed on) in another subsystem. A sign-on prompt is to be displayed at each work station associated with this work entry. If a work station becomes allocated to a different subsystem, interactive jobs associated with the work station are allowed to enter this subsystem through the Transfer Job (TFRJOB) command.

**ENTER:* The work stations associated with this work entry are not to be allocated when the subsystem is started. However, the interactive jobs associated with the work stations are allowed to enter this subsystem through the TFRJOB command.

DSPFMT Parameter: Specifies the name of the device file and the name of the record format to be used when the subsystem obtains routing data from the user (that is, when RTGDTA(*GET) is specified in the job description).

*SYSRTGFMT: If routing data is not defined in the referenced job description, the subsystem obtains the routing data from the user using the system-supplied routing data format. This format is described in the *CPF Programmer's Guide.*

qualified-device-file-name record-format-name: Enter the qualified name of the device file to be used by the subsystem to obtain the routing data. (If no library qualifier is given, *LIBL is used to find the device file description.) If the device file does not exist when the work station entry is added, a library qualifier must be specified because the qualified name of the device file is retained in the subsystem description. Also, enter the name of the record format that defines the format to be used when the subsystem obtains the routing data from the user.

Examples

```
ADDWSE  SBSD(ORDER.LIB7) WRKSTNTYPE(5251) +
    JOBD(QCTL) AT(*SIGNON)
```

This command adds a work station job entry to a subsystem description named ORDER in library LIB7. All 5251 work stations are allocated to this subsystem when the subsystem is started (unless they are already active in a previously started subsystem). The work stations are to be signed on at demand. When sign-on is complete, the IBM-supplied job description QCTL is used to initiate the routing step.

```
ADDWSE  SBSD(ORDER.LIB7) WRKSTN(A12) +
    JOBD(ORDENT.LIB7) AT(*SIGNON)
```

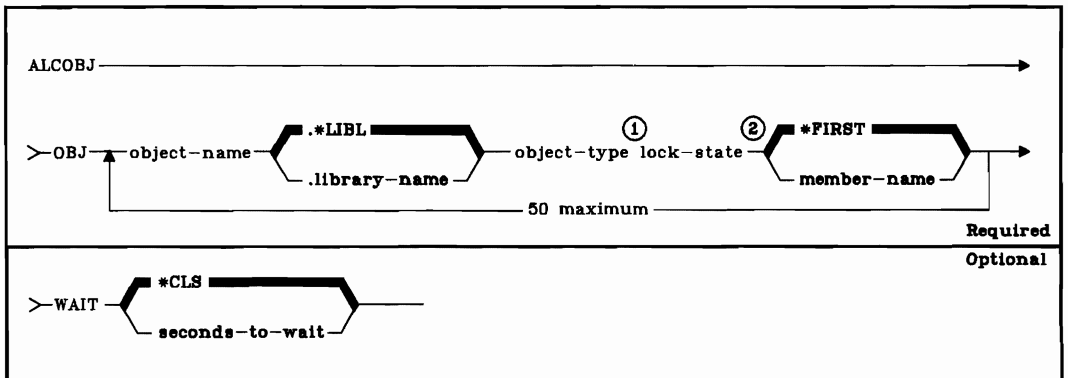
This command adds a work station job entry to a subsystem description named ORDER in library LIB7. Work station A12 is to be signed on at demand. When sign-on is complete, the system-supplied routing data format is displayed at the work station if the job description ORDENT in LIB7 specifies *GET as the routing data.

ALCOBJ (Allocate Object) Command

The Allocate Object (ALCOBJ) command is used in a routing step to reserve an object or list of objects for use later in the routing step. If an object that is needed in the routing step is not specified in an ALCOBJ command, an allocation is made automatically when the object is used. The objects are deallocated either automatically at the end of the routing step or when the DLCOBJ command is used.

For this command to be executed successfully: (1) the object must exist on the system, (2) the user issuing the command must have object existence, object management, or operational rights for the object, and (3) the object must not be allocated to another job in a lock state that inhibits or restricts the requested lock state for the entire time specified in the WAIT parameter. If the allocation cannot be completed, none of the locks are granted and a message is sent to the job that issued the command. If the command is issued from a program, the Monitor Message (MONMSG) command can be used to determine that the allocation was not successful.

Note: If a file is being allocated that is affected by a file override, the ALCOBJ command ignores the override and attempts to allocate the file named in the OBJ parameter.



① The values of only six object types are valid: *DEVD, *DTAARA, *FILE, *LIB, *MSGQ, and *SBSD. Refer to the OBJ parameter description for the lock states that are valid for each object type.
② If valid for the specified object type, one of the following lock states can be specified: *SHRRD, *SHRUPD, *SHRNUP, *EXCLRD, or *EXCL.

Job:B,I Pgm:B,I

OBJ Parameter: Specifies the qualified names of one or more CPF objects that are to be allocated to the job, the type of each object specified, the lock state of each object, and if the object is a data base file, a member name can optionally be specified. (If no library qualifier is given for an object, *LIBL is used to find the object. Note that the LIB and DEVD object types do not reside in user libraries and, therefore, cannot be qualified with a library name.)

If the member name is not specified for a data base file, the member name defaults to *FIRST and the first member of the file is allocated. (If the specified file is a logical file, the physical file members associated with the members of the logical file are also allocated to the job.)

For each object named, enter: the object name (optionally qualified) followed by the object type, one lock state value, and (if applicable) the file member name to be allocated.

The lock states that can be specified are:

Value	Lock State Meaning
*SHRRD	Shared for read
*SHRUPD	Shared for update
*SHRNUP	Shared, no update
*EXCLRD	Exclusive, allow read
*EXCL	Exclusive, no read

For an explanation of each lock state, refer to the *CPF Programmer's Guide.*

Multiple locks can be specified for the same object within the same job with duplicate or different lock states. Each lock is held separately. For example, if an *EXCL lock is already held for an object and a second *EXCL lock request occurs, the second lock is acquired. Both locks must be released in the job (deallocated with the DLCOBJ command) before another job can access the same object. If a user already has an object allocated with one lock state and wants to use a different lock state, he should first use the ALCOBJ command to request the new lock with the desired lock state and then use the DLCOBJ command to release the old lock (with the old lock state).

To determine if a device description can be allocated, the DSPDEVSTS (Display Device Status) command should be entered. You can determine if the device description can be allocated by using information shown by the DSPDEVSTS command and from the following table. If, for the appropriate device type,

- No job name is associated with the device, or if the job name associated with the device is of the same job that is to issue the ALCOBJ command, and

- The status field of the display indicates the following value,

you can attempt to allocate the device description object.

Device Type	Status
3203, 3262, 3410, 5211, 5424, 72MD, *BSCT	VARIED ON
CONSOLE	VARIED ON or SIGNON DISPLAY
*BSC, *PLU1	VARY ON PENDING[1] or VARIED ON
5224, 5225, 5251, 5252, 5256, 5291, 5292	VARY ON PENDING[1], VARIED ON[2], or SIGNON DISPLAY
[1]Switched device [2]Device is powered on	

The device description will not be allocated if one of the following conditions exist:

- Another job is allocating the device description
- Another job or object is opening a file to the device
- Another job is varying the device off

If the device description object cannot be allocated, reissue the DSPDEVSTS command to determine the status of the device.

Only six of the CPF object types can be specified on the ALCOBJ command. Of these six, some cannot use all of the lock states. The following table shows the CPF object types that can be specified and the lock states allowed for each one (A = allowed).

	Lock States				
Object Type	***SHRRD**	***SHRUPD**	***SHRNUP**	***EXCLRD**	***EXCL**
*DEVD				A	
*DTAARA	A	A	A	A	A
*FILE	A	A	A	A	A
*LIB	A	A	A	A	
*MSGQ	A				A
*SBSD					A

When the user requests an exclusive lock on a logical file member, the lock occurs on both the logical file member and the associated physical file members. No other user can use the physical file members, even through some other logical file member.

WAIT Parameter: Specifies the number of seconds that the program is to wait for the object to be allocated. If the object cannot be allocated in the specified wait time, a message, which can be detected by a MONMSG command, is sent to the program.

*CLS: The default wait time specified in the class description used by the routing step is to be used as the wait time for the object to be allocated.

seconds-to-wait: Enter the number of seconds that the program is to wait for (all of) the specified objects to be allocated. Valid values are 0 through 32767 (32 767 seconds). If 0 is specified, no wait time is allowed.

Example

```
ALCOBJ  OBJ((FILEA.LIBB *FILE *EXCL MEMBERA))
```

This command exclusively allocates MEMBERA of FILEA in LIBB to the routing step in which the allocate command is used. If MEMBERA is unavailable, the number of seconds to wait for it to become available is the default wait time defined for the class used by the routing step.

ANSLIN (Answer Line) Command

The Answer Line (ANSLIN) command identifies a communication line that has been manually answered by the system operator. This command indicates that the operator has manually answered an incoming call and validated the requirements of the caller. When this command is entered, CPF executes the manual answer sequence for the line and, when completed, instructs the operator to select data mode on the modem.

```
                                                                    Required
ANSLIN ——— LINE line-description-name ———
                                                                    Job:I
```

LINE Parameter: Enter the name of the communication line that is being answered.

Example

```
ANSLIN LINE(LINE01)
```

This command answers an incoming call on a line named LINE01.

APYJRNCHG (Apply Journaled Changes) Command

The Apply Journaled Changes (APYJRNCHG) command applies the changes that have been journaled (for a particular member of a data base file) to a backup version of the file to recover the file after an operational error or some form of damage. The journaled changes are applied from the indicated starting point, either at the point at which a file was last saved or at a particular entry on the journal, until the designated ending point has been reached. The ending point can be the point at which the file has had all changes applied, a designated entry has been reached, a designated time has been reached, or the file was opened or closed by a job.

Note: The DSPJRN command can be used to help determine the desired starting and/or ending points.

A list of physical files and members can be specified. The journaled changes for physical file members are applied in the order that the journal entries are encountered on the journal (the same order the changes were made to the physical file members).

If an error is encountered at any point during the application of the journaled entries, the command terminates and the file member(s) may be only partially updated from the journal entries. (Termination errors include partial damage to a receiver and any logical error in the file member, such as a duplicate key.) The command also terminates when a journal entry is encountered that indicates that:

- The member was reorganized
- The member was restored
- Journaling was stopped for the member
- The member was deleted or saved with storage freed
- Journal IPL synchronization failed, or
- The member had its changes applied or removed (through the APYJRNCHG or RMVJRNCHG command.

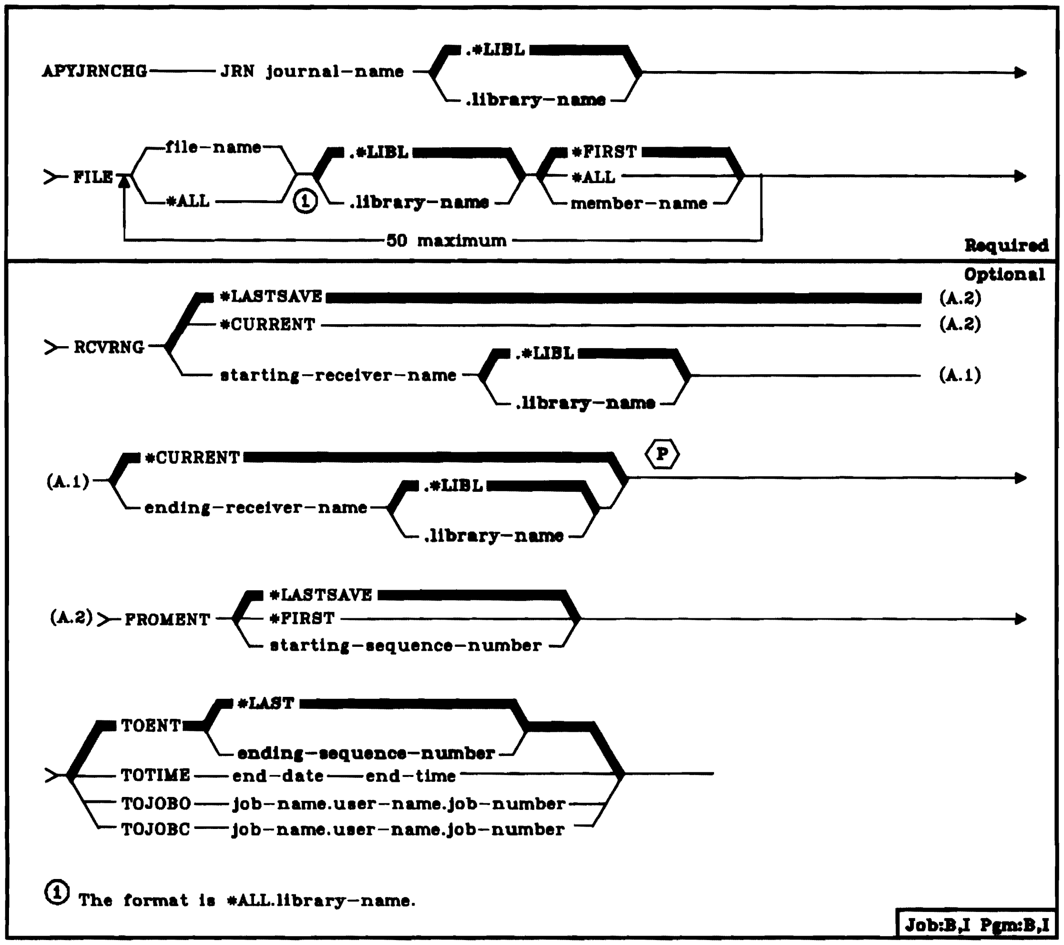
The user of the command may reissue the command, specifying a new starting sequence number, if a restart is possible.

It is possible to apply changes even if the sequence numbers have been reset. The system will handle this condition, send an informational message, and continue to apply the changes. If journal receivers are attached and detached in pairs (dual receivers), the system will always attempt to use the first of the two receivers (the first of the two shown in the DSPJRNA receiver directory). When the first of the pair is not accessible (for example, damaged or not found), the system will attempt to use the second receiver of the pair. If neither receiver is accessible, the application of changes will terminate.

Restrictions: The files specified on this command must currently be having their changes journaled and they must have been journaled to the specified journal throughout the period indicated on the command. The files indicated on the command are allocated exclusively while the changes are being applied. If a file cannot be allocated, the command terminates and no journaled changes are applied. If there is no journal entry that corresponds to the 'FROM' or 'TO' option, the command is terminated and no journaled changes are applied.

If the journal sequence numbers have been reset within the range of receivers specified, the first occurrence of the FROMENT or TOENT parameter will be used, if they are specified.

Note: If the application terminates for one of the members specified, it terminates for all of the members specified.



JRN Parameter: Specifies the qualified name of the journal associated with the journal entries that are to be applied. (If no library qualifier is given, *LIBL is used to find the journal.)

FILE Parameter: Specifies the qualified name of the physical data base file to which journal entries are to be applied.

file-name: Enter the name of the physical data base file that is to have its journal entries applied. (If no library qualifier is given, *LIBL is used to find the file.)

**ALL:* All physical files within the specified library whose changes are being journaled to the specified journal will have their journal entries applied. The library name *must* be specified. If *ALL is specified and you do not have the required authority for all the files in the library, a message is sent and the application terminates.

The FILE parameter also specifies the name of the member within the file that is to have its journal entries applied.

FIRST: The first member in the file is to have journal entries applied.

**ALL:* All members in the file are to have their journal entries applied.

member-name: Enter the name of the member within the file that is to have its journal entries applied.

If *ALL is specified for the first part of this parameter, the value specified for the member name is used for all applicable files within the library. For example, if *FIRST is specified, the first member of all applicable files in the library will have the changes applied.

Note: A maximum of 256 members can have their changes applied with one invocation of the command. If this maximum is exceeded, an exception is signaled and no changes are applied. You must change the values entered on the FILE parameter so that the limit is not exceeded.

RCVRNG Parameter: Specifies the first and last journal receivers to be used in applying the journal entries. The system will begin the application with the first journal receiver (specified by the first value) and will proceed through the receivers until the last receiver (specified by the last value) is processed. If dual receivers were used at any time, the first of the receivers will always be used when chaining through the set of receivers. If any problem is encountered in the receiver chain (such as a damaged receiver or a receiver not online) before the journal entries are applied, the system will attempt to use the second of the dual receivers. If the second of the receivers is damaged or offline, or if the problem is encountered during the application of journal entries, the operation will terminate.

*LASTSAVE: The range of journal receivers to be used will be determined by the system, based on save information for the files that are to have their journaled changes applied. This parameter value is only valid if FROMENT(*LASTSAVE) is also specified.

**CURRENT:* Only the currently attached receiver will be used in applying the journal entries.

First Parameter Value

starting-receiver-name: Enter the name of the journal receiver to be used as the first (oldest) receiver. (If no library qualifier is given, *LIBL is used to find the receiver.)

Second Parameter Value

*CURRENT: Application of journal entries will continue for all journal receivers in the chain, beginning with the receiver specified by the first parameter value through the currently attached journal receiver.

ending-receiver-name: Enter the name of the journal receiver to be used as the last (newest) receiver with journal entries to be applied. If the end of the receiver chain is reached before encountering a receiver of this name, the operation is not performed and an escape message is sent. (If no library qualifier is given, *LIBL is used to find the receiver.)

Note: The maximum number of receivers that can be used in a range of receivers is 256. If this maximum is exceeded, an exception will be signaled and no changes will be applied.

FROMENT Parameter: Specifies the entry to be used as the starting point for applying changes that have been journaled.

*LASTSAVE: Specifies that the journal entries are to be applied beginning with the first journal entry after the file member that was last saved. The system will determine the actual starting position for each of the files specified on the command. The parameter value implies that the file was just restored onto the system.

Some validation is performed by the system for each member specified, such as whether the date and time of the restore is after the date and time of the last save. The system also verifies that the date and time of the saved version of the file member that was restored onto the system match the date and time that the file member was last saved, as indicated on the journal.

If the dates and times do not match, the application of journaled changes is not attempted and an inquiry message is sent to the system operator requesting a cancel or ignore response. (If an ignore response is given to the message, the operation is attempted. A cancel response causes the operation to terminate.)

**FIRST:* The journal entries are to be applied beginning with the first journal entry in the first receiver supplied to this command.

starting-sequence-number: Specifies the sequence number of the first journal entry that is to be applied from the journal entries supplied.

TOENT Parameter: Specifies the entry to be used as the ending point for applying changes that have been journaled.

*LAST: Specifies that journal entries are to be applied through the last entry.

ending-sequence-number: Specifies the sequence number of the last entry that is to be applied to the file member.

TOTIME Parameter: Specifies the time and date of the last journal entry to be applied to the file member. The first entry with that or the next earlier time will be the ending point for the application of journal entries. The format of the date must be defined by the system values QDATFMT and, if separators are used, QDATSEP. The time can be entered as four or six digits (hhmm or hhmmss) where hh = hours, mm = minutes and ss = seconds. If colons are used to separate the time values, the string must be enclosed in apostrophes ('hh:mm:ss').

TOJOBO Parameter: Specifies that the journal entries are to be applied only until the indicated job (fully qualified job name) first opens any physical file member (or logical member defined over the physical member) in the list of members specified on the FILE parameter that are to have their journal entries applied. (This will be the ending point for all members specified.)

TOJOBC Parameter: Specifies that the journal entries are only to be applied until the indicated job (fully qualified job name) last closes any physical file member (or logical member defined over the physical member) that is in the list of members specified on the FILE parameter that are to have their journal entries applied, or until the indicated job was terminated. (This will be the ending point for all members specified.)

Examples

```
APYJRNCHG  JRN(JRNACT.FIN)  FILE(RCVABLE.FIN)
```

This command will cause the system to apply to the first member of file RCVABLE in library FIN all changes that were recorded in journal JRNACT in library FIN since the file was last saved. The receiver range will be determined by the system. The changes will be applied (beginning with the first recorded change on the receiver chain after the file was last saved) and will continue through all applicable journal entries.

```
APYJRNCHG  JRN(JRNA)  FILE((PAYROLL.LIB2 JAN))
      RCVRNG(RCV22 RCV25)  FROMENT(*FIRST)
```

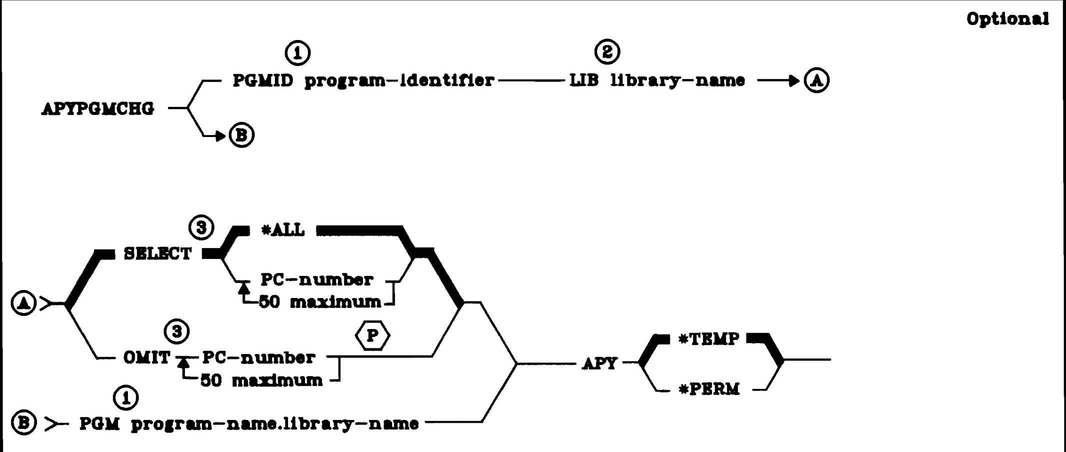
This command will cause the system to apply all changes recorded in journal JRNA to member JAN of file PAYROLL in library LIB2. The journal receivers containing the journaled changes are contained in the receiver chain starting with receiver RCV22 and ending with receiver RCV25. The application will begin with the first change recorded on this receiver chain. The library search list (*LIBL) is used to find the journal JRNA and the journal receivers RCV22 and RCV25.

APYPGMCHG (Apply Programming Change) Command

The Apply Programming Change (APYPGMCHG) command applies programming changes (PCs) or program patches to a program in the specified library. If a PC is to be applied, it must have been loaded by the LODPGMCHG command. If a program patch is to be applied, it must have been created by the PCHPGM command.

When a PC is applied, it completely replaces the affected objects in the licensed program. Either PCs or program patches can be applied temporarily or permanently. If they are applied temporarily, the replaced object is saved by the system and can later be restored to the program by the RMVPGMCHG command. If PCs or program patches are applied permanently, the replaced object is deleted from the system.

The APYPGMCHG command can be used to apply only immediate PCs, not deferred PCs. Deferred PCs must be applied through the deferred programming changes display. This display is explained in the *System/38 Operator's Guide*.



① PGMID and PGM are mutually exclusive. One or the other must be specified. To code the following parameters *positionally*, you must code them in this order, using *N for those not being specified: PGMID, LIB, SELECT, and OMIT.
② If PGMID is specified, LIB is required.
③ SELECT or OMIT is valid only if PGMID is specified.

Job:B,I Pgm:B,I

PGMID Parameter: Specifies the identifier of the program to which PCs are to be applied. The PGM parameter cannot be specified if PGMID is specified. If PGMID is not specified, PGM must be specified.

LIB Parameter: Specifies the name of the library that contains the program specified by the PGMID parameter. If PGMID is specified, LIB must be specified.

SELECT Parameter: Specifies which of the previously loaded PCs are to be applied to the specified program. The OMIT parameter cannot be specified if SELECT is specified.

*ALL: All the PCs that were loaded are to be applied to the program. If all PCs cannot be applied, messages are sent indicating the PCs that were not applied and the reasons they were not applied (for example, prerequisite PCs had not been applied).

PC-number: Enter the PC identification numbers of the individual programming changes that are to be applied. A maximum of 50 PC numbers can be specified.

OMIT Parameter: Specifies that all the loaded PCs are to be applied except for those specified in this parameter. Enter the PC numbers of the programming changes that are to be omitted (not applied) when all the rest are applied. A maximum of 50 PC numbers can be specified. The OMIT parameter cannot be specified if individual PC numbers are specified in the SELECT parameter.

PGM Parameter: Specifies the qualified name of the program to which a program patch is to be applied. This parameter is valid only for applying program patches. It cannot be specified if PGMID is specified. If PGM is not specified, PGMID must be specified.

APY Parameter: Specifies whether the PCs or program patches are to be applied on a temporary basis or permanently applied. Permanently applied changes cannot be removed; temporary changes can be removed by the RMVPGMCHG command.

*TEMP: The changes are to be applied as temporary changes.

**PERM:* The changes are to be applied permanently.

Examples

```
APYPGMCHG  PGMID(5714SS1) LIB(QSYS)
```

This command applies all the programming changes currently in the library QSYS that affect CPF (program number 5714SS1). The changes are temporarily applied.

```
APYPGMCHG  PGMID(5714SS1) LIB(QSYS) +
      SELECT(00003 00008 00012) APY(*PERM)
```

This command permanently applies PCs 00003, 00008, and 00012 to the CPF in library QSYS.

```
APYPGMCHG  PGM(PAYPGM3.PAYLIB)
```

This command temporarily applies the program patch (that was created by the PCHPGM command) to the program PAYPGM3 in library PAYLIB.

CALL (Call Program) Command

The Call (CALL) command invokes an executable program named on the command, and passes control to it. Optionally, the program or user issuing the CALL command can pass parameters to the called program. The CALL command can be used in batch jobs, in interactive jobs, and in both compiled and interpreted CL. When the called program completes its execution, it can return control to the calling program by issuing the RETURN command.

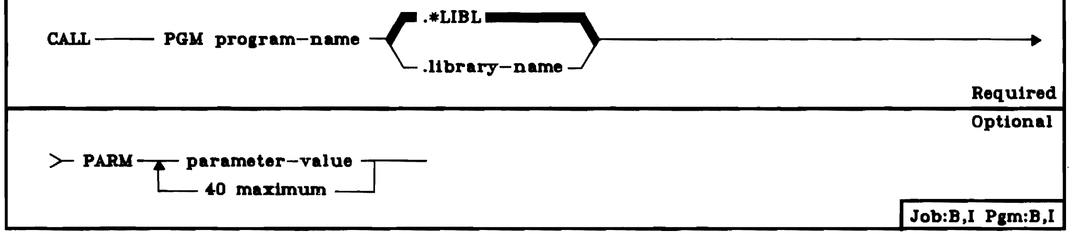
When the CALL command is issued by a CL program, each parameter value passed to the called program can be a character string constant, a numeric constant, a logical constant, or a CL program variable. When parameters are passed, the value of the constant or CL variable is available to the program that is called. Parameters cannot be passed in any of the following forms: lists of values, qualified names, expressions, null parameters (that is, a parameter whose value is null, specified by *N), or keyword parameters. A maximum of 40 parameters can be passed to the called program.

When parameters are passed to a program using the CALL command, the values of the parameters are passed in the order in which they appear on the CALL command; this order must match the order in which they appear in the parameter list in the calling program.

Parameters in a called program can be used in place of its variables. However, no storage in the called program is associated with the variables it receives. Instead, when a variable is passed, the storage for the variable is in the program in which it was originally declared. When a constant is passed, a copy of the constant is made in the calling program and that copy is passed to the program called.

The result is that when a variable is passed, the called program can change its value and the change is reflected in the calling program. When a constant is passed, and its value is changed by the called program, the changed value is not known to the calling program. So, if the calling program calls the same program again, it reinitializes the values of constants, but not variables.

Restriction: The user must have operational rights or one of the data rights for the program being called.



PGM Parameter: Specifies the qualified name of the program to be invoked by the calling program. (If no library qualifier is given, *LIBL is used to find the called program.)

PARM Parameter: Specifies one or more parameter values that are to be passed to the called program. Each of the values can be specified only in one of the following forms: a character string constant, a numeric constant, a logical constant, or a program variable.

The type and length of each parameter must match in both the calling and receiving programs. The number of parameters and the order in which they are sent and received must also match. If the CALL command is entered interactively or in noncompiled batch mode, you must ensure that, for each parameter being passed on the command, its type and length matches that expected by the called program.

Parameters can be passed and received as follows:

- Character string constants of 32 bytes or less are *always* passed with a length of *32* bytes (padded on the right with blanks). If a character constant is longer than 32 bytes, the entire length of the constant is passed. If the parameter is defined to contain more than 32 bytes, the calling program must pass a constant containing exactly that number of bytes. Constants longer than 32 characters are *not* padded to the length expected by the receiving program.

  The receiving program can receive less than the number of bytes passed (in this case, no message is sent). For example, if a program specifies that 4 characters are to be received and ABCDEF is passed (padded with blanks in 26 positions), only ABCD is accepted and used by the program. Quoted character strings can also be passed.

- Decimal constants are passed in packed form and with a length of (15 5), where the value is 15 digits long, of which 5 digits are decimal positions. Thus if a parameter of 12345 is passed, the receiving program must declare the decimal field as (15 5); the parameter is received as 1234500000 (which is 12 345.00000).

- Logical constants are passed as 1 byte with a logical value of '1' or '0'.

- A program variable can be passed if the call is made from a CL program, in which case the receiving program must declare the field to match the variable defined in the calling CL program. For example, if a CL program defines a decimal variable named &CHKNUM as (5 0), the receiving program must declare the field as packed with 5 digits total, with no decimal positions.

If either a decimal constant or a program variable can be passed to the called program, the parameter should be defined as (15 5), and any calling program must adhere to that definition. If the type, number, order, and length of the parameters do not match between the calling and receiving programs (other than the length exception noted previously for character constants), unpredictable results will occur.

The value *N cannot be used to specify a null value because a null value cannot be passed to another program.

Examples

```
CALL  PGM(PAYROLL)
```

The program named PAYROLL is called with no parameters being passed to it. The library list is used to locate the called program.

```
CALL  PAYROLL '1'
```

The program named PAYROLL is called with a character constant passed as a quoted string. The program must declare a field of from 1 to 32 characters to receive the constant. The library list is used to locate the called program.

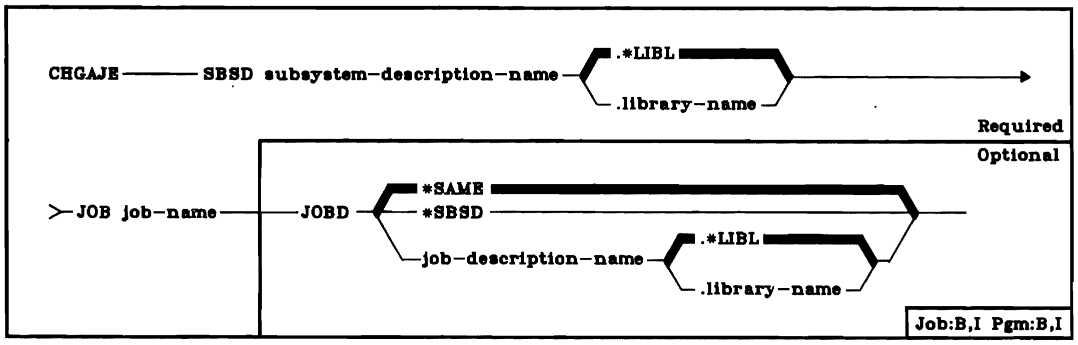
```
CALL  PAYROLL.LIB1 (CHICAGO 1234 &VAR1)
```

The program named PAYROLL located in library LIB1 is invoked by the calling program. The calling program is passing three parameters: a character string (CHICAGO), a decimal value (1234.00000), and the contents of the CL variable &VAR1. The attributes of the variable determine the attributes of the third parameter.

CHGAJE (Change Autostart Job Entry) Command

The Change Autostart Job Entry (CHGAJE) command is used to specify a different job description for a previously defined autostart job entry in the specified subsystem description. The subsystem associated with the subsystem description must be inactive when the change is made.

Restriction: To use this command, you must have operational and object management rights for the subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description containing the autostart job entry to be changed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

JOB Parameter: Specifies the simple name that identifies the autostart job entry in the subsystem description whose attributes are to be changed.

JOBD Parameter: Specifies the name of the job description to be used for the job that is initiated by this autostart job entry.

<u>*SAME:</u> The job description specified in the existing autostart job entry is to be used.

**SBSD:* The job description having the same qualified name as the subsystem description, specified by the SBSD parameter, is to be used for the initiated job.

qualified-job-description-name: Enter the qualified name of the job description to be used for the job initiated by this autostart job entry. (If no library qualifier is given, the library list (*LIBL) of the job in which this CHGAJE command is executed is used to find the job description.) If the job description does not exist when the entry is changed, a library qualifier must be specified because the qualified job description name is retained in the subsystem description.

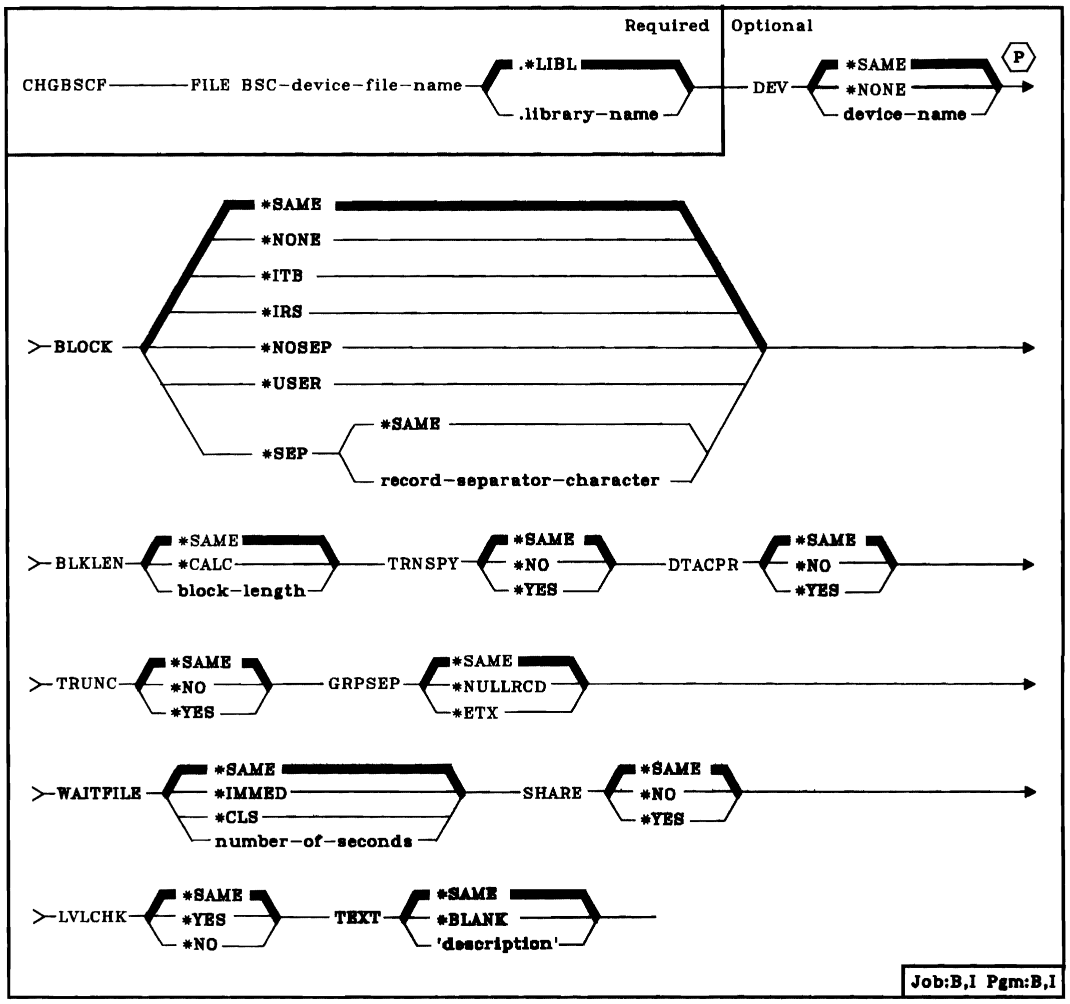
Example

```
CHGAJE  SBSD(PAYROLL.QGPL) JOB(INIT) +
     JOBD(MANAGER)
```

This command changes the JOBD parameter, for the autostart job entry INIT, to MANAGER. The work entry is in the PAYROLL subsystem description that is in the QGPL library. The library list is used to locate the job description MANAGER. When the correct library is determined, the qualified job description name is placed in the subsystem description for this autostart job entry.

CHGBSCF (Change BSC File) Command

The Change BSC File (CHGBSCF) command can be used to change certain attributes of a BSC device file. If nothing is specified, or if *SAME is specified, that attribute of the file remains unchanged.



FILE Parameter: Specifies the qualified name of the BSC device file whose description is being changed. (If no library qualifier is given, *LIBL is used to find the file.)

DEV Parameter: Specifies the name of the System/38 BSC device that is to be used with the BSC device file to send and receive data records.

SAME: The device name, if any, specified in the device file description remains the same.

**NONE:* No device name is to be specified. Any device names to be specified must be specified later in an OVRBSCF command, in another CHGBSCF command, or in the HLL (high-level language) program that opens the file.

device-name: Enter the name of the BSC device that is to be used with this BSC file. The device name must be known to the system via a device description.

BLOCK Parameter: Specifies whether the system or the user will block and deblock transmitted records. With this parameter, you may specify one of the following conditions of record formatting:

no blocking/deblocking: The record format described in the DDS (data description specifications) is the format for both the record and the block.

user blocking/deblocking: You must provide the BSC controls needed to describe the record format to the system.

system blocking with record separator characters: You specify the record separator character used by the system to determine record boundaries within the block.

system blocking of fixed-length records: The system uses fixed-length records, and blocks/deblocks records accordingly. The record separator character is added when a record is transmitted, and removed before the record is returned to your program.

If you specify a parameter value other than *NONE, or *USER, records will be blocked as required by the system for output and deblocked on input. Blocking may be done with or without record separator characters. If TRNSPY(*YES) is specified, the records may be blocked without record separator characters, by specifying BLOCK(*NOSEP), or the records may be transmitted one record at a time by specifying BLOCK(*NONE). By specifying BLOCK(*USER), you may block records to include the BSC transparency controls. If TRNSPY(*NO) is specified, all blocking options are valid. The record length, when used, is obtained from the device file. A maximum of 512 records will be blocked for transmitting. When the system blocks and deblocks the records, record separator characters and control characters will not be passed to your program as data.

<u>*SAME:</u> Specifies that the BLOCK parameter value is to remain the same.

**NONE:* Specifies that no blocking or deblocking will be done by the system.

**ITB:* Specifies that the records are to be blocked or deblocked, based on the location of an ITB (intermediate text block) control character. For input files, a record will be delimited by locating the next ITB character. An ETX (end of text) or ETB (end-of-transmission block) character will be used as an ITB character to delimit records. For output files, an ITB character will be inserted after the record. If that is the last character of the block, the ITB will be replaced by an ETX or an ETB character.

**IRS:* Specifies that the records are to be blocked or deblocked, based on the location of an IRS (interrecord separator) character. For input files, a record will be delimited by locating the next IRS character. For output files, an IRS character will be inserted after the record.

**NOSEP:* Specifies that no record separator character is contained within the transmission block sent to or received from the device. The system will block and deblock the records according to a fixed record length, as specified in the DDS (data description specifications) format specifications.

**USER:* Specifies that your program is to provide all control characters, including record separator characters, BSC framing characters, transparency characters, and so forth, necessary to transmit records.

When transmitting records, BSC device support will scan the buffer for the last nonblank byte to determine the length of the data to be transmitted. For this reason, you must ensure that the unused portion of the buffer contains blanks.

For receiving, you must specify with an ETX control character the end of the received text. BSC device support will pad the remaining buffer space with blanks.

This method of blocking allows you to transmit and receive variable-length data blocks by using a single record format capable of accommodating the maximum block length. Except for the padding and truncating with blanks, BSC device support passes the data to and from the system when user blocking is specified.

If you are using the Remote Job Entry Facility, BLOCK(*USER) must be specified. For more information on RJEF, refer to the *RJEF Programmer's Guide*.

Before selecting this option, you should have a good understanding of the device and of the BSC support characteristics. For more information on BSC support characteristics, refer to the *IBM System/38 Data Communications Programmer's Guide*, SC21-7825.

**SEP:* Specifies that the records are to be blocked or deblocked, based on the location of a user-specified record separator character. For input files, a record will be delimited by locating the next record separator character. For output files, a record separator character will be inserted after the record.

record-separator-character: Specifies a unique one-byte record separator character. The record separator character may be specified as two hexadecimal characters, as in BLOCK(*SEP X'FD'), or as a single character, as in BLOCK(*SEP @).

The following is a list of BSC control characters that must not be used as record separators:

EBCDIC	**BSC Control**
X'01'	SOH (Start of header)
X'02'	STX (Start of text)
X'03'	ETX (End of text)
X'10'	DLE (Data link escape)
X'1D'	IGS (Interchange group separator)
X'1F'	ITB (Intermediate text block)
X'26'	ETB (End-of-transmission block)
X'2D'	ENQ (Enquiry)
X'32'	SYN (Synchronization)
X'37'	EOT (End of transmission)
X'3D'	NAK (Negative acknowledgment)

You must be certain that none of these control characters are specified in your data as record separator characters.

BLKLEN Parameter: Specifies the maximum block length (in bytes) for data to be transmitted. This parameter changes the block length specified in the program or in the device file.

<u>*SAME</u>: The block length is not to be changed.

**CALC:* The block length is to be determined by the system. The length will be 512 bytes or the length of the largest record in the device file, whichever is greater.

block-length: The maximum block length of records to be sent when using this device file. The value must be at least the size of the largest record to be sent. Valid values are 1 through 8192.

TRNSPY Parameter: Specifies whether the text transparency feature is to be used when sending blocked records. The text transparency feature permits the transmission of all 256 EBCDIC character codes; you should use this feature when transmitting packed or binary data fields.

*SAME: The usage condition of the text transparency is not to be changed.

*NO: The text transparency feature is not to be used.

*YES: The text transparency feature is to be used, which permits the use of all 256 EBCDIC character codes. *YES is valid only when BLOCK(*NONE), BLOCK(*NOSEP), or BLOCK(*USER) is specified.

Note: Transparency of received data is determined by the data stream; therefore, this parameter is not relevant for received data. If TRNSPY(*YES) is specified with BLOCK(*USER), BSC ignores the transparency indicator during put operations. You must provide the proper controls with the data in order to get transparent transmission of data. For example, you must initially specify the DLE and STX control characters; System/38 provides the remaining control characters for transparent transmission of data.

DTACPR Parameter: Specifies whether blanks in BSC data will be compressed for output and decompressed for input. If TRNSPY(*YES) is specified, or if the line description specifies CODE(*ASCII), DTACPR(*YES) is ignored.

*SAME: The data compression is to remain as specified.

*NO: No data compression or decompression is to occur.

*YES: Data is to be compressed on output and decompressed on input.

TRUNC Parameter: Specifies whether trailing blanks are to be removed from output records. TRUNC(*YES) cannot be specified if BLOCK(*NOSEP) or TRNSPY(*YES) is specified.

*SAME: The TRUNC parameter is not to be changed.

*NO: Trailing blanks are not to be removed from output records.

*YES: Trailing blanks are to be removed from output records.

GRPSEP Parameter: Specifies a separator for groups of data (data sets, documents, and so forth).

*SAME: The value specified in the BSC file description is not to be changed.

**NULLRCD:* Specifies that a null record (STXETX) is to be used as a data group separator.

**ETX:* A transmission block ending with the BSC control character ETX is to be used as a data group separator.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter changes the wait time specified in the program or in the device file. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*SAME: The wait time specified in the device file description for the needed objects is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is to be made.

**CLS:* The default wait time specified in the class description is to be used as the wait for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the BSC device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the BSC device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a write operation in that program produces the next output record.

*SAME: The value specified in the BSC file description is not to be changed.

**NO:* An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file.

This parameter changes the value specified in the program or in the device file. Level checking cannot be done unless the program contains the record format identifiers.

*SAME: The value specified in the BSC file description is not to be changed.

**YES:* The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not match, an open exception occurs and an error message is sent to the program requesting the open.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

TEXT Parameter: Specifies the user-defined text that describes the BSC device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

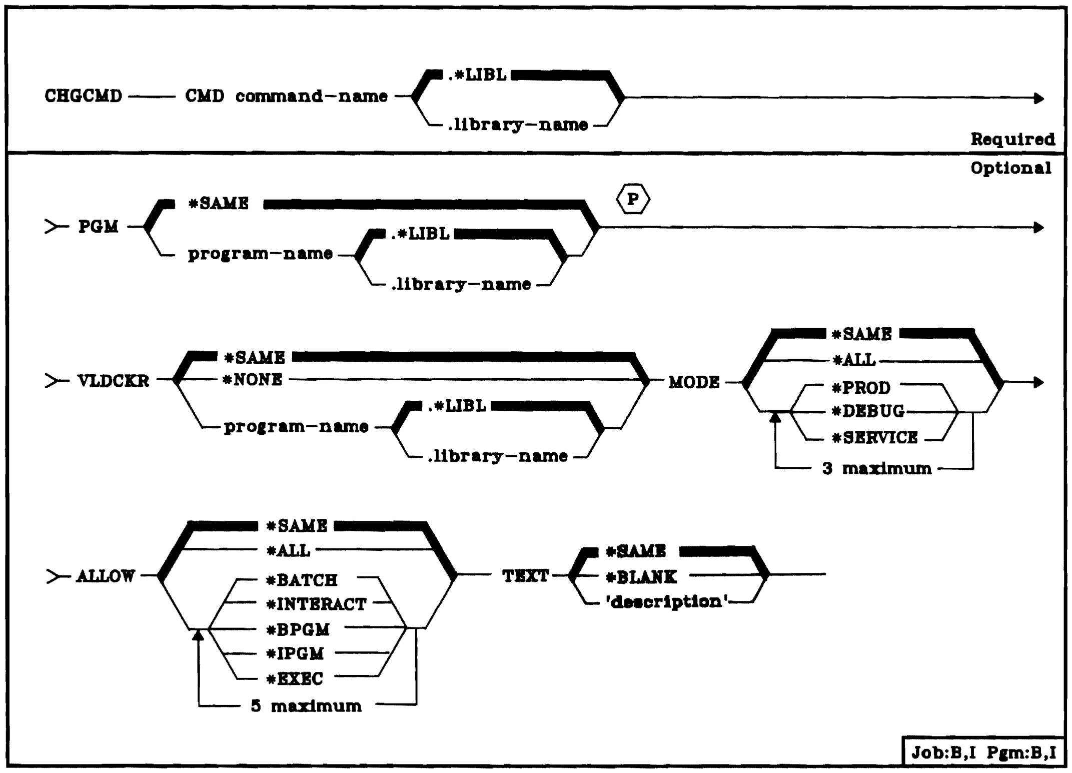
```
CHGBSCF  FILE(TRANSD1.COMM1)  BLOCK(SEP X'EE')  WAITFILE(10)
```

This command changes the record separator character the system uses for record blocking to hex EE. This command also changes to 10 seconds the period of time the program will wait for file resources to be allocated. All other values specified for device file TRANSD1 in library COMM1 remain as specified in the CRTBSCF command.

CHGCMD (Change Command) Command

The Change Command (CHGCMD) command changes some of the attributes of a command definition. It can specify a different command processing program (CPP) to execute the command; it can also change the mode or where it can be executed, and the text description for the command. CL programs that use the command being changed by the CHGCMD command do *not* have to be re-created. The CHGCMD command does *not* change the parameter descriptions or validity checking information in the command definition object.

Restrictions: The user must be authorized to use the CHGCMD command and have object management and operational rights for the command that is being changed. The CHGCMD command can be used to change only the attributes of a created CL command (that is, those attributes that were specified on the CRTCMD command). The CHGCMD command cannot be used to change attributes of *statements*, such as command definition statements.



CMD Parameter: Specifies the name of the command to be changed. The command can be a user-defined or IBM-supplied command. (If no library qualifier is given, *LIBL is used to find the command.)

PGM Parameter: Specifies the name of the command processing program (CPP) that is to execute the command.

SAME: The current CPP is not to be changed.

qualified-program-name: Enter the name of the CPP that is to process the command specified in CMD. (If no library qualifier is given, *LIBL is used to find the CPP at command execution time.)

VLDCKR Parameter: Specifies the name of a program that, at compile time, performs additional validity checking on the parameters in the command to be executed. The validity checker is invoked to perform additional user-defined validity checking beyond that specified by the command definition statements in the source file, and beyond the syntax checking done on the command when it is compiled.

SAME: The current validity checking program is to be used for this command.

**NONE:* There is no separate validity checking program for this command. All validity checking is done by the command analyzer and the command processing program.

qualified-program-name: Enter the qualified name of the validity checker that is to check the validity of the command whenever the command is executed or validity checked (provided variables and expressions are not used). (If no library qualifier is given, *LIBL is used to find the program at command execution time.)

MODE Parameter: Specifies the modes of operation that the command can be used in. One or more of the modes can be specified.

SAME: The modes of operation in which the command can be used remain the same.

**ALL:* The command is to be valid in all the modes of operation: production, debug, and service.

**PROD:* The command is to be valid in the production mode.

**DEBUG:* The command is to be valid in the debugging mode.

**SERVICE:* The command is to be valid in the service mode.

ALLOW Parameter: Specifies where the command can be executed. One or more of the following options can be specified.

*SAME: Where the command can be executed is not to be changed.

**ALL:* The command is valid in a batch input stream, in a CL program, or when executed interactively. It can also be passed to the system program QCAEXEC to be executed.

**BATCH:* The command is valid in a batch input stream, external to a compiled CL program.

**INTERACT:* The command is valid when executed interactively, external to a compiled CL program.

**BPGM:* The command can be included in a compiled CL program that executes in the batch input stream.

**IPGM:* The command can be included in a compiled CL program that executes interactively.

**EXEC:* The command can be used as a parameter on the CALL command and be passed as a character string to the system program QCAEXEC to be executed. If *EXEC is specified, either *BATCH or *INTERACT must also be specified.

TEXT Parameter: Specifies the user-defined text that briefly describes this command and its function. The text specified here replaces any previous text. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

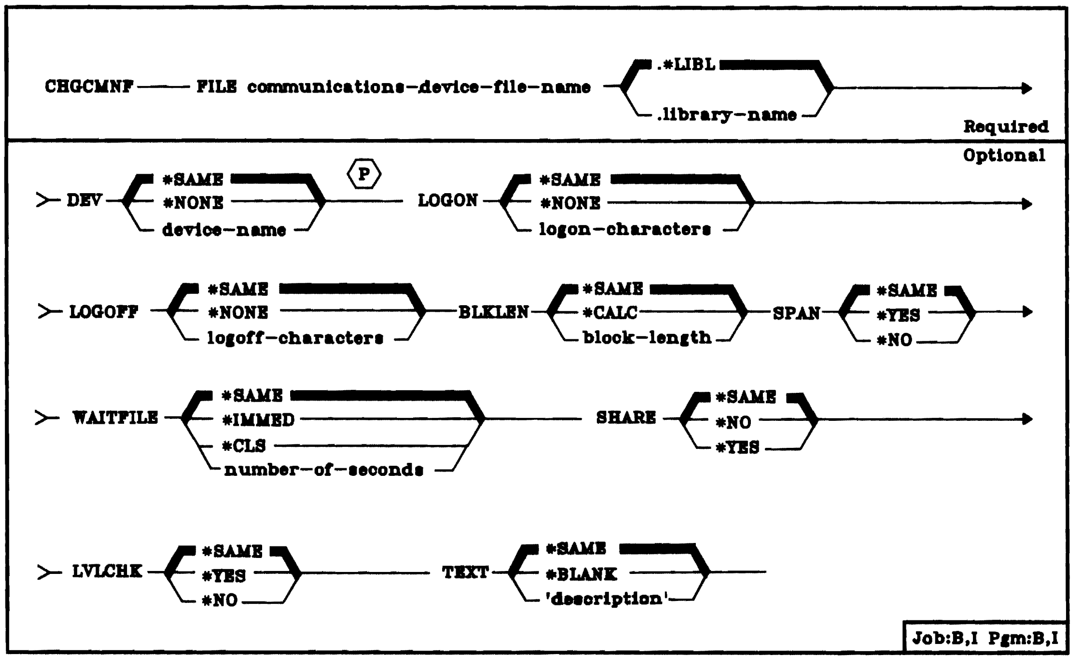
Examples

```
CHGCMD  CMD(PAYROLL) VLDCKR(PAYVLDPGM.LIB01)
```

The validity checking program for the PAYROLL command is the program named PAYVLDPGM located in library LIB01. All other attributes of the PAYROLL command remain the same.

CHGCMNF (Change Communications File) Command

The Change Communications File (CHGCMNF) command changes attributes in the file description of a communications device file.



FILE Parameter: Specifies the qualified name of the communications device file whose description is being changed. (If no library qualifier is given, *LIBL is used to find the file.)

DEV Parameter: Specifies the name of the System/38 communications device that is to be used with this device file to send and receive data records from another system.

SAME: The device name, if any, specified in the device file description, remains the same.

**NONE:* No device is to be specified. It must be specified later in an OVRCMNF command, in another CHGCMNF command, or in the HLL program that opens the file.

device-name: Enter the name of the communications device that is to be used with this communications file. The device name must already be known on the system via a device description.

LOGON Parameter: Specifies the text that is to be transmitted to the primary logical unit host when the file is opened. The text is limited to 80 characters, and its format is host-dependent.

*SAME: The logon text specified in the communications file description is not to be changed.

**NONE:* No logon text is to be specified.

logon-characters: Enter the text that is to be transmitted to the primary logical unit host when this file is opened.

LOGOFF Parameter: Specifies the logoff text that is to be transmitted to the primary logical unit host when the file is closed. The text is limited to 80 characters, and its format is host-dependent.

*SAME: The logoff text specified in the communications file description is not to be changed.

**NONE:* No logoff text is to be specified.

logoff-characters: Enter the text that is to be transmitted to the primary logical unit host when this file is closed.

BLKLEN Parameter: Specifies, in bytes, the maximum block length for data that is to be transmitted or received by the communications file.

*SAME: The block length specified in the device file description stays the same.

**CALC:* The device support chooses an optimum value based on the record sizes in the device file. Device support calculates the smallest multiple of 1792 that is greater than or equal to the largest record in the device file. The calculated value includes the new line (NL) or form feed (FF) characters that follow each record when RCDSEP(*YES) is specified.

block-length: Enter a value (256 through 32767) that specifies the maximum block length of records to be processed by this communications device file. This value must be at least the size of the largest message expected to be transmitted or received. Also, it must include the new line (NL) or form feed (FF) characters that follow each record when RCDSEP(*YES) is specified.

SPAN Parameter: Specifies whether logical records are to be allowed to span request unit boundaries during output operations.

<u>*SAME:</u> The boundary characteristics of request units are not to be changed.

**YES:* The system places as much data as possible into a request unit. When this parameter value is specified, a request unit may contain any of the following:

- One or more complete records
- One or more complete records plus a partial record
- A partial record

**NO:* The system places as many complete records as possible into a request unit, but will never allow a request unit to contain a partial record.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

<u>*SAME:</u> The wait time specified in the device file description for the needed objects is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the communications device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the communications device file can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used.

SAME: The value specified in the communications file description is not to be changed.

**NO:* An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file, a new ODP to the file is created and activated.

**YES:* An ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given an internal system identifier when the format is created.

SAME: The value specified in the communications file description stays the same.

**YES:* The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not all match or they have not been specified in the program, an open error message is sent to the program requesting the open.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

TEXT Parameter: Specifies the user-defined text that describes the communications device file. (For an expanded description of the TEXT parameter, see Appendix A.)

**SAME*: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

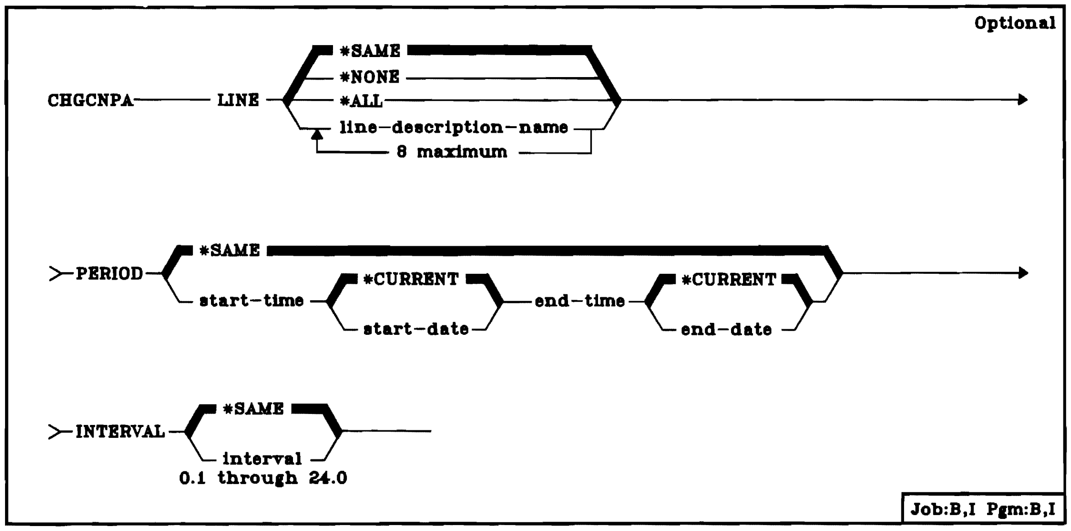
Example

```
CHGCMNF  FILE(FILEB)  WAITFILE(*IMMED)
```

This command changes the number of seconds that the program waits for FILEB resources to be allocated such that the program does not wait; when FILEB is opened, its file resources must be allocated immediately or an error occurs.

CHGCNPA (Change CSNAP Attributes) Command

The Change CSNAP Attributes (CHGCNPA) command changes the CSNAP (communications statistics network analysis procedure) current short-term statistics sampling parameters.



LINE Parameter: Specifies the name(s) of the line(s) CSNAP is to monitor for short-term statistics. Up to 8 line names may be used, or if *ALL is specified, all communications lines that are described to the system will have this set of changed CSNAP parameters applied.

*SAME: CSNAP is to monitor for short-term statistics for the same line(s) that are currently set in the system. If no lines are being sampled and *NONE is specified, an error message will be sent.

**NONE:* No lines are to be sampled.

**ALL:* All communications lines currently described to the system will be sampled for statistics.

line-description-name: Enter up to 8 line description names to be sampled by CSNAP.

PERIOD Parameter: Specifies the period of time for which CSNAP start-time statistics are to be sampled and recorded.

This parameter contains two lists of two values each. Refer to the syntax diagram for the order in which the values are specified. If this parameter is not specified, the default of *SAME is used and the period that is set in the system is used. The period of sampling is defined by using the start-time and start-date, followed with the end-time and end-date. Under any of the following conditions, an error message will result:

- If the end-time, end-date is earlier than the start-time, start-date.
- If the end-time, end-date is more than 120 hours later than the start-time, start-date.
- If the end-time, end-date is more than 3 hours later than the start-time, start-date and the INTERVAL parameter specifies a sampling interval of less than one hour.
- If the period and interval values have been reset to zero and new values have not been entered.

*SAME: The CSNAP short-term statistics values that are currently set in the system will continue to be used.

**current:* The samplings that are to be taken for the CSNAP short-term statistics are for the current date, between the specified starting and ending times.

start-time: Enter the time at which CSNAP short-term statistics are to begin.

start-date: Enter the date on which the first CSNAP statistic samplings are to be taken. The starting date specified is not to exceed 5 days (120 hours) from the present system date.

end-time: Enter the time at which CSNAP statistics are to be ended.

end-date: Enter the date on which CSNAP statistic samplings are to end. The ending date specified is not to exceed 5 days (120 hours) from the present system date.

INTERVAL Parameter: Specifies the interval spacing for which CSNAP recording is to be done. This sampling interval can range from 0.1 hours up to 24 hours in 0.1 increments. The value should be entered in the form HH.H (hours and one-tenth hours).

*SAME: The recording interval of CSNAP statistics is to remain the same as that currently set in the system.

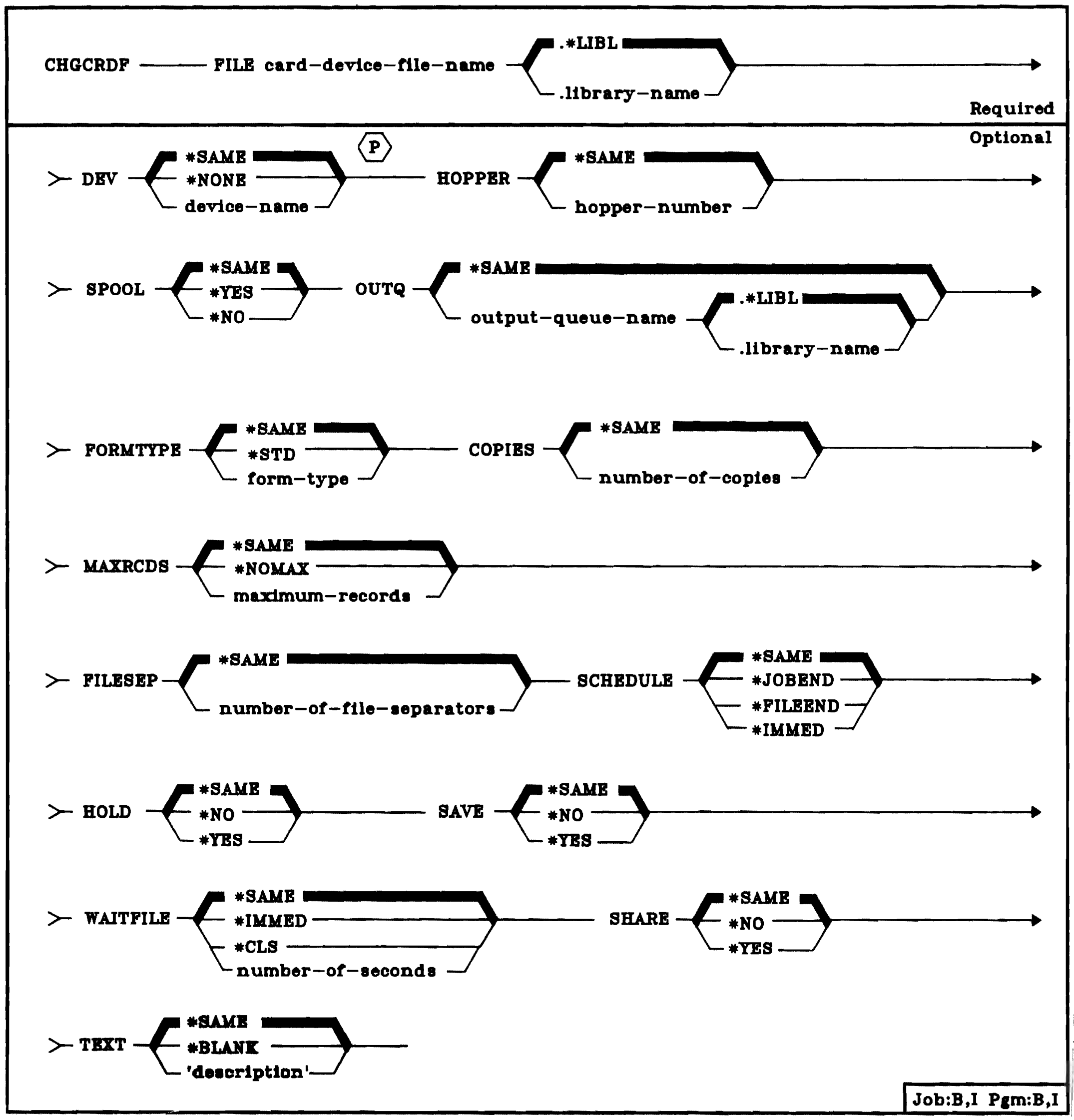
Example

```
CHGCNPA LINE(LN1) PERIOD((133000 *CURRENT) (153000 *CURRENT)) +
      INTERVAL(.3)
```

This command sets the CSNAP short-term attributes to start recording at 13:30 on today's date, and to end sampling at 15:30 on today's date, sampling at intervals of 0.3 hour.

CHGCRDF (Change Card File) Command

The Change Card File (CHGCRDF) command changes, in the file description, one or more of the attributes of the specified card device file.



FILE Parameter: Specifies the qualified name of the card device file whose description is being changed. (If no library qualifier is given, *LIBL is used to find the file.)

DEV Parameter: Specifies the name of the card device that is to be used with this device file to perform input/output data operations. The device name of the IBM-supplied card device description is QCARD96.

*SAME: The device name, if any, specified in the device file description remains the same.

**NONE:* No device name is to be specified. It can be specified later on an OVRCRDF command or when the card device file is opened.

device-name: Enter the name of the device that is to be used with this card device file. The device name must already be known on the system via a device description.

HOPPER Parameter: Specifies from which hopper of the MFCU the cards are to be fed when this card device file is used. Valid entries are 1 (for the primary hopper) and 2 (for the secondary hopper).

*SAME: The hopper number specified in the device file description is not to be changed.

hopper-number: Enter either a 1 or a 2 to indicate which hopper of the MFCU is to be used.

SPOOL Parameter: Specifies whether the input or output data for the card device file is to be spooled. If SPOOL(*NO) is specified, the following parameters in this command are ignored: OUTQ, FORMTYPE, COPIES, MAXRCDS, FILESEP, SCHEDULE, HOLD, and SAVE.

*SAME: The value specified in the device file description is not to be changed.

**YES:* The data is to be spooled. If this file is opened for input, an inline data file having the specified name is processed; otherwise, the next unnamed inline spooled file is processed. (For a discussion of named and unnamed inline files, see the *CPF Programmer's Guide.*) If this file is opened for output, the data is spooled for processing by a card, diskette, or print writer.

**NO:* The data is not to be spooled. If this file is opened for input, the data is read directly from the card device. If this is an output file, the data is sent directly to the device to be punched or printed as the output becomes available.

OUTQ Parameter: Specifies, for spooled output only, the name of the output queue for the spooled output file.

<u>*SAME:</u> The same output queue specified in the device file description is to be used.

qualified-output-queue-name: Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.) The IBM-supplied output queue that can be used by the card file is the QPUNCH output queue, stored in the QGPL library.

FORMTYPE Parameter: Specifies, for spooled output only, the type of form (cards) on which the card device is to produce the output. The identifiers used to indicate the type of cards are user-defined and must not be longer than 10 characters.

<u>*SAME:</u> The type of cards specified in the device file description remains the same.

**STD:* The standard card type used in your installation is to be used for output from jobs using this card device file.

form-type: Enter the identifier of the card type to be used for output from jobs using this card device file. A maximum of 10 alphameric characters can be specified.

COPIES Parameter: Specifies, for spooled output files only, the number of copies (card decks) of the output to be produced by the card device.

<u>*SAME:</u> The number of copies specified in the device file description is not to be changed.

number-of-copies: Enter a value, 1 through 99, that indicates the number of identical card decks to be produced when this device file is used.

MAXRCDS Parameter: Specifies the maximum number of records that can be in the spooled output file for this card device file.

<u>*SAME:</u> The maximum number of records specified in the device file description remains the same.

**NOMAX:* No maximum is specified for the number of records that can be in the spooled file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of records that can be in the spooled output file.

FILESEP Parameter: Specifies, for spooled output files only, the number of separator cards to be placed at the beginning of each output card deck, including between multiple copies of the same output. Each separator card will contain the file name, file number, job name, the user name, job number, and the date and time when the job was executed.

*SAME: The number of separator cards specified in the device file description is not to be changed.

number-of-file-separators: Enter the number of separator cards to be placed at the beginning of each card deck produced by spooled jobs that use this card device file. Valid values are 0 through 9. If 0 is specified, at the end of each output file a message is sent to the message queue specified on the STRCRDWTR command that started the writer; the message indicates that the output just produced is to be removed from the device.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a spooling writer.

*SAME: The time specified in the device file description when spooled output can begin remains the same.

**JOBEND:* The spooled output file is to be made available to the spooling writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the spooling writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the spooling writer as soon as the file is opened in the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The spooled file is made available to a spooling writer when it is released by the RLSSPLF (Release Spooled File) command.

*SAME: The value specified in the device file description is not to be changed.

**NO:* The spooled output file is not to be held on the output queue. The spooled output is made available to a spooling writer based on the SCHEDULE parameter.

**YES:* The spooled output file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled file is to be saved (left on the output queue) after the output has been produced.

SAME: The value specified in the device file description is not to be changed.

**NO:* The spooled file data is not to be retained on the output queue after it has been produced.

**YES:* The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

SAME: The wait time specified in the device file description is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the card device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the card device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

SAME: The value specified in the device file description is not to be changed.

**NO:* An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Specifies the user-defined text that describes the card device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

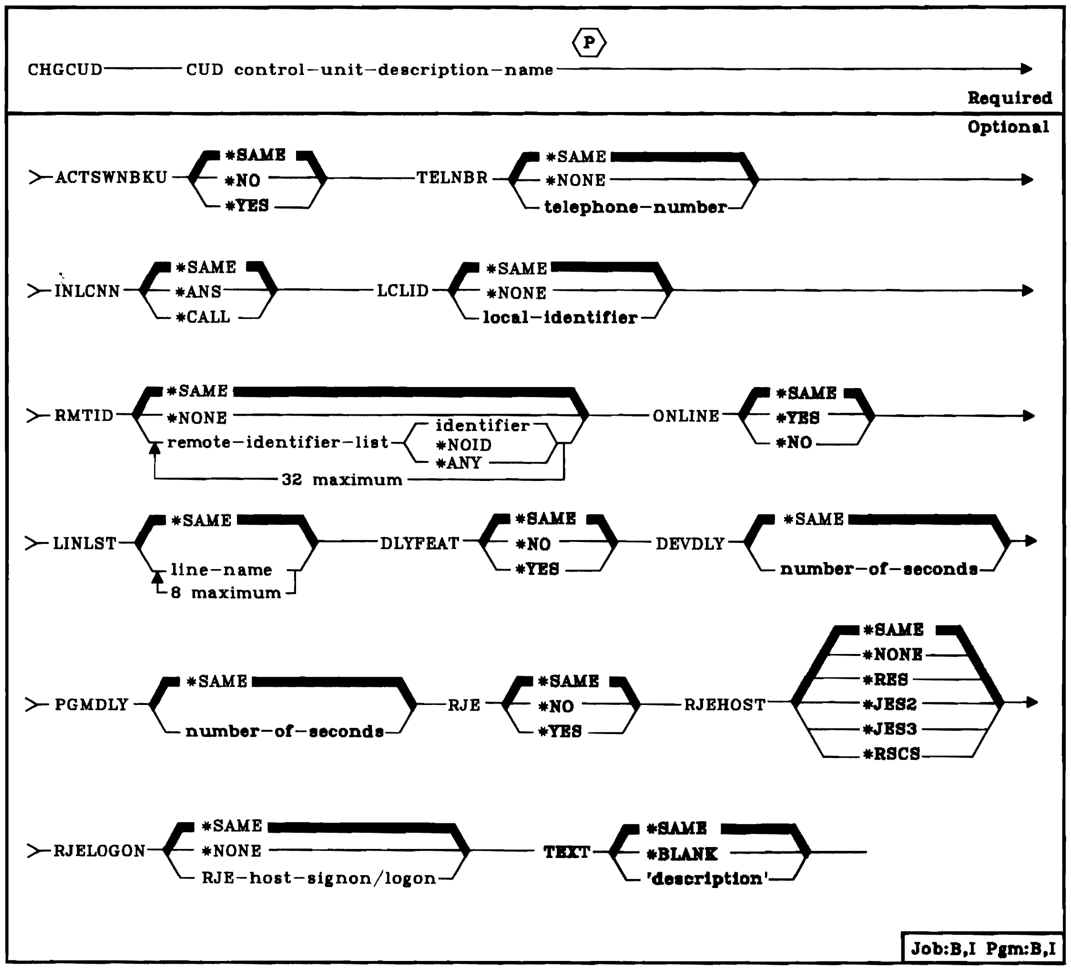
Examples

```
CHGCRDF  FILE(PCHRPT.ACCREC)  COPIES(3)
```

This command changes the description of the card device file named PCHRPT stored in the ACCREC library. The number of copies (card decks) to be punched is changed to three. No other values in the file description are changed.

CHGCUD (Change Control Unit Description) Command

The Change Control Unit Description (CHGCUD) command changes some of the attributes in the description of the specified control unit. The control unit must be varied offline before the attributes can be changed, except for the ONLINE and TEXT attributes. The changes become effective when the control unit is varied online.


CHGCUD
CUD control-unit-description-name
P
Required
Optional
ACTSWNBKU *SAME *NO *YES
TELNBR *SAME *NONE telephone-number
INLCNN *SAME *ANS *CALL
LCLID *SAME *NONE local-identifier
RMTID *SAME *NONE remote-identifier-list identifier *NOID *ANY
32 maximum
ONLINE *SAME *YES *NO
LINLST *SAME line-name
8 maximum
DLYFEAT *SAME *NO *YES
DEVDLY *SAME number-of-seconds
PGMDLY *SAME number-of-seconds
RJE *SAME *NO *YES
RJEHOST *SAME *NONE *RES *JES2 *JES3 *RSCS
RJELOGON *SAME *NONE RJE-host-signon/logon
TEXT *SAME *BLANK 'description'
Job:B,I Pgm:B,I

CUD Parameter: Specifies the name of the control unit description that is to have one or more of its attributes changed.

ACTSWNBKU Parameter: Specifies, for BSC, PU2, or 5251 control units attached to nonswitched lines only, whether the switched network backup feature (if the feature is installed) is to be activated or de-activated. This feature lets you bypass a broken nonswitched connection (leased line) by converting the line to a switched line operation. (This parameter applies only if SWITCHED(*NO) and SWNBKU(*YES) are specified in the control unit description; *SAME must be specified for TYPE(*BSCT).)

*SAME: The value specified in the control unit description is not to be changed.

**NO:* The backup feature is to be de-activated if it was active.

**YES:* The backup feature is to be activated if it is not active.

TELNBR Parameter: Specifies, for remote control units only, the telephone number of this control unit if it is associated with a switched line, or if it is associated with a nonswitched line and has the switched network backup feature. The telephone number (1 to 16 digits long) is dialed at the System/38 site to establish a connection to this control unit. (This parameter applies only to switched lines and to nonswitched lines with SWNBKU(*YES) specified in the control unit description.) The telephone number is:

- Sent to the autocall unit, if automatic calling is used to establish a connection to this control unit
- Displayed to the system operator, if manual calling is used to call this control unit

*SAME: The value specified in the control unit description is not to be changed.

**NONE:* The line is nonswitched, so no telephone number is specified.

telephone-number: Enter the telephone number that is to be used to call this control unit.

INLCNN Parameter: Specifies, for remote control units only, the method to be used to make the initial connection over a switched line between System/38 and the control unit. (This parameter applies to switched lines and to control units that have the switched network backup feature activated because ACTSWNBKU(*YES) was specified.)

SAME: The method of initial connection remains the same.

**ANS:* The initial connection is made by System/38 when it answers an incoming call from this control unit.

**CALL:* The initial connection is made by a call initiated from System/38.

LCLID Parameter: Specifies the local identifier for identifying System/38 to the remote BSC control unit.

SAME: The local identifier is not to be changed.

**NONE:* No local identifier is to be specified.

local-identifier: A string of from 2 to 15 characters for identifying System/38 to a remote BSC control unit. If a 2-character identifier is specified, both characters must be the same. The identifier cannot contain BSC control characters.

RMTID Parameter: Specifies a list of identifiers for remote BSC control units.

SAME: The list of identifiers is not to be changed.

**NONE:* Specifies that there are to be no remote identifiers. *NONE is valid only for BSC control units with SWITCHED(*NO) and SWNBKU(*NO) specified. This parameter value should not be confused with *NOID, which is a valid remote identifier.

remote-identifier-list: Enter the identifier or a list of identifiers (32 maximum) used by remote BSC control units. If a 2-character identifier is specified, both characters must be the same. The identifier cannot contain BSC control characters. *NOID specifies a null identifier; a null identifier can be specified by itself or within a list of identifiers. *ANY instructs System/38 to accept any identifier sent by a remote BSC control unit. If *ANY is specified, it must be the last or only identifier in the list.

ONLINE Parameter: Specifies whether the control unit is to be varied online automatically when the Control Program Facility (CPF) is started. After CPF is started, the VRYCTLU (Vary Control Unit) command can be used to modify the status of the control unit.

*SAME: The value specified in the control unit description is not to be changed.

**YES:* The control unit is to be online when CPF is started.

**NO:* The control unit is to be offline when CPF is started. The VRYCTLU command must be used to put the control unit online, making it operational.

LINLST Parameter: Specifies, for switched connections only, a list of line names that identify the lines that can be connected to this control unit. (This parameter is valid only if SWITCHED(*YES) or SWNBKU(*YES) is specified in the associated CRTCUD command. The parameter does not apply to the 3411 tape control unit or to the work station controller.)

*SAME: The list of line names is not to be changed.

line-name: Enter the names of up to eight lines that can be connected to this control unit. The same line name can be used more than once. For each line name specified, a line description by that name must already exist. The number of line names specified here cannot exceed the number of line names currently in the line list of this control unit description.

By specifying one or more entries here, the entire existing list is replaced; that is, if two line names are specified here to change an existing list of four names, the first two names in the existing list are changed to the specified names, and the last two are replaced with null lines.

DLYFEAT Parameter: Specifies, for nonswitched lines only, whether periodic attempts should be made to contact this control unit (to establish a delayed connection) if the initial attempt to establish a connection is not successful. (This parameter is valid only for 5251 work station control units.)

*SAME: The value specified in the control unit description is not to be changed.

**NO:* Only one attempt is to be made to establish a connection between the line and the control unit.

**YES:* Periodic attempts are to be made to establish a delayed connection between the line and the control unit.

DEVDLY Parameter: Specifies, for BSC and BSCT only, the number of seconds the control unit will wait while receiving WACK (wait before transmit positive acknowledgment) or TTD (temporary text delay) sequences from the remote device before time-out occurs.

*SAME: The time interval the control unit will wait is not to be changed.

number-of-seconds: The number of seconds the control unit will wait before time-out occurs.

PGMDLY Parameter: Specifies, for BSC and BSCT only, the number of seconds the control unit will continue sending delay signals to the remote device because of delays in issuing READ or WRITE requests.

*SAME: The time interval for sending delay signals is not to be changed.

number-of-seconds: The number of seconds the control unit will continue to send delay signals before time-out occurs.

RJE Parameter: Specifies, for BSC only, whether this control unit description is to be used by the Remote Job Entry Facility (RJEF).

*SAME: The value specified in the control unit description is not to be changed.

**NO:* This control unit description is not to be used by RJEF.

**YES:* This control unit description is to be used by RJEF.

RJEHOST Parameter: Specifies, for BSC only, the subsystem type of the host to which RJEF is connected.

*SAME: The value specified in the control unit description is not to be changed.

**NONE:* No RJEF host subsystem type is to be specified.

**RES:* RJEF is connected to a VS1/RES subsystem.

**JES2:* RJEF is connected to a VS2/JES2 subsystem.

**JES3:* RJEF is connected to a VS2/JES3 subsystem.

**RSCS:* RJEF is connected to a VM/370 RSCS subsystem.

RJELOGON Parameter: Specifies, for BSC only, logon information for the RJEF host system.

*SAME: The logon information specified in the control unit description is not to be changed.

**NONE:* No logon information is to be specified; the control unit is not to be used for RJEF.

'RJE-host-signon/logon': Enter up to 80 characters of text enclosed in apostrophes to be used as signon/logon information for the RJEF host system.

TEXT Parameter: Specifies the user-defined text that describes the control unit description. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

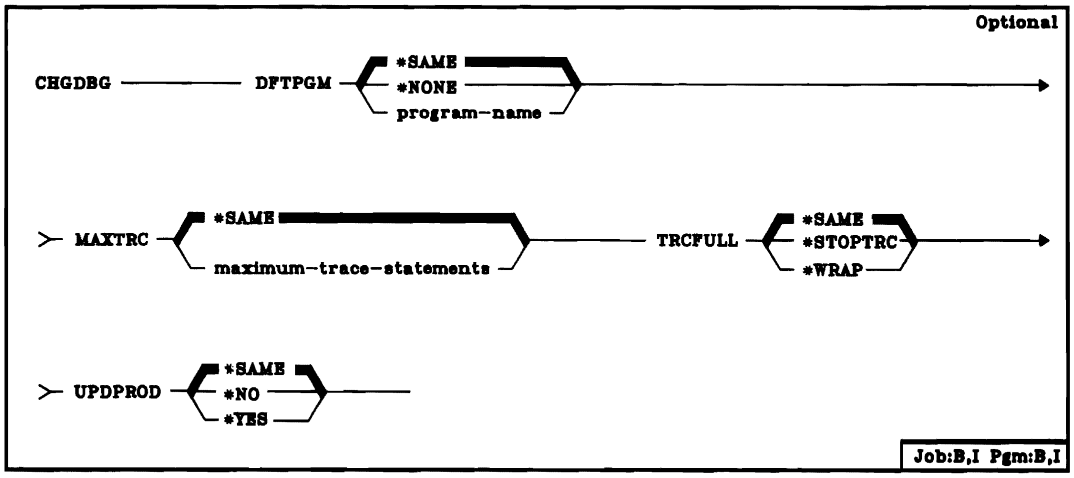
```
CHGCUD  CUD(CONTROL01) TELNBR(nnnnnnnnnn) +
      LINLST(LINE01)
```

This command changes the control unit description of the control unit named CONTRL01. The line list now contains the line name LINE01, and the telephone number is changed to the number represented here by the letter *n* (nnn-nnn-nnnn). Because the line list is always changed from the beginning of the list, LINE01 replaced whatever line name was the first name in the list.

CHGDBG (Change Debug Mode) Command

The Change Debug (CHGDBG) command changes the attributes of the debugging environment currently in effect for a job. All of the attributes can be changed, except which programs are to be debugged. Use the ADDPGM or RMVPGM commands to add or remove a program from debug mode.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*



DFTPGM Parameter: Specifies the name of the program that is to be the default program in the job's debugging environment. The program specified here can be used as the default program for any of the other debug commands that specify *DFTPGM on their PGM parameter.

*SAME: The same program, if any, currently specified as the default program is to be used.

**NONE:* No program is to be specified as the default program. Either the default program must be named later in the ADDPGM command or another CHGDBG command, or *DFTPGM cannot be the specified value (or taken as the default) on any of the other debug commands.

program-name: Enter the simple name of the program that is to be the default program for the job's debugging environment. The same name (in qualified form) must already have been specified in the PGM parameter of the ENTDBG or ADDPGM command.

MAXTRC Parameter: Specifies the maximum number of trace statements that the system is to put into the job's trace file before either terminating tracing or wrapping around (overlaying) on the trace file. When the trace file contains the maximum specified, the system performs the actions specified in the TRCFULL parameter.

*SAME: The maximum for the number of trace statements in the file is not to be changed.

maximum-trace-statements: Enter the maximum number of trace statements that can be in the trace file.

TRCFULL Parameter: Specifies what is to happen when the job's trace file is full (that is, it contains the maximum number of trace statements specified by the MAXTRC parameter).

*SAME: The action to be taken when the trace file is full is not to be changed.

**STOPTRC:* In batch mode, tracing stops but the program continues to execute. In interactive mode, a breakpoint occurs on the next trace statement encountered, and control is given to the user.

**WRAP:* The trace file is overlaid with new trace statements as they occur, wrapping from the beginning of the file. The program continues to execute until finished with no message to indicate that wrapping has occurred. The trace file will never have more than the maximum specified statements, and they will be the more recently recorded statements.

UPDPROD Parameter: Specifies whether or not data base files in a production library can be opened for changes (that is, for adding, deleting, or updating records in the file) while the job is in debug mode. If not, the files must be copied into a test library before an attempt is made to execute a program that uses the files.

*SAME: The previously specified value for this parameter is not to be changed.

**NO:* Data base files in production libraries cannot be changed in debug mode. However, a data base file can be opened for reading only.

**YES:* Data base files in production libraries can be changed while the job is in debug mode.

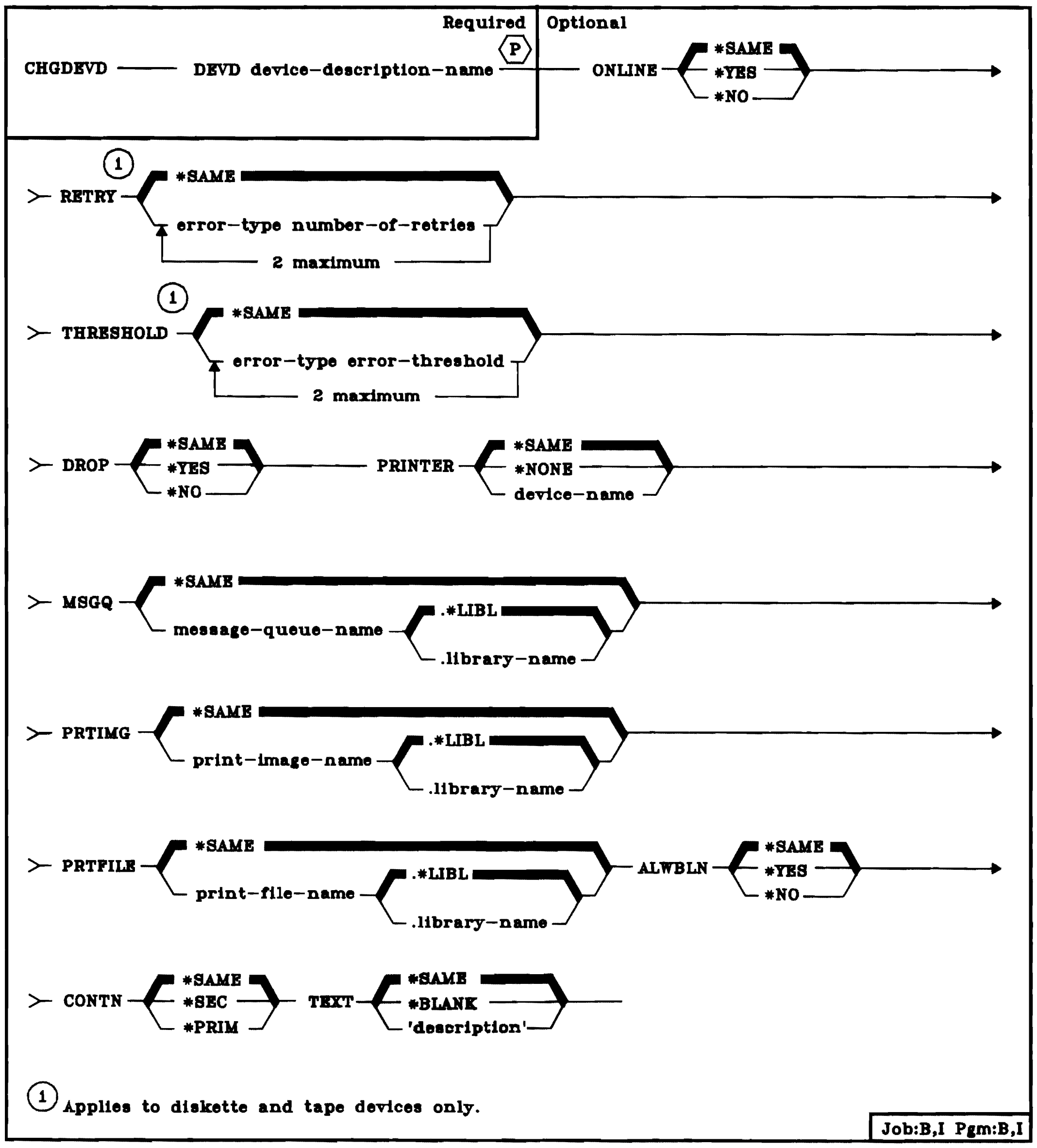
Example

```
CHGDBG  MAXTRC(400) TRCFULL(*STOPTRC)
```

This command changes the maximum number of trace statements that can be put in the trace file to 400. The tracing is to be terminated when the file is full.

CHGDEVD (Change Device Description) Command

The Change Device Description (CHGDEVD) command changes some of the attributes in the device description of the specified device. The device attributes can be changed at any time, regardless of whether the device is online or offline. With the exception of parameter PRTIMG, the device attributes become effective immediately. The attribute specified for PRTIMG becomes effective when the system printer is next used.



DEVD Parameter: Specifies the name of the device description that is to have one or more of its attributes changed. The system console name, QCONSOLE, cannot be specified in this parameter, because its description cannot be changed.

ONLINE Parameter: Specifies whether this device is to be varied online automatically when the Control Program Facility (CPF) is started. After CPF is started, the VRYDEV (Vary Device) command can be used to modify the status of the device.

<u>*SAME:</u> The value specified in the device description is not to be changed.

**YES:* The device is to be online when CPF is started.

**NO:* The device is to be offline when CPF is started. The VRYDEV command must be used to put the device online, making it operational.

RETRY Parameter: Specifies, for diskette and tape data errors only, the number of times the system should attempt to recover from a data error when data is read or written. The system operator is notified if the device cannot recover from the data error in the specified number of retries.

If a retry value is to be specified, both the error type and retry values must be specified. The range of valid values is shown in the following chart:

Error Type	Applicable Device	Number of Retries	Error Threshold
1 – Read error	Diskette	40-80	1-100
	Tape	10-20	1-10
2 – Write error	Tape	15-30	1-64

<u>*SAME:</u> The number of retries is not to be changed.

error-type number-of-retries: Enter the type code followed by the maximum number of retries that the system can have to recover from the specified device data error.

THRESHOLD Parameter: Specifies, for diskette and tape data errors only, the error threshold values that are used to determine when an entry for an error type is to be entered in the error log. The first occurrence of the error type is always logged automatically. This parameter is used to specify the number of errors that can occur before an error is logged again.

*SAME: The values specified in the device description are not changed.

error-type error-threshold: Enter the error type code followed by a valid error threshold value, after which the same error message is to be repeated in the system error log. The values that are valid for each error type are shown in the RETRY parameter chart. Both values must be entered for each type of data error being specified.

DROP Parameter: Specifies, only for 5251 and 5252 devices attached to a control unit that is on a switched line, whether the line is to be disconnected by the system when all work stations on the line are no longer being used. When multiple work stations are attached to the same control unit, the line is disconnected only if: (1) the device description for this device specifies DROP(*YES) or DROP(*YES) is specified on the SIGNOFF command when the user signs off at the device; (2) all of the other display stations connected to the control unit have signed off and are not in use; and (3) all 5224/5225/5256 Printers attached to the control unit are not in use.

The value specified in the device description can be overridden by a user signing off at the device if he specifies the DROP parameter on the SIGNOFF command.

*SAME: The value specified in the device description is not to be changed.

*YES: The switched line to the control unit to which this device is attached is to be disconnected when this device and all the other attached devices are no longer in use.

*NO: The switched line is not to be disconnected from the control unit when all of its attached devices are no longer in use.

PRINTER Parameter: This parameter is valid only to change the device description of a 5251 or 5252 Display Station. It specifies the device name of the 5224/5225/5256 Printer to be associated with the display station. (The printer and the display station must be attached to the same control unit.) The device description of the printer named in this parameter must have already been created in a CRTDEVD command and must currently exist on the system.

Note: A printer attached to a remote work station must have the Expanded Function feature to support this parameter.

*SAME: The same 5224/5225/5256 Printer, if any, is to be associated with this display station.

**NONE:* No 5224/5225/5256 Printer is to be associated with this display station.

device-name: Enter the name of the 5224/5225/5256 Printer (that is, the same name as specified in the device description created for this printer) to be associated with this display station. Both the printer and the display must be attached to the same control unit.

MSGQ Parameter: Specifies, for 5224/5225/5256 Printers only, the message queue to which operational messages for this device are to be sent.

*SAME: The message queue specified in the device description is not to be changed.

qualified-message-queue-name: Enter the qualified name of the message queue to which operational messages are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

PRTIMG Parameter: Specifies, for a system printer device description only, the name of the print image that is to be the standard print image for the 3203, 3262, or 5211 Printer.

*SAME: The print image specified in the device description is not to be changed.

qualified-print-image-name: Enter the qualified name of the print image for the printer. (If a library qualifier is not given, *LIBL is used to find the print image.)

PRTFILE Parameter: Specifies an alternate printer to use when no associated work station printer exists, or when an error occurs during an attempt to use the work station printer.

*SAME: The printer device file specified in the device description is not to be changed.

qualified-print-file-name: Enter the name of the printer device file that is to perform default system printing. (If no library qualifier is given, *LIBL is used to find the device file.)

ALWBLN Parameter: Allows users to suppress the (software-controlled) blinking cursor.

*SAME: The value in the ALWBLN parameter is not to be changed.

**YES:* Allows the cursor to blink for 5250 display devices.

**NO:* The blinking cursor is to be suppressed.

CONTN Parameter: Specifies which BSC station is primary and which is secondary, in order to resolve contention for BSC point-to-point and multipoint lines.

**SAME:* The assignment of rank to the BSC stations is not to be changed.

**SEC:* Specifies that the local System/38 is the secondary station and will yield to the other station when line contention occurs.

**PRIM:* Specifies that the local System/38 is the primary station.

TEXT Parameter: Specifies the user-defined text that describes the device description. (For an expanded description of the TEXT parameter, see Appendix A.)

**SAME:* The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

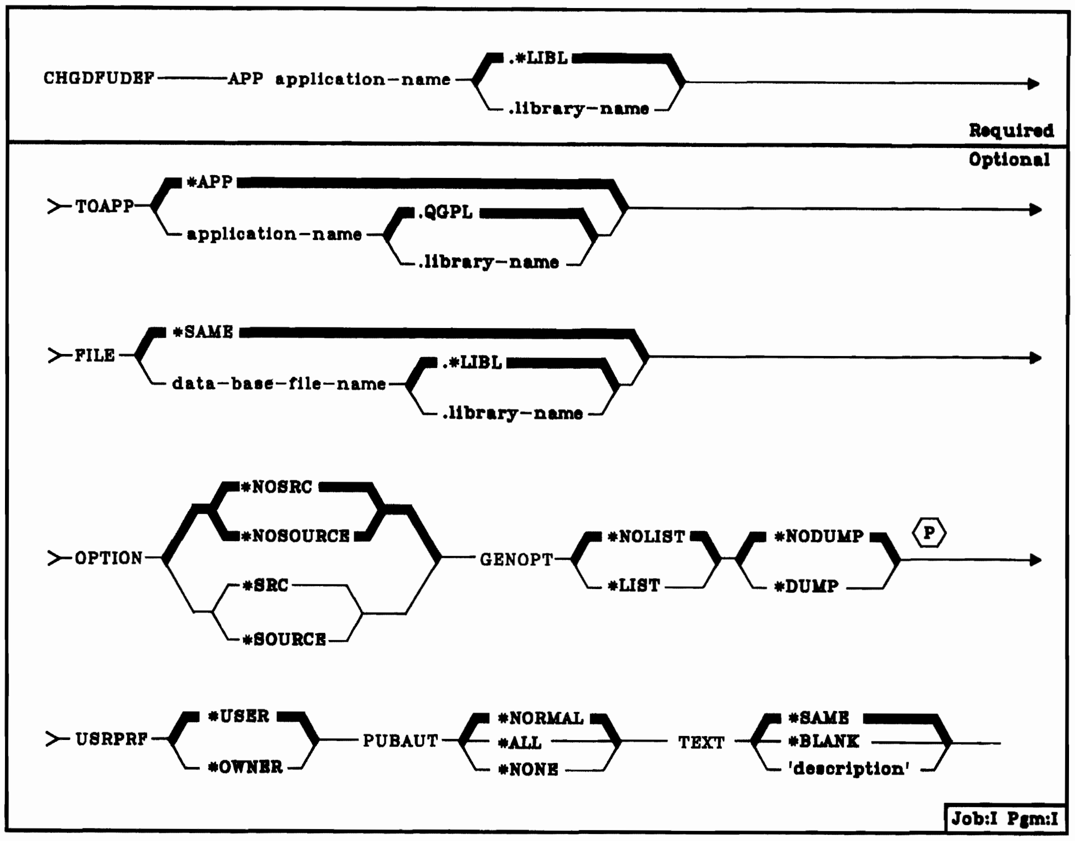
```
CHGDEVD  DEVD(DISP01) PRINTER(PRINTMASK1)
```

This command changes the device description of the display station named DISP01 to include a printer named PRINTMASK1.

CHGDFUDEF (Change DFU Definition) Command

The Change DFU Definition (CHGDFUDEF) command begins a prompting sequence for interactive modification of a DFU application. Your responses to the prompts are used to create a new application or to replace the original application.

The Data File Utility is part of the IBM System/38 Interactive Data Base Utilities Program Licensed Program Product, Program 5714-UT1. For more information on the Data File Utility, refer to the *IBM System/38 DFU Utility Reference Manual and User's Guide,* SC21-7714.



APP Parameter: Specifies the qualified name of the application being changed. (If no library name is given, *LIBL is used to find the application.)

TOAPP Parameter: Specifies the qualified name of the application in which the changed application is to be stored.

<u>*APP:</u> Specifies that the original application is to be replaced by the changed application.

application-name: Enter the name of the application in which the changed application is to be stored. The application definition specified in the APP parameter will remain as originally defined, and can be executed as originally defined. (If no library name is given, the new application is stored in the general-purpose library, QGPL.)

FILE Parameter: Specifies the name of an existing data base file with record formats that will be referred to by the application you are changing. The file is defined by DDS (see the *CPF Reference Manual—DDS*). The file contains record formats that will be referred to by the application you are changing.

<u>*SAME:</u> The data base file specified in the original application definition is to be used.

data-base-file-name: Specify the name of an existing data base file to be referred to during execution of the application. (If no library qualifier is specified, *LIBL is used to find the file.)

OPTION Parameter: Specifies whether a listing of the UDS (utility definition source) statements is to be printed, which may be helpful if problems occur.

<u>*NOSRC</u> or <u>*NOSOURCE:</u> Specifies that DFU is not to print a listing of the UDS. The *NOSRC and *NOSOURCE values are equivalent.

**SRC* or **SOURCE:* Specifies that DFU is to print a listing of the UDS. The *SRC and *SOURCE values are equivalent.

GENOPT Parameter: Specifies whether the IDU program listings for your application are to be produced. These listings may be helpful if a problem occurs.

<u>*NOLIST:</u> Specifies that an internal representation of the application program is not to be printed.

**LIST:* Specifies that an internal representation of the application program is to be printed.

<u>*NODUMP:</u> Specifies that the application program template is not to be printed.

**DUMP:* Specifies that the application program template is to be printed. *DUMP will provide the template only if *LIST has been specified.

USRPRF Parameter: Specifies a user profile under which the application is to be executed. This parameter allows a programmer to define a DFU application for someone who does not have full authority over the data base file that the application reads.

*USER: The user profile of the application user is in effect when the application is executed.

OWNER: The user profiles of both the application owner and the application user are in effect when the application is executed.

When you create or change an application that is to be used by someone else, you must authorize the user for the use of the application and any objects associated with the application. You can grant each user specific rights to such objects, or by specifying USRPRF(*OWNER) when an application is created or changed, you can permit a user to temporarily assume your authority to use objects associated with the application.

PUBAUT Parameter: Specifies what authority over the application is extended to all system users. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: All system users can execute or read the application, but not all users can delete the application.

*ALL: All system users have complete authority over the application.

*NONE: All users but the owner are restricted from using the application. Of course, the owner can grant rights to other users.

TEXT Parameter: Specifies a brief description of the changed application.

*SAME: The description of the application is to remain as originally defined.

*BLANK: There is to be no description of this application.

'description': Enter no more than 50 characters, enclosed in apostrophes, to describe the changed application.

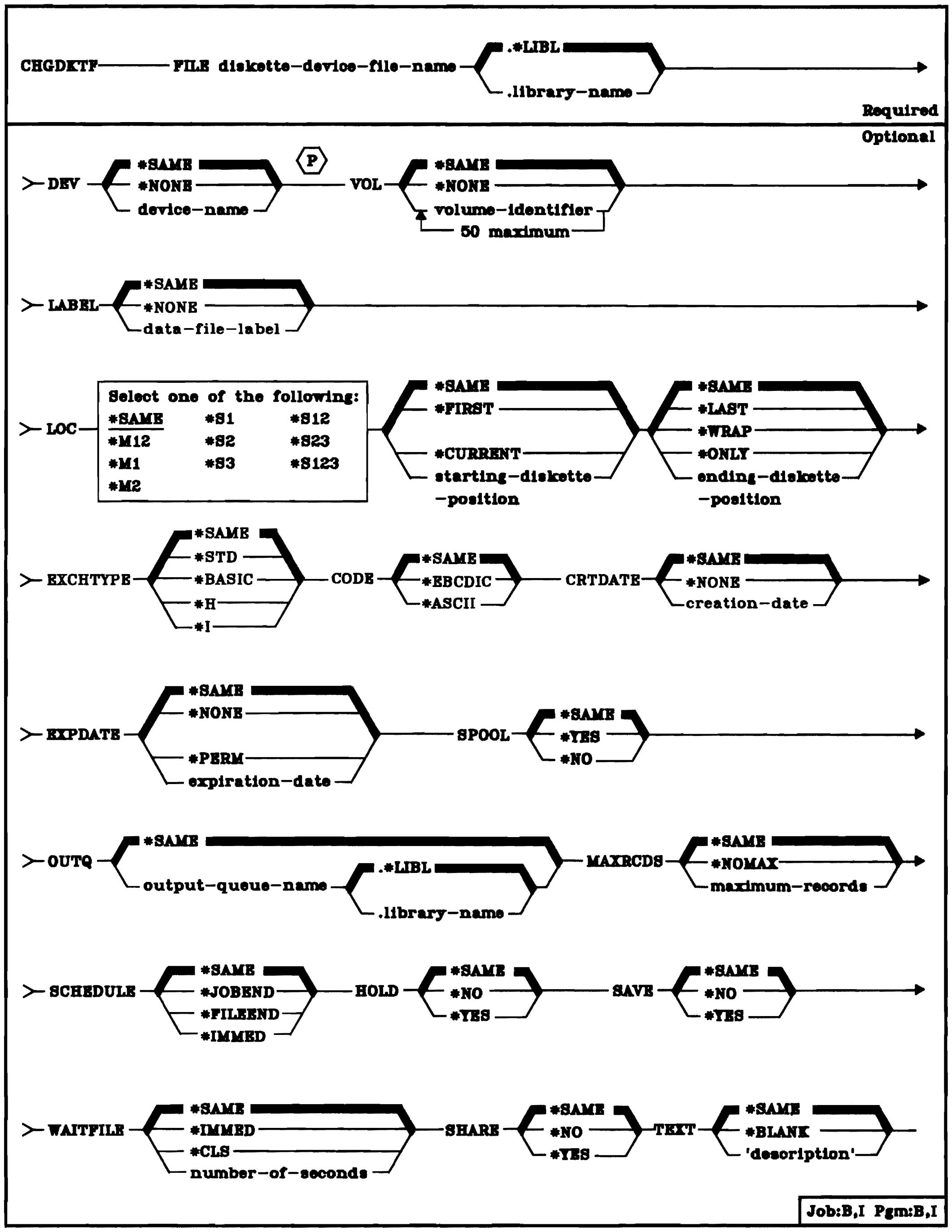
Example

```
CHGDFUDEF  APP(TEST1) TOAPP(TEST2) +
     TEXT('Create application for TEST2, based on TEST1')
```

This command begins a prompting sequence which allows you to create an application named TEST2 in library QGPL based on application TEST1 in your library list. Your responses to the prompts can result in changes to the TEST2 application attributes (which differ from the based-on application TEST1). Application TEST1 is not changed in any way. Application TEST2 uses data from the data base file specified for application TEST1. No UDS or internal representations of application TEST2 will be printed. Any system users can execute or read TEST2, but only the owner of the application can delete it.

CHGDKTF (Change Diskette File) Command

The Change Diskette File (CHGDKTF) command changes, in the file description, one or more of the attributes of the specified diskette device file.



FILE Parameter: Specifies the qualified name of the diskette device file whose description is being changed. (If no library qualifier is given, *LIBL is used to find the file.)

DEV Parameter: Specifies the name of the diskette device that is to be used with this device file to perform I/O data operations. The device name of the IBM-supplied diskette device description is QDKT.

SAME: The device name, if any, specified in the device file description remains the same.

**NONE:* No device name is to be specified. It can be specified later on an OVRDKTF command or when the diskette device file is opened.

device-name: Enter the name of the device that is to be used with this diskette device file. The device must already be known on the system via a device description.

VOL Parameter: Specifies one or more volume identifiers of diskettes (either in magazines or slots) to be used by the diskette device file. The diskettes (volumes) must be mounted on the device in the same order as the identifiers are specified here. The identifiers are matched, one by one, with the diskette locations specified in the LOC parameter. (For an expanded description of the VOL parameter, see Appendix A.)

SAME: The volume identifiers specified in the device file description remain the same.

**NONE:* No diskette volume identifiers are specified. They can be supplied before the device file is opened, either in the OVRDKTF (or another CHGDKTF) command or in the HLL program. If not specified, no volume identifier checking is performed.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used by this device file. Each identifier can have 6 alphameric characters or fewer.

LABEL Parameter: Specifies the data file label of the data file on diskette that is to be used with this diskette device file. For input files (diskette input to system), this label specifies the identifier of the file that exists on the diskette. For output files (system output to diskette), it specifies the identifier of the file that is to be created on the diskette. (For an expanded description of the LABEL parameter, see Appendix A.)

*SAME: The data file label specified in the device file description is not to be changed.

**NONE:* No data file label is to be specified. It must be supplied before the device file is opened, either in the OVRDKTF (or another CHGDKTF) command or in the HLL program.

data-file-label: Enter the identifier (8 characters maximum) of the data file to be used with this diskette device file. (See Appendix A for details.)

LOC Parameter: Specifies which diskette location(s) in the magazines or slots are to be used by this diskette device file. Three values are needed: (1) the unit type and location, (2) the starting diskette position, and (3) the ending diskette position in the unit. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit and location on the diskette magazine drive are to be used by the device file for diskette input/output. Enter one of the following values for the unit type and location (the valid starting and ending positions for each unit type are also listed):

Unit Type/Location	**Diskette Starting and Ending Position**
*M12	1 through 10
*M1	1 through 10
*M2	1 through 10
*S1	1
*S2	2
*S3	3
*S12	1 through 2
*S23	2 through 3
*S123	1 through 3

*SAME: The unit location specified in the device file description that is to be used with this device file remains the same.

location: Enter one of the following values to specify the unit type and location on the diskette magazine drive to be used with this device file: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123. (See Appendix A for their meanings.)

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette used first by the device file. Enter one of the following values to specify the starting diskette positions:

*SAME: The same starting diskette position specified in the device file description is to be used.

**FIRST:* The first diskette position in the location contains the diskette to be used first in the read or write operation. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used.

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine or manual slot that contains the first diskette to be used.

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette used last by the device file. Enter one of the following values to specify the ending diskette position:

*SAME: The same ending diskette position specified in the device file description is to be used.

**LAST:* The last diskette position in the location contains the diskette to be used last in the read or write operation. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**WRAP:* If the end of the last diskette in the location is reached before the end of the data file is reached, a message is sent to the system operator to mount another magazine or diskette to continue. (See Appendix A for details and restrictions on using *WRAP.)

**ONLY:* Only the diskette position specified by the second value is to be used, and used only once.

ending-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine or manual slot that contains the last diskette to be used.

EXCHTYPE Parameter: Specifies, for diskette output files only, the exchange type to be used by the device file when the system is writing diskette data. (For an expanded description of the EXCHTYPE parameter, refer to Appendix A.)

*SAME: The exchange type specified in the device file description is not to be changed.

**STD:* The basic exchange format will be used for a type 1 or a type 2 diskette. The H exchange type will be used for a type 2D diskette.

**BASIC:* The basic exchange type will be used.

**H:* The H exchange type will be used.

**I:* The I exchange type will be used.

CODE Parameter: Specifies the type of character code to be used when diskette data is read or written by a job that uses this device file.

*SAME: The type of character code specified in the device file description is not to be changed.

**EBCDIC:* The EBCDIC character code is to be used with this device file.

**ASCII:* The ASCII character code is to be used with this device file.

CRTDATE Parameter: Specifies when the diskette data file was created on diskette. The creation date parameter is valid for input data files only. If the creation date written on the diskette does not match the date specified for the device file when it is opened, an error message is sent to the user program.

*SAME: The creation date of the diskette data file specified in the device file description remains the same.

**NONE:* The creation date of the diskette data file is not to be checked.

creation-date: Enter the creation date of the diskette data file to be used by this device file. The date must be specified in the format defined by the system values QDATFMT and QDATSEP. However, the specified date is put in the diskette label as *yymmdd.*

EXPDATE Parameter: Specifies the expiration date of the diskette data file used by this device file. The data file is protected and cannot be written over until the day after the specified expiration date.

*SAME: The expiration date of the data file specified in the device file description remains the same.

**NONE:* The data file is protected for only one day, the day it is created on the diskette.

**PERM:* The data file is to be protected permanently. The date written on the diskette is 999999.

expiration-date: Enter the date after which the data file expires. The date must be specified in the format defined by the system values QDATFMT and QDATSEP. However, the specified date is put in the diskette label as *yymmdd*.

SPOOL Parameter: Specifies whether the input or output data for the diskette device file is to be spooled. If SPOOL(*NO) is specified, the following parameters in this command are ignored: OUTQ, MAXRCDS, SCHEDULE, HOLD, and SAVE.

*SAME: The value specified in the device file description is not to be changed.

**YES:* The data is to be spooled. If this file is opened for input, an inline data file having the specified name is processed; otherwise, the next unnamed inline spooled file is processed. (For a discussion of named and unnamed inline files, see the *CPF Programmer's Guide*.) If this is an output file, the data is spooled for processing by a card, diskette, or print writer.

**NO:* The data is not to be spooled. If this file is opened for input, the data is read directly from the diskette. If this is an output file, the data is written directly to the diskette as it is processed by the program.

OUTQ Parameter: Specifies, for spooled output only, the name of the output queue for the spooled output file.

*SAME: The same output queue specified in the device file description is to be used.

qualified-output-queue-name: Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.) The IBM-supplied output queue that can be used by the diskette file is the QDKT output queue, stored in the QGPL library.

MAXRCDS Parameter: Specifies the maximum number of records that can be in the spooled output file for this diskette device file.

*SAME: The maximum number of records specified in the device file description remains the same.

**NOMAX:* No maximum is specified for the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of diskette records that can be in the spooled output file.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a writer.

*SAME: The time specified in the device file description when spooled output can begin remains the same.

**JOBEND:* The spooled output file is to be made available to the writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the writer as soon as the file is opened in the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The spooled file is made available to a writer when it is released by the Release Spooled File (RLSSPLF) command.

*SAME: The value specified in the device file description is not to be changed.

**NO:* The spooled output file is not to be held by the output queue. The spooled output is made available to a writer based on the SCHEDULE parameter value.

**YES:* The spooled output file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled file is to be saved (left on the output queue) after the output has been produced.

*SAME: The value specified in the device file description is not to be changed.

**NO:* The spooled file data is not to be retained on the output queue after it has been produced.

**YES:* The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*SAME: The wait time specified in the device file description is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the diskette device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

*SAME: The value specified in the device file description is not to be changed.

**NO:* An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Specifies the user-defined text that describes the diskette device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

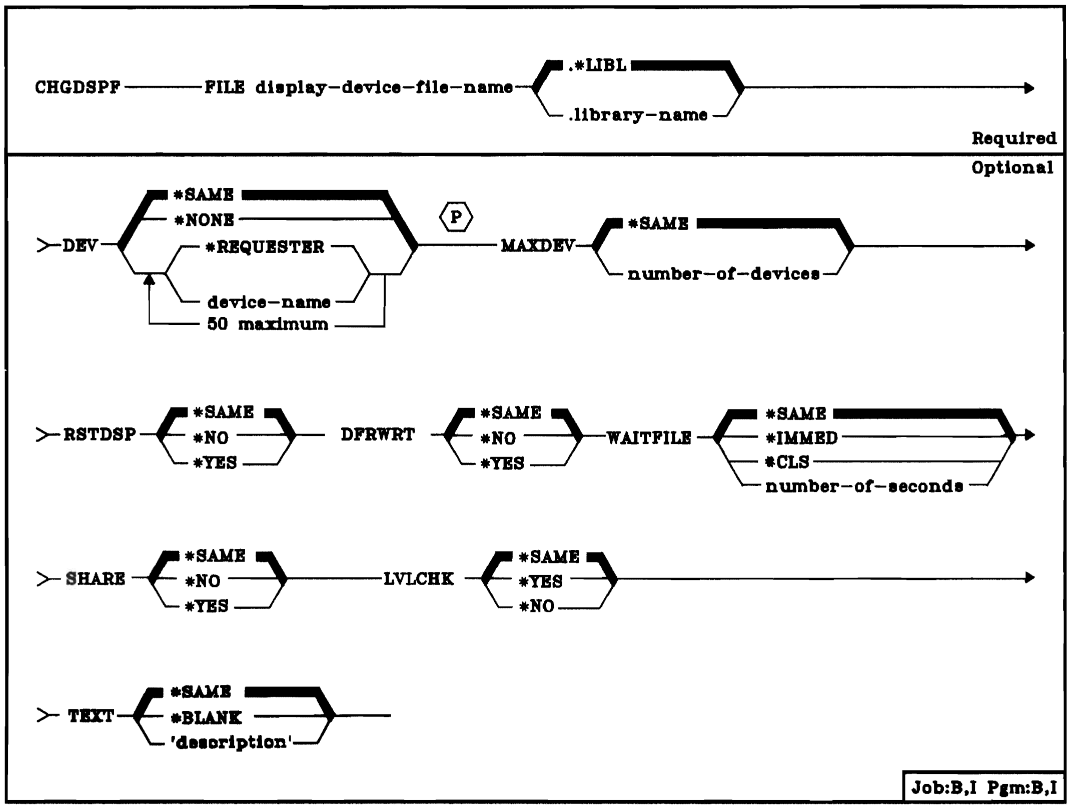
Example

```
CHGDKTF  FILE(PRNTRPT.ACCREC) SPOOL(*NO)
```

This command changes the description of the diskette device file named PRNTRPT stored in the ACCREC library. The device file now causes all I/O operations between the program and the diskette to be direct (without spooling). All the other values in the file description are not changed.

CHGDSPF (Change Display File) Command

The Change Display File (CHGDSPF) command changes, in the file description, one or more of the attributes of the specified display device file.



FILE Parameter: Specifies the qualified name of the display device file whose description is being changed. (If no library qualifier is given, *LIBL is used to find the file.)

DEV Parameter: Specifies the names of one or more display devices that are to be used with this display device file to pass data records between the users of the display devices and their jobs.

SAME: The device names specified in the device file description are not changed.

**NONE:* No device name is to be specified. It can be specified later on an OVRDSPF command, another CHGDSPF command, or in the HLL program that opens the file.

**REQUESTER:* The device that requests the program that uses this device file is the device that is assigned to the file.

device-name: Enter the names of one or more display devices that are to be used with this device file to pass data records between the users of the devices and the system. Each device name must already be known on the system via a device description. *REQUESTER can be specified as one of the names.

The list of names specified here replaces the previous list, if any, contained in the file description. A maximum of 50 device names (including *REQUESTER, if it is specified) can be specified, but the total number cannot exceed the number specified in the MAXDEV parameter when the file is opened.

MAXDEV Parameter: Specifies the maximum number of display devices that can be connected to the display device file at the same time, while the file is open. The names of the devices can be specified in the DEV parameter of this command, in an OVRDSPF command, or in the HLL program that opens the file.

<u>*SAME</u>: The maximum number of display devices specified in the device file description remains the same.

number-of-devices: Enter a value, 1 through 255, that specifies the maximum number of devices that can be connected to this display file at the same time.

RSTDSP Parameter: Specifies whether data being displayed at a display device by this display file is to be saved at the time the file is suspended (temporarily inactive) so that a different display file can be used to display different data on the same device. If the data for this file is saved, it is restored to the screen of the device when the file is used again.

This parameter must be considered if, within the same routing step, any program can be called that uses a different display file for the same device. If *all* programs that use this file always display new data when control is returned to them, the display data for this file need not be saved for any of them; RSTDSP(*NO) can be specified or assumed. If *any* program using this file requires that the contents of the screen be exactly the same as it was before it called another program, RSTDSP(*YES) must be specified. If certain display fields are to remain unchanged while others are erased or rewritten, or if the program containing the file can be interrupted (for messages to be displayed, for example), you should specify RSTDSP(*YES). (For additional information about suspended display files, see the *CPF Programmer's Guide.*)

<u>*SAME</u>: The value specified in the device file description is not to be changed.

**NO:* The data being displayed by this file is not to be saved when the file is suspended. None of the programs using this file need the data restored when control is returned to them.

**YES:* The data being displayed when the file is suspended is to be saved so it can be restored to the screen of the device when the file is used again.

DFRWRT Parameter: Specifies that the writing of data is to be deferred until it can be written out with other data when a read request is made. Control is returned to the program immediately after the data is received. This may result in improved performance.

<u>*SAME:</u> The value specified in the device file description is not to be changed.

**NO:* After a write operation, the user program does not regain control until the I/O is completed (with the data displayed and the I/O feedback information available).

**YES:* When the program issues a write request, control is returned after the buffer is processed. The data might not be displayed immediately; the actual display of the data might take place later when a read or combined read/write operation is performed. The buffer is then available to be prepared for the next read or combined read/write operation.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources (including at least one of the display devices) cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

<u>*SAME:</u> The wait time specified in the device file description is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the display device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

<u>*SAME:</u> The value specified in the device file description is not to be changed.

**NO:* An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check, (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given an internal system identifier when the format is created.

*SAME: The value specified in the device file description is not to be changed.

**YES:* The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not all match, an error message is sent to the program requesting the open.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

TEXT Parameter: Specifies the user-defined text that describes the display device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CHGDSPF  FILE(ORDENT)  DEV(WS1 WS2 WS3)  MAXDEV(3)
```

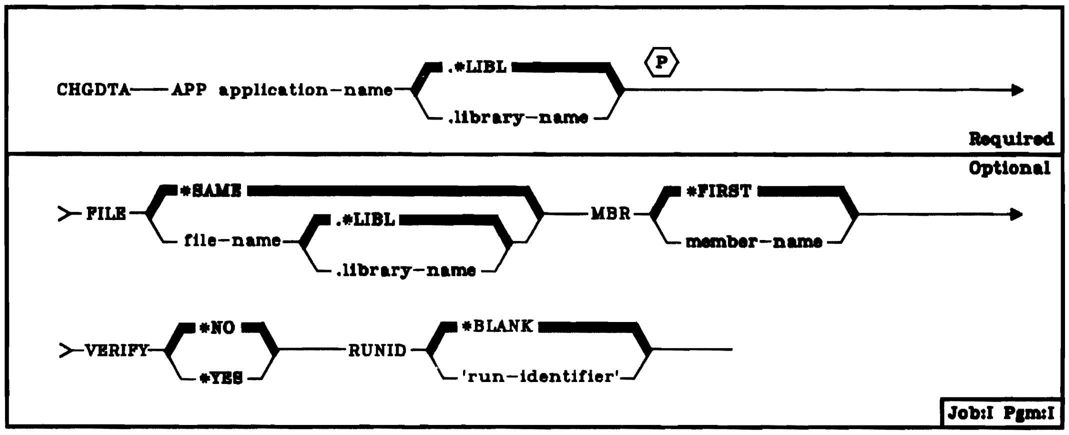
This command changes the description of the display device file named ORDENT. The file is located through the library list. The devices to be used with this file are the work stations WS1, WS2, and WS3. All three of the devices can be used concurrently with this display file.

CHGDTA (Change Data) Command

The Change Data (CHGDTA) command allows you to add, change, delete, or display records in an existing data base file.

The Data File Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Data File Utility, refer to the *IBM System/38 DFU Reference Manual and User's Guide,* SC21-7714.

Note: The first member of the file named when you defined the application is processed unless you specify a different member.



APP Parameter: Specifies the qualified name of the DFU application controlling the interactive update of data. (If no library qualifier is specified, *LIBL is used to find the application.)

FILE Parameter: Specifies the name of the data base file you want to process.

*SAME: DFU will use the same file used to define the application.

file-name: Enter the qualified name of the data file you want DFU to process. The file should have at least one record format name in common with the file used to define the application. (If no library name is specified, *LIBL is used to find the file.)

MBR Parameter: Specifies which member in the file you want to process.

*FIRST: DFU will process the first member of the file.

member-name: Enter the name of the member you want DFU to process.

VERIFY Parameter: Indicates whether the updates are intended to verify the contents of existing data records.

<u>*NO:</u> Adds, changes, or deletes are not to be compared to existing data.

**YES:* Data being reentered is to be compared with previously entered data. Discrepancies are highlighted with reverse image characters on the display screen.

RUNID Parameter: Specifies a character string of eight characters or less that can be used to set an initial value in each data base record added during a given processing session. Either an alphameric field must be defined in the DFU application with an initial value, or *RUNID must be specified.

<u>*BLANK:</u> No run identifier is to be specified.

'run-identifier': Enter a character string to identify the records added during this session.

Example

```
CHGDTA  APP(DATA.LIB1)  FILE(FILEA)  RUNID('NEWSALES')
```

This command uses the application named DATA in library LIB1 to process the file named FILEA. Every record added will be identified by the characters NEWSALES.

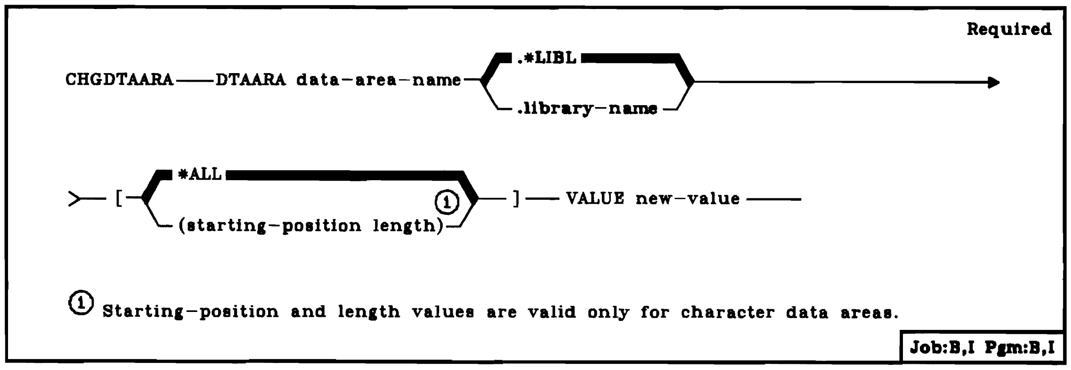
CHGDTAARA (Change Data Area) Command

The Change Data Area (CHGDTAARA) command changes the value of the specified data area that is stored in a library. This command does not change the data attributes nor any of the object attributes of the data area. The new value must have the same type and a length less than or equal to the data area length or the specified substring length.

For character data areas, a substring of the data area may be changed without affecting the rest of the data area. This substring is defined by specifying the starting position and the length of the substring. In this case, the new value must have a length less than or equal to the substring length.

When the CHGDTAARA command is executed, the data area is locked to the program during the change operation so that commands in other jobs cannot change or destroy it until the operation is completed. If the data area is shared with other jobs and it is updated in steps involving more than one command in a job, the data area should be explicitly allocated to that job until all the steps have been performed. The data area can be explicitly allocated with the ALCOBJ command.

Restriction: To use this command, you must have operational and update rights for the data area being changed and read rights for the library in which it is stored.


Required
CHGDTAARA
DTAARA data-area-name
.*LIBL
.library-name
[
*ALL
(starting-position length)
①
]
VALUE new-value
① Starting-position and length values are valid only for character data areas.
Job:B,I Pgm:B,I

DTAARA Parameter: Specifies the qualified name of the data area whose value is to be changed. (If no library qualifier is given, *LIBL is used to find the data area.) Optionally specifies, for character data areas only, the starting position and length of the character string that is to be changed in the data area.

<u>*ALL:</u> The entire data area is to be changed. The length, if specified, must not be less than the length of the VALUE specified.

starting-position length: Enter the starting position and the length of the character string that is to be changed in the data area. Starting position and length must be specified together if used; neither may be specified alone. The beginning and end of this string must be within the data area. If the length is greater than the length specified on the VALUE parameter, padding on the right with blanks will occur.

VALUE Parameter: Specifies the new value to be stored in the data area. Enter a value that is valid for the data attributes specified in the data area's description. If TYPE(*CHAR) or TYPE(*LGL) was specified when the data area was created and the value specified here is numeric, the value must be enclosed in apostrophes. If TYPE(*DEC) was specified, the value must not be enclosed in apostrophes.

Examples

```
CHGDTAARA  DTAARA(MYDATA.MYLIB)  VALUE(GOODNIGHT)
```

This command changes the value of the data area named MYDATA in library MYLIB to GOODNIGHT. The data area must be for character data and must be 9 or more characters long.

```
CHGDTAARA  PAYROLLSW  '0'
```

This command changes the logical value of the data area named PAYROLLSW to zero. The library search list is used to locate the data area.

```
CHGDTAARA  DTAARA(MYDATA.MYLIB  (5 4))  VALUE('TWO')
```

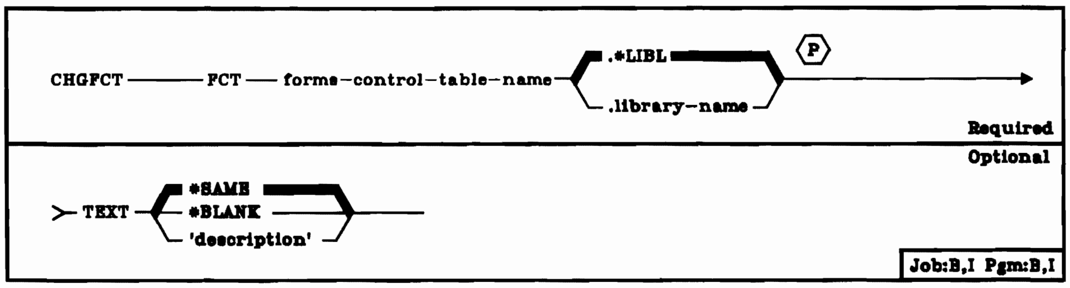
This command changes characters 5 through 8 of the data area MYDATA. Because the new value is shorter than the substring, it will be padded with a blank. If MYDATA is a character data area that previously contained 'ONE TOOOTHREE', MYDATA will now contain 'ONE TWO THREE'.

CHGFCT (Change Forms Control Table) Command

The Change Forms Control Table (CHGFCT) command changes attributes in an existing forms control table (FCT).

Restriction: To use this command, you must have operational rights for the FCT and read rights for the library in which the FCT is stored.

The Change Forms Control Table (CHGFCT) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



FCT Parameter: Specifies the qualified name of the FCT that is to be changed. (If no library qualifier is given, *LIBL is used to find the FCT.)

TEXT Parameter: Lets the user enter text that briefly describes the FCT. (For an expanded description of the TEXT parameter, see Appendix A.)

SAME: The text, if any, is not changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CHGFCT  FCT(FORMCTRL.USERLIB) +
     TEXT('Forms control table number two')
```

This command changes the description of forms control table named FORMCTRL in library USERLIB.

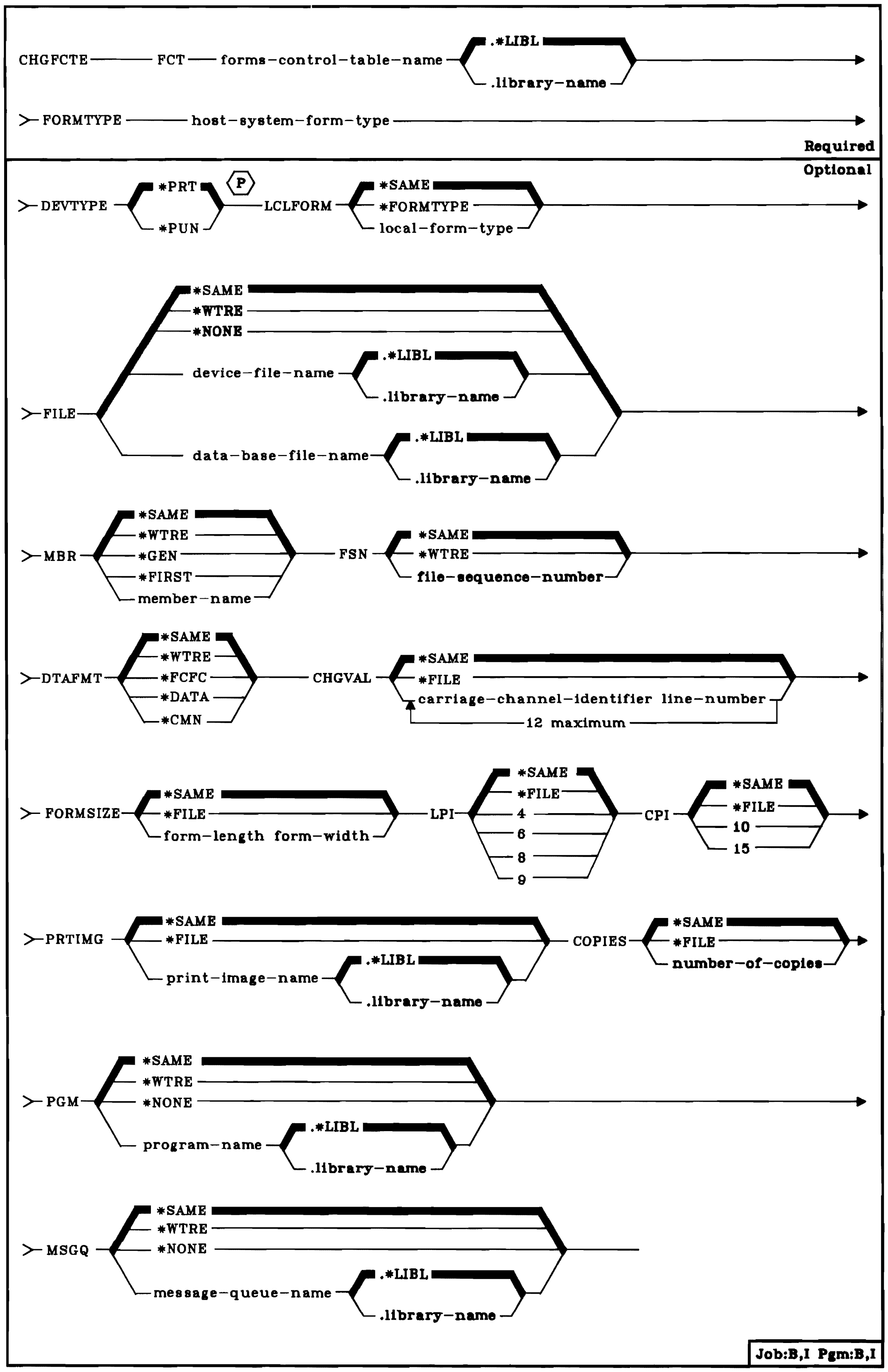
CHGFCTE (Change Forms Control Table Entry) Command

The Change Forms Control Table Entry (CHGFCTE) command changes the attributes in an existing forms control table (FCT) entry.

FCT entries are read by an active RJEF session when an RJEF writer is started and when a forms mount message is received from the host system. If an FCT entry is changed and not ready by the active RJEF session, the change does not affect the processing of host system data. For example, if you were receiving data for forms type xxxx and issued a CHGFCTE command for this forms type, the change would not take effect until the RJEF writer is canceled and restarted or another forms mount message is received from the host system.

Restriction: To use this command, you must have operational rights to the FCT and read rights to the library in which the FCT is stored.

The Change Forms Control Table Entry (CHGFCTE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.


CHGFCTE
FCT
forms-control-table-name
.*LIBL
.library-name
FORMTYPE
host-system-form-type
Required
Optional
DEVTYPE
*PRT
*PUN
P
LCLFORM
*SAME
*FORMTYPE
local-form-type
FILE
*SAME
*WTRE
*NONE
device-file-name
.*LIBL
.library-name
data-base-file-name
.*LIBL
.library-name
MBR
*SAME
*WTRE
*GEN
*FIRST
member-name
FSN
*SAME
*WTRE
file-sequence-number
DTAFMT
*SAME
*WTRE
*FCFC
*DATA
*CMN
CHGVAL
*SAME
*FILE
carriage-channel-identifier line-number
12 maximum
FORMSIZE
*SAME
*FILE
form-length form-width
LPI
*SAME
*FILE
4
6
8
9
CPI
*SAME
*FILE
10
15
PRTIMG
*SAME
*FILE
print-image-name
.*LIBL
.library-name
COPIES
*SAME
*FILE
number-of-copies
PGM
*SAME
*WTRE
*NONE
program-name
.*LIBL
.library-name
MSGQ
*SAME
*WTRE
*NONE
message-queue-name
.*LIBL
.library-name
Job:B,I Pgm:B,I

FCT Parameter: Specifies the qualified name of the forms control table (FCT) in which the entry is to be changed. (If no library qualifier is given, *LIBL is used to find the FCT.)

FORMTYPE Parameter: Specifies the host system form type that is to be associated with the FCT entry. This value (one through eight alphameric characters in length) will be returned by the host system in a forms mount message. A host system form type of blanks can be entered as FORMTYPE('   '). The LCLFORM parameter can be used to change this value to one more understandable to the System/38 user.

DEVTYPE Parameter: Specifies the device type with which the FCT entry is to be associated.

*PRT: This FCT entry can be used only when processing printer output streams.

**PUN:* This FCT entry can be used only when processing punch output streams.

LCLFORM Parameter: Specifies the local form type. This value is to be substituted for the FORMTYPE value used by the host system, to make the forms mount message more understandable to the System/38 user.

*SAME: The local form type to be substituted for the host system form type specified in the FCT entry remains the same.

**FORMTYPE:* No local form type is to be substituted for the host system form type (therefore, the host system form type is to be used).

local-form-type: Enter the name of the local form type to be substituted for the host system form type when the output from the job is actually received. Valid values can be one through ten alphameric characters in length.

FILE Parameter: Specifies the qualified name of the file that is to receive data from the host system.

*SAME: The file name specified in the FCT entry remains the same.

**WTRE:* The file specified in the session description writer entry is to be associated with the FCT entry. (However, if the FILE parameter of the Start RJE Writer (STRRJEWTR) command defaults to the value specified in the RJEF writer entry, that value is used.)

**NONE:* No file is to be associated with the FCT entry. The session description writer entry must be used to determine where the data is to be sent. None of the information in the FCT entry is to be used.

device-file-name: Enter the qualified name of the program-described printer file that is to receive the data. (If no library qualifier is given, *LIBL is used to find the printer file.)

data-base-file-name: Enter the qualified name of the System/38 physical file to receive the data. (If no library qualifier is given, *LIBL is used to find the data base file.)

MBR Parameter: Specifies the data base file member to which the output is to be directed (if a data base file was specified either in the FILE parameter of this command or in the associated session description writer entry).

<u>*SAME</u>: The data base file member name in the session description FCT entry remains the same.

**WTRE:* The data base file member is to be generated according to the method specified in the associated session description writer entry.

**GEN:* RJEF creates a member name as follows:

Afffffccc or Bffffffccc

Where:

A	=	file member names beginning with the character A contain print data.
B	=	file member names beginning with the character B contain punch data.
ffffff	=	first six characters of the forms name specified in the FCT or received from the host system. **Note:** Only characters that are valid in a System/38 name are valid in the forms type used to generate data base file member names.
ccc	=	three-digit sequence value controlled by the RJEF session to maintain member uniqueness (refer also to the FSN parameter description of this command).

If a member with this name already exists in the data base file, the three-digit sequence value is incremented by one and another attempt is made to create a member. Incrementing of the sequence value continues until a unique name is generated and a member is created or until all 1000 possibilities have been exhausted without creating a member. If no member is created, the RJEF operator receives a message indicating the failure and a request to retry or cancel this file.

**FIRST:* The output is to be directed to the first member of the data base file (if a data base file is specified in the FILE parameter of this command or the associated session description writer entry).

member-name: Enter the name of the data base file member to which output is to be directed (if a data base file is specified in the FILE parameter of this command or the associated session description writer entry). If the member does not exist when it is needed, an inquiry message is sent to the RJEF message queue.

FSN Parameter: Specifies the initial three-digit file sequence number to be used when creating data base file member names. This parameter is ignored unless MBR(*GEN) is specified for this command or in the associated session description writer entry.

*SAME: The file sequence number specified in the FCT entry remains the same.

**WTRE:* The initial file sequence number to be used is the same as the number specified in the session description writer entry.

file-sequence-number: Enter the initial three-digit file sequence number to be used. Leading zeros are not required for sequence numbers less than 100.

DTAFMT Parameter: Specifies the format of the output data.

*SAME: The data format designation specified in the FCT entry remains the same.

**WTRE:* The output data is to be in the format specified in the session description writer entry.

**FCFC:* The output data is to be in the FCFC data format, with the first character of every record being the ANSI forms control character. Specify *FCFC if the data is to be printed. If DEVTYPE(*PUN) is specified, *FCFC is not valid.

The data can be written to a data base file in the FCFC data format and then printed later by issuing the Copy File (CPYF) command and specifying an FCFC printer file on the TOFILE parameter.

**DATA:* The output data is to be in the normal data format (that is, no FCFC characters are embedded in the data). Specify *DATA if the data is to go to a data base file and be processed by a program. If the data is directed to a printer file, a single space ANSI control character is the first character in each record.

**CMN:* The output data is to be in the communications data format (that is, still compressed or truncated). *CMN should be used to decrease communications time. However, before the data can be used, the Format RJE Data (FMTRJEDTA) command must be used to change the data to *FCFC or *DATA. If *CMN is specified, the output file must be a data base file with a length of 256.

CHLVAL Parameter: Specifies the printer carriage channel information.

<u>*SAME:</u> The carriage information specified in the FCT entry remains the same.

**FILE:* The carriage information specified in the device file is to be used.

carriage-channel-identifier line-number: Enter the channel identifiers and line numbers to be used.

Each identifier can be specified only once per command invocation. The identifiers are 1 through 12, corresponding to printer channels 1 through 12. Single spacing is used for any channel not associated with a line number.

The maximum valid line number is 255.

The CHLVAL parameter associates the channel identifier with a page line number; for example, CHLVAL((1 5)(10 55)) means to associate channel 1 with line 5 and channel 10 with line 55.

FORMSIZE Parameter: Specifies the form size to be used on the System/38 printer.

<u>*SAME:</u> The form size specified in the FCT entry remains the same.

**FILE:* The form size specified in the device file is to be used.

form-length form-width: Enter the form length and width to be used. The maximum valid form length is 255 and the maximum valid form width is 198.

LPI Parameter: Specifies the number of lines of print per inch to be used on the System/38 printer.

<u>*SAME:</u> The number of lines of print per inch specified in the FCT entry remains the same.

**FILE:* The number of lines of print per inch specified in the device file is to be used.

4: The number of lines of print per inch is 4.

6: The number of lines of print per inch is 6.

8: The number of lines of print per inch is 8.

9: The number of lines of print per inch is 9.

CPI Parameter: Specifies the number of characters per inch to be used on the System/38 printer.

*SAME: The number of characters per inch specified in the FCT entry remains the same.

**FILE:* The number of characters per inch specified in the device file is to be used.

10: The number of characters per inch is 10.

15: The number of characters per inch is 15.

PRTIMG Parameter: Specifies the qualified print image name to be used on the System/38 printer.

*SAME: The print image specified in the FCT entry remains the same.

**FILE:* The print image specified in the device file is to be used.

print-image-name: Enter the qualified name of the print image to be used. (If no library qualifier is given, *LIBL is used to find the print image.)

COPIES Parameter: Specifies the number of copies to be printed. This parameter applies only for spooled files.

*SAME: The number of copies of print or punch output specified in the FCT entry remains the same.

**FILE:* The number of copies specified in the device file is to be used.

number-of-copies: Enter the number of copies to be printed.

PGM Parameter: Specifies the qualified name of a user-supplied program to be used for processing data received from the host system.

*SAME: The user-supplied program name specified in the FCT entry remains the same.

**WTRE:* The associated session description writer entry is to be used.

**NONE:* No user-supplied program is to be used.

program-name: Enter the qualified name of the user-supplied program to be used. (If no library qualifier is given, *LIBL is used to find the user-supplied program.)

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF writer are to be recorded.

Note: Messages for RJEF writers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands. If inquiry messages are issued by RJEF, they are sent to the user message queue (if specified) where they must receive a response.

*SAME: The message queue specified in the FCT entry remains the same.

**WTRE:* The message queue specified in the session description writer entry is to be used.

**NONE:* No user message queue exists on which the messages for the FCT entry are to be recorded.

message-queue-name: Enter the qualified name of the user message queue on which the messages for the RJEF writer job's messages are to be recorded. (If no library qualifier is given, *LIBL is used to find the message queue.)

Example

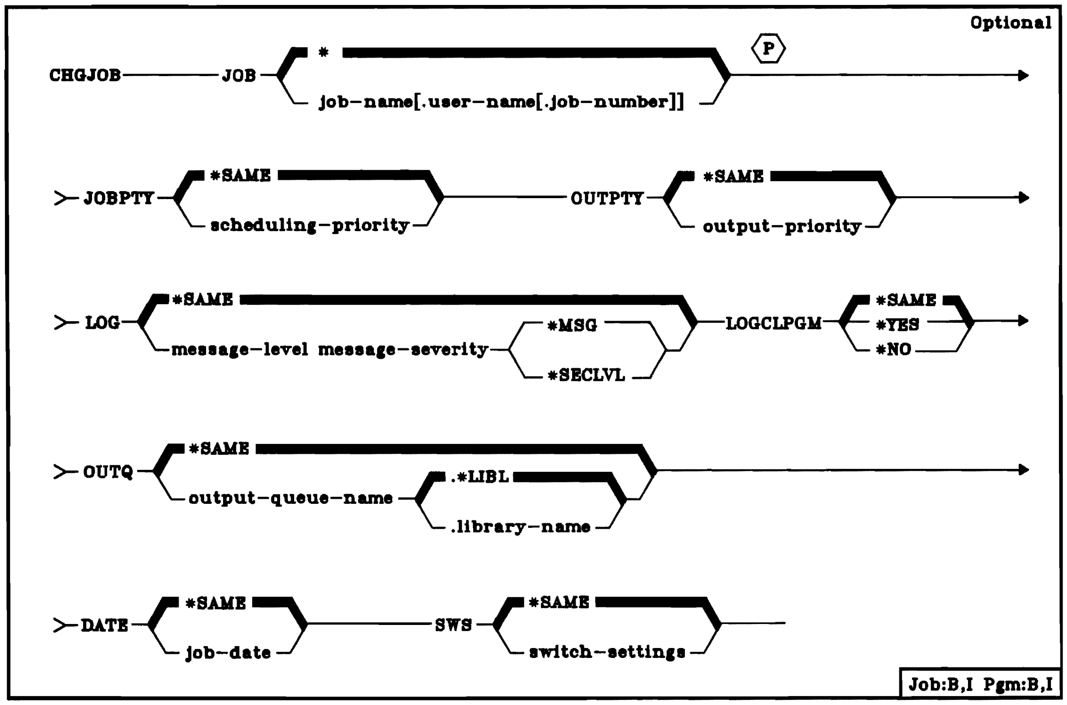
```
CHGFCTE  FCT(FORMCTRL.USERLIB) +
      FORMTYPE(MEDICAL) +
      DEVTYPE(*PUN) +
      FSN(200)
```

This command changes the forms control entry named MEDICAL associated with punch devices. The file sequence number is changed to 200.

CHGJOB (Change Job) Command

The Change Job (CHGJOB) command changes some of the attributes of a job, including priorities, message logging controls, and job switch settings. The job can be on a job or output queue, or it can be active within a subsystem. The new attributes remain in effect for the duration of the job unless changed by another CHGJOB command. If an attribute that no longer affects the job is changed, a message is sent to the user of the command. For example, if the job has already completed execution, it is too late to change the OUTQ and JOBPTY parameters; but if any output files are still on the output queue, a change to the OUTPTY parameter would change their output priority.

Restriction: To use this command, you must be changing your own job or you must have the special job control authority.



JOB Parameter: Specifies the name of the job whose attributes are to be changed.

*: The job whose attributes are to be changed is the job in which this CHGJOB command is issued.

qualified-job-name: Enter the qualified name of the job whose attributes are to be changed. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. If duplicates of the specified name are found, a qualified job name must be specified. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

JOBPTY Parameter: Specifies the scheduling priority to be used for the job being changed. Valid values are 1 through 9, where 1 is the highest priority and 9 is the lowest. (For an expanded description of the JOBPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

*SAME: The scheduling priority is not to be changed.

scheduling-priority: Enter a value, 1 through 9, for the scheduling priority that the job is to have. If the job is currently on the job queue, its position on the queue in relation to other jobs may be changed. The scheduling priority specified here cannot be higher than the priority specified in the user profile under which the job (in which this command is entered) is executing.

OUTPTY Parameter: Specifies the priority that the job's spooled output files are to have for producing output. The highest priority is 1 and the lowest is 9. (For an expanded description of the OUTPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

*SAME: The job's priority for spooled output is not to be changed.

output-priority: Enter a value, 1 through 9, for the priority that the job's output files are to have. The output priority specified here cannot be higher than the priority specified in the user profile under which the job entering the command is executing.

LOG Parameter: Specifies the message logging values to be used by the job. They determine the amount and type of information to be logged in the job log. There are three message logging values; if one value is to be changed, all three must be specified.

*SAME: None of the message logging values are to be changed.

message-level: Enter a value, 0 through 4, that specifies the message logging level to be used for the job's messages. (For additional information on the message levels, refer to *Message Level* under the CRTJOBD command's LOG parameter.)

message-severity: Enter a value, 00 through 99, that specifies the lowest severity level that causes an error message to be logged in the job's log. Only messages that have a severity greater than or equal to this value are logged in the job's log. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

**MSG:* Only first-level message text is to be written to the job's log.

**SECLVL:* Both the first-level and second-level text of the error message is to be written to the job's log.

LOGCLPGM Parameter: Specifies whether the executed commands in a control language program are to be logged to the job log by way of the CL program's message queue. This parameter sets the status of the job's logging flag; if *YES is specified and the LOG(*JOB) value has been specified in the Create CL Program (CRTCLPGM) command, all commands in the CL program that can be logged will be logged to the job log. The commands will be logged in the same manner as requests are logged. Otherwise, the logging flag status is *off* and CL commands will not be logged.

For more information on request logging, refer to the LOG parameter in the CRTJOBD command description.

*SAME: The current state of the job's logging flag is not to be changed.

**YES:* Specifies that commands in a CL program are to be logged to the job log.

**NO:* Specifies that commands in a CL program are not to be logged to the job log.

OUTQ Parameter: Specifies the name of the output queue that is to be used for spooled output produced when OUTQ(*JOB) is specified. This change does not affect files already created in active jobs or files in completed jobs where the files were spooled.

*SAME: The same default output queue is to be used for the job.

qualified-output-queue-name: Enter the qualified name of the default output queue that is to be used by the job. (If no library qualifier is given, *LIBL is used to find the queue.)

DATE Parameter: Specifies the date that is to be assigned to the job.

*SAME: The job date is not to be changed.

job-date: Enter the value that is to be used as the job date for the job; the date must be in the format specified by the system value QDATFMT. (See the *CPF Programmer's Guide* for the description of the possible date formats.) If no job date is specified for a job, the system date is used as the default for any function requiring a job date. The date specified in this parameter overrides the system date for this execution of the job only.

SWS Parameter: Specifies the switch settings for a group of eight job switches to be used with the job. These switches can be set or tested in a CL program and used to control the flow of the program. For example, if a certain switch is on, another program could be called. The job switches may also be valid in other HLL programs. The only values that are valid for each one-digit switch are 0 (off), 1 (on), or X. The X indicates that a switch value is not to be changed.

*SAME: None of the values in the eight job switches are to be changed.

switch-settings: Enter any combination (either in quoted or unquoted form) of eight zeros, ones, or Xs to change the job switch settings. If a switch value is not to be changed, enter an X in the position representing that switch.

Examples

```
CHGJOB  JOB(WS1.DEPT2.123581)  LOG(2 40 *SECLVL)
```

This command changes the job WS1, which is associated with the user profile DEPT2, and has the job number 123581, so that it will receive only commands and associated diagnostic messages (level 2) if the messages have a severity greater than or equal to 40. Second-level text, in addition to first-level message text, is to be logged in the job log.

```
CHGJOB  JOB(PAYROLL)  JOBPTY(4)  +
     OUTPTY(3)  SWS(10XXXX00)
```

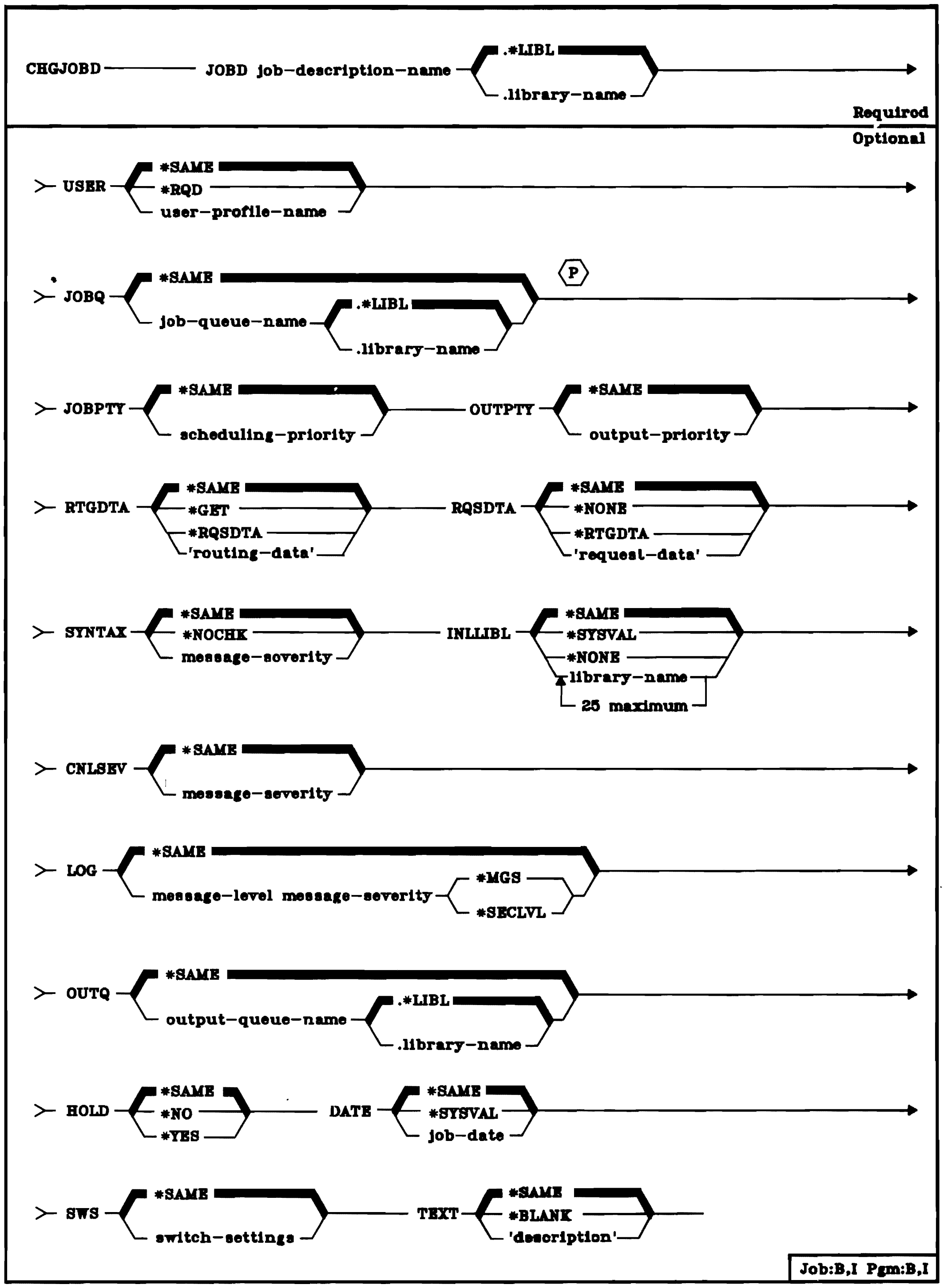
This command changes the scheduling priority of the job PAYROLL to 4 and the priority of the job's spooled output to 3. Also, four of the eight job switches are changed: switches 1 and 2 are set to 1 and 0, switches 3 through 6 remain the same, and switches 7 and 8 are both set to 0. Because only the simple name of the job is specified, there can be only one job named PAYROLL in the system.

CHGJOBD (Change Job Description) Command

The Change Job Description (CHGJOBD) command changes the job-related attributes specified for a job description object through the Create Job Description (CRTJOBD) command. The changes become effective upon command execution.

Any attribute may be changed, except for the public authority attribute. Refer to the *RVKOBJAUT (Revoke Object Authority) Command* and *GRTOBJAUT (Grant Object Authority) Command* for more information on changing object authorizations.

Restrictions: To use this command, you must have operational rights for the user profile named in the USER parameter (if any); that is, you must have that user's authority to initiate a job. You must also have object management and operational authority for the job description, and read authority for the library the job description resides in.


CHGJOBD
JOBD job-description-name
.*LIBL
.library-name
Required
Optional
USER
*SAME
*RQD
user-profile-name
JOBQ
*SAME
job-queue-name
.*LIBL
.library-name
P
JOBPTY
*SAME
scheduling-priority
OUTPTY
*SAME
output-priority
RTGDTA
*SAME
*GET
*RQSDTA
'routing-data'
RQSDTA
*SAME
*NONE
*RTGDTA
'request-data'
SYNTAX
*SAME
*NOCHK
message-severity
INLLIBL
*SAME
*SYSVAL
*NONE
library-name
25 maximum
CNLSEV
*SAME
message-severity
LOG
*SAME
message-level message-severity
*MGS
*SECLVL
OUTQ
*SAME
output-queue-name
.*LIBL
.library-name
HOLD
*SAME
*NO
*YES
DATE
*SAME
*SYSVAL
job-date
SWS
*SAME
switch-settings
TEXT
*SAME
*BLANK
'description'
Job:B,I Pgm:B,I

JOBD Parameter: Specifies the qualified name of the job description being changed. (If no library qualifier is given, *LIBL is used to find the job description.)

USER Parameter: Specifies the name of the user profile to be associated with this job description. The names QSECOFR, QSPL, and QSYS are not valid entries for this parameter.

<u>*SAME:</u> The name of the user profile is not to be changed.

**RQD:* A user name is required in order to use the job description. For work station entries, the user must enter a password when signing on at the work station; the associated user name becomes the name used for the job. *RQD is not valid for job descriptions specified for autostart job entries, or for those used by the JOB command. (It is valid on the SBMJOB command only if USER(*CURRENT) is specified.)

user-profile-name: Enter the user name that identifies the user profile that is to be associated with batch jobs using this job description. For interactive jobs, this is the default user name used when a user signs on without entering a password.

JOBQ Parameter: Specifies the name of the job queue into which jobs using this job description are to be placed.

<u>*SAME:</u> The name of the job queue is not to be changed.

qualified-job-queue-name: Enter the qualified name of the job queue that is to be associated with this job description. (If no library qualifier is given, *LIBL is used to find the job queue.) If the job queue does not exist when the job description is changed, a library qualifier must be specified because the qualified job queue name is retained in the job description.

JOBPTY Parameter: Specifies the scheduling priority to be used for jobs that use this job description. Valid values are 1 through 9, where 1 is the highest priority and 9 is the lowest. (For an expanded description of the JOBPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

<u>*SAME:</u> The scheduling priority is not to be changed.

scheduling-priority: Enter a value, 1 through 9, for the scheduling priority for jobs that use this job description.

OUTPTY Parameter: Specifies the output priority of spooled output files that are produced by jobs that use this job description. The highest priority is 1 and the lowest is 9. (For an expanded description of the OUTPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

<u>*SAME:</u> The output priority for spooled output is not to be changed.

output-priority: Enter a value, 1 through 9, for the output priority of the spooled output files that are produced by jobs that use this job description.

RTGDTA Parameter: Specifies the routing data to be used with this job description to initiate jobs.

<u>*SAME:</u> The routing data is not to be changed.

**GET:* The routing data is obtained from the work station user, by using the display format specified in the work station entry that references this job description.

**RQSDTA:* Up to the first 80 characters of the request data specified in the RQSDTA parameter are to be used as the routing data for the job.

'routing-data': Enter the character string that is to be used as the routing data for jobs that use this job description. For example, the value QCMDI is the routing data used by the IBM-supplied interactive subsystem (QINTER) to route interactive jobs to the IBM-supplied control language processor, QCL. A maximum of 80 characters can be entered (enclosed in apostrophes if necessary).

RQSDTA Parameter: Specifies the request data that is to be placed as the last entry in the job's message queue for jobs using this job description. For example, when a CL command is supplied as request data, it becomes a message that can be read by the control language processor, QCL (if the job is routed to QCL).

<u>*SAME:</u> The request data is not to be changed.

**NONE:* No request data is to be placed in the job's message queue.

**RTGDTA:* The routing data specified in the RTGDTA parameter is to be placed as the last entry in the job's message queue.

'request-data': Enter the character string that is to be placed as the last entry in the job's message queue as a single request. A maximum of 256 characters can be entered (enclosed in apostrophes if necessary). When a CL command is entered, it must be enclosed in single apostrophes, and where apostrophes would normally be used *within* the command, double apostrophes must be used instead.

SYNTAX Parameter: Specifies whether requests placed on the job message queue (for jobs using this job description) are to be syntax-checked as CL commands. When syntax checking is specified, the commands are syntax-checked as they are submitted rather than when the job is executed, thus providing an earlier diagnosis of syntax errors. If checking is specified, the message severity that causes a syntax error to terminate processing of a job is also specified.

*SAME: The SYNTAX parameter value is not to be changed.

**NOCHK:* The request data is not to be syntax-checked as CL commands.

message-severity: The request data is to be syntax-checked as CL commands; if a syntax error occurs that is equal to or greater than the error message severity specified here, the execution of the job containing the erroneous command is suppressed. Enter a value, 00 through 99, that specifies the lowest message severity that can cause job execution to end. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

If the message severity is specified, it is used only when the job description is used by a job command that also has RQSDTA(*) specified and the requests are CL commands.

INLLIBL Parameter: Specifies the initial user part of the library list that is to be used for jobs using this job description. For more information on the use of library lists, see the *CPF Programmer's Guide.*

*SAME: The initial user part of the library list is not to be changed.

**SYSVAL:* The system default library list is to be used for jobs that use this job description. The default library list contains the library names that were specified in the system values QSYSLIBL and QUSRLIBL at the time that a job using this job description is initiated.

**NONE:* The user part of the initial library list is to be empty; only the system portion is to be used.

library-name: Enter the names of one or more libraries that are to be in the user part of the library list for jobs that use this job description. No more than 25 names can be specified; the libraries are searched in the same order as they are listed here.

CNLSEV Parameter: Specifies the message severity level of escape messages that can cause a batch job to be canceled. The batch job is canceled when a request in the batch input stream sends to the request processing program an escape message whose severity code is equal to or greater than that specified here. This parameter value is compared with the severity of any unmonitored escape message that occurs as a result of executing a noncompiled CL command in a batch job.

For a description of each IBM-defined severity code level, refer to the expanded description of the SEV parameter in Appendix A.

*SAME: The message severity level for canceling batch jobs is not to be changed.

message-severity: Enter a value, 00 through 50, that specifies the message severity of an escape message that results from a request in the batch input stream and that causes the jobs that use this job description to be canceled. Because escape messages typically have a maximum severity level of 50, a value of 50 or lower must be specified in order for a job to be canceled as a result of an escape message. An unhandled escape message whose severity is equal to or greater than the value specified causes the jobs to be canceled. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

LOG Parameter: Specifies the message logging values to be used by jobs that use this job description. They determine the amount and type of information to be logged in the job log. There are three message logging values; if one value is to be changed, all three must be specified.

*SAME: None of the message logging values are to be changed.

message-level: Enter a value, 0 through 4, that specifies the message logging level to be used for the job's messages. (For additional information on the message levels, refer to *Message Level* under the CRTJOBD command's LOG parameter.)

message-severity: Enter a value, 00 through 99, that specifies the lowest severity level that causes an error message to be logged in the job's log. Only messages that have a severity greater than or equal to this value are logged in the job's log. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

***MSG:** Only first-level message text is to be logged in the job's log.

***SECLVL:** Both the first-level and second-level text of the error message is to be logged in the job's log.

OUTQ Parameter: Specifies the name of the output queue to be used as the default output queue for jobs that use this job description.

*SAME: The default output queue is not to be changed.

qualified-output-queue-name: Enter the qualified name of the default output queue that is to be used by jobs that use this job description. (If no library qualifier is given, *LIBL is used to find the queue.) If the output queue does not exist when the job description is changed, a library qualifier must be specified, because the qualified output queue name will be retained in the job description.

HOLD Parameter: Specifies whether jobs using this job description are to be put on the job queue in the hold state. A job placed on the job queue in the hold state is held until it is released by the Release Job (RLSJOB) command or canceled, either by the Cancel Job (CNLJOB) command or by the Clear Job Queue (CLRJOBQ) command. If the job is not executed before CPF is terminated, the job queue can be cleared (and the job canceled) when CPF is started again.

<u>*SAME:</u> The value of the HOLD parameter is not to be changed.

**NO:* Jobs using this job description are not to be held when they are put on the job queue.

**YES:* Jobs using this job description are to be held when they are put on the job queue.

DATE Parameter: Specifies the date that is to be assigned for jobs that use this job description.

<u>*SAME:</u> The job date is not to be changed.

**SYSVAL:* The value in the QDATE system value at the time that the job is initiated is to be used as the job date.

job-date: Enter the value that is to be used as the job date for the job being initiated; the format that is currently specified for the system value QDATFMT must be used. (See the *CPF Programmer's Guide* for the QDATFMT system value.)

SWS Parameter: Specifies the initial switch settings for a group of eight job switches for jobs that use this job description. These switches can be set or tested in a CL program and used to control the flow of the program. For example, if a certain switch is on, another program could be called. The job switches may also be valid in other HLL programs. The only values that are valid for each one-digit switch are 0 (off) or 1 (on).

<u>*SAME:</u> None of the values in the eight job switches are to be changed.

switch-settings: Enter any combination (either in quoted or unquoted form) of eight zeros or ones to change the job switch settings.

TEXT Parameter: Lets the user enter text that briefly describes the job description. (For an expanded description of the TEXT parameter, see Appendix A.)

<u>*SAME:</u> The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CHGJOBD  JOBD(QPGMR.QGPL) JOBPTY(2) OUTPTY(2)
```

This command allows jobs using the IBM-supplied job description QPGMR in library QGPL to process with a higher job and output priority than originally specified for QPGMR. QPGMR originally has priorities of 5 set for job execution and output; refer to the *CPF Programmer's Guide* for IBM-supplied job description parameter values.

Assume that your user profile was created as follows:

```
CRTUSRPRF  USRPRF(JLRAY) PASSWORD(GAMMA) SPCAUT(*JOBCTL) +
      PTYLMT(4) PUBAUT(*NONE)
```

Then you attempt to modify the priority limits of the job description BATCH5 with the following command:

```
CHGJOBD  JOBD(BATCH5) USER(JLRAY) JOBPTY(1) OUTPTY(1)
```

Because the priority limit specified in the user profile takes precedence over any limit specified in a job description, an error message is sent and a priority of 4 is assumed for both job and output priority levels.

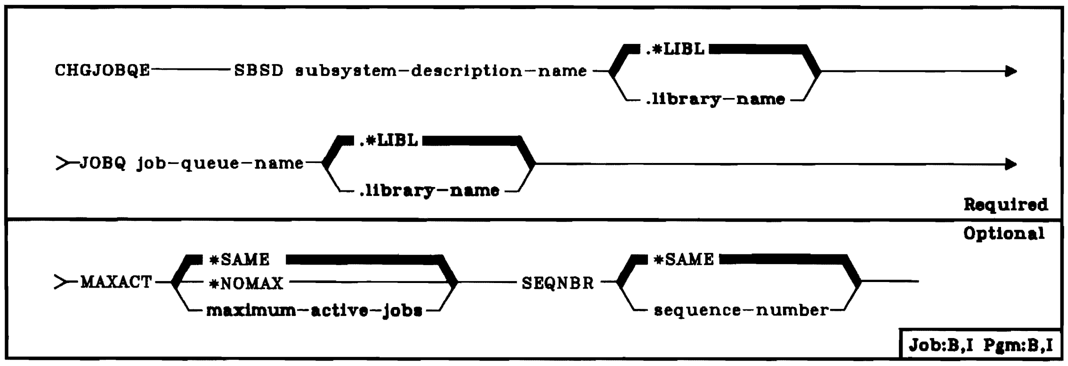
CHGJOBQE (Change Job Queue Entry) Command

The Change Job Queue Entry (CHGJOBQE) command changes an existing job queue entry within the specified subsystem description; the associated subsystem must be inactive when the change is made. A job queue entry identifies the job queue from which jobs are to be selected for execution within the subsystem. Jobs can be placed on a job queue by spooling readers or by using the following commands:

- Submit Job (SBMJOB)
- Submit Card Jobs (SBMCRDJOB)
- Submit Data Base Jobs (SBMDBJOB)
- Submit Diskette Jobs (SBMDKTJOB)
- Transfer Job (TFRJOB)

Within a subsystem, job queues with lower sequence numbers are processed first. For more information, refer to the SEQNBR parameter.

Restriction: To use this command, you must have operational and object management rights for the subsystem description being changed.



SBSD Parameter: Specifies the qualified name of the subsystem description that contains the job queue entry being changed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

JOBQ Parameter: Specifies the unique qualified name of the job queue that is to be a source of batch jobs that are to be initiated by the subsystem. (If no library qualifier is given, *LIBL is used to find the job queue.) If the job queue does not exist when the entry is changed, a library qualifier must be specified because the qualified job queue name is retained in the subsystem description.

MAXACT Parameter: Specifies the maximum number of jobs that can be concurrently initiated from this job queue. (For an expanded description of the MAXACT parameter, see Appendix A.)

*SAME: The maximum number of jobs that can be concurrently active is not to be changed.

**NOMAX:* There is to be no maximum on the number of jobs that can be concurrently initiated. However, the maximum activity level of the routing entries might prevent routing steps from being initiated. If *NOMAX is specified, all the jobs on the job queue will be initiated (within the limit specified by the MAXJOBS parameter in the subsystem description), even though the activity level of the storage pool might prohibit them from executing concurrently.

maximum-active-jobs: Enter a value that specifies the new maximum for the number of jobs that can be concurrently active.

SEQNBR Parameter: Specifies a sequence number for this job queue, to be used by the subsystem to determine the order in which the job queues are to be processed.

SAME: The sequence number assigned to this job queue is not to be changed.

sequence-number: Enter the sequence number to be assigned to this job queue. The sequence number must be unique within the subsystem description. Valid values are 1 through 9999.

The subsystem first selects jobs from the job queue with the lowest sequence number. Once all jobs on that queue have been processed or the number of jobs specified on the MAXACT parameter has been reached, the subsystem processes jobs on the queue with the next higher sequence number. This sequence continues until all job queue entries have been processed or until the subsystem has reached its limit for overall maximum jobs (as specified by the MAXJOBS parameter in the subsystem description). In some cases, this sequence is interrupted and the subsystem processes a queue with a lower sequence number. This occurs for this subsystem when:

- A held job or job queue is released
- A job is placed on or transferred to a queue
- A new queue is allocated
- A job terminates

Example

```
CHGJOBQE  SBSD(QBATCH.QGPL) JOBQ(QBATCH.QGPL) +
     MAXACT(4)
```

This command changes the maximum number of jobs submitted from the job queue QBATCH via the job queue entry to the QBATCH subsystem for concurrent processing. A maximum of four jobs from the QBATCH job queue can be concurrently active. The sequence number of the job queue entry is not changed.

This page is intentionally left blank.

CHGJRN (Change Journal) Command

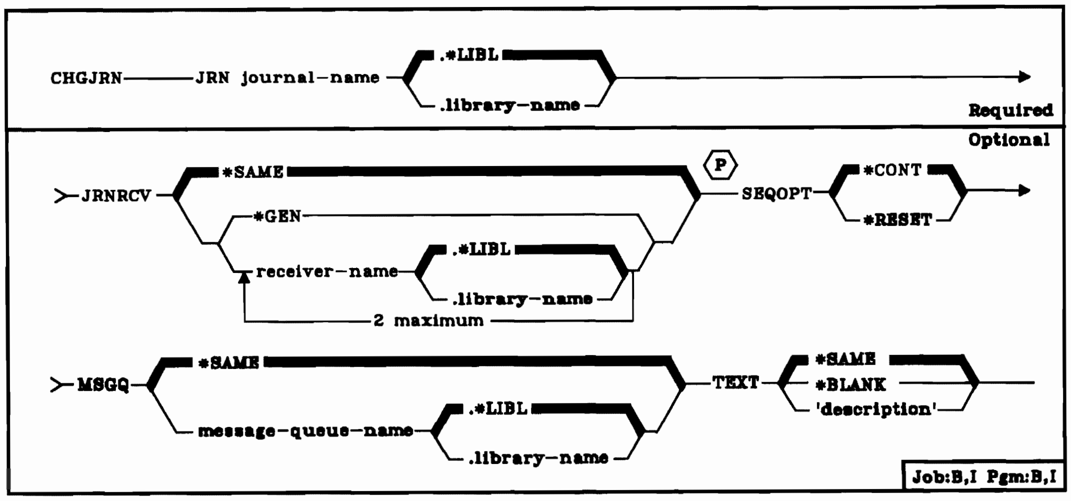
The Change Journal (CHGJRN) command changes the journal receivers, the message queue, or the descriptive text associated with the indicated journal. The command allows up to two journal receivers to be attached to the specified journal. These replace all previously attached journal receivers. The designated journal receivers will begin receiving journal entries for the journal immediately. The sequence numbering of journal entries can be reset when the receivers are changed. If the sequencing is not reset, an informational message is sent indicating the first sequence number in the newly attached receivers. If the first sequence number is greater than 2000 000 000, an informational message is sent to the system operator.

If JRNRCV is *SAME, the currently attached journal receivers will remain attached.

Restrictions: Receivers that already contain journal entries cannot be reattached to a journal. There can be no more than two journal receivers attached to the journal at any specific time.

If journaled changes are being applied or removed while this command is being executed, you cannot switch journal receivers (JRNRCV(*SAME) must be specified).

Resetting of sequence numbers is not valid if JRNRCV is *SAME, or if any files being journaled are open *and* contain changes that have not yet been forced to auxiliary storage. When the maximum sequence number is reached, an exception will be signaled (entry not journaled) and all subsequent activity requiring journaling will terminate.



JRN Parameter: Specifies the qualified name of the journal to have its journal receivers or operational attributes changed. (If no library qualifier is given, *LIBL is used to find the journal.)

JRNRCV Parameter: Specifies which journal receivers are to be attached to the designated journal.

*SAME: Specifies that the journal receivers currently attached to the journal are to remain attached.

**GEN:* Specifies that the journal receiver(s) are to be created by the system and then attached to the designated journal. The journal receiver(s) will be created with the same attributes and in the same library as the currently attached journal receiver(s) and will be owned by the same owner. The name of the new journal receiver will be derived by appending a four-digit number to a portion of the name of the current receiver, or by incrementing the number in the current journal receiver. The name of the journal receiver created and attached will be returned in an informational message. For more information on generation of journal receiver names, refer to *CPF Programmer's Guide.*

receiver-name: Enter the qualified names of the journal receivers that are to be attached to the designated journal. (If no library qualifier is given, *LIBL is used to find the journal receiver.) The journal receivers must have been previously created in the specified library, and must not have been previously attached to any journal.

A maximum of two journal receivers may be attached at one time. Any combination of *GEN and receiver name is valid.

SEQOPT Parameter: Specifies whether the journal sequence numbers are to continue being incremented or whether the journal sequence number is to be reset to one in the newly attached journal receivers.

*CONT: Specifies that the journal sequence number of the next journal entry generated is to be one greater than the sequence number of the last journal entry in the currently attached journal receiver(s).

**RESET:* Specifies that the journal sequence numbers in the newly attached journal receivers are to be reset to one. *RESET is not valid if JRNRCV(*SAME) is specified or if any file being journaled is open *and* contains changes that have not yet been forced to auxiliary storage.

MSGQ Parameter: Specifies whether the message queue associated with the journal is to be changed. The message issued when a journal receiver's storage limit (threshold) is exceeded is sent to this message queue. To set the threshold value, refer to the CRTJRNRCV command.

*SAME: Specifies that the message queue is not to change.

message-queue-name: Enter the qualified name of the message queue to which the message will be sent, which will replace the message queue previously specified.

TEXT Parameter: Specifies whether the descriptive text associated with the journal is to be changed.

*SAME: Specifies that the text is not to be changed.

**BLANK:* The text is to be replaced by blanks.

'description': Enter no more than 50 characters, enclosed in apostrophes. The value entered becomes the new text associated with this journal.

Examples

```
CHGJRN  JRN(JRNLA) JRNRCV(RCV10) SEQOPT(*RESET)
```

This command causes all journal receivers currently attached to journal JRNLA to be detached (JRNLA is found using the library search list *LIBL). Journal receiver RCV10 (found using the library search list *LIBL) is attached to journal JRNLA. The first journal entry in journal receiver RCV10 will have a sequence number of one.

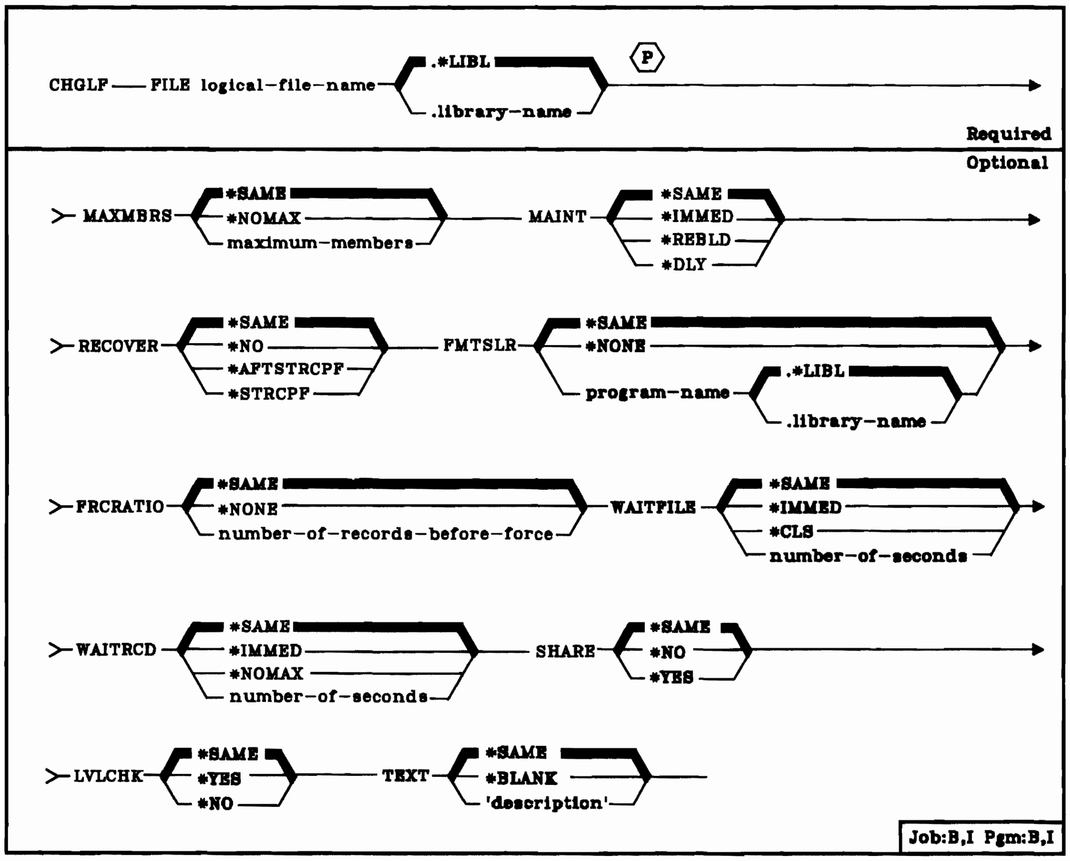
```
CHGJRN  JRN(JRNLA) JRNRCV(*GEN *GEN)
```

This command causes all journal receivers currently attached to journal JRNLA to be detached. Two new journal receivers will be created and attached to journal JRNLA. The libraries and owners of the new journal receivers will be the same as the libraries and owners of the detached receivers. The names of the new receivers depend on the names of the detached receivers. (For example, if one receiver was named RCVJRNA, the new receiver will be named RCVJRN0001. If the receiver was named RCVJRN0001, the new receiver will be named RCVJRN0002.) The first journal entry in the new journal receivers will have a sequence number of one greater than the last sequence number in the detached receivers.

CHGLF (Change Logical File) Command

The Change Logical File (CHGLF) command changes the attributes of a logical file and its members. The changed attributes will be used for all members subsequently added to the file. To change the attributes of a specific member, execute the CHGLFM (Change Logical File Member) command.

Restrictions: To change a logical file, you must have object management and operational rights for the file and read rights to the library. In order to change the file, an exclusive no read lock is necessary; no one may be using the file for any purpose.



FILE Parameter: Specifies the qualified name of the logical file to be changed. (If no library qualifier is given, *LIBL is used to find the file.)

MAXMBRS Parameter: Specifies the maximum number of members that the logical file can have at any time. The maximum number of members specified must be greater than or equal to the current number of members in the file.

SAME: The maximum number of members should not be changed.

**NOMAX:* No maximum is specified for the number of members; the system maximum of 32 767 members per file is used.

maximum-members: Enter the value for the maximum number of members that the logical file can have. A value of 1 through 32767 is valid.

MAINT Parameter: Specifies the type of access path maintenance to be used for all members of the logical file. This parameter is valid only if a keyed access path is used.

Only the following changes to a file's access path maintenance are allowed: *REBLD to *IMMED (if the file was originally created as *IMMED or *REBLD), *IMMED to *REBLD, *DLY to *REBLD, and *REBLD to *DLY (if the file was originally created as *DLY).

Existing MAINT Value	**CHGLF MAINT Parameter Value**		
	***REBLD**	***DLY**	**IMMED**
*REBLD	N/A	Note 1	Note 2
*DLY	YES	N/A	NO
*IMMED	YES	NO	N/A

Notes:
1. Allowed only if file was originally created with MAINT(*DLY).
2. Allowed only if file was originally created with MAINT(*IMMED) or MAINT(*REBLD).

*SAME: The files access path maintenance is not to be changed.

**IMMED:* The access path is to be continuously (immediately) maintained for each logical file member. The access path is updated each time a record is changed, added to, or deleted from the member. The records can be changed through a logical file that uses the logical file member regardless of whether the logical file is opened or closed. *IMMED must be specified for all files requiring unique keys to ensure uniqueness in all inserts and updates.

**REBLD:* The access path is to be rebuilt when a file member is opened during program execution. The access path is continuously maintained until the member is closed; access path maintenance is then terminated. *REBLD is not valid for access paths that are to contain unique key values.

**DLY:* The maintenance of the access path is to be delayed until the member is opened for use. The access path is then updated only for records that have been added, deleted, or updated since the file was last closed. (While the file is open, all changes made to based-on members are immediately reflected in the access paths of the opened files members, no matter what is specified for MAINT.) To prevent a lengthy rebuild time when the file is opened, *DLY should be specified only when the number of changes to the access path between a close and the next open are small (when key fields in records for this access path change infrequently). *DLY is not valid for access paths that require unique key values.

If the number of changes saved reaches approximately 10 per cent of the access path size, the system will stop saving changes and the access path will be completely rebuilt the next time the file is opened.

RECOVER Parameter: Specifies, for files having immediate or delayed maintenance on their access paths, when recovery processing of the file is to be performed if a system failure occurred while the access path was being changed.

An access path having immediate or delayed maintenance can be rebuilt during start CPF (before any user can execute a job), or after start CPF has finished (during concurrent job execution), or when the file is next opened. While the access path is being rebuilt, the file cannot be used by any job.

An access path having rebuild maintenance will be rebuilt the next time its file is opened, the time that it normally is rebuilt. This parameter is valid only if a keyed access path is used. For more information on recovery processing, refer to the *CPF Programmer's Guide*.

*SAME: The files recovery attribute should not be changed.

**NO:* The access path of the file is not to be rebuilt. The file's access path is rebuilt the next time the file is opened.

**AFTSTRCPF:* The file is to have its access path rebuilt after the start CPF operation has been completed. This option allows other jobs not using this file to begin processing immediately after the CPF has been started. If a job tries to allocate the file while its access path is being rebuilt, a file open exception occurs if the specified wait time for the file is exceeded.

**STRCPF:* The file is to have its access path rebuilt during the start CPF operation. This ensures that the file's access path will be rebuilt before the first user program tries to use it; however, no jobs can begin execution until after all files that specify RECOVER(*STRCPF) have their access paths rebuilt.

FMTSLR Parameter: Specifies the name of a record format selector program that is to be called when the logical file member contains more than one logical record format. The user-written selector program is called when a record is to be inserted into the data base file and a record format name is not included in the HLL program. The selector program receives the record as input, determines the record format to be used, and returns it to the data base. This program must perform this function for every member in the logical file that has more than one record format, unless the HLL program itself specifies the record format name. (More information about the use of format selector programs is contained in the *CPF Programmer's Guide*.)

This parameter is not valid if the logical file has only one record format.

**SAME:* The files format selector program should not be changed.

**NONE:* There is no selector program for this logical file. The file cannot have more than one logical record format, or the HLL program itself must specify the record format name.

program-name: Enter the qualified name of the format selector program to be called when a record is to be inserted into a member having more than one format. The selector program name can be optionally qualified by the name of the library in which the program is stored. (If no library qualifier is given, *LIBL is used to find the program.)

A program specified as the format selector program cannot be created with USRPRF(*OWNER) specified in its create program command.

FRCRATIO Parameter: The force write ratio parameter specifies the number of inserted, updated, or deleted records that are processed before they are forced to auxiliary (permanent) storage. (For an expanded description of the FRCRATIO parameter, see Appendix A.)

The force write ratio specified for a logical file must be less than or equal to the smallest force write ratio of its based-on files. If a larger force write ratio is specified it is ignored and a message is sent informing you of the action.

If a physical file associated with this logical file is being journaled, a larger force write ratio or *NONE may be specified. Refer to the *CPF Programmer's Guide* for more information on the Journal Management Facility.

SAME: The files force write ratio is not to be changed.

**NONE:* There is no force write ratio; the system determines when the records are written in auxiliary storage.

number-of-records-before-force: Enter the number of new or changed records that are processed before they are explicitly forced into auxiliary storage.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

SAME: The wait attribute of the file is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

WAITRCD Parameter: Specifies the number of seconds that the program is to wait for a record that is to be updated or deleted. If the record cannot be allocated in the specified wait time, an error message is sent to the program.

*SAME: The record wait attribute of the file is not to be changed.

**IMMED:* The program is not to wait; when a record is locked, an immediate allocation of the record is required.

**NOMAX:* The wait time will be the maximum allowed by the system (32 767 seconds).

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether an ODP (open data path) to the logical file member is to be shared with other programs in the same job. When an ODP is shared, the programs accessing the file share such things as the position being accessed in the file, the file status, and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

If SHARE is specified, all members in the file will be changed.

*SAME: The ODP sharing value of the member is not to be changed.

**NO:* An ODP created by the program when the file member is opened is not to be shared with other programs in the job. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* The same ODP is to be shared with each program in the job that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the record format identifiers are to be level checked to verify that the current record format identifier is the same as that specified in the program that opens the logical file. This value can be overridden on the OVRDBF command at execution time.

*SAME: The level check value of the member is not to be changed.

**YES:* The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not match, an error message is sent to the program requesting the open, and the file is not opened.

**NO:* The level identifiers are not to be checked when the file is opened.

TEXT Parameter: Enter text that briefly describes the logical file member. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text that describes the member is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

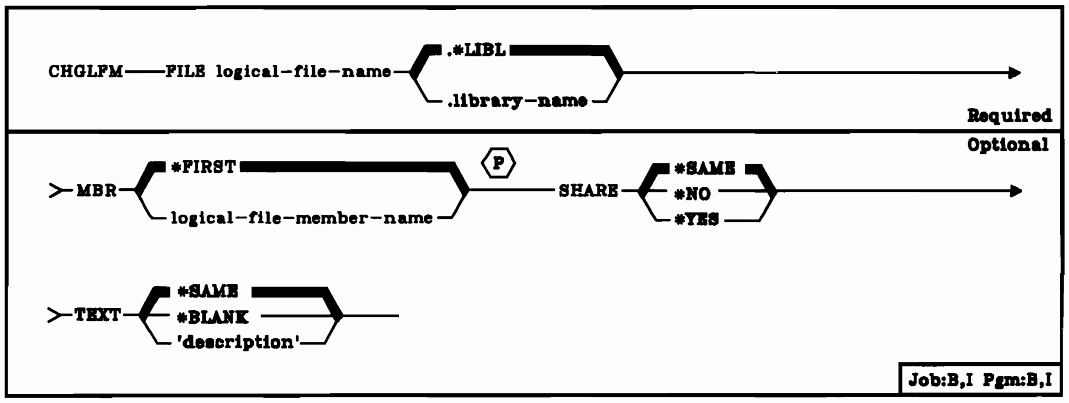
```
CHGLF  FILE(INV.QGPL)  MBR(FEB)  FMTSLR(INVFMTS)
```

The member named FEB in the logical file INV that is stored in the QGPL library is to be changed so that the new format selector program to be used with the logical file will be INVFMTS. *LIBL will be used to find the format selector program.

CHGLFM (Change Logical File Member) Command

The Change Logical File Member (CHGLFM) command changes the attributes of a logical file member.

Restrictions: To change a logical member, you must have object management and operational rights for the logical file that contains the member. You must also have read rights for the file library. In order to change the member, no other user may be holding the file for exclusive use. Concurrent users may have the member open, but changes made to the member will not be reflected in any open members. In order to effect the changes in any open members, you must first close the member (this must be a full close if the member is open SHARE(*YES)) and then open it again.



FILE Parameter: Specifies the qualified name of the logical file that contains the member to be changed. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name of the member, or the first member (*FIRST), to be changed.

**FIRST:* The first member of the specified logical file is to be changed.

logical-file-member-name: Enter the name of the logical file member to be changed.

SHARE Parameter: Specifies whether an ODP (open data path) to the logical file member is to be shared with other programs in the same job. When an ODP is shared, the programs accessing the file share such things as the position being accessed in the file, the file status, and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

**SAME:* The member's ODP sharing value should not be changed.

**NO:* An ODP created by the program when the file member is opened is not to be shared with other programs in the job. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* The same ODP is to be shared with each program in the job that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Enter text that briefly describes the logical file member. (For an expanded description of the TEXT parameter, see Appendix A.)

**SAME:* The text that describes the member is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

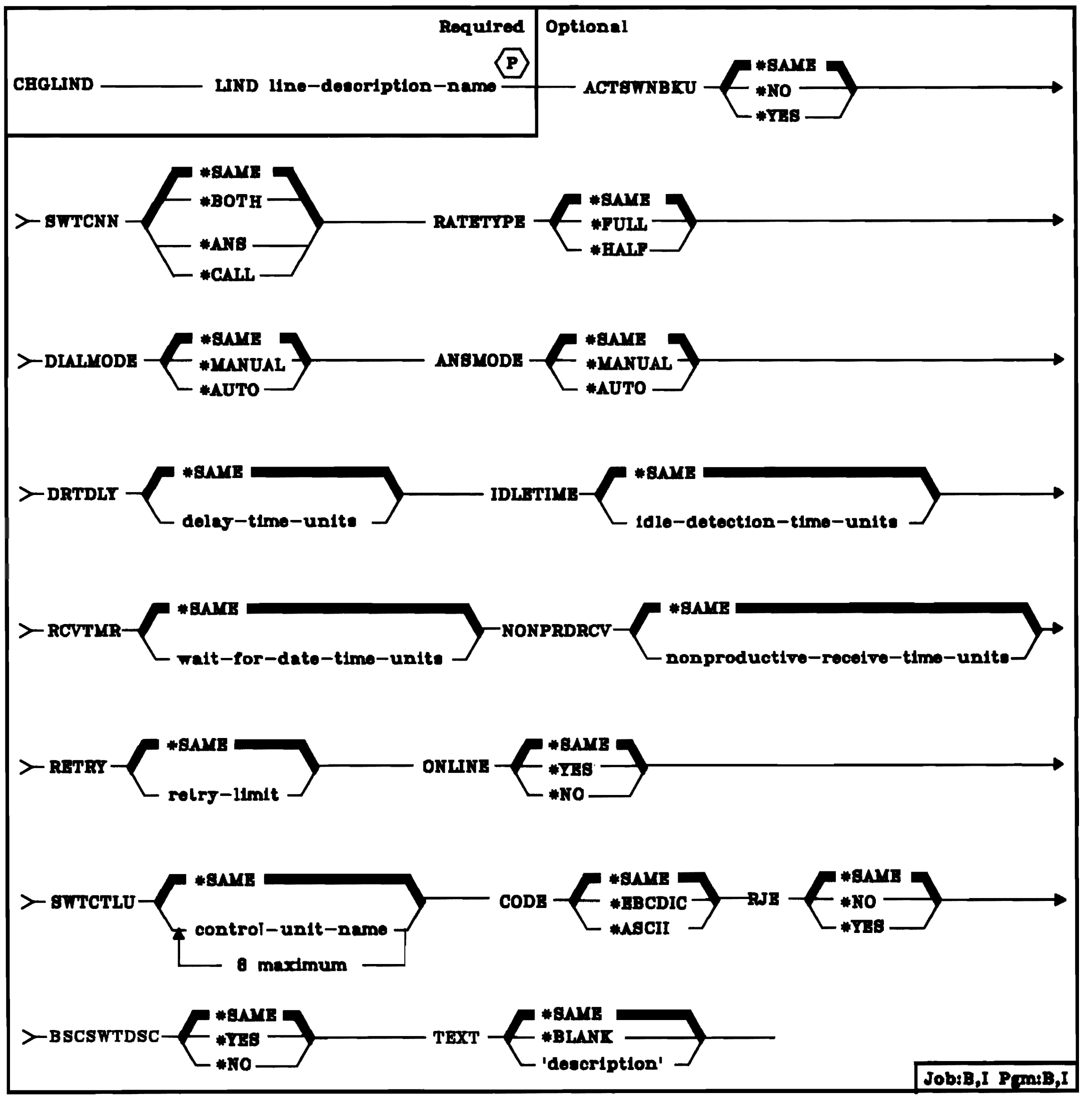
Example

```
CHGLFM  FILE(INV.QGPL) MBR(FEB) +
     TEXT('Logical file member for FEB')
```

The member named FEB in the logical file INV that is stored in the QGPL library is to be changed so that the member will have new text.

CHGLIND (Change Line Description) Command

The Change Line Description (CHGLIND) command changes some of the parameter values in the named line description. The RETRY, ONLINE, and TEXT parameters can be changed while the line itself is still online (the changed values become effective immediately). All other changes must be made with the line varied offline.



LIND Parameter: Specifies the name of the line description that is to have one or more of its attributes changed.

ACTSWNBKU Parameter: Specifies whether the switched network backup feature is to be activated (if the feature is installed) or de-activated. This feature lets you bypass a broken, nonswitched (leased line) connection by converting the line to a switched line operation as specified by the SWTCNN, DIALMODE, and ANSMODE parameters. This parameter must be *SAME for TYPE(*BSCT).

SAME: The value specified in the line description is not to be changed.

*NO: The backup feature is to be de-activated if it was active. (The line is back in normal operation.)

**YES:* The backup feature is to be activated if it is not active.

SWTCNN Parameter: Specifies whether the line is to be used for incoming calls, outgoing calls, or both. This parameter must be *SAME unless CNN(*SWT) and SWNBKU(*YES) were specified when this line description was created.

SAME: The use of the line remains the same.

**BOTH:* The line can be used for both incoming and outgoing calls.

**ANS:* The line can be used for incoming calls only.

**CALL:* The line can be used for outgoing calls only.

RATETYPE Parameter: Specifies the speed at which the line operates if the line has the data rate select function. RATETYPE(*HALF) is valid only if SELECT(*YES) was specified on the CRTLIND command that created this line description.

SAME: The line speed remains the same.

**FULL:* The line is operated at full speed.

**HALF:* The line is operated at half speed.

DIALMODE Parameter: Specifies whether the line connection is to be made manually or automatically. DIALMODE(*AUTO) is valid only if AUTOCALL(*YES) is specified.

*SAME: The value specified in the line description is not to be changed.

**MANUAL:* The line connection is made by the user manually dialing the connection (that is, the called station). If AUTOCALL(*NO) is specified, *MANUAL is the default.

**AUTO:* The line connection is made by the system automatically dialing the called station. If AUTOCALL(*YES) is specified, *AUTO is the default.

ANSMODE Parameter: Specifies how incoming calls to System/38 can be answered (that is, how the switched line connection is to be made through the autoanswer facilities for calls coming from a remote control unit or work station). ANSMODE(*AUTO) is valid only if AUTOANS(*YES) was specified in the associated CRTLIND command that created this line description.

*SAME: The method of answering incoming calls remains the same.

**MANUAL:* The incoming call must be manually answered.

**AUTO:* The incoming call is automatically answered by the autoanswer modem feature.

DTRDLY Parameter: The data terminal ready (DTR) delay parameter specifies the maximum length of time that the system is to pause before ending a command that resets the DTR condition. The delay time cannot exceed 3 seconds.

*SAME: The maximum delay time specified in the line description is not to be changed.

delay-time-units: Enter a value, 0 through 15, that is multiplied by the base time unit of 200 milliseconds to determine the maximum delay time before the system resets the DTR condition. For most networks, 200 milliseconds (specified here by a 1) is appropriate. If 0 is specified or assumed, a default time of 100 milliseconds is used.

IDLETIME Parameter: Specifies, for any transmission sent by the primary station that requires a response, the maximum time within which the beginning of the secondary station's response must be detected (received). This time should be greater than the sum of the:

- Transmission time to the secondary station
- Processing time of the control unit's response at the secondary station (not including customer program processing time or operator response time)
- Clear-to-send time at the secondary station modem
- Transmission time from the secondary station

This parameter is not valid for secondary SDLC lines or BSC lines.

*SAME: The maximum time during which the secondary station's response can be detected remains the same.

idle-detection-time-units: Enter a value, 0 through 255, that is multiplied by the base time unit of 53.3 milliseconds to determine the maximum detection time for the secondary station's response (53.3 milliseconds through 13.6 seconds). If 0 is specified or assumed, a default time of 500 milliseconds is used.

RCVTMR Parameter: The receive timer parameter, valid for BSC lines only, specifies the time the system will wait for data before a time-out occurs. The time is measured in 200-millisecond intervals; a period of 3 seconds (a RCVTMR value of 15) is appropriate for most systems.

*SAME: The time interval is not to be changed.

wait-for-data-time-units: The time period the system will wait for data. The maximum time-out period allowed is 25.4 seconds (a RCVTMR value of 127).

NONPRDRCV Parameter: The nonproductive receive parameter specifies the maximum length of time in which to receive an intelligible transmission. The time is specified by a value that is multiplied by the base time unit of 500 milliseconds. Because the nonproductive receive time depends upon the line speed, refer to the table given in the NONPRDRCV parameter description of the CRTLIND command.

*SAME: The maximum delay time to wait for intelligible data remains the same.

nonproductive-receive-time-units: Enter a value, 0 through 255, that is multiplied by the base time unit of 500 milliseconds to determine the maximum time to wait for intelligible data. If 0 is specified or assumed, a default time of 128 seconds is used.

RETRY Parameter: Specifies the maximum number of retries that can be made to correct an error that occurs.

*SAME: The retry limit remains the same.

retry-limit: Enter a value, 0 through 21, that is to be multiplied by a base number of 1 or 7 to determine the *maximum* number of retries that can be attempted if necessary. All errors associated with making a switched connection to the line use the base multiplier 1; all other line errors use the base multiplier 7. If 0 is specified, no retries occur.

In no case does the system attempt more than 21 retries. Therefore, a value of 0 through 21 is valid for retrying errors that use the multiplier 1. A value of 0 through 3 is valid for those using the multiplier 7; in this case, any value specified that is greater than 3 is assumed to be 3, and a maximum of 21 retries (3 times 7) can be attempted if necessary.

ONLINE Parameter: Specifies whether the line is to be varied online automatically when the Control Program Facility (CPF) is started. After CPF is started, the Vary Line (VRYLIN) command can be used to modify the status of the line. ONLINE(*YES) should be specified for only one line description per line number; if it is specified for more than one, the system chooses the first line description based on alphabetic order.

*SAME: The value specified in the line description is not to be changed.

**YES:* The line is to be online when CPF is started.

**NO:* The line is to be offline when CPF is started. The VRYLIN command must be used to put the line online, making it operational.

SWTCTLU Parameter: Specifies up to 8 control unit names that can establish a connection with this switched BSC line.

*SAME: The control unit names are not to be changed.

control-unit-name: Specifies the previously created control unit name (up to 8).

CODE Parameter: Specifies the BSC line code to be used for communications. This parameter is valid for BSC lines only.

*SAME: The line code is not to be changed.

**EBCDIC:* The EBCDIC character set code is to be used.

**ASCII:* The ASCII character set code is to be used. ASCII is not valid if RJE(*YES) is specified.

RJE Parameter: Specifies, for BSC only, whether this line description is to be used by the Remote Job Entry Facility (RJEF).

*SAME: The value specified in the line description is not to be changed.

**NO:* This line description is not to be used by RJEF.

**YES:* This line description is to be used by RJEF.

BSCSWTDSC Parameter: Specifies whether inactivity on this BSC switched line (while in contention mode) should cause a line disconnect due to a 30-second timeout. Some CL commands may cause a timeout disconnect in a debugging or problem determination situation; in that case, you could use this parameter to disable the automatic timeout and continue. This parameter is valid only if TYPE(*BSC) and CNN(*SWT) are specified, or if TYPE(*BSC) and CNN(*PP) and SWNBKU(*YES) are specified.

*SAME: The value specified in the line description is not to be changed.

**YES:* The switched BSC line will be automatically disconnected after a 30-second period of inactivity (while in contention mode).

**NO:* The switched BSC line will not be automatically disconnected after a 30-second period of inactivity.

Note: When the last file is closed, the normal BSC switched line disconnect is not affected by this parameter.

TEXT Parameter: Specifies the user-defined text that describes the line description. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

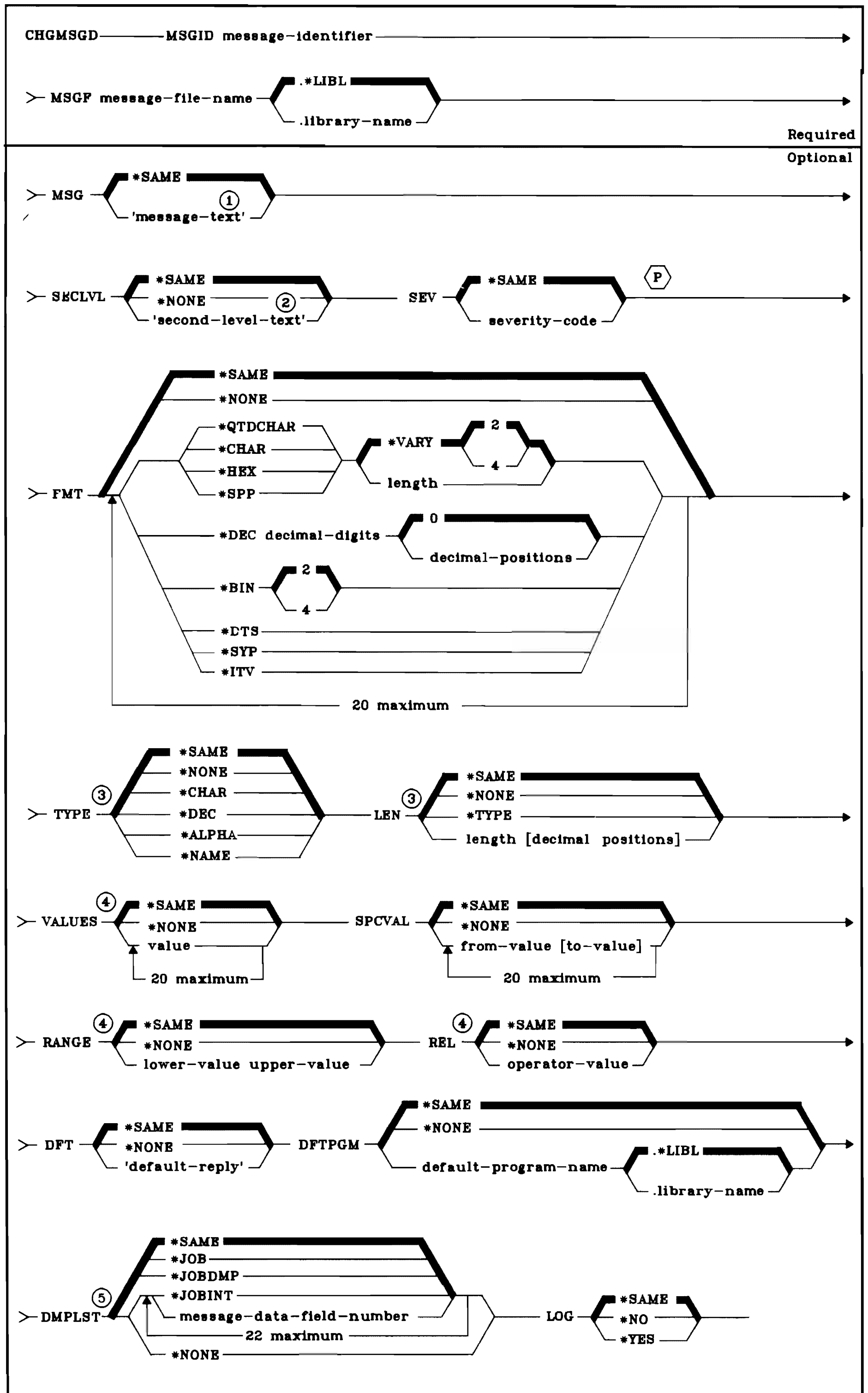
```
CHGLIND  LIND(LINE01)  RETRY(15)  ONLINE(*NO)
```

This command changes the line description of the line named LINE01. The RETRY parameter is changed to a maximum of 15 retries for errors occurring while a switched connection is being attempted and a maximum of 21 retries for all other errors associated with the line. The ONLINE parameter is changed to *NO, specifying that the line will not be operational when CPF is started.

CHGMSGD (Change Message Description) Command

The Change Message Description (CHGMSGD) command describes changes to an existing message description stored in a message file, and stores those changes in that message file for later use. The message description remains in the message file until the file is deleted, until the Remove Message Description (RMVMSGD) command is used to remove the message from the file, or until another change to the message is made with CHGMSGD.

Restriction: To change a message description in a message file, you must have operational rights for the message file and for the library in which the file is to be stored.



①No more than 132 characters can be specified.
②No more than 1435 characters can be specified.
③If either TYPE or LEN is specified as *NONE, the other of the two parameters must also be specified as *NONE.
④VALUES, RANGE, and REL are mutually exclusive; only one of them can be specified.
⑤If any of the parameter values are specified for DMPLST, *JOB is assumed to be part of the value(s).

Job:B,I Pgm:B,I

MSGID Parameter: Specifies the message identifier of the message to be changed. The message identifier must be 7 characters long and in the following format:

```
pppnnnn
```

The first character must be an alphabetic character, followed by two alphanumeric characters, and the last 4 characters must be a decimal number (0001 through 9999).

MSGF Parameter: Specifies the qualified name of the message file in which the message to be changed is stored. (If no library qualifier is specified, *LIBL is used to find the file.) This command ignores any message file overrides in effect for the job.

MSG Parameter: Specifies the first-level message text of the message being changed.

*SAME: The first-level text is to remain as originally defined.

'message-text': This text is the message that is initially displayed, printed, or sent to a program or log. A maximum of 132 characters (enclosed in apostrophes) can be specified, but the display station screen size may cause additional limitations.

Note: If the message text is changing, the *entire* original message text is replaced with the specified change.

One or more substitution variables can be embedded in the message text string to indicate positional replacement fields that allow the program to substitute variable data in the message before the message is sent. The variables must be specified in the form &n, where n is a one-digit number identifying the data field to be substituted. Each variable can be immediately followed by any non-numeric character (such as &2M or &9?), but not by another digit (such as &99). (The variables in the text do not have to be in ascending sequence by these data field identifiers. Also, blanks do not have to precede or follow each variable. The variables can be enclosed in apostrophes if only the variables themselves make up the message. For example, to show a two-part decimal value, the message '&1.&2' can be specified.) The data fields are described positionally in the FMT parameter and are specified positionally in the MSGDTA parameter of the SNDPGMMSG command. Refer to the *CPF Programmer's Guide* for details on substituting data fields in message text.

SECLVL Parameter: Specifies any second-level text to be changed. Second-level text can also be written to the job log, if *SECLVL is specified on the LOG parameter of the job commands.

*SAME: The second-level text is not to be changed.

**NONE:* There is to be no second-level text for this message description. Any second-level text in the original message description will be removed.

'second-level-text': Enter the text to be displayed as second-level text. No more than 1435 characters (enclosed in apostrophes) can be specified, but display limitations must be considered. One or more substitution variables can be embedded in the second-level text, as described in the MSG parameter. If second-level text is changing, the *entire* original second-level text will be replaced with the specified change.

SEV Parameter: Specifies the severity code of the message being changed. The severity code indicates the severity level of the condition that causes the message to be sent.

*SAME: The severity code of this message is not to be changed.

severity-code: Enter a value, 00 through 99, to represent the severity level of this message. The assigned code for the message should correspond to the IBM predefined severity codes. (These codes and their meanings are given in the chart under the SEV parameter, in Appendix A.) Any two-digit value can be entered, even if no severity code has been defined for it (either predefined or user-defined).

FMT Parameter: Specifies the formats of from one to 20 message data fields to be changed. Each field is described in this parameter by a list of attributes. The first nine message data fields can be used as substitution values in the first-level and second-level text messages defined in this message description. All 20 of the fields can be specified in the DMPLST parameter of this command. When specified in the MSGDTA parameter of the SNDPGMMSG command, the data fields must be concatenated to form one character string of no more than 132 characters and must match the format and sequence specified here.

Note: If any of the previously defined formats are to be changed, all existing formats must be included in the FMT parameter. For example, if seven formats had been previously defined and now the third of the seven formats is to be changed from *CHAR 24 to *HEX 8, all seven of the formats (including their types and lengths) must be included in the FMT parameter.

*SAME: The formats of the message are not to be changed.

**NONE:* No format is being described for message fields, or the original formats are to be removed. If *NONE is specified, no references can be made to message data fields in the MSG, SECLVL, or DMPLST parameters.

Note: If FMT had been originally specified, but now FMT(*NONE) is specified, all references to those formats must be removed from the first- and second-level message texts and from the dump list.

type (length[decimal-positions]): A list of attributes defines each message data field (up to a maximum of 20 fields) in this message description. These attributes specify the type of data in the field, the total length of the field, and, optionally, the number of decimal digits to the right of the decimal point. Certain data types do not require a length field. Boundary alignment requirements must be considered (for example, pointers are always aligned on 16-byte boundaries). While 20 fields may be defined, &1 through &9 can appear in the message text; the others can appear only in the dump list.

Type of Message Data: The first value specifies the type of data the substitution field contains and how the data is to be formatted in the message text. The contents of the second and third values vary depending on the type specified. One of the following types can be specified for each field described by this parameter:

**QTDCHAR:* A character string to be formatted (by CPF) with enclosing apostrophes ('Monday, the 1st').

**CHAR:* A character string to be formatted without enclosing apostrophes. It is an alphameric string that can be used to specify a name, for example, BOB. Trailing blanks are truncated.

**HEX:* A string of bytes to be formatted as a hexadecimal value (X'C0F4').

**SPP:* A 16-byte space pointer to data in a space object. When referenced in the DMPLST parameter, the data in the space object (from the offset indicated by the pointer) for the length specified is to be dumped. *SPP is not valid as a replacement field in message text.

**DEC:* A packed decimal number (X'058C') that is formatted in the message as a signed decimal value with a decimal point (58.). Values for length (required) and decimal positions (optional) specified *DEC indicate the number of decimal digits and the number of digits to the right of the decimal point. If the number of decimal positions is not specified, zero is assumed.

**BIN:* A binary value that is either 2 or 4 bytes long (B'0000 0000 0011 1010'), formatted in the message as a signed decimal value (58).

The following formats are valid only in IBM-provided message descriptions and should not be used for other messages.

**DTS:* An 8-byte field that contains a system-date time-stamp. The date in the output message is in the format specified by the system values QDATFMT and QDATSEP. The time is formatted as hh:mm:ss.

**SYP:* A 16-byte system pointer to a system object. When referenced in message text, the simple name of the system object is formatted as described in the name type, *CHAR. When referenced by the DMPLST parameter, the object itself is to be dumped.

**ITV:* An 8-byte field that contains a time interval. The time is formatted in the output message in the form of seconds.

Length of Message Data: After the type specification, a second value (length) can be specified to indicate the number of characters or digits that are passed in the message data. How the second value is used depends upon the type specified in the first value.

- If a length is not specified for *QTDCHAR, *CHAR, *HEX, or *SPP, then *VARY is assumed for the length. If *VARY is assumed, the message data field passed by the SNDPGMMSG command must be preceded by a 2-byte or 4-byte binary field that indicates the actual number of bytes of data being passed. However, when *SPP is specified, the first bytes pointed to by the space pointer contain the field length. Therefore, the 2- or 4-byte field must precede the data pointed to by the space pointer, and not precede the space pointer that is passed as part of the message data.

- If the type *DEC is specified, the total number of decimal digits (including the fraction) must be specified as the second value; the number of digits in the fraction can be specified optionally as the third value.

- If the type *BIN is specified, the message data field can be only 2 or 4 bytes long; the default is 2 bytes.

Length Field Size/Decimal Positions: The third value is used in one of two ways, depending upon the type specified in the first value: (1) if *QTDCHAR, *CHAR, *HEX, or *SPP is specified, and if *VARY is specified or assumed for the second value, the third value is used with *VARY to indicate the size of the length field actually passed. The third value can be either a 2 or a 4, which is the number of bytes specifying the length (in binary) of the passed value; (2) if *DEC is specified, the third value indicates the number of decimal positions in the decimal value. If not specified for a decimal substitution value, the default is 0 decimal positions.

Note: If an object has been damaged or deleted, the substitution variable, when displayed, will not be replaced by the name of the object. Instead, the object will appear as &n (where n = number).

Reply Validity Specification Parameters

If the message is to be sent as an inquiry message or as a notify message (specified by MSGTYPE(*INQ) or MSGTYPE(NOTIFY) on the SNDPGMMSG command) and a reply is expected, seven parameters can be used to specify some requirements that validate the reply received. The seven validity-checking parameters are: TYPE, LEN, VALUES, SPCVAL, RANGE, REL, and DFT.

These parameters are not necessary for a message to allow a reply, but they can be used to define valid replies made to the message. The VALUES, RANGE, and REL are mutually exclusive—only one of them can be specified in this command.

Note: If the reply type or length is changed, and if VALUES, RANGE, or REL had been previously specified, the existing VALUES, RANGE, REL, SPCVAL and DFT must also be changed to be compatible with the new reply type and/or length. If the reply type is changed, LEN must be changed also. If the reply type is changed to *NONE, then LEN and (if they had been coded previously) VALUES, SPCVAL, RANGE, REL, and DFT must be coded as *NONE.

TYPE Parameter: Specifies, only if the message is sent as an inquiry or notify message, the type of valid reply to this message.

SAME: The reply TYPE is not to be changed.

**NONE:* There is to be no reply validity checking. Any existing reply type will be removed. LEN(*NONE) must also be specified.

**CHAR:* Any character string is valid. If it is a quoted character string, the apostrophes are passed as part of the character string.

**DEC:* Only a decimal number is a valid reply.

**ALPHA:* Only an alphabetic (A through Z, $, #, and @) character string is valid. Blanks are not allowed.

**NAME:* Only a simple name is a valid reply. The name does not have to be a CPF object name, but must begin with an alphabetic character; the rest must be alphameric.

LEN Parameter: Specifies, only if the message is sent as an inquiry or notify message, the maximum reply length.

*SAME: The reply length is not to be changed.

**NONE:* There is to be no reply validity checking. The existing LEN specification, if any, will be removed. TYPE(*NONE) must also be specified.

**TYPE:* The maximum length is determined by the type of reply specified in the TYPE parameter. The maximum length for each type of reply is:

- 132 characters for types *CHAR and *ALPHA. If any further validity checking is to be performed (VALUES, RANGE, REL, SPCVAL, or DFT are specified), the maximum length allowed for *CHAR and *ALPHA is 32 characters.
- 15 digits for *DEC, of which a maximum of 9 digits can be to the right of the decimal point.
- 10 alphameric characters for *NAME.

length (decimal-positions): Enter the maximum reply length. The length specified here cannot exceed the maximums shown above. If the reply type is a decimal value, the number of decimal positions can be optionally specified; if a decimal is not specified, zero decimal positions are assumed.

VALUES Parameter: Specifies, only if the message is sent as an inquiry or notify message, a list of values of which one can be received as a valid reply. No more than 20 values can be specified in the list. Each value in the list must meet the requirements specified for message replies by the TYPE and LEN parameters. If VALUES is specified, the RANGE and REL parameters cannot be specified.

*SAME: The existing values list is not to be changed.

**NONE:* No list of reply values is specified. The reply can have any value that is consistent with the other validity-checking parameters. Any existing VALUES will be removed.

value: Enter one or more values, up to a maximum of 20, that, to be valid, must match a reply value sent in response to the message defined in this message description. The maximum length of each value is 32 characters.

SPCVAL Parameter: Specifies, only if the message is sent as an inquiry or notify message, a list of up to 20 sets of special values of which one set (if the from-value is matched by the sent reply) is used as the reply. These values are special in that they may not meet all the validity checking specifications given in the other reply-oriented parameters. The reply sent is compared to the from-value in each set; if a match is found, and a to-value was specified in that set, the to-value is sent as the reply. If no to-value was specified, the from-value is sent as the reply. If the reply sent does not match any from-value, then the reply is validity-checked by the specifications in the other reply-oriented parameters.

*SAME: The special values list is not to be changed.

**NONE:* No special values are specified for the replies to this message. Any existing special values will be removed from the message description.

from-value (to-value): Enter one or more sets of values, up to a maximum of 20 sets, that are used to determine the reply sent to the sender of the message. Each set must have a from-value that the reply is to be compared with, and an optional to-value to be sent as the reply (if its from-value matches the reply).

RANGE Parameter: Specifies, only if the message is sent as an inquiry or notify message, the upper and lower value limits for valid replies to this message. These values must meet the requirements specified for replies by the TYPE and LEN parameters, and both values must be of the same type. If both values are not of the same length, the shorter value is padded on the right with blanks. For type *CHAR and *ALPHA replies, the reply is padded on the right with blanks or truncated on the right (to the length of the specified values) before the value range is validity-checked. If RANGE is specified, the VALUES and REL parameters cannot be specified.

*SAME: The upper and lower range limits are not to be changed.

**NONE:* No range values are specified for the replies to this message. Any existing range values will be removed from the message description.

lower-value upper-value: Enter the lower and upper limit values for valid replies to this message.

REL Parameter: Specifies, only if the message is sent as an inquiry or notify message, the relationship that must exist for a reply to be valid. The value specified must meet the requirements specified for replies by the TYPE and LEN parameters. For replies of the types *CHAR and *ALPHA, the reply is padded on the right with blanks or truncated on the right to match the length of the value specified, before the system performs the test on the reply value sent.

*SAME: The relationship is not to be changed.

**NONE:* No relationship is to be specified for replies to this message. Any existing relationship specifications will be removed from the message description.

operator-value: Enter one of the relational operators and the value against which the message reply is to be checked. If the reply is valid in the relational test, it is sent to the originator of the message. The relational operators that can be entered are:

*LT	Less than
*LE	Less than or equal to
*GT	Greater than
*GE	Greater than or equal to
*EQ	Equal to
*NL	Not less than
*NG	Not greater than
*NE	Not equal to

Note: If VALUES, RANGE, or REL had been specified on the existing message, and they are to be changed to another type of reply validity-checking, then the existing check must be removed by specifying *NONE. For example, if VALUES had been specified originally, but now you want to specify a RANGE, you must specify VALUES(*NONE) and RANGE(x y) in the CHGMSGD command.

DFT Parameter: Specifies, only if the message is sent as an inquiry or notify message, the default reply (enclosed in apostrophes, if it contains special characters) to be used when the receiver of the message has indicated that all messages to him are to use default replies, or when a message is deleted from a message queue and no reply was specified. The default reply can also be used to answer unmonitored notify messages. The default reply must meet the requirements specified for replies by the validity-checking parameters.

*SAME: The default reply is not to be changed.

**NONE:* No default reply is to be specified. Any existing default reply will be removed.

'default-reply': Enter the reply (enclosed in apostrophes) to be used as the default reply.

DFTPGM Parameter: Specifies the name of the default program (if any) to take default action when this message is sent as an escape message to a program that is not monitoring for it. This parameter is ignored if the message is not sent as an escape message. If it is sent as an escape message, the following parameters are passed to the program:

- Program message queue name (10 characters). The name of the program message queue to which the message was sent. (This is the same name as that of the program.)
- Message reference key (4 characters). The message reference key of the escape message on the program message queue.

*SAME: The default program is not to be changed.

**NONE:* No default program is specified for this message. Any existing default program will be removed from the message description.

qualified-default-program-name: Enter the qualified name of the default program to be called when an escape message is sent. (If no library qualifier is given, *LIBL is used to find the default program.)

DMPLST Parameter: Specifies the data to be dumped when this message is sent as an escape message to a program that is not monitoring for it. This parameter can specify that data related to the job be dumped, that data from message data fields be dumped, or that a combination of these be dumped. When data from message data fields is to be dumped, this parameter specifies one or more numbers that positionally identify the data fields to be dumped.

The system objects indicated by system pointers are to be dumped. The data in a space object, indicated by a space pointer, is to be dumped starting from the offset indicated by the space pointer for the length indicated in the field description. The standard job dump can also be requested.

Note: If any of these values are specified for DMPLST, *JOB is assumed to be part of the values. For example, DMPLST(1 2 *JOBDMP) results in the equivalent of DMPLST(*JOB 1 2 *JOBDMP).

*SAME: The dump list is not to be changed.

**JOB:* This value is the equivalent of specifying DSPJOB JOB(*) OUTPUT(*LIST); refer to *DSPJOB (Display Job) Command* for more information.

**JOBDMP:* The data areas of the job are to be dumped as specified by the DMPJOB command. *JOBDMP can be specified by itself, with *JOB, with *JOBINT, or with a list of message data field numbers.

**JOBINT:* The internal machine data structures related to the job execution are to be dumped to the machine error log as specified by the DMPJOBINT command. *JOBINT can be specified by itself, with *JOBDMP, *JOB, or with a list of message data field numbers.

message-data-field-number: Enter the numbers of the message data fields that identify the data to be dumped when this escape message is sent but not monitored. As many as 20 data field numbers can be specified in the list; additionally, the list can contain the values *JOB and *JOBINT.

**NONE:* There is no dump list for this message. Any existing dump list will be removed from the message description.

LOG Parameter: Specifies whether or not the message is to be logged in the system service log, when it is sent as an escape message that is not monitored.

*SAME: The logging specification is not to be changed.

**NO:* The unmonitored escape message is not to be logged in the system service log.

**YES:* Every unmonitored escape message is to be logged in the system service log.

Examples

```
CHGMSGD MSGID(UIN0115) MSGF(INV) +
      MSG('Enter your name')
      SEV(55)
```

This command changes the first-level text and the severity of message UIN0115 stored in the message file INV. The rest of the message description will remain as originally specified in the ADDMSGD command.

As another example, assume you created message UPY0047 as follows:

```
ADDMSGD MSGID(UPY0047) MSGF(TIMECARD.PAYLIB) +
      MSG('Enter department number:') +
      TYPE(*DEC)  LEN(4) +
      VALUES(0816 0727 0319 8774)
```

Now, if you would like to change to a range of valid replies (RANGE parameter), rather than specific reply values (as specified with the VALUE parameter):

```
CHGMSGD MSGID(UPY0047) MSGF(TIMECARD.PAYLIB) +
      VALUES(*NONE) +
      RANGE(0300 8900)
```

The VALUES as originally defined are removed and the RANGE parameters are added to the message description. The type and length of the reply values remain the same.

Note: All changes made to an existing message description must be compatible with the existing message description. For example, the following change would be diagnosed as invalid because the RANGE values are not compatible with the reply length as defined on the original ADDMSGD command.

```
ADDMSGD MSGID(XYZ0202) MSGF(XYZMSGF) +
      MSG('Enter routing code:') +
      TYPE(*CHAR) LEN(2) +
      VALUES(AA BB CC DD EE)

CHGMSGD MSGID(XYZ0202) MSGF(XYZMSGF) +
      VALUES(*NONE) RANGE(AAA ZZZ)
```

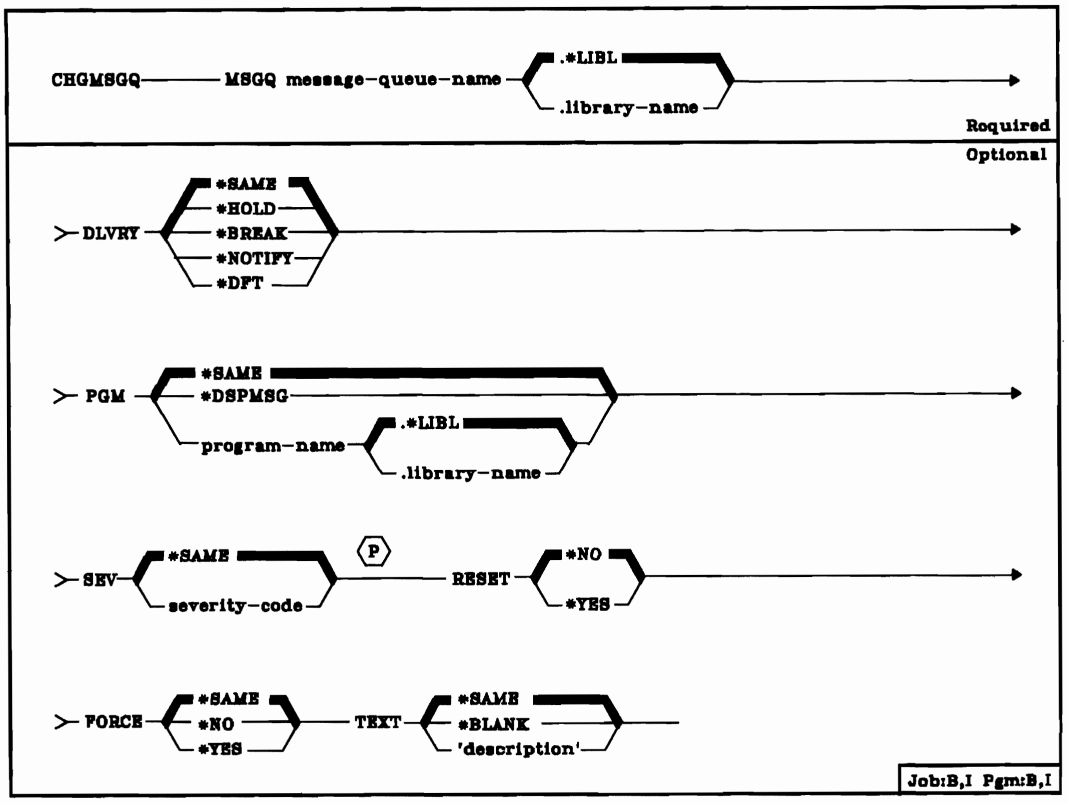
To make the change to the range of reply values valid, you must also change the length (LEN parameter). The correct command coding would be as follows:

```
CHGMSGD MSGID(XYZ0202) MSGF(XYZMSGF) +
      LEN(3) +
      VALUES(*NONE) RANGE(AAA ZZZ)
```

CHGMSGQ (Change Message Queue) Command

The Change Message Queue (CHGMSGQ) command changes the attributes of the specified message queue. If the delivery mode is being changed to *BREAK or *NOTIFY and if the message queue is not allocated to the job in which this command is entered, it is implicitly allocated by this command. (The DLVRY, PGM, and SEV parameters are not contained in the CRTMSGQ command, but default values are assigned to them by the system when the message queue is created.) This command can also be used to reset the status of old messages to new messages so they can be received again without the use of message reference keys.

Restriction: To change the message queue, you must have operational and read rights for the queue.



MSGQ Parameter: Specifies the qualified name of the message queue whose attributes are to be changed. (If no library qualifier is given, *LIBL is used to find the message queue.)

DLVRY Parameter: Specifies how the messages that are sent to this message queue are to be delivered. The method of delivery is in effect only as long as the message queue is allocated to the job. When the queue is deallocated, the delivery mode is changed to *HOLD for work station, system operator, and user message queues. However, if *DFT is specified for a system operator or user message queue, the delivery mode remains *DFT.

Note: Changing the delivery mode to *BREAK or *NOTIFY initiates an attempt to allocate the message queue in the *EXCL (exclusive, no read) lock state. If another job already has the message queue allocated in the *EXCL lock state, the message queue is not allocated to this job and the CHGMSGQ command is not executed.

SAME: The method of message delivery is not to be changed. (If this parameter has not been changed previously in another CHGMSGQ command, *SAME means that *HOLD is the method of delivery, because there is no DLVRY parameter on the CRTMSGQ command.) However, if the specified message queue is a work station message queue, it is automatically changed to *NOTIFY by the system at sign-on. If any messages that meet the notify conditions for the message queue were put on the queue before sign-on, the user is notified immediately.

**HOLD:* The messages are held in the message queue until they are requested by the user or program. The work station user uses the DSPMSG (Display Messages) command to display the messages; a program must issue a RCVMSG (Receive Message) command to receive a message and handle it.

**BREAK:* The job to which the message queue is allocated is interrupted when a message arrives at the message queue, and the program specified in the PGM parameter is invoked; also, if the job is an interactive job, the audible alarm is sounded (if the feature is installed). The delivery mode cannot be changed to *BREAK if the message queue is also being used by another job.

**NOTIFY:* The job to which the message queue is allocated is notified when a message arrives at the message queue. For interactive jobs at a work station, the audible alarm is sounded and the Message Waiting indicator is turned on; at the console, the Attention indicator on the display is turned on and the Attention light is turned on (if the feature is installed on the console). Note that the audible alarm does *not* sound at the console for a notify message. For batch jobs, no notification occurs; the message is simply held in the queue (the same as for *HOLD). The delivery mode cannot be changed to *NOTIFY if the message queue is also being used by another job.

**DFT:* Messages requiring replies are answered with their default reply, and information only messages are ignored.

PGM Parameter: Specifies the name of the program to be called when a message arrives at the message queue and break delivery has been specified. (Because the QSYSOPR message queue receives messages that require manual operator action, only *DSPMSG should be specified or assumed if the message queue being changed is QSYSOPR.) The following parameters are passed to the program:

- Message queue name (10 characters). The name of the message queue to which the message was sent.
- Library name (10 characters). The name of the library containing the message queue.
- Message reference key (4 characters). The reference key of the message sent to the message queue.

*SAME: The same program, if any, is to be called. (If this parameter has not been changed previously in another CHGMSGQ command, *SAME means that *DSPMSG is assumed.)

**DSPMSG:* The Display Message (DSPMSG) command is executed when a message arrives for break delivery. For interactive jobs, the messages are displayed. Also, at the work station, the audible alarm is sounded and the Message Waiting indicator is turned on. At the console, the Attention indicator is turned on and the audible alarm is sounded if the feature is installed on the console. For batch jobs, the message is sent to a spooled printer file.

qualified-program-name: Enter the qualified name of the program that is to be called when a message arrives for break delivery. (If no library qualifier is given, *LIBL is used to find the program.)

SEV Parameter: Specifies the lowest severity code that a message can have and still be delivered to a user in break or notify mode. Messages arriving at the message queue whose severities are lower than that specified here do not interrupt the job or turn on the Attention indicator; they are held in the queue until they are displayed by the DSPMSG command. If *BREAK or *NOTIFY is specified on the DLVRY parameter, and is in effect when a message arrives at the queue, the message is delivered if the severity code associated with the message is equal to or greater than the value specified here. Otherwise, the message is held in the queue until it is requested.

*SAME: The severity code is not to be changed. (If this parameter has not been changed previously in another CHGMSGQ command, *SAME means that the severity code is 00, which is the system default.)

severity-code: Enter a value, 00 through 99, that specifies the lowest severity code that a message can have and still be delivered if the message queue is in break or notify delivery mode. (System messages are shipped with a set of predefined severity codes. These codes and their meanings are given in the chart under the SEV parameter, in Appendix A.) Any two-digit value can be entered, even if no severity code has been defined for it (either predefined or user-defined).

RESET Parameter: Specifies whether old messages (messages that have been received once and were not removed from the message queue) held in the message queue are to be reset to the new message status. The messages can then be received in first-in, first-out (FIFO) order as they were originally. This parameter applies only to receiving messages by a program; it does not affect message displays. If all messages are to be cleared, refer to *RMVMSG (Remove Message) Command.*

*NO: Old messages in the message queue are not to be reset to new message status. To receive an old message, reply to it, or remove it, you must enter the message reference key.

**YES:* All messages in the message queue are to be reset to the new message status. These messages can then be received as new messages in the same order that they were sent to the message queue.

FORCE Parameter: Specifies whether changes made to the message queue description or messages added to or removed from the queue are to be immediately forced into auxiliary storage; this ensures that changes to the queue, or messages sent or received, are not lost if a system failure occurs.

*SAME: The value specified in the referenced message queue is not to be changed.

**NO:* Changes made to the message queue, including its messages, do not have to be immediately forced to auxiliary storage.

**YES:* All changes to the message queue description and to the messages in the queue are to be immediately forced to auxiliary storage.

TEXT Parameter: Specifies the user-defined text that describes the message queue. The text specified here replaces any previous text. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CHGMSGQ  MSGQ(JONES) DLVRY(*NOTIFY)
```

This command changes the method of delivery of the message queue named JONES to notify mode. The user will be immediately notified by the attention light and audible alarm (if installed) when a message has been sent to his queue.

```
CHGMSGQ  MSGQ(INV) DLVRY(*BREAK) PGM(INVUPDT)
```

This command changes the delivery mode of the message queue named INV to *BREAK and calls a program named INVUPDT when a message arrives at INV.

CHGOBJOWN (Change Object Owner) Command

The Change Object Owner (CHGOBJOWN) command transfers object ownership from one user to another. The rights that other users have to the object are not changed. Also, the user profile that no longer owns the object still retains the explicit object authority for the object that it had before the transfer.

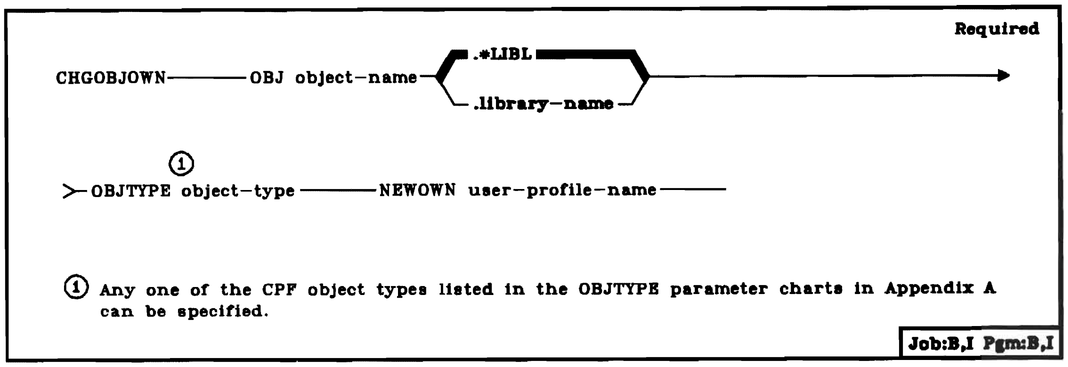
The owner of an object always has all the rights applicable to the object unless they are explicitly revoked. As the owner, he has the authority to grant any rights to any user for his objects. He can also grant to himself any rights that were explicitly revoked previously. The owner may, for example, revoke some of his specific rights as a precautionary measure, and then, when the need arises, he can again grant those same rights to himself.

To transfer ownership, any user (including the object's present owner) must have:

- Object existence rights for the object
- Add rights for the new owner's user profile
- Delete rights for the present owner's user profile

The security officer has complete authority for all objects; therefore, he can transfer the ownership of any object. All users have add and delete rights for their own user profiles; that is, a user can add objects to or delete objects (that he created) from his own user profile by transferring the ownership of the object.

Restrictions: (1) For any program that is created with USRPRF(*OWNER) specified on the command that creates it, only the security officer or a program that executes under the security officer's user profile can transfer the program's ownership. (2) Before this command can be used to change the owner of a device, control unit, or line description, its associated device, control unit, or line must be varied on. (3) For display work stations, if this command is not entered at the device whose ownership is being changed, this command should be preceded by the ALCOBJ command and followed by the DLCOBJ command.



① Any one of the CPF object types listed in the OBJTYPE parameter charts in Appendix A can be specified.

OBJ Parameter: Specifies the qualified name of the object that is being assigned to the new owner. (If no library qualifier is given, *LIBL is used to find the specified object.) The library name can be entered to ensure that the correct object changes ownership.

OBJTYPE Parameter: Specifies the object type of the object whose ownership is being transferred. Enter the predefined value that specifies the object type. (For an expanded description of the OBJTYPE parameter and a list of the valid values for the CPF object types, see Appendix A.)

NEWOWN Parameter: Specifies the name of the user to whom the object is being assigned. Enter the user profile name under which the new owner is enrolled on the system.

Example

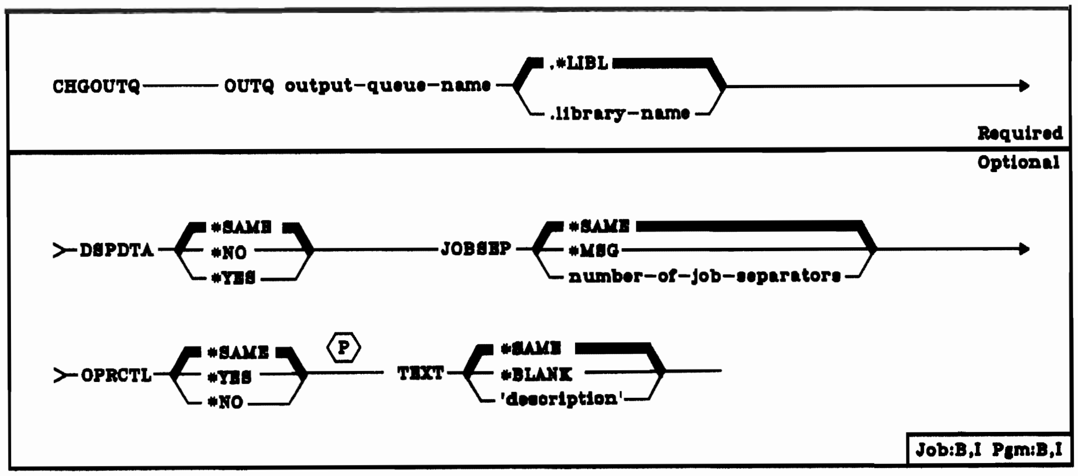
```
CHGOBJOWN  OBJ(PROGRAM1.USERLIB) OBJTYPE(*PGM) +
     NEWOWN(ANN)
```

This command assigns ownership of the program named PROGRAM1, located in the user library named USERLIB, to the user named ANN.

CHGOUTQ (Change Output Queue) Command

The Change Output Queue (CHGOUTQ) command changes the attributes of the specified output queue. The attributes of the output queue, except the number of job separators, can be changed while a writer is producing spooled files from the output queue.

Restriction: To change an output queue's attributes, you must either be the queue's owner or you must have object management, read, add, and delete rights for the queue.



OUTQ Parameter: Specifies the qualified name of the output queue that is to have its attributes changed. (If no library qualifier is given, *LIBL is used to find the output queue.)

DSPDTA Parameter: Specifies whether users that have authority to read the output queue can display the data from any output file on the queue or only data from their own files.

*SAME: The current value of the display data attribute that is specified for the output queue is not to be changed.

_*NO:_ Users authorized to use the queue can display the output data of their own files only; they cannot display the output data of another user's file on the queue.

*YES: Any user with read rights for the output queue can display the output of any file on the queue.

JOBSEP Parameter: Specifies, for each job having spooled file entries on this output queue, the number of separators to be placed at the beginning of the output for each job. Each separator (card or printer page) contains information that identifies the job such as its name, the job user's name, the job number, and the time and date when the job was executed. The number of separators can be from zero through nine. This parameter can only be changed when the output queue is not being processed by a writer.

*SAME: The number of job separators is not to be changed.

**MSG:* No job separators are to be placed before each job's output. A message is sent to a message queue notifying the operator of the end of each job. The message queue receiving the message is identified by the MSGQ parameter of the Start Writer command.

number-of-job-separators: Enter the new number (0 through 9) of separators to be placed before the output of each job.

OPRCTL Parameter: Specifies whether a user with job control rights is allowed to control and make changes to spooled output files with entries on this output queue. A user has job control authority if SPCAUT(*JOBCTL) is specified in his user profile.

*SAME: The current value specified for the operator control attribute of the output queue is not to be changed.

**YES:* A user with job control rights can control the queue and make changes to the entries on the queue.

**NO:* This queue and its entries cannot be manipulated or changed by a user with job control rights unless he also has object management rights, and read, add, and delete rights for the queue.

TEXT Parameter: Specifies the user-defined text that describes the output queue. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

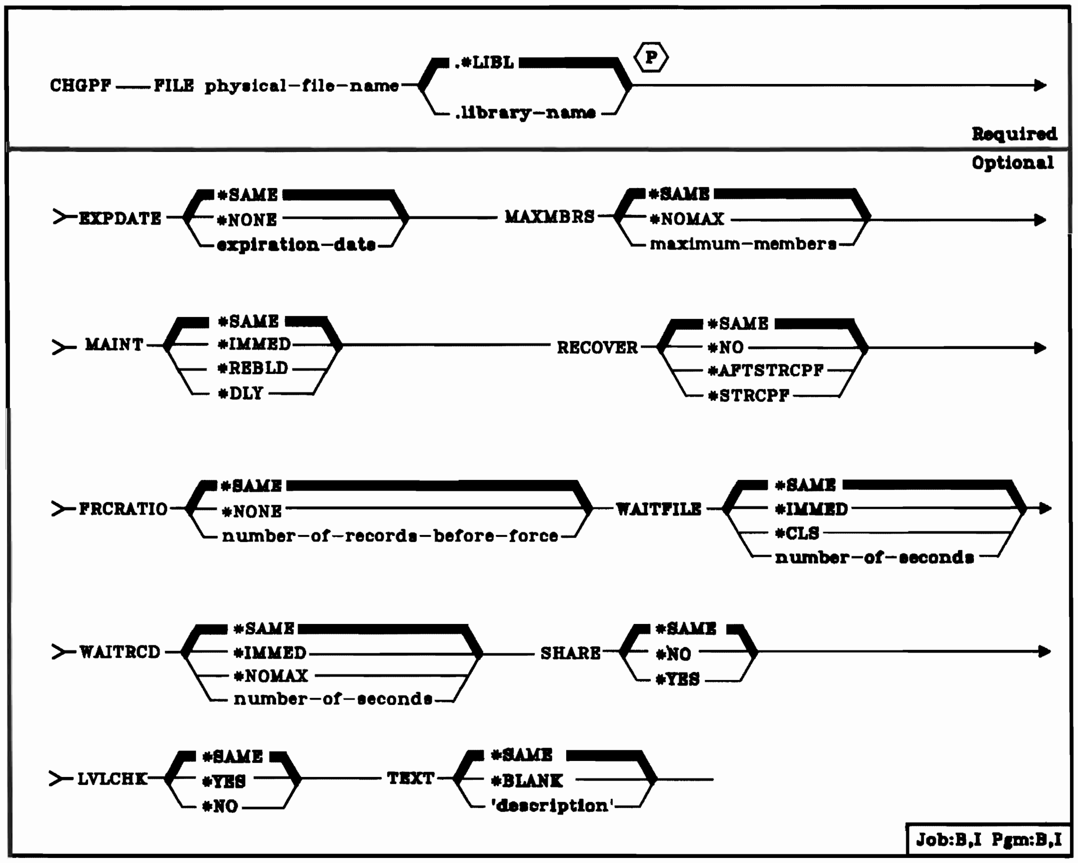
```
CHGOUTQ  OUTQ(QPUNCH)  JOBSEP(4)  TEXT('Default queue +
      for all 96-col cards')
```

This command changes the number of job separators and the text that describes the output queue named QPUNCH. Four job separator cards are to precede all of the spooled output for each job produced from the QPUNCH output queue.

CHGPF (Change Physical File) Command

The Change Physical File (CHGPF) command changes the attributes of a physical file and all its members. The changed attributes will be used for all members subsequently added to the file. To change the attributes of a specific member, execute the CHGPFM (Change Physical File Member) command.

Restrictions: To change a physical file, you must have object management and operational rights for the file and read rights to the library. In order for you to change the file, an exclusive-no-read lock is necessary, which means no one may be using the file for any purpose.



FILE Parameter: Specifies the qualified name of the physical file to be changed. (If no library qualifier is given, *LIBL is used to find the file.)

EXPDATE Parameter: Specifies the expiration date of all the file's members. Any attempt to open a file member that has expired causes an error message to be sent to the user. (The RMVM command is used to remove the member.) If EXPDATE is specified, all members in the file will be changed. The expiration date must be later than or equal to the current day's date. An expired member may be changed to non-expired by changing the EXPDATE parameter.

*SAME: The expiration date of the file is not to be changed.

**NONE:* The member has no expiration date.

expiration-date: Enter the date after which the member should not be used. The date must be specified in the format defined by the system values, QDATFMT and QDATSEP. The date must be enclosed in apostrophes if special characters are used in the format.

MAXMBRS Parameter: Specifies the maximum number of members that the physical file can have at any time. The maximum number of members specified must be greater than or equal to the current number of members in the file.

*SAME: The maximum number of members in the file is not to be changed.

**NOMAX:* No maximum is specified for the number of members; the system maximum of 32 767 members per file is used.

maximum-members: Enter the value for the maximum number of members that the physical file can have. A value of 1 through 32767 is valid.

MAINT Parameter: Specifies the type of access path maintenance to be used for all members of the physical file. This parameter is valid only if a keyed access path is used.

Only the following changes to a file's access path maintenance are allowed: *REBLD to *IMMED (if the file was originally created as *IMMED or *REBLD), *IMMED to *REBLD, *DLY to *REBLD, and *REBLD to *DLY (if the file was originally created as *DLY).

Existing MAINT Value	CHGPF MAINT Parameter Value		
	*REBLD	*DLY	IMMED
*REBLD	N/A	Note 1	Note 2
*DLY	YES	N/A	NO
*IMMED	YES	NO	N/A

Notes:
1. Allowed only if file was originally created with MAINT(*DLY).
2. Allowed only if file was originally created with MAINT(*IMMED) or MAINT(*REBLD).

<u>*SAME</u>: The access path maintenance of the file is not to be changed.

**IMMED:* The access path is to be continuously (immediately) maintained for each physical file member. The path is updated each time a record is changed, added to, or deleted from the member. The records can be changed through a logical file that uses the physical file member regardless of whether the physical file is opened or closed. *IMMED must be specified for all files requiring unique keys to ensure uniqueness in all inserts and updates.

**REBLD:* The access path is to be rebuilt when a file member is opened during program execution. The access path is continuously maintained until the member is closed; the access path maintenance is then terminated. *REBLD is not valid for access paths that are to contain unique key values.

**DLY:* The maintenance of the access path is to be delayed until the member is opened for use. The access path is then updated only for records that have been added, deleted, or updated since the file was last closed. (While the file is open, all changes made to based-on members are immediately reflected in the access paths of the opened files members, no matter what is specified for MAINT.) To prevent a lengthy rebuild time when the file is opened, *DLY should be specified only when the number of changes to the access path between a close and the next open are small (when key fields in records for this access path change infrequently). *DLY is not valid for access paths that require unique key values.

If the number of changes saved reaches approximately 10 per cent of the access path size, the system will stop saving changes and the access path will be completely rebuilt the next time the file is opened.

RECOVER Parameter: Specifies, for files having immediate or delayed maintenance on their access paths, when recovery processing of the file is to be performed if a system failure occurred while the access path was being changed.

An access path having immediate or delayed maintenance can be rebuilt during start CPF (before any user can execute a job), or after start CPF has finished (during concurrent job execution), or when the file is next opened. While the access path is being rebuilt, the file cannot be used by any job. For more information on recovery processing, refer to the *CPF Programmer's Guide*.

An access path having rebuild maintenance will be rebuilt the next time its file is opened, the time that it normally is rebuilt.

This parameter is valid only if a keyed access path is used.

*SAME: The recovery attribute of the file is not to be changed.

**NO:* The access path of the file is not to be rebuilt. The file's access path is rebuilt when the file is next opened.

**AFTSTRCPF:* The file is to have its access path rebuilt after the start CPF operation has been completed. This option allows other jobs not using this file to begin processing immediately after the CPF has been started. If a job tries to allocate the file while its access path is being rebuilt, a file open exception occurs if the specified wait time for the file is exceeded.

**STRCPF:* The file is to have its access path rebuilt during the start CPF operation. This ensures that the file's access path will be rebuilt before the first user program tries to use it; however, no jobs can begin execution until after all files that specify RECOVER(*STRCPF) have their access paths rebuilt.

FRCRATIO Parameter: The force write ratio parameter specifies the number of inserted, updated, or deleted records that are processed before they are forced to auxiliary (permanent) storage. (For an expanded description of the FRCRATIO parameter, see Appendix A.)

If the physical file is being journaled, a larger force write ratio or *NONE may be specified. Refer to the *CPF Programmer's Guide* for more information on the Journal Management Facility.

*SAME: The force write ratio of the file is not to be changed.

**NONE:* There is no force write ratio; the system determines when the records are written in auxiliary storage.

number-of-records-before-force: Enter the number of new or changed records that are processed before they are explicitly forced into auxiliary storage.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*SAME: The wait attribute of the file is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

WAITRCD Parameter: Specifies the number of seconds that the program is to wait for a record that is to be updated or deleted. If the record cannot be allocated in the specified wait time, an error message is sent to the program.

*SAME: The record wait attribute of the file is not to be changed.

**IMMED:* The program is not to wait; when a record is locked, an immediate allocation of the record is required.

**NOMAX:* The wait time will be the maximum allowed by the system (32 767 seconds).

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether an ODP (open data path) to the physical file member is to be shared with other programs in the same job. When an ODP is shared, the programs accessing the file share such things as the position being accessed in the file, the file status, and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record. If SHARE is specified, all members in the file will be changed.

*SAME: The ODP sharing value of the member is not to be changed.

**NO:* An ODP created by the program when the file member is opened is not to be shared with other programs in the job. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* The same ODP is to be shared with each program in the job that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the record format identifiers are to be level checked to verify that the current record format identifier is the same as that specified in the program that opens the physical file. This value can be overridden on the OVRDBF command at execution time.

*SAME: The level check value of the member is not to be changed.

**YES:* The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not match, an error message is sent to the program requesting the open, and the file is not opened.

**NO:* The level identifiers are not to be checked when the file is opened.

TEXT Parameter: Enter text that briefly describes the physical file member. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text that describes the member is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

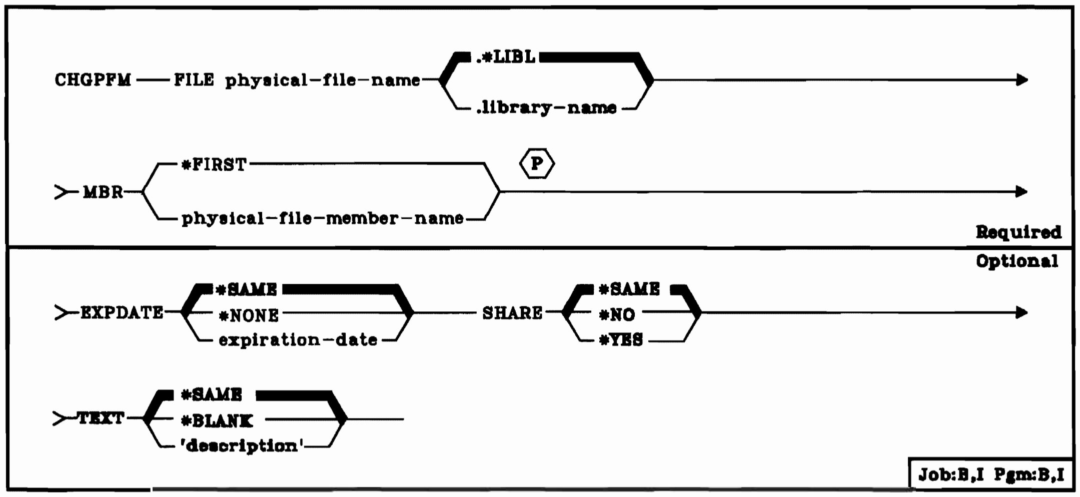
```
CHGPF  FILE(INV.QGPL) EXPDATE('10/31/87')
```

This command changes the expiration date for all members in physical file INV to October 31, 1987.

CHGPFM (Change Physical File Member) Command

The Change Physical File Member (CHGPFM) command changes the attributes of a physical file member.

Restrictions: To change a physical member, you must have object management and operational rights for the physical file that contains the member, and read rights to the file library. In order for you to change the member, no other user may be clearing or initializing the member, nor may any user be holding the file for exclusive use. Concurrent users may have the member open, but the changes made to the member will not be reflected in any open members. In order for the changes in open members to be effective, you must first close the member (this must be a full close if the member is open SHARE(*YES)) and open it again.



FILE Parameter: Specifies the qualified name of the physical file that contains the member to be changed. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name of the member, or the first member (*FIRST) to be changed.

<u>*FIRST:</u> The first member of the specified physical file is to be changed.

physical-file-member-name: Enter the name of the physical file member to be changed.

EXPDATE Parameter: Specifies the expiration date of the member. Any attempt to open a file member that has expired causes an error message to be sent. (The RMVM command is used to remove the member.) An expired member may be changed to non-expired by changing the EXPDATE parameter. The expiration date must be later than or equal to the current day's date.

<u>*SAME:</u> The expiration date of the member is not to be changed.

**NONE:* The member has no expiration date.

expiration-date: Enter the date after which the member should not be used. The date must be specified in the format defined by the system values, QDATFMT and QDATSEP. The date must be enclosed in apostrophes if special characters are used in the format.

SHARE Parameter: Specifies whether an ODP (open data path) to the physical file member is to be shared with other programs in the same job. When an ODP is shared, the programs accessing the file share such things as the position being accessed in the file, the file status, and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

<u>*SAME:</u> The ODP sharing value of the member is not to be changed.

**NO:* An ODP created by the program when the file member is opened is not to be shared with other programs in the job. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* The same ODP is to be shared with each program in the job that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Enter text that briefly describes the physical file member. (For an expanded description of the TEXT parameter, see Appendix A.)

**SAME:* The members text should not be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CHGPFM  FILE(INV.QGPL)  MBR(FEB)  EXPDATE('10/31/87')
```

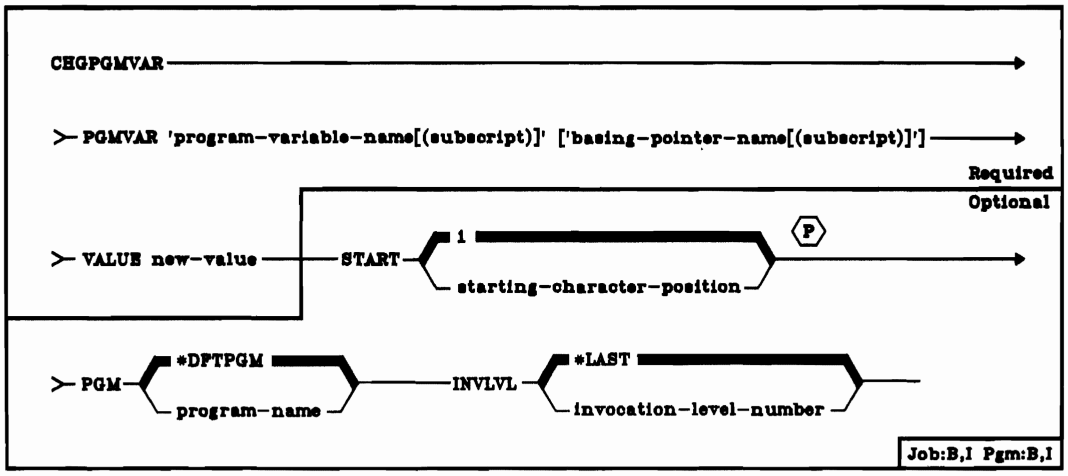
The member named FEB in the physical file INV that is stored in the QGPL library is to be changed so that the expiration date of the member will now be October 31, 1987.

CHGPGMVAR (Change Program Variable) Command

The Change Program Variable (CHGPGMVAR) command, used only in debug mode, changes the value of a variable in a program being debugged. Only character and numeric variables can be changed. Depending upon whether the variable to be changed is in static or automatic storage, the duration of the change varies. For a static variable, the change lasts for the duration of the program's activation. For an automatic variable, the change lasts until the invocation of the program is terminated.

A portion of a character string can be changed; the length of the data being changed is the length of data specified in the VALUE parameter.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*



PGMVAR Parameter: Specifies the name of the program variable to be changed in an HLL (high-level language) or MI (machine interface) program.

'program-variable-name': Enter the name of the program variable to be changed. If the variable name contains special characters (such as the & in a CL variable name), it must be enclosed in apostrophes. An example is: PGMVAR('&VAR2')

An RPG indicator or an MI ODV (object definition table directory vector) number can be specified instead of a program variable name. An example of an RPG indicator is: PGMVAR('*IN22'). The ODV number must be preceded by a slash: PGMVAR('/264') for example.

COBOL-qualified program variable names may be specified in this parameter. These names have the following syntax:

```
var-name-1 OF/IN var-name-2 OF/IN varname-3...varname-N
```

where varname-N is the last possible variable name that will fit into the input field of the PGMVAR parameter. The input field length for each variable in the PGMVAR parameter is 98 characters. The subscript specified for a qualified variable name may also be a qualified variable name. A qualified variable name (or one with a subscript), including blanks and parentheses, must be contained within the 98-character limit. The 98-character limit includes the necessary keywords (OF/IN) and blanks, but does not include the enclosing apostrophes.

'program-variable-name(subscript)': For variables in an array, enter the name of the variable and the subscript representing the positional element in the array that is to be changed. The subscript must be enclosed in parentheses, and the variable name and subscript number must be enclosed in apostrophes. An example is: PGMVAR('A(5)')

Either an integer or another variable name can be specified for each subscript.

For COBOL-qualified program variable names, any combination of variable name length and subscript length that will fit into the 98-character limit is used. For example, one qualified variable name 98 characters in length (including the keywords OF or IN) can be used with no subscript, or a one-character variable name may be used with a qualified variable name (used as a subscript that uses the other 97 spaces, including parentheses).

For COBOL, the following apply:

- Variable names used in qualifying strings must be high-level language variable names (qualification with ODVs is not allowed).
- Either keyword (OF or IN) is allowed.
- Each OF/IN keyword must be separated from adjacent variable names by at least one blank.
- A qualified variable name can be used as a variable subscript.
- The order the variable names are specified must be from the lowest to the highest levels in the structure.
- Structure levels may be skipped; enough levels must be specified, however, to uniquely identify the variable.
- Qualified variable names must be enclosed in apostrophes, since they contain blank characters.

['basing-pointer-name[(subscript)]']: This set of values in the PGMVAR parameter applies only to MI or HLL programs that support based-on variables. The values can optionally be used with either of the previous two choices to also specify the value in an array that is based on a pointer. The same description of the coding syntax applies here. An example is:

```
PGMVAR('VAR1(5)' 'PTR1(5)')
```

The field length for the basing pointer name is 24 characters.

VALUE Parameter: Specifies the new value of the program variable. Depending on the type specified for the variable in a DCL command, its value must be specified according to the following rules:

- The value for a character variable must be enclosed in apostrophes if it contains special characters or numeric characters (for example, 'ABC 67', which contains a blank and numeric characters, or '37.92', which contains a decimal point and numeric characters).

- The value for a decimal variable can be coded with or without a decimal point (. or ,), and with or without a plus or minus sign. If a negative value is to be specified, it must be preceded by a minus (-) sign. If a decimal point is not entered in the coded value, it is assumed to be on the right of the last digit entered; that is, the coded value is assumed to be an integer (whole number) only. If the number of either integer or fractional digits entered exceeds the defined number of integer or fractional digits, an error occurs.

  If, for example, a decimal variable is defined as a five-position decimal value of which two positions are the fraction portion, the following values can be coded:

Coded Value	Assumed Value
2.7 or 2,7	2.70
27 or 27.00	27.00
-27	-27.00

- Values for all variable types can be entered in hexadecimal form (X'058C' for packed decimal 58). However, if decimal values are entered in hexadecimal form, care should be used because no validity checking is performed on the hexadecimal string.

START Parameter: Specifies the starting position of the value in the program variable that is changed. This parameter is valid only if the program variable is a character string.

1: The first position of the program variable is the starting position in the string to be changed.

starting-character-position: Enter the position number within the program variable that specifies the first character to be changed in the string.

PGM Parameter: Specifies the name of the program that contains the program variable whose value is to be changed.

*DFTPGM: The program previously specified as the default program contains the variable to be changed.

program-name: Enter the name of the program that contains the variable to be changed. The same name (in qualified form) must already have been specified in the ENTDBG or ADDPGM command.

INVLVL Parameter: Specifies which invocation level of the program contains the variable whose value is to be changed. Changes made to static variables automatically affect all invocations. Invocation level 1 is the first (or earliest) invocation of the program, invocation level 2 is the second invocation, and so on down to the last (most recent) invocation level in the stack. For example, if program A calls program B, then program B calls program A, a new invocation of program A is formed. If the first invocation of program A contains the variable to be changed, INVLVL(1) must be specified.

*LAST: The last (most recent) invocation of the specified program has the variable to be changed.

invocation-level-number: Enter the number of the invocation level of the program that has the variable whose value is to be changed.

Examples

```
DCL  VAR(&AMT)  TYPE(*DEC)  LEN(5 2)
       •
       •
CHGPGMVAR  PGMVAR('&AMT')  VALUE(16.2)
```

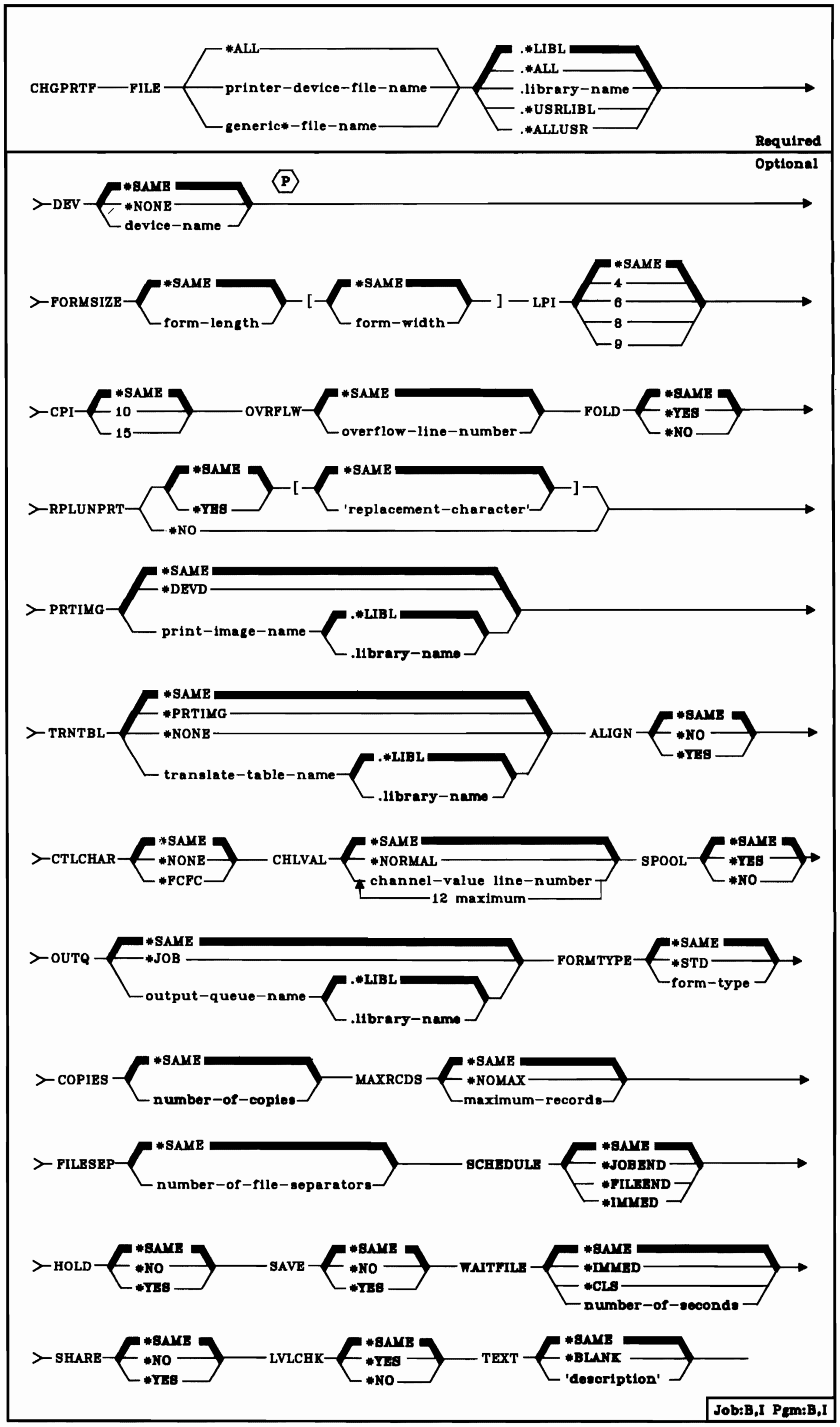
The first command, which is used in a CL program, declares the CL variable &AMT as a five-position decimal value having a three-digit integer and a two-digit fraction. The CHGPGMVAR command (entered in debug mode) is used to change the value of &AMT to 16.20. If VALUE is coded as 16 or 16.00, the value accepted is 16.00; if -16 is coded, the value accepted is -16.00. However, if 1600 is coded, an error occurs because the system assumes that, if no decimal point is coded, it is always on the right of the last digit coded.

```
CHGPGMVAR  PGMVAR(PARTNO)  VALUE('56')  START(4)
```

This command changes, starting in position 4, the program variable PARTNO to 56. Because the START parameter is specified, PARTNO must be a character variable. Because PARTNO is a character variable, the numeric value must be enclosed in apostrophes.

CHGPRTF (Change Printer File) Command

The Change Printer File (CHGPRTF) command changes, in the file description, one or more of the attributes of the specified printer device file.


CHGPRTF
FILE
*ALL
printer-device-file-name
generic*-file-name
.*LIBL
.*ALL
.library-name
.*USRLIBL
.*ALLUSR
Required
Optional
DEV
*SAME
*NONE
device-name
P
FORMSIZE
*SAME
form-length
[
*SAME
form-width
]
LPI
*SAME
4
6
8
9
CPI
*SAME
10
15
OVRFLW
*SAME
overflow-line-number
FOLD
*SAME
*YES
*NO
RPLUNPRT
*SAME
*YES
[
*SAME
'replacement-character'
]
*NO
PRTIMG
*SAME
*DEVD
print-image-name
.*LIBL
.library-name
TRNTBL
*SAME
*PRTIMG
*NONE
translate-table-name
.*LIBL
.library-name
ALIGN
*SAME
*NO
*YES
CTLCHAR
*SAME
*NONE
*FCFC
CHLVAL
*SAME
*NORMAL
channel-value line-number
12 maximum
SPOOL
*SAME
*YES
*NO
OUTQ
*SAME
*JOB
output-queue-name
.*LIBL
.library-name
FORMTYPE
*SAME
*STD
form-type
COPIES
*SAME
number-of-copies
MAXRCDS
*SAME
*NOMAX
maximum-records
FILESEP
*SAME
number-of-file-separators
SCHEDULE
*SAME
*JOBEND
*FILEEND
*IMMED
HOLD
*SAME
*NO
*YES
SAVE
*SAME
*NO
*YES
WAITFILE
*SAME
*IMMED
*CLS
number-of-seconds
SHARE
*SAME
*NO
*YES
LVLCHK
*SAME
*YES
*NO
TEXT
*SAME
*BLANK
'description'
Job:B,I Pgm:B,I

FILE Parameter: Specifies the qualified name of the printer device file whose description is being changed. A generic printer device file name may be specified. For more information on changes made to files with generic filenames, refer to Appendix A.

DEV Parameter: Specifies, for *nonspooled* output only, the name of the printer that is to be used with this printer file to produce printed output. The device name of the IBM-supplied printer device description is QSYSPRT. If System/38 has two system printers attached, another printer device description named QSYSPRT2 is also provided. If SPOOL(*YES) is specified, this parameter is ignored.

SAME: The device name, if any, specified in the device file description is not to be changed.

**NONE:* No device name is to be specified. It can be specified later on an OVRPRTF command, another CHGPRTF command, or in the HLL (high-level language) program that opens the file.

device-name: Enter the name of the device that is to be used with this printer file. The device name must already be known on the system via a device description.

FORMSIZE Parameter: Specifies the length and width of the printer forms to be used by this device file. The length is in lines per page, and the width is in print positions (characters) per line.

SAME: The length specified in the printer file description is not to be changed.

form-length: Enter the form length (in print lines per page) that is to be used by this device file. Although a value of 1 through 255 can be specified as the form length, the value specified should not exceed the actual length of the forms used. The following chart shows the number of lines per page that are valid for each printer type, depending on whether 6 or 8 lines per inch is specified in the LPI parameter for the 3203, 3262, and 5211 Printers, or is manually set on the 5256 Printer. For 5224 and 5225 Printers, 4, 6, 8, or 9 lines per inch can be specified.

	Lines per Page			
Printer	**4 lines/inch**	**6 lines/inch**	**8 lines/inch**	**9 lines/inch**
3203	–	2-144	2-192	–
3262 5211	–	2-84	2-112	–
5224 5225	1-255	1-255	1-255	1-255
5256	–	1-255	1-255	–

*SAME: The width specified in the printer file description is not to be changed.

form-width: Enter the form width (in characters per printed line) that is to be used by this device file. Valid values for the 3203, 3262, 5211, and 5256 Printers are 1 through 132. Valid values for the 5224 and 5225 Printers are 1 through 198. The value specified should not exceed the actual width of the forms used.

LPI Parameter: Specifies the line spacing setting on the printer, in lines per inch, to be used by this device file. The line spacing on the 5256 Work Station Printer must be set manually.

*SAME: The printer line spacing specified in the device file description is not to be changed.

4: The line spacing on the printer is to be 4 lines per inch. This spacing is valid for only the 5224/5225 Work Station Printers.

6: The line spacing on the printer is to be 6 lines per inch.

8: The line spacing on the printer is to be 8 lines per inch.

9: The line spacing on the printer is to be 9 lines per inch. This spacing is valid for only the 5224/5225 Work Station Printers.

CPI Parameter: Specifies the printer character density, in characters per inch, to be used by this device file.

*SAME: The character density specified in the printer device file description is to be used.

10: Character density is to be 10 characters per inch.

15: Character density is to be 15 characters per inch. This density is valid only for 5224/5225 Printers.

OVRFLW Parameter: Specifies the line number on the page when overflow to a new page is to occur. Generally, after the specified line is printed, the printer overflows to the next page before printing continues. Refer to the *CPF Programmer's Guide* for details about controlling page overflow.

*SAME: The line number (after which the printer overflows to a new page) that is specified in the printer file description remains the same.

overflow-line-number: Enter the line number of the line that causes page overflow after the line is printed. The value specified must not exceed the form length specified for the file (FORMSIZE parameter).

FOLD Parameter: Specifies whether all positions in a record are to be printed when the record length exceeds the form width (specified by the FORMSIZE parameter). When folding is specified and a record exceeds the form width, any portion of the record that cannot be printed on the first line will be continued (folded) on the next line or lines until the entire record has been printed.

SAME: The same value specified in the printer file description is to be used.

*YES: Records whose length exceeds the form width are to be folded on the following line(s).

*NO: Records are not to be folded; if a record is longer than the form width, only the first part of the record that fits on one line will be printed.

RPLUNPRT Parameter: The replace unprintable character parameter specifies (1) whether unprintable characters are to be replaced and (2) which substitution character (if any) is to be used. An *unprintable* character is a character that is not on the print belt or train, or in the print image used by the printer.

For 5224, 5225, and 5256 Printers, one of the following occurs when an unprintable character is encountered:

- If you specify RPLUNPRT(*YES), the specified substitution character is printed in place of each unprintable character.
- If you specify RPLUNPRT(*NO) and the value of the unprintable character is hex 00 through hex 3F, or is hex FF, undesirable results may occur. Most characters in this range cause an unrecoverable error to be signaled by the printer, and either the file is held for spooling or it is not processed. Some characters in this range, however, control forms movement and character representation on the printer. If the unprintable character is one of these control characters, additional spacing or skipping may occur. If control characters are specifically placed in the data, other system functions (such as the displaying or copying of a spooled file, or restarting or backing up of a print writer) may cause unpredictable results.
- If you specify RPLUNPRT(*NO) and the value of the unprintable character is in the range of hex 40 through hex FE, a recoverable error is signaled by the device and an inquiry message is sent to the operator, informing him of the error and giving him the chance to cancel the file or to continue processing. If the continue option is selected, subsequent unprintable characters will appear as blanks in the output, and no further inquiry messages will be sent to the operator.

For 3203, 3262, and 5211 Printers, the following occurs when an unprintable character is encountered:

- If you specify RPLUNPRT(*YES) and the value of the unprintable character is in the range of hex 00 through hex 3F, or is hex FF, the specified substitution character is printed instead. If no substitution character was specified, the blank is used. If no characters in this range are expected to be in the data to be printed, *NO can be specified for this parameter to gain some performance improvement. However, if *NO *is* specified and an unprintable character in this range does occur, the only recovery is to rerun the job.

- If you specify RPLUNPRT(*YES) and the value of the unprintable character is in the range of hex 40 through hex FE, a translate table should be used to translate unprintable characters to different printable characters; each unprintable hex value can be translated to its own printable character. The translate table, which is specified by the TRNTBL parameter, should also match the print image used by the printer.

- If you specify RPLUNPRT(*NO) and the value of the unprintable character is hex 00 through hex 3F, undesirable results may occur. Most characters in this range cause an unrecoverable error to be signaled by the printer, and either the file is held for spooling or it is not processed. Some characters in this range, however, control forms movement and character representation on the printer. If the unprintable character is one of these control characters, additional spacing or skipping may occur. If control characters are specifically placed in the data, other system functions (such as the displaying or copying of a spooled file, or restarting or backing up of a print writer) may cause unpredictable results.

- If you specify RPLUNPRT(*NO) and the value of the unprintable character is in the range of hex 40 through hex FE, a recoverable error is signaled by the device and a notify message is sent to the program. If you choose to continue processing or if the message is unmonitored, the error will be ignored and processing will continue. Subsequent unprintable characters will appear as blanks in the output, and no further inquiry messages will be sent to the program.

*SAME: The value specified in the printer file description remains the same concerning whether a message is sent when an unprintable character is detected.

**YES:* Unprintable characters are to be replaced. The program is not notified when unprintable characters are detected.

**NO:* Unprintable characters are not to be replaced. When an unprintable character is detected, a message is sent to the program.

*SAME: The same substitution character specified in the printer file description is to be used when an unprintable character is detected and *YES is specified.

'replacement-character': If *YES is also specified in this parameter, enter the substitution character that is to be used each time an unprintable character is detected. Any printable EBCDIC character can be specified.

PRTIMG Parameter: Specifies, for 3203, 3262, and 5211 Printers only, the name of the print image to be used by this printer device file.

*SAME: The same print image specified in the device file description is to be used.

**DEVD:* The standard print image for the printer (specified in the device description) is to be used.

qualified-print-image-name: Enter the qualified name of the print image to be used by this device file. (If no library qualifier is given, *LIBL is used to find the print image.)

TRNTBL Parameter: Specifies, for 3203, 3262, and 5211 Printers only, the name of the translate table (if any) to be used by this device file when the output data is to be translated before it is printed. The translate table is used to convert each unprintable character having a hexadecimal code of 40 through FE to the printable character specified in the table that is also on the print belt or train. Each hexadecimal code can specify a different printable character.

For each IBM-supplied print image shipped with the system, a matching translate table is also supplied; the name of the table is the same as the name of the image.

*SAME: The translate table name, if any, specified in the printer file description is not to be changed.

**PRTIMG:* The translate table with the same qualified name as the print image is to be used.

**NONE:* No translation is needed when this device file is used.

qualified-translate-table-name: Enter the qualified name of the translate table to be used by this device file and the 3203, 3262, or 5211 Printer. (If no library qualifier is given, *LIBL is used to find the translate table.)

ALIGN Parameter: Specifies, for *nonspooled* output only, whether the forms must be aligned in the printer before printing is started. If ALIGN(*YES) and SPOOL(*NO) are specified, and forms alignment is required, the system sends a message to the QSYSOPR message queue (or any message queue specified for 5224, 5225, and 5256 Printers), and waits for a reply to the message. This parameter is ignored if SPOOL(*YES) is specified. (For spooled output, the message is sent to the message queue specified on the STRPRTWTR command whenever the spooling writer is started and whenever the forms are to be changed.)

SAME: The same value specified in the printer file description is to be used.

**NO:* No forms alignment is required.

**YES:* The forms are to be aligned before the output is printed.

CTLCHAR Parameter: Specifies whether the printer device file will support input with print control characters. Any invalid control characters that are encountered will be ignored, and single spacing is assumed.

SAME: The specification for the control characters is to remain as originally defined.

**NONE:* No print control characters will be passed in the data to be printed.

**FCFC:* Specifies that the first character of every record will contain an ANSI forms control character. If *FCFC is specified, the record length must include one position for the first-character forms-control code. This value is not valid for externally described printer files; that is, SRCFILE(*NONE) was specified on the Create Printer File (CRTPRTF) command.

CHLVAL Parameter: Specifies a list of channel numbers with their assigned line numbers. Use this parameter only if CTLCHAR(*FCFC) has been specified.

Note: If one or more channel-number/line-number combinations are changed, all other combinations must be re-entered.

<u>*SAME:</u> The specification for the channel and line values is to remain as originally defined.

**NORMAL:* The default values for skipping to channel identifiers will be used. The following are the default values:

ANSI First-Character Forms-Control Codes

Code	Action Before Printing a Line
' '	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to line 1
2-11	Space one line
12	Skip to overflow line (OVRFLW parameter)

channel-number: Specifies a channel number to be associated with corresponding 'skip to' line number. The only valid values for this parameter are 1 through 12, corresponding to channels 1 through 12. The CHLVAL parameter associates the channel number with a page line number.

If no line number is specified for a channel identifier, and that channel identifier is encountered in the data, a default of 'space one line' before printing is taken. Each channel number may be specified only once per CHGPRTF command invocation.

line-number: The line number assigned for the channel number in the same list. The range of valid line numbers is 1 through 255. If no line number is assigned to a channel number and that channel number is encountered in the data, a default of 'space one line' before printing is taken. Each line number may be specified only once per CHGPRTF command invocation.

SPOOL Parameter: Specifies whether the output data for the printer device file is to be spooled. If SPOOL(*NO) is specified, the following parameters in this command are ignored: OUTQ, FORMTYPE, COPIES, MAXRCDS, FILESEP, SCHEDULE, HOLD, and SAVE.

SAME: The value specified in the printer file description is not to be changed.

*YES: The data is to be spooled for processing by a card, diskette, or print writer.

*NO: The data is not to be spooled; it is sent directly to the device to be printed as the output becomes available.

OUTQ Parameter: Specifies, for spooled output only, the name of the output queue for the spooled output file.

SAME: The same output queue specified in the device file description is to be used.

*JOB: The output queue specified in the job description associated with the job is to be used.

qualified-output-queue-name: Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.) The IBM-supplied output queues that can be used by the printer file are the QPRINT, QPRINT2, and QPRINTS output queues, stored in the QGPL library.

FORMTYPE Parameter: Specifies, for spooled output only, the type of forms to be used in the printer when it uses this device file to produce printed output. The identifiers used to indicate the type of forms are user-defined and must not be longer than 10 characters.

*SAME: The type of printer forms specified in the printer file description remains the same.

**STD:* The standard form used in your installation is to be used with this device file for printed output. The system assumes (for *STD only) that the standard forms are already in the printer; no message is sent when this device file is opened.

form-type: Enter the identifier of the form type to be used with this device file for printed output from jobs. A maximum of 10 alphameric characters can be specified. When the device file is opened, the system sends a message identifying the form type to the system operator, and requests that the identified forms be mounted in the printer.

COPIES Parameter: Specifies, for spooled output only, the number of copies (regardless of whether it is one-part or multipart paper) of the output to be printed when this printer device file is used.

*SAME: The number of copies specified in the printer file description is not to be changed.

number-of-copies: Enter a value, 1 through 99, that indicates the number of copies to be produced when this device file is used.

MAXRCDS Parameter: Specifies the maximum number of records that can be in the spooled output file for spooled jobs using this printer device file. If this maximum is exceeded, an error message is sent to the program message queue and the program is terminated.

*SAME: The maximum number of records specified in the printer file description remains the same.

**NOMAX:* No maximum is specified for the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of records that can be in the spooled output file.

FILESEP Parameter: Specifies, for spooled output files only, the number of separator pages to be placed at the beginning of each printed file, including those between multiple copies of the same output. Each separator page has the following items printed on it: file name, file number, job name, user name, and job number.

*SAME: The number of separator pages specified in the printer file description is not to be changed.

number-of-file-separators: Enter the number of separator pages to be used at the beginning of each printed output file produced by this device file. Valid values are 0 through 9.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a writer.

*SAME: The time specified in the printer file description when spooled output can begin remains the same.

**JOBEND:* The spooled output file is to be made available to the writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the writer as soon as the file is opened in the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The spooled file is made available to a writer when it is released by the Release Spooled File (RLSSPLF) command.

*SAME: The same value specified in the printer file description is to be used.

**NO:* The spooled printer file is not to be held by the output queue. The spooled output is made available to a writer based on the SCHEDULE parameter value.

**YES:* The spooled printer file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled file is to be saved (left on the output queue) after the output has been produced.

*SAME: The value specified in the printer file description is not to be changed.

*NO: The spooled file data is not to be retained on the output queue after it has been produced.

*YES: The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*SAME: The wait time specified in the device file description for the needed objects is not to be changed.

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

*CLS: The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the printer device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the printer device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a write operation in that program produces the next output record.

*SAME: The value specified in the printer file description is not to be changed.

*NO: An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

*YES: An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given an internal system identifier when the format is created.

*SAME: The value specified in the printer file description is not to be changed.

**YES:* The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not all match, an open exception occurs and an error message is sent to the program requesting the open.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

TEXT Parameter: Specifies the user-defined text that describes the printer device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CHGPRTF  FILE(PRTRPT.ACCREC)  LPI(6)  ALIGN(*YES)
```

This command changes two parameters in the description of the device file named PRTRPT that is stored in the ACCREC library. The system operator must align the forms in the printer before the system begins printing that file. The file is to be printed in 6 lines per inch on the forms.

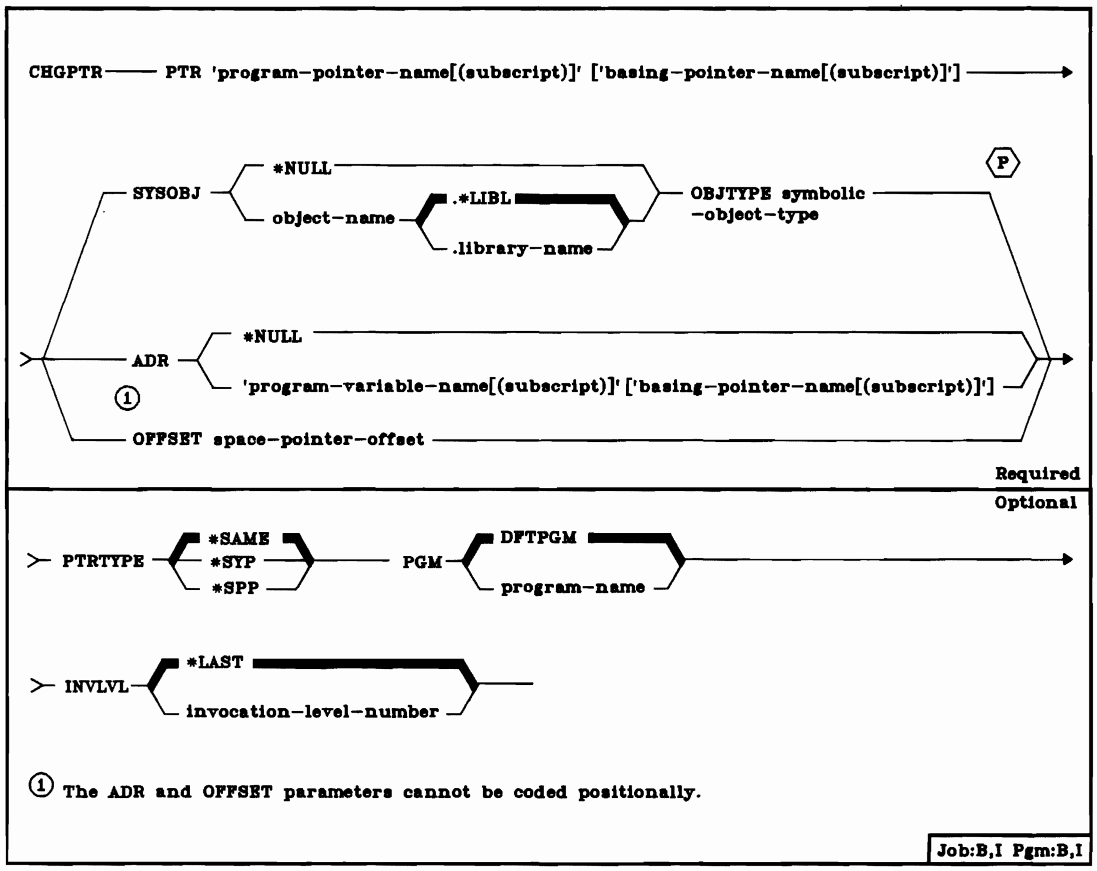
```
CHGPRTF  FILE(Q*.QSYS)  FORMSIZE(88132)  LPI(8)  OVERFLOW(80)
```

This command changes all IBM-supplied print files in library QSYS to use 88 lines of 132 characters (8 lines per inch), but to skip to the next page after 80 lines.

CHGPTR (Change Pointer) Command

The Change Pointer (CHGPTR) command changes the value of a pointer variable in a program. The value of the program pointer specified can be changed to point to a new system object (SYSOBJ), to a new space pointer address (ADR) or to a new offset within a space object (OFFSET). This command is not normally used in high-level language programs.

Restrictions: This command is valid only for changing program variables that are used as pointers; also, the command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*


CHGPTR
PTR 'program-pointer-name[(subscript)]' ['basing-pointer-name[(subscript)]']
SYSOBJ
*NULL
object-name
.*LIBL
.library-name
OBJTYPE symbolic -object-type
P
ADR
*NULL
'program-variable-name[(subscript)]' ['basing-pointer-name[(subscript)]']
1
OFFSET space-pointer-offset
Required
Optional
PTRTYPE
*SAME
*SYP
*SPP
PGM
DFTPGM
program-name
INVLVL
*LAST
invocation-level-number
1 The ADR and OFFSET parameters cannot be coded positionally.
Job:B,I Pgm:B,I

PTR Parameter: Specifies the name of the pointer (program variable) whose value is to be changed, allowing the pointer to point to a different system object or space pointer address.

'program-pointer-name': Enter the name of the pointer whose value is to be changed. The name must be enclosed in apostrophes if it contains special characters. An example of a CL variable used as a pointer is: PTR('&PTRVAR')

An MI ODV number can be specified instead of a pointer name. The ODV number must be preceded by a slash: PTR('/264') for example.

'program-pointer-name(subscript)': If the pointer is in an array, enter the name of the pointer and the subscript representing the positional element in the array whose value is to be changed. The subscript must be enclosed in parentheses, and the pointer name and subscript number must be enclosed in apostrophes. An example is: PTR('A(5)')

Either an integer or another variable name can be specified for the subscript.

['basing-pointer-name[(subscript)]']: This additional set of values in the PTR parameter applies only to MI or HLL programs that support based-on variables. The values can optionally be used with either of the previous two choices to also specify the value in an array that is based on a pointer or is an array of pointers. The same description of the coding syntax applies here. An example is:

```
PTR('VAR1(5)' 'PTR1(5)')
```

Coding Relationships: The next four parameters have a mutually exclusive relationship; only one of the following three combinations must be coded:

- SYSOBJ and OBJTYPE
- ADR
- OFFSET

SYSOBJ Parameter: Specifies that the pointer is a system pointer, or it is to be changed to a system pointer, whose value is to be changed. The pointer can be set to address MI system objects or to CPF objects. If a CPF object is specified that is stored in a library, the object name can be optionally qualified.

***NULL:** The system pointer is to be set to a null; that is, it no longer points to any system object nor does it have a pointer type. The OBJTYPE parameter cannot be specified if *NULL is specified here.

qualified-object-name: Enter the qualified name of the CPF or system object to which the system pointer is to be set. (If no library qualifier is given, *LIBL is used to find the object.) The user who codes this value must have at least read authority for the specified object. If a CPF object name is specified, the OBJTYPE parameter must also be specified.

OBJTYPE Parameter: Specifies the object type of the CPF or system object specified in the SYSOBJ parameter to which the pointer named in the PTR parameter is to be set. The CPF object type specified can be any one of those listed in the OBJTYPE parameter charts in Appendix A. For a chart of the system object types used by CPF that can be specified, refer to the *IBM System/38 Diagnostic Aids Manual,* SY21-0584.

ADR Parameter: Specifies the name of the program variable (if any) to which the specified space pointer is to point (that is, the program variable's address).

**NULL:* The space pointer is to be set to a null; it no longer points to the address of any space object nor does it have a pointer type.

'program-variable-name': Enter the name of the program variable to which the space pointer is to be set to point. If the variable name contains special characters (such as the & in a CL variable name), it must be enclosed in apostrophes. An example is:

```
ADR('&PTRADR')
```

An MI ODV number can be specified instead of a program variable name. The ODV number must be preceded by a slash. An example is:

```
ADR('/264')
```

'program-variable-name[(subscript)]': If the variable is in an array, enter the name of the variable and (optionally) the subscript representing the positional element in the array to which the pointer is to be set. If a subscript is not specified, it is set to point to the beginning of the array. The subscript, if specified, must be enclosed in parentheses, and the variable name and subscript number must be enclosed in apostrophes. An example is:

```
ADR('A(5)')
```

Either an integer or another variable name can be specified for the subscript.

['basing-pointer-name[(subscript)]']: This additional set of values in the ADR parameter applies only to MI or HLL programs that support based-on variables. The values can optionally be used with either of the previous two choices to also specify the value in an array that is based on a pointer. The same description of the coding syntax applies here. An example is:

```
ADR('VAR1(5)' 'PTR1(5)')
```

OFFSET Parameter: Specifies the value to which the offset portion of the specified space pointer is to be set. Enter the offset value that indicates the number of bytes from the start of the space object that the space pointer is to be set.

PTRTYPE Parameter: Specifies the type of pointer to which the pointer named in the PTR parameter is to be set.

*SAME: The type of pointer remains the same.

**SYP:* The pointer type is to be a system pointer.

**SPP:* The pointer type is to be a space pointer.

PGM Parameter: Specifies the name of the program that contains the pointer whose value is to be changed.

*DFTPGM: The program previously specified as the default program contains the pointer whose value is to be changed.

program-name: Enter the name of the program that contains the pointer whose value is to be changed. The same name (in qualified form) must already have been specified in the ENTDBG or ADDPGM command.

INVLVL Parameter: Specifies the invocation level of the program in which the pointer is to be changed. Invocation level 1 is the first (or earliest) invocation of the program, invocation level 2 is the second invocation, and so on down to the last (most recent) invocation level in the stack. If the pointer to be changed is a static pointer, INVLVL is ignored.

*LAST: The value of the specified pointer is to be changed in the last (most recent) invocation of the specified program.

invocation-level-number: Enter the number of the invocation level of the program that has the pointer whose value is to be changed.

Example

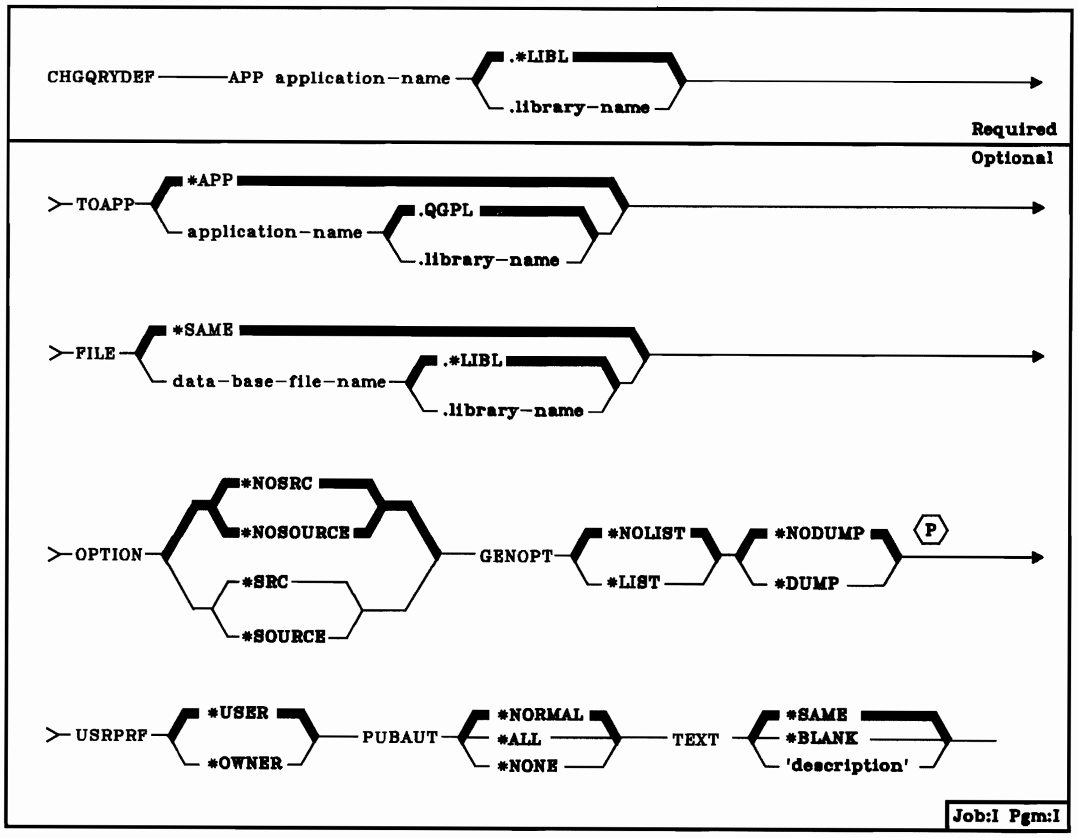
```
CHGPTR  PTR(DATAFILPTR) SYSOBJ(MYFILE.QGPL) +
     OBJTYPE(*FILE)
```

This command changes the value of the pointer DATAFILPTR that is used in the default program in the debugging session. The pointer value is changed so that it now points to the file called MYFILE, which is stored in the QGPL library.

CHGQRYDEF (Change Query Definition) Command

The Change Query Definition (CHGQRYDEF) command begins a prompting sequence for interactive modification of a Query application. Your response to the prompts are used to create a new application or to replace the original application.

The Query Utility is part of the IBM System/38 Interactive Data Base Utilities Program Licensed Program Product, Program 5714-UT1. For more information on the Query Utility, refer to the *IBM System/38 Query Utility Reference Manual and User's Guide,* SC21-7724.



APP Parameter: Specifies the qualified name of the application definition being changed. If no library name is given, the application is stored in the general-purpose library (QGPL).

TOAPP Parameter: Specifies the qualified name of the application in which the changed application is to be stored.

*APP: Specifies that the original application is to be replaced by the changed application.

application-name: Enter the name of the application in which the changed application is to be stored. The application definition specified in the APP parameter will remain as originally defined, and can be executed as originally defined. If no library name is given, the new application is stored in the general-purpose library (QGPL).

FILE Parameter: Specifies the name of an existing data base file with record formats that will be referred to by the application you are changing. The file is defined by DDS (see the *CPF Reference Manual—DDS*).

Note: Query has access to only the records included in the access path for the file; the access path is defined in DDS for the file. To determine whether DDS for a file contains select/omit logic that restricts the records available to Query, use the Display File Description (DSPFD) command.

*SAME: The data base file specified in the original application definition is to be used.

data-base-file-name: Specify the name of an existing data base file to be referred to during execution of the application. (If no library qualifier is specified, the library list (*LIBL) is used to find the file.)

OPTION Parameter: Specifies whether a listing of UDS (utility definition source) statements is to be printed, which may be helpful if problems occur.

*NOSRC or *NOSOURCE: Specifies that Query is not to print a listing of the UDS. The *NOSRC and *NOSOURCE values are equivalent.

**SRC* or **SOURCE:* Specifies that Query is to print a listing of the UDS. The *SRC and *SOURCE values are equivalent.

GENOPT Parameter: Specifies whether the IDU program listings created for your application program are to be produced. These listings may be helpful if a problem occurs.

*NOLIST: Specifies that an internal representation of the application program is not to be printed.

**LIST:* Specifies that an internal representation of the application program is to be printed.

*NODUMP: Specifies that the application program template is not to be printed.

**DUMP:* Specifies that the application program template is to be printed. *DUMP will provide the template only if *LIST has been specified.

USRPRF Parameter: Specifies a user profile under which the application is to be executed. This parameter allows a programmer to define a Query application for someone who does not have full authority over the data base file that the application reads.

*USER: The user profile of the application user is in effect when the application is executed.

**OWNER:* The user profiles of both the application owner and the application user are in effect when the application is executed.

When you create or change an application that is to be used by someone else, you must authorize the user for the use of the application and any objects associated with the application. You can grant each user specific rights to such objects. By specifying USRPRF(*OWNER) when an application is created or changed, you can permit a user to temporarily assume your authority to use objects associated with the application.

PUBAUT Parameter: Specifies what authority over the application is extended to all system users. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: All system users can execute or read the application, but not all users can delete the application.

**ALL:* All system users have complete authority over the application.

**NONE:* All users but the owner are restricted from using the application. Of course, the owner can grant rights to other users.

TEXT Parameter: Specifies a brief description of the changed application.

*SAME: The description of the application is to remain as originally defined.

**BLANK:* There is to be no description of this application.

'description': Enter no more than 50 characters, enclosed in apostrophes, to describe the changed application.

Example

```
CHGQRYDEF  APP(TEST1)  TOAP(TEST2) +
      TEXT('Create application TEST2, based on TEST1')
```

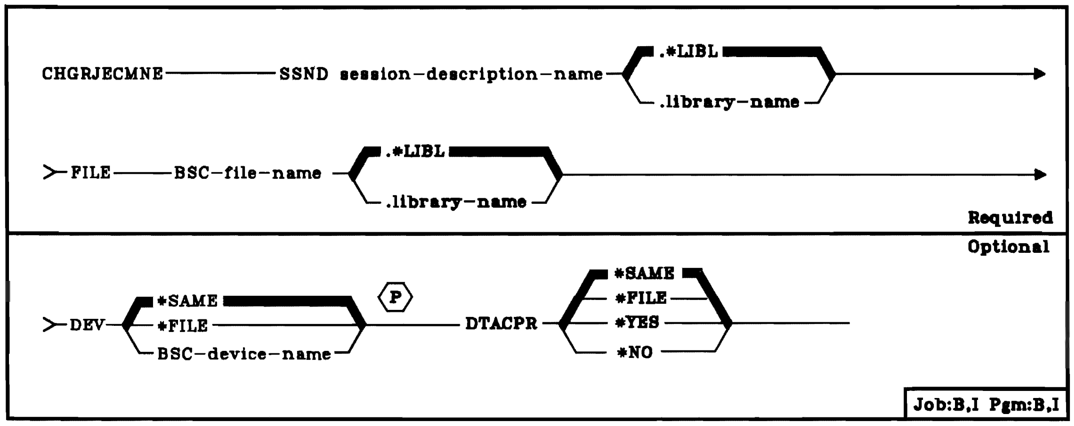
This command begins a prompting sequence that allows you to create an application named TEST2 in library QGPL based on application TEST1 in your library list. Your responses to the prompts can result in changes to the TEST2 application attributes (which differ from the based-on application TEST1). Application TEST1 is not changed in any way. Application TEST2 uses data from the data base file specified for application TEST1. No UDS or internal representations of application TEST2 will be printed. Any system users can execute or read TEST2, but only the owner of the application can delete it.

CHGRJECMNE (Change RJE Communications Entry) Command

The Change RJE Communications Entry (CHGRJECMNE) command changes attributes in an existing session description communications device file entry. This command can be issued while the RJEF session is active; however, the change does not take effect until the next Start RJE Session (STRRJESSN) command is issued.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Change RJE Communications Entry (CHGRJECMNE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description in which the communications entry is to be changed. (If no library qualifier is given, *LIBL is used to find the session description.)

FILE Parameter: Specifies the qualified name of the BSC device file entry to be changed in the session description. (If no library qualifier is given, *LIBL is used to find the file.)

DEV Parameter: Specifies the BSC device to be used with the specified communications device file for sending and receiving data.

*SAME: The BSC device named in the session description communications entry remains the same.

**FILE:* The device name specified in the BSC device file is to be used.

BSC-device-name: Enter the name of the BSC device to be used. This device name specified overrides the device that was specified when the BSC device file was created.

DTACPR Parameter: Specifies whether data compression is to be performed for the communications entry.

*SAME: The value specified in the session description communications entry remains the same.

**FILE:* Data compression is to be performed, based on the specification in the BSC device file.

**YES:* Data compression is to be performed for the communications entry.

**NO:* Data compression is not to be performed for the communications entry.

Example

```
CHGRJECMNE  SSND(RJE.USERLIB) +
     FILE(DEVPRT1.USERLIB) +
     DTACPR(*NO)
```

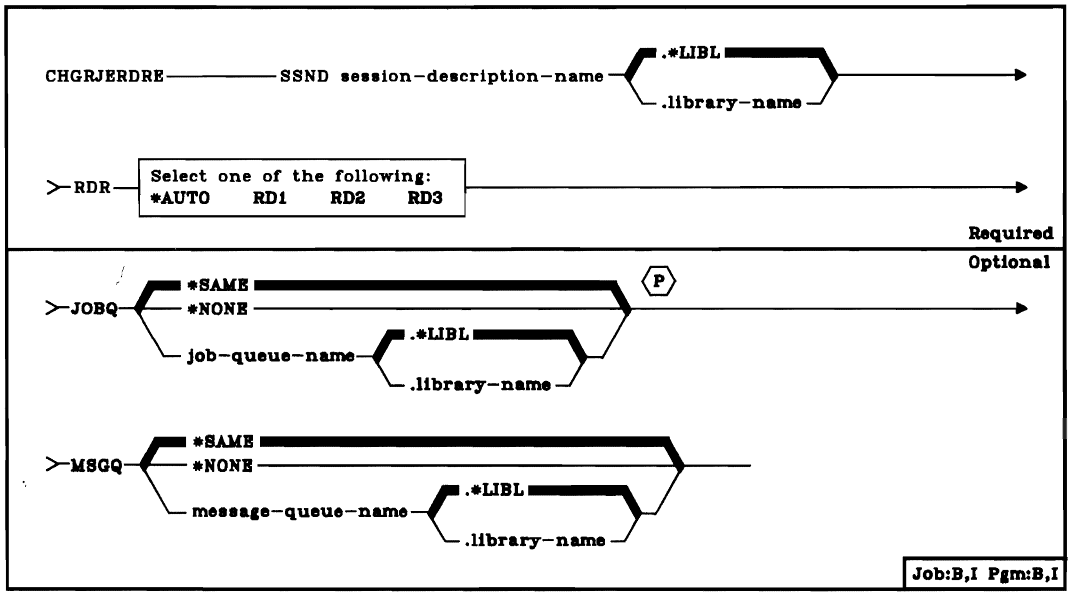
This command changes the communication entry named DEVPRT1.USERLIB in session description named RJE in library USERLIB. The entry is changed to prevent data compression.

CHGRJERDRE (Change RJE Reader Entry) Command

The Change RJE Reader Entry (CHGRJERDRE) command changes the attributes in an existing session description RJEF reader entry. The change takes effect immediately for readers identified in a Submit RJE Job (SBMRJEJOB) command with OPTION(*IMMED) specified. The change can also take effect when the next Start RJE Reader (STRRJERDR) command is issued for a reader identified in a SBMRJEJOB command with OPTION(*QUEUE) specified.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Change RJE Reader Entry (CHGRJERDRE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSND Parameter: Specifies the qualified name of the session description in which the RJEF reader entry is to be changed. (If no library qualifier is given, *LIBL is used to find the session description.)

RDR Parameter: Identifies the RJEF reader that is to be associated with this reader entry.

**AUTO:* Any RJEF reader input stream that is available at the time the Submit RJE Job (SBMRJEJOB) command executes is to be used.

RD1: RJEF Reader 1 input stream is to be used.

RD2: RJEF Reader 2 input stream is to be used.

RD3: RJEF Reader 3 input stream is to be used.

JOBQ Parameter: Specifies the job queue on which the reader jobs for this session are to be placed for transmission to the host system.

**SAME:* The RJEF job queue named in the session description RJEF reader entry remains the same.

**NONE:* No reader job queue is to be associated with this session description reader entry. RJEF reader data streams can be reserved for the interactive user issuing the SBMRJEJOB command and specifying OPTION(*IMMED). Therefore, the interactive user does not have to compete with the batch RJEF reader jobs that are started from the RJEF reader job queue.

job-queue-name: Enter the qualified name of the job queue on which reader jobs for this session description are to be placed, or are already placed, for transmission to the host system. (If no library qualifier is given, *LIBL is used to find the job queue.)

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF reader are to be recorded.

Note: Messages for RJEF readers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands.

**SAME:* The user message queue name, specified in the session description reader entry, remains the same.

**NONE:* No user message queue exists on which the messages for these RJEF reader jobs are to be recorded.

message-queue-name: Enter the qualified name of the user message queue on which this RJEF reader job's messages are to be recorded. (If no library qualifier is given, *LIBL is used to find the message queue.)

Example

```
CHGRJERDRE  SSND(RJE.USERLIB) +
      RDR(RD1) +
      MSGQ(BROWN.DEPT52)
```

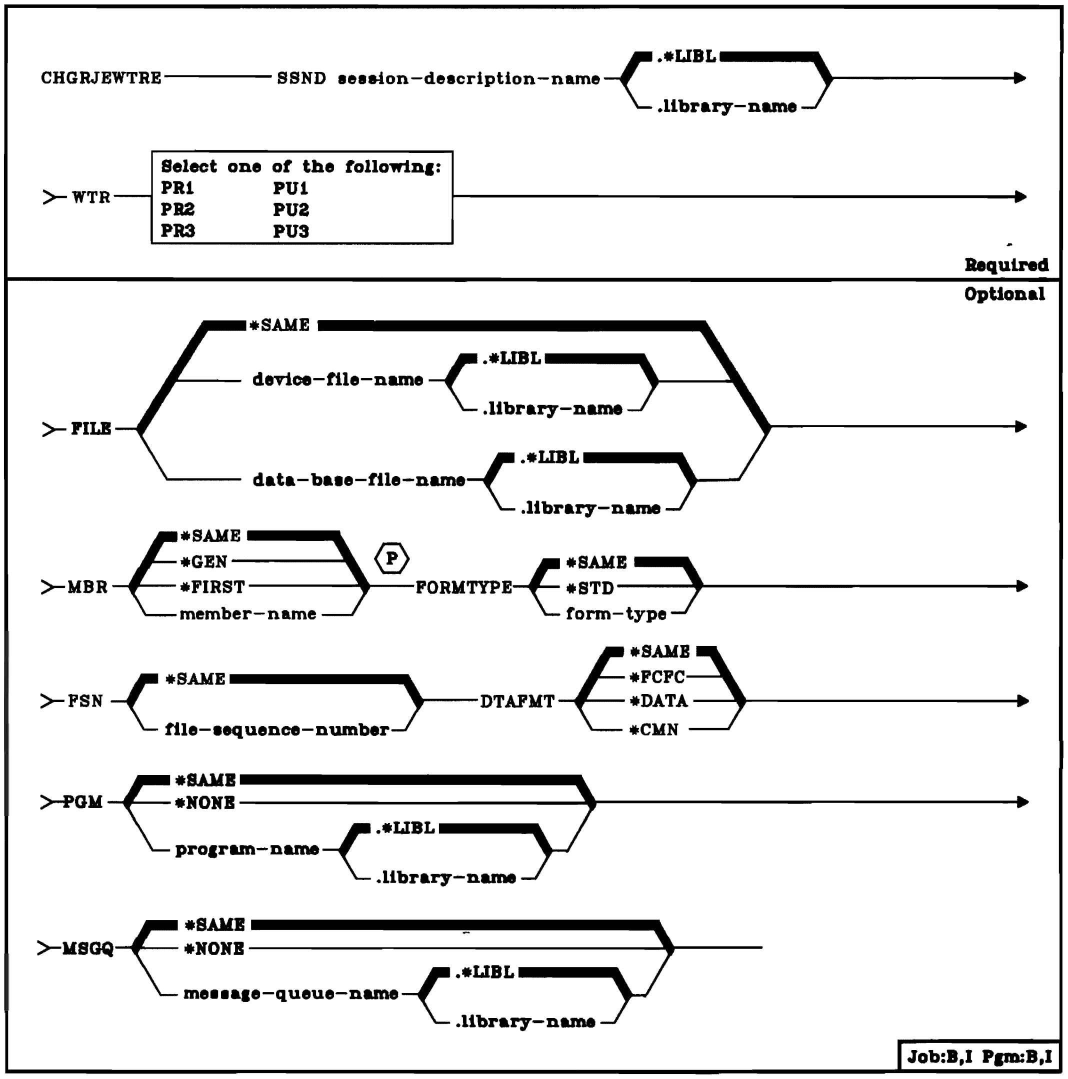
This command changes RD1 reader entry in session description named RJE in library USERLIB. The user message queue is changed to BROWN in library DEPT52. Messages associated with jobs submitted to RD1 will be written to message queue named BROWN in library DEPT52.

CHGRJEWTRE (Change RJE Writer Entry) Command

The Change RJE Writer Entry (CHGRJEWTRE) command changes the attributes in an existing session description RJEF writer entry. The change takes effect when the next Start RJE Writer (STRRJEWTR) command is issued for the writer specified in this command.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Change RJE Writer Entry (CHGRJEWTRE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description in which the RJEF writer entry is to be changed. (If no library qualifier is given, *LIBL is used to find the session description.)

WTR Parameter: Identifies the RJEF writer that is to be associated with this writer entry.

PR1: RJEF Printer 1 output stream is to be used.

PR2: RJEF Printer 2 output stream is to be used.

PR3: RJEF Printer 3 output stream is to be used.

PU1: RJEF Punch 1 output stream is to be used.

PU2: RJEF Punch 2 output stream is to be used.

PU3: RJEF Punch 3 output stream is to be used.

FILE Parameter: Specifies the qualified name of the RJEF writer file (printer only) or the System/38 data base file to be changed within the session description to receive output data from the host system.

*SAME: The RJEF writer device file name in the session description writer entry remains the same.

device-file-name: Enter the qualified name of the program-described device file to receive data. (If no library qualifier is given, *LIBL is used to find the device file.)

data-base-file name: Enter the qualified name of the System/38 physical data base file to receive the data. (If no library qualifier is given, *LIBL is used to find the data base file.)

MBR Parameter: Specifies the data base file member to which the output is to be directed (if a data base file was specified in the FILE parameter of this command).

**SAME:* The data base file member name in the session description writer entry remains the same.

**GEN:* RJEF creates a member name as follows:

Afffffccc or Bffffffccc

Where:

A	=	file member names beginning with the character A contain print data.
B	=	file member names beginning with the character B contain punch data.
ffffff	=	first six characters of the forms name specified in the FCT or received from the host system. **Note:** Only characters that are valid in a System/38 name are valid in the forms type used to generate data base file member names.
ccc	=	three-digit sequence value controlled by the RJEF session to maintain member uniqueness (refer also to the FSN parameter description of this command).

If a member with this name already exists in the data base file, the three-digit sequence value is incremented by one and another attempt is made to create a member. Incrementing of the sequence value continues until a unique name is generated and a member is created or until all 1000 possibilities have been exhausted without creating a member. If no member is created, the RJEF operator receives a message indicating the failure and a request to retry or cancel this file.

**FIRST:* The output is to be directed to the first member of the data base file (if a data base file is specified in the FILE parameter of this command).

member-name: Enter the name of the data base file member to which output is to be directed (if a data base file is specified in the FILE parameter of this command). If the member does not exist when it is needed, an inquiry message is sent to the RJEF message queue.

FORMTYPE Parameter: Specifies the initial form type to be used.

<u>*SAME:</u> The initial form type specified in the session description writer entry remains the same.

**STD:* The initial form type to be used is *STD.

form-type: Enter the initial form type. Valid values can be one through eight alphameric characters in length.

FSN Parameter: Specifies the initial three-digit file sequence number to be used when creating data base file member names. This parameter is ignored unless MBR(*GEN) is specified for this command or in the associated session description writer entry.

<u>*SAME:</u> The file sequence number specified in the session description writer entry remains the same.

file-sequence-number: Enter the initial three-digit file sequence number to be used. Leading zeros are not required for sequence numbers less than 100.

DTAFMT Parameter: Specifies the format of the output data.

<u>*SAME:</u> The data format designation specified in the session description writer entry remains the same.

**FCFC:* The output data is to be in the FCFC data format, with the first character of every record being the ANSI forms control character. Parameter value WTR(PUn) is invalid with parameter value DTAFMT(*FCFC). Specify *FCFC if the data is to be printed.

The data can be written to a data base file in the FCFC data format and be printed later by using the Copy File (CPYF) command and specifying an FCFC printer file on the TOFILE parameter.

**DATA:* The output data is to be in the normal data format (that is, no FCFC characters are embedded in the data). Specify *DATA if the data is to go to a data base file and be processed by a program. If the data is directed to a printer device file, a single space ANSI control character is the last character in each record.

**CMN:* The output data is to be in the communications data format (that is, still compressed or truncated). *CMN should be used to decrease communications time. However, before the data can be used, the Format RJE Data (FMTRJEDTA) command must be used to change the data to *FCFC or *DATA. If *CMN is specified, the output file must be a data base file with a length of 256.

PGM Parameter: Specifies the qualified name of a user-supplied program to be used.

***SAME:** The value of the program entry in the session description writer entry remains the same.

**NONE:* A null value is used for the program value for the writer entry.

program-name: Enter the qualified name of the user-supplied program to be used. (If no library qualifier is given, *LIBL is used to find the user-supplied program.)

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF writer are to be recorded.

Note: Messages for RJEF writers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands. If inquiry messages are issued by RJEF, they are sent to the user message queue (if specified) where they must receive a response.

***SAME:** The user message queue name remains the same.

**NONE:* No user message queue exists on which the messages for these RJEF writer jobs are to be recorded.

message-queue-name: Enter the qualified name of the user message queue on which this RJEF writer job's messages are to be recorded. If no library qualifier is given, *LIBL is used to find the message queue.

Example

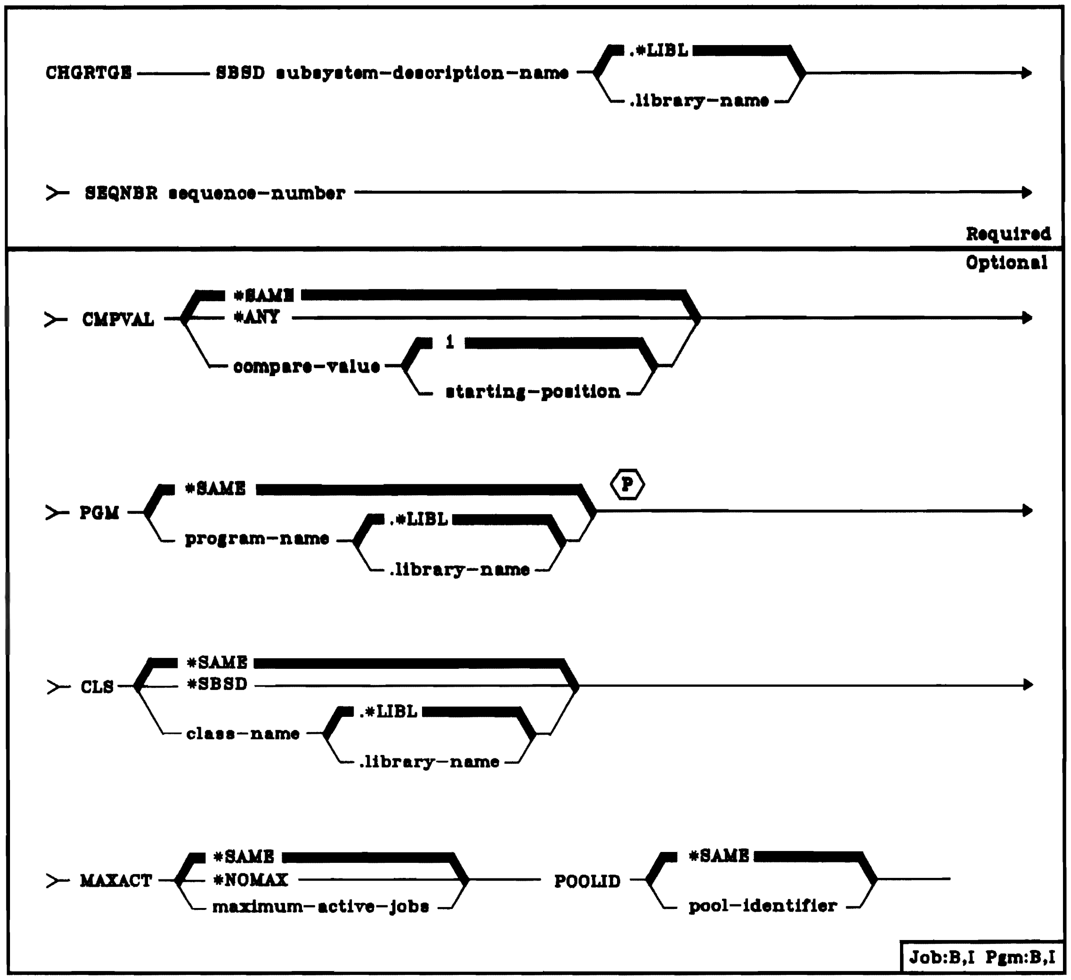
```
CHGRJEWTRE  SSND(RJE.USERLIB) +
      WTR(PR1) +
      FILE(NIGHTPRT.USERLIB) +
      DTAFMT(*FCFC) +
      MSGQ(*NONE)
```

This command changes an RJEF writer entry called PR1 in session description named RJE in library USERLIB. The file is changed to a printer device file called NIGHTPRT in library USERLIB. Output data will be written in the normal *FCFC format. Messages associated with printer 1 are not to be written to a user message queue. They will be written only to the RJE message queue specified on the CRTSSND command.

CHGRTGE (Change Routing Entry) Command

The Change Routing Entry (CHGRTGE) command changes a routing entry in the specified subsystem description; the associated subsystem must be inactive when the changes are made. The routing entry specifies the parameters used to initiate a routing step.

Restriction: To use this command, you must have operational and object management rights for the subsystem description being changed.



SBSD Parameter: Specifies the qualified name of the subsystem description containing the routing entry to be changed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

SEQNBR Parameter: Specifies the sequence number of the routing entry that is to be changed. Enter the sequence number, 1 through 9999, of the routing entry.

CMPVAL Parameter: Specifies a value that is to be compared with the routing data to determine whether this is the routing entry to be used for initiating a routing step. Optionally, a new starting position within the routing data character string can be specified for the comparison. If CMPVAL is not specified, *SAME is assumed; if a starting position value is not specified, 1 is assumed.

SAME: The compare value and starting position are not to be changed.

**ANY:* Any routing data is considered to be a match. To specify *ANY, this routing entry must also have the highest SEQNBR value of any routing entry in the subsystem description.

compare-value: Enter a new value (a character string not exceeding 80 characters) that is to be compared with routing data for a match. When a match occurs, this routing entry is used to initiate a routing step. A starting position within the routing data character string can be specified for the comparison; if no position is specified, 1 is assumed.

1: The comparison between the compare value and the routing data begins with the first position in the routing data character string.

starting-position: Enter a value indicating which position in the routing data character string is the starting position for the comparison. The last character position compared must be less than or equal to the length of the routing data used in the comparison.

PGM Parameter: Specifies the name of the program to be invoked as the (first) program to be executed in the routing step. (No parameters can be passed to the specified program.)

SAME: The program to be called is not to be changed.

qualified-program-name: Enter the qualified name of the program to be invoked and executed in the routing step. (If no library qualifier is given, *LIBL is used to find the program.) If the program does not exist when this routing entry is changed, a library qualifier must be specified because the qualified program name is retained in the subsystem description.

CLS Parameter: Specifies the qualified name of the class to be used for the routing steps initiated through this routing entry. The class defines the attributes of the execution environment for processing the routing step associated with this routing entry. (For an expanded description of the CLS parameter, see Appendix A.) If the class does not exist when this routing entry is changed, a library qualifier must be specified because the qualified class name is retained in the subsystem description.

*SAME: The same class for this entry is to be used.

**SBSD:* The class having the same qualified name as the subsystem description, specified by the SBSD parameter, is to be used for routing steps initiated through this entry.

qualified-class-name: Enter the qualified name of the class that is to be used for routing steps initiated through this routing entry. If no library qualifier is specified, the library list (*LIBL) of the job in which this CHGRTGE command is executed is used to find the class.

MAXACT Parameter: Specifies the maximum number of routing steps (jobs) that can be concurrently active through this routing entry. (Within a job, only one routing step is active at a time.) When a subsystem is active and the maximum number of routing steps is reached, any subsequent attempts to initiate a routing step through this routing entry will fail. If the routing data was entered interactively, an error message is sent to the user. Otherwise, the job is terminated and a message is sent by the subsystem to the job's log. (For an expanded description of the MAXACT parameter, see Appendix A.)

*SAME: The maximum number of routing steps that can be concurrently active is not to be changed.

**NOMAX:* There is no limit to the number of routing steps that can be concurrently active through this routing entry. (This value is normally used when there is no reason to control the number of routing steps.)

maximum-active-jobs: Enter a value that specifies the new maximum number of routing steps that can be concurrently active through this routing entry. If a routing step would exceed this number if it were started, the job is implicitly terminated.

POOLID Parameter: Specifies the pool identifier of the storage pool in which the program is to run. The pool identifier specified here relates to the storage pools in the subsystem description.

*SAME: The pool identifier is not to be changed.

pool-identifier: Enter the identifier of another existing storage pool in which the routing step is to run. Valid values are 1 through 10.

Examples

```
CHGRTGE  SBSD(ORDER.LIB5)  SEQNBR(1478)  +
      CLS(SOFAST.LIB6)  POOLID(3)
```

This command changes routing entry 1478 in the subsystem description ORDER found in library LIB5. The same program is used, but now it will run in storage pool 3 using class SOFAST in library LIB6.

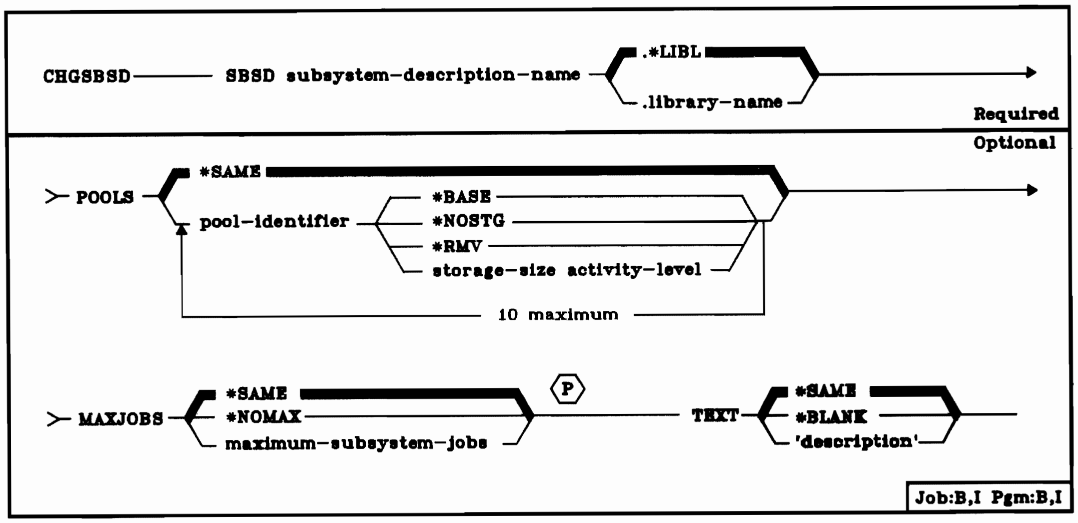
```
CHGRTGE  SBSD(PGMR.T7)  SEQNBR(157)  +
      PGM(INTDEV.T7)
```

This command changes routing entry 157 in the subsystem description PGMR found in library T7. The program INTDEV in library T7 will now be invoked whenever this routing entry is selected. The other routing entry parameters remain unchanged.

CHGSBSD (Change Subsystem Description) Command

The Change Subsystem Description (CHGSBSD) command changes the operational attributes of the specified subsystem description. Note that this is the only command affecting the subsystem description that can be issued while the subsystem is active.

Restriction: You must have operational and object management rights for the subsystem description before you can change it. You must also have operational rights for the library containing the subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description to which changes are to be made. (If no library qualifier is given, *LIBL is used to find the subsystem description.) The name of the IBM-supplied controlling subsystem, QCTL, should *not* be specified. The CRTSBSD command, however, can be used to create a similar subsystem description, and you can make it the controlling subsystem by specifying its name in the QCTLSBSD system value instead of QCTL.

POOLS Parameter: Specifies the identifiers of one or more storage pool definitions and the changes that are to be made to them. If a new storage pool definition is to be added or an existing pool definition is to be removed, the subsystem must be inactive. For each existing pool definition that is not specified, its size and activity level are not changed.

*SAME: No changes are to be made to the storage pool definitions in the subsystem description.

pool-identifier: Enter the pool identifier, 1 through 10, of the storage pool definition to be added or deleted, or whose attributes are to be changed. If more than one pool definition is specified, the attributes of each one must follow its identifier.

**BASE:* The specified pool definition is to be the system pool, which can be shared with other subsystems. The size and activity level of the shared system pool are specified in the system values QBASPOOL and QBASACTLVL (see the *CPF Programmer's Guide.*)

**NOSTG:* No storage and no activity level are to be assigned to the pool at the present time.

**RMV:* The specified pool definition is to be removed from the subsystem description. If the specified pool definition is used for any routing entries in the subsystem description, an error message is sent to the user and the pool definition is *not* removed. *RMV cannot be specified if the subsystem is active.

storage-size activity-level: Enter the storage size in K-bytes that the specified storage pool is to have, and enter the maximum number of jobs that can execute concurrently in the pool. Both values must be specified. A value of at least 16 (meaning 16 K-bytes) must be specified for the storage size.

MAXJOBS Parameter: Specifies the maximum number of jobs that can be concurrently active within the subsystem controlled by this subsystem description. The maximum applies to all initiated jobs that are waiting or executing, except for jobs on the job queue or jobs that have finished executing.

*SAME: The maximum number of concurrent jobs allowed within the subsystem is not to be changed.

**NOMAX:* There is no maximum number of concurrent jobs allowed within this subsystem.

maximum-subsystem-jobs: Enter the maximum number of jobs to be allowed in this subsystem.

TEXT Parameter: Specifies the user-defined text that describes the subsystem description. The text specified here replaces any previous text. (For an expanded description of the TEXT parameter, see Appendix A.)

Note: The text can be changed only when the subsystem description is not active.

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CHGSBSD  SBSD(PAYCTL.QGPL)  POOLS((2 150 3))
```

This command changes the definition of storage pool 2, which is used by the PAYCTL subsystem, to a storage size of 150 K and an activity level of 3.

```
CHGSBSD  SBSD(ORDER.LIB6) +
      POOLS((1 *BASE)(2 75 4)(3 *RMV)(4 *NOSTG)) +
      MAXJOBS(5)
```

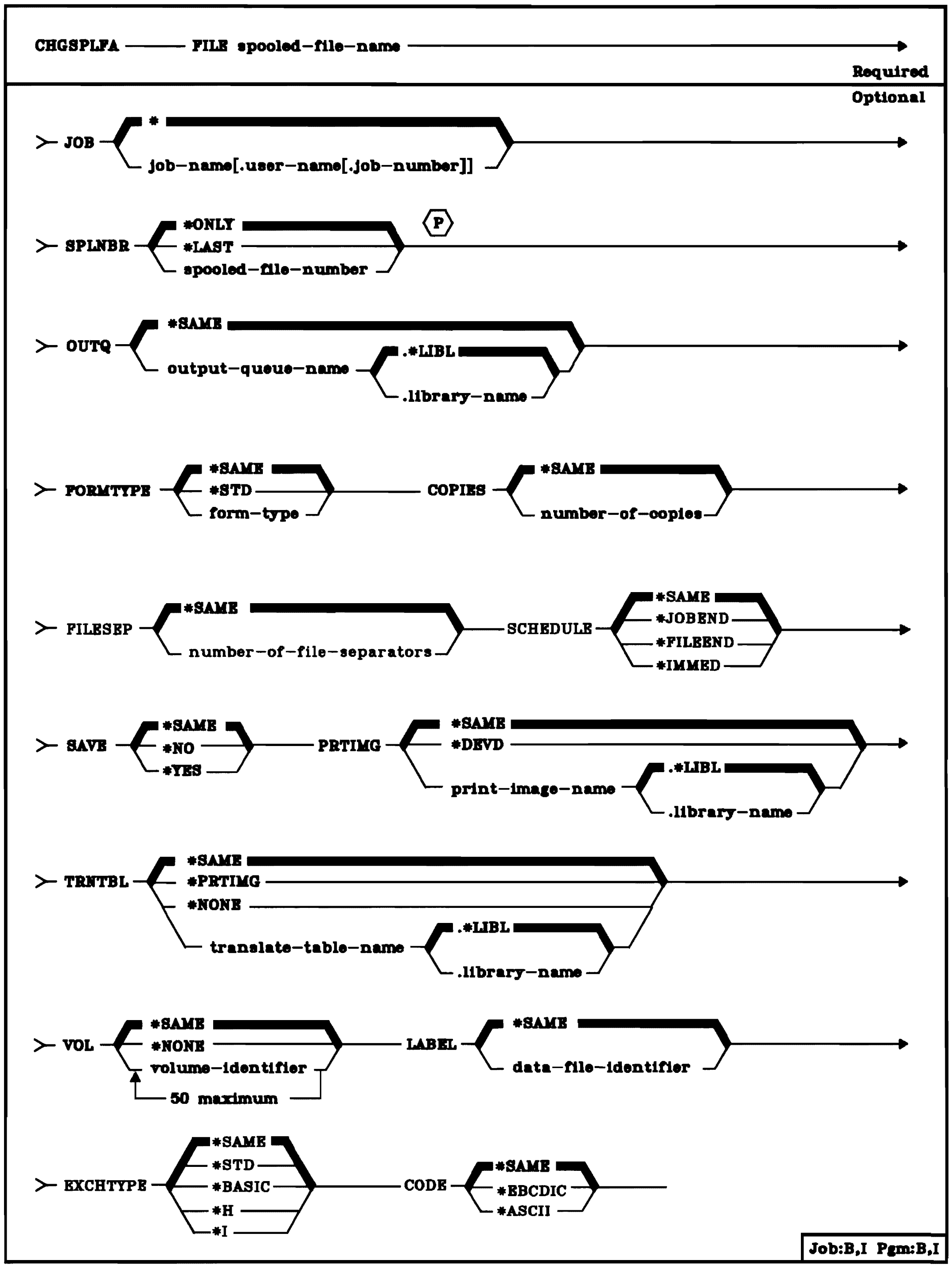
This command changes the maximum number of jobs that the subsystem ORDER can support to five. (The description of the subsystem is stored in library LIB6.) The definition of storage pool 1 is changed to be the shared system pool; the definition of pool 2 is changed to have a storage size of 75 K and an activity level of 4; the definition of pool 3 is removed from the subsystem; and the definition of pool 4 is changed to have no storage and no activity level.

CHGSPLFA (Change Spooled File Attributes) Command

The Change Spooled File Attributes (CHGSPLFA) command changes the attributes of a spooled output file while it is on an output queue. The changes affect only the current processing of the file. The next time the job runs and the file is produced, the file attributes are derived from the device file description, from the program, and from any override commands.

If the file is currently being produced on an output device, the only parameters that can be changed are COPIES and SAVE. An attempt to change any other parameter results in an error, and no file attributes are changed. However, if the file is being held on an output queue because of spooling attribute errors, this command can be used to change the attributes, and a spooling writer can then be started to produce the file.

Restrictions: To change the attributes of a spooled file, you must: (1) be the owner of the spooled file; or (2) have read, add, and delete rights for the output queue; or (3) have the job control rights (*JOBCTL) *and* the output queue must have OPRCTL(*YES) specified. In addition, if the OUTQ parameter is to be changed, you must have add rights for the new output queue.


CHGSPLFA
FILE spooled-file-name
Required
Optional
JOB
*
job-name[.user-name[.job-number]]
SPLNBR
*ONLY
*LAST
spooled-file-number
P
OUTQ
*SAME
output-queue-name
.*LIBL
.library-name
FORMTYPE
*SAME
*STD
form-type
COPIES
*SAME
number-of-copies
FILESEP
*SAME
number-of-file-separators
SCHEDULE
*SAME
*JOBEND
*FILEEND
*IMMED
SAVE
*SAME
*NO
*YES
PRTIMG
*SAME
*DEVD
print-image-name
.*LIBL
.library-name
TRNTBL
*SAME
*PRTIMG
*NONE
translate-table-name
.*LIBL
.library-name
VOL
*SAME
*NONE
volume-identifier
50 maximum
LABEL
*SAME
data-file-identifier
EXCHTYPE
*SAME
*STD
*BASIC
*H
*I
CODE
*SAME
*EBCDIC
*ASCII
Job:B,I Pgm:B,I

FILE Parameter: Specifies the name of the spooled file in a job that is to have its attributes changed. Enter the name of the spooled device file.

JOB Parameter: Specifies the name of the job that created the spooled file.

<u>*:</u> The job that issued this CHGSPLFA command is the job that created the spooled file.

qualified-job-name: Enter the qualified name of the job that contains the spooled file. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the unique number of the spooled output file in the job whose attributes are to be changed. (For an expanded description of the SPLNBR parameter, see Appendix A.)

<u>*ONLY:</u> Only one spooled output file from the job has the specified file name; therefore, the number of the spooled file is not necessary. If *ONLY is specified and more than one spooled output file has the specified file name, an error message is displayed.

**LAST:* The highest numbered spooled output file with the specified file name is the file whose attributes are to be changed.

spooled-file-number: Enter the number of the spooled output file having the specified file name whose attributes are to be changed.

OUTQ Parameter: Specifies the name of the output queue to which the spooled file is to be moved. This parameter is used only when the specified file is to be moved from one output queue to another.

<u>*SAME:</u> The file remains on the same output queue.

qualified-output-queue-name: Enter the qualified name of the output queue to which the file is to be moved. (If no library qualifier is given, *LIBL is used to find the queue.) You must have add authority for the queue specified in this parameter.

FORMTYPE Parameter: Specifies, for printer or card output, the forms identifier that identifies the type of forms on which this output file is to be produced. The forms identifiers are user-defined and must not be longer than 10 characters. *SAME must be specified if the output file is a diskette file.

<u>*SAME:</u> The type of forms is not to be changed.

**STD:* The standard form used in your installation is to be used to produce this spooled file. The system assumes (for *STD) that the standard forms are already in the print or card device; no message is sent when this spooled file is opened.

form-type: Enter the forms identifier that specifies on which forms the output of this spooled file is to be produced. A maximum of 10 alphameric characters can be specified. Strings with embedded blanks must be enclosed in quotes.

COPIES Parameter: Specifies the number of copies of the output file to be produced.

**SAME:* The number of copies remains unchanged.

number-of-copies: Enter the new number of identical copies that are to be produced. Valid values are 1 through 99.

FILESEP Parameter: Specifies the number of separator pages or cards to be produced at the beginning of each output file to separate the file from the other files being spooled to an output device. The identifying information included on each file separator is the file name, file number, the job name and number, and the user's name. *SAME must be specified if the output file is a diskette file.

**SAME:* The number of separator pages or cards is not to be changed.

number-of-file-separators: Enter the new number (0 through 9) of pages or cards that are to be used as file separators. If 0 is specified for card output, at the end of each output file a message is sent to the message queue (usually QSYSOPR) specified on the STRCRDWTR command that started the writer; the message indicates that the output just produced is to be removed from the device.

SCHEDULE Parameter: Specifies when the output file is made available to the writer.

**SAME:* The schedule attribute of the spooled file is not to be changed.

**JOBEND:* The spooled output file is to be made available to the writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the writer, as soon as the file has been closed in the program.

**IMMED:* The spooled output file (already open) is to be made available to the writer when the file is opened.

SAVE Parameter: Specifies whether the spooled output file is to be saved after it has been written to an output device. After the file has been written, the number of copies (COPIES parameter) is set to 1, and the status of the file is changed to held. The file can be retained on the output queue (saved) so that it can be used to produce other copies of the output.

*SAME: The save attribute of the spooled output file is not to be changed.

**NO:* The file is not saved.

**YES:* The file is saved.

PRTIMG Parameter: Specifies, for printer output files only, the name of the print image that is to be used to produce the spooled file on a printer. This parameter is ignored for work station printers.

*SAME: The print image associated with the spooled output file remains the same.

**DEVD:* The standard print image, specified in the printer device description, is to be used.

qualified-print-image-name: Enter the qualified name of the print image that is to be used to print this output file. (If no library qualifier is given, *LIBL is used to find the print image.)

TRNTBL Parameter: Specifies, for printer output files only, the name of the translate table (if any) to be used with this spooled file when the output data is to be translated before it is printed. The translate table is used to convert each unprintable character having a hexadecimal code of 40 through FE to the printable character specified in the table that is also on the print belt. Each hexadecimal code can specify a different character.

For each IBM-supplied print image shipped with the system, a matching translate table is also supplied; the name of the table is the same as the name of the image.

*SAME: The translate table associated with the spooled output file remains the same.

**PRTIMG:* The translate table with the same qualified name as the print image is to be used.

**NONE:* No translation is to be done when this spooled output file is produced.

qualified-translate-table-name: Enter the qualified name of the translate table that is to be used to convert unprintable characters before this spooled output file is printed. (If no library qualifier is given, *LIBL is used to find the translate table.)

VOL Parameter: Specifies, for diskette output files only, one or more volume identifiers of the diskettes (either in magazines or in slots) on which this spooled file is to be written. The diskettes (volumes) must be mounted on the device in the same order as the identifiers are specified here; a message is sent to the system operator if the order is different. The identifiers are matched, one by one, with the diskette locations specified in the LOC parameter. (For an expanded description of the VOL parameter, see Appendix A.)

*SAME: The volume identifiers associated with the spooled output file are not changed.

**NONE:* No diskette volume identifiers are to be specified. This output file is to be written on the first available diskette, based on the diskette writer's current position. No volume identifier checking is performed.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used for this output file. No more identifiers can be specified here than were initially specified for the diskette device file.

Each volume identifier contains a maximum of six characters. A blank is used as the separator character when listing multiple identifiers. The number of volumes possible in the list is 50, but if more than 10 volume names were specified when the file was first opened, then only that number of files may be entered on the change command. Up to 10 volumes may always be specified.

LABEL Parameter: Specifies, for diskette output files only, the data file identifier of the data file to be written on diskette from this spooled output file. The data file identifier is stored in a label in the volume label area of the diskette. (For an expanded description of the LABEL parameter, see Appendix A.)

*SAME: The data file identifier associated with the spooled output file remains the same.

data-file-identifier: Enter the identifier (8 characters maximum) to be assigned to the data file being written on diskette from this spooled output file.

EXCHTYPE Parameter: Specifies the exchange type to be used to write the spooled file. This parameter must be coded EXCHTYPE(*SAME) if the spooled file is not a diskette file. (For an expanded description of the EXCHTYPE parameter, refer to Appendix A).

*SAME: The current value is not changed.

**STD:* The basic exchange format will be used for a type 1 or a type 2 diskette. The H exchange type will be used for a type 2D diskette.

**BASIC:* The basic exchange type will be used.

**H:* The H exchange type will be used.

**I:* The I exchange type will be used.

CODE Parameter: Specifies, for diskette output files only, the type of character code to be used when this spooled output file is written to diskette.

*SAME: The type of character code associated with the spooled output file remains the same.

**EBCDIC:* The EBCDIC character code is to be used with this output file.

**ASCII:* The ASCII character code is to be used with this output file.

Examples

```
CHGSPLFA  FILE(SALES) JOB(BILLING.JONES.000147) +
     OUTQ(QPRINT2) FORMTYPE('1140-6')
```

This command moves the file named SALES (of the BILLING job numbered 000147) from the present queue to the QPRINT2 queue. It also changes the forms identifier to 1140-6, which means that that form type is to be used in the printer.

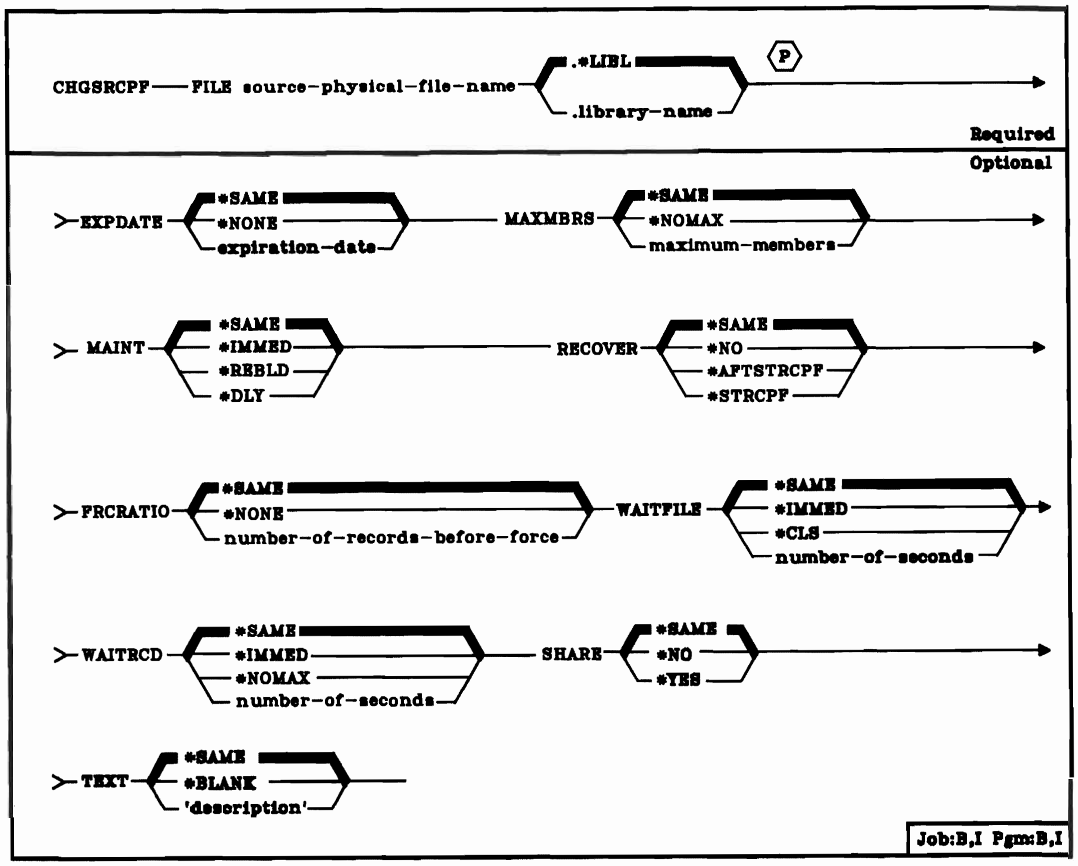
```
CHGSPLFA  FILE(DEPT511) COPIES(2) FILESEP(5)
```

This command changes the attributes of the output file DEPT511 that is produced by the submitter's job. It changes the number of output copies to two and specifies that five separator pages (or cards) are to precede each copy.

CHGSRCPF (Change Source Physical File) Command

The Change Source Physical File (CHGSRCPF) command changes the attributes of a source physical file and all its members. The changed attributes will be used for all members subsequently added to the file.

Restrictions: To change a source physical file, you must have object management and operational rights for the file and read rights to the library. In order for you to change the file, an exclusive no read lock is necessary; no one may be using the file for any purpose.



FILE Parameter: Specifies the qualified name of the physical file to be changed. (If no library qualifier is given, *LIBL is used to find the file.)

EXPDATE Parameter: Specifies the expiration date of all the file's members. Any attempt to open a file member that has expired causes an error message to be sent. (The RMVM command is used to remove the member.) If EXPDATE is specified, all members in the file will be changed. An expired member may be changed to non-expired by changing the EXPDATE parameter. The expiration date must be later than or equal to the current day's date.

<u>*SAME:</u> The expiration date of the file is not to be changed.

**NONE:* The member has no expiration date.

expiration-date: Enter the date after which the member should not be used. The date must be specified in the format defined by the system values, QDATFMT and QDATSEP. The date must be enclosed in apostrophes if special characters are used in the format.

MAXMBRS Parameter: Specifies the maximum number of members that the physical file can have at any time. The maximum number of members specified must be greater than or equal to the current number of members in the file.

<u>*SAME:</u> The maximum number of members in the file is not to be changed.

**NOMAX:* No maximum is specified for the number of members; the system maximum of 32 767 members per file is used.

maximum-members: Enter the value for the maximum number of members that the physical file can have. A value of 1 through 32767 is valid.

MAINT Parameter: Specifies the type of access path maintenance to be used for all members of the physical file. This parameter is valid only if a keyed access path is used.

Only the following changes to a file's access path maintenance are allowed: *REBLD to *IMMED (if the file was originally created as *IMMED or *REBLD), *IMMED to *REBLD, *DLY to *REBLD, and *REBLD to *DLY (if the file was originally created as *DLY).

Existing MAINT Value	CHGSRCPF MAINT Parameter Value		
	***REBLD**	***DLY**	**IMMED**
*REBLD	N/A	Note 1	Note 2
*DLY	YES	N/A	NO
*IMMED	YES	NO	N/A

Notes:
1. Allowed only if file was originally created with MAINT(*DLY).
2. Allowed only if file was originally created with MAINT(*IMMED) or MAINT(*REBLD).

*SAME: The access path maintenance of the file is not to be changed.

**IMMED:* The access path is to be continuously (immediately) maintained for each physical file member. The path is updated each time a record is changed, added to, or deleted from the member. The records can be changed through a logical file that uses the physical file member, regardless of whether the physical file is opened or closed. *IMMED must be specified for all files requiring unique keys to ensure uniqueness in all inserts and updates.

**REBLD:* The access path is to be rebuilt when a file member is opened during program execution. The access path is continuously maintained until the member is closed; the access path maintenance is then terminated. *REBLD is not valid for access paths that are to contain unique key values.

**DLY:* The maintenance of the access path is to be delayed until the member is opened for use. The access path is then updated only for records that have been added, deleted, or updated since the file was last closed. (While the file is open, all changes made to based-on members are immediately reflected in the access paths of the opened files' members, no matter what is specified for MAINT.) To prevent a lengthy rebuild time when the file is opened, *DLY should be specified only when the number of changes to the access path between a close and the next open are small (when key fields in records for this access path change infrequently). *DLY is not valid for access paths that require unique key values.

If the number of changes saved reaches approximately 10 per cent of the access path size, the system will stop saving changes and the access path will be completely rebuilt the next time the file is opened.

RECOVER Parameter: Specifies, for files having immediate or delayed maintenance on their access paths, when recovery processing of the file is to be performed if a system failure occurred while the access path was being changed.

The access path having immediate or delayed maintenance can be rebuilt during start CPF (before any user can execute a job), or after start CPF has finished (during concurrent job execution), or when the file is next opened. While the access path is being rebuilt, the file cannot be used by any job.

The access path having rebuild maintenance will be rebuilt the next time its file is opened, the time that it normally is rebuilt. This parameter is valid only if a keyed access path is used.

**SAME:* The recovery attribute of the file is not to be changed.

**NO:* The access path of the file is not to be rebuilt. The file's access path is rebuilt the next time the file is next opened.

**AFTSTRCPF:* The file is to have its access path rebuilt after the start CPF operation has been completed. This option allows other jobs not using this file to begin processing immediately after the CPF has been started. If a job tries to allocate the file while its access path is being rebuilt, a file open exception occurs if the specified wait time for the file is exceeded.

**STRCPF:* The file is to have its access path rebuilt during the start CPF operation. This ensures that the file's access path will be rebuilt before the first user program tries to use it; however, no jobs can begin execution until after all files that specify RECOVER(*STRCPF) have their access paths rebuilt.

FRCRATIO Parameter: The force write ratio parameter specifies the number of inserted, updated, or deleted records that are processed before they are forced to auxiliary (permanent) storage. (For an expanded description of the FRCRATIO parameter, see Appendix A.)

<u>*SAME:</u> The force write ratio of the file is not to be changed.

**NONE:* There is no force write ratio; the system determines when the records are written in auxiliary storage.

number-of-records-before-force: Enter the number of new or changed records that are processed before they are explicitly forced into auxiliary storage.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

<u>*SAME:</u> The wait attribute of the file is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

WAITRCD Parameter: Specifies the number of seconds that the program is to wait for a record that is to be updated or deleted. If the record cannot be allocated in the specified wait time, an error message is sent to the program.

<u>*SAME:</u> The record wait attribute of the file is not to be changed.

**IMMED:* The program is not to wait; when a record is locked, an immediate allocation of the record is required.

**NOMAX:* The wait time will be the maximum allowed by the system (32 767 seconds).

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether an ODP (open data path) to the physical file member is to be shared with other programs in the same job. When an ODP is shared, the programs accessing the file share such things as the position being accessed in the file, the file status, and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record. If SHARE is specified, all members in the file will be changed.

SAME: The ODP sharing value of the member is not to be changed.

**NO:* An ODP created by the program when the file member is opened is not to be shared with other programs in the job. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* The same ODP is to be shared with each program in the job that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Enter text that briefly describes the physical file member. (For an expanded description of the TEXT parameter, see Appendix A.)

SAME: The text that describes the member is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CHGSRCPF  FILE(INV.QGPL)  EXPDATE('10/31/87')
```

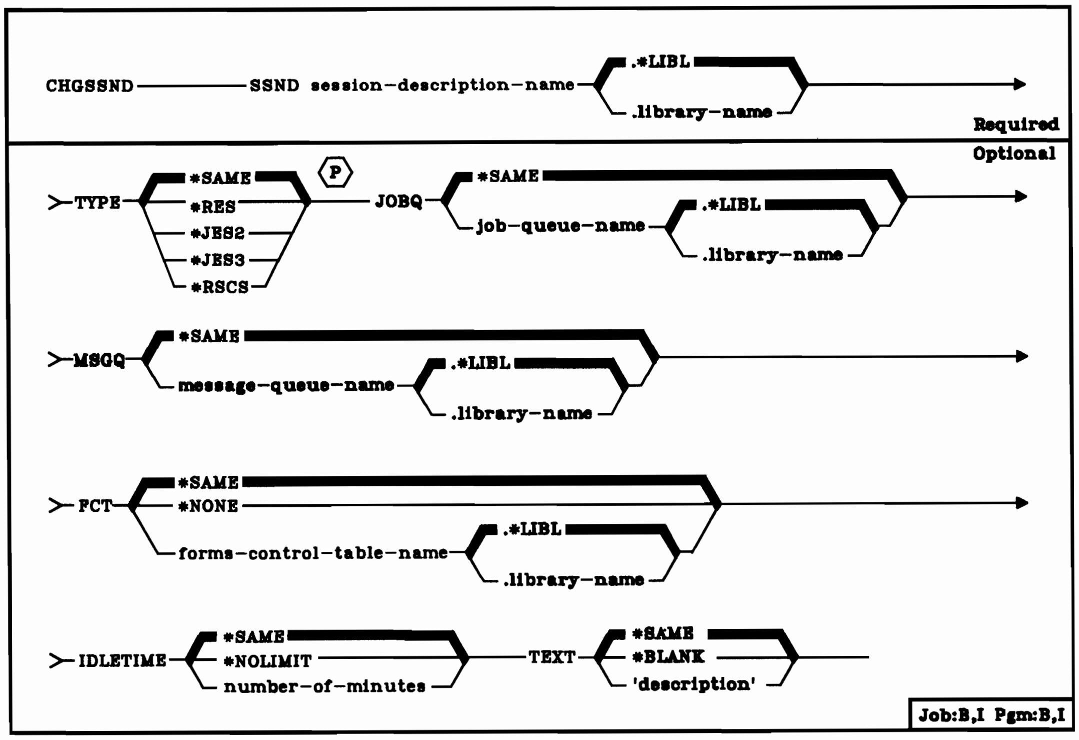
This command changes the expiration date of all members in file INV to October 31, 1987.

CHGSSND (Change Session Description) Command

The Change Session Description (CHGSSND) command changes attributes in an existing RJEF session description.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Change Session Description (CHGSSND) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description that is to be changed. (If no library qualifier is given, *LIBL is used to find the session description.)

TYPE Parameter: Specifies the type of remote job entry host subsystem with which this RJEF session is to communicate. Enter the value that applies to this session description.

*SAME: The host subsystem type for this session description remains the same.

**RES:* VS1/RES.

**JES2:* VS2/JES2.

**JES3:* VS2/JES3.

**RSCS:* VM/370 RSCS.

JOBQ Parameter: Specifies the name of the default RJEF job queue on which all the RJEF session jobs are to be placed. The session jobs are all those jobs associated with RJEF, except the RJEF reader jobs. RJEF reader jobs are placed on the job queues defined in the session description reader entries.

*SAME: The job queue named in the session description remains the same.

job-queue-name: Enter the qualified name of the job queue on which all the RJEF session jobs are to be started. (If no library qualifier is given, *LIBL is used to find the job queue.)

MSGQ Parameter: Specifies the qualified name for the RJEF message queue in which all the RJEF messages are to be recorded.

*SAME: The message queue named in the session description remains the same.

message-queue-name: Enter the qualified name of the message queue that is to contain a record of all the RJEF messages for this session description. (If no library qualifier is given, *LIBL is used to find the message queue.)

FCT Parameter: Specifies a forms control table (FCT) to be used with this session description.

*SAME: The FCT named in the session description remains the same.

**NONE:* No FCT is to be used with this session description.

forms-control-table-name: Enter the qualified name of the FCT that is to be used with this session description. (If no library qualifier is given, *LIBL is used to find the FCT.)

IDLETIME Parameter: Specifies the minimum number of minutes that the RJEF session should remain idle after the line connection has been established before transmitting the LOGOFF or SIGNOFF command to the host system. During this time no files are transmitted or received.

When the number of minutes is set equal to zero, and if the line connection has been established, the LOGOFF or SIGNOFF command is transmitted immediately. Also, RJEF holds all RJEF reader job queues defined for this RJEF session.

The idle time countdown begins following the end-of-file of the last input stream sent or output stream received.

The idle time countdown is reset each time data becomes available for transmitting or receiving.

If there are any input streams that have started but have not ended (that is, received end-of-file) except for the console input streams, the idle time countdown will not begin.

If a Terminate RJE Session (TRMRJESSN) command specifies a controlled cancel, the IDLETIME parameter value of the TRMRJESSN command overrides the CRTSSND command IDLETIME parameter value. This parameter is ignored if OPTION(*IMMED) is specified on the TRMRJESSN command.

SAME: The idle time value, if any, specified in the session description remains the same.

**NOLIMIT:* A LOGOFF or SIGNOFF command is not to be transmitted unless a TRMRJESSN command is issued specifying OPTION(*CNTRLD).

number-of-minutes: Enter the number of minutes that the RJEF session should remain idle before transmitting the LOGOFF or SIGNOFF command to the host system. Valid values are 0 through 99.

TEXT Parameter: Specifies a brief description of the session description. (For an expanded description of the TEXT parameter, see Appendix A.)

**SAME:* The text is to remain as specified when the session description was created.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CHGSSND  SSND(RJE.USERLIB) +
     FCT(FCT1.USERLIB) +
     IDLETIME(30)
```

This command changes the session description named RJE in library USERLIB. The forms control table name is changed to FCT1 in library USERLIB. Also, the idletime is changed to 30 minutes.

CHGSYSVAL (Change System Value) Command

The Change System Value (CHGSYSVAL) command changes the current value of the specified system value. System values are provided as part of the system. They are used by the system to control certain operations in CPF and to communicate the status of certain conditions to the user. Changes to some system values take effect immediately, some do not take effect until new jobs are started, and others do not take effect until CPF is started again. For more information about system values, see the *CPF Programmer's Guide*.

```
                                                                          Required
                                                       ①
CHGSYSVAL ──── SYSVAL system-value-name ──── VALUE new-value ────────────────────

① Some system values can contain a list of values.  They must be enclosed in apostrophes.
                                                                    Job:B,I Pgm:B,I
```

SYSVAL Parameter: Specifies the name of the system value that is to have its value changed. Most of the system values can be specified; however, some cannot have their values changed by this command. (For more information on which values can be specified, see the *CPF Programmer's Guide.*)

VALUE Parameter: Specifies the new value that the system value is to have. Some system values, such as QUSRLIBL and QCTLSBSD, may be made up of multiple character strings. These strings must be separated by blanks; apostrophes must surround the entire contents of the VALUE parameter. For those system values that accept alphabetic characters, any letters that are entered in lowercase (a-z) are translated into uppercase (A-Z) even if they are enclosed in apostrophes. Some system values, such as QDATE and QDBRCVYWT, are zoned-decimal values (character in nature) and must also be enclosed in apostrophes when specified in this parameter. For numeric system values, apostrophes *cannot* be used. (See the *CPF Programmer's Guide* for the descriptions of all system values.) Enter the new value(s) that meet the type, length, and range requirements for that system value.

Examples

```
CHGSYSVAL  SYSVAL(QHOUR)  VALUE('12')
```

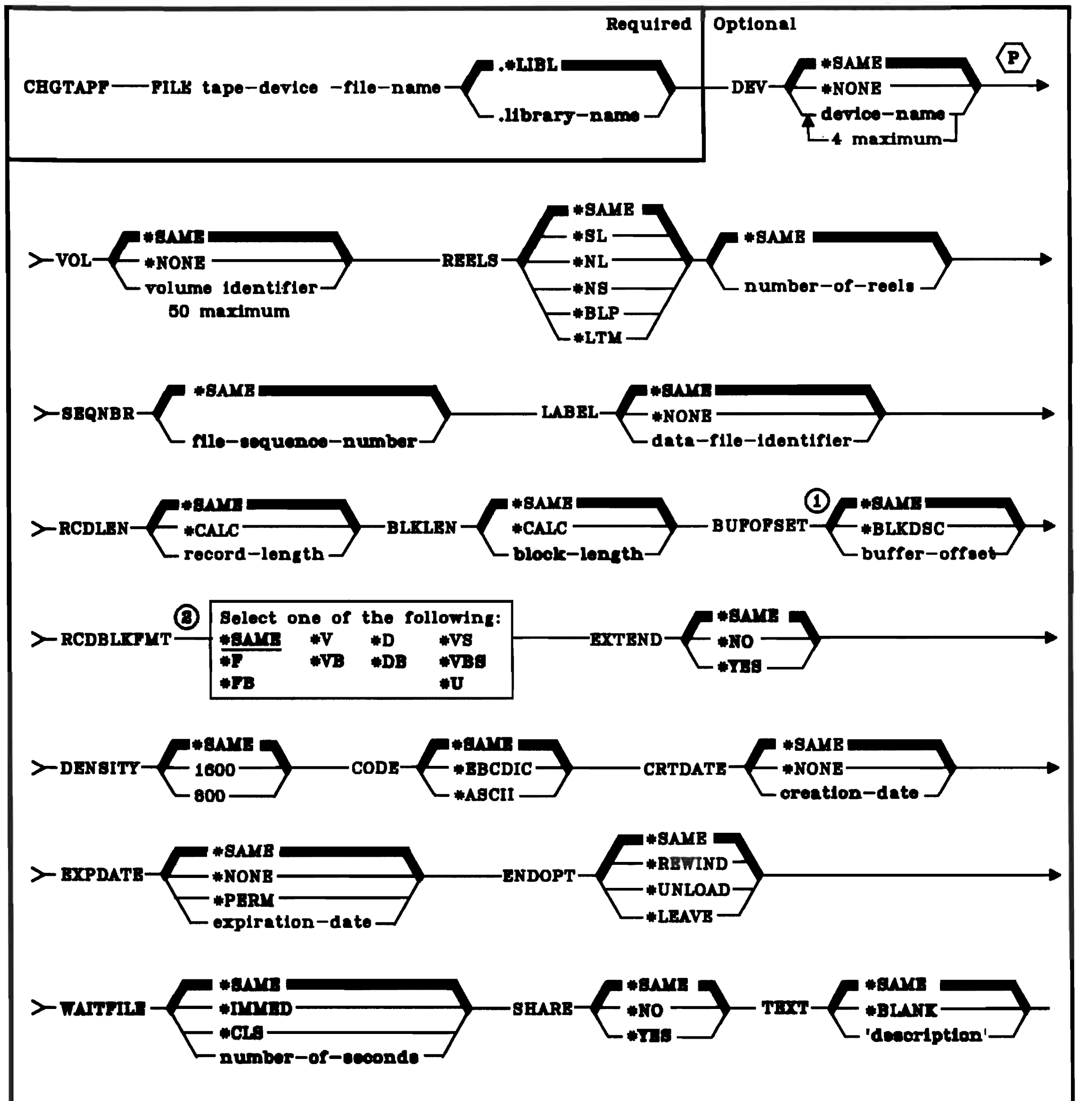
This command changes the value of the system value QHOUR (which is a subvalue of the QTIME system value) to 12. Because QHOUR is a character variable, 2 characters long, the system value is set to the character representation of 12, which is hex F1F2 and, therefore, must be enclosed in apostrophes. Also, the QTIME system value is updated with this value because QHOUR is a subvalue of QTIME.

```
CHGSYSVAL  SYSVAL(QUSRLIBL)  VALUE('INVLIB  STOCKLIB  +
     MYLIB')
```

This command changes the value of the system value QUSRLIBL, which specifies the default list of libraries in the user portion of the library list to be used for a job at the time the job is started. The user portion of the library list is to contain the libraries INVLIB, STOCKLIB, and MYLIB.

CHGTAPF (Change Tape File) Command

The Change Tape File (CHGTAPF) command changes, in the file description, one or more of the attributes of the specified tape device file.



① The value *BLKDSC is valid only if the file is record block format *D or *DB.

② The values *F, *FB, *VS, *VBS and *U are valid for both EBCDIC and ASCII codes; *V and *VB are valid only for EBCDIC; and *D and *DB are valid only for ASCII.

Job:B,I Pgm:B,I

FILE Parameter: Specifies the qualified name of the tape device file whose description is being changed. (If no library qualifier is given, *LIBL is used to find the file.)

DEV Parameter: Specifies the names of one or more tape devices that are to be used with this device file to perform I/O data operations.

*SAME: The device name, if any, specified in the device file description remains the same.

**NONE:* No device names are to be specified. They must be supplied later on an OVRTAPF command or when the tape device file is opened.

device-name: Enter the names of one or more devices (no more than four) that are to be used with this tape device file. The order in which the device names are specified here is the order in which tapes mounted on the devices are processed. Each device must already be known on the system via a device description. When more volumes are to be processed than the number of devices in the DEV list, the devices are used in the same order as specified, wrapping around to the first device as needed.

VOL Parameter: Specifies one or more volume identifiers of tapes to be used by the tape device file. The tapes (volumes) must be mounted on the devices in the same order as the identifiers are specified here (and as they are specified in the DEV parameter). If the tape file is opened for read backward, then the volume identifiers in the list are processed from last to first (while the devices in the device list are used in first to last order). An inquiry message is sent to the system operator for either *SL or *BLP processing if an incorrect volume is mounted, or if no volume is mounted (for any type of label processing). When a list of volume identifiers is provided for the file, operator mount messages indicate the name of the volume which is required. (For an expanded description of the VOL parameter, see Appendix A.)

*SAME: The volume identifiers specified in the device file description remain the same.

**NONE:* No tape volume identifiers are specified for this file. They can be supplied before the device file is opened, either in the CHGTAPF or OVRTAPF command or in the HLL program. If no volume identifiers are specified before the device file is opened, no volume checking is performed beyond verifying that the correct label type volume is mounted, and no volume names are provided in operator mount messages. The maximum number of reels processed for a *NL, *LTM, *NS, or *BLP input file when VOL(*NONE) is specified is determined by the REELS(number-of-reels) parameter value.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used by this device file. Each identifier can have six alphameric characters or less. The maximum number of reels processed for a *NL, *LTM, *NS, or *BLP input file is determined by the number of volume identifiers in the list.

REELS Parameter: Specifies the type of labeling used on the tape reels and the maximum number of reels to be processed, if there is no list of volume identifiers specified (VOL parameter) and this device file is used with either *NL, *LTM, *NS, or *BLP input files. When the number of reels are specified, the volume identifiers on the mounted volumes are ignored if labeled tapes are being processed; the order in which the reels are mounted must be checked by the operator.

The number of reels value (the second part of the REELS parameter) is not a limiting value for standard-label or output files. For a standard-label input file, the data file labels limit the number of volumes processed by indicating end-of-file. For an output file, the maximum number of reels value is ignored; the system requests that additional volumes be mounted until the file is closed.

The system checks the block at the beginning of the tape to see (1) if it has exactly 80 bytes for EBCDIC or at least 80 bytes for ASCII and (2) if the first 4 bytes contain the values VOL and 1. If so, the reel contains a standard labeled tape. *SL and *BLP tape files require standard-label tape volumes. *NL, *LTM, and *NS tape files cannot process standard-label volumes.

Note: The values *SL, *NL, and *LTM can be specified if the device file is to be used for either reading or writing on tapes. The values *NS and *BLP are valid only if the device file is used to read tapes.

SAME: The type of labeling specified in the device file description is not to be changed.

**SL:* The volumes have standard labels. The volume identifiers are to be ignored; instead, the number-of-reels value is to be checked.

**NL:* The volumes have no labels. On a nonlabeled volume, tape marks are used to indicate the beginning and end of the volume and each data file on it.

**NS:* The volumes have nonstandard labels. The load point on the tape may be immediately followed by an optional tape mark and some kind of volume and/or file information, but they are to be ignored. All file label information, if any, contained on the tape is also ignored; instead, the tape marks are to be used to determine the beginning and end of the data file and to determine whether the file is continued on another tape. Only a single data file can exist on a nonstandard tape.

**BLP:* Standard label processing is to be bypassed. Each reel must have standard labels. Although each reel is checked for a standard volume label and each file must have at least one standard header label (HDR1) and one standard trailer label (EOV1 or EOF1), most other label information (such as the data file record length or block length) is ignored. The sequence number of each file on the volume is determined only by the number of tape marks between it and the beginning of tape (in contrast to *SL processing where the file sequence number stored in the header and trailer labels of each file are used to locate a data file). Bypass label processing can be used when some file label information is incorrect.

**LTM:* The volumes have no labels, but have a single leading tape mark before the first data file. REELS(*LTM) is processed the same way as REELS(*NL) except that when SEQNBR(1) is specified for an output file to create the first data file on the tape, a leading tape mark is written at the beginning of the tape before the first data block.

**SAME:* The number of reels specified in the device file description is not to be changed.

number-of-reels: Enter the maximum number of reels that are to be processed for a *NL, *LTM, *NS, or *BLP input tape operation when there is no list of volume identifiers specified (VOL parameter). If the next reel is not mounted when the end of the currently-processing tape is reached, a message is sent to the operator requesting that the next tape be mounted on the next tape device. The number-of-reels value is ignored for a standard label (*SL) file or for any output file.

SEQNBR Parameter: Specifies the sequence number of the data file on the tape that is to be processed. When standard labeled tapes are used, the four-position file sequence number is read from the first header label of the data file. When bypass label processing is used or when standard-labeled tapes are not used, the system uses the tape marks and the value specified (or assumed) here to locate the correct data file to be processed. (When multifile, multivolume tapes are processed using REELS(*SL), the file sequence numbers continue consecutively through all of the volumes; that is, each new data file has a sequence number that is one greater than the previous file, regardless of which volume it is on.)

**SAME:* The file sequence number specified in the device file description is not to be changed.

file-sequence-number: Enter the sequence number of the file to be processed on this tape.

LABEL Parameter: Specifies the data file identifier of the data file that is to be processed by this tape device file. The data file identifier is defined only for standard-label tapes and is stored in the header label immediately preceding the data file that the header describes. If a data file identifier is specified for any type of label processing other than *SL, it is ignored. A label identifier is *required* for a standard label output file, but is optional for an input file (since the sequence number uniquely identifies which data file to process).

For an input file or output file with EXTEND(*YES) specified, this parameter specifies the data file identifier of the file that exists on the tape. The specified identifier must be the same as the one in the labels of the data file that the SEQNBR parameter specifies; otherwise, an error message is sent to the program using this device file. For output files with EXTEND(*NO) specified, the LABEL parameter specifies the identifier of the file that is to be created on the tape. (For an expanded description of the LABEL parameter, see Appendix A.)

SAME: The data file identifier specified in the device file description is not to be changed.

**NONE:* The data file identifier is not specified.

data-file-identifier: Enter the identifier (17 alphameric characters maximum) of the data file to be used with this tape device file. If this identifier is for a tape that is written in the basic exchange format, and it is to be used on a system other than System/38, a maximum of 8 characters should be used or a qualified identifier having no more than 8 characters per qualifier should be used. (See Appendix A for details.)

RCDLEN Parameter: Specifies, in bytes, the length of the records contained in the data file that is to be processed with this device file. The system will always use the record length and block length specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified (if a second header label (HDR2) is found on the tape and *BLP label processing has not been specified).

*SAME: The record length specified in the device file description is not to be changed.

**CALC:* No record length is specified for the data file to be processed. If *CALC is specified the system will attempt to calculate an appropriate record length when the file is opened. RCDLEN(*CALC) can be used for nonlabeled tapes or when there is no HDR2 label if a BLKLEN value other than *CALC is specified for the file and the RCDBLKFMT does not specify spanned or blocked records. In this case, the system calculates an appropriate record length from the block length, record block format, and buffer offset (for an ASCII file) specified for the file. In any other case, the actual record length must be specified by a CHGTAPF or OVRTAPF command, or in the HLL program that opens the device file.

record-length: Enter a value (1 through 32767) that specifies the length of each record in the data file. The minimum and maximum record length that will be allowed for a file is dependent on the record block format, block length, buffer offset (for an ASCII file), and recording code. The following table shows the minimum and maximum record length values allowed for each record block format, assuming the block length value is large enough to support the maximum record length:

Absolute RCDLEN Ranges

		FILETYPE(*DATA)		FILETYPE(*SRC)	
CODE	RCDFBLKFMT	Minimum RCDLEN	Maximum RCDLEN	Minimum RCDLEN	Maximum RCDLEN
*EBCDIC	*F *FB *U	18	32767	30	32767
*ASCII	*F *FB *U	18	32767	30	32767
*EBCDIC	*V *VB	1	32759	13	32767
*ASCII	*D *DB	1	9995	13	10007
*EBCDIC	*VS *VBS	1	32759	13	32767
*ASCII	*VS *VBS	1	32759	13	32767

BLKLEN Parameter: Specifies, in bytes, the maximum length of the data blocks that will be transferred to or from the tape for output or input operations. The system will always use the block length and record length specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified (if a second header label (HDR2) is found on the tape and *BLP label processing has not been specified).

SAME: The block length specified in the device file description is not to be changed.

**CALC:* No block length is specified for the data file to be processed. If *CALC is specified, the system will attempt to calculate an appropriate block length when the file is opened. BLKLEN(*CALC) can be used for nonlabeled tapes or when there is no HDR2 label if a RCDLEN value other than *CALC is specified for the file and the RCDBLKFMT does not specify spanned or blocked records. In this case, the system calculates an appropriate block length from the record length, record block format, and buffer offset (for an ASCII file) specified for the file. In any other case, the actual block length must be specified by a CHGTAPF or OVRTAPF command, or in the HLL program that opens the device file.

block-length: Enter a value, not exceeding 32767 bytes, that specifies the maximum length of each block in the data file to be processed. The minimum block length that can be successfully processed is determined by the tape device hardware and System/38 machine support functions. The minimum value for the 3410/3411 tape drive is 18 bytes. The maximum block length is always 32 767 for an input file, but is limited to 9999 if block descriptors must be created for an ASCII output file. The following table shows the minimum and maximum block length values allowed for an output file:

Absolute BLKLEN Ranges

CPDE	BUFOFSET	Minimum BLKLEN	Maximum BLKLEN
*EBCDIC	ignored	18	32767
*ASCII	0	18	32767
*ASCII	*BLKDSC	18	9999

BUFOFSET Parameter: Specifies the buffer offset value for the start of the first record in each block in the tape data file. A buffer offset value can be used for any record block format ASCII file, and is ignored for an EBCDIC tape file. The system will always use the buffer offset specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified, if a value is contained in the second header label (HDR2) on the tape and *BLP label processing has not been specified.

The buffer offset parameter specifies the length of any information that precedes the first record in the block. For record block formats *D, *DB, *VS, and *VBS each record or record segment is preceded by a descriptor that contains the length of the record or segment. A buffer offset value is used to indicate that there is information *ahead* of the descriptor word for the first record in each block, or *ahead* of the data of the first fixed-length or undefined format record in each block.

This parameter is not needed for a standard label file processed for input if the tape includes a second file header label (HDR2) that contains the buffer offset value. A buffer offset must be provided by the CRTTAPF, CHGTAPF, or OVRTAPF command, or by the file labels for an input file that contains any information (such as a block descriptor) ahead of the first record in each block. If you do not specify a buffer offset when a tape file is created, it is not necessary to specify an offset value when the file is read.

The only buffer offset values allowed for an output file are zero and *BLKDSC. An existing standard label data file with a buffer offset value in the HDR2 label can be extended only if the offset value is either zero or four. An offset of zero in the HDR2 label adds data blocks with *no* buffer offset. BUFOFSET(*BLKDSC) must be specified to extend an existing tape data file that contains an offset value of four in the HDR2 label.

*SAME: The buffer offset value specified in the device file description is not to be changed.

**BLKDSC:* Specifies that 4-byte block descriptors are to be created in any tape file created using this device file, and that any input file read using this device file should assume 4-bytes of buffer offset information preceding the first record in each data block. This value is only valid for a record block format *D or *DB file. When BUFOFSET(*BLKDSC) is specified, the contents of the buffer offset part of each output data block is the actual length of the data block, in zoned decimal format.

buffer-offset: Enter a value (zero through 99) that specifies the length of the buffer offset information which preceeds the first record in each data block.

RCDBLKFMT Parameter: Specifies the type and blocking attribute of records in the tape data file to be processed.

Record block format *V and *VB records can only be processed for an EBCDIC file; *D and *DB records can only be processed for an ASCII file. If a standard label tape (label type *SL or *BLP) is being processed and an inconsistent record block format is specified for the volume code, the correct record type is assumed (V or D) for the volume code and a warning message is sent to the progam that opens the file. If the record type and code are inconsistent for a nonlabeled volume (label type *NL, *LTM, or *NS), an error message is sent and the file is *not* opened, because there are no labels to verify the correct volume code.

If a valid record length, block length, and buffer offset (for an ASCII file) are specified for fixed length records, but the block attribute is incorrect, the correct block attribute will be assumed (changing record block format *F to *FB or record block format *FB to *F), and a warning message will be sent to the program that opens the file.

If a block length is specified that is longer than required to process a maximum length record, then record block format *V, *D, or *VS will be changed to *VB, *DB, or *VBS, and a warning message will be sent to the program that opens the file.

The following chart shows the required relationship between the record length, block length, and buffer offset (for ASCII) file parameters for an output file or an input file where the file parameters are not determined from a second file header label (HDR2):

Required RCDLEN/BLKLEN/BUFOFSET Relation[1]

CODE	RCDBLKFMT	BLKLEN = fcn(RCDLEN,BUFOFSET)
*EBCDIC	*F *U	BLKLEN = RCDLEN
*ASCII	*F *U	BLKLEN = RCDLEN + BUFOFSET
*EBCDIC	*FB	BLKLEN = RCDLEN * n
*ASCII	*FB	BLKLEN = (RCDLEN * n) + BUFOFSET
		n is the number of records in a maximum-length block
*EBCDIC	*V	BLKLEN = RCDLEN + 8
*ASCII	*D	BLKLEN = RCDLEN + 4 + BUFOFSET
*EBCDIC	*VB	BLKLEN >= RCDLEN + 8
*ASCII	*DB	BLKLEN >= RCDLEN + 4 + BUFOFSET
*EBCDIC	*VS *VBS	BLKLEN >= 18
*ASCII	*VS *VBS	BLKLEN >= 6 + BUFOFSET (18 minimum)

[1]When BUFOFSET(*BLKDSC) is specified for the file, a value of 4 should be used for the BUFOFSET part of any BLKLEN calculations, unless existing file labels on the tape specify a different value.

**F:* Fixed length, unblocked, unspanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *FB, based on other file parameters. See the explanation preceding the chart for more information.

**FB:* Fixed length, blocked, unspanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *F, based on other file parameters. See the explanation preceding the chart for more information.

**V:* Variable length, unblocked, unspanned records in EBCDIC type V format are to be processed. The system may change this record block format to *VB, *D, or *DB, based on other file parameters. See the explanation preceding the chart for more information.

**VB:* Variable length, blocked, unspanned records in EBCDIC type V format are to be processed. The system may change this record block format to *DB, based on the volume code. See the explanation preceding the chart for more information.

**D:* Variable length, unblocked, unspanned records in ASCII type D format are to be processed. The system may change this record block format to *DB, *V, or *VB, based on other file parameters. See the explanation preceding the chart for more information.

**DB:* Variable length, blocked, unspanned records in EBCDIC type D format are to be processed. The system may change this record block format to *VB, based on the volume code. See the explanation preceding the chart for more information.

**VS:* Variable length, unblocked, spanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *VBS, based on other file parameters. See the explanation preceding the chart for more information. Note that the representation of spanned records on the tape is different for EBCDIC and ASCII files, but the system selects the correct format based on the file code.

**VBS:* Variable length, blocked, spanned records in either EBCDIC or ASCII code are to be processed. Note that the representation of spanned records on the tape is different for EBCDIC and ASCII files, but the system selects the correct format based on the file code.

**U:* Undefined format records in either EBCDIC or ASCII code are to be processed. RCDBLKFMT(*U) records are processed as variable length records, where each record written or read is in a separate tape block. This format can be useful for processing tape files that do not meet the formating requirements of any other record block format.

EXTEND Parameter: Specifies, for output operations to tape, whether new records are to be added to the end of a data file that is currently on the tape. (The specific data file is identified by the SEQNBR parameter and, for a standard-label file, the LABEL parameter.) If the data file is to be extended, it becomes the last file on the tape volume; any data files that follow it are overwritten as the specified file is extended.

*SAME: The value specified in the device file description is not to be changed.

**NO:* Records are not to be added to the end of the specified data file. Regardless of whether there is already a data file with the specified SEQNBR on the tape, a new data file is created (overwriting an existing data file and any files that follow it).

**YES:* New records are to be added to the end of the specified data file on tape when this device file is used.

DENSITY Parameter: Specifies, in bits per inch, the density of the data that is to be written on the tape volume when this device file is used. This parameter is used only for tapes written as nonlabeled volumes (*NL); it is not valid unless the *first* data file is being written on the nonlabeled volume. The density of a standard-label volume is specified on the INZTAP command, which initializes tapes as standard-label volumes by writing volume labels on them. If a labeled or nonlabeled output file is written with a different density than specified here, a warning message is issued.

SAME: The data density specified in the device file description is not to be changed.

1600: The data density on this tape volume is to be 1600 bits per inch.

800: The data density on this tape volume is to be 800 bits per inch.

CODE Parameter: Specifies the type of character code to be used when tape data is read or written by a job that uses this tape device file. If a labeled volume is recorded in a different code than the value specified for the file, a warning message is sent to the program that opened the file and the volume code is assumed for the file.

SAME: The type of character code specified in the tape file description is not to be changed.

**EBCDIC:* The EBCDIC character code is to be used with this tape device file.

**ASCII:* The ASCII character code is to be used with this tape device file.

CRTDATE Parameter: Specifies, for tape input data files and for tape output for which EXTEND(*YES) is specified, the date when the data file was created (written on tape). The data file creation date is stored in file labels on the tape. If a creation date is specified for any type of label processing other than *SL, it is ignored. If the creation date written on the tape containing the data file does not match the date specified in this device file description, an inquiry message is sent to the operator.

SAME: The creation date of the tape data file specified in the device file description remains the same.

**NONE:* The creation date is not specified. It will not be checked unless it is supplied in the OVRTAPF command or in the HLL program.

creation-date: Enter the creation date of the data file to be used by this device file. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP.

EXPDATE Parameter: Specifies, for tape output data files only, the expiration date of the data file used by this device file. The data file expiration date is stored in file labels on the tape. If an expiration date is specified for any type of label processing other than *SL, it is ignored. If a date is specified, the data file is protected and cannot be written over until the specified expiration date.

SAME: The expiration date of the data file specified in the device file description remains the same.

**NONE:* No expiration date for the data file is to be specified; the file is not to be protected. An expired date is written in the data file labels so the file can be used as a scratch data file.

**PERM:* The data file is to be protected permanently. The date written in the tape data file labels consists of all nines.

expiration-date: Enter the date on which the data file expires. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP.

ENDOPT Parameter: Specifies the positioning operation to be performed automatically on the tape volume when the device file is closed. In the case of a multiple-volume data file, this parameter applies to the *last* reel only; all the other reels are rewound and unloaded when the end of the tape is reached.

SAME: The value specified in the device file description is not to be changed.

**REWIND:* The tape is to be rewound, but not unloaded, after the file is closed.

**UNLOAD:* The tape is to be rewound and unloaded after the file is closed.

**LEAVE:* The tape should be left in its current position when the file is closed; it is not to be rewound or unloaded. This option can be used to reduce the time required to position the tape if the next tape file to open to this device uses a data file that is on the same volume.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*SAME: The wait time specified in the device file description is not to be changed.

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the tape device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

*SAME: The value specified in the device file description is not to be changed.

**NO:* An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

TEXT Parameter: Specifies the user-defined text that describes the tape device file. (For an expanded description of the TEXT parameter, see Appendix A).

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

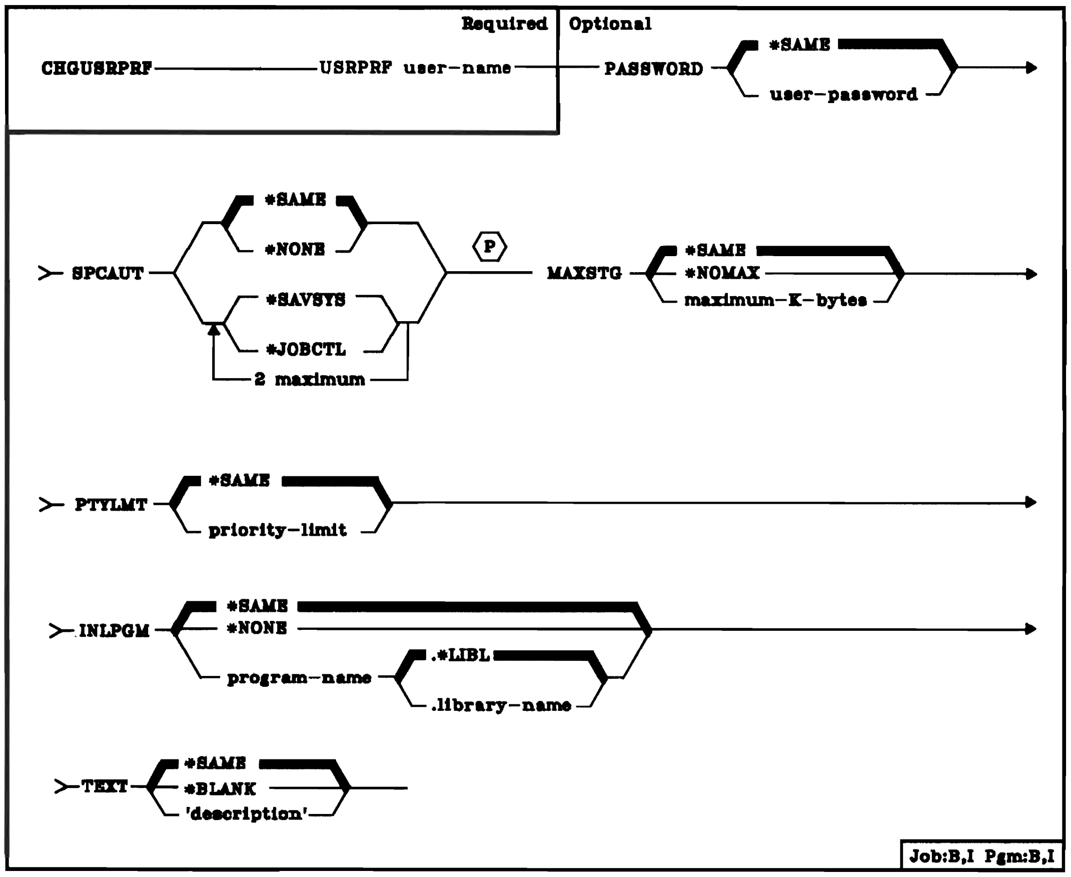
```
CHGTAPF  FILE(TAPE01) LABEL(TUESDAY)
```

This command changes the description of the tape device file named TAPE01. The LABEL parameter now contains the data file identifier TUESDAY.

CHGUSRPRF (Change User Profile) Command

The Change User Profile (CHGUSRPRF) command changes the attributes that were specified for a user in his user profile. The command may be used by the security officer to change the password of each user every month, for example.

Restriction: Only the system security officer can use this command. For the QSECOFR (security officer) user profile, only the PASSWORD, INLPGM, and TEXT attributes can be changed. None of the attributes of the QSYS, QDBSHR, or QSPL user profiles can be changed.



USRPRF Parameter: Specifies the name of the user profile being changed. Enter the name of the user profile that is to have its attributes changed.

PASSWORD Parameter: Specifies the password that lets the user sign on to the system. The password is associated with a unique user profile used by the system to represent the user within the system and to contain his object rights and special rights. The password should be known only to the user(s) himself and to the security officer.

*SAME: The password is not to be changed.

user-password: Enter the alphameric character string (10 characters or less) that identifies the user with his own user profile. The standard rule for specifying names also applies to passwords. The first character must be alphabetic and the other characters must be alphameric.

SPCAUT Parameter: Specifies the special rights that a user is authorized to use. Special rights are *required* to perform certain functions on the system. The special rights are grouped into save system rights (*SAVSYS) and job control rights (*JOBCTL). The security officer can authorize these rights for any user profile; however, *SAVSYS and *JOBCTL are normally given only to the user who operates the system.

*SAME: The special rights, if any, assigned to the user profile are not to be changed.

**NONE:* Any previously granted special rights are to be revoked.

**SAVSYS:* The save system rights are to be granted to the user named in the USRPRF parameter. The named user is given the authority to save, restore, and free storage for all objects on the system, regardless of whether he has object existence rights for the objects.

**JOBCTL:* The job control rights are to be granted to the user named in the USRPRF parameter. The named user is given the authority to change, display, hold, release, and cancel all jobs that are executing on the system or that are on a job queue or output queue that has OPRCTL(*YES) specified.

MAXSTG Parameter: Specifies the maximum amount of auxiliary storage that can be allocated to store permanent objects that are owned by this user profile. If the maximum is exceeded when an interactive user tries to create an object, an error message is displayed and the object is not created. If the maximum is exceeded when an object is created in a batch job, an error message is sent to the job log (depending on the logging level of the job) and the object is not created.

*SAME: The maximum amount of storage that can be allocated to the user remains the same.

**NOMAX:* As much storage as required can be allocated to this profile.

maximum-K-bytes: Enter the maximum amount of storage in K-bytes that can be allocated to this user profile. (1 K equals 1024 bytes.)

PTYLMT Parameter: Specifies the highest scheduling priority that the user is allowed to have for each job that he submits to the system. This value controls the job processing priority and output priority that *any* job running under this user profile can have; that is, values specified in the JOBPTY and OUTPTY parameters of any job command cannot exceed the PTYLMT value of the user profile under which the job is to be run. The scheduling priority can have a value of 1 through 9, where 1 is the highest priority and 9 is the lowest. (For an expanded description of the PTYLMT parameter, see the *Scheduling Priority Parameters* in Appendix A.)

*SAME: The highest scheduling priority that the user can assign to a job remains the same.

priority-limit: Enter a value, 1 through 9, for the highest scheduling priority that the user is allowed.

INLPGM Parameter: Specifies, for an interactive job, the name of the program that is to be invoked whenever a new routing step that has QCL as the request processing program is initiated. (No parameters can be passed to the initial program.) The named program can cause a menu to be displayed or perform some other function. If the initial program fails to function properly, the user may not be able to use the system. However, the security officer can use the CHGUSRPRF command to resolve the problem.

*SAME: The program that is to be invoked after this user signs on remains the same.

**NONE:* No initial program is to be invoked when the user signs on. The command entry display is shown instead.

qualified-program-name: Enter the qualified name of the program that is to be invoked after the user signs on. (If no library qualifier is given, *LIBL is used to find the program.) One of the IBM-supplied programs that can be invoked, if installed, is the QCALLMENU program. This program causes the program call menu to be displayed. This menu is described in the *Programmer's/User's Work Station Guide.*

TEXT Parameter: Specifies the user-defined text that describes the user profile named in the USRPRF parameter. The text specified here replaces any previous text. (For an expanded description of the TEXT parameter, see Appendix A.)

*SAME: The text, if any, is not to be changed.

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CHGUSRPRF  USRPRF(JJADAMS) PASSWORD(SECRET) +
     SPCAUT(*JOBCTL) INLPGM(DSPMENU.ARLIB)
```

This command makes the following changes to the user profile named JJADAMS:

- Changes the password to SECRET.
- Authorizes JJADAMS to use the special job control rights.
- Changes the initial program to be invoked following a successful sign-on to a program named DSPMENU, which is located in a library named ARLIB.

All the other command parameters default to *SAME and are not to be changed.

CHGVAR (Change Variable) Command

The Change Variable (CHGVAR) command is used in CL programs to change the value of a CL variable or part of a character variable (using the substring built-in function). The value can be changed to the value of a constant, to the value of another variable, or to the value obtained from the evaluation of an expression or a built-in function. (Expressions and built-in functions are described in Appendix B.) Also, implicit conversion between decimal and character values is performed according to the rules given in the VALUE parameter description.

The substring built-in function (%SUBSTRING or %SST) can be used in either the VAR or the VALUE parameter of this command. When %SUBSTRING or %SST is specified for VAR, the part of the value of the CL character variable that %SUBSTRING designates is changed to the value of the expression given in the VALUE parameter. When %SUBSTRING or %SST is used for VALUE, the character string specified in VAR is set equal to the part of the character string specified by the substring function designated for VALUE.

The %SWITCH built-in function can be used in the VALUE parameter as a substitute for a logical variable declared in the program. %SWITCH contains an 8-character mask that indicates which of the eight job switches in a job are to be tested for 1s and 0s. When %SWITCH is specified for VALUE, the logical variable specified by VAR is set to a '1' if the logical results of the built-in function are all true. If any of the job switches tested yields a false condition, the variable is set to a '0'.

Restriction: The CHGVAR command is valid only in CL programs.

```
                                                                      Required

CHGVAR ──────── VAR CL-variable-name ──────── VALUE expression ────────

                                                                      Pgm:B,I
```

VAR Parameter: Specifies the name of the CL variable that is to be changed in value. The type of variable does not have to be the same as the type of the constant or variable specified in the VALUE parameter, unless an expression is being evaluated or VAR specifies a logical variable.

If the substring built-in function is to be used to change a portion of a character variable (that is, a substring of the character string in the variable) specified in VAR to a value specified in VALUE, enter the name of the character variable, followed by the starting position and the number of characters to be changed within the character string specified by the variable name.

VALUE Parameter: Specifies the expression that is to be used to change the value of the variable. (Note that variables, constants, or a built-in function can be used within the expression.) For a description of expressions, see Appendix B.

If a constant is to be used as a simple expression, its value must be specified according to the following rules, depending on the type of constant being specified and whether the variable was declared as a decimal, character, or logical variable.

Coding Decimal Values for Decimal Variables. When a numeric value is specified for a decimal variable:

- It can be coded with or without a decimal point (. or ,), and with or without a plus or minus sign.
- If a negative value is to be specified, it must be preceded by a minus (-) sign.
- If a decimal point is not entered in the coded value, it is assumed to be on the right of the last digit entered; that is, the coded value is assumed to be an integer (whole number) only.
- If the number of either integer or fractional digits entered exceeds the defined number of integer or fractional digits, an error occurs.

If, for example, a decimal variable is defined as a five-position decimal value of which two positions are the fraction portion, the following values can be coded:

Coded Value	Assumed Value
2.7 or 2,7	2.70
27 or 27.00	27.00
-27	-27.00

Coding Character Values for Decimal Variables. When a character value is specified for a decimal variable:

- Only the digits 0 through 9, a decimal point (. or ,), and a + or - sign can be used.
- If a + or - sign is specified, it must immediately precede (no blanks between) the first digit in the character value. If no sign character is specified, the value is converted as a positive value.

- The number of decimal positions in the converted result is determined by the decimal point specified in the character value. If no decimal point is specified, it is assumed to be to the right of the last digit in the converted value.

- Decimal alignment occurs in the converted result. (The number of decimal positions in the converted result is determined by the number declared for the variable.) If the specified character value has more decimal positions than the declared variable, the extra positions on the right are truncated. If the integer portion of the character value has more digits than that declared for the variable, an error message is sent to the user.

The following examples show the results of converting the indicated character values for character variable &A to decimal values for decimal variable &B.

```
CHGVAR  VAR(&B) VALUE(&A)
```

Character Variable &A		**Decimal Variable &B**	
Length	**Specified Value**	**Length**	**Converted Result**
10	'ƀƀ+123.1ƀƀ'	5,2	123.10
10	'ƀƀƀƀ123.00'	5,0	123
10	'-123ƀƀƀƀƀƀ'	5,2	-123.00

Coding Character Values for Character Variables. When a character string is specified for a character variable, it must be enclosed in apostrophes if it contains special characters or consists entirely of numeric characters. (For example, 'ABC 67', which contains a blank, or '37.92', which contains a decimal point and consists entirely of numeric characters. If 37.92 is not enclosed in apostrophes, it is treated as a decimal value instead of a character value.)

Character variables are padded with blanks (or are truncated) on the right if the character string for the VALUE parameter is shorter (or longer) than the variable specified by the VAR parameter.

If a character variable is to be set equal to a portion of another character variable, enter, as parameters on the substring built-in function, the name of the variable containing the substring, the starting character position, and the number of characters to be replaced. The starting position and the number of characters can be specified in CL variables.

Coding Decimal Values for Character Variables. When a decimal value is specified for a character variable:

- The same digits, decimal point, and sign character (if the value is negative) are used in the converted result. The value is right-justified in the character variable and padded on the left with zeros, if needed. (This is unique to converted CL decimal values.)

- The converted result has as many decimal positions as were specified in the decimal value or as defined for the decimal variable being used. If no decimal positions are specified in the decimal value or defined for the decimal variable, no decimal point is placed in the result.

- A minus sign is placed in the leftmost position of the character variable if the specified decimal value is negative. No plus sign is placed in the character variable for positive values.

The following examples show the results of converting the indicated decimal values for decimal variable &B to character values for character variable &A.

```
CHGVAR VAR(&A) VALUE(&B)
```

Decimal Variable &B		**Character Variable &A**	
Length	**Specified Value**	**Length**	**Converted Result**
5, 2	23.00 or +23	7	0023.00
5, 2	-3.9	7	-003.90
5, 2	-123.67	7	-123.67

Note: The character variable must be long enough to accommodate the decimal point and sign character if the value can have a decimal point and a negative value in it. In the last example, although the decimal value is defined as (5, 2), the character variable must be at least 7 characters long for the value shown. In the next-to-last example, the character variable could be only 5 characters long and the converted result -3.90 would be valid.

The substring built-in function can be used to change a substring of a character variable specified in the VAR parameter to a decimal value in the VALUE parameter.

Coding Logical or Character Values for Logical Variables. The value for a logical variable must be a logical value of either '1' or '0'. It must be enclosed in apostrophes. Note, however, that the %SWITCH built-in function can be used in place of a logical variable in the VALUE parameter. Refer to Appendix B for a description of the %SWITCH built-in function.

Note: Values for decimal and character variable types can be entered in hexadecimal form (X'580F' for decimal 58.0). However, if character values are entered in hexadecimal form, care should be used because no validity checking is performed on the hexadecimal string.

Examples

The following examples of the CHGVAR command show how the values of decimal, logical, and character variables can be changed.

Changing Decimal Variables

```
CHGVAR &A &B
```

The value of variable &A is set to the value of the variable &B. If &B has a value of 37.2, then &A becomes 37.2 also.

```
CHGVAR &Y (&Y + 1)
```

The value of variable &Y is increased by 1. If &Y has a value of 216, its value is changed to 217.

Changing Logical Variables

```
CHGVAR &X (&Y *OR &Z)
```

The value of the logical variable &X is set to the value of the result of ORing the logical variable &Y with the logical variable &Z. (Both variables *must* be logical variables when *OR is used.) If &Y equals '0' and &Z equals '1', then &X is set to '1'.

```
CHGVAR &A %SWITCH(10XXXX10)
```

The value of the logical variable &A is determined by the logical results of the built-in function %SWITCH. Positions 1, 2, 7, and 8 of the 8-character mask indicate that the corresponding job switches for the job are to be tested for the values indicated in the mask. Job switches 1 and 7 are to be tested for 1s, and switches 2 and 8 are to be tested for 0s. (Switches 3 through 6 are not to be tested.) If all four switches contain the values specified in the %SWITCH mask, the logical result of the built-in function is true, and the variable &A is set to a '1'. If any of the four switches contain a value not indicated in the mask, the result is false and &A is set to '0'.

Changing Character Variables

```
CHGVAR  VAR(&A) VALUE(AB *CAT CD)
CHGVAR  &A ('AB' *CAT 'CD')
```

These two commands set the value of the variable &A equal to the character string ABCD, which is the result of the concatenation performed on the two character strings AB and CD. The first command is coded in keyword form with unquoted strings; the second is coded in positional form with the VALUE parameter specifying two quoted character strings.

```
CHGVAR  &VAR1  &VAR2
```

This example shows a 6-character variable whose value is changed by a shorter character string. If &VAR1 = ABCDEF and &VAR2 = XYZ before the command is executed, the result in &VAR1 is padded on the right with blanks: XYZƀƀƀ.

```
CHGVAR  &VAR1  '12'
```

Assuming &VAR1 is a character variable that is 6 characters long, the result is again padded on the right with blanks: 12ƀƀƀƀ. The apostrophes are required in this example.

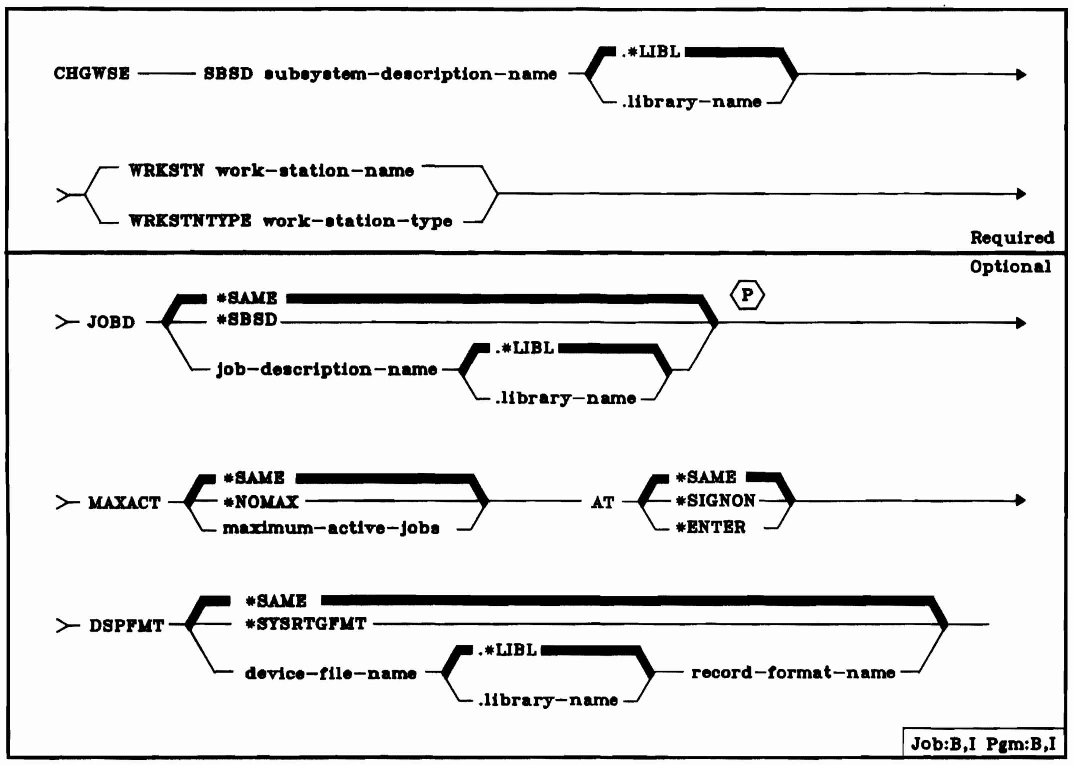
```
CHGVAR  VAR(%SUBSTRING(&A 4 3)) VALUE(REP)
                    or
CHGVAR  VAR(%SST(&A 4 3)) VALUE(REP)
```

The substring built-in function is used to change 3 characters of the character constant in the variable named &A. If &A has a value of ABCDEFGH, the fourth, fifth, and sixth characters in &A are set to REP, and the result is ABCREPGH.

CHGWSE (Change Work Station Entry) Command

The Change Work Station Entry (CHGWSE) command changes one or more attributes of a work station entry in the specified subsystem description; the associated subsystem must be inactive when the changes are made.

Restriction: To use this command, you must have operational and object management rights for the subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description that contains the work station entry
that is to be changed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

WRKSTN Parameter: Specifies the name of the work station whose work station entry is to be changed. The work station must have a type code of CONS, 5251, or 5252 specified in its device description (by the DEVTYPE parameter of the CRTDEVD command).

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

WRKSTNTYPE Parameter: Specifies the type of work station whose work station entry is to be changed. This work entry applies to all work stations of this type that do not have specific work entries for an individual work station. The following type codes are valid:

Type Code	**Device**
*CONS	System console display
5251	5251 Display Station
5252	5252 Dual Display Station

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

JOBD Parameter: Specifies the qualified name of the job description to be used for jobs that are created and processed through this work station entry. If the job description does not exist when this work station entry is being changed, a library qualifier must be specified because the qualified job description name is retained in the subsystem description.

<u>*SAME:</u> The same job description is to be used.

**SBSD:* The job description having the same qualified name as the subsystem description, specified by the SBSD parameter, is to be used for jobs created through this entry.

qualified-job-description-name: Enter the qualified name of the job description that is to be used for jobs created through this entry. If no library qualifier is specified, the library list (*LIBL) of the job in which this CHGWSE command is executed is used to find the job description.

MAXACT Parameter: Specifies, for work stations that use this work station entry, the maximum number of work station jobs that can be concurrently active (or signed on). (For an expanded description of the MAXACT parameter, see Appendix A.)

<u>*SAME:</u> The maximum number of jobs that can be concurrently active is not to be changed.

**NOMAX:* There is no maximum on the number of jobs that can be concurrently active through this entry.

maximum-active-jobs: Enter the new maximum number of jobs that can be concurrently active through this entry.

AT Parameter: Specifies when the work stations associated with this work entry are to be allocated. For more information on how work stations are allocated to subsystems, see the Start Subsystem (STRSBS) command.

Note: The following should be considered if two or more work station entries specify AT(*SIGNON), they apply to the same work station, and they are in more than one subsystem description: If the work station is varied on while more than one of the subsystems are active, you cannot predict to which subsystem the work station will be assigned.

*SAME: The job entry specification is not to be changed.

**SIGNON:* The work stations are to be allocated when the subsystem is started. A sign-on prompt is to be displayed at each work station associated with this work entry. If a work station becomes allocated to a different subsystem, interactive jobs associated with the work station are allowed to enter this subsystem through the Transfer Job (TFRJOB) command.

**ENTER:* The work stations associated with this work entry are not to be allocated when the subsystem is started. However, the interactive jobs associated with the work stations are allowed to enter this subsystem through the TFRJOB command.

DSPFMT Parameter: Specifies the name of the device file and the name of the record format to be used when the subsystem obtains routing data from the user.

*SAME: The value specified in the work station entry is not to be changed.

**SYSRTGFMT:* If routing data is not defined in the referenced job description, the subsystem obtains the initial routing data from the user using the system-supplied routing data format.

qualified-device-file-name record-format-name: Enter the qualified name of the new device file to be used by the subsystem to obtain the routing data. (If no library qualifier is given, *LIBL is used to find the device file description.) If the device file does not exist when the work station entry is changed, a library qualifier must be specified because the qualified name of the device file is retained in the subsystem description. Also, enter the name of the record format to be used when the subsystem obtains the routing data from the user.

Examples

```
CHGWSE  SBSD(BAKER.QGPL) WRKSTN(A12) +
          AT(*SIGNON)
```

This command changes the work station entry for work station A12 in subsystem BAKER found in the general purpose library. A job will be created for work station A12 when a user enters his password on the sign-on prompt and presses the Enter key.

```
CHGWSE  SBSD(BAKER.QGPL) WRKSTN(B28) +
          DSPFMT(*SYSRTGFMT)
```

This command changes the job entry for work station B28 in subsystem BAKER found in the general purpose library. If the routing data format is not defined in the specified job description, the subsystem obtains user data through the system-supplied routing data format.

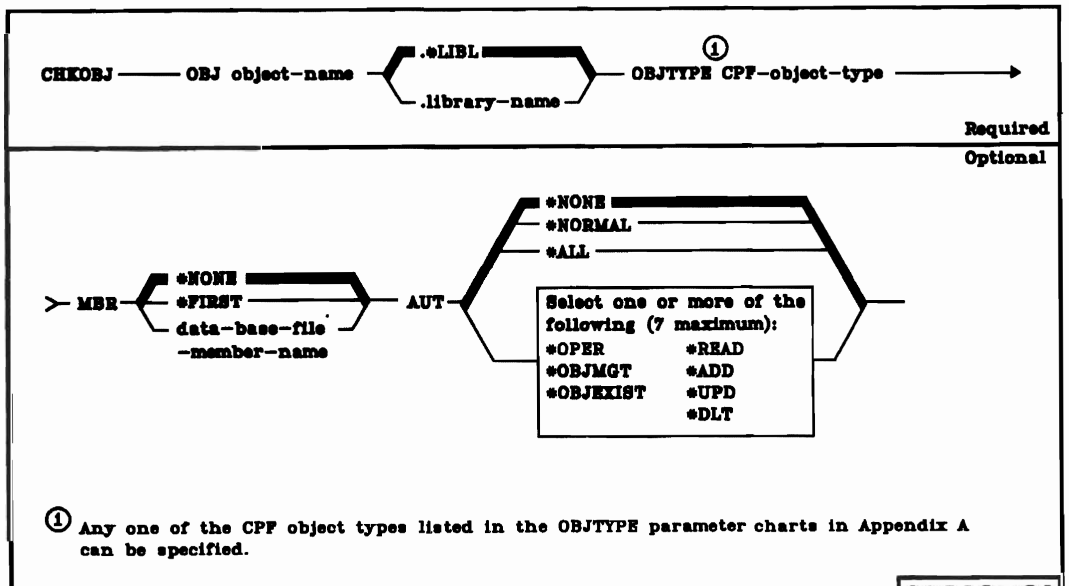
CHKOBJ (Check Object) Command

The Check Object (CHKOBJ) command checks object existence and, optionally, object authorization so that a user can verify that an object exists and verify his rights to the object before trying to access it. For verification, as many as seven specific rights of use can be specified in the command.

These checks can be particularly useful before the user tries to access multiple objects at the same time. The CHKOBJ command is also used to check the validity of object names contained in CL variables and to verify object authorizations under program control.

When the command executes, the system searches for the specified object. If the object is found, the system verifies that the user is authorized for that object in the manner he specified on the CHKOBJ command. If the object is not found or the user does not have the rights specified on the CHKOBJ command, an escape message is sent to the user.

When the CHKOBJ command is used in a CL program, at least one MONMSG command should follow the CHKOBJ command to monitor for any messages that result from the execution of the command. (Refer to Appendix E, *Error Messages That Can Be Monitored*, for the list of error messages that can be monitored for each command.)



① Any one of the CPF object types listed in the OBJTYPE parameter charts in Appendix A can be specified.

Job:B,I Pgm:B,I

OBJ Parameter: Specifies the qualified name of the object being checked. (If no library qualifier is given, *LIBL is used to find the object.)

OBJTYPE Parameter: Specifies the object type of the CPF object that is being checked. Enter the predefined value that specifies the object type. (For an expanded description of the OBJTYPE parameter and a list of the valid values for the CPF object types, see Appendix A.)

MBR Parameter: Specifies, if a member of a data base file is being checked, the name of the file member.

Note: For the specified logical file member, the data rights specified by AUT are checked for each of the physical file members on which the logical file member is based.

NONE: For data base files, *NONE means that no member is to be checked, but the existence and (optionally) the authority of the file itself are to be checked. For all other object types (including device files), *NONE is the only valid value for the MBR parameter.

**FIRST:* The first member in the data base file is to be checked.

data-base-file-member-name: Enter the name of a physical or logical file member that is to be checked by the CHKOBJ command. The values specified for the OBJ and OBJTYPE parameters must be valid for a data base file and the member specified must be a member of the data base file specified in the OBJ parameter.

AUT Parameter: Specifies the rights of use that are to be checked for the specified object.

Note: Refer to the chart in the *CPF Programmer's Guide* that shows the applicable rights of use for each object type.

NONE: Authority is not to be checked.

**NORMAL:* Normal rights of use are to be checked.

**ALL:* All rights of use applicable to the specified object are to be checked.

**OPER:* Operational rights, which provide the authority to use an object and to look at its description, are to be checked.

**OBJMGT:* Object management rights, which provide the authority to manage the access and availability of an object, are to be checked. A user with object management rights can grant (and revoke) the rights he has to an object, as well as move and rename objects and add members to data base files.

**OBJEXIST:* Object existence rights, which provide the authority to control object ownership and existence, are to be checked. This right of use allows the user to delete, save, restore, transfer ownership of, and free the storage of an object.

**READ:* Read rights, which provide the authority to retrieve the contents of an object entry, are to be checked. (See note on MBR parameter.)

**ADD:* Add rights, which provide the authority to add entries to an object, are to be checked. (See note on MBR parameter for checking logical file members.)

**UPD:* Update rights, which provide the authority to change the entries in an object, are to be checked. (See note on MBR parameter.)

**DLT:* Delete rights, which provide the authority to delete entries in an object, are to be checked. (See note on MBR parameter.)

Examples

```
CHKOBJ  OBJ(PROG1.LIB1) OBJTYPE(*PGM)
```

This command checks for the existence of a program named PROG1 in library LIB1. The user's rights of use for PROG1 are not to be checked.

```
CHKOBJ  OBJ(SOURCE1) OBJTYPE(*FILE) +
          MBR(MBR3) AUT(*NORMAL)
```

This command checks the user's authority for normal rights of use to member MBR3 in the file SOURCE1.

```
CHKOBJ  OBJ(PROG1.LIB1) OBJTYPE(*PGM) AUT(*NORMAL)
```

This command checks the existence of and the user's rights of use for PROG1 in LIB1.

The following list identifies messages that can be monitored by the Monitor Message (MONMSG) command if sent by the CHKOBJ command:

CPF9801	OBJECT NOT FOUND – PROG1 does not exist.
CPF9802	OBJECT NOT AUTHORIZED – The user that issued this command does not have *NORMAL authority to PROG1.
CPF9810	LIBRARY NOT FOUND – LIB1 cannot be located.
CFP9820	NOT AUTHORIZED TO LIBRARY – The user that issued this command is not authorized to the library named LIB1, or is not authorized to a library in the library search list named LIB1.
CPF9830	UNABLE TO ALLOCATE LIBRARY – The library named LIB1 or a library in the library search list named LIB1 is locked and cannot be accessed.

```
CHKOBJ  OBJ(FILEA) OBJTYPE(*FILE) MBR(MBR1) AUT(*NORMAL)
```

This command checks the user's authority for normal rights of use to logical file member MBR1, and each physical file member on which MBR1 is based.

The following are messages that can be monitored by the MONMSG command, in addition to the messages shown in the previous example:

CPF9815	MEMBER IN FILE NOT FOUND – MBR1 cannot be found in FILEA. If FILEA does not contain members, a CPF001 (invalid parameter) is sent. If FILEA is a device file, a CPF9899 message is sent.
CPF9899	FUNCTION NOT PERFORMED – This message is a summary escape message that is always preceded by a diagnostic message. If FILEA is a device file, message CPF2168 precedes message CPF9899. If FILEA is locked, message CPF3202 precedes this message. If MBR1 is a logical data base file member and *ALL, *READ, *UPD, or *DLT is specified, message CPF9899 is preceded by diagnostic message CPF3274.

```
CHKOBJ  OBJ(FILEA) OBJTYPE(*FILE) MBR(MBR1) +
     AUT(*ADD *DLT)
MONMSG  MSGID(CPFXXXX) EXEC(GOTO ERROR1)
```

These two commands are used to verify that the user has both add and delete rights for each of the physical file members on which the logical file member MBR1 in the logical file FILEA is based. If he does not have both of the data rights for all of the based-on physical file members, the escape message CPF9802 is sent to the program, and control is passed in the program to the command that has the label ERROR1.

CLNPRT (Clean Printer) Command

The Clean Printer (CLNPRT) command is used to clean the type faces of the print train character slugs on the 3203 Printer. For instructions on preparing the 3203 to clean the print train, refer to the *IBM 3203 Printer Model 5 Component Description and Operator's Guide,* GA33-1529.

```
                                                                  Required
  CLNPRT ——— DEV 3203-device-name ———
                                                           Job:B,I Pgm:B,I
```

DEV Parameter: Specifies the device name assigned to the 3203 Printer on which the print train is to be cleaned.

Example

```
CLNPRT  DEV(PRINTER1)
```

This command cleans the print train mounted on the 3203 Printer named PRINTER1.

CLRDKT (Clear Diskette) Command

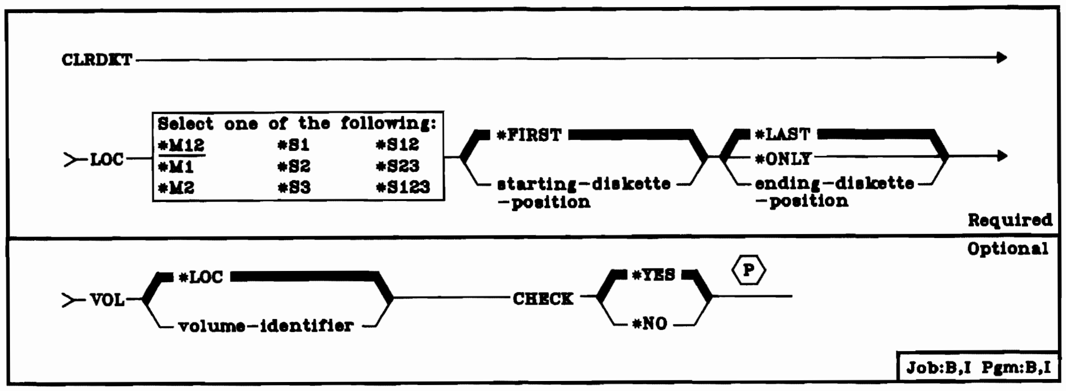
The Clear Diskette (CLRDKT) command deletes all files, active and inactive, from one or more diskettes by deleting the data file identifiers from the diskette label area on each diskette. A single (expired) file is defined, covering the entire diskette, and is identified as DATA. The data contained in the files is not erased. Refer to *DLTDKTLBL (Delete Diskette Label) Command* and the *INZDKT (Initialize Diskette) Command* to erase the data in the files.

The CLRDKT command does not test the diskette for defects nor does it change the volume identifier and owner identifier fields. The error map also is not altered.

A maximum of two magazines or three diskettes in manual slots can be mounted and cleared by one CLRDKT command. If no volume identifier is specified, the command can clear more than one volume at a time, either in the basic exchange format or in the save/restore format. If an identifier is specified that is the same on multiple diskettes currently mounted in magazines or slots, all diskettes with an identical volume identifier are cleared. (The volume identifier of a save/restore *magazine* cannot be specified because the last character (diskette position number) in the identifier changes for each diskette in that magazine.)

Note: When processing diskettes with non-IBM standard labels, you may get unpredictable results. To initialize the diskette, execute the Initialize Diskette (INZDKT) command, with CHECK(*NO) specified.

Restriction: A diskette that has an extended label area cannot be cleared; it must be initialized by the INZDKT command.



LOC Parameter: Specifies which diskette location(s) in the magazines or slots are to have their diskettes cleared. Three values are needed: (1) the unit type and location (that is, the magazines or slots used), (2) the starting diskette position, and (3) the ending diskette position in the unit. (For an expanded description of the LOC parameter, see Appendix A.) A value must be specified for the first of the three values; if no values are specified for the other two, *FIRST and *LAST are assumed by the system.

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit (magazine or slot) and diskette location are to be cleared. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette that is to be cleared first. Enter one of the following values to specify the starting diskette position:

*FIRST: The first diskette position in the location contains the diskette to be cleared first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the first diskette to be cleared. (A value is not valid for manual slots.)

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette that is to be cleared last. Enter one of the following values to specify the ending diskette position:

*LAST: The last diskette position in the location contains the diskette to be cleared last. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

ONLY: Only the diskette position specified by the second value is to be cleared.

ending-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the last diskette to be cleared. (A value is not valid for manual slots.)

VOL Parameter: Specifies whether a check of the volume identifier field on the diskette should be made before the specified diskettes are cleared. If so, the volume identifier of the volume to be checked must be specified.

*LOC: No volume identifier check is to be made; the diskettes currently mounted in the location specified by the LOC parameter are to be cleared without checking. If multiple diskettes in the save/restore format are to be cleared, *LOC is the only valid value for VOL.

volume-identifier: Enter a volume identifier that is to be compared with the diskette label volume identifier field on the diskette being cleared. The identifier can have no more than 6 characters; any combination of letters and digits can be used. For magazines in the save/restore format, if only a single diskette is to be cleared, both the *magazine* identifier (5 characters maximum) and the diskette's position in the magazine must be specified. (For example, the volume identifier SVLIB4 indicates diskette 4 in the magazine volume SVLIB is to be cleared. Note that, for LOC, *M1 or *M2 followed by 4 and *ONLY must also be specified.)

If the volume identifiers do not match, a message is issued to the system operator. The operator can then either insert the correct diskette and try again or continue with the next diskette as specified by the LOC parameter.

Note: If multiple diskettes having identical volume identifiers are mounted in the location specified by LOC, all the diskettes that are identically named are cleared.

CHECK Parameter: Specifies whether a check for active files is to be performed on each diskette in the specified location before it is to be cleared. Active files are files having an expiration date greater than the system date.

*YES: A check is to be performed on files whose labels are in cylinder 0 only. File labels in an extended file label area (not supported by System/38) are not checked. If any active files are found on a diskette, a message is sent to the system operator. The operator can continue the clear function, destroying any active files, or he can terminate the operation. If more than one diskette is being cleared, the process continues on the next diskette in the sequence.

**NO:* The diskettes are to be cleared without being checked for active files.

Examples

```
CLRDKT  LOC(*M1 5 *ONLY) VOL(MASTER)
```

This command clears only the fifth diskette in magazine 1 if its volume identifier is MASTER.

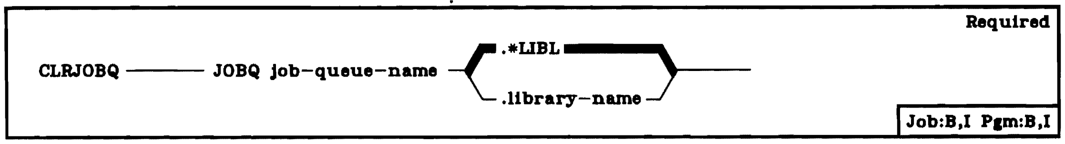
```
CLRDKT  LOC(*M12)
```

This command clears all diskettes in magazines 1 and 2. Because VOL(*LOC) is assumed, the diskettes could be in either the basic exchange or save/restore formats, and a volume identification check is not made. However, because CHECK(*YES) is also assumed, each diskette (in both magazines) is checked for active files before it is cleared.

CLRJOBQ (Clear Job Queue) Command

The Clear Job Queue (CLRJOBQ) command removes, from the specified job queue, all the job entries for batch jobs (including jobs that are in the hold state). Any jobs that are currently being read in and any interactive jobs that have been rerouted to the job queue remain on the queue. The execution of jobs that were started from the job queue is not affected.

Restriction: You must have read, add, and delete rights for the job queue; or you must have job control rights and the job queue must have OPRCTL(*YES) specified, which allows you to clear the queue.



JOBQ Parameter: Specifies the qualified name of the job queue that is to be cleared of all waiting or held jobs. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
CLRJOBQ  JOBQ(QBATCH)
```

This command removes all jobs currently in the IBM-supplied job queue, QBATCH. Any job currently being read in is not affected.

CLRLIB (Clear Library) Command

The Clear Library (CLRLIB) command deletes all of the objects from the specified library that a user has the authority to delete. The CLRLIB command does not delete the specified library, only the objects for which the user has object existence authority; the other objects remain in the library. If any objects are being used by any other job when this command is entered, those objects are not deleted.

Restrictions: (1) The user must have operational rights for the library being cleared as well as object existence rights for the objects to be deleted. (2) This command cannot be used to clear the QSYS library.

```
                                                              Required
CLRLIB ——— LIB library-name ———
                                                     Job:B,I Pgm:B,I
```

LIB Parameter: Specifies the name of the library that is to be cleared of all objects that the user has object existence authority for. If the user does not have object existence rights for an object, that object remains in the library.

Example

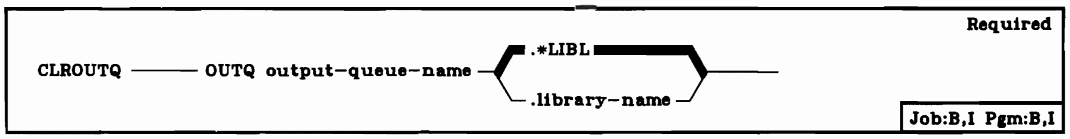
```
CLRLIB  LIB(A)
```

This command deletes all of the objects in library A for which the user has object existence authority.

CLROUTQ (Clear Output Queue) Command

The Clear Output Queue (CLROUTQ) command removes from the specified queue the entries for all spooled files that are waiting to be written on an output device, including files that are in the hold state. Spooled output files that are currently being produced by programs or that are being written to an output device are not removed from the queue.

Restriction: You must have read, add, and delete rights for the output queue; or you must have job control rights and the output queue must have OPRCTL(*YES) specified.



OUTQ Parameter: Specifies the qualified name of the output queue that is to be cleared. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
CLROUTQ  OUTQ(QPUNCH)
```

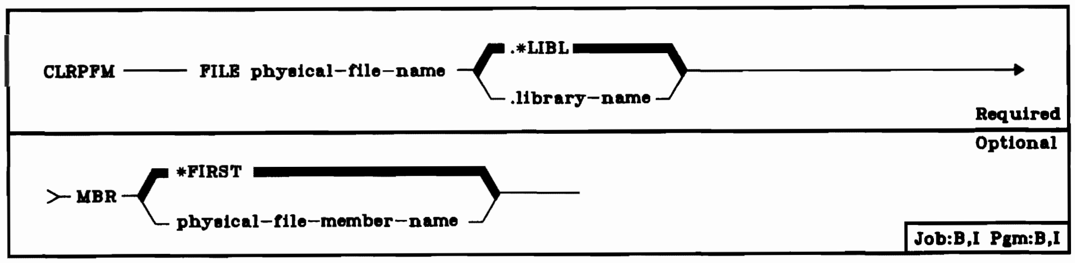
This command removes from the output queue, QPUNCH, the entries for all spooled files that are waiting to be punched or are being held. The entries for the file currently being punched and those files still receiving records from executing programs are not affected.

CLRPFM (Clear Physical File Member) Command

The Clear Physical File Member (CLRPFM) command removes all the data (including deleted records) from the specified member of a physical file. The record count for the member is set to zero, and the member size is set to the optimum size (as determined by the system), depending upon the manner in which the file was created. For more information, refer to the ALLOCATE parameter for the *CRTPF (Create Physical File) Command.* Any attempt to retrieve a record from the cleared member results in an error message being sent to the user or program that attempted the retrieve.

Note: The CLRPFM command ignores all file overrides that are currently in effect for the job.

Restrictions: To clear a member, the user must have object management and delete rights for the physical file that contains the member. If any of the access paths to the member are in use when this command is entered, the command is not executed. Also, if MAINT(*IMMED) is specified for any access path associated with this physical file member and that access path or any other physical file member associated with that access path is currently open for update (this may be through another logical file), the clear operation does not occur.



FILE Parameter: Specifies the qualified name of the physical file that contains the member to be cleared. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name of the member, or the first member, to be cleared.

<u>*FIRST:</u> The first member of the specified physical file is to be cleared.

physical-file-member-name: Enter the name of the physical file member to be cleared.

Example

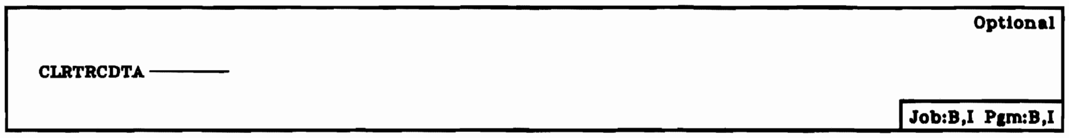
```
CLRPFM  FILE(INV.QGPL)  MBR(FEB)
```

The member named FEB in the physical file INV that is stored in the QGPL library is to be cleared. It cannot be cleared until all jobs currently using the member and all jobs using the access paths over the member are finished with it.

CLRTRCDTA (Clear Trace Data) Command

The Clear Trace Data (CLRTRCDTA) command is used to clear (destroy) all of the data from any previous trace operations in this debugging session. Once cleared, the data can no longer be displayed.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command*.



Example

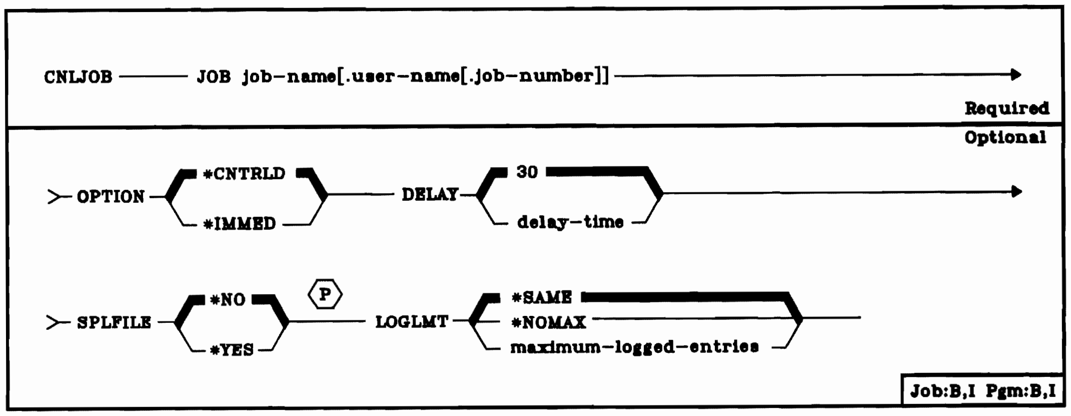
```
CLRTRCDTA
```

This command clears all of the data recorded from any and all previous tracing operations in all of the programs currently being debugged.

CNLJOB (Cancel Job) Command

The Cancel Job (CNLJOB) command cancels the specified job and its associated inline data files, if any. The job may be on a job queue, it may be active within a subsystem, or it may have already completed execution. All spooled output files associated with the job being canceled can also be canceled or allowed to remain on the output queue.

Restriction: To use this command, you must be canceling your own job or you must have the special job control rights.



JOB Parameter: Specifies the qualified name of the job to be canceled. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. If duplicates of the specified name are found, a qualified job name must be specified. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

OPTION Parameter: Specifies whether the job is to be canceled in a controlled manner (which lets the application program perform termination processing) or immediately. In either case, the system does perform certain job cleanup functions.

CNTRLD: The job is to be terminated in a controlled manner. This allows the executing program to perform cleanup (termination processing).

**IMMED:* The job is to be terminated immediately, meaning the executing program is not allowed to perform any cleanup. (This option might cause undesirable results if data has been partially updated and, therefore, should be used only after a controlled cancel has been attempted unsuccessfully.)

DELAY Parameter: Specifies the amount of time (in seconds) allowed, for the routing step to complete its cleanup processing during a controlled cancel. This parameter is not used if OPTION(*IMMED) is specified. If the cleanup is not completed before the end of the delay time, the job is immediately canceled. (Only system cleanup is performed.)

30: A maximum delay time of 30 seconds is allowed for cleanup before the job is canceled.

delay-time: Enter the maximum amount of delay time in seconds before the job is canceled. Valid values are 1 through 999999 seconds.

SPLFILE Parameter: Specifies whether spooled output files created by this job are to be retained for normal processing a writer or whether they are to be deleted.

*NO: The spooled output files created by the job being canceled are to be retained for normal processing by a writer.

**YES:* The spooled output files created by the job being canceled are to be deleted. The job log is not deleted.

LOGLMT Parameter: Specifies the maximum number of entries, in the message queue of the job being canceled, that are to be written to the job log. This parameter can be used to limit the number of messages written to the job log printer file (QPJOBLOG) for a job that is being canceled. This option is particularly useful when a job is canceled and its message queue contains an excessive number of entries.

If the CNLJOB command is used to change the message logging limit while the messages for the canceled job are being written to the spooled file, and the new limit is greater than the number written at the time the command is entered, messages continue to be written until the new limit is reached. If the new limit is less than the number of messages already written to the spooled file, a message indicating that the limit has been reached is immediately put in the spooled file as the last entry, and the rest of the messages on the queue are ignored. If the limit is set to zero before any messages are written to the spooled file, no job log is produced for the canceled job.

*SAME: The message logging limit is not to be changed. (If the logging limit was not changed for this job on a previous command, *NOMAX is the value used by the system.)

**NOMAX:* There is no limit on the number of messages to be logged; *all* messages on the job message queue are to be written to the job log.

maximum-logged-entries: Enter a value that specifies the maximum number of messages to be written to the job log. This value is the maximum only if it is entered *before* the job log contains that many messages; otherwise, the limit just stops the process of writing any more messages to the job log. If 0 is specified before any messages are written to the log, no job log is produced.

Examples

```
CNLJOB  JOB(PAYROLL)  OPTION(*IMMED)  SPLFILE(*YES)
```

This command cancels a job called PAYROLL immediately. Any spooled output produced by the job is deleted; the job log is saved.

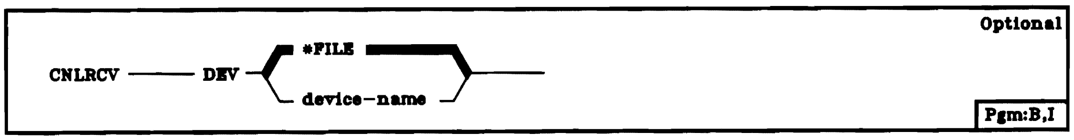
```
CNLJOB  JOB(WSTATION2)  OPTION(*CNTRLD)  +
      DELAY(50)  SPLFILE(*NO)
```

This command cancels a job called WSTATION2. Any spooled output is saved for normal processing by the spooling writer. The job has 50 seconds to perform any cleanup routines, after which it is canceled immediately.

CNLRCV (Cancel Receive) Command

The Cancel Receive (CNLRCV) command is used to cancel a request for input made by a previously issued RCVF or SNDRCVF command that had WAIT(*NO) specified. The CNLRCV command will cancel an input request even if the user enters the requested data at the work station at the same time that the command is executed. If the requested data is entered and is enroute to the program when the cancel receive operation is performed, the entered data is lost. If there is no outstanding input request, the command is ignored.

Restriction: This command is valid only within CL programs.



DEV Parameter: Specifies the name of the display device for which the request for input is to be canceled.

*FILE: The name of the device having the response from it canceled is contained in the device file that was declared in the FILE parameter of the DCLF command. If the device file has more than one device name specified in it, *FILE cannot be specified.

device-name: Enter the name of the display device from which a response is being canceled.

Example

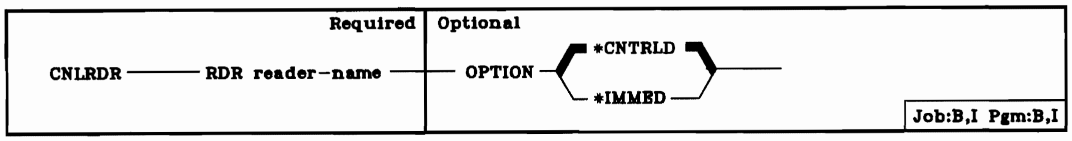
```
CNLRCV  DEV(MYDISPLAY)
```

In this example, assume that a RCVF command with WAIT(*NO) was issued earlier in the CL program to request input from the device file declared earlier in the DCLF command and from the display device MYDISPLAY. When this CNLRCV command is executed, that request for input from MYDISPLAY is canceled.

CNLRDR (Cancel Reader) Command

The Cancel Reader (CNLRDR) command terminates the specified card, diskette, or data base reader and makes its associated input device available to the system. The reader can be terminated either immediately, without completing the current job being read, or at the end of the current job. If the reader is in a hold state when this command is issued, the reader is terminated immediately.

Restriction: To cancel a reader, you must have started the reader or you must have the special job control rights in your user profile.



RDR Parameter: Specifies the name of the card, diskette, or data base reader to be canceled. The reader's associated input device is made available to the system.

OPTION Parameter: Specifies when the canceled reader should terminate processing.

*CNTRLD: The reader is to terminate processing after the current job is read and an entry for the job is placed on the job queue.

**IMMED:* The reader is to terminate processing immediately. The job being read in is not placed on the job queue.

Example

```
CNLRDR  RDR(CARD)
```

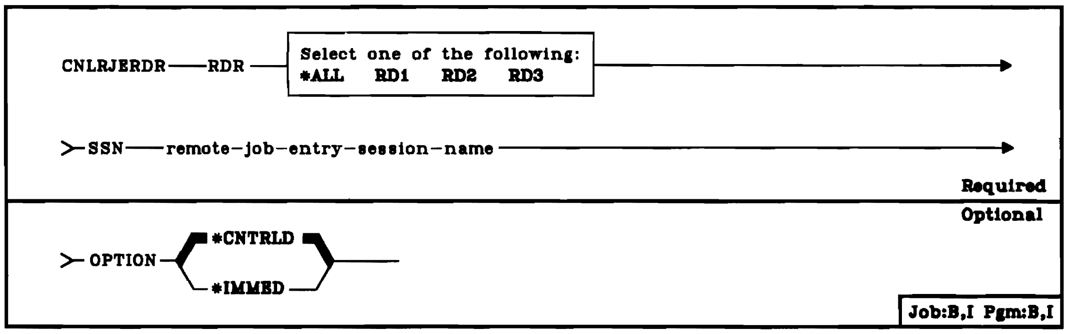
This command stops the reader CARD as soon as the current job is completely read in and releases that device to the system. To process any jobs that remain in the input stream, another reader can be started, but the system operator may have to put the job card and other cards needed for the next job back in the hopper with the rest of the input stream.

CNLRJERDR (Cancel RJE Reader) Command

The Cancel RJE Reader (CNLRJERDR) command cancels the specified RJEF reader job and holds the associated RJEF reader job queue. Other RJEF reader job queues are not affected.

Restriction: To use this command, you must have operational rights to the session description.

The Cancel RJE Reader (CNLRJERDR) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



RDR Parameter: Identifies the RJEF reader that is to be canceled.

**ALL:* All RJEF readers associated with the specified RJEF session are to be canceled.

RD1: RJEF Reader 1 is to be canceled.

RD2: RJEF Reader 2 is to be canceled.

RD3: RJEF Reader 3 is to be canceled.

SSN Parameter: Specifies the name of the RJEF session in which the RJEF reader is to be canceled.

OPTION Parameter: Specifies when the canceled RJEF reader should terminate processing. For both parameter values, issuing a Start RJE Reader (STRRJERDR) command is required to resume RJEF reader operations.

CNTRLD: The specified RJEF readers are to terminate processing in a controlled manner by holding the RJEF job queue associated with the specified RJEF reader(s). Controlled termination prevents any new RJEF reader jobs from executing and allows the job currently executing to complete normally.

**IMMED:* The specified RJEF readers are to terminate processing immediately. No more data records are sent to the host system and no new RJEF reader jobs are allowed to start. A normal end-of-file sequence is sent to the host system.

Example

```
CNLRJERDR RDR(RD1) +
     SSN(RJE) +
     OPTION(*IMMED)
```

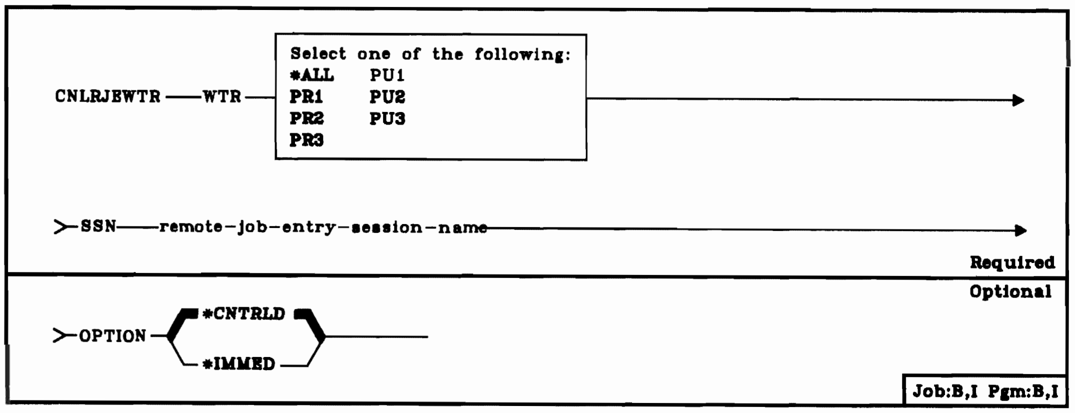
This command cancels reader 1 in the active RJEF session named RJE. The reader is canceled immediately. The file currently being sent to the host by RD1 is not allowed to complete. If RD1 was started from an RJEF reader job queue, the job queue is held. No new files will be sent to the host by RD1 until it is restarted by the Start RJE Reader (STRRJERDR) command.

CNLRJEWTR (Cancel RJE Writer) Command

The Cancel RJE Writer (CNLRJEWTR) command cancels the specified RJEF writer.

Restriction: To use this command, you must have operational rights to the session description and read rights to the library in which the session description is stored.

The Cancel RJE Writer (CNLRJEWTR) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



WTR Parameter: Identifies the RJEF writer that is to be canceled.

**ALL:* All RJEF writers associated with the specified RJEF session are to be canceled.

PR1: RJEF Printer 1 output stream is to be canceled.

PR2: RJEF Printer 2 output stream is to be canceled.

PR3: RJEF Printer 3 output stream is to be canceled.

PU1: RJEF Punch 1 output stream is to be canceled.

PU2: RJEF Punch 2 output stream is to be canceled.

PU3: RJEF Punch 3 output stream is to be canceled.

SSN Parameter: Specifies the name of the RJEF session in which the RJEF writer is to be canceled.

OPTION Parameter: Specifies when the canceled RJEF writer should terminate processing. For both parameter values, issuing a Start RJE Writer (STRRJEWTR) command is required to resume RJEF writer operations.

*CNTRLD: The specified RJEF writers are to terminate processing in a controlled manner. Controlled termination prevents any new writer jobs from being accepted from the host system for the specified RJEF writer. RJEF writer jobs currently executing are allowed to complete normally.

**IMMED:* The specified RJEF writers are to terminate processing immediately. No new RJEF writer jobs are allowed to start, which causes the data set at the host system to be placed again on the output queue.

Note: For an RES host system, the data set is placed again on the output queue and held. In order to release the data set, enter the following command from the RJE console:

```
RELEASE jobname, OUT=x
```

where x is the output class. (The host system 'D N' command can be used to show the names of any jobs held and their output class.)

Example

```
CNLRJEWTR WTR(PR1) +
      SSN(RJE) +
      OPTION(*IMMED)
```

This command cancels printer 1 (PR1) in the active RJEF session named RJE. The printer is canceled immediately. The file currently being received from the host system by printer 1 is not allowed to complete, but is placed again on the output queue. No new files will be accepted from the host system by printer 1 until it is restarted by the Start RJE Writer (STRRJEWTR) command.

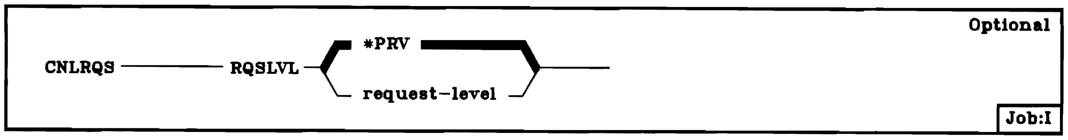
CNLRQS (Cancel Request) Command

The Cancel Request (CNLRQS) command cancels a previously requested operation (command). This command can be entered during a breakpoint that occurs in a program being tested or it can be entered in response to a message that was not monitored by the executing program. If this command is entered in response to an unmonitored message, it cancels the request that caused the message.

When a request is canceled, an escape message (see Appendix E) is sent to the request processing program that is currently invoked at the request level being canceled. Request processing programs can monitor for the escape message so that cleanup processing can be performed when the request is canceled. Otherwise, the executing program is not allowed to perform any termination processing, but the static storage and the files associated with the request are reclaimed.

Note: External objects that are locked by the Allocate Object (ALCOBJ) command are not unlocked (deallocated) by the cancel request.

The CNLRQS command can only be used interactively with nested commands. For more information on nested commands, see the *Programmer's/User's Work Station Guide.*



RQSLVL Parameter: Specifies the command (request) nesting level at which the command to be canceled was entered.

*PRV: The command entered at the immediately previous level is to be canceled.

request-level: Enter the number of the command nesting level at which the command to be canceled was entered. All nesting levels from the level specified to the current level are canceled.

Examples

```
CALL PROGA        (This is level 1)
   •
   •
   •
   Breakpoint occurs

CALL PROGB        (This is level 2)
   •
   •
   •
   Breakpoint occurs

CNLRQS            (This is level 3)
```

In this example, because RQSLVL(*PRV) is the default, the request made at level 2 is canceled. The user can then enter another command at level 2 or press the CF1 key to redisplay the PROGA breakpoint display.

```
CALL PROGA        (This is level 1)
   •
   •
   •
   Breakpoint occurs

CALL PROGB        (This is level 2)
   •
   •
   •
   Breakpoint occurs

CNLRQS RQSLVL(1)      (This is level 3)
```

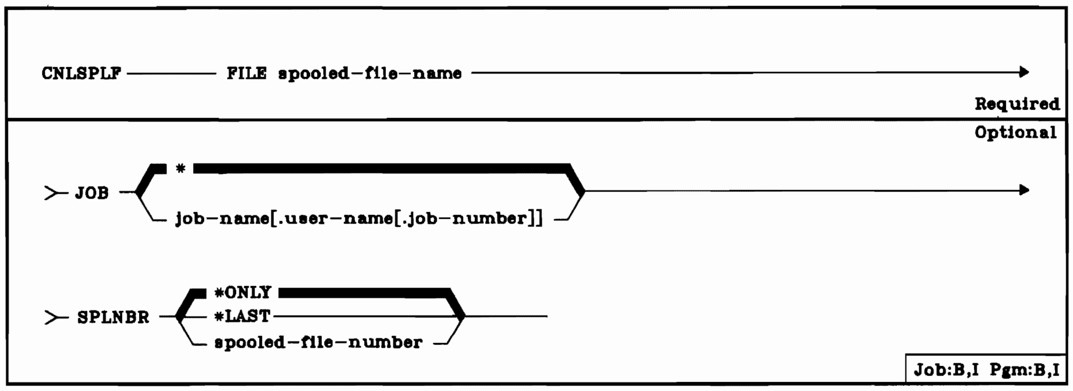
In this example, the request made at the highest level (CALL PROGA) is canceled. Consequently, any requests made between level 1 and level 3 are also canceled.

CNLSPLF (Cancel Spooled File) Command

The Cancel Spooled File (CNLSPLF) command is used to remove the specified spooled output file from the output queue. If the spooled file is currently being produced on a device, it is immediately stopped and removed. Any output that has not been produced is lost. Only one file can be canceled with a CNLSPLF command.

For information on canceling multiple spooled files of a job, refer to the *Additional Considerations* section of the Display Job (DSPJOB) command. For information on canceling *all* spooled files of a job, refer to the Cancel Job (CNLJOB) command.

Restrictions: You must be the owner of the job that created the file being canceled; or have read, add, and delete rights for the output queue containing the file; or have job control rights, and the output queue must have OPRCTL(*YES) specified.



FILE Parameter: Specifies the name of the spooled file that is to be removed from the output queue. The file name is the name of the device file that was used by the program to produce the spooled output file.

JOB Parameter: Specifies the name of the job that produced (or is producing) the spooled file that is to be removed from the output queue.

*: The job that issued this CNLSPLF command is the job that produced the file to be canceled.

qualified-job-name: Enter the qualified name of the job that produced the file to be canceled. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the job's spooled output file that is to be removed from the output queue. (For an expanded description of the SPLNBR parameter, see Appendix A.)

*ONLY: Only one spooled output file in the job has the specified file name; therefore, the number of the spooled file is not necessary. If *ONLY is specified and more than one file on the output queue has the specified name, an error message is displayed to the user who issued this command.

**LAST:* The highest numbered spooled file created for the job that has the specified file name is the file that is being canceled.

spooled-file-number: Enter the number of the spooled file with the specified file name that is being canceled.

Example

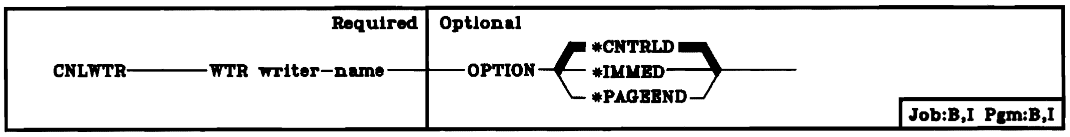
```
CNLSPLF  FILE(WEEKLY) JOB(PAYROLL5.SMITH.000146)
```

This command removes from the output queue the file named WEEKLY whose job number is 000146. Any output files with different names produced by the job PAYROLL5 are not affected by this command. If the job produced more than one file named WEEKLY, no file is canceled because SPLNBR(*ONLY) is assumed.

CNLWTR (Cancel Writer) Command

The Cancel Writer (CNLWTR) command terminates the specified spooling writer and makes its associated output device available to the system. The writer can be canceled immediately or in a controlled manner. If canceled immediately, the writer stops writing the file, and the file is made available again on the output queue. If canceled in a controlled manner, the writer finishes writing the current file (or a copy of a file), or it finishes printing a page of the file, before it is canceled.

Restrictions: The user must have read, add, and delete rights for the output queue associated with the writer; or he must have job control rights (*JOBCTL) in his user profile and the output queue must have OPRCTL(*YES) specified.



WTR Parameter: Specifies the name of the spooling writer to be canceled. The writer's output device is available to the system.

OPTION Parameter: Specifies when the canceled writer should terminate processing.

**CNTRLD:* The spooling writer is to terminate processing in a controlled manner. Output stops at the end of the output file (or copy of a file) currently being written to an output device.

**IMMED:* The writer is to terminate processing immediately. (The file being output remains on the output queue.)

**PAGEEND:* The writer is canceled at the end of a page. This value is valid only if the spooling writer is a printer writer.

Example

```
CNLWTR  WTR(PRINTER)
```

This command stops the writer, PRINTER, at the end of the spooled file whose output is being printed and releases the device to the system.

CPYF (Copy File) Command

The Copy File (CPYF) command has many uses. It can:

- Copy data and source between data base files. Records can be copied *from* physical or logical files; records can be copied *to* physical files, but *not* to logical files.
- Copy data and source files from external devices, such as the MFCU or diskettes, to the data base.
- Copy data and source files from the data base to external devices.
- Copy data and source files from external devices to other external devices.
- Copy data and source from inline data files to the data base or to external devices.

The combinations of device and/or data base files for which copy operations can be performed are shown in the following chart. An X indicates that the corresponding file types are a valid combination for copying a file.

	To File					
From File	**Physical**	**Undefined**[1]	**Printer**	**Diskette**	**Card**	**Tape**
Physical	X	X	X	X	X	X
Logical	X	X	X	X	X	X
Diskette	X		X		X	X
Card	X		X	X	X	X
Tape	X		X	X	X	X
Inline Data[2]	X		X	X	X	X

[1]An undefined file is one that does not exist. It is created if CRTFILE(*YES) is specified on the CPYF command. A physical file is always created, even if the from-file is a logical file.
[2]An inline data file is a data file that is included as part of a batch job when the job is read by a reader program.

Besides copying the records, the CPYF command can also perform the following functions:

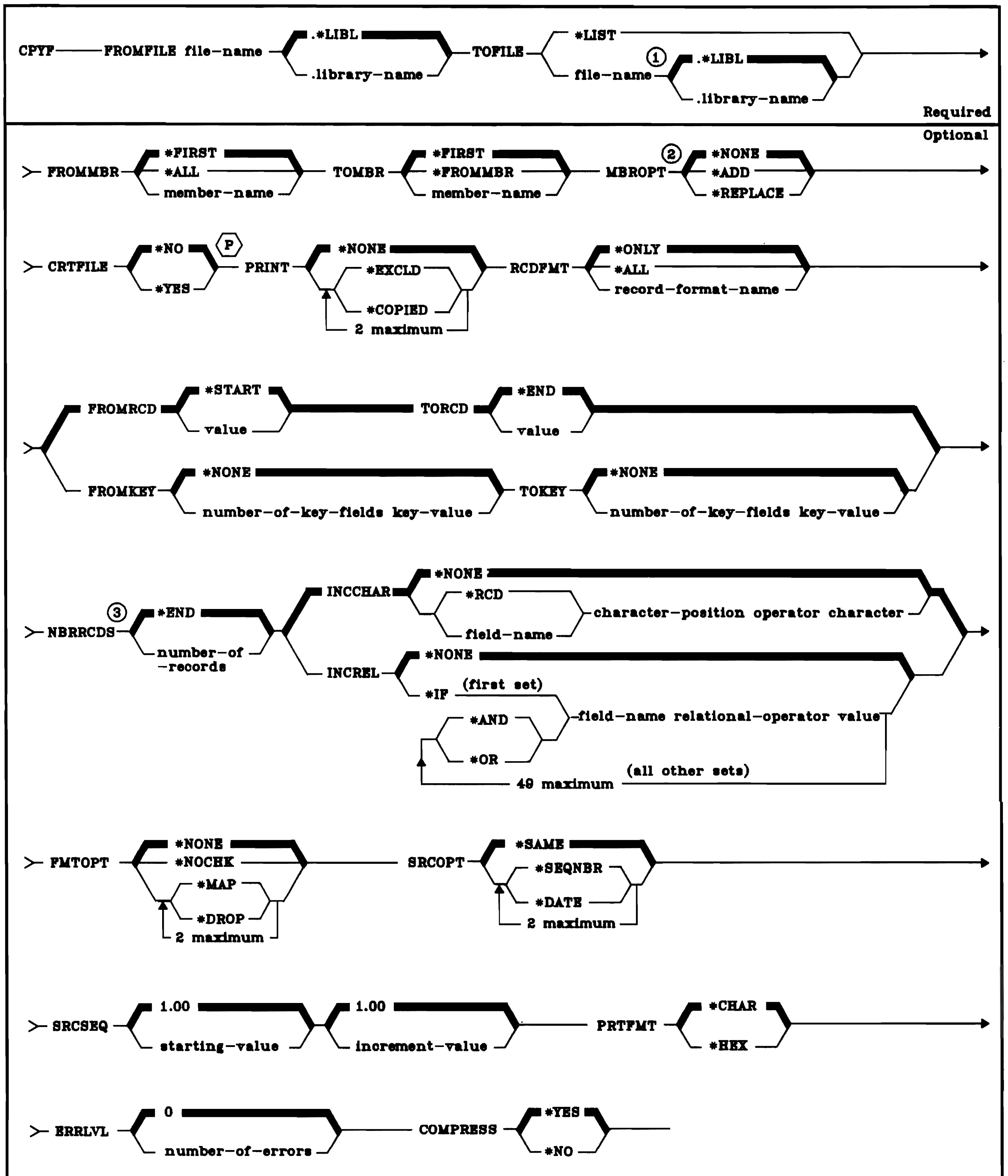
- Copying from and to the first data base file member, a particular file member or all file members (FROMMBR and TOMBR parameters).
- Adding records to an existing file member or replacing the contents of the file member (see MBROPT parameter description).
- Selecting certain records for copying by one of the following methods:
  - Basing the selection on the contents of a character position in the record or in a field in the record (INCCHAR parameter).
  - Basing the selection on the values of one or more fields in the record (INCREL parameter).
  - Specifying the number of records to be copied (NBRRCDS parameter).
  - Specifying records beginning at a relative record number and/or ending at a relative record number (FROMRCD and TORCD parameters).
  - Specifying records beginning with a specific record key value and/or ending with another specific record key value (FROMKEY and TOKEY parameters).
  - Specifying a record format to use when copying a multiformat logical file (RCDFMT parameter).
- Copying records whose from-file and to-file record formats are different (FMTOPT parameter). When formats are different, CPYF can:
  - Map fields whose names are the same in the from-file and to-file record formats (FMTOPT(*MAP)).
  - Drop fields from the from-file record format that do not exist in the to-file record format (FMTOPT(*DROP)).
  - Copy data directly (left to right), disregarding the differences (FMTOPT(*NOCHK)).
- Inserting a sequence number and/or a zero date in the source fields when copying source files (SRCOPT parameter). When renumbering is to be done, the starting sequence number and the increment value can be specified (SRCSEQ parameter).
- Create the to-file as part of the copy operation (CRTFILE(*YES)).
- Terminate the copy after a specified number of recoverable data base errors are encountered (ERRLVL(number-of-errors)).

  Note: When a physical file is being copied, the relative record numbers of the to-file may not correspond to the from-file if records are being added to the to-file (MBROPT(*ADD) parameter), or if compression is used to prevent deleted records from being copied (COMPRESS(*YES) parameter).

- When the CPYF command is used to copy a file to a printer using the TOFILE(*LIST) or PRINT parameters, two formats can be specified to print the records: character format, or both character and hexadecimal format (PRTFMT parameter).

If a program-described printer file is specified for the to-file (rather than the TOFILE(*LIST) value), a listing of records will be produced that does not have headers or record sequence numbers (unlike the TOFILE(*LIST) value, which does have headers and record sequence numbers). If a program-described printer file is specified for the to-file, then a straight listing of the records will be produced. If the program-described printer file specified has the CTLCHAR(*FCFC) attribute, the first character in each record will control the spacing and skipping of the records printed.

Note: Source files can be copied only to source files, and nonsource files can be copied only to nonsource files. Records in a data base file on a device can be processed as source or nonsource, depending on the device file used.



① A library qualifier must be specified if CRTFILE(*YES) is specified and the file does not exist.
② *NONE is the default for copies to device files only. *ADD is the default for copies to a *new* physical file, created when CRTFILE(*YES) is specified. When copying to an *existing* physical file, MBROPT must specify *ADD or *REPLACE.
③ If TORCD or TOKEY is specified, NBRRCDS cannot be specified. If NBRRCDS is specified, neither TORCD nor TOKEY can be specified.

Job:B,I Pgm:B,I

The following chart shows all of the CPYF parameters and indicates for which file types each parameter is valid. (The parameters that can be used with all the data base and device files are: PRINT, NBRRCDS, and INCCHAR.) The parameters are listed down the left side, and the file types (and whether each can be a from-file and/or a to-file) are shown across the top. An X indicates that the associated parameter is valid for the type and use of file under which it occurs. No X is shown when the parameter is either invalid or ignored (does not apply).

	Data Base Files				Device Files									
	Physical		Logical		Diskette		Tape		Card		Printer		Inline Data	
Parameter	From	To	From	To	From	To	From	To	From	To	From	To	From	To
FROMFILE	X		X	Never applicable	X[1]		X		X		Never applicable		X	Never applicable
TOFILE		X		Never applicable		X[1]		X		X	Never applicable	X[2]		Never applicable
FROMMBR	X		X	Never applicable	X[3]		X[3]				Never applicable			Never applicable
TOMBR		X		Never applicable		X[3]		X[3]			Never applicable			Never applicable
MBROPT		X		Never applicable							Never applicable			Never applicable
CRTFILE	X		X	Never applicable							Never applicable			Never applicable
PRINT	X	X	X	Never applicable	X	X	X	X	X	X	Never applicable	X	X	Never applicable
RCDFMT			X	Never applicable							Never applicable			Never applicable
FROMRCD	X[4]		X[5]	Never applicable	X		X		X		Never applicable		X	Never applicable
TORCD	X[4]		X[5]	Never applicable	X		X		X		Never applicable		X	Never applicable
FROMKEY	X[6]		X[6]	Never applicable							Never applicable			Never applicable
TOKEY	X[6]		X[6]	Never applicable							Never applicable			Never applicable
NBRRCDS	X		X	Never applicable	X		X		X		Never applicable		X	Never applicable
INCCHAR	X		X	Never applicable	X		X		X		Never applicable		X	Never applicable
INCREL	X		X	Never applicable							Never applicable			Never applicable
FMTOPT	X[7]	X	X[7]	Never applicable							Never applicable			Never applicable
SRCOPT	X[7]	X	X[7]	Never applicable							Never applicable			Never applicable
SRCSEQ	X[7]	X	X[7]	Never applicable							Never applicable			Never applicable
PRTFMT[8]				Never applicable							Never applicable	X		Never applicable
ERRLVL	X	X	X	Never applicable							Never applicable			Never applicable
COMPRESS	X	X		Never applicable							Never applicable			Never applicable

[1]Valid only if the other file (being copied to, or from) is not another diskette file.
[2]Valid only if *LIST or a program-described printer file is specified. An externally described printer file *cannot* be specified.
[3]Valid only if a data file identifier is not previously provided on a file override.
[4]Valid only if FROMKEY and TOKEY are not specified.
[5]Valid only if the logical file has an arrival sequence access path.
[6]Valid only if FROMRCD and TORCD are not specified.
[7]Valid only if copy is to a physical file.
[8]Valid for TOFILE(*LIST) or all files if PRINT(*EXCLD), PRINT(*COPIED), or PRINT(*EXCLD *COPIED) is specified.

FROMFILE Parameter: Specifies the qualified name of the data base file or device file that contains the data to be copied. (If no library qualifier is given, *LIBL is used to find the file.) The data base file can be a physical or logical file. The device file can be a card, diskette, tape, or inline data file.

TOFILE Parameter: Specifies the physical file or device file into which the data is to be copied. The device file can be a card, diskette, tape, or printer file. However: (1) Diskette files cannot be specified for both the FROMFILE and TOFILE parameters. (2) An externally described printer file cannot be specified.

**LIST*: The data is to be copied to the IBM-supplied printer device file QSYSPRT and the listing formatted according to the PRTFMT parameter attributes. If a formatted listing is not wanted or if CTLCHAR(*FCFC) is specified, then a program-described printer device file name (can be user-defined or QSYSPRT) must be specified instead of *LIST.

Note: If *LIST is specified, an OVRPRTF command can be used to override any of the attributes of the QSYSPRT printer file, except for the file name (TOFILE parameter of the OVRPRTF command), the replace unprintable character attribute (RPLUNPRT parameter), and the control character attribute (CTLCHAR parameter).

qualified-file-name: Enter the qualified name of the physical file or device file that is to receive the copied data. (If no library qualifier is given, *LIBL is used to find the file. However, if CRTFILE(*YES) is specified and the specified physical file cannot be found, the file name *must* be qualified with a library name; then when the file is created, it is stored in the library specified.)

FROMMBR Parameter: Specifies, for data base files only, the name of the member within the file specified by the FROMFILE parameter that is to be copied. It can also be used to specify the data file identifier when data is being copied from diskette or tape and a data file identifier is not provided on an override command. This parameter is not valid for any other device files.

<u>*FIRST</u>: The first member in the data base file specified by the FROMFILE parameter is the member to be copied.

**ALL:* All members of the *data base* file specified by the FROMFILE parameter are copied to corresponding members of the data base or diskette file specified by the TOFILE parameter. (In this case, TOMBR(*FROMMBR) must also be specified.) If a *diskette device* file is to be copied to a data base file, *ALL means that all diskette data files in the device file are to be copied to a single data base member. (In this case, TOMBR must specify *FIRST or a member name.) If you are copying from the data base to a diskette specifying FROMMBR(*ALL), the data base member names are used for the data set identifiers. Member names are truncated on the left if they are longer than 8 characters. If this truncation results in duplicate names, the copy operation terminates with a diskette error.

member-name: Enter the name of the member within the file specified by the FROMFILE parameter that is to be copied. (A data file identifier can be specified here if the copy is from a diskette or tape and the identifier is not provided on an override command.) If the tape data file identifier is longer than 10 characters or contains special characters, it must be specified on the CRTTAPF, CHGTAPF, or OVRTAPF command before CPYF is executed.

TOMBR Parameter: Specifies the name of the member within the physical file, specified by the TOFILE parameter, that is to receive the copied data. If TOFILE is specified as a diskette or tape device file, TOMBR specifies the data file identifier of the diskette or tape to which the data is to be copied. This parameter is valid only for physical files and diskette or tape device files.

*FIRST: The first member in the physical file specified by the TOFILE parameter is to receive the copied data.

**FROMMBR:* Corresponding from- and to-member names are to be used. Corresponding member names means that each from-member name is used as a to-member name when all members are copied. If a corresponding to-member name already exists, the MBROPT parameter is used to determine whether records are added or replaced in the member. If TOMBR(*FROMMBR) is specified and the to-member does not exist, it is added to the file.

If TOMBR(*FROMMBR) is specified when a *data base* file is to be copied, FROMMBR(*ALL) *must* also be specified. However, TOMBR(*FROMMBR) cannot be specified if a *single* member is to be copied to a physical file *and* either the name of the from-member or *FIRST is specified.

member-name: Enter the name of the physical file member or the data file identifier of the diskette or tape data file that is to receive the copied data. If the receiving data base file is being created by this command (CRTFILE(*YES) is specified), the name specified here is the name of the member added to the created file. If the tape data file identifier is longer than 10 characters or contains special characters, you must specify it on the CRTTAPF, CHGTAPF, or OVRTAPF command before executing the CPYF command. If you are copying from the data base to a diskette specifying FROMMBR(*ALL), the data base member names are used for the data set identifiers. Member names are truncated on the left if they are longer than 8 characters. If this truncation results in duplicate names, the copy operation terminates with a diskette error.

MBROPT Parameter: Specifies, for copies to physical files only, that copied data either is to be added or is to replace the existing data in a physical file member. If the copy is to a physical file, this parameter must specify either *ADD or *REPLACE.

*NONE: No data records are to be added or replaced in a member. This value is valid only for copies to device files, not to physical files.

**ADD:* Data records are to be added to existing records in a member. The new records are physically added to the end of the member. If the file being copied to is a keyed file, the added records will be in the correct keyed sequence when the file is processed with its keyed sequence access path. New records are automatically added to the new file if CRTFILE(*YES) is specified (the MBROPT parameter is ignored) and a new file is created.

**REPLACE:* Data records are to replace existing records in a member. Existing records are cleared from the member before the new data records are copied into the member.

CRTFILE Parameter: Specifies, when the CPYF command is used to copy from a physical file or logical file, whether a physical file that is to receive the data is to be created if the to-file does not exist. If the to-file does not exist, CRTFILE(*YES) is required and the name of the file to be created must be qualified with the name of an existing library for which the user has the necessary authority.

A member is also added to the created file. Its name is that specified in the TOMBR parameter of the CPYF command or in the MBR parameter of the OVRDBF command; otherwise, its name (by default) is the same as that of the from-file specified in the FROMFILE parameter of the CPYF command.

*NO: A to-file is not to be created.

**YES:* If the to-file does not exist, it must be created. If the file is created, the MBROPT parameter is ignored and records are automatically added to the new file. Note that you must have operational rights for the CRTPF command (which is implicitly invoked to create the file), and you must have operational and add rights for the library that is to contain the created file.

If the CPYF command creates a new physical file when the from-file is a physical file, the new file is given the same attributes as the from-file. If, however, the from-file is a logical file that has multiple record formats, the RCDFMT parameter must specify a record format name. Then, when the physical file is created, it has the attributes of the logical file and the specified record format; in addition, the following physical file attributes are assigned by the system: SIZE(*NOMAX), ALLOCATE(*NO), and CONTIG(*NO).

PRINT Parameter: Specifies whether copied records and/or excluded records are to be printed. The values *EXCLD and *COPIED can both be specified for PRINT. If they are, all of the excluded records are in one listing and all of the copied records are in another listing.

If multiple members are being copied, there will be separate listings for each member of the excluded records and of the copied records. The records will be printed using the IBM-supplied printer device file QSYSPRT and the listing formatted according to the PRTFMT parameter attributes.

NONE: No records are to be printed.

**EXCLD:* Only the records excluded from the copy operation by the INCCHAR and INCREL parameters are to be printed.

**COPIED:* Only the copied records are to be printed.

Note: If you specify PRINT(*EXCLD), PRINT(*COPIED), or PRINT(*EXCLD *COPIED), an OVRPRTF command can be used to override any of the attributes of the QSYSPRT printer device file, except for the file name (TOFILE parameter), the replace unprintable character attribute (RPLUNPRT parameter), and the control character attribute (CTLCHAR parameter).

RCDFMT Parameter: Specifies, for logical file copies only, the name of the record format that is copied when the from-file is a logical file with multiple formats.

ONLY: Only one record format exists in the logical file being copied. That is the format in which the data is to be copied. If the file contains multiple members and they all use the same format, all the members are copied.

**ALL:* All the record formats in the logical file specified by the FROMFILE parameter are to be used when the data in the file is copied to a device file. RCDFMT(*ALL) is valid only if a logical file is copied to a device file. In this case, when the record formats have different lengths, the shorter length records are padded with blanks when they are copied.

record-format-name: Enter the name of the record format to be copied when the from logical file has more than one format. If the file has multiple members, all members that use the specified format are copied.

FROMRCD Parameter: Specifies the relative record number of the first record to be copied from the specified file. The value (n) specified indicates that the *n*th record from the beginning of the file is the first record to be copied. This parameter is not valid if a value other than *NONE is specified on the FROMKEY or TOKEY parameter.

*START: The copy is to begin with the first record in the file, as determined by the access path, which (for *START only) can be either in keyed sequence or in arrival sequence.

value: Enter a relative record number, no more than nine digits in length, that identifies the first record to be copied from the file.

TORCD Parameter: Specifies the relative record number of the last record to be copied from the specified file. This parameter is not valid if the values for the TOKEY, the FROMKEY, or the NBRRCDS parameter are anything other than *NONE, and *END, respectively.

*END: Records are to be copied until the end-of-file condition is indicated from the file.

value: Enter a relative record number, no more than nine digits in length, that identifies the last record to be copied from the file. The value must be equal to or greater than the FROMRCD value.

Note: Keyed files are copied, by default, in the order of their keyed sequence access path. However, if either FROMRCD or TORCD is specified, the file will be copied in the order of its arrival sequence access path.

FROMKEY Parameter: Specifies, when files are copied by key values, the key field value of the first record to be copied. This parameter is valid only for keyed data base files if FROMRCD and TORCD are not specified. For example, FROMKEY(1 JONES) requests CPYF to copy starting with the record whose first key field has a value of JONES, while FROMKEY(2 JONES7) requests CPYF to copy starting with the record whose first key field has a value of JONES and whose second key field has a value of character 7.

*NONE: The first record to be copied is not selected by key.

number-of-key-fields key-value: Enter the two values that identify the first keyed record to be included in the copied file. The first value specifies the number of key fields to be used in searching the record keys (beginning with the first key field), and the second value specifies the actual key value of the first record to be copied.

All positions in the key value should be specified, or a generic search producing undesired results may occur. If a key value is specified whose length is shorter than the defined key field length, the value is padded on the right with zeros. For example, if a key field is defined as a five-position decimal field with no decimal positions and the key value is to be 8, FROMKEY(1 00008) should be specified; not FROMKEY(1 8), which causes a search for a key equal to 80 000.

If the key value is a character string that has blanks or special characters, it must be enclosed in apostrophes. If the key value contains one or more key fields whose data types were defined in DDS as either packed decimal or binary, the key must be coded as a hexadecimal value. For example, if two key fields were defined as a character field of three positions and a binary field of two positions, and if the first key field contains ABC and the second contains 15, the from-key value must be coded as FROMKEY(2 X'C1C2C3000F').

TOKEY Parameter: Specifies, when files are copied by key values, the key value of the last record to be included in the copied file. This parameter is valid only for keyed data base files and if no values are specified for the TORCD, FROMRCD, or NBRRCDS parameters.

*NONE: The last record to be copied is not selected by key.

number-of-key-fields key-value: Enter the two values that identify the last keyed record to be included in the copied file. The first value specifies the number of key fields to be used in searching the record keys (beginning with the first key field), and the second value specifies the actual key value of the last record to be copied. If the key value is a character string that has blanks or special characters, it must be enclosed in apostrophes.

Note: If the key value specified in the TOKEY parameter is less than the value specified in the FROMKEY parameter, a descending keyed sequence is assumed.

Unpredictable results may occur if a starting position is specified for the POSITION parameter in the Override with Data Base File (OVRDBF) command, or if the from-file is opened with SHARE(*YES) specified, or if the SEQONLY parameter is specified on an override.

NBRRCDS Parameter: Specifies the number of records to copy beginning with the record specified by the FROMRCD or the FROMKEY parameter. The TORCD and TOKEY parameters cannot be used if a numeric value is specified for NBRRCDS.

*END: Records are to be copied until the end-of-file condition is indicated for the file, unless either the TOKEY or the TORCD parameter has been specified.

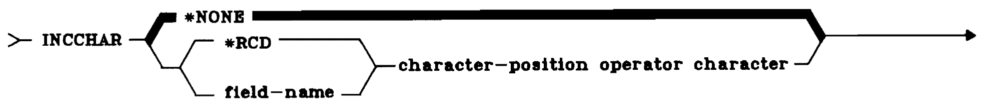
number-of-records: Enter a value, no more than nine digits in length, that specifies the number of records to be copied from the file.

INCCHAR Parameter: Specifies that records are to be included in the copy based on the result of a comparison with a user-supplied character value. The specified character can be compared with the character in a specified character position of each record, or with the character in the specified position in a field of each record. If INCCHAR is specified with RCDFMT(*ALL), the include character is used for selecting records from all the formats.

Note that the INCCHAR and INCREL parameters are mutually exclusive; if INCCHAR is specified, INCREL cannot be specified.

*NONE: No comparison with a specific character is to be used to determine which records are to be included in the copied file.

Comparison Values: To specify the comparison that is used to determine which records are to be copied, four values must be entered. Either *RCD or the name of a field must be entered, followed by the three comparison values: character position, operator, and the actual character. All records that meet the relationship specified by the four values are copied into the file specified by TOFILE.



**RCD:* The character that is in the specified position of each *record* is to be compared with the specified character (the fourth value).

field-name: Enter the name of the field in the record format that is to have one position in the field compared with the specified character.

character-position: Enter the position within the field or record that is to be used in the comparison.

operator: Enter the relationship of the character position to the specified character. Relational operators that can be used are:

*EQ	Equal
*GT	Greater than
*LT	Less than
*NE	Not equal
*GE	Greater than or equal to
*NL	Not less than
*LE	Less than or equal to
*NG	Not greater than

character: Enter the character to be used for the comparison with the specified field or record position. If the character is a special character, it must be enclosed in apostrophes. A hexadecimal value can be used to specify the character, if necessary; it must be specified as X'*nn*' where *nn* is the two-digit hex value that represents the character. (For example, X'5A' represents the exclamation point, !.)

INCREL Parameter: Specifies that records are to be included in the copy based on the specified contents of the records meeting the specified value relationships. As many as 50 value relationships can be specified to determine whether each record is to be copied. INCREL is not valid if a device file is specified in the FROMFILE parameter.

Note that the INCREL and INCCHAR parameters are mutually exclusive; if INCREL is specified, INCCHAR cannot be specified.

*NONE: No relational comparison between any record or field values and a specified relationship is to be made to determine which records are to be copied.

Relationship Values: To specify the conditions under which records are to be copied, a set of values is specified for each condition. Each set must contain exactly four values:

1. One of the logical operators *IF, *AND, or *OR

2. The name of the field to be compared

3. One of the relational operators (from the list that follows)

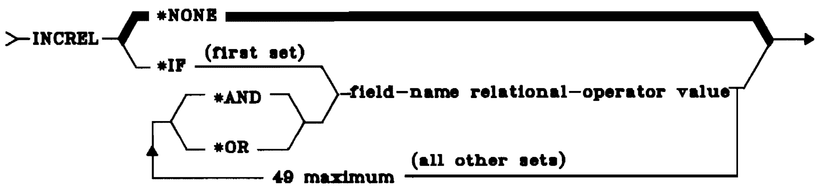
4. The comparison value

Values 2 and 4 are compared to see if they have the relationship specified by value 3.

The value *IF *must* be specified as the first value in the first set of comparison values, whether there is only one set or several sets. If more than one set of comparison values are specified, either *AND or *OR must be specified as the first value in each set after the first one.

In the following discussion, an *IF group* refers to an IF set optionally followed by one or more AND sets. An *OR group* refers to an OR set optionally followed by one or more AND sets. The objective is to perform all the comparisons specified in each group until a complete group (which can be a single IF set or OR set having no AND sets following it) yields all true results. If one group has true results, the tested record is included in the copied file.

The first set of comparison values (*IF field-name operator value) and any AND sets logically connected with the IF set are evaluated first. If the results in all of the sets in the IF group are all true, the testing ends and the record is copied. If any of the results in the IF group are false and an OR group follows, another comparison begins. The OR set and any AND sets that follow it are evaluated (up to the next OR set). If the results in the OR group are all true, the record is included. If any result is false and another OR group follows, the process continues until either an OR group is all true or until there are no more OR groups. If the results are not all true for any IF or OR group, the record is not included in the copied file.



**IF:* Identifies the initial relationship which must be satisfied by the record before that record can be copied.

**AND:* The relational groups on both sides of the *AND value must all be satisfied by the record before that record can be copied.

**OR:* If either relational group on either side of the value *OR is satisfied, the record is copied.

field-name: Enter the name of the field to be compared.

relational-operator: Enter the relationship of the specified field contents and the specified values. Operators that can be used are:

*EQ	Equal
*GT	Greater than
*LT	Less than
*NE	Not equal
*GE	Greater than or equal to
*NL	Not less than
*LE	Less than or equal to
*NG	Not greater than

value: Enter the value to be compared with the contents of the specified field. The value cannot be another field name. (If the value is a character string containing blanks or special characters, it may need to be enclosed in apostrophes; refer to Chapter 2 for more information.) If a CL variable is specified for the value, it must be a character variable. A coded example of the INCREL parameter is:

```
((*IF FIELD1 *GT 100) (*AND FIELD2 *EQ DAILY) (*OR FIELD5 *GE &FLD5TEST))
```

Each record whose fields meet one of the following conditions would be included in the copied file:

- Field 1 is greater than 100, and field 2 contains DAILY.
- Field 5 has a value that is greater than or equal to the value contained in the CL character variable &FLD5TEST.

FMTOPT Parameter: Specifies, when a data base file is being copied to another data base (that is, physical) file, what actions are to be taken as a result of the record format checking done by the system. A value other than *NONE *must* be specified for FMTOPT if the record formats of the from and to data base files are different. (For any other type of copy, such as data base to device file copy, records are to be copied without any checking. In these cases, if the record lengths of the from- and to-files are different, they are truncated or padded with blanks or zeros, depending upon the characteristics of the to-file.)

Note: See the topic on different data base record formats in the *CPF Programmer's Guide* for additional information.

<u>*NONE:</u> No field format checking is to be done during this copy operation. This value is valid only if this is not a data-base-file to data-base-file copy or if both data base files have identical record formats.

**NOCHK:* If the record formats of the data base files are different, the copy operation is to continue, despite the differences. Data is copied directly (left to right) from one file to the other. If the data is copied into a file that has a longer record format, the copied record is padded with zeros or blanks. If the to-file has a shorter record format, the copied records are truncated on the right. Messages will be sent indicating the differences in the record formats.

**MAP:* Individual fields with the same names in both formats are copied even if their field attributes are different, except in some cases. If they are different, the data is converted into the format of the file into which it is copied. Fields in the to-file record format that are not in the from-file record format are padded with blanks or set to zero. *MAP must be specified if both formats have identical field names but different field attributes. If *MAP is specified, *DROP can also be specified.

Mapping is *not* allowed in the following situations:

- From a character field to any type of numeric field, and vice versa
- From a binary field to a binary field that has a different number of decimal positions
- From a zoned or packed numeric field to a binary numeric field, and vice versa, if the binary field has a number of decimal positions greater than zero

**DROP:* Any fields that are in the from-file record format are to be dropped if the same fields are not in the to-file record format. All other fields in both record formats must have the same names, attributes, and relative order within the record; otherwise, *MAP must also be specified. *DROP must be specified if all of the fields in the from-file are not in the to-file. If *DROP is specified, *MAP can also be specified.

SRCOPT Parameter: Specifies, for copying between data base source files only, whether to insert new sequence numbers in the sequence number field, whether to place zeros in the date field, or whether to update both the sequence number and the date fields.

When copying from a device source file to a data base source file, the system automatically adds sequence number and date fields to the beginning of the records, so the SRCOPT parameter does not apply. Also, when copying from a data base source file to a device source file, the system removes the sequence number and date fields.

<u>*SAME:</u> When copying a data base source file to another data base source file, the sequence number and date fields are not to be changed.

**SEQNBR:* The copied data base source file records are to have new sequence numbers inserted. If *SEQNBR is specified, *DATE can also be specified. The SRCSEQ parameter is used to specify the start and increment values.

**DATE:* The copied data base source file records are to have a null (all zeros) date inserted. If *DATE is specified, *SEQNBR can also be specified.

SRCSEQ Parameter: Specifies, only when SRCOPT(*SEQNBR) is also specified, what sequence number is to be given to the first record copied to the to-file and what increment value is to be used to renumber all other records copied. This parameter allows the copied file to have as many as 999 999 records with unique sequence numbers. If a copied source file is to be renumbered but SRCSEQ is not specified, SRCSEQ(1.00 1.00) is assumed; the copied records are renumbered sequentially beginning with 1.00, and the whole number increment of 1 is used.

1.00: The first source record copied to the to-file is to have a sequence number of 0001.00.

starting-value: Enter a value in the range of 0000.01 through 9999.99 that is to be the sequence number of the first source record copied to the to-file. A whole number of no more than four digits and/or a fraction of no more than two digits can be specified. If the starting value contains a fraction, a decimal point must be used. Examples are .01 and 3250.4. (If a value has a fraction of .00, such as 5000.00, it can be coded without the fraction; either 5000 or 5000.00 is valid.)

1.00: The copied source records are to be renumbered in the to-file with whole number increments of 1. (1.00, 2.00, 3.00...).

increment-value: Enter a value in the range of 0000.01 through 9999.99 that is to be used as the increment value for renumbering all source records copied after the first one. A whole number of no more than four digits and/or a fraction of no more than two digits can be specified. If the increment value contains a fraction, a decimal point must be used. For example, if SRCSEQ(5000 10) is specified, the first record copied to the file is numbered 5000.00, the second is 5010.00, the third is 5020.00, and so on. If SRCSEQ(*N .25) is specified, the copied records are numbered 1.00, 1.25, 1.50, 1.75, 2.00, and so on.

If the maximum sequence number of 9999.99 is reached when you are copying between data base source files, the remaining records copied will be assigned the sequence number 9999.99 also. If, when you are copying from a device source file to a data base source file, the maximum sequence number is reached, the sequencing will wrap back to 1 and increment from there for the remaining records. The Reorganize Physical File Member (RGZPFM) command can be used to reassign unique sequence numbers to the records.

PRTFMT Parameter: Specifies whether records are to be printed in character format, or in both character and hexadecimal format. PRTFMT is valid only when the to-file is specified as *LIST, or when the PRINT parameter specifies either *EXCLD or *COPIED (or both of them).

*CHAR: Records are to be printed in character format.

**HEX:* Records are to be printed in character and hexadecimal format.

ERRLVL Parameter: Specifies, only for recoverable errors detected during copies to or from data base files, the maximum number of recoverable errors to be allowed per file member if mapping errors or duplicate keys are encountered during the copy operation, or if the from-file contains damage. If the maximum is exceeded, the copy operation is terminated and a message is sent to the user. When multiple file members are being copied (either to or from) in the same operation, the error count restarts at zero at the beginning of each one. But if the count exceeds the error level for a file member, the copy operation terminates and any remaining members are not copied.

For read operations, the recoverable errors are those that occur when data is converted (mapped) and those caused by a damaged area on the disk (in auxiliary storage). For write operations, the recoverable errors are those that occur when data is converted and those that occur when duplicate keys are encountered. Any record that causes an error is not copied and is not printed (regardless of the values specified for the PRINT parameter). A diagnostic message that identifies the record causing the error is sent to the user who issued the command.

0: If *any* recoverable error occurs, the copy operation is terminated with the file member in which the error occurs.

number-of-errors: Enter a value that specifies the maximum number of recoverable errors that can occur within each file member being copied, after which another error causes the copy operation to be terminated.

COMPRESS Parameter: Specifies, only when a physical file that is accessed in arrival sequence is being copied to another physical file, whether the to-file is to be compressed. Compression occurs when deleted records in the from-file are not copied to the to-file.

If a keyed file is copied, the keyed sequence access path does not contain deleted records. Therefore, the COMPRESS parameter does not apply. However, if the FROMRCD or TORCD parameter specifies a numeric value, the keyed physical file (or based-on physical file) is copied in *arrival* sequence, and the COMPRESS parameter determines whether deleted records are to be copied.

*YES: The to-file is to be compressed; deleted records that may exist in the from-file are not to be copied to the to-file. Only undeleted records are to be copied, and are to be renumbered consecutively in the to-file. That is, the relative record numbers of all undeleted records that occur *after* the first deleted record in the from-file will be different in the to-file. (No physical record data, such as source sequence numbers, are changed by the copy operation as a result of specifying COMPRESS(*YES).)

**NO:* Both the deleted and undeleted records are to be copied to the to-file, and the relative record numbers are not changed. If *NO is specified, the following parameters *cannot* be specified: PRINT, INCCHAR, INCREL, SRCOPT, ERRLVL, FMTOPT(*MAP) if mapping is required, and FMTOPT(*DROP) if dropping is required.

Examples

The following examples of the CPYF command show the type(s) of files that can be copied and the function provided by various parameters.

Example 1: Physical Data Base File to Physical Data Base File

```
CPYF  FROMFILE(PAYROLL.PERSONNEL) +
      TOFILE(PAYROLL.TESTPAY)  MBROPT(*ADD) +
      CRTFILE(*YES)  ERRLVL(2)
```

This command copies all of the records in the physical file named PAYROLL in the PERSONNEL library to the file PAYROLL in the TESTPAY library. If the from-file contains more than one member, only the first member is copied. If PAYROLL.TESTPAY does not exist, it is created before the data records are copied. A member with the same name as the from-file is created in PAYROLL.TESTPAY to correspond with the member being copied from PAYROLL.PERSONNEL.

New records are automatically added to the end of the file, because MBROPT(*ADD) is specified. The to-file PAYROLL.TESTPAY will have the same record format and access path as the file PAYROLL.PERSONNEL. If the to-file existed before the copy and contained records, it now contains more records than the from-file does. If more than two recoverable errors occur during the copy, the operation is terminated.

If FROMMBR(*ALL) and TOMBR(*FROMMBR) had also been specified, all of the members in the from-file would be copied to corresponding members (having the same names) in the to-file. For each from-member that has no corresponding to-member, a to-member is created and all the records in the from-member are copied to it. For each to-member that already exists, only new records are added to the member (no updates are made to existing records on any copy operation). If the to-file contained members for which there were no corresponding members in the from-file, the to-file contains more members than the from-file after the copy operation. If more than two recoverable errors occur within any member being copied, the copy operation terminates at that point, and any remaining members are not copied.

Example 2: Physical File to Physical File

```
CPYF  FROMFILE(EMP1.PERSONNEL) TOFILE(VACLEFT.PERSONNEL) +
      FROMMBR(VAC) MBROPT(*REPLACE) +
      FROMKEY(1 438872) TOKEY(1 810199) +
      INCREL((*IF VAC *GT 005.0)) FMTOPT(*MAP *DROP)
```

In this example, the to physical file, VACLEFT, is defined, but its format is different from the physical file named EMP1, which is to be copied. Both files are in the PERSONNEL library. The from-file EMP1, which contains employee records, is keyed on employee number, and records are to be selected by key from employee numbers 438872 through 810199. Only records with more than five days of vacation (VAC) are to be mapped to the to-file. Records are to be selected from member VAC and are to replace the first member of file VACLEFT.

Example 3: Physical Source File to Physical Source File

```
CPYF  FROMFILE(SYSSRC80.RPGLIB) TOFILE(TESTRPG.RPGLIB) +
      FROMMBR(A1) TOMBR(A1) +
      MBROPT(*REPLACE) SRCOPT(*SEQNBR *DATE)
```

This command updates the sequence number field, and inserts zeros in the date field in all the records copied to member A1 of the source file TESTRPG in the RPGLIB library. These records are read from member A1 in the SYSSRC80 file and replace all records in member A1 of the file TESTRPG in the library RPGLIB.

The sequence number starts at 0001.00, is incremented by 1.00, and is placed in the 6-byte sequence field of each record in member A1 of the TESTRPG file. The null date (000000) is placed in the date field of each record copied.

Example 4: Logical File to Physical File

```
CPYF  FROMFILE(SALES.DEPTS) TOFILE(YTDSALES.DEPTS) +
      FROMMBR(TOTSALES) RCDFMT(AA) +
      NBRRCDS(5) MBROPT(*REPLACE)
```

This command copies five records from member TOTSALES of logical file SALES in library DEPTS to the first member in the physical file YTDSALES in library DEPTS. Only records from the logical file SALES in library DEPTS that use the record format AA are copied, and they are copied to YTDSALES in the same format. After the copy, the first member in YTDSALES contains only five nondeleted records because all the records in that member are first deleted and only the data in the first five records (in keyed sequence) in the TOTSALES member are copied to it.

Example 5: Device File to Physical File

```
CPYF FROMFILE(MFCU1) TOFILE(PAYROLL.FINANCE) +
     TOMBR(MBR1) MBROPT(*ADD)
```

This command copies the records for new employees from the card reader to the PAYROLL file in the FINANCE library. The records are to be added to member MBR1. The device file name for the card reader is MFCU1.

Input from MFCU1

Name	Emp#	Position
Jane P	15678	Clerk
Tom J	23451	Clerk
Bob H	85712	Sec
Joe P	71567	Cashier

Existing PAYROLL.FINANCE File, MBR1

Name	Emp#	Position
Mary P	13467	Sec
Tommy J	21457	Sr Clerk
Jake T	22444	Jr Clerk
Richard S	26517	Clerk
Bob K	38751	Sp Clerk
Jeannie H	38753	Clerk
Kathy H	71051	Clerk
Susan H	72342	Sr Sec

Upon completion of the processing for the preceding command, the PAYROLL.FINANCE file, which is keyed on EMP#, contains:

Name	Emp#	Position	
Mary P	13467	Sec	
Tommy J	21457	Sr Clerk	
Jake T	22444	Jr Clerk	
Richard S	26517	Clerk	
Bob K	38751	Sp Clerk	
Jeannie H	38753	Clerk	
Kathy H	71051	Clerk	
Susan H	72342	Sr Sec	
Jane P	15678	Clerk	Records added
Tom J	23451	Clerk	Records added
Bob H	85712	Sec	Records added
Joe P	71567	Cashier	Records added

Example 6: Physical File to System Printer

```
CPYF FROMFILE(TEMPFILE) TOFILE(*LIST) FROMMBR(EMP1) +
     FROMKEY(1 448762) NBRRCDS(2) PRTFMT(*HEX)
```

This command copies two records from member EMP1 of the file named TEMPFILE. The records are employee records; one key field, the employee number, is used to select the records. The first employee number is 448762. The records are to be copied to the system-supplied printer device file in both character and hexadecimal format. The system-supplied printer device file is indicated by coding TOFILE(*LIST).

Example 7: Physical File to Device File

```
CPYF  FROMFILE(PAYROLL.PERSONNEL) +
      TOFILE(DISK1) FROMMBR(VAC1) +
      INCREL((*IF VAC *GT 005.0) (*AND HOL *EQ 0))
```

This command copies all employee records containing more than five days' vacation (none of those five being holidays) from the PAYROLL file in the PERSONNEL library to a diskette. The member to be copied is VAC1. The vacation and holiday fields (VAC and HOL) each contain a packed decimal number whose attributes are DEC(4 1), and whose format is ddd.d, where d = days. The diskette device file name is DISK1, and the diskette label to which the records are copied has been specified by an Override with Diskette File (OVRDKTF) command.

Example 8: Physical File to Device File

```
CPYF  FROMFILE(PAYROLL.PERSONNEL) TOFILE(DISK1) +
      FROMMBR(*ALL) TOMBR(*FROMMBR)
```

This command copies all members of file PAYROLL in the PERSONNEL library to a diskette (device file DISK1).

The physical file member names are used for the label names on the diskette. If the member name is longer than 8 characters, it is truncated on the left.

Assume the from-member names are varying in length. An example of from-member names and corresponding diskette data set names might be:

From-Member (10 bytes)	**Diskette Data Set Name (8 bytes)**
PYROLLREC1	ROLLREC1
PYROLLREC2	ROLLREC2
PAYROLL1	PAYROLL1
PAY1	PAY1

Example 9: Device File to Device File

```
CPYF  FROMFILE(CARDIN) TOFILE(TAPEOUT) TOMBR(TAPLABEL) +
      INCCHAR(*RCD 80 *EQ X) PRINT(*EXCLD)
```

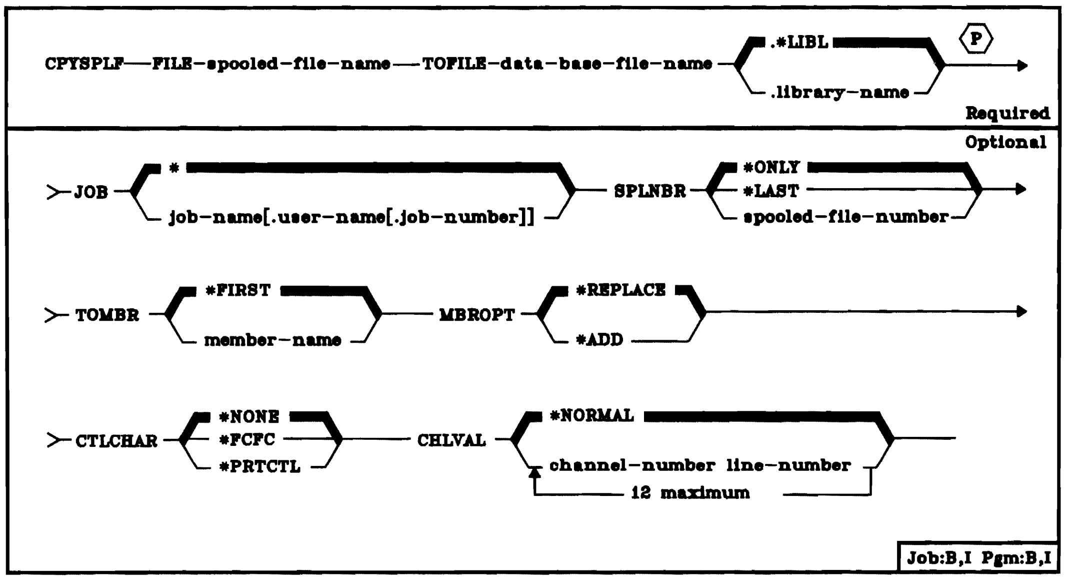
This command copies all card records with an X in column 80 to the tape device. The device file name for the card reader is CARDIN. The device file name for the tape device is TAPEOUT. All cards that are not copied are to be printed. The TOMBR parameter has specified the tape label to which the records will be copied.

CPYSPLF (Copy Spooled File) Command

The Copy Spooled File (CPYSPLF) command copies the data records in the specified spooled output file to a user-defined physical data base file. This conversion allows the use of spooled files in applications using microfiche, data communications, or data processing.

Restrictions: Before you execute this command, one of the following must be true:

- You created the spooled file.
- You have read rights for the output queue containing the spooled file.
- The output queue has DSPDTA(*YES) specified as its display data attribute.



FILE Parameter: Specifies the name of the spooled output file which is to be copied to a data base file. The file name is the name of the device file that was used by the program to produce the spooled output file.

TOFILE Parameter: Specifies a user-defined physical data base file to which the spooled records are to be copied. If this file does not exist at the time of the copy, the copy will fail.

data-base-file-name: Specifies the qualified file name of the physical file. (If no library qualifier is given, *LIBL is used to find the file.)

JOB Parameter: Specifies the name of the job that created the spooled output file whose data records are to be copied.

<u>*:</u> The job that issued this CPYSPLF command is the job that created the spooled file.

qualified-job-name: The qualified name of the job that created the spooled file. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file, from the job whose data records are to be copied. (For an expanded description of the SPLNBR parameter, see Appendix A.)

<u>*ONLY:</u> Only one spooled output file from the job has the specified file name; therefore, the number of the spooled file is not necessary. If *ONLY is specified and more than one spooled output file has the specified name, an error message will be issued.

**LAST:* The highest-numbered spooled output file with the specified file name will be copied.

spooled-file-number: The number of the spooled file having the specified file name whose data records are to be copied.

TOMBR Parameter: Specifies the member in the physical file (specified by the TOFILE parameter) to which the spooled records are to be copied.

<u>*FIRST:</u> The first member of the physical file (specified by the TOFILE parameter) will be used.

member-name: Enter the member name of the physical file. If this member does not exist, a member will be created and the copy will continue.

MBROPT Parameter: Specifies whether copied data be added to or replace data that already exists in the receiving data base file member.

<u>*REPLACE:</u> The member is to be cleared before copied records are added.

**ADD:* The newly copied records are to be added to the existing records in the member.

CTLCHAR Parameter: Specifies print control characters (if any) to replace the spooled file's internal print control characters. Any invalid internal print control characters that are encountered will be ignored and resultant formatting may be unpredictable.

*NONE: No print control characters will be generated. (This option is required for diskette and spooled card files.)

**FCFC:* Specifies that the first character of every record will contain one of the ANSI forms control codes listed below. This option may be useful for microfiche production.

ANSI First Character Forms Control Codes

Code	Action Before Printing a Line
' '	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to next channel 1
2	Skip to next channel 2
3	Skip to next channel 3
4	Skip to next channel 4
5	Skip to next channel 5
6	Skip to next channel 6
7	Skip to next channel 7
8	Skip to next channel 8
9	Skip to next channel 9
A	Skip to next channel 10
B	Skip to next channel 11
C	Skip to next channel 12

**PRTCTL:* Specifies that the first four characters of every record will contain skip- and space-before values useful in HLL (high-level language) programs. This code can be viewed as 'SSSL', where 'SSS' is the skip-before line value and 'L' is the space-before value. 'SSS' can be from 001 to 255 to cause a skip to the specified line (1 to 255); once there, 'L' can be used to specify a spacing of from 0, 1, 2, or 3 lines before printing the record. When one part of the code is used (SSS or L), the other part will be blank. Sample control codes and their meanings are as follows:

Code	Action Before Printing a Line
'001 '	Skip to line 1
'010 '	Skip to line 10
'099 '	Skip to line 99
' 1'	Space one line
' 0'	Do not space (or skip)

CHLVAL Parameter: Specifies a list of channel numbers with their assigned line numbers. Specify this parameter only if CTLCHAR(*FCFC) has been specified. If the spooled file to be printed has data on a line that precedes a line number assigned to a channel, the copy will terminate.

*NORMAL: Indicates channel 1 is the only assigned channel number. The assigned line number for channel 1 will be line 1.

channel-number: Specifies which ANSI FCFC channels are to be used to generate first-character forms control codes. The only valid values for this parameter are 1 through 12. Each channel number may be specified only once per CPYSPLF command.

line-number: The line number assigned for the channel number in the same list. The range of valid line numbers is 1 through 255. Each line number may be specified only once per CPYSPLF command.

Notes:

1. The order in which the channels are specified on the command is not important. For example, the following would be identical:

   ```
   CHLVAL((2 1)(6 15)(8 40))
   CHLVAL((6 15)(2 1)(8 40))
   ```

2. Channel numbers and line numbers do not have to be specified in ascending order.

Examples

```
CPYSPLF FILE(QPRINT) JOB(PAYROLL01) SPLNBR(4) TOFILE(MYFILE) +
      TOMBR(MYMBR) CTLCHAR(*PRTCTL)
```

In this example, the file QPRINT (which is the fourth file produced by the job PAYROLL01) will be copied to the member MYMBR of the physical file MYFILE (which resides in a library found by searching the library list). The newly copied data will replace all old data in the member, because any old records will have been cleared. The 4-byte print control code will be generated.

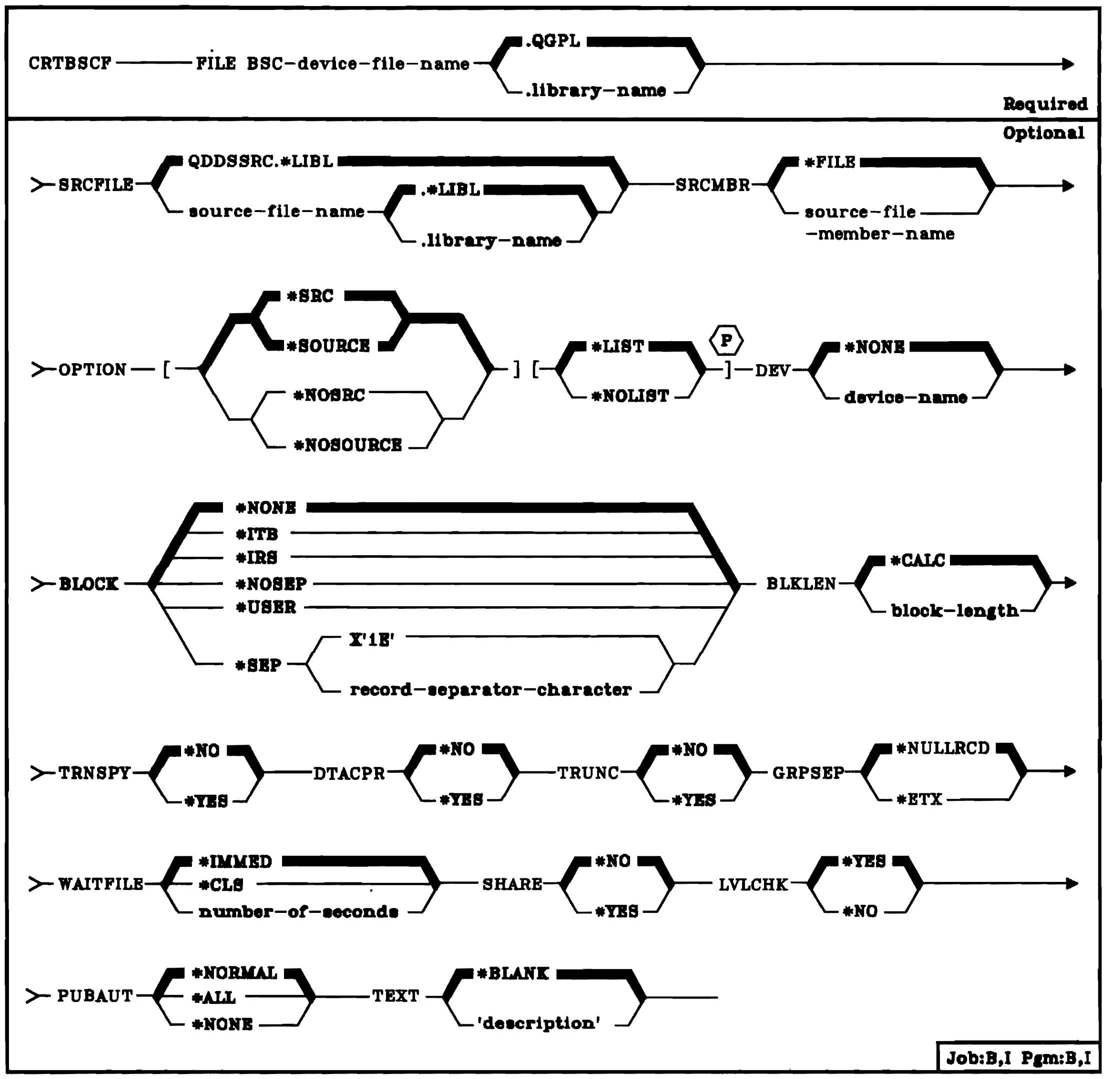
```
CPYSPLF FILE(QPRINT) TOFILE(MYFILE.MYLIB) JOB(PAYROLL02) +
      MBROPT(*ADD) CTLCHAR(*FCFC) CHLVAL( (1 3) (4 15) )
```

In this example, the file QPRINT (the only file of that name left in the job PAYROLL02) will be copied to the first member of the physical file found in library MYLIB. The newly copied data is added to data existing in the member. The FCFC 1-byte print control characters will be used and will take advantage of the assigned channel values in formatting the output. The assigned channel values as specified on the command are as follows:

Line 3 assigned to channel 1
Line 15 assigned to channel 4

CRTBSCF (Create BSC File) Command

The Create BSC File (CRTBSCF) command creates a device file for use with BSC devices. You select the appropriate BSC file parameters on the basis of the type of BSC device with which your system is to communicate.



FILE Parameter: Specifies the qualified name of the BSC device file being created. If no library qualifier is given, the file is stored in QGPL. (If the file is to be used by an HLL (high-level language) program, the file name should be consistent with the naming rules of that language and should be unique within the library; otherwise, the file must be renamed in the program itself.)

SRCFILE Parameter: Specifies the name of the source file that contains the DDS (data description specifications) used to create the record formats for the device file. (For the specifications that can be made in DDS, refer to the *CPF Reference Manual-DDS*.)

QDDSSRC: The IBM-supplied DDS source file in QGPL containing the source descriptions used to create the BSC file. Each member of QDDSSRC contains the source description of one physical, logical, or device file. Initially, QDDSSRC contains no source descriptions.

qualified-source-file-name: Enter the qualified name of the source file that contains the DDS to be used to create the BSC device file. (If no library qualifier is given, *LIBL is used to find the source file.)

SRCMBR Parameter: Specifies the name of the member in the data base source file that contains the DDS for this BSC device file.

*FILE: The source file name is the same as the device file name specified in the FILE parameter.

source-file-member-name: Enter the name of the member in the source file (specified by SRCFILE) that is to be used to create the BSC device file.

OPTION Parameter: Specifies the type of output listing to be produced when the file is created.

*SRC or *SOURCE:* A listing of source statements used to create the file, and of any errors that occur.

**NOSRC* or **NOSOURCE:* No listing of the source statements is to be generated unless errors are detected. If errors are detected, they are listed, along with the keyword or record format that caused the error.

*LIST: An expanded source listing is to be generated, showing a detailed list of the file specifications that result from the source statements and references to other file descriptions. This listing shows the file and field keywords and attributes.

**NOLIST:* No expanded source listing is to be generated.

DEV Parameter: Specifies the name of the System/38 BSC device that is to be used with the BSC device file to send and receive data records.

*NONE: No device name is to be specified. Any device names to be specified must be specified later in the CHGBSCF command or the OVRBSCF command, or in the HLL (high-level language) program that opens the file.

device-name: Enter the name of the BSC device that is to be used with this BSC file. The device name must be known to the system via a device description.

BLOCK Parameter: Specifies whether the system or user will block and deblock transmitted records. With this parameter, you may specify one of the following conditions of record formatting:

No blocking/deblocking: The record format described in the DDS is the format for both the record and the block.

User blocking/deblocking: You must provide the BSC controls needed to describe the record format to the system.

System blocking with record separator characters: You specify the record separator character used by the system to determine record boundaries within the block.

System blocking of fixed-length records: The system uses fixed-length records, and blocks/deblocks records accordingly. The record separator character is added when a record is transmitted, and removed before the record is returned to your program. This occurs for every case but *user blocking/deblocking*.

If you specify a parameter value other than *NONE or *USER, records will be blocked as required by the system for output and deblocked on input. Blocking may be done with or without record separator characters. If TRNSPY(*YES) is specified, the records may be blocked without record separator characters by specifying BLOCK(*NOSEP), or the records may be transmitted one record at a time by specifying BLOCK(*NONE). By specifying BLOCK(*USER), you may block records to include the BSC transparency controls. If TRNSPY(*NO) is specified, all blocking options are valid. The record length, when used, is obtained from the device file. A maximum of 512 records will be blocked for transmitting. When the system blocks and deblocks the records, record separator characters and control characters will not be passed to your program as data.

*NONE: Specifies that no blocking or deblocking will be done by the system.

**ITB:* Specifies that the records are to be blocked or deblocked, based on the location of an ITB (intermediate text block) control character. For input files, a record will be delimited by locating the next ITB character. An ETX (end of text) or ETB (end-of-transmission block) character will be used as an ITB character to delimit records. For output files, an ITB character will be inserted after the record. If that is the last character of the block, the ITB will be replaced by an ETX or an ETB character.

**IRS:* Specifies that the records are to be blocked or deblocked based on the location of an IRS (interrecord separator) character. For input files, a record will be delimited by locating the next IRS character. For output files, an IRS character will be inserted after the record.

**NOSEP:* Specifies that no record separator character is contained within the transmission block sent to or received from the device. The system will block and deblock the records according to a fixed record length, as specified in the DDS format specifications.

**USER:* Specifies that your program is to provide all control characters, including record separator characters, BSC framing characters, transparency characters, and so forth, necessary to transmit records.

When transmitting records, BSC device support will scan the buffer for the last non-blank byte to determine the length of the data to be transmitted. For this reason, you must ensure that the unused portion of the buffer contains blanks.

For receiving, you must specify with an ETX control character the end of the received text. BSC device support will pad the remaining buffer space with blanks.

This method of blocking allows you to transmit and receive variable-length data blocks by using a single record format capable of accommodating the maximum block length. Except for the padding and truncating with blanks, BSC device support simply passes the data to and from the system when user blocking is specified.

If you are using the Remote Job Entry Facility, BLOCK(*USER) must be specified. For more information on RJEF, refer to the *RJEF Programmer's Guide*.

Before selecting this option, you should have a good understanding of the device and of the BSC support characteristics. For more information on BSC support characteristics, refer to the *IBM System/38 Data Communications Programmer's Guide*, SC21-7825.

**SEP:* Specifies that the records are to be blocked or deblocked based on the location of a user-specified record separator character. For input files, a record will be delimited by locating the next record separator character. For output files, a record separator character will be inserted after the record.

record-separator-character: Specifies a unique one-byte record separator character. The record separator character may be specified as two hexadecimal characters, as in BLOCK(*SEP X'FD'), or the character may be specified as a single character, as in BLOCK(*SEP @). If a record separator character is not specified, the record separator character of X'1E' is used.

The following is a list of BSC control characters that must not be used as record separator characters:

EBCDIC	**BSC Control**
X'01'	SOH (Start of header)
X'02'	STX (Start of text)
X'03'	ETX (End of text)
X'10'	DLE (Data link escape)
X'1D'	IGS (Interchange group separator)
X'1F'	ITB (Intermediate text block)
X'26'	ETB (End-of-transmission block)
X'2D'	ENQ (Enquiry)
X'32'	SYN (Synchronization)
X'37'	EOT (End of transmission)
X'3D'	NAK (Negative acknowledgment)

You must be certain that none of these control characters are specified in your data as record separator characters.

BLKLEN Parameter: Specifies the maximum block length (in bytes) for data to be transmitted.

*CALC: The block length is to be determined by the system. The length will be the greater of 512 bytes or the length of the largest record in the device file.

block-length: The maximum block length of records to be sent when using this device file. The value must be at least the size of the largest record to be sent. Valid values are 1 through 8192.

TRNSPY Parameter: Specifies whether the text transparency feature is to be used when sending blocked records. The text transparency feature permits the transmission of all 256 EBCDIC character codes; you should use this feature when transmitting packed or binary data fields.

*NO: The text transparency feature is not to be used.

*YES: The text transparency feature is to be used, which permits the transmission of all 256 EBCDIC character codes. *YES is valid only when BLOCK(*NONE), BLOCK(*NOSEP), or BLOCK(*USER) is specified.

Note: Transparency of received data is determined by the data stream; therefore, this parameter is not relevant for received data. If TRNSPY(*YES) is specified with BLOCK(*USER), BSC ignores the transparency indicator during put operations. You must provide the proper controls with the data in order to get transparent transmission of data. For example, you must initially specify the DLE and STX control characters; System/38 provides the remaining control characters for transparent transmission of data.

DTACPR Parameter: Specifies whether blanks in BSC data will be compressed for output and decompressed for input. DTACPR(*YES) cannot be specified if TRNSPY(*YES) is specified, or if the line description specifies CODE(*ASCII).

*NO: No data compression or decompression is to occur.

*YES: Data is to be compressed for output and decompressed for input.

TRUNC Parameter: Specifies whether trailing blanks are to be removed from output records. TRUNC(*YES) cannot be specified if BLOCK(*NOSEP) or TRNSPY(*YES) is specified.

*NO: Trailing blanks are not to be removed from output records.

*YES: Trailing blanks are to be removed from output records.

GRPSEP Parameter: Specifies a separator for groups of data (data sets, documents, and so forth).

*NULLRCD: Specifies that a null record (STXETX) is to be used as a data group separator.

*ETX: A transmission block ending with the BSC control character ETX is to be used as a data group separator.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is to be made.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the BSC device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the BSC device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a write operation in that program produces the next output record.

*NO: An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Level checking cannot be done unless the program contains the record format identifiers.

*YES: The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not match, an open exception occurs and an error message is sent to the program requesting the open.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

PUBAUT Parameter: Specifies what authority for the BSC device file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the device file.

**ALL:* The public has complete authority for the device file.

**NONE:* The public cannot use the device file.

TEXT Parameter: Specifies the user-defined text that describes the BSC device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTBSCF FILE(TRANSD1.COMM1) DEV(DEVS34) BLOCK(*IRS) +
     WAITFILE(30) TEXT('S38 to S34 data transfer')
```

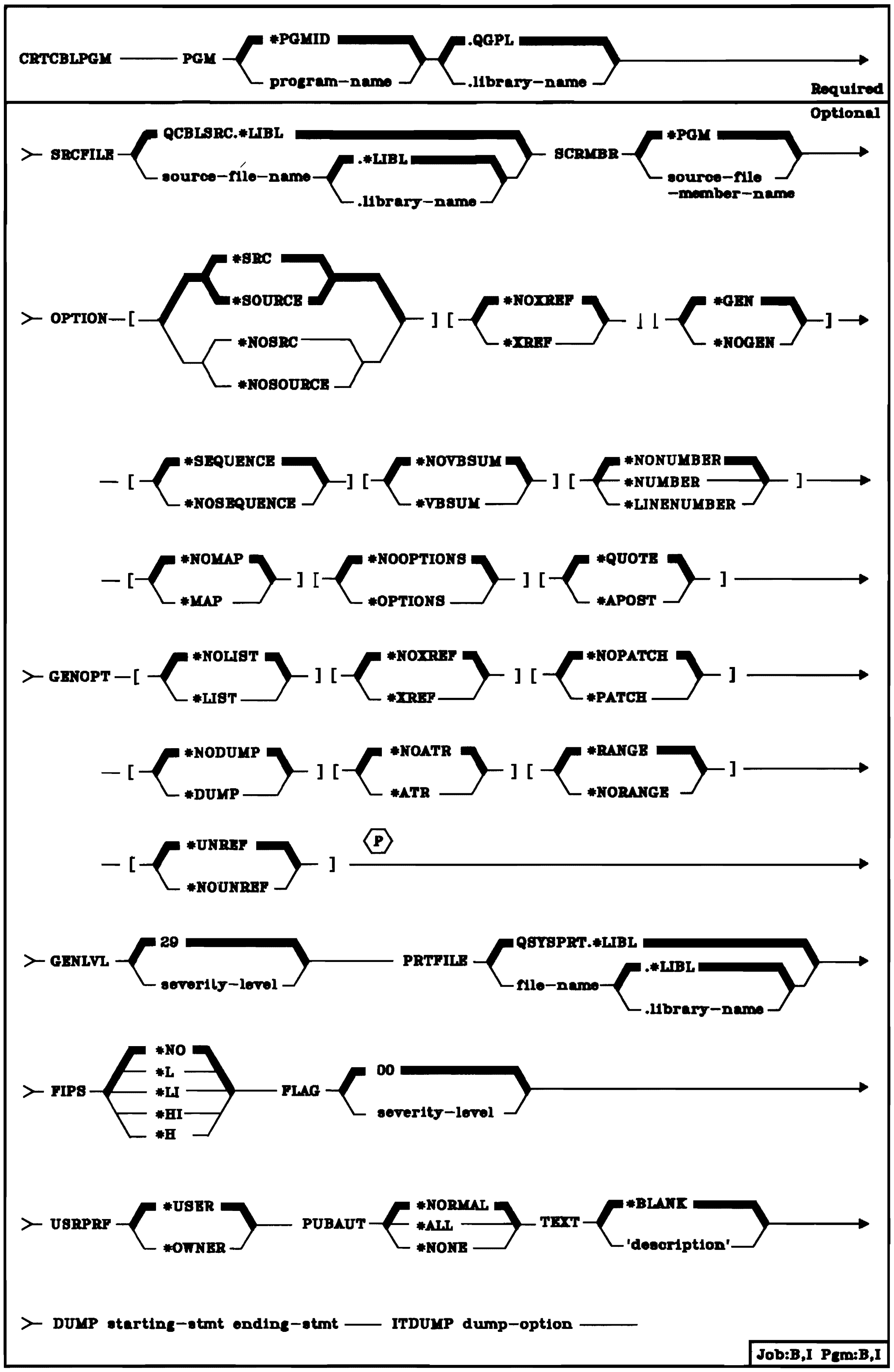
This command creates a BSC device file named TRANSD1 in library COMM1. The record formats for this device file will be taken from a source member (also named TRANSD1) in source file QDDSSRC. The device description to be used with this device is named DEVS34. Record blocking will be performed by the system, using IRS as the record separator character. The length of the transmission block will be calculated by the system, based on the record format length. The program in which this command occurs will wait up to 30 seconds for the file resources to be allocated.

CRTCBLPGM (Create COBOL Program) Command

The Create COBOL Program (CRTCBLPGM) command invokes the COBOL compiler, to compile a COBOL source member into an executable program. The command is valid in batch and interactive jobs and from other programs.

The COBOL high-level language is part of the IBM System/38 COBOL Program Product, Program 5714-CB1. For more information, refer to the *IBM System/38 COBOL Reference Manual and Programmer's Guide,* SC21-7718.

Restriction: All object names specified on the CRTCBLPGM command must be composed of no more than 10 alphameric characters, the first of which must be alphabetic.


CRTCBLPGM
PGM
*PGMID
program-name
.QGPL
.library-name
Required
Optional
SRCFILE
QCBLSRC.*LIBL
source-file-name
.*LIBL
.library-name
SCRMBR
*PGM
source-file
-member-name
OPTION
*SRC
*SOURCE
*NOSRC
*NOSOURCE
*NOXREF
*XREF
*GEN
*NOGEN
*SEQUENCE
*NOSEQUENCE
*NOVBSUM
*VBSUM
*NONUMBER
*NUMBER
*LINENUMBER
*NOMAP
*MAP
*NOOPTIONS
*OPTIONS
*QUOTE
*APOST
GENOPT
*NOLIST
*LIST
*NOXREF
*XREF
*NOPATCH
*PATCH
*NODUMP
*DUMP
*NOATR
*ATR
*RANGE
*NORANGE
*UNREF
*NOUNREF
P
GENLVL
29
severity-level
PRTFILE
QSYSPRT.*LIBL
file-name
.*LIBL
.library-name
FIPS
*NO
*L
*LI
*HI
*H
FLAG
00
severity-level
USRPRF
*USER
*OWNER
PUBAUT
*NORMAL
*ALL
*NONE
TEXT
*BLANK
'description'
DUMP starting-stmt ending-stmt
ITDUMP dump-option
Job:B,I Pgm:B,I

Note: The number of entries in the Object Definition Table (ODT) and the amount of storage required by a COBOL program varies with the number and kinds of COBOL statements used in the program. A combination of these factors can cause System/38 internal size limits for the program to be exceeded. If this occurs, GENOPT(*NOUNREF) can be specified. If the problem still exists, the program must be rewritten as multiple programs rather than as one program.

PGM Parameter: Specifies the qualified name by which the compiled COBOL program is to be known and the library in which the compiled program is to be located. (If no library qualifier is specified, QGPL is used to find the program.)

PGMID: The name specified as the PROGRAM-ID is used.

program-name: Specifies the name by which the compiled COBOL program is known. The first program in the batch job uses this name, while all other programs use the name specified in the PROGRAM-ID paragraph in the source program.

SRCFILE Parameter: Specifies the name of the source file that contains the COBOL source program to be compiled.

QCBLSRC.*LIBL: Specifies that the IBM-supplied source file, QCBLSRC, contains the COBOL source to be compiled.

qualified-source-file-name: Enter the qualified name of the source file that contains the COBOL source program to be compiled. (If no library qualifier is given, *LIBL is used to find the file.) This source file should have a record length of 92. The source file can be a data base file, a device file, or an inline data file.

SRCMBR Parameter: Specifies the name of the member of the source file that contains the COBOL source program to be compiled. This parameter can be specified only if the source-file name in the SRCFILE parameter is a data base file.

*PGM: The COBOL source program to be compiled is in the member of the source file that has the same name as that specified for the compiled program in the PGM parameter. If *PGMID is specified for the PGM parameter, the SRCMBR parameter is not used. For a data base source file, the first member is used.

source-file-member-name: Enter the name of the member that contains the COBOL source.

OPTION Parameter: Specifies the options to use when the COBOL source is compiled.

<u>*SOURCE</u> or **SRC:* The compiler produces a source listing, consisting of the COBOL source input and all compile-time error messages.

**NOSOURCE* or **NOSRC:* The compiler does not produce a source listing.

<u>*NOXREF:</u> The compiler does not produce a cross-reference listing for the source program.

**XREF:* The compiler produces a cross-reference listing for the source program.

<u>*GEN:</u> The compiler generates an executable program after the program is compiled.

**NOGEN:* The compiler does not generate an executable program after the program is compiled.

<u>*SEQUENCE:</u> The reference numbers are checked for sequence errors. Sequence errors do not occur if the *LINENUMBER option is specified.

**NOSEQUENCE:* The reference numbers are not checked for sequence errors. Because SEQUENCE is the default option, sequence errors are flagged until the NOSEQUENCE option is recognized. When NOSEQUENCE is the last item specified on a record, sequence checking is in effect between that record and the next record.

<u>*NOVBSUM:</u> Verb usage counts are not printed.

**VBSUM:* Verb usage counts are printed.

<u>*NONUMBER:</u> The source file sequence numbers are used for reference numbers.

**NUMBER:* The user-supplied sequence numbers (columns 1 through 6) are used for reference numbers.

**LINENUMBER:* The compiler-generated sequence numbers are used for reference numbers. This option combines program source code and source code introduced by COPY statements into one consecutively numbered sequence. Use this option when you specify FIPS flagging.

<u>*NOMAP:</u> The compiler does not list the Data Division map.

**MAP:* The compiler lists the Data Division map.

*NOOPTIONS: Options in effect are not listed for this compilation.

**OPTIONS:* Options in effect are listed for this compilation.

*QUOTE: Specifies the delimiter " used for nonnumeric literals and Boolean literals.

**APOST*: Specifies the delimiter ' used for nonnumeric literals and Boolean literals.

GENOPT Parameter: Specifies the options to use when the executable program is created. The listings could be required if a problem occurs in COBOL.

*NOLIST: No IRP (intermediate representation of a program), associated hexadecimal code, or error messages are listed.

**LIST:* The IRP, its associated hexadecimal code, and any error messages are listed.

*NOXREF: A cross-reference listing of all objects defined in the IRP is not produced.

**XREF:* A cross-reference listing of all objects defined in the IRP is produced.

*NOPATCH: Space is not reserved in the compiled program for a program patch area.

**PATCH:* Space is reserved in the compiled program for a program patch area. The program patch area can be used for debugging purposes.

*NODUMP: The program template is not listed.

**DUMP:* The program template is listed.

*NOATR: The attributes for the IRP source are not listed.

**ATR:* The attributes for the IRP source are listed.

*RANGE: Execution-time checks are performed for substring and subscript ranges.

**NORANGE:* Execution-time checks are not performed.

*UNREF: Unreferenced data items are included in the compiled program.

**NOUNREF:* Unreferenced data items are not included in the compiled program. This option reduces the number of ODT entries used, allowing a larger program to be compiled.

GENLVL Parameter: Specifies when a program is generated. A severity-level value, corresponding to the severity level of the messages produced during compilation, can be specified in this parameter. If errors occur in a program with a severity level greater than the value specified in this parameter, an executable program is not generated. For example, if you do not want a program generated if you have messages with a severity level of 20 or greater, specify 19 in this parameter.

29: If a severity-level value is not specified, the default severity-level is 29.

severity-level: A two-digit number, 00 through 29, can be specified.

PRTFILE Parameter: Specifies the qualified name of the file to which the compiler listing is directed and the library in which the file is located. The file should have a minimum record length of 132. If a file with a record length less than 132 is specified, information is lost.

QSYSPRT.*LIBL: If a file-name is not specified, the compiler listing is directed to the IBM-supplied file, QSYSPRT.

file-name: Enter the name of the file to which the compiler listing is directed.

FIPS Parameter: The source program is FIPS-flagged (Federal Information Processing Standard) for the following specified level (and higher). Use the *LINENUMBER option to ensure unique reference numbers in FIPS flagging messages.

*NO: The source program is not FIPS-flagged.

**L:* FIPS flag for low level and higher.

**LI:* FIPS flag for low-intermediate level and higher.

**HI:* FIPS flag for high-intermediate level and higher.

**H:* FIPS flag for high level.

FLAG Parameter: Specifies the minimum severity level of messages to be listed.

00: All messages are to be listed.

severity-level: Enter a two-digit number that specifies the minimum severity level of messages that are to be listed. Messages that have severity levels of the specified value or higher are listed.

USRPRF Parameter: Specifies under which user profile the compiled COBOL program is to be executed. The profile of either the program owner or the program user is used to execute the program and control which objects can be used by the program (including what authority the program has for each object).

*USER: The program user's user profile is to be used when the program is executed.

**OWNER:* The user profiles of both the program's owner and user are to be used when the program is executed. The collective sets of object authority in both user profiles are to be used to find and access objects during the program's execution. Any objects that are created during the program are owned by the program's user.

PUBAUT Parameter: Specifies what authority for the program and its description is being granted to the public. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the compiled program. Any user can execute the program, but cannot change it or debug it.

**ALL:* The public has complete authority for the program.

**NONE:* The public cannot use the program.

TEXT Parameter: Lets the user enter text that briefly describes the program and its function.

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

DUMP Parameter: An IBM COBOL debugging aid. For IBM service personnel.

ITDUMP (n) Parameter: An IBM debugging aid. Causes the compiler to dump the internal text at various times during the compilation. For IBM service personnel.

Example

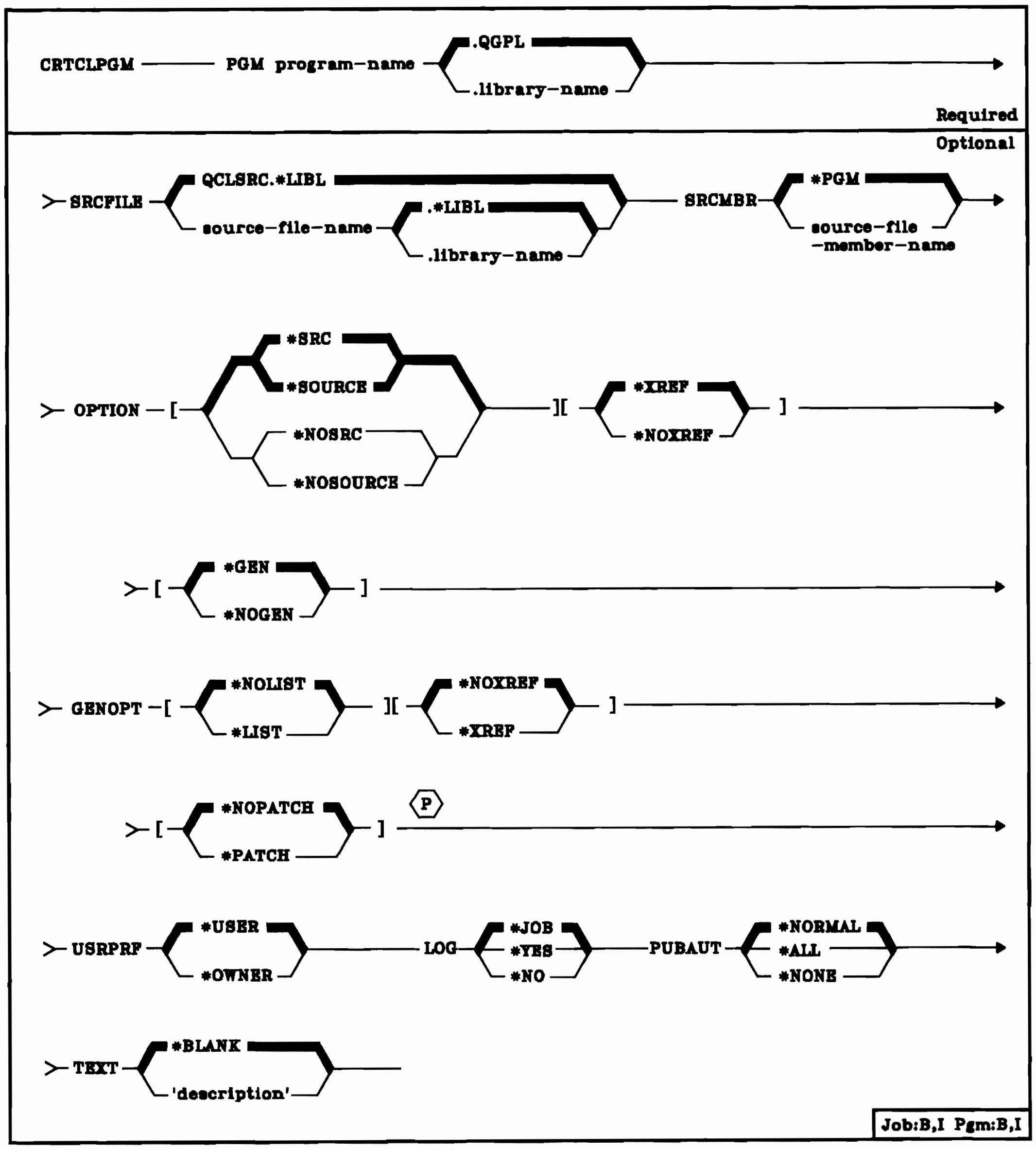
```
CRTCBLPGM PGM(STATS.ACCTS) SRCFILE(ACTIVE.ACCTS) GENOPT(*PATCH) +
    GENLVL(19) FLAG(19) TEXT('Statistical analysis program for active +
    accounts')
```

This command creates a COBOL program named STATS in library ACCTS. The source program ACTIVE also resides in library ACCTS. The compiler will reserve space in the compiled program for program patches. If messages with a severity level of 20 or higher are generated during compilation, they will be listed and an executable program will not be generated.

CRTCLPGM (Create Control Language Program) Command

The Create CL Program (CRTCLPGM) command creates an executable CL program from the specified CL source program. The command is valid in batch and interactive jobs, and in both compiled and interpreted CL.

Restriction: The amount of auxiliary storage occupied by a compiled program varies by the number of commands in the program, the kinds of functions performed by the commands (for example: display, create, add, call), and the kinds of parameter values specified (variables versus constants). Some combinations of these factors can cause System/38 internal size limits for the program to be exceeded (an unlikely occurrence). When the limits are exceeded, the program must be rewritten, usually as multiple programs rather than one program.



PGM Parameter: Specifies the qualified name by which the compiled CL program is to be known. (If no library qualifier is given, the created program is stored in the general purpose library, QGPL).

SRCFILE Parameter: Specifies the name of the source file that contains the CL source program to be compiled. The program, created in executable form, is known by the name given in the PGM parameter.

QCLSRC: The IBM-supplied source file, QCLSRC, contains the CL source program to be compiled. (If no library qualifier is given, *LIBL is used to find the file.)

qualified-source-file-name: Enter the qualified name of the source file that contains the CL source program to be compiled. (If no library qualifier is given, *LIBL is used to find the file.) The source file can be a data base file, a device file, or an inline data file.

SRCMBR Parameter: Specifies the name of the member of the source file that contains the CL source program to be compiled.

*PGM: The CL source program to be compiled is in the member of the source file that has the same name as that specified in the PGM parameter for the compiled program.

source-file-member-name: If the member name is not the same as the name of the program being created, enter the name of the member that contains the CL source program.

OPTION Parameter: Specifies the types of output listings to be produced when this command is executed, and whether an executable program is to be generated.

*SRC or **SOURCE:* The compiler is to generate a listing of the source input used to compile the program. If neither *SOURCE nor *NOSOURCE is specified, *SOURCE is assumed.

**NOSRC* or **NOSOURCE:* A complete compiler source listing is not to be generated; only compiler errors are to be listed.

*XREF: The compiler is to generate a cross-reference listing of references to variables and/or labels in the source. If *NOSOURCE is specified, *NOXREF is always assumed. Otherwise, if neither *XREF nor *NOXREF is specified or if they are both specified, *XREF is assumed.

**NOXREF:* No cross-reference listing of references to variables and data items in the source is to be generated.

<u>*GEN:</u> The compiler is to generate an executable program and place it in the appropriate library. If neither *GEN nor *NOGEN is specified or if they are both specified, *GEN is assumed.

**NOGEN:* An executable program is not to be generated. The compiler is to syntax check the source and (if *SOURCE is specified) produce a source listing.

GENOPT Parameter: Specifies the options to be used when the CL program is compiled. This parameter specifies whether a listing of the IRP and the machine instructions generated by the program resolution monitor is to be produced, whether a cross-reference listing is to be produced, and whether a program patch area is to be included in the compiled program. (The IRP is an intermediate representation of the program being compiled.)

<u>*NOLIST:</u> No listing is to be produced of the IRP and the generated machine instructions. If neither *LIST nor *NOLIST is specified or if they are both specified, *NOLIST is assumed.

**LIST:* A listing is to be produced of the IRP and the generated machine instructions.

<u>*NOXREF:</u> No cross-reference listing is to be generated of variable and data item references in the IRP. If *NOLIST is specified, *NOXREF is always assumed. Otherwise, if neither *XREF nor *NOXREF is specified or if they are both specified, *NOXREF is assumed.

**XREF:* A cross-reference listing of variable and data item references in the IRP is to be produced.

<u>*NOPATCH:</u> No space is to be reserved in the compiled program for a program patch area. If neither *PATCH nor *NOPATCH is specified or if they are both specified, *NOPATCH is assumed.

**PATCH:* Space is to be reserved in the compiled CL program for a program patch area.

USRPRF Parameter: Specifies under which user profile the compiled CL program is to be executed. The profile of either the program owner or the program user is used to execute the program and control which objects can be used by the program (including what authority the program has for each object).

<u>*USER:</u> The program user's user profile is to be used when the program is executed.

**OWNER:* The user profiles of both the program's owner and user are to be used when the program is executed. The collective sets of object authority in both user profiles are to be used to find and access objects during the program's execution. Any objects that are created during the program are owned by the program's user.

LOG Parameter: Specifies the logging options for a CL program that is to be created.

**JOB:* Specifies that logging of commands in an executing CL program depends upon the status of the job's logging flag (see the LOGCLPGM parameter of the *CHGJOB (Change Job) Command*). For the logged commands to be listed, the logging level of the jobs must be 3 or 4.

**YES:* Specifies that logging of commands is to be performed in all cases.

**NO:* Specifies that logging of commands is not to be performed.

PUBAUT Parameter: Specifies what authority for the class and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

**NORMAL:* The public has only operational rights for the compiled CL program. Any user can execute the program, but cannot change it or debug it.

**ALL:* The public has complete authority for the program.

**NONE:* The public cannot use the program.

TEXT Parameter: Lets the user enter text that briefly describes the compiled CL program. (For an expanded description of the TEXT parameter, see Appendix A.)

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTCLPGM  PAYROLL TEXT('Payroll Program')
```

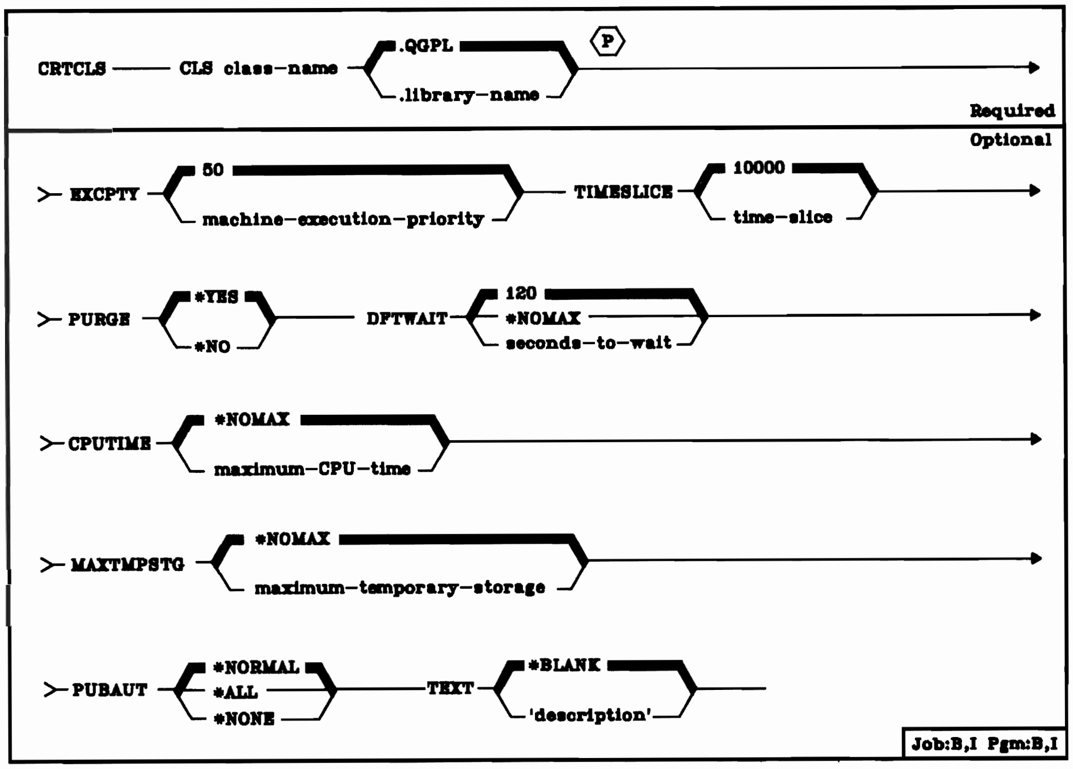
This command invokes the CL compiler to compile a program named PAYROLL. The source program is in the default source file QCLSRC, in a member named PAYROLL. A compiler listing will be generated. The program will be executed under the program user's user profile and can be executed by any system user.

```
CRTCLPGM  PGM(PARTS) SRCFILE(PARTDATA.MYLIB) PUBAUT(*NONE) +
      TEXT('This program displays all parts data')
```

This command creates a CL program named PARTS and stores it in the library QGPL. The source for the program is in the PARTS member of the source file PARTDATA in the library MYLIB. A listing will be generated. This program, which has no public use, can be executed under the profile of the user that is running the program, who could be the owner or another user that the owner has explicitly authorized by name in the GRTOBJAUT command.

CRTCLS (Create Class) Command

The Create Class (CRTCLS) command creates a class object and specifies the attributes to be contained in the class. The class defines the processing parameters for routing steps that are to use the class; the class to be used by a routing step is specified in the subsystem description routing entry that is used to initiate the routing step.



CLS Parameter: Specifies the qualified name by which the class of attributes will be known. (If no library qualifier is given, the class is stored in the general purpose library, QGPL.) (For an expanded description of the CLS parameter, see Appendix A.)

EXCPTY Parameter: Specifies the execution priority for routing steps that will use the class being created. Machine execution priority is a value, 1 (highest priority) through 99 (lowest), that represents the importance of the routing step when it competes with other routing steps in the class for the machine resources. This value represents the relative, not absolute, importance of the routing step. For example, a routing step with an execution priority of 25 is *not* twice as important as one with an execution priority of 50.

50: Routing steps that use this class are to have an execution priority of 50.

machine-execution-priority: Enter the execution priority that routing steps using this class are to have.

TIMESLICE Parameter: Specifies the maximum amount of processor time, in milliseconds, given to a routing step using this class before other routing steps, waiting to use the same storage pool, are given the opportunity to execute. The time slice establishes the amount of time needed by the routing step to accomplish a meaningful amount of processing. At the end of the time slice, the routing step might be put in an inactive state so that other routing steps can become active in the storage pool.

10000: A maximum execution time of 10 000 milliseconds is allocated to each routing step each time it is allowed to process.

time-slice: Enter the maximum amount of time, in milliseconds, that each routing step in this class can have to execute when it is given processing time. Valid entries are 1 through 9999999 (that is, 9 999 999 milliseconds or 9999.999 seconds).

PURGE Parameter: Specifies whether or not the job is to be marked eligible to be moved out of main storage and put into auxiliary storage at the end of a time slice or upon entering a long wait (such as waiting for a work station user's response).

*YES: The job is eligible to be moved out of main storage and put into auxiliary storage.

**NO:* The job is not to be moved out of main storage. However, when some of main storage is needed to execute the routing steps of other jobs running in the same storage pool, pages belonging to this job are moved (one at a time) to auxiliary storage to accommodate pages needed for other jobs. Then, when this job executes again, its pages are returned to main storage as they are needed.

DFTWAIT Parameter: Specifies the default maximum wait time (in seconds) that processing of a routing step is to be held up until a System/38 instruction that performs a wait completes its execution. This default wait time is used when a wait time is not otherwise specified for a given situation. Normally, this would be the amount of time the system user would be willing to wait for the system before the request is canceled.

If the wait time for any one instruction is exceeded, an error message can be displayed or it can be automatically handled by a MONMSG command.

120: An instruction has a maximum of 120 seconds in which to complete execution.

**NOMAX:* There is no time limit on how long the system is to wait for the execution of an instruction to be completed.

seconds-to-wait: Enter a value, 1 through 9999999 (9 999 999 seconds), that specifies the maximum time that the system is to wait for the System/38 instruction to be completely executed.

CPUTIME Parameter: Specifies the maximum CPU time (in milliseconds) that a routing step using this class can have to completely execute the entire routing step. If not finished before the maximum time is used up, execution of the routing step is terminated.

*NOMAX: There is no time limit on how long the routing step may take.

maximum-CPU-time: Enter the maximum amount of CPU time, in milliseconds, that the routing step has in which to complete execution. Valid entries are 1 through 9999999 (that is, 9 999 999 milliseconds or 9999.999 seconds).

MAXTMPSTG Parameter: Specifies the maximum amount of temporary (auxiliary) storage (in K-bytes) that a routing step in this class can use for processing. This temporary storage is used for storage required by the program itself and by implicitly created internal system objects used to support the routing step. (It is not storage in the QTEMP library.) If the maximum temporary storage is exceeded by a routing step, the routing step is terminated. This parameter does not apply to the use of permanent storage, which is controlled through the user profile.

*NOMAX: There is no maximum amount of temporary storage for the routing step that uses this class.

maximum-temporary-storage: Enter a value in K-bytes (1 through 9999999) that specifies the maximum amount of temporary storage that a routing step in this class can have.

PUBAUT Parameter: Specifies what authority for the class and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the class.

**ALL:* The public has complete authority for the class.

**NONE:* The public cannot use the class.

TEXT Parameter: Lets the user enter text that briefly describes the class being created. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTCLS  CLS(CLASS1) EXCPTY(60) TIMESLICE(900) +
      TEXT('This class for all batch jobs +
      from Dept 4836')
```

This command creates a class called CLASS1 that has public operational rights; the class is stored in the QGPL library. The user text 'This class for all batch jobs from Dept 4836' describes the class. The attributes of this class provide a machine execution priority of 60 and a time slice of 900 milliseconds. If the routing step has not finished execution at the end of a time slice, it is eligible to be moved out of main storage until it is allocated another time slice. The defaults for the other parameters are assumed.

CRTCMD (Create Command) Command

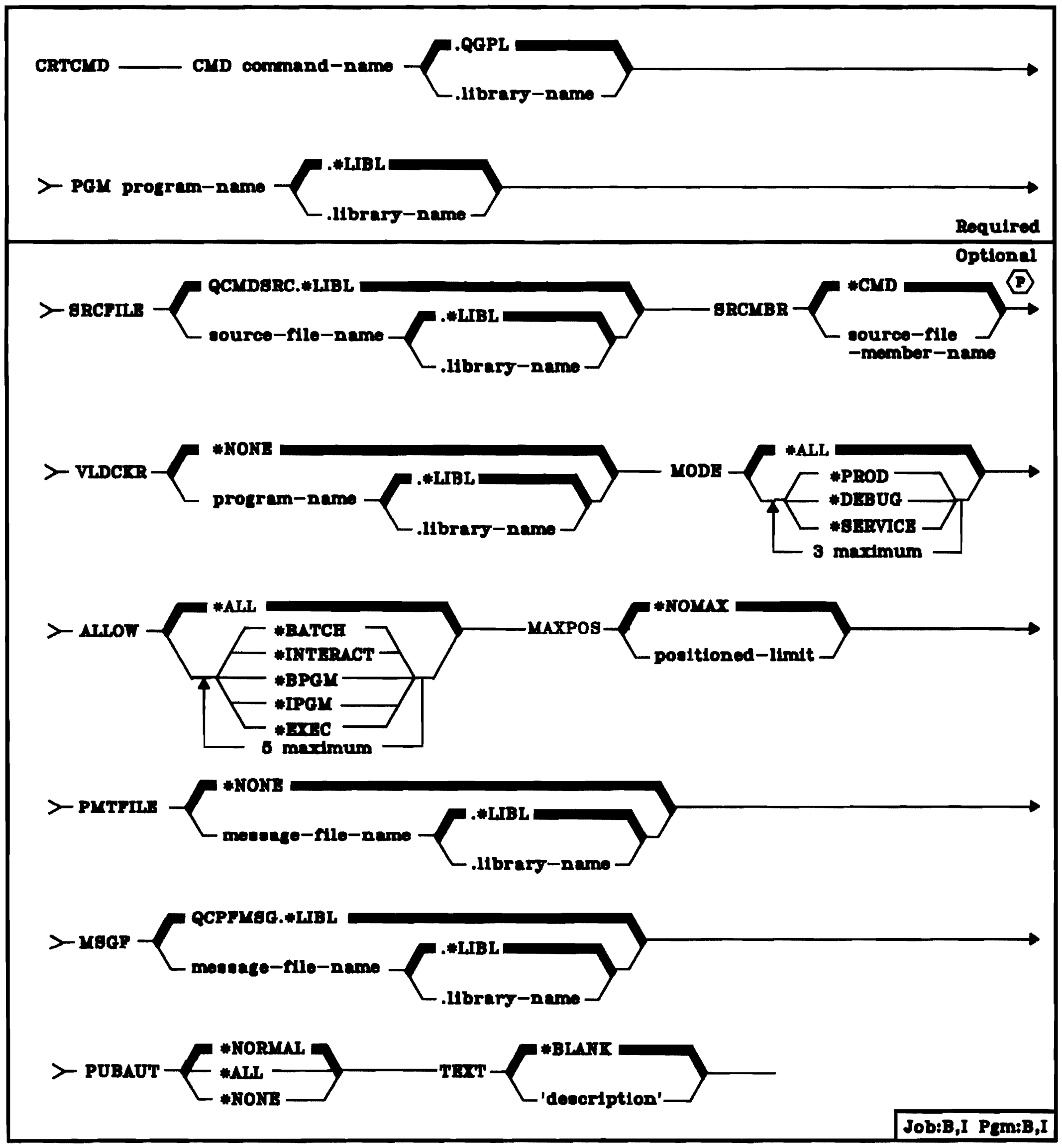
The Create Command (CRTCMD) command creates a user-defined command (that is, a command definition) that can use the same command processing support that is used by IBM-supplied commands. The command definition is an object that can be stored in the general purpose library (QGPL) or in a user library. To update an existing command (for example, change the name of one of its parameter keywords), the command must first be deleted by the DLTCMD command and then created again by the CRTCMD command. However, some of the values can be changed by the CHGCMD command.

To create a command, a set of command definition statements are entered into a source file. The CRTCMD command is used to process the source file and create a command definition object. The following command definition statements are used as needed:

- Command statement (CMD): One CMD statement is needed for each command being defined.
- Parameter statement (PARM): One PARM statement is required for each command parameter in the command being defined. It defines the parameter to be passed to the CPP.
- Element statement (ELEM): An ELEM statement further defines a parameter that is to be a list of values. One statement is required for each possible element of the list.
- Qualifier statement (QUAL): A QUAL statement is required to describe each part of a qualified name that can be accepted for a parameter (defined in a PARM statement) or for an element in a list of values (defined in an ELEM statement).
- Dependent statement (DEP): The DEP statement indicates which parameters are dependent upon each other.

Refer to Chapter 5 for the descriptions of the five command definition statements and to the *CPF Programmer's Guide* for information about command definition.

Restriction: The CRTCMD command can be used only to create the command definition of an actual CL command. That is, it cannot be used to create definitions of *statements*, such as the command definition statements themselves.


CRTCMD
CMD command-name
.QGPL
.library-name
PGM program-name
.*LIBL
.library-name
Required
Optional
P
SRCFILE
QCMDSRC.*LIBL
source-file-name
.*LIBL
.library-name
SRCMBR
*CMD
source-file -member-name
VLDCKR
*NONE
program-name
.*LIBL
.library-name
MODE
*ALL
*PROD
*DEBUG
*SERVICE
3 maximum
ALLOW
*ALL
*BATCH
*INTERACT
*BPGM
*IPGM
*EXEC
5 maximum
MAXPOS
*NOMAX
positioned-limit
PMTFILE
*NONE
message-file-name
.*LIBL
.library-name
MSGF
QCPFMSG.*LIBL
message-file-name
.*LIBL
.library-name
PUBAUT
*NORMAL
*ALL
*NONE
TEXT
*BLANK
'description'
Job:B,I Pgm:B,I

CMD Parameter: Specifies the qualified name of the command to be created. (If the library qualifier is not specified, the command definition object is placed in the general purpose library, QGPL.)

PGM Parameter: Specifies the qualified name of the command processing program (CPP) that is to execute the command. (If no library qualifier is given, *LIBL is used to find the command's CPP at execution time.) The CPP does not have to exist until command execution time.

The parameters passed to the CPP are the ones defined by the command definition statements in the source file specified by the SRCFILE parameter.

SRCFILE Parameter: Specifies the name of the source file that is to contain the command definition statements.

QCMDSRC: The IBM-supplied source file, QCMDSRC, is to contain the command definition source. (If no library qualifier is given, *LIBL is used to find the file.)

qualified-source-file-name: Enter the qualified name of the source file that is to contain the source for the command being created. (If no library qualifier is given, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of the source file member containing the command definition statements.

*CMD: If this option is specified and the file specified in the SRCFILE parameter is a data base file, the name of the source file member is the name specified in the CMD parameter of this command.

source-file-member-name: Enter the name of the member in the source file specified by the SRCFILE parameter that contains the command definition statements that are to be used to create the command.

VLDCKR Parameter: Specifies the name of a program that, when the command is used, performs additional validity checking on the parameters in the command being created. The same parameters passed to the CPP (identified in the PGM parameter) are also passed to the validity checking program. The validity checker is invoked to perform additional user-defined validity checking beyond that specified by the command definition statements in the source file, and beyond normal control language syntax checking.

<u>*NONE:</u> There is no separate validity checking program for this command. All validity checking is done by the command analyzer and the command processing program.

qualified-program-name: Enter the qualified name of the validity checker that is to check the validity of the command being created. (If no library qualifier is given, *LIBL will be used to find the validity checker at execution time.) The validity checker does not have to exist until command execution time.

MODE Parameter: Specifies the modes of operation to which the newly defined command applies. One or more of the modes can be specified.

<u>*ALL:</u> The command is valid in all the modes of operation: production, debug, and service.

**PROD:* The command is valid in the production mode.

**DEBUG:* The command is valid in the debugging mode.

**SERVICE:* The command is valid in the service mode.

ALLOW Parameter: Specifies where the command can be executed. One or more of the following options can be specified.

<u>*ALL:</u> The command is valid in a batch input stream, in a CL program, or when executed interactively. It can also be passed to the system program QCAEXEC to be executed.

**BATCH:* The command is valid in a batch input stream, external to a compiled CL program.

**INTERACT:* The command is valid when executed interactively, external to a compiled CL program .

**BPGM:* The command can be included in a compiled CL program that executes in the batch input stream.

**IPGM:* The command can be included in a compiled CL program that executes interactively.

**EXEC:* The command can be used as a parameter on the CALL command and be passed as a character string to the system program QCAEXEC to be executed. If *EXEC is specified, either *BATCH or *INTERACT must also be specified.

MAXPOS Parameter: Specifies the maximum number of parameters that can be entered positionally for this command. This parameter value must be greater than the number of nonconstant required parameters and less than the total number of nonconstant parameters. Not included in the number specified for this parameter are those parameters of TYPE(*ZEROELEM), parameters with the CONSTANT attribute, and lists and qualified names whose ELEMs and QUALs have the CONSTANT attribute or are of TYPE(*ZEROELEM).

<u>*NOMAX:</u> No maximum positional coding limit is to be specified for this command.

positional limit: Specifies the maximum number of parameters that can be coded positionally for this command. Valid values are 0 through 75.

PMTFILE Parameter: Specifies the name of the message file from which the prompt text for the command is to be retrieved.

<u>*NONE:</u> No message file is needed for the prompt text. The text, if any, is supplied in the definition statements that define the command.

qualified-message-file-name: Enter the qualified name of the message file that is to contain the prompt text for the command. (If no library qualifier is given, *LIBL is used to find the file.)

MSGF Parameter: Specifies the name of the message file from which messages identified on the DEP statements used to define the command are to be retrieved. (The messages are those that can be sent if the command, while executing, encounters a parameter dependency error.) The messages referred to by this MSGF parameter are those whose message identifiers are specified in the MSGID parameter of one or more DEP definition statements, but whose identifiers do not have the 3-character prefix of *CPF*. (QCPFMSG is always used for messages that do have the CPF prefix.)

<u>QCPFMSG:</u> The IBM-supplied CPF message file, QCPFMSG, is the file from which CPF and non-CPF dependency error messages are to be retrieved.

qualified-message-file-name: Enter the qualified name of the message file from which the non-CPF prefixed error messages are to be retrieved. (If no library qualifier is given, the file is found by the library list that is in effect for the job in which the created command is being executed when a dependency error is detected.)

PUBAUT Parameter: Specifies what authority for the command and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. The name of the CPP that executes the command must be qualified so that the use of the command can be limited. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the command.

**ALL:* The public has complete authority for the command.

**NONE:* The public cannot use the command.

TEXT Parameter: Lets the user enter text that briefly describes this command and its function. The text in the command description can be displayed by the DSPOBJD command.

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTCMD  CMD(PAYROLL) PGM(PAY076) +
      SRCFILE(PAYSOURCE) PUBAUT(*NONE)
```

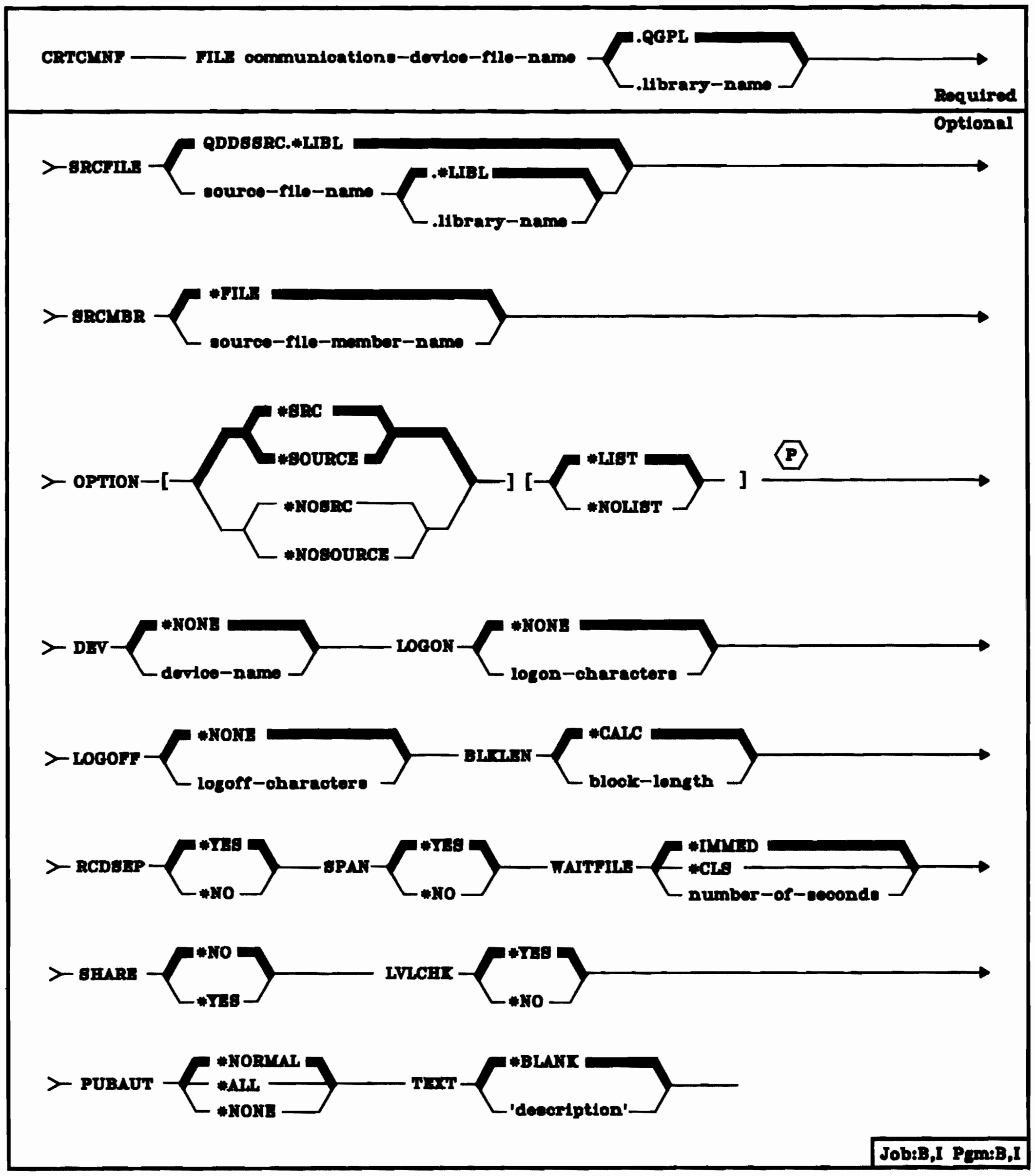
The command named PAYROLL is created from the source file PAYSOURCE. The command is private and calls the CPP named PAY076. It is a valid command when entered in a batch input stream, when compiled in a control language program, when entered interactively, or when passed to the QCAEXEC program.

CRTCMNF (Create Communications File) Command

The Create Communications File (CRTCMNF) command creates a communications device file. This file is used to send records to and receive records from a host system, such as an IBM System/370. The host system can be using IMS/VS (Information Management System/Virtual Storage) or CICS (Customer Information Control System) with the OS/VS1 or OS/VS2 operating system. Communications occur in an SNA (systems network architecture) environment using the SDLC (synchronous data link control) protocol. Refer to the *System/38 Data Communications Programmer's Guide* for additional information on using communications device files.

The device file contains the file description, which identifies the communications device to be used, and the record formats used by the application programs; the file does not contain data. The same device file can be used for both input and output operations. The file description is made up of information that is specified in two places: (1) in the source file that contains the data description specifications (if used); and (2) in the CRTCMNF command itself. The DDS contains the specifications for each record format in the device file and for each of the fields within each record format.

The CHGCMNF or OVRCMNF command can be used in a program to change or override the parameter values specified in the communications file description. Each changed value in the device file remains changed after the program ends. Each overridden value remains altered only for the execution of the program; once the program ends, the original parameter values specified for the communications file are used. Override commands must be executed before the communications file to be affected is opened for use by the program.



FILE Parameter: Specifies the qualified name by which the communications device file will be known. If no library qualifier is given, the file is stored in QGPL. (If the file is to be used by an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

SRCFILE Parameter: Specifies the name of the source file (if any) that contains the data description specifications to be used to create the communications device file. (The specifications that can be made in DDS are described in the *CPF Reference Manual—DDS*.)

QDDSSRC: The IBM-supplied DDS source file named QDDSSRC in the QGPL library contains the source descriptions to be used to create the communications file. Each member of QDDSSRC contains the source description of one file. (When shipped, QDDSSRC contains no descriptions.) (If no library qualifier is specified, *LIBL is used to find the source file.)

qualified-source-file-name: Enter the qualified name of the source file that contains the DDS to be used to create the communications device file. (If no library qualifier is given, *LIBL is used to find the source file.)

SRCMBR Parameter: Specifies the name of the member in the data base source file that contains the DDS for this communications device file.

*FILE: The source file member name is the same as the communications file name specified in the FILE parameter.

source-file-member-name: Enter the name of the member in the source file that contains the DDS to be used to create the device file.

OPTION Parameter: Specifies the type of output listing to be produced when the file is created.

*SRC or **SOURCE:* A listing of the source statements used to create the file, and of any errors that occur, is to be generated.

**NOSRC* or **NOSOURCE:* No listing of the source statements is to be generated unless errors are detected. If errors are detected, they are listed along with the keyword or record format that caused the error.

*LIST: An expanded source listing is to be generated, showing a detailed list of the file specifications that result from the source statements and references to other file descriptions. This listing shows file and field keywords and attributes.

**NOLIST:* No expanded source listing is to be generated.

DEV Parameter: Specifies the name of the System/38 device to be used with the communications device file to communicate with another system.

*NONE: No device name is to be specified. The name of the communications device must be specified later in the CHGCMNF or OVRCMNF command, or in the HLL program that opens the file.

device-name: Enter the name of the communications device that is to be used with this communications device file. The device name must already be known on the system (via a device description) before this device file is created.

LOGON Parameter: Specifies the text that is to be transmitted to the primary logical unit host when the file is opened. The text is limited to 80 characters, and its format is host-dependent.

*NONE: No logon text is to be specified.

logon-characters: Enter the text that is to be transmitted to the primary logical unit host when this file is opened.

LOGOFF Parameter: Specifies the text that is to be transmitted to the primary logical unit host when the file is closed. The text is limited to 80 characters, and its format is host-dependent.

*NONE: No logon text is to be specified.

logoff-characters: Enter the text that is to be transmitted to the primary logical unit host when this file is closed.

BLKLEN Parameter: Specifies, in bytes, the maximum block length for data that is to be transmitted or received by the device file.

*CALC: The device support chooses an optimum value based on the record sizes in the device file. Device support calculates the smallest multiple of 1792 that is greater than or equal to the largest record in the device file. The calculated value includes the new line (NL) or form feed (FF) characters that follow each record when RCDSEP(*YES) is specified.

block-length: Enter a value, 256 through 32767, that specifies the maximum block length of records to be processed by this communications device file. This value must be at least the size of the largest message expected to be transmitted or received. Also, it must include the new line (NL) or form feed (FF) characters that follow each record when RCDSEP(*YES) is specified.

RCDSEP Parameter: Specifies whether SNA character stream support is to be used to delimit records.

*YES: The system delimits the data records by inserting new line (NL) or form feed (FF) characters between records. The system scans for and removes NL and FF characters during input operations.

**NO:* The system does not insert, scan for, or remove the NL and FF characters in the data records. NL and FF characters, if present, are treated as data characters.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If they cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the communications device file. Valid values are 1 through 32767 (32 767 seconds).

SPAN Parameter: Specifies whether logical records are to be allowed to span request unit boundaries during output operations.

*YES: The system places as much data as possible into a request unit. When this parameter value is specified, a request unit may contain any of the following:

- One or more complete records
- One or more complete records plus a partial record
- A partial record

**NO:* The system places as many complete records as possible into a request unit but will never allow a request unit to contain a partial record.

SHARE Parameter: Specifies whether the ODP (open data path) for the communications device file can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used.

*NO: An ODP created by the program is not to be shared with other programs in the routing step. Every time a program opens the file, a new ODP to the file is created and activated.

**YES:* An ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given an internal system identifier when the format is created.

*YES: The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not all match or they have not been specified in the program, an open error message is sent to the program that attempted to open the file.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

PUBAUT Parameter: Specifies the authority that is being granted to the public (all users) for the communications device file and its description. Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the device file.

**ALL:* The public has complete authority for the device file.

**NONE:* The public cannot use the device file.

TEXT Parameter: Lets the user enter text that briefly describes the communications device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTCMNF  FILE(FILEB.LIBA) SRCFILE(QDDSSRC)
```

This command creates a description of the communications device file named FILEB in library LIBA using the device source file named QDDSSRC. The defaults for all the other parameters are assumed. The device name must be specified in another CL command or in each program that uses the device file.

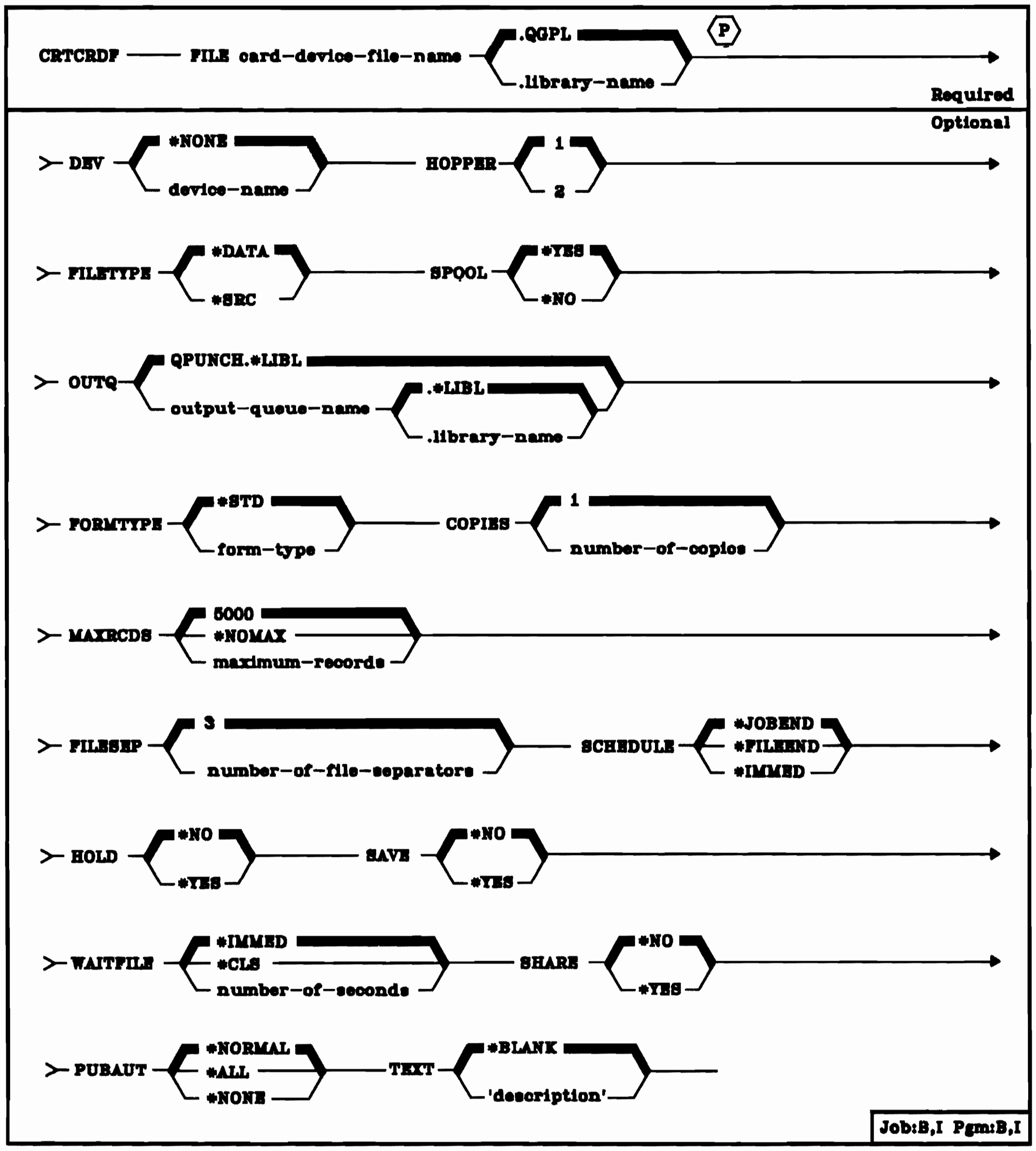
No logon or logoff text is transmitted when data is being sent or received. The block length is to be calculated by device support, and record delimiters are to be inserted. The level identifiers of the record formats used by the communications file are to be checked when the file is opened. The public has only operational rights for the device file.

CRTCRDF (Create Card File) Command

The Create Card File (CRTCRDF) command creates a card device file. The device file contains the file description, which identifies the device to be used and specifies the spooling requirements; the file does not contain data. The card device file is used to get data from a card device (one record per card) and to send data to the card device. The same device file can be used for both input and output operations.

All the information in the card file description is contained in the command that creates it; there is no DDS (data description specifications) for card device files. The card file has only one record format for input/output operations. The record format consists of one character field that contains the input data retrieved from the device or the output data to be written to the device. The program using the device file must describe the fields in the record format so that the program can arrange the data received from or sent to the card device in the manner specified by the card file description.

The CHGCRDF or OVRCRDF command can be used in a program to change or override the parameter values specified in the card file description. Each changed value in the device file remains changed after the program ends. Each overridden value remains altered only for the execution of the program (unless the override is deleted by a DLTOVR command); once the program ends, the original parameter values specified for the card file are used. Override commands must be executed before the card file to be affected is opened for use by the program.



FILE Parameter: Specifies the qualified name by which the card device file being created will be known. If no library qualifier is given, the file is stored in QGPL. (If the file is to be used by an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

DEV Parameter: Specifies the name of the card device that is to be used with this card device file to perform input/output data operations. The device name of the IBM-supplied card device description is QCARD96.

NONE: No device name is to be specified. The name of the card device must be specified later in the CHGCRDF or OVRCRDF command, or in the HLL program that opens the file.

device-name: Enter the name of the device that is to be used with this card device file. The device name must already be known on the system (via a device description) before this device file is created.

HOPPER Parameter: Specifies from which hopper of the MFCU the cards are to be fed when this device file is used. Valid entries are 1 (for the primary hopper) and 2 (for the secondary hopper).

1: The primary hopper is to be used with this card device file.

hopper-number: Enter either a 1 or a 2 to indicate which hopper of the MFCU is to be used.

FILETYPE Parameter: Specifies whether the card device file being created describes data records or describes source records (statements) for a program or another file. (For an expanded description of the FILETYPE parameter, see Appendix A.)

DATA: The card file describes data records.

**SRC:* The card file describes source records.

SPOOL Parameter: Specifies whether the input or output data for the card device file is to be spooled. If SPOOL(*NO) is specified, the following parameters in this command are ignored: OUTQ, FORMTYPE, COPIES, MAXRCDS, FILESEP, SCHEDULE, HOLD, and SAVE.

YES: The data is to be spooled. If this file is opened for input, an inline data file having the specified name is processed; otherwise, the next unnamed inline spooled file is processed. (For a discussion of named and unnamed inline files, see the *CPF Programmer's Guide.*) If this file is opened for output, the data is spooled for processing by a spooling writer.

**NO:* The data is not to be spooled. If this file is opened for card input, the data is read directly from the card device. If this is an output file, the data is sent directly to the device to be punched or printed as the output becomes available.

OUTQ Parameter: Specifies, for spooled output only, the qualified name of the output queue for the spooled output file. (If no library qualifier is given, the queue is found by the library list (*LIBL) that is in effect for the job that uses the card file.)

QPUNCH: The spooled output data is sent to the IBM-supplied output queue, QPUNCH, which is in the QGPL library.

qualified-output-queue-name: Enter the qualified name of the output queue to which the output data is to be spooled. The IBM-supplied output queue that can be used by the card file is the QPUNCH output queue, stored in the QGPL library.

FORMTYPE Parameter: Specifies, for spooled output only, the type of form (cards) on which the card device is to produce the output. The identifiers used to indicate the type of cards are user-defined and must not be longer than 10 characters.

*STD: The standard card type used in your installation is to be used for output from jobs using this card device file.

form-type: Enter the identifier of the card type to be used for output from jobs using this card device file. A maximum of 10 alphameric characters can be specified.

COPIES Parameter: Specifies, for spooled output files only, the number of copies (card decks) of the output to be produced by the card device.

1: Only one copy (card deck) of the output is to be produced.

number-of-copies: Enter a value, 1 through 99, that indicates the number of identical card decks to be produced when this device file is used.

MAXRCDS Parameter: Specifies, for spooled output only, the maximum number of records that can be in the spooled output file for spooled jobs using this card device file.

5000: A maximum of 5000 records can be in the spooled output file for this card device file if the job is to be spooled.

**NOMAX:* No maximum is specified for the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of records that can be in the spooled output file.

FILESEP Parameter: Specifies, for spooled output files only, the number of separator cards to be placed at the beginning of each output card deck, including between multiple copies of the same output. Each separator card contains the file name, file number, job name, user name, job number, and the time and date when the job was executed.

3: Three separator cards are placed at the beginning of each card deck produced by spooled jobs that use this device file.

number-of-file-separators: Enter the number of separator cards to be placed at the beginning of each card deck produced by spooled jobs that use this device file. Valid values are 0 through 9. If 0 is specified, at the end of each output file a message is sent to the message queue (usually QSYSOPR) specified on the STRCRDWTR command that started the writer; the message indicates that the output just produced is to be removed from the device.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a writer.

*JOBEND: The spooled output file is to be made available to the writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the writer as soon as the file is opened in the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The spooled file is made available to a writer when it is released by the Release Spooled File (RLSSPLF) command.

*NO: The spooled output file is not to be held by the output queue. The spooled output is made available to a writer based on the SCHEDULE parameter value.

**YES:* The spooled output file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled file is to be saved (left on the output queue) after the output has been produced.

*NO: The spooled file data is not to be retained on the output queue after it has been produced.

YES: The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

CLS: The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used.

When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

*NO: An ODP created by the program in which this command is used is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

YES: An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

PUBAUT Parameter: Specifies what authority for the card device file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

**NORMAL:* The public has only operational rights for the device file.

**ALL:* The public has complete authority for the device file.

**NONE:* The public cannot use the device file.

TEXT Parameter: Lets the user enter text that briefly describes the card device file. (For an expanded description of the TEXT parameter, see Appendix A.)

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTCRDF  FILE(DSPHST)
```

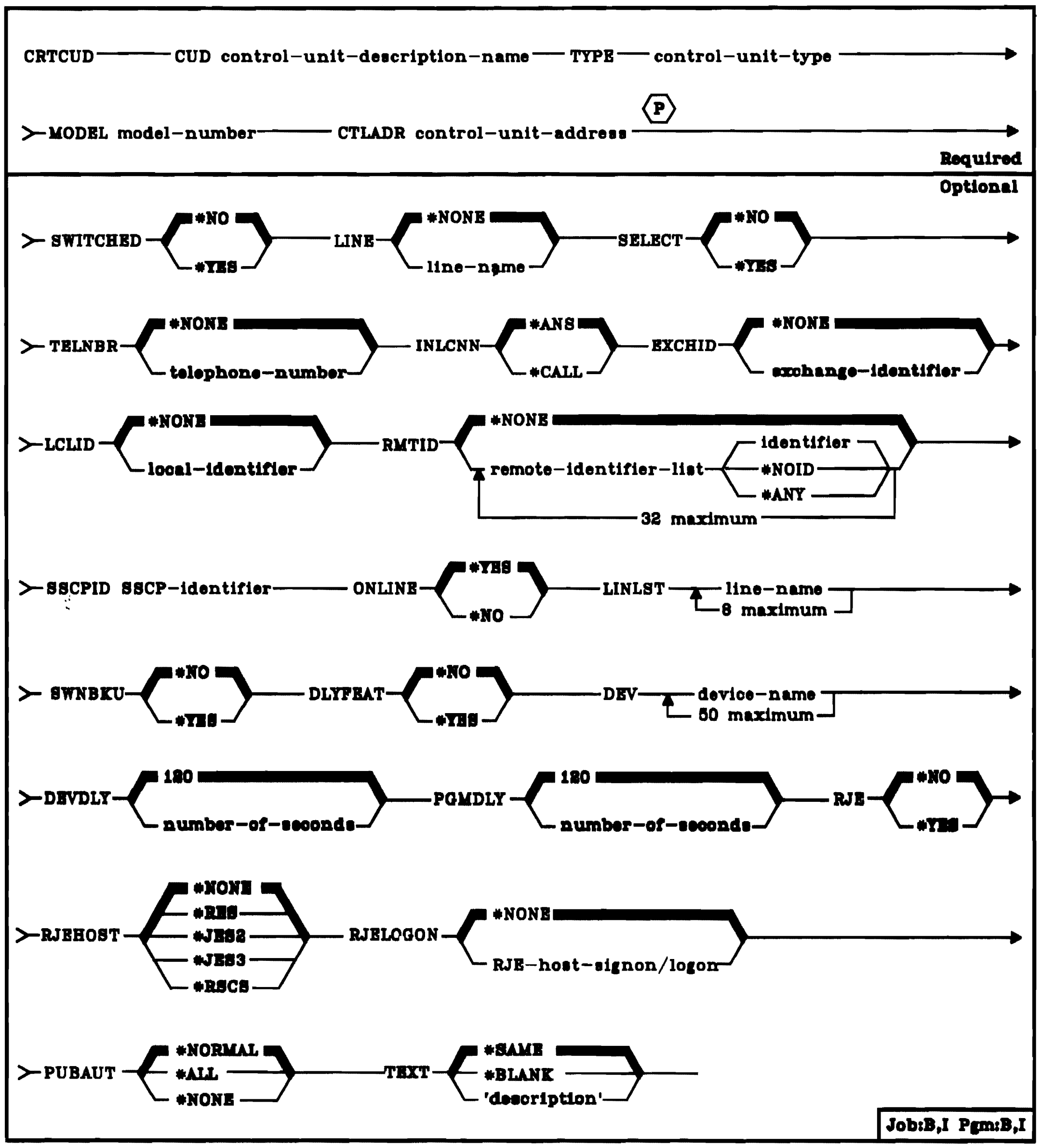
This command creates a description of the card device file named DSPHST. The defaults for all the other parameters are assumed. The device name must be specified in another CL command or in each program that uses the device file. The device file describes card data files that will be spooled for both input and output. Output goes to the QPRINT output queue, and cannot go to the card device until the job is completed on the system. Data cards of the installation's standard type are to be fed from hopper 1 if the device is the MFCU. When output is produced from the output queue, only one copy is produced, and it is preceded by one separator card that gives the file name, job name, and job number.

CRTCUD (Create Control Unit Description) Command

The Create Control Unit Description (CRTCUD) command identifies a control unit and describes its features to the system. The control unit can be a 3411 tape control unit, a 5251 work station control unit attached to a communications line, a work station controller, a BSC control unit, a physical unit (type 2) control unit, or the System/38 itself operating as a multipoint tributary station to an IBM Series 1, System/3, or System/370. When the control unit description is created, it is stored as part of the internal system, and it appears as though it exists in the QSYS (system) library.

This command should be used to create the control unit description *after* the associated line description has been created for the line attached to the control unit; it should also be used *before* the associated device descriptions are created for the devices attached to the line. However, this sequence for creating descriptions is not a required sequence.

Restriction: If the control unit is to be attached to a nonswitched line, that line (identified in the LINE parameter of this command) must first be varied offline.


CRTCUD CUD control-unit-description-name TYPE control-unit-type
MODEL model-number CTLADR control-unit-address
P
Required
Optional
SWITCHED *NO *YES LINE *NONE line-name SELECT *NO *YES
TELNBR *NONE telephone-number INLCNN *ANS *CALL EXCHID *NONE exchange-identifier
LCLID *NONE local-identifier RMTID *NONE remote-identifier-list identifier *NOID *ANY 32 maximum
SSCPID SSCP-identifier ONLINE *YES *NO LINLST line-name 8 maximum
SWNBKU *NO *YES DLYFEAT *NO *YES DEV device-name 50 maximum
DEVDLY 120 number-of-seconds PGMDLY 120 number-of-seconds RJE *NO *YES
RJEHOST *NONE *RES *JES2 *JES3 *RSCS RJELOGON *NONE RJE-host-signon/logon
PUBAUT *NORMAL *ALL *NONE TEXT *SAME *BLANK 'description'
Job:B,I Pgm:B,I

CUD Parameter: Specifies the name of the control unit description that is being created.

TYPE Parameter: Specifies the type of control unit being described. Enter the value shown under *Type* in the table that applies to this control unit.

Type of Control Unit	Type	Model
Tape control unit	3411	1 (for Model 1 3410 tape units) 2 (for Model 2 3410 tape units) 3 (for Model 3 3410 tape units)
Work station control unit	5251	2 (960 characters) 12 (1920 characters)
Work station controller	*WSC	*NONE
Physical unit (type 2)	*PU2	0
BSC device (also for RJEF)	*BSC	0
BSC multipoint tributary	*BSCT	0

MODEL Parameter: Specifies the model number of the control unit. This number indicates to the system the features that the control unit has. (Refer to the table in the description of the MODEL parameter of *CRTDEVD (Create Device Description) command* for the differences in 5251 and 3411 device models.)

For 5251 or 3411 control units, or for physical units (type 2) for SNA, enter one of the values shown under *Model* in the TYPE parameter description. (The model number of the 3411 *must* be the same as the model number of all the 3410 tape units associated with the control unit.) For the work station controller, enter *NONE. For TYPE(*BSC), (*BSCT), or (*PU2), enter MODEL(0).

CTLADR Parameter: Specifies the address of the 5251 or 3411 control unit, of the System/38 work station controller (WSC), of the type-2 physical unit, of the BSC device, or of this System/38 as a multipoint tributary station. (Additional information about the control unit address can be obtained from the *Guide to Program Product Installation and Device Configuration* and the *IBM 5250 Information Display System Planning and Site Preparation Guide,* GA21-9337.)

Enter a four-digit hexadecimal number, consisting of the controller station address (digits 1 and 2) and the operational unit (OU) number of the line or control unit (digits 3 and 4). The following table shows the valid two-digit values used to form the complete address.

Type	**Controller Station Address (Digits 1 & 2)**	**OU Number (Digits 3 & 4)**
Control unit:		
3411	00	15
5251	01-FE	00[1], 20-23, or 60-63
PU2	00	00[1], 20-23, or 60-63
BSC	00	00[1], 20-23, or 60-63
BSCT	01-FE	20-23 or 60-63
Controller:		
WSC 1	00	30
WSC 2	00	70
WSC 3	00	B0
WSC 4	00	F0

[1]00 is used if the control unit is attached to a switched line.
[2]For BSCT, must be the same as the STNADR on the line.

SWITCHED Parameter: Specifies, for 5251, PU2, and BSC control units, whether the remote control unit has a switched line connection. (This parameter does not apply to the 3411 tape control unit, to BSCT, or to the work station controller.)

<u>*NO:</u> The control unit is not attached to a switched line.

**YES:* The control unit is attached to a switched line.

The following chart shows only those parameters in this command that are dependent on the value specified in the SWITCHED parameter. The parameters in the left column can be specified only if SWITCHED(*NO) is also specified; those in the right column are valid only if SWITCHED(*YES) is specified.

SWITCHED(*NO)	**SWITCHED(*YES)**
LINE	
TELNBR[1]	TELNBR
INLCNN[1]	INLCNN
LINLST[1]	LINLST
SWNBKU	LCLID[2]
DLYFEAT	RMTID[2]
RMTID[1]	
LCLID[1]	

[1]Valid only if SWNBKU(*YES) is also specified.
[2]Valid for BSC only.

LINE Parameter: Specifies, for 5251, PU2, BSC, and BSCT control units, the line name of a nonswitched line (if any) that is connected to this remote control unit. (This parameter does not apply to the 3411 tape control unit or to the work station controller.)

*NONE: No nonswitched line is to be attached to the control unit.

line-name: Enter the name of the nonswitched line that is attached to the control unit; the line description must have been created and the associated line must have been varied offline before this command is entered. (The line name must be the same as the name specified in the line description that describes this line.)

SELECT Parameter: Specifies, for 5251, PU2, and BSC control units, whether the modem attached to the remote control unit has the data rate select function or whether it can operate at full speed only. (This parameter does not apply to the 3411 tape control unit, or to the work station controller.)

*NO: The remote modem cannot operate at half speed; it can operate at full speed only.

**YES:* The remote modem has the data rate select function and can operate at either full or half speed.

TELNBR Parameter: Specifies the telephone number of this remote control unit if it is associated with a switched line, or of a nonswitched line if the switched network backup feature is used. The telephone number (1 to 16 digits long) is dialed at the System/38 site to establish a connection with this control unit. (This parameter is required for and valid only for switched lines and for nonswitched lines with SWNBKU(*YES) specified.) The telephone number is:

- Sent to the autocall unit, if automatic calling is used to establish a connection to this control unit
- Displayed to the system operator, if manual calling is used to call this control unit

*NONE: No telephone number is specified for the control unit.

telephone-number: Enter the telephone number that is to be used to call this control unit, using only the digits 0 through 9 and two other special characters: the separator character and the end-of-number character. The separator character is designated by the keyboard's apostrophe symbol, and the end-of-number by the asterisk symbol. Refer to the *IBM System/38 Guide to Program Product Installation and Device Configuration,* GC21-7775 for more information regarding the use of these characters with ACE (autocall equipment).

INLCNN Parameter: Specifies, for remote control units only, the method to be used to make the initial connection over a switched line between System/38 and the control unit. (This parameter applies to switched lines and to control units that have the switched network backup feature activated because ACTSWNBKU(*YES) is specified later on a CHGCUD command.)

*ANS: The initial connection is made by System/38 when it answers an incoming call from this control unit.

**CALL:* The initial connection is made by a call initiated from System/38.

EXCHID Parameter: Specifies, for 5251 remote control units only, the exchange identifier of the control unit. The control unit sends (exchanges) its identifier to another location when a connection is established. Identifiers must be specified for all 5251 control units attached to SDLC lines. The eight-digit hexadecimal identifier contains three digits for the block number and five digits for the identifier of the specific control unit.

*NONE: The control unit has no exchange identifier; it is not a 5251 control unit.

exchange-identifier: Enter the hexadecimal value, eight digits long (using the hexadecimal digits 0 through 9 and A through F) that will identify this control unit to System/38. For the 5251 Model 2 or 12, the value is 020000xx, where 020 is the block number and 000xx is the control unit identifier. The first three digits of the control unit identifier are always zeros and *xx* equals the setting of the Controller Station Address switches on the 5251.

LCLID Parameter: Specifies the local identifier for identifying System/38 to the remote BSC control unit.

*NONE: No local identifier is to be specified.

local-identifier: A string of from 2 to 15 characters for identifying System/38 to a remote BSC control unit. If a two-character identifier is specified, both characters must be the same. The identifier cannot contain BSC control characters.

RMTID Parameter: Specifies a list of identifiers for remote BSC control units. This parameter is valid for switched lines only, and is required if SWITCHED(*YES) or if SWNBKU(*YES) is specified.

*NONE: Specifies that there are to be no remote identifiers. *NONE is valid only for BSC control units with SWITCHED(*NO) and SWNBKU(*NO) specified. This parameter value should not be confused with *NOID, which is a valid remote identifier.

remote-identifier-list: Enter the identifier or a list of identifiers (32 maximum) used by remote BSC control units. If a two-character identifier is specified, both characters must be the same. The identifier cannot contain BSC control characters. *NOID specifies a null identifier; a null identifier can be specified by itself or within a list of identifiers. *ANY instructs System/38 to accept any identifier sent by a remote BSC control unit. If *ANY is specified, it must be the last or only identifier in the list.

SSCPID Parameter: Specifies, if this control unit is to communicate using SNA with a host system, the SSCP (system service control point) identifier of the host system. The SSCP identifier is a 12-digit hexadecimal value, with the first two digits being hexadecimal 05. This parameter is required for and valid for PU2 controllers only.

ONLINE Parameter: Specifies whether the control unit is to be varied online automatically when the Control Program Facility (CPF) is started. After CPF is started, the Vary Control Unit (VRYCTLU) command can be used to modify the status of the control unit.

*YES: The control unit is to be online when CPF is started.

**NO:* The control unit is to be offline when CPF is started. The VRYCTLU command must be used to put the control unit online, making it operational.

LINLST Parameter: Specifies, for switched connections only, a list of line names that identify the lines that can be connected to this control unit. The same line name can be used more than once. This allows the user to add lines later by using the CHGCUD command to replace one or more of the duplicate line names with new line names. If no names are specified, an entry of eight null lines is the default. This parameter is valid only if SWITCHED(*YES) or SWNBKU(*YES) is specified. Also, for each line name specified, a line description by that name must already exist. (This parameter does not apply to the 3411 tape control unit, to the work station controller, or to a BSCT control unit.)

SWNBKU Parameter: Specifies whether a nonswitched modem attached to a remote control unit has the switched network backup feature. The backup feature is used to allow the user to bypass a broken nonswitched connection by manually dialing a telephone number to establish a switched connection. The CHGCUD command must be used to actually activate the feature. (This parameter does not apply to the 3411 tape control unit, to the BSCT control unit, or to the work station controller.) SWNBKU(*YES) is valid only if SWITCHED(*NO) is specified.

*NO: The nonswitched line modem does not have the switched backup feature.

YES: The nonswitched modem does have the switched backup feature. To activate the feature when the nonswitched connection is broken, specify ACTSWNBKU(*YES) on the CHGCUD command.

DLYFEAT Parameter: Specifies, for nonswitched lines only, whether periodic attempts should be made to contact this control unit (to establish a delayed connection) if the initial attempt to establish a connection is not successful. (This parameter is valid only for 5251 work station control units.)

*NO: Only one attempt is to be made to establish a connection between the line and the control unit.

YES: Periodic attempts are to be made to establish a delayed connection between the line and the control unit.

DEV Parameter: Specifies the names of one or more devices to be attached to this control unit. Each device name must be the same as that specified when the associated device description was created.

The following table describes the maximum number of devices that can be attached to the various types of control units:

Control Unit Type	Maximum Number of Devices
*WSC	20
*PU2	50
5251	9
3411	4
*BSC	24[1]
*BSCT	32

[1]Maximum of one Model 0 BSC device and 23 Model 1 BSC devices.

Enter the name of each device to be attached to the control unit.

Do not use this parameter when following the normal procedure of creating the descriptions for lines first, control units second, and devices last (using the CRTLIND, CRTCUD, and CRTDEVD commands). Use this parameter only when the associated device descriptions have already been created before this control unit description.

DEVDLY Parameter: Specifies, for BSC and BSCT only, the number of seconds System/38 will wait while receiving WACK (wait before transmit positive acknowledgment) or TTD (temporary text delay) sequences from the remote device before time-out occurs.

*120: The system will wait for a delay of 120 seconds before time-out occurs.

number-of-seconds: The number of seconds the control unit will wait before time-out occurs.

PGMDLY Parameter: Specifies, for BSC and BSCT only, the number of seconds System/38 will send WACK or TTD sequences to the remote device because of delays by the System/38 application in issuing READ or WRITE requests.

*120: The system will send delay signals for 120 seconds before time-out occurs.

number-of-seconds: The number of seconds the control unit will continue to send delay signals before time-out occurs.

RJE Parameter: Specifies, for BSC only, whether this control unit description is to be used by the Remote Job Entry Facility (RJEF).

*NO: This control unit description is not to be used by RJEF.

**YES:* This control unit description is to be used by RJEF. If RJE(*YES) is specified with SWITCHED(*YES), at least one remote identifier must be specified with the RMTID parameter.

RJEHOST Parameter: Specifies, for BSC only, the subsystem type of the host to which which RJEF is connected.

*NONE: No RJEF host subsystem type is to be specified.

**RES:* RJEF is connected to a VS1/RES subsystem.

**JES2:* RJEF is connected to a VS2/JES2 subsystem.

**JES3:* RJEF is connected to a VS2/JES3 subsystem.

**RSCS:* RJEF is connected to a VM/370 RSCS subsystem.

RJELOGON Parameter: Specifies, for BSC only, logon information for the RJEF host system.

*NONE: No logon information is to be specified; the control unit is not to be used for RJEF.

'RJE-host-signon/logon': Enter up to 80 characters of text enclosed in apostrophes to be used as signon/logon information for the RJEF host system.

PUBAUT Parameter: Specifies what authority for the control unit and its description is being granted to the public. Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

Note: *NORMAL should be specified so that users who are authorized to use work stations attached to this control unit are not hindered from doing so because they might not also have been given explicit authority for the control unit.

*NORMAL: The public has only operational rights for the control unit.

**ALL:* The public has complete authority for the control unit.

**NONE:* The public cannot use the control unit.

TEXT Parameter: Lets the user enter text that briefly describes the control unit and its location. (For an expanded description of the TEXT parameter, see Appendix A.)

<u>*BLANK:</u> No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

Control unit description work sheets are provided at the back of the *Guide to Program Product Installation and Device Configuration* that you can use to collect the information needed before creating the control unit descriptions. Refer to that publication for information about device configuration, system installation procedures, and how to use the work sheets.

```
CRTCUD  CUD(WSC1) TYPE(*WSC) MODEL(*NONE) +
     CTLADR(0030) PUBAUT(*NORMAL) +
     TEXT('Work station controller 1')
```

This command creates a description for a work station controller (*WSC). Because it is the basic work station controller, the address is 0030. Normal public authority is granted for the control unit description.

```
CRTCUD  CUD(NYC1) TYPE(5251) MODEL(12) +
     CTLADR(0100) SWITCHED(*YES) +
     SELECT(*YES) TELNBR(2866894) +
     INLCNN(*ANS) EXCHID(02000001) LINLST(NYC) +
     TEXT('NYC sales branch 1, Room 308')
```

This command creates a description for a control unit named NYC1. The control unit is a 5251 Model 12 and is at address 0100. The control unit is on a switched line at telephone number 286-6894. Connection to the control unit is initiated by an incoming call to System/38.

```
CRTCUD  CUD(S370CU) TYPE(*PU2) MODEL(0) +
     CTLADR(0020) SSCPID(050000000080) LINE(SECLINE)
```

This command creates a description for a control unit named S370CU that enables System/38 to function as a secondary communications station. The control unit is a physical unit (type 2)—which is a host system. The SSCP identifier of the host system is 050000000080. The address of the control unit is 0020, where 00 is the controller station address and 20 is the OU number of the line description SECLINE.

CRTDEVD (Create Device Description) Command

The Create Device Description (CRTDEVD) command creates a device description for the specified device and describes all of the features of the device to the system. The device description is stored as part of the internal system and appears as though it exists in the QSYS (system) library.

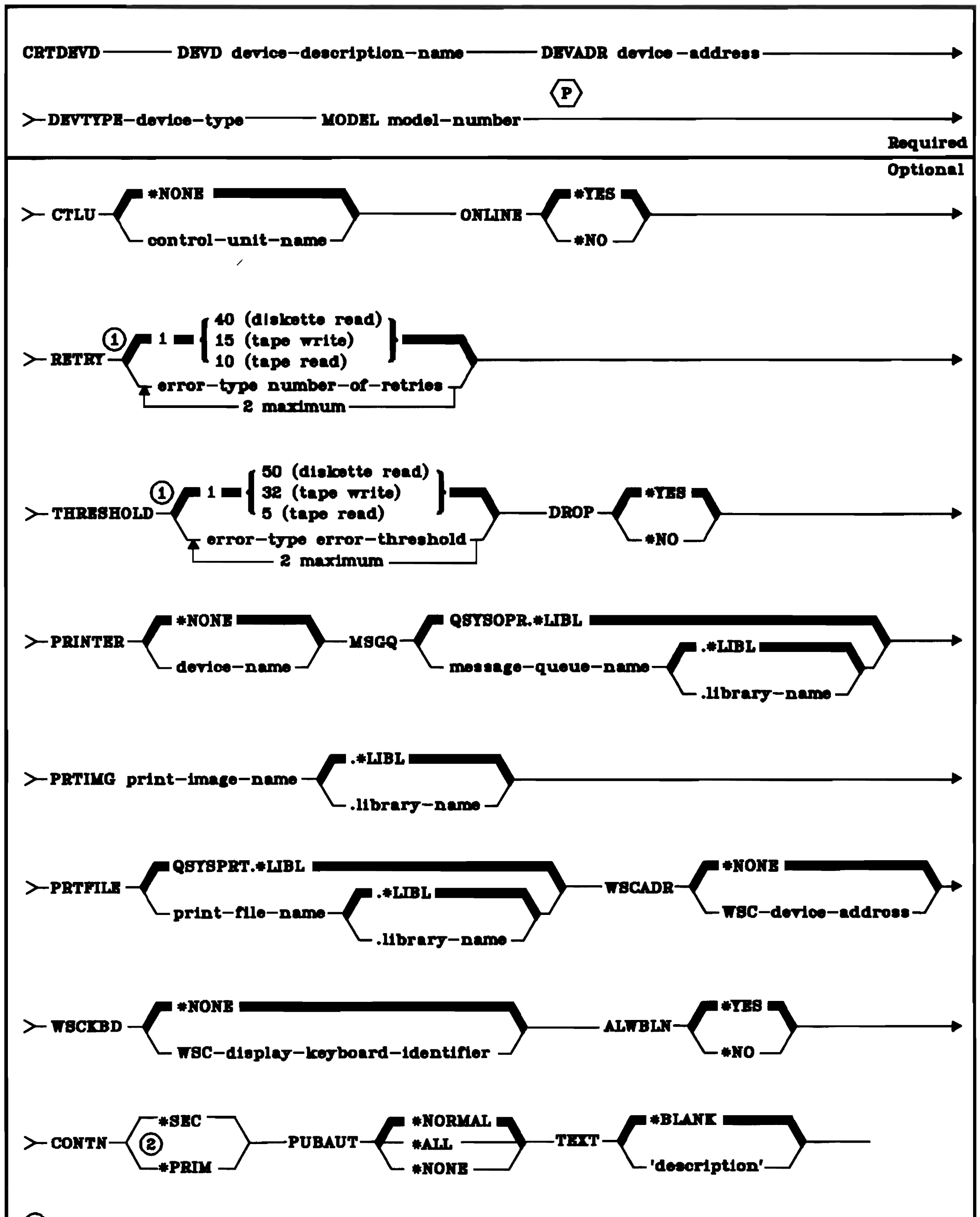
Each device attached to the system must have a device description before the system can use the device. Some device descriptions, such as QDKT and QCONSOLE, are predefined by IBM and are shipped with the system.

For devices that require a line description and/or control unit description, this command should be used to create each device description after the associated line description and control unit description are created for the line and control unit associated with the device. Also, the device description for each remote printer should be created before it can be referenced by the device description for its associated display station. If the descriptions are created out of sequence, the system rejects any commands referring to names of descriptions not yet created.

Restrictions: (1) If the device is to be attached to a control unit, the control unit must first be varied offline. (2) If the CRTDEVD command is used to change the name and/or address of a work station printer, the CHGDEVD command must be used to incorporate the new attribute(s) in the device description of each display station associated with the printer. (If the address of the printer is changed, the CHGDEVD command makes the new address association when the new (or unchanged) printer name is specified in the PRINTER parameter.) (3) Only one device description may be created per BSC line or controller. (4) No more than 50 devices can be assigned per control unit.

The following table shows the maximum number of devices that can be attached to a single control unit of a specified type:

Control Unit Type	Maximum Number of Devices
*BSC	1
3411	4
5251	9
*WSC	20
*PU2	50



(1) Applies to diskette and tape devices only. The default values for RETRY are 1 and 40 for diskette, and 1 and 20 for tape.
(2) *SEC is the default for DEVTYPE(*BSC). *PRIM is the default for DEVTYPE(*BSCT).

Job:B,I Pgm:B,I

DEVD Parameter: Specifies the name of the device description that is being created. The name of an existing device description cannot be specified. For example, QCONSOLE cannot be used, because it is already used as the name of the system console.

DEVADR Parameter: Specifies a six-digit hexadecimal number that identifies the physical address of the device. Additional information about this address can be obtained from the *Guide to Program Product Installation and Device Configuration* and the *IBM 5250 Information Display System Planning and Site Preparation Guide*, GA21-9337.

For work stations attached to the work station controller (locally attached) and for switched BSC devices, this address must be 000000. Note that the actual work station device address is specified in the WSCADR parameter, and it is the value displayed by the DSPDEVD command as the device address.

The physical address of the device contains a combination of three values:

- Unit (device) address. Digits 1 and 2 must specify:
  - 00, if the device is directly attached.
  - The unit address, if the device is attached to a control unit, a communication line, or both.
  - The logical unit address used by the host system (contained in the SNA destination address field), to address System/38, if the device and SNA are used by System/38 to communicate with a host system.

- Controller station address. Digits 3 and 4 must specify:
  - 00, if the device is not attached to *both* a control unit and a communications line.
  - The controller station address, if the device is attached to *both* a control unit and a communications line. (For example, a 5251 Model 11 attached to a 5251 Model 12 control unit.)

- OU number (for non-work-station devices). Digits 5 and 6 must specify:
  - 00, if the device is attached to a line and a control unit, when a switched line connection is used.
  - The line OU (operational unit) number, if the device is attached to a nonswitched line and a control unit.
  - The device OU number, if the device is directly attached.
  - The control unit OU number, if the device is attached to a control unit only.

Enter the appropriate values that specify the correct configuration and addresses. The following chart shows the possible values for this parameter:

Device	Unit Address (Digits 1 & 2)	Controller Station Address (Digits 3 & 4)	OU Number (Digits 5 & 6)
BSC	For BSC devices 00	00	00, or 20-23, 60-63
	For RJE devices: Console input 01 Console output 02 Reader 1 11 Reader 2 12 Reader 3 13 Printer 1 21 Printer 2 22 Printer 3 23 Punch 1 31 Punch 2 32 Punch 3 33	00	00, or 20-23, 60-63
BSCT	00[1]	01-FE	20-23 or 60-63
PLU1	00-FF	00	00, or 20-23, 60-63
Console	00	00	02
Diskette magazine drive	00	00	12
MFCU	00	00	19
First system printer			
3262 or 5211	00	00	18
3203	00	00	40
Second system printer			
3262 or 5211	00	00	58
3203	00	00	40 or 41[2]
Tape unit	00-03	00	15
Remote work station	00, or 02-09[3]	01-FE	00, or 20-23, 60-63
WSC work station (see WSCADR parameter)	00[4]	00[4]	00[4]

[1]Any hexadecimal digits can be specified for the BSCT unit address, except for hex FE, 7F, or BSC control characters (control character hex 2D may be specified).

[2]If only one 3203 is installed on the system, its OU number is always 40, regardless of whether it is installed as the first or second system printer. If two 3203s are on the system, the OU number of the second 3203 is 41.

[3]Any 5251 Model 2 or 12 control unit has a unit address of 00. Any cluster-attached work station has a unit address of 02-05 (if part of the first cluster) or 06-09 (if part of the second cluster).

[4]For work stations attached to a work station controller, the DEVADR parameter must have all zeros; the actual address of the connected device is specified in the WSCADR parameter.

DEVTYPE Parameter: Specifies the type code for this device. Enter one of the following four-character type codes that describes this type of device:

Type Code	Device Name	Type Code	Device Name
3203	Printer (system)	5252	Dual Display Station
3262	Printer (system)	5256	Printer (work station)
3410	Magnetic Tape Unit	5424	Multi-Function Card Unit
5211	Printer (system)	72MD	Diskette magazine drive
5224	Printer (work station)	PLU1	Primary logical unit, type 1 (for SNA)
5225	Printer (work station)	*BSC	All BSC-supported IBM equipment including RJEF
5251	Display Station	*BSCT	This System/38 as a BSC multipoint tributary station

MODEL Parameter: Specifies the model number of the device. This number indicates to the system the operational capabilities of the device. Enter one of the following model numbers (containing 1 to 4 characters) that matches the device.

Device Type	**Description**	**Model Number**
3203	Printer (system)	5 (1200 lines per minute)
3262	Printer (system)	A1,B1 (650 lines per minute) (see Note 1)
3410	Tape unit	1, 2, or 3 (see Notes 2 and 3)
5211	Printer (system)	2 (300 lines per minute)
5224	Printer (work station)	1 (137 lines per minute) 2 (240 lines per minute)
5225	Printer (work station)	1 (280 lines per minute) 2 (400 lines per minute) 3 (490 lines per minute) 4 (560 lines per minute)
5251	Display Station	1 (960 characters) 11 (1920 characters)
5252	Dual Display Station	1 (960 characters each)
5256	Printer (work station)	1 (40 characters per second) 2 (80 characters per second) 3 (120 characters per second)
5424	Multi-Function Card Unit	A1, A2, K1, K2, or K3 (see Note 4)
72MD	Diskette magazine drive	1001
PLU1	Primary logical unit, type 1 (for SNA)	0
BSC	All devices	0
BSC/RJE	All devices	1
BSCT	All devices	0

Notes:

1. Two 3262 Model A1 Printers cannot be attached to System/38. Also if a 3262 Model B1, a 5211, or a 3203 is already attached and a 3262 Model A1 is to be added, it must be installed as the first printer and the device address of the original printer must be changed to that of a second printer.
2. All 3410 tape units associated with a 3411 tape control unit must have the same model number as that of the control unit.

3. The following are the characteristics of the 3410 models:

3410 Characteristics	**Model 1**	**Model 2**	**Model 3**
Tape speed (in inches and millimeters per second)	12.5 in 317.5 mm	25 in 635 mm	50 in 1270 mm
Data rate:			
1600 bits/inch (63 bytes/mm) (standard rate)	20 kb/sec	40 kb/sec	80 kb/sec
800 bits/inch (37.5 bytes/mm)	10 kb/sec	20 kb/sec	40 kb/sec
Read access time (in milliseconds)	15 ms	12 ms	6 ms

4. The following are the characteristics of the 5424 MFCU:

5424 Characteristics **Speed (in cards per minute) for:**	**Models A1, K1, and K2**	**Models A2 and K3**
Read	250	500
Punch (print or punch/print 3 lines)	60	120
Print (4 lines)	48	96

CTLU Parameter: Specifies the name of the control unit to which the device is attached. The control unit name must be the same as the name specified in the control unit description. This parameter is valid only if this is the device description of a 5251 or 5252 display station, a 5224/5225/5256 Printer, a 3410 tape unit, secondary logical unit, or a BSC or BSCT device.

*NONE: The device is not attached to a control unit.

control-unit-name: Enter the name of the control unit (which must be varied offline before this command is executed) to which this display, printer, tape, PLU1 or BSC device is attached.

ONLINE Parameter: Specifies whether the device is to be varied online automatically when the Control Program Facility (CPF) is started. After CPF is started, the Vary Device (VRYDEV) command can be used to modify the status of the device.

<u>*YES:</u> The device is to be online when CPF is started.

**NO:* The device is to be offline when CPF is started. The VRYDEV command must be used to put the device online, making it operational.

RETRY Parameter: Specifies, for diskette and tape data errors only, the number of times the system should attempt to recover from a data error when data is read or written. The system operator is notified if the device cannot recover from the data error in the specified number of retries.

If this parameter is specified, the error type and retry values must *both* be specified. If this parameter is not specified, the default error type is <u>1</u> (for diskette or tape read errors) and the default value for the number of retries is one of those shown in the following chart:

Error Type	Applicable Device	Number of Retries		Error Threshold	
		Range	Default	Range	Default
1 (read error)	Diskette	40-80	40	1-100	50
	Tape	10-20	10	1-10	5
2 (write error)	Tape	15-30	15	1-64	32

error-type number-of-retries: Enter the type code followed by the maximum number of retries that the system can have to recover from the specified device data error.

THRESHOLD Parameter: Specifies, for diskette and tape data errors only, the error threshold value that is to be used to determine when an entry should be written to the error log to indicate data errors. The first occurrence of the error is always logged. This parameter is used to specify the number of times the error can occur before the error is logged again. For example, if the threshold for tape read errors was set to five, and 10 errors have occurred, the error would have been logged three times (on the first, fifth, and tenth errors).

error-type error-threshold: Enter the error type code followed by a valid error threshold value, after which the same error message is repeated in the error log. The values that are valid for each error type (and the default values) are shown in the RETRY parameter chart. Both values must be entered for each type of data error being specified.

DROP Parameter: Specifies, only for 5251 and 5252 devices attached to a control unit that is on a switched line, whether the line is to be disconnected by the system when all work stations on the line are no longer being used. When multiple work stations are attached to the same control unit, the line is disconnected only if: (1) the device description for this device specifies DROP(*YES) or DROP(*YES) is specified on the SIGNOFF command when the user signs off at the device; (2) all of the other display stations connected to the control unit have signed off and are not in use; and (3) all 5224/5225/5256 Printers attached to the control unit are not in use.

The value specified in the device description can be overridden by a user signing off at the device if he specifies the DROP parameter on the SIGNOFF command.

*YES: The switched line to the control unit to which this device is attached is to be disconnected when this device and all the other attached devices are no longer in use.

_*NO:_ The switched line is not to be disconnected from the control unit when all of its attached devices are no longer in use.

PRINTER Parameter: This parameter is valid only when this CRTDEVD command is used to describe a 5251 or 5252 Display Station. It specifies the device name of the 5224/5225/5256 Printer to be associated with the display device. The device description of the work station printer named in this parameter must have already been created in another CRTDEVD command and must currently exist on the system. Both the printer and display must be attached to the same control unit. The relationship created by this parameter is used when a related printer (PRINT keyword in DDS) is referred to in a device file used to access this work station.

Note: A printer attached to a remote work station must have the Expanded Function feature to support this parameter's function.

*NONE: No printer is to be associated with this display.

device-name: Enter the name of the printer to be associated with this 5251 or 5252 display.

MSGQ Parameter: Specifies, for 5224/5225/5256 Printers only, the message queue to which operational messages for this device are to be sent.

QSYSOPR: Messages are to be sent to the QSYSOPR message queue.

qualified-message-queue-name: Enter the qualified name of the message queue to which operational messages are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

PRTIMG Parameter: Specifies, for system printer device descriptions only, the qualified name of the print image that is to be the standard print image for the 3203, 3262, or 5211 Printer. (If no library qualifier is given, *LIBL is used to find the print image.)

PRTFILE Parameter: Specifies an alternate print file to be used when no associated work station printer exists or when an error occurs during an attempt to use the work station printer.

QSYSPRT: The print processing will be performed by the system printer device file.

qualified-print-file-name: Enter the name of the printer device file that is to perform default system printing. (If no library qualifier is specified, *LIBL is used to locate the device file.)

WSCADR Parameter: Specifies the address of a device that is attached to a work station controller (WSC). This address must be specified only when the device being described is attached to a WSC. For remote work stations, this address must be 00 00 00. Additional information about the first two parts of this address can be obtained from the *Guide to Program Product Installation and Device Configuration.*

The address specified in this parameter is made up of six digits (xxyyzz), as follows:

- xx (00-19): Specifies the unit address assigned to the device by the customer. The unit address of each device attached to a WSC must be unique. The devices attached to a WSC should be numbered consecutively in ascending sequence.

- yy (00-63): Specifies the number of the WSC connector (identified on the WSC connector panel at the rear of the System/38 system unit) to which this device is connected. The valid values are 00-15 for WSC1, 16-31 for WSC2, 32-47 for WSC3, and 48-63 for WSC4.

- zz (00-06): Specifies the work station address established by the switch settings of the address switches on the device. Each work station address must be unique among the devices attached to the WSC via a particular *WSC port*. The 5252 Dual Display Station is recognized as two work stations; therefore, the primary work station address will be an even number (such as 00 or 02), and the secondary address will default to the next consecutive odd number (such as 01 or 03). For more information about the work station address, refer to the description of the address switches in the *IBM 5250 Information Display System Planning and Site Preparation Guide*, GA21-9337.

*NONE: The device is not attached to a work station controller.

work-station-controller-device-address: Enter the six-digit device address in the format xxyyzz.

WSCKBD Parameter: Specifies, for display work stations, the type of keyboard on the device. This parameter is used only for display devices that are attached to the work station controller (WSC). The identifier specified consists of 4 characters (yzzz), as follows:

- y (T, D, or P): Specifies a typewriter keyboard (T), a data entry keyboard without a Proof Feature (D), or a data entry keyboard with the Proof Feature (P).
- zzz: Specifies a character combination (from the table of keyboard identifiers shown later) to identify the keyboard. The last character indicates whether the character set is the basic set (B) or multinational set (I, for international).

For example, WSCKBD(TUSB) indicates a typewriter keyboard using the basic United States character set.

The maximum number of devices that can be supported on one WSC is dependent on the number of different keyboard types used with the display devices attached to that WSC. The following chart shows the maximum number of devices (which includes both work station displays and printers) that can be supported on one WSC for a given number of keyboard types used by those devices.

Number of Keyboard Types	Maximum Number of Devices Allowed	Number of Keyboard Types	Maximum Number of Devices Allowed
1-2	20	11-12	15
3-4	19	13-14	14
5-6	18	15-16	13
7-8	17	17-18	12
9-10	16	19-20	11

Data entry keyboards with and without the proof feature (P and D) that are in the same language group are considered to be the same keyboard type. (For example, PUSB and DUSB are considered one type.)

If the device maximum is exceeded, then when the VRYCTLU command is used to vary on the control unit, an error message is sent to the system operator.

*NONE: The device being described in this command is not a display work station or is a display work station without a keyboard attached.

WSC-display-keyboard-identifier: Enter the four-character identifier that specifies the type of keyboard and the language group to be used with the work station display.

Country	Keyboard Identifiers	
	Basic (96-Character Set)	Multinational (188-Character Set)
Austria/Germany	AGB	AGI
Belgium	BLB	BLI
Brazil	BRB	BRI
Canada (French)	CAB	CAI
Denmark	DMB	DMI
Finland	FNB	FNI
France (Azerty)	FAB	FAI
France (Qwerty)	FQB	FQI
International	INB	INI
Italy	ITB	ITI
Japan (English)	JEB	JEI
Japan (Katakana)	KAB	
Norway	NWB	NWI
Portugal	PRB	PRI
Spain	SPB	SPI
Spanish Speaking	SSB	SSI
Sweden	SWB[1]	SWI[1]
United Kingdom	UKB	UKI
United States	USB	USI
United States ASCII	UAB[2]	UAI[2]

[1]Typewriter and data entry with proof feature keyboards only.
[2]Typewriter keyboard only.

ALWBLN Parameter: Allows users to suppress the (software-controlled) blinking cursor.

<u>*YES:</u> Allows the cursor to blink for the 5250 display devices.

**NO:* The blinking cursor is to be suppressed.

CONTN Parameter: Specifies which BSC station is primary and which is secondary, in order to resolve contention for BSC point-to-point and multipoint lines.

**SEC:* Specifies the local System/38 as the secondary station, which will yield to the other station when line contention occurs. *SEC is the default for DEVTYPE(*BSC).

**PRIM:* Specifies the local System/38 as the primary station. *PRIM is the default for DEVTYPE(*BSCT).

PUBAUT Parameter: Specifies what authority for the device and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

<u>*NORMAL:</u> The public has only operational rights for the device.

**ALL:* The public has complete authority for the device.

**NONE:* The public cannot use the device.

TEXT Parameter: Lets the user enter text that briefly describes the device and its location. (For an expanded description of the TEXT parameter, see Appendix A.)

<u>*BLANK:</u> No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

Various device description work sheets are provided at the back of the *Guide to Program Product Installation and Device Configuration* that you can use to collect the information needed before creating the device descriptions. Refer to that publication for information about device configuration, system installation procedures, and how to use the work sheets.

```
CRTDEVD  DEVD(DP1) DEVADR(000000) DEVTYPE(5251) +
     MODEL(11) CTLU(WSC1) PRINTER(DP2) +
     WSCADR(000001) WSCKBD(PUSB) +
     PUBAUT(*NONE) TEXT('Programmer''s +
     display work station – Dept 522')
```

This command creates a description for a 5251 Display Station named DP1, which is attached to the work station controller. The work station has the US Basic (96-character set) keyboard with the proof feature included with the data entry keyboard. A work station printer (named DP2) is associated with this display work station. No public authority is granted to this device description and device.

```
CRTDEVD  DEVD(NYC2) DEVADR(000120) DEVTYPE(5251) +
     MODEL(11) CTLU(NYC1) PRINTER(NYC3) +
     TEXT('NYC sales Br 1 display work station')
```

This command creates a description for a 5251 Display Station named NYC2, which is attached to the remote control unit NYC1. A work station printer (named NYC3) is associated with this display work station.

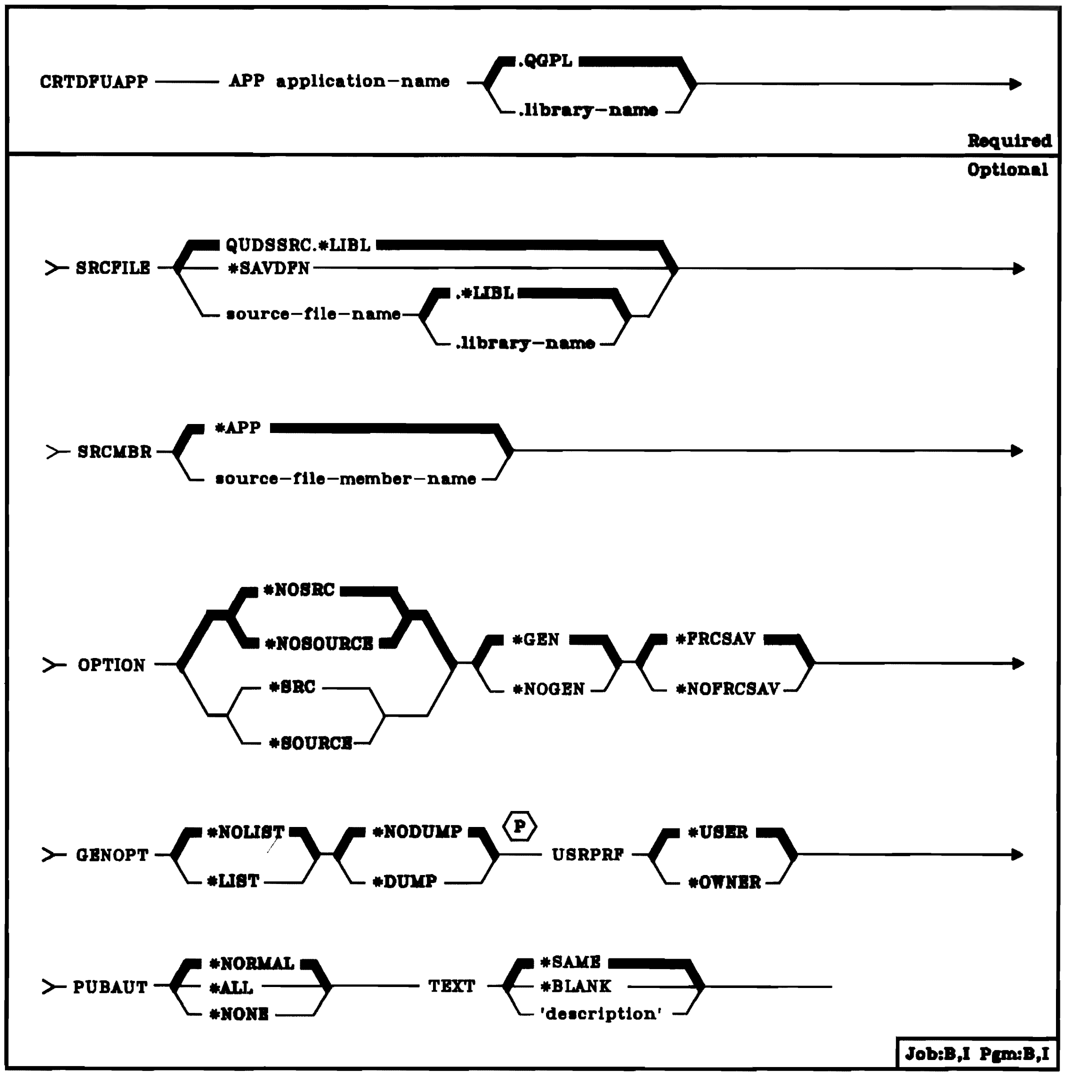
```
CRTDEVD  DEVD(PRTR1) DEVADR(000040) DEVTYPE(3203) +
     MODEL(5) PRTIMG(HN)
```

This command creates a description for a 3203 Model 5 Printer. The device description is named PRTR1, and it has a device address of 000040, indicating that it is the first 3203 Printer attached to the system. The standard print image for PRTR1 is to be HN.

CRTDFUAPP (Create DFU Application) Command

The Create DFU Application (CRTDFUAPP) command creates an executable DFU application from utility definition source statements or from an existing definition.

The Data File Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Data File Utility, refer to the *IBM System/38 DFU Utility Reference Manual and User's Guide,* SC21-7714.



APP Parameter: Specifies the name of the application you are creating and specifies the library in which it is to be stored. (If no library name is given, the application is stored in the general-purpose library QGPL.) The application name must be unique in the library where it is stored. No program or file in the library can have the same name.

SRCFILE Parameter: Specifies the application or the name of the source file that contains the definition of the application. (If no library qualifier is specified, the library list *LIBL is used to find the file.)

QUDSSRC: When IDU is distributed by IBM the QUDSSRC source file is provided in the library QIDU.

**SAVDFN:* The definition of the application is saved in the application specified in the APP parameter, rather than in a source file. If *SAVDFN is specified, the Retrieve DFU Application (RTVDFUAPP) command is not used.

source-file-name: An existing source file other than the provided QUDSSRC.

Note: The CRTDFUAPP command ignores overrides to source files that contain UDS statements.

SRCMBR Parameter: Specifies the name of the source member that contains the definition of the application.

*APP: The definition of the application is in a source member that has the same name as the name specified in the APP parameter.

source-file-member-name: The definition of the application is in a source member that has a name that is different from the name in the APP parameter.

OPTION Parameter: Specifies whether a listing of the source UDS is printed; specifies whether an executable application is actually created, or whether the source UDS is only checked for errors; specifies whether service information is to be printed. Select one value from each of the following groups: *SOURCE and *NOSOURCE; *GEN and *NOGEN; *NODUMP, *DUMP, and *EXCDUMP; *NOTRACE and *TRACE.

*NOSOURCE or *NOSRC: The *NOSOURCE and *NOSRC values are equivalent. When you specify *NOSOURCE or *NOSRC, DFU does not print a listing of the source UDS; however, DFU does print a listing of errors found in source UDS.

**SOURCE* or **SRC:* The *SOURCE and *SRC values are equivalent. When you specify *SOURCE or *SRC, DFU prints a listing of the source UDS.

*GEN: Create an executable application.

**NOGEN:* Do not create an executable application; only perform error checking.

<u>*FRCSAV:</u> Specifies that the UDS (possibly in a nonexecutable application) is to be saved, regardless of whether the application was created successfully. If *FRCSAV is not specified, the UDS is not saved if the application fails to create.

**NOFRCSAV:* Specifies that the UDS will not be saved if the application fails to be created.

GENOPT Parameter: Specifies the printing of IDU program listings created for your application. The listings may be required if a problem occurs in IDU.

USRPRF Parameter: Specifies under which user profile the application is to be executed.

<u>*USER:</u> The user profile for the application user is in effect when the application is executed.

**OWNER:* The user profiles of both the application owner and the application user are in effect when the application is executed.

To execute a DFU application, the user must be authorized to the CHGDTA and DSPDTA commands, the generated application (file and program objects), the installed DFU device files (QDTALOG, QDTAPRT, and QDFUSVCF), the data base file associated with the application, and any libraries that contain these objects. Authority to most of these objects is granted to all users unless restricted by your installation. Normally, you will only need to consider the user's authority to the application and associated data base file.

PUBAUT Parameter: Specifies what authority over the application is extended to all system users. (For an expanded description of the PUBAUT parameter, see Appendix A.)

<u>*NORMAL:</u> All system users can execute the application, but all users cannot change the application.

**ALL:* All system users have complete authority over the application.

**NONE:* All users but the owner are restricted from the application. The owner can subsequently grant some or all rights to some or all other users. Because a DFU application consists of two objects (FILE and PGM), each having the same name assigned to the application, you must issue two commands to grant authority to the application.

TEXT Parameter: Lets you specify a description of the application.

*SAME: Copy the description from the original definition.

**BLANK:* There is no description of this application.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

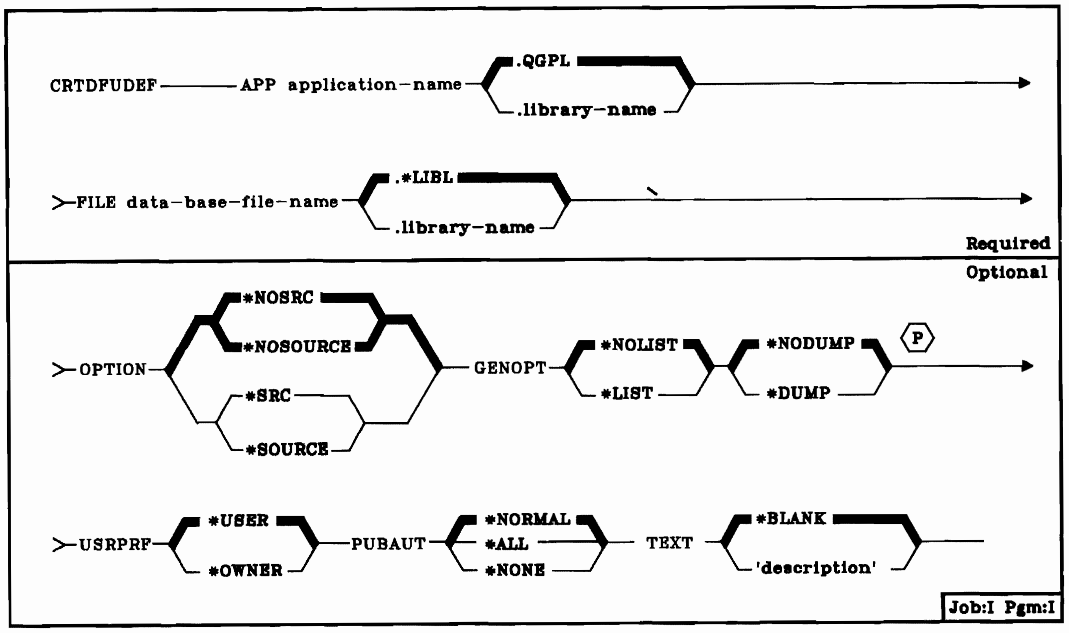
```
CRTDFUAPP  APP(TEST1) SRCFILE(FILE1) SRCMBR(TEST2) +
     TEXT('Test application for TEST1')
```

This command creates an application named TEST1 using source member TEST2, which resides in source file FILE1.

CRTDFUDEF (Create DFU Definition) Command

The Create DFU Definition (CRTDFUDEF) command begins the prompting sequence for interactive definition of a DFU application. Your responses to the prompts are used to create a DFU application.

The Data File Utility is part of the IBM System/38 Interactive Data Base Utilities Program Licensed Program Product, Program 5714-UT1. For more information on the Data File Utility, refer to the *IBM System/38 DFU Utility Reference Manual and User's Guide*, SC21-7714.



APP Parameter: Specifies the qualified name of the application being defined and the library in which it is to be stored. (If no library name is given, the application is stored in the general-purpose library QGPL.)

FILE Parameter: Specifies the name of an existing data base file with record formats that will be referred to by the application you are defining. The file is defined by DDS (see the *CPF Reference Manual—DDS*). (If no library qualifier is specified, *LIBL is used to find the file.)

OPTION Parameter: Specifies whether a listing of the UDS (utility definition source) statements is to be printed, which may be helpful if problems occur.

NOSRC or *NOSOURCE:* Specifies that DFU is not to print a listing of the UDS. The *NOSRC and *NOSOURCE values are equivalent.

**SRC* or **SOURCE:* Specifies that DFU is to print a listing of the UDS. The *SRC and *SOURCE values are equivalent.

GENOPT Parameter: Specifies whether the IDU program listings for your application are to be produced. These listings may be helpful if a problem occurs.

*NOLIST: Specifies that an internal representation of the application program is not to be printed.

**LIST:* Specifies that an internal representation of the application program is to be printed.

*NODUMP: Specifies that the application program template is not to be printed.

**DUMP:* Specifies that the application program template is to be printed. *DUMP should be specified only if *LIST has been specified.

USRPRF Parameter: Specifies a user profile under which the application is to be executed. This parameter allows a programmer to define a DFU application for someone who does not have full authority over the data base file that the application reads.

*USER: The user profile of the application user is in effect when the application is executed.

**OWNER:* The user profiles of both the application owner and the application user are in effect when the application is executed.

When you create an application that is to be used by someone else, you must authorize the user for the use of the application and any objects associated with the application. You can grant each user specific rights to such objects. By specifying USRPRF(*OWNER) when an application is created, you can permit a user to temporarily assume your authority to use objects associated with the application.

PUBAUT Parameter: Specifies what authority over the application is extended to all system users. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: All system users can execute or read the application, but not all users can delete the application.

**ALL:* All system users have complete authority over the application.

**NONE:* All users but the owner are restricted from using the application. Of course, the owner can grant rights to other users.

TEXT Parameter: Enter a brief description of the application.

*BLANK: There is to be no description of the application.

'description': Enter no more than 50 characters, enclosed in apostrophes, to describe the application.

Example

```
CRTDFUDEF  APP(TEST1) FILE(FILE1) +
      TEXT('Create application for TEST1')
```

This command begins a prompting sequence which allows you to create an application named TEST1 in library QGPL. Your responses to the prompts define TEST1. Application TEST1 uses data from the data base file FILE1. No UDS or internal representations of TEST1 will be printed. Any system users can execute or read TEST2, but only the owner of the application can delete it.

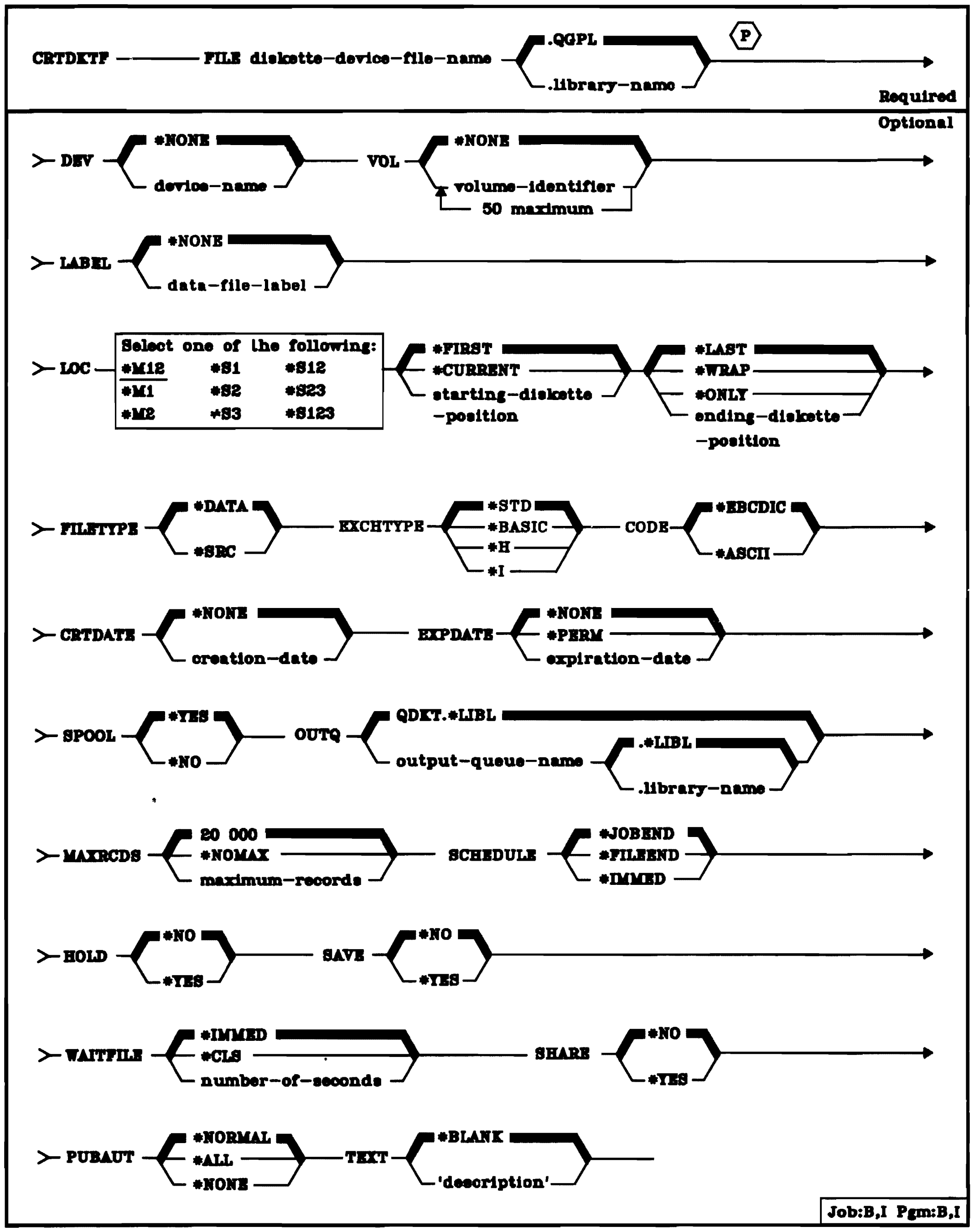
CRTDKTF (Create Diskette File) Command

The Create Diskette File (CRTDKTF) command creates a diskette device file. The device file contains the file description, which identifies the device to be used and specifies the spooling requirements; the device file does not contain data. The diskette device file is used to read and write records on diskettes that are in the diskette device and that have been initialized in the basic, H or I exchange format. The same device file can be used for both input and output operations.

Note: This command is not to be used to create device files for use in save/restore operations. User-created device files are not needed for save/restore operations.

All the information in the diskette file description is contained in the command that creates it; there is no DDS (data description specifications) for diskette device files. The diskette file has only one record format for input/output operations. The record format consists of one character field containing the input data retrieved from the device or the output data to be written to the device. The program using the device file must describe the fields in the record format so that the program can arrange the data received from or sent to the device in the manner specified by the diskette file description.

The CHGDKTF or OVRDKTF command can be used in a program to change or override the parameter values specified in the diskette file description. Each changed value in the device file remains changed after the program ends. Each overridden value remains altered only for the execution of the program (unless the override is deleted by a DLTOVR command); once the program ends, the original parameter values specified for the diskette file are used. Override commands must be executed before the diskette file to be affected is opened for use by the program.


CRTDKTF
FILE diskette-device-file-name
.QGPL
.library-name
P
Required
Optional
DEV
*NONE
device-name
VOL
*NONE
volume-identifier
50 maximum
LABEL
*NONE
data-file-label
LOC
Select one of the following:
*M12 *S1 *S12
*M1 *S2 *S23
*M2 *S3 *S123
*FIRST
*CURRENT
starting-diskette -position
*LAST
*WRAP
*ONLY
ending-diskette -position
FILETYPE
*DATA
*SRC
EXCHTYPE
*STD
*BASIC
*H
*I
CODE
*EBCDIC
*ASCII
CRTDATE
*NONE
creation-date
EXPDATE
*NONE
*PERM
expiration-date
SPOOL
*YES
*NO
OUTQ
QDKT.*LIBL
output-queue-name
.*LIBL
.library-name
MAXRCDS
20 000
*NOMAX
maximum-records
SCHEDULE
*JOBEND
*FILEEND
*IMMED
HOLD
*NO
*YES
SAVE
*NO
*YES
WAITFILE
*IMMED
*CLS
number-of-seconds
SHARE
*NO
*YES
PUBAUT
*NORMAL
*ALL
*NONE
TEXT
*BLANK
'description'
Job:B,I Pgm:B,I

FILE Parameter: Specifies the qualified name by which the diskette device file being created is to be known. If no library qualifier is given, the file is stored in QGPL. (If the file is to be used by an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

DEV Parameter: Specifies the name of the diskette device that is to be used with this diskette device file to perform input/output data operations. The device name of the IBM-supplied diskette device description is QDKT.

*NONE: No device name is to be specified. The name of the diskette device must be specified later in the CHGDKTF or OVRDKTF command, or in the HLL program that opens the file.

device-name: Enter the name of the device that is to be used with this diskette device file. The device name must already be known on the system via a device description before this device file is created.

VOL Parameter: Specifies one or more volume identifiers of the diskettes (either in magazines or in slots) to be used by this device file. The diskettes (volumes) must be mounted on the device in the same order as the identifiers are specified here; a message is sent to the system operator if they are not. The identifiers are matched, one by one, with the diskette locations specified in the LOC parameter. (For an expanded description of the VOL parameter, see Appendix A.)

*NONE: The diskette volume identifiers are not specified for this file. They can be supplied before the device file is opened, either in the OVRDKTF (or CHGDKTF) command or in the HLL program. Otherwise, no volume identifier checking is performed.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used by this device file. Each identifier can have 6 alphameric characters or fewer.

LABEL Parameter: Specifies the data file label of the data file on diskette that is to be used with this diskette device file. This data file is stored in a label in the volume label area of the diskette. For input files (diskette input to system), it specifies the identifier of the file that exists on the diskette. For output files (system output to diskette), the label specifies the identifier of the file that is to be created on the diskette. (For an expanded description of the LABEL parameter, see Appendix A.)

*NONE: The data file label is not specified. It must be supplied before the device file is opened, either in the CHGDKTF (or OVRDKTF) command or in the HLL program.

data-file-label: Enter the identifier (8 characters maximum) of the data file to be used with this diskette device file. (See Appendix A for details.)

LOC Parameter: Specifies which diskette location(s) in the magazines or slots is to be used by this diskette device file. Three values are needed: (1) the unit type and location (that is, the magazines or slots used), (2) the starting diskette position, and (3) the ending diskette position in the unit. (For an expanded description of the LOC parameter, see Appendix A.) If LOC is not specified, *M12, *FIRST, and *LAST are assumed by the system.

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit and location on the diskette magazine drive are to be used by the device file for diskette input/output. Enter one of the following values for the unit type and location (the valid starting and ending positions for each unit type are also listed):

Unit Type/Location	Diskette Starting and Ending Position
*M12	1 through 10
*M1	1 through 10
*M2	1 through 10
*S1	1
*S2	2
*S3	3
*S12	1 through 2
*S23	2 through 3
*S123	1 through 3

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette used first by the device file. Enter one of the following values to specify the starting diskette position:

FIRST: The first diskette position in the location contains the diskette to be used first in the read or write operation. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used.

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine or the manual slot that contains the first diskette to be used.

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette used last by the device file. Enter one of the following values to specify the ending diskette position:

LAST: The last diskette position in the location contains the diskette to be used last in the read or write operation. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**WRAP:* If the end of the last diskette in the location is reached before the end of the data file is reached, a message is sent to the system operator to mount another magazine or diskette to continue. (See Appendix A for details and restrictions on using *WRAP.)

**ONLY:* Only the diskette position specified by the second value is to be used, and used only once.

ending-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine or the manual slot that contains the last diskette to be used.

FILETYPE Parameter: Specifies whether the diskette device file being created describes data records or describes source records (statements) for a program or another file. (For an expanded description of the FILETYPE parameter, see Appendix A.)

**DATA:* The diskette file describes data records.

**SRC:* The diskette file describes source records.

EXCHTYPE Parameter: Specifies, for diskette output files only, the exchange type to be used by the device file when the system is writing diskette data. (For an expanded description of the EXCHTYPE parameter, refer to Appendix A.)

**STD:* The basic exchange format will be used for a type 1 or a type 2 diskette. The H exchange type will be used for a type 2D diskette.

**BASIC:* The basic exchange type will be used.

**H:* The H exchange type will be used.

**I:* The I exchange type will be used.

CODE Parameter: Specifies the type of character code to be used when diskette data is read or written by a job that uses this device file.

**EBCDIC:* The EBCDIC character code is to be used with this device file.

**ASCII:* The ASCII character code is to be used with this device file.

CRTDATE Parameter: Specifies when the diskette data file was created on diskette. The creation date parameter is valid for diskette input data files only. If the creation date written on the diskette containing the data file does not match the date specified for the device file when it is opened, an error message is sent to the user program.

**NONE:* The creation date is not specified. It is not checked unless it is supplied before the device file is opened, either in the OVRDKTF (or CHGDKTF) command or in the HLL program.

creation-date: Enter the creation date of the data file to be used by this device file. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP. However, the specified date is put in the diskette label as *yymmdd*.

EXPDATE Parameter: Specifies, for diskette output data files only, the expiration date of the data file used by this device file. If a date is specified, the data file is protected and cannot be written over until the day after the specified expiration date.

*NONE: No expiration date for the data file is to be specified; the file is to be protected one day. Its protection expires the day after it is created.

**PERM:* The data file is to be protected permanently. The date written on the diskette is 999999.

expiration-date: Enter the expiration date after which the data file expires. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP. However, the specified date is put in the diskette label as *yymmdd*.

SPOOL Parameter: Specifies whether the input or output data for the diskette device file is to be spooled. If SPOOL(*NO) is specified, the following parameters in this command are ignored: OUTQ, MAXRCDS, SCHEDULE, HOLD, and SAVE.

*YES: The data is to be spooled. If this file is opened for input, an inline data file having the specified name is processed; otherwise, the next unnamed inline spooled file is processed. (For a discussion of named and unnamed inline files, see the *CPF Programmer's Guide*.) If this is an output file, the data is spooled for processing by a card, diskette, or printer writer.

**NO:* The data is not to be spooled. If this file is opened for input, the data is read directly from the diskette. If this is an output file, the data is written directly to the diskette as it is processed by the program.

OUTQ Parameter: Specifies, for spooled output only, the name of the output queue for the spooled output file.

QDKT: The spooled output data is sent to the IBM-supplied QDKT output queue. (If no library qualifier is specified, *LIBL is used to find the output queue.)

qualified-output-queue-name: Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.)

MAXRCDS Parameter: Specifies, for spooled output only, the maximum number of records that can be in the spooled output file for spooled jobs using this diskette device file.

20000: A maximum of 20 000 records can be in the spooled output file for the diskette data file that is produced by this device file.

**NOMAX:* No maximum is specified for the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of diskette records that can be in the spooled output file.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a writer.

*JOBEND: The spooled output file is to be made available to the writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the writer as soon as the file is opened in the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The spooled file is made available to a writer when it is released by the Release Spooled File (RLSSPLF) command.

*NO: The spooled output file is not to be held by the output queue. The spooled output is made available to a writer based on the SCHEDULE parameter value.

**YES:* The spooled output file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled file is to be saved (left on the output queue) after the output has been produced.

*NO: The spooled file data is not to be retained on the output queue after it has been produced.

**YES:* The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the diskette device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used.

When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

*NO: An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

PUBAUT Parameter: Specifies what authority for the diskette device file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the device file.

**ALL:* The public has complete authority for the device file.

**NONE:* The public cannot use the device file.

TEXT Parameter: Lets the user enter text that briefly describes the diskette device file. (For an expanded description of the TEXT parameter, see Appendix A.)

<u>*BLANK:</u> No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTDKTF  FILE(DSPHST)
```

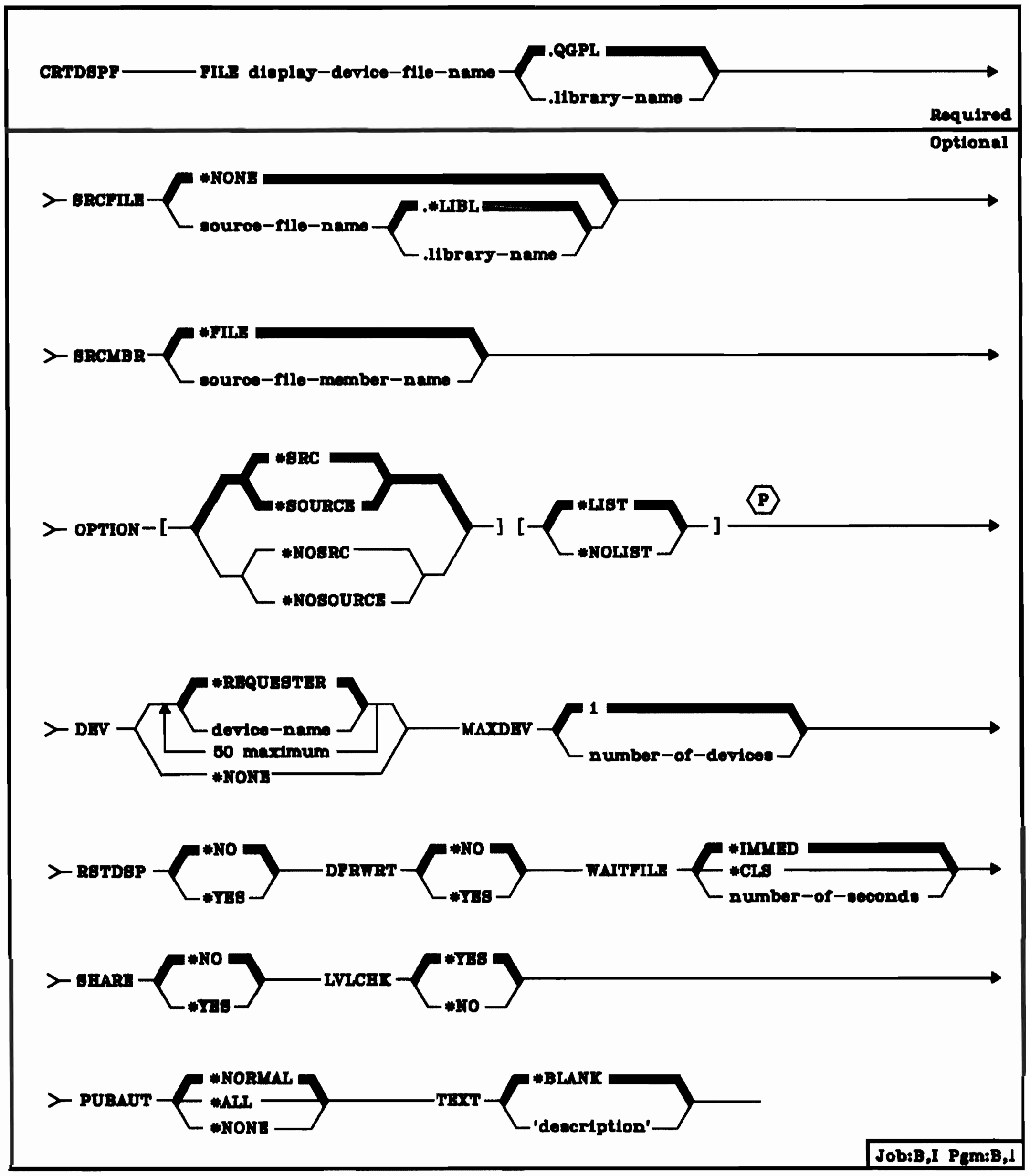
This command creates a description of the diskette device file named DSPHST. The defaults for all the other parameters are assumed. The device name, diskette volume, file label, and the creation date of the data file on diskette must be specified in another CL command or in each program that uses the device file. The device file describes diskette data files that are in EBCDIC code and that are to be spooled for both input and output. Output goes to the QDKT output queue, and cannot go on diskette until the job is completed on the system. When output is produced from the output queue, only one copy is produced.

CRTDSPF (Create Display File) Command

The Create Display File (CRTDSPF) command creates a display device file. The device file contains the file description, which identifies the device to be used and, optionally, the record formats used by the device (if specified in DDS); the device file does not contain data. The display device file is used to send records to one or more display devices associated with the device file, and to receive records from the display devices.

The display file description is made up of information that is specified in two places: (1) in the source file that contains the data description specifications (if used); and (2) in the CRTDSPF command itself. The DDS contains the specifications for each record format in the device file and for the fields within each record format.

The CHGDSPF or OVRDSPF command can be used in a program to change or override the parameter values specified in the display file description; the override command must be executed before the display file is opened for use by the program. Overridden values are altered only for the execution of the program; once the program ends, the original parameter values specified for the display file are used.



FILE Parameter: Specifies the qualified name by which the display device file being created is to be known. If no library qualifier is given, the file is stored in QGPL. (If the file is to be used by an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

SRCFILE Parameter: Specifies the name of the source file (if there is one) that contains the data description specifications for the records in the display device file. (The specifications that can be made in DDS are described in the *CPF Reference Manual–DDS*.)

<u>*NONE:</u> There is no DDS source file for this display device file; the device file has only one record format with no fields, or else the program that uses the file must describe the record formats and their fields.

qualified-source-file-name: Enter the qualified name of the source file that contains the DDS for this display device file. (If no library qualifier is given, *LIBL is used to find the source file.)

SRCMBR Parameter: Specifies the name of the member in the source file that contains the DDS for this display device file.

<u>*FILE:</u> The source file member name is the same as the device file name specified in the FILE parameter.

source-file-member-name: Enter the name of the member in the source file specified by SRCFILE that is to be used to create the display device file.

OPTION Parameter: Specifies the type of output listing to be produced when the file is created.

<u>*SRC</u> or **SOURCE:* A listing of the source statements used to create the file, and of any errors that occur, is to be generated.

**NOSRC* or **NOSOURCE:* No listing of the source statements is to be generated unless errors are detected. If errors are detected, they are listed along with the record format containing the error.

<u>*LIST:</u> An expanded source listing is to be generated showing a detailed list of the file specifications that result from the source statements and references to other file descriptions. This listing shows file and field keywords and attributes and, for data base files, key and select/omit keywords.

**NOLIST:* No expanded source listing is to be generated.

DEV Parameter: Specifies the names of one or more display devices that are to be used with this display device file to pass data records between the users of the display devices and their jobs.

*REQUESTER: The device from which the program was invoked is the device that is assigned to the file when the file is opened.

**NONE:* No device name is to be specified. The name of the display device must be specified later in the CHGDSPF or OVRDSPF command, or in the HLL program that opens the file.

device-name: Enter the names of one or more display devices that are to be used with this device file to pass data records between the users of the devices and the system. Each device name must already be known on the system (via a device description) before this device file is created. *REQUESTER can be specified as one of the names.

A maximum of 50 device names (including *REQUESTER, if it is specified) can be specified in this command, but the total number cannot exceed the number specified in the MAXDEV parameter when the file is opened.

MAXDEV Parameter: Specifies the maximum number of display devices that can be connected to the display device file at the same time, while the file is open. A value of 1 is normally specified because each work station user has his own copy of the program that uses the file. However, if a CL program is written to access more than one work station through the same file (through a single execution of the program), this parameter must specify a value greater than 1.

The names of the devices can be specified in the DEV parameter of this command, an OVRDSPF command, or in the HLL program that opens the file.

1: Only one device name or *REQUESTER can be specified for this display device file.

number-of-devices: Enter a value, 1 through 255, that specifies the maximum number of devices that can be connected to this display file at the same time.

RSTDSP Parameter: Specifies whether data being displayed at a display device by this display file is to be saved at the time the file is suspended (temporarily inactive) so that a different display file can be used to display different data on the same device. If the data for this file is saved, it is restored to the screen of the device when the file is used again.

This parameter must be considered if, within the same routing step, any program can be called that uses a different display file for the same device. If *all* programs that use this file always display new data when control is returned to them, the display data for this file need not be saved for any of them; RSTDSP(*NO) can be specified or assumed. If *any* program using this file requires that the contents of the screen be exactly the same as it was before it called another program, RSTDSP(*YES) must be specified. If certain display fields are to remain unchanged while others are erased or rewritten, or if the program containing the file can be interrupted (for messages to be displayed, for example), you should specify RSTDSP(*YES). (For additional information about suspended display files, see the *CPF Programmer's Guide.*)

NO: The data being displayed by this file is not to be saved when the file is suspended. None of the programs using this file need the data restored when control is returned to them.

**YES:* The data being displayed when the file is suspended is to be saved so it can be restored to the screen of the device when the file is used again.

DFRWRT Parameter: Specifies that the writing of data is to be deferred until it can be written out with other data when a read request is made. Control is returned to the program immediately after the data is received. This may result in improved performance.

**NO:* After a write operation, the user program does not regain control until the I/O is completed (with the data displayed and the I/O feedback information available).

**YES:* When the program issues a write request, control is returned after the buffer is processed. The data might not be displayed immediately; the actual display of the data might take place later when a read or combined read/write operation is performed. The buffer is then available to be prepared for the next read or combined read/write operation.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the display device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used.

When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

*NO: An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given an internal system identifier when the format is created.

*YES: The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not all match, an open error message is sent to the program that attempted to open the file.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

PUBAUT Parameter: Specifies what authority for the display device file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

**NORMAL:* The public has only operational rights for the device file.

**ALL:* The public has complete authority for the device file.

**NONE:* The public cannot use the device file.

TEXT Parameter: Lets the user enter text that briefly describes the display device file. (For an expanded description of the TEXT parameter, see Appendix A.)

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTDSPF  FILE(DSPHIST)  SRCFILE(JOBHIST.PRSNNL)
```

This command creates a description of the display device file named DSPHIST using the device source file description named JOBHIST that is stored in the PRSNNL library. The defaults for all the other parameters are assumed. Only the device requesting the program that uses this device file (that is, *REQUESTER) is assigned to the device file. The level identifiers of the record formats are to be checked when the file is opened. The public has only operational rights for the device file.

CRTDTAARA (Create Data Area) Command

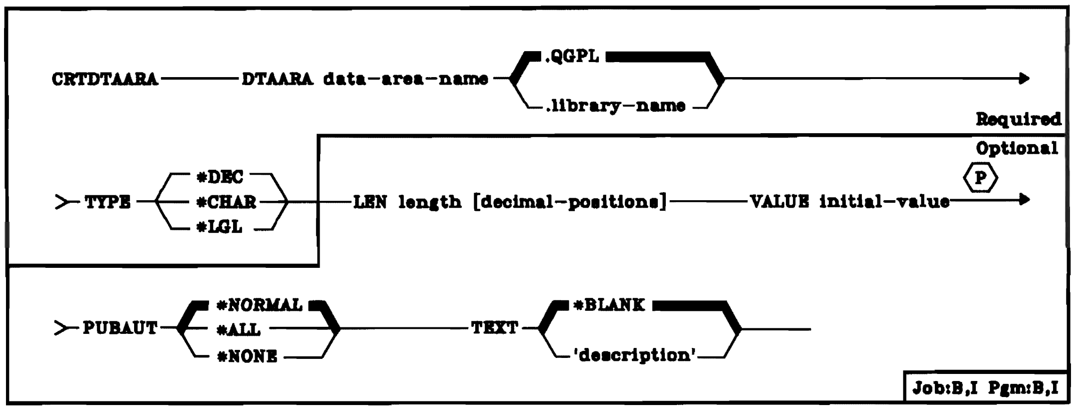
The Create Data Area (CRTDTAARA) command creates a data area and stores it in a specified library. It also specifies the attributes of the data. The data area can optionally be initialized to a specific value.

Data areas (which are a type of CPF object) are used to communicate and store data used by several programs in a job or between jobs. A program can use the value of a data area by declaring the data area in the program, using the DCLDTAARA command.

If a data area is not to be used by more than one job at a time, it can be explicitly allocated to the appropriate job. If a data area is used by two or more jobs concurrently, it is protected from simultaneous changes occurring from different jobs. A data area is changed by using the Change Data Area (CHGDTAARA) command. The system does not allow two CHGDTAARA commands to change the same data area at the same time.

A data area is updated in auxiliary storage any time the data area is changed. This ensures that the changes are not lost in the event of a program or system failure.

Restrictions: To use this command, you must have operational and add rights for the library in which the data area is to be placed.



DTAARA Parameter: Specifies the qualified name of the data area being created. (If no library qualifier is given, the data area is stored in the general purpose library, QGPL.)

TYPE Parameter: Specifies the type of value to be contained in the data area being created. The data area can contain a character value, a decimal value, or a logical one or zero. Enter one of the following types.

**DEC:* This data area contains a decimal value.

**CHAR:* This data area contains a character string value.

**LGL:* This data area contains a logical value of either one ('1') or zero ('0') that can be used to represent two opposing conditions such as on/off, true/false, or yes/no.

LEN Parameter: Specifies the length of the data area being created. If it is a decimal data area, the number of decimal digits to the right of the decimal point can be optionally specified. The type of data area (specified by the TYPE parameter) determines the maximum length that its value can have and the default length that is assumed if LEN is not specified. The maximum lengths and the defaults for each of the three types are:

Type	**Maximum Length**	**Default Length**
Decimal	15 digits, 9 decimal positions	15 digits, 5 decimal positions
Character	2000 characters	32 characters
Logical	1 character	1 character

Note: For character types, the default length is the same as the length of the initial value, if one is specified in the VALUE parameter.

length: Enter the length that the value in this data area can have; the length cannot exceed the maximum for this type of variable.

length [decimal-positions]: This option is valid only for *decimal* data areas. The length of the value in the data area includes the number of decimal positions in the value. The maximum length of the decimal value is 15 digits, of which no more than nine can be to the right of the decimal point. (If nine decimal positions are specified, the value to the *left* of the decimal point could never be greater than 999 999; only six of the 15 digits are left for the integer value.) If TYPE(*DEC) is specified and the number of decimal positions is not specified, a value of 0 is assumed; 15 digits to the left of the decimal point are then allowed.

VALUE Parameter: Specifies the initial value that is assigned to the data area when it is created. The initial value must be of the type specified by the TYPE parameter. If no initial value is specified, a character data area is initialized to blanks, a decimal data area is initialized to a value of 0, and a logical data area is initialized to '0'.

PUBAUT Parameter: Specifies what authority for the data area is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

**NORMAL:* The public has only operational rights for the data area.

**ALL:* The public has complete authority for the data area.

**NONE:* The public cannot use the data area.

TEXT Parameter: Lets the user enter text that briefly describes the data area. (For an expanded description of the TEXT parameter, see Appendix A.)

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTDTAARA  DTAARA(TOTSALES) +
     TYPE(*DEC) LEN(15 2) VALUE(0) +
     TEXT('Total sales accumulator')
```

This command creates a data area named TOTSALES and stores it in the QGPL library. TOTSALES has the following data attributes: it is a 15-position numeric data area with two decimal positions and with an initial value of 0.

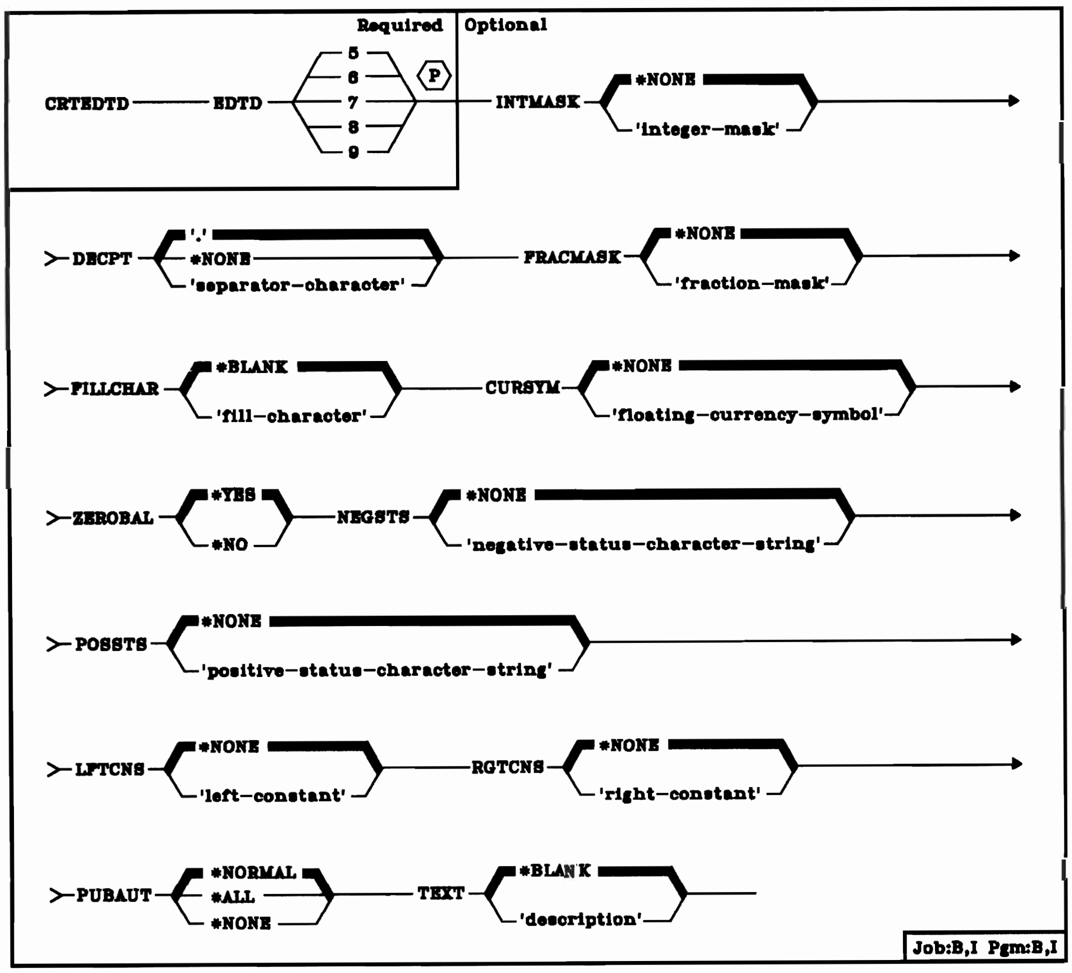
```
CRTDTAARA  DTAARA(CUSTOMER) TYPE(*CHAR) LEN(148) +
     TEXT('Customer name area')
```

This command creates the data area named CUSTOMER. It can contain as many as 148 characters in the character string. Because no initial value is specified, the data area is initialized to blanks.

CRTEDTD (Create Edit Description) Command

The Create Edit Description (CRTEDTD) command defines an edit mask for the specified edit description and stores it in the QSYS library. As many as five edit descriptions can be defined by the user. They must be identified as edit descriptions 5, 6, 7, 8, and 9. The actual object names for the edit descriptions are *QEDITn*, where *n* is the single digit identifying code. CPF provides a version of each of these edit descriptions in QSYS. (For a description of the IBM-supplied versions, see the *CPF Programmer's Guide*.) To create a new version, the IBM-supplied version must first be deleted by the DLTEDTD command.

Edit descriptions can be used in data description specifications and high-level language programs to edit numeric fields.



EDTD Parameter: Specifies a single digit code (5, 6, 7, 8, or 9) that identifies the user-defined edit description being created. This digit is used in DDS to refer to the edit mask that is created by this CRTEDTD command. The actual name of the created object (which is stored in the QSYS library) is *QEDITn*, where *n* is the single digit edit code entered here.

INTMASK Parameter: Specifies a character string (mask) that describes the editing of the integer portion of a decimal field. Characters other than a space (blank), a zero, or an ampersand (&) are treated as constants in the editing process. Space, zero, and ampersand have the following meanings:

- Space (blank): Each blank is replaced with a fill character or with a digit from the number being edited once zero suppression has been terminated (by a significant digit or by the leftmost zero in the mask).
- Zero (0): The leftmost zero is a digit replacement character and also terminates zero suppression. All other zeros in the integer mask are treated as constants.
- Ampersand (&): Blank substitution.

*NONE: No editing mask is to be used on the integer portion of decimal fields.

'integer-mask': Enter the character string that is to be used as the editing mask for the integer portion of a decimal field. A maximum of 31 characters, enclosed in apostrophes, can be used in the integer mask.

DECPNT Parameter: Specifies, for decimal fields, a single character to be used as a decimal point to separate the integer (INTMASK) and fraction (FRACMASK) portions of the edited result. If the field has no decimal places, this character is not used and need not be considered in the width of the edited results.

Note: If the separator character specified for DECPNT is also used in the INTMASK parameter, it has no special meaning in the integer mask; it is treated only as a constant or as a digit replacement character in the integer mask.

'.': The period (or decimal point) is the separator character. If specified, it must be enclosed in apostrophes.

**NONE:* No separator character is specified; a decimal point is not needed in the edited result.

'separator-character': Enter the separator character, such as the comma (,), that is to be used as a decimal point. Any alphameric or special character can be used, but a special character must be enclosed in apostrophes.

FRACMASK Parameter: Specifies a character string (mask) that describes the editing of the fraction portion of a decimal field (to the right of the decimal point). The characters have the same meaning as described for the INTMASK parameter except that all zeros are treated as constants and blanks are not replaced with a fill character.

*NONE: No editing mask is to be used on the fraction portion of decimal fields.

'fraction-mask': Enter the character string that is to be used as the editing mask for the fraction portion of a decimal field. A maximum of 31 characters, enclosed in apostrophes, can be used in the fraction mask.

FILLCHAR Parameter: Specifies the character that is used in each position of a result that is zero suppressed. The specified character replaces all leading zeros that are to the left of the first significant digit in the integer mask (or a forced zero).

*BLANK: The fill character is a blank (a space).

'fill-character': Enter the character that is to be used as the fill character. Any alphameric or special character can be used, but a special character must be enclosed in apostrophes.

CURSYM Parameter: Specifies the character string that is to be used as the floating currency symbol. If CURSYM is specified, the character string appears immediately to the left of the first significant digit (or constant). If the first significant digit is a zero, occurring in the position that terminated zero suppression, the CURSYM character string ends in the position occupied by that zero.

*NONE: No floating currency symbol is specified; none is needed in the edited result.

'floating-currency-symbol': Enter the character string that is to be used as the floating currency symbol for monetary amount fields. A maximum of 15 alphameric and special characters, enclosed in apostrophes, can be entered.

ZEROBAL Parameter: Specifies the editing action for zero values.

*YES: The normal editing rules are followed. (Refer to *Editing Rules*, following the description of the CRTEDTD command parameters.)

*NO: The entire field (integer, decimal point, or fraction) is replaced by the fill character, including constants within the edit mask, if the field being edited has a value of zero.

NEGSTS Parameter: Specifies the character string that immediately follows the body of the edited result if the field is negative. If the field is positive, blanks are substituted for the length of the string, unless POSSTS is also specified.

*NONE: No character string is specified; blanks will be used to the right of the field in the edited result.

'negative-status-character-string': Enter the character string that is to immediately follow the edited field when the field is negative in value. A maximum of 31 characters, enclosed in apostrophes, can be entered as the negative status character string.

POSSTS Parameter: Specifies the character string that immediately follows the body of the edited result if the field is positive or zero. If the field is negative, blanks are substituted for the length of the string, unless NEGSTS is also specified.

*NONE: No character string is specified; blanks will be used to the right of the field in the edited result.

'positive-status-character-string': Enter the character string that is to immediately follow the edited field when the field is positive in value. A maximum of 31 characters, enclosed in apostrophes, can be entered as the positive status character string.

LFTCNS Parameter: Specifies the character string constant that always appears as the leftmost portion of the edited result.

*NONE: No constant is to appear on the left side of edited fields.

'left-constant': Enter the character string that is to always appear on the left side of an edited field. A maximum of 31 characters, enclosed in apostrophes, can be entered.

RGTCNS Parameter: Specifies the character string constant that always appears as the rightmost portion of the edited result.

*NONE: No constant is to appear on the right side of edited fields.

'right-constant': Enter the character string that is to always appear on the right side of an edited field. A maximum of 31 characters, enclosed in apostrophes, can be entered.

PUBAUT Parameter: Specifies what authority for the edit description is being granted to the public (all users). Additional authority can be granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

<u>*NORMAL</u>: The public has only operational rights for the edit description.

**ALL:* The public has complete authority for the edit description.

**NONE:* The public cannot use the edit description.

TEXT Parameter: Lets the user enter text that briefly describes the edit description. (For an expanded description of the TEXT parameter, see Appendix A.)

<u>*BLANK</u>: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Editing Rules

- The field to be edited is aligned with respect to the two portions of the edit mask (integer and fraction).
- The integer mask INTMASK is truncated on the left side immediately preceding the leftmost digit replace character that could be used, based on the number of integer digits in the field to be edited. If a leading zero occurs in the truncated portion of the integer mask, this is remembered by the system and no zero suppression occurs.
- The separator character used as the decimal point (DECPNT) immediately follows the integer mask.
- The fraction mask FRACMASK immediately follows the separator character (or the integer mask if DECPNT(*NONE) is specified). The fraction mask is truncated on the right side immediately following the rightmost digit replace character that could be used, based on the number of decimal positions in the field to be edited.
- The width of the edited result can be calculated as follows:

  (length of LFTCNS) + (length of CURSYM) +
  (length of truncated INTMASK) +
  (1 (or 0 if DECPNT equals *NONE)) +
  (length of truncated FRACMASK) +
  (length of NEGSTS or POSSTS) +
  (length of RGTCNS) = width of edited result

- If either the integer mask or the fraction mask does not contain sufficient digit replace characters to contain the digits that can be contained in the respective portions of the field, editing of the field is diagnosed and ignored, and an error message is sent to the user (or program).

- Changing the edit description has no effect on previously created file formats. These file formats must be recreated if the new (changed) edit mask is desired.

Examples

The examples assume the following:

FIELDA	– Six digits (four integer and two decimal positions) with a value of 001234
FIELDB	– Same as FIELDA but with a negative value (-001234)
FIELDC	– Same as FIELDA but with a zero value (000000)
DATE	– Six digits (0 decimal positions) with a value of 091878

The character ƀ is used to represent blank spaces.

Example 1

```
CRTEDTD  EDTD(5)  INTMASK('ƀƀƀ,ƀƀƀ,ƀƀ0') +
      FRACMASK('ƀƀƀƀ') +
      NEGSTS('DBƀƀƀƀ')  POSSTS('CREDIT') +
      LFTCNS('$')  RGTCNS('ƀ**')
```

FIELDA	– Logical mask is '$ƀ,ƀƀ0.ƀƀ DBƀƀƀƀ ƀ**' for a negative value or '$ƀ,ƀƀ0.ƀƀ CREDIT ƀ**' for a positive value
	– Edited result is $ƀƀƀ12.34CREDITƀ**
FIELDB	– Same logical mask
	– Edited result is $ƀƀƀ12.34DBƀƀƀƀƀ**
FIELDC	– Same logical mask
	– Edited result is $ƀƀƀƀƀ.00CREDITƀ** or, if ZEROBAL(*NO) had been specified, $ƀƀƀƀƀƀƀƀCREDITƀ**

Example 2

```
CRTEDTD  EDTD(6) INTMASK('ƀƀƀ.ƀ0ƀ') DECPNT(',') +
     FRACMASK('ƀƀƀ') CURSYM('DM') NEGSTS('-ƀ**')
```

FIELDA – Logical mask is 'ƀƀƀ.ƀ0ƀ,ƀƀ-ƀ**' with floating DM

– Edited result is ƀƀƀDM12,34ƀƀƀƀ

FIELDB – Same logical mask

– Edited result is ƀƀƀDM12,34-ƀ**

FIELDC – Same logical mask

– Edited result is ƀƀƀƀDM0,00ƀƀƀƀ or, if ZEROBAL(*NO) had been specified, ƀƀƀƀƀƀƀƀƀƀƀƀƀƀ

Example 3

```
CRTEDTD  EDTD(7) INTMASK('0ƀMONTHƀƀDAY&ƀƀYEAR') +
     LFTCNS('DATEƀISƀ')
```

DATE – Logical mask is equal to the INTMASK parameter value

– Edited result is DATEƀISƀƀ9MONTH18DAYƀ78YEAR'

Example 4

```
CRTEDTD  EDTD(9) INTMASK('ƀƀ,ƀƀ0') DECPNT('.') +
     FRACMASK('ƀƀ') FILLCHAR('*') NEGSTS('ƀERRORƀ**')
```

FIELDA – Logical mask is 'ƀ,ƀƀ0.ƀƀƀƀƀƀƀƀƀƀ' or 'ƀ,ƀƀ0.ƀƀƀERRORƀ**' (Both use the * as the fill character)

– Edited result is ***12.34ƀƀƀƀƀƀƀƀƀ

FIELDB – Same logical mask

– Edited result is ***12.34ƀERRORƀ**

FIELDC – Same logical mask

– Edited result is *****.00ƀƀƀƀƀƀƀƀƀ or, if ZEROBAL(*NO) had been specified, ********ƀƀƀƀƀƀƀƀƀ

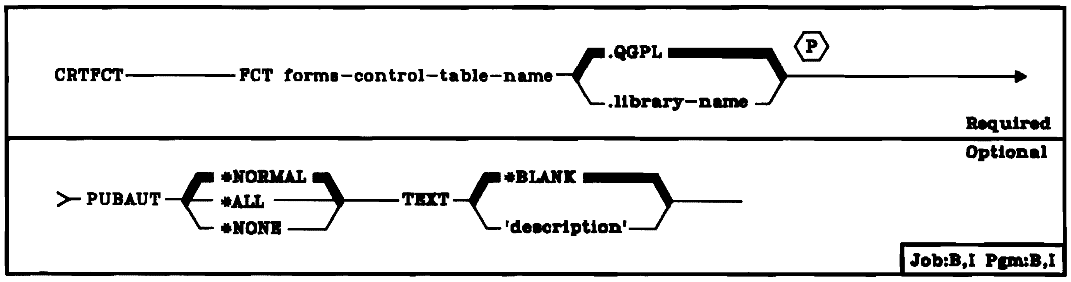
CRTFCT (Create Forms Control Table) Command

The Create Forms Control Table (CRTFCT) command creates a forms control table (FCT) with no entries.

After it is created, the FCT can contain up to 999 entries. Refer to the CRTSSND command for making the FCT entries available to the RJEF session. Refer to the ADDFCTE command for adding entries to the FCT.

Restriction: To use this command, you must have add rights to the library in which the FCT is to be stored.

The Create Forms Table (CRTFCT) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



FCT Parameter: Specifies the qualified name of the FCT that is to be created. (f no library qualifier is given, the FCT is stored in QGPL.)

PUBAUT Parameter: Specifies the authority that is being granted to the public (all users) for the FCT. Additional authority can be explicitly granted to specific users by the Grant Object Authority (GRTOBJAUT) command.

NORMAL: The public has only operational rights for the FCT.

**ALL:* The public has complete authority for the FCT.

**NONE:* The public cannot use the FCT.

TEXT Parameter: Lets the user enter text that briefly describes the FCT. (For an expanded description of the TEXT parameter, see Appendix A.)

**BLANK:* No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTFCT  FCT(FORMCTRL.USERLIB) +
     TEXT('Forms control table')
```

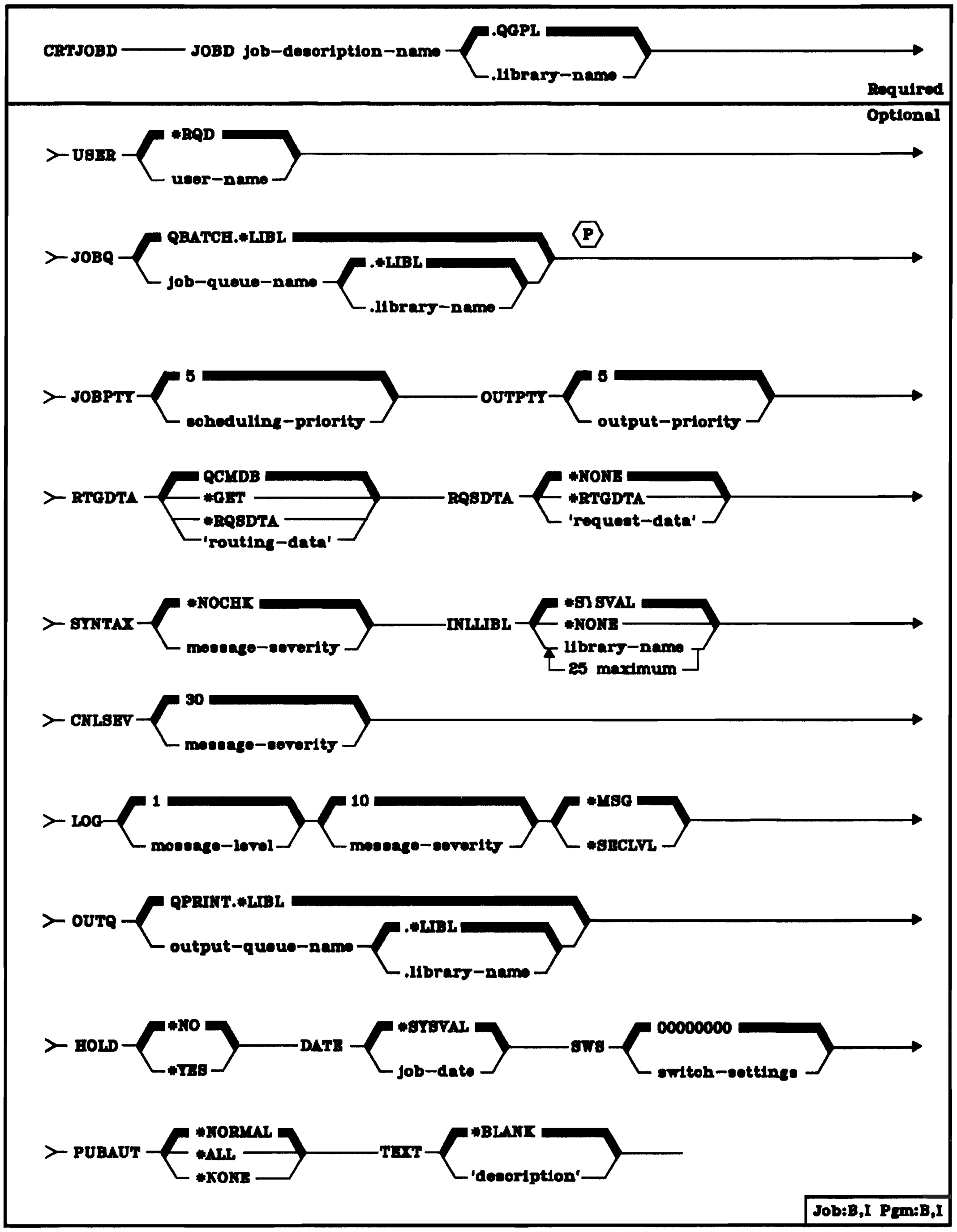
This command creates a forms control table called FORMCTRL in library USERLIB.

CRTJOBD (Create Job Description) Command

The Create Job Description (CRTJOBD) command creates a job description object that contains a specific set of job-related attributes that can be used by one or more jobs. The attributes determine how each job is to be executed on the system. The same job description can be used by multiple jobs. The values in the job description are usually used as the default values of the corresponding parameters in the JOB and SBMJOB commands when their parameters are not specified.

The values in the job description can be overridden by the values specified in the JOB and SBMJOB commands.

Restrictions: To use this command, you must have operational rights for the user profile named in the USER parameter (if any); that is, you must have the authority to initiate a job on behalf of that user. You must also have operational and add rights for the library into which the job description is to be placed.


CRTJOBD
JOBD job-description-name
.QGPL
.library-name
Required
Optional
USER
*RQD
user-name
JOBQ
QBATCH.*LIBL
job-queue-name
.*LIBL
.library-name
P
JOBPTY
5
scheduling-priority
OUTPTY
5
output-priority
RTGDTA
QCMDB
*GET
*RQSDTA
'routing-data'
RQSDTA
*NONE
*RTGDTA
'request-data'
SYNTAX
*NOCHK
message-severity
INLLIBL
*SYSVAL
*NONE
library-name
25 maximum
CNLSEV
30
message-severity
LOG
1
message-level
10
message-severity
*MSG
*SECLVL
OUTQ
QPRINT.*LIBL
output-queue-name
.*LIBL
.library-name
HOLD
*NO
*YES
DATE
*SYSVAL
job-date
SWS
00000000
switch-settings
PUBAUT
*NORMAL
*ALL
*NONE
TEXT
*BLANK
'description'
Job:B,I Pgm:B,I

JOBD Parameter: Specifies the qualified name of the job description being created. (If no library qualifier is given, the job description is stored in the general purpose library, QGPL.)

USER Parameter: Specifies the name of the user profile to be associated with this job description. The names QSECOFR, QSPL, and QSYS are not valid entries for this parameter.

*RQD: A user name is required in order to use the job description. For work station entries, the user must enter his password when he signs on at the work station; the associated user name becomes the name used for the job. *RQD is not valid for job descriptions specified for autostart job entries, or for those used by the JOB command. (It is valid on the SBMJOB command only if USER(*CURRENT) is specified.)

user-profile-name: Enter the user name that identifies the user profile that is to be associated with batch jobs using this job description. For interactive jobs, this is the default user name used when a user signs on without entering a password.

JOBQ Parameter: Specifies the name of the job queue into which jobs using this job description are to be placed.

QBATCH: The QBATCH job queue is the queue into which the jobs are to be placed. (If no library qualifier is specified, *LIBL is used to find the job queue.)

qualified-job-queue-name: Enter the qualified name of the job queue that is to be associated with this job description. (If no library qualifier is given, *LIBL is used to find the job queue.) If the job queue does not exist when the job description is created, a library qualifier must be specified because the qualified job queue name is retained in the job description.

JOBPTY Parameter: Specifies the scheduling priority for each job that uses this job description. The highest priority is 1 and the lowest is 9. (For an expanded description of the JOBPTY parameter, see Appendix A.)

5: The scheduling priority that any job using this job description is to have is 5.

scheduling-priority: Enter a value, 1 through 9, for the scheduling priority that any job using this job description is to have.

OUTPTY Parameter: Specifies the output priority for spooled output files that are produced by jobs using this job description. The highest priority is 1 and the lowest is 9. (For an expanded description of the OUTPTY parameter, see Appendix A.)

5: The output priority for spooled files produced using this job description is 5.

output-priority: Enter a value, 1 through 9, for the priority that spooled files produced using this job description are to have.

RTGDTA Parameter: Specifies the routing data that is to be used with this job description to initiate jobs.

QCMDB: The default routing data QCMDB is to be used by the IBM-supplied batch subsystem (QBATCH) to route the job to the IBM-supplied control language processor QCL, in the QSYS library.

**GET:* The routing data to be used is obtained from the work station that uses the display format defined in the work station entry that references this job description.

**RQSDTA:* Up to the first 80 characters of the request data specified in the RQSDTA parameter are to be used as the routing data for the job.

'routing-data': Enter the character string that is to be used as the routing data for jobs that use this job description. For example, the value QCMDI is the routing data used by the IBM-supplied interactive subsystem (QINTER) to route interactive jobs to the IBM-supplied control language processor, QCL. A maximum of 80 characters can be entered (enclosed in apostrophes if necessary).

RQSDTA Parameter: Specifies the request data that is to be placed as the last entry in the job's message queue for jobs using this job description. For example, when a CL command is supplied as request data on a SBMJOB (Submit Job) command, it becomes a message that can be read by the control language processor, QCL (if the submitted job is routed to QCL).

*NONE: No request data is to be placed in the job's message queue.

**RTGDTA:* The routing data specified in the RTGDTA parameter is to be placed as the last entry in the job's message queue.

'request-data': Enter the character string that is to be placed as the last entry in the job's message queue as a single request. A maximum of 256 characters can be entered (enclosed in apostrophes if necessary). When a CL command is entered, it must be enclosed in single apostrophes, and where apostrophes would normally be used *within* the command, double apostrophes must be used instead.

SYNTAX Parameter: Specifies whether requests placed on the job's message queue are to be syntax checked as CL commands. When syntax checking is specified, the commands are syntax checked as they are submitted rather than when the job is executed, thus providing an earlier diagnosis of syntax errors. If checking is specified, the message severity that causes a syntax error to terminate processing of a job is also specified.

NOCHK: The request data is not to be syntax checked as CL commands.

message-severity: The request data is to be syntax checked as CL commands, and, if a syntax error occurs that is equal to or greater than the error message severity specified here, the execution of the job containing the erroneous command is suppressed. Enter a value, 00 through 99, that specifies the lowest message severity that can cause job execution to be terminated. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

If the message severity is specified, it is used only when the job description is used by a job command that also has RQSDTA(*) specified and the requests are CL commands.

INLLIBL Parameter: Specifies the initial user portion of the library list that is to be used for jobs using this job description. For more information on the use of library lists, see the *CPF Programmer's Guide.*

SYSVAL: The system default library list is to be used for jobs that use this job description. The default library list contains the library names that were specified in the system values QSYSLIBL and QUSRLIBL at the time that a job using this job description is initiated.

NONE: The user portion of the initial library list is to be empty; only the system portion is to be used.

library-name: Enter the names of one or more libraries that are to be in the user portion of the library list for jobs that use this job description. No more than 25 names can be specified; the libraries are searched in the same order as they are listed here.

CNLSEV Parameter: Specifies the message severity level of escape messages that can cause a batch job to be canceled. The batch job is canceled when a request in the batch input stream sends to the request processing program an escape message whose severity code is equal to or greater than that specified here. This parameter value is compared with the severity of any unmonitored escape message that occurs as a result of executing a noncompiled CL command in a batch job.

For a description of each IBM-defined severity code level, refer to the expanded description of the SEV parameter in Appendix A.

30: An escape message resulting from a request in the batch input stream whose severity is equal to or greater than 30 causes the job to be canceled.

message-severity: Enter a value, 00 through 50, that specifies the message severity of an escape message that results from a request in the batch input stream and that causes the job using this job description to be canceled. Because escape messages typically have a maximum severity level of 50, a value of 50 or lower must be specified in order for a job to be canceled as a result of an escape message. An unhandled escape message whose severity is equal to or greater than the value specified causes the job to be canceled. (Refer to the expanded description of the SEV parameter in Appendix A for a list of the IBM-defined severity codes.)

LOG Parameter: Specifies the message logging values to be used for the jobs that use this job description. They determine the amount and type of information to be logged in the job log. The LOG parameter is made up of a list of three values: the message (or logging) level, the message severity, and the level of message text. If no values are specified for the LOG parameter, the values 1, 10, and *MSG are assumed by the system.

Message Level: The first of the three values in the LOG parameter specifies one of five logging levels, which are described as follows:

0 No data is to be logged.

1 The only information to be logged is all messages sent to the job's external message queue with a severity greater than or equal to the message severity specified (this includes the indications of job start, job end, and job completion status).

2 The following information is to be logged:
- Logging level 1 information
- Any requests or commands being logged from a CL program for which messages are issued with a severity code greater than or equal to the severity specified
- All messages associated with a request or commands being logged from a CL program that results in a high-level message with a severity greater than or equal to the severity specified.

3 The following information is to be logged:
- Logging level 1 information
- All requests or commands being logged from a CL program
- All messages associated with a request or commands being logged from a CL program that results in a high-level message with a severity greater than or equal to the severity specified.

4 The following information is to be logged: all requests or commands being logged from a CL program and all messages, including trace messages.

Note: *A high-level* message is one that is sent to the program message queue of the program that received the request or commands being logged from a CL program.

1: A message logging level of 1 is to be used for job messages generated when this job description is used.

message-level: Enter a value, 0 through 4, that specifies the message logging level to be used for job messages produced when this job description is used.

Message Severity: The second of the three values in the LOG parameter specifies the minimum severity level that causes error messages to be logged in the job logs of jobs that use this job description. (For a description of the severity codes, see the SEV parameter in Appendix A.)

10: The message severity level is set at 10. Any errors that have a severity code of 10 or greater causes an error message to be logged in the job log of the job that caused the error.

message-severity: Enter a value, 00 through 99, that specifies the lowest severity level that causes an error message to be logged in the job's log.

Message Text Level: The third of the three values in the LOG parameter specifies the level of message text that is written in the job log or displayed to the user when an error message is generated according to the first two values.

*MSG: Only first-level message text is to be written to the job log.

**SECLVL:* Both the first-level and second-level text of the error message is to be written to the job log.

OUTQ Parameter: Specifies the name of the output queue that is to be used as the default output queue for jobs that use this job description.

QPRINT: The QPRINT output queue is the default output queue to be used with this job description. (If no library qualifier is specified, *LIBL is used to find the queue.)

qualified-output-queue-name: Enter the qualified name of the default output queue that is to be used with this job description. (If no library qualifier is given, *LIBL is used to find the queue.) If the output queue does not exist when the job description is created, a library qualifier must be specified because the qualified output queue name is retained in the job description.

HOLD Parameter: Specifies whether jobs using this job description are to be put on the job queue in the hold state. A job placed on the job queue in the hold state is held until it is released by the RLSJOB (Release Job) command or canceled, either by the CNLJOB (Cancel Job) command or by the CLRJOBQ (Clear Job Queue) command. If the job is not executed before CPF is terminated, the job queue can be cleared (and the job canceled) when CPF is started again.

*NO: Jobs using this job description are not to be held when they are put on the job queue.

**YES:* Jobs using this job description are to be held when they are put on the job queue.

DATE Parameter: Specifies the date that is to be assigned to the job that uses this job description when the job is initiated.

*SYSVAL: The value in the QDATE system value at the time that the job is initiated is to be used as the job date.

job-date: Enter the value that is to be used as the job date for the job being initiated; the format that is currently specified for the system value QDATFMT must be used. (See the *CPF Programmer's Guide* for the QDATFMT system value.)

SWS Parameter: Specifies the initial settings for a group of eight job switches used by jobs that use this job description. Only zeros (off) and ones (on) can be used. These switches can be set or tested in a CL program and used to control the flow of the program. For example, if a certain switch is on, another program could be called. The job switches may also be valid in other HLL programs.

00000000: The initial setting for the job switches is to be all zeros for jobs that use this job description.

switch-settings: Enter any combination of eight zeros and ones that is to be used as the initial switch setting for jobs using this job description.

PUBAUT Parameter: Specifies what authority for the job description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the job description.

**ALL:* The public has complete authority for the job description.

**NONE:* The public cannot use the job description.

TEXT Parameter: Lets the user enter text that briefly describes the job description. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTJOBD  JOBD(INT4) USER(*RQD) RTGDTA(QCMDI) +
      TEXT('Interactive #4 job description +
      for Department 127')
```

This command creates a job description named INT4 in the general purpose library (QGPL). According to the text description, this job description is for interactive jobs and is to be used by Department 127. When the user signs on, he must enter his password. The characters QCMDI are used as routing data to be compared with the routing table of the subsystem under which the job is to be run.

```
CRTJOBD  JOBD(BATCH3) USER(*RQD) +
      JOBQ(NIGHTQ) JOBPTY(4) OUTPTY(4) +
      RTGDTA(QCMDB) TEXT('Batch #3 job +
      description for high priority night work')
```

This command creates a job description named BATCH3 in the general purpose library (QGPL). The jobs using this description are placed on the job queue NIGHTQ. The priority for jobs using this description and their spooled output is 4. QCMDB is the routing data that is to be compared with entries in the routing table of the subsystem in which the job is to be executed.

```
CRTJOBD  JOBD(PAYWK) USER(QPGMR) RTGDTA(QCMDB) +
      RQSDTA('CALL PAY025 PARM(WEEKLY UNION)')
```

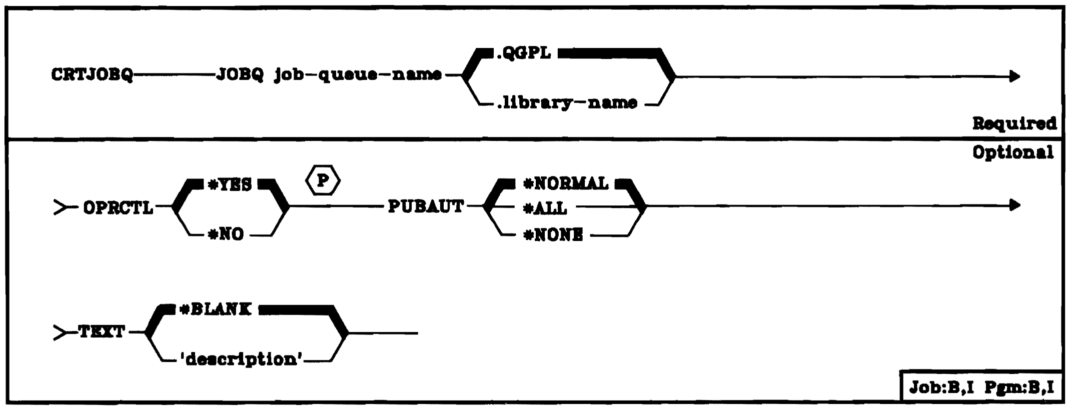
This command creates a job description named PAYWK in the general purpose library (QGPL). Jobs using this job description execute under the IBM-supplied user profile for the programmer, QPGMR. The routing data QCMDB is to be compared with entries in the routing table of the subsystem in which the job is to be run. The request data passed to the command processing program is a CALL command that names the application program to be executed and passes a parameter to it.

CRTJOBQ (Create Job Queue) Command

The Create Job Queue (CRTJOBQ) command creates a new job queue. A job queue contains entries for jobs that are to be processed by the system. To add this job queue to the subsystem, use the Add Job Queue Entry (ADDJOBQE) command.

- Submit Job (SBMJOB)
- Submit Card Jobs (SBMCRDJOB)
- Submit Data Base Jobs (SBMDBJOB)
- Submit Diskette Jobs (SBMDKTJOB)
- Transfer Job (TFRJOB)

To add an entry for this job queue to a subsystem description, use the Add Job Queue Entry (ADDJOBQE) command.



JOBQ Parameter: Specifies the qualified name of the job queue being created. (If no library qualifier is given, the queue is placed in QGPL.)

OPRCTL Parameter: Specifies whether a user who has job control authority is allowed to control this job queue and manipulate the job entries on the queue. A user has job control authority if SPCAUT(*JOBCTL) is specified in his user profile.

YES: A user with job control authority can control the queue and manipulate the entries on the queue.

NO: This queue and its entries cannot be controlled by anyone except the queue's owner.

PUBAUT Parameter: Specifies what authority for the job queue is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only read and add rights for the queue. Any user can put job entries on the job queue and display all entries on the queue.

**ALL:* The public has complete authority for the queue.

**NONE:* The public has no authority for the queue.

TEXT Parameter: Lets the user enter text that briefly describes the job queue. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTJOBQ  JOBQ(DEPTA) PUBAUT(*NONE) +
      TEXT('Special queue for Dept A jobs')
```

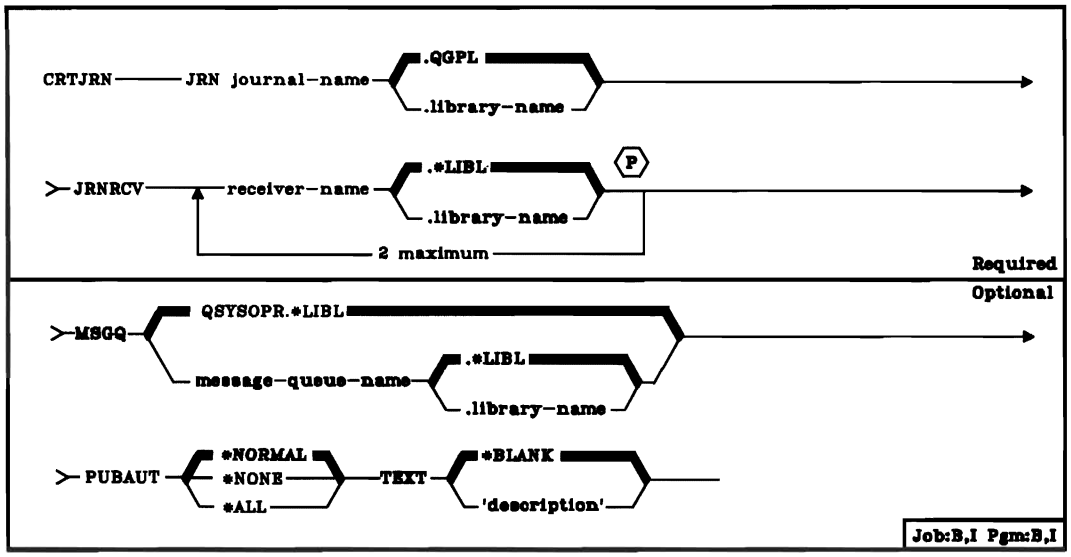
This command creates a job queue named DEPTA and puts it in the QGPL library. Because PUBAUT(*NONE) is specified and OPRCTL(*YES) is assumed, the job queue can be used and controlled only by the user who created the queue and by users with job control rights (*JOBCTL).

CRTJRN (Create Journal) Command

The Create Journal (CRTJRN) command creates a journal with the specified attributes, and attaches the specified receivers to the journal. Once a journal has been created, file changes may be journaled to it.

Restriction: The receivers specified must have been created before the execution of this command and they must be empty (that is, the receivers must not have been previously attached to any journal).

A journal cannot be created in library QTEMP.



JRN Parameter: Specifies the qualified name of the journal that is being created. (If a library name is not specified, the journal is placed in library QGPL).

JRNRCV Parameter: Specifies the qualified names of the journal receivers to be attached to the specified journal. (If no library qualifier is given, *LIBL is used to find the journal receiver.) The journal receivers must not have been previously attached to any journal. A maximum of 2 journal receivers may be specified.

MSGQ Parameter: Specifies the message queue to be associated with this journal. The message issued when a journal receiver's storage limit (threshold) is exceeded is sent to this message queue.

QSYSOPR.*LIBL: The message will be sent to the QSYSOPR message queue.

message-queue-name: Enter the qualified name of the message queue to which the message will be sent. If this message queue is not available when the journal receiver threshold is exceeded for a journal receiver, the message will be sent to the QSYSOPR message queue. To set the threshold value, refer to the CRTJRNRCV command.

PUBAUT Parameter: Specifies what authority for the journal is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

Note: No specific authority is required to journal physical file member changes once journaling has been started for that file.

*NORMAL: The public has operational, read, add, and update authority for the journal.

**NONE:* The public has no specific authority for the journal.

**ALL:* The public has complete authority for the journal.

TEXT Parameter: Specifies the user-defined text that briefly describes the journal. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

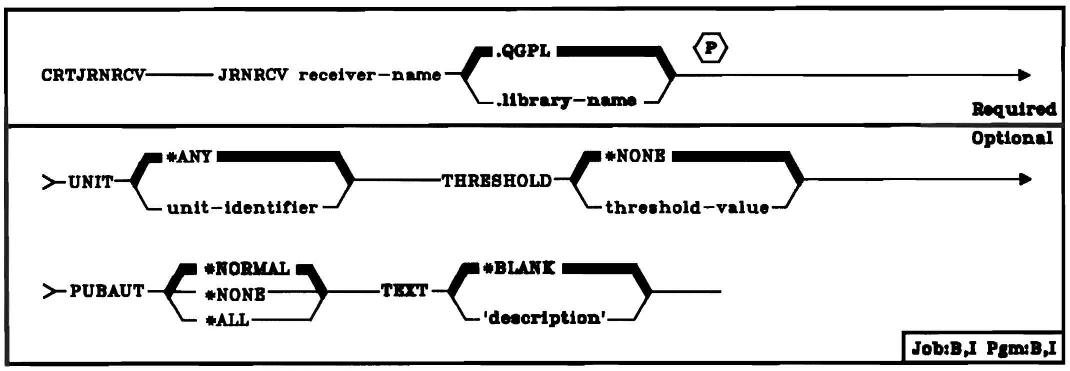
```
CRTJRN  JRN(JRNLA.MYLIB) JRNRCV(RCV01.MYLIB RCV01A.MYLIB)
```

This command causes a journal named JRNLA to be created in library MYLIB. The public will have normal authority for the journal. Two journal receivers (named RCV01 and RCV01A in library MYLIB) are attached to journal JRNLA.

CRTJRNRCV (Create Journal Receiver) Command

The Create Journal Receiver (CRTJRNRCV) command creates a journal receiver. Once a journal receiver has been attached to a journal (with the CRTJRN or CHGJRN command), journal entries may be placed on it. A preferred storage unit and a storage space threshold value can be specified for the journal receiver. When the size of the journal receiver exceeds the size specified, a message is sent to the message queue associated with the journal (the operation will not terminate).

Restriction: A journal receiver cannot be created in library QTEMP.



JRNRCV Parameter: Specifies the qualified name of the journal receiver that is being created. (If a library name is not specified, the journal receiver is placed in library QGPL).

UNIT Parameter: Specifies the unit identifier of the auxiliary storage unit on which the system will attempt to allocate the storage space for the journal receiver. This includes the initial allocation for the receiver and any later extensions. If the system cannot allocate the storage space on the specified unit, it allocates the space on any available unit. The journal receiver is entirely usable in all cases.

*ANY: The storage space for the journal receiver can be allocated on any available auxiliary storage unit.

unit-identifier: Enter a value of 1 through 14 to specify the identifier of the auxiliary storage unit on which you prefer to have the storage space of the receiver allocated. Valid values depend on how many storage units are on the system, and on their types (62PC disk and 3370 disk).

The following chart shows the valid unit identifiers for each device type and unit:

Device Type	Unit	Unit Identifier
62PC	1-6	1-6
3370	Module 1, actuator 1	7
	Module 1, actuator 2	8
	Module 2, actuator 1	9
	Module 2, actuator 2	10
	Module 3, actuator 1	11
	Module 3, actuator 2	12
	Module 4, actuator 1	13
	Module 4, actuator 2	14

Note: These identifiers remain the same for systems that have 3370 devices and fewer than six 62PC devices.

The system attempts to make all space allocations on the unit specified. If it cannot, either because that unit is full or because an invalid identifier was specified, it allocates the remainder of the space on any available unit.

THRESHOLD Parameter: Specifies a storage space threshold value (in K-bytes) for the journal receiver. If the threshold value is exceeded during journaling, a message (CPF7099) is sent to the message queue designated on the CRTJRN or CHGJRN command. No action is taken by the system other than sending the message; you may want to take some action, however, such as issuing a CHGJRN command or saving the receivers.

*NONE: No threshold value is specified (no message will be sent).

threshold-value: Enter a value between 1 and 1,920,000, which is to indicate K-bytes of storage; for example, an entry of 1000 specifies 1024000 bytes. When the size of the space for the journal receiver exceeds the size specified by this value, a message will be sent to the designated message queue and journaling will continue.

Note: If the specified threshold value is less than 9, the message will be sent when the journal receiver is attached to a journal (with the CRTJRN or CHGJRN command).

PUBAUT Parameter: Specifies what authority for the journal receiver is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

Note: No specific authority is required to journal physical file member changes once journaling has been started for that file.

NORMAL: The public has operational and read authority for the journal receiver.

**NONE:* The public has no specific authority for the journal receiver.

**ALL:* The public has complete authority for the journal receiver.

TEXT Parameter: Specifies the user-defined text that briefly describes the journal receiver. (For an expanded description of the TEXT parameter, see Appendix A.)

BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTJRNRCV  JRNRCV(JRNRCLA.MYLIB) THRESHOLD(1000) PUBAUT(*ALL)
        TEXT('RECEIVER FOR WEEK 37')
```

This command causes a journal receiver named JRNRCLA to be created in library MYLIB. When the size of JRNRCLA exceeds 1000K (1,024,000) bytes, a message will be sent to the message queue associated with the journal. The public has complete authority for the journal receiver.

CRTLF (Create Logical File) Command

The Create Logical File (CRTLF) command creates a logical file in the System/38 data base. The logical file is created from the file description parameters in this CRTLF command and from the previously entered data description specifications (DDS) that contain the source description of the logical file. (To override the attributes of the logical file after it has been created, use the Override Data Base File (OVRDBF) command before the file is opened.)

A logical file is a data base file through which data from one or more physical files can be accessed in a format and organization that is different from the physical representation of the data. Logical files do not actually contain data. A logical file is structured over one or more physical files, which are referred to as *based-on physical files*. The logical file cannot be created unless all the physical file(s) specified in the logical file description already exist; however, the physical files do not have to contain data. The following attributes are contained in the logical file description:

- The record format specifications used by the file. A logical file can have more than one record format. The record formats can describe records having different attributes. Each logical file record format can describe records from one or more based-on physical files. There will usually be one logical file record format specified for each physical file record format that is accessed. Fields from more than one physical file cannot be combined in one logical file record format.

- The access path specifications used to process data from the file. A logical file can be in arrival sequence or in keyed sequence and can have many optional access path specifications.

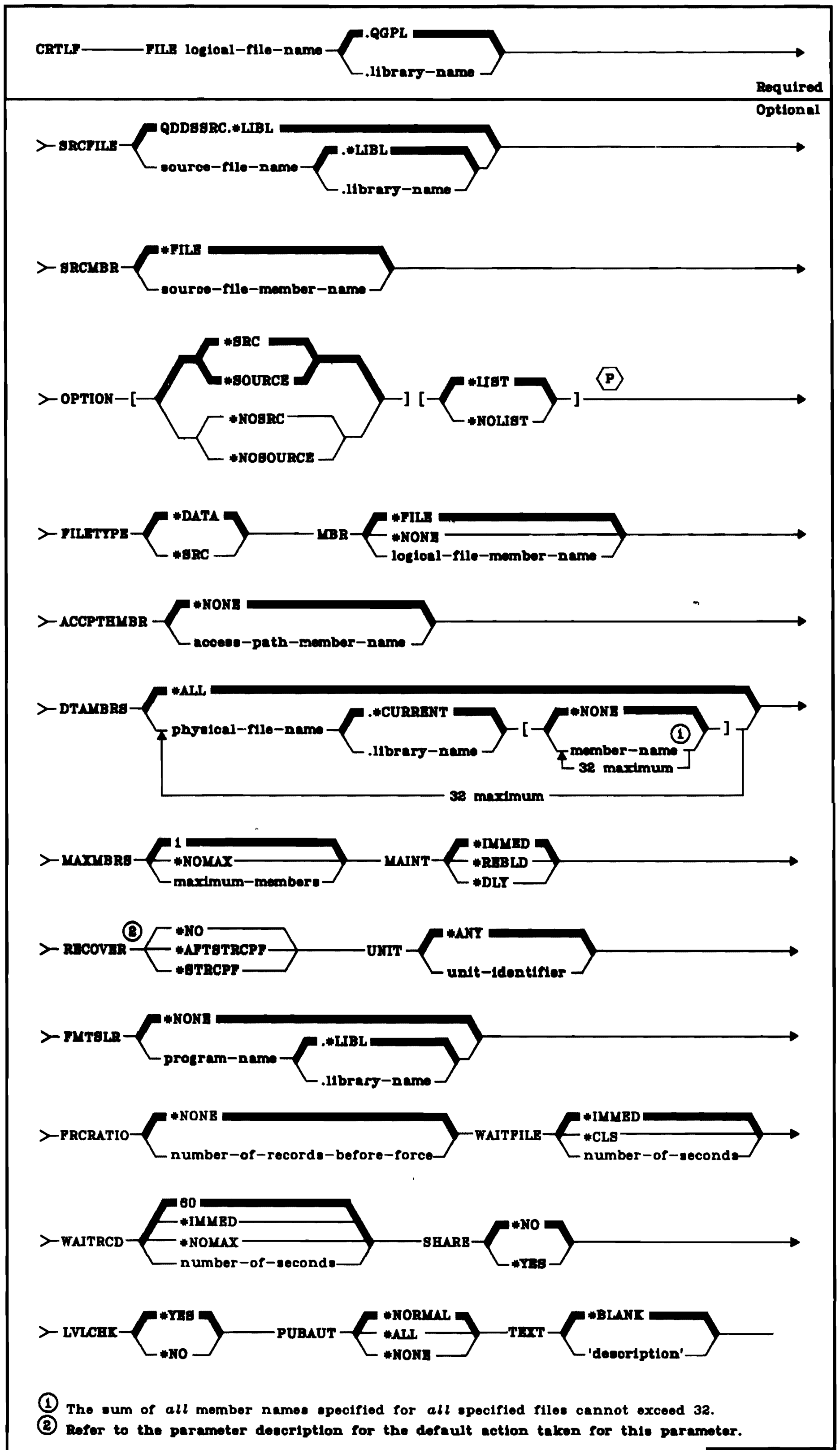
- The data association specifications for the file. These specifications identify the physical files that contain data used by the logical file.

Each logical file can have one or more members; each member, except (optionally) the first member, is added separately, but has many of the same attributes of the entire file. Each member in the logical file can have different based-on physical file members for data; it has a separate organization of data. Each member within the file has its own access path, but the *type* of access path (keyed or arrival sequence) and maintenance rule is the same for every member.

Each member within the logical file can access records that have one or more record formats. The logical file member is really an index to the records located in the physical files specified in the DDS source input to the CRTLF command. The member can access records from all based-on physical files specified by PFILE keywords in the file's DDS, or a subset of them. The record format combinations can be different for each logical file member. But each logical record format can be based only on existing record format(s) in the based-on physical file. The format can be the same as, or a subset of, the based-on physical file format.

A logical file without any members can exist, but data operations cannot be performed through the logical file until a member is added. To add a member, either specify one in the MBR parameter of this command or use the Add Logical File Member (ADDLFM) command. (The default is to create one member with the same name as the file.)

Restrictions: To create a logical file over one or more physical files, you must have object management rights and operational rights for each of the based-on files (specified by PFILE keywords in DDS). And, if the logical file is to share the *keyed* sequence access path of another logical or physical file (specified by ACCPTH in DDS), you must have operational rights for that file. Similar restrictions also apply to based-on physical file members when a member is to be added to the logical file (by the ADDLFM command) after it has been created.



(1) The sum of *all* member names specified for *all* specified files cannot exceed 32.
(2) Refer to the parameter description for the default action taken for this parameter.

Job:B,I Pgm:B,I

FILE Parameter: Specifies the qualified name by which the logical file being created will be known. If no library qualifier is given, the logical file is stored in QGPL. (If the file is to be used in an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

SRCFILE Parameter: Specifies the name of the source file containing the DDS to be used to create the logical file. The source file contains the specifications that describe the record format(s) and their fields, and the access paths for the file and its members. (For the specifications that can be made in DDS, refer to the *CPF Reference Manual—DDS*.)

QDDSSRC: The IBM-supplied DDS source file named QDDSSRC in the QGPL library contains the source descriptions to be used to create the logical file. QDDSSRC can contain source descriptions for many files; each member of QDDSSRC contains the source description of one physical, logical, or device file. (When shipped, QDDSSRC contains no descriptions.) (If no library qualifier is specified, *LIBL is used to find the source file.)

qualified-source-file-name: Enter the qualified name of the source file that contains the DDS to be used to create the logical file. (If no library qualifier is given, *LIBL is used to find the source file.)

SRCMBR Parameter: Specifies the name of the source file member that contains the DDS for the logical file being created; the member is in the source file specified in the SRCFILE parameter (or its default, QDDSSRC). The SRCMBR parameter is valid only if SRCFILE specifies the name of a data base file. If not specified, the member name is the same as the name of the logical file being created; the default value *FILE implies that the name of the logical file being created is to be used. A member name must be specified when the source file member to be processed does not have the same name as the logical file being created.

*FILE: The source file member name is the same as the name of the logical file being created.

source-file-member-name: Enter the name of the member in the source file specified by SRCFILE to be used to create the logical file.

OPTION Parameter: Specifies the type of output listing to be produced when the file is created.

*SRC or **SOURCE:* A listing of the source statements used to create the file, and of any errors that occur, is to be generated.

**NOSRC* or **NOSOURCE:* No listing of the source statements is to be generated unless errors are detected. If errors are detected, they are listed along with the record format containing the error.

*LIST: An expanded source listing is to be generated, showing a detailed list of the file specifications that result from the source statements and references to other file descriptions. This listing shows file, field, key, and select/omit keywords and attributes.

**NOLIST:* No expanded source listing is to be generated.

FILETYPE Parameter: Specifies whether each member of the logical file being created is to contain data records, or is to contain source records (statements) for a program or another file. The file could contain, for example, RPG source statements for an RPG program or DDS source statements for another physical, logical, or device file. (For an expanded description of the FILETYPE parameter, see Appendix A.)

*DATA: The logical file is to contain data records.

**SRC:* The logical file is to contain source records. (Each source record must have at least three fields defined by DDS.)

MBR Parameter: Specifies the name of the logical file member (if a member is to exist immediately) to be added when the logical file is created.

*FILE: The member being added is to have the same name as that of the logical file that contains the member (specified in the FILE parameter).

**NONE:* No member is to be added when the file is created.

logical-file-member-name: Enter the name of the member that is to be added when the logical file is created.

ACCPTHMBR Parameter: Specifies, for the logical file member being added, the name of the member of the existing file that describes the access path to be shared with the logical file member. This parameter is required and can be used only when a logical file member is to be added at the time the file is created and it is to share an access path. The parameter is not valid if MBR(*NONE) is specified.

For information on access path sharing, refer to the description of the ACCPTH keyword in the *CPF Reference Manual—DDS*. The ACCPTH keyword can be specified in the logical file source description.

*NONE: The logical file member being added (if there is one) is not to share the access path of another file member.

access-path-member-name: Enter the name of the member of the file that describes the access path to be shared with the member being added to the logical file. The name of the file is specified in the logical file data description specifications, on the ACCPTH keyword.

DTAMBRS Parameter: Specifies the names of the physical files and members that contain the data to be associated with the logical file member being added when this logical file is created. The scope of the logical file member can contain all of the physical files and members that the logical file itself contains, specified by DTAMBRS(*ALL); or the member can contain a subset of the total files and members, specified by DTAMBRS(qualified-file-name(s) [member-name(s)]). DTAMBRS cannot be specified if MBR(*NONE) is specified.

In addition, the total of all member names cannot exceed 32; that is, *all* of the member names specified for all of the files specified cannot be greater than 32. For example, one file can specify 32 members, two files can each have 16 members, or 32 files can each have one member specified.

*ALL: If no access path is shared, the scope of the logical file member being added is to be the same as that for the entire logical file. That is, the data to be associated with the member is in all the physical files and members (that exist when this command is entered) used by the logical file. The files are specified by the PFILE keyword in the DDS source file named in the SRCFILE and SRCMBR parameters in this command.

If *ALL is specified (or is the default) and the logical file is to share an access path (not data) with an existing physical or logical file, the data for the logical file member is the same as the data associated with the member specified by the ACCPTHMBR parameter; that is, the same based-on physical file(s) and member(s) are used.

qualified-physical-file-name [member-name]: Enter the names of one or more physical files and their members that contain the data to be accessed by the logical file member being added. Each entry for a physical file in the PFILE keyword in DDS should have a corresponding entry in the DTAMBRS parameter. Also, each physical file specified in the DTAMBRS parameter must correspond to one of the physical files specified by the PFILE parameter when the logical file was created. If no member name is specified for a physical file that is specified, *NONE is assumed and the logical file scope list or the based-on member's scope list is bypassed. (For more details, refer to *Additional Considerations* at the end of this command description.)

A maximum of 32 qualified physical file names and physical file member names can be specified. Also, the total of *all* members cannot exceed 32; that is, all of the member names specified for all of the files specified cannot be greater than 32. For example, one file can specify 32 members, two files can each have 16 members, or 32 files can each have one member specified.

When the file is created, the DDS PFILE keyword is used to specify physical file names, and, optionally, the library qualifiers of the physical files being associated with the logical file. If a library qualifier is not specified, *LIBL is used to find the physical file when the logical file is created. (The physical file and the library in which it is stored are saved in the description of the logical file when the logical file is entered.) When members are added to the file, each physical file name specified in the DTAMBRS parameter can be optionally qualified by the name of the library; however, the library name must be specified only if the logical file is based on more than one physical file of the same name, as defined in the PFILE keyword. If a library name is not specified for a physical file, the current library name (*CURRENT) for the specified file is determined from the qualified file name saved in the description of the logical file (not the current *LIBL library list).

The following examples show the syntax for specifying single and multiple members for single and multiple physical files. In the examples, the abbreviation PF represents a physical file name, LIB represents a library qualifier, and M represents a member name. Physical file names need only be qualified if the PFILE Keyword in the DDS specifies multiple physical files of the same name.

```
Single physical file and member:
   DTAMBRS((PFA M1))
Single file with multiple members:
   DTAMBRS((PFA (M1 M2 M3)))
Multiple files with single members and no members:
   DTAMBRS((PFA M1) (PFB M4) (PFE.NONE))
Multiple files with multiple members:
   DTAMBRS((PFA (M1 M3 M4)) (PFB (M1 M2 M4)))
Multiple files with the same name in different libraries:
   DTAMBRS(PFA.LIBX M1) (PFA.LIBY (M1 M2)))
Multiple files with the same name in the same library:
   DTAMBRS((PFA.LIBX M1) (PFA.LIBX M1))
```

As shown in the preceding example, each physical file specified in the PFILE keyword in the DDS should have a corresponding entry in the DTAMBRS parameter, even though it may mean specifying the same qualified physical file and member many times.

When more than one physical file member is specified for a physical file, specify the member names in the order in which records are retrieved when duplicate key values occur across those members.

MAXMBRS Parameter: Specifies the maximum number of members that the logical file being created can have at any time. (This value cannot be changed after the file is created.)

1: Only one member can be contained in the file.

**NOMAX:* No maximum is specified for the number of members; the system maximum of 32 767 members per file is used.

maximum-members: Enter the value for the maximum number of members that the logical file can have. A value of 1 through 32767 is valid.

MAINT Parameter: Specifies, for files with keyed sequence access paths only, the type of access path maintenance to be used for every member of the logical file. This parameter is not valid for files that have arrival sequence access paths.

*IMMED: The access path is to be continuously (immediately) maintained for each logical file member. The path is updated each time a record is changed, added, or deleted in a physical file member contained in the scope of that logical file member regardless of whether the logical member is open or closed. This means that, for *every* file that is continuously maintained, the access path of each one is immediately updated—whether it is open or closed. *IMMED must be specified for all files requiring unique keys to ensure uniqueness in all inserts and updates.

**DLY:* The maintenance of the access path is to be delayed until the logical file member is opened for use. Then the access path is updated only for records that have been added, deleted, or updated since the file was last opened. (While the file is *open,* all changes made to based-on file members are immediately reflected in the access paths of the opened file's own members, no matter what is specified for MAINT.) To prevent a lengthy rebuild time when the file is opened, *DLY should be specified only when the number of changes to the access path between successive opens are small; that is, when the file is opened frequently or when the key fields in records for this access path change infrequently. *DLY is not valid for access paths that require unique key values.

If the number of changes saved reaches approximately 10 percent of the access path size, the system will stop saving changes and the access path will be completely rebuilt the next time the file is opened.

**REBLD:* The access path is to be completely rebuilt when a logical file member is opened during program execution. The access path is continuously maintained within that member until the member is closed; the access path is then destroyed. *REBLD is not valid for access paths that require unique key values.

RECOVER Parameter: Specifies, for files having immediate maintenance on their access paths, when recovery processing of the file is to be performed after a system failure has occurred while the access path was being changed. This parameter is valid only if a keyed access path is used.

The access path having *immediate maintenance* can be rebuilt during start CPF (before any user can execute a job), after start CPF has finished (during concurrent job execution), or when the file is next opened. While the access path is being rebuilt, the file cannot be used by any job.

The access path having *rebuild maintenance* will be rebuilt the next time its file is opened, the time that it normally is rebuilt.

**NO:* The access path of the file is not to be rebuilt. The file's access path is rebuilt when the file is next opened if it has rebuild maintenance. *NO is the default for all files that do not require unique keys.

**AFTSTRCPF:* The file is to have its access path rebuilt after the start CPF operation has been completed. This option allows other jobs not using this file to begin processing immediately after the CPF has been started. If a job tries to allocate the file while its access path is being rebuilt, a file open exception occurs if the specified wait time for the file is exceeded. *AFTSTRCPF is the default for files that require unique keys.

**STRCPF:* The file is to have its access path rebuilt during the start CPF operation. This ensures that the file's access path will be rebuilt before the first user program tries to use it; however, no jobs can begin execution until after all files that specify RECOVER(*STRCPF) have their access paths rebuilt.

UNIT Parameter: Specifies, if the user prefers that a file be stored on a specific unit, the unit identifier of the auxiliary storage unit on which the system will attempt to allocate the storage space for the file and for all its members and their keyed sequence access paths. This includes the initial allocation for each member and any additional extensions needed later. If the system cannot allocate the storage space on the specified unit, it allocates the space on any available unit and sends a message to the job log. In any case, the file is entirely usable.

*ANY: The storage space for each member can be allocated on any available auxiliary storage unit.

unit-identifier: Enter a valid value of 1 through 14 to specify the identifier of the auxiliary storage unit on which you prefer to have the storage space of all members allocated. The values that are valid depend on how many storage units are on the system, and on their types (62PC disk and 3370 disk). Refer to the chart in the CRTPF command description, UNIT parameter, for the type and unit that correspond to the unit identifiers.

The system attempts to make all space allocations on the unit specified. If it cannot, either because that unit is full or because an invalid identifier was specified, it allocates the remainder of the space on any available unit and sends a message to the job log.

FMTSLR Parameter: Specifies the name of a record format selector program that is to be called when the logical file member contains more than one logical record format. The user-written selector program is called when a record is to be inserted into the data base file and a record format name is not included in the HLL program. The selector program receives the record as input, determines the record format to be used, and returns it to the data base. This program must perform this function for every member in the logical file that has more than one record format, unless the HLL program itself specifies the record format name. (More information about the use of format selector programs is contained in the *CPF Programmer's Guide*.)

This parameter is not valid if the logical file has only one record format.

*NONE: There is no selector program for this logical file. The file cannot have more than one logical record format, or the HLL program itself must specify the record format name.

qualified-program-name: Enter the qualified name of the format selector program to be called when a record is to be inserted into a member having more than one format. The selector program name can be optionally qualified by the name of the library in which the program is stored. (If no library qualifier is given, *LIBL is used to find the program.)

A program specified as the format selector program cannot be created with USRPRF(*OWNER) specified in its create program command.

FRCRATIO Parameter: The force write ratio parameter specifies the number of inserted or updated records that are processed before they are forced into auxiliary (permanent) storage. (For an expanded description of the FRCRATIO parameter see Appendix A.)

For example, if the force ratios of three physical files are 2, 6, and 8, the logical file force ratio that is based upon these three physical files must be as restrictive as the least of them; that is 2 in this case. Two would be used even if FRCRATIO is not specified. Thus, each time a program updates two records in the logical file (regardless of which based-on physical files are used), those changes are forced into permanent storage.

If a physical file associated with this logical file is being journaled, a large force write ratio or *NONE may be specified. Refer to the *CPF Programmer's Guide* for more information on the Journal Management Facility.

*NONE: There is no specified force ratio; the system determines when the records are written in auxiliary storage.

number-of-records-before-force: Enter the number of new or changed records that are processed before they are explicitly forced into auxiliary storage.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

WAITRCD Parameter: Specifies the number of seconds that the program is to wait for a record that is to be updated or deleted. If the record cannot be allocated in the specified wait time, an error message is sent to the program.

60: The program is to wait for 60 seconds.

**IMMED:* The program is not to wait; when a record is locked, an immediate allocation of the record is required.

**NOMAX:* The wait time will be the maximum allowed by the system (32 767 seconds).

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the logical file member can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used. This parameter is not valid if a member is not being added when the logical file is created.

When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record. A write operation produces the next output record.

*NO: An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the record format identifiers are to be level checked to verify that the current record format identifier is the same as that specified in the program that opens the logical file. This value can be overridden on the OVRDBF command at execution time.

*YES: The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not match, an error message is sent to the program requesting the open, and the file is not opened.

**NO:* The level identifiers are not to be checked when the file is opened.

PUBAUT Parameter: Specifies what authority for the logical file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. To use the logical file, the user profile must have the appropriate rights to use the based-on physical files. (For an expanded description of the PUBAUT parameter, see Appendix A.)

Note: Because data rights do not apply to a logical file, they are not included with the rights given by *NORMAL and *ALL.

*NORMAL: The public has only operational rights for the logical file.

**ALL:* The public has complete authority for the logical file.

**NONE:* The public cannot use the logical file.

TEXT Parameter: Lets the user enter text that briefly describes the logical file. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTLF  FILE(STOCKCTL.INVEN) SRCFILE(STKLFSRC.SRCLIB) +
         MBR(*NONE)
```

This command creates a logical file named STOCKCTL, which is to be stored in the INVEN library. The source descriptions in the source file STKLFSRC in the SRCLIB library are used to create the logical file. The file is created without any members (*NONE was specified), and only one member can be added later (because one member is the default). Also, by default, the logical file will access the data contained in the physical files specified in the DDS source file used to create this logical file. (For the CRTLF command to complete successfully, the user must have object management authority for all specified based-on physical files.)

```
CRTLF  FILE(PAYCODESEQ.PAYLIB) SRCFILE(PAYTXSRC.PAYLIB) +
     DTAMBRS(PAYTRANS FIRSTQTR) UNIT(02) PUBAUT(*NONE) +
     TEXT('Pay taxes in code sequence')
```

This command creates a logical file and logical file member, both named PAYCODESEQ, to be stored in the PAYLIB library. The file and its member are created from the PAYTXSRC source file that is in the same library. The user prefers that the logical file be stored on auxiliary storage unit 02. The logical file member will access the data contained in the FIRSTQTR member of the physical file PAYTRANS. The logical file is to be secured for the private use of the owner. The owner must have object management authority for the PAYTRANS file to create the member.

Additional Considerations

This section supplies additional information for coding the DTAMBRS parameter when physical file names and member names are to be specified.

Non-Shared Access Paths

The following considerations apply when an access path is *not* to be shared:

- If the name of the library in which the physical file is stored is not specified, the name of the library is determined from the logical file description (the library name may have been specified in DDS. The library name should be specified if the logical file is based on more than one physical file of the same name (for example, PF1 in LIB1 and PF1 in LIB2).

- When more than one physical file member is specified for a physical file, the member names are specified in the order in which records are to be presented when a duplicate key value occurs across those members. If multiple members from one physical file are specified, add operations are not possible for that record format from high-level language programs.

- The logical file description contains a scope list of the based-on physical files associated with the logical file. The scope list contains the content and order of the based-on physical files. This list can be displayed by the DSPFD (Display File Description) command if TYPE(*ACCPTH) is specified. If a based-on physical file is used with more than one record format, the DTAMBRS file parameters are order dependent.
  - The file parameters, for the based-on files used with more than one record format, must be specified in the same order as they appear in the logical file scope list.
  - If the user does not want to use a file in the logical file's scope list, that file name must be specified without member names for the corresponding entry in the logical file scope list.

  For example, assume that two record formats (FMT1 and FMT2) in this logical file are based on one physical file (PFA) and FMT1 is specified before FMT2 in the DDS. If the logical file member being added will use only FMT2 from member 2 (M2) of the physical file, the following DTAMBRS parameter would be specified: DTAMBRS((PFA)(PFA M2))

Shared Access Paths

The following considerations apply when an access path *is* to be shared:

- The based-on file member must be specified in the ACCPTHMBR parameter. (The based-on member is the member of the file specified by the ACCPTH keyword in DDS whose access path is to be shared.)

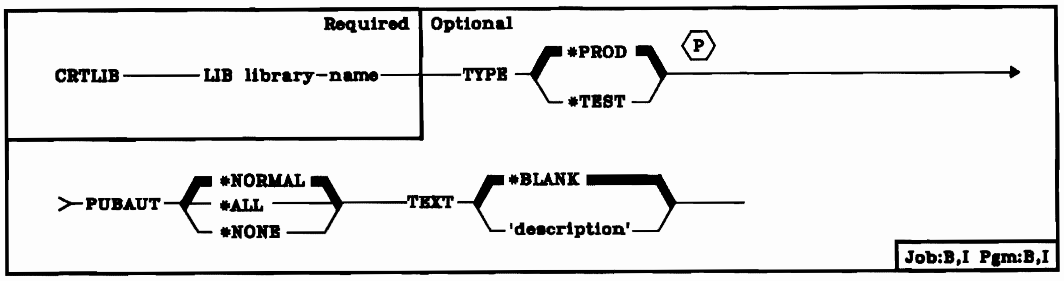
- The file(s) and member(s) specified in the DTAMBRS parameter must also exist in the list of data members on which the existing ACCPTHMBR member is based. This list is called the member scope list.

- If the name of the library in which the physical file is stored is not specified, the unqualified file name and the specified member name are used to search for a matching entry in the based-on member's scope list. The library name must be specified if more than one physical file with the same file name and the same member name are included in the based-on member's scope list. The member scope list contains the content and order of the based-on physical files and members. This list can be displayed by the Display File Description (DSPFD) command if TYPE(*ACCPTH) is specified.

- If a based-on physical file and member appear more than once in the member scope list, the DTAMBRS file parameters are order dependent.
  - The file parameters, for the file member appearing more than once in the based-on member, must be specified in the same order as they appear in the member scope list.
  - If the user does not want data to be associated with a particular entry in the member scope list, that file name must be specified without a member name for the corresponding entry in the member scope list.

CRTLIB (Create Library) Command

The Create Library (CRTLIB) command adds a new library to the system. Before any object can be placed into a library, the library must have been created. When the library is created, it is actually stored as part of the internal system. However, although it is itself a library, it appears as though it exists in the QSYS (system) library.



LIB Parameter: Specifies the name of the library to be created.

TYPE Parameter: Specifies the type of library being created.

*PROD: This is to be a production library. Data base files in production libraries cannot be opened for updating if a user is in debug mode and he requested that production libraries be protected. A user can protect all data base files in production libraries by specifying UPDPROD(*NO) on the ENTDBG command to begin testing. However, this protection does not prevent the program from deleting data base files or from changing other objects (such as data areas) in the library.

**TEST:* This is to be a test library. All objects in a test library can be updated during testing, even if special protection is requested for production libraries.

PUBAUT Parameter: Specifies what authority for the library is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational, read, add, update, and delete rights for the library.

**ALL:* The public has complete authority for the library.

**NONE:* The public cannot use the library.

TEXT Parameter: Lets the user enter text that briefly describes the library. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTLIB  LIB(MYLIB) TEXT('My Production Library')
```

The library MYLIB is added to the system. The library is a production library; only the owner has object existence and object management rights for it. But all users have operational, read, add, update, and delete rights for the library because *NORMAL was assumed for PUBAUT. The text 'My Production Library' is displayed whenever the library description for MYLIB is displayed.

```
CRTLIB  LIB(Z) TYPE(*TEST) PUBAUT(*NONE) +
      TEXT('This is a test library')
```

Test library Z is added to the system. Only the owner of Z can use it because no other users have been granted any authority. The specified text ('This is a test library') is displayed whenever the library description for Z is displayed.

CRTLIND (Create Line Description) Command

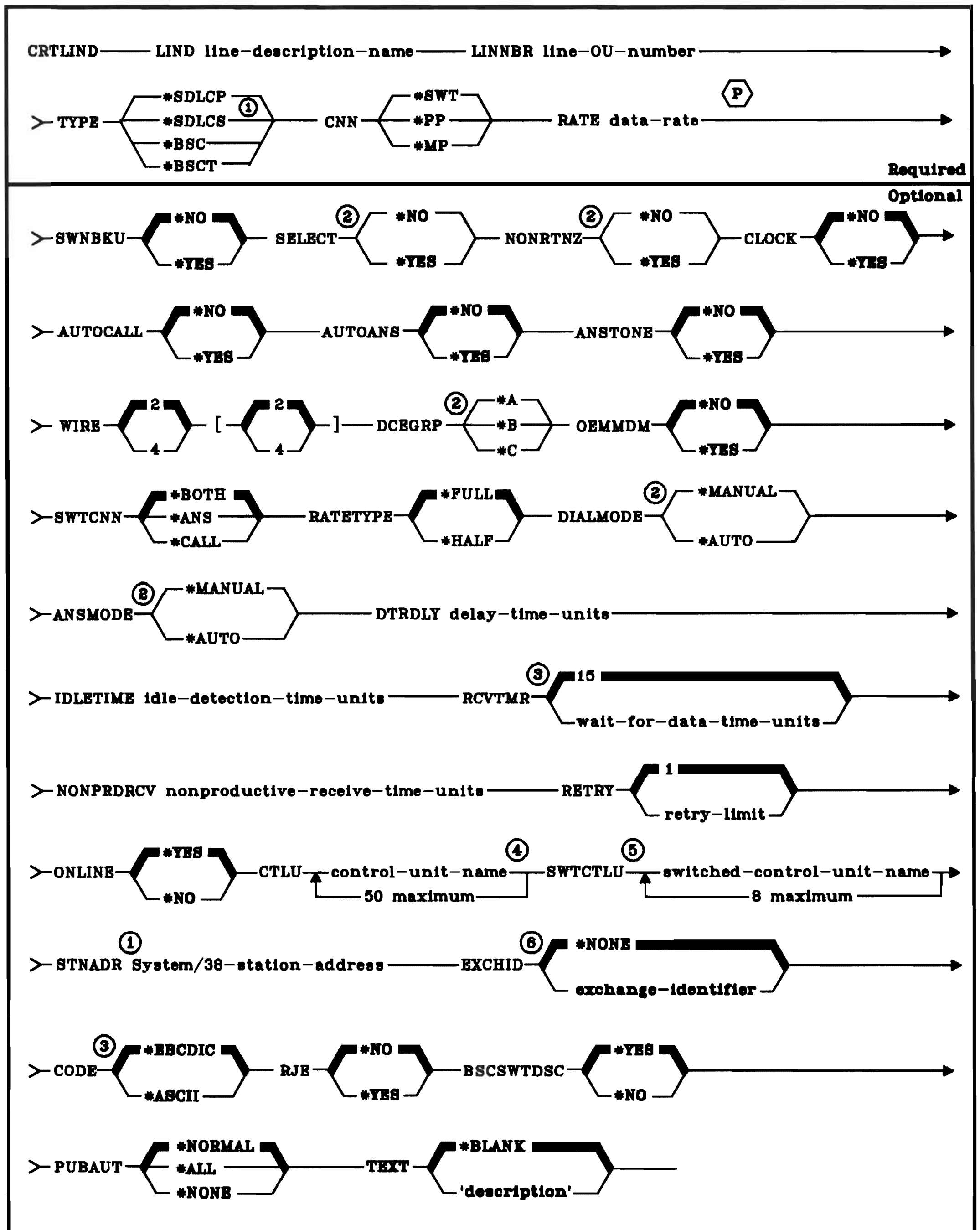
The Create Line Description (CRTLIND) command creates a line description for the specified communications line and describes its features to the system; this includes the description of the modem (and its features) connected to the line. When the line description is created, it is stored as part of the internal system, and it appears as though it exists in the QSYS (system) library.

One CRTLIND command must be entered for each communications line on the system. Up to 10 line descriptions can be created for each line number. Only one line description and its associated network can be varied on at one time. This enables you to change the line description or communications protocol of a particular line number by simply varying the network off, then varying on the preferred network description. Devices directly attached to System/38, such as the MFCU and work stations attached to the work station controller, do not require a corresponding CRTLIND command.

This command should be used to create a line description before the associated control unit descriptions and device descriptions are created for the control units and devices attached to the line. This sequence for creating descriptions is not a required sequence. However, if the descriptions are created out of sequence, any commands referring to names of descriptions not yet created are rejected by the system.

Note: Before using this command, you should refer to the *Guide to Program Product Installation and Device Configuration* for considerations relating to the communications line interfaces, and to the modem characteristics, types, and features. (Appendix C contains many details on the modem types.) This information will help you to properly specify the parameters on the CRTLIND command.

Restrictions: If the CRTLIND command is used to change the name and/or line number of a switched line, the CHGCUD command must be used to incorporate the new attribute(s) in the control unit description of each control unit that is associated with that line and that specifies either SWITCHED(*YES) or SWNBKU(*YES) in its control unit description. (If the address of the line is changed, the CHGCUD command makes the new address association when the new (or unchanged) line name is specified in the LINLST parameter.) The LINLST parameter must specify all the unchanged line names as well as the new name; however, their order can be changed.



① If TYPE(*SDLCS) or TYPE(*BSCT) is specified, an address *must* be specified in the STNADR parameter.
② Refer to the parameter description for the default action taken for this parameter.
③ Valid for TYPE(*BSC) or TYPE(*BSCT) only.
④ Multiple control unit names can be specified only if TYPE(*SDLCP) and CNN(*MP) are also specified. Only one name is valid if TYPE(*SDLCS), TYPE(*BSC) or TYPE(*BSCT) is specified.
⑤ Valid only if TYPE(*BSC) and CNN(*SWT) are specified, or if SWNBKU(*YES) is specified.
⑥ Valid for TYPE(*SDLCS) only.

Job:B,I Pgm:B,I

Table of Valid Parameters by Line Types

Parameter	SDLCP	SDLCS	BSC	BSCT
LIND	R	R	R	R
LINNBR	R	R	R	R
TYPE	R	R	R	R
CNN	R	R	R	R
RATE	R	R	R	R
SWNBKU	O	O	O	I
SELECT[1]	O	O	O	O
NONRTNZ[2]	O	O	I	I
CLOCK	O	O	O	O
AUTOCALL[3,5]	O	O	O	O
AUTOANS[3,5]	O	O	O	O
ANSTONE[4]	O	O	O	O
WIRE[5]	O	O	O	O
DCEGRP	O	O	O	O
OEMMDM	O	O	O	O
SWTCNN	O	O	O	I
RATETYPE[6]	O	O	O	O
DIALMODE[7]	O	O	O	O
ANSMODE[4]	O	O	O	O
DTRDLY	O	O	O	O
IDLETIME	O	O	I	I
RCVTMR	I	I	O	O
NONPRDRCV	O	O	I	I
RETRY	O	O	O	O
ONLINE	O	O	O	O
CTLU	O	O	O	O
SWTCTLU[10]	I	I	O	I
STNADR	I	R	I	R
EXCHID[8]	I	O	I	I
CODE[9]	I	I	O	O
RJE[9]	I	I	O	I
BSCSWTDSC	I	I	O	I
PUBAUT	O	O	O	O
TEXT	O	O	O	O

R = Required O = Optional I = Invalid

[1]Dependent on OEMMDM parameter.
[2]If TYPE(*SDLCP) or TYPE(*SDLCS) is specified, NONRTNZ(*YES) must be specified (default).
[3]Dependent on CNN(*SWT) parameter.
[4]Dependent on AUTOANS(*YES) parameter.
[5]Dependent on SWNBKU(*YES) parameter.
[6]SELECT(*YES) must be specified for RATETYPE(*HALF) selection.
[7]Dependent on AUTOCALL(*YES) parameter.
[8]Must be a valid EBCDIC value or a translatable ASCII value.
[9]CODE(*ASCII) and RJE(*YES) are mutually exclusive.
[10]At least one entry required for switched point to point.

LIND Parameter: Specifies the name of the communication line description that is being created.

LINNBR Parameter: Specifies a value that is to identify to the system the line being described. The value specified is to be the operational unit (OU) number of the line description, and it must correspond to the I/O controller (IOC) being used and to the number of the line connector to which the line is attached. The line connectors for controller 1 are on the back of the 5381 System Unit, and those for controller 2 are on the back of the expansion unit; both sets of connectors are numbered 00 through 03.

Communication I/O Controller 1		**Communication I/O Controller 2**[1]	
Line Number (OU Number)	**Line Position**	**Line Number (OU Number)**	**Line Position**
20	First (00)	60	Fifth (00)
21	Second (01)	61	Sixth (01)
22	Third (02)	62	Seventh (02)
23	Fourth (03)	63	Eighth (03)

[1]If installed.

If the line is moved to another line connector, the line description must be recreated with the new OU number. If AUTOCALL(*YES) is specified, line 23 or line 63 cannot be specified (refer to the AUTOCALL parameter description). Up to 10 line descriptions can be created for each line number, but only one described line can be varied on at one time.

Note: When more than one line description exists for a line number, only one should specify ONLINE(*YES); if multiple line descriptions specify ONLINE(*YES), the system chooses, in alphabetic order, the first line description and varies it on during CPF start up.

TYPE Parameter: Specifies the type of communication line used.

**SDLCP:* The line is a primary synchronous data link control (SDLC) line.

**SDLCS:* The line is a secondary SDLC line that is to be used by a host system to communicate with this System/38. The SDLC link (station) address of System/38 must be specified in the STNADR parameter.

**BSC:* The line is used for point-to-point BSC or RJEF.

**BSCT:* The line is a BSC multipoint tributary line.

CNN Parameter: Specifies the type of line connection used for the line.

**SWT:* This is a switched connection, accomplished by telephone dial-up. *SWT is not valid for TYPE(*BSCT).

**PP:* This is a point-to-point nonswitched connection.

**MP:* This is a multipoint nonswitched connection. This value is not valid for TYPE(*BSC) (and, therefore, RJEF).

The following chart shows which parameters are valid for each of the types of connection. Only those parameters that are dependent on the value specified in the CNN parameter are shown in the chart.

CNN(*SWT)	CNN(*PP)	CNN(*MP)
	SWNBKU	SWNBKU
AUTOCALL		
AUTOANS	AUTOANS[1]	AUTOANS[1]
ANSTONE	ANSTONE[1]	ANSTONE[1]
DCEGRP	DCEGRP[2]	DCEGRP[2]
DIALMODE		
ANSMODE	ANSMODE[1]	ANSMODE[1]
SWTCTLU	SWTCTLU[1]	
	CTLU	CTLU

[1]Valid only if SWNBKU(*YES) is specified.
[2]If SWNBKU(*YES) is specified, *A, *B, or *C (depending on the country and the modem type) can be specified for DCEGRP; otherwise, only *A is valid.

RATE Parameter: Specifies the data rate (speed) for the line. Enter one of the following line speeds: 1200, 2000, 2400, 4800, 7200, 9600, or 56 000 bits per second.

SWNBKU Parameter: Specifies, for nonswitched line modems only, whether the modem has the switched network backup feature. SWNBKU(*YES) is *not* valid if CNN(*SWT) or TYPE(*BSCT) is specified. The backup feature is used to allow the user to bypass a broken nonswitched (leased line) connection by dialing a telephone number to establish a switched connection. The CHGLIND command must be used to actually activate the switched backup feature.

**NO:* The nonswitched line modem does not have the switched backup feature.

**YES:* The nonswitched line modem does have the switched backup feature. To activate the feature when the nonswitched connection is broken, specify ACTSWNBKU(*YES) on both the CHGLIND and CHGCUD commands.

SELECT Parameter: Specifies whether the line has the data rate select function or whether it can operate at full speed only.

**NO:* The line cannot operate at half speed; it can operate at full speed only. If OEMMDM(*YES) is specified, *NO is the default.

**YES:* The line has the data rate select function and can operate at either full or half speed. If OEMMDM(*NO) is specified, *YES is the default.

NONRTNZ Parameter: Specifies, for SDLC lines, whether the data communications equipment on the line requires the NRZI (nonreturn-to-zero-inverted) transmission method. All data communication equipment on the line must use the same transmission method.

**NO:* NRZI is not required for data transmission.

**YES:* NRZI is required for data transmission. If TYPE(*SDLCP) or TYPE(*SDLCS) is specified, *YES is the default for this parameter.

CLOCK Parameter: Specifies whether the clocking function for this line is provided by System/38 or by the data communications equipment (DCE).

*NO: The clocking function for the line is provided by the DCE.

**YES:* The clocking function for the line is provided by System/38. For SDLC, if CLOCK(*YES) is specified, NONRTNZ(*YES) must also be specified. CLOCK(*YES) is not valid for TYPE(*BSCT).

AUTOCALL Parameter: Specifies whether the automatic calling feature is installed. This feature automatically dials the telephone number of the modem of another station to connect System/38 to that station. AUTOCALL(*YES) is valid only if CNN(*SWT) or SWNBKU(*YES) is specified. Each line that has the automatic calling feature uses two line positions (the one specified on the LINNBR parameter, and the next higher number), which reduces the number of communication lines that can be supported.

Note: If you specify AUTOCALL(*YES), you cannot use the next sequential line number or line number 23 or 63.

*NO: The autocall feature is not installed.

**YES:* The autocall feature is installed.

AUTOANS Parameter: Specifies whether the automatic answer modem feature is installed. The autoanswer feature allows incoming calls on the line to be automatically connected to the communication equipment. AUTOANS(*YES) is valid only if CNN(*SWT) or SWNBKU(*YES) is specified.

*NO: The autoanswer feature is not installed.

**YES:* The autoanswer feature is installed. The following parameters in this command, which are associated with the autoanswer feature, can be specified only if AUTOANS(*YES) is specified: ANSMODE, and ANSTONE.

ANSTONE Parameter: Specifies whether System/38 provides the answer-tone signal needed by some autoanswer feature modems. ANSTONE(*YES) is valid only if AUTOANS(*YES) is specified.

*NO: The system does not provide the answer-tone signal.

**YES:* The system provides the answer-tone signal.

WIRE Parameter: Specifies the type of physical connection used for the line. If the switched network backup feature is installed and SWNBKU(*YES) is specified, the type of connection for the backup line can also be specified as the second value. If SWNBKU(*YES) is specified, the defaults are 2-wire for both lines; if SWNBKU(*NO) is specified or assumed, the second default is ignored. If both the normal and the backup lines are to use 4-wire connections, for example, WIRE(4 4) is specified.

2: The physical connection is by a 2-wire link.

4: The physical connection is by a 4-wire link.

DCEGRP Parameter: Specifies the types of modems that can be used on this line. This parameter indicates the modem type (an IBM integrated modem or another supported modem) used on a switched line either for the US and Canada or for all other countries. If DCEGRP is not specified, the default is *C for switched lines and *A for nonswitched lines.

Note: If this is a nonswitched line with the switched network backup feature, then specify the appropriate value for operating the line in switched backup mode.

**A:* If this is a switched line in countries other than the US and Canada, an IBM integrated modem is to be used on the line. If this is a nonswitched line, *A should be specified or assumed for *all* countries.

**B:* For switched lines in countries other than the US and Canada, any supported modem *other than* the IBM integrated modem is to be used on the line.

**C:* For switched lines in the US and Canada, any supported modem, *including* the IBM integrated modem, is to be used on the line.

OEMMDM Parameter: Specifies whether a non-IBM modem is used.

*NO: An IBM modem is used.

**YES:* A non-IBM modem is used.

SWTCNN Parameter: Specifies whether the line is to be used for incoming calls, outgoing calls, or both. This parameter is valid only if CNN(*SWT) or SWNBKU(*YES) is specified.

*BOTH: The line can be used for both incoming and outgoing calls.

**ANS:* The line can be used for incoming calls only.

**CALL:* The line can be used for outgoing calls only.

RATETYPE Parameter: Specifies the speed at which the line operates if the line has the data rate select function. RATETYPE(*HALF) is valid only if SELECT(*YES) is specified.

*FULL: The line is operated at full speed.

**HALF:* The line is operated at half speed.

DIALMODE Parameter: Specifies whether the line connection is to be made manually or automatically. DIALMODE(*AUTO) is valid only if AUTOCALL(*YES) is specified.

**MANUAL:* The line connection is made by the user manually dialing the connection (that is, the called station). If AUTOCALL(*NO) is specified, *MANUAL is the default.

**AUTO:* The line connection is made by the system automatically dialing the called station. If AUTOCALL(*YES) is specified, *AUTO is the default.

ANSMODE Parameter: Specifies how incoming calls to System/38 can be answered (that is, how the switched line connection is to be made through the autoanswer facilities for calls coming from a remote control unit or work station). ANSMODE(*AUTO) is valid only if AUTOANS(*YES) is also specified.

**MANUAL:* The incoming call must be manually answered. If AUTOANS(*NO) is specified, *MANUAL is the default.

**AUTO:* The incoming call is automatically answered by the autoanswer modem feature. If AUTOANS(*YES) is specified, *AUTO is the default.

DTRDLY Parameter: The data terminal ready (DTR) delay parameter specifies the maximum length of time that the system is to pause before ending a command that resets the DTR condition. (The delay time cannot exceed 3 seconds.) Enter a value (0 through 15) that is multiplied by the base unit of 200 milliseconds to determine the maximum delay time before resetting the DTR condition. For most networks, 200 milliseconds (specified here by a 1) is appropriate. If 0 is specified or assumed, a default time of 100 milliseconds is used.

IDLETIME Parameter: Specifies, for any transmission sent by the primary station that requires a response, the maximum time within which the beginning of the secondary station's response must be detected (received). This time should be greater than the sum of the:

- Transmission time to the secondary station
- Processing time of the control unit's response at the secondary station (not including customer program processing time or operator response time)
- Clear-to-send time at the secondary station modem
- Transmission time from the secondary station

Enter a value, 0 through 255, that is multiplied by the base time unit of 53.3 milliseconds to determine the maximum detection time for the secondary station's response (53.3 milliseconds through 13.6 seconds). A recommended minimum time is 2 seconds (specified here by a value of 38). If 0 is specified or assumed, a default time of 500 milliseconds is used. IDLETIME is not valid if TYPE(*SDLCS), TYPE(*BSC), or TYPE(*BSCT) is specified.

For more information on the idle state time and nonproduction receive time considerations, refer to *IBM Synchronous Data Link Control—General Information*, GA27-3093.

RCVTMR Parameter: The receive timer parameter, valid for BSC or BSCT lines only, specifies the amount of time the system will wait for data before a time-out occurs. Measured in 200-millisecond intervals, the timer allows a maximum time-out period of 25.4 seconds (value of 127). For most systems, the default value of 15 (3 seconds) is appropriate.

NONPRDRCV Parameter: The nonproductive receive parameter specifies, for TYPE(*SDLCP) only, the maximum length of time in which to receive an intelligible transmission. The time is specified by a value that is multiplied by the base time unit of 500 milliseconds. The nonproductive receive time is dependent upon the data rate (line speed) specified by the RATE parameter. Use the following table to determine, for a given line speed, the recommended value that should be specified for the NONPRDRCV parameter. (The times given in the last column are the resulting maximum times in which to receive intelligible data. They provide enough time for 5250 devices, which can have a maximum number of 266 bytes transmitted per frame.) This parameter is not valid for TYPE(*BSC).

Primary Line Speed	Recommended Parameter Value[1]	Nonproductive Receive Timer Setting (266 bytes per frame)
600	11	5.5 seconds
1200	6	3.0 seconds
2400	4	2.0 seconds
4800	2	1.0 second
9600	2	1.0 second
56 000	2	1.0 second

[1]If SELECT(*YES) is specified, enter the value for the lowest speed (half speed).

Enter a value, 0 through 255, that is multiplied by the base time unit of 500 milliseconds to determine the maximum time. If 0 is specified or assumed, a default time of 128 seconds is used.

Note: This parameter is ignored for nonswitched secondary lines. For switched secondary or switched network backup secondary lines, specify a parameter value of 60 to 255. A value of 60 will allow a 30-second period of no transmission from the host system to elapse before the switched line is disconnected.

RETRY Parameter: Specifies the maximum number of retries that can be performed when an error occurs before the line (or a station on the line) is considered inoperative. If the retry limit is reached without a successful completion, an error message is sent to the system operator.

Enter a value, 0 through 21, that is to be multiplied by a base number of 1 or 7 to determine the maximum number of retries that can be attempted if necessary. All errors associated with making a switched connection to the line use the base multiplier 1; all other line errors use the base multiplier 7. If 0 is specified, no retries occur.

In no case does the system attempt more than 21 retries. Therefore, a value of 0 through 21 is valid for retrying errors that use the multiplier 1. A value of 0 through 3 is valid for those using the multiplier 7; in this case, any value specified that is greater than 3 is assumed to be 3, and a maximum of 21 retries (3 times 7) can be attempted.

Table of Retry Multipliers

Error	Value
Switched connect retry (BSC)	1
Data line occupied	1
PRESENT NEXT DIGIT inactive after CALL REQUEST set (AUTODIAL)	1
PRESENT NEXT DIGIT active after DIGIT PRESENT reset (AUTODIAL)	1
DISTANT STATION CONNECTED inactive	1
DISTANT STATION CONNECTED inactive and ACR (Abandon Call and Retry)	1
PRESENT NEXT DIGIT inactive after CALL REQUEST set – ACR (Abandon Call and Retry) (AUTODIAL)	1
PRESENT NEXT DIGIT active after DIGIT PRESENT set – ACR (Abandon Call and Retry) (AUTODIAL)	1
PRESENT NEXT DIGIT inactive after DIGIT PRESENT set – ACR (Abandon Call and Retry) (AUTODIAL)	1
REQUEST TO SEND (RTS) inactive	1
General autodial error	1
CLEAR TO SEND (CTS) inactive	7
CLEAR TO SEND (CTS) active before REQUEST TO SEND (RTS)	7
Error during contention (BSC)	7
Error during data transfer (BSC)	7
Adapter overrun/underrun	7
Unrecognizable SDLC control field	7
SDLC sequence number error, transmit error	7
SDLC sequence number error, receive error	7
CRC error	7
Frame abort detected (pattern '0111111'B)	7
CPU buffer overflow (on input)	7
Idle state detected	7
Nonproductive receive timeout	7
Data overrun (receive)	7
Data underrun (transmit)	7

ONLINE Parameter: Specifies whether the line is to be varied online automatically when the Control Program Facility (CPF) is started. After CPF is started, the Vary Line (VRYLIN) command can be used to modify the status of the line. ONLINE(*YES) should be specified for only one line description per line number; if is is specified for more than one, the system chooses the first line description based on alphabetic order.

YES: The line is to be online when CPF is started.

**NO:* The line is to be offline when CPF is started. The VRYLIN command must be used to put the line online, making it operational.

CTLU Parameter: Specifies, for nonswitched lines only, the names of one or more control units to which this line is physically attached. Do not use this parameter when following the normal procedure of creating the descriptions for lines first, control units (CRTCUD) second, and devices (CRTDEVD) last. Use this parameter only when the associated control unit descriptions have already been created before this line description is created. This parameter is valid only if CNN(*PP) or CNN(*MP) is specified; it is not valid for switched lines.

SWTCTLU Parameter: Specifies the names of up to 8 control units that can establish a connection with this switched BSC line. The control unit names should be created before using them in this parameter. This parameter is valid only if TYPE is *BSC, and CNN(*SWT) or SWNBKU(*YES) is specified. If CNN(*SWT) is specified, the control unit names will default to 8 null entries, to be used for answering only.

Note: To use this parameter, you should first create the line, ignoring the SWTCTLU parameter. Next, the controllers that can establish connections with this BSC switched line should be created, using the Create Control Unit Description (CRTCUD) command. Then, using the Change Line Description (CHGLIND) command, enter the names in the SWTCTLU parameter of the controller(s) that can establish connections with this BSC switched line.

STNADR Parameter: Specifies, only if TYPE(*SDLCS) or TYPE(*BSCT) is specified, the station address used by a host system to communicate with this System/38. If TYPE(*SDLCS) or TYPE(*BSCT) is specified, this parameter is required. The station address is the SDLC link address or BSCT polling address that has been assigned to this System/38. It is a two-digit hexadecimal value in the range 01 through FE.

EXCHID Parameter: Specifies, only if TYPE(*SDLCS) is specified, a user-defined exchange identifier for use with SNA communications networks.

**NONE:* No exchange identifier is to be specified. If an exchange identifier is needed, System/38 will generate one internally.

exchange-identifier: Specifies the identifier, with a format of 022xxxxx; xxxxx can be any combination of characters 0 through 9 and A through F.

CODE Parameter: Specifies the BSC line code to be used for communications.

<u>*EBCDIC:</u> The EBCDIC character set code is to be used.

**ASCII:* The ASCII character set code is to be used. ASCII is not valid if RJE(*YES) is specified.

RJE Parameter: Specifies, for BSC only, whether this line description is to be used by the Remote Job Entry Facility (RJEF). RJE(*YES) is not valid for TYPE(*BSCT).

<u>*NO:</u> This line description is not to be used by RJEF.

**YES:* This line description is to be used by RJEF.

BSCSWTDSC Parameter: Specifies whether inactivity on this BSC switched line (while in contention mode) should cause a line disconnect due to a 30-second timeout. Some CL commands may cause a timeout disconnect in a debugging or problem determination situation; in that case, you could use this parameter to disable the automatic timeout and continue. This parameter is valid only if TYPE(*BSC) and CNN(*SWT) are specified, or if TYPE(*BSC) and CNN(*PP) and SWNBKU(*YES) are specified.

<u>*YES:</u> The switched BSC line will be automatically disconnected after a 30-second period of inactivity (while in contention mode).

**NO:* The switched BSC line will not be automatically disconnected after a 30-second period of inactivity.

Note: When the last file is closed, the normal BSC switched line disconnect is not affected by this parameter.

PUBAUT Parameter: Specifies what authority for the line and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

Note: *NORMAL should be specified so that users who are authorized to use work stations attached to this line are not hindered from doing so because they might not also have been given explicit authority for the line.

<u>*NORMAL:</u> The public has only operational rights for the line.

**ALL:* The public has complete authority for the line.

**NONE:* The public cannot use the line.

TEXT Parameter: Lets the user enter text that briefly describes the line and its location. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

Line description work sheets are provided at the back of the *Guide to Program Product Installation and Device Configuration* that you can use to collect the information needed before creating the line descriptions. Refer to that publication for information about device configuration, system installation procedures, and how to use the work sheets.

```
CRTLIND  LIND(NYC)  LINNBR(20)  TYPE(*SDLCP)  +
      CNN(*SWT)  RATE(1200)  SELECT(*NO)  +
      NONRTNZ(*YES)  AUTOANS(*YES)  +
      WIRE(2)  DCEGRP(*C)  SWTCNN(*ANS)  +
      DTRDLY(1)  IDLETIME(38)  +
      NONPRDRCV(6)  RETRY(1)  +
      TEXT('Switched line between Chicago  +
      and New York City')
```

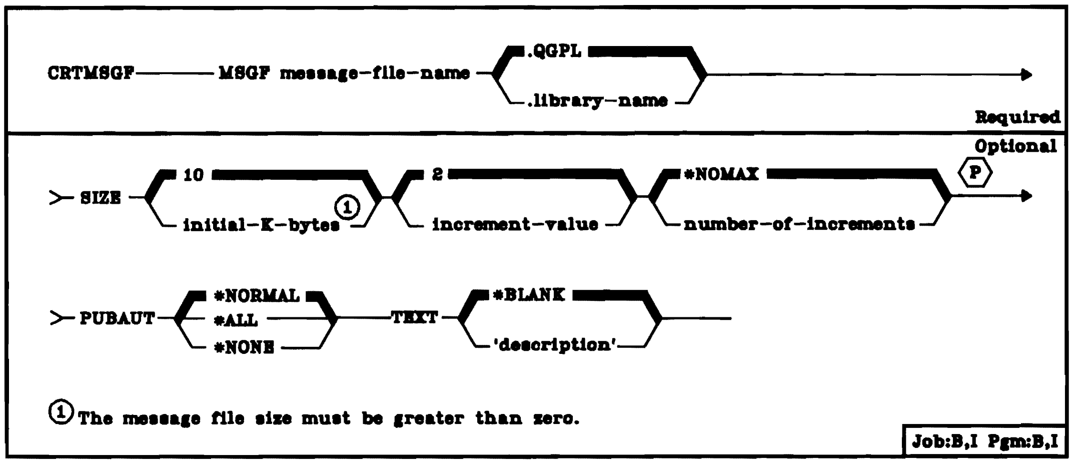
This command describes a line named NYC that is attached to the first line position (OU 20). This is a remote SDLCP line using a switched connection. The data rate is 1200 bits per second, and the line does not have the data rate select function. The line uses the NRZI data transmission method and has the automatic answering feature.

```
CRTLIND  LIND(MIL__MAD)  LINNBR(21)  TYPE(*SDLCP)  +
      CNN(*MP)  RATE(4800)  WIRE(4)  +
      DTRDLY(1)  IDLETIME(15)  +
      NONPRDRCV(2)  TEXT('MP line between  +
      Chicago, Milwaukee, & Madison')
```

This command describes a line named MIL__MAD that is attached to the second line position (OU 21). This is a remote SDLCP line using a nonswitched, multipoint connection. Because the line is a nonswitched line without the switched network backup feature, the DCEGRP parameter defaults to *A. The data rate is 4800 bits per second. A 4-wire physical connection is used.

CRTMSGF (Create Message File) Command

The Create Message File (CRTMSGF) command creates a user-defined message file for storing message descriptions. The message file should be stored in a library for which all users who are to use the predefined messages have operational rights. The system is shipped with the following IBM-supplied message files, which are all stored in the system library, QSYS: the CPF message file, QCPFMSG (for CPF and machine interface messages); and the licensed program message files, such as QRPGMSG (for RPG messages).



MSGF Parameter: Specifies the qualified name of the message file being created. (If no library qualifier is given, the file is stored in QGPL.)

SIZE Parameter: Specifies the initial storage size of the message file, the size of each increment added to its storage, and the number of times the size can be incremented. The storage size is expressed in K-bytes. The message file size is increased when a message description is added to the message file and there is no room for it in the file. The minimum size allowed is 1 K, the maximum allowed is 16 000 K. If SIZE is not specified, SIZE(10 2 *NOMAX) is assumed.

Initial Size: One of the following is used to specify the initial storage size of the message file.

10: Initially, the message file has 10 K of storage assigned to it. (1 K equals 1024 bytes of storage.)

initial-K-bytes: Enter the value that specifies the initial size of the file (cannot equal 0).

Increment Amount: One of the following is used to specify the amount of storage in K-bytes to be added to the message file's size.

2: The message file size is to be increased by 2 K of storage for each increment added.

increment-value: Enter the value that specifies the number of K-bytes to be added for each increment.

Number of Increments: One of the following is used to specify the maximum number of increments that can be added to the message file's size.

*NOMAX: The number of increments that can be added to the message file is not limited by the user. The maximum size is determined by the system.

number-of-increments: Enter the maximum number of increments that can be added to the file size. Enter a 0 to prevent any additions to the initial size of the file.

PUBAUT Parameter: Specifies what authority for the message file is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational and read rights for the message file.

**ALL:* The public has complete authority for the message file.

**NONE:* The public cannot use the message file.

TEXT Parameter: Lets the user enter text that briefly describes the message file and its description. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

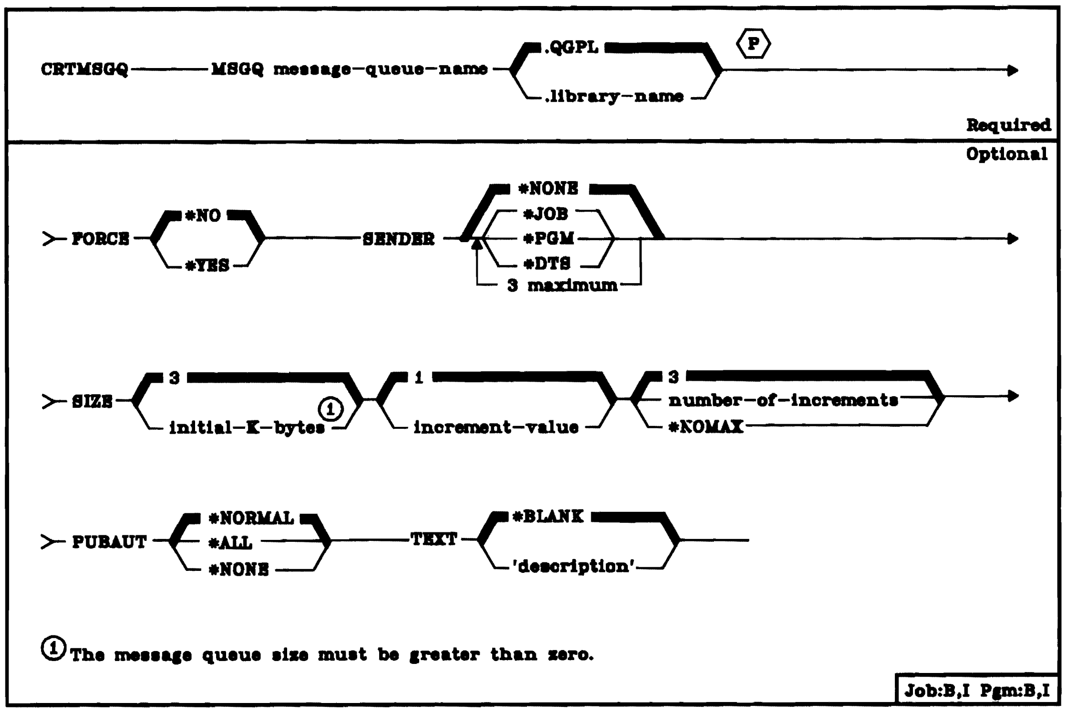
Example

```
CRTMSGF  MSGF(INVMSGS.INVLIB) +
       TEXT('Inventory Application Messages')
```

This command creates a message file named INVMSGS in which predefined inventory application messages are to be stored. The file is stored in the library INVLIB, for which all users of the file must have operational authority. Because PUBAUT was not specified, all users have operational authority for the file, meaning they can retrieve messages from the file.

CRTMSGQ (Create Message Queue) Command

The Create Message Queue (CRTMSGQ) command creates a user-defined message queue and stores it in a library. The message queue should be put in a library for which all users who are to send messages to and receive messages from the queue have operational authority. The messages sent can be predefined or impromptu messages. The message queue has the following attributes initialized when it is created: the DLVRY parameter is set to *HOLD, PGM is *DSPMSG, SEV is 00, and RESET is *NO. These initialized attributes cannot be specified on the CRTMSGQ command; if these values are not desired as defaults, the CHGMSGQ command must be used to change these values after the queue is created.



MSGQ Parameter: Specifies the qualified name of the message queue being created. (If no library qualifier is given, the queue is stored in QGPL.)

FORCE Parameter: Specifies whether changes made to the message queue description or messages added to or removed from the queue are to be immediately forced into auxiliary storage; this ensures that changes to the queue, or messages sent or received, are not lost if a system failure occurs.

*NO: Changes made to the message queue, including its messages, do not have to be immediately forced to auxiliary storage.

*YES: All changes to the message queue description and to the messages in the queue are to be immediately forced to auxiliary storage.

SENDER Parameter: Specifies one or more types of sender identifiers that are to be sent with each message sent to this user-defined message queue. These identifiers are supplied by the system when the message is sent; they are not specified by the sender. When the second-level text for a message is displayed by the Help key, the sender identifiers for that message are also displayed. If SENDER is not specified, no identifiers are sent with the message.

*NONE: No sender identifier is sent with the message.

**JOB:* The qualified name of the job is sent with the message. (The qualified job name includes the user name.) For interactive jobs, the job name is always the same as the work station name.

**PGM:* The name of the program (and the statement number within the program) sending the message is sent with the message.

**DTS:* The system date/time stamp is sent with the message to identify when the message was sent.

SIZE Parameter: Specifies the initial storage size of the message queue, the size of each increment added to its storage, and the number of times the size can be incremented. The storage size is expressed in K-bytes. The message queue size is increased when a message is sent to the message queue and there is no room for it in the queue. If SIZE is not specified, SIZE(3 1 3) is assumed.

Initial Size: One of the following is used to specify the initial storage size of the message queue.

3: Initially, the message queue has 3 K of storage assigned to it. (1 K equals 1024 bytes of storage.)

initial-K-bytes: Enter the value that specifies the initial size of the queue (cannot equal 0).

Increment Amount: One of the following is used to specify the amount of storage in K-bytes to be added to the message queue's size.

1: The message queue size is to be increased by 1 K of storage for each increment added.

increment-value: Enter the value that specifies the number of K-bytes to be added for each increment.

Number of Increments: One of the following is used to specify the maximum number of increments that can be added to the message queue's size.

3: A maximum of three increments can be added to the queue's size.

number-of-increments: Enter the maximum number of increments that can be added to the queue size. Enter a 0 to prevent any additions to the initial size of the queue.

**NOMAX:* The number of increments that can be added to the message queue is not limited by the user. The maximum size is determined by the system.

PUBAUT Parameter: Specifies what authority for the message queue is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational, read, add, and delete rights for the message queue.

**ALL:* The public has complete authority for the message queue.

**NONE:* The public cannot use the message queue.

TEXT Parameter: Lets the user enter text that briefly describes the message queue and its description. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTMSGQ  MSGQ(MYQ) SENDER(*JOB)
```

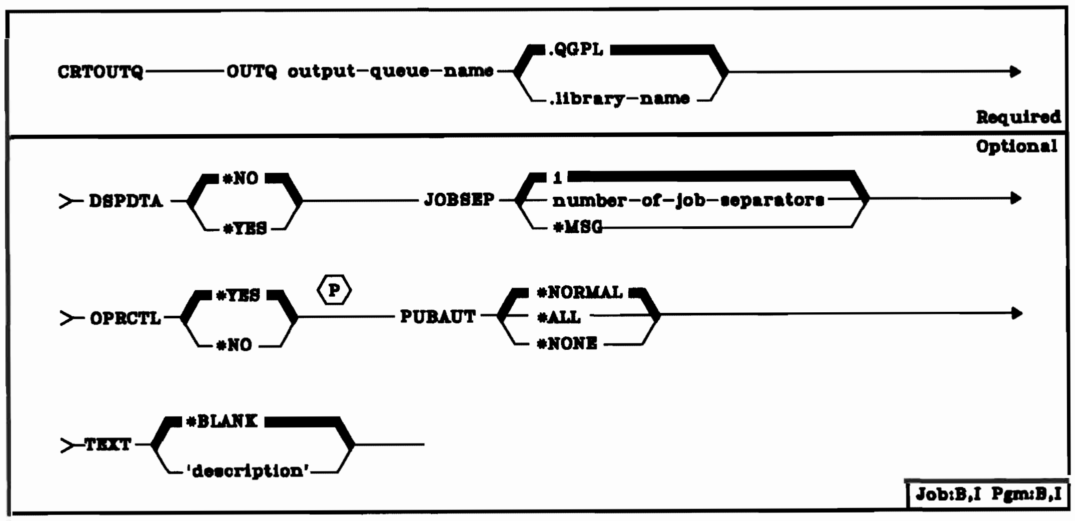
This command creates a message queue named MYQ and stores it in the general purpose library, QGPL, by default. All users are authorized to send messages to the queue and to read its description. Each message sent will be identified by the name of the job sending the message.

```
CRTMSGQ  MSGQ(INV.SPECIAL) +
      TEXT('This message queue is for inventory transactions')
```

This command creates a message queue named INV and stores it in a library named SPECIAL. The sender of the message is not identified.

CRTOUTQ (Create Output Queue) Command

The Create Output Queue (CRTOUTQ) command creates a new output queue for spooled output files. An entry is placed on the output queue for each spooled output file. The order in which the files are written to the output device is determined by the output priority of the job that produced each file and when the entry is made available to the writer.



OUTQ Parameter: Specifies the qualified name of the output queue being created. (If no library qualifier is given, the queue is stored in QGPL.)

DSPDTA Parameter: Specifies whether users who have authority to read the output queue can display the output data of any output file on the queue or only the data in their own files.

<u>*NO:</u> Users authorized to use the queue can display the output data of their own files only; they cannot display the output data of any file on the queue that they do not own.

**YES:* Any user having authority to read the queue can display the data of any file on the queue.

JOBSEP Parameter: Specifies, for each job with entries on the output queue, the number of separators to be placed at the beginning of the output for the job. Each separator (card or printer page) contains information that identifies the job, such as the job name, the job user's name, the job number, and the time and date of job execution.

1: One job separator is to be placed before each job's output.

number-of-job-separators: Enter a value, 0 through 9, that specifies the number of separators that are to be placed before the output of each job.

**MSG:* No job separators are to be placed before each job's output. A message is sent to a message queue notifying the operator of the end of each job. This message queue is identified by the MSGQ parameter of the Start Writer command.

OPRCTL Parameter: Specifies whether a user who has job control rights is allowed to manipulate or control the entries on this output queue. A user has job control rights if SPCAUT(*JOBCTL) is specified in his user profile.

*YES: A user with job control rights can control the queue and make changes to the entries on the queue.

**NO:* This queue and its entries cannot be controlled or changed by a user with job control rights unless he also has object management rights, and read, add, and delete rights for the queue.

PUBAUT Parameter: Specifies what authority for the output queue is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational, read, and add rights for the queue. Any user can put output file entries on the output queue and can display all entries on the queue. If DSPDTA(*YES) is specified, he can also display the data in *any* output file.

**ALL:* The public has complete authority for the queue.

**NONE:* The public has no authority for the queue.

TEXT Parameter: Lets the user enter text that briefly describes the output queue. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'*description*': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTOUTQ  OUTQ(DEPTAPRT) PUBAUT(*NONE) +
     TEXT('SPECIAL PRINT FILES FOR +
     DEPTA')
```

This command creates an output queue named DEPTAPRT and puts it in the QGPL library. Because PUBAUT(*NONE) is specified and OPRCTL(*YES) is assumed, the output queue can be used and controlled only by the user who created the queue and users who have job control authority. If users in Department A are to be authorized to use this output queue, the Grant Object Authority (GRTOBJAUT) command must be used to grant them the necessary authority. Data contained in files on this queue can be displayed only by those users who own the files. By default, one job separator will be printed at the beginning of the output for each job.

CRTPF (Create Physical File) Command

The Create Physical File (CRTPF) command creates a named physical file in the data base. A physical file is created from the file description parameters in the CRTPF command and (optionally) from a previously entered DDS source file that contains the source description of the file. If the physical file is to have a record format with only one character field and be in arrival sequence or if the file is to be a source file, a DDS source file is not needed. (To change attributes of the file after it has been created, use the Override Data Base File (OVRDBF) command before the file is opened.)

A physical file is the data base structure in which data is actually stored as records. The organization of the data is described in the record format, which is named and described in the data description specifications (DDS), using the data description specifications form or SEU (the Source Entry Utility).

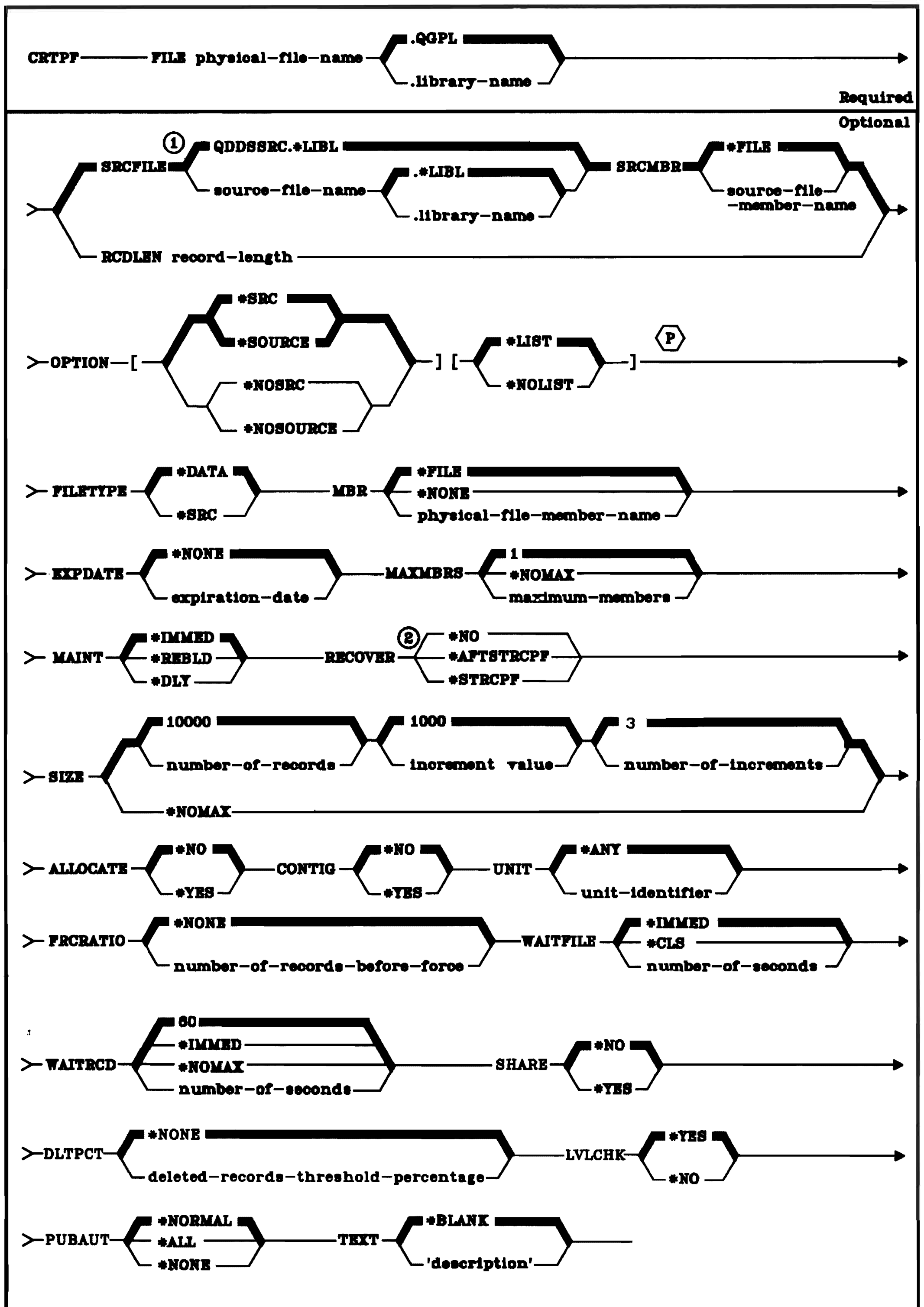
A physical file has only one record format, and the entire file contains records having only that format. This means that, from the system's viewpoint, all records are of the same length (fixed-length) and have the same fields. Programs may describe fields within fields as done in traditional systems.

Data is stored and referenced as fields within records, and the records are physically stored in the order in which they are written to the file (arrival sequence). However, the records can be logically processed in any order (arrival sequence or keyed sequence); the processing order is established by the access path specifications given for the file in DDS. If a file is defined as having a keyed sequence access path and there is only one member, it can still be processed in arrival sequence or by relative record number.

Each physical file can have one or more members; each physical file member is a separate collection of records, whose record format is the same as all other members of the file. Each member within the file has its own access path, of the same type as the file itself.

No records can be stored in the file being created until at least one member has also been added to the file. Either the MBR parameter of this command or the Add Physical File Member (ADDPFM) command can be used to add a member. However, the descriptive portion of the named file does exist within the data base even if there are no members.

Restriction: If a physical file is to be saved, it cannot contain more than 1999 members if it is a keyed file, or 3999 members if it is a nonkeyed file.



① To code the following parameters *positionally*, you must code them in this order, using *N for those not being specified: SRCFILE, SRCMBR, and RCDLEN.
② Refer to the parameter description for the default action taken for this parameter.

Job:B,I Pgm:B,I

FILE Parameter: Specifies the qualified name by which the physical file being created will be known. If no library qualifier is given, the physical file is stored in QGPL. (If the file is to be used in an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

SRCFILE Parameter: Specifies the name of the source file to be used when the physical file is created. Unless the RCDLEN parameter is specified instead, a source file name must be specified or QDDSSRC must contain the data description specifications describing the physical file. The source file contains the specifications that describe the record format and its fields, the access path, and the storage requirements for the file and its members. (For the specifications that can be made in DDS, refer to the *CPF Reference Manual—DDS*.)

If SRCFILE is specified, the RCDLEN parameter cannot be specified.

QDDSSRC: The system DDS source file named QDDSSRC in the QGPL library contains the source descriptions to be used to create the physical file. QDDSSRC can contain source descriptions for many files; each member of QDDSSRC contains the source description of one physical, logical, or device file. (When shipped, QDDSSRC contains no descriptions.) (If no library qualifier is specified, *LIBL is used to find the file.)

qualified-source-file-name: Enter the qualified name of the source file that contains the DDS to be used to create the physical file. (If no library qualifier is given, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of the source file member that contains the DDS for the physical file being created; the member is in the source file specified in the SRCFILE parameter (or its default, QDDSSRC). If SRCMBR is not specified, the member name is the same as the name of the physical file being created; the default value *FILE implies that the name of the physical file being created is to be used. A member name must be specified when the source file member to be processed does not have the same name as the physical file being created. If RCDLEN is specified, SRCMBR cannot be specified.

*FILE: The source file member name is the same as the name of the physical file being created.

source-file-member-name: Enter the name of the member in the source file specified by SRCFILE to be used to create the physical file.

RCDLEN Parameter: Specifies the record length, in bytes, of the records to be stored in the physical file. If RCDLEN and FILETYPE(*DATA) are specified, the physical file is created with a record format that has only one field. The file is then restricted to an arrival sequence access path. The record format and the field are both assigned the same name as that of the file itself, specified in the FILE parameter. The field is also assigned the data type of character whose length is the same as the record length specified here. A value of 1 through 32766 (32 766 bytes) can be specified for the record length.

If RCDLEN and FILETYPE(*SRC) are specified, the record format has three fields: source sequence number, date, and source statement. Also, the RCDLEN parameter must provide 12 positions for the source sequence number and date fields required in each record. If records are copied into the file by the CPYF command and the records are longer than the length specified here, the records are truncated on the right. These fields are defined with fixed attributes and names, and have a keyed access path over the sequence number. (See the *CPF Programmer's Guide* for details.)

If RCDLEN is specified, SRCFILE and SRCMBR cannot be specified; RCDLEN is used to specify a fixed record length for the record format when a source file is not needed (when only one field exists in each record or when the file being created is a source file). The HLL program that processes the file must describe the fields in the record within the program itself.

OPTION Parameter: Specifies the type of output listing to be produced when the file is created.

*SRC or **SOURCE:* A listing of the source statements used to create the file, and of any errors that occur, is to be generated.

**NOSRC* or **NOSOURCE:* No listing of the source statements is to be generated unless errors are detected. If errors are detected, they are listed along with the keyword or record format that caused the error.

*LIST: An expanded source listing is to be generated, showing a detailed list of the file specifications that result from the source statements and references to other file descriptions. This listing shows file, field, and key attributes.

**NOLIST:* No expanded source listing is to be generated.

FILETYPE Parameter: Specifies whether each member of the physical file being created is to contain data records or is to contain source records (statements) for a program or another file. The file could contain, for example, RPG source statements for an RPG program or DDS source statements for another physical, logical, or device file. (For an expanded description of the FILETYPE parameter, see Appendix A.)

Note: FILETYPE(*SRC) should be specified only when you are including DDS field definitions in the source file. Otherwise, you should use the CRTSRCPF (Create Source Physical File) to create a source file.

*DATA: The physical file is to contain data records.

**SRC:* The physical file is to contain source records. (Each source record must have at least three fields; see Appendix A.)

MBR Parameter: Specifies the name of the physical file member (if a member is to exist immediately) to be added when the physical file is created. (You can add other members to the file after it is created by using the ADDPFM command.)

*FILE: The member being added is to have the same name as that of the physical file that contains the member (specified in the FILE parameter).

**NONE:* No member is to be added when the file is created.

physical-file-member-name: Enter the name of the member that is to be added when the physical file is created.

EXPDATE Parameter: Specifies, if a physical file member is to be added when the file is created, the expiration date of the member. Any attempt to open a file that uses a member that has expired causes an error message to be sent to the user. (The expiration date of each member added later to the file must be specified in the ADDPFM command that adds it.)

*NONE: The member has no expiration date.

expiration-date: Enter the date after which the physical file member should not be used. The date must be in the format specified by the QDATFMT and QDATSEP system values.

MAXMBRS Parameter: Specifies the maximum number of members that the physical file being created can have at any time.

1: Only one member can be contained in the file.

**NOMAX:* No maximum is specified for the number of members; the system maximum of 32 767 members per file is used.

maximum-members: Enter the value for the maximum number of members that the physical file can have. A value of 1 through 32767 is valid.

MAINT Parameter: Specifies, for files with keyed sequence access paths only, the type of access path maintenance to be used for all members of the physical file. This parameter is not valid for files that have arrival sequence access paths.

IMMED: The access path is to be continuously (immediately) maintained for each physical file member. The path is updated each time a record is changed, added to, or deleted from the member. The records can be changed through a logical file that uses the physical file member regardless of whether the physical file is opened or closed. *IMMED must be specified for all files requiring unique keys to ensure uniqueness in all inserts and updates.

**REBLD:* The access path is to be completely rebuilt when a file member is opened during program execution. The access path is continuously maintained until the member is closed; the access path maintenance is then terminated. *REBLD is not valid for access paths that are to contain unique key values.

**DLY:* The maintenance of the access path is to be delayed until the physical file member is opened for use. Then, the access path is updated only for records that have been added, deleted, or updated since the file was last opened. (While the file is *open*, all changes made to its members are immediately reflected in the access paths of those members, no matter what is specified for MAINT.) To prevent a lengthy rebuild time when the file is opened, *DLY should be specified only when the number of changes to the access path between successive opens are small; that is, when the file is opened frequently or when the key fields in records for this access path change infrequently. *DLY is not valid for access paths that require unique key values.

If the number of changes saved reaches approximately 10 percent of the access path size, the system will stop saving changes and the access path will be completely rebuilt the next time the file is opened.

RECOVER Parameter: Specifies, for files having immediate maintenance on their access paths, when recovery processing of the file is to be performed after a system failure has occurred while the access path was being changed. This parameter is valid only if a keyed access path is used.

The access path having *immediate maintenance* can be rebuilt during start CPF (before any user can execute a job), or after start CPF has finished (during concurrent job execution), or when the file is next opened. While the access path is being rebuilt, the file cannot be used by any job.

The access path having *rebuild maintenance* will be rebuilt the next time its file is opened, the time that it normally is rebuilt.

**NO:* The access path of the file is not to be rebuilt. The file's access path is rebuilt when the file is next opened if it has rebuild maintenance. <u>*NO</u> is the default for all files that do not require unique keys.

**AFTSTRCPF:* The file is to have its access path rebuilt after the start CPF operation has been completed. This option allows other jobs not using this file to begin processing immediately after the CPF has been started. If a job tries to allocate the file while its access path is being rebuilt, a file open exception occurs if the specified wait time for the file is exceeded. <u>*AFTSTRCPF</u> is the default for all files that require unique keys.

**STRCPF:* The file is to have its access path rebuilt during the start CPF operation. This ensures that the file's access path will be rebuilt before the first user program tries to use it; however, no jobs can begin execution until after all files that specify RECOVER(*STRCPF) have their access paths rebuilt.

SIZE Parameter: Specifies the *initial* number of records in each member of the file, the number of records in each increment that can be automatically added to the member size, and the number of times the increment can be automatically applied. The number of records for each file member is expressed as the number of *undeleted* records that can be placed in it.

When the maximum number of records has been reached, a message (stating that the member is full) is sent to the system operator, giving him the choice of terminating the job or extending the member size himself. The operator can extend the member by the amount specified as the increment value (in the second value) one time for each time he receives the message.

A list of three values can be specified to indicate the initial size of each member and the automatic extensions that can be added when needed. Or *NOMAX can be specified instead. If SIZE is not specified, SIZE(10000 1000 3) is assumed by the system.

Records: One of the following is used to specify the *initial* number of records in the member before any automatic extension of the member occurs. The ALLOCATE parameter determines when the required space for the initial allocation occurs: If *YES is specified, the space is allocated when the file is created, or when a new member is added. If *NO is specified, the initial space is allocated as determined internally by the system.

<u>10000:</u> Initially, up to 10 000 records can be inserted into each member of the file before any extension occurs.

number-of-records: Enter the number of records that are inserted before an automatic extension occurs. A value of 0 cannot be used; the maximum value cannot exceed 16 777 215 records, or, if ALLOCATE(*YES) is specified, the amount of total system storage remaining for all permanent objects, whichever is less. If you do not want any automatic extensions, enter a 0 for the second and third values in the list.

Increment Amount: One of the following is used to specify the maximum number of records that can be additionally inserted in the member when the initial member size is exceeded and an automatic extension is made.

1000: A maximum of 1000 additional records can be inserted into the member after an automatic extension occurs.

increment-value: Enter the value (0 through 32767) that specifies the maximum number of additional records that can be inserted into the member after an automatic extension occurs. Enter a 0 to prevent automatic extensions.

Number of Increments: One of the following is used to specify the maximum number of increments that can be automatically added to the member. If 0 is specified for the increment amount, the number of increments need not be specified; 0 will be the default value instead of 3 (and a message is sent to the user issuing the command).

3: A maximum of three increments can be automatically added to the member size.

number-of-increments: Enter the maximum number of increments (0 through 32767) that can be automatically added to the member. Enter a 0 to prevent automatic extensions.

Unlimited Size: The following value can be specified to allow an unlimited number of records in each member.

**NOMAX:* The number of records that can be inserted into each member of the file is not limited by the user. The maximum size of each member is determined by the system.

ALLOCATE Parameter: Specifies whether *initial* storage space is to be allocated for each physical file member when it is added. The allocation provides enough space to hold the number of records specified by the SIZE parameter. Allocations which occur when a record cannot be added to a member without exceeding its capacity are determined by the system and by the SIZE parameter values.

*NO: When a new member is to be added, the system determines if additional space is needed, and allocates that amount.

**YES:* The amount of storage space specified in the first value of the SIZE parameter is allocated each time a new member is added. If that amount of storage space is not available, the member is not added, and a message is sent to the user. If this parameter value is used, SIZE(*NOMAX) cannot be specified.

CONTIG Parameter: The contiguous parameter specifies whether the user prefers that all records in the initial allocation in each physical file member are stored together without separations. If so, and the necessary contiguous space is not available, the system sends a message to the job log and allocates the storage space noncontiguously. The file is still entirely usable. This parameter does not indicate anything about the additional allocations that might be needed later, which most likely would be noncontiguous.

*NO: The storage space for each member does not have to be contiguous.

**YES:* The user wants the system to allocate contiguous space for each member of the physical file being added, and to notify the user and put a message in the job log if it cannot. The affected member is still added, even if the storage space has to be allocated noncontiguously. The member is just as usable in noncontiguous form. If *YES is specified for CONTIG, then ALLOCATE(*YES) must also be specified.

UNIT Parameter: Specifies, if the user prefers that a file be stored on a specific unit, the unit identifier of the auxiliary storage unit on which the system will attempt to allocate the storage space for the file and for all its members and their associated access paths. This includes the initial allocation when each member is added and any extensions that occur later for each member in the file. If the system cannot allocate the storage space for each member on the specified unit, it allocates the space on any available unit and sends a message to the job log. The file is entirely usable in all cases.

*ANY: The storage space for the file and its members can be allocated on any available auxiliary storage unit.

unit-identifier: Enter a valid value of 1 through 14 to specify the identifier of the auxiliary storage unit on which you prefer to have the storage space of all members allocated. The values that are valid depend on how many storage units are on the system, and on their types (62PC disk and 3370 disk).

Device Type	Unit	Unit Identifier
62PC	1-6	1-6
3370	Module 1, actuator 1	7
	Module 1, actuator 2	8
	Module 2, actuator 1	9
	Module 2, actuator 2	10
	Module 3, actuator 1	11
	Module 3, actuator 2	12
	Module 4, actuator 1	13
	Module 4, actuator 2	14

Note: These identifiers remain the same for systems that have 3370 devices and fewer than six 62PC devices.

The system attempts to make all space allocations on the unit specified. If it cannot, either because that unit is full or an invalid identifier was specified, it allocates the remainder of the space on any available unit and sends a message to the job log.

FRCRATIO Parameter: The force write ratio parameter specifies the number of inserted or updated records that are processed before they are forced into auxiliary (permanent) storage. (For an expanded description of the FRCRATIO parameter, see Appendix A.)

If this physical file is being journaled, a larger force write ratio or *NONE may be specified. Refer to the *CPF Programmer's Guide* for more information on the Journal Management Facility.

*NONE: There is no force write ratio; the system determines when the records are written in auxiliary storage.

number-of-records-before-force: Enter the number of new or changed records that are processed before they are explicitly forced into auxiliary storage.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

WAITRCD Parameter: Specifies the number of seconds that the program is to wait for a record that is to be updated or deleted. If the record cannot be allocated in the specified wait time, an error message is sent to the program.

60: The program is to wait for 60 seconds.

**IMMED:* The program is not to wait; when a record is locked, an immediate allocation of the record is required.

**NOMAX:* The wait time will be the maximum allowed by the system (32 767 seconds).

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the physical file member can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used. This parameter is not valid if a member is not being added when the physical file is created.

When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

**NO:* An ODP created by the program in which this command is used is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

DLTPCT Parameter: Specifies the maximum percentage of deleted records that any member in the physical file should have. The percentage is based on the number of deleted records compared with the total record count in a member. The percentage check is made when any member of the file is closed, and if the number of deleted records exceeds the percentage, a message is sent to the system history log to inform the user.

*NONE: No percentage is to be specified; the number of deleted records in the file members is not to be checked when a member is closed.

deleted-records-threshold-percentage: Enter a value, 1 through 100, that specifies the largest percentage of deleted records in any member in the file can have. If this percentage is exceeded, a message is sent to the system history log whenever the file is closed. This check will be made for logical file processing also.

LVLCHK Parameter: Specifies whether the record format identifiers are to be level checked to verify that the current record format identifier is the same as that specified in the program that opens the physical file. This value can be overridden on the OVRDBF command at execution time.

*YES: The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not match, an error message is sent to the program requesting the open, and the file is not opened.

**NO:* The level identifiers are not to be checked when the file is opened.

PUBAUT Parameter: Specifies what authority for the physical file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has operational, read, add, delete, and update rights for the physical file.

**ALL:* The public has complete authority for the file.

**NONE:* The public cannot use the file.

TEXT Parameter: Lets the user enter text that briefly describes the physical file. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTPF  FILE(PAYTXS.PAYLIB) SRCFILE(PAYTXS.SRCLIB) +
       MBR(*NONE) MAXMBRS(5)
```

This command creates a physical file named PAYTXS that is to be stored in the PAYLIB library. The source descriptions in the member PAYTXS source file also called PAYTXS in the SRCLIB library are used to create the physical file. The file is created without any members (*NONE was specified); therefore, no data can be put into the file until a member is added later. As many as five members can be contained in the file.

By default, each file member added later will contain data records. The access path of each member will be continuously maintained. Each member can have up to 10 000 records before automatic extensions (three increments maximum) occur that add 1000 records to the capacity of the member. The storage space for each member will be allocated only as needed, with no restrictions on which unit is used or whether the space is contiguous; there is no initial storage allocation. The public has operational, read, add, delete, and update authority for the file, but no object rights.

```
CRTPF  FILE(ORDERS.ORDERCTL) SRCFILE(ORDERSRC.ORDERCTL) +
       SRCMBR(MFGORD) MAXMBRS(50) SIZE(1000 100 5) ALLOCATE(*YES) +
       UNIT(01)
```

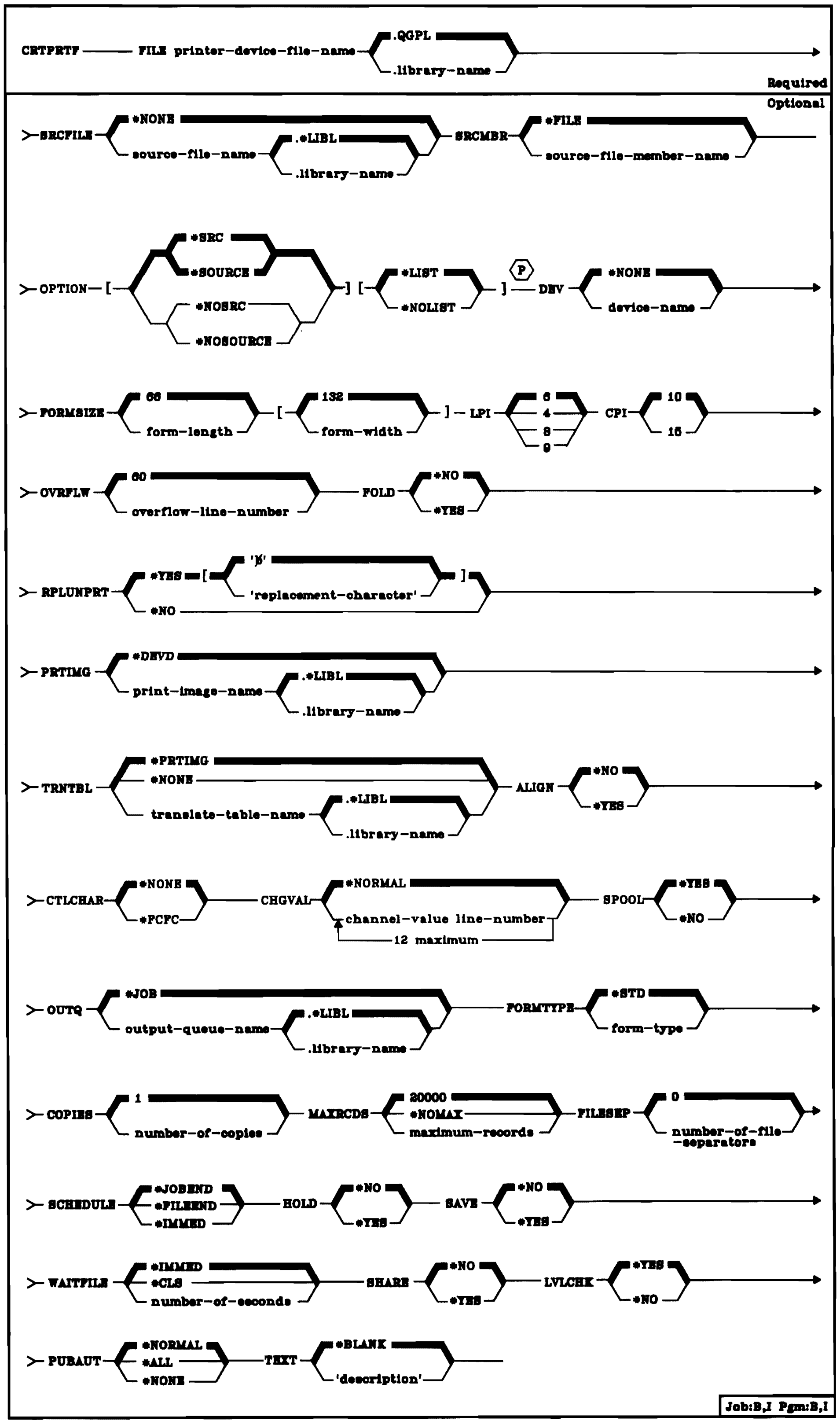
This command creates a physical file and physical file member, both named ORDERS, to be stored in the ORDERCTL library. The file and its member are created from the MFGORD source member of the ORDERSRC source file that is stored in the same library. The user prefers that all records placed in the file are to be stored on auxiliary storage unit 01, but the space does not have to be contiguous. A maximum of 50 members can be contained in the file. The initial allocation of storage provides for a maximum of 1000 records, and up to five increments of additional space for 100 records each can be added automatically. These allocation values also apply to each member of this physical file that is added later.

CRTPRTF (Create Printer File) Command

The Create Printer File (CRTPRTF) command creates a printer device file. The device file contains the file description, which identifies the device to be used and specifies the spooling requirements; the device file does not contain data. The printer device file is used to send records to the printer.

The printer file description is made up of information that is specified in two places: (1) in the source file that contains the data description specifications (if used); and (2) in the CRTPRTF command itself. The DDS contains the specifications for each record format in the device file and for the fields within each record format. A printer device file can have several record formats; the record format names must be unique within the file and the field names within each record format must be unique (however, the same field name can appear in more than one of the record formats).

The CHGPRTF or OVRPRTF command can be used in a program to change or override the parameter values specified in the printer file description. Each changed value in the device file remains changed after the program ends. Each overridden value remains altered only for the execution of the program (unless the override is deleted by a DLTOVR command); once the program ends, the original parameter values specified for the printer file are used. Override commands must be executed before the printer file to be affected is opened for use by the program.


CRTPRTF
FILE printer-device-file-name
.QGPL
.library-name
Required
Optional
SRCFILE
*NONE
source-file-name
.*LIBL
.library-name
SRCMBR
*FILE
source-file-member-name
OPTION
*SRC
*SOURCE
*NOSRC
*NOSOURCE
*LIST
*NOLIST
P
DEV
*NONE
device-name
FORMSIZE
66
form-length
132
form-width
LPI
6
4
8
9
CPI
10
15
OVRFLW
60
overflow-line-number
FOLD
*NO
*YES
RPLUNPRT
*YES
'ƀ'
'replacement-character'
*NO
PRTIMG
*DEVD
print-image-name
.*LIBL
.library-name
TRNTBL
*PRTIMG
*NONE
translate-table-name
.*LIBL
.library-name
ALIGN
*NO
*YES
CTLCHAR
*NONE
*FCFC
CHGVAL
*NORMAL
channel-value line-number
12 maximum
SPOOL
*YES
*NO
OUTQ
*JOB
output-queue-name
.*LIBL
.library-name
FORMTYPE
*STD
form-type
COPIES
1
number-of-copies
MAXRCDS
20000
*NOMAX
maximum-records
FILESEP
0
number-of-file-separators
SCHEDULE
*JOBEND
*FILEEND
*IMMED
HOLD
*NO
*YES
SAVE
*NO
*YES
WAITFILE
*IMMED
*CLS
number-of-seconds
SHARE
*NO
*YES
LVLCHK
*YES
*NO
PUBAUT
*NORMAL
*ALL
*NONE
TEXT
*BLANK
'description'
Job:B,I Pgm:B,I

FILE Parameter: Specifies the qualified name by which the printer device file being created will be known. If no library qualifier is given, the file is stored in QGPL. (If the file is to be used by an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

SRCFILE Parameter: Specifies the name of the source file (if there is one) that contains the data description specifications for the records in the printer device file. (For the specifications that can be made in DDS, refer to the *CPF Reference Manual–DDS*.)

*NONE: There is no DDS source file for this printer device file; the device file has only one record format with no fields, and the program that uses the file must describe the record formats and their fields.

qualified-source-file-name: Enter the qualified name of the source file that contains the DDS for this printer device file. (If no library qualifier is given, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of the member in the data base source file that contains the DDS for this printer device file.

*FILE: The source file member name is the same as the device file name specified in the FILE parameter.

source-file-member-name: Enter the name of the member in the source file specified by SRCFILE that is to be used to create the printer device file.

OPTION Parameter: Specifies the type of output listing to be produced when the file is created.

*SRC or **SOURCE:* A listing of the source statements used to create the file, and of any errors that occur, is to be generated.

**NOSRC* or **NOSOURCE:* No listing of the source statements is to be generated unless errors are detected. If errors are detected, they are listed along with the keyword or record format that caused the error.

*LIST: An expanded source listing is to be generated, showing a detailed list of the file specifications that result from the source statements and references to other file descriptions. This listing shows file and field keywords and attributes.

**NOLIST:* No expanded source listing is to be generated.

DEV Parameter: Specifies, for *nonspooled* output only, the name of the printer that is to be used with this printer device file to produce printed output. The device name of the IBM-supplied printer device description is QSYSPRT. If System/38 has two system printers attached, another printer device description named QSYSPRT2 is also provided. If SPOOL(*YES) is specified, this parameter is ignored.

*NONE: No device name is to be specified. The name of the printer device must be specified later in the CHGPRTF or OVRPRTF command, or in the HLL (high-level language) program that opens the file.

device-name: Enter the name of the device that is to be used with this printer device file. The device name must already be known on the system (via a device description) before this device file is created.

FORMSIZE Parameter: Specifies the length and width of the printer forms to be used by this device file. The length is in lines per page, and the width is in print positions (characters) per line. The defaults for FORMSIZE are 66 lines per page and 132 characters per line.

66: The form length is 66 print lines per page.

form-length: Enter the form length (in print lines per page) that is to be used by this device file. Although a value of 1 through 255 can be specified as the form length, the value specified should not exceed the actual length of the forms used. The following chart shows the number of lines per page that are valid for each printer type, depending on whether 6 or 8 lines per inch is specified in the LPI parameter for the 3203, 3262, and 5211 Printers, or is manually set on the 5256 Printer. For 5224 and 5225 Printers, 4, 6, 8, or 9 lines per inch can be specified.

	Lines per Page			
Printer	**4 lines/inch**	**6 lines/inch**	**8 lines/inch**	**9 lines/inch**
3203	–	2-144	2-192	–
3262 5211	–	2-84	2-112	–
5224 5225	1-255	1-255	1-255	1-255
5256	–	1-255	1-255	–

132: The form width is 132 printed characters per line.

form-width: Enter the form width (in characters per printed line) that is to be used by this device file. The value specified should not exceed the actual width of the forms used. Valid values for the 3203, 3262, 5211, and 5256 Printers are 1 through 132. Valid values for the 5224 and 5225 Printers are 1 through 198. The value specified should not exceed the actual width of the forms used.

LPI Parameter: Specifies the line spacing setting on the printer, in lines per inch, to be used by this device file. The line spacing on the 5256 Printer must be set manually.

6: The line spacing on the printer is to be 6 lines per inch.

4: The line spacing on the printer is to be 4 lines per inch.

8: The line spacing on the printer is to be 8 lines per inch.

9: The line spacing on the printer is to be 9 lines per inch.

Line spacings of 4 and 9 lines per inch are valid for only 5224 and 5225 Printers.

CPI Parameter: Specifies the printer character density, in characters per inch, to be used by this device file. 15 characters per inch is valid only for the 5224 and 5225 Printers.

10: Character density is to be 10 characters per inch.

15: Character density is to be 15 characters per inch.

OVRFLW Parameter: Specifies the line number on the page when overflow to a new page is to occur. Generally, after the specified line is printed, the printer overflows to the next page before printing continues. Refer to the *CPF Programmer's Guide* for details about controlling page overflow.

60: After line 60 has been printed, the printer overflows to a new page.

overflow-line-number: Enter the line number of the line that causes page overflow after the line is printed. The value specified must not exceed the length specified in the FORMSIZE parameter.

FOLD Parameter: Specifies whether all positions in a record are to be printed when the record length exceeds the form width (specified by the FORMSIZE parameter). When folding is specified and a record exceeds the form width, any portion of the record that cannot be printed on the first line will be continued (folded) on the next line or lines until the entire record has been printed.

*NO: Records are not to be folded; if a record is longer than the form width, only the first part of the record that fits on one line will be printed.

**YES:* Records whose length exceeds the form width are to be folded on the following line(s).

RPLUNPRT Parameter: The replace unprintable character parameter specifies (1) whether unprintable characters are to be replaced and (2) which substitution character (if any) is to be used. An *unprintable* character is a character that is not on the print belt or train, or in the print image used by the printer. The default values for RPLUNPRT are *YES and the blank (shown here as ƀ).

For 5224, 5225, and 5256 Printers, one of the following occurs when an unprintable character is encountered:

- If you specify RPLUNPRT(*YES), the specified substitution character is printed in place of each unprintable character.
- If you specify RPLUNPRT(*NO) and the value of the unprintable character is hex 00 through hex 3F, or is hex FF, undesirable results may occur. Most characters in this range cause an unrecoverable error to be signaled by the printer, and either the file is held for spooling or it is not processed. Some characters in this range, however, control forms movement and character representation on the printer. If the unprintable character is one of these control characters, additional spacing or skipping may occur. If control characters are specifically placed in the data, other system functions (such as the displaying or copying of a spooled file, or restarting or backing up of a print writer) may cause unpredictable results.
- If you specify RPLUNPRT(*NO) and the value of the unprintable character is in the range of hex 40 through hex FE, a recoverable error is signaled by the device and an inquiry message is sent to the operator, informing him of the error and giving him the chance to cancel the file or to continue processing. If the continue option is selected, subsequent unprintable characters will appear as blanks in the output, and no further inquiry messages will be sent to the operator.

For 3203, 3262, and 5211 Printers, one of the following occurs when an unprintable character is encountered:

- If you specify RPLUNPRT(*YES) and the value of the unprintable character is in the range of hex 00 through hex 3F, or is hex FF, the specified substitution character is printed instead. If no substitution character was specified, the blank is used. If no characters in this range are expected to be in the data to be printed, *NO can be specified for this parameter to gain some performance improvement. However, if *NO *is* specified and an unprintable character in this range does occur, the only recovery is to rerun the job.
- If you specify RPLUNPRT(*YES) and the value of the unprintable character is in the range of hex 40 through hex FE, a translate table should be used to translate unprintable characters to different printable characters; each unprintable hex value can be translated to its own printable character. The translate table, which is specified by the TRNTBL parameter, should also match the print image used by the printer.

- If you specify RPLUNPRT(*NO) and the value of the unprintable character is hex 00 through hex 3F, or is hex FF, undesirable results may occur. Most characters in this range cause an unrecoverable error to be signaled by the printer, and either the file is held for spooling or it is not processed. Some characters in this range, however, control forms movement and character representation on the printer. If the unprintable character is one of these control characters, additional spacing or skipping may occur. If control characters are specifically placed in the data, other system functions (such as the displaying or copying of a spooled file, or restarting or backing up of a print writer) may cause unpredictable results.

- If you specify RPLUNPRT(*NO) and the value of the unprintable character is in the range of hex 40 through hex FE, a recoverable error is signaled by the device and a notify message is sent to the program. If you choose to continue processing or if the message is unmonitored, the error will be ignored and processing will continue. Subsequent unprintable characters will appear as blanks in the output, and no further inquiry messages will be sent to the program.

YES: Unprintable characters are to be replaced. The program is not notified when unprintable characters are detected.

_*NO:_ Unprintable characters are not to be replaced. When an unprintable character is detected, a message is sent to the program.

'ƀ': A blank is to be used as the substitution character when an unprintable character is detected and *YES is specified.

'replacement-character': If *YES is also specified in this parameter, enter the substitution character that is to be used each time an unprintable character is detected. Any printable EBCDIC character can be specified.

PRTIMG Parameter: Specifies, for 3203, 3262, and 5211 Printers only, the name of the print image to be used by this printer device file. (This parameter does not apply to the 5224, 5225, or 5256 Printers.)

DEVD: The standard print image for the printer (specified in the device description) is to be used.

qualified-print-image-name: Enter the qualified name of the print image to be used by this device file. (If no library qualifier is given, *LIBL is used to find the print image.)

TRNTBL Parameter: Specifies, for 3203, 3262, and 5211 Printers only, the name of the translate table (if any) to be used by this device file when the output data is to be translated before it is printed. The translate table is used to convert each unprintable character having a hexadecimal code of 40 through FE to the printable character specified in the table that is also on the print belt or train. Each hexadecimal code can specify a different character.

For each IBM-supplied print image shipped with the system, a matching translate table is also supplied; the name of the table is the same as the name of the image.

PRTIMG: The translate table with the same qualified name as the print image is to be used.

**NONE:* No translation is needed when this device file is used.

qualified-translate-table-name: Enter the qualified name of the translate table to be used by this device file and the 3203, 3262, or 5211 Printer. (If no library qualifier is given, *LIBL is used to find the translate table.)

ALIGN Parameter: Specifies, for *nonspooled* output only, whether the forms must be aligned in the printer before printing is started. If ALIGN(*YES) and SPOOL(*NO) are specified, and forms alignment is required, the system sends a message to the QSYSOPR message queue (or any message queue specified for 5224, 5225, or 5256 Printers), and waits for a reply to the message. This parameter is ignored if SPOOL(*YES) is specified. (If the file is spooled, the message is sent to the message queue specified on the STRPRTWTR command whenever the printer writer is started and whenever the forms are to be changed.)

NO: No forms alignment is required.

**YES:* The forms are to be aligned before the output is printed.

CTLCHAR Parameter: Specifies whether the printer device file will support input with print control characters. Any invalid control characters that are encountered will be ignored, and single spacing is assumed.

NONE: No print control characters will be passed in the data to be printed.

**FCFC:* Specifies that the first character of every record will contain an ANSI forms control character. If *FCFC is specified, the record length must include one position for the first-character forms-control code. This value is not valid for externally described printer files (SRCFILE(*NONE) was specified).

CHLVAL Parameter: Specifies a list of channel numbers with their assigned line numbers. Use this parameter only if CTLCHAR(*FCFC) has been specified.

*NORMAL: The default values for skipping to channel identifiers will be used. The following are the default values:

ANSI First-Character Forms-Control Codes

Code	Action Before Printing a Line
' '	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to line 1
2-11	Space one line
12	Skip to overflow line (OVRFLW parameter)

channel-number: Specifies a channel number to be associated with corresponding 'skip to' line number. The only valid values for this parameter are 1 through 12, corresponding to channels 1 through 12. The CHLVAL parameter associates the channel number with a page line number.

If no line number is specified for a channel identifier, and that channel identifier is encountered in the data, a default of 'space one line' before printing is taken. Each channel number may be specified only once per CHGPRTF command invocation.

line-number: The line number assigned for the channel number in the same list. The range of valid line numbers is 1 through 255. If no line number is assigned to a channel number, and that channel number is encountered in the data, a default of 'space one line' before printing is taken. Each line number may be specified only once per CHGPRTF command invocation.

SPOOL Parameter: Specifies whether the output data for the printer device file is to be spooled. If SPOOL(*NO) is specified, the following parameters in this command are ignored: OUTQ, FORMTYPE, COPIES, MAXRCDS, FILESEP, SCHEDULE, HOLD, and SAVE.

*YES: The data is to be spooled for processing by a card, diskette, or print writer.

NO: The data is not to be spooled; it is sent directly to the device to be printed as the output becomes available.

OUTQ Parameter: Specifies, for spooled output only, the name of the output queue for the spooled output file.

JOB: The output queue specified in the job description associated with this job is to be used for the spooled output data.

qualified-output-queue-name: Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.) The IBM-supplied output queues that can be used by the printer file are QPRINT, QPRINT2, and QPRINTS output queues, stored in the QGPL library.

FORMTYPE Parameter: Specifies, for spooled output only, the type of forms to be used in the printer when it uses this device file to produce printed output. The identifiers used to indicate the type of forms are user-defined and must not be longer than 10 characters.

STD: The standard form used in your installation is to be used with this device file for printed output. The system assumes, (for *STD) that the standard forms are already in the printer; no message is sent when this device file is opened.

form-type: Enter the identifier of the form type to be used with this device file for printed output from jobs. A maximum of 10 alphameric characters can be specified. When the device file is opened, the system sends a message identifying the form type to the system operator, and requests that the identified forms be mounted in the printer.

COPIES Parameter: Specifies, for spooled output only, the number of copies (regardless of whether it is one-part or multipart paper) of the output to be printed when this printer device file is used.

1: Only one copy of the output is to be printed.

number-of-copies: Enter a value, 1 through 99, that indicates the number of identical print runs to be produced when this device file is used.

MAXRCDS Parameter: Specifies, for spooled output only, the maximum number of records that can be in the spooled output file for spooled jobs using this printer device file. If this maximum is exceeded, an error message is sent to the program message queue and the program is terminated.

20000: A maximum of 20 000 records can be in the spooled output file for each job that uses this printer device file.

**NOMAX:* No maximum is specified for the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of records that can be in the spooled output file.

FILESEP Parameter: Specifies, for spooled output files only, the number of separator pages to be placed at the beginning of each printed file, including those between multiple copies of the same output. Each separator page has the following items printed on it: file name, file number, job name, user name, and the job number.

0: No separator pages are to be used at the beginning of each spooled file produced by this device file.

number-of-file-separators: Enter the number of separator pages to be used at the beginning of each printed output file produced by this device file. Valid values are 0 through 9. If 0 is specified, no separator pages are printed for the file. In this case, the printed output for each file (or copy of a file) starts at the top of a new page.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a writer.

*JOBEND: The spooled output file is to be made available to the writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the writer as soon as the file is opened by the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The spooled file is made available to a writer when it is released by the Release Spooled File (RLSSPLF) command.

*NO: The spooled printer file is not to be held by the output queue. The spooled output is made available to a writer based on the SCHEDULE parameter value.

**YES:* The spooled printer file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled file is to be saved (left on the output queue) after the output has been produced.

*NO: The spooled file data is not to be retained on the output queue after it has been produced.

**YES:* The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the printer device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used.

When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a write operation in that program produces the next output record.

*NO: An ODP created by the program in which this command is used is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given an internal system identifier when the format is created.

*YES: The level identifiers of the record formats are to be checked when the file is opened. If the level identifiers do not all match, an error message is sent to the program requesting the open, and the file is not opened.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

PUBAUT Parameter: Specifies what authority for the printer device file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the Grant Object Authority (GRTOBJAUT) command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the device file.

**ALL:* The public has complete authority for the device file.

**NONE:* The public cannot use the device file.

TEXT Parameter: Lets the user enter text that briefly describes the printer device file. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTPRTF  FILE(DSPHIST)  SRCFILE(JOBHIST.PRSNNL)  FILESEP(3)
```

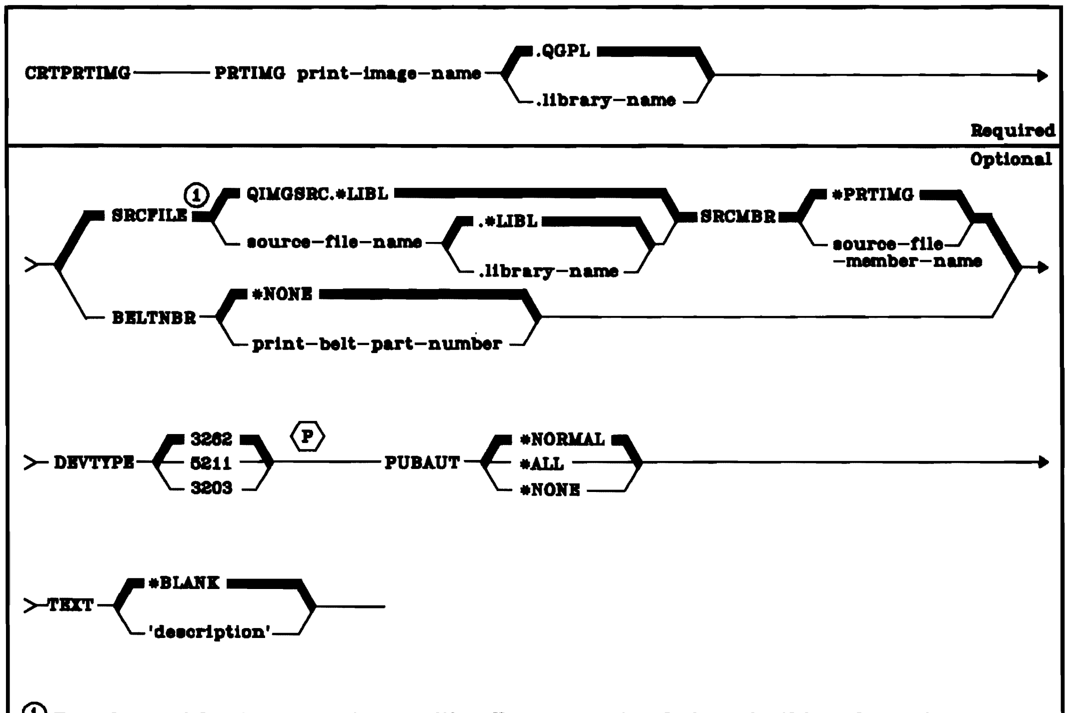
This command creates a description of the printer device file named DSPHIST using the device source file description named JOBHIST that is stored in the PRSNNL library. The defaults for all the other parameters are assumed, except for FILESEP. The device name must be specified in another CL command or in each program that uses the device file.

The printer uses standard forms for that installation that are 66 lines long and 132 print positions wide. It prints 6 lines per inch and overflows to a new page after line 60 is printed. The print image specified in the device description is used. Output is spooled to the output queue specified for the job and cannot be printed until the job ends. The spooled file is not to be held or saved after printing. One copy of the output will be printed, preceded by three separator pages, each containing the file name, the spooled number, and the job name and number.

CRTPRTIMG (Create Print Image) Command

The Create Print Image (CRTPRTIMG) command creates a print image that describes to the system a specific print belt (or print train, on the 3203 Printer) that is to be used on a system printer. A print image can be created from a source file that describes the print image or from IBM-supplied source information that is provided for each IBM print belt. The IBM-supplied source is used when the print belt number specified in this command has a standard IBM part number. When the part number is specified, a translate table that corresponds to the print image is also created. The translate table is given the same name as the print image. The print image is specified in a printer device file. When the printer file is opened, the print image is loaded into the 3203, 3262, or 5211 Printer, and the corresponding print train or belt is mounted to produce the output file.

Restrictions: (1) Before this command is executed, the diskette volume labeled IBM Service Library, Volume 1 must be mounted on magazine 1 of the diskette magazine drive. (2) When the system printer is to use a new print image and table, and their names are the same as the old print image and table, the printer must be varied offline and back online. First, however, the old print image and table must be deleted, then the new ones must be created. If the names are different, the new print image and table can be used by a printer device file that specifies the new names in the PRTIMG and TRNTBL parameters of its file description, or those same parameters default to the names specified in the device description that has been changed by the Change Device Description (CHGDEVD) command. The new names are used the next time the printer device file is opened.



① To code the following parameters *positionally*, you must code them in this order, using *N for those not being specified: SRCFILE, SRCMBR, and BELTNBR.

Job:B,I Pgm:B,I

PRTIMG Parameter: Specifies the qualified name of the print image whose description is being created. (If no library qualifier is given, the print image is stored in the general purpose library, QGPL.)

SRCFILE Parameter: Specifies the name of the source file containing the description of the print image being created. If SRCFILE or SRCMBR is specified, BELTNBR cannot be specified.

Note: Information about the format of records in the print image source file is contained in the *CPF Programmer's Guide*.

QIMGSRC: The source file named QIMGSRC in the QGPL library contains the source records to be used with this command to create the print image. (If no library qualifier is specified, *LIBL is used to find the file.)

qualified-source-file-name: Enter the qualified name of the source file that contains the source records to be used with this command to create the print image. (If no library qualifier is given, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of the source file member containing the description of the print image being created.

*PRTIMG: The source file member name is the same as the name of the print image.

source-file-member-name: Enter the name of the member in the source file specified by SRCFILE to be used to create the print image.

BELTNBR Parameter: Specifies (for 3262 and 5211 Printers) the IBM part number of the print belt or (for 3203 Printers) the train arrangement identification of the print train for which the print image and translate table are being created. Refer to the *Guide to Program Product Installation and Device Configuration* for the list of standard IBM print belts and print trains and their part numbers or identifiers. If BELTNBR is specified, SRCFILE and SRCMBR cannot be specified.

*NONE: No print belt number is to be specified. A nonstandard print belt is being used.

print-belt-part-number: For a 3262 or 5211 Printer, enter the part number of the IBM print belt for which the print image and associated translate table are being created. For a 3203 Printer, enter the train arrangement identification of the print train. Only digits are allowed for a print belt number, and only letters are allowed for a print train identifier.

DEVTYPE Parameter: Specifies the device type of the system printer for which the print image is to be used.

3262: The print image is for a 3262 Printer.

5211: The print image is for a 5211 Printer.

3203: The print image is for a 3203 Printer.

PUBAUT Parameter: Specifies what authority for the print image and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the GRTOBJAUT command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the print image.

**ALL:* The public has complete authority for the print image.

**NONE:* The public cannot use the print image.

TEXT Parameter: Lets the user enter text that briefly describes the print image description. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTPRTIMG  PRTIMG(CHAR48PI) SRCFILE(PRINTSRC) +
      SRCMBR(CHAR48SC) +
      TEXT('Print image for 48 character print belt')
```

This command creates a print image description for the print image named CHAR48PI; the print image is created from the source records contained in the PRINTSRC source file's CHAR48SC source member. The print image is (by default) to be used for a 3262 Printer.

Additional Considerations

The following example shows how a set of source records is coded for a print image in both formats for a 48-character print image.

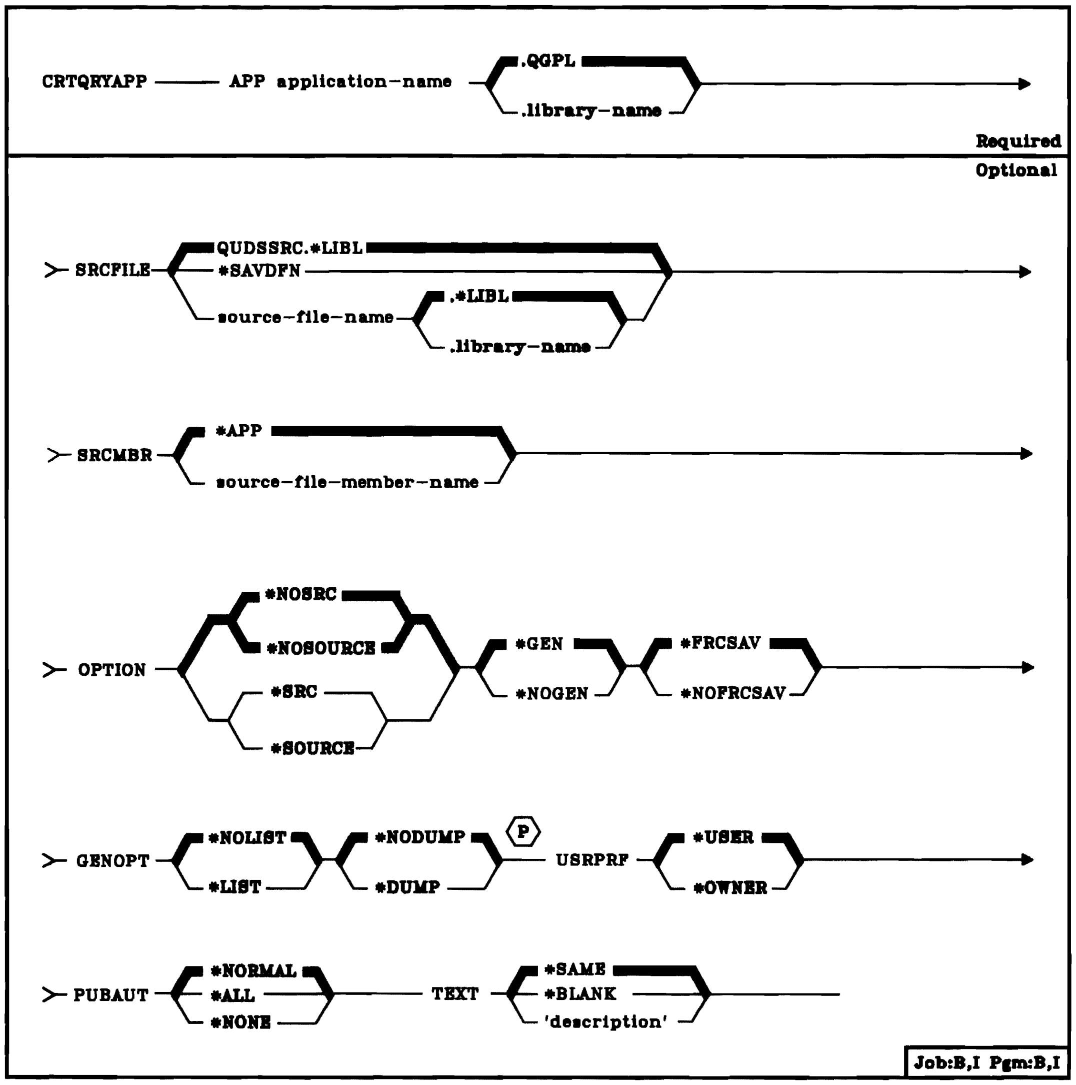
```
IMAGE CHAR,048
1234567890#@/STUVWXYZ&,%JKLMNOPQR-$*ABCDEFGHI+.'

IMAGE HEX,048
F1F2F3F4F5F6F7F8F9F07B7C61E2E3E4E5E6E7E8E9506B6C
D1D2D3D4D5D6D7D8D9605B5CC1C2C3C4C5C6C7C8C94E4B7D
```

Note that, in the hex format, the total number of characters entered in the source input records is always double that of the character count (identical to that specified in *Size of Character Set*) for the character format. It takes two source records in the HEX format for each source record in the CHAR format. More information about print image source records is contained in the *CPF Programmer's Guide*.

CRTQRYAPP (Create Query Application) Command

The Create Query Application (CRTQRYAPP) command creates an executable query application from an existing definition.

The Query Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Query Utility, refer to the *IBM System/38 Query Utility Reference Manual and User's Guide,* SC21-7755.



APP Parameter: Specifies the name of the application you are creating and specifies the library in which it is to be stored. (If no library name is given, the application is stored in the general-purpose library (QGPL.) The application name must be unique in the library where it is stored.

SRCFILE Parameter: Specifies the application or the name of the source file that contains the definition of the application. (If no library qualifier is specified, *LIBL is used to find the file.)

QUDSSRC: The QUDSSRC source file is provided in the library QIDU.

**SAVDFN:* The definition of the application is saved in the application specified in the APP parameter, rather than in a source file.

source-file-name: An existing source file other than the provided QUDSSRC.

Note: The CRTQRYAPP command ignores overrides to source files that contain UDS statements.

SRCMBR Parameter: Specifies the name of the source member that contains the source of the application.

*APP: The source of the application is in a source member that has the same name as the name specified in the APP parameter, which is described in the preceding paragraph.

source-file-member-name: The definition of the application is in a source member that has a name that is different from the name in the APP parameter. APP is described in the preceding paragraph.

OPTION Parameter: Specifies whether or not a listing of the source UDS is printed; specifies whether an executable application is actually created, or whether the source UDS is only checked for errors; specifies whether or not service information is to be printed. Select one value from each of the following groups: *NOSOURCE or *NOSRC and *SOURCE or *SRC; *GEN and *NOGEN; *NODUMP, *DUMP, and *EXCDUMP; *NOTRACE and *TRACE.

*NOSOURCE or *NOSRC: The *NOSOURCE and *NOSRC values are equivalent. When you specify *NOSOURCE or *NOSRC query does not print a listing of the source UDS. However, query does print a listing of errors found in source UDS.

**SOURCE* or **SRC:* The *SOURCE and *SRC values are equivalent. When you specify *SOURCE or *SRC query prints a listing of the source UDS.

*FRCSAV: Specifies that the definition of an application is saved regardless of whether the application will be executable. If *FRCSAV is not specified, the UDS is not saved if the application fails to create.

**NOFRCSAV:* Specifies that the definition will not be saved if the application fails to be created.

GEN: Create an executable application.

**NOGEN:* Do not create an executable application: perform error checking only.

GENOPT Parameter: Specifies the printing of IDU program listings created for your application. The listings may be required if a problem occurs in IDU.

USRPRF Parameter: Specifies under which user profile the application is to be executed.

USER: The user profile for the application user is in effect when the application is executed.

**OWNER:* The user profiles of both the application owner and the application user are in effect when the application is executed.

PUBAUT Parameter: Specifies what authority over the application is extended to all system users. (For an expanded description of the PUBAUT parameters, refer to Appendix A.)

NORMAL: All system users can execute the application, but all users cannot change the application.

**ALL:* All system users have complete authority over the application.

**NONE:* All users but the owner are restricted from the application. The owner can subsequently grant some or all rights to some or all other users.

TEXT Parameter: Lets you specify a description of the application.

SAME: Copy the description from the original definition.

**BLANK:* There is no description of this application.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

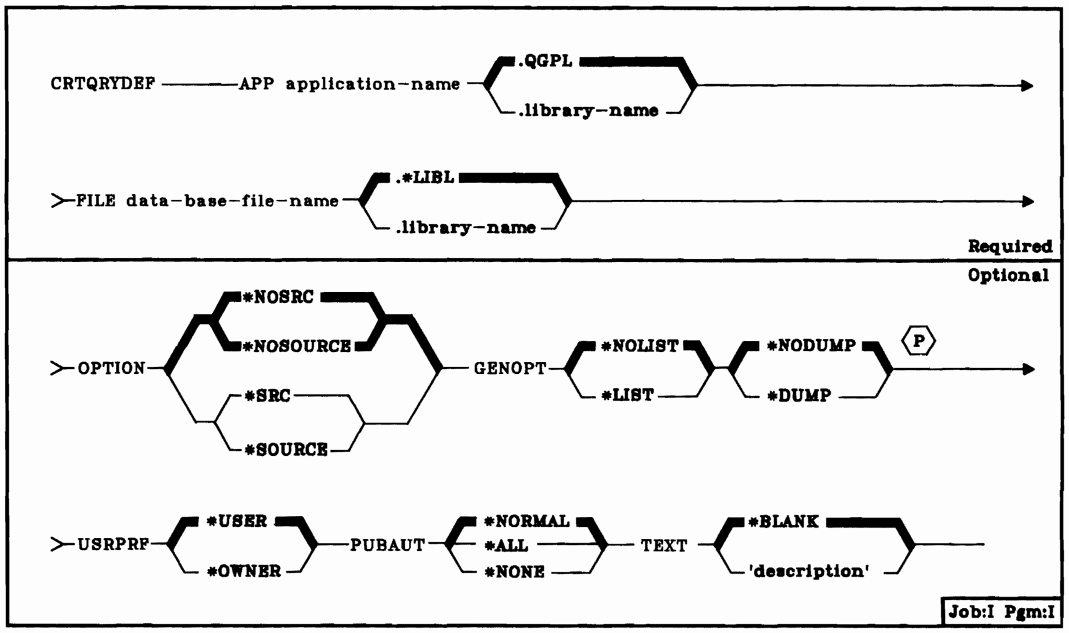
```
CRTQRYAPP APP(TEST1) SRCFILE(FILE1) SRCMBR(TEST2) +
      TEXT('Test application for TEST1')
```

This command creates an application named TEST1 using source member TEST2, which resides in source file FILE1.

CRTQRYDEF (Create Query Definition) Command

The Create Query Definition (CRTQRYDEF) command begins the prompting sequence for interactive definition of a Query application. Your responses to the prompts are used to create a Query application.

The Query Utility is part of the IBM System/38 Interactive Data Base Utilities Program Licensed Program Product, Program 5714-UT1. For more information on the Query Utility, refer to the *IBM System/38 Query Utility Reference Manual and User's Guide*, SC21-7724.



APP Parameter: Specifies the qualified name of the application being defined and the library in which it is to be stored. (If no library name is given, the application is stored in the general-purpose library, QGPL.)

FILE Parameter: Specifies the name of an existing data base file with record formats that will be referred to by the application you are defining. The file is defined by DDS (see the *CPF Reference Manual—DDS*). (If no library qualifier is specified, *LIBL is used to find the file.)

Note: Query has access to only the records included in the access path for the file; the access path is defined in DDS for the file. To determine whether DDS for a file contains select/omit logic that restricts the records available to Query, you can use the Display File Description (DSPFD) command.

OPTION Parameter: Specifies whether a listing of UDS (utility definition source) statements is to be printed, which may be helpful if problems occur.

*NOSRC or *NOSOURCE: Specifies that Query is not to print a listing of the UDS. The *NOSRC and *NOSOURCE values are equivalent.

**SRC* or **SOURCE:* Specifies that Query is to print a listing of the UDS. The *SRC and *SOURCE values are equivalent.

GENOPT Parameter: Specifies whether the IDU program listings for your application are to be produced. These listings may be helpful if a problem occurs.

*NOLIST: Specifies that an internal representation of the application program is not to be printed.

**LIST:* Specifies that an internal representation of the application program is to be printed.

*NODUMP: Specifies that the application program template is not to be printed.

**DUMP:* Specifies that the application program template is to be printed. *DUMP should be specified only if *LIST has been specified.

USRPRF Parameter: Specifies a user profile under which the application is to be executed. This parameter allows a programmer to define a Query application for someone who does not have full authority over the data base file that the application reads.

*USER: The user profile of the application user is in effect when the application is executed.

**OWNER:* The user profiles of both the application owner and the application user are in effect when the application is executed.

When you create an application that is to be used by someone else, you must authorize the user for the use of the application and any objects associated with the application. You can grant each user specific rights to such objects. By specifying USRPRF(*OWNER) when an application is created, you can permit a user to temporarily assume your authority to use objects associated with the application.

PUBAUT Parameter: Specifies what authority over the application is extended to all system users. (For an expanded description of the PUBAUT parameter, see Appendix A.)

<u>*NORMAL:</u> All system users can execute or read the application, but not all users can delete the application.

**ALL:* All system users have complete authority over the application.

**NONE:* All users but the owner are restricted from using the application. Of course, the owner can grant rights to other users.

TEXT Parameter: Enter a brief description of the application.

<u>*BLANK:</u> There is to be no description of the application.

'description': Enter no more than 50 characters, enclosed in apostrophes, to describe the application.

Example

```
CRTQRYDEF  APP(TEST1)  FILE(FILE1)  +
      TEXT('Create application for TEST1')
```

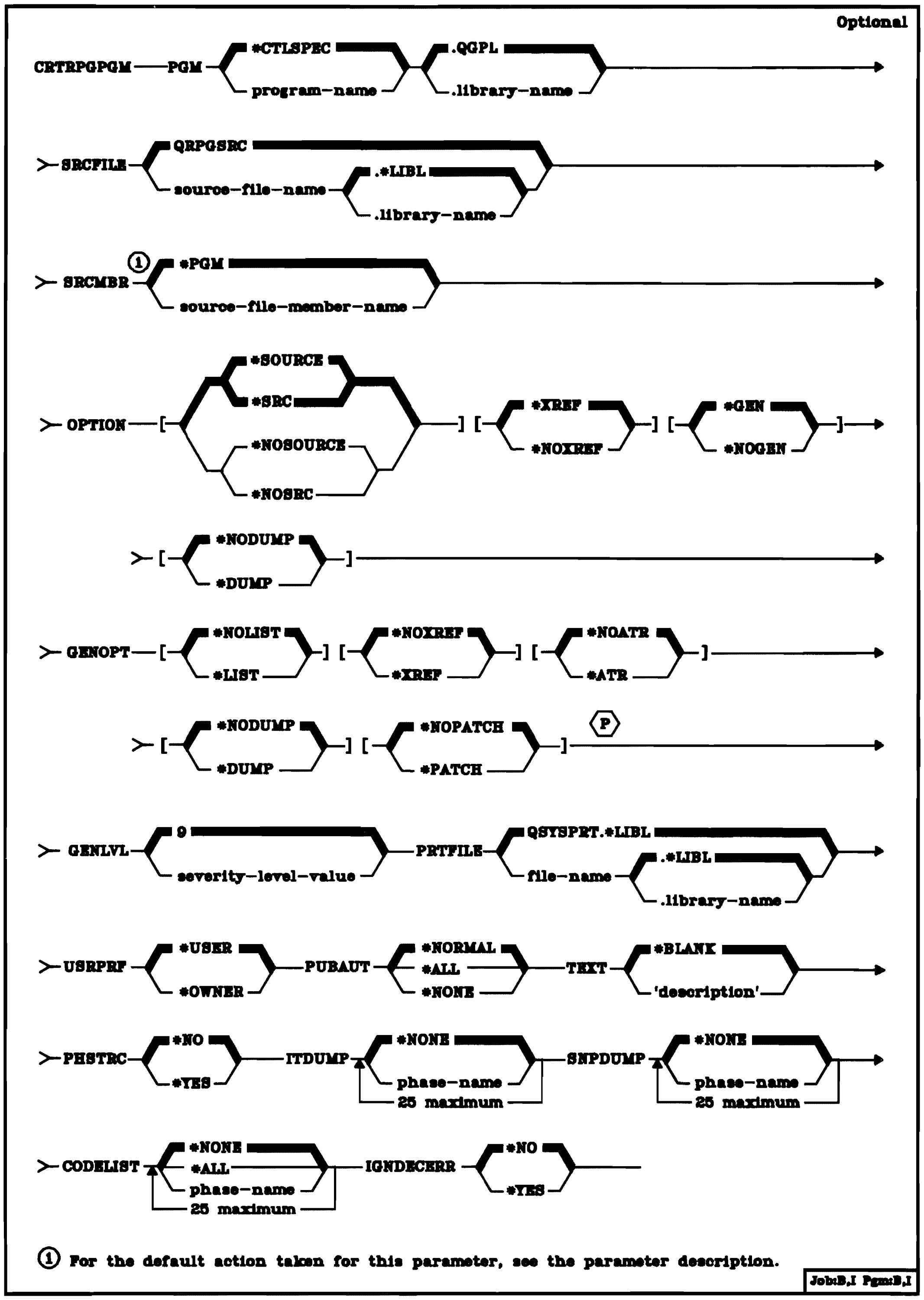
This command begins a prompting sequence which allows you to create an application named TEST1 in library QGPL. Your responses to the prompts define TEST1. Application TEST1 uses data from the data base file FILE1. No UDS or internal representation of the TEST1 will be printed. Any system users can execute or read TEST2, but only the owner of the application can delete it.

CRTRPGPGM (Create RPG Program) Command

The Create RPG Program (CRTRPGPGM) command invokes the RPG compiler, to compile RPG source statements into a program.

The RPG high-level language is part of the IBM System/38 RPG III Program Product, Program 5714-RG1. For more information, refer to the *IBM System/38 RPG III Reference Manual and Programmer's Guide,* SC21-7775.

Restriction: All object names specified on the CRTRPGPGM command must be composed of alphameric characters, the first of which must be alphabetic. The length of the names cannot exceed 10 characters.


Optional
CRTRPGPGM
PGM
*CTLSPEC
program-name
.QGPL
.library-name
SRCFILE
QRPGSRC
source-file-name
.*LIBL
.library-name
SRCMBR
①
*PGM
source-file-member-name
OPTION
*SOURCE
*SRC
*NOSOURCE
*NOSRC
*XREF
*NOXREF
*GEN
*NOGEN
*NODUMP
*DUMP
GENOPT
*NOLIST
*LIST
*NOXREF
*XREF
*NOATR
*ATR
*NODUMP
*DUMP
*NOPATCH
*PATCH
P
GENLVL
9
severity-level-value
PRTFILE
QSYSPRT.*LIBL
file-name
.*LIBL
.library-name
USRPRF
*USER
*OWNER
PUBAUT
*NORMAL
*ALL
*NONE
TEXT
*BLANK
'description'
PHSTRC
*NO
*YES
ITDUMP
*NONE
phase-name
25 maximum
SNPDUMP
*NONE
phase-name
25 maximum
CODELIST
*NONE
*ALL
phase-name
25 maximum
IGNDECERR
*NO
*YES
① For the default action taken for this parameter, see the parameter description.
Job:B,I Pgm:B,I

PGM Parameter: Specifies the qualified name by which a compiled RPG program is to be known. (If no library qualifier is specified, the created program is stored in the general purpose library, QGPL.) The program must not already exist in QGPL.

*CTLSPEC: The system uses the program name specified in positions 75 through 80 of the control specification. If the program name is not specified on the control specification, the program assumes the name specified on the SRCMBR parameter. If a program name is not specified on the control specification, and if a member name is not specified by using the SRCMBR parameter, the default program name is RPGOBJ.

program-name: Enter the name by which the program will be known.

QGPL: If a library name is not specified, the program is stored in QGPL.

.library-name: Enter the name of the library in which the compiled program is to be stored.

SRCFILE Parameter: Specifies the name of the source file that contains the RPG source to be compiled and the library in which the source file is located.

QRPGSRC.*LIBL: If a source file name is not specified, the IBM-supplied source file QRPGSRC contains the RPG source to be compiled.

source-file-name: Enter the name of the source file that contains the RPG source program to be compiled. (If no library qualifier is specified, *LIBL is used to find the program.)

SRCMBR Parameter: Specifies the name of the member of the source file that contains the RPG source program to be compiled. This parameter can be specified only if the source file name in the SRCFILE parameter is a data base file.

*PGM: The system uses the name specified on the PGM parameter as the source file member name. If no program name is specified by using the PGM parameter, the system uses the first member created in or added to the source file as the source member name.

source-file-member-name: Enter the name of the member that contains the RPG source program.

OPTION Parameter: Specifies whether the following options are to be used when the RPG source is compiled.

*SOURCE or *SRC: The compiler produces a source listing, consisting of RPG source input and all compile-time errors.

**NOSOURCE* or **NOSRC:* The compiler does not produce a source listing. If either *NOSOURCE or *NOSRC is specified, the system defaults to *NOXREF.

*XREF: The compiler produces a cross-reference listing and key field information table (when appropriate) for the source program.

**NOXREF:* The compiler does not produce a cross-reference listing for the source program. This is the default when either *NOSOURCE or *NOSRC is specified.

*GEN: The compiler creates an executable program after the program is compiled.

**NOGEN:* The compiler does not create an executable program after the program is compiled.

*NODUMP: When an error occurs during compilation, the compiler does not dump major data areas.

**DUMP:* The compiler dumps major data areas when an error occurs during compilation.

GENOPT Parameter: Specifies the printing of the IRP (intermediate representation of a program), a cross-reference listing of objects defined in the IRP, an attribute listing from the IRP, and the program template; and specifies the reservation of a program patch area. These listings may be required if a problem occurs in RPG. For a description of the GENOPT parameter and the information it provides, see Appendix E in the *IBM System/38 RPG III Reference Manual and Programmer's Guide,* SC21-7725.

GENLVL Parameter: Specifies whether a program is to be generated, depending on the severity of messages generated as a result of compile-time errors. If errors occur in a program with a severity level equal to or greater than the value specified in this parameter, the compile will terminate. The severity level value of RPG messages does not exceed 50.

9: If a severity level value is not specified, the default severity level is 9. If a severity level greater than 9 is specified, the program may contain errors that will cause unpredictable results when the program is executed.

severity-level-value: A two-digit number, 01 through 50, can be specified.

PRTFILE Parameter: Specifies the name of the file in which the compiler listing is to be placed and the library in which the file is located.

QSYSPRT.*LIBL: If a file name is not specified, the compiler listing is placed in the IBM-supplied file, QSYSPRT. If the file is spooled, the file goes to the QPRINT queue. The file QSYSPRT has a record length of 132. If you specify a file whose record length is less than 132, information will be lost.

file-name: Enter the qualified name of the file in which the compiler listing is to be placed. (If no library qualifier is given, *LIBL is used to find the file.)

USRPRF Parameter: Specifies under which user profile the compiled RPG program is to be executed. The profile of either the program owner or the program user is used to execute the program and control which objects can be used by the program (including what authority the program has for each object).

*USER: The program user's user profile is to be used when the program is executed.

**OWNER:* The user profiles of both the program's owner and user are to be used when the program is executed. The collective sets of object authority in both user profiles are to be used to find and access objects during the program's execution. Any objects that are created during the program are owned by the program's user.

PUBAUT Parameter: Specifies what authority for the program and its description is being granted to the public. (For an expanded description of the PUBAUT parameter, refer to Appendix A.)

*NORMAL: The public has only operational rights for the compiled program. Any user can execute the program, but cannot change it or debug it.

**ALL:* The public has complete authority for the program.

**NONE:* The public cannot use the program.

TEXT Parameter: Lets the user enter text that briefly describes the program and its function. The text appears whenever the program appears.

*BLANK: No text is to be specified.

'*description*': Enter no more than 50 characters, enclosed in apostrophes.

PHSTRC Parameter: Specifies whether information about compiler phases is provided on the listing.

*NO: Information about compiler phases is not provided.

**YES:* Information about compiler phases is provided.

ITDUMP Parameter: This parameter specifies whether a dynamic listing of intermediate text for one or more specified phases is to be printed at compile time as each IT record is being built. This parameter also specifies whether a flow of the major routines executed in one or more specified phases is to be printed.

*NONE: No intermediate dump is produced.

phase-name: Last two characters of phase name.

SNPDUMP Parameter: Specifies whether the major data areas are to be printed after the execution of one or more specified phases.

*NONE: No snap dump is produced.

phase-name: Last two characters of phase name.

CODELIST Parameter: Specifies whether a dynamic listing of the IRP is to be printed during execution of one or more specified phases.

*NONE: No code listing is produced.

**ALL:* A code listing is produced for each phase executed.

phase-name: Last two characters of phase name.

IGNDECERR Parameter: Specifies whether decimal data errors detected by the system are ignored by the program.

*NO: Decimal data errors are not ignored.

**YES:* Decimal data errors are ignored by the program. The result of the operation being performed when the error occurs is unknown. The compiler generates an error message on the compiler listing to notify the user that this option was specified. Incorrect results that occur during the execution of a program when this option is specified are the user's responsibility.

Example

```
CRTRPGPGM  PGM(ARBR5.JONES)  GENLVL(30)  TEXT('Accounts Receivable +
      Branch 5')
```

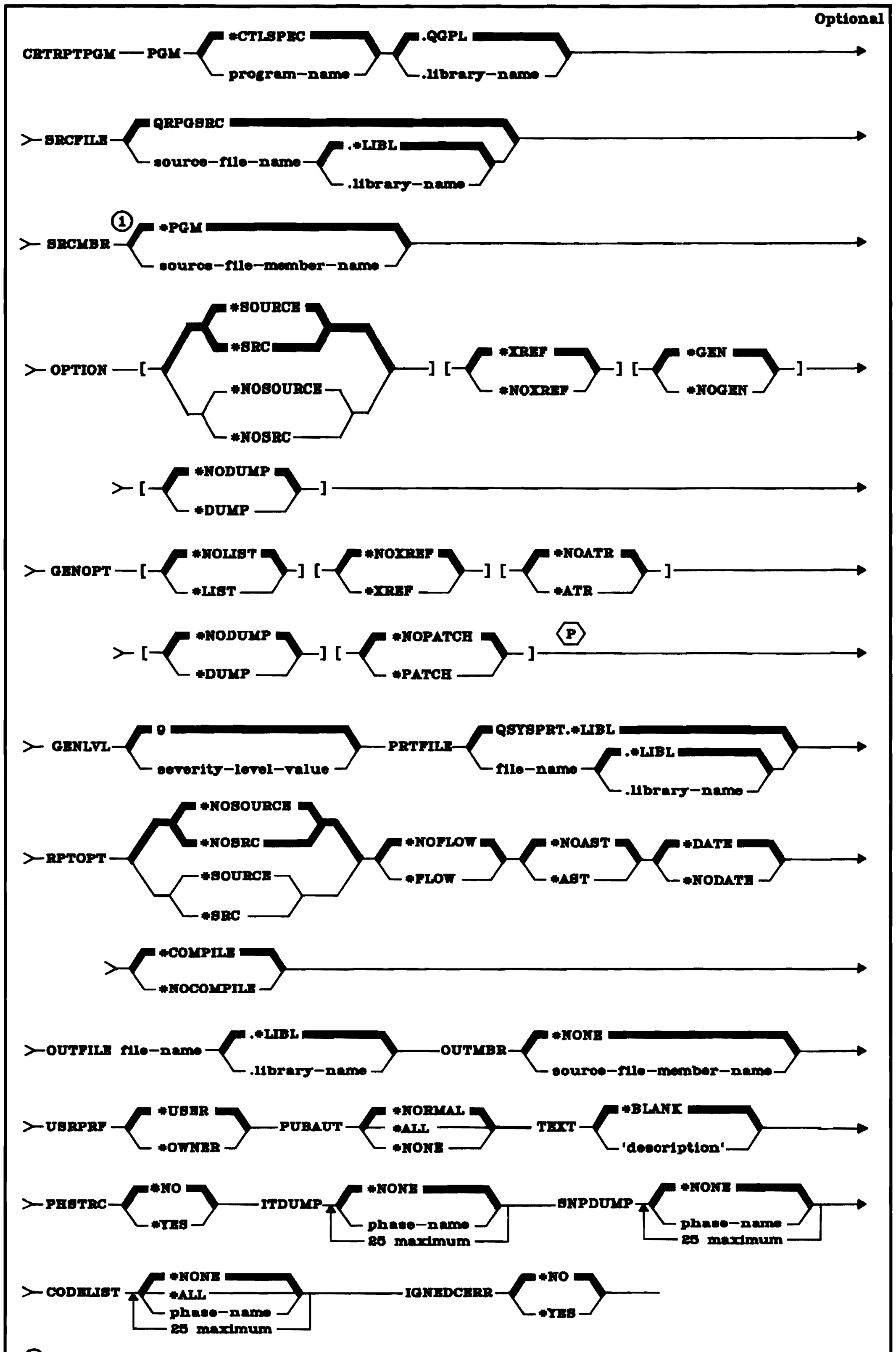
This command invokes the RPG compiler to produce an executable RPG program called ARBR5 in library JONES. The source program that is compiled resides in a member also named ARBR5, in source file QRPGSRC. Any errors occurring during the compile will appear on a source listing of the RPG source input, printed on the system printer. If an error occurs with a severity level of 30 or higher, the compile will terminate.

CRTRPTPGM (Create Auto Report Program) Command

The Create Auto Report Program (CRTRPTPGM) command generates the compilation of an RPG program that contains auto report specifications.

The RPG high-level language is part of the *IBM System/38 RPG III Program Product*, Program 5714-RG1. For more information, refer to the *IBM System/38 RPG III Reference Manual and User's Guide*, SC21-7775.

Restriction: All object names specified on the CRTRPTPGM command must be composed of alphameric characters, the first of which must be alphabetic. The length of the names cannot exceed 10 characters.



① For the default action taken for this parameter, see the parameter description.

Job:B,I Pgm:B,I

PGM Parameter: Specifies the qualified name by which a compiled RPG program is to be known. (If no library qualifier is specified, the created program is stored in the general purpose library, QGPL.) The program must not already exist in QGPL.

*CTLSPEC: The system uses the program name specified in positions 75 through 80 of the control specification. If the program name is not specified on the control specification, the program assumes the name specified on the SRCMBR parameter. If a program name is not specified on the control specification, and if a member name is not specified by using the SRCMBR parameter, the default program name is RPGOBJ.

program-name: Enter the name by which the program will be known.

QGPL: If a library name is not specified, the program is stored in QGPL.

.library-name: Enter the name of the library in which the compiled program is to be stored.

SRCFILE Parameter: Specifies the name of the source file that contains the RPG source to be compiled and the library in which the source file is located.

QRPGSRC.*LIBL: If a source file name is not specified, the IBM-supplied source file QRPGSRC contains the RPG source to be compiled.

source-file-name: Enter the name of the source file that contains the RPG source program to be compiled. (If no library qualifier is specified, *LIBL is used to find the source file.)

SRCMBR Parameter: Specifies the name of the member of the source file that contains the RPG source program to be compiled. This parameter can be specified only if the source file name in the SRCFILE parameter is a data base file.

*PGM: The system uses the name specified on the PGM parameter as the source file member name. If no program name is specified by using the PGM parameter, the system uses the first member created in or added to the source file as the source member name.

source-file-member-name: Enter the name of the member that contains the RPG source program.

OPTION Parameter: Specifies whether the following options are to be used when the RPG source is compiled.

*SOURCE or *SRC: The compiler produces a source listing, consisting of RPG source input and all compile-time errors.

**NOSOURCE* or **NOSRC:* The compiler does not produce a source listing. If either *NOSOURCE or *NOSRC is specified, the system defaults to *NOXREF.

*XREF: The compiler produces a cross-reference listing and key field information table (when appropriate) for the source program.

**NOXREF:* The compiler does not produce a cross-reference listing for the source program. This is the default when either *NOSOURCE or *NOSRC is specified.

*GEN: The compiler creates an executable program after the program is compiled.

**NOGEN:* The compiler does not create an executable program after the program is compiled.

*NODUMP: When an error occurs during compilation, the compiler does not dump major data areas.

**DUMP:* The compiler dumps major data areas when an error occurs during compilation.

GENOPT Parameter: Specifies the printing of the IRP (intermediate representation of a program), a cross-reference listing of objects defined in the IRP, an attribute listing from the IRP, and the program template; and specifies the reservation of a program patch area. These listings may be required if a problem occurs in RPG. For a description of the GENOPT parameter and the information it provides, see Appendix E in the *IBM System/38 RPG III Reference Manual and Programmer's Guide*, SC21-7725.

GENLVL Parameter: Specifies whether a program is to be generated, depending on the severity of messages generated as a result of compile-time errors. If errors occur in a program with a severity level equal to or greater than the value specified in this parameter, the compile will terminate. The severity level value of RPG messages does not exceed 50.

9: If a severity level value is not specified, the default severity level is 9. If a severity level greater than 9 is specified, the program may contain errors that will cause unpredictable results when the program is executed.

severity-level-value: A two-digit number, 01 through 50, can be specified.

PRTFILE Parameter: Specifies the name of the file in which the compiler listing is to be placed and the library in which the file is located.

QSYSPRT.*LIBL: If a file name is not specified, the compiler listing is placed in the IBM-supplied file, QSYSPRT. If the file is spooled, the file goes to the QPRINT queue. The file QSYSPRT has a record length of 132. If you specify a file whose record length is less than 132, information will be lost.

file-name: Enter the qualified name of the file in which the compiler listing is to be placed. (If no library qualifier is given, *LIBL is used to find the file.)

RPTOPT Parameter: Specifies whether the following options are to be used when the auto report source program is compiled.

*NOSOURCE or **NOSRC:* A source listing is not written.

**SOURCE* or **SRC:* A source listing, consisting of auto report source input and all compile-time errors, is written.

*NOFLOW: A flow of the major routines executed is not written.

**FLOW:* A flow of the major routines executed while the auto report source program is compiled is written.

*NOAST: Asterisk indication is suppressed from generated total output lines.

**AST:* Asterisks are generated for total output lines.

*DATE: The page number and date are included on the first *AUTO page heading line.

**NODATE:* The page number and date are suppressed on the first *AUTO page heading line.

*COMPILE: The RPG compiler is called after the auto report source program is compiled.

**NOCOMPILE:* The RPG compiler is not called.

OUTFILE Parameter: Specifies the qualified name of the file where the output from the auto report compiled program is to be placed and the library in which the file is located. The file specified on the OUTFILE parameter is also used as the source input file to the RPG compiler unless RPTOPT(*NOCOMPILE) is specified. If the OUTFILE parameter is not specified, auto report creates a file in library QTEMP to pass the generated RPG source to the RPG compiler.

OUTMBR Parameter: Specifies the name of the member of the file that will contain the output from auto report.

<u>*NONE:</u> Uses the first member created in or added to the file as the member name.

source-file-member-name: Enter the name of the member that is to contain the output of auto report.

USRPRF Parameter: Specifies under which user profile the compiled RPG program is to be executed. The profile of either the program owner or the program user is used to execute the program and control which objects can be used by the program (including what authority the program has for each object).

<u>*USER:</u> The program user's user profile is to be used when the program is executed.

**OWNER:* The user profiles of both the program's owner and user are to be used when the program is executed. The collective sets of object authority in both user profiles are to be used to find and access objects during the program's execution. Any objects that are created during the program are owned by the program's user.

PUBAUT Parameter: Specifies what authority for the program and its description is being granted to the public. (For an expanded description of the PUBAUT parameter, refer to Appendix A.)

<u>*NORMAL:</u> The public has only operational rights for the compiled program. Any user can execute the program, but cannot change it or debug it.

**ALL:* The public has complete authority for the program.

**NONE:* The public cannot use the program.

TEXT Parameter: Lets the user enter text that briefly describes the program and its function. The text appears whenever the program appears.

<u>*BLANK:</u> No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

PHSTRC Parameter: Specifies whether information about compiler phases is provided on the listing.

*NO: Information about compiler phases is not provided.

**YES:* Information about compiler phases is provided.

ITDUMP Parameter: This parameter specifies whether a dynamic listing of intermediate text for one or more specified phases is to be printed at compile time as each IT record is being built. This parameter also specifies whether a flow of the major routines executed in one or more specified phases is to be printed.

*NONE: No intermediate dump is produced.

phase-name: Last two characters of phase name.

SNPDUMP Parameter: Specifies whether the major data areas are to be printed after the execution of one or more specified phases.

*NONE: No snap dump is produced.

phase-name: Last two characters of phase name.

CODELIST Parameter: Specifies whether a dynamic listing of the IRP is to be printed during execution of one or more specified phases.

*NONE: No code listing is produced.

**ALL:* A code listing is produced for each phase executed.

phase-name: Last two characters of phase name.

IGNDECERR Parameter: Specifies whether decimal data errors detected by the system are ignored by the program.

*NO: Decimal data errors are not ignored.

**YES:* Decimal data errors are ignored by the program. The result of the operation being performed when the error occurs is unknown. The compiler generates an error message on the compiler listing to notify the user that this option was specified. Incorrect results that occur during the execution of a program when this option is specified are the user's responsibility.

Example

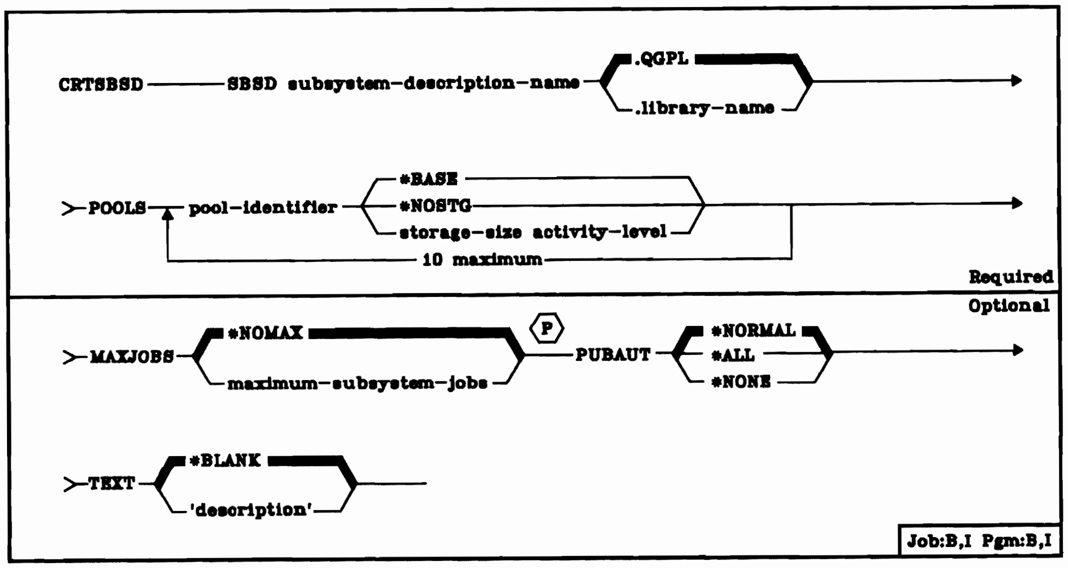
```
CRTRPTPGM  PGM(ARINQ5.JONES)  GENLVL(29)  TEXT('Accounts Receivable +
      Inquiry, Branch 5')
```

This command invokes the auto report function to generate and compile the RPG program called ARINQ5 in library JONES. The source program that is compiled resides in a member also named ARINQ5, in source file QRPGSRC. Any RPG source specifications from a different source file member can be copied into the source file member ARINQ5 by using the /COPY statement on the input specifications to name the existing source file member. The auto report program is generated first; if a program cannot be successfully generated because of errors in the auto report specifications, the auto report function terminates and escape message RPT 9001 is issued (the MONMSG command can be used to monitor for this message). Once an auto report program has been successfully generated, the program passes control to the RPG compiler. Any errors occurring during the compile of the RPG program will appear on a source listing of the RPG source input, printed on the system printer. If an error occurs with a severity level of 30 or higher, the compile will terminate.

CRTSBSD (Create Subsystem Description) Command

The Create Subsystem Description (CRTSBSD) command creates a subsystem description, which defines the operational attributes of a subsystem. After the subsystem description is created, it can be specialized by commands that add, change, and remove work entries and routing entries in the subsystem description.

Restriction: To use this command, you must have operational and add rights for the library into which the subsystem description is to be placed.



SBSD Parameter: Specifies the qualified name of the subsystem description being created. The subsystem description is stored in the specified library. (If no library qualifier is given, the subsystem description is stored in the general purpose library, QGPL.) Five IBM-supplied subsystem descriptions are shipped with the system; they are QCTL (in the QSYS library), QINTER, QBATCH, QSPL, and QPGMR (all in the QGPL library).

POOLS Parameter: Specifies one or more storage pool definitions that are to exist in this subsystem description. Each definition specifies for one storage pool:

- *Pool definition identifier:* The identifier, *within* the subsystem description, of the storage pool definition. The same identifiers (1 through 10 are valid) can be used for pool definitions in different subsystem descriptions.
- *Size:* The size of the storage pool, expressed in K-byte (1024 bytes) multiples. This is the amount of main storage that can be used by the pool.
- *Activity level:* The maximum number of jobs that can execute concurrently in the pool.

A maximum of 10 storage pool definitions can be specified for the subsystem description being created. Although each subsystem description can have as many as 10, there is an operational limitation on how many active storage pools there can be in the *system*. Within the system, no more than 16 storage pools can be active at any time, including the base storage pool and a machine storage pool. (A storage pool for which *NOSTG has been specified is not considered to be active, and it is not allocated to any subsystem.)

The base storage pool is the only pool that can be shared among subsystems. If a subsystem is started for which all of its storage pools cannot be allocated without exceeding the 16-pool *system* maximum, the pools that can be allocated (up to the limit) are allocated and the remainder are not. Then, for each routing step initiated by that subsystem that normally is routed into one of the pools that was not allocated, the base pool is used instead. For additional information about storage pools, see the *CPF Concepts* and the *CPF Programmer's Guide.*

pool-identifier: Enter the pool identifier (1 through 10) of the storage pool definition to be in this subsystem. The attributes of the pool also must be specified by one of the following values. As many as 10 sets of values can be specified in the POOLS parameter to define as many as 10 storage pools in the subsystem.

**BASE:* The specified pool definition is defined to be the base system pool, which can be shared with other subsystems. The size and activity level of the shared system pool are specified in the system values QBASPOOL and QBASACTLVL (see the *CPF Programmer's Guide.*)

**NOSTG:* No storage and no activity level are to be assigned to the pool initially. (It is to be inactive.)

storage-size activity-level: Enter the storage size in K-bytes that the specified storage pool is to have, and enter the maximum number of jobs that can execute concurrently in the pool. Both values must be specified. A value of at least 16 (meaning 16 K-bytes) must be specified for the storage size.

MAXJOBS Parameter: Specifies the maximum number of jobs allowed within the subsystem. The maximum applies to all initiated jobs that are waiting or executing, except for jobs on the job queue or jobs that have finished executing.

<u>*NOMAX:</u> There is no maximum number of jobs within this subsystem.

maximum-subsystem-jobs: Enter the maximum number of jobs to be allowed in this subsystem.

PUBAUT Parameter: Specifies what authority for the subsystem and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the Grant Object Authority (GRTOBJAUT) command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the subsystem description.

*ALL: The public has complete authority for the subsystem description.

*NONE: The public cannot use the subsystem description.

TEXT Parameter: Lets the user enter text that briefly describes the subsystem description. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTSBSD  SBSD(BAKER)  POOLS((1 *BASE)(2 200 4)) +
         TEXT ('Subsystem for running Baker +
         Department jobs')
```

This command creates a subsystem description named BAKER and stores it in the general purpose library (QGPL). Storage pool definition 1 specifies that pool 1 is to share the base system pool; the definition of storage pool 2 is to have 200 K of storage and an activity level of 4. There is no limit in this subsystem description on the number of jobs that can be active concurrently. The activity levels within the subsystem may, however, be controlled by MAXACT parameters specified in work station entries, job queue entries, and routing entries that are in the subsystem.

```
CRTSBSD  SBSD(MEDICAL.MEDLIB) +
         POOLS((1 150 2)(2 *BASE) (3 *NOSTG) +
         MAXJOBS(5) TEXT('Medical files +
         Inquiry and update')
```

This command creates a subsystem description named MEDICAL and stores it in the MEDLIB library. The subsystem description contains three storage pool definitions: storage pool 1 is defined to have 150 K of storage and an activity level of 2; pool 2 is to share the base system pool; and pool 3 is defined initially to be inactive when the other pools are active—it is to have no storage and no activity level. A maximum of five jobs can be active concurrently in this subsystem. A text string briefly describes the subsystem.

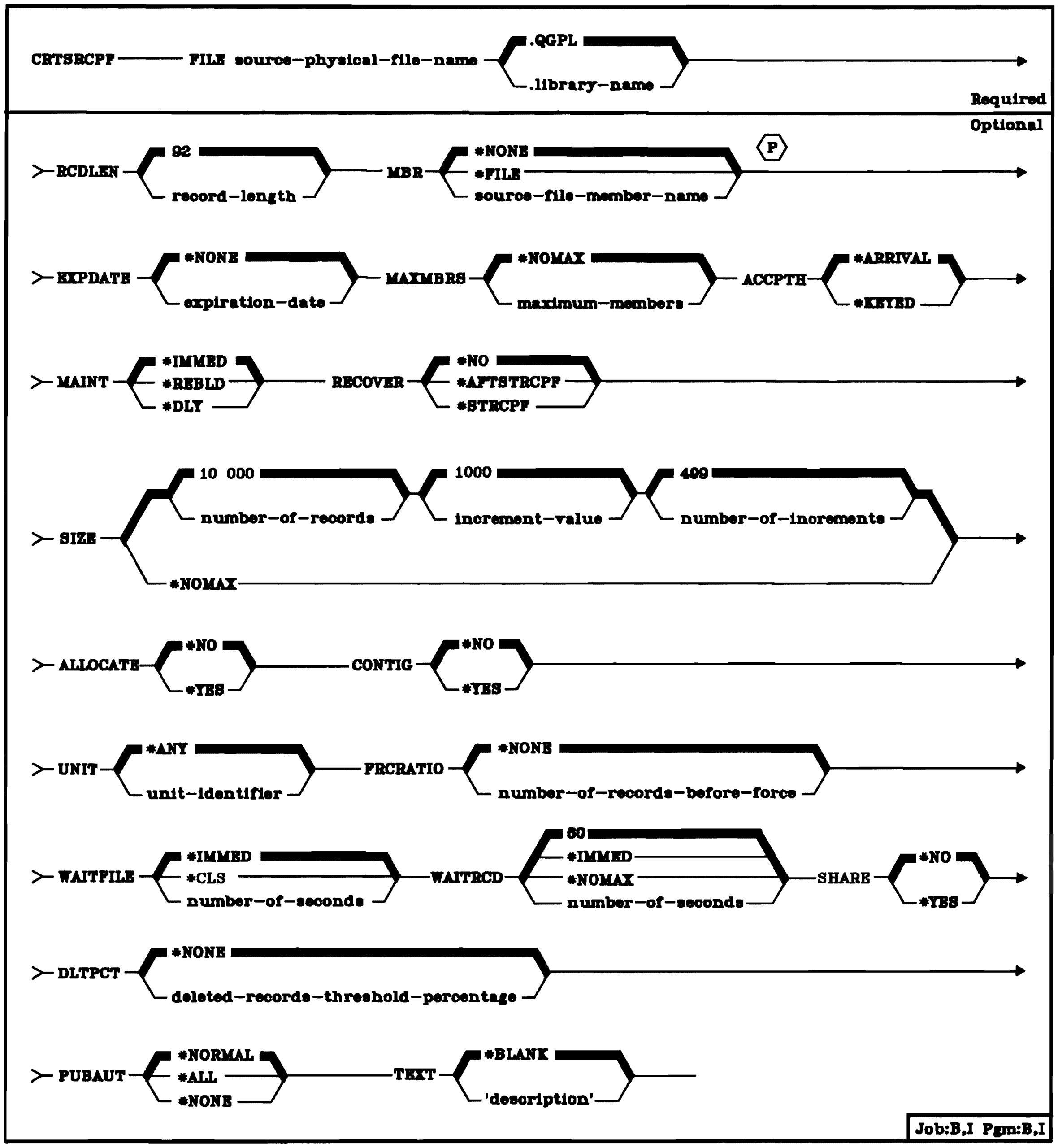
CRTSRCPF (Create Source Physical File) Command

The Create Source Physical File (CRTSRCPF) command creates a named source physical file in the data base. A source file is created from the file description parameters in the CRTSRCPF command; it is used to store source records to be used as input to IBM-supplied source processors, such as the data description specifications (DDS) processor, CL compiler, or RPG III compiler. (To override attributes of the file after it has been created, use the Override Data Base File (OVRDBF) command before the file is opened.)

A source file has only one record format, and the entire file contains records having only that format. All records in the file have the same length and have the same fields. (No level checking is performed for source files created by the CRTSRCPF command.)

Each source file can have one or more members; each source file member is a separate collection of records, whose record format is the same as all other members of the file. Each member within the file has its own access path, of the same type as the file itself.

No source records can be stored in the file being created until at least one member has also been added to the file. Either the MBR parameter of this command or the Add Physical File Member (ADDPFM) command can be used to add a member. However, the descriptive portion of the named file does exist within the data base even if there are no members.


CRTSRCPF
FILE source-physical-file-name
.QGPL
.library-name
Required
Optional
RCDLEN
92
record-length
MBR
*NONE
*FILE
source-file-member-name
P
EXPDATE
*NONE
expiration-date
MAXMBRS
*NOMAX
maximum-members
ACCPTH
*ARRIVAL
*KEYED
MAINT
*IMMED
*REBLD
*DLY
RECOVER
*NO
*AFTSTRCPF
*STRCPF
SIZE
10 000
number-of-records
1000
increment-value
499
number-of-increments
*NOMAX
ALLOCATE
*NO
*YES
CONTIG
*NO
*YES
UNIT
*ANY
unit-identifier
FRCRATIO
*NONE
number-of-records-before-force
WAITFILE
*IMMED
*CLS
number-of-seconds
WAITRCD
60
*IMMED
*NOMAX
number-of-seconds
SHARE
*NO
*YES
DLTPCT
*NONE
deleted-records-threshold-percentage
PUBAUT
*NORMAL
*ALL
*NONE
TEXT
*BLANK
'description'
Job:B,I Pgm:B,I

FILE Parameter: Specifies the qualified name by which the source physical file being created will be known. If no library qualifier is given, the physical file is stored in QGPL. (If the file is to be used in an HLL program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

RCDLEN Parameter: Specifies the record length, in bytes, of the records to be stored in the source file. The format of each record contains three fields: the sequence number of the record, a date field, and the source statement. The record format name is the same as that of the file itself, specified in the FILE parameter. For information about the fields in a source record, refer to the expanded parameter description of the FILETYPE parameter in Appendix A, and to the *CPF Programmer's Guide.*

The RCDLEN parameter must provide 12 positions for the source sequence number and date fields required in each record. If the Copy File (CPYF) command is used to copy records into the file, and the records are longer than the length specified here, the records are truncated on the right. These fields are defined with fixed attributes and names, and have a keyed access path over the sequence number. (See the *CPF Programmer's Guide* for details.)

92: The default record length is to be 92 characters. Six characters are for the record sequence number, six are for the record date, and the remaining 80 characters are for the source statement itself.

record-length: Enter a value, 13 through 32766, that indicates the record length of each source record in the file; the value must include 12 positions for the sequence number and date fields.

MBR Parameter: Specifies the name of the source file member (if a member is to exist immediately) to be added when the source file is created. (You can add other members to the file after it is created by using the ADDPFM command.)

*NONE: No member is to be added when the file is created.

**FILE:* The member being added is to have the same name as that of the source file that contains the member (specified in the FILE parameter).

source-file-member-name: Enter the name of the member that is to be added when the source file is created.

EXPDATE Parameter: Specifies, if a source file member is to be added when the file is created, the expiration date of the member. Any attempt to open a file that uses a member that has expired causes an error message to be sent to the user. (The expiration date of each member added later to the file must be specified in the Add Physical File Member (ADDPFM) command that adds it.)

*NONE: The member has no expiration date.

expiration-date: Enter the date after which the source file member should not be used. The date must be in the format specified by the QDATFMT and QDATSEP system values.

MAXMBRS Parameter: Specifies the maximum number of members that the source file being created can have at any time.

*NOMAX: No maximum is specified for the number of members; the system maximum of 32 767 members per file is used.

maximum-members: Enter the value for the maximum number of members that the source file can have. A value of 1 through 32767 is valid.

ACCPTH Parameter: Specifies the type of access path to be used by all members of the file.

*ARRIVAL: The access path is to be of arrival sequence order. Using this parameter value will reduce the size of the file and eliminate maintenance of the keyed access path.

**KEYED:* The access path is to be of keyed sequence order.

For more information on keyed and arrival sequence orders of source file access paths, refer to the *IBM System/38 CPF Programmer's Guide,* SC21-7730.

MAINT Parameter: Specifies the type of access path maintenance to be used for all members of the source file (which always have keyed access paths).

*IMMED: The access path is to be continuously (immediately) maintained for each source file member. The path is updated each time a record is changed, added to, or deleted from the member. The records can be changed through a logical file that uses the physical file member regardless of whether the source file is opened or closed.

**REBLD:* The access path is to be completely rebuilt when a file member is opened during program execution. The access path is continuously maintained until the member is closed; the access path maintenance is then terminated.

**DLY:* The maintenance of the access path is to be delayed until the physical file member is opened for use. Then, the access path is updated only for records that have been added, deleted, or updated since the file was last opened. (Records that change while the file is open have their access paths immediately rebuilt.) To prevent a lengthy rebuild time when the file is opened, *DLY should be specified only when the number of changes to the access path is small.

RECOVER Parameter: Specifies, for files having immediate maintenance on their access paths, when recovery processing of the file is to be performed after a system failure has occurred while the access path was being changed. This parameter is valid only if a keyed access path is used.

The access path having *immediate maintenance* can be rebuilt during start CPF (before any user can execute a job), or after start CPF has finished (during concurrent job execution), or when the file is next opened. While the access path is being rebuilt, the file cannot be used by any job.

The access path having *rebuild maintenance* will be rebuilt the next time its file is opened, the time that it normally is rebuilt.

**NO:* The access path of the file is not to be rebuilt. The file's access path is rebuilt when the file is next opened if it has rebuild maintenance.

**AFTSTRCPF:* The file is to have its access path rebuilt after the start CPF operation has been completed. This option allows other jobs not using this file to begin processing immediately after the CPF has been started. If a job tries to allocate the file while its access path is being rebuilt, a file open exception occurs if the specified wait time for the file is exceeded.

**STRCPF:* The file is to have its access path rebuilt during the start CPF operation. This ensures that the file's access path will be rebuilt before the first user program tries to use it; however, no jobs can begin execution until after all files that specify RECOVER(*STRCPF) have their access paths rebuilt.

SIZE Parameter: Specifies the *initial* number of records in each member of the file, the number of records in each increment that can be automatically added to the member size; and the number of times the increment can be automatically applied. The number of records for each file member is expressed as the number of *undeleted* records that can be placed in it.

When the maximum number of records has been reached, a message (stating that the member is full) is sent to the system operator, giving him the choice of terminating the job or extending the member size himself. The operator can extend the member by the amount specified as the increment value (in the second value) one time for each time he receives the message.

A list of three values can be specified to indicate the initial size of each member and the automatic extensions that can be added when needed. Or *NOMAX can be specified instead. If SIZE is not specified, SIZE(10 000, 1000, 3) is assumed by the system.

Records: One of the following is used to specify the *initial* number of records in the member before any automatic extension of the member occurs. The ALLOCATE parameter determines when the required space for the initial allocation occurs: If *YES is specified, the space is allocated when the file is created, or when a new member is added. If *NO is specified, the initial space is allocated as determined internally by the system.

10 000: Initially, up to 10 000 records can be inserted into each member of the file before any extension occurs.

number-of-records: Enter the number of records that are inserted before an automatic extension occurs. A value of 0 cannot be used; the maximum value cannot exceed 16 777 215 records, or, if ALLOCATE(*YES) is specified, the amount of total system storage remaining for all permanent objects, whichever is less. If you do not want any automatic extensions, enter a 0 for the second and third values in the list.

Increment Amount: One of the following is used to specify the maximum number of records that can be additionally inserted in the member when the initial member size is exceeded and an automatic extension is made.

1000: A maximum of 1000 additional records can be inserted into the member after an automatic extension occurs.

increment-value: Enter the value, 0 through 32767, that specifies the maximum number of additional records that can be inserted into the member after an automatic extension occurs. Enter a 0 to prevent automatic extensions.

Number of Increments: One of the following is used to specify the maximum number of increments that can be automatically added to the member. If 0 is specified for the increment amount, the number of increments need not be specified; 0 will be the default value instead of 499 (and a message is sent to the user issuing the command).

499: A maximum of 499 increments can be automatically added to the member size.

number-of-increments: Enter the maximum number of increments, 0 through 32767, that can be automatically added to the member. Enter a 0 to prevent automatic extensions.

Unlimited Size: The following value can be specified to allow an unlimited number of records in each member.

**NOMAX:* The number of records that can be inserted into each member of the file is not limited by the user. The maximum size of each member is determined by the system. If *NOMAX is specified, ALLOCATE(*NO) must also be specified.

ALLOCATE Parameter: Specifies whether an *initial* allocation of storage space is to be performed for each source file member as it is added. The allocation provides enough space to hold the initial number of records specified by the SIZE parameter. (Later allocations, which occur when a record cannot be added to a member without exceeding its capacity, are determined by the system and by the SIZE parameter values.)

*NO: Minimum storage space is initially allocated when the member is added. The system determines when space allocations are necessary and the size of each allocation.

**YES:* Storage space is to be initially allocated as each member is added. The amount specified in the first value of the SIZE parameter (the number of records) is allocated. If the space cannot be allocated, a message is sent to the user and the affected member is not added. If *YES is specified, SIZE(*NOMAX) cannot be specified.

CONTIG Parameter: The contiguous parameter specifies whether the user prefers that all records in the initial allocation in each source file member are stored together without separations. If so, and the necessary contiguous space is not available, the system sends a message to the job log and allocates the storage space noncontiguously. The file is still entirely usable. This parameter does not indicate anything about the additional allocations that might be needed later, which most likely would be noncontiguous.

*NO: The storage space for each member does not have to be contiguous.

**YES:* The user wants the system to allocate contiguous space for each member of the source file being added, and to notify the user and put a message in the job log if it cannot. The affected member is still added, even if the storage space has to be allocated noncontiguously. The member is just as usable in noncontiguous form. If *YES is specified for CONTIG, then ALLOCATE(*YES) must also be specified.

UNIT Parameter: Specifies, if the user prefers that a file be stored on a specific unit, the unit identifier of the auxiliary storage unit on which the system will attempt to allocate the storage space for the file and for all its members and their associated access paths. This includes the initial allocation when each member is added and any extensions that occur later for each member in the file. If the system cannot allocate the storage space for each member on the specified unit, it allocates the space on any available unit and sends a message to the job log. The file is entirely usable in all cases.

*ANY: The storage space for the file and its members can be allocated on any available auxiliary storage unit.

unit-identifier: Enter a valid value of 1 through 14 to specify the identifier of the auxiliary storage unit on which you prefer to have the storage space of all members allocated. The values that are valid depend on how many storage units are on the system, and on their types (62PC disk and 3370 disk). Refer to the chart in the CRTPF command, UNIT parameter, for the type and unit that correspond to the unit identifiers.

The system attempts to make all space allocations on the unit specified. If it cannot, either because that unit is full or because an invalid identifier was specified, it allocates the remainder of the space on any available unit and sends a message to the job log.

FRCRATIO Parameter: The force write ratio parameter specifies the number of inserted or updated records that are processed before they are forced into auxiliary (permanent) storage. (For an expanded description of the FRCRATIO parameter, see Appendix A.)

<u>*NONE:</u> There is no force write ratio; the system determines when the records are written in auxiliary storage.

number-of-records-before-force: Enter the number of new or changed records that are processed before they are explicitly forced into auxiliary storage.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

<u>*IMMED:</u> The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

WAITRCD Parameter: Specifies the number of seconds that the program is to wait for a record that is to be updated or deleted. If the record cannot be allocated in the specified wait time, an error message is sent to the program.

60: The program is to wait for 60 seconds.

**IMMED:* The program is not to wait; when a record is locked, an immediate allocation of the record is required.

**NOMAX:* The wait time will be the maximum allowed by the system (32 767 seconds).

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the job. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the source file member can be shared with other programs in the same routing step. If so, when the same file is opened by other programs that also specify SHARE(*YES), they use the same ODP to the file. If a program that specifies SHARE(*NO) opens the file, a new ODP is used. This parameter is not valid if a member is not being added when the source file is created.

When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

*NO: An ODP created by the program in which this command is used is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

DLTPCT Parameter: Specifies the maximum percentage of deleted records that any member in the source file should have. The percentage is based on the number of deleted records compared with the total record count in a member. The percentage check is made when any member of the file is closed, and if the number of deleted records exceeds the percentage, a message is sent to the system history log to inform the user.

*NONE: No percentage is to be specified; the number of deleted records in the file members is not to be checked when a member is closed.

deleted-records-threshold-percentage: Enter a value, 1 through 100, that specifies the largest percentage of deleted records in any member in the file can have. If this percentage is exceeded, a message is sent to the system history log whenever the file is closed. This check will be made for logical file processing also; if more than one based-on physical file has its percentage exceeded, a message is logged for each one that was exceeded.

PUBAUT Parameter: Specifies what authority for the source file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the Grant Object Authority (GRTOBJAUT) command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has operational, read, add, delete, and update rights for the source file.

**ALL:* The public has complete authority for the file.

**NONE:* The public cannot use the file.

TEXT Parameter: Lets the user enter text that briefly describes the source file. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTSRCPF  FILE(PAYTXS.SRCLIB)
```

This command creates a source file named PAYTXS that is to be stored in the SRCLIB library. The file is created without any members; therefore, no data can be put into the file until a member is added later. As many as 32 767 members (*NOMAX) can be added to the file.

The access path of each member is continuously maintained. Each member can have up to 20 records before automatic extensions (499 increments maximum) occur that add 20 records to the capacity of the member. The initial storage allocated for each member will hold 20 records, with no restrictions on which unit is used or whether the space is contiguous; there is no initial storage allocation. The public has operational, read, add, delete, and update authority for the file, but no object rights.

```
CRTSRCPF  FILE(ORDERS.ORDERCTL) +
     SIZE(100 50 5) UNIT(01)
```

This command creates a source file and source file member, both named ORDERS, to be stored in the ORDERCTL library. The user prefers that all records placed in the file are to be stored on auxiliary storage unit 01, but the space does not have to be contiguous. The initial allocation of storage provides for a maximum of 100 records, and up to five increments of additional space for 50 records each can be added automatically. These allocation values also apply to each member of this source file that is added later.

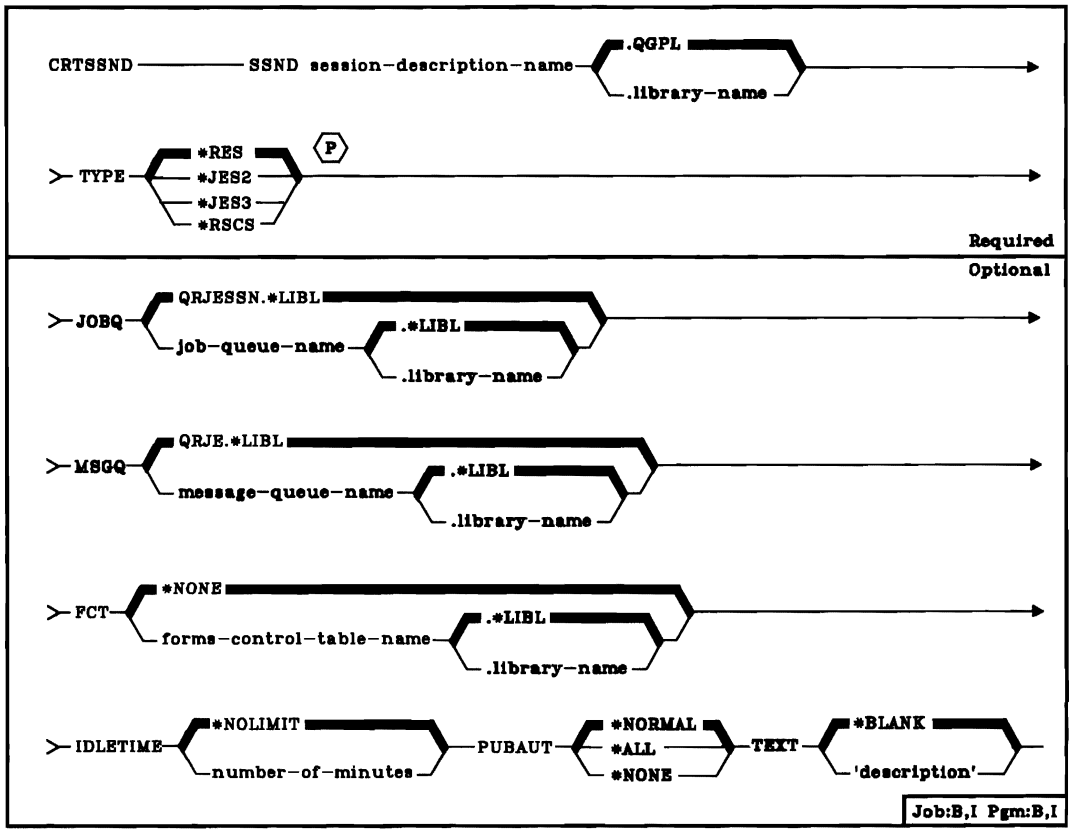
CRTSSND (Create Session Description) Command

The Create Session Description (CRTSSND) command creates a session description that defines the attributes of an RJEF session. A session description is necessary for each RJEF session.

After the session description is created, it can be specialized through commands that add RJEF reader entries, RJEF writer entries, and communications entries. Refer to the Add RJE Reader Entry (ADDRJERDRE), Add RJE Writer Entry (ADDRJEWTRE), and Add RJE Communications Entry (ADDRJECMNE) commands respectively.

Restriction: To use this command, you must have add rights to the library in which the session description is to be stored.

The Create Session Description (CRTSSND) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSND Parameter: Specifies the qualified name of the session description that is to be created. (If no library qualifier is given, the session description is stored in QGPL.)

TYPE Parameter: Specifies the type of remote job entry host subsystem with which RJEF session is to communicate. Enter the value that applies to this session description.

**RES:* VS1/RES.

**JES2:* VS2/JES2.

**JES3:* VS2/JES3.

**RSCS:* VM/370 RSCS.

JOBQ Parameter: Specifies the name of the default RJEF job queue on which all the RJEF session jobs are to be started.

QRJESSN.*LIBL: The RJEF job queue named QRJESSN in the QRJE library is to be used for this session description. (If no library qualifier is specified, *LIBL is used to find the job queue.)

job-queue-name: Enter the qualified name of the job queue on which all the RJEF session jobs are to be started. (If no library qualifier is given, *LIBL is used to find the job queue.)

MSGQ Parameter: Specifies the qualified name for the RJEF message queue in which all the RJEF messages are to be recorded.

This message queue will contain all the messages received from the host system, all commands sent to the host system, and all the messages issued by RJEF jobs. In addition, this message queue serves as a job log for all RJEF jobs in the active RJEF session. The message queue can be displayed by issuing the STRRJECSL command.

QRJE.*LIBL: The RJEF message queue named QRJE in the QRJE library is to be used for this session description. If no library qualifier is specified, *LIBL is used to find the message queue.

message-queue-name: Enter the qualified name of the message queue that is to contain a record of all the RJEF messages for this session description. (If no library qualifier is given, *LIBL is used to find the message queue.)

FCT Parameter: Specifies a forms control table (FCT) to be used with this session description.

*NONE: No FCT is to be specified.

forms-control-table-name: Enter the qualified name of the FCT that is to be used. (If no library qualifier is given, *LIBL is used to find the FCT.)

IDLETIME Parameter: Specifies the minimum number of minutes that the RJEF session should remain idle after the line connection has been established before transmitting the LOGOFF or SIGNOFF command to the host system. During this time no files are transmitted or received.

When the number of minutes is set equal to zero, and if the line connection has been established, the LOGOFF or SIGNOFF command is transmitted immediately. Also, RJEF holds all RJEF reader job queues defined for this RJEF session.

The idle time countdown begins following the end-of-file of the last input stream sent or output stream received.

The idle time countdown is reset each time data becomes available for transmitting or receiving.

If there are any input streams that have started but have not ended (that is, received end of file) except for the console input streams, the idle time countdown will not begin.

If a Terminate RJE Session (TRMRJESSN) command is issued, it overrides the session idle time processing. If the TRMRJESSN command specifies a controlled cancel, the IDLETIME parameter value of the TRMRJESSN command overrides the CRTSSND command IDLETIME parameter value. This parameter is ignored if OPTION(*IMMED) is specified in the TRMRJESSN command.

*NOLIMIT: A LOGOFF or SIGNOFF command is not to be transmitted unless a TRMRJESSN command is issued specifying OPTION(*CNTRLD).

number-of-minutes: Enter the number of minutes that the RJEF session should remain idle before transmitting the LOGOFF or SIGNOFF command to the host system. Valid values are 0 through 99.

PUBAUT Parameter: Specifies the authority that is being granted to the public (all users) for the session description. Additional authority can be explicitly granted to users by the Grant Object Authority (GRTOBJAUT) command.

NORMAL: The public has only operational rights for the session description.

**ALL:* The public has complete authority for the session description.

**NONE:* The public cannot use the session description.

TEXT Parameter: Lets the user enter text that briefly describes the session description. (For an expanded description of the TEXT parameter, see Appendix A.)

BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTSSND SSND(RJE.USERLIB) +
      TYPE(JES2) +
      JOBQ(RJEF.USERLIB) +
      MSGQ(RJEF.USERLIB) +
      FCT(FCT.USERLIB) +
      TEXT('Session description for JES2 host')
```

This command creates a session description named RJE in library USERLIB. The host type is JES2. The RJE job queue, named RJEF, is in library USERLIB. The RJEF message queue is named RJEF.USERLIB. This session will use forms control entries from a forms control table named FCT in library USERLIB. A text string briefly describes the session description.

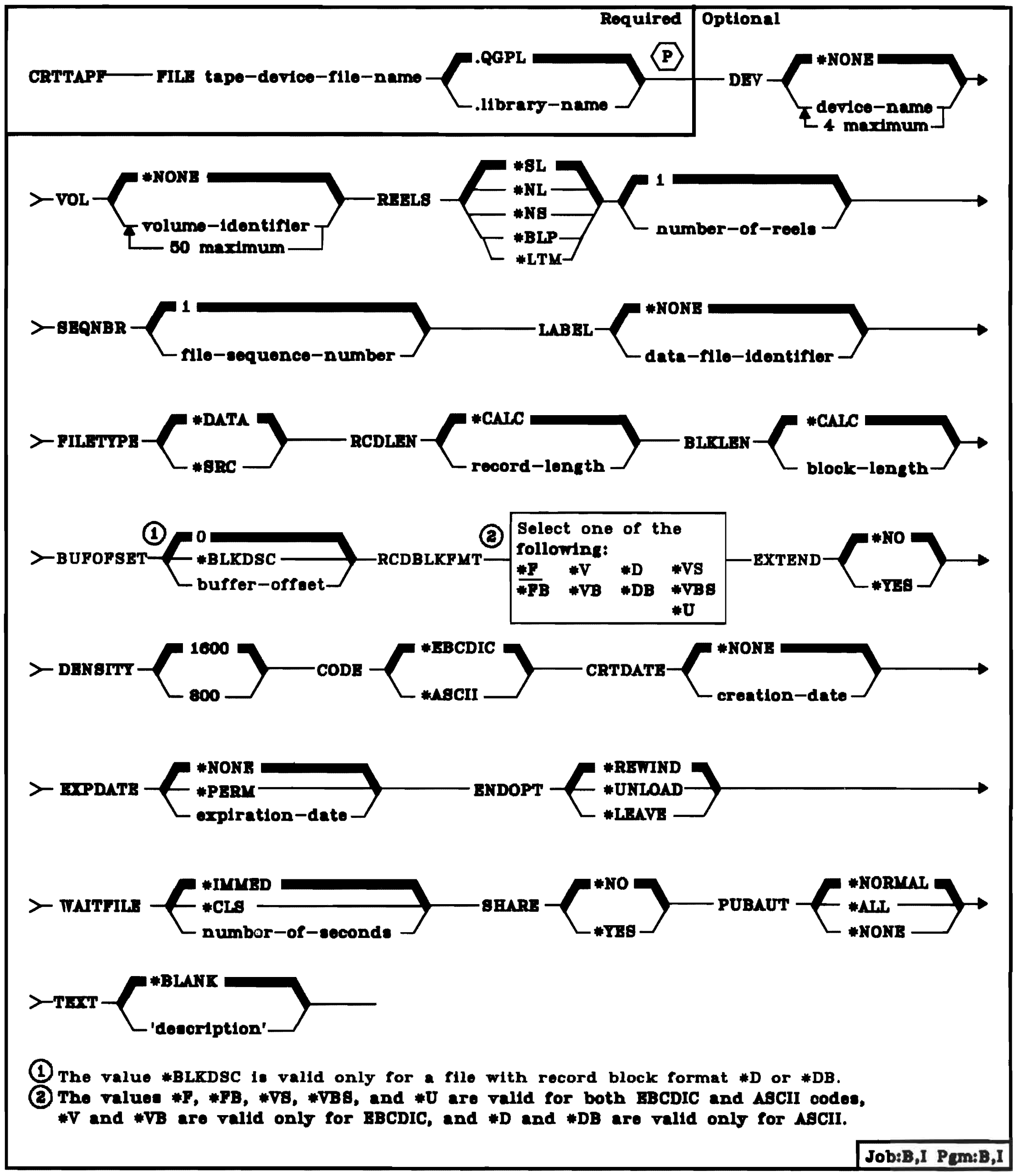
CRTTAPF (Create Tape File) Command

The Create Tape File (CRTTAPF) command creates a tape device file. The device file contains the file description, which identifies the device to be used; the device file does not contain data. The tape device file is used to read and write records on tape. The same device file can be used for both input and output operations.

Note: This command is not to be used to create device files for use in save/restore operations. User-created device files are not needed for save/restore operations.

All the information in the tape file description comes from the command that creates it; there is no DDS (data description specifications) for tape device files. The tape file has only one record format for input/output operations. The record format consists of one character field containing the input data retrieved from the device or the output data to be written to the device. The program using the device file must describe the fields in the record format so that the program can arrange the data received from or sent to the device in the manner specified by the tape file description.

The CHGTAPF or OVRTAPF command can be used in a program to change or override the parameter values specified in the tape file description. Each changed value in the device file remains changed after the program ends. Each overridden value remains altered only for the execution of the program (unless the override is deleted by a Delete Override (DLTOVR) command); once the program ends, the original parameter values specified for the tape file are used. Override commands must be executed before the tape file to be affected is opened for use by the program.



(1) The value *BLKDSC is valid only for a file with record block format *D or *DB.
(2) The values *F, *FB, *VS, *VBS, and *U are valid for both EBCDIC and ASCII codes, *V and *VB are valid only for EBCDIC, and *D and *DB are valid only for ASCII.

FILE Parameter: Specifies the qualified name by which the tape device file being created is to be known. If no library qualifier is given, the file is stored in QGPL. (If the file is to be used by an HLL (high-level language) program, the file name should be consistent with the naming rules of that language; otherwise, the file must be renamed in the program itself.)

DEV Parameter: Specifies the names of one or more tape devices to be used with this tape device file to perform input/output data operations.

*NONE: No device names are to be specified. They must be specified later in the CHGTAPF or OVRTAPF command, or in the HLL program that opens the file.

device-name: Enter the names of one or more devices (no more than four) that are to be used with this tape device file. The order in which the device names are specified here is the order in which tapes mounted on the devices are processed. When more volumes are to be processed than the number of devices in the DEV list, the devices are used in the same order as specified, wrapping around to the first device as needed. Each device name must already be known on the system via a device description before this device file is created.

VOL Parameter: Specifies volume identifiers for the tapes to be used by this device file. The tapes (volumes) must be mounted on the devices in the same order as the identifiers are specified here and as the device names are specified in the DEV parameter. If the tape file is opened for read backward, the volume identifiers in the list are processed from last to first (while the devices in the device list are used in first to last order). An inquiry message is sent to the system operator if either *SL or *BLP processing is specified or if an incorrect volume is mounted, or if no volume is mounted (for any type of label processing). When a list of volume identifiers is provided for the file, operator mount messages indicate the name of the volume which is required. (For an expanded description of the VOL parameter, see Appendix A.)

*NONE: No tape volume identifiers are specified for this file. They can be supplied before the device file is opened, either in the CHGTAPF or OVRTAPF command or in the HLL program. If no volume identifiers are specified before the device file is opened, no volume checking is performed beyond verifying that the correct label type volume is mounted, and no volume names are provided in operator mount messages. The maximum number of reels processed for a *NL, *NS, *BLP, or *LTM input file when VOL(*NONE) is specified is determined by the REELS(number-of-reels) parameter value.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used by this device file. Each identifier can have 6 alphameric characters or less. These identifiers are used in any mount messages that are sent to the operator during processing. The maximum number of reels processed for a *NL, *NS, *BLP, or *LTM input file is determined by the number of volume identifiers in the list.

REELS Parameter: Specifies the type of labeling used on the tape reels and the maximum number of reels to be processed, if there is no list of volume identifiers specified (VOL parameter) and this device file is used with either *NL, *LTM, *NS, or *BLP input files. When the number of reels are specified, the volume identifiers on the mounted volumes are ignored if labeled tapes are being processed; instead, the order in which the reels are mounted must be checked by the operator.

The number of reels value (the second part of the REELS parameter) is not a limiting value for standard-label or output files. For a standard-label input file, the data file labels limit the number of volumes processed by indicating end-of-file. For an output file, the maximum number of reels value is ignored; the system requests that additional volumes be mounted until the file is closed.

The system checks the first record following the load point on the tape to see (1) if it has exactly 80 bytes for EBCDIC or at least 80 bytes for ASCII and (2) if the first 4 bytes contain the values VOL and 1. If so, the reel contains a standard-label tape. *SL and *BLP files require standard-label tape volumes. *NL, *LTM, and *NS tape files cannot process standard-label volumes.

Note: The values *SL, *NL, and *LTM can be specified if the device file is to be used for either reading or writing on tapes. The values *NS and *BLP are valid only if the device file is used to read tapes.

SL: The volumes have standard labels. If a list of volume identifiers is specified (with the VOL parameter), the system checks that the correct tape volumes are mounted in the specified sequence. If no volume identifier list is given and the file is opened for output, any standard-label volumes may be mounted. If no volume identifier list is given and the file is opened for input, the first volume may have any volume identifier, but if the file is continued, the system requires the correct continuation volumes to be processed (verified by checking the data file labels). For an input file, the end-of-file message will be sent to the using program when the labels on the last volume processed indicate that it is the last volume for the data file.

NL: The volumes have no labels. On a nonlabeled volume, tape marks are used to indicate the end of each data file and the end of the volume. For an input file, the end-of-file message will be sent to the using program when the number of volumes specified in the volume list have been processed, or (if no list of volume identifiers is provided) when the number of reels specified in the REELS parameter have been processed.

**NS:* The volumes have nonstandard labels. Each volume must begin with some kind of label information, optionally preceded by a tape mark (and always followed by a tape mark). This nonstandard-label information is ignored, and the system spaces forward to a point beyond the tape mark that follows the nonstandard labels to position the tape at the file's data. Each reel must have a tape mark at the end of the file's data. Any information beyond this ending tape mark is ignored. Only a single data file can exist on a nonstandard tape. Standard-label volumes *cannot* be processed using *NS label processing. For an input file, the end-of-file message will be sent to the using program when the number of volumes specified in the volume list have been processed, or, if no list of volume identifiers is provided, when the number of reels specified in the REELS parameter have been processed.

**BLP:* Standard-label processing is to be bypassed. Each reel *must* have standard labels. Although each reel is checked for a standard volume label and each file must have at least one standard header label (HDR1) and one standard trailer label (EOV1 or EOF1), most other label information (such as the data file record length or block length) is ignored. The sequence number of each file on the volume is determined only by the number of tape marks between it and the beginning of tape (in contrast to *SL processing where the file sequence number stored in the header and trailer labels of each file are used to locate a data file).

Most of the information in the data file trailer label is ignored, but if an end-of-file (EOF) trailer label is found, the end-of-file message is signaled to the program using the tape file. If no end-of-file trailer label is encountered by the time the specified number of volumes or reels have been processed (volume identifier list and REELS parameter), the end-of-file message is immediately sent to the program using the tape file. Bypass label processing can be used when you do not know the name of the file to be used or (for example) when some file label information is incorrect.

**LTM:* The volumes have no labels but have a single leading tape mark before the first data file. REELS(*LTM) is processed the same way as REELS(*NL) except that when SEQNBR(1) is specified for an output file to create the first data file on the tape, a leading tape mark is written at the beginning of the tape before the first data block.

1: A maximum of one tape reel can be processed for the *NL, *LTM, *NS, or *BLP tape file input operation if there is no list of volume identifiers provided (VOL parameter).

number-of-reels: Enter the maximum number of reels that are to be processed for a *NL, *LTM, *NS, or *BLP input tape operation when there is no list of volume identifiers specified (VOL parameter). If the next reel is not mounted when the end of the currently-processing tape is reached, a message is sent to the operator requesting that the next tape be mounted on the next tape device. The number-of-reels value is ignored for a standard label (*SL) file or for any output file.

SEQNBR Parameter: Specifies the sequence number of the data file on the tape that is to be processed. When standard-label tapes are used, the four-position file sequence number is read from the first header label of the data file. When bypass label processing is used or when standard-label tapes are *not* used, the system counts the tape marks from the beginning of the tape to locate the correct sequence number data file to be processed. (When multifile, multivolume tapes are processed using REELS(*SL), the file sequence numbers continue consecutively through all of the volumes; that is, each new data file has a sequence number that is one greater than the previous file, regardless of which volume it is on.)

1: For standard-label tapes (not using bypass label processing), the data file having the sequence number 1 is the file to be processed. For nonlabeled tapes and for bypass label processing of standard-label tapes, the first data file on the tape is to be processed.

file-sequence-number: Enter the sequence number of the file to be processed on this tape.

LABEL Parameter: Specifies the data file identifier of the data file that is to be processed by this tape device file. The data file identifier is defined for only standard-label tapes and is stored in the header label immediately preceding the data file the label describes. If a data file identifier is specified for any type of label processing other than *SL, it is ignored. A label identifier is *required* for a standard label output file, but is optional for an input file (since the sequence number uniquely identifies which data file to process).

For an input file or output file with EXTEND(*YES) specified, this parameter specifies the data file identifier of the file that exists on the tape. The specified identifier must be the same as the one in the labels of the data file that the SEQNBR parameter specifies; otherwise, an error message is sent to the program using this device file. For output files with EXTEND(*NO) specified, the LABEL parameter specifies the identifier of the file that is to be created on the tape. (For an expanded description of the LABEL parameter, see Appendix A.)

*NONE: The data file identifier is not specified.

data-file-identifier: Enter the identifier (17 alphameric characters maximum) of the data file to be used with this tape device file. If this identifier is for a tape that is written in the basic exchange format, and it is to be used on a system other than System/38, a maximum of 8 characters should be used or a qualified identifier having no more than 8 characters per qualifier should be used. (See Appendix A for details.)

FILETYPE Parameter: Specifies whether the tape device file being created describes data records or describes source records (statements) for a program or another file. (For an expanded description of the FILETYPE parameter, see Appendix A.)

*DATA: The tape file describes data records.

**SRC:* The tape file describes source records.

Note: If *SRC is specified, the system will add 12 bytes to the beginning of every record (to replace the sequence number and date fields).

RCDLEN Parameter: Specifies, in bytes, the length of the records contained in the data file that is to be processed with this device file. The system will always use the record length and block length specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified (if a second header label (HDR2) is found on the tape and *BLP label processing has not been specified).

*CALC: No record length is specified for the data file to be processed. If *CALC is specified the system will attempt to calculate an appropriate record length when the file is opened. RCDLEN(*CALC) can be used for nonlabeled tapes or when there is no HDR2 label if a BLKLEN value other than *CALC is specified for the file and the RCDBLKFMT does not specify spanned or blocked records. In this case, the system calculates an appropriate record length from the block length, record block format, and buffer offset (for an ASCII file) specified for the file. In any other case, the actual record length must be specified by a CHGTAPF or OVRTAPF command, or in the HLL program that opens the device file.

record-length: Enter a value (1 through 32767) that specifies the length of each record in the data file. The minimum and maximum record length that will be allowed for a file is dependent on the record block format, block length, buffer offset (for an ASCII file), and recording code. The following table shows the minimum and maximum record length values allowed for each record block format, assuming the block length value is large enough to support the maximum record length:

Absolute RCDLEN Ranges

		FILETYPE(*DATA)		FILETYPE(*SRC)	
CODE	RCDFBLKFMT	Minimum RCDLEN	Maximum RCDLEN	Minimum RCDLEN	Maximum RCDLEN
*EBCDIC	*F *FB *U	18	32767	30	32767
*ASCII	*F *FB *U	18	32767	30	32767
*EBCDIC	*V *VB	1	32759	13	32767
*ASCII	*D *DB	1	9995	13	10007
*EBCDIC	*VS *VBS	1	32759	13	32767
*ASCII	*VS *VBS	1	32759	13	32767

BLKLEN Parameter: Specifies, in bytes, the maximum length of the data blocks that will be transferred to or from the tape for output or input operations. The system will always use the block length and record length specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified (if a second header label (HDR2) is found on the tape and *BLP label processing has not been specified).

*CALC: No block length is specified for the data file to be processed. If *CALC is specified the system will attempt to calculate an appropriate block length when the file is opened. BLKLEN(*CALC) can be used for nonlabeled tapes or when there is no HDR2 label if a RCDLEN value other than *CALC is specified for the file and the RCDBLKFMT does not specify spanned or blocked records. In this case, the system calculates an appropriate block length from the record length, record block format, and buffer offset (for an ASCII file) specified for the file. In any other case, the actual block length must be specified by a CHGTAPF or OVRTAPF command, or in the HLL program that opens the device file.

block-length: Enter a value, not exceeding 32767 bytes, that specifies the maximum length of each block in the data file to be processed. The minimum block length which can be successfully processed is determined by the tape device hardware and System/38 machine support functions. The minimum value for the 3410/3411 tape drive is 18 bytes. The maximum block length is always 32767 for an input file, but is limited to 9999 if block descriptors must be created for an ASCII output file. The following table shows the minimum and maximum block length values allowed for an output file:

Absolute BLKLEN Ranges

CPDE	BUFOFSET	Minimum BLKLEN	Maximum BLKLEN
*EBCDIC	ignored	18	32767
*ASCII	0	18	32767
*ASCII	*BLKDSC	18	9999

BUFOFSET Parameter: Specifies the buffer offset value for the start of the first record in each block in the tape data file. A buffer offset value can be used for any record block format ASCII file, and is ignored for an EBCDIC tape file. The system will always use the buffer offset specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified if a value is contained in the second header label (HDR2) on the tape, and *BLP label processing has not been specified.

The buffer offset parameter specifies the length of any information that precedes the first record in the block. For record block formats *D, *DB, *VS, and *VBS each record or record segment is preceded by a descriptor that contains the length of the record or segment. A buffer offset value is used to indicate that there is information *ahead* of the descriptor word for the first record in each block, or *ahead* of the data of the first fixed-length or undefined format record in each block.

This parameter is not needed for a standard label file processed for input if the tape includes a second file header label (HDR2) that contains the buffer offset value. A buffer offset must be provided by the CRTTAPF, CHGTAPF, or OVRTAPF command, or by the file labels for an input file that contains any information (such as a block descriptor) ahead of the first record in each block. If you do not specify a buffer offset when a tape file is created, it is not necessary to specify an offset value when the file is read.

The only buffer offset values allowed for an output file are zero and *BLKDSC. An existing standard label data file with a buffer offset value in the HDR2 label can be extended only if the offset value is either zero or four. An offset of zero in the HDR2 label adds data blocks with *no* buffer offset. BUFOFSET(*BLKDSC) must be specified to extend an existing tape data file that contains an offset value of four in the HDR2 label.

0: Specifies that no buffer offset information will precede the first record in each data block.

**BLKDSC:* Specifies that 4-byte block descriptors are to be created in any tape file created using this device file, and that any input file read using this device file should assume 4-bytes of buffer offset information preceding the first record in each data block. This value is only valid for a record block format *D or *DB file. The contents of the buffer offset part of each output data block when BUFOFSET(*BLKDSC) is specified is the actual length of the data block, in zoned decimal format.

buffer-offset: Enter a value (zero through 99) that specifies the length of the buffer offset information that precedes the first record in each data block.

RCDBLKFMT Parameter: Specifies the type and blocking attribute of records in the tape data file to be processed.

Record block format *V and *VB records can only be processed for an EBCDIC file; *D and *DB records can only be processed for an ASCII file. If a standard label tape (label type *SL or *BLP) is being processed and an inconsistent record block format is specified for the volume code, the correct record type is assumed (V or D) for the volume code and a warning message is sent to the progam that opens the file. If the record type and code are inconsistent for a nonlabeled volume (label type *NL, *LTM, or *NS), an error message is sent and the file is *not* opened, because there are no labels to verify the correct volume code.

If a valid record length, block length, and buffer offset (for an ASCII file) are specified for fixed length records but the block attribute is incorrect, the correct block attribute will be assumed (changing record block format *F to *FB or record block format *FB to *F), and a warning message sent to the program that opens the file.

If a block length is specified that is longer than required to process a maximum length record, then record block format *V, *D, or *VS will be changed to *VB, *DB, or *VBS and a warning message sent to the program that opens the file.

The following chart shows the required relationship between the record length, block length, and buffer offset (for ASCII) file parameters for an output file or an input file where the file parameters are not determined from a second file header label (HDR2):

Required RCDLEN/BLKLEN/BUFOFSET Relation[1]

CODE	RCDBLKFMT	BLKLEN = fcn(RCDLEN,BUFOFSET)
*EBCDIC	*F *U	BLKLEN = RCDLEN
*ASCII	*F *U	BLKLEN = RCDLEN + BUFOFSET
*EBCDIC	*FB	BLKLEN = RCDLEN * n
*ASCII	*FB	BLKLEN = (RCDLEN * n) + BUFOFSET
		n is the number of records in a maximum-length block
*EBCDIC	*V	BLKLEN = RCDLEN + 8
*ASCII	*D	BLKLEN = RCDLEN + 4 + BUFOFSET
*EBCDIC	*VB	BLKLEN >= RCDLEN + 8
*ASCII	*DB	BLKLEN >= RCDLEN + 4 + BUFOFSET
*EBCDIC	*VS *VBS	BLKLEN >= 18
*ASCII	*VS *VBS	BLKLEN >= 6 + BUFOFSET (18 minimum)

[1]When BUFOFSET(*BLKDSC) is specified for the file, a value of 4 should be used for the BUFOFSET part of any BLKLEN calculations, unless existing file labels on the tape specify a different value.

**F:* Fixed length, unblocked, unspanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *FB, based on other file parameters. See the previous explanation for more information.

**FB:* Fixed length, blocked, unspanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *F, based on other file parameters. See the previous explanation for more information.

**V:* Variable length, unblocked, unspanned records in EBCDIC type V format are to be processed. The system may change this record block format to *VB, *D, or *DB, based on other file parameters. See the previous explanation for more information.

**VB:* Variable length, blocked, unspanned records in EBCDIC type V format are to be processed. The system may change this record block format to *DB, based on the volume code. See the previous explanation for more information.

**D:* Variable length, unblocked, unspanned records in ASCII type D format are to be processed. The system may change this record block format to *DB, *V, or *VB, based on other file parameters. See the previous explanation for more information.

**DB:* Variable length, blocked, unspanned records in EBCDIC type D format are to be processed. The system may change this record block format to *VB, based on the volume code. See the previous explanation for more information.

**VS:* Variable length, unblocked, spanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *VBS, based on other file parameters. See the previous explanation for more information. Note that the representation of spanned records on the tape is different for EBCDIC and ASCII files, but the system selects the correct format based on the file code.

**VBS:* Variable length, blocked, spanned records in either EBCDIC or ASCII code are to be processed. Note that the representation of spanned records on the tape is different for EBCDIC and ASCII files, but the system selects the correct format based on the file code.

**U:* Undefined format records in either EBCDIC or ASCII code are to be processed. RCDBLKFMT(*U) records are processed as variable length records, where each record written or read is in a separate tape block. This format can be useful for processing tape files that do not meet the formatting requirements of any other record block format.

EXTEND Parameter: Specifies, for output operations to tape, whether new records are to be added to the end of a data file that is currently on the tape. (The specific data file is identified by the SEQNBR parameter and, for a standard-label file, the LABEL parameter.) If the data file is extended, it becomes the last file on the tape volume; any data files that follow it are overwritten as the specified file is extended.

<u>*NO:</u> Records are not to be added to the end of the specified data file. Regardless of whether there is already a data file with the specified SEQNBR on the tape, a new data file is created (overwriting an existing data file and any files that follow it).

**YES:* New records are to be added to the end of the specified data file on tape when this device file is used.

DENSITY Parameter: Specifies, in bits per inch, the density of the data that is to be written on the tape volume when this device file is used. This parameter is used only for tapes written as nonlabeled volumes (*NL); it is ignored unless the *first* data file is being written on the nonlabeled volume. The density of a standard-label volume is specified on the INZTAP command, which initializes tapes as standard-label volumes by writing volume labels on them. If a labeled or nonlabeled output file is written with a different density than specified here, a warning message is issued.

<u>1600:</u> The data density on this tape volume is to be 1600 bits per inch.

800: The data density on this tape volume is to be 800 bits per inch.

CODE Parameter: Specifies the type of character code to be used when tape data is read or written by a job that uses this tape device file. If a labeled volume is recorded in a different code than the value specified for the file, a warning message is sent to the program that opened the file and the volume code is assumed for the file. This parameter is used only for tapes written as nonlabeled volumes (*NL or *NS). The code for a standard-label volume is specified on the INZTAP command, which initializes tapes as standard-label volumes by writing volume labels on them.

<u>*EBCDIC:</u> The EBCDIC character code is to be used with this tape device file.

**ASCII:* The ASCII character code is to be used with this tape device file.

CRTDATE Parameter: Specifies, for tape input data files and for tape output for which EXTEND(*YES) is specified, the date when the data file was created (written on tape). The data file creation date is stored in file labels on the tape. If a creation date is specified for any type of label processing other than *SL, it is ignored. If the creation date written on the tape containing the data file does not match the date specified in this device file description, an inquiry message is sent to the operator.

NONE: The creation date is not specified. It will not be checked unless it is supplied before the device file is opened, either in the OVRTAPF (or CHGTAPF) command or in the HLL program.

creation-date: Enter the creation date of the data file to be used by this device file. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP.

EXPDATE Parameter: Specifies, for tape output data files only, the expiration date of the data file used by this device file. The data file expiration date is stored in file labels on the tape. If an expiration date is specified for any type of label processing other than *SL, it is ignored. If a date is specified, the data file is protected and cannot be overwritten until the specified expiration date.

NONE: No expiration date for the data file is specified; the file is not to be protected. An expired date is written in the data file labels so the file can be used as a scratch data file.

**PERM:* The data file is to be protected permanently. The date written in the tape data file labels consists of all nines.

expiration-date: Enter the expiration date on which the data file expires. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP.

ENDOPT Parameter: Specifies the positioning operation to be performed automatically on the tape volume when the device file is closed. In the case of a multiple-volume data file, this parameter applies to the *last* reel only; all other reels are rewound and unloaded when the end of the tape is reached.

REWIND: The tape is to be rewound, but not unloaded, when the file is closed.

**UNLOAD:* The tape is to be rewound and unloaded when the file is closed.

**LEAVE:* The tape should be left in its current position when the file is closed; it is not to be rewound or unloaded. This option can be used to reduce the time required to position the tape if the next tape file to open to this device uses a data file is on this volume.

Note: Even if ENDOPT(*LEAVE) is specified, the next tape file opened to this reel will be positioned at the beginning of some data file on the volume (or end of a data file, for either read backward or for output that extends an existing data file on the volume). A tape file is always positioned at the start or end of a data file when it is opened.

WAITFILE Parameter: Specifies the number of seconds the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. (For an expanded description of the WAITFILE parameter, see Appendix A.)

*IMMED: The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds the program is to wait for the file resources to be allocated to the tape device file. Valid values are 1 through 32767 (32 767 seconds).

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record.

*NO: An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens the file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file.

PUBAUT Parameter: Specifies what authority for the tape device file and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the Grant Object Authority (GRTOBJAUT) command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

**NORMAL:* The public has only operational rights for the device file.

**ALL:* The public has complete authority for the device file.

**NONE:* The public cannot use the device file.

TEXT Parameter: Specifies the user-defined text that briefly describes the tape device file. (For an expanded description of the TEXT parameter, see Appendix A).

<u>*BLANK:</u> No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTTAPF  FILE(BACKHST) DEV(QTAPE1 QTAPE2 QTAPE3) REELS(*BLP 10) +
           RCDLEN(256) BLKLEN(1024) RCDBLKFMT(*FB) EXTEND(*YES) +
           ENDOPT(*UNLOAD) WAITFILE(60)
```

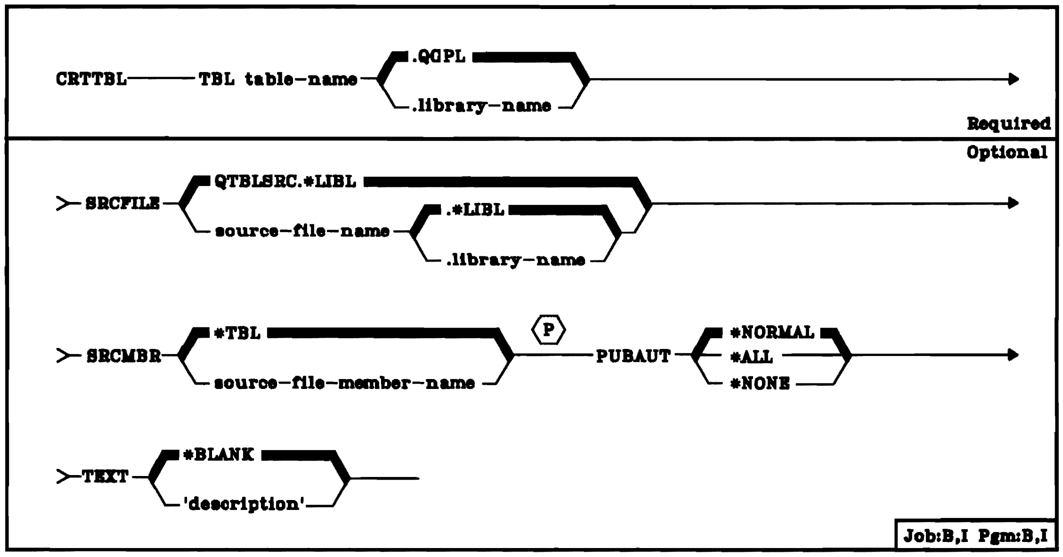
This command creates a description of the tape device file named BACKHST in library QGPL, to be used with the tape devices QTAPE1, QTAPE2, and QTAPE3. All volumes processed on these devices with this device file must have standard labels. Each block of data (EBCDIC character code) on the tape volumes contains four records of 256 bytes each. When records are written to the tape, they are added to the end of the data file. No creation or expiration date is specified for this tape, and both unloading and rewinding operations will occur when the device file is closed at the last tape volume processed. The program using this tape device file will wait 60 seconds for file resources to be allocated when this file is opened, and this device file is dedicated to the current program invocation.

CRTTBL (Create Table) Command

The Create Table (CRTTBL) command creates a named table. The table can be used for the translation of data that is transferred between the system and a device (a printer, for example). The table can also be used to specify an alternate collating sequence or for field translation functions.

The table is stored internally as a 256-byte character string. The input must consist of 512 hexadecimal characters because 2 hexadecimal characters of input equal 1 internal byte. Each record of input must contain 64 characters, and eight records (512 characters) must be entered. Record sizes greater than 64 characters are valid, but only the first 64 characters are used. For more information about creating tables, see the *CPF Programmer's Guide.*

Restriction: If a table is to be used by a system printer, the name of the table must be specified in either the TRNTBL or PRTIMG parameter of the printer device file that is opened by the printer. However, if a specified table is deleted and recreated during the time that it is being used by the system printer, the printer must be varied offline, then online, before the new table can be used.



TBL Parameter: Specifies the qualified name of the table being created. (If no library qualifier is given, the table is stored in the general purpose library, QGPL.)

SRCFILE Parameter: Specifies the name of the source file containing the description of the table being created. Information about the format of records in the source file is contained in the *CPF Programmer's Guide*.

QTBLSRC: The system source file named QTBLSRC contains the source records to be used with this command to create the table. (If no library qualifier is specified, *LIBL is used to find the file.)

qualified-source-file-name: Enter the qualified name of the source file that contains the source records to be used with this command to create the table. (If no library qualifier is given, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of the source file member containing the description of the table being created.

*TBL: The source file member name is the same as the name of the table.

source-file-member-name: Enter the name of the member in the source file specified by SRCFILE to be used to create the table.

PUBAUT Parameter: Specifies what authority for the table and its description is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the Grant Object Authority (GRTOBJAUT) command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NORMAL: The public has only operational rights for the table.

**ALL:* The public has complete authority for the table.

**NONE:* The public cannot use the table.

TEXT Parameter: Lets the user enter text that briefly describes the table. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Example

```
CRTTBL  TBL(SCRAMTBL) SRCFILE(USERTABLES) +
      SRCMBR(SCRAMBLE) TEXT('Translate table for +
      scrambling text characters')
```

This command creates a table named SCRAMTBL and stores it in the QGPL library (default). The source file named USERTABLES contains the source records used when the table is created; the name of the source file member is SCRAMBLE. The TEXT parameter describes this table as being used as a translate table for scrambling text characters. Information about the format of source file records is contained in the *CPF Programmer's Guide*.

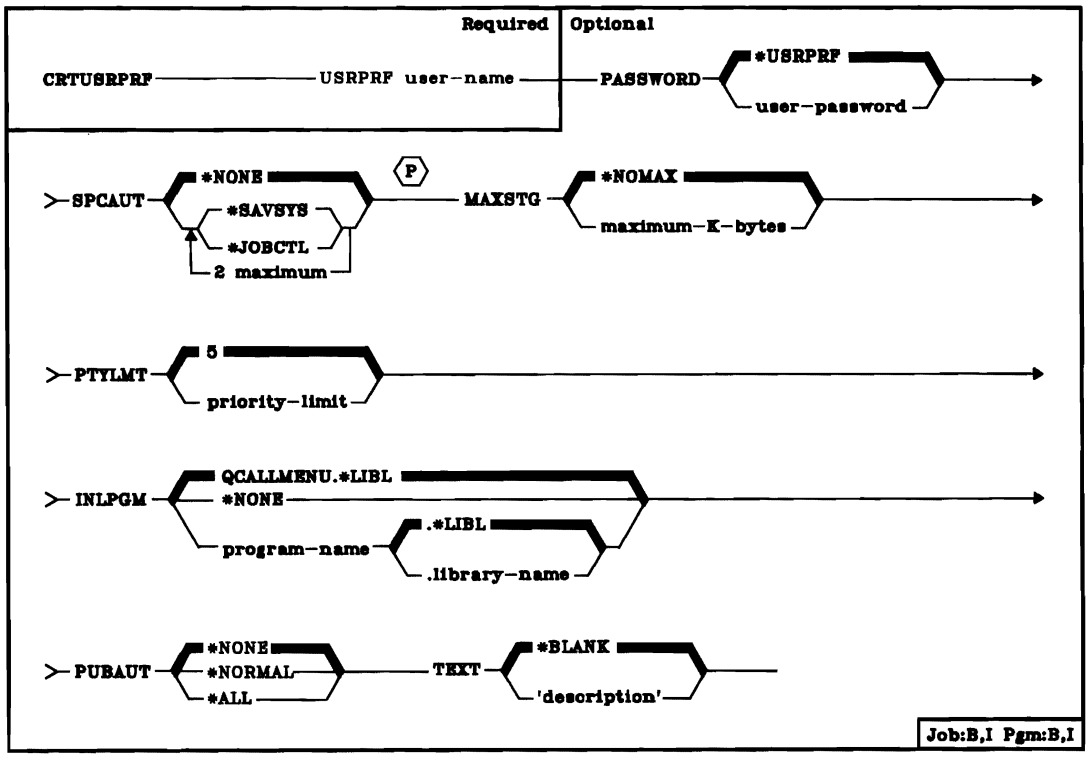
CRTUSRPRF (Create User Profile) Command

The Create User Profile (CRTUSRPRF) command identifies a user to the system and creates a user profile containing only the attributes assigned to that user. These attributes are used by the system to control any jobs submitted by the user or any jobs that are executed under the constraints of this user profile but are submitted by other system users. When the user profile is created, it is stored as part of the internal system, and it appears as though it exists in the QSYS (system) library.

When the user identified by this user profile uses the system, the user profile is checked by the system to determine what objects the user is authorized to use. When the user profile is created, the user is granted read, add, delete, and object management rights for the profile itself, and the user can use only those objects, commands and devices that have public authority. Subsequently, the security officer and other object owners can explicitly grant rights of use for other objects to the user through the use of the Grant Object Authority (GRTOBJAUT) command. The profile identifies:

- Objects owned by the user.
- Commands and system devices that are authorized for use by the user.
- Objects (including commands and devices) that have been explicitly authorized for use by the user. The names of the objects and the rights granted are stored in the profile.
- The number of objects owned by the user and/or authorized for his use.
- Special rights granted to the user.
- The maximum amount of storage allowed, and the amount currently in use, for the storage of owned objects.
- The maximum scheduling priority for jobs and spooled output, the initial program that is invoked after sign-on, and the text that describes the profile.

Restriction: Only the system security officer can use this command.



USRPRF Parameter: Specifies the name of the user profile by which the user is to be known in the system. A maximum of 10 alphameric characters can be used; the first character must be alphabetic.

PASSWORD Parameter: Specifies the unique password that indicates the user profile used by the system to control the user's jobs. The password is the security key that allows the user to sign on to the system. The user signs on the system by entering the password *exactly* as it is specified here. The password is to be used only by the user(s) it identifies—it should be known only to the user himself and to the system security officer, who assigns it.

*USRPRF: If a password is not entered, it is to be the same as the user profile name specified in the USRPRF parameter.

user-password: Enter the alphameric character string (10 characters or less) that identifies the user with his own user profile. The standard rule for specifying names also applies to passwords. The first character must be alphabetic and the other characters must be alphameric.

SPCAUT Parameter: Specifies the special rights that are to be granted to a user. Special rights are *required* to perform certain functions on the system. The special rights are save system rights (*SAVSYS) and job control rights (*JOBCTL). The special rights *SAVSYS and *JOBCTL are normally given to the user who operates the system. The security officer can, however, grant any of these rights to any user profile.

*NONE: No special rights are to be granted to this user profile.

**SAVSYS:* The save system rights are to be granted to this user profile. This user is given the authority to save, restore, and free storage for all objects on the system, regardless of whether he has object existence rights for the objects.

**JOBCTL:* The job control rights are to be granted to this user profile. This user is given the authority to change, display, hold, release, and cancel all jobs that are executing on the system or that are on a job queue or output queue that has OPRCTL(*YES) specified.

MAXSTG Parameter: Specifies the maximum amount of auxiliary storage that can be allocated to store permanent objects that are owned by this user profile including objects placed in QTEMP during a job. If the maximum is exceeded when the user creates an object, an error message is displayed and the object is not created.

*NOMAX: As much storage as required can be allocated to this profile.

maximum-K-bytes: Enter the maximum amount of storage in K-bytes that can be allocated to this profile. (1 K equals 1024 bytes of storage.)

PTYLMT Parameter: Specifies the highest scheduling priority that the user is allowed to have for each job that he submits to the system. This value controls the job processing priority and output priority that *any* job running under this user profile can have; that is, values specified in the JOBPTY and OUTPTY parameters of any job command cannot exceed the PTYLMT value of the user profile under which the job is to be run. The scheduling priority can have a value of 1 through 9, where 1 is the highest priority and 9 is the lowest. (For an expanded description of the PTYLMT parameter, see *Scheduling Priority Parameters* in Appendix A.)

5: The user named in this profile can have a priority value no higher than 5 for scheduling any of his jobs on the system. All jobs having this priority value will be executed before all jobs having values of 6 through 9, and after all jobs having values of 1 through 4.

priority-limit: Enter a value, 1 through 9, for the highest scheduling priority that the user is allowed.

INLPGM Parameter: Specifies, for an interactive job, the name of the program that is to be invoked whenever a new routing step that has QCL as the request processing program is initiated. (No parameters can be passed to the initial program.) The named program can cause a menu to be displayed or perform some other function. If this program ends or returns (via the RETURN command), the command entry display is presented at the work station. If the program sends the escape message CPF2320, the command entry display is not shown and QCL ends normally. If this program terminates abnormally, QCL terminates and an abnormal termination message is sent to the work station.

QCALLMENU: The program named QCALLMENU is invoked automatically when the user signs on. This program causes the program call menu to be displayed. (If no library qualifier is specified, *LIBL is used to find the program.) This menu is described in the *System/38 Programmer's/User's Work Station Guide.*

**NONE:* No initial program is to be invoked when the user signs on. The command entry display is shown instead.

qualified-program-name: Enter the qualified name of the program that is to be invoked after the user signs on. (If no library qualifier is given, *LIBL is used to find the program.) The following IBM-supplied programs can be invoked, if installed: QCALLMENU (program call menu), QOPRMENU (system operator menu), and QPGMMENU (programmer menu).

PUBAUT Parameter: Specifies what authority for the user profile is being granted to the public (all users). Additional authority can be explicitly granted to specific users by the Grant Object Authority (GRTOBJAUT) command. (For an expanded description of the PUBAUT parameter, see Appendix A.)

*NONE: The public cannot use the user profile.

**NORMAL:* The public has normal authority for the user profile, which is the same as operational authority.

**ALL:* The public has complete authority for the user profile.

TEXT Parameter: Lets the user enter text that briefly describes the user profile being created. (For an expanded description of the TEXT parameter, see Appendix A.)

*BLANK: No text is to be specified.

'description': Enter no more than 50 characters, enclosed in apostrophes.

Examples

```
CRTUSRPRF  USRPRF(JJADAMS)  PASSWORD(SECRET) +
      SPCAUT(*SAVSYS)  INLPGM(DSPMENU.ARLIB)
```

This command creates a user profile with the user name of JJADAMS and a password of SECRET. After sign-on, a program called DSPMENU in the ARLIB library is invoked. The user is granted the special save system rights. Because the other parameters were not coded: (1) the profile has no limit on the amount of storage allocated to it for owned permanent objects; (2) a scheduling priority of 5 is the highest priority that any of the user's jobs can have; and (3) the user-defined description text is blank. No public authority is granted.

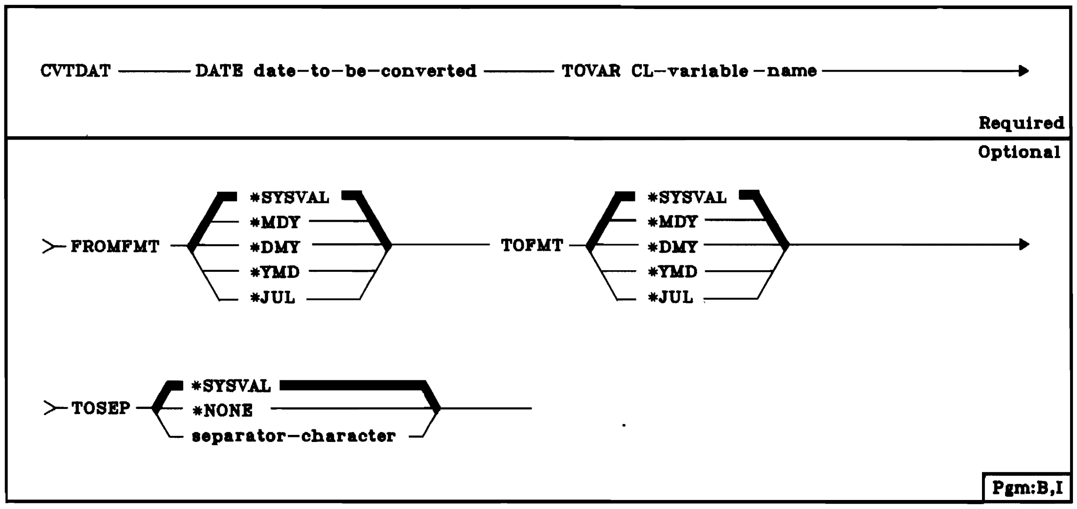
```
CRTUSRPRF  USRPRF(TMSMITH)  MAXSTG(10) +
      INLPGM(CALC.PROGMR) +
      TEXT('Ted M. Smith, Dept 41, +
      Application Programs')
```

This command creates a user profile with the user name of TMSMITH; the password is also TMSMITH because the password was not specified. The maximum amount of permanent storage that the user can use for all his objects is 10 K-bytes (or 10 240 bytes). The initial program to be invoked following sign-on is CALC, which is located in the library named PROGMR. The text parameter provides the user's name and department. The other parameters have their default values assigned.

CVTDAT (Convert Date) Command

The Convert Date (CVTDAT) command converts the format of a date value from one format to another, without changing its value. The command ignores any date separators used in the old format, but if separators are to be included in the converted result, a separator character can be specified on the command.

Restriction: This command is valid only within a CL program.



DATE Parameter: Specifies the constant or CL variable containing the date that is to be converted. When a constant is specified that contains separator characters, it must be enclosed in apostrophes (the separator characters are ignored in the conversion). If separators are used in a constant, leading zeros in each part of the date can be omitted (3/3/80 or 03/03/80 are both valid). If a variable is specified, it must be long enough to contain the date type and its date separators, if used. The valid date separators are the slash (/), hyphen (-), period (.), and comma (,).

TOVAR Parameter: Specifies the name of the CL variable that is to contain the converted date value. The variable must be declared with a minimum length of 8 characters if the converted date is to contain date separators (6 are required for the Julian format); the length must be at least 6 characters (5 for Julian) if the converted date will not contain date separators.

For every format except Julian, the month, day, and year subfields in the converted result are each 2 bytes in length, are right-justified, and (if necessary) a leading zero is used as a padding character to fill each 2-byte field. For the Julian format, the day field is 3 bytes long and padded with leading zeros (if necessary), and the year field is 2 bytes long.

FROMFMT Parameter: Specifies the current format of the date being converted.

<u>*SYSVAL:</u> The date has the format specified by the system value QDATFMT.

**MDY:* The date has the month, day, year format.

**DMY:* The date has the day, month, year format.

**YMD:* The date has the year, month, day format.

**JUL:* The date has the Julian format.

TOFMT Parameter: Specifies the format to which the date is being converted.

<u>*SYSVAL:</u> The date format is converted to the format specified by the system value QDATFMT.

**MDY:* The date format is converted to month, day, year.

**DMY:* The date format is converted to day, month, year.

**YMD:* The date format is converted to year, month, day.

**JUL:* The date format is converted to Julian.

TOSEP Parameter: Specifies the type of date separators, if any, to be used in the converted date.

<u>*SYSVAL:</u> The converted date is to have the separators specified by the system value QDATSEP.

**NONE:* No separator characters are to be contained in the converted date.

separator-character: Enter the character that is to be used as the date separator in the converted date. The valid separator characters are the slash (/), hyphen (-), period (.), and comma (,).

Examples

```
DCL  VAR(&DATE) TYPE(*CHAR) LEN(8)
   •
   •
   •
CVTDAT  DATE('12-24-80') TOVAR(&DATE) TOFMT(*DMY)
```

This command converts the date 12-24-80, which is in the MDY format; because the FROMFMT parameter was not specified, its default *SYSVAL indicates that the system value QDATFMT contains MDY. The date is converted to the DMY format, and the separator character specified in the system value QDATSEP is inserted. If QDATSEP contains a slash, the converted result is 24/12/80.

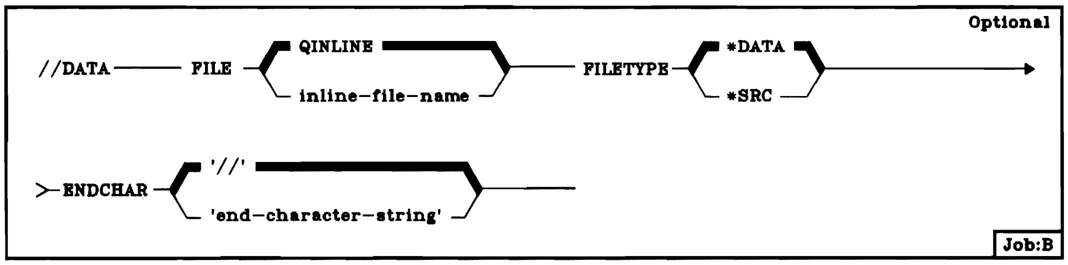
```
DCL  &PAYDAY *CHAR 6
DCL  &NEWPDAY *CHAR 6
   •
   •
   •
CVTDAT  DATE(&PAYDAY) TOVAR(&NEWPDAY) +
      FROMFMT(*YMD) TOSEP(*NONE)
```

This command converts the format of the date stored in &PAYDAY from year, month, day to the format specified by the system value QDATFMT. If, for example, QDATFMT contains MDY, the format of the converted date is month, day, and year. The converted date is stored in the variable &NEWPDAY. Because &NEWPDAY was declared as a 6-character variable, TOSEP(*NONE) is required; the converted result cannot include separator characters.

DATA (Data) Command

The Data (//DATA) command must be used to indicate the beginning of an inline data file in an input stream that is to be read by a spooling reader. It also specifies what delimiter must be used to indicate the end of the data file. Inline data files exist only for the duration of the job; after the job is finished, they are destroyed. Unnamed inline files can be used only once in the job.

Restrictions: The DATA command cannot be executed from a work station. The DATA command must be preceded by two slashes (//) in positions 1 and 2 in the data record. Blanks can separate the slashes from the command name (// DATA).



FILE Parameter: Specifies the name of the inline data file. This name is also specified in the program that is to process the file.

<u>QINLINE</u>: The name of the inline data file is to be QINLINE. The file is processed as an unnamed inline file. An unnamed file can be processed if the program specifies QINLINE as the file name, or if the device file that specifies SPOOL(*YES) is opened for input. Unnamed inline files can be used only once by the job.

inline-file-name: Enter the name of the inline data file to be used by one or more programs in the job. The file is connected to the program when the program opens the file by specifying its file name. Named inline data files can be accessed more than once by the job.

FILETYPE Parameter: Specifies whether the inline data following this command is to be put in the standard format for source files or in the data file format. The standard source file format is a sequence number (a 6-character source number) followed by the 6-character system date that precedes the data. (For an expanded description of the FILETYPE parameter, see Appendix A.)

<u>*DATA</u>: The inline data is not in the standard format for source files. The data file is passed to the program using it in the same form as it was read in.

**SRC:* The inline data is to be sequence numbered; it is to be a source file that can be used to create another file or a program.

ENDCHAR Parameter: Specifies a string of characters used to indicate the end of an inline data file. To be recognized, the character string must begin in position 1 of the record. If you specify a character string other than // (the default value) as the delimiter, all records up to the end-of-file record (the record containing the specified character string starting in column 1) are treated as data. This allows you to imbed reader commands (//JOB, //DATA, or //ENDJOB) in the data stream. It also allows the characters /* to be included as data without causing the job input to be terminated on the MFCU. The end-of-file record for non-default ENDCHAR values is not put to the data file, nor is it checked to see if it is a valid reader command. It is used only to determine the end of the data stream and then it is discarded.

'//': The default value is two slashes. The command will work the same way whether two slashes are coded into the parameter or the parameter itself is defaulted.

Using the default, the slashes in positions 1 and 2 of a record (in either a data file or a source file) identify the first record beyond the file. Thus, the commands //JOB, //DATA, and //ENDJOB also indicate the end of the inline file. The end of the inline file can also be caused by the '/*' card on the MFCU. The characters '/*' are recognized by the MFCU and cause a hardware EOF (end-of-file) condition. The hardware EOF will be recognized as the end of the spooled job.

'end-character-string': A character string (up to 25 characters long and enclosed in apostrophes) can be entered to identify the last record in the file. The character string may contain both alphameric and special characters. If a character combination other than '//' is specified on the ENDCHAR parm, reader commands and '/*' records may be safely imbedded in the data. The reader ignores all other data while searching for the specified string, including reader commands. Should the '/*' record be read by the MFCU, the reader does not recognize it as the hardware EOF, since the '/*' is part of the imbedded data stream.

Examples

```
//DATA  FILE(FILE1)
```

This command assigns the name FILE1 to the data that follows it, until an end of inline data condition is found (two slashes in positions 1 and 2).

```
// DATA  FILE(FILE2) ENDCHAR('STOPIT')
```

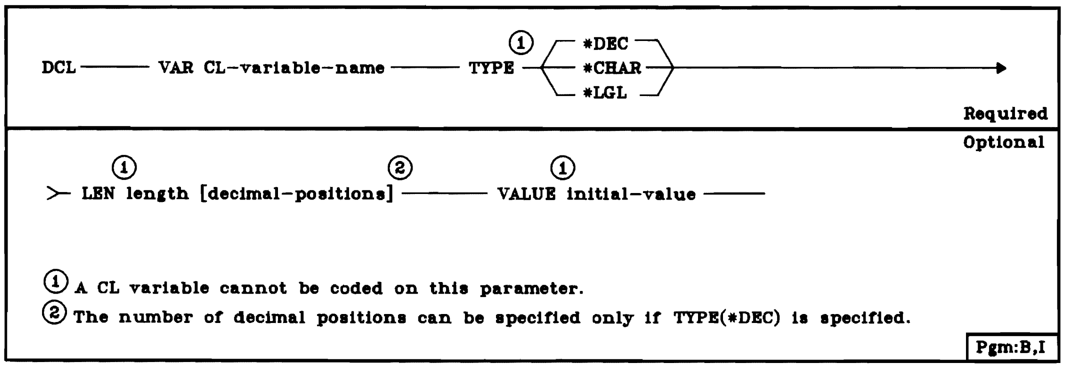
This command assigns the name FILE2 to the data following it; the file continues until a record is found that contains the characters STOPIT in positions 1 through 6. This delimiter allows the //JOB, //ENDJOB, and //DATA commands and records with /* in positions 1 and 2 to be embedded in an inline file.

DCL (Declare CL Variable) Command

The Declare CL Variable (DCL) command defines CL program variables used in CL programs. CL variables are used to store and update data, and to receive parameters from another program on a call. CL variables are known by name only within the program that declares them; they cannot be used outside a CL program, except when they are referenced by some commands (such as the Display Program Variable (DSPPGMVAR) command) used for debugging programs. (If a variable is declared but not referenced by another command in a CL program, the variable is not included in the program when it is compiled.) However, the value in the variable can be passed to another program as a parameter. Each DCL command defines the attributes of one CL variable and declares its name in the program in which it is to be used.

Each CL variable in a program must be identified by one of the three declare commands. The DCLF and DCLDTAARA commands declare CL variables for display device files and data areas. The DCL command declares all other CL variables.

Restriction: The DCL command is valid only within a CL program. All declares (DCL, DCLF, and DCLDTAARA) must follow the PGM (Program) command and precede all other commands in the program. The three types of declares can be intermixed in any order.



① A CL variable cannot be coded on this parameter.

② The number of decimal positions can be specified only if TYPE(*DEC) is specified.

VAR Parameter: Specifies the name of the CL variable being declared within the CL program. The variable exists only within the program in which it is defined. It can be passed as a parameter on a call to another program, in which case it can be processed by the called program. Enter the name of the variable here; it must begin with an ampersand (&).

TYPE Parameter: Specifies the type of value to be contained in the CL variable being declared. The value of the variable can be a character constant, a decimal constant, or a logical one or zero. (The value for this TYPE parameter cannot be specified via a CL variable.) Enter one of the following types:

**DEC:* This is a decimal variable that contains a packed decimal value.

**CHAR:* This is a character variable that contains a character string value.

**LGL:* This is a logical variable that contains a logical value of either '1' or '0'.

LEN Parameter: Specifies the length of the CL variable being declared. If the variable is a decimal value, the number of decimal digits to the right of the decimal point can be optionally specified. The type of CL variable (specified by the TYPE parameter) determines the maximum length that the variable can have and the default length assumed if LEN is not specified. (The value for this LEN parameter cannot be specified via a CL variable.) The maximum lengths and the defaults for each of the three types are:

Type	**Maximum Length**	**Default Length**[1]
Decimal	15 digits, 9 decimal positions	15 digits, 5 decimal positions
Character	2000 characters	32 characters
Logical	1 character	1 character

[1]For decimal and character types, the default length is the same as the length of the initial value, if one is specified in the VALUE parameter.

length: Enter the length that the value in this CL variable can have; the length cannot exceed the maximum for this type of variable.

length [decimal-positions]: This option is valid only for *decimal* variables. The length of the value in the variable includes the number of decimal positions in the value. The maximum length of the decimal value is 15 digits, including the digits to the right of the decimal point. A maximum of nine decimal positions can be specified. (If nine decimal positions are specified, the value to the *left* of the decimal point can never be greater than 999 999 because only six of the 15 digits are left for the integer value.)

If a length is specified for a decimal variable and the number of decimal positions is not, 0 decimal positions is assumed.

VALUE Parameter: Specifies the initial value that is assigned to the CL variable when it is declared in the program. The initial value must be of the type specified by the TYPE parameter. If no initial value is specified, a character variable is initialized to blanks, a decimal variable is initialized to a value of zero, and a logical variable is initialized to '0'. (The value for the VALUE parameter can not be specified via a CL variable.)

If the name of the declared variable is specified on the PARM parameter of the PGM command in the same program in which the variable is declared, an initial value *cannot* be specified for the variable. That is, the variable is to receive its value from the calling program instead.

Examples

```
DCL  &ABLE *DEC  LEN(5 2)
```

This command declares a CL variable named &ABLE that contains a decimal value. The value can never exceed 999.99 because LEN specifies a maximum of five digits, of which two are to the right of the decimal point. Because VALUE was not specified and it is a numeric value, &ABLE is initialized to a value of zero (000.00).

```
DCL  &SWITCH  *LGL
```

This command declares a CL variable named &SWITCH to contain a logical value. Because the type parameter specifies logical, the variable is one character and is initialized to '0'.

```
DCL  &FILNAM  *CHAR  VALUE(FILEA)
```

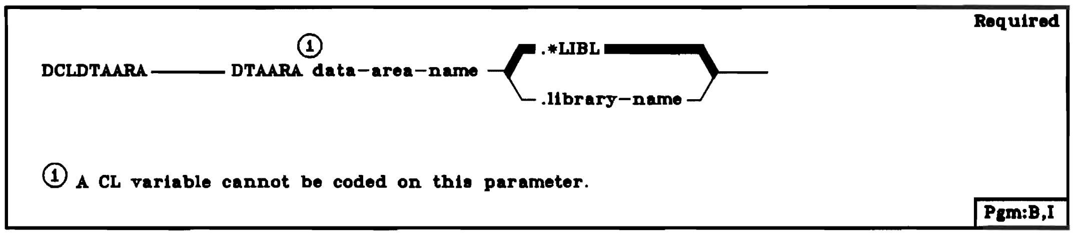
This command declares a CL variable named &FILNAM whose initial value is FILEA. Because the initial value contains 5 characters and the LEN parameter was not specified, the length of the variable is also 5 characters.

DCLDTAARA (Declare Data Area) Command

The Declare Data Area (DCLDTAARA) command declares the name of a previously created data area that is to be used within a CL program. Each data area referenced on the Send Data Area (SNDDTAARA) and the Receive Data Area (RCVDTAARA) commands in a CL program must be declared in the program by a DCLDTAARA command. (The data area was created by the CRTDTAARA command.)

When the CL program that has a DCLDTAARA command in it is compiled, a CL variable is automatically declared in the program with the same name and data attributes that the referenced data area has. Then, when the program is executed, program data can be passed between the data area and the corresponding CL variable by the SNDDTAARA and RCVDTAARA commands. (However, if a data area is declared but the associated CL variable is not referenced by another command in the program, no variable is included in the program for that data area when the program is compiled.)

Restrictions: This command is valid only within a CL program. All declares (DCL, DCLF, and DCLDTAARA) must follow the PGM (Program) command and precede any other commands. The three types of declares can be intermixed in any order.



DTAARA Parameter: Specifies the qualified name of the data area that is to be made known to the CL program. (If no library qualifier is given, *LIBL is used to find the data area.) Two data areas with the same name cannot be declared in the same program even if they are located in different libraries. A CL variable cannot be used to identify the data area.

Example

```
DCLDTAARA  DTAARA(CHECKNUM.MYLIB)
```

This command indicates that a data area named CHECKNUM that is stored in MYLIB is to be used in the program containing this command. When the program is compiled, a CL variable named &CHECKNUM is automatically declared in the program with the same data attributes as the data area. Program data can later be passed between the variable and the data area by the SNDDTAARA and RCVDTAARA commands.

DCLF (Declare File) Command

The Declare File (DCLF) command declares one display device file (by name) to a CL program. Only one DCLF command is allowed in a CL program; the command specifies the name of the device file and the record formats to be used in the program. The program can then contain the data manipulation commands (SNDF, RCVF, SNDRCVF, CNLRCV, and WAIT) that reference the file, enabling the program to interact with its user by sending data to and receiving data from a work station.

When the CL program is compiled, a CL variable is automatically declared for each field in each record format that is used in the program. The field name will become the variable name with an ampersand (&) added at the beginning of the name. (The attributes of each declared field are the same as the attributes of the fields in the device file. Fields defined in the record format as numeric are defined as decimal variables.) Also, indicators used in the referenced device file are declared as logical variables in the form &INnn, where nn is the indicator number.

The variables that are automatically declared by the DCLF command can be used in the program in the same manner as the variables declared by a DCL command. For example, indicators can be used in expressions and IF statements because they are declared as logical variables.

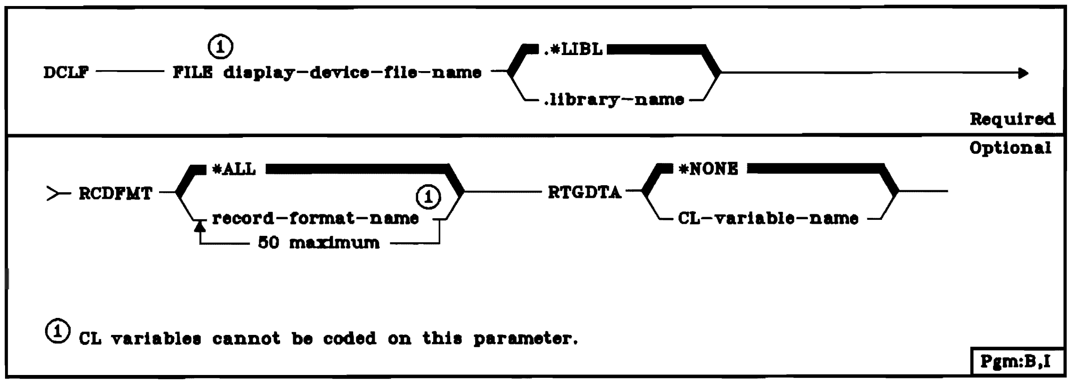
(The contents of the variables, not the variable names, are seen by the user; the display shows one, some, or all of the fields in the record format that the user can fill in. DDS determines the display format.)

The DCLF command also allows certain routing information to be returned to the variable named in the RTGDTA parameter with the user's data. The routing data to be returned is defined in the device file.

Restrictions: This command is valid only within CL programs. All declares (DCL, DCLF, and DCLDTAARA) must follow the PGM (Program) command and precede any other command. The three types of declares can be intermixed in any order. The device file must be a display device file, and it must exist before the program is created.

Because CL variables are automatically declared for each field in a referenced file's record formats, the following restrictions apply:

- If the display file is changed (and the file description specifies that level checking is to be performed), the CL program must be recompiled to match the new file description. More information on level checking is contained in the *CPF Programmer's Guide.*
- If any field name is defined in more than one record format of the display file, then the attributes in each record format for the commonly named field must match.
- Any CL variable declared in the program by a DCL command and having the same name as an automatically declared CL variable (for a referenced field) must also have the same attributes specified in DDS for the referenced field.
- The variables used in the file must have data types supported for CL variables. (However, fields defined as numeric are declared as decimal variables.)



FILE Parameter: Specifies the qualified name of the display device file to be used by the CL program. (If no library qualifier is given, *LIBL is used to find the file.) A CL variable name cannot be used to specify the file name.

RCDFMT Parameter: Specifies the names of one or more record formats contained in the display device file that are to be used by the SNDF, RCVF, and SNDRCVF commands in the CL program. CL variable names cannot be specified in RCDFMT; only names of record formats can be used. For every field and indicator in each record format specified in RCDFMT, one CL variable is automatically declared in the program.

*ALL: Every record format in the device file, up to a maximum of 99, is to have its fields declared in the CL program as variables. If there are more than 99 record formats in the file, only the first 99 are used.

record-format-name: Enter one or more record format names whose fields are to be declared as variables in the CL program. No more than 50 record format names can be specified; CL variables cannot be used to specify the names.

RTGDTA Parameter: Specifies whether the data that is defined as routing data in the device file (by one or more of the DDS routing keywords) is to be passed to the program along with the user's data. (For example, the device name can be defined as routing data and passed to the program to identify which work station is sending the other data.) If the routing data is to be passed, the parameter also specifies the name of the CL variable into which the data is to be placed. The specified variable must be a character variable that is at least 80 characters long. (For more information on routing data in the device file, see the *CPF Reference Manual—DDS*.)

*NONE: No routing data is to be returned to the CL program when send or receive file commands are executed.

CL-variable-name: Enter the name of the CL character variable into which the routing data defined for the device file is to be placed when RCVF or SNDRCVF commands are executed. The named variable must be declared in the CL program as a character variable and must be at least 80 characters long.

Examples

```
DCLF  FILE(ABLE)
```

This command specifies that the display device file named ABLE is to be used by the CL program to pass data between the user and the program. RCDFMT was not specified; therefore, all the fields and indicators in all the record formats are automatically declared as variables, and data from any field in any record format (up through the first 99) in the device file can be passed between the program and the user. No device file routing data is to be received by the program.

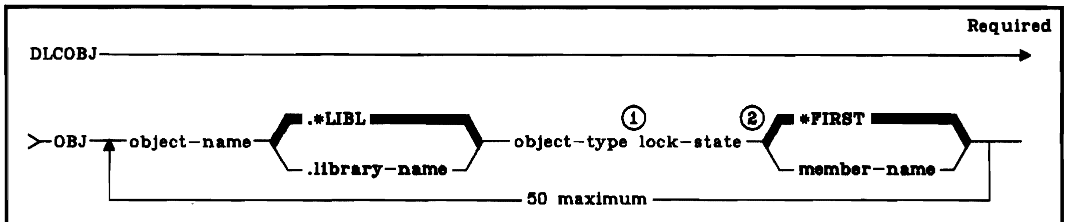
```
DCLF  FILE(BAKER) RCDFMT(REC2 REC6) RTGDTA(&FB)
```

The display device file BAKER is to be used by the CL program to pass data between the user and the program. Both the REC2 and REC6 record formats are to be used. Device file routing data is to be returned to the program and stored in the CL variable &FB for each RCVF and SNDRCVF command.

DLCOBJ (Deallocate Object) Command

The Deallocate Object (DLCOBJ) command releases the allocations of the specified objects for the specified lock states. The objects, allocated earlier in the same routing step by one or more Allocate Object (ALCOBJ) commands, are freed for use by other jobs. If the DLCOBJ command is not used, the objects are automatically deallocated at the end of the routing step.

A single DLCOBJ command can, for each allocated object, release only one lock state. That is, if one object has multiple lock states applied, a separate DLCOBJ command must be used to release each one.



① The values of only six object types are valid: *DEVD, *DTAARA, *FILE, *LIB, *MSGQ, and *SBSD. Refer to the OBJ parameter description for the lock states that are valid for each object type.
② If valid for the specified object type, one of the following lock states can be specified: *SHRRD, *SHRUPD, *SHRNUP, *EXCLRD, or *EXCL.

Job:B,I Pgm:B,I

OBJ Parameter: Specifies the qualified names of one or more CPF objects that are to be deallocated from the job, the type of each object deallocated, the lock state of each object, and if the object is a data base file, a member name can optionally be specified. (If no library qualifier is given for an object, *LIBL is used to find the object. Note that the LIB and DEVD object types do not reside in user libraries and, therefore, cannot be qualified with a library name.) If the member name is not specified for a data base file, the member name defaults to *FIRST and the first member of the file is deallocated.

For each object named, enter the object name (optionally qualified) followed by the object type, one lock state value, and (if applicable) the file member name to be deallocated. The possible lock states are:

Value	Lock State Meaning
*SHRRD	Shared for read
*SHRUPD	Shared for update
*SHRNUP	Shared, no update
*EXCLRD	Exclusive, allow read
*EXCL	Exclusive, no read

For an explanation of each lock state, refer to the *CPF Programmer's Guide*.

Only six of the CPF object types can be specified on the DLCOBJ command. Of these six, some cannot use all of the lock states. Refer to the lock state table under the OBJ parameter of the *ALCOBJ Command*.

Example

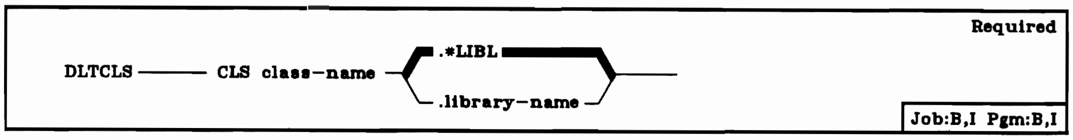
```
DLCOBJ  OBJ((FILEA.LIBB  *FILE  *SHRRD))
```

This command releases the shared-for-read allocation of the first member of FILEA in the library LIBB.

DLTCLS (Delete Class) Command

The Delete Class (DLTCLS) command deletes a class object from the system. Any executing routing steps using the class are not affected by its deletion. However, additional routing steps that use the class cannot be initiated. If the deleted class is referred to in any existing routing entry, either the routing entry should be changed (to refer to a different class) or another class should be created with the same name. If the subsystem is active when the class is deleted, errors might occur in the subsystem.

Restriction: You must have object existence rights for the class being deleted.



CLS Parameter: Specifies the qualified name of the class description to be deleted. (If no library qualifier is given, *LIBL is used to find the class description. If a library qualifier is specified, no completion message will be sent.)

Example

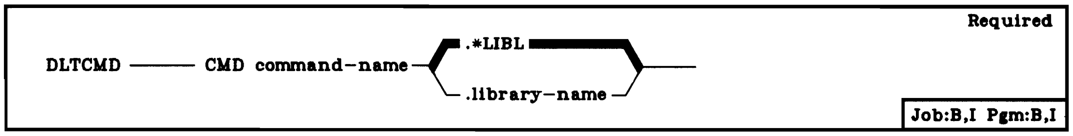
```
DLTCLS  CLS(CLASS1)
```

This command deletes the class named CLASS1 from the system.

DLTCMD (Delete Command) Command

The Delete Command (DLTCMD) command removes a user-specified command from the library in which it is located. Only the command definition object is removed; the command definition source, the command processing program, and the validity checker are not affected.

Restriction: To use this command, you must have object existence rights for the command to be deleted.



CMD Parameter: Specifies the qualified name of the command to be deleted. (If no library qualifier is given, *LIBL is used to find the command.) The specified command whose command definition object is to be deleted can be a user-defined or IBM-supplied command. The usage rights for the command are deleted from the user profiles of all users authorized to use the command.

Example

```
DLTCMD CMD(PAYROLL.LIB01)
```

The command named PAYROLL is to be deleted from library LIB01. Any rights of use for the command are removed from the profiles of all authorized users.

DLTCUD (Delete Control Unit Description) Command

The Delete Control Unit Description (DLTCUD) command deletes the specified control unit description.

If any devices are attached to the control unit, they are detached when the command is executed; each device cannot be used until it is associated with another control unit. If a new control unit description is created, the devices can be specified in the DEV parameter of the Create Control Unit Description (CRTCUD) command. Or, the devices can be attached to other control units that already have control unit descriptions. In this case, the device descriptions can be deleted and recreated, giving the new control unit names in the CTLU parameter of each Create Device Description (CRTDEVD) command.

Restriction: The control unit identified in the control unit description, and any devices or lines attached to the control unit, must be varied offline before this command is entered.

```
                                                                      Required
  DLTCUD ——— CUD control-unit-description-name ———
                                                              Job:B,I Pgm:B,I
```

CUD Parameter: Specifies the name of the control unit description to be deleted.

Example

```
DLTCUD  CUD(CONTROL01)
```

This command deletes the control unit description named CONTROL01 from the system. If the control unit description to be deleted has any device descriptions associated with it, they are detached and a message containing their names is sent to the system operator.

DLTDEVD (Delete Device Description) Command

The Delete Device Description (DLTDEVD) command deletes the specified device description.

Restrictions: The device identified in the device description must be varied offline before this command is issued; if the device is attached to a control unit, it too must be varied offline.

```
                                                                    Required
   DLTDEVD ──────── DEVD device-description-name ─────

                                                           Job:B,I Pgm:B,I
```

DEVD Parameter: Specifies the name of the device description to be deleted. The device description for the system console (named QCONSOLE) cannot be deleted.

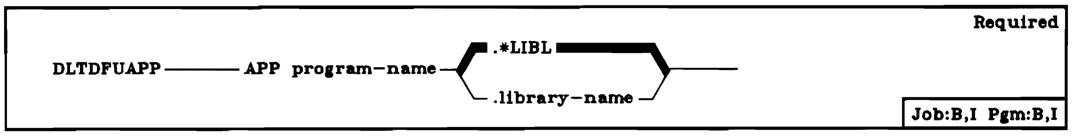
Example

```
DLTDEVD  DEVD(ARCTIC01)
```

This command deletes the device description of the device named ARCTIC01 from the system.

DLTDFUAPP (Delete DFU Application) Command

The Delete DFU Application (DLTDFUAPP) command deletes an existing DFU application and the utility definition statements from a library. The Data File Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Data File Utility, refer to the *IBM System/38 DFU Reference Manual and User's Guide,* SC21-7714.



APP Parameter: Specifies the name of the DFU application you are deleting. (If no library qualifier is specified, *LIBL is used to find the application.)

Example

```
DLTDFUAPP APP(DATA.LIB1)
```

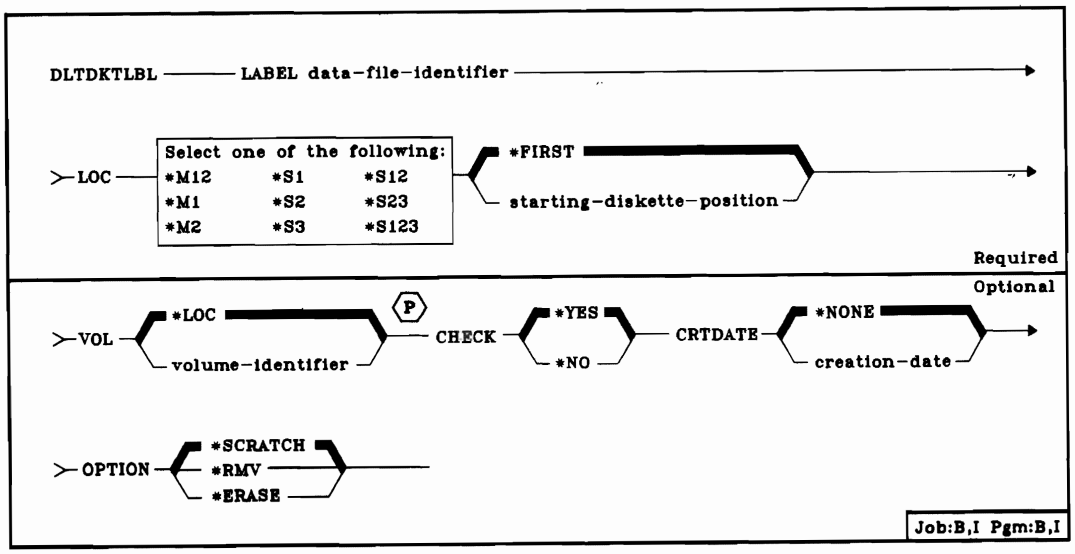
This command deletes the application DATA from the library named LIB1.

DLTDKTLBL (Delete Diskette Label) Command

The Delete Diskette Label (DLTDKTLBL) command deletes the label (that is, the data file identifier) of a named data file from a diskette; the data file must be in the basic exchange format. The data in the file can optionally be overwritten with binary zeros. If the file is active (the file expiration date is greater than the system date), a message is issued to the system operator. The operator can then either continue to delete the file or terminate the command.

Note: When processing diskettes with non-IBM standard labels, you may have unpredictable results. To initialize the diskette, execute the Initialize Diskette (INZDKT) command with CHECK(*NO) specified.

Restrictions: Diskettes that have an extended label area (not supported by System/38) do not have the extended label area accessed in the search for the label of the file to be deleted.



LABEL Parameter: Specifies the data file identifier of the file to be deleted.

LOC Parameter: Specifies the location in the magazine or slot that contains the diskette having the file that is to be deleted. For magazines, two values must be specified: one for the magazine and one for the first diskette used in the magazine. No ending diskette position need be specified because the operation stops when an end-of-file indication is given. (Either *FIRST or a diskette position can be specified for the second value.) Only one value is needed to identify the manual slot used.

Enter one of the following values to specify which magazine or slot is to be used: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

<u>*FIRST:</u> The first diskette in the location specified by the first value in the LOC parameter is the diskette at which to start. If *FIRST is used and the first value is:

*M12	Start at diskette 1 of magazine 1.
*M1	Start at diskette 1 of magazine 1.
*M2	Start at diskette 1 of magazine 2.
*S12	Start at slot 1.
*S23	Start at slot 2.
*S123	Start at slot 1.

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the first diskette from which the file is to be deleted. (A value is not valid for manual slots.)

VOL Parameter: Specifies whether a check of the volume identifier field on the diskette should be made before the specified file is deleted. If so, the volume identifier of the volume to be checked must be specified.

<u>*LOC:</u> No volume identifier check is made; if the named file is on the diskette specified by the LOC parameter, it is deleted without being checked. If the CRTDATE parameter is specified, its check must be met.

volume-identifier: Enter a volume identifier that is to be compared with the diskette label volume identifier field on the diskette that contains the file to be deleted. The identifier can have no more than 6 characters; any combination of letters and digits can be used. If the volume identifiers do not match, a message is issued to the system operator. The operator can then either insert the correct diskette and try again or terminate the command.

CHECK Parameter: Specifies whether a check for active files (those with an expiration date greater than the system date) is to be performed.

*YES: If the file is active, an operator message is issued. The operator can continue or terminate the deletion of the file.

**NO:* The file is to be deleted without the active file check being performed.

CRTDATE Parameter: Specifies the creation date of the file to be deleted. If CRTDATE is specified and the date on the file to be deleted does not match, the file is not deleted and a message is issued to the system operator. The operator can then either retry the operation or terminate the command.

*NONE: No creation date test is made.

creation-date: Enter the date that must match the creation date of the file to be deleted before that file can be deleted. The date must be entered in the format specified by the system values QDATFMT and QDATSEP.

OPTION Parameter: Specifies how the file is to be deleted from the diskette.

*SCRATCH: The expiration date of the file is to be changed to the current system date. The file can still be referenced for input data. However, when a new file is written by the system on the diskette, all expired files are deleted to free space for the new file.

**RMV:* The data file identifier is to be removed from cylinder 0; therefore, the file cannot be referenced for input.

**ERASE:* The data file identifier is to be deleted from cylinder 0, and data in the file is to be overwritten with binary zeros.

Examples

```
DLTDKTLBL  LABEL(FILEA) LOC(*S1)
```

This command scratches (assumed by the system) FILE A on the diskette in slot 1.

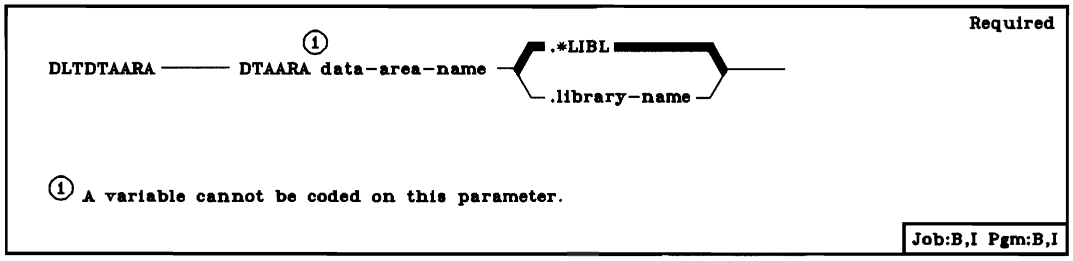
```
DLTDKTLBL  LABEL(MONDAY) LOC(*M1 2) OPTION(*ERASE)
```

This command deletes the file label MONDAY from the second diskette in magazine 1 and overwrites the data with binary zeros.

DLTDTAARA (Delete Data Area) Command

The Delete Data Area (DLTDTAARA) command deletes the specified data area from a library.

Restriction: To use this command, you must have object existence rights for the data area, and read rights for the library.



(1) A variable cannot be coded on this parameter.

DTAARA Parameter: Specifies the qualified name of the data area to be deleted from a library. (If no library qualifier is given, *LIBL is used to find the data area.)

Example

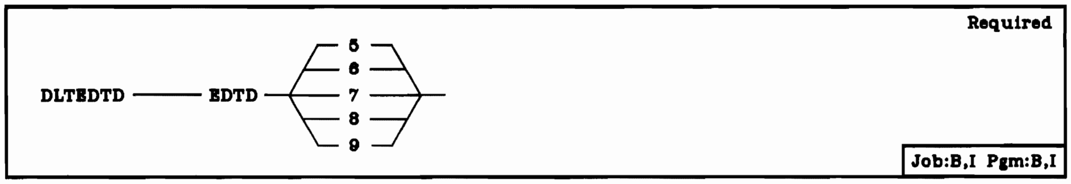
```
DLTDTAARA  DTAARA(MYDATA.MYLIB)
```

This command deletes the data area named MYDATA from the library MYLIB if the user has the proper authority for the data area and the library to do so.

DLTEDTD (Delete Edit Description) Command

The Delete Edit Description (DLTEDTD) command deletes a specified user-defined edit description.

Note: Any DDS (data description specifications) or high-level language programs that have already been created are not affected.



EDTD Parameter: Specifies a digit code (5, 6, 7, 8, or 9) that identifies the user-defined edit description being deleted.

Example

```
DLTEDTD  EDTD(5)
```

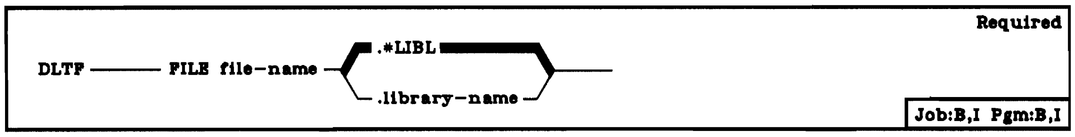
This command deletes the user-defined edit description 5 from the system.

DLTF (Delete File) Command

The Delete File (DLTF) command deletes the specified data base file or device file from the system. Deleting the file also frees the storage space allocated to the file. If a data base file (physical or logical) is deleted, all members contained in the file are also deleted. If the file is in use by a program (opened), the file is not deleted.

Deleting a *damaged* physical or logical file can produce unpredictable side effects, depending on the type and the extent of damage to the file. For example, any logical files based on a damaged physical file may reflect the same damage. Because it is possible that pieces of a damaged file can be left in the system after the DLTF command has been executed, the damaged file should first be copied to another file to recover as much of the file as possible. (The ERRLVL parameter in the CPYF command, by counting the number of errors in the file, determines whether the file is copied; therefore, the error level must be set high enough to permit the copy to take place.) After the damaged file has been copied, the file should be deleted and recreated.

Restrictions: (1) To delete a file, you must have object existence rights for the file and operational authority for the library containing the file. (2) If a physical file is being deleted and a logical file is sharing the access path of, or using the data in, the physical file, the logical file must be deleted first. (3) If a logical file is being deleted and another logical file is sharing its access path, the dependent logical file must be deleted first. (4) If the DLTF command is entered in debug mode and UPDPROD(*NO) was specified on the ENTDBG or CHGDBG command, the name of a physical file that contains data and is in a production library cannot be specified.



FILE Parameter: Specifies the qualified name of the file to be deleted. (If no library qualifier is given, *LIBL is used to find the file.)

Example

```
DLTF  FILE(ORDERS.BILLING)
```

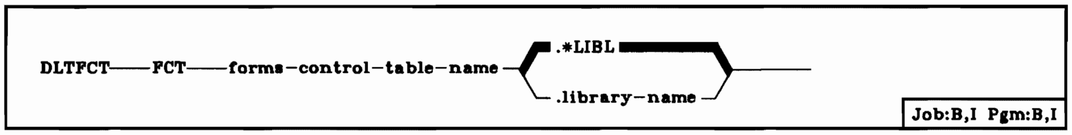
This command deletes the file named ORDERS, stored in the BILLING library. Only the BILLING library is searched for the file.

DLTFCT (Delete Forms Control Table) Command

The Delete Forms Control Table (DLTFCT) command deletes an inactive forms control table (FCT). If any session is active that references the FCT to be deleted, that FCT cannot be deleted.

Restriction: To use this command, you must have object existence rights for the FCT and read rights for the library in which the FCT is stored.

The Delete Forms Control Table (DLTFCT) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



FCT Parameter: Specifies the qualified name of the FCT that is to be deleted. (If no library qualifier is given, *LIBL is used to find the FCT.)

Example

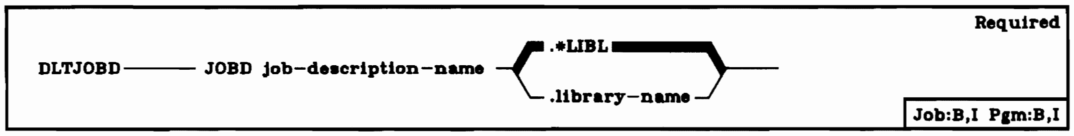
```
DLTFCT FCT(FORMCTRL.USERLIB)
```

This command deletes the forms control table named FORMCTRL in library USERLIB.

DLTJOBD (Delete Job Description) Command

The Delete Job Description (DLTJOBD) command deletes the specified job description object from the system. Any executing jobs using the job description are not affected by its deletion.

Restriction: You must have object existence rights for the job description being deleted, and operational and delete rights for the library containing the job description.



JOBD Parameter: Specifies the qualified name of the job description to be deleted. (If no library qualifier is given, *LIBL is used to find the job description.)

Example

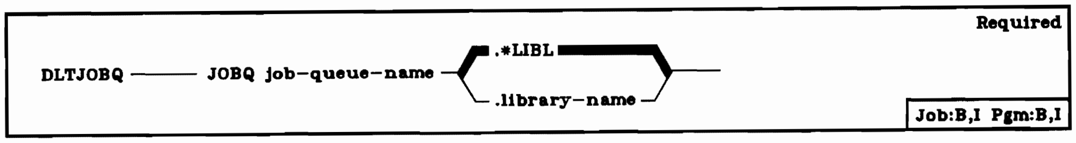
```
DLTJOBD  JOBD(MYJOBD.MYLIB)
```

This command deletes the job description named MYJOBD from the library MYLIB.

DLTJOBQ (Delete Job Queue) Command

The Delete Job Queue (DLTJOBQ) command removes the specified job queue from the system.

Restrictions: To delete a queue, you must have object existence rights and read, add, and delete rights for the specified queue. The queue cannot contain any job entries nor can it be in use by an active subsystem.



JOBQ Parameter: Specifies the qualified name of the job queue to be deleted. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

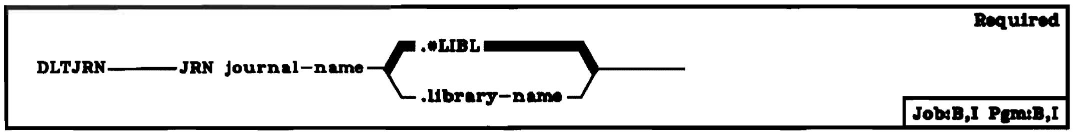
```
DLTJOBQ  JOBQ(SPECIALJQ)
```

This command deletes the job queue named SPECIALJQ from the system.

DLTJRN (Delete Journal) Command

The Delete Journal (DLTJRN) command deletes the specified journal from the system.

Restriction: The journal must not have any objects journaling changes through it when the command is issued. To determine whether the journal is being used, issue the DSPJRNA (Display Journal Attributes) command. If any objects are being journaled, issue the ENDJRNPF (End Journaling Physical File Changes) command to terminate the journaling.



JRN Parameter: Specifies the qualified name of the journal that is to be deleted. (If no library qualifier is specified, *LIBL is used to find the journal.)

Example

```
DLTJRN  JRN(JRNLA.MYLIB)
```

The above command causes the journal named JRNLA in library MYLIB to be deleted from the system.

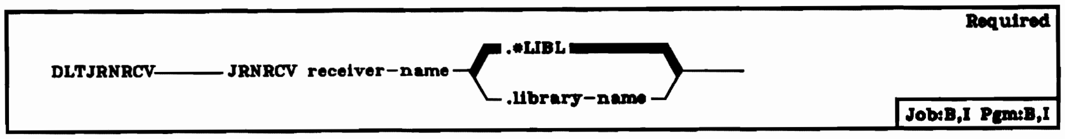
DLTJRNRCV (Delete Journal Receiver) Command

The Delete Journal Receiver (DLTJRNRCV) command deletes the specified journal receiver from the system, which frees the storage space allocated to the receiver.

Restrictions: The journal receiver must not be attached to a journal at the time the command is issued.

The journal receiver must not be in the middle of a chain of online receivers unless it is damaged. (The receivers must be deleted in the same order in which they were detached, to prevent gaps from occuring within the range of receivers. This restriction does not apply if the journal receiver is damaged or if the other of the dual receiver pair is usable).

You can use this command to delete multiple receivers by selecting the appropriate option on the Journal Receiver Directory of the Journal Attributes Display. For more information, refer to the *Additional Considerations* section of the DSPJRNA command.



JRNRCV Parameter: Specifies the qualified name of the journal receiver that is to be deleted. (If no library qualifier is specified, *LIBL is used to find the journal receiver.)

Example

```
DLTJRNRCV  JRNRCV(JRNRCLA.MYLIB)
```

The above command causes the journal receiver named JRNRCLA in library MYLIB to be deleted from the system.

DLTLIB (Delete Library) Command

The Delete Library (DLTLIB) command deletes a specified library from the system after all objects in the library have been deleted. If a library that is to be deleted contains objects, this command first deletes all of the objects and then deletes the library. If the user submitting this command does not have the authority to delete every object in the library, only those for which he does have the authority are deleted. In this case, the library and all the other objects in the library remain unchanged. If any object in the library is in use (locked to a program) when this command is entered, that object is not deleted.

If a library has been damaged, the user should not delete it without first trying to resolve the damage. In most cases, he can resolve the damage by initiating the IMPL sequence to rebuild a user library (including QGPL), or by reinstalling CPF to rebuild the QSYS library. (Refer to the *System Operator's Guide* for the procedures.) Then, if the library is still damaged, it should be deleted. Either a saved version of the library can be restored in its place or the library can be recreated.

Restrictions: (1) To delete a library, the user must have object existence and operational rights for the specified library and object existence rights for every object in it. If he does not have object existence rights for the library, nothing is deleted. If he does not have object existence rights for one or more objects in the library, those objects and the library are not deleted. (2) A library cannot be deleted if it is in the library list of the job in which this command is entered. (3) This command cannot be used to delete the QSYS and QTEMP libraries.

```
                                                                  Required
DLTLIB ——— LIB library-name ———
                                                          Job:B,I Pgm:B,I
```

LIB Parameter: Specifies the name of the library to be deleted.

Example

DLTLIB LIB(W)

This command deletes library W after all its objects have been deleted. If library W contains objects and the user has the authority to delete all of the objects, library W and all of the objects are deleted. If the user does not have authority to delete all of the objects, only those for which he has object existence rights are deleted and the library is not deleted.

DLTLIND (Delete Line Description) Command

The Delete Line Description (DLTLIND) command deletes the specified line description. The line identified in the line description must be varied offline before this command is issued.

If any control units are attached to the line, they must also be varied offline (as well as their associated devices) before the DLTLIND command is issued. The control units are detached when the command is executed; a control unit cannot be used until it is associated with another line. If a new line description is created, the control units can be specified in the CUD parameter of the CRTLIND command. Or the control units can be attached to other lines that already have line descriptions. In this case, the control unit descriptions can be deleted and recreated, with the new line names given in the LINE (if just one) or LINLST (if several, for switched network) parameter of each CRTCUD command.

If a switched line or a nonswitched line with the switched network backup feature is deleted, the name of the deleted line description must be removed from every control unit description that contains that name in its LINLST parameter. For each control unit description, the CHGCUD command must be used to respecify the LINLST parameter without the name of the deleted line.

```
                                                                    Required

DLTLIND——— LINE line-description-name ——

                                                            Job:B,I Pgm:B,I
```

LIND Parameter: Specifies the name of the line description to be deleted.

Example

```
DLTLIND  LIND(LINE01)
```

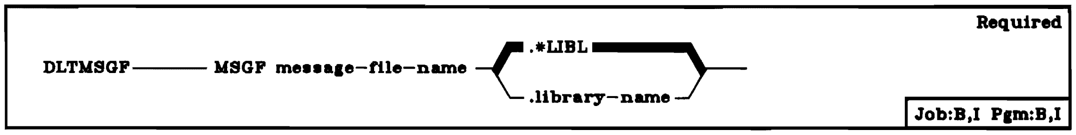
This command deletes the line description of the line named LINE01 from the system.

If the line description to be deleted has any control unit descriptions associated with it, they are detached and a message containing those control unit names is sent to the system operator. The detached control unit descriptions can be associated with a new line description if their names are supplied in the optional CTLU parameter of the CRTLIND command that creates the line description.

DLTMSGF (Delete Message File) Command

The Delete Message File (DLTMSGF) command deletes the specified message file from the system, including all the message descriptions stored in the file. If any messages that use this file exist on queues, the file is deleted and no message text will be available for those messages.

Restrictions: To delete the specified message file, you must have object existence rights for the file and operational rights for the library in which the file is stored. The IBM-supplied message files, QCPFMSG (for CPF and machine interface messages) and the licensed program message files (such as QRPGMSG), cannot be deleted (unless authorized by the security officer).



MSGF Parameter: Specifies the qualified name of the message file to be deleted. (If no library qualifier is given, *LIBL is used to find the file.)

Example

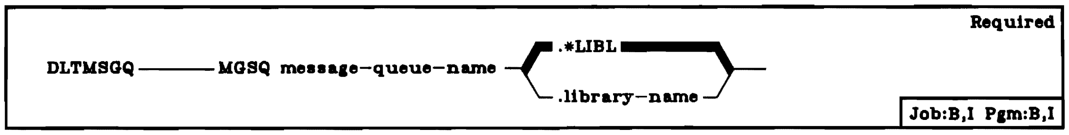
```
DLTMSGF  MSGF(INV)
```

This command deletes the message file named INV. All message descriptions stored in INV are also removed.

DLTMSGQ (Delete Message Queue) Command

The Delete Message Queue (DLTMSGQ) command deletes the specified message queue as well as any messages in it. Any message in the queue that requires a reply is answered with the default reply supplied by that message. If the message queue is being used by another job, the message queue cannot be deleted.

Restrictions: To delete the specified message queue, you must have object existence, operational, read, and delete rights for the queue, and object management rights for the library in which it is stored. The system operator message queue QSYSOPR cannot be deleted. The message queue of each work station (and the system console) cannot be deleted at all.



MSGQ Parameter: Specifies the qualified name of the message queue to be deleted. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

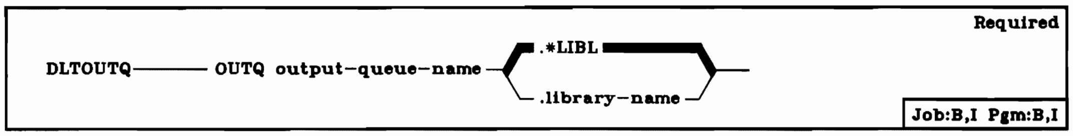
```
DLTMSGQ  MSGQ(JONES)
```

This command deletes the message queue named JONES. All messages stored in JONES are also removed. The library list is used to find the message queue.

DLTOUTQ (Delete Output Queue) Command

The Delete Output Queue (DLTOUTQ) command deletes the specified output queue from the system.

Restrictions: The queue to be deleted must not contain any entries and cannot be in use by a spooling writer. The output for each file must have already been produced or canceled. To delete the queue, you must have object existence rights and read, add, and delete rights for the queue.



OUTQ Parameter: Specifies the qualified name of the output queue to be deleted. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

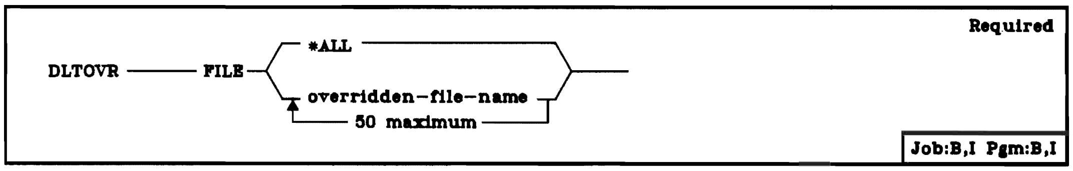
```
DLTOUTQ  OUTQ(PUNCH2)
```

This command deletes the output queue named PUNCH2 from the system.

DLTOVR (Delete Override) Command

The Delete Override (DLTOVR) command deletes one or more file overrides (including message file overrides) that were previously specified in an invocation. That is, for each overridden file named in the DLTOVR command, the override specified in the same invocation as the DLTOVR command is deleted. When the command is entered interactively or outside a program in a batch job, the file overrides for the invocation are deleted; when the command is used in a CL program, the file overrides for that program invocation are deleted. A file override is the result of an override file command.

The DLTOVR command can delete all the file overrides for all the files in the same invocation, or it can delete the file override(s) for a specific file(s) in the same invocation. Only the invocation in which the command is entered has its file overrides deleted. For example, if an override command is entered in one program in a routing step, and then another program is called that also contains override commands, a DLTOVR command entered in the second program can delete only overrides that occurred in that program. The DLTOVR command has no effect on the override command that was entered before the program was called. The deleted file overrides have no effect on subsequent uses of the file.



FILE Parameter: Specifies the names of the overridden files in the invocation that are to have the file overrides that affect them deleted. One or more overridden files can be specified by name, or all files can be specified.

**ALL:* All the file overrides that still exist in the invocation in which this command is entered are to be deleted.

overridden-file-name: Enter the names of one or more overridden files for which the overrides in the invocation are to be deleted.

Examples

Example 1: Deleting Overrides in a Single Invocation

```
1.   OVRDBF  FILE(A) TOFILE(B)
2.   OVRPRTF  FILE(C) TOFILE(D)
3.   OVRCRDF  FILE(E) TOFILE(F)
          •
          •
          •
4.   DLTOVR  FILE(A C)
5.   DLTOVR  FILE(*ALL)
```

If the first three override commands had been specified earlier in the invocation, the files B, D, and F would be overriding files A, C, and E. The fourth command deletes only the file overrides that affect files A and C. The last command deletes all the file overrides that exist in the invocation, which in this case is the command overriding file E, the third command.

Example 2: Deleting Overrides in Multiple Invocations

This example assumes that commands 1, 2, and 11 are entered in the same invocation, invocation number 1. The rest of the commands are in invocation 2.

```
1.   OVRDBF  FILE(A) TOFILE(B)
2.   CALL  PGMA                                       Invocation 1
     -------------------------------------------------------------
Program PGMA
      3.   OVRPRTF  FILE(B) TOFILE(C)
      4.   TFRCTL  PGMB                               Invocation 2

Program PGMB
      5.   OVRCRDF  FILE(E) TOFILE(F)
      6.   CALL  QCAEXEC ('OVRMSGF  FILE(G) TOFILE(H)' 25)
      7.   DLTOVR  FILE(A B)
      8.   MONMSG  MSGID(CPF9841)
      9.   CALL QCAEXEC ('DLTOVR  FILE(*ALL)' 17)
     10.   RETURN
     -------------------------------------------------------------
11.  DLTOVR  FILE(*ALL)                               Invocation 1
```

Command 1 causes an override in invocation 1 from file A to file B. Command 2 calls PGMA and generates another invocation, invocation 2.

In program PGMA, command 3 causes an override in invocation 2 from file B to file C. Command 4 transfers control from PGMA to PGMB in the same invocation, invocation 2. Unlike the CALL command, the TFRCTL command does not generate a new invocation.

In program PGMB, command 5 causes an override in invocation 2 from file E to file F. Command 6 calls QCAEXEC and causes an override from file G to file H. When it is called, QCAEXEC executes as though it is just another command in PGMB, rather than executing as a called program. That is, QCAEXEC executes in the same invocation (invocation 2); it does not generate another invocation (invocation 3).

Command 7 deletes any overrides affecting files A and B in invocation 2. In this case, the override specified by command 3 is deleted, but the override specified by command 1 is not. Because an override for file A is not found in invocation 2, the escape message CPF9841 (override not found) is sent to PGMB. To prevent a function check, a MONMSG command is needed after the DLTOVR command. In this example, command 8 monitors for CPF9841, but specifies no action to be taken if the message is sent. Therefore, when CPF9841 is received, it is monitored and ignored by command 8, and control is passed to the next command in the program.

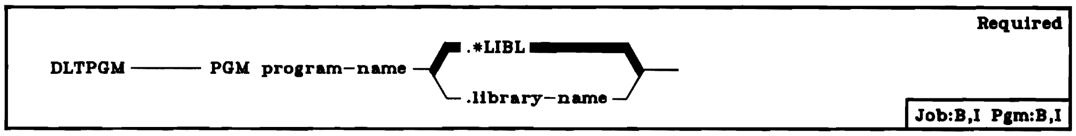
Command 9 deletes all remaining overrides in invocation 2. The overrides specified by commands 5 and 6 are deleted, but the override specified by command 1 is not.

Command 10 causes a return to invocation 1, and invocation 2 is terminated. If any overrides had been specified in invocation 2 that were not deleted by DLTOVR commands, they are deleted when invocation 2 terminates. Command 11 causes all remaining overrides in invocation 1 to be deleted; the override specified by command 1 is deleted.

DLTPGM (Delete Program) Command

The Delete Program (DLTPGM) command deletes an executable program from a library. If the program is currently being executed, the program execution is abnormally terminated when this command is issued. Any HLL or CL program can be deleted.

Restrictions: (1) To use this command, you must have object existence rights for the program, and update and delete rights for the library in which it is stored. (2) If you delete a program that is currently in debug mode, a function check occurs if an implicit reference is made to the deleted program (for example, a CHGVAR command specifies PGM(*DFTPGM)). To prevent function checks, you can use the RMVPGM command to remove the program from debug mode before you delete it. If the program is to be recompiled while you are in debug mode, you should: remove the program from debug mode (RMVPGM), delete it from the system (DLTPGM), change and recompile the program, and add the new version of the program to debug mode (ADDPGM).



PGM Parameter: Specifies the qualified name of the program to be deleted. (If no library qualifier is given, *LIBL is used to find the library.)

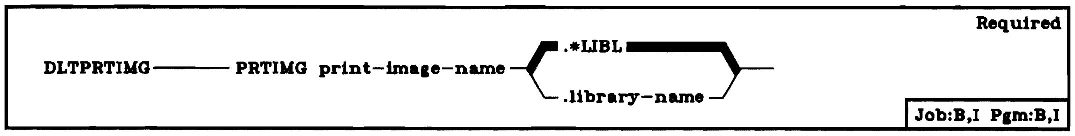
Example

```
DLTPGM  PGM(PROG1.LIB1)
```

This command deletes the program PROG1 from the library LIB1 if the user has the proper authority for the program and library to do so.

DLTPRTIMG (Delete Print Image) Command

The Delete Print Image (DLTPRTIMG) command deletes the specified print image.



PRTIMG Parameter: Specifies the qualified name of the print image to be deleted. (If no library qualifier is given, *LIBL is used to find the print image.)

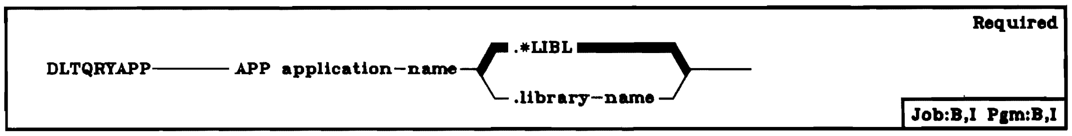
Example

```
DLTPRTIMG  PRTIMG(CHAR48PI)
```

This command deletes the print image named CHAR48PI from the system.

DLTQRYAPP (Delete Query Application) Command

The Delete Query Application (DLTQRYAPP) command deletes an existing query application. The Query Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Query Utility, refer to the *IBM System/38 Query Utility Reference Manual and User's Guide,* SC21-7724.



APP Parameter: Specifies the qualified name of the query application you are deleting. (If no library qualifier is specified, *LIBL is used to find the application.)

Example

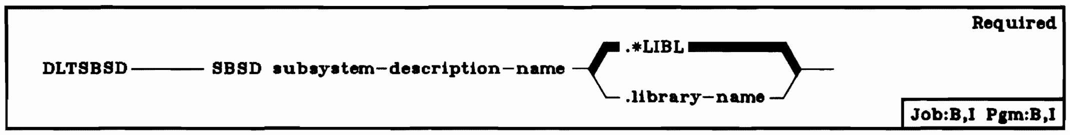
```
DLTQRYAPP APP(QDATA.LIB1)
```

This command deletes the query application QDATA from the library named LIB1.

DLTSBSD (Delete Subsystem Description) Command

The Delete Subsystem Description (DLTSBSD) command deletes the specified subsystem description (including any work entries or routing entries that were added to it) from the system. The associated subsystem must be inactive before it can be deleted.

Restrictions: This command cannot be executed if an active subsystem is associated with this subsystem description. You must have object existence rights for the subsystem description being deleted, and operational and delete rights for the library.



SBSD Parameter: Specifies the qualified name of the subsystem description that is to be deleted. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

Example

```
DLTSBSD  SBSD(BAKER.LIB1)
```

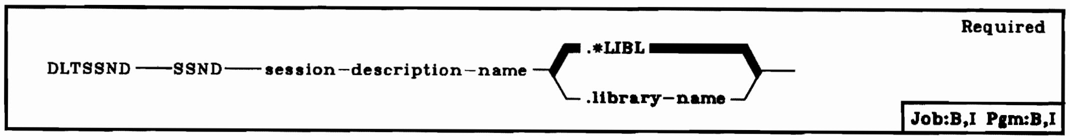
This command deletes the inactive subsystem description called BAKER from library LIB1.

DLTSSND (Delete Session Description) Command

The Delete Session Description (DLTSSND) command deletes an inactive RJEF session description.

Restriction: To use this command, you must have object existence rights for the session description and read rights for the library in which the session description is stored.

The Delete Session Description (DLTSSND) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description that is to be deleted. (If no library qualifier is given, *LIBL is used to find the session description.)

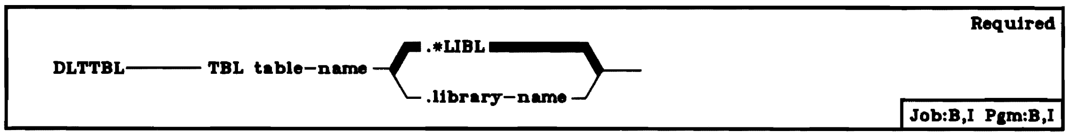
Example

```
DLTSSND SSND(RJE.USERLIB)
```

This command deletes the inactive session description called RJE from library USERLIB.

DLTTBL (Delete Table) Command

The Delete Table (DLTTBL) command deletes the specified table.



TBL Parameter: Specifies the qualified name of the table to be deleted. (If no library qualifier is given, *LIBL is used to find the table.)

Example

```
DLTTBL  TBL(SCRAMTBL)
```

This command deletes the table named SCRAMTBL from the system.

DLTUSRPRF (Delete User Profile) Command

The Delete User Profile (DLTUSRPRF) command deletes a user profile from the system. The user who enters this command must have object existence authority for the user profile being deleted.

If a user profile has been damaged by some program logic error or system failure, it should be deleted by the DLTUSRPRF command and recreated by the CRTUSRPRF command. Before the profile is deleted, the objects that it owns should be transferred to a new or different profile by the CHGOBJOWN command. After a user profile has been recreated, the owned objects can be transferred back to it. Also, any authority that was granted to the damaged profile has to be regranted to the new profile by the GRTOBJAUT command.

If multiple profiles have been damaged, a saved version of the user profiles should be restored to the system via the RSTUSRPRF and RSTAUT commands.

Restrictions: (1) The user profile cannot be deleted if a user is currently executing under the profile, or if it owns any objects. All objects in the user profile must first be transferred to new owners (by the CHGOBJOWN command) or be deleted from the system. Authority granted to the user does not have to be explicitly revoked (by the RVKOBJAUT command); it is automatically revoked when the user profile is deleted. (2) To delete any object, including a user profile, you must have object existence rights for the object. User profiles QSECOFR, QPSR, QSYS, QCE, QDBSHR, QSPL, and QRJE cannot be deleted.

```
                                                                    Required
DLTUSRPRF———— USRPRF user-name————
                                                          Job:B,I Pgm:B,I
```

USRPRF Parameter: Specifies the name of the user profile to be deleted from the system.

Example

```
DLTUSRPRF  USRPRF(JJADAMS)
```

This command causes the user profile named JJADAMS to be deleted from the system (if no objects are owned by the user profile and no user is currently executing under it).

DMPCLPGM (Dump CL Program) Command

The Dump CL Program (DMPCLPGM) command dumps all variables (declared in the CL program in which the command executes) and all messages on the program's message queue to a spooled printer file (QPPGMDMP). This command is valid only in a CL program; after the program is dumped, it continues to execute. To use this command, you must have authority to read the program.

```
                                                                  Required
  DMPCLPGM ———
                                                                  Pgm:B,I
```

There are no parameters for this command.

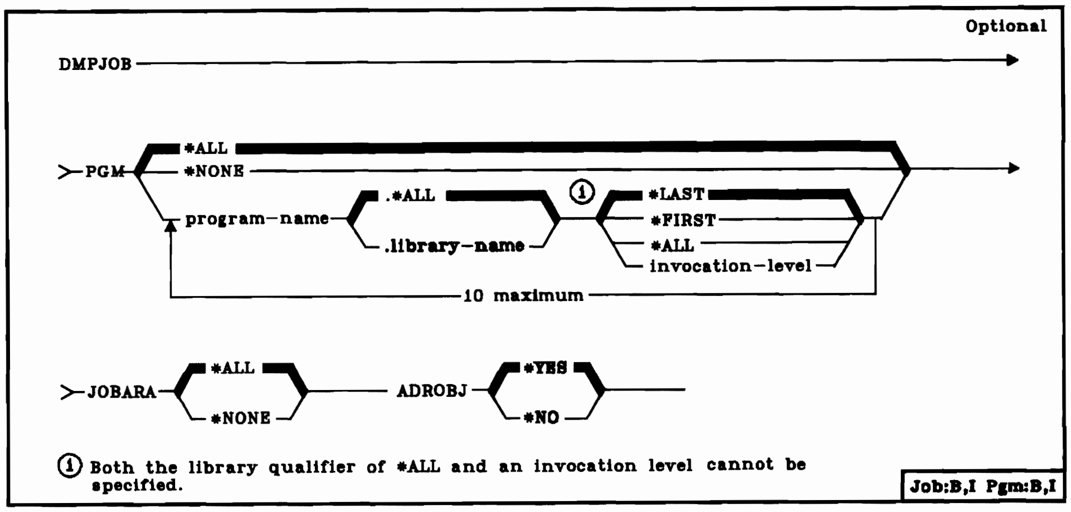
Example

```
PGM
DCL . . .
DLC . . .
MONMSG  MSGID(CPF9999)  EXEC(GOTO DUMP)
   •
   •
   •
RETURN
DUMP:  DMPCLPGM
ENDPGM
```

This CL program monitors for the function check message CPF9999. If a function check occurs in the program, control is passed to the command at label DUMP. This causes a dump of the program's message queue and causes the program's variables to be printed. This dump can be used as an aid in determining the cause of the function check.

DMPJOB (Dump Job) Command

The Dump Job (DMPJOB) command dumps the basic data structures, or specific invocations of the current job or of the job being serviced as a result of the Service Job (SRVJOB) command. The information is dumped to a spooled printer file (QPSRVDMP) to be printed. If the output is not to be spooled and the printer is not available, the printer file is overridden. The dump includes formatted information about the specified programs, and dumps of individual CPF objects and system objects associated with the job.



PGM Parameter: Specifies a program to be dumped. PGM can be either a single value or a list of values (10 maximum).

ALL: All programs on the invocation stack are to be dumped.

**NONE:* No programs are to be dumped. Only the invocation and activation lists are dumped.

qualified-program-name: Enter the qualified name of the invoked program to be dumped. A maximum of 10 characters for the program name can be specified. If no library qualifier is specified, *ALL is used to determine which program invocation to dump. Only the program name is to be used in the selection of invocations to be dumped. If *ALL is specified, an invocation-level cannot be specified.

LAST: The last invocation with the name specified is to be dumped.

**FIRST:* The first invocation with the name specified is to be dumped.

**ALL:* All invocations with the name specified are to be dumped.

invocation-level: Enter the level of invocation for a program with multiple invocations in the stack. If *ALL is specified for the library name, the invocation level cannot be specified.

JOBARA Parameter: Specifies whether the job structure areas of the process will be dumped. Job structure areas consist of the following:

- Work Control Block
- Library Search List
- Job Temporary Library
- Job Message Queue
- Spool Control Block
- Data Management Communications Queue
- Service Communication Object
- Process Definition Template
- Process Lock List
- MI Response Queue

**ALL:* The job structure areas are to be dumped.

**NONE:* The job structure areas are not to be dumped.

ADROBJ Parameter: Specifies whether objects addressed from the program storage of a program being dumped will also be dumped. If *NONE is specified for the PGM parameter, no addressed objects will be dumped.

**YES:* The addressed objects are to be dumped.

**NO:* The addressed objects are not to be dumped.

Example

```
DMPJOB  PGM((UPDATE.QGPL *FIRST) (MASTER.PAYROLL *ALL)) +
          JOBARA(*ALL) ADROBJ(*NO)
```

This command dumps the first occurrence of UPDATE.QGPL in the invocation stack and all occurrences of MASTER.PAYROLL. Also dumped will be the job structure areas.

```
DMPJOB
```

This command dumps the entire job structure.

```
DMPJOB  PGM(*NONE) JOBARA(*NONE)
```

This command dumps the invocation and activation lists.

DMPJOBINT (Dump Job Internal) Command

The Dump Job Internal (DMPJOBINT) command dumps the machine internal data that is related to the machine process in which the current job or the job being serviced as a result of the Service Job (SRVJOB) command is executing. This data is for use by IBM service representatives. When the internal data is dumped, a dump identifier is sent in a message to the requester of the job issuing the command. The List Internal Data (LSTINTDTA) command can be used to print the dump.

Restriction: Only the programmer, system operator, IBM service representative, or security officer can use this command.

```
DMPJOBINT ────────
                                                        Job:B,I Pgm:B,I
```

There are no parameters for this command.

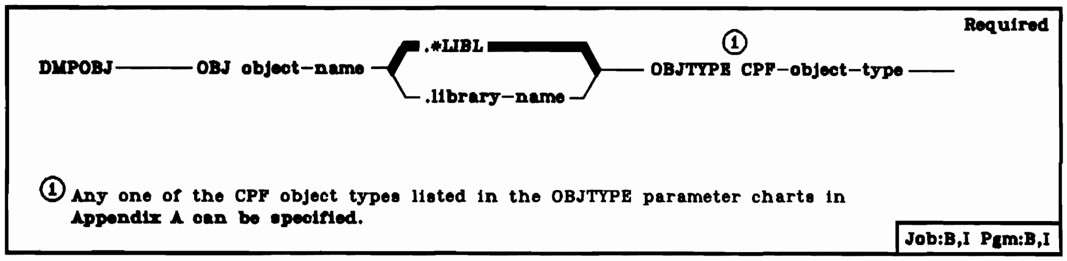
Example

DMPJOBINT

This command dumps, for the job in which the command is entered, the machine internal data associated with the job. A message that includes the dump identifier is sent to the user entering the command.

DMPOBJ (Dump Object) Command

The Dump Object (DMPOBJ) command dumps the contents and/or attributes of the specified CPF object to a spooled printer file named QPSRVDMP. (Whether the contents and/or attributes *can* be dumped depends upon the object type.) If the printed output is not to be spooled, and the printer is not available, the printer file (QPSRVDMP) is overridden. Any library or CPF object that is stored in a library can be dumped, but only one object can be specified at a time on this command.



① Any one of the CPF object types listed in the OBJTYPE parameter charts in Appendix A can be specified.

OBJ Parameter: Specifies the qualified name of the CPF object to be dumped. (If no library qualifier is given, *LIBL is used to find the specified object.) Only the CPF objects that are stored in libraries can be dumped. Refer to the OBJTYPE parameter for the valid types of objects.

OBJTYPE Parameter: Specifies the object type of the CPF object to be dumped. Any one of the CPF object types can be specified; refer to the charts in the expanded description of the OBJTYPE parameter in Appendix A. To dump a program, for example, enter the value *PGM.

Examples

```
DMPOBJ  OBJ(ORDERIN.ORDENT) OBJTYPE(*FILE)
```

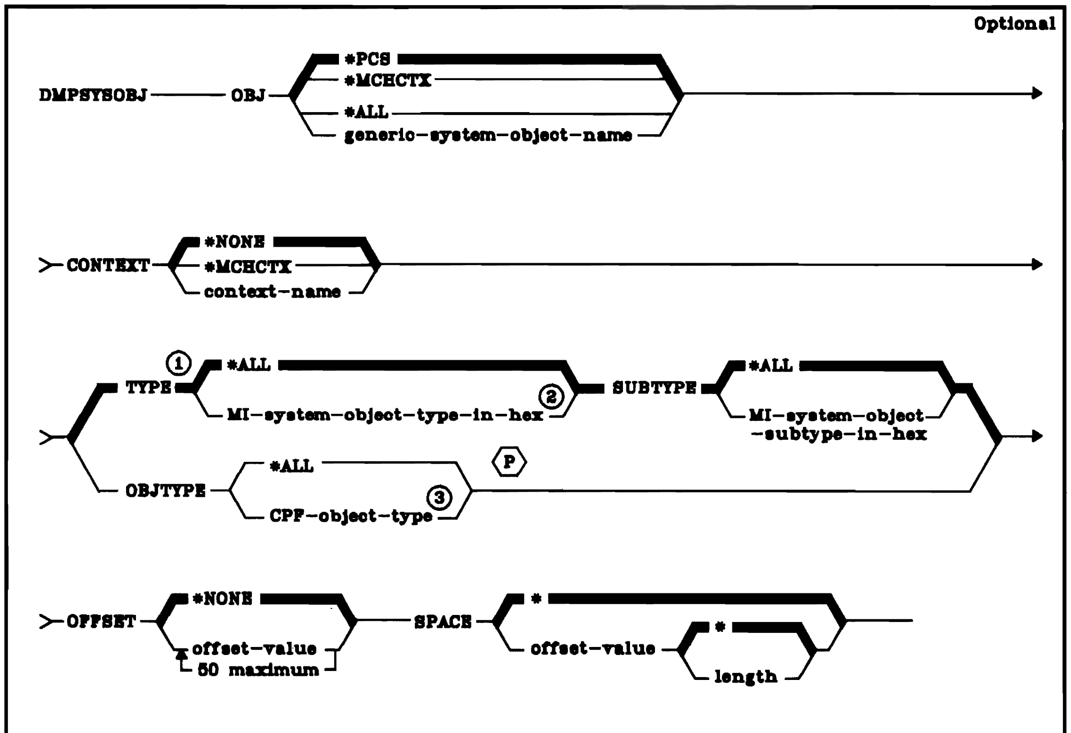
This command dumps the contents of the file named ORDERIN that is stored in the ORDENT library.

```
DMPOBJ  OBJ(MYPROG) OBJTYPE(*PGM)
```

This command dumps the first occurrence of the program MYPROG that is found by the library list. The dump is spooled to the printer output file QPSRVDMP.

DMPSYSOBJ (Dump System Object) Command

The Dump System Object (DMPSYSOBJ) command is used primarily for problem determination. It dumps the contents and/or attributes of MI system objects to a spooled printer file named QPSRVDMP. If the printed output is not to be spooled, and the printer is not available, the printer file (QPSRVDMP) is overridden. Any MI object that is stored in any context or that is addressable through an object stored in a context can be dumped. A specific object, a generic group, or all of the MI objects in a context can be specified. The dump can also be limited to objects of a specified type and, optionally, of a specified subtype.



(1) To code the following parameters *positionally*, you must code them in this order, using *N for those not being specified: TYPE, SUBTYPE, and OBJTYPE.
(2) To specify an MI system object type, refer to the TYPE parameter description for a list of the valid MI type codes.
(3) To specify one of the CPF object types, refer to the charts in the *expanded* description of the OBJTYPE parameter in Appendix A.

Job:B,I Pgm:B,I

OBJ Parameter: Specifies which of the MI system objects are to be dumped. The name of a specific object, the generic name of a group of objects, the process control space of the job, the machine context, or all of the MI objects in a context can be specified. If a library name is specified, the library is dumped, but not the objects in it.

If OBJ(QTEMP) is specified along with either OBJTYPE(*LIB) or TYPE(04) SUBTYPE(01), then the temporary job context associated with the job in which this command is entered, or the job being serviced as a result of the SRVJOB command, is dumped. In either case, the CONTEXT parameter is ignored.

*PCS: The process control space to be dumped is that of the current job or that of the job being serviced as a result of the SRVJOB command. OBJ(*PCS) can be used with the OFFSET and SPACE parameters to dump objects in the job structure. If OBJ(*PCS) is specified, the CONTEXT, TYPE, SUBTYPE, and OBJTYPE parameters are ignored.

**MCHCTX:* The machine context (which contains a list of the objects in the context) is to be dumped. If OBJ(*MCHCTX) is specified, all the other parameters in this command are ignored.

**ALL:* All the MI system objects in the specified context are to be dumped if they match the requirements specified in TYPE and SUBTYPE (for MI objects), or OBJTYPE (for CPF objects).

generic-system-object-name: Enter the CPF or MI generic object name that identifies the group of MI system objects to be dumped. An MI object name can have as many as 30 characters in it.

system-object-name: Enter the name of the CPF or MI object that is to be dumped. A maximum of 30 characters can be entered. If more than one object has the same name, all objects having that name and matching the attributes specified by the CONTEXT parameter and either the TYPE and SUBTYPE parameters or the OBJTYPE parameter are dumped. If a specific object is to be dumped, the CONTEXT, TYPE, and SUBTYPE parameters or the CONTEXT and OBJTYPE parameters should be specified.

CONTEXT Parameter: Specifies in which context or library the objects to be dumped are to be found.

*NONE: The object specified by the OBJ parameter is not in any context. *NONE is valid only if *PCS or *MCHCTX is specified or assumed for the OBJ parameter, or if OBJ(QTEMP) is specified along with either OBJTYPE(*LIB) or TYPE(04) SUBTYPE(01).

**MCHCTX:* The objects to be dumped are in the machine context. The following CPF object types, whose MI system object names are given in parentheses, can reside *only* in the *machine* context: library (context), user profile, device (logical unit) description, line (network) description, and control unit (controller) description. (These types are included in the table given in the TYPE parameter description.) *MCHCTX is valid only if one of these five object types is to be dumped.

context-name: Enter the name of the context containing the objects to be dumped. The name of a library, such as QGPL or QTEMP, can be specified. If QTEMP is specified, the objects to be dumped are in the temporary job context associated with the job in which this command is entered or the job being serviced as a result of the SRVJOB command.

TYPE Parameter: Specifies the type of MI objects to be dumped.

*ALL: All MI object types in the specified context that have the specified name (if used) are to be dumped.

MI-system-object-type-in-hex: Enter the hexadecimal value that specifies the type of MI system objects to be dumped. The following table shows the MI system objects and their hexadecimal type codes. The value must be specified with both characters, but it does not have to be enclosed in apostrophes.

MI System Object	MI Type Code
Access group	01
Program	02
Context (library)[1]	04
User profile[1]	08
Queue	0A
Data space	0B
Data space index	0C
Cursor	0D
Index	0E
Logical unit (device) description[1]	10
Network (line) description[1]	11
Controller (control unit) description[1]	12
Space	19
Process control space	1A

[1]If this object is specified for TYPE, then CONTEXT(*MCHCTX) must also be specified.

SUBTYPE Parameter: Specifies the subtype of the specified MI objects to be dumped, or specifies that all subtypes are to be dumped.

*ALL: All the subtypes of the specified MI objects are to be dumped.

MI-system-object-subtype-in-hex: Enter the specific subtype of the MI system objects to be dumped. The subtypes are in the range of 00 through FF. However, the subtype specified must be for an MI object actually in the specified context. If TYPE(*ALL) is specified, a specific subtype *cannot* be specified.

OBJTYPE Parameter: Specifies the object type of CPF objects that are to have their associated MI system objects dumped. If OBJTYPE is specified, neither TYPE nor SUBTYPE can be specified.

**ALL:* The specified MI objects of all CPF object types are to be dumped.

CPF-object-type: Enter the specific CPF object type that is to have its associated MI system objects dumped. (For a list of the valid object types that can be specified, see the chart in the expanded description of the OBJTYPE parameter in Appendix A.)

OFFSET Parameter: Specifies a list of values to be used as offsets to indirectly address a single object that is to be dumped. The values must be positive hexadecimal values or zeros that, when added to a pointer, result in valid addresses. If an offset of zero is added to a system pointer, the result is a space pointer to the start of the space associated with the object that is addressed by the system pointer. (In this discussion, the associated space of a space object is the space itself.)

Note: The OFFSET and SPACE parameters cannot be specified if *ALL or a generic object name is specified for the OBJ, TYPE, SUBTYPE, or OBJTYPE parameters.

*NONE: No offset is being specified. The object located through the context is dumped.

offset-value: Enter the list of offsets to pointers that are used to address the object or space to be dumped. The values specified in this parameter are used as follows:

1. The first offset is added to a space pointer that points to the associated space of the object located through the context. The result is a space pointer that points to a location further into the space.
   a. If only one offset value is specified in this parameter, step 2 is not performed and the dump, as indicated by the rest of the parameters in the command, is taken.
   b. If more than one offset is specified in this parameter, step 2 is repeated for each additional offset given.

2. Regarding the location pointed to by the space pointer produced in the previous step:
   a. If the location does not contain another pointer, the command is terminated, an error message is sent to the user, and no dump is taken.
   b. If the location contains a space pointer, the (next) offset is added to it. The result is another space pointer that points to either the same or a different space or associated space.
   c. If the location contains a system pointer, the associated space pointer is set from the system pointer, and the (next) offset is added to the space pointer. The result is a space pointer that points to a location in the associated space of the object addressed by the system pointer.

The result of step 2b or 2c is a space pointer that is used to perform step 2 again if there is another offset. If the last offset has been used, the final result is a location contained in a space pointer that is used as follows:

- If the resulting location contains a system pointer and the SPACE parameter is not specified, the system object pointed to by the system pointer is dumped. If the SPACE parameter is specified, the SPACE specification determines the portion of the system object that is to be dumped.

- If the resulting location contains a space pointer and the SPACE parameter is not specified, the portion of the space that starts at the location pointed to by the space pointer is dumped. If the SPACE parameter is specified, the SPACE specification determines the portion of the space to be dumped.

The following chart shows the offsets into the process control space (PCS) at which there are pointers to the (other) components of a job structure. If one of these offsets is specified, OBJ(*PCS) must be specified or assumed.

Object (Descriptive Name and Abbreviation)	**Object Name**	**Offset**
Data management communications queue (DMCQ)	QDMDMCQ	20
Job message queue (JMQ)	QJOBMSGQ	F0
Job temporary context (QTEMP)	QTEMP	40
MI response queue (MIRQ)	(none)	80
Process definition template (PDT)	PDT	60
Process automatic storage area (PASA)	PASA	60 E0
Process static storage area (PSSA)	PSSA	60 F0
Process access group (PAG)	PAG	60 100
Spooling control block (SCB)	QSPSCB	100
Work control block table (WCBT)	QWCBT	10

SPACE Parameter: Specifies the area of a space or associated space to be dumped. The space is pointed to by the final pointer determined by the OFFSET parameter. If the OFFSET parameter is not specified, the final pointer is a system pointer to the specified object in the context. (See *Note* in the OFFSET parameter description.)

*: If the final pointer is a system pointer, the object pointed to by that pointer is dumped. If the final pointer is a space pointer, the portion of the space that starts at the location pointed to by that pointer is dumped.

offset-value: Enter the value to be added to the final pointer to point to the beginning of the area to be dumped. The value specified must be a positive hexadecimal value or zero and, when added to the final pointer, must result in a valid address.

*: The rest of the space pointed to as a result of the offset value is to be dumped.

length: Enter a positive hexadecimal value that specifies the length of the area to be dumped. If the length specified is greater than the actual length of the space, only the actual space available is dumped.

Examples

```
DMPSYSOBJ  CONTEXT(QTEMP)  TYPE(0E)
```

This command dumps the contents and attributes of all the indexes in the temporary job context to a spooled output file for printing. MI indexes are identified by the type code 0E.

```
DMPSYSOBJ  OBJ(WS1)  CONTEXT(*MCHCTX)  OBJTYPE(*DEVD)
```

This command dumps the device description for work station WS1, which is stored in the machine context.

```
DMPSYSOBJ  OBJ(*PCS)  SPACE(0  2A0)
```

This command dumps the work control block from the space associated with the process control space for the job.

```
DMPSYSOBJ  OBJ(*PCS)  OFFSET(60 E0 10 10)  SPACE(0 20)
```

This command dumps the second invocation entry of the process automatic storage area (offset 60 E0) for a length of 32 bytes (SPACE(0 20)). If the third invocation level were to be dumped, OFFSET(60 E0 10 10 10) would be specified.

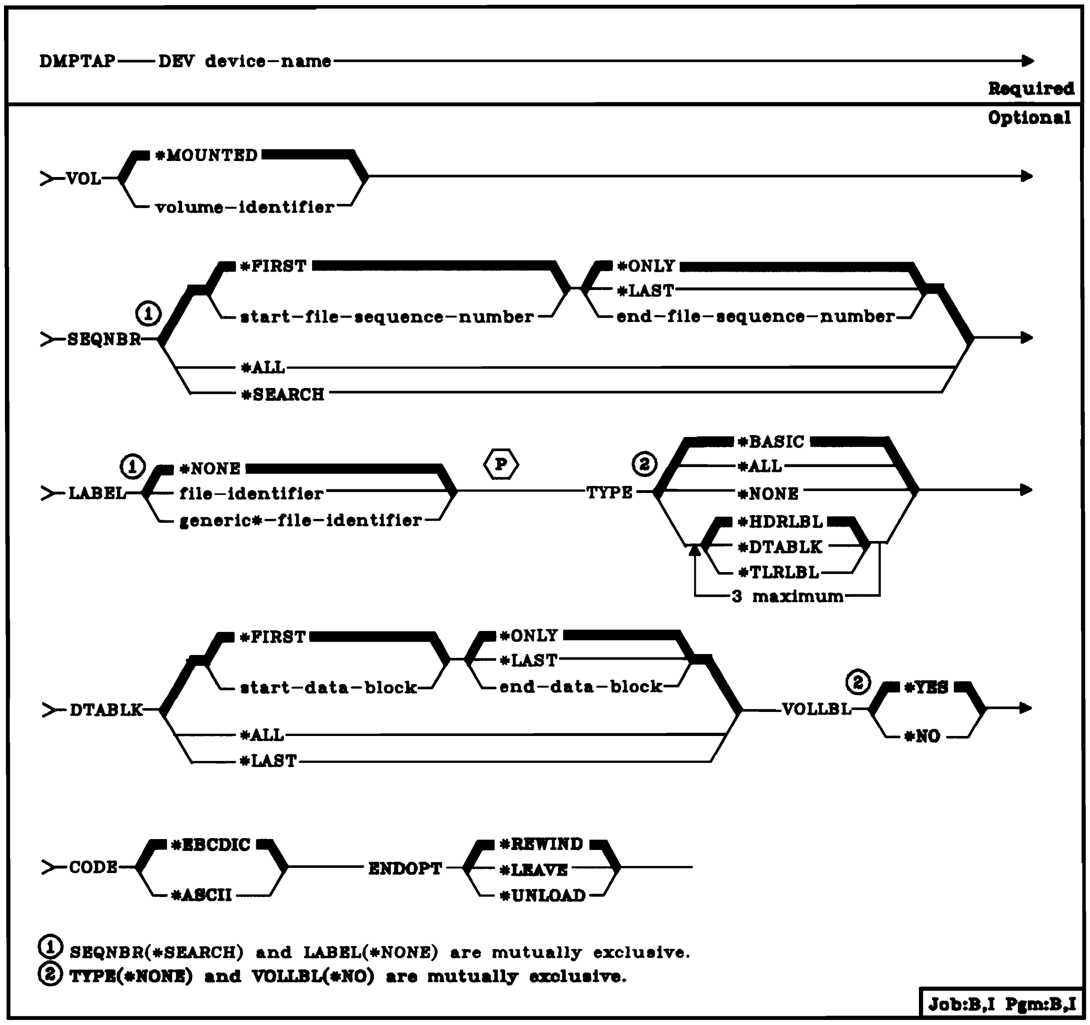
DMPTAP (Dump Tape) Command

The DMPTAP (Dump Tape) command dumps label information or data blocks or both from standard labeled or nonlabeled magnetic tape to a spooled printer file named QPTAPDMP. This command allows you to dump one or more data files from the tape volume, writing the information to a print file.

The tape volume to be dumped must be mounted on the specified device. After the DMPTAP command is entered, as much of the tape as necessary is read before the requested information is printed.

Data files on secured tapes can be dumped by the security officer only; any user can dump label information on secured tapes.

The defaults are such that execution of the DMPTAP command with defaults results in the printing of the tape label areas and a minimal amount of data from the first file. This command can help determine the record format of a nonlabeled tape data file, or to determine the exact contents of all label information for a labeled data file.



① SEQNBR(*SEARCH) and LABEL(*NONE) are mutually exclusive.
② TYPE(*NONE) and VOLLBL(*NO) are mutually exclusive.

DEV Parameter: Specifies the name of the tape device where the labeled or nonlabeled volume to be dumped is mounted.

VOL Parameter: Specifies the volume identifier of the labeled tape to be dumped, or indicates that any mounted tape reel is to be dumped.

*MOUNTED: Specifies that any labeled or nonlabeled volume mounted on the specified device is to be dumped. Note that VOL(*MOUNTED) and LABEL(*NONE) must be specified to dump a nonlabeled volume.

volume-identifier: Enter the identifier of the labeled volume to be dumped. This value can be specified only for dumping a labeled volume. If the mounted tape has a different volume identifier than specified or is a nonlabeled volume, an error message is sent to the user of the DMPTAP command and the tape is not dumped.

SEQNBR Parameter: Specifies the range of sequence numbers for the data files that should be dumped. If SEQNBR(*ALL) is specified, then all data files on the tape are dumped. If a range of sequence numbers is specified, then only data files in that range of data file sequence numbers are dumped. Note that the data files dumped may be further restricted using the LABEL parameter.

The sequence number for a labeled tape data file is stored on labels that precede and follow the data in the file. For a nonlabeled volume the data file sequence number is determined by the number of tape marks from beginning of tape.

*FIRST: The range of data files to be dumped should begin with the first file on the tape volume (regardless of its sequence number).

start-file-seqnbr: The range of data files to be dumped should begin with the data file with the specified sequence number. Enter a number that is less than or equal to the end-file-seqnbr value.

<u>*ONLY:</u> Only a single data file (specified by the start-file-seqnbr) will be dumped.

**LAST:* The range of data files to be dumped should proceed from the start-file-seqnbr data file to last data file on the end of the reel.

end-file-seqnbr: The range of data files to be dumped should end with the specified sequence number data file. Enter a number that is greater than or equal to the start-file-seqnbr.

**ALL:* All data files on the mounted volume should be dumped.

**SEARCH:* The mounted volume is searched for a data file with an identifier that matches the LABEL parameter value; when a match is found, the data file is dumped. If VOLLBL(*NO) is specified and the last operation on the device specified ENDOPT(*LEAVE) (the tape is positioned at the location at which the last operation terminated), the file search begins with the first data file beyond the current tape position. If VOLLBL(*YES) is specified or ENDOPT(*LEAVE) was not used for the last operation (or if the tape was manually rewound since an ENDOPT(*LEAVE) operation), the search begins with the first data file on the volume. SEQNBR(*SEARCH) is not valid when LABEL(*NONE) is specified, and cannot be used to dump a nonlabeled tape volume.

LABEL Parameter: Specifies the identifier of the specific data files that should be dumped. The file identifier for a tape data file is stored on labels that precede and follow the data in the file.

<u>*NONE:</u> All data files on the mounted volume in the specified SEQNBR range will be dumped. Note that VOL(*MOUNTED) and LABEL(*NONE) *must* be used to dump a nonlabeled tape volume.

file-identifier: Enter the data file identifier (17 alphameric characters maximum) of the data files to be dumped. The system will compare the LABEL identifier with the data file identifier on the labels of each file in the range specified by the SEQNBR parameter. All data files with an identifier that matches the LABEL identifier are dumped; any data file with an identifier that does not match the LABEL identifier is not dumped.

generic-label-file-identifier:* Specifies a character string for a generic label identifier (17 alphameric characters maximum), which contains at least one character followed by an asterisk ('*'). Any tape file that has a file identifier with the same prefix as the generic data-file-identifier will be dumped.

TYPE Parameter: Specifies the type of information that is to be dumped. The dump may consist of the data file header labels or trailer labels (for a labeled tape volume only), data blocks from the data portion of the file, or all three types of information. If a nonlabeled tape volume is mounted, only *BASIC, *ALL, or *DTABLK can be specified or an error message is sent to the user of the DMPTAP command and the volume is not dumped.

BASIC: For a standard-labeled volume, the dump includes header labels and the data blocks specified by the DTABLK parameter value. For a nonlabeled volume, only the data blocks (DTABLK parameter) are dumped.

**ALL:* For a standard-labeled volume, the dump includes header labels, trailer labels, and data blocks. For a nonlabeled volume, TYPE(*ALL) dumps only data blocks (since there are no labels).

**NONE:* No data file is to be dumped. If TYPE(*NONE) is specified, the tape volume to be dumped must be labeled, and VOLLBL(*NO) cannot be specified, or an error message is sent to the user of the DMPTAP command.

**HDRLBL:* The data file header labels should be dumped. Header labels immediately precede the data in the file to which they apply. All header labels for the specified data files will be dumped, including user-specified header labels (if any exist). TYPE(*HDRLBL) is not valid for nonlabeled volumes.

**DTABLK:* One or more data blocks from the file data should be dumped. The blocks within the data file that should be dumped are specified by the DTABLK parameter value.

**TLRLBL:* All data file trailer labels should be dumped. Trailer labels immediately follow the data in the file to which they apply. All the trailer labels for the specified data files will be dumped, including user-specified trailer labels (if any exist). TYPE(*TLRLBL) is not valid for nonlabeled volumes.

DTABLK Parameter: Specifies which data blocks should be dumped. This parameter is used to limit the amount of tape file data dumped to the printer. If neither TYPE(*BASIC) nor TYPE(*ALL) is specified and the TYPE parameter value does not include *DTABLK, this parameter is ignored.

<u>*FIRST:</u> The data blocks to be dumped should begin with the first block in the data file.

start-data-block: Enter the number of the first data block within each file to be dumped. If this number is greater than the number specified for the end-data-block portion of the DTABLK parameter, an error message is sent to the user who requested the dump, and the tape is not dumped. If the start-data-block value is larger than the actual number of data blocks in the data file, then the last data block in the file is dumped (with no error messages).

<u>*ONLY:</u> Only the data block specified by the first part of the DTABLK parameter is to be dumped.

**LAST:* The range of data blocks to be dumped should proceed from the data block specified by the start-data-block value to the last block in the file.

end-data-block: Enter the number of the last data block within each file to be dumped. If this number is less than the number specified for the start-data-block part of the DTABLK parameter, an error message is sent to the user who requested the dump, and the tape is not dumped. If the end-data-block value is larger than the actual number of data blocks in the data file, then all blocks from the start-block number to the end of the file are dumped (with no error messages).

**ALL:* All data blocks in the specified data file(s) on this volume should be dumped. If a data file is continued from another volume or continues onto another volume, only the part of the data file that is stored on this volume will be dumped.

**LAST:* Only the last data block in the data file is to be dumped.

VOLLBL Parameter: Specifies whether volume labels are to be dumped. This parameter is ignored for nonlabeled volumes.

<u>*YES:</u> All volume labels (including user-specified labels) are to be dumped.

**NO:* No volume labels are to be dumped; the volume listing does, however, include the volume identifier of a labeled volume and other basic information for any dumped tape.

CODE Parameter: Specifies the type of character code used for the data recorded on the tape. For a labeled volume, the CODE parameter is ignored because the tape labels determine whether the data is recorded in EBCDIC or ASCII character code.

<u>*EBCDIC:</u> The tape contains data in the EBCDIC character code. The dump output will contain the hexadecimal value and the EBCDIC character equivalent of each data byte.

**ASCII:* The tape contains data in the ASCII character code. The dump output will contain the hexadecimal value and the ASCII character equivalent of each data byte.

ENDOPT Parameter: Specifies whether the tape should be rewound, left positioned where the dump ends, or rewound and unloaded after it has been dumped.

<u>*REWIND:</u> The tape is to be rewound after it has been dumped.

**LEAVE:* The tape is to be left wherever it is positioned when the dump is completed. If an error is encountered during the dump, *LEAVE is ignored and the tape is rewound when the dump is completed or before the next tape operation starts.

**UNLOAD:* The tape is to be rewound and unloaded after it has been dumped.

Example

```
DMPTAP  DEV(QTAPE2) SEQNBR(5) TYPE(*DTABLK) DTABLK(3 7)
```

This command will dump information from the tape volume mounted on device QTAPE2. Data blocks 3 through 7 within the data file specified by sequence number 5 will be dumped to a print file.

DO (Do) Command

The Do (DO) command provides for grouping commands within a CL program; it is used with the ENDDO command to identify a group of commands that are to be executed together as a group. Typically, the DO command specifies the beginning of a group of commands that are to be executed as a result of a decision made by the execution of an IF command. (However, the DO command does not have to be associated with an IF command.) When used with an IF command, the DO command can be either the true part of the decision (that is, the value of the THEN parameter of the IF command), or the false part of a decision (on the ELSE command). Every do group must be terminated by the ENDDO command. Do groups can be nested within other do groups, but each group must have an ENDDO command to terminate its level of nesting.

Restrictions: This command is valid only within a CL program. A maximum of 10 levels of do groups can be nested within each other.

```
DO ────
                                                          Pgm:B,I
```

There are no parameters for this command.

Examples

```
DO
  •
  • (group of control language commands)
  •
ENDDO
```

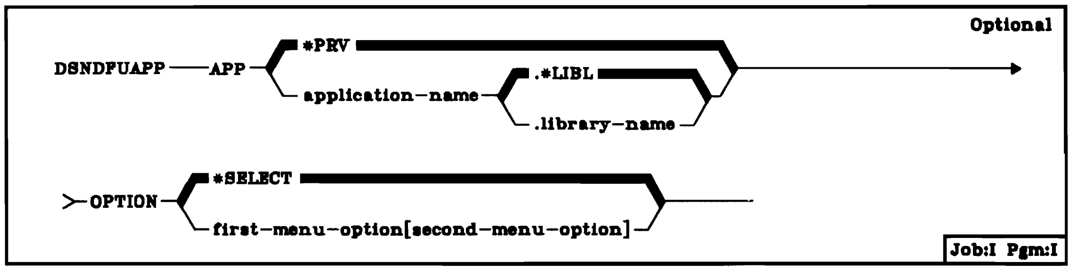
The commands between the DO and ENDDO commands are executed once, as a group of commands.

```
If &SWITCH DO
  •
  • (group of CL commands)
  •
ENDDO
```

The commands between the DO and ENDDO commands are executed if the value in the logical variable &SWITCH is '1'. If &SWITCH is not '1', then control passes immediately to the next command following the ENDDO command.

DSNDFUAPP (Design DFU Application) Command

The Design DFU Application (DSNDFUAPP) command begins the DFU prompting sequence for interactive definition and management of a DFU application. The Data File Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Data File Utility, refer to the *IBM System/38 DFU Reference Manual and User's Guide,* SC21-7714.



APP Parameter: Specifies the qualified name of the DFU application.

<u>*PRV:</u> The qualified name of the application used during your last DFU session is to be used.

application-name: Specify the name of the DFU application to be designed.

OPTION Parameter: Specifies the options you intend to use from the first two DFU displays (the second display to be presented is dependent upon your option selection from the first display). If you preselect the options on this parameter, DFU will skip the displays and will present the next logical display.

<u>*SELECT:</u> The first two DFU displays will appear in sequence with the options you use to define or manage a DFU application.

first-menu-option: Enter one of the three options from the DFU menu display. Possible options are:

1 – Create or change an application

2 – Execute an application

3 – Manage existing applications

[second-menu-option]: Enter one of the options from the selected second menu. Possible second menus and their options are as follows:

DFU Create/Change Menu (option 1)

1 – Display information about an application

2 – Create a new application

3 – Change an existing application

4 – Delete an existing application

DFU Execution Menu (option 2)

1 – Display information about an application

2 – Change data (add, delete, change, or verify records)

3 – Display data (display data base records)

DFU Management Menu (option 3)

1 – Display information about an application

2 – Rename or move an application

3 – Add or remove application users

4 – Change application owner

Example

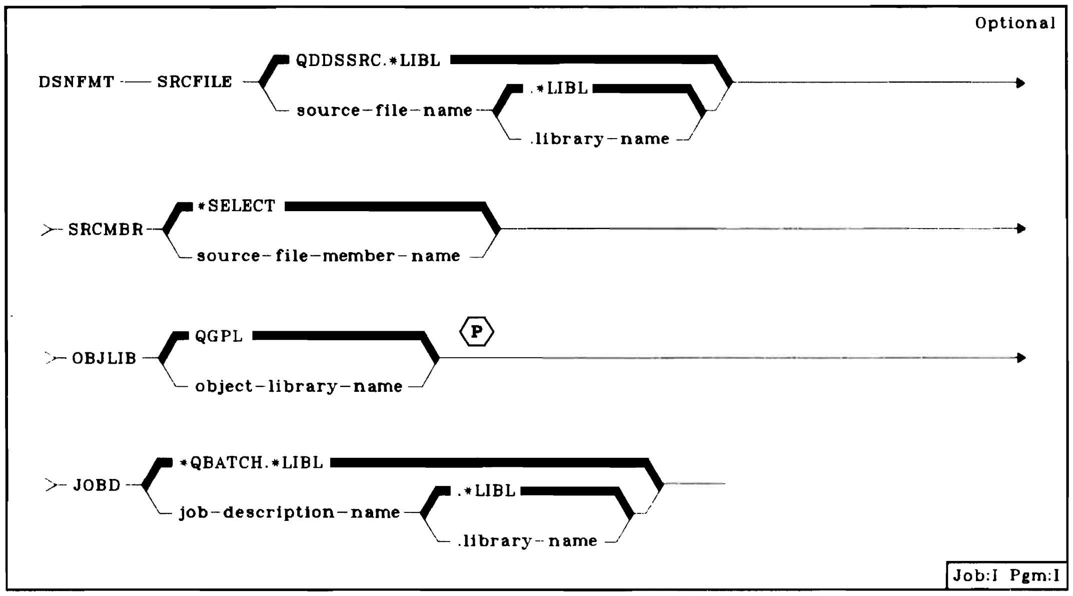
```
DSNDFUAPP  APP(DATA.LIB1)
```

This command calls the displays associated with the application named DATA in library LIB1.

DSNFMT (Design Format) Command

The Design Format (DSNFMT) command requests the SDA (Screen Design Aid) and displays the initial SDA display (SDA option menu).

The Screen Design Aid is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Screen Design Aid, refer to the *IBM System/38 SDA Reference Manual and User's Guide,* SC21-7755.



Note: All parameter values can be changed later during SDA execution.

SRCFILE Parameter: Specifies the qualified name of an existing source file that contains source file members to be updated or to which new source file members will be added.

QDDSSRC: The DDS source file QDDSSRC is assumed if the SRCFILE parameter is not specified. (If no library qualifier is specified, *LIBL is used to find the file.)

qualified-source-file-name: Enter the qualified name of an existing source file to be used by SDA. (If no library qualifier is specified, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of a new or existing source file member that contains or will contain DDS for the display formats or the control language for a menu to be updated or created by SDA.

*SELECT: If the SRCMBR parameter is not specified, a source file member name is not displayed on the design record format menu. The member list display is displayed next. The SRCMBR value can be entered on the member list display or when the design record format menu is redisplayed.

source-file-member-name: Enter the name of the source file member to be created or updated.

OBJLIB Parameter: Specifies the name of the library into which programs and display device files that are created by SDA will be stored.

QGPL: If the OBJLIB parameter is not specified, objects created by SDA will be stored in library QGPL.

object-library-name: Enter the name of the library into which objects created by SDA are to be stored.

JOBD Parameter: Specifies the qualified name of the job description to be used with jobs being submitted to SDA.

QBATCH.*LIBL: If the JOBD parameter is not specified, the job description QBATCH is to be used with submitted jobs. (If no library qualifier is specified, *LIBL is used to find the job description.)

job-description-name: Enter the qualified name of the job description to be used with submitted jobs. (If no library qualifier is specified, *LIBL is used to find the job description.)

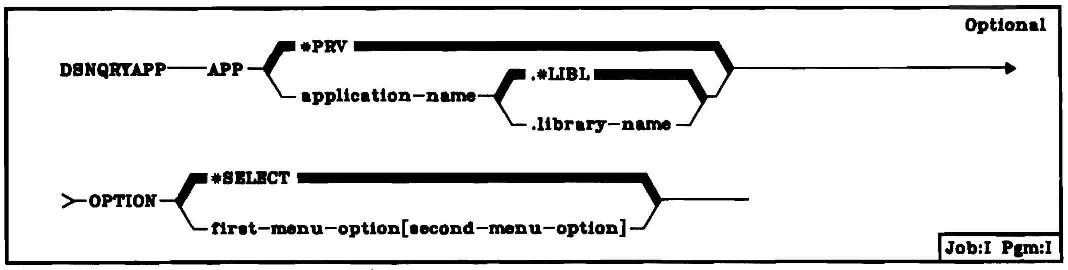
Example

```
DSNFMT
```

This command requests the initial SDA option menu. From this menu, you can select SDA functions to design display record formats, to design a menu, or to test a record format. With this command execution, all parameter defaults are taken. Once the source statements for a menu or a record format have been generated, you may wish to change one of the default parameter values. To change values, use the SAVE DDS/CREATE DISPLAY DEVICE FILE display. For more information, refer to the *Screen Design Aid Reference Manual and User's Guide*.

DSNQRYAPP (Design Query Application) Command

The Design Query Application (DSNQRYAPP) command begins the query prompting sequence for interactive definition and management of a query application. The Query Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Query Utility, refer to the *IBM System/38 Query Utility Reference Manual and User's Guide,* SC21-7724.



APP Parameter: Specifies the qualified name of the query application.

*PRV: The qualified name of the application used during your last query session is to be used.

application-name: Specify the name of the query application to be designed.

OPTION Parameter: Specifies the options you intend to use from the first two query displays. If you preselect the options on this parameter, Query will skip the displays and will present the next logical display.

*SELECT: The first two query displays will appear with the options you use to manage or define a query application.

first-menu-option: Enter one of the three options from the query menu. The resulting menu will be displayed next. Possible options are:

1 – Create or change a query

2 – Execute queries and display output

3 – Manage existing queries

[second-menu-option]: Enter one of the four options from the selected second menu. Possible second menus and their options are as follows:

Query Create/Change Menu (option 1)

1 – Display information about a query

2 – Create a new query

3 – Change an existing query

4 – Delete an existing query

Query Execution and Report Menu (option 2)

1 – Display information about a query

2 – Submit a query for execution

3 – Display status of queries submitted for execution

4 – Display output at work station from last execution

Query Management Menu (option 3)

1 – Display information about a query

2 – Rename or move a query

3 – Add or remove query users

4 – Change query owner

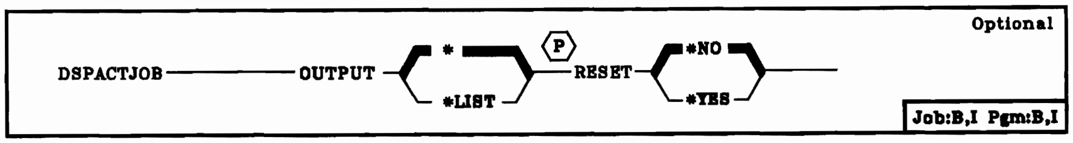
Example

```
DSNQRYAPP APP(QDATA.LIB1)
```

This command calls the displays associated with the query named QDATA in library LIB1.

DSPACTJOB (Display Active Jobs) Command

The Display Active Jobs (DSPACTJOB) command displays performance and status information for the active jobs in the system.



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the file(s) used by this command.)

*: The output is to be displayed (if requested by an interactive job) or listed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

RESET Parameter: Specifies whether the active job statistics are to be reset.

*NO: The active job statistics are not to be reset. The measurement time interval is extended (similar to pressing CF5) if a previous display active jobs command has executed in the current job. All active jobs are displayed.

**YES:* The active job statistics are to be reset. A measurement time interval of zero is used (similar to pressing CF7). All active jobs are displayed.

Example

```
DSPACTJOB  OUTPUT(*LIST)
```

This command directs the active job information to the job's spooled output on a printer. If OUTPUT(*) is specified instead and the command was entered from a work station, the information about the active jobs is displayed at the work station.

Additional Considerations

The display produced by the DSPACTJOB command has the following format when initially presented on the work station:

```
XX/XX/XX XX:XX:XX          ACTIVE JOBS DISPLAY          CPU: XX.X%
Elapsed: XX:XX:XX                 -------ELAPSED-------   Active jobs: XXXX
  SBS/JOB       TYP PL PTY    CPU  INT   RSP AUXIO   CPU  FUNCTION      STS
_ XXXXXXXXXX    XXX XX XX XXXXX.X  XXX XXX.X XXXXX XX.X%  X-XXXXXXXXX   XXXXX
_   XXXXXXXXXX XXX XX XX XXXXX.X  XXX XXX.X XXXXX XX.X%  X-XXXXXXXXX   XXXXX
      .
      .
      .

 1-DSPJOB   2-Spl files  3-HLDJOB       5-Inv stack   6-RLSJOB   7-Locks
 8-Exclude  9-CNLJOB     CF5-Redisplay  CF6-Restart   CF7-Reset  CF8-DSPSYSSTS
```

The first line of the active jobs display gives the current system date and time and the CPU utilization during the elapsed time.

The current system date and time is the time either when the display is first presented (after the DSPACTJOB command is entered) or when it is redisplayed with updated statistics (after CF5, CF6, or CF7 key is pressed).

The percent of CPU time used by the system during the elapsed time (CPU) compares the total amount of CPU time used during the elapsed time to the elapsed time. This field is normally higher than the sum of the CPU percentages used by the active jobs displayed because it includes CPU used by system overhead, excluded jobs, and jobs that have ended during the measurement time interval. This field is zero when the elapsed time is zero.

The second line gives the elapsed time and the number of active jobs.

The elapsed time (Elapsed) is the amount of time that has elapsed between the measurement start time and the current system time. This field is presented in hours, minutes, and seconds. This field is zero when the display is initially requested or when the display is reset (CF7 is pressed). A reset is forced if the elapsed time is negative (the system date/time has been set back) or is greater than approximately 100 hours.

The number of active jobs (Active jobs) is the current number of jobs active in the system, including both user and system jobs, to be printed or displayed (if no jobs have been excluded from the display).

The remainder of the display gives a list of all jobs that are currently active in the system. All information is gathered on a job basis (as opposed to routing step). The jobs are ordered on the basis of the subsystem they are executing in. Jobs that execute in a subsystem (autostart jobs, interactive jobs, batch jobs, readers, and writers) are alphabetized by job name and indented under the subsystem monitor job field they are associated with. Subsystem monitors (with the jobs in the subsystem grouped under each monitor job) are alphabetized and presented before system (SYS) jobs. The system jobs (start CPF, system arbiter) are alphabetized by job name and presented after the subsystem monitors and jobs within the subsystems. For each active job in the system the following information is displayed:

- Simple job name (SBS/JOB): The simple job name of the active job. Jobs that execute in a subsystem (autostart jobs, interactive jobs, batch jobs, readers, and writers) are indented two positions under the subsystem monitor job field they are associated with. The indentation shows the jobs that are 'contained' in a subsystem. Subsystem monitors and system jobs are not indented.

- Job type (TYP): The type of the active job. The identifiers for job types are as follows:
  - ASJ (autostart)
  - BCH (batch)
  - INT (interactive)
  - RDR (reader)
  - SBS (subsystem monitor)
  - SYS (system)
  - WTR (writer)

- System pool identifier (PL): The system-related pool identifier that the job's main storage is allocated from. These identifiers are not the same as those specified in the subsystem description, but are the same as the system pool identifiers shown on the system status display.

- Execution priority (PTY): The execution priority of the job. System jobs (subsystem monitors, system arbiter, start CPF) with an execution priority higher than priorities allowed for user jobs will display a priority of 0 (a lower number indicates a higher priority).

- Total CPU time used (CPU): The total CPU time used by the job expressed in seconds.

- Elapsed number of interactions (INT): The number of operator interactions (enter or CF key pressed) during the measurement time interval. This field will be blank for jobs that have no interactions (job types other than interactive) and the console job.

- Average response time (RSP): The average system response time over the measurement time interval, expressed in seconds. The transmission line time is not included. This field will be blank for jobs that have no interactions (job types other than interactive) and the console job.

- Elapsed number of auxiliary storage I/O operations (AUXIO): The number of auxiliary storage read and write operations the job has made during the measurement time interval. This includes both data base and non-data base paging.

- Percent CPU used (CPU): The percent of CPU time attributed to this job over the elapsed time compared to the measurement time interval.

- Function (FUNCTION): The high level function being performed by the job. This field will be blank when a logged function has not been performed. The first character of this field indicates what the characters that follow the hyphen represent:

  C – command. The command name will be a command executed interactively, in a batch job stream, or requested from a system menu (QCALLMENU will not log the functions it performs). Commands in CL programs will not be logged.

  P – program. The program name will be the high level program called interactively, a program called in a batch job stream, the initial program specified in the user profile, or the name of a system request processor (QMNSYSRQ, QOPRMENU, QPGMMENU, or QCALLMENU). If the high level program does a transfer control, it will remain in the function field even though it is no longer in the program stack.

  L – message queue. The message queue being produced or copied to a data base file. The previously logged value is replaced when the logging is finished.

    QHST – QHST is being logged to a DB file.

    QSRV – QSRV is being logged to a DB file.

    QCHG – QCHG is being logged to a DB file.

* – special value (previous log value replaced on completion).

JOBLOG – joblog is being produced.

DUMP – a dump is in progress.

ADLACTJOB – auxiliary storage is being allocated for the number of active jobs specified in the QADLACTJ system value. This may indicate that the system value for the initial number of active jobs was set too low. (See Chapter 19 of the *Programmer's Guide*.)

ADLTOTJOB – auxiliary storage is being allocated for the number of jobs specified in the QADLTOTJ system value. (See Chapter 19 of the *Programmer's Guide*.)

CMDENT – the command entry display is being used.

- Status (STS): The status of the job. Only one status is displayed per job. A blank status represents a job that is in transition. If the hold job, release job, or cancel job functions are executed against a job through the active jobs display, the job is identified with *HLD, *RLS, or *CNL in this field. Possible status values listed in order of precedence are:
  - CNL. The job has been canceled with the *IMMED option or delay time has expired with the *CNTRLD option.
  - HLD. The job is held.
  - SRQ. The job is the inactive half of a system request job pair.
  - LCKW. The job is waiting for a lock.
  - EVTW. The job is waiting on an event.
  - DEQW. The job is waiting on a dequeue operation.
  - DEQA. The job is waiting on a dequeue operation in the pool activity level.
  - EXC. The job is currently executing in the pool activity level.
  - INEL. The job is ineligible and not currently in the pool activity level.

You can update the statistics on the Active Jobs Display by pressing the CF5 key. This causes the previous start time to continue to be used as the start time for the new measurement interval. Jobs that have been excluded will remain excluded. (This is similar to the command being entered again with RESET(*NO) specified.)

You can restart the display by pressing the CF6 key. This causes the start time for the new measurement interval to be set to the previous display time (shown on line 1 of display). The measurement time interval is the amount of time that has elapsed between the time the previous display was presented and the time CF6 was pressed. Jobs that have been excluded will remain excluded.

You can reset the display by pressing the CF7 key. This causes the start time to be set to the current time. The measurement time interval will be zero and elapsed fields will contain zero, and excluded jobs will be added. The beginning of the list of jobs will be displayed (this has the same effect as reentering the command with RESET(*YES) specified.)

If more active jobs exist than will fit on one display, a single plus sign (+) appears to the right of the last job displayed. The roll keys can be used to view the additional jobs. When CF5 or CF6 is pressed an attempt is made to show the same set of jobs that had previously been displayed. The job that was previously at the top of the display will be in the new set of jobs displayed (if it is still active). If the job that was at the top of the display has finished, the job that would appear after it in the presentation order will be in the new set of jobs displayed. The beginning of the list of active jobs is displayed if the job that was at the top of the display has finished and no job would have appeared after it in the presentation order.

An input field (to the left of each job name) can be used to enter any one of the numbers shown at the bottom of the display. When the enter key is pressed, the function associated with the entered number is performed for that job. If numbers are placed in the input fields preceding several jobs before the enter key is pressed, the specified functions are performed (one at a time) on the jobs in the order in which the jobs are shown on the display. The system executes each command using the default values of all of its parameters. The following functions can be specified:

1–DSPJOB:	The display job menu is presented from which several displays can be selected to show the job's definition and execution attributes, the job's status, and the job's spooled output files. When this option is selected for a spooling reader or spooling writer job, the DSPRDR or DSPWTR display is presented. This option is not valid for system or subsystem monitor jobs. Job control special authority is required to display a job with a user name different than the job requesting the display.
2–Spl files:	The job's spooled output files are displayed. This option is valid for all jobs. Job control special authority is required to display spooled files of a job with a user name different than the job requesting the display.

4–HLDJOB:	The job is held, but its spooled files are not held. The HLDRDR or HLDWTR (with OPTION(*IMMED)) command is executed if this option is selected for a spooling reader or spooling writer job. This option is not valid for system or subsystem monitor jobs. Job control special authority is required to to hold a job with a user name different than the job requesting the display. *HLD replaces the status field if the command was successfully executed.
5–Inv stack:	The job's program invocation stack is displayed. This option is valid for all jobs. Job control special authority is required to display the program invocation stack of a job with a user name different than the job requesting the display.
6–RLSJOB:	The job, which must be in the held state, is released. The RLSRDR or RLSWTR command (with OPTION(*CURRENT)) is executed if this option is selected for a spooling reader or spooling writer job. This option is not valid for system or subsystem monitor jobs. Job control special authority is required to release a job with a user name different than the job requesting the display. *RLS replaces the status field if the command was successfully executed.
7–Locks:	The job's locks are displayed. (Data base record locks and some types of internal lock functions are not displayed.) This option is valid for all jobs. Job control special authority is required to display the locks for a job with a user name different than the job requesting the display.
8–Exclude:	The job is excluded from the display. This option has no effect on the job, only the display. Pressing CF7 will reset the display and all active jobs will be displayed.
9–CNLJOB:	The job is canceled, but the spooled files produced by the job are not canceled. A controlled cancel is performed as if the CNLJOB command were entered with all the default parameter values assumed. The CNLRDR or CNLWTR command (with OPTION(*CNTRLD)) is executed if this option is selected for a spooling reader or spooling writer job. This option is not valid for system or subsystem monitor jobs. Job control special authority is required to cancel a job with a user name different from that of the job requesting the display. *CNL replaces the status field if the command was successfully executed.

When all of the commands have been executed, the active jobs display is reshown with no updated information. The same set of jobs will be shown unless an error occurred during command processing; in that case, the first job with an error is shown. Any error or completion messages are shown at the bottom of the display. An indication of successful non-display commands will be placed in the status field of the job the command was entered against (*HLD, *RLS, *CNL).

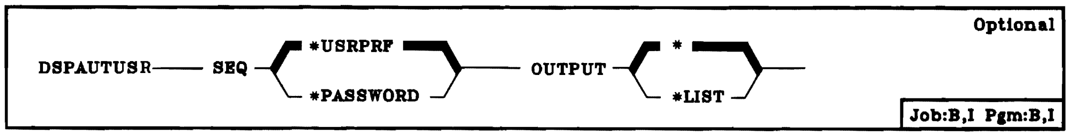
If there are more jobs than can be shown on a single display, the roll keys can be used and options can be placed in the input fields on multiple displays before the enter key is pressed.

The CF1 key can be used to exit from the display shown above, or to exit from a display presented as a result of executing the commands entered on the display. The CF1 key prevents options requested for jobs (following the job currently being displayed) from being executed.

DSPAUTUSR (Display Authorized Users) Command

The Display Authorized Users (DSPAUTUSR) command displays or prints the names of the authorized system users and their passwords in alphabetic sequence.

Restriction: Only the security officer can use this command.



SEQ Parameter: Specifies that the list of system users is to be in alphabetic sequence either by user name or by user password.

<u>*USRPRF:</u> The list is to be in alphabetic sequence by user profile name.

**PASSWORD:* The list is to be in alphabetic sequence by password.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

```
DSPAUTUSR
```

This command causes the list of authorized users and their passwords to be displayed or printed. The list will be in alphabetic sequence by user profile name because SEQ(*USRPRF) is assumed. Because OUTPUT(*) is also assumed, the list will be displayed or printed depending on whether the command was submitted at a work station or as part of the batch input stream.

```
DSPAUTUSR SEQ(*PASSWORD) OUTPUT(*LIST)
```

This command causes the user profile names and the associated passwords of the authorized users of the system to be printed. The listing is to be printed in alphabetic sequence by the password.

Additional Considerations

The display produced by the DSPAUTUSR command has the following format:

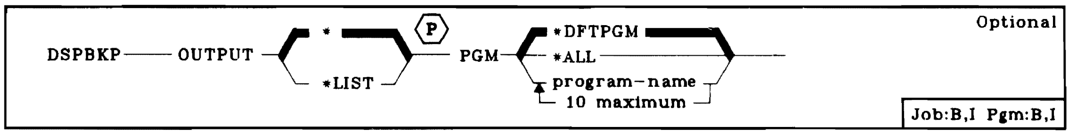
```
 XX/XX/XX          AUTHORIZED USERS DISPLAY
USER NAME    PASSWORD            USER NAME    PASSWORD
XXXXXXXXXX   XXXXXXXXXX          XXXXXXXXXX   XXXXXXXXXX
XXXXXXXXXX   XXXXXXXXXX          XXXXXXXXXX   XXXXXXXXXX
   .                                .
   .                                .
   .                                .
```

Regardless of whether the display shows the information in alphabetic order by user profile name or by password, the format of the display does not change. That is, the password always appears to the right of the user profile name.

DSPBKP (Display Breakpoints) Command

The Display Breakpoints (DSPBKP) command displays the locations of all the breakpoints currently set in the specified programs that are in debug mode. The breakpoints and the names of the program variables associated with each breakpoint are displayed.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

PGM Parameter: Specifies which programs in the debugging environment are to have their breakpoint locations and associated program variables displayed.

*DFTPGM: Only the default program is to have its breakpoint locations displayed.

**ALL:* All the programs currently in debug mode are to have their breakpoint locations displayed.

program-name: Enter the simple names of one or more programs that are to have their breakpoint locations displayed. The programs specified must already be in debug mode.

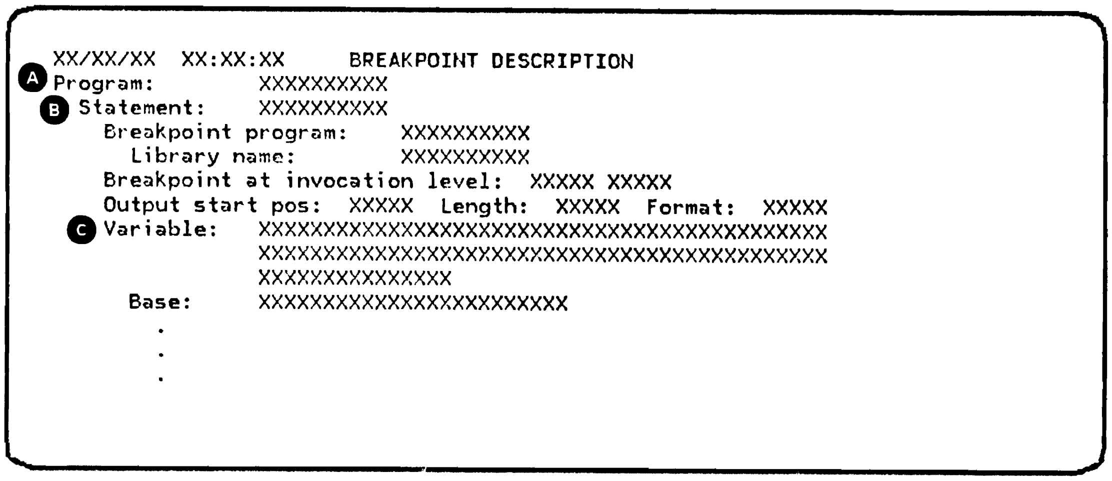
Example

```
DSPBKP
```

Assuming that MYPROG is the default program in an interactive debugging environment, this command causes the work station to display all of the breakpoint locations that are currently set in MYPROG. The names of all the program variables associated with each breakpoint are also listed.

Additional Considerations

The DSPBKP command produces the following display, which shows the *description* of all the breakpoints set in the program(s) specified on the PGM parameter.



```
  XX/XX/XX  XX:XX:XX     BREAKPOINT DESCRIPTION
Ⓐ Program:          XXXXXXXXXX
 Ⓑ Statement:       XXXXXXXXXX
    Breakpoint program:    XXXXXXXXXX
      Library name:        XXXXXXXXXX
    Breakpoint at invocation level:  XXXXX XXXXX
    Output start pos:  XXXXX  Length:  XXXXX  Format:  XXXXX
  Ⓒ Variable:       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                    XXXXXXXXXXXXXXX
      Base:         XXXXXXXXXXXXXXXXXXXXXXXX
        .
        .
        .
```

The information displayed identifies all the statements on which breakpoints are currently set in the programs and, for each breakpoint, the names of all the variables whose values are to be displayed when the breakpoint is reached. (Each statement in each program can have different groups of variables displayed.)

If more than one program is specified on the DSPBKP command, the descriptions of all the statements defined as breakpoints in the first program are displayed before those set in the next program are displayed. That is, for the program named at Ⓐ, all of its breakpoints identified at Ⓑ and all of its variables named at Ⓒ are shown before another program name is shown.

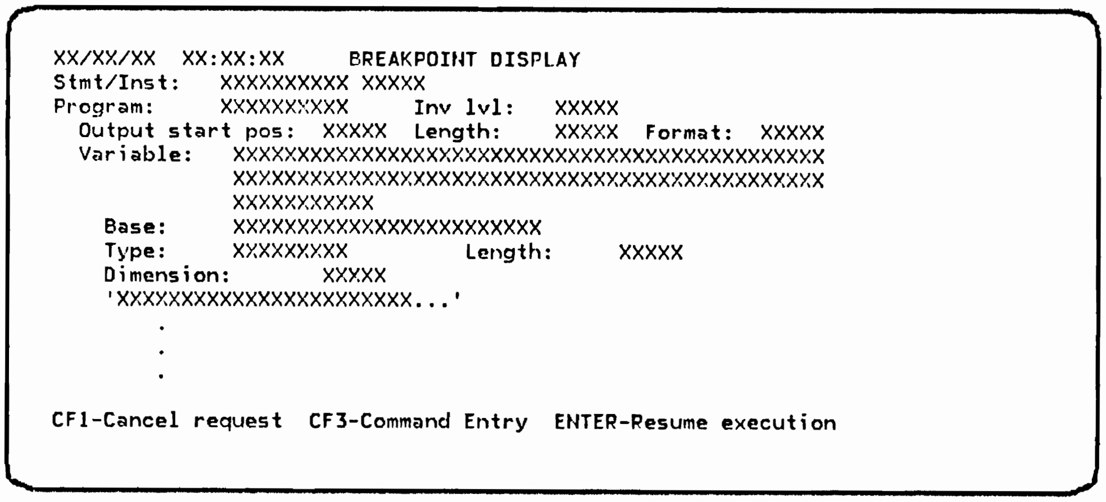
Beginning at C, as many as 10 repetitions of the lines identifying program variables can be displayed for each statement identified at B. Beginning at B, for each statement on which a breakpoint is set in the program identified at A, a set of lines describing the breakpoint is displayed. And as many as 10 programs can have their breakpoint information displayed successively after one DSPBKP command is entered.

For each program, the following information can be displayed:

- The name of the program (at Ⓐ) whose breakpoints are displayed on the following lines.
- The label name, the System/38 instruction number, or the statement number of the statement (at Ⓑ) at which a breakpoint is set.
- If the program named at A is executed in a batch job and has a breakpoint program specified, the *breakpoint program* field gives the name of the program to which control is transferred when the breakpoint is reached. The name of the library containing the breakpoint program is given on the next line. (If the program shown at A has no breakpoint program specified, these two lines are not shown.)

- If the DSPBKP command is entered interactively after the job has stopped execution, the *breakpoint at invocation level* field shows the invocation level(s) of the program (identified at A) in which job execution has stopped.
- The start, display length, and output format fields contain the common information about the variable(s) identified on the following lines (beginning at C) that are to be displayed when program execution stops at this statement (identified at B). The values displayed in these three fields are those that were specified in the START, LEN, and OUTFMT parameters of the ADDBKP command.

The following display is *not* produced by the DSPBKP command; it occurs when a breakpoint is reached in a program being executed and debugging is being done interactively.

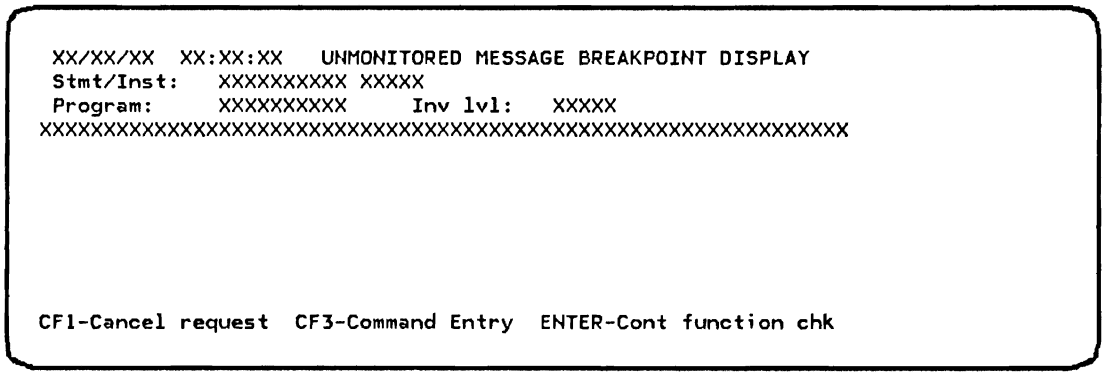


```
XX/XX/XX  XX:XX:XX     BREAKPOINT DISPLAY
Stmt/Inst:   XXXXXXXXXX XXXXX
Program:     XXXXXXXXXX     Inv lvl:   XXXXX
  Output start pos:  XXXXX  Length:    XXXXX  Format:  XXXXX
  Variable:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              XXXXXXXXXXX
    Base:     XXXXXXXXXXXXXXXXXXXXXXXX
    Type:     XXXXXXXXX          Length:     XXXXX
    Dimension:       XXXXX
    'XXXXXXXXXXXXXXXXXXXXXXX...'
         .
         .
         .

CF1-Cancel request  CF3-Command Entry  ENTER-Resume execution
```

Line 2 identifies the breakpoint at which program execution has stopped, giving the statement label or statement number and the machine instruction number of the breakpoint statement. Line 3 gives the name and invocation level number of the program containing the breakpoint statement. The other lines on the display identify all the program variables and gives their current values (that were requested on the associated ADDBKP command) when the breakpoint was reached. For a complete description of the displayed variable information, refer to *Additional Considerations* in the DSPPGMVAR command description.

When a breakpoint is reached, the Enter key can be used to resume program execution, or the CF3 key can be used to display the command entry display. Any CL command can then be entered at that breakpoint.

If an unmonitored escape message occurs during program execution in interactive debug mode, program execution stops and the following display of the escape message is presented. The display gives the user a chance to take action on the unmonitored error condition instead of letting the programs associated with the requested function terminate in a function check because of the error.



```
 XX/XX/XX  XX:XX:XX   UNMONITORED MESSAGE BREAKPOINT DISPLAY
 Stmt/Inst:    XXXXXXXXXX XXXXX
 Program:      XXXXXXXXXX     Inv lvl:    XXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX






CF1-Cancel request  CF3-Command Entry  ENTER-Cont function chk
```

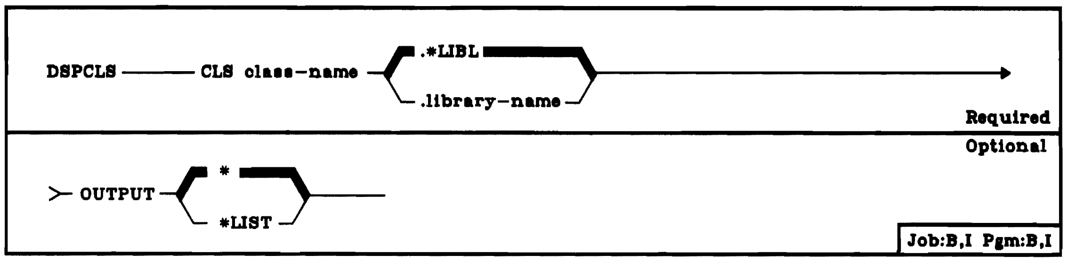
Again, line 2 identifies the statement/machine instruction number causing the error, and line 3 indicates the program and invocation level of the program in which the error occurred. The error message itself (first-level text only) is shown on the following lines. The second-level text can be displayed if you move the cursor to the message line and press the Help key.

- The Enter key can be used to allow program execution to continue, which will result in a function check. Also, if the displayed message indicates that data can be dumped for the displayed error, a system dump of the job and/or message data fields is taken. (For details, see the DMPLST parameter in the ADDMSGD command description.)
- The CF3 key can be used to get the command entry display, and then enter any CL command (except ENDDBG or RSMBKP) to gather more information about the error condition. For example, you can enter DSPDBG, DSPPGMVAR, or DSPTRCDTA to view current debug status, or you can enter DMPJOB to get a dump of the job. (To allow the function check to continue after the command entry display has been shown, you must press the CF1 key and then the Enter key.)
- The CF1 key can be used to cancel the last request you entered (that caused the unmonitored escape message). This is equivalent to entering CNLRQS RQSLVL(*PRV) on the command entry display.

DSPCLS (Display Class) Command

The Display Class (DSPCLS) command displays the attributes of a class.

Restriction: You must have operational rights for the class before you can display its attributes.



CLS Parameter: Specifies the qualified name of the class that is to have its attributes displayed. If no library qualifier is given, *LIBL is used to find the class description. (For an expanded description of the CLS parameter, see Appendix A.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

_*:_ The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPCLS  CLS(CLASS1) OUTPUT(*LIST)
```

This command directs the attributes of class CLASS1 to the job's output spooling queue to be printed.

Additional Considerations

The display produced by the DSPCLS command has the following format:

```
 XX/XX/XX  XX:XX:XX          CLASS DISPLAY

Class name:                   XXXXXXXXXX
  Library name:               XXXXXXXXXX
Execution priority:           XX
Time slice in millisec:       XXXXXXX
Eligible for purge:           XXXX
Default wait time in sec:     XXXXXXX
Max CPU time in millisec:     XXXXXXX
Max temp storage in K-bytes:  XXXXXXX
Text: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

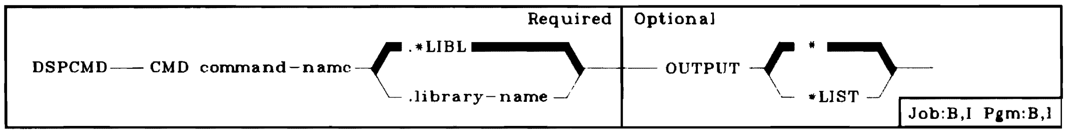
Line 1 of the class display shows the current date and time for the job. Lines 3 and 4 identify the class being displayed and the remaining lines show the attributes of the class.

For an explanation of each class attribute shown, refer to the associated parameter description given in the CRTCLS command description; for example, the execution priority attribute is explained in the EXCPTY parameter.

To exit from this display and return to the working display, such as the command entry display, programmer menu, operator menu, and so forth, press the CF1 key. Press the CF2 key to return to the calling display.

DSPCMD (Display Command) Command

The Display Command (DSPCMD) command displays a subset of the values that were specified for parameters in the Create Command (CRTCMD) command.



CMD Parameter: Specifies the qualified name of the user-defined or IBM-supplied command. (If no library qualifier is specified, *LIBL is used to find the command.)

OUTPUT Parameter: Specifies whether the output is to be directed to the work station screen or to a printer.

<u>*</u>: The command attributes are to be displayed at the work station. If the command is executing in batch mode, the attributes are to be printed on a printer.

**LIST:* The command attributes are to be printed on a printer.

Example

```
DSPCMD  CMD(PAYROLL)
```

This command displays at the work station all current user-assigned parameter values for the user-defined command PAYROLL.

Additional Considerations

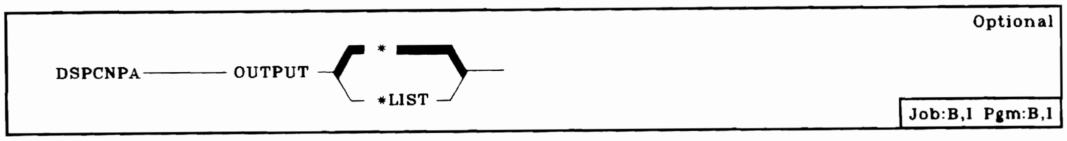
The display produced by the DSPCMD command has the following format:

```
XX/XX/XX XX:XX:XX              COMMAND DISPLAY
Command name:  XXXXXXXXXX    Library:  XXXXXXXXXX
  Program to execute command:    PGM        XXXXXXXXXX
    Library name:                             XXXXXXXXXX
  Source file name:              SRCFILE    XXXXXXXXXX
    Library name:                             XXXXXXXXXX
  Source member name:            SRCMBR     XXXXXXXXXX
  Validity checking program:     VLDCKR     XXXXXXXXXX
    Library name:                             XXXXXXXXXX
  Mode in which valid:           MODE       XXXXXXXXXXXXXX
                                            XXXXXXXXXXXXXX
  Where allowed to execute:      ALLOW      XXXXXXXXXXXXXX
                                            XXXXXXXXXXXXXX
  Max positional parameters:     MAXPOS     XXXXXXXXXX
  Message file for prompt text:  PMTFILE    XXXXXXXXXX
    Library name:                             XXXXXXXXXX
  Message file name:             MSGF       XXXXXXXXXX
    Library name:                             XXXXXXXXXX
  Text description:              TEXT       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXXXXXXXX
```

All attributes but the PUBAUT parameter are displayed, in the same order as the parameters appear in the CRTCMD command. For an explanation of the parameters and values, refer to the CRTCMD command description.

DSPCNPA (Display CSNAP Attributes) Command

The Display CSNAP Attributes (DSPCNAP) command is used to display the CSNAP (communications statistical network analysis procedure) short-term statistics attributes that are currently set in the system.



OUTPUT Parameter: Specifies whether the attributes are to be displayed at the work station or listed on the printer.

*: The CSNAP short-term statistics attributes are to be displayed at the work station. When a batch job uses the * default, the attributes are listed on the printer.

**LIST:* The CSNAP short-term statistics attributes are to be listed on the printer.

Examples

```
DSPCNPA  OUTPUT(*)
```

This command displays the CSNAP short-term statistic attributes on the device at which the command was entered.

```
DSPCNPA  OUTPUT(*LIST)
```

This command lists the CSNAP short-term statistics attributes on the printer.

Additional Considerations

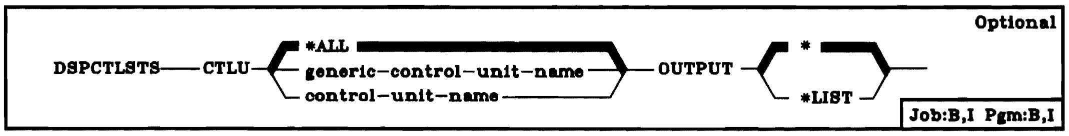
The DSPCNPA command produces the following display:

```
 XX/XX/XX XX:XX:XX      CSNAP (CNP) ATTRIBUTES
Line names:                      LINE       XXXXXXXXXX
  XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX
  XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX
Period for recording:            PERIOD
  Start time/date:                          XX:XX:XX  XX/XX/XX
  End time/date:                            XX:XX:XX  XX/XX/XX
Sampling interval in hours:      INTERVAL   XX.XX
```

The display shows the current CSNAP attributes, as they were originally set in the system or the values they were changed to with the Change Communication Network Program Attributes (CHGCNPA) command.

DSPCTLSTS (Display Control Unit Status) Command

The Display Control Unit Status (DSPCTLSTS) command displays the configuration of specified control units on a system, with their attached devices. If a single control unit is named on a DSPCTLSTS command, the line to which it is attached (if applicable) is also displayed. The status of each specified control unit is also displayed, with the job names of all interactive, batch, autostart, reader, or writer jobs that are holding a lock on a device.



CTLU Parameter: Specifies whether the status of all control units and attached devices on the system is to be displayed, or only the status information for a specific control unit and its attached devices.

*ALL: The status information for all control units and attached devices on the system is to be displayed. No line name information for these control units will be directly displayed.

generic-control-unit-name:* The status information for this control unit and any attached devices is to be displayed, or the status information for all control units with the same generic name is to be displayed. To specify a generic name, add an asterisk after the last character in the generic name (ABC*, for example). If an asterisk is not included with the name, the system assumes that the name is a complete control unit name.

control-unit-name: The status information for this control unit and attached devices is to be displayed. Line name information will also be displayed, if applicable.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled printer output. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the file(s) used by this command.)

*: The output is to be displayed (if requested by an interactive job) or listed with the job's system output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled printer output.

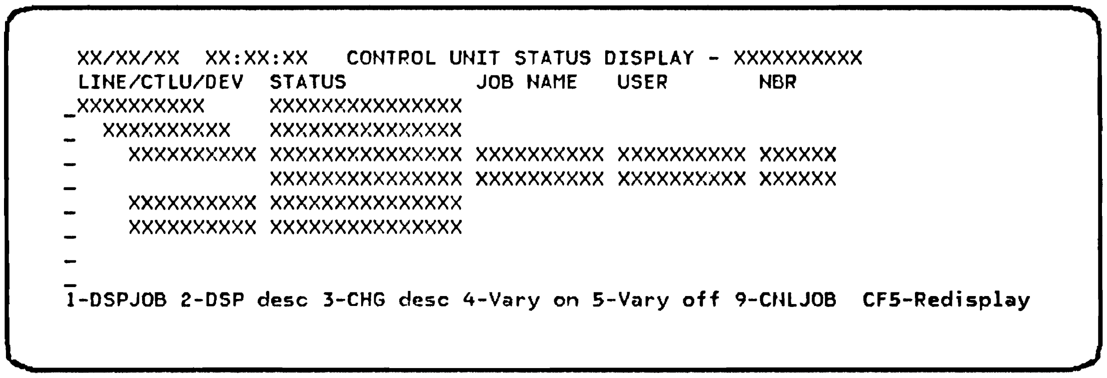
Example

```
DSPCTLSTS  CTLU(NDA01)
```

This command displays the name and status of control unit NDA01 and the name and status of any attached devices. If device NDA01 is attached to a line, the name and status of the line are displayed. In addition, the names of jobs using that device will be displayed. The information is displayed on the work station from which the command was submitted or it is spooled to a printer output queue to be printed on the system printer, if the command was part of the batch input stream.

Additional Considerations

The displays produced by the DSPCTLSTS command will show various amounts of control unit configuration information, depending on what you specify for the CTLU parameter. As an example, if you specify DSPCTLSTS CTLU control-unit-name (where the control unit is attached to a line), the following display is shown:



```
 XX/XX/XX  XX:XX:XX   CONTROL UNIT STATUS DISPLAY - XXXXXXXXXX
  LINE/CTLU/DEV  STATUS          JOB NAME   USER       NBR
_XXXXXXXXXX     XXXXXXXXXXXXXXX
_  XXXXXXXXXX   XXXXXXXXXXXXXXX
_    XXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
_               XXXXXXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
_    XXXXXXXXXX XXXXXXXXXXXXXXX
_    XXXXXXXXXX XXXXXXXXXXXXXXX
_
_
1-DSPJOB 2-DSP desc 3-CHG desc 4-Vary on 5-Vary off 9-CNLJOB  CF5-Redisplay
```

Header Information

The first line of the display shows the current job date and time fields followed by the value specified for the CTLU parameter (either '*ALL', a generic control unit name, or a specific control unit name).

First-column Input Field

The leftmost column of the display consists of a single-digit input field in which a number can be entered. You can enter the number (any one of those shown at the bottom of the display) in the input field to cause the function (a command) associated with that number to be performed for that line/control-unit/device when the you press the Enter key. If you enter numbers in the input fields of several items before pressing the Enter key, the specified functions are performed on the items in the order in which the items are displayed. The following functions can be specified:

1–DSPJOB: Executes the DSPRDR command for a reader job, executes the DSPWTR command for a writer job, executes the DSPJOB command for all other jobs, or returns 'Job .. not found' if no job is associated with the input record.

2–DSP desc: Executes the DSPLIND command for a line, executes the DSPCUD command for a control unit, or executes the DSPDEVD command for a device.

3–CHG desc: Prompts for the CHGLIND command for a line, prompts for the CHGCUD command for a control unit, or prompts for the CHGDEVD command for a device.

4–Vary on:
For a line, 1) the line is varied on,
2) all attached control units are varied on, or
3) all attached devices are varied on.
For a control unit, 1) the control unit is varied on, or
2) all attached devices are varied on.
For a device, the device is varied on.

5–Vary off:
For a line, 1) all attached devices are varied off, if possible,
2) all attached control units are varied off, if possible, or
3) the line is varied off, if possible.
For a control unit, 1) all attached devices are varied off, if possible,
2) the control unit is varied off, if possible.
For a device, the device is varied off.

Note: As is the case when the VRYxxx commands are entered individually, there is a noticeable delay when varying off a line/control-unit/device that has already been varied off.

9–CNLJOB: Executes the CNLRDR command (with OPTION(*CNTRLD)) for a reader job, executes the CNLWTR command (with OPTION(*CNTRLD)) for a writer job, executes the CNLJOB command (with OPTION(*CNTRLD)) for all other types of jobs, and returns 'Job .. not found' if no job is associated with the input record.

When all of the commands have been executed, the display is shown again with the status fields of the objects updated and with any error messages that occurred when the commands were executed. The display can be shown at any time *before* all the commands have been executed by pressing the CF5 key.

If the configuration has more elements than can be shown on a single display, the Roll Up key can be used to display them all. You can enter numbers in the input fields on multiple displays before pressing the Enter key.

After the commands have been executed, if there are more error messages than can fit on that display, a '+' is shown at the end of the last message. You must position the cursor at the first message and use the Roll Up key to view all of the error messages.

Line/Ctlu/Dev

The second vertical column displays the name of the item whose information is being displayed on that line. Control unit (CTLU) names are indented two spaces and device (DEV) names are indented four spaces. The heading of this column varies with the value specified for the CTLU parameter. If you specify *ALL, or a generic control unit name, or the name of a control unit that is not attached to a line, the CTLU/DEV field values will be shown. The LINE/CTLU/DEV field values will be shown if you specify the name of a control unit that is attached to a line.

Status

The third column lists the status of the line/control-unit/device. One of the following values is used to indicate status:

- ACTIVE. The line, control unit, or device is currently in use. For a display device, the device is signed on or has been allocated by a batch, auto-start, or interactive job.
- ACTIVE/RDR. A spool reader is using this device.
- ACTIVE/WTR. A spool writer is using this device.
- CONNECT PENDING. A VRYLIN command has been issued for this line, and the system is waiting for an action to be completed, such as a switched connection to be made.
- DIAGNOSTIC MODE. The line, control unit, or device is being serviced or has otherwise been set to diagnostic mode.
- FAILED. The line, control unit, or device is in an unusable state; it can possibly be made usable again by varying it off, then on. A failed device may still be allocated to a job.
- FAILED/RDR. This device, which is in an unusable state, is still allocated to a spool reader job.

- FAILED/WTR. This device, which is in an unusable state, is still allocated to a spool writer job.
- POWERED OFF. The control unit or device is in a varied-off and powered-off state.
- SIGNON DISPLAY. This display device currently has the 'Enter password to signon' screen displayed.
- SYSREQ. This display device has been requested by the system, and the job associated with this status does not have a lock on the device. SYSREQ status coexists only with an ACTIVE or SIGNON DISPLAY status for this device.
- VARIED OFF. For a control unit or device that can be powered on or off by the PWRCTLU or PWRDEV command, the status is indicated after the control unit or device is powered on. For a line, this status indicates that the line is varied off.
- VARIED ON. The line, control unit, or device is varied online, although it may not be physically powered on.
- VARY ON PENDING. A VRYCTLU or VRYDEV command has been issued for this control unit or device, respectively, and the system is waiting for an action to be completed, such as a switched connection to be made.
- *DAMAGED. The line, control unit, or device has incurred hard or partial damage; it is not possible to obtain any further status information.
- *LOCKED. The line, control unit, or device is allocated to another job with an *EXCL lock, and its attributes can not be determined at this time.
- *UNKNOWN. All of the status bits for the line, control unit, or device have been checked, and none are set. This is an exceptional condition.

Job Name

The fourth column shows the names of the jobs that are currently using any of the named devices.

User

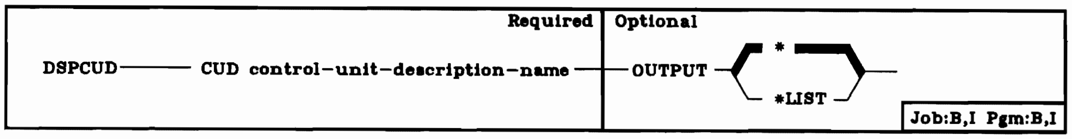
The fifth column shows the name of the user profile under which the job that holds the lock on the device is running.

Nbr

The rightmost column shows the six-digit number assigned by the system to identify the job.

DSPCUD (Display Control Unit Description) Command

The Display Control Unit Description (DSPCUD) command displays information about the specified control unit and its description.



CUD Parameter: Specifies the name of the control unit description to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

**:* The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

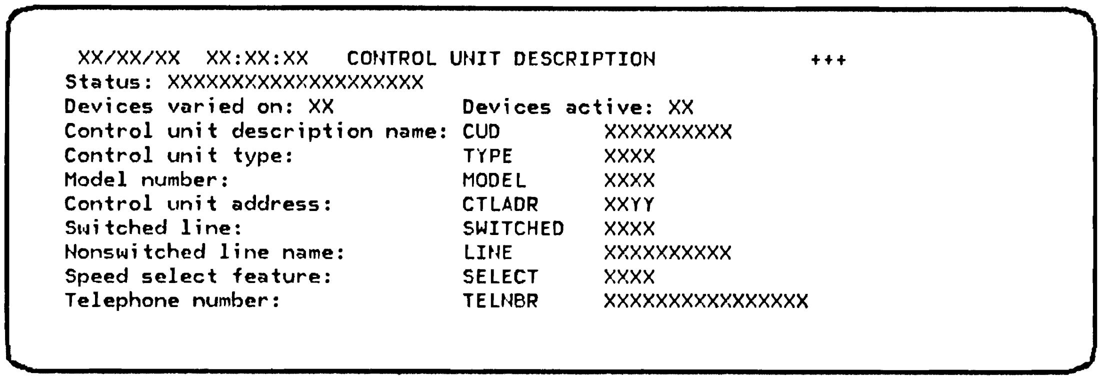
Example

```
DSPCUD  CUD(CONTROL01)
```

This command displays information about the control unit description named CONTROL01. The information is displayed on the work station from which the command was submitted; or it is spooled to a printer output queue to be printed on the system printer, if the command was part of the batch input stream.

Additional Considerations

Four displays are used to display the attributes of the control unit description identified in the DSPCUD command. The attributes are displayed in the same order as their associated parameters (whose keywords are shown here) occur in the CRTCUD command. For an explanation of each attribute of the control unit description, refer to the associated parameter description given in the CRTCUD command description.



```
 XX/XX/XX  XX:XX:XX   CONTROL UNIT DESCRIPTION           +++
Status: XXXXXXXXXXXXXXXXXXXX
Devices varied on: XX         Devices active: XX
Control unit description name: CUD        XXXXXXXXXX
Control unit type:             TYPE       XXXX
Model number:                  MODEL      XXXX
Control unit address:          CTLADR     XXYY
Switched line:                 SWITCHED   XXXX
Nonswitched line name:         LINE       XXXXXXXXXX
Speed select feature:          SELECT     XXXX
Telephone number:              TELNBR     XXXXXXXXXXXXXXXX
```

Lines 2 and 3 of the first display contain three fields not specified in the CRTCUD command, but whose information is kept current in the control unit's description. The fields are:

- *Status:* Indicates one of the following about the control unit:
  - ACTIVE. The control unit is varied on, and an attached device is currently in use.
  - DIAGNOSTIC MODE. The control unit is being serviced.
  - POWERED OFF. The control unit is in a varied-off and powered-off state.
  - VARIED OFF. The control unit is varied offline. If it can be powered on or off by the PWRCTLU command (such as the 3411 tape control unit), the status is indicated after the control unit is powered on.
  - VARIED ON. The control unit is varied online and is powered on.
  - VARY ON PENDING. The VRYCTLU command has been issued and the system is waiting for an action to be completed, such as a switched connection being made.
  - FAILED. The control unit is in an unusable state; it can possibly be usable again by varying it off, then on.

- *Devices varied on:* The number of devices attached to the control unit that are varied online.

- *Devices active:* The number of devices that are currently in use (including work station printers and signed-on displays).

The control unit address (specified as four digits in the CRTCUD command) is displayed in the format *xxyy*, where xx is the controller station address and yy is the operational unit (OU) number of the control unit.

The second, third, and fourth displays show the following:

```
 XX/XX/XX  XX:XX:XX   CONTROL UNIT DESCRIPTION            +++
Switched initial connection:   INLCNN     XXXXX
Exchange identifier:           EXCHID     XXXXXXXX
BSC local identifier:          LCLID
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
BSC remote identifier:         RMTID
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
SSCP identifier:               SSCPID     XXXXXXXXXXXX
Online at CPF start:           ONLINE     XXXX
```

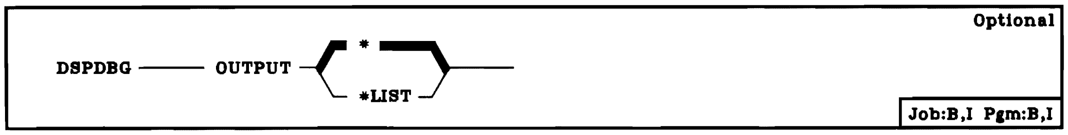
```
 XX/XX/XX  XX:XX:XX   CONTROL UNIT DESCRIPTION             +++
Authorized switched lines:     LINLST
  XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX
  XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX    XXXXXXXXXX
Current switched line:                    XXXXXXXXXX
Switched network backup:       SWNBKU    XXXX
Activate swt network backup?   ACTSWNBKU XXXX
Allow delayed connection:      DLYFEAT   XXXX
Attached device names:         DEV
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
BSC device delay in sec:       DEVDLY    XXXX
```

```
 XX/XX/XX  XX:XX:XX   CONTROL UNIT DESCRIPTION
BSC program delay in sec:      PGMDLY     XXXX
Text description:              TEXT
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

DSPDBG (Display Debug) Command

The Display Debug (DSPDBG) command displays the current debug status as entered through the ENTDBG or the last CHGDBG command. It shows the invocation stack, indicating which programs are currently being debugged, the instruction number of the calling instruction or the instruction number of each breakpoint at which program execution is stopped, and the program invocation level. Also displayed are the names of the programs that are in debug mode but have not been invoked.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPDBG
```

This command, if entered interactively, displays on the work station the current attributes of debug mode. Also displayed are the breakpoints at which any of the programs being debugged are stopped, the invocation levels of the programs that are currently active, and the names of the programs that have not been invoked.

Additional Considerations

The following display is produced by the DSPDBG command:

```
 XX/XX/XX  XX:XX:XX     DEBUG DISPLAY
Default program:         XXXXXXXXXX
Trace full action:       XXXXXXXX  Maximum traces:      XXXXX
Update production files: XXX
DBG RQS PROGRAM    LIBRARY    STMT       INST INVLVL BKP TYPE
XXX XXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX XXXXX  XXXXXXXX
XXX XXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX XXXXX  XXXXXXXX
YES     XXXXXXXXXX XXXXXXXXXX                 NOT INVOKED
  .                                                    .
  .
```

Line 2 of the display gives the name of the default program in debug mode (if a default program has been designated). Line 3 indicates the action to be taken when a trace full condition occurs (either STOPTRC or WRAP is shown). Also shown is the maximum number of trace statements that can be put in the trace file. Line 4 specifies whether files in production libraries that are used by the job in debug mode can also be updated at the same time. The values shown on these lines depend on the values specified on the ENTDBG and CHGDBG commands entered.

The remainder of the debug display lists the names of all programs that are currently in the job's invocation stack, followed by the names of all other programs that were specified as being in debug mode but which are not currently invoked in the job. The program names are listed in the order that the programs appear in the invocation stack; if a program has been invoked (called) multiple times, the same program name appears with a higher invocation level number. The uninvoked programs in debug mode (if any) are listed at the end with NOT INVOKED specified, starting under the INVLVL column.

Beginning with the fifth line on the display, a separate line is used to show, for each program:

- Whether the program is in debug mode (via the ENTDBG or ADDPGM command). If it is not, the DBG field is blank.

- The request level of the request processing program, which normally occurs when the QCL program is called and when the CF3 key is used either at a breakpoint or from a menu to get the command entry display. If the request level exceeds 999, three plus (+) symbols will indicate the level.

- The name of the program that has been invoked in the job or that is in debug mode.

- The name of the library containing the program.

- The HLL (high-level language) statement identifier, if available, of the last instruction executed.
- The instruction number of the last instruction executed.
- The number of the invocation level for the program.
- If the program is stopped at a breakpoint, a value that describes the breakpoint is displayed in the fifth column:
  - USR. A breakpoint defined by the user has been reached.
  - TRCFULL. The breakpoint occurred because the program's trace file has been filled with trace records.
  - USR,FULL. The trace file became full at a user-defined breakpoint.
  - MSG. An unmonitored message caused the breakpoint.

DSPDBR (Display Data Base Relations) Command

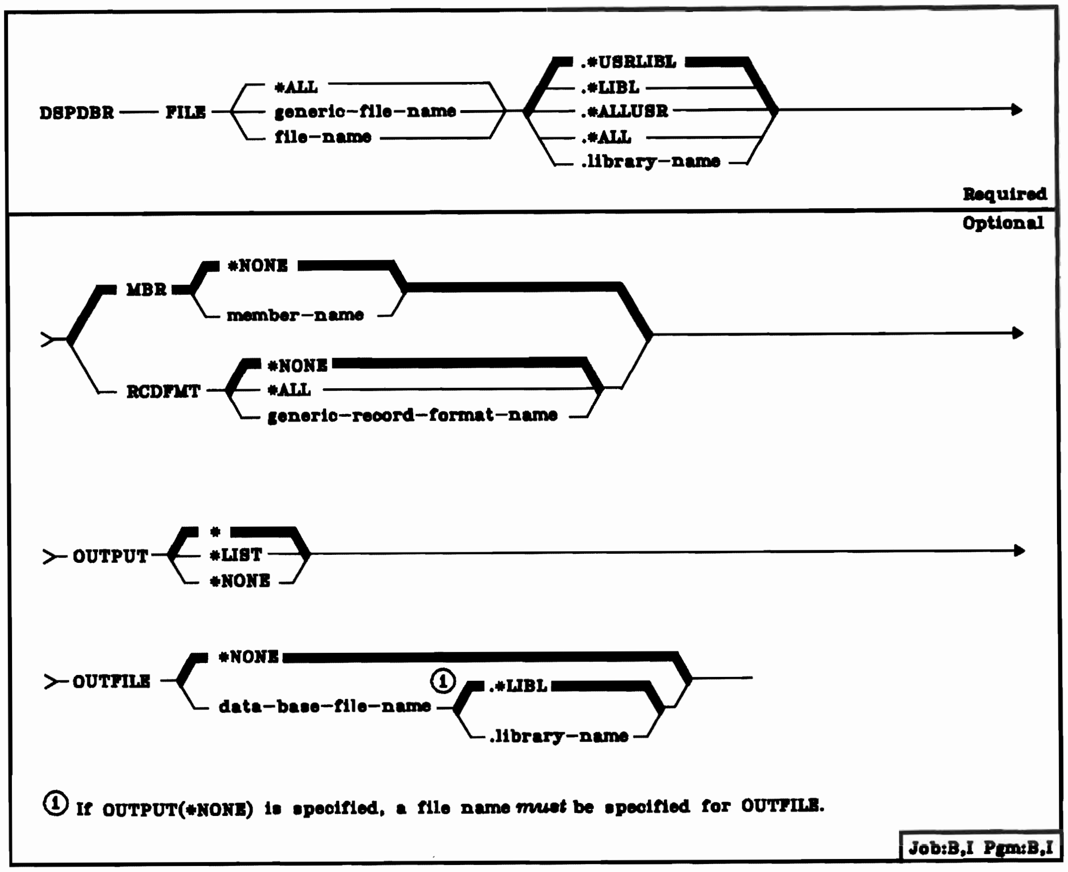
The Display Data Base Relations (DSPDBR) command provides relational information about data base files. The information identifies the physical and logical files that are dependent on a specific file, those files that use a specific record format, or the members that are dependent on a specific member. This command can be used to actually display or print the information, and/or to place the information in a data base file so that a user program can extract data from it as needed. (This command does not apply to device files.)

If the information is put in a data base file, a record composed of the following fields is produced. (The format of the record in the data base is *not* related to the format of the printed output.) The data base format is the same as that used in the IBM-supplied data base file QADSPDBR; the format is described in the *Application Documentation* chapter of the *CPF Programmer's Guide.*

- For each file specified in the command, the data base record contains:
  - The name of the specified file, its library name, and the file type.
  - The name of the record format used by the file, if a name is specified for RCDFMT.
  - The information retrieval date(s) for the file information. The latest date contains the most accurate information if changes have been made to the files.

- One of the following (and only one) is also included in the record:
  - The names of all the files that are dependent on the specified file for access path sharing or data sharing. The names of the libraries containing the files and the type of sharing are also included.
  - The names of all the file members that are dependent on the specified member, their library names, and the type of sharing.
  - The names of all the files that are dependent on the specified record format, and their library names.

Restrictions: To display each file specified, you must have operational authority for the file. Also, of the libraries specified by the library qualifier, only the libraries for which you have read rights are searched for the files.



FILE Parameter: Specifies the qualified name of the file or the generic name of several files about which relational information is to be displayed. Or, this parameter can specify that all files in the specified library or libraries (*ALL.*LIBL, for example), or all files in all libraries (*ALL.*ALL), are to have relational information about them displayed. Only the libraries in the specified library qualifier that the user either owns or is authorized to use are searched for the file(s).

Depending on the library qualifier specified or assumed, the following libraries (for which the user has the authority) are to be searched for the dependent file relationships:

- .*USRLIBL (user library list). Only the libraries listed in the user portion of the job's library list. If a specific file name is given, only the first file found by that name is displayed.
- .*LIBL* (library list). All the libraries in the user *and* system portions of the job's library list. If a specific file name is given, only the first file found by that name is displayed.
- .*ALLUSR* (all user libraries). All the nonsystem libraries, which include *all* user-defined libraries and the QGPL library, not just those in the job's library list. Libraries other than QGPL that begin with the letter Q are not included.

- .**ALL* (all libraries). All the libraries in the system, including QSYS.

- .*library-name* (one library). Only the library named in this parameter.

**ALL:* All files in the specified library (or all libraries identified in the library qualifier to which the user has access) are to have relational information about them displayed.

qualified-generic-file-name: Enter the qualified name of the file or the generic name of several files in the specified library qualifier that are to have relational information about them displayed. To specify a generic file name, add an asterisk (*) at the end of the characters that are in the names of all the files desired.

MBR Parameter: Specifies the name of the member in a data base file about which dependent information is desired.

*NONE: No information about the file member's dependencies is to be displayed. Either file dependencies or record format dependencies are displayed.

member-name: Enter the name of the member about which dependency information is to be displayed. If a member name is specified, either one file name or a generic file name must still be specified. If MBR is specified, RCDFMT cannot be specified.

RCDFMT Parameter: Specifies the name of the record format about which dependent information is to be displayed. If RCDFMT is specified, MBR cannot be specified.

*NONE: No dependent record format information is desired.

**ALL:* Information about all record formats in the specified file(s) is desired.

generic-record-format-name: Enter the name of the record format or the generic name of several record formats in the specified files about which information is to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the names of the files used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

**NONE:* The only output is to be to the data base file specified in OUTFILE.

OUTFILE Parameter: Specifies the name of the data base file into which the specified relational information is to be stored. If the specified file does not exist, this command causes a data base file and member to be created.

*NONE: The specified information is not to be put into a data base file.

qualified-data-base-file-name: Enter the qualified name of the data base file into which the relational information is to be placed. (If no library qualifier is given, *LIBL is used to find the file.) If the specified file name is not found, a file and member by that name are created and stored in the specified library, or in QGPL if not qualified. This file can be used when other DSPDBR commands are entered. Output from the file always starts at the beginning of the file member. Any data that exists in the file when this command is used is replaced by the output from this command. (The IBM-supplied data base file QADSPDBR *cannot* be specified.)

Examples

The following examples assume that there is an interactive environment and that the user of the command is authorized to access all relevant libraries and objects.

```
DSPDBR  FILE(FILE1.LIBRARY1) RCDFMT(FORMAT1)
```

This command displays a list of all files that use the FORMAT1 format and are associated with FILE1 in LIBRARY1. They are displayed at the work station initiating the DSPDBR command.

```
DSPDBR  FILE(FILE1.LIB1)
```

This command displays all files that are dependent on FILE1 in LIB1 for data sharing or access path sharing. They are displayed at the work station initiating the DSPDBR command.

```
DSPDBR  FILE(FILE1.LIB1) MBR(MEMBER1)
```

This command displays all members that are dependent on MEMBER1 of FILE1 in LIB1 for data sharing or access path sharing. They are displayed at the work station initiating the DSPDBR command.

Additional Considerations

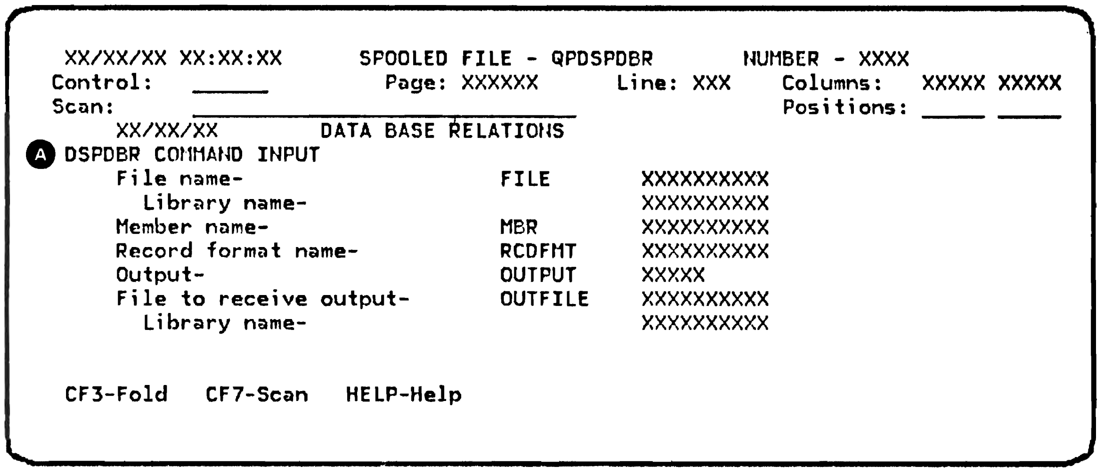
When the DSPDBR command is entered, the data base is searched for the specified relationships and a record is generated for each relationship (dependency) found. The records are placed in the printer device file named QPDSPDBR. If OUTPUT(*LIST) is specified on the command, the records are listed on the printer in the following order:

- Header information, which lists the DSPDBR command input values and the files identified in the data base that match the command's request.
- The names of the files that are dependent on the file(s) identified by the FILE parameter. (A file can be dependent on another file for its access path or its data.) This information is listed when neither MBR nor RCDFMT is specified.
- If a member name was specified in the MBR parameter, the names of the file members that are dependent on a member in the file or on members in different files all having the same member name.
- The names of the files that are dependent on a record format of the specified file(s), if a record format name was specified in the RCDFMT parameter.

If the DSPDBR command is entered interactively and OUTPUT(*) is specified or assumed, the records in the printer device file are displayed rather than printed.

The first line of each display shows the following information:

- The current job date and time.
- The name of the printer device file (QPDSPDBR) into which all records containing the desired relational information are placed after they are generated.
- The spooled file number of the display device file created by the DSPDBR command. If this is the first spooled output file produced in the job in which the command was entered, its number is 0001.
- The page, line, and columns of the spooled file being shown.



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPDBR       NUMBER - XXXX
Control:  ______           Page: XXXXXX       Line: XXX    Columns:   XXXXX XXXXX
Scan:                                                      Positions: _____ _____
     XX/XX/XX        DATA BASE RELATIONS
(A) DSPDBR COMMAND INPUT
     File name-                     FILE         XXXXXXXXXX
       Library name-                             XXXXXXXXXX
     Member name-                   MBR          XXXXXXXXXX
     Record format name-            RCDFMT       XXXXXXXXXX
     Output-                        OUTPUT       XXXXX
     File to receive output-        OUTFILE      XXXXXXXXXX
       Library name-                             XXXXXXXXXX


 CF3-Fold   CF7-Scan   HELP-Help
```

For each file or member in the data base (or in the library specified in the FILE parameter) that has the specified relationship to the file(s) named in the FILE parameter, a record is created that identifies the file or member and its dependent relationship. The record is placed in the printer device file to be printed or displayed. (Also, if a data base file name was specified on the OUTFILE parameter, the records are saved in the named file.)

The format given in the following series of displays is used to display all of the data base relations that exist between the file(s) specified on the DSPDBR command and the other files in the data base. The header information and spooled file control fields are always presented. The information in the other areas (C, D, and E) is displayed only when needed, based on what is specified on the MBR and RCDFMT parameters.

Refer to the *Additional Considerations* section of the DSPSPLF (Display Spooled File) command for information on the use of the scan and control fields.

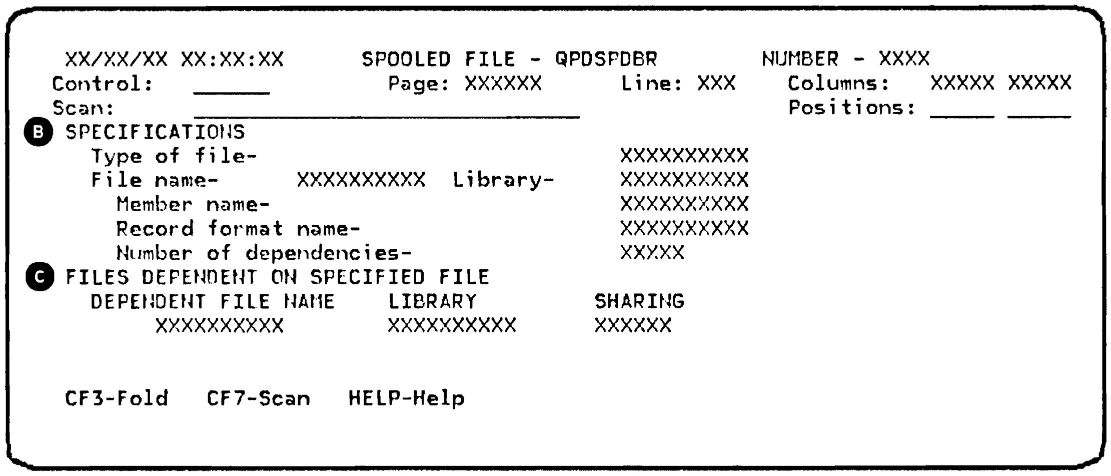
The first information to be displayed includes:

(A) The command input, containing the parameter values specified on the DSPDBR command.

(B) The file or files that were specifically or generically named in the FILE parameter. For each file identified in area B, one of the groups of information in areas C, D, or E that applies to the file is shown immediately following the file name.

Line 4 of the first display gives the current job date and the name of the displayed information. (This is the first line of the first page in the printer device file.)

Beginning at A, the rest of the first display shows all of the values specified on the DSPDBR command. Area B on the second display contains information about a file for which the relational information is to be displayed. If more than one file having the name specified in the FILE parameter exists in the data base (or in the library specified), area B is repeated for the next file after all the dependent relationships for the previously identified file have been displayed.



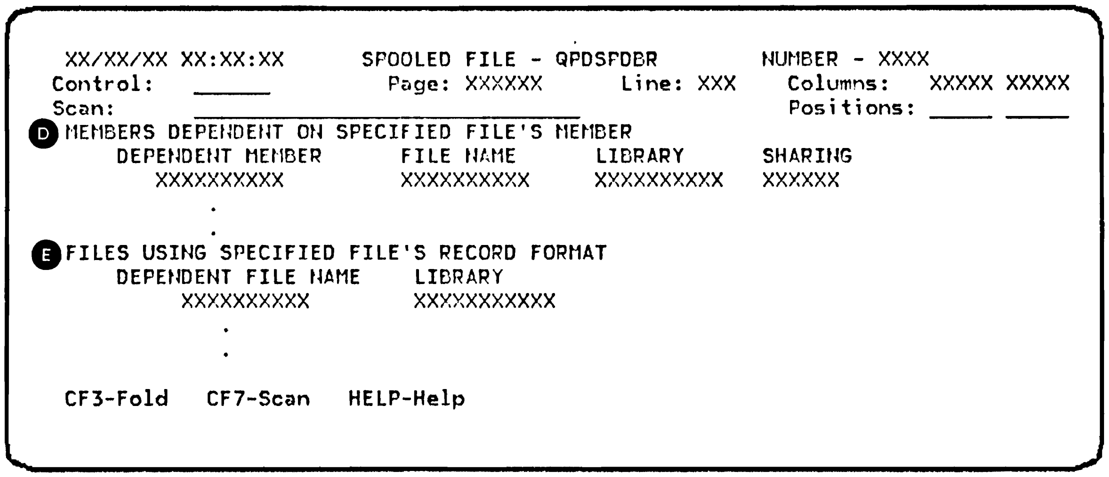
```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPDBR       NUMBER - XXXX
Control:  _____          Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________                Positions: _____ _____
B SPECIFICATIONS
   Type of file-                                XXXXXXXXXX
   File name-     XXXXXXXXXX  Library-          XXXXXXXXXX
     Member name-                               XXXXXXXXXX
     Record format name-                        XXXXXXXXXX
     Number of dependencies-                    XXXXX
C FILES DEPENDENT ON SPECIFIED FILE
   DEPENDENT FILE NAME    LIBRARY          SHARING
        XXXXXXXXXX        XXXXXXXXXX       XXXXXX


 CF3-Fold   CF7-Scan   HELP-Help
```

The five lines in area B identify, for each file having dependent files:

- The type of file.
- The name of the file that was specifically or generically named in the FILE parameter.
- The name of the file member, if it was specified in the MBR parameter.
- The record format name, if it was specified in the RCDFMT parameter.
- The number of dependencies that exist for the specified file, file member, or file record format. (1) If no member or record format name is specified, the number of dependencies correspond to the number of dependent *files* listed in area C. (2) If a member name is specified, the number corresponds to the number of dependent *members* listed in area D. (3) If a record format name is specified, the number corresponds to the number of *files*, listed in area E, that are dependent on the record format.

For each file that is dependent on the file identified in B, the following are displayed on one line in area C:

- The name of each file that is dependent on the specified files
- The name of the library in which the dependent file is located
- Whether the dependent file is sharing the access path (ACCPTH) or the data (DATA) in the specified files



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPDBR          NUMBER - XXXX
Control:  ______        Page: XXXXXX     Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________              Positions: _____ _____
D MEMBERS DEPENDENT ON SPECIFIED FILE'S MEMBER
      DEPENDENT MEMBER      FILE NAME      LIBRARY       SHARING
         XXXXXXXXXX         XXXXXXXXXX     XXXXXXXXXX    XXXXXX
                 .
                 .
E FILES USING SPECIFIED FILE'S RECORD FORMAT
      DEPENDENT FILE NAME    LIBRARY
           XXXXXXXXXX        XXXXXXXXXXX
                 .
                 .

 CF3-Fold   CF7-Scan   HELP-Help
```

If a member name is specified in the MBR parameter of the DSPDBR command, the information in area D is shown. For each file member that is dependent upon the specified member in the files, the following is displayed:

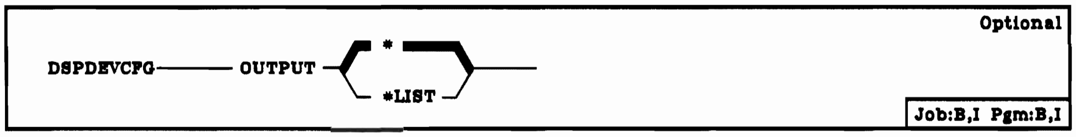
- The name of the dependent member
- The name of the file in which the dependent member exists
- The name of the library in which the file and member exist
- The item shared with the dependent member, either an access path or the data in a member

If a record format name is specified in the RCDFMT parameter of the DSPDBR command, area E will list the names of all the files (and the names of their libraries) in the data base that use the specified record format and file.

Note that, on the actual displays: (1) only the information requested is displayed, (2) the displayed information is shown in the same sequence shown here, and (3) the records displayed are listed continuously with no blank lines between groups.

DSPDEVCFG (Display Device Configuration) Command

The Display Device Configuration (DSPDEVCFG) command displays the configuration information of all line descriptions, control unit descriptions, and device descriptions in the system.



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

**:* The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPDEVCFG
```

This command displays all the system device configuration information in three sets of displays.

Additional Considerations

The information about the device configuration of the entire system, which is generated by the DSPDEVCFG command, is shown in three sets of displays. Each set contains one or more displays that describe only one line, control unit, or device.

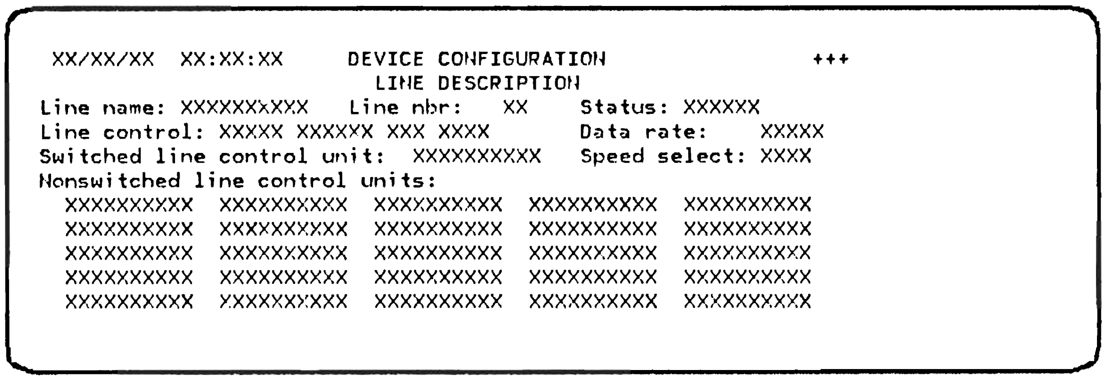
The first set shows:

- The name of the line attached to the system. The remainder of the attributes shown on this display apply only to that line. For each additional line attached to the system, another display is presented.

- The line number (OU number) and the speed (data rate) of the line, and whether the line has the data rate select function (YES or NO).

- The line control characteristics (shown in four fields):
  - The line type (SDLCP, SDLCS, or BSC)
  - The type of character code used on the line (EBCDIC or ASCII)
  - The type of line connection (SWT, PP, or MP)
  - NRZI, if the NRZI transmission method is used on the line

- The names of the control units attached to the line. Only one control unit can be on a switched line.



```
 XX/XX/XX  XX:XX:XX     DEVICE CONFIGURATION              +++
                          LINE DESCRIPTION
Line name: XXXXXXXXXX   Line nbr:   XX    Status: XXXXXX
Line control: XXXXX XXXXXX XXX XXXX       Data rate:    XXXXX
Switched line control unit:  XXXXXXXXXX   Speed select: XXXX
Nonswitched line control units:
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
```

All three sets of displays have a *status* field. One of the following is shown as the status of the line, control unit, or device being displayed:

- ACTIVE. Varied online and in use.

- DIAG. In diagnostic mode.

- PWROFF. Powered off.

- VRYOFF. Varied offline.

- VRYON. Varied online.

- VRYONP. (Vary on pending). Currently being varied online.

- FAILED. The line is in an unusable state; it can possibly be made usable again by varying if off, then on.

The second set shows each control unit attached to the system; the control unit's name, address, type, model, and exchange identifier; the name and type of line to which it is attached; and the names of all devices attached to that control unit. The control unit address is made up of the controller station address (yy) and the OU number (zz). (Each control unit and its associated information are shown on a separate display.)

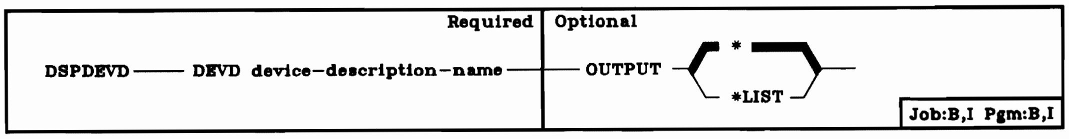
```
 XX/XX/XX  XX:XX:XX    DEVICE CONFIGURATION             +++
                     CONTROL UNIT DESCRIPTION
Control unit name: XXXXXXXXXX  Address: YYZZ  Status: XXXXXX
Type: XXXX   Model: XXXX
Exchange ID: XXXXXXXX
Nonswitched line:  XXXXXXXXXX     Switched line:  XXXXXXXXXX
Attached device names:
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
```

The third set of displays shows all of the devices attached to the system. Each device is listed with its name, address, type, model number, and the name of the control unit to which it is attached. The device address is made up of the unit address (xx), the controller station address (yy), and the OU number (zz).

```
 XX/XX/XX  XX:XX:XX    DEVICE CONFIGURATION
                       DEVICE DESCRIPTIONS
DEVICE       DEVICE   DEVICE   MODEL    DEVICE     CONTROL
NAME         ADDRESS   TYPE    NUMBER   STATUS     UNIT NAME
XXXXXXXXXX   XXYYZZ    XXXX     XXXX    XXXXXX     XXXXXXXXXX
XXXXXXXXXX   XXYYZZ    XXXX     XXXX    XXXXXX     XXXXXXXXXX
XXXXXXXXXX   XXYYZZ    XXXX     XXXX    XXXXXX     XXXXXXXXXX
XXXXXXXXXX   XXYYZZ    XXXX     XXXX    XXXXXX     XXXXXXXXXX
XXXXXXXXXX   XXYYZZ    XXXX     XXXX    XXXXXX     XXXXXXXXXX
XXXXXXXXXX   XXYYZZ    XXXX     XXXX    XXXXXX     XXXXXXXXXX
XXXXXXXXXX   XXYYZZ    XXXX     XXXX    XXXXXX     XXXXXXXXXX
```

DSPDEVD (Display Device Description) Command

The Display Device Description (DSPDEVD) command displays information about the specified device and its description.



DEVD Parameter: Specifies the name of the device description to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPDEVD  DEVD(WRKSTN01)
```

This command displays information about the device description named WRKSTN01. The information is displayed on the work station from which the command was submitted; or it is spooled to a printer output queue to be printed on the system printer, if the command was part of the batch input stream.

Additional Considerations

Four displays are used to display the attributes of the device description identified in the DSPDEVD command. The attributes are displayed in the same order as their associated parameters (whose keywords are shown here) occur in the CRTDEVD command. For an explanation of each attribute of the device description, refer to the associated parameter description given in the CRTDEVD command description.

The first display has the following format:

```
 XX/XX/XX  XX:XX:XX   DEVICE DESCRIPTION                    +++
Status: XXXXXXXXXXXXXXXXXXXX
Device description name:        DEVD        XXXXXXXXXX
Device address:                 DEVADR      XXYYZZ
Device type code:               DEVTYPE     XXXX
Model number:                   MODEL       XXXX
Control unit description name:  CTLU        XXXXXXXXXX
Online at CPF start:            ONLINE      XXXX
```

Line 2 of the first display contains one field not specified in the CRTDEVD command, but whose information is kept current in the device's description. The *status* field indicates one of the following about the device:

- ACTIVE. The device is varied on and is either currently in use or displaying a sign-on display.
- DIAGNOSTIC MODE. The device is being serviced.
- POWERED OFF. The device is in a varied-off and powered-off state.
- VARIED OFF. For a device that can be powered on or off by the PWRDEV command, the status is indicated after the device is powered on.
- VARIED ON. The device is powered on and is varied online.
- VARY ON PENDING. The VRYDEV command has been issued and the system is waiting for an action to be completed, such as a switched connection being made.
- FAILED. The device is in an unusable state; it can possibly be made usable again by varying it off, then on.

The device address (specified as six digits in the CRTDEVD command) is displayed in the format *xxyyzz*, where xx is the hexadecimal unit (device) address, yy is the controller station address, and zz is the operational unit (OU) number of the line, device, or control unit. If the device description being displayed is for a locally attached work station or work station printer, the device address attribute field contains zeros.

The second display shows all of the error retry and threshold values that have been specified for the device:

```
 XX/XX/XX  XX:XX:XX    DEVICE DESCRIPTION                     +++
Device error retries:                RETRY
 (type - times)
  XX-XX
  XX-XX
Device error log threshold:          THRESHOLD
 (type - number allowed)
  XX-XXXX
  XX-XXXX
```

The third and fourth displays show the remainder of the displayable attributes in the device description, except for the command authorization as specified in the PUBAUT parameter.

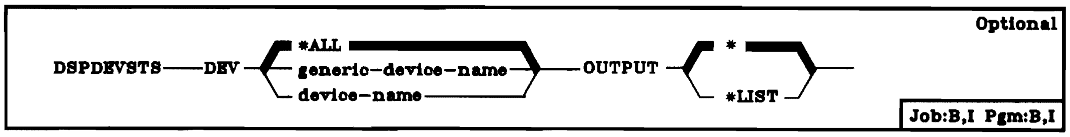
```
 XX/XX/XX  XX:XX:XX  DEVICE DESCRIPTION                     +++
 Drop line at sign off:        DROP       XXXX
 Associated work stn printer:  PRINTER    XXXXXXXXXX
 Associated message queue:     MSGQ       XXXXXXXXXX
   Library name:                          XXXXXXXXXX
 Print image name:             PRTIMG     XXXXXXXXXX
   Library name:                          XXXXXXXXXX
 Print device file name:       PRTFILE    XXXXXXXXXX
   Library name:                          XXXXXXXXXX
```

```
 XX/XX/XX  XX:XX:XX   DEVICE DESCRIPTION
Work stn controller address:   WSCADR     NNCCSS
Work stn controller keyboard:  WSCKBD     YZZZ
Allow blinking cursor:         ALWBLN     XXXX
BSC contention resolution:     CONTN      XXXXX
Text description:              TEXT
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

The work station controller address is displayed in the format *nnccss*, where nn is the decimal unit address assigned by the customer to the device, cc is the WSC connector number for the device, and ss is the work station address switch settings on the device. The work station controller keyboard identifier is displayed as *yzzz*, where y specifies the type of keyboard and zzz specifies the language group identifier.

DSPDEVSTS (Display Device Status) Command

The Display Device Status (DSPDEVSTS) command displays the names and status of specified devices on a system, or the hierarchical configuration (if applicable) of a single specified device. The status of each of these objects is also displayed, in addition to job names of all interactive, batch, autostart, reader, or writer jobs that are holding a lock on a device.



DEV Parameter: Specifies whether the status of all devices on the system is to be displayed or only the status information for a specific device.

<u>*ALL:</u> The status information for all devices is to be displayed. No control unit or line name information for these devices will be directly displayed.

generic-device-name:* The status information for all devices with this generic name is to be displayed. To specify a generic name, add an asterisk after the last character in the generic name (ABC*, for example). If an asterisk is not included with the name, the system assumes that the name is a complete device name. If a generic name is specified, no control unit or line information will be displayed.

device-name: Specifies the name of the device for which status information is to be displayed. If a control unit is attached to this device, information about that control unit will precede the device status information on the display. If a line is attached to the control unit, information about that line will precede the control unit status information. All of the devices associated with the attached control unit will *not* be displayed when a device name is specified; this information can be obtained using the DSPLINSTS or DSPCTLSTS commands.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or is to be listed with the job's spooled printer output. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the file(s) used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or listed with the job's system output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled printer output.

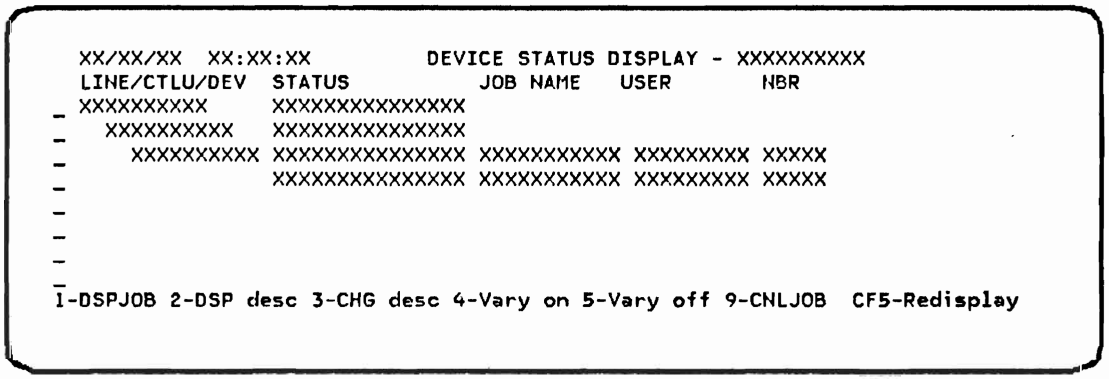
Example

DSPDEVSTS DEV(QTAPE1)

This command displays the name and status of device QTAPE1 and the name and status of any control unit and line (if applicable) to which it is attached, as well as the names of any jobs using an active device. The information is displayed on the work station from which the command was submitted; or if the command was part of the batch input stream, it is spooled to a printer output queue to be printed on the system printer.

Additional Considerations

The displays produced by the DSPDEVSTS command will show various amounts of device configuration information, depending on what you specify for the DEV parameter. As an example, if you specify DSPDEVSTS DEV device-name (where the device is attached to a control unit and a line), the following display is shown:



```
 XX/XX/XX  XX:XX:XX         DEVICE STATUS DISPLAY - XXXXXXXXXX
 LINE/CTLU/DEV  STATUS          JOB NAME   USER       NBR
_XXXXXXXXXX     XXXXXXXXXXXXXXX
_  XXXXXXXXXX   XXXXXXXXXXXXXXX
_    XXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXX XXXXXXXXX XXXXX
_               XXXXXXXXXXXXXXX XXXXXXXXXXX XXXXXXXXX XXXXX
_
_
_
_
1-DSPJOB 2-DSP desc 3-CHG desc 4-Vary on 5-Vary off 9-CNLJOB  CF5-Redisplay
```

Header Information

The first line of the display shows the current job date and time fields followed by the value specified for the DEV parameter (either '*ALL', a generic device name, or a specific device name).

First-column Input Field

The leftmost column of the display consists of a single-digit input field in which a number can be entered. You can enter the number (any one of those shown at the bottom of the display) in the input field to cause the function (a command) associated with that number to be performed for that line/control-unit/device when the you press the Enter key. If you enter numbers in the input fields of several items before pressing the Enter key, the specified functions are performed on the items in the order in which the items are displayed. The following functions can be specified:

```
1–DSPJOB:    Executes the DSPRDR command for a reader job,
             executes the DSPWTR command for a writer job,
             executes the DSPJOB command for all other jobs,
             or returns 'Job .. not found' if no job
             is associated with the input record.

2–DSP desc:  Executes the DSPLIND command for a line,
             executes the DSPCUD command for a control unit, or
             executes the DSPDEVD command for a device.

3–CHG desc:  Prompts for the CHGLIND command for a line,
             prompts for the CHGCUD command for a control unit, or
             prompts for the CHGDEVD command for a device.

4–Vary on:   For a line,  1) the line is varied on,
                          2) all attached control units are varied on, or
                          3) all attached devices are varied on.
             For a control unit, 1) the control unit is varied on, or
                                 2) all attached devices are varied on.
             For a device, the device is varied on.

5–Vary off:  For a line,  1) all attached devices are varied off, if possible,
                          2) all attached control units are varied off, if possible, or
                          3) the line is varied off, if possible.
             For a control unit, 1) all attached devices are varied off,
                                    if possible,
                                 2) the control unit is varied off,
                                    if possible.
             For a device, the device is varied off.
```

Note: As is the case when the VRYxxx commands are entered individually, there is a noticeable delay when varying off a line/control-unit/device that has already been varied off.

```
9–CNLJOB:    Executes the CNLRDR command (with OPTION(*CNTRLD))
             for a reader job,
             executes the CNLWTR command (with OPTION(*CNTRLD))
             for a writer job,
             executes the CNLJOB command (with OPTION(*CNTRLD))
             for all other types of jobs, and
             returns 'Job .. not found' if no job is associated with
             the input record.
```

When all of the commands have been executed, the display is shown again with the status fields of the objects updated and with any error messages that occurred when the commands were executed. The display can be shown at any time *before* all the commands have been executed by pressing the CF5 key.

If the configuration has more elements than can be shown on a single display, the Roll Up key can be used to display them all. You can enter numbers in the input fields on multiple displays before pressing the Enter key.

After the commands have been executed, if there are more error messages than can fit on that display, a '+' is shown at the end of the last message. You must position the cursor at the first message and use the Roll Up key to view all of the error messages.

Line/Ctlu/Dev

The second vertical column displays the name of the item whose information is being displayed on that line. Control unit (CTLU) names are indented two spaces and device (DEV) names are indented four spaces. The heading of this column varies with the value specified for the CTLU parameter. If you specify *ALL, or a generic control unit name, or the name of a device that is not attached to a control unit, only the DEV field value will be shown. If you specify the name of a device that is attached to a control unit, the CTLU/DEV field values will be shown. The LINE/CTLU/DEV field values will be shown if you specify the name of a device that is attached to a control unit and a line.

Status

The third column lists the status of the line/control-unit/device. One of the following values is used to indicate status:

- ACTIVE. The line, control unit, or device is currently in use. For a display device, the device is signed on or has been allocated by a batch, auto-start, or interactive job.

- ACTIVE/RDR. A spool reader is using this device.

- ACTIVE/WTR. A spool writer is using this device.

- CONNECT PENDING. A VRYLIN command has been issued for this line, and the system is waiting for an action to be completed, such as a switched connection to be made.

- DIAGNOSTIC MODE. The line, control unit, or device is being serviced or has otherwise been set to diagnostic mode.

- FAILED. The line, control unit, or device is in an unusable state; it can possibly be made usable again by varying it off, then on. A failed device may still be allocated to a job.

- FAILED/RDR. This device, which is in an unusable state, is still allocated to a spool reader job.

- FAILED/WTR. This device, which is in an unusable state, is still allocated to a spool writer job.
- POWERED OFF. The control unit or device is in a varied-off and powered-off state.
- SIGNON DISPLAY. This display device currently has the 'Enter password to signon' screen displayed.
- SYSREQ. This display device has been requested by the system, and the job associated with this status does not have a lock on the device. SYSREQ status coexists only with an ACTIVE or SIGNON DISPLAY status for this device.
- VARIED OFF. For a control unit or device that can be powered on or off by the PWRCTLU or PWRDEV command, the status is indicated after the control unit or device is powered on. For a line, this status indicates that the line is varied off.
- VARIED ON. The line, control unit, or device is varied online, although it may not be physically powered on.
- VARY ON PENDING. A VRYCTLU or VRYDEV command has been issued for this control unit or device, respectively, and the system is waiting for an action to be completed, such as a switched connection to be made.
- *DAMAGED. The line, control unit, or device has incurred hard or partial damage; it is not possible to obtain any further status information.
- *LOCKED. The line, control unit, or device is allocated to another job with an *EXCL lock, and its attributes can not be determined at this time.
- *UNKNOWN. All of the status bits for the line, control unit, or device have been checked, and none are set. This is an exceptional condition.

Job Name

The fourth column shows the names of the jobs that are currently using any of the named devices.

User

The fifth column shows the name of the user profile under which the job that holds the lock on the device is running.

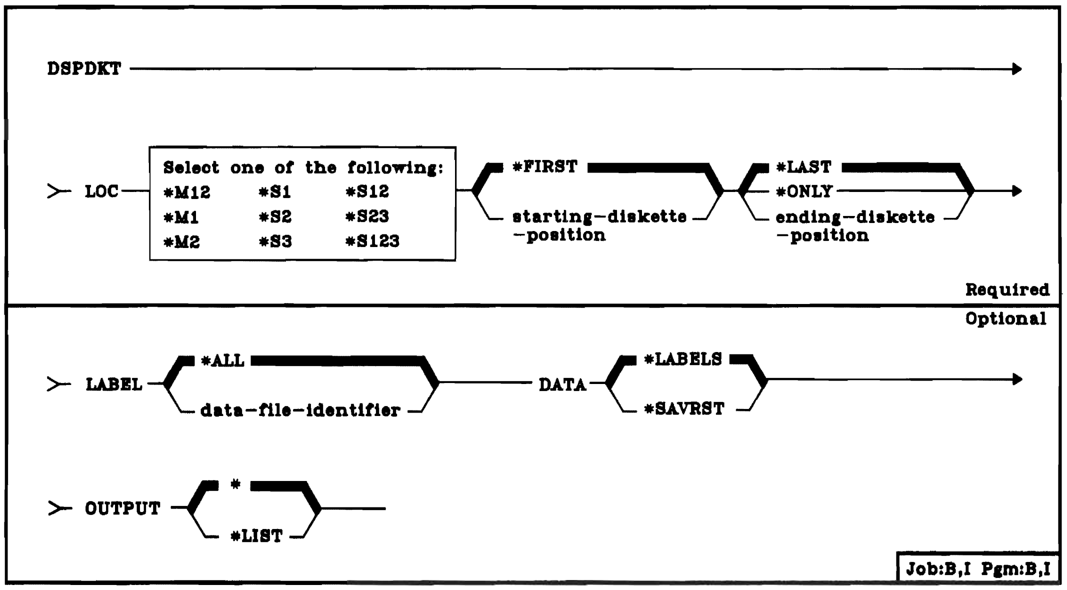
Nbr

The rightmost column shows the six-digit number assigned by the system to identify the job.

DSPDKT (Display Diskette) Command

The Display Diskette (DSPDKT) command displays the volume label and data file identifier information that is contained on one or more diskettes. The information can be written on a printer or displayed on a display device.

Note: When displaying diskettes with non-IBM standard labels, you may have unpredictable results. To initialize the diskette, execute the Initialize Diskette (INZDKT) command with CHECK(*NO) specified.



LOC Parameter: Specifies which diskette location(s) in the magazines or slots are to be displayed. Three values are needed: (1) the unit type and location (that is, the magazines or slots used), (2) the starting diskette position, and (3) the ending diskette position in the unit. (For an expanded description of the LOC parameter, see Appendix A.) A value must be specified for the first of the three values; if no values are specified for the other two, *FIRST and *LAST are assumed by the system.

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit and location are to be displayed. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette that is to be displayed first. Enter one of the following values to specify the starting diskette position:

<u>*FIRST:</u> The first diskette position in the location contains the diskette to be displayed first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

starting-diskette-position: Enter the number of the diskette position that contains the first diskette to be displayed.

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be displayed last. Enter one of the following values to specify the ending diskette position:

*LAST: The last diskette position in the location contains the diskette to be displayed last. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**ONLY:* Only the diskette position specified by the second value is to be displayed.

ending-diskette-position: Enter the number of the diskette position that contains the last diskette to be displayed.

LABEL Parameter: Specifies the data file identifier, or all the identifiers, of the data file(s) on the diskette that is to be displayed. The data file identifier is stored in a label in the volume label area of the diskette, and it specifies the identifier of the file that exists on the diskette.

*ALL: All data file identifiers on the diskettes specified by the LOC parameter are to be displayed.

data-file-identifier: Enter the data file identifier (8 or 15 characters maximum) of the data file that is to be displayed. (If this identifier is on a diskette in the save/restore format, the identifier can contain a maximum of 15 characters: up to 10 characters in a library name, followed by a period, a Q character, and a three-digit sequence number. For data files not in the save/restore format [basic, H-, or I-format], the identifier can contain a maximum of 8 characters.)

Note: If a range of diskettes is specified (for example, LOC(*M1)), all diskettes in the specified range are searched for the given file identifier. If DATA(*LABELS) is specified, the labels of all those that match are displayed; the search does not stop when a match occurs.

DATA Parameter: Specifies the type of information that is to be displayed.

*LABELS: Volume and data file identifiers are to be displayed.

**SAVRST:* Save/restore information is to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the names of the files used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPDKT  LOC(*M12)
```

This command displays all the volume labels and data file identifiers of all the diskettes in magazines 1 and 2.

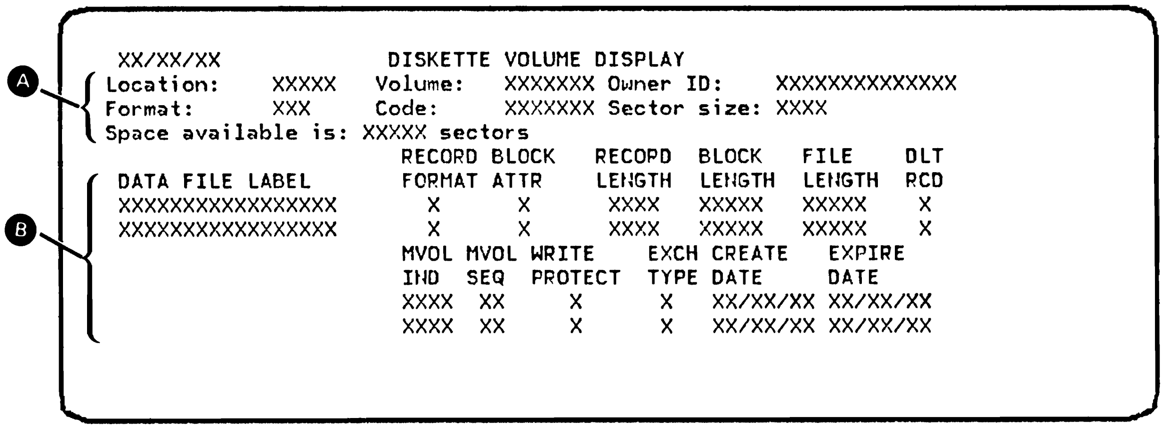
Additional Considerations

The DSPDKT command can produce two types of displays, depending on the value specified in the DATA parameter:

- If *LABELS is specified, the information in the VTOC (volume table of contents) area of a diskette can be displayed. The same display is used to display the VTOCs of diskettes that are in either the basic, H-, or I-format or the save/restore format.
- If *SAVRST is specified, a series of displays that identify the libraries, objects, and data base file members saved on a diskette can be displayed. These displays apply only to diskettes in the save/restore format.

Displaying Diskette and Data File Attributes

If DATA(*LABELS) is specified or assumed, the following diskette volume display is shown:



```
 XX/XX/XX              DISKETTE VOLUME DISPLAY
Location:     XXXXX   Volume:    XXXXXXX Owner ID:     XXXXXXXXXXXXXX
Format:       XXX     Code:      XXXXXXX Sector size: XXXX
Space available is: XXXXX sectors
                       RECORD BLOCK   RECORD  BLOCK   FILE    DLT
 DATA FILE LABEL       FORMAT ATTR    LENGTH  LENGTH  LENGTH  RCD
 XXXXXXXXXXXXXXXXX       X      X      XXXX   XXXXX   XXXXX    X
 XXXXXXXXXXXXXXXXX       X      X      XXXX   XXXXX   XXXXX    X
                       MVOL MVOL WRITE     EXCH CREATE    EXPIRE
                       IND  SEQ  PROTECT   TYPE DATE      DATE
                       XXXX  XX     X       X   XX/XX/XX XX/XX/XX
                       XXXX  XX     X       X   XX/XX/XX XX/XX/XX
```

Each display of this type shows information in the volume label area (VTOC) of the diskette, followed by all of the information in the file label area associated with one or all data files, depending on the value specified for the LABEL parameter. If one data file identifier is specified for LABEL, only one line of information (about that data file) follows each set of column headings.

If there are multiple data files on the diskette and LABEL(*ALL) is specified or assumed, the information for as many data files as can fit completely on the display is presented. That is, for each data file, two lines are used (one under each set of column headings) to display all of its file label information; in the 12-line display above, two data files are shown. If all the data files cannot be displayed on one display, a + sign is shown in the lower right of the display, indicating that the Roll Up key should be used to display the rest. The first four lines containing the volume label information are repeated on each display.

If multiple diskettes are to be displayed (as specified by the LOC parameter), +++ is shown at the upper right of each display that is followed by another one. A display for each diskette is presented each time the Enter key is pressed. For multivolume diskette files, the volume label information is the same on each display.

A Volume Label Fields

Location: The magazine or slot location containing the diskette being displayed. *S1, *S2, or *S3 identifies a slot location. *M1 or *M2 followed by a number identifies the diskette's position in a magazine.

Volume: The volume identifier of the volume being displayed.

Owner ID: The identifier of the diskette's owner.

Format: The physical characteristics of the recording surface of the diskette. The following values can be shown:

1	Single-sided diskette
2	Double-sided diskette with single-density recording capability
2D	Double-sided diskette with double-density recording capability

Code: The character set (EBCDIC or ASCII) in use on the diskette.

Sector Size: Indicates the number of bytes available in each sector.

Space Available: Indicates the number of blocks (sectors) available after the last *unexpired* file on the diskette and the number of bytes (characters) per block.

B File Label Area Fields

Data File Label: The data file identifier of the data file, which was assigned by the user when the file was created.

Record Format: Indicates, if either an F is displayed or the field is blank, that fixed length records are in the data file. The length is given in the record length field.

Block Attribute: Indicates whether the records in the data file are blocked (multiple records are contained in each sector) or unblocked (one record is contained in each sector), and whether each record can span multiple sectors or must be contained in a single sector.

(The block attribute does not apply to diskettes in the basic or H data exchange format.)

ƀ = Unblocked, unspanned
B = Blocked, unspanned
S = Unblocked, spanned
R = Blocked, spanned

Record Length: Indicates the number of bytes in each record. The length of one record could be the same as the block length, meaning a sector contains only one record. (This attribute does not apply to diskettes in the basic or H data exchange format.) If this field is blank, the record length equals the block length.

Block Length: Indicates the number of bytes in each block (sector).

File Length: Indicates the number of sectors in the specified data file.

DLT RCD (Logical Record Delete Indicator): Displays whether this I-format file contains logically deleted records. A character in this field indicates that records have been logically deleted; a blank indicates that records have not been logically deleted. Records that contain (in the last position) the character indicated here are considered logically deleted.

MVOL Ind (Multivolume Indication): Indicates whether the data file is complete on the specified diskette, is continued on another diskette, or is the last diskette of a multivolume data file.

Blank = Complete on this diskette
C = Continued on another diskette
L = Last diskette of data file

MVOL Seq (Multivolume Sequence): Displays, for a multivolume file only, a number that indicates the sequential order of this diskette relative to the other diskettes on which this data file is stored.

Write Protect: Indicates whether the data file is protected from being overwritten. A P indicates that the file is protected from write operations (the file can only be read). If this field is blank, the file is not protected.

EXCH Type (Exchange Type): Indicates the type of data exchange used on the diskette. A blank field indicates that the diskette is in the basic data exchange format (128 bytes per sector). H indicates exchange type H (two-sided diskettes similar to the basic data exchange but having 256 bytes per sector). If E is shown, all fields in the file label area must be checked to determine the attributes of the data file. The save/restore format (which has 1024 bytes per sector) is included in type E, but it is not shown by this display. I indicates exchange type I (128, 256, 512, or 1024 bytes per sector).

Create Date: The date on which the data file was created. The date is displayed in the format specified by the system value QDATFMT. If no date is shown, the file label has no date.

Expire Date: The date *after which* the data file can be deleted. No date indicates the data file can be deleted; the date has expired. If *PERM is displayed, the file is a permanent file and should not be deleted. A data file is designated as a permanent file when the device file used to write the data on diskette specifies 99/99/99 as the expiration date (in the EXPDATE parameter).

Diskettes and Tapes in Save/Restore Format

Note: Because the save/restore displays are nearly identical for both diskettes and tapes written in the save/restore format, they are described here only.

If DATA(*SAVRST) is specified in the Display Diskette (DSPDKT) or Display Tape (DSPTAP) commands to display the contents of a diskette or tape volume in the *save/restore format*, one or more displays are produced. The resulting displays depend on how many data files are on the volume, what commands were used to save the libraries or objects, and whether you specified LABEL(*ALL) or a single data file identifier on the DSPDKT or DSPTAP command.

(If the diskette being displayed was used to save information with the SAVSYS command or it is one of the IBM-supplied diskettes used to install the system, two additional displays (not shown here) are always shown first. Both displays identify an installation file used when CPF is installed on the system.)

Each library involved in a save operation is stored on diskette or tape in different data files; no data file can contain objects from more than one library. When multiple libraries are involved in a single save operation, an additional data file that contains only the names of the libraries being saved is first created on the volume, followed by a data file for each library containing all of the objects in that library.

Library-Oriented Displays

If LIB(*NONSYS) was specified on a Save Library (SAVLIB) command and three libraries were then saved, four data files would be created on the volume. Assuming that no other save operation has occurred on this volume, all four data files would be displayed in the following order if LABEL(*ALL) were specified on the DSPDKT or DSPTAP commands. The first display would contain the information found in the first data file, which is *library oriented.*

```
XX/XX/XX XX:XX:XX  SAVE/RESTORE VOLUME - XXXXXX  LIBRARIES XXX
Libraries:  XXXXX                        Save cmd:    XXXXXXXXXX
Save date/time:   XX/XX/XX XX:XX:XX      Expiration:  XX/XX/XX
File label ID:    XXXXXXXXXXXXXXXXX
  LIBRARY NAME
  XXXXXXXXXX
  XXXXXXXXXX
      .
      .
      .

CF3 - Next volume      CF6 - Previous volume
```

The first line gives the current job date and time, identifies the volume identifier of the diskette being displayed, and indicates that this is a library-oriented display. The second line indicates how many libraries were saved and how they were saved (in this case, SAVLIB would be displayed in the save command/type field).

The third line specifies the date and time when the libraries were saved and the expiration date of the save/restore files. The file label identifier is shown on the *fourth* line; for tape, the fourth line also shows the file sequence number of the save/restore file on the tape volume.

After that, the name of each library that was saved by the save command is listed on the display. To display the contents of each library (that is, all of the objects in the library that were saved), press the Enter key. The CF3 and CF6 keys apply only to diskette volumes; they cannot be used to display different tape volumes.

Object-Oriented Displays

The remaining displays would contain the information in the last three data files. Information about all of the objects in the first library would be displayed in one or more displays, followed by a series of displays for each of the other two saved libraries. The format of the object-oriented display is:

```
XX/XX/XX XX:XX:XX  SAVE/RESTORE VOLUME - XXXXXX  OBJECTS
Library:  XXXXXXXXXX Objects:  XXXXXX  Save cmd:     XXXXXXXXXX
Save date/time:   XX/XX/XX XX:XX:XX    Expiration:  XX/XX/XX
File label ID:    XXXXXXXXXXXXXXXXX    File seq:     XXXX
  OBJECT        OBJECT    SUB   OWNER           SYS STG   DATA
  NAME          TYPE      TYPE  NAME           REQUIRED  ON (dkt or tape)
_ XXXXXXXXXX    XXXXXXXX  XXXX  XXXXXXXXXX  XXXXXXXXXX    XXX
    XXXXXXXXXX  XXXXXXXX  XXXX  XXXXXXXXXX  XXXXXXXXXX    XXX
      .
      .
      .

1-Display saved members
```

The first line gives the current job date and time, identifies the volume identifier of the diskette or tape being displayed, and indicates that this is an object-oriented display. The second line identifies the library from which the objects were saved, how many objects were saved, and what command was used to save them.

The third line specifies the date and time when the objects were saved and the expiration date of the file containing the saved objects. The file label identifier is shown on the *fourth* line; for tape, the fourth line also shows the file sequence number of the first data file on the tape volume.

The CF3 key can be used to advance the next diskette save/restore volume display. CF6 can be used to return to the save/restore display shown for a previous diskette volume.

The fifth and sixth lines of the display specify six column headings below which the following are displayed for each object saved in the specified file:

1. The name of the saved object.

2. The object type. Only the following CPF object types are valid on this display.

*CLS	Class
*CMD	Command
*DTAARA	Data area
*EDTD	Edit description
*FCT	Forms control table
*FILE	File
*JOBD	Job description
*JRN	Journal
*JRNRCV	Journal receiver
*LIB	Library
*MSGF	Message file
*PGM	Program
*PRTIMG	Print image
*SBSD	Subsystem description
*SSND	Session description
*TBL	Table

   If the Save System (SAVSYS) command was used to save the system on the volume being displayed, the following object types can also be displayed:

*CUD	Control unit description
*DEVD	Device description
*LIND	Line description
*USRPRF	User profile

3. The subtype of the file or program objects. For files, the following subtypes can be displayed.

DSP	Display
PRT	Printer
PHY	Physical
LGL	Logical

   For program objects CL and RPG can be displayed.

4. The name of the object's owner.

5. The number of bytes in the system used to store the object's contents.

6. Whether the data portion of the object has been saved on diskette or tape. If the object's storage was freed when the object was saved, NO is shown in this field even if STG(*FREE) was specified on the command performing the save. This is because one save operation cannot free the storage on the system while simultaneously saving the object on diskette in the freed state.

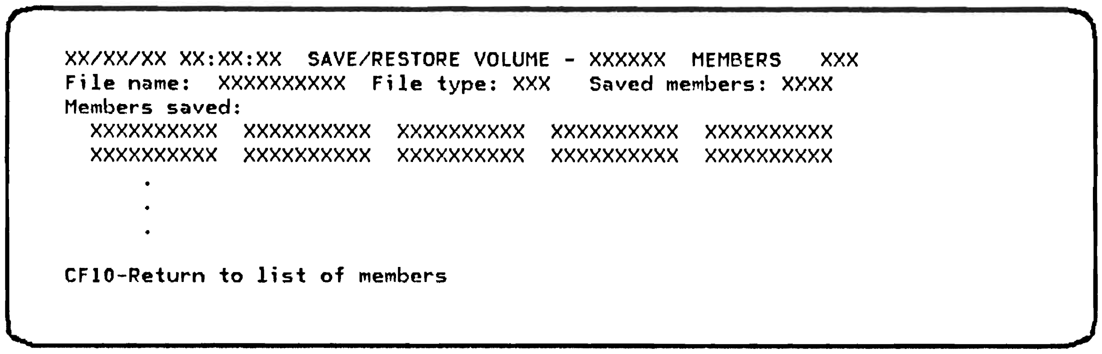
The following chart shows, for each type of save operation, what can be displayed from a diskette or tape in the save/restore format.

Type of Save	Displayed Information
SAVOBJ: One, several, or all objects in a library. SAVCHGOBJ: All changed objects in a library. SAVLIB: One library only.	The object-oriented displays are shown: the library is identified and its objects are displayed. Members of data base files can also be displayed.
SAVLIB: LIB(*NONSYS) is specified, saving all user libraries, including QGPL. SAVSYS: Saves all system libraries and other system objects.	The library-oriented display is shown first, identifying the libraries saved. Then the object-oriented displays for each library are shown.

The function keys used with the DSPDKT and DSPTAP functions have the following effect on the library and object-level displays:

Key	Action
Roll Up, Roll Down	Roll forward or backward (within a list of saved libraries or saved objects) on tape or the current diskette
Enter	Display next file on same diskette or tape
CF1	Cancel diskette or tape display function
CF2	Display previous file on same diskette or tape
CF3	Display next diskette in volume
CF6	Display previous diskette in volume

The following display results when the Display saved members option is taken for an object shown in the previous display:



```
XX/XX/XX XX:XX:XX  SAVE/RESTORE VOLUME - XXXXXX  MEMBERS   XXX
File name:  XXXXXXXXXX  File type: XXX   Saved members: XXXX
Members saved:
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
      .
      .
      .

CF10-Return to list of members
```

The first line shows the date and time, identifies the volume identifier of the diskette or tape, and indicates that this is a display showing the names of the saved members. The second line of the display shows the file name and file type of the saved object, as well as the total number of members in the saved file.

Beginning with the fourth line, the names of the saved files members are shown. The function keys used on the member-level display have the following effect:

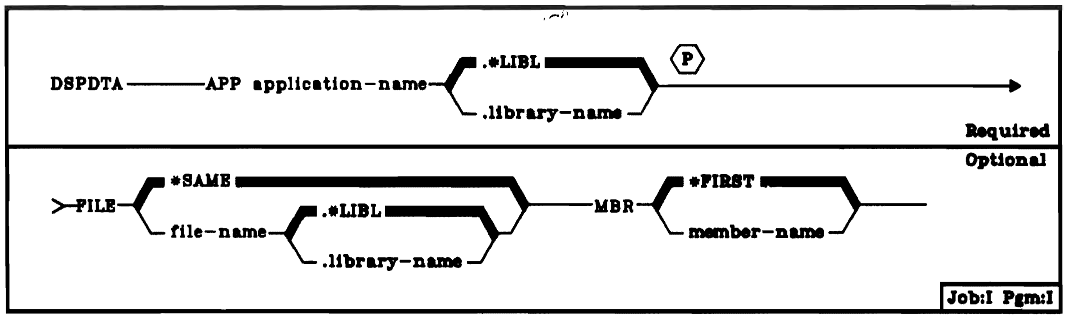
CF1	Cancel diskette or tape display function
CF2	Return to the last display on which the Enter key was pressed
CF10	Return to the object-level display

DSPDTA (Display Data) Command

The Display Data (DSPDTA) command allows you to display a data file, but you are not allowed to modify the data within it in any way. Any DFU application can be used by the DSPDTA command; it is not necessary to define the application specifically for display.

The Data File Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program*, Program 5714-UT1. For more information on the Data File Utility, refer to *IBM System/38 DFU Reference Manual and User's Guide*, SC21-7714.

Note: The first member of the file named when you define the application is displayed unless a different member is named.



APP Parameter: Specifies the qualified name of the DFU application controlling the interactive display of data. (If no library qualifier is specified, *LIBL is used to find the application.)

FILE Parameter: Specifies the name of the data base file you want to display.

<u>*SAME:</u> DFU will use the same file used to define the application.

file-name: Enter the qualified name of the data file you want DFU to display. (If no library qualifier is specified, *LIBL is used to find the file.)

MBR Parameter: Specifies which member in the file you want to display.

<u>*FIRST:</u> DFU will display the first member of the file.

member-name: Enter the name of the member you want DFU to display.

Example

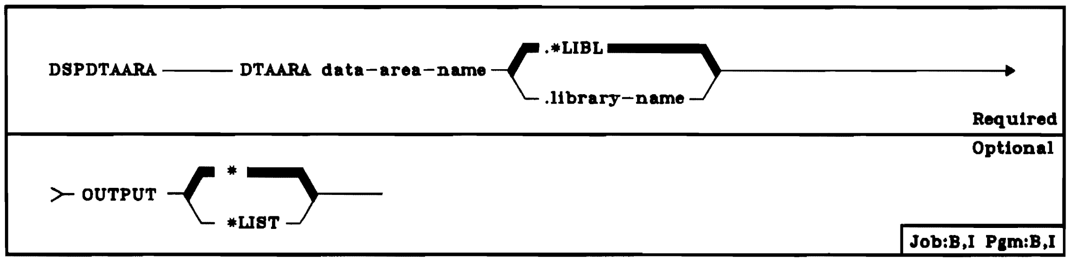
```
DSPDTA  APP(DATA.LIB1) FILE(FILEA)
```

This command displays the data in the first member of the data file FILEA of application DATA in library LIB1. If you want to add, change, or delete data in the data file, execute the CHGDTA (Change Data) command.

DSPDTAARA (Display Data Area) Command

The Display Data Area (DSPDTAARA) command displays the data attributes and value of the specified data area. The following data attributes are displayed: the type and length of the data area, and its current value. The following object attributes are also displayed: the library in which the data area is located, and the text describing the data area.

Restriction: To use this command, you must have operational rights for the data area and read rights for the library.



DTAARA Parameter: Specifies the qualified name of the data area whose attributes and value are to be displayed. (If no library qualifier is given, *LIBL is used to find the data area.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

<u>*</u>: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST*: The output is to be listed with the job's spooled output on a printer.

Example

```
DSPDTAARA  DTAARA(TIME) OUTPUT(*)
```

The value and attributes of the data area TIME are displayed to the user if he has the proper authority to display it. The library list is used to find the data area.

Additional Considerations

The display produced by the DSPDTAARA command has the following format:

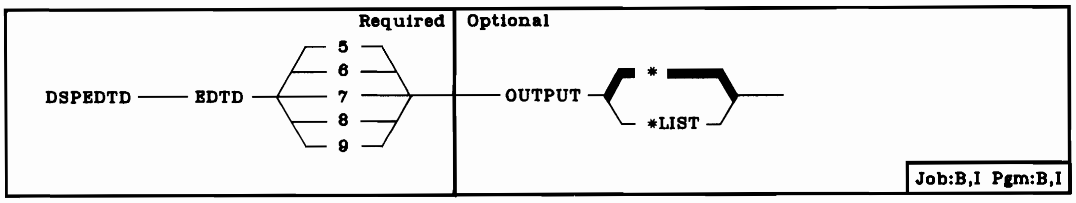
```
 XX/XX/XX  XX:XX:XX     DATA AREA DISPLAY
Data area: XXXXXXXXXX     Library: XXXXXXXXXX
Type:      XXXX           Length:  XXXX XX
Text:      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Value:    'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           XXXXXXXXXXXXXXXXXXXXXXXX...'
```

The first line of the data area display gives the current job date and time. The second line identifies the data area being displayed and the name of the library in which it is stored. The type and length attributes that were specified in the Create Data Area (CRTDTAARA) command are displayed on the third line; if TYPE(*DEC) was specified, the second portion of the length attribute gives the number of decimal positions in the value. The text specified when the data area was created is displayed on the fourth line.

The current value contained in the data area is displayed on the fifth and subsequent lines (as required for character string values). If the data area was specified as TYPE(*CHAR), the value is displayed within apostrophes on each line. The value displayed is either the initial value assigned on the CRTDTAARA command, or the changed value assigned by the most recent Change Data Area (CHGDTAARA) or Send Data Area (SNDDTAARA) command.

DSPEDTD (Display Edit Description) Command

The Display Edit Description (DSPEDTD) command displays information about the specified user-defined edit description.



EDTD Parameter: Specifies the single-digit code (5, 6, 7, 8, or 9) that identifies the user-defined edit description to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPEDTD  EDTD(6)
```

This command displays the user-defined edit description 6 on either a printer or a display.

Additional Considerations

The display produced by the DSPEDTD command has the following format:

```
 XX/XX/XX  XX:XX:XX    EDIT CODE DESCRIPTION
Edit code: X    Edit object name: XXXXXX   Library: XXXXXXXXXX
Decimal point: X    Edit zero values: XXX   Fill character: X
Currency symbol: XXXXXXXXXXXXXXX
Integer mask:    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Fraction mask:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Negative status: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Positive status: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Left constant:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Right constant:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Text:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

The second line of the display gives the number of the edit code description, its name, and the name of the library (always QSYS) in which it is stored.

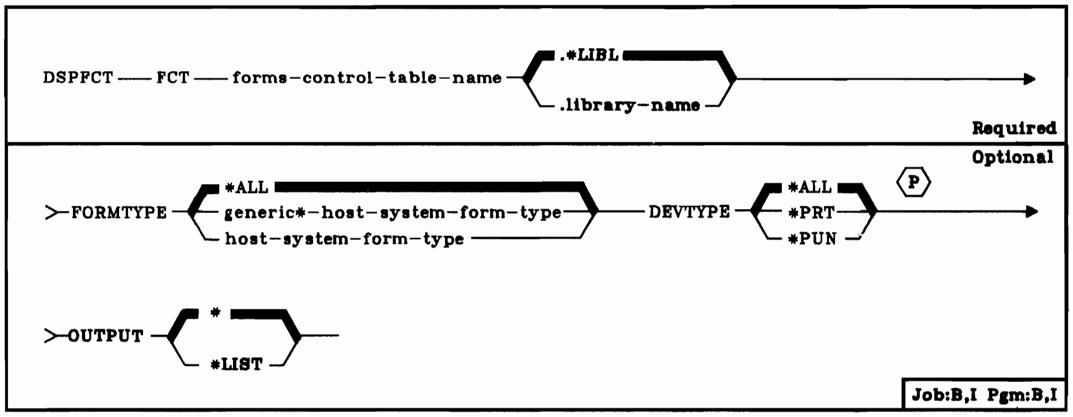
For an explanation of each edit description attribute (starting with the third line), refer to the associated parameter description given in the Create Edit Description (CRTEDTD) command description; for example, the decimal point attribute is explained in the DECPNT parameter description.

DSPFCT (Display Forms Control Table) Command

The Display Forms Control Table (DSPFCT) command displays the entries in the forms control table (FCT). Output can be displayed or printed.

Restriction: To use this command, you must have read rights for the FCT and the library in which the FCT is stored.

The Delete Forms Control Table (DLTFCT) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



FCT Parameter: Specifies the qualified name of the FCT to be displayed. (If no library qualifier is given, *LIBL is used to find the FCT.)

FORMTYPE Parameter: Specifies the form type to be displayed.

<u>*ALL:</u> All forms types for the specified device type are to be displayed.

generic-host-system-form-type:* Enter the generic host system form type of the entries that are to be displayed. To specify a generic host system form type, add an asterisk after the last character in the generic name (ABC*, for example). If an * is not included with the name, the system assumes that the name is complete.

host-system-form-type: Enter the host system form type of the entry to be displayed.

DEVTYPE Parameter: Specifies the device type for which the specified forms type is to be displayed.

**ALL:* The specified forms type for all device types is to be displayed.

**PRT:* The specified forms type for printer device types is to be displayed.

**PUN:* The specified forms type for punch device types is to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer.

**:* The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer. The IBM-supplied device file QPDSPFCT is used to print the FCT.

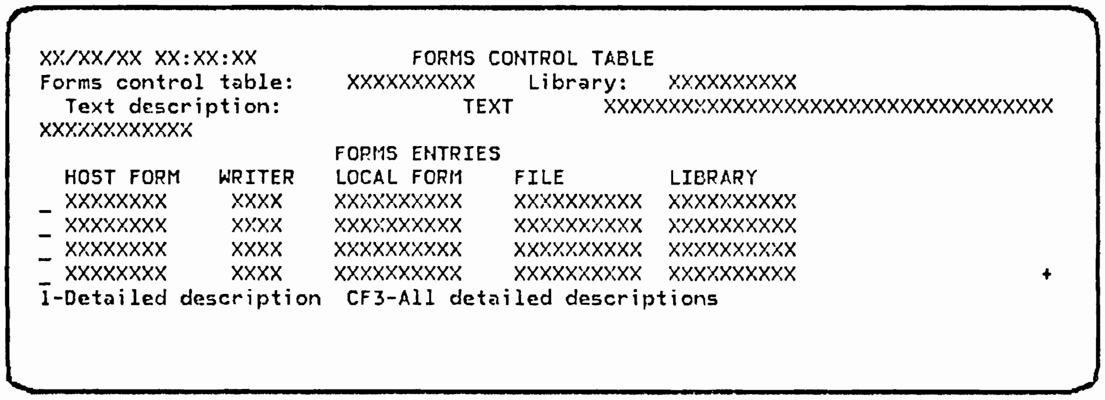
Example

```
DSPFCT  FCT(FORMCTRL.USERLIB) +
      FORMTYPE(MED*) +
      DEVTYPE(*PRT) +
      OUTPUT(*)
```

This command (if entered from a batch job) causes selected forms control entries to be sent to the job's spooling queue to be printed. If the command is entered in an interactive job, a menu of selected forms control entries are displayed from which detail selections can be made. The menu displayed will contain all printer entries beginning with the characters MED.

Additional Considerations

The FCT displays are in two sets. Execution of the Display Forms Control Table (DSPFCT) command causes the following display to appear:



```
XX/XX/XX XX:XX:XX          FORMS CONTROL TABLE
Forms control table:    XXXXXXXXXX    Library:   XXXXXXXXXX
  Text description:              TEXT       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX
                             FORMS ENTRIES
  HOST FORM    WRITER    LOCAL FORM    FILE        LIBRARY
_ XXXXXXXX      XXXX     XXXXXXXXXX    XXXXXXXXXX  XXXXXXXXXX
_ XXXXXXXX      XXXX     XXXXXXXXXX    XXXXXXXXXX  XXXXXXXXXX
_ XXXXXXXX      XXXX     XXXXXXXXXX    XXXXXXXXXX  XXXXXXXXXX
_ XXXXXXXX      XXXX     XXXXXXXXXX    XXXXXXXXXX  XXXXXXXXXX                 +
1-Detailed description  CF3-All detailed descriptions
```

This display shows information that identifies the FCT. It also presents a listing of FCT entries that can be selected for detailed display, either by entering a 1 in the first column for one or more entries or by using the CF3 key for a progression of detailed displays for all entries in the FCT. When you are displaying the FCT entries, use the CF2 key to return to a previous display, the CF10 key to return to the FCT entry selection list, and the CF1 to terminate the DSPFCT command.

Each line of the forms entry selection list portion of the display presents information about each FCT entry. This information includes the type of form that the host requests to be used with the entry, and the type of writer used, which is indicated by *PRT for print and *PUN for punch. The identifier to be used in place of the host form type is displayed as the local form type, and the name and library of the device or data base file to be used to receive host data is shown last for each entry.

The following three displays show the information presented when selecting an FCT entry for detailed display. For more information on each value, refer to the Add Forms Control Table Entry (ADDFCTE) command.

```
XX/XX/XX XX:XX:XX          FORMS CONTROL TABLE ENTRY
Host form type:          XXXXXXXX
Forms control table:     XXXXXXXXXX    Library:   XXXXXXXXXX
  Host writer type:               DEVTYPE    XXXX
  Local form type:                LCLFORM    XXXXXXXXXX
  Channel values:                 CHLVAL     XXXXX
    Channel line equivalences (channel - line):
    1-XXX    3-XXX    5-XXX    7-XXX    9-XXX    11-XXX
    2-XXX    4-XXX    6-XXX    8-XXX   10-XXX    12-XXX
  Form size (length  width):      FORMSIZE   XXXXX XXX                     +
CF10-Return to selection list
```

Note: A zero (0) for the line value means no line number is associated with that channel in the FCT entry. A value of zero is not allowed when adding or changing FCT entries.

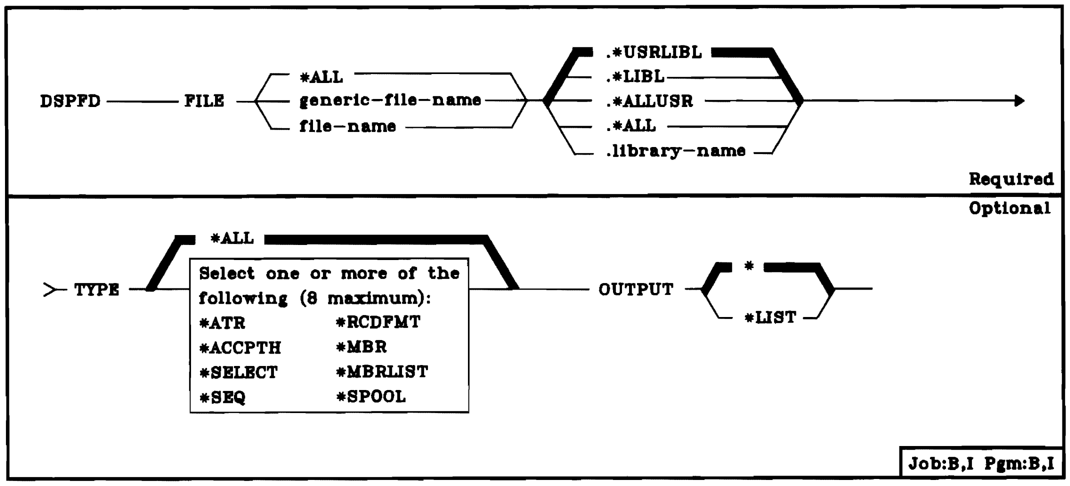
```
XX/XX/XX XX:XX:XX          FORMS CONTROL TABLE ENTRY
Host form type:          XXXXXXXX
Forms control table:     XXXXXXXXXX    Library:   XXXXXXXXXX
  Lines per inch:                 LPI        XXXXX
  Characters per inch:            CPI        XXXXX
  Print image name:               PRTIMG     XXXXXXXXXX
    Library name:                              XXXXXXXXXX
  Number of copies:               COPIES     XXXXX
  File name:                      FILE       XXXXXXXXXX
    Library name:                              XXXXXXXXXX                  +
CF10-Return to selection list
```

```
XX/XX/XX XX:XX:XX          FORMS CONTROL TABLE ENTRY
Host form type:          XXXXXXXX
Forms control table:     XXXXXXXXXX    Library:   XXXXXXXXXX
  Member name:                    MBR        XXXXXXXXXX
  File sequence number:           FSN        XXX
  Data format:                    DTAFMT     XXXXX
  User program name:              PGM        XXXXXXXXXX
    Library name:                              XXXXXXXXXX
  Message queue name:             MSGQ       XXXXXXXXXX
    Library name:                              XXXXXXXXXX
CF10-Return to selection list
```

DSPFD (Display File Description) Command

The Display File Description (DSPFD) command displays one or more types of information that is retrieved from the file descriptions of one or more data base and/or device files. The information is displayed for each file that has the specified name and that is found in all the libraries named in the specified library list to which the user has access. The information can be displayed or printed.

Restrictions: Before you can display each file specified, you must have operational authority for or own the file. Also, of the libraries specified by the library qualifier, only the libraries for which you have read rights are searched for the files.



FILE Parameter: Specifies the qualified name of the file or the generic name of several files whose descriptions are to be displayed or printed. Or, this parameter can specify that all files in the specified library or libraries (*ALL.*LIBL for example), or all files in all libraries (*ALL.*ALL), are to have their descriptions displayed. Only the libraries in the specified library qualifier that the user either owns or is authorized to use are searched for the file. All files found within the libraries listed in the specified library list that have the specified file name are displayed.

Depending on the library qualifier specified or assumed, the following libraries (for which the user has the authority) are to be searched for the specified file:

- .*USRLIBL (user library list). Only the libraries listed in the user portion of the job's library list. If a specific file name is given, only the first file found by that name is displayed.

- *.*LIBL* (library list). All the libraries in the user *and* system portions of the job's library list. If a specific file name is given, only the first file found by that name is displayed.

- .**ALLUSR* (all user libraries). All the nonsystem libraries, which include *all* user-defined libraries and the QGPL library, not just those in the job's library list. Libraries other than QGPL that begin with the letter Q are not included.

- .**ALL* (all libraries). All the libraries in the system, including QSYS.

- .*library-name* (one library). Only the library named in this parameter. The user must have read rights for the specified library.

**ALL:* All files in the specified library (or all libraries identified in the library qualifier to which the user has access) are to have their descriptions displayed.

qualified-generic-file-name: Enter the qualified name of the file or the generic name of several files in the specified library qualifier that are to have their descriptions displayed. To specify a generic file name, add an asterisk (*) at the end of the characters that are in the names of all the files desired.

TYPE Parameter: Specifies the types of information about the specified file that are to be printed or displayed. If *ALL is not specified, one or more of the other values can be specified.

Note: In the following type descriptions, the type of file that the value can be specified for is indicated as follows: P represents physical files, L represents logical files, and D represents device files.

**ALL:* (P,L,D) All of the following types of information that are applicable to the specified file are displayed.

**ATR:* (P,L,D) The file attributes of the specified file are to be displayed. For data base files, the attributes include the access path type and the maximum number of members.

**ACCPTH:* (P,L) The access paths of the specified file are to be displayed. For keyed access paths, the composite key description is also displayed. Refer to *Additional Considerations* at the end of the CRTLF command description for more information about displaying the access paths of a logical file.

**SELECT:* (L) The select/omit specifications for the specified logical file are to be displayed.

**SEQ:* (P,L) The collating sequence specification for the specified physical or logical file is to be displayed.

**RCDFMT:* (P,L,D) The record format names and record format level information about the specified file is to be displayed. This includes the record format name(s) and data association information for referenced files.

**MBR:* (P,L) Information about the members in the specified file is to be displayed. The member names, creation dates, sizes, types, and other attributes are displayed.

**MBRLIST:* (P, L) A list is to be displayed containing the names of all the members in the specified file and containing a brief description of each member.

**SPOOL:* (D) The spooling attributes of the specified device file are to be displayed. The spooling attributes include the number of output copies produced (if card or printer), the output priority, and the maximum number of records in the file.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

The following examples assume that the commands are entered at a display work station and the user of the command is authorized to access all relevant libraries and objects.

```
DSPFD  FILE(FILE1.*ALL)
```

This command displays the definition of FILE1 as defined in all libraries authorized for the user. The information is displayed at the work station initiating the command.

```
DSPFD  FILE(FILE1.LIBRARY1)
```

This command displays the definition of FILE1 as defined in LIBRARY1. The information is displayed at the work station initiating the command.

Additional Considerations

When the DSPFD command is entered, the data base is searched for the file or files specified by the FILE parameter; then a group of records that give file-level information about each file is generated. The records are placed in the printer device file named QPDSPFD. If OUTPUT(*LIST) is specified on the command, the records are listed on the printer in the following order:

1. On the first line, the leftmost fields show system identifier digits. The right side shows the date and time the job executed and the page number.

2. The file, library, and type of information are shown (page 1 only).

3. The specific file, library, and type of file (to which the following information applies) are shown.

4. Header information that identifies the type of information being presented is shown, followed by the attributes.

If more than one file is to have its attributes displayed or printed, the *basic* attributes of the next file are displayed only after all of the (specified) attributes for the first file have been displayed. For printed output, a separate page is used for each type of information that is printed, as indicated by the TYPE parameter.

For displayed or printed output, an appropriate indication is given under the section heading for that type of display if no detailed information exists for the specified type. For example, if a logical file is being displayed and TYPE(*MBR *MBRLIST) was specified, but no members have yet been defined for the file, *(no members in file)* is displayed or printed under the section headings for the *member attributes display* and for the *member list display*.

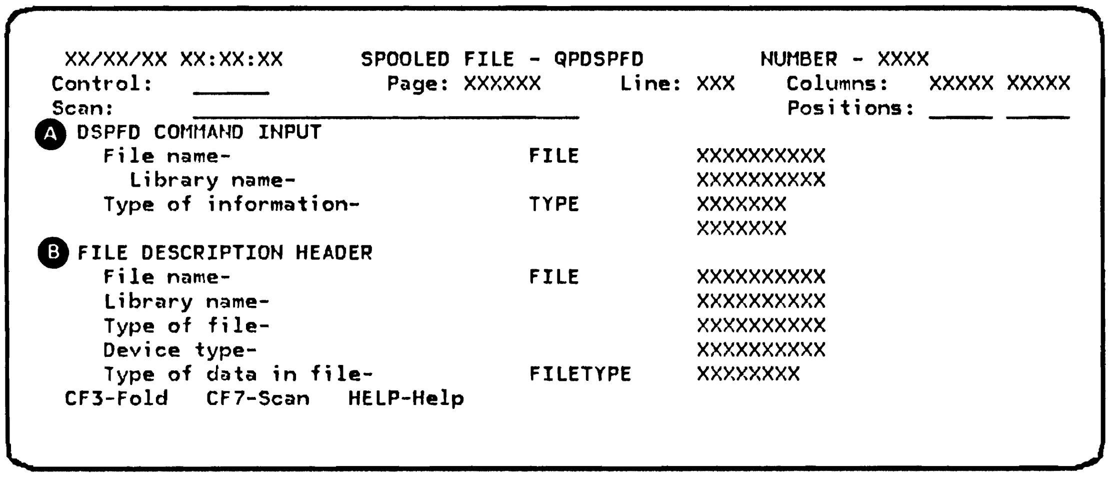
For displayed output, the first three lines of each display contain information about the spooled file and fields used for display handling functions. For more information on the uses and meanings of these fields, refer to the *Additional Considerations* section of the Display Spooled File (DSPSPLF) command.

The information described on the following series of displays shows all of the *potentially* displayable attributes for each file type (physical, logical, or device) and each attribute type.

Displayed Output

If the DSPFD command is entered interactively and OUTPUT(*) is specified or assumed, the records in the printer device file are displayed rather than printed. Also, additional information that appears on the first three lines of each display is generated and updated for every display presented to the user who entered the command.

- The current job date and time
- The name of the printer device file (QPDSPFD) into which all records containing the file descriptions are placed after they are generated
- The spooled file number of the spooled printer file created by the DSPFD command
- The page, line, and column numbers of the spooled file being shown



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:    _____         Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:       ______________________________               Positions: _____ _____
Ⓐ DSPFD COMMAND INPUT
    File name-                          FILE         XXXXXXXXXX
      Library name-                                  XXXXXXXXXX
    Type of information-                TYPE         XXXXXXX
                                                     XXXXXXX
Ⓑ FILE DESCRIPTION HEADER
    File name-                          FILE         XXXXXXXXXX
    Library name-                                    XXXXXXXXXX
    Type of file-                                    XXXXXXXXXX
    Device type-                                     XXXXXXXXXX
    Type of data in file-               FILETYPE     XXXXXXXX
 CF3-Fold   CF7-Scan   HELP-Help
```

Ⓐ DSPFD Command Input

Line 4 of the first display gives the current job date and the name of the displayed information. (This is page 1, line 1 in the printer device file.) Lines 5 through 8 of the first display show the values specified on the DSPFD command when it was entered. If multiple attribute types were specified on the TYPE parameter, they are listed under each other (on separate lines).

Ⓑ File Description Header

The following basic information is shown for each file whose attributes are to be displayed. It includes:

- The name of the file, and the name of the library in which the file is stored.
- The type of file: physical, logical, or a device file.
- If the file is a device file, the type of device.
- If the file type can be a source file, whether the file is a source file or a data file.

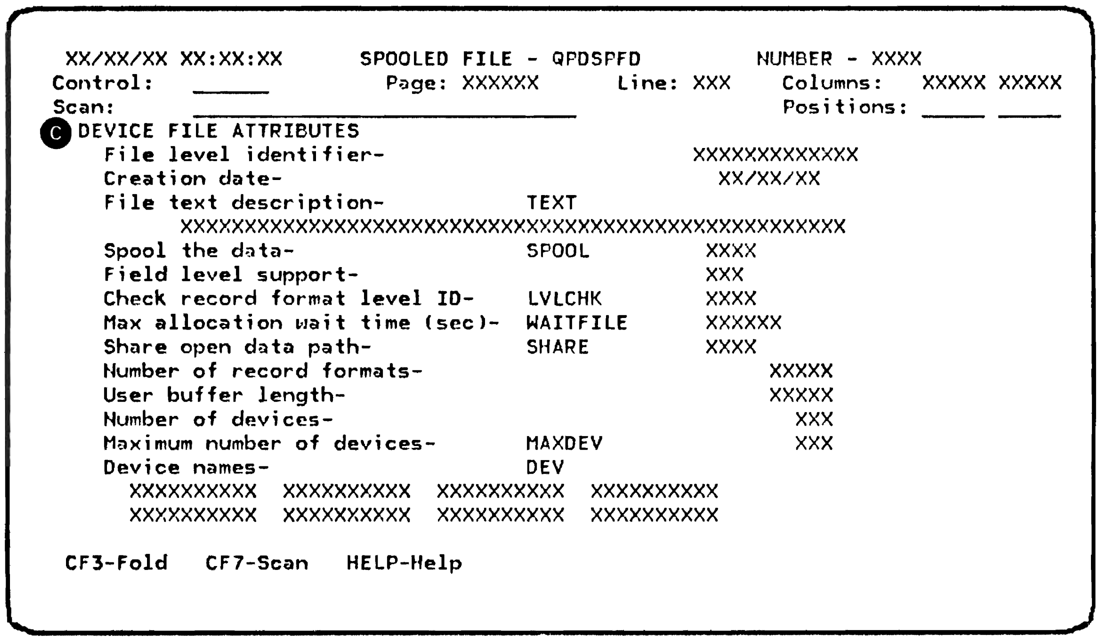
C Device File Attributes (Common to All Device Files)

If the specified file is a device file and either *ATR or *ALL is specified on the TYPE parameter of the DSPFD command, the attributes are displayed in two logical groups:

- The attributes that are common to all types of device files. The same common attributes display is used for any device file being displayed.

- The remaining device attributes (excluding spooling attributes) unique to that device type are then displayed; for example, LABEL, CODE, and LOC.

The common device file attributes are displayed first:



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:  _____          Page: XXXXXX     Line: XXX    Columns:  XXXXX XXXXX
Scan:     ________________________________              Positions: ____ ____
C DEVICE FILE ATTRIBUTES
    File level identifier-                          XXXXXXXXXXXXX
    Creation date-                                    XX/XX/XX
    File text description-          TEXT
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    Spool the data-                 SPOOL          XXXX
    Field level support-                           XXX
    Check record format level ID-   LVLCHK         XXXX
    Max allocation wait time (sec)- WAITFILE       XXXXXX
    Share open data path-           SHARE          XXXX
    Number of record formats-                           XXXXX
    User buffer length-                                 XXXXX
    Number of devices-                                    XXX
    Maximum number of devices-      MAXDEV                XXX
    Device names-                   DEV
      XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
      XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX

 CF3-Fold   CF7-Scan   HELP-Help
```

Additional information on some of the attributes shown are:

- Field level support: Whether the device file has field-level support (if the file has a record format containing defined fields).

- Number of record formats: The number of record formats defined for the device file.

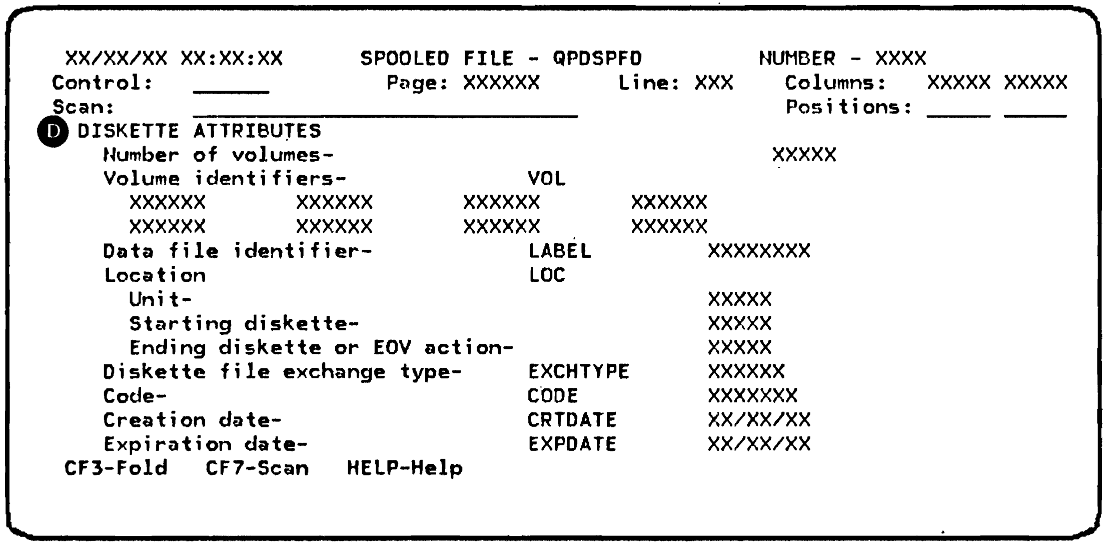
- User buffer length: The number of bytes in the user's input/output buffer associated with this file.

- Number of devices: The number of devices that are associated with this device file. The name of each one (that is, the device descriptions) are displayed under *device names.*

- Maximum number of devices: The number of devices that can be associated with this device file. Refer to the CRTDSPF command for an explanation of the MAXDEV parameter.

D Device-Dependent File Attributes

The remaining device attributes that are not related to spooling are then displayed. As an example, the following display shows the attributes for a diskette device file:



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD         NUMBER - XXXX
Control:  _____             Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________                 Positions: _____ _____
D DISKETTE ATTRIBUTES
    Number of volumes-                                    XXXXX
    Volume identifiers-                VOL
      XXXXXX       XXXXXX       XXXXXX       XXXXXX
      XXXXXX       XXXXXX       XXXXXX       XXXXXX
    Data file identifier-              LABEL         XXXXXXXX
    Location                           LOC
      Unit-                                          XXXXX
      Starting diskette-                             XXXXX
      Ending diskette or EOV action-                 XXXXX
    Diskette file exchange type-       EXCHTYPE      XXXXXX
    Code-                              CODE          XXXXXXX
    Creation date-                     CRTDATE       XX/XX/XX
    Expiration date-                   EXPDATE       XX/XX/XX
 CF3-Fold   CF7-Scan   HELP-Help
```

E Data Base File Attributes (Physical and Logical Files)

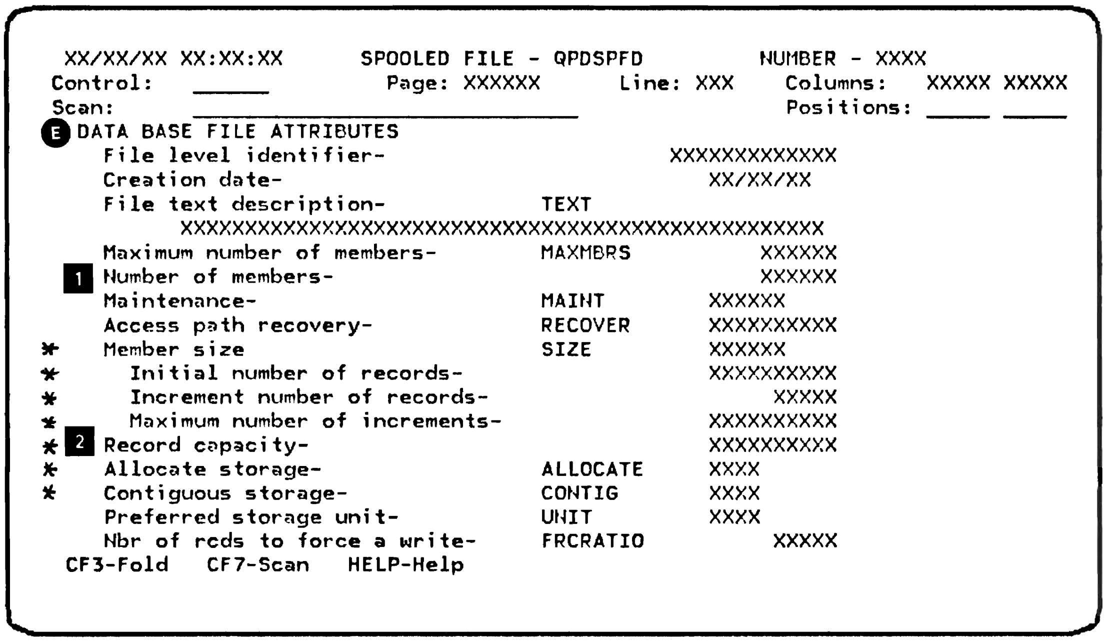
If *ATR or *ALL is specified on the TYPE parameter of the DSPFD command and if the file is a data base file, the *data base file attributes* of the specified file are displayed on one or more displays. If the data base file is a physical file, the following displays are presented. Those attributes shown on the following display that do not apply to the specified file are not shown.

Note: Because many of the attributes shown in the following physical file displays also occur on the logical file displays, only the physical file attributes are shown as a *complete* set. An * is used at the left edge of each line that shows an attribute that is unique to the physical file display. All lines having no * can appear on both physical or logical file displays. (Those attributes unique to a logical file display are presented later.)

Also note that some attributes come from the file description itself (identified by its associated command parameter keyword); others come from DDS specifications (those specified by DDS keywords are identified by *DDS* after its DDS keyword, the rest are not); and other attributes are determined by the system when it generates all the records placed in the printer device file QPDSPFD or when the file was created.

For the description of those attributes identified with either command or DDS keywords, refer to the appropriate command description (such as the CRTPF command) or to the *CPF Reference Manual—DDS* for the DDS keyword descriptions. The other attributes are described here only if an explanation is necessary.

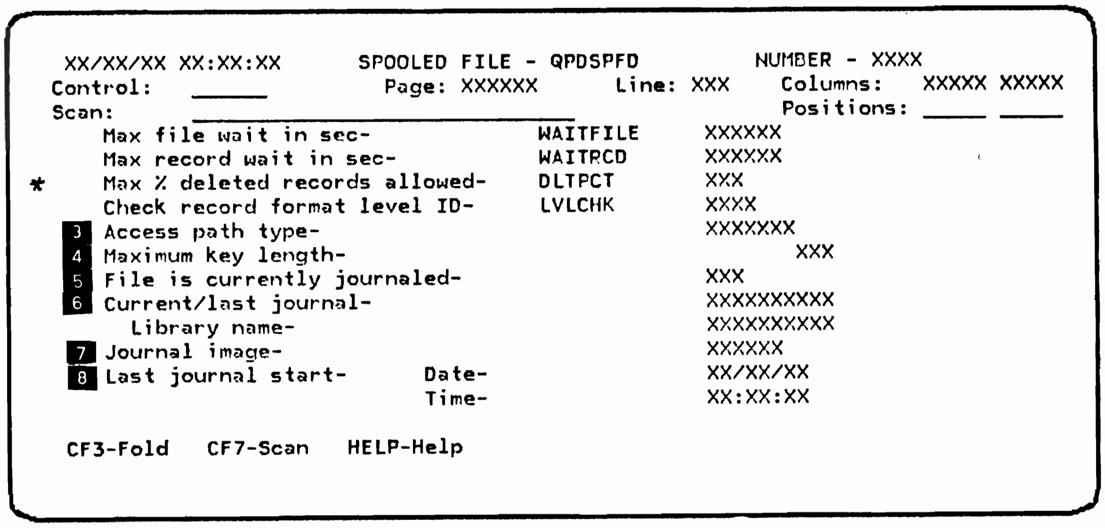
For each physical file, the attributes on the next two displays can appear:



```
   XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
  Control:  ______         Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
  Scan:            ____________________________           Positions: _____ _____
E DATA BASE FILE ATTRIBUTES
     File level identifier-                           XXXXXXXXXXXXX
     Creation date-                                      XX/XX/XX
     File text description-         TEXT
       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
     Maximum number of members-     MAXMBRS                XXXXXX
   1 Number of members-                                    XXXXXX
     Maintenance-                   MAINT            XXXXXX
     Access path recovery-          RECOVER          XXXXXXXXXX
*    Member size                    SIZE             XXXXXX
*      Initial number of records-                    XXXXXXXXXX
*      Increment number of records-                       XXXXX
*      Maximum number of increments-                 XXXXXXXXXX
*  2 Record capacity-                                XXXXXXXXXX
*    Allocate storage-              ALLOCATE         XXXX
*    Contiguous storage-            CONTIG           XXXX
     Preferred storage unit-        UNIT             XXXX
     Nbr of rcds to force a write-  FRCRATIO              XXXXX
   CF3-Fold   CF7-Scan   HELP-Help
```

1 Number of members: The current number of members in the file

2 Record capacity: The total number of records that can be placed in each member in the file, based on the values specified in the SIZE parameter of the CRTPF command



```
   XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
  Control:    ______          Page: XXXXXX      Line: XXX   Columns:   XXXXX XXXXX
  Scan:       ______________________________                Positions: _____ _____
      Max file wait in sec-              WAITFILE      XXXXXX
      Max record wait in sec-            WAITRCD       XXXXXX
*     Max % deleted records allowed-     DLTPCT        XXX
      Check record format level ID-      LVLCHK        XXXX
   3  Access path type-                                XXXXXXX
   4  Maximum key length-                                     XXX
   5  File is currently journaled-                     XXX
   6  Current/last journal-                            XXXXXXXXXX
        Library name-                                  XXXXXXXXXX
   7  Journal image-                                   XXXXXX
   8  Last journal start-       Date-                  XX/XX/XX
                                Time-                  XX:XX:XX

   CF3-Fold   CF7-Scan   HELP-Help
```

3 Access path type: Indicates whether the file is accessed in arrival sequence, is accessed in keyed sequence, or is shared with another file

4 Maximum key length: Indicates, for keyed files only, the maximum length of the composite key when all of the key fields defined for the file are used

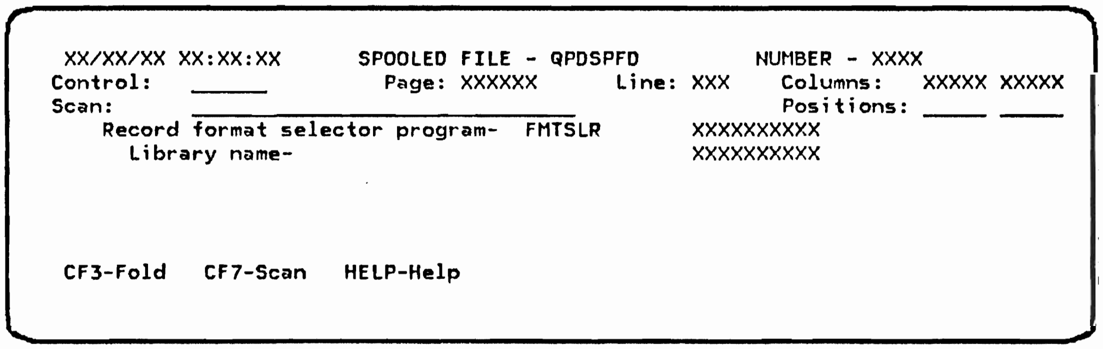
5 File is currently journaled: Indicates whether the file is being journaled

6 Current/last journal: Shows the qualified name of the journal to which changes are being sent, or the name of the journal to which changes were sent when the file was last journaled

7 Journal image: Indicates whether after images or both before and after images are being generated for changes to records in this file

8 Last journal start: Shows the date and the time on which journaling was lasted started

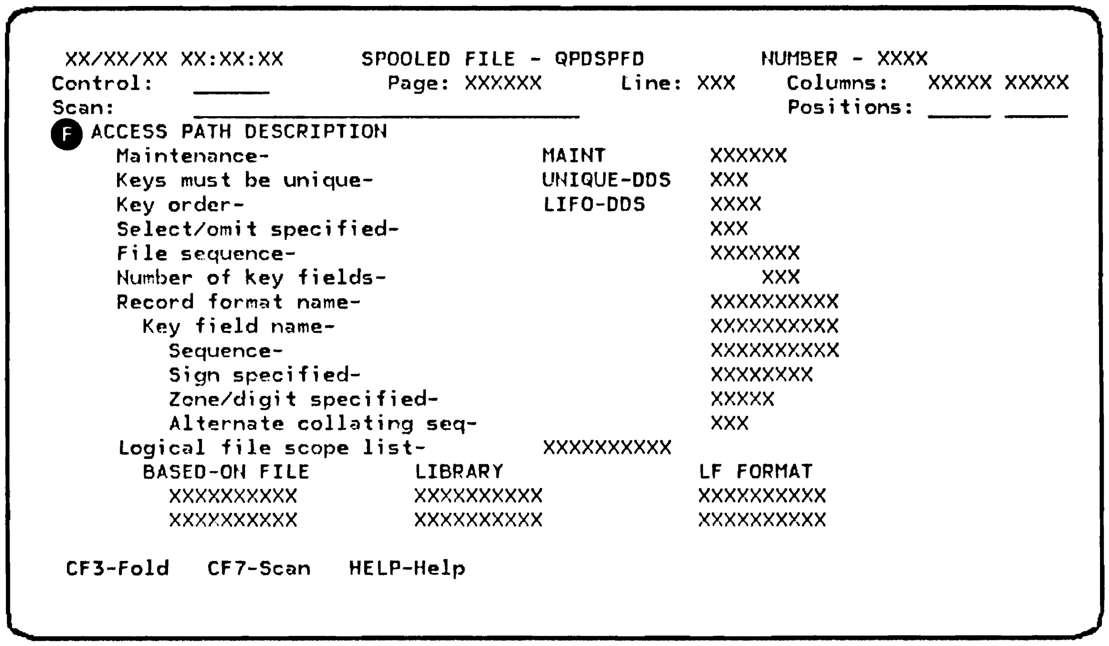
Logical File Attributes: The following display contains the only displayable attributes unique to a *logical* file that can be displayed when TYPE(*ATR) is specified on the DSPFD command.



```
   XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
  Control:    ______          Page: XXXXXX      Line: XXX   Columns:   XXXXX XXXXX
  Scan:       ______________________________                Positions: _____ _____
      Record format selector program-  FMTSLR       XXXXXXXXXX
        Library name-                               XXXXXXXXXX



   CF3-Fold   CF7-Scan   HELP-Help
```

F Access Path Description (Physical and Logical Files)

If *ACCPTH or *ALL is specified on the TYPE parameter of the DSPFD command and if the file is a data base file, the *access path attributes* of the file are displayed. The same displays are presented for physical and logical files, but only the attributes applicable to that file type are displayed.



```
 XX/XX/XX XX:XX:XX       SPOOLED FILE - QPDSPFD         NUMBER - XXXX
Control:   ______          Page: XXXXXX      Line: XXX   Columns:   XXXXX XXXXX
Scan:      _____________________________                 Positions: _____ _____
F ACCESS PATH DESCRIPTION
     Maintenance-                          MAINT        XXXXXX
     Keys must be unique-                  UNIQUE-DDS   XXX
     Key order-                            LIFO-DDS     XXXX
     Select/omit specified-                             XXX
     File sequence-                                     XXXXXXX
     Number of key fields-                                XXX
     Record format name-                                XXXXXXXXXX
       Key field name-                                  XXXXXXXXXX
         Sequence-                                      XXXXXXXXXX
         Sign specified-                                XXXXXXXX
         Zone/digit specified-                          XXXXX
         Alternate collating seq-                       XXX
     Logical file scope list-              XXXXXXXXXX
       BASED-ON FILE        LIBRARY                    LF FORMAT
         XXXXXXXXXX         XXXXXXXXXX                 XXXXXXXXXX
         XXXXXXXXXX         XXXXXXXXXX                 XXXXXXXXXX

 CF3-Fold    CF7-Scan    HELP-Help
```

Additional information on some of the attributes shown are:

- Select/omit specified: Indicates, for keyed logical files only, whether select/omit specifications were made in DDS for the file.
- File sequence: Indicates whether the file is in arrival sequence, is in a keyed sequence, or is sharing an access path with another file.
- Number of key fields (for keyed files): The maximum number of fields used for accessing records in the file by keyed sequence. Each key field is described one after the other by the attributes identified by items 5 through 9.
- Record format name: The name of the record format used by the physical or logical file. If there are multiple record formats, in the case of logical files only, the name of the next record format used in the file is displayed *after* all the access path attributes for the first record format have been displayed.
- Key field name (for keyed files): The name of the field that is used as a key field. If there are multiple key fields for this file, they are displayed here in the same order as they are used in keyed sequence access paths.
- Sequence (for keyed files): Indicates whether the values in this key field are to be accessed in ascending or descending order.

- Sign specified (for keyed files): Indicates, for key fields containing numeric values, whether the signed value, the absolute value, or a bit string (default value) is used as the key value.

- Zone/digit specified (for keyed files): Indicates, for key fields containing character or zoned values, whether neither the zoned or digit portions, only the digit portion, or only the zoned portion of each character in the key field is to be forced to zero.

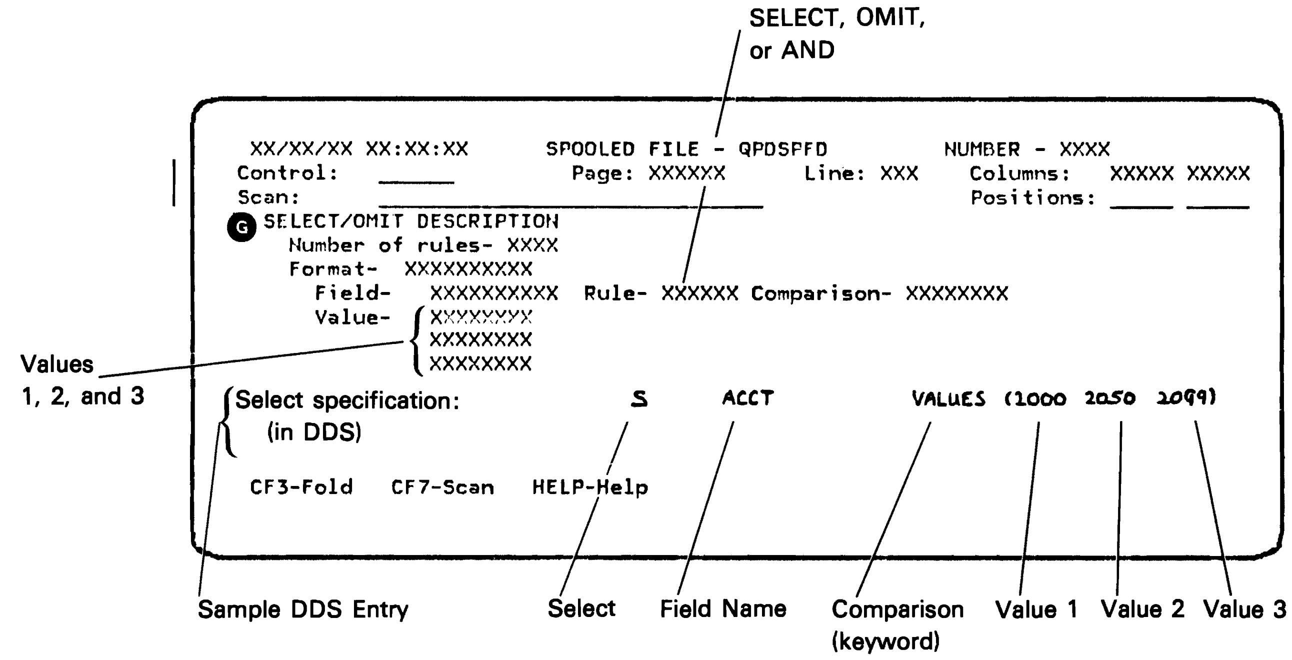
- Alternate collating sequence (for keyed files): Indicates whether an alternate collating sequence is used for this key field.

- Logical file scope list: Indicates, for logical files only, the name of each physical file on which this logical file is based, and the name of the corresponding logical file record format that is used to present data from the based-on physical file. Each based-on file is listed on separate lines and described under three column headings.

  If the logical file does not have a shared access path, the value PFILE-DDS is displayed on the line above the three column headings. If the file does have a shared access path, the value ACCPTH-DDS is displayed, and the first file listed under the column headings is the physical file whose access path is shared with the logical file.

G Select/Omit Description (Logical File)

If *SELECT or *ALL is specified on the TYPE parameter of the DSPFD command and the specified file is a logical file for which select/omit attributes were specified in DDS, the *select/omit attributes* are displayed, if any exist.



```
XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:  _____           Page: XXXXXX      Line: XXX   Columns:   XXXXX XXXXX
Scan:                                                   Positions: _____ _____
G SELECT/OMIT DESCRIPTION
    Number of rules- XXXX
    Format-  XXXXXXXXXX
      Field-   XXXXXXXXXX  Rule- XXXXXX Comparison- XXXXXXXX
      Value- { XXXXXXXX
               XXXXXXXX
               XXXXXXXX
{Select specification:            S       ACCT          VALUES (1000  2050  2099)
   (in DDS)

  CF3-Fold   CF7-Scan   HELP-Help
```

Below the display fields is a sample DDS entry that helps associate the displayed fields with a coded DDS specification.

The meanings of the fields that display the select/omit specifications for the file are:

- Number of rules: The number of comparison keywords specified for the file. This is the total for all the fields in the record format that is named on the next line.
- Format: The name of the record format to which the following rules apply.
- Field: The name of the field to which the rule or rules apply.
- Rule: The type of rule or operation (select, omit, or and) applied to the file.
  - SELECT. The record is to be selected if the field contains a value that satisfies the condition specified by the comparison value.
  - OMIT. The record is to be omitted if the field contains a value that satisfies the condition specified by the comparison value.
  - AND. If the condition specified by the comparison value on this line *and* the condition(s) on the preceding line(s) associated with the same field are all met, then the record is to be selected or omitted, whichever is specified on the preceding line.

- Comparison: The DDS keyword and op code (in the case of the CMP keyword) used with the parameter value.
- Value: One or more values that are specified with the DDS comparison keyword.

If multiple DDS keywords are used, or if multiple fields within the same record format have select/omit logic applied, then lines 5 and 6 are repeated one group after another for each keyword or field used. If multiple record formats have select/omit logic applied, then lines 4, 5, and 6 are repeated.

H Alternate Collating Sequence (Physical and Logical Files; display not shown)

If *SEQ or *ALL is specified on the TYPE parameter of the DSPFD command and the specified file is a physical or logical file that uses an alternate collating sequence, that sequence is displayed. The display contains no other information besides the actual sequence.

I Member Description (Physical and Logical Files)

If *MBR or *ALL is specified on the TYPE parameter of the DSPFD command and if the file is a data base file, the *member attributes* of the file are displayed.

The following displays show the complete set of member attributes potentially displayable for a physical file. Again, those attributes unique to a physical file member are identified by an * at the left end of their display lines; lines without an * apply to both physical and logical file members. (Attributes unique to a logical file member are presented later.)

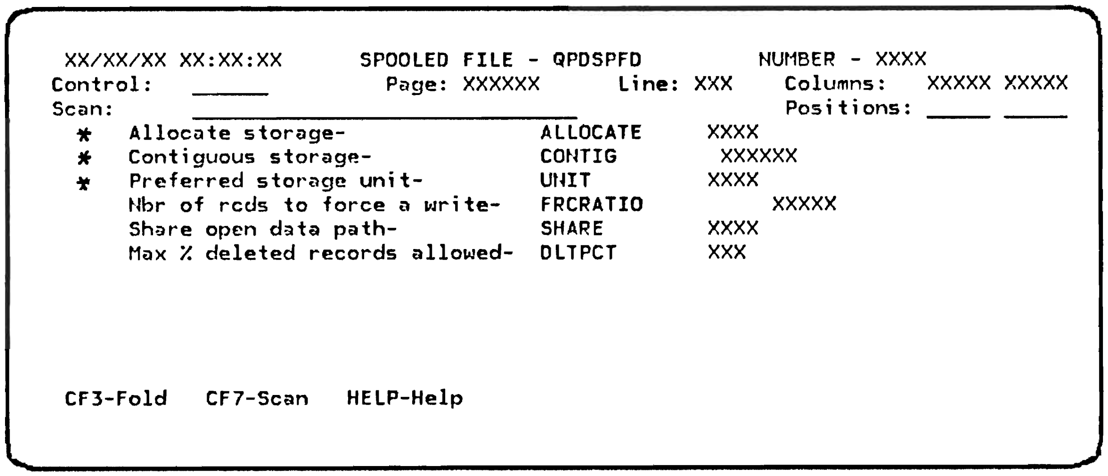
```
  XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
 Control:   ______          Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
 Scan:      ______________________________                 Positions: _____ _____
I MEMBER DESCRIPTION
    Member name-                          MBR         XXXXXXXXXX
   1  Member level identifier-                     XXXXXXXXXXXXX
      Member creation date-                           XX/XX/XX
      Text description-                   TEXT
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      Expiration date for member-         EXPDATE     XX/XX/XX
      Maintenance                         MAINT       XXXXXX
      Access path recovery                RECOVER     XXXXXXXXXX
 *    Member size-                        SIZE
 *      Initial number of records-                    XXXXXXXXXX
 *      Increment number of records-                    XXXXXXXX
 *      Maximum number of increments-                   XXXXXXXX
 *    Current number of increments-                   XXXXXXXX
 *  2 Record capacity-                                XXXXXXXXXX
 *  3 Current number of records-                      XXXXXXXXXX
 *  4 Number of deleted records-                      XXXXXXXXXX
  CF3-Fold    CF7-Scan    HELP-Help
```

1 Member level identifier: The member identifier assigned by the system to this member.

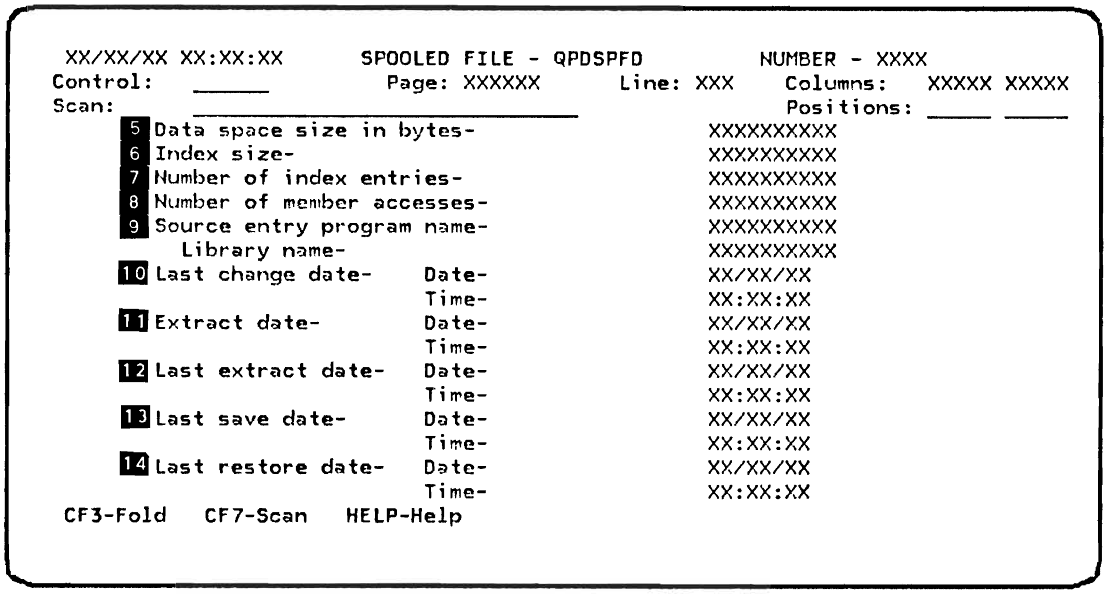
2 Record capacity: The total number of records that can be placed in this member without additional manual extents.

3 Current number of records: The total number of records currently in the member, not including deleted records.

4 Number of deleted records: The current number of deleted records in the member.



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:  ______          Page: XXXXXX      Line: XXX   Columns:   XXXXX XXXXX
Scan:     ______________________________                Positions: _____ _____
  *   Allocate storage-               ALLOCATE     XXXX
  *   Contiguous storage-             CONTIG        XXXXXX
  *   Preferred storage unit-         UNIT         XXXX
      Nbr of rcds to force a write-   FRCRATIO         XXXXX
      Share open data path-           SHARE        XXXX
      Max % deleted records allowed-  DLTPCT       XXX




 CF3-Fold   CF7-Scan   HELP-Help
```



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:  ______          Page: XXXXXX      Line: XXX   Columns:   XXXXX XXXXX
Scan:     ______________________________                Positions: _____ _____
       5 Data space size in bytes-             XXXXXXXXXX
       6 Index size-                           XXXXXXXXXX
       7 Number of index entries-              XXXXXXXXXX
       8 Number of member accesses-            XXXXXXXXXX
       9 Source entry program name-            XXXXXXXXXX
           Library name-                       XXXXXXXXXX
      10 Last change date-    Date-            XX/XX/XX
                              Time-            XX:XX:XX
      11 Extract date-        Date-            XX/XX/XX
                              Time-            XX:XX:XX
      12 Last extract date-   Date-            XX/XX/XX
                              Time-            XX:XX:XX
      13 Last save date-      Date-            XX/XX/XX
                              Time-            XX:XX:XX
      14 Last restore date-   Date-            XX/XX/XX
                              Time-            XX:XX:XX
 CF3-Fold   CF7-Scan   HELP-Help
```

5 Data space size (in bytes): The current number of bytes used in the data space (an MI system object) associated with this member.

6 Index size (for keyed files): The current number of bytes used in the data space index (an MI system object) associated with this member.

7 Number of index entries (for keyed files): The current number of index entries in the data space index.

8 Number of member accesses (for keyed files): The total number of times that records in this member have been accessed (including read, add, delete, and update operations), since the last extract (request for information about this keyed file member) was performed.

9 Source entry program name: Indicates, only if this is a source file member, the name of the source entry program used to make changes to source records in this member.

10 Last change date: The date and time when the member was last changed.

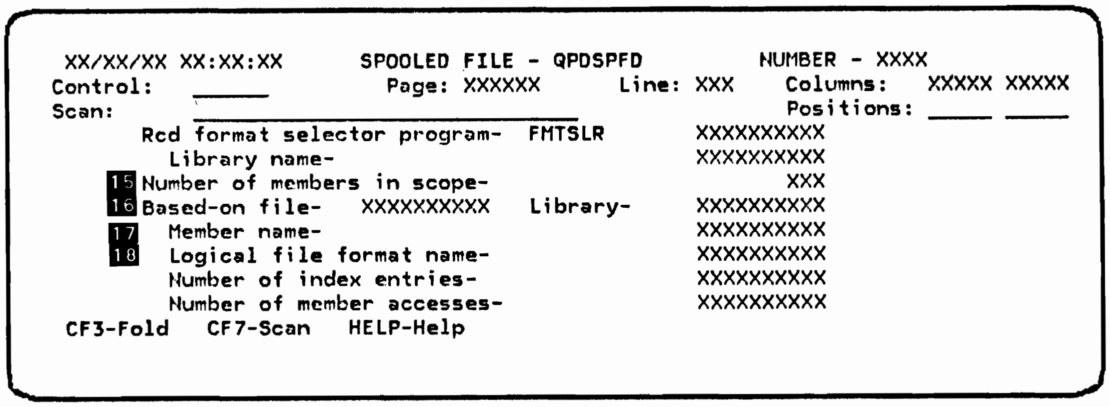
11 Extract date (for keyed files): The current date and time (when the DSPFD command was executed) when any type of attribute information about this file member was extracted (requested).

12 Last extract date (for keyed files): The date and time of the last request for information about this file member.

13 Last save date: The date and time this file member was last saved.

14 Last restore date: The date and time this file member was last restored.

Logical File Member Attributes: The following display contains the only displayable attributes unique to a *logical* file member that can be displayed when TYPE(*MBR) is specified on the DSPFD command.



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD          NUMBER - XXXX
Control:  ______          Page: XXXXXX      Line: XXX   Columns:   XXXXX XXXXX
Scan:     ______________________________                Positions: _____ _____
       Rcd format selector program-  FMTSLR       XXXXXXXXXX
         Library name-                            XXXXXXXXXX
     15 Number of members in scope-                      XXX
     16 Based-on file-    XXXXXXXXXX   Library-    XXXXXXXXXX
     17  Member name-                              XXXXXXXXXX
     18  Logical file format name-                 XXXXXXXXXX
         Number of index entries-                  XXXXXXXXXX
         Number of member accesses-                XXXXXXXXXX
 CF3-Fold   CF7-Scan   HELP-Help
```

Note: For each based-on physical file member (on which this logical member is based), the based-on file and library names, the member name, the logical file format, the number of index entries, and the number of member accesses are repeated on the display.

The number of index entries and member accesses are displayed only for keyed file members.

15 Number of members in scope: The number of members in the physical files upon which this logical member is based. The attributes identified (and two other attributes already described for a physical file member: index entry count and member access count) are repeated as a group for each based-on physical member.

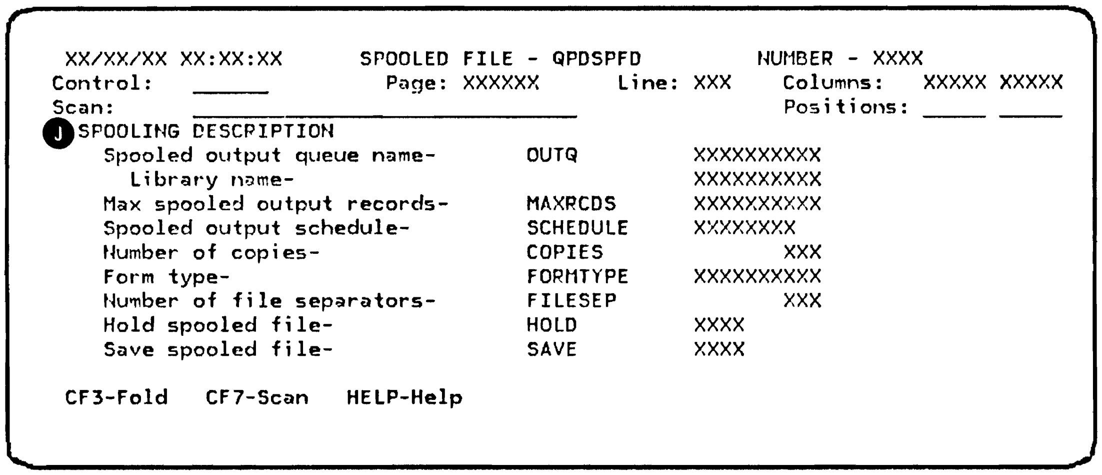
16 Based-on file: The name of the physical file, and its library name, upon which this logical member is based.

17 Member name: The name of the physical file member upon which this logical member is based.

18 Logical file format name: The name of the logical file's record format, which is made up of fields in the based-on physical file member.

J Spooling Description (Device File)

If the specified file is a device file and either *SPOOL or *ALL is specified on the TYPE parameter of the DSPFD command, the *spooling attributes* for the device file are displayed. The same display format is used for any device file having spooling attributes.



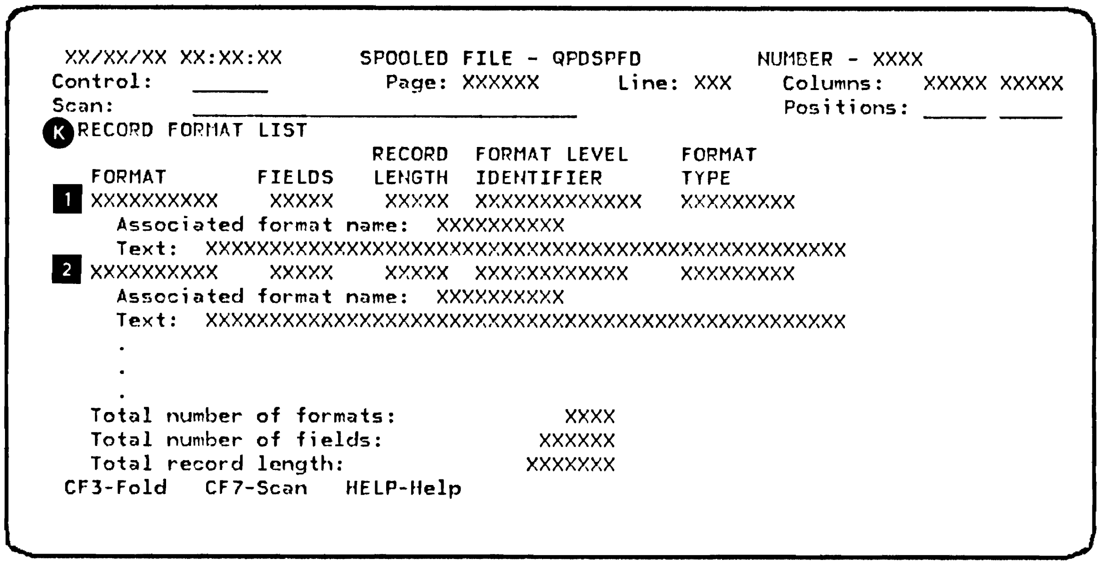
```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:  ______          Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________                Positions: _____ _____
J SPOOLING DESCRIPTION
    Spooled output queue name-         OUTQ          XXXXXXXXXX
      Library name-                                  XXXXXXXXXX
    Max spooled output records-        MAXRCDS       XXXXXXXXXX
    Spooled output schedule-           SCHEDULE      XXXXXXXX
    Number of copies-                  COPIES               XXX
    Form type-                         FORMTYPE      XXXXXXXXXX
    Number of file separators-         FILESEP              XXX
    Hold spooled file-                 HOLD          XXXX
    Save spooled file-                 SAVE          XXXX

 CF3-Fold   CF7-Scan   HELP-Help
```

For an explanation of each spooling attribute shown, refer to the associated parameter description given in the create command for that type of device file.

K Record Format List (Physical, Logical, and Device Files)

If *RCDFMT or *ALL is specified on the TYPE parameter of the DSPFD command, and the file is a display device file, the *record format list* is displayed. The same display, excluding the format type and associated format name attributes, is presented for all other file types.

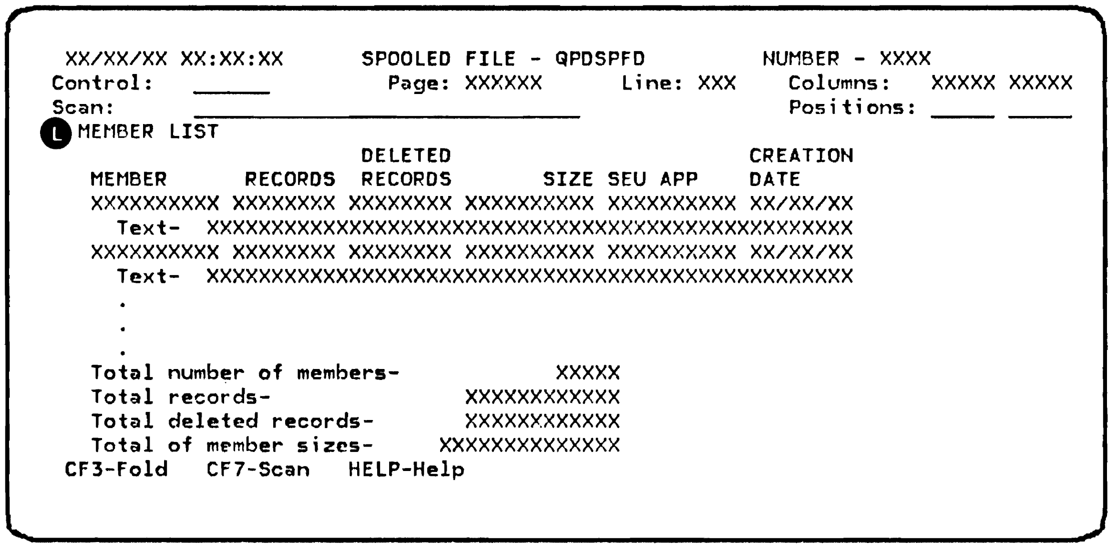
If the specified file has multiple record formats, the set of attributes for the next record format is shown following the set for the first one. For example, both formats 1 and 2 are shown in the following display.



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:  ______          Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________                Positions: _____ _____
K RECORD FORMAT LIST
                              RECORD   FORMAT LEVEL     FORMAT
   FORMAT        FIELDS      LENGTH   IDENTIFIER       TYPE
1  XXXXXXXXXX    XXXXX       XXXXX    XXXXXXXXXXXXX    XXXXXXXXX
     Associated format name:  XXXXXXXXXX
     Text:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
2  XXXXXXXXXX    XXXXX       XXXXX    XXXXXXXXXXXX     XXXXXXXXX
     Associated format name:  XXXXXXXXXX
     Text:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
     .
     .
     .
   Total number of formats:             XXXX
   Total number of fields:            XXXXXX
   Total record length:              XXXXXXX
 CF3-Fold   CF7-Scan   HELP-Help
```

L Member List (Physical and Logical Files)

If *MBRLIST or *ALL is specified on the TYPE parameter of the DSPFD command and if the file is a data base file, a *list of members* in the file is displayed, including a description of each member.



```
XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFD        NUMBER - XXXX
Control:  ______         Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________                Positions: _____ _____
L MEMBER LIST
                      DELETED                          CREATION
   MEMBER      RECORDS  RECORDS       SIZE SEU APP    DATE
   XXXXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXXXX XXXXXXXXXX XX/XX/XX
     Text-  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXXXX XXXXXXXXXX XX/XX/XX
     Text-  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      .
      .
      .
   Total number of members-           XXXXX
   Total records-              XXXXXXXXXXXX
   Total deleted records-      XXXXXXXXXXXX
   Total of member sizes-    XXXXXXXXXXXXXX
 CF3-Fold   CF7-Scan   HELP-Help
```

Two lines are used as column headings to identify the following fields. Then, for each member in the file, two lines are used to show the attributes of that member. (The text attribute extends through all of the columns.) If a field is not valid for the type of file being displayed, the field is blank.

- Member: The name of the file member.
- Records (for physical file members only): The total number of records currently in the member, not including deleted records.
- Deleted records (for physical file members only): The current number of deleted records in the member.
- Size: The current number of bytes used in the member.
- SEU application (for source file members only): The name of the source entry application program used to make changes to source records in the member.
- Creation date: The date when the member was added to the file.
- Last change date/time: This field appears in printed output only; it is never displayed. It gives the date and time when the member was last changed.
- Text: The text, describing the member, that was specified on the TEXT parameter of the ADDPFM or ADDLFM command. If there is no text describing the member, this field is blank.

After the description of the last member in the file, additional lines show totals for the entire file:

- Total number of members: The total number of members in the file.
- Total records (for physical files only): The total number of records in all members of the file, not including deleted records.
- Total deleted records (for physical files only): The total number of deleted records in all members of the file.
- Total of member sizes: The total of all the sizes (in bytes) of all members of the file.

DSPFFD (Display File Field Description) Command

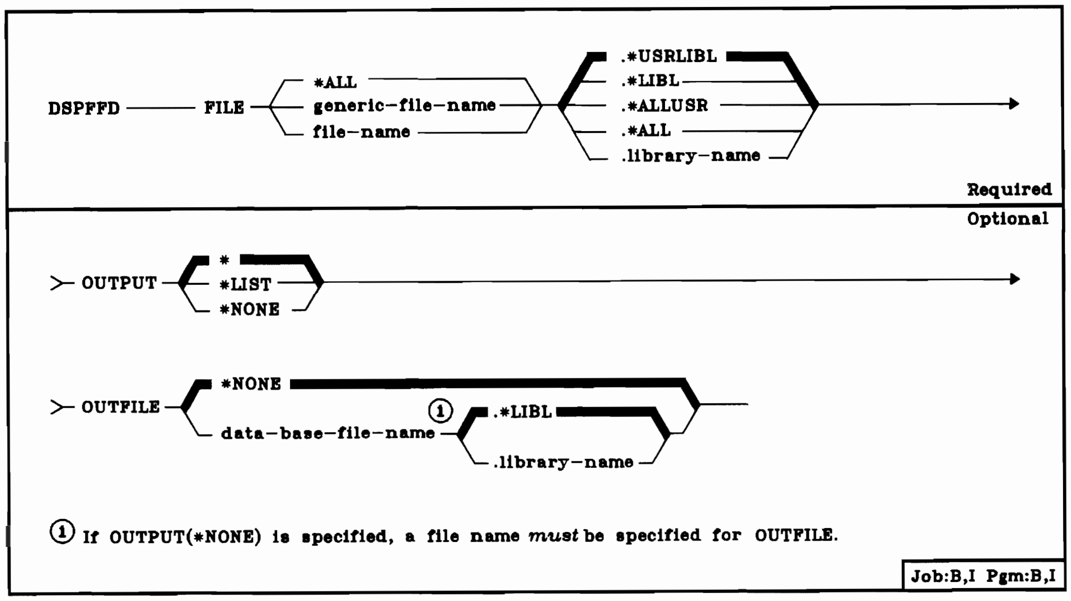
The Display File Field Description (DSPFFD) command provides field-level information for one or more files in a specific library or all the libraries to which the user has access. This command can be used to actually display or print the information, or to place the information in a data base file so that a user program can extract data from it as needed.

If the information is put in a data base file, a record composed of the following fields is produced. (The format of the record in the data base is *not* related to the format of the printed output.) The data base format is the same as that used in the IBM-supplied data base file QADSPFFD; the format is described in the *Application Documentation* chapter of the *CPF Programmer's Guide.*

- For each file specified in the command, the data base record contains:
  - The name of the file, the name of the library containing the file, and the file type.
  - The name of the record format used by the file.
  - The information retrieval date(s).

- For each field in the record format, the record also contains the following, if applicable:
  - The field name and external field name.
  - The type and length of the field.
  - For fields referencing other fields, the name of the referenced file, record format, and field. If any attributes of the referenced field were changed, the attribute type is given.
  - The edit code, edit word, and column headings associated with the field, if any.
  - An indication of whether validity checking is performed on the field.
  - For fields in device files, the I/O attribute of the field.

Restrictions: Before you can display each file specified, you must have operational authority for the file. Also, of the libraries specified by the library qualifier, only the libraries for which you have read rights are searched for the files.



FILE Parameter: Specifies the qualified name of the file or the generic name of several files, or specifies that all files in the specified library or group of libraries are to have field-level information about them displayed. A specific file name or a generic file name can be specified; either type can be optionally qualified by a library name. If no library qualifier is specified, all the libraries in the user portion of the job's library list are assumed (by *.*USRLIBL*). Only the libraries in the specified library qualifier that the user either owns or is authorized to use are searched for the file(s).

Depending on the library qualifier specified or assumed, the following libraries (for which the user has the authority) are to be searched for the files specified:

- .*USRLIBL (user library list). Only the libraries listed in the user portion of the job's library list. If a specific file name is given, only the first file found by that name is displayed.

- *.*LIBL* (library list). All the libraries in the user *and* system portions of the job's library list. If a specific file name is given, only the first file found by that name is displayed.

- *.*ALLUSR* (all user libraries). All the nonsystem libraries, which include *all* user-defined libraries and the QGPL library, not just those in the job's library list. Libraries other than QGPL that begin with the letter Q are not included.

- *.*ALL* (all libraries). All the libraries in the system, including QSYS.

- *.library-name* (one library). Only the library named in this parameter. The user must have read rights for the specified library.

**ALL:* All files in the specified library (or all libraries identified in the library qualifier to which the user has access) are to have field-level information about them displayed.

qualified-generic-file-name: Enter the qualified name of the file or the generic name of several files in the specified library qualifier that are to have field-level information about them displayed. To specify a generic file name, add an asterisk (*) at the end of the characters that are in the names of all the files desired.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the names of the files used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

**NONE:* The only output is to be to the data base file specified in OUTFILE.

OUTFILE Parameter: Specifies the name of the data base file to which the displayed information is to be sent and stored. If the specified file does not exist, this command causes a data base file and member to be created.

*NONE: No data base file is specified because the output is not to be stored in the data base.

qualified-data-base-file-name: Enter the qualified name of the data base file in which the displayed field level information is to be stored. (If no library qualifier is given, *LIBL is used to find the file.) If no file is found by that name, a file and member by that name are created and stored in the specified library, or in QGPL if not qualified. This file can be reused when other DSPFFD commands are entered. Output always starts at the beginning of the file member. (The IBM-supplied data base file QADSPFFD *cannot* be specified.)

Example

```
DSPFFD  FILE(FILE2.LIB1)
```

This command displays the field-level information in FILE2 in LIB. The information is displayed at the work station initiating the command.

```
DSPFFD  FILE(FLDREF.QGPL) OUTPUT(*NONE) +
        OUTFILE(FLDREFX.QGPL)
```

This command puts the field level information for the file (FLDREF in QGPL) into a data base file named FLDREFX in the general purpose library. That file can then be processed by a program. For example, a program such as Query in the Interactive Data Base Utilities could be used to print the field names in the file in alphabetic order.

Additional Considerations

When the DSPFFD command is entered, the data base is searched for the file or files specified on the FILE parameter; then a group of records that give field level information about each file is generated. The records are placed in a spooled printer file named QPDSPFFD. If OUTPUT(*LIST) is specified on the command, the records are listed on the printer in the following order:

1. Header information. Lists the DSPFFD command input values.

2. File information.

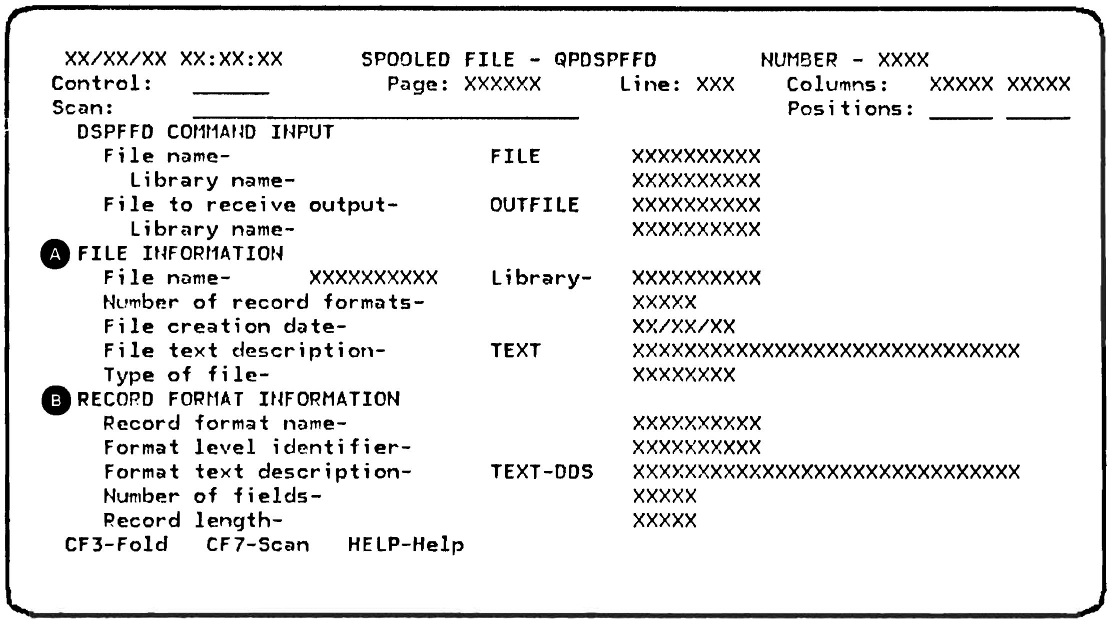
3. Record format information.

4. Field-level information. Lists, for each file identified by the FILE parameter, all of the applicable field attributes. If more than one file is identified, the beginning of the next one follows the end of the previous one.

If the DSPFFD command is entered interactively and OUTPUT(*) is specified or assumed, the records in the printer device file are displayed rather than printed. The first line of each display contains:

- The current job date and time
- The name of the spooled printer file (QPDSPFFD) into which all records containing the file field descriptions are placed after they are generated
- The spooled file number of the spooled printer file created by the DSPFFD command
- The page, line, and column numbers of the spooled file being shown

The format given in the following series of displays is used to display all of the file field attributes that exist for the file or files specified on the DSPFFD command. The header information, file information, and record format information are presented on the first display.



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFFD       NUMBER - XXXX
Control:  ______          Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________                 Positions: _____ _____
  DSPFFD COMMAND INPUT
    File name-                    FILE        XXXXXXXXXX
      Library name-                           XXXXXXXXXX
    File to receive output-       OUTFILE     XXXXXXXXXX
      Library name-                           XXXXXXXXXX
Ⓐ FILE INFORMATION
    File name-        XXXXXXXXXX  Library-    XXXXXXXXXX
    Number of record formats-                 XXXXX
    File creation date-                       XX/XX/XX
    File text description-        TEXT        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    Type of file-                             XXXXXXXX
Ⓑ RECORD FORMAT INFORMATION
    Record format name-                       XXXXXXXXXX
    Format level identifier-                  XXXXXXXXXX
    Format text description-      TEXT-DDS    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    Number of fields-                         XXXXX
    Record length-                            XXXXX
 CF3-Fold   CF7-Scan   HELP-Help
```

The first three lines contain information about the spooled file and fields used for display handling functions. For more information on the uses and meanings of these fields, refer to the *Additional Considerations* section of the Display Spooled File (DSPSPLF) command.

The displays for the DSPFFD command show file-level, record format-level, and field-level attributes. If more than one file is to be displayed, *all* of the first file's attributes (all three levels) are shown before the next file is displayed (beginning the first display and continuing through the last display).

If a file has multiple record formats, all of the attributes (record format-level and field-level) for the first record format and its fields are shown before the next record format in the same file is displayed (beginning at Ⓑ (RECORD FORMAT INFORMATION) and continuing through the last display). If a record format has multiple fields, all of the first field's attributes are shown before the next field is displayed (beginning at Ⓒ (FIELD LEVEL INFORMATION) and continuing through the last display).

For each file being displayed, the following applicable attributes are displayed in the order shown. If one or more attributes do not apply to the named file, record format, or field, those lines are not shown, and the lines that would be left blank are filled with the next attributes that do apply.

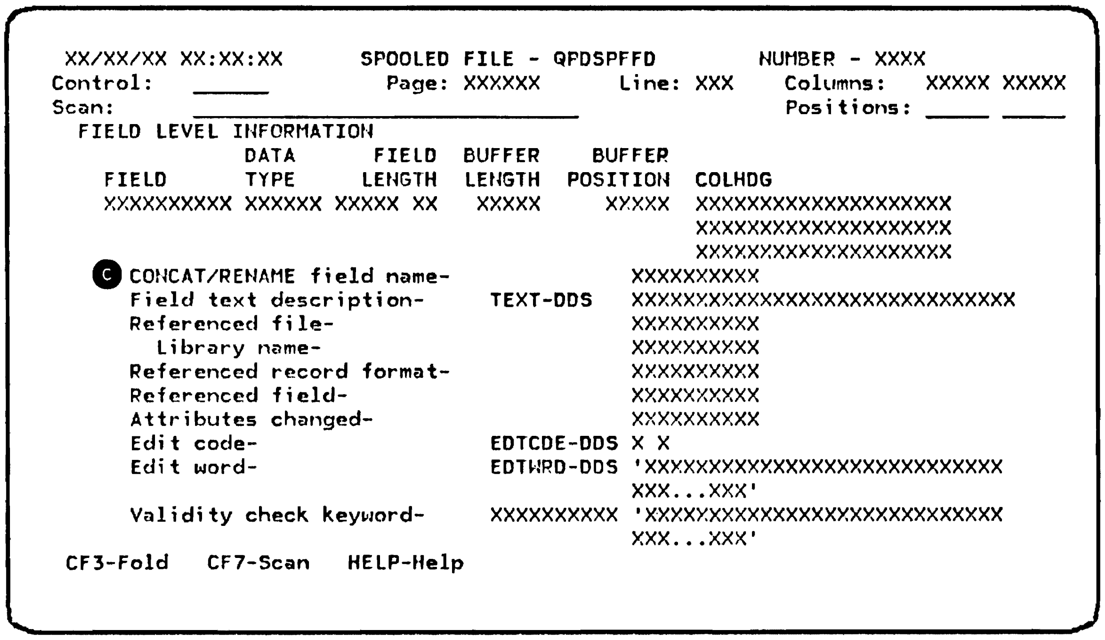
A Beginning with the file name under FILE INFORMATION of the first display, the following file-level attributes are shown:

- The name of the file whose attributes are being displayed, and the name of the library in which the file is stored
- The number of record formats used in the file
- The date the file was created
- The text that describes the file (the text is stored in the file's description)
- The type of the file (physical, logical, or device)

B Beginning with RECORD FORMAT INFORMATION, the following record format level attributes are shown:

- The name of the record format (specified in DDS)
- The most recent level identifier assigned to the record format by the system
- The text (specified in DDS) that describes the record format
- The number of fields in the record format
- The total length of the record format (in bytes)

The following display is the field level display for data base files:



```
  XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPFFD        NUMBER - XXXX
 Control:  ______        Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
 Scan:     ______________________________               Positions: _____ _____
   FIELD LEVEL INFORMATION
                DATA      FIELD   BUFFER   BUFFER
     FIELD      TYPE      LENGTH  LENGTH  POSITION  COLHDG
     XXXXXXXXXX XXXXXX XXXXX XX    XXXXX     XXXXX  XXXXXXXXXXXXXXXXXXXX
                                                    XXXXXXXXXXXXXXXXXXXX
                                                    XXXXXXXXXXXXXXXXXXXX
     C CONCAT/RENAME field name-                XXXXXXXXXX
       Field text description-     TEXT-DDS     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
       Referenced file-                         XXXXXXXXXX
         Library name-                          XXXXXXXXXX
       Referenced record format-                XXXXXXXXXX
       Referenced field-                        XXXXXXXXXX
       Attributes changed-                      XXXXXXXXXX
       Edit code-                  EDTCDE-DDS   X X
       Edit word-                  EDTWRD-DDS   'XXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                XXX...XXX'
       Validity check keyword-     XXXXXXXXXX   'XXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                XXX...XXX'
  CF3-Fold   CF7-Scan   HELP-Help
```

Ⓒ For each field, the following field level attributes are shown:

- The name of the field (specified in DDS).
- The type of data in the field.
- If the field is a decimal field, the number of digits in the field and, of that number, the number of decimal positions in the field.
- The length of the field (in bytes).
- The position in storage (the location) of the buffer.
- Up to three column headings that can be specified over the field and used by a utility, such as Query.

Beginning with Ⓒ, the following values are shown:

- The internal name of the field (specified with the CONCAT or RENAME keywords in DDS).
- The text (specified in DDS) that describes the field.
- The name of the file that contains the referenced field and the name of the library in which it is stored.
- The name of the record format that contains the referenced field.
- The name of the referenced field whose attributes are to be used by the field being displayed.
- Any of the attributes from the referenced field that have been changed.
- The edit code or edit word to be used to edit the field. Two single-character values are displayed if an edit code was specified: the edit code identifier and an * (if asterisk fill was specified in DDS); or a blank in the second position.
- If the field is validity checked, the DDS validity checking keyword and value is used. The validity check value does not contain a decimal point.

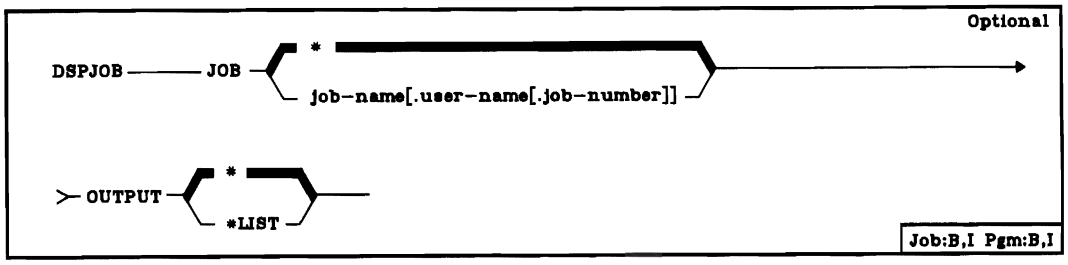
For device files, INPUT BUFFER and OUTPUT BUFFER fields replace the BUFFER POSITION field to indicate the offset values, which are the number of bytes from the beginning of the input and output buffers. Replacing the COLHDG field is one that indicates whether the usage for the field is input or output or both.

Also for device files, DSPFFD shows attributes for indicators, including the indicator name, whether it is a response or option indicator, and the text description. For indicators in the buffer area, the input and output buffer offsets are shown. Indicator information is shown following the RECORD FORMAT INFORMATION field.

DSPJOB (Display Job) Command

The Display Job (DSPJOB) command displays, for the specified user job, any of the following information: job status attributes, job definition attributes, job execution attributes, program invocation stack information, spooled file information, and job lock information. The information can be displayed regardless of where the user's job is in the system: on the job queue, on an output queue, or active in the system. Note, however, that the job is not considered to be in the system until all of its input has been completely read in; only then is an entry placed on the job queue.

Restrictions: (1) You can display only your own job unless you have the special job control rights (*JOBCTL) or unless the job you want to display has the same user name you have. (2) This command cannot be used to display system jobs, but can be used to display spooling readers and writers.



JOB Parameter: Specifies the name of the user job whose status information is to be displayed.

<u>*:</u> The job whose status information is to be displayed is the job in which this display command is issued.

qualified-job-name: Enter the qualified name of the job that is to have status information about it displayed. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. If duplicates of the specified name are found, a list of messages containing the qualified job names of all duplicates is displayed. (For an expanded description of the JOB parameter and non-unique job names, see Appendix A.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the names of the files used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

DSPJOB JOB(PAYROLL.SMITH) OUTPUT(*LIST)

This command directs the status information for the job named PAYROLL submitted by the user named SMITH to the job's output spooling queue for printing.

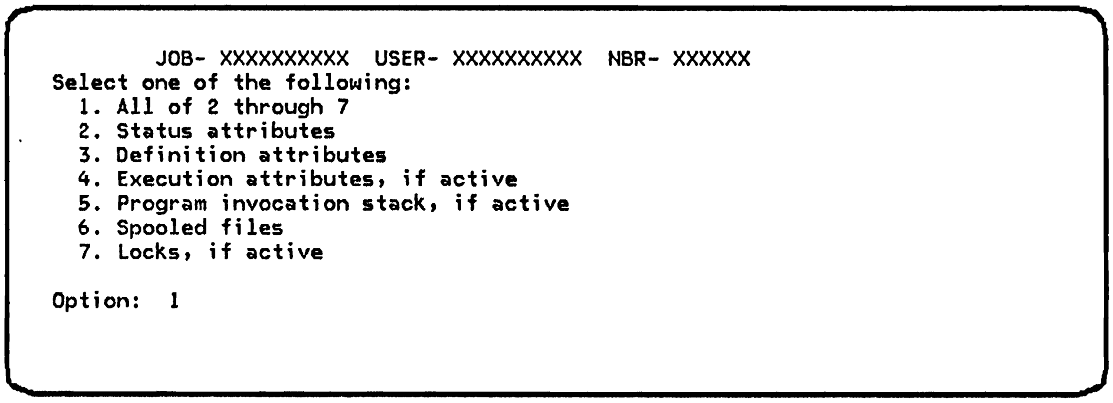
Additional Considerations

A menu and a series of displays can be presented through the DSPJOB command. When the DSPJOB command is entered interactively, and OUTPUT(*) is specified or assumed, the display job menu is presented. The display job menu allows you to select one of seven options. These options display the job's definition and execution attributes, the job's status, the job's program invocation stack, information about the job's associated spooled input or output files, and information about locks associated with the job.

Note: This same menu and the following set of displays can also be accessed for the current job by selecting Option 3 from the system request menu, by entering a 1 on displays produced by the DSPSBS, DSPJOBQ, DSPSBMJOB, DSPLINSTS, DSPCTLSTS, DSPDEVSTS, DSPOBJLCK, or DSPACTJOB commands, or by keying a CF3 on the detailed attributes display of the DSPRDR or DSPWTR commands.

If the DSPJOB command is part of a batch job, or if OUTPUT(*LIST) is specified for an interactive job, all of the available information for Options 2 through 7 on the menu is written to a spooled file to be printed, including the details for every spooling entry shown for Option 6.

If the information is to be displayed at a work station, the display job menu is presented when the DSPJOB command is entered.



```
        JOB- XXXXXXXXXX  USER- XXXXXXXXXX  NBR- XXXXXX
Select one of the following:
  1. All of 2 through 7
  2. Status attributes
  3. Definition attributes
  4. Execution attributes, if active
  5. Program invocation stack, if active
  6. Spooled files
  7. Locks, if active

Option:  1
```

Line 1 of the display job menu shows the job name, user name, and job number. Seven available options are listed next. To select one of the seven options from the menu, enter the associated option number (1 through 7) in the *Option* field and press the Enter key. (If the field is blank when Enter is pressed, you will return to the display from which the job display menu was requested.)

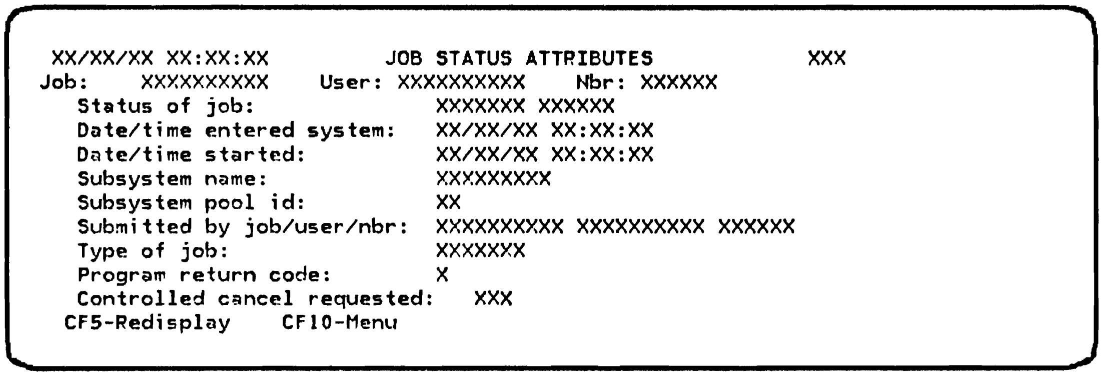
If option 1 is selected, all of the displays for options 2 through 7 are displayed in the order listed. Briefly, the information displayed is:

2. Status attributes: This display shows the identifying characteristics and the status of the job.
3. Definition attributes: This display shows the job definition attributes that are in the job description associated with this job.
4. Execution attributes: This display, shown only if the job is active, gives the job execution attributes that are defined in the class associated with the job.
5. Program invocation stack: This display, shown only if the job is active, lists information for all programs in the invocation stack.
6. Spooled files: One of two displays provides information about the job's spooled input or output file(s).
7. Locks: This display, shown only if the job is active, presents a list of all lock requests outstanding for the job. All locks held and pending are displayed, except for locks on data base records.

As you advance through the displays for option 1, you can use the command function keys as follows:

CF1	Ends the sequence of job displays and returns control to the working display, such as the command entry display, programmer menu, operator menu, and so forth.
CF2	Returns the sequence to the previous display.
CF5	Reshows the display you are viewing with updated information.
CF10	Returns the sequence to the primary DSPJOB menu.

Option 2. If option 2 is selected, the *job status attributes* display is presented:



```
 XX/XX/XX XX:XX:XX         JOB STATUS ATTRIBUTES          XXX
Job:    XXXXXXXXXX    User: XXXXXXXXXX    Nbr: XXXXXX
   Status of job:               XXXXXXX XXXXXX
   Date/time entered system:    XX/XX/XX XX:XX:XX
   Date/time started:           XX/XX/XX XX:XX:XX
   Subsystem name:              XXXXXXXXX
   Subsystem pool id:           XX
   Submitted by job/user/nbr:   XXXXXXXXXX XXXXXXXXXX XXXXXX
   Type of job:                 XXXXXXX
   Program return code:         X
   Controlled cancel requested:   XXX
  CF5-Redisplay    CF10-Menu
```

Line 1 of the job status attributes display gives the current job date and time. Lines 2 through 10 display the following job attributes:

- Job name, user name, and job number: These three attributes are derived from the qualified job name, which is also the name that was specified, or used, when the DSPJOB command was entered. (For additional information about the parts of a qualified job name and where each part comes from, refer to the expanded description of the JOB parameter in Appendix A.)

- Status of job (in two fields):
  - The first field indicates whether the job has been started (ACTIVE), is currently on an output queue (OUTQ) or on a job queue (JOBQ or TFRJOB if the job was transferred to another job), has been suspended by the system request key (SYSREQ), has finished (FIN) or was canceled (CANCEL).
  - If a job is terminating because a CNLJOB (Cancel Job) *IMMED or a TRMSBS (Terminate Subsystem) *IMMED has been specified, or, if CNLJOB(*CNTRLD) or TRMSBS(*CNTRLD) has been specified and delay time has expired, the first status field will indicate CANCEL.
  - The second field indicates whether the job is being held (HELD) or not held (the field is blank).
  - If the system failed while the job was active, the status JOBLOG PENDING is shown in the two fields until the job log is written.

- Date/time entered system: The date/time the job was entered into the system.

- Date/time started: The date/time when the execution of the job began.

- Subsystem name: The name of the subsystem in which the job is being processed. If the field is blank, the job is not active at the time this display is requested.

- Subsystem pool id: The pool defined for the subsystem in which the job is executing. The subsystem pool identifier is defined in the routing entry used to initiate the routing step that is currently active for the job.

- Submitted by job/user name/job number: The qualified job name of the job from which this job was submitted. This attribute is displayed only if the job originated as a result of a SBMJOB command being executed.

- Type of job (INTER, BATCH, AUTO, RDR, or WTR).

- Program return code: If the job contains any RPG, COBOL, DFU, or conversion reformat utility programs, the completion status of the last program that has finished execution is shown.

- Controlled cancel requested: Indicates whether a controlled cancel of the job has been requested, or whether the subsystem in which the job is executing is being terminated.

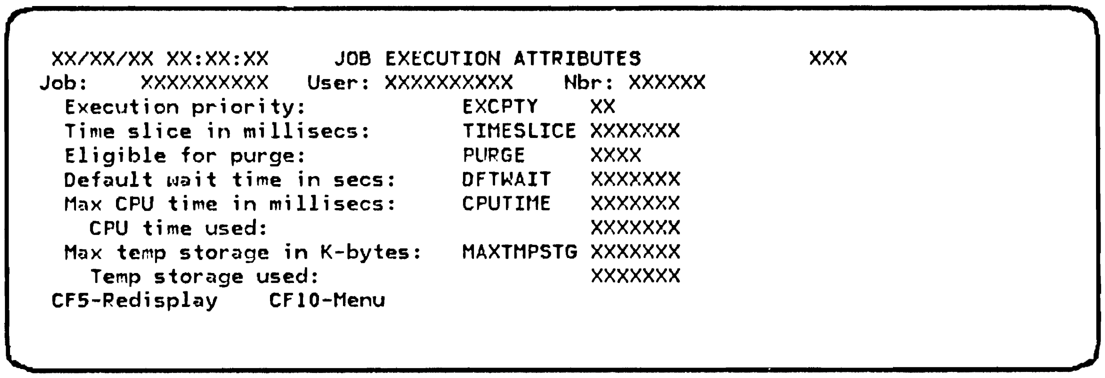
Option 3. If option 3 is selected, the *job definition attributes display* is presented:

```
 XX/XX/XX XX:XX:XX         JOB DEFINITION ATTRIBUTES        XXX
Job:    XXXXXXXXXX   User: XXXXXXXXXX    Nbr: XXXXXX
  Job queue name:                   JOBQ      XXXXXXXXXX
    Library name:                                XXXXXXXXXX
  Job priority (on JOBQ):           JOBPTY    X
  Output priority (on OUTQ):        OUTPTY    X
  Cancel severity:                  CNLSEV    XX
  Job logging (lvl sev text):       LOG       X  XX  XXXXXXX
  Default output queue name:        OUTQ      XXXXXXXXXX
    Library name:                                XXXXXXXXXX
  Job date:                         DATE      XX/XX/XX
  Job switches:                     SWS       XXXXXXXX   CF5-Redisplay   CF10-Menu
```

Lines 1 and 2 of the job definition attributes display repeat the current job date and time and the qualified job name. The rest of the display shows attributes that were specified in the job description associated with this job. They are explained in their associated parameter descriptions in the CRTJOBD command description. (The names of the parameters associated with these attributes are given in the second column of this display.) For example, the job date attribute is described in the DATE parameter.

If the job is no longer on the job queue when the job display is requested, the following attribute fields are blank: job queue name, job queue library name, and job priority.

Option 4. If option 4 is selected, the *job execution attributes display* is presented:



```
 XX/XX/XX XX:XX:XX     JOB EXECUTION ATTRIBUTES           XXX
Job:    XXXXXXXXXX   User: XXXXXXXXXX    Nbr: XXXXXX
  Execution priority:            EXCPTY    XX
  Time slice in millisecs:       TIMESLICE XXXXXXX
  Eligible for purge:            PURGE     XXXX
  Default wait time in secs:     DFTWAIT   XXXXXXX
  Max CPU time in millisecs:     CPUTIME   XXXXXXX
    CPU time used:                         XXXXXXX
  Max temp storage in K-bytes:   MAXTMPSTG XXXXXXX
    Temp storage used:                     XXXXXXX
 CF5-Redisplay    CF10-Menu
```

This display is shown only if the job is active; otherwise, the display heading lines and the phrase *(job not active)* are displayed.

Lines 1 and 2 of the job execution attributes display repeat the current job date and time and the qualified job name. The rest of the display shows attributes that were specified in the class object associated with this job. They are explained in their associated parameter descriptions given in the CRTCLS command description.

Lines 8 and 10 show two attributes that indicate how much of the specified maximums on lines 7 and 9 have currently been used by the job. Line 8 shows how much CPU time (in milliseconds) has been used by the current routing step of the job at the time this display is presented. Line 10 shows the amount of storage (in K-bytes) that is currently allocated to this job. (CF5 can be used to reshow the display with the updated CPU time and storage amount used.)

Option 5. If option 5 is selected, the *job program stack* display is presented:

```
 XX/XX/XX XX:XX:XX          JOB PROGRAM STACK            XXX
Job:    XXXXXXXXXX  User: XXXXXXXXXX   Nbr: XXXXXX    XXXXXX
  RQS  PROGRAM    LIBRARY     STMT       INST
  XXX  XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX  XXXX
  XXX  XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX  XXXX
   .
   .



 CF5-Redisplay    CF10-Menu
```

This display is shown only if the job is active; otherwise, the display heading lines and the phrase *job not active* are not displayed.

The column headings in line 3 represent the following information for each program in the program invocation stack:

- RQS. Indicates the level of command entry nesting in the request processing program; displayed only if the program has received a request message.
- PROGRAM. Name of the program at this level in the invocation stack.
- LIBRARY. Name of the library that contains the program.
- STMT. The high-level language statement identifier. This identifier is not displayed if the machine instruction number is not displayed or if the debugging tables do not exist for the program.
- INST. The hexadecimal representation of the current or next machine instruction number in the program. This value is not displayed if the program cannot be displayed or has been suspended.

If the request level in the invocation stack exceeds 999, RQS is represented with characters +++. When the program is destroyed, the PROGRAM and LIBRARY will indicate *DESTROYED; they will indicate *LOCKED when the library is locked.

If the job being displayed is held, suspended because of a system request, or cannot be interrupted, only the PROGRAM and LIBRARY fields will be displayed.

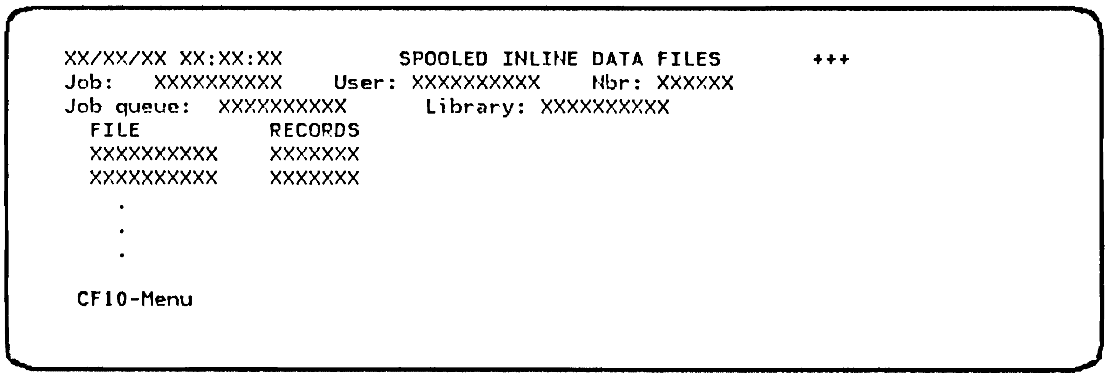
The programs used to obtain the display will not appear in the program stack.

Option 6. If the job is on a job queue or has spooled output on one or more output queues when the DSPJOB command is entered, you can request additional information about the job's spooled files by selecting option 6. (If the DSPJOB command is *not* entered at a work station, the information about the job's spooled files is automatically included in a spooled output file to be printed.)

These are the same spooled file displays that can be accessed when a 2 is entered for a job shown on the DSPSBS or DSPSBMJOB displays.

Restriction: The CF5 key cannot be used to redisplay the contents of the spooled inline data files display.

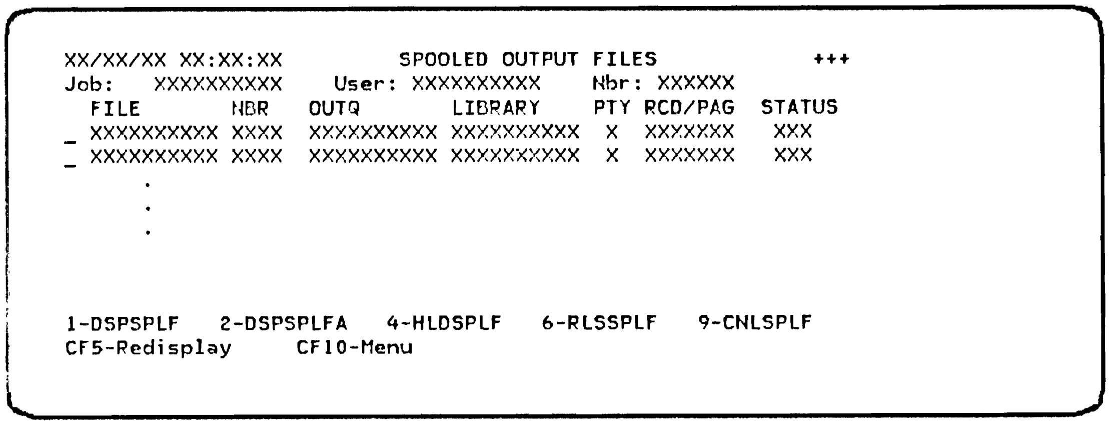
If option 6 is specified while the job is still on the job queue, a display of the job's inline files is shown:



```
XX/XX/XX XX:XX:XX         SPOOLED INLINE DATA FILES       +++
Job:   XXXXXXXXXX    User: XXXXXXXXXX    Nbr: XXXXXX
Job queue:  XXXXXXXXXX       Library: XXXXXXXXXX
  FILE          RECORDS
  XXXXXXXXXX    XXXXXXX
  XXXXXXXXXX    XXXXXXX
     .
     .
     .

 CF10-Menu
```

The first and second lines of the *spooled inline data files display* gives the current job date and time, and the qualified name of the job being displayed. The third line identifies the job queue and the library in which it is stored. Then all of the spooled input files associated with the job are identified, and the number of records in each file is shown. If the job has no inline files, the phrase *(no inline files)* is displayed on line 6.

If option 6 is specified after the job has started execution or the job has finished and output is still on the queue(s), the following display of the job's spooled output files is shown:



```
XX/XX/XX XX:XX:XX         SPOOLED OUTPUT FILES          +++
Job:   XXXXXXXXXX    User: XXXXXXXXXX    Nbr: XXXXXX
  FILE       NBR   OUTQ       LIBRARY    PTY RCD/PAG  STATUS
_ XXXXXXXXXX XXXX  XXXXXXXXXX XXXXXXXXXX  X  XXXXXXX   XXX
_ XXXXXXXXXX XXXX  XXXXXXXXXX XXXXXXXXXX  X  XXXXXXX   XXX
      .
      .
      .



1-DSPSPLF   2-DSPSPLFA   4-HLDSPLF   6-RLSSPLF   9-CNLSPLF
CF5-Redisplay     CF10-Menu
```

The first and second lines of the *spooled output files display* gives the current job date and time, and the qualified name of the job being displayed. Then, on a separate line for each spooled output file produced by the job, the following is displayed:

- An input field (to the left of the file name) in which a number can be entered. The number can be any one of those shown at the bottom of the display and can be entered in the input field to cause the function (a command) associated with that number to be performed for that spooled file when the Enter key is pressed. If numbers are placed in the input fields of several files before the Enter key is pressed, the specified functions are performed on the files in the order in which the files are shown on the display. The system executes each command using the default values of all its parameters. The following functions can be specified:

  1–DSPSPLF: Display the data in the spooled file.
  2–DSPSPLFA: Display the attributes of the spooled file.
  4–HLDSPLF: Hold the spooled file.
  6–RLSSPLF: Release the spooled file.
  9–CNLSPLF: Cancel the spooled file.

  - When all of the commands have been executed, the output files display is reshown with the status fields of the files updated; also shown at the bottom of the display are any error or completion messages that occurred when the commands were executed.
  - If the job has more spooled files than can be shown on a single display, the Roll Up key can be used to display them all. Numbers can be placed in the input fields on multiple displays before the Enter key is pressed.
  - After the commands have been executed, if there are more messages than can fit on that display, a + is shown at the end of the last (or only) message displayed. To view additional messages, position the cursor anywhere on a message line and use the Roll Up key.
  - The CF1 key can be used to exit from the display shown above, or to exit from a display presented as a result of executing the commands entered on the display shown above. If you are viewing a display that resulted from a 1 or 2 being entered to execute the DSPSPLF or DSPSPLFA command, the CF1 key causes all of the functions indicated for the files that followed the file currently being displayed to *not* be executed. That is, all files further down in the initial file list do not have any of their requested functions performed.

- The name of the device file that was opened to produce this spooled output file.

- The number of the spooled file, assigned by spooling to ensure uniqueness. The number indicates its sequential order (within the job) in relation to the other output files produced by the same job.

- The name of the output queue on which the file has an entry, and the name of the library in which the queue is stored.

- The output priority assigned to the file from a job command or from the job description associated with the job that created the output file.

- The total number of records or pages (for printer output) in the file. If the file is still open, this field is blank.

- The status of the file. One of the following values can be specified for the status:
  - FIN (finished). The file's output has already been produced on an output device.
  - RDY (ready). The file is complete and ready to be spooled to an output device.
  - OPN (open). The file has not been completely processed by a program and is not ready to be selected by a spooling writer.
  - CLO (closed). The file has been completely processed by a program, but SCHEDULE(*JOBEND) was specified in the associated device file and the job that produced the file has not yet finished.
  - HLD (held). The file has been held: by a HLDSPLF command; by a HLDJOB command that specified SPLFILE(*YES); by a spooling writer after a copy was produced because SAVE(*YES) was specified; because the device file was opened with HOLD(*YES); or because an error affecting some attribute of the file was detected as the file was being processed by a writer.
  - WTR (writing). The file is currently being produced on an output device by a spooling writer.

For both of the above displays, the CF10 key can be used to return to the primary DSPJOB menu. For the spooled output data files display, the CF5 key can be used to redisplay the screen with the most current information.

For file objects, the OBJTYPE field displays the specific type of file, as follows:

- *FILE-PHY. Physical file.

- *FILE-LGL. Logical file.

- *FILE-xxx. Device file, where xxx is the abbreviation of the specific type of device file.

The lock state is shown in the LOCK field as one of the following values:

- *SHRRD. Lock shared for read.

- *SHRUPD. Lock shared for update.

- *SHRNUP. Lock shared no update.

- *EXCLRD. Lock exclusive allow read.

- *EXCL. Lock exclusive no read.

- *NONE. Lock entry is a place holder used to select display of lower level locks.

The status of the lock is shown as one of the following values:

- HELD. The lock is currently held by the job.
- WAIT. The job is in a synchronous wait for the lock.
- REQ. The job has a lock request outstanding for the object.

For data base files, members may be locked and share/locked. When these data base file locks exist, the MBR LOCKS field appears as one of the following:

- YES. Additional locks exist.
- WAIT. This job is waiting for an additional lock.
- NO. There are no additional lock requests.

If there are additional locks, you can display them by using the job member locks option (2). The job member locks display has the following format:

```
XX/XX/XX XX:XX:XX      JOB MEMBER LOCKS
Job:     XXXXXXXXXX   User:     XXXXXXXXXX   Nbr:  XXXXXX
File:    XXXXXXXXXX   Library: XXXXXXXXXX   Type: XXX
   MEMBER        LCKTYP  LOCK     STS   SHR
_  XXXXXXXXXX    XXXXXX  XXXXXXX  XXXX  XXXX
_                XXXXXX  XXXXXXX  XXXX
_                XXXXXX  XXXXXXX  XXXX
_  XXXXXXXXXX    XXXXXX  XXXXXXX  XXXX  XXXX
                   .
                   .
                   .
1-DSPOBJLCK     3-Shared member locks          CF5-Redisplay
```

The first two lines of the job member locks display contain the date and time of the job, job and user identifiers, the qualified name of the file for which member locks are being displayed, and the type of the data base file (PHY or LGL).

Beginning with the fifth line, the member lock entries are shown. The entries contain:

- A one character input field
- The name of the member for which the lock was requested
- The lock type
- The lock state
- The lock status
- A field that indicates whether shared member locks are associated with the member (this field is not displayed for physical file members)

The lock entries are shown alphabetically by file member name and by job name within the member name.

The lock type (LCKTYP) field may take on any of the following values:

- MBR, for member control block locks
- DATA, for locks on the actual data within a member
- ACCPTH, for locks on the access path used to access a member's data

The shared member locks field (SHR) is displayed for logical member locks only. If additional locks associated with the member are held, the SHR field will indicate 'YES', or if the locks are being waited for, 'WAIT' will be indicated.

You can enter the shared member locks option (option 3) to display the shared member locks display:

```
 XX/XX/XX XX:XX:XX   JOB SHARED MEMBER LOCKS
Job:        XXXXXXXXXX  User:  XXXXXXXXXX  Nbr:       XXXXXX
Lgl mbr:    XXXXXXXXXX  File:  XXXXXXXXXX  Library:   XXXXXXXXXX
   MEMBER     FILE       LIBRARY     LCKTYP LOCK    STS
 _ XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX  XXXXXX XXXXXXX XXXX
 _                                   XXXXXX XXXXXXX XXXX
 _ XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX  XXXXXX XXXXXXX XXXX
                   .
                   .
                   .
 1-DSPOBJLCK                                   CF5-Redisplay
```

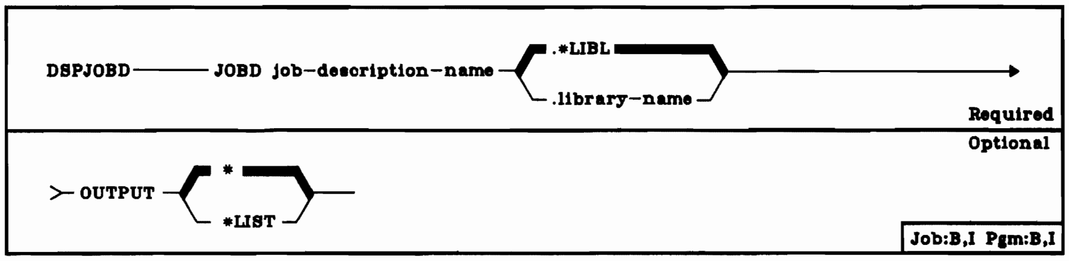
The first line of the display shows the date and time of the current job and the display title. The second line shows the qualified job name of the job for which locks are being displayed. The third line shows the qualified logical file member name that is sharing members for which locks are displayed.

Beginning with the fifth line, the shared member lock entries are shown. The entries are displayed with most of the same fields shown in the member locks display (SHR is not shown). The entries are displayed in alphabetic order by member name within the file.

DSPJOBD (Display Job Description) Command

The Display Job Description (DSPJOBD) command displays the contents of the specified job description.

Restriction: You must have operational and read rights for the job description before you can display its contents.



JOBD Parameter: Specifies the qualified name of the job description to be displayed. (If no library qualifier is given, *LIBL is used to find the job description.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

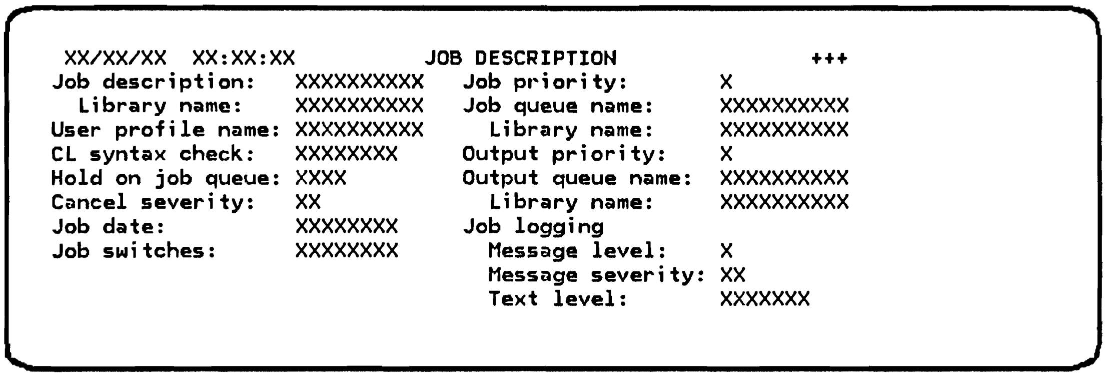
Example

```
DSPJOBD  JOBD(SPECIAL.MYLIB)
```

This command displays the job description named SPECIAL that is stored in the library MYLIB.

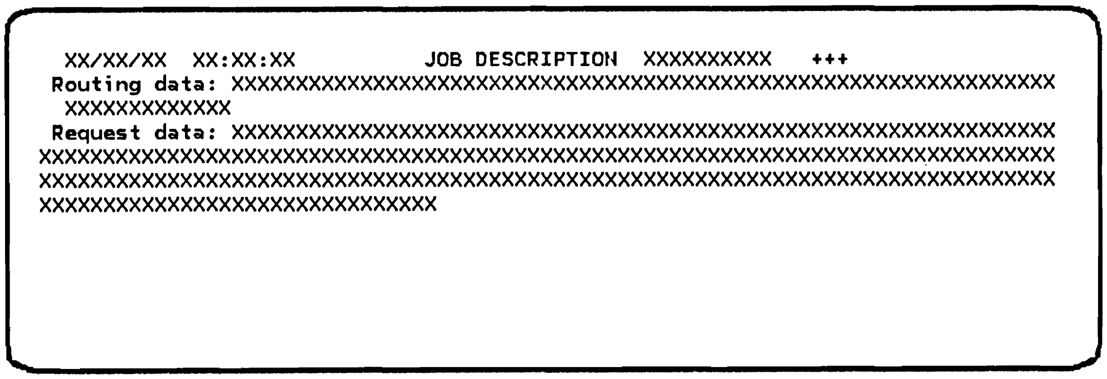
Additional Considerations

Three displays are used to present all of the attributes of the job description specified on the DSPJOBD command. The first display has the following format:



```
 XX/XX/XX  XX:XX:XX          JOB DESCRIPTION            +++
Job description:   XXXXXXXXXX   Job priority:       X
  Library name:    XXXXXXXXXX   Job queue name:     XXXXXXXXXX
User profile name: XXXXXXXXXX     Library name:     XXXXXXXXXX
CL syntax check:   XXXXXXXX     Output priority:    X
Hold on job queue: XXXX         Output queue name:  XXXXXXXXXX
Cancel severity:   XX             Library name:     XXXXXXXXXX
Job date:          XXXXXXXX     Job logging
Job switches:      XXXXXXXX       Message level:    X
                                  Message severity: XX
                                  Text level:       XXXXXXX
```

The first display shows the current job date and time, identifies the job description being displayed, and shows many of the attributes that can be assumed by one or more jobs. For an explanation of each attribute, refer to the associated parameter description given in the CRTJOBD command description; for example, the job switches attribute is explained in the SWS parameter.

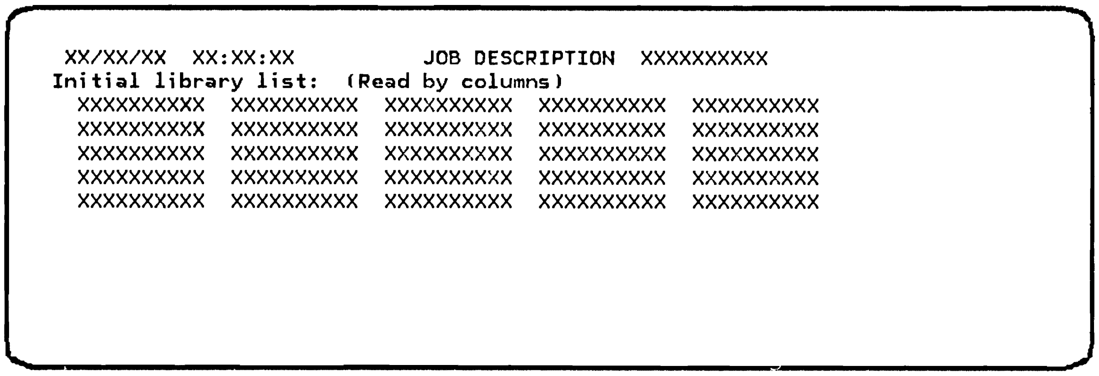


```
  XX/XX/XX  XX:XX:XX          JOB DESCRIPTION  XXXXXXXXXX   +++
 Routing data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXXXXX
 Request data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

The second display presents, for each job using this job description, the:

- Routing data to be submitted with each job. The routing data is used by the subsystem to initiate the correct routing step for the job.
- Request data to be placed on the job's message queue as a request.

The third display presents, for each job using this job description, the initial library list to be used for the job. Only the libraries in the user portion of the library list are included.



```
 XX/XX/XX  XX:XX:XX          JOB DESCRIPTION  XXXXXXXXXX
Initial library list:  (Read by columns)
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
```

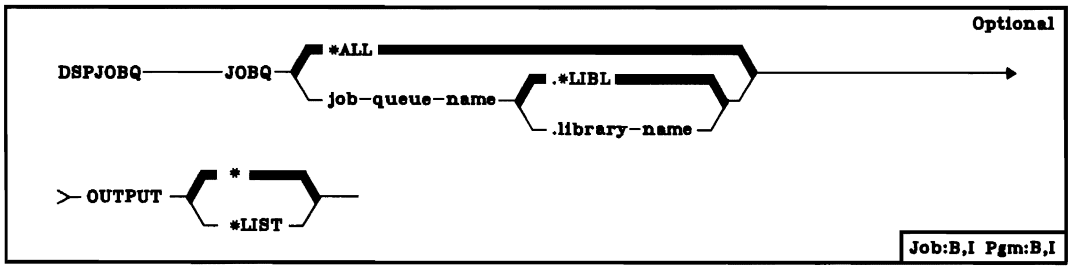
You can use the command function keys as follows:

CF1	Ends the sequence of job description displays and returns control to the working display, such as the command entry display, programmer menu, operator menu, and so forth.
CF2	Returns to the previous display. On the first display, returns to the caller.

DSPJOBQ (Display Job Queue) Command

The Display Job Queue (DSPJOBQ) command displays the overall status of all job queues or the detailed status of a specific job queue. The status of the queues may change while the command is being executed.

Restrictions: If only one job queue is to be displayed, the user must have read rights for the queue to be displayed or he must have job control rights in his user profile and the job queue must have the OPRCTL(*YES) attribute. If all the job queues are to be displayed, the user needs authority only for the DSPJOBQ command.



JOBQ Parameter: Specifies that all job queues are to be displayed, or specifies the name of the job queue that is to have its status displayed.

*ALL: The overall status of all job queues is to be displayed. The following information about each job queue is given: qualified job queue name, the queue's status, its subsystem, and the number of entries on the queue.

qualified-job-queue-name: Enter the qualified name of the job queue about which detailed status information is to be displayed. (If no library qualifier is given, *LIBL is used to find the queue.) The following information about each job entry on the specified queue is given: the job's name and number, user name, and the job's priority and status.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPJOBQ  JOBQ(QBATCH.QGPL)
```

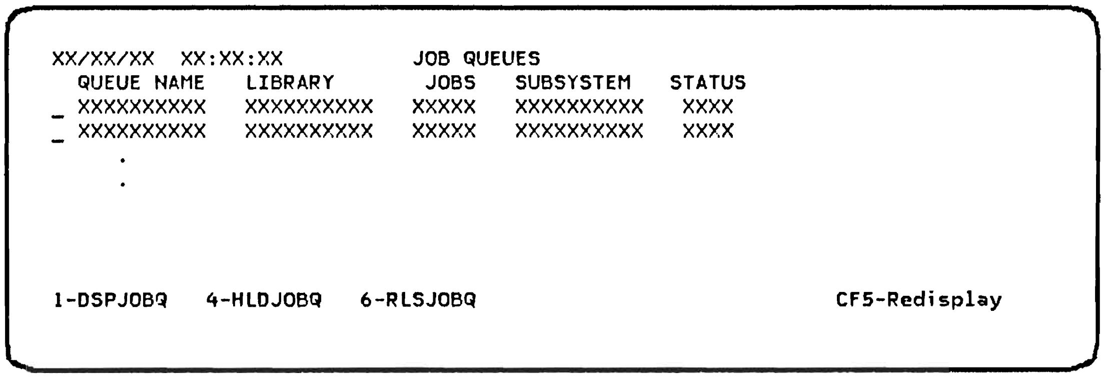
This command displays the detailed status information about the job queue named QBATCH qualified with library name QGPL. Each job entry on the QBATCH job queue is identified by job name, user name, and job number; the job's priority and status are also displayed.

Additional Considerations

One of two display formats is produced by the DSPJOBQ command to display the job queues and their current status. The display presented depends on whether you request the detailed status of a specific job queue or the overall status of all job queues.

Displaying All Job Queues

If JOBQ(*ALL) is specified or assumed on the DSPJOBQ command, the *job queues display* is presented:



```
XX/XX/XX  XX:XX:XX          JOB QUEUES
  QUEUE NAME   LIBRARY       JOBS   SUBSYSTEM   STATUS
_ XXXXXXXXXX   XXXXXXXXXX   XXXXX   XXXXXXXXXX   XXXX
_ XXXXXXXXXX   XXXXXXXXXX   XXXXX   XXXXXXXXXX   XXXX
     .
     .



1-DSPJOBQ   4-HLDJOBQ   6-RLSJOBQ                          CF5-Redisplay
```

The overall status of each of the job queues defined in the system is displayed. For each job queue, a separate line is used to show:

- An input field (to the left of the job queue name) in which a number can be entered. The number can be any one of those shown at the bottom of the display and can be entered in the input field to cause the function (a command) associated with that number to be performed for that job queue when the Enter key is pressed. If numbers are placed in the input fields of several job queues before the Enter key is pressed, the specified functions are performed (one at a time) on the queues in the order in which the queues are shown on the display. The system executes each command using the default values of all its parameters. The following functions can be specified:

  1–DSPJOBQ: Display the status of the jobs on the job queue.
  4–HLDJOBQ: Hold the job queue.
  6–RLSJOBQ: Release the job queue.

  - When all of the commands have been executed, the job queues display is reshown with the status fields of the job queues updated; also shown at the bottom of the display are any error or completion messages that occurred when the commands were executed.
  - If there are more job queues than can be shown on a single display, the Roll Up key can be used to display the rest of them. Numbers can be placed in the input fields on multiple displays before the Enter key is pressed.
  - After the commands have been executed, if there are more messages than can fit on that display, a + is shown at the end of the last (or only) message displayed. To view additional messages, position the cursor anywhere on a message line and press the Roll Up key. If no input values are specified in the input fields and the Enter key is pressed, it causes an exit back to the display from which the current display was requested.
  - The CF1 key can be used to exit from the display shown above, or to exit from a display presented as a result of executing the commands entered on the display shown above. (In both cases, the exit causes a return to the basic working display or menu from which the last primary request was entered.) If, for example, you are viewing a display that resulted from a 1 being entered to execute the DSPJOBQ command, the CF1 key causes all of the functions indicated for the job queues that followed the queue currently being displayed to *not* be executed. That is, all queues further down in the initial job queue list do not have any of their requested functions performed.
  - The CF5 key can be used to ignore (cancel) any functions not yet executed that were entered in the input fields, and to reshow the job queues display with the updated status of all the queues.

- The name of the job queue and the name of the library in which it is stored.

- The number of job entries on the queue that are waiting to be scheduled for processing.

- The name of the subsystem that is currently associated with the job queue (that is, the subsystem that can remove jobs from the queue). If no name is shown, the job queue is not associated with an active subsystem and the jobs cannot be processed at the present time.

- The current status of the job queue. If the queue has been held by a HLDJOBQ command, HELD is shown in the status column. If jobs can be removed from the queue for processing, this field is blank.

Note: You can use the CF5 key to get an update on the status of all the job queues (if viewing the overall status of the queues) or of the jobs on a single job queue (if viewing the status of a specified queue) without having to reenter the DSPJOBQ command.

Displaying a Single Job Queue

If a single job queue name is specified on the JOBQ parameter of the DSPJOBQ command, the *job queue display* presents the detailed status of that queue:

```
XX/XX/XX XX:XX:XX  JOBQ - XXXXXXXXXX LIB - XXXXXXXXXX   XXXXXXX
  JOB NAME     USER         NBR        PTY    STATUS
 _XXXXXXXXXX   XXXXXXXXXX   XXXXXX      X      XXXX
 _XXXXXXXXXX   XXXXXXXXXX   XXXXXX      X      XXXX
      .
      .



1-DSPJOB   4-HLDJOB   6-RLSJOB   9-CNLJOB       CF5-Redisplay
```

All the jobs that are currently on the job queue identified on line 1 are shown on this display. Also shown on line 1 are the current job date and time, the name of the library in which the job queue is stored, and the status of the subsystem and/or queue. If a subsystem is using this job queue, the status field indicates SBS. If the job queue is being held, HLD is displayed in the status field. If the subsystem is active and the job queue is held, SBS/HLD is displayed.

All of the jobs are displayed in an order determined by: whether the job is available or unavailable (available jobs are all shown first), the priority of the job (jobs with a priority of 1 are shown first), and the number of the job (shown in the NBR column).

For each job that has a job entry on the queue, a separate line is used to display:

- An input field (to the left of the job name) in which a number can be entered. The number, which can be any one of the following, causes the function shown with the number to be performed for that job when the Enter key is pressed. The information given in the dashed list describing the input field on the previous display (of all job queues) also applies to the `input field on this display for jobs on a single job queue.

  1–DSPJOB: Display (via the display job menu) the attributes and status of the job.
  4–HLDJOB: Hold the job on a job queue or output queue.
  6–RLSJOB: Release the job.
  9–CNLJOB: Cancel the job.

- The qualified name of the job (shown in the first three columns).
  - Job name. Jobs are listed in the order in which they will be selected for execution unless they are held. (For information on where the job name comes from, refer to the expanded description of the JOB parameter in Appendix A.)
  - User name. This name identifies the user profile under which the job is running.
  - Job number. This six-digit number is assigned by the system.

- The job priority assigned to the job from a job command or from the job description associated with the job.

- The status of the job. If HELD is shown for the job, it is being held on the queue.

DSPJRN (Display Journal) Command

The Display Journal (DSPJRN) command allows you to convert journal entries (contained on one or more receivers) into a form for external representation. Output of the command can be displayed or listed with the job's spooled printer output and, optionally, directed to a data base output file. If the data base output file exists, records will replace the current data in the file (member) indicated. The system will create the data base file and member if it does not exist. When created by the system, the data base output file will have a standard format. A warning message is issued if any of the converted entries are longer than the record length of the output files; if so, those records are truncated.

The entire contents of selected entries of the journal receivers may be converted to an external representation and displayed or directed to a data base output file. It is also possible to selectively limit the entries that are to be converted. If no journal entries satisfy the selection or limitation criteria, an escape message will be sent indicating this result.

It is possible to display journal entries whose journal sequence numbers have been reset within the chain of receivers specified.

If journal receivers are attached and detached in pairs, the system will always attempt to use the first of the receivers in the pair (the first of the two shown in the DSPJRNA receiver directory display). When the first of the pair is not accessible (for example, not found or damaged), the system will attempt to use the second receiver of the pair. If both receivers are not accessible, the conversion will terminate.

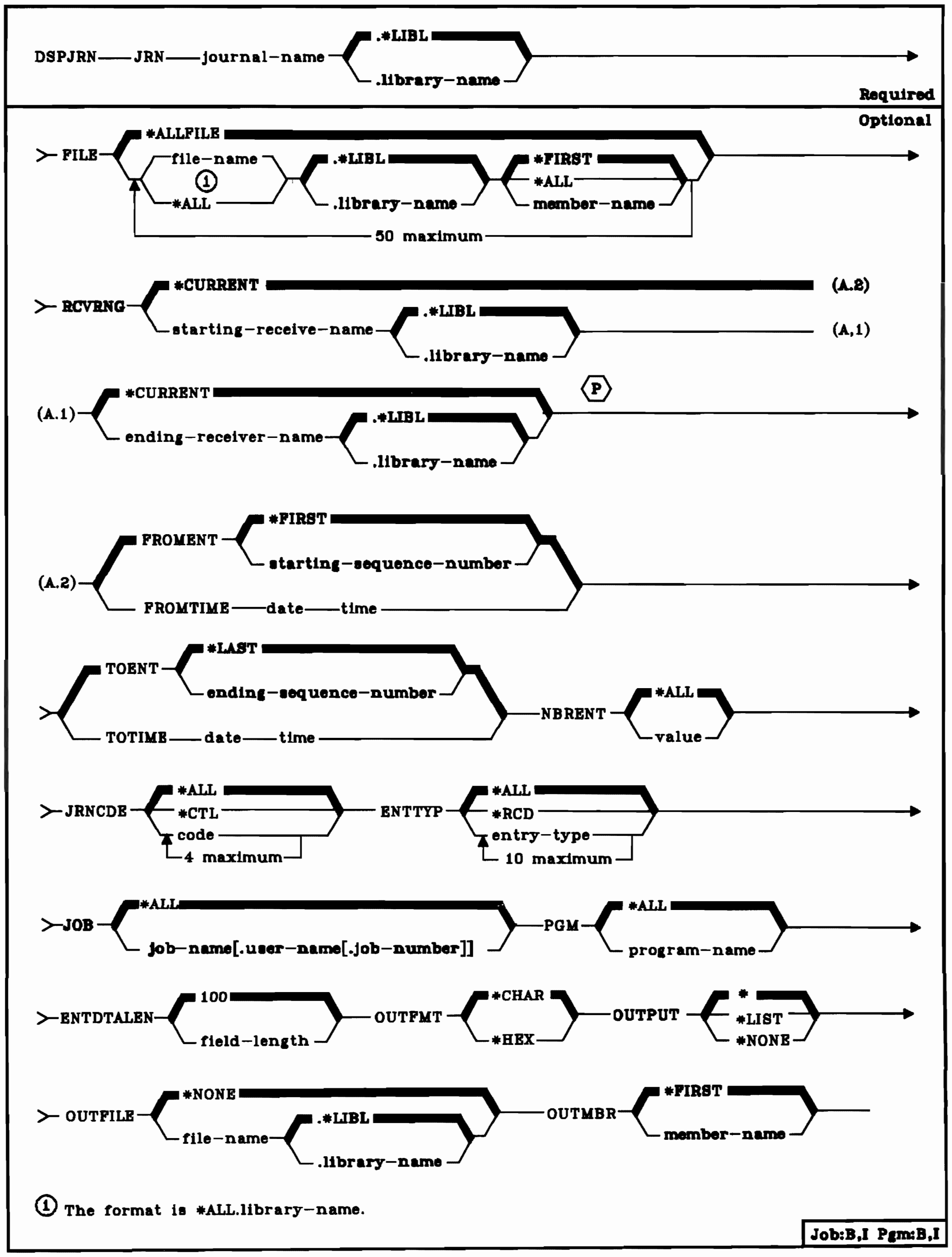
When the output of the command is directed to the requesting work station, basic information on the journal entries is displayed. The next sequential set of entries can be displayed by using the roll key. From the basic display, an option can be selected to display detailed information for any journal entry shown.

If the output is to be listed with the job's spooled printer output, then all of the information that would be displayed is listed.

If the output is to be displayed at the requesting work station or listed with the job's spooled printer output, a hexadecimal representation of the entry specific data may be requested.

Restrictions: The record format of the data base output file must match the record format of the IBM-supplied output file QADSPJRN (except for the length of the last field in the format). The file specified for the data base output file must not be having its changes journaled at the time it will receive data.

If the sequence number has been reset within the range of receivers specified, the first occurrence of FROMENT or TOENT will be used if they are specified. For more information on data base file record formats, refer to the *CPF Programmer's Guide.*



JRN Parameter: Specifies the qualified name of the journal from which the journal entries are to be obtained, for conversion to an external representation. (If no library qualifier is given, *LIBL is used to find the journal.)

FILE Parameter: Specifies up to 50 qualified file names whose journal entries are to be converted for external representation.

ALLFILE: Specifies that all journal entries on the specified journal receivers for all files will be considered for conversion.

file-name: Enter the name of the physical data base file whose journaled changes are to be considered for conversion. (If no library qualifier is given, *LIBL is used to find the journal.)

If a specified physical file member is not found, all journal entries in the specified receiver range that correspond to a physical file member of the same qualified name will be considered for conversion. If a specified physical file member is found, all journal entries in the specified receiver range that correspond to *that* physical file member of the same qualified name will be considered for conversion.

**ALL:* Specifies that journal entries will be considered for conversion for all physical files within the specified library (the library name *must* be specified) that are currently having their changes journaled to the journal (specified by JRN). If *ALL is specified and you do not have the required authority to all of the files, an exception is signaled and the command terminates.

The FILE parameter also specifies the name of the member within the file that is to have its journal entries converted.

FIRST: Specifies that entries for the first member in the file will be considered for conversion.

**ALL:* Specifies that entries for all members of the file will be considered for conversion.

member-name: Enter the name of the member whose entries are to be considered for conversion.

If *ALL is specified for the file-name value, this member name is used for all applicable files within the library. For example, if FILE(*ALL.library-name *FIRST) is specified, the first member of all applicable files in the library will have the changes applied.

Note: If the maximum number of members is exceeded (1024 if a list of files is specified), an exception is signaled and no entries will be converted. This restriction is ignored if *ALLFILE is specified.

RCVRNG Parameter: Specifies the first (beginning) and last (ending) journal receivers to be used in converting the journal entries. The system will start the conversion with the first journal receiver (as specified by the first value) and will proceed through the receiver chain until the last receiver (as specified by the last value) is processed. If dual receivers were used at any time, the first of the receivers will be used when chaining through the receivers. (The DSPJRNA command can be used to display the order of the receivers in the receiver chain.) If any problem is encountered in the receiver chain (such as damaged or offline receivers) before the conversion, the system will attempt to use the second of the dual receivers. If the second of the receivers is damaged or offline, or if a problem is encountered during the conversion, the conversion will terminate.

First Receiver Name

*CURRENT: The journal entries in the currently attached receiver are to be converted.

starting-receiver-name: Enter the qualified name of the first journal receiver whose entries are to be converted. (If no library qualifier is given, *LIBL is used to find the receiver.)

Second Receiver Name

*CURRENT: Specifies that conversion of journal entries will continue for all journal receivers (in the chain that began with the receiver specified by the first parameter value) through the currently attached journal receiver.

ending-receiver-name: Enter the qualified name of the last journal receiver whose entries are to be converted. If the end of the receiver chain is reached before encountering a receiver of this name, an exception is signaled and no journal entries will be converted. (If no library qualifier is given, *LIBL is used to find the receiver.)

Note: If the maximum number of receivers in the range is exceeded (256), an exception is signaled and no journal entries will be converted.

FROMENT Parameter: Specifies the first journal entry to be considered for conversion.

*FIRST: The first journal entry in the journal receiver range specified will be the first entry considered for conversion.

starting-sequence-number: Specifies that the journal entry with the designated sequence number will be the first entry considered for conversion.

FROMTIME Parameter: Specifies the date and time of the first journal entry to be considered for conversion.

date time: The first journal entry with the designated or the next later time encountered will be the starting point for the conversion of journal entries. The time can be entered as four or six digits (hhmm or hhmmss) where hh = hours, mm = minutes, and ss = seconds. If colons are used to separate the time values, they must be enclosed in apostrophes ('hh:mm:ss'). The format of the date must be defined by the system values QDATFMT and, if separators are used, QDATSEP.

TOENT Parameter: Specifies the last journal entry to be considered for conversion.

*LAST: The last journal entry in the journal receiver chain specified will be the final entry considered for conversion.

ending-sequence-number: The journal entry with the designated sequence number will be the final entry considered for conversion.

TOTIME Parameter: Specifies the date and time of the last entry to be considered for conversion.

date time: The first journal entry with the designated or the next earlier time encountered will be the ending point for the conversion of journal entries. The time can be entered as four or six digits (hhmm or hhmmss) where hh = hours, mm = minutes, and ss = seconds. If colons are used to separate the time values, the string must be enclosed in apostrophes ('hh:mm:ss'). The format of the date must be defined by the system values QDATFMT and, if separators are used, QDATSEP.

NBRENT Parameter: Specifies the total number of journal entries that are to be converted.

*ALL: All journal entries in the journal receivers specified that satisfy the selection criteria will be converted.

value: Specifies the maximum number of journal entries to be converted. If the specified journal entry designated by the TOENT or TOTIME parameter is reached before the value specified for NBRENT is met, the conversion will terminate normally.

JRNCDE Parameter: Specifies that the entries being considered for conversion are to be limited to the journal entries that contain the designated journal code. For more information on the journal codes, refer to the *CPF Programmer's Guide.*

*ALL: The conversion is not to be limited to entries with a particular journal code.

**CTL:* Specifies that only journal entries that are generated to control the journal facilities are to be considered for conversion (journal codes = 'J' and 'F').

code: Enter the journal code(s) that will limit the conversion. Up to four valid journal codes may be specified. Only journal entries that contain the designated journal codes will be considered for conversion.

ENTTYP Parameter: Specifies that the journal entries being considered for conversion are to be limited to the journal entries that contain the designated entry types. For more information on the journal types, refer to the *CPF Programmer's Guide.*

*ALL: The conversion is not to be limited to entries with a particular entry type.

**RCD:* The conversion is to be limited to entries with an entry type for record level operations (entry types PT, UB, UP, and DL).

entry-type: Enter the entry type(s) that will limit the conversion. Up to ten valid entry types may be specified. Only journal entries that contain the designated entry types will be considered for conversion.

JOB Parameter: Specifies that the journal entries to be considered for conversion are to be limited to the journal entries for a particular job. Only journal entries for the designated job will be considered for conversion.

*ALL: The conversion is not to be limited to entries for a particular job.

job-name: Enter the qualified name of the job whose journaled changes are to be considered for conversion. Only journaled changes for this job will be be considered for conversion. If no job qualifier is given, all of the journal entries containing the simple job name will be considered for conversion.

PGM Parameter: Specifies that the journal entries to be considered for conversion are to be limited to the journal entries generated by a particular program. Only journal entries for the designated program will be considered for conversion.

*ALL: The conversion is not to be limited to entries for a particular program.

program-name: Specifies the name of the program whose journaled changes are to be considered for conversion. Only journaled changes for this program will be considered for conversion.

Note: If multiple selection criteria are specified, a journal entry must satisfy all of the selection criteria to be converted. Select criteria are FILE, JOB, PGM, ENTTYP and JRNCDE. If none are specified, all entries will be converted as specified in the values given for the FROMENT, TOENT, FROMTIME, TOTIME and NBRENT parameters.

Gaps may exist in the sequence numbers of the entries converted. These occur because some of the journal entries represent internal system information. These entries are not converted.

ENTDTALEN Parameter: Specifies the field length of the entry-specific data portion of the journal entry when the output file is created by the system. This field will contain the variable portion of the journal entries (such as the after image of records for update journal entries). If the output file is not specified or if it already exists, this parameter is ignored.

100: Specifies that the field length of the entry-specific data portion of the output file is to be 100 characters.

field-length: Enter a value greater than 0 and less than 32 641. (A non-zero value is required by the system to ensure that the field exists when the output file is created. The maximum value prevents the total record length of the output file from exceeding the length allowed for physical data base files.)

OUTFMT Parameter: Specifies whether the entry-specific data portion of the journal entry information is to appear in character or hexadecimal format. This parameter is ignored if OUTPUT(*NONE) is specified.

*CHAR: The entry-specific data portion of the journal entry will be shown in character format.

**HEX:* The entry-specific data portion of the journal entry will be shown in hexadecimal format.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled printer output.

<u>*:</u> Specifies that the output is to be displayed at the requesting work station. For a batch job, specifies that the output is to be listed with the job's spooled printer output.

**LIST:* The output is to be listed with the job's spooled printer output.

**NONE:* The output is to be neither listed nor displayed. OUTPUT(*NONE) is valid only if a value is specified for the OUTFILE parameter.

OUTFILE Parameter: Specifies the physical data base file to which the output from the conversion is to be directed. If the output file already exists, the system will attempt to use it. Records will replace the current data in the file member. If the output file does not exist, the system will create a data base physical file (of the name specified in the OUTFILE parameter) in the designated library. A member will be created for the file with the name specified in the OUTMBR parameter. The format name will be QJORDJE. The format provided defines all the normal output fields for the common fields in each journal entry.

<u>*NONE:</u> Specifies that the output is not to be directed to any data base file.

file-name: Enter the qualified name of the file to receive the converted journal entries. (If no library qualifier is given, *LIBL is used to find the file. If the file is to be created and the file name is not qualified, it is created in QGPL.)

OUTMBR Parameter: Specifies the name of the member within the file that is to receive the converted journal entries. If the OUTFILE is to be created by the system, a member of the same name will also be created for the OUTFILE. If the OUTFILE exists but the OUTMBR does not, a member with the specified name will be added to the output file.

<u>*FIRST:</u> The first member of the file specified by the OUTFILE parameter will receive the converted journal entries. If the OUTFILE is to be created and *FIRST is specified, the name of the created member will be the same as the name of the created file. If OUTMBR is not specified, *FIRST is assumed.

member-name: Enter the name of the member within the file specified by the OUTFILE parameter that is to receive the converted journal entries.

Examples

```
DSPJRN  JRN(JRNLA.MYLIB)
```

When entered at a work station, the above command causes the first series of journal entries in the current journal receiver attached to the journal JRNLA in library MYLIB to be displayed. Subsequent entries can be displayed by using the Roll key. When entered from a batch job, the above command causes all journal entries in the journal receiver currently attached to journal JRNLA in library MYLIB to be listed with the job's spooled printer output.

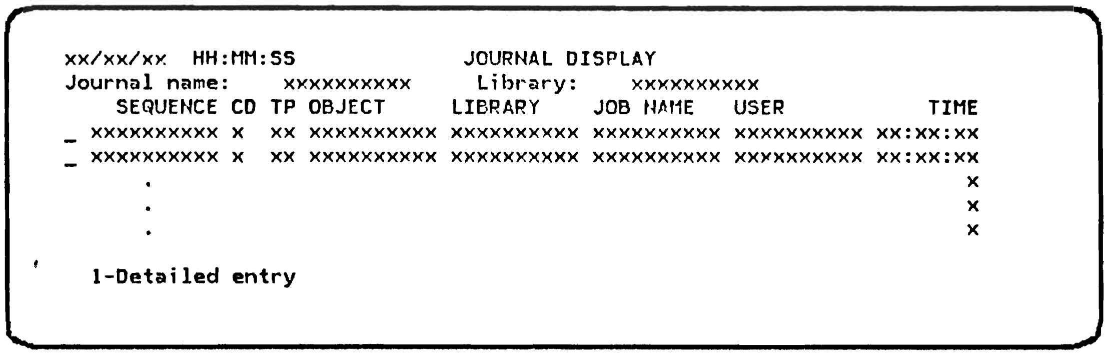
The entry-specific data portion of the journal entries will be shown in character format.

```
DSPJRN  JRN(JRNLA.MYLIB)  FILE((A.LIB1 MBR3)  (C.LIB1)  (*ALL.LIB2*ALL))  +
    RCVRNG((RCV27.RCVLIB  RCV30.RCVLIB))  FROMENT(4736)  +
    ENTTYP(UP DL)  JOB(WORKST01.QPGMR.000666)  PGM(TSTPGMA)  +
    ENTDTALEN(280)  OUTPUT(*NONE)  OUTFILE(JRNENTFIL1.MYLIB)
```

The above command causes selected journal entries in the journal receiver chain (from receiver RCV27 in library RCVLIB to receiver RCV30 in library RCVLIB) that are journaled through journal JRNLA in library MYLIB to be converted and placed in the first member of the data base file JRNENTFIL1 in library MYLIB. If the data base file does not exist, it will be created with a format of QJORDJE. The last field in the format will be 280 bytes in length. The conversion will start with the journal entry that has a sequence number of 4736. Only entries will be converted that reflect 'UPDATE' and 'DELETE' changes made by program TSTPGMA in the job WORKST01.QPGMR.000666 to member MBR3 of file A in library LIB1, the first member of file C in library LIB1, and all members of all files in library LIB2.

Additional Considerations

The information produced by the DSPJRN command can be listed with the job's spooled printer output or displayed at the requesting work station. When the information is directed to the requesting work station, two displays are used. The first display shows basic information for a number of journal entries, and has the following format:



```
xx/xx/xx  HH:MM:SS              JOURNAL DISPLAY
Journal name:     xxxxxxxxxx      Library:     xxxxxxxxxx
    SEQUENCE CD TP OBJECT     LIBRARY    JOB NAME   USER         TIME
_ xxxxxxxxxx x  xx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xx:xx:xx
_ xxxxxxxxxx x  xx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xx:xx:xx
       .                                                              x
       .                                                              x
       .                                                              x

  1-Detailed entry
```

The second line of the display identifies the journal whose entries are being displayed. Beginning with the fourth line of the display, the following information is displayed for each journal entry:

- SEQUENCE. The sequence number of the journal entry
- CD. The journal code of the entry
- TP. The journal type of the entry
- OBJECT LIBRARY. The qualified name of the object for which this journal entry was generated
- JOB NAME. The name of the job that generated the entry
- USER. The name of the user whose job generated the entry
- TIME. The time the entry was generated

For any of the journal entries displayed, detailed information can be displayed by using the detailed entry option. The display produced has the following format:

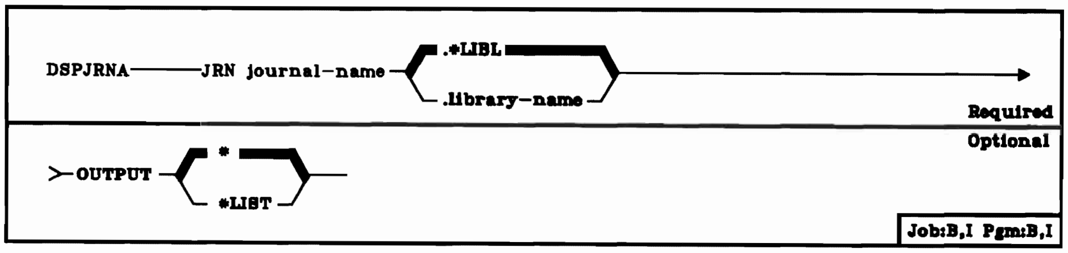
```
xx/xx/xx  HH:MM:SS         JOURNAL ENTRY DISPLAY
Journal name:    xxxxxxxxxx     Library:    xxxxxxxxxx

  Sequence:   x,xxx,xxx,xxx
  Code:                   x
  Type:                  xx
  Object:        xxxxxxxxxx     Library:    xxxxxxxxxx
  Member:        xxxxxxxxxx
  Date:          xx/xx/xx
  Time:          xx:xx:xx
  Job:           xxxxxxxxxx.xxxxxxxxxx.xxxxxx
  Program:       xxxxxxxxxx
  RRN:        x,xxx,xxx,xxx
  Flag:                   x
  Entry specific data:
         *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5
        'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
        'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
        'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
        'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
        'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
        'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'                +
  CF10-Return to journal display
```

Beginning with the second line of the display, information similar to that shown on the first display appears. If the object for which this entry was generated is a data base file, the member name is displayed with the qualified name of the object. Also shown are the date and time the entry was generated, the qualified name of the job and the name of the program that generated the entry, and, if the entry is a record level change, the relative record number. Also shown is the flag indicator value assigned to this journal entry. For more information on the meaning of the flag indicator, refer to the *Programmer's Guide*. Following the value shown as the content of the FLAG field, the entry-specific data portion of the display shows an unformatted representation of the converted journal entry.

DSPJRNA (Display Journal Attributes) Command

The Display Journal Attributes (DSPJRNA) command displays the creation and operational attributes of a journal, including the names of the journal receivers currently attached to the journal. From the display produced, options can be selected to display the names of all physical files currently being journaled, to display the receiver directory, and to display detailed information about a journal receiver.

If output is to be listed with the job's spooled printer output, then all of the information that would be optionally displayed is listed (except for detailed information on journal receivers; for that information, use the DSPJRNRCVA command).



JRN Parameter: Specifies the qualified name of the journal whose attributes are to be displayed. (If a library qualifier is not specified, *LIBL is used to find the journal.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled printer output.

**:* Specifies, for an interactive job, that the output is to be displayed at the requesting work station. For a batch job, the output will be listed with the job's spooled printer output.

**LIST:* The output is to be listed with the job's spooled printer output.

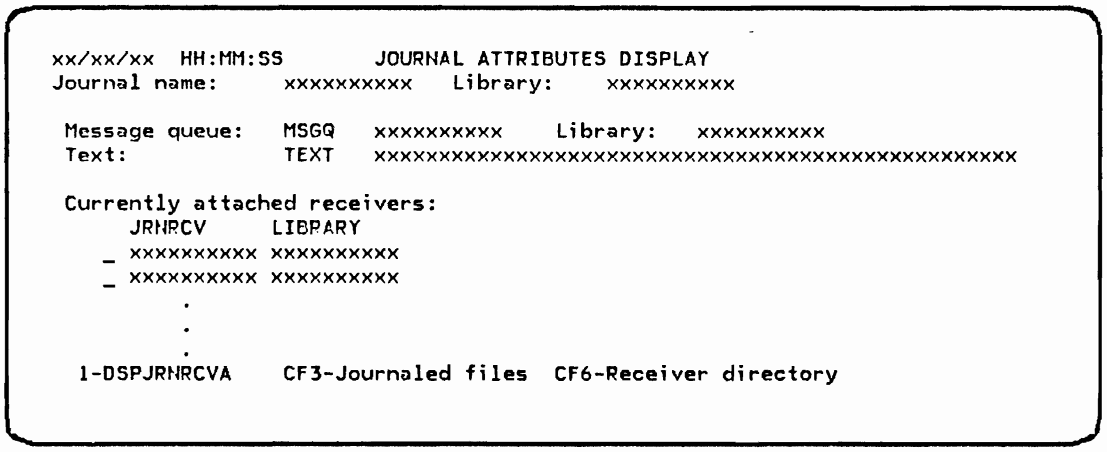
Example

```
DSPJRNA  JRN(JRNLA.MYLIB)
```

The above command causes the current attributes of JRNLA in library MYLIB to be displayed.

Additional Considerations

The DSPJRNA command produces a display of journal information. The information can be listed with the job's spooled printer ouput or it can be directed to the work station that requested it.



```
xx/xx/xx  HH:MM:SS       JOURNAL ATTRIBUTES DISPLAY
Journal name:      xxxxxxxxxx   Library:    xxxxxxxxxx

 Message queue:   MSGQ   xxxxxxxxxx    Library:   xxxxxxxxxx
 Text:            TEXT   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

 Currently attached receivers:
      JRNRCV     LIBRARY
    _ xxxxxxxxxx xxxxxxxxxx
    _ xxxxxxxxxx xxxxxxxxxx
            .
            .
            .
  1-DSPJRNRCVA    CF3-Journaled files  CF6-Receiver directory
```

The first display shows the journal attributes assigned when the journal was created with the CRTJRN command or changed with the CHGJRN command (refer to those command descriptions for explanations of the values).

Also shown on this display is a list of the receivers that are currently attached to the journal. Detailed information about the receivers can be obtained by selecting the DSPJRNRCVA option (refer to the DSPJRNRCVA command for an example and explanation of the display information).

The CF3 key can be used to display a list of qualified names of the files that are being journaled through this journal, in alphabetic order. Only files that contain members will be displayed. The display produced has the following format:

```
xx/xx/xx  HH:MM:SS       JOURNAL ATTRIBUTES DISPLAY
Journal name:     xxxxxxxxxx   Library:    xxxxxxxxxx
                                  JOURNALED FILES
 FILE       LIBRARY       FILE       LIBRARY       FILE       LIBRARY
 xxxxxxxxxx xxxxxxxxxx    xxxxxxxxxx xxxxxxxxxx    xxxxxxxxxx xxxxxxxxxx
 xxxxxxxxxx xxxxxxxxxx    xxxxxxxxxx xxxxxxxxxx    xxxxxxxxxx xxxxxxxxxx
                .                                                      x
                .                                                      x
                .                                                      x

                    Total number of files journaled   --   xxxxxxx
                    Total number of members journaled --   xxxxxxx
```

From the first display, you can also use the CF6 key to request a directory of receivers that have been or are currently attached to the journal. The receiver directory display has the following format:

```
xx/xx/xx  HH:MM:SS       JOURNAL ATTRIBUTES DISPLAY
Journal name:     xxxxxxxxxx   Library:    xxxxxxxxxx
                              JOURNAL PECEIVER DIRECTORY
     NBR  JRNRCV     LIBRARY     ATTACHED      SAVED  STATUS              SIZE
_  xxxxx  xxxxxxxxxx xxxxxxxxxx   x/xx/xx    x/xx/xx  xxxxxxxx  x,xxx,xxx,xxx
_  xxxxx  xxxxxxxxxx xxxxxxxxxx   x/xx/xx    x/xx/xx  xxxxxxxx  x,xxx,xxx,xxx
              .
              .
              .
                           Total size of receivers --       xxx,xxx,xxx,xxx

  1-DSPJRNRCVA    9-DLTJRNRCV
```

The display shows the qualified names of all receivers associated with the journal, in the order of oldest (first attached) to newest (last attached). The information presented for each journal receiver includes:

- NBR. A unique number supplied by the system when the receiver was attached. This number can be used with the date shown in the ATTACHED field to identify members of a common receiver chain associated with this journal. If dual receivers are being used, the number will be the same for both receivers.

- JRNRCV LIBRARY. The qualified name of the journal receiver.

- ATTACHED. The date the receiver was attached to the journal.

- SAVED. The date the receiver was saved (if it was saved).

- STATUS. The status of the receiver, which can be:
  - SAVED. The journal receiver was saved after it was detached.
  - FREED. The journal receiver was saved with storage freed after it was detached.
  - PARTIAL. The journal receiver was restored from a version that was saved while attached to the journal.
  - ON-LINE. The receiver is still online (no full save of the receiver has been performed).
  - ATTACHED. The journal receiver is attached to the journal.

- SIZE. The current size of the receiver or the size when it was detached. The sum of the sizes for all receivers associated with this journal is shown near the bottom of the display.

Detailed information about any receivers can be obtained by selecting the DSPJRNRCVA option (refer to the DSPJRNRCVA command for an example and explanation of the display information).

Receivers can be deleted by entering the DLTJRNRCV option on the appropriate display line. This option executes the DLTJRNRCV command, but with the following restrictions:

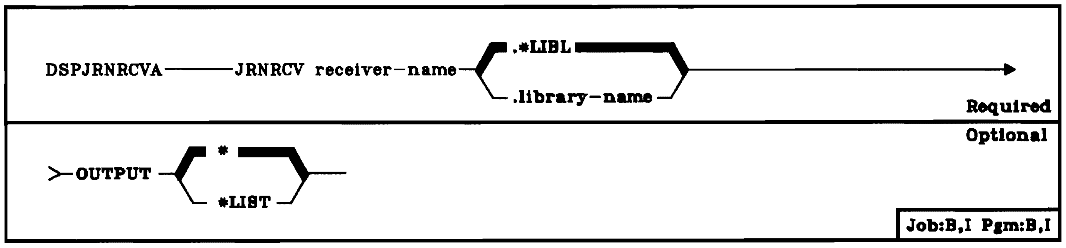
- The journal receiver must not be attached to a journal at the time the command is issued.

- The journal receiver must not be in the middle of a chain of online receivers unless it is damaged. The receivers must be deleted in the same order in which they were detached.

DSPJRNRCVA (Display Journal Receiver Attributes) Command

The Display Journal Receiver Attributes (DSPJRNRCVA) command displays the creation and current operational attributes of a journal receiver, including the name of the journal the receiver is now attached to or was last attached to (if the receiver is not currently attached) and the names of the journal receivers that were attached before and after the specified receiver. If a dual receiver was used, the name of the journal receiver is shown that was attached concurrently with the specified receiver. The information also includes the number of journal entries contained in the journal receiver, the length of the largest journal entry, the journal sequence numbers of the first and last entries on the journal receiver, and the date and time that the receiver was attached and detached.

From the display produced by the command, an option can be selected to display the previous receiver, the next receiver or the dual receiver.



JRNRCV Parameter: Specifies the qualified name of the journal receiver that is to be displayed. (If a library qualifier is not specified, *LIBL is used to find the journal receiver.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled printer output.

: Specifies that the output is to be displayed at the requesting work station. For a batch job, specifies that the output is to be listed with the job's spooled printer output.

**LIST:* The output is to be listed with the job's spooled printer output.

Example

```
DSPJRNRCVA  JRNRCV(JRNRCLA.MYLIB) OUTPUT(*LIST)
```

The above command causes the current status information of journal receiver JRNRCLA in library MYLIB to be listed with the job's spooled printer output.

Additional Considerations

The information produced by the DSPJRNRCVA command can be listed with the job's spooled printer output or displayed at the requesting work station. When the information is directed to the requesting work station, a display is presented with the following format:

```
xx/xx/xx  HH:MM:SS      JOURNAL RECEIVER ATTRIBUTES DISPLAY
Journal receiver:   xxxxxxxxxx          Library:  xxxxxxxxxx
  Unit:          UNIT        xx
  Threshold:     THRESHOLD   x,xxx,xxx,xxx
  Text:          TEXT        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

  Journal name:      xxxxxxxxxx          Library:  xxxxxxxxxx
  Status:            xxxxxxxx
  Attached:          xx/xx/xx xx:xx:xx   Detached:   xx/xx/xx xx:xx:xx
  Saved:             xx/xx/xx xx:xx:xx
  Associated receivers:   JRNRCV               LIBRARY
         Previous:      _ xxxxxxxxxx           xxxxxxxxxx
                        _ xxxxxxxxxx           xxxxxxxxxx
         Next:          _ xxxxxxxxxx           xxxxxxxxxx
                        _ xxxxxxxxxx           xxxxxxxxxx
         Dual:          _ xxxxxxxxxx           xxxxxxxxxx
  Number of entries:         x,xxx,xxx,xxx
  Maximum entry length:      x,xxx,xxx,xxx
  First sequence number:     x,xxx,xxx,xxx
  Last sequence number:      x,xxx,xxx,xxx
  Size:                      x,xxx,xxx,xxx

   1-DSPJRNRCVA           CF10-Return to selection display
```

Information about the journal receiver named on line 2 of the display includes the following:

- Parameter values assigned to the receiver when it was created with the CRTJRNRCV command, including:
  - UNIT. The preferred storage unit on which the receiver is to reside.
  - THRESHOLD. The threshold value for the receiver (when the storage used in the receiver reaches this value, a message is sent to the user).
  - TEXT. The descriptive text used to identify the receiver.

- Journal name: The name of the journal the receiver is now attached to or was last attached to, if the receiver is not currently attached. If the receiver has never been attached to a journal, *NONE will be displayed in this field.

- Status: The current status of the receiver, which can be:
  - SAVED. The journal receiver was saved after it was detached.
  - FREED. The journal receiver was saved with storage freed after it was detached.
  - PARTIAL. The journal receiver was restored from a version that was saved while attached to the journal.
  - ON-LINE. The receiver is still online (no full save of the receiver has been performed).
  - ATTACHED. The journal receiver is attached to the journal.
  - EMPTY. The journal receiver contains no journal entries.

- Attached/Detached: The date and time that the journal receiver was attached to and detached from the journal. If the journal receiver was never attached to a journal, both fields will be displayed as zeros.

- Saved: The date and time the journal receiver was saved.

- Associated receivers:
  - Previous: The qualified names of the journal receivers that were detached when this receiver was attached. These journal receivers can be individually displayed with the DSPJRNRCVA option.
  - Next: The qualified names of the journal receivers that were attached when this receiver was detached. These journal receivers can be individually displayed with the DSPJRNRCVA option.
  - Dual: The qualified name of the journal receiver that was attached when this receiver was attached (the other receiver in a dual receiver pair). This receiver can be displayed with the DSPJRNRCVA option.

- Number of entries: The total number of journal entries on this receiver.

- Maximum entry length: The length of the longest journal entry on this receiver. This information can be useful in determining the field length to specify (in the ENTDALEN parameter of the DSPJRN command) for the entry-specific portion of the output file. The length of the longest entry may be larger than the entry displayed by the DSPJRN command, due to selection and limitation criteria and the reduction in length occurring during conversion to the external format.

- First sequence number: The sequence number of the first journal entry on this receiver.

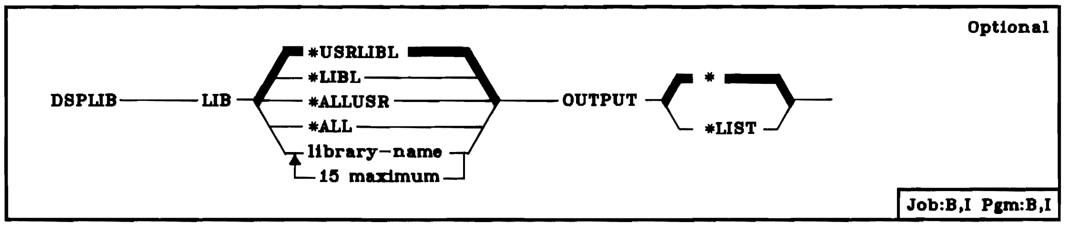
- Last sequence number: The sequence number of the last journal entry on this receiver.

- Size: The size of the journal receiver when it was detached or its current size if it is attached.

DSPLIB (Display Library) Command

The Display Library (DSPLIB) command displays the contents of one or more specified libraries; that is, it displays a list of the names and types of all objects contained within each library, regardless of the authorization on each object. (If you want a list of the library names, or you want to display the *description* of one or more libraries, use the DSPOBJD command.)

Restriction: The user must have read rights for each library specified in order to display its contents.



LIB Parameter: Specifies the names of one or more libraries that are to have their objects (names and types) displayed or printed. If the user does not have read rights for a specified library, that library is not displayed.

*USRLIBL: The libraries named in the user portion of the job's library list are the libraries whose objects are to be displayed or printed.

**LIBL:* The libraries named in the job's library list are the libraries whose objects are to be displayed or printed.

**ALLUSR:* All the nonsystem libraries that the user owns or is authorized to use are to be displayed. Nonsystem libraries include all user-defined libraries and the QGPL library, not just those in the job's library list. Libraries other than QGPL that begin with the letter Q are not included.

**ALL:* All of the libraries that the user owns or is authorized to use are to be displayed.

library-name: Enter the names of one or more libraries whose objects are to be displayed. Any system and/or user-defined library for which the user has operational rights can be specified.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

DSPLIB LIB(QGPL)

The names, types, and basic descriptions of all the objects located in the library QGPL are displayed on the work station from which the command was submitted, or are printed on the system printer if the command was executed in a batch job.

Additional Considerations

The following display format results from the DSPLIB command:

```
XX/XX/XX  XX:XX:XX    LIBRARY DISPLAY                  XXX
LIBRARY:      XXXXXXXXXX   TYPE:        XXXXXXXXXX
NBR OF OBJS:  XXXXXXXXXX   TOTAL SIZE:  XXXXXXXXXX
  OBJECT     TYPE     ATTR      FREED      SIZE TEXT
  XXXXXXXXXX XXXXXXXX XXXXXXXXXX XXX XXXXXXXXXX XXXXXXXXXXXXXXXX
  XXXXXXXXXX XXXXXXXX XXXXXXXXXX XXX XXXXXXXXXX XXXXXXXXXXXXXXXX
```

If more than one library is to be displayed, they are displayed one at a time and in the following order, depending upon what is specified in the LIB parameter:

- If *USRLIBL or *LIBL is specified, the libraries are displayed in the *positional* order in which they occur in the user portion, or in both the user and system portions, of the job's library list.

- If *ALLUSR or *ALL is specified, all the user-defined libraries (including QGPL), or all the libraries, in the system for which the user has read rights are displayed in *alphabetic* order by library name.

- If multiple library names are specified, the libraries are displayed in the order specified.

Lines 2 and 3 of the display list the following:

- The name and type (production or test) of each library.
- The number of objects in the library.
- The size of the combined objects. If this value is followed by a plus (+) sign, all objects in the library are not currently available, are damaged, being locked, or are not authorized. The plus sign indicates that the actual total of all objects is greater than the value displayed.

For each object in the library, the following information is displayed:

- The name and type of the object. Objects for which the user has no operational authority show names and types only; no additional attributes are displayed.
- Any extended attributes or status indicators. This field lists types of high-level languages or status information of objects not currently available. Valid values grouped by object type are:

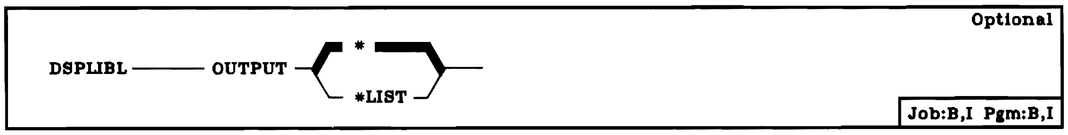
Object Type	**Values**		
File	BSC	DKT	PHY
	CMN	DSP	PRT
	CRD	LGL	TAP
Library	PROD	TEST	
Program	CBL	DFU NOTEXC	
	CL	DFU EXC	
	DFU	QRY NOTEXC	
	QRY	QRY EXC	
	RPG		

- Whether storage for the object has been freed.
- The size of the object, in bytes. If the object has been allocated to another user, the size of the object is not displayed or included in the total size value for all objects in the library.
- The text description of the object.

The object names are listed in alphabetic order within each object type. Multiple displays for all the objects in one library are presented before the next library can be displayed.

DSPLIBL (Display Library List) Command

The Display Library List (DSPLIBL) command displays the name and position of each library in the user's portion of the library list associated with his job. Only the libraries that are in the user portion of the job's library list are displayed; the system portion of the library list is not displayed.



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPLIBL
```

The names and positions of the libraries that are in the user portion of the job's library list are displayed on the user's work station from which the command was submitted, or the list is printed on the system printer if the command was part of the batch input stream.

Additional Considerations

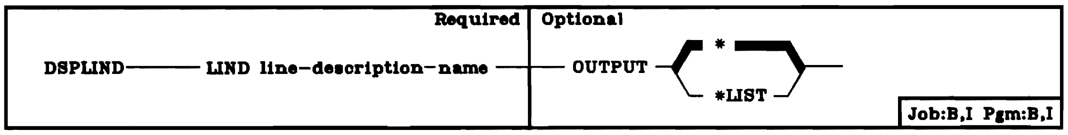
The display produced by the DSPLIBL command shows the names of the libraries that are in the user portion of the library list associated with the user's job.

```
XX/XX/XX  XX:XX:XX   USER LIBRARY LIST DISPLAY
POSITION  LIBRARY            POSITION  LIBRARY            POSITION  LIBRARY
   XX     XXXXXXXXXX            XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX
   XX     XXXXXXXXXX            XX     XXXXXXXXXX
```

The libraries are listed in the order in which they are searched for object names that are specified without their library qualifiers.

DSPLIND (Display Line Description) Command

The Display Line Description (DSPLIND) command displays information about the specified line and its description.



LIND Parameter: Specifies the name of the line description to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

**:* The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPLIND  LIND(LINE21)
```

This command displays information about the line description named LINE21. The information is displayed on the work station from which the command was submitted; or it is spooled to a printer output queue to be printed on the system printer, if the command was part of the batch input stream.

Additional Considerations

Five displays are used to show the attributes of the line description identified in the DSPLIND command. The attributes are displayed in the same order as their associated parameters (whose keywords are shown here) occur in the CRTLIND command. For an explanation of each attribute of the line description, refer to the associated parameter description given in the CRTLIND command description.

Note: A blank attribute field indicates that the associated parameter is not valid for the given line type.

The first display contains the first group of attributes in the line description. The name of the line description being displayed appears on the third line.

```
 XX/XX/XX  XX:XX:XX   LINE DESCRIPTION                       +++
Status: XXXXXXXXXXXXXXXXXXXX
Line description name:          LIND        XXXXXXXXXX
OU number of line port:         LINNBR      XX
Line type:                      TYPE        XXXXX
Connection type:                CNN         XXXX
Data rate:                      RATE        XXXXXX
Switched network backup:        SWNBKU      XXXX
Activate swt network backup?    ACTSWNBKU   XXXX
Speed select feature:           SELECT      XXXX
```

Also shown on the first display is the *status* field (on line 2), which is not specified in the CRTLIND command; however, the operational status of the line is kept current in the line's description. The status field indicates one of the following about the line:

- ACTIVE. Data is currently being transmitted over the line.
- DIAGNOSTIC MODE. The line is being serviced.
- VARIED OFF. The line is varied offline.
- VARIED ON. The line is varied online.
- CONNECTION PENDING. A VRYLIN command has been issued, and the system is waiting for the action to be completed.
- FAILED. The line is in an unusable state; it can possibly be made usable again by varying it off, then on.

The second, third, fourth, and fifth displays contain the rest of the attributes in the line description. All but two of the attributes were specified in the CRTLIND command. The attached *switched* control unit attribute (on the fourth display) specifies the name of a control unit that is attached to this line only while the physical switched connection is established. This line description was associated with the control unit description in the LINLST parameter of the CRTCUD (or CHGCUD) command, which is used to describe 5251 PU2 and BSC control units only. There may be as many as 50 attached nonswitched line control units; however, only the first 10 of them are displayed here.

```
 XX/XX/XX  XX:XX:XX   LINE DESCRIPTION                    +++
NRZI decoding:      :          NONRTNZ    XXXX
S/38-provided clock:           CLOCK      XXXX
Autocall feature:              AUTOCALL   XXXX
Autoanswer feature:            AUTOANS    XXXX
S/38 answer tone feature:      ANSTONE    XXXX
Wire link type:                WIRE       X X
Data comm equipment group:     DCEGRP     XX
Non-IBM modem:                 OEMMDM     XXXX
Switched connection type:      SWTCNN     XXXXX
Speed rate type:               RATETYPE   XXXXX
```

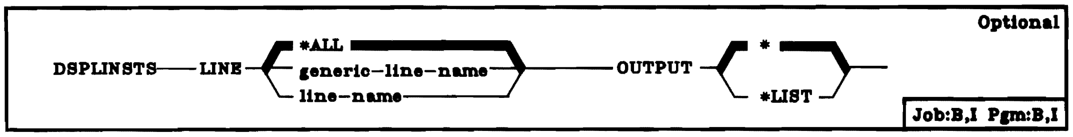
```
 XX/XX/XX  XX:XX:XX   LINE DESCRIPTION                    +++
Dial mode:                       DIALMODE    XXXXXXX
Answer mode:                     ANSMODE     XXXXXXX
Data terminal ready delay:       DTRDLY      XXXX
Idle detection time:             IDLETIME    XXXX
BSC receive timeout timer:       RCVTMR      XXXX
Nonproductive receive time:      NONPRDRCV   XXXX
Number of error retries:         RETRY       XXXX
Online at CPF start:             ONLINE      XXXX
```

```
 XX/XX/XX  XX:XX:XX   LINE DESCRIPTION                   +++
Attached switched control unit:           XXXXXXXXXX
Attached nonswitched ctl units:  CTLU
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
BSC switched control units:      SWTCTLU
  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
S/38 SDLC secondary address:     STNADR     XX
S/38 exchange identifier:        EXCHID     XXXXXXXX
BSC character code:              CODE       XXXXXXX
```

```
 XX/XX/XX  XX:XX:XX    LINE DESCRIPTION
Text description:                TEXT
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

DSPLINSTS (Display Line Status) Command

The Display Line Status (DSPLINSTS) command displays the hierarchial configuration of specified lines on a system, with the attached control units and devices. The status of each of these objects is also displayed, along with the job names of all interactive, batch, autostart, reader, or writer jobs that are holding a lock on a device.



LINE Parameter: Specifies whether the status of all lines and attached control units and devices on the system is to be displayed, or if only the status information for a specific line and its attached control units and devices is to be displayed.

*ALL: The status information for all lines and attached control units and devices on the system is to be displayed.

generic-line-name: The status information is to be displayed for the specified line and its attached control units and devices, or for all lines with the same generic name. To specify a generic name, add an asterisk after the last character in the name (ABC*, for example). If an asterisk is not included with the name, the system assumes that the name is a complete line name.

line-name: The status information for the specified line and its attached control units and devices is to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or is to be listed with the job's spooled printer output. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the file(s) used by this command.)

*: The output is to be displayed (if requested by an interactive job) or listed with the job's system output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled printer output.

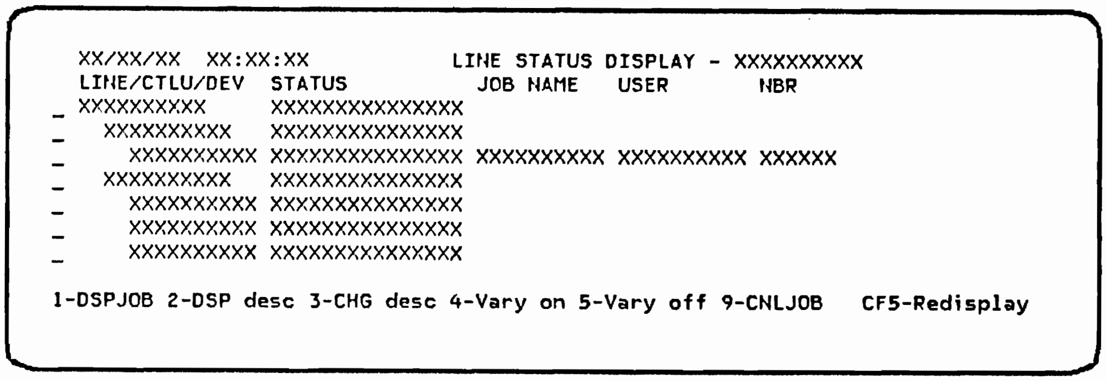
Example

DSPLINSTS LINE(LINE1)

This command displays the name and status of line LINE1, the name and status of any attached control units, and the name and status of any devices attached to the control unit(s), as well as the names of any jobs using an active device. The information is displayed on the work station from which the command was submitted, or it is spooled to a printer output queue to be printed on the system printer, if the command was part of the batch input stream.

Additional Considerations

The displays produced by the DSPLINSTS command will show various amounts of line configuration information, depending on what is currently attached to the line. As an example, if you specify DSPLINSTS LINE line-name (and the line has connected to it a control unit with an active device), the following display is shown:



```
   XX/XX/XX  XX:XX:XX          LINE STATUS DISPLAY - XXXXXXXXXX
   LINE/CTLU/DEV  STATUS          JOB NAME   USER       NBR
 _ XXXXXXXXXX     XXXXXXXXXXXXXXX
 _   XXXXXXXXXX   XXXXXXXXXXXXXXX
 _     XXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 _   XXXXXXXXXX   XXXXXXXXXXXXXXX
 _     XXXXXXXXXX XXXXXXXXXXXXXXX
 _     XXXXXXXXXX XXXXXXXXXXXXXXX
 _     XXXXXXXXXX XXXXXXXXXXXXXXX

 1-DSPJOB 2-DSP desc 3-CHG desc 4-Vary on 5-Vary off 9-CNLJOB   CF5-Redisplay
```

Header Information

The first line of the display shows the current job date and time fields followed by the value specified for the LINE parameter (either '*ALL', a generic line name, or a specific line name).

First-column Input Field

The leftmost column of the display consists of a single-digit input field in which a number can be entered. You can enter the number (any one of those shown at the bottom of the display) in the input field to cause the function (a command) associated with that number to be performed for that line/control-unit/device when you press the Enter key. If you enter numbers in the input fields of several items before pressing the Enter key, the specified functions are performed on the items in the order in which the items are displayed. The following functions can be specified:

1–DSPJOB:	Executes the DSPRDR command for a reader job, executes the DSPWTR command for a writer job, executes the DSPJOB command for all other jobs, or returns 'Job .. not found' if no job is associated with the input record.
2–DSP desc:	Executes the DSPLIND command for a line, executes the DSPCUD command for a control unit, or executes the DSPDEVD command for a device.
3–CHG desc:	Prompts for the CHGLIND command for a line, prompts for the CHGCUD command for a control unit, or prompts for the CHGDEVD command for a device.
4–Vary on:	For a line, 1) the line is varied on, 2) all attached control units are varied on, or 3) all attached devices are varied on. For a control unit, 1) the control unit is varied on, or 2) all attached devices are varied on. For a device, the device is varied on.
5–Vary off:	For a line, 1) all attached devices are varied off, if possible, 2) all attached control units are varied off, if possible, or 3) the line is varied off, if possible. For a control unit, 1) all attached devices are varied off, if possible, 2) the control unit is varied off, if possible. For a device, the device is varied off. **Note:** As is the case when the VRYxxx commands are entered individually, there is a noticeable delay when varying off a line/control-unit/device that has already been varied off.
9–CNLJOB:	Executes the CNLRDR command (with OPTION(*CNTRLD)) for a reader job, executes the CNLWTR command (with OPTION(*CNTRLD)) for a writer job, executes the CNLJOB command (with OPTION(*CNTRLD)) for all other types of jobs, and returns 'Job .. not found' if no job is associated with the input record.

When all of the commands have been executed, the display is shown again with the status fields of the objects updated and with any error messages that occurred when the commands were executed. The display can be shown at any time *before* all the commands have been executed by pressing the CF5 key.

If the configuration has more elements than can be shown on a single display, the Roll Up key can be used to display them all. You can enter numbers in the input fields on multiple displays before pressing the Enter key.

After the commands have been executed, if there are more error messages than can fit on that display, a '+' is shown at the end of the last message. You must position the cursor at the first message and use the Roll Up key to view all of the error messages.

Line/Ctlu/Dev

The second vertical column displays the name of the item whose information is being displayed on that line. Control unit (CTLU) names are indented two spaces and device (DEV) names are indented four spaces.

Status

The third column lists the status of the line, control unit, or device. One of the following values is used to indicate status:

- ACTIVE. The line, control unit, or device is currently in use. For a display device, the device is signed on or has been allocated by a batch, autostart, or interactive job.
- ACTIVE/RDR. A spool reader is using this device.
- ACTIVE/WTR. A spool writer is using this device.
- CONNECT PENDING. A VRYLIN command has been issued for this line, and the system is waiting for an action to be completed, such as a switched connection to be made.
- DIAGNOSTIC MODE. The line, control unit, or device is being serviced or has otherwise been set to diagnostic mode.
- FAILED. The line, control unit, or device is in an unusable state; it can possibly be made usable again by varying it off, then on. A failed device may still be allocated to a job.
- FAILED/RDR. This device, which is in an unusable state, is still allocated to a spool reader job.
- FAILED/WTR. This device, which is in an unusable state, is still allocated to a spool writer job.

- POWERED OFF. The control unit or device is in a varied-off and powered-off state.

- SIGNON DISPLAY. This display device currently has the 'Enter password to signon' screen displayed.

- SYSREQ. This display device has been requested by the system, and the job associated with this status does not have a lock on the device. SYSREQ status coexists only with an ACTIVE or SIGNON DISPLAY status for this device.

- VARIED OFF. For a control unit or device that can be powered on or off by the PWRCTLU or PWRDEV command, the status is indicated after the control unit or device is powered on. For a line, this status indicates that the line is varied off.

- VARIED ON. The line, control unit, or device is varied online, although it may not be physically powered on.

- VARY ON PENDING. A VRYCTLU or VRYDEV command has been issued for this control unit or device, respectively, and the system is waiting for an action to be completed, such as a switched connection to be made.

- *DAMAGED. The line, control unit, or device has incurred hard or partial damage; it is not possible to obtain any further status information.

- *LOCKED. The line, control unit, or device is allocated to another job with an *EXCL lock, and its attributes can not be determined at this time.

- *UNKNOWN. All of the status bits for the line, control unit, or device have been checked, and none are set. This is an exceptional condition.

Job Name

The fourth column shows the names of the jobs that are currently using any of the named devices.

User

The fifth column shows the name of the user profile under which the job that holds the lock on the device is running.

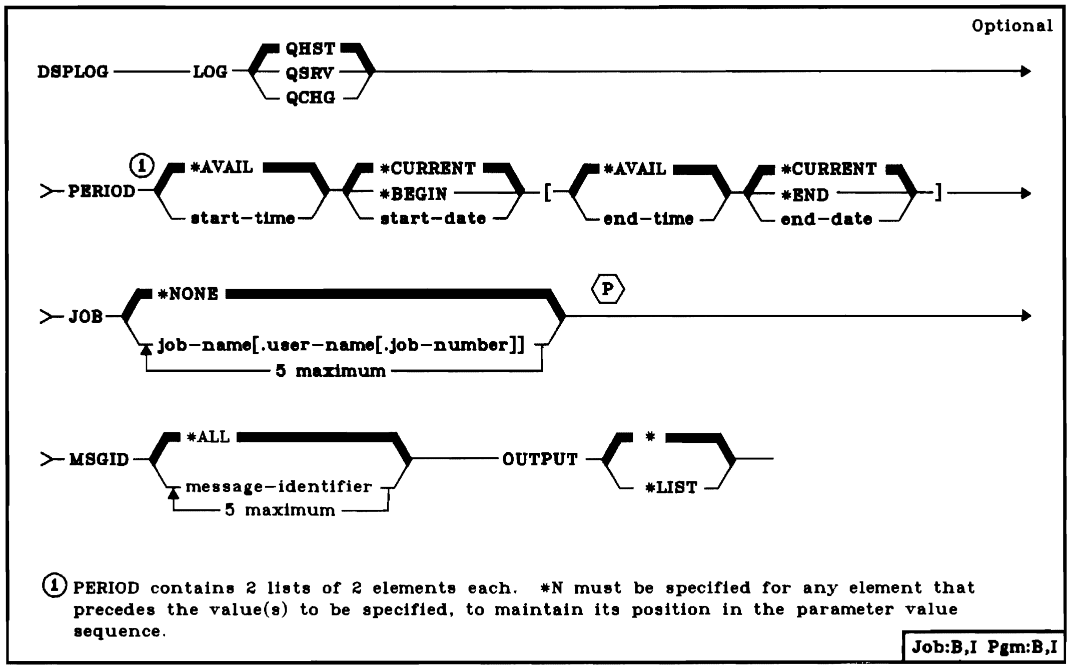
Nbr

The rightmost column shows the six-digit number assigned by the system to identify the job.

DSPLOG (Display Log) Command

The Display Log (DSPLOG) command displays the specified system log. The system history log (QHST), the system service log (QSRV), or the programming change log (QCHG) can be specified. The history log contains information about the normal operation of the system and system status. The service log contains information that is used to service the system. The programming change log contains information about programming changes and program patches that have been loaded, applied, or deleted.

The display contains the messages sent to the log, the date and time the message was sent, and the name of the job that sent it.



① PERIOD contains 2 lists of 2 elements each. *N must be specified for any element that precedes the value(s) to be specified, to maintain its position in the parameter value sequence.

LOG Parameter: Specifies which of the system logs is to be displayed.

QHST: The system history log QHST is to be displayed.

QSRV: The service log QSRV is to be displayed.

QCHG: The programming change log QCHG is to be displayed.

PERIOD Parameter: Specifies the period of time for which the logged message data is to be displayed. The following values can be coded in this parameter, which contains two lists of two elements each. If PERIOD is not specified, the following values are assumed:

PERIOD((*AVAIL *CURRENT) (*AVAIL *CURRENT))

Starting Time: One of the following is used to specify the starting time at which or after which the data must have been logged. Any entries logged before the specified time and date are not displayed.

*AVAIL: The logged data that is available for the specified starting *date* is to be displayed.

start-time: Enter the starting *time* for the specified starting date that indicates the logged data to be displayed. The time can be entered as four or six digits (hhmm or hhmmss) where hh = hours, mm = minutes, and ss = seconds. If colons are used to separate the time values, the string must be enclosed in apostrophes ('hh:mm:ss').

Starting Date: One of the following is used to specify the starting date on which or after which the data must have been logged. Any entries logged before the specified date are not displayed.

*CURRENT: The logged data for the current day and between the specified starting and ending times (if specified) is to be displayed.

**BEGIN:* The logged data from the beginning of the log is to be displayed.

start-date: Enter the starting date for which logged data is to be displayed. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

Ending Time: One of the following is used to specify the ending time before which the data must have been logged.

*AVAIL: The logged data that is available for the specified ending *date* is to be displayed.

end-time: Enter the ending time for the specified ending date that determines the logged data to be *printed.* The time can be entered as four or six digits, and colons can also be used ('hh:mm:ss').

Note: The values specified for the *ending* date and time are ignored if the output is to be displayed. That is, *all* data in the log that was logged on or after the specified starting date and time can be displayed, regardless of the ending date and time specified.

Ending Date: One of the following is used to specify the ending date before which or on which the data must have been logged.

*CURRENT: The current day is the last day for which logged data is to be displayed.

**END:* The last day on which data was logged is the last day for which the logged data is to be displayed.

end-date: Enter the ending date for which logged data is to be *printed*. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

Note: If no output is received after you execute the DSPLOG command with *LIST specified, the dates of some message data may be out of sequence. To *LIST the data in that event, specify PERIOD((*AVAIL *BEGIN)(*AVAIL *END)).

JOB Parameter: Specifies the names of the jobs (if any) for which messages in the system log are to be displayed.

*NONE: No job name is used to indicate which messages are to be displayed.

qualified-job-name: Enter the qualified names of one or more jobs (no more than five) that are to have their logged messages displayed. If a job name is not qualified, all jobs by that name in the log will have their messages displayed. (For an expanded description of the job name, see *JOB Parameter* in Appendix A.)

MSGID Parameter: Specifies the message identifiers (if any) of the logged messages that are to be displayed. These messages are displayed only if they were logged in the period of time specified in this command and in the jobs specified, if job names were given in this command.

*ALL: All logged messages, regardless of their identifiers, are displayed if they meet the previous parameters' specifications.

message-identifier: Enter the identifiers of one or more messages (no more than five) that are to be displayed. (Refer to the description of the MSGID parameter in the ADDMSGD command description for more information on message identifiers.)

To display specific generic types of messages, specify the 3-character code that identifies the message file followed by all zeros. (For example, CPF0000 specifies that all CPF messages that meet the specifications of the previous parameters are to be displayed.) If an identifier is specified as pppnn00, any message beginning with the specified 5 characters can be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPLOG  LOG(QHST)
```

This command displays all the logged messages (and their associated data) that are available in the history log for the current date.

```
DSPLOG  JOB(MYJOB)  PERIOD((*AVAIL 090178) (*AVAIL 093078)) +
      MSGID(CPF0000)
```

This command displays all CPF messages, in the history log for MYJOB, that were logged during September 1978.

Additional Considerations

The display produced by the DSPLOG command has the following format:

```
 XX/XX/XX  XX:XX:XX     SYSTEM LOG - XXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
CF7 - Display all messages
```

This display shows the messages that were sent to the system log specified on the LOG parameter after the time indicated on the PERIOD parameter (the ending time is ignored for *displayed* output). Depending upon the values specified on the JOB and MSGID parameters, the messages displayed can also be restricted to those for specific jobs or those having specific message identifiers.

The first line of the display identifies which system log is being displayed (QHST, QSRV, or QCHG), and the current job date and time.

Lines 2 through 10 display the messages that were sent to the specified log and that meet the requirements specified on the DSPLOG command. For each message, only the first-level message text is displayed and only one line is used for each message. If the text string for the message is longer than one display line, the end of the first-level text is truncated. If more messages follow the last message shown on the display, a plus sign (+) is displayed on the right end of the bottom line.

To display the complete message text, or other information about the message (such as its message identifier or second-level text), position the cursor in the space preceding the message (on the same line) and press the Help key. For an explanation of the second-level text display, refer to *Additional Considerations* in the DSPMSG command description.

The Roll keys are used to roll forward or backward through the messages that can be displayed. You can roll beyond either the specified start or stop times, provided the required version of the log files are online. Also, the CF7 key can be used to override all the values specified on the DSPLOG command except for the type of log being displayed. If CF7 is pressed, *all* the messages in the log are displayed, starting with the message on the line at which the cursor is positioned when the CF7 key is pressed. The Roll keys can be used to roll forward or backward through all the messages, regardless of the time the message was sent, what job sent it, or what its message identifier is.

If OUTPUT(*LIST) is specified on the DSPLOG command, only the messages selected by the LOG, PERIOD, JOB, and MSGID parameters are listed on the printer. Each logged message is printed in the same format used to print the job log, and contains the following information:

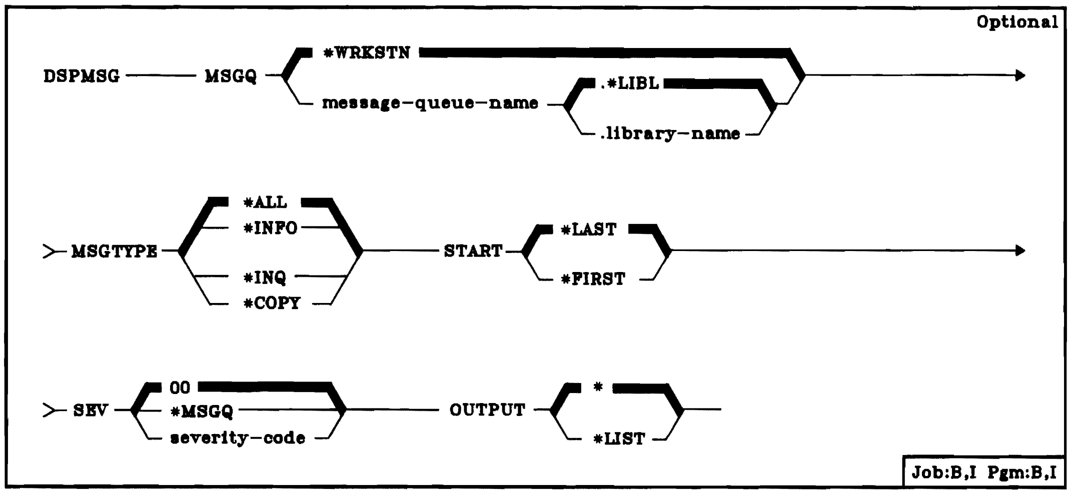
- The message identifier
- The severity code of the message
- The type of message
- The first-level message text (not truncated)
- The name of the job sending the message
- The date and time when the message was sent

If the printed information is from the service log (QSRV), the following is also provided:

- The name of the program that sent the message
- The statement number of the statement within the sending program causing the message
- The name of the program that received the message
- The number of the statement within the receiving program that handled the message

DSPMSG (Display Messages) Command

The Display Messages (DSPMSG) command is used by the work station user to display the messages received at the specified message queue. If the message queue is not allocated to the job in which this command is entered, it is implicitly allocated by this command for the duration of the command. When the messages are displayed, options are also displayed that allow the user to remove one or more messages from the queue or enter a reply to each inquiry message.



MSGQ Parameter: Specifies the name of the message queue from which messages are to be displayed.

<u>*WRKSTN:</u> Messages are displayed from the work station's own message queue.

qualified-message-queue-name: Enter the qualified name of the message queue from which messages are displayed. (If no library qualifier is given, *LIBL is used to find the queue.) The system operator can specify QSYSOPR to display messages sent to the system operator message queue.

MSGTYPE Parameter: Specifies the type of messages in the message queue that are to be displayed.

<u>*ALL:</u> All messages that are in the message queue are to be displayed.

**INFO:* Only informational messages (those not requiring a reply) are to be displayed.

**INQ:* Only inquiry messages (those requiring a reply) are to be displayed.

**COPY:* A copy of the sender's messages, each requiring a reply, that were sent to other message queues is to be displayed.

START Parameter: Specifies, whether the newest messages or the oldest messages in the message queue (of those specified by the MSGTYPE and SEV parameters) are to be displayed first. The work station user can roll the display up or down to see other messages, if the message list occupies more than one display screen. This parameter is ignored if the output is to a spooled printer file.

*LAST: The last (newest) message on the message queue is to be displayed on the bottom line of the display. The work station user can press the Roll Down key to display older messages.

**FIRST:* The first (oldest) message on the message queue is to be displayed on the top line of the display. The work station user can press the Roll Up key to display newer messages.

SEV Parameter: Specifies the lowest value for the severity code that a message can have and be displayed. If the message's severity code is lower than the value specified here, the message will not be displayed.

00: All messages in the specified message queue are to be displayed.

**MSGQ:* All messages having a severity code greater than or equal to the severity code specified for the message queue are to be displayed.

severity-code: Enter a value, 00 through 99, that specifies the lowest severity code that a message can have and still be displayed. (For an expanded description of severity codes and a list of the IBM-defined codes, see *SEV Parameter* in Appendix A.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

DSPMSG

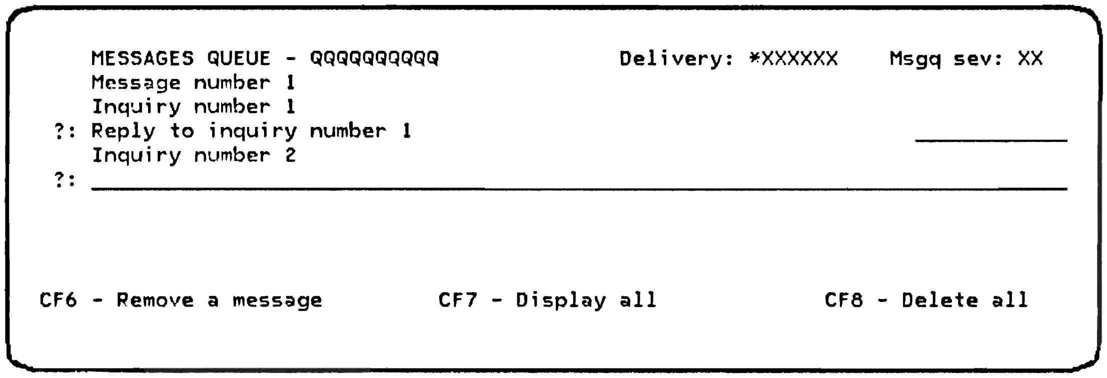
This command displays any new messages in the work station queue associated with the requester's work station.

DSPMSG MSGQ(SMITH) MSGTYPE(*INFO)

This command displays on the requester's work station any new informational messages in the message queue named SMITH.

Additional Considerations

Two display formats are used to display the complete information about messages sent to the message queue whose contents are being displayed. The first display is presented when the DSPMSG command is entered, or when a message arrives at the queue and the queue is set to break delivery mode.



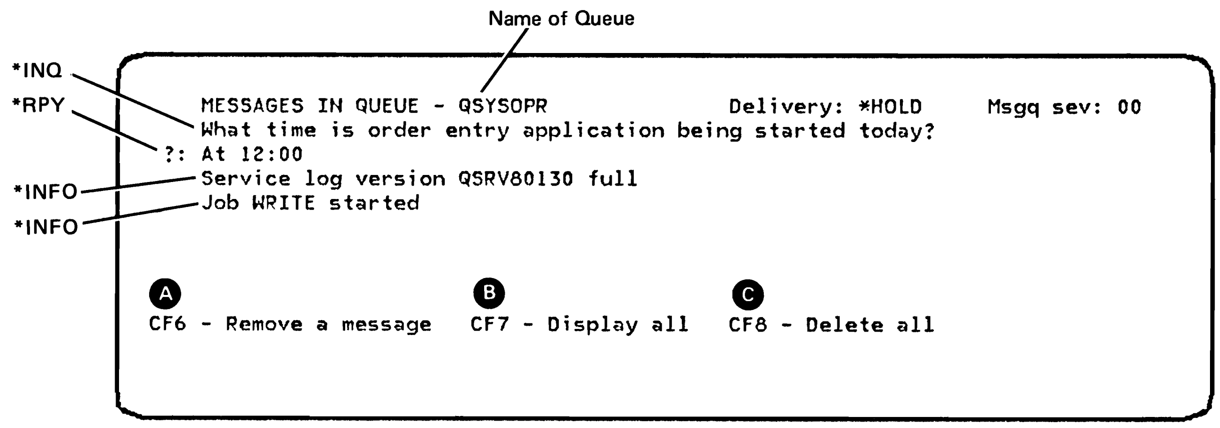
```
    MESSAGES QUEUE - QQQQQQQQQQ            Delivery: *XXXXXX    Msgq sev: XX
    Message number 1
    Inquiry number 1
 ?: Reply to inquiry number 1                                   ____________
    Inquiry number 2
 ?: ____________________________________________________________________________



CF6 - Remove a message        CF7 - Display all            CF8 - Delete all
```

The messages displayed can be restricted to those of a certain type and/or severity code, depending on the values specified in the MSGTYPE and SEV parameters. Also, the oldest or newest message of those specified can be displayed first. If the oldest message is to be first, it is immediately followed (on the same display) by the next oldest. If the newest is to be first (the default on the START parameter), it is displayed on the *bottom* line with older messages above it.

Line 1 gives the name of the message queue being displayed and the delivery mode and severity of messages sent to the queue; lines 2 through n display the selected messages, where n depends on the size of the screen (lines 2 through 10 in this case). For each message, only the first-level message text is displayed and only one line is used for each message stored in a message file (up to two lines are used for impromptu messages). If the text string for the message is longer than one display line, the end of the first-level text is truncated. If more messages follow the last message shown on the display, a plus sign (+) is displayed on the right end of the bottom line.

The display shown on the preceding page shows the format of each type of message appearing on the first-level text display. Line 2 shows an information message line (of MSGTYPE(*INFO)). Lines 3 and 5 show inquiry messages (*INQ) to which responses can be entered by the user on lines 4 and 6 (*RPY); line 4 shows the reply to the inquiry message of line 3.

Assuming that MSGTYPE(*ALL) is specified on the DSPMSG command, an example of some messages that might appear on a work station follows:



```
   MESSAGES IN QUEUE - QSYSOPR            Delivery: *HOLD     Msgq sev: 00
   What time is order entry application being started today?
?: At 12:00
   Service log version QSRV80130 full
   Job WRITE started

CF6 - Remove a message   CF7 - Display all   CF8 - Delete all
```

The message types for each message are indicated by the predefined values shown to the left of each line. These message types are all described in the MSGTYPE parameter description of the RCVMSG command.

You can use command function key 6 (indicated at A) to remove a message by first positioning the cursor anywhere on the message line that is to be deleted. The CF8 key (at C) can be used to delete all the messages in the queue. If an inquiry message is deleted by the CF6 key, a default reply is sent to the message queue of the requester, if a reply hasn't already been sent. If the CF8 key is pressed, the appropriate default reply is sent for each inquiry message in the queue for which no reply has yet been sent.

The CF7 key (at B) can be used to override all the values specified on the MSGTYPE, START, and SEV parameters on the DSPMSG command and cause *all* the messages in the specified queue to be displayed, beginning with the first message on the display (if the cursor is on line 1) or with the message on the line where the cursor is positioned when the CF7 key is pressed. The Roll keys are used to roll forward or backward through the messages that can be displayed.

To display the information on the second display, which is the *second-level message display*, the Help key must be used. The second display shows all the available information about the message selected by the position of the cursor on the first display when the Help key was pressed.

```
                    SECOND LEVEL MESSAGE DISPLAY
 Msg ID: XXXXXXX  Sev: XX  Type: XXXXXXXXXX   XX/XX/XX  XX:XX:XX
 Job: XXXXXXXXXX User: XXXXXXXXXX Nbr: XXXXXX
 From pgm: XXXXXXXXXXX  Inst: XXXX To pgm: XXXXXXXXX  Inst: XXXX
    XXXXXX  (first level message text)  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    XXXXXX  (second level message text)  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

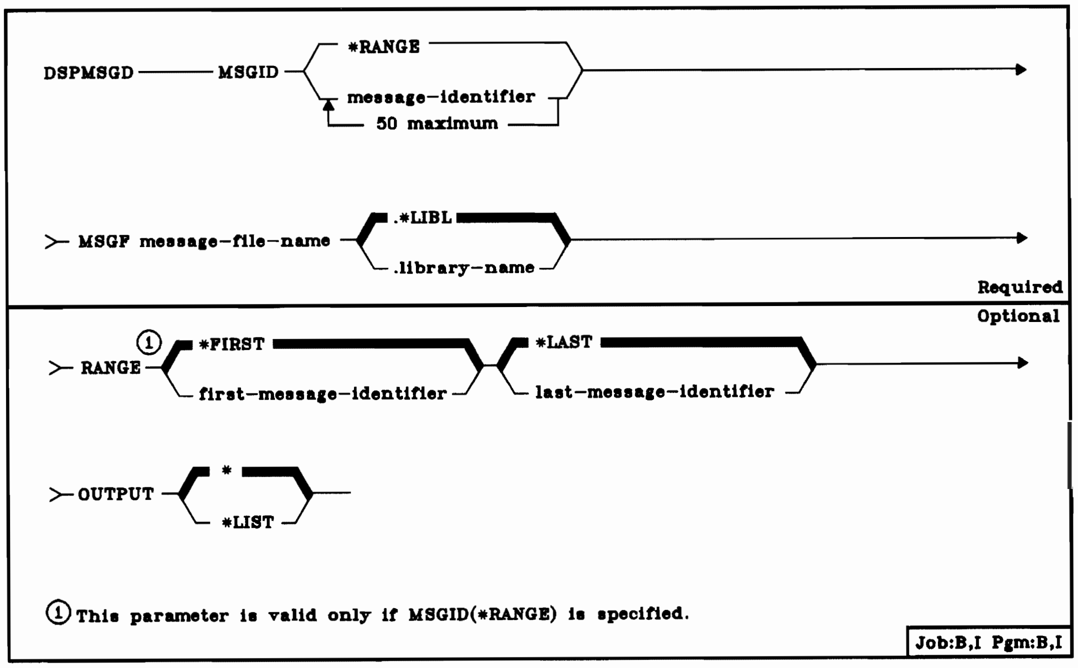
Beginning with line 2, the information shown on the second-level display for the specified message is:

- The message identifier of the specified message
- The severity code of the message
- The type of the message
- The date and time the message was sent to the message queue, if the message queue requires this information
- The qualified name of the job (shown in three fields) that sent the message
- The name of the program (and the number of the statement within the program) that sent the message, and the name of the program (and the number of its statement) that handled the message
- The complete text of the first-level message (on lines 5 and 6)
- The complete text of the second-level message that explains or gives additional information about the first-level message

(If the Enter key is pressed after the second-level display is shown, the first-level display is again displayed exactly as it was before the Help key was pressed.)

DSPMSGD (Display Message Description) Command

The Display Message Description (DSPMSGD) command displays the message descriptions of one or more messages contained in a message file. The descriptions of specific messages or a range of messages in one message file can be specified by their identifiers, or all messages in one message file can be specified.



MSGID Parameter: Specifies the message identifiers of one or more messages whose descriptions are to be displayed. (To specify a *consecutive* sequence of message identifiers, you can use the RANGE parameter instead.) The message identifier is the value under which the message is stored in the message file.

**RANGE:* All the messages that are in the file and range specified by the FILE and RANGE parameters are to have their message descriptions displayed.

message-identifier: Enter the message identifiers of one or more messages whose descriptions are to be displayed. The message identifiers must each be 7 characters long and in the following format: *pppmmnn*. The first 3 characters must be an alphabetic program code, and the last 4 characters must be a decimal number.

If the messages are specified by their individual identifiers, and one of them is not in the specified file, none of the messages are displayed and an error message is sent to the user.

MSGF Parameter: Specifies the qualified name of the message file from which the message descriptions are taken. (If no library qualifier is given, *LIBL is used to find the file.) The following table lists the alphabetic identifier codes used in the message identifiers and the corresponding IBM-supplied message files:

Identifier Code	Message Type	IBM-Supplied Message File
CBE	COBOL execution time	QCBLMSGE.QSYS
CBL	COBOL compiler	QCBLMSG.QCBL
CBX	COBOL titles and texts	
CSC	COBOL syntax checker	QCSCMSG.QCBL
CPF	Control program facility (CPF)	QCPFMSG.QSYS
CPI	CPF informational messages	
CPX	CPF titles and texts	
KBD	Keyboard	
MCH	System/38 machine instruction interface	
EDT	Editor	QEDTMSG.QSYS
EDX	Editor titles and texts	
FMT	Reformat utility	QFMTMSG.QS3E
FMX	Reformat utility titles and texts	QFMTTXT.QS3E
IDU	Interactive data base utilities (IDU)	QIDUMSG.QIDU
IDX	IDU titles and texts	QIDXMSG.QIDU
QRG	RPG language compiler	QRPGMSG.QRPG
RPT	RPG auto report	
RSC	RPG syntax checker	
RTX	RPG auto report titles and texts	
RXT	RPG relational diagnostic texts	
RPG	RPG execution time	QRPGMSGE.QRPG
SDA	Screen design aid (SDA)	QSDAMSG.QIDU
SDX	Screen design aid titles and texts	

RANGE Parameter: Specifies the range of message identifiers in the specified message file for which message descriptions are to be displayed. This parameter is valid only if MSGID(*RANGE) is specified.

The range is specified by a list of two values. All messages having message identifiers in the specified range are displayed. (Message identifiers must be specified in the same format described in the MSGID parameter.) The first message displayed is the one whose identifier value is equal to or greater than the identifier specified as the starting value; the last message displayed is the one whose identifier value is equal to or less than the identifier specified as the ending value.

Starting Message Identifier: The first of the two values in the RANGE parameter specifies the message identifier for the first message description to be displayed.

*FIRST: The first message in the file is the first one whose description is to be displayed.

first-message-identifier: Enter the 7-character identifier of the first message description to be displayed.

Ending Message Identifier: The second of the two values in the RANGE parameter specifies the message identifier for the last message description to be displayed.

*LAST: The last message in the file is the last one whose description is to be displayed.

last-message-identifier: Enter the 7-character identifier of the last message description to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

```
DSPMSGD  MSGID(CPF1165  CPF1167  CPF3512) +
     MSGF(QCPFMSG.QSYS)
```

If this command is entered from a work station, the descriptions of the three messages listed are displayed. If this command is entered from a batch job, the descriptions of the three messages are sent to the job's output spooling queue.

```
DSPMSGD  MSGID(*RANGE)  MSGF(QIDUMSG.QIDU) +
     RANGE(*FIRST  IDU0571)  OUTPUT(*LIST)
```

This command lists the message descriptions for those message identifiers in the file that are in the following range: the first message in the IDU message file through the message whose identifier is IDU0571.

Additional Considerations

For each message description specified by the MSGID or RANGE parameter, one or more displays are presented that give all the attributes of the predefined message. If more than one display is needed for the same message description, a + sign appears in the lower right corner of each display, except the last. (The Roll Up key must be used to display the remainder of its attributes.) All of the attributes for one message are shown before the first display of the next message description is presented.

Although a series of three displays are shown here, as few as one may be actually presented, depending on the number of attributes specified on the ADDMSGD command used to define the message. Fields for which no attributes were specified in the message description are not displayed.

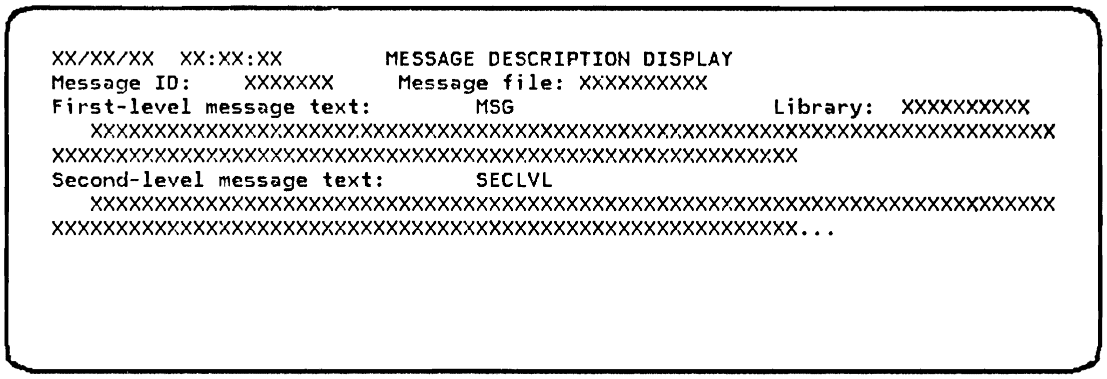
The Roll keys are used to roll forward or backward through the displays for the specified range of message descriptions.

Line 1 of each display gives the current job date and time. Line 2 always displays the attributes that were specified in the MSGID and MSGF parameters of the ADDMSGD command:

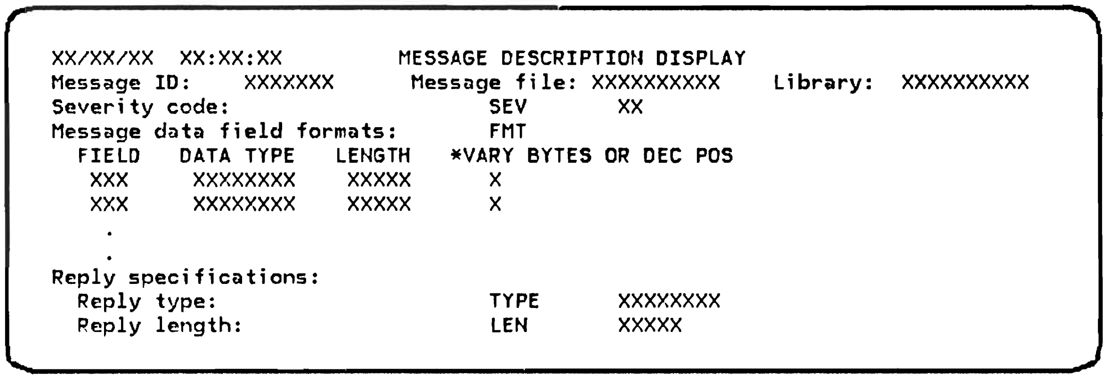
- The message identifier of the message whose description is currently being displayed.
- The qualified name of the message file in which the message descriptions are stored.

The remaining lines on each of the displays show all of the attributes of the predefined message that were specified in the ADDMSGD command. For the explanation of each attribute, refer to the associated parameter description in the ADDMSGD command description. The first-level message text attribute, for example, is described in the MSG parameter. The parameter keywords for the attributes are shown in a column in the center of each display.

The first two message description displays have the following format:



```
XX/XX/XX  XX:XX:XX        MESSAGE DESCRIPTION DISPLAY
Message ID:    XXXXXXX     Message file: XXXXXXXXXX
First-level message text:        MSG                        Library:  XXXXXXXXXX
   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Second-level message text:       SECLVL
   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...
```



```
XX/XX/XX  XX:XX:XX         MESSAGE DESCRIPTION DISPLAY
Message ID:    XXXXXXX      Message file: XXXXXXXXXX    Library:  XXXXXXXXXX
Severity code:                     SEV       XX
Message data field formats:        FMT
  FIELD   DATA TYPE   LENGTH   *VARY BYTES OR DEC POS
   XXX     XXXXXXXX    XXXXX      X
   XXX     XXXXXXXX    XXXXX      X
    .
    .
Reply specifications:
  Reply type:                      TYPE      XXXXXXXX
  Reply length:                    LEN       XXXXX
```

Four fields are displayed for the FMT parameter, but only the last three were specified in the ADDMSGD command.

- Field: The number of the message data field (this displayed attribute is not specified in the FMT parameter). The number indicates the order in which the data field was defined in FMT, with respect to the other data fields also defined.

- Data type: The type of data the substitution field contains and how the data is to be formatted when substituted in the message text.

- Length: The total number of characters or digits that are passed in the message data.

- *VARY bytes/dec pos: Indicates, if *VARY was specified or assumed, the number of bytes in which the length field is passed. Otherwise, it indicates the number of decimal positions in the decimal value.

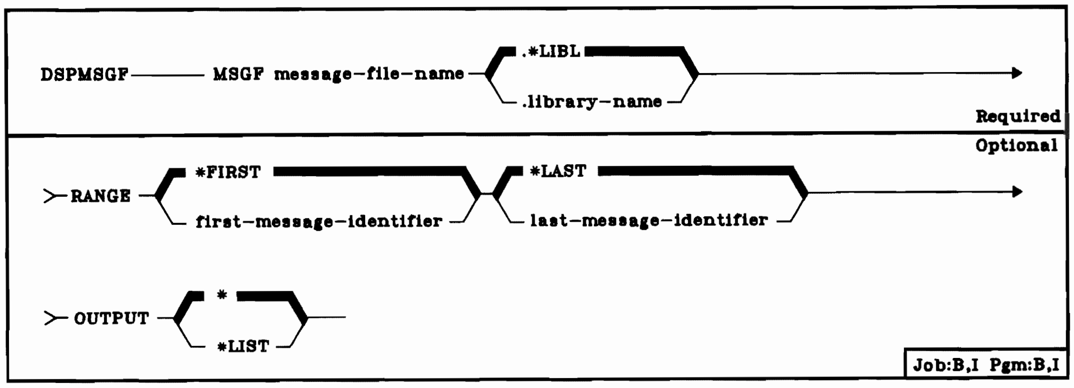
The last display shows the following:

```
XX/XX/XX  XX:XX:XX        MESSAGE DESCRIPTION DISPLAY
Message id:  XXXXXXX        Message file:  XXXXXXXXXX     Library:  XXXXXXXXXX
  Special reply values:             SPCVAL
     ORIGINAL FROM-VALUE                 REPLACEMENT TO-VALUE
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      .
      .
  Reply range:                      RANGE
     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  Relationship for valid replies: REL        XXX        XXXXXXXXXXXXXXXXXXXXXX
    XXXXXXXXXX
  Default reply:                    DFT         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Default program:                    DFTPGM      XXXXXXXXXX
  Library:                                        XXXXXXXXXX
Data to be dumped:                  DMPLST     XXXXXXXX  XXXXXXXX  XXXXXXXX
     XXXXXXXX  XXXXXXXX  XXXXXXXX  XXXXXXXX  XXXXXXXX  XXXXXXXX  XXXXXXXX
Log to service log:                 LOG        XXXX
Level of message when added:        LVL        XX/XX/XX  XX
  Modification level:                          XX/XX/XX  XX
```

DSPMSGF (Display Message File) Command

The Display Message File (DSPMSGF) command allows the user to display or print basic attributes of the specified messages in the specified file. The information displayed for each message is its message identifier, its severity code, and its first-level text.

If the information is displayed, options can be used on the display to display or print the complete message descriptions of messages selected on the display.



MSGF Parameter: Specifies the qualified name of the message file from which the message information is to be taken. (If no library qualifier is given, *LIBL is used to find the file.)

RANGE Parameter: Specifies the range of messages in the file for which their identifiers, severity codes, and first-level text are to be displayed. The range is specified by a list of two values. All messages having message identifiers in the specified range are displayed.

When specific message identifiers are given, they must be specified as 7 characters in the following format: *pppmmnn*. The first 3 characters must be an alphabetic program code, and the last 4 characters must be a decimal number. The first message displayed is the one whose identifier is equal to or greater than the identifier value specified as the starting value; the last message displayed is the one whose identifier is equal to or less than the identifier value specified as the ending value. (The range limits initially specified in the RANGE parameter can be removed later while the messages are being displayed.) If RANGE is not specified, RANGE(*FIRST *LAST) is assumed, and all messages in the file are displayed.

Starting Message Identifier: The first of the two values in the RANGE parameter specifies the message identifier of the first message to be displayed.

*FIRST: The first message to be displayed is the first message in the file.

first-message-identifier: Enter the 7-character identifier of the first message whose attributes are to be displayed.

Ending Message Identifier: The second of the two values in the RANGE parameter specifies the message identifier of the last message to be displayed.

*LAST: The last message to be displayed is the last message in the file.

last-message-identifier: Enter the 7-character identifier of the last message whose attributes are to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

```
DSPMSGF  MSGF(MESSAGES.LIB1)
```

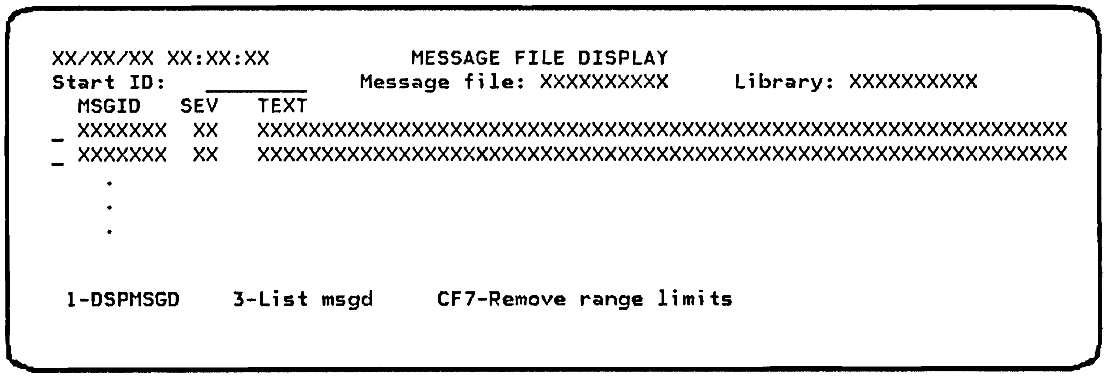
If this command is executed in an interactive job, all of the message identifiers in the file MESSAGES can be displayed. The severity codes and first-level text for those messages are also displayed. If this command is executed in a batch job, the same information is listed with the job's spooled output.

```
DSPMSGF  MSGF(MESSAGES.LIB1) +
     RANGE(IDU0571 *LAST) OUTPUT(*LIST)
```

This command lists with the spooled output the message identifiers, severity codes, and first-level text for those messages in the file that are in the following range: identifier IDU0571 through the last message identifier in the IDU message file.

Additional Considerations

When the DSPMSGF command is executed as an interactive job and OUTPUT(*) is specified, the *message file display* appears:



```
XX/XX/XX XX:XX:XX          MESSAGE FILE DISPLAY
Start ID:  ________     Message file: XXXXXXXXXX     Library: XXXXXXXXXX
  MSGID   SEV   TEXT
_ XXXXXXX  XX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
_ XXXXXXX  XX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
     .
     .
     .

 1-DSPMSGD    3-List msgd     CF7-Remove range limits
```

The message file display shows all of the messages in the specified message file that are in the range specified by the RANGE parameter values. Line 2 contains the name of the message file whose message descriptions are displayed, the name of the library containing the file, and the starting message identifier input field. To specify the message to be displayed *next*, the user can key the message identifier in the input field and press the Enter key. (If an identifier is specified on line 2, options 1 and 3 on line 11 cannot be specified for any message on the same display.)

Line 3 contains the headings of the three attributes that are displayed for each message. Then, for each message, one line is used to display an input field and the identifier, severity code, and first-level text of the message. The first-level text is truncated; the user can display the complete text by selecting option 1.

The Roll keys can be used to move forward and backward in the file being displayed. The starting identifier field on line 2 can be used to specify the complete message identifier of the first message to be shown on the next display when the Enter key is pressed. If there is no message in the file whose identifier is specified on line 2, the first message displayed is the first one whose message identifier is greater than the identifier specified.

If the CF7 key has not been pressed, you will not be able to roll beyond the range initially specified by the RANGE parameter on the DSPMSGF command. If you want to display other messages that are outside the range specified on the command, you can press the CF7 key to remove the range limits. The display on which you pressed the CF7 key is then redisplayed, and you can specify the 7-character message identifier of the message you want to display next.

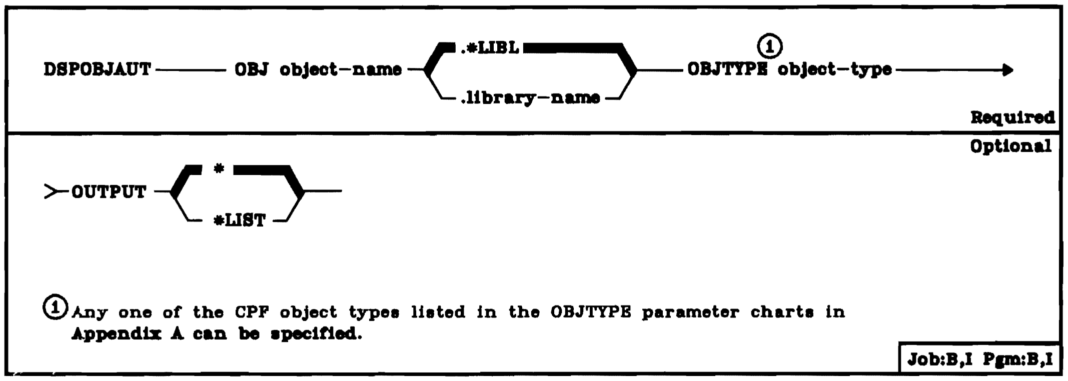
Line 11 shows, in addition to the function of the CF7 key, two other options for obtaining more information about a message. Option 1 displays the detailed message description for the specified message. Option 3 causes the detailed description for the message to be printed with the job's spooled output. To select one of these options for a specific message, enter the option number in the input field to the left of the message identifier for that message.

Options can be entered in multiple input fields on the same display, but if options are specified, a 7-character message identifier cannot be specified on line 2 of the same display. Once all of the specified options have been performed, another starting identifier can be specified on line 2 if desired.

DSPOBJAUT (Display Object Authority) Command

The Display Object Authority (DSPOBJAUT) command displays the list of authorized users of an object and their associated rights of use. This command can be entered by the security officer or any other user who is authorized to use the command.

The following are displayed for the specified object: the object name, the name of the library containing the object, the name of the object's owner, the object's type, and a list of all the users who are authorized to use the object and the rights that each user has for the object. However, if the user entering the command does not have object management rights for the object, only his name and rights are displayed—the names of the other users and their rights for the object are not displayed.



①Any one of the CPF object types listed in the OBJTYPE parameter charts in Appendix A can be specified.

OBJ Parameter: Specifies the qualified name of the object for which the authorized users and their rights are to be displayed. (If no library qualifier is given, or if the object is OBJTYPE(*LIB), *LIBL is used to find the object.)

OBJTYPE Parameter: Specifies the object type of the object that is to have its authorized users and rights displayed. Any one of the CPF object types can be specified; refer to the charts in the expanded description of the OBJTYPE parameter in Appendix A. To display a list of users authorized to use a file, for example, enter the value *FILE.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

```
DSPOBJAUT  OBJ(PROG1.ARLIB)  OBJTYPE(*PGM)
```

This command causes the list of authorized users and their rights of use for the object named PROG1 to be listed for the user who entered the command, but only if he has object management rights for the object. (If he does not, only his rights are displayed.) PROG1 is a program (PGM) located in the library named ARLIB. The system assumes * for the device that is to display the output list. If the command was entered in the batch subsystem, the output is placed in the default output queue for the job. If the command was entered in the interactive subsystem, the output is displayed on the device at which the user entered the command.

```
DSPOBJAUT  OBJ(PROG2.ARLIB)  OBJTYPE(*PGM)  +
     OUTPUT(*LIST)
```

This command causes the list of authorized users of the program named PROG2 in the ARLIB library to be printed. If the user who enters the command does not have object management rights for the program, only that user's name and rights are printed, not those of other authorized users.

Additional Considerations

The information display for the object specified on the DSPOBJAUT command gives details about the object itself and then lists all the authorized users and their rights of use for the object.

```
XX/XX/XX              OBJECT AUTHORITY DISPLAY
Object name: XXXXXXXXXX         Library:     XXXXXXXXXX
Owner name:  XXXXXXXXXX         Object type: XXXXXXXX
                      |   OBJECT RIGHTS  |    DATA RIGHTS
     USER NAME        | OPER MGT  EXIST  | READ ADD  UPD  DLT
     *PUBLIC          |  X    X     X    |  X    X    X    X
     XXXXXXXXXX       |  X    X     X    |  X    X    X    X
     XXXXXXXXXX       |  X    X     X    |  X    X    X    X
     XXXXXXXXXX       |  X    X     X    |  X    X    X    X
         .            |                  |
         .            |                  |
         .            |                  |
```

The name of the object, the object owner, the library containing the object, and the object type are shown. The information about individual user profiles is given under three groups of column headings:

- User Profile Name: Specifies the name of each user profile that has been granted explicit rights to the object by the GRTOBJAUT command.

  The value *PUBLIC is displayed when the public can use the object in some way. This occurs if:
  - The object was created with *NORMAL or *ALL specified in the PUBAUT parameter of its create command.
  - The public was granted one or more rights to the object by the GRTOBJAUT command.

  The value *ADOPTED is displayed if the adopted user profile you are currently using has more authority than the one you used to sign on.

- Object Rights: Indicates, for each user profile, which of the *object control rights* the users operating under that user profile have for the object.
  - Operational rights
  - Object management rights
  - Object existence rights

- Data Rights: Indicates, for each user profile, which of the *data rights* the users operating under that user profile have for the object. The data rights are read (READ), add (ADD), update (UPD), and delete (DLT).

For each right that the specified user profile has, an X appears in the appropriate column.

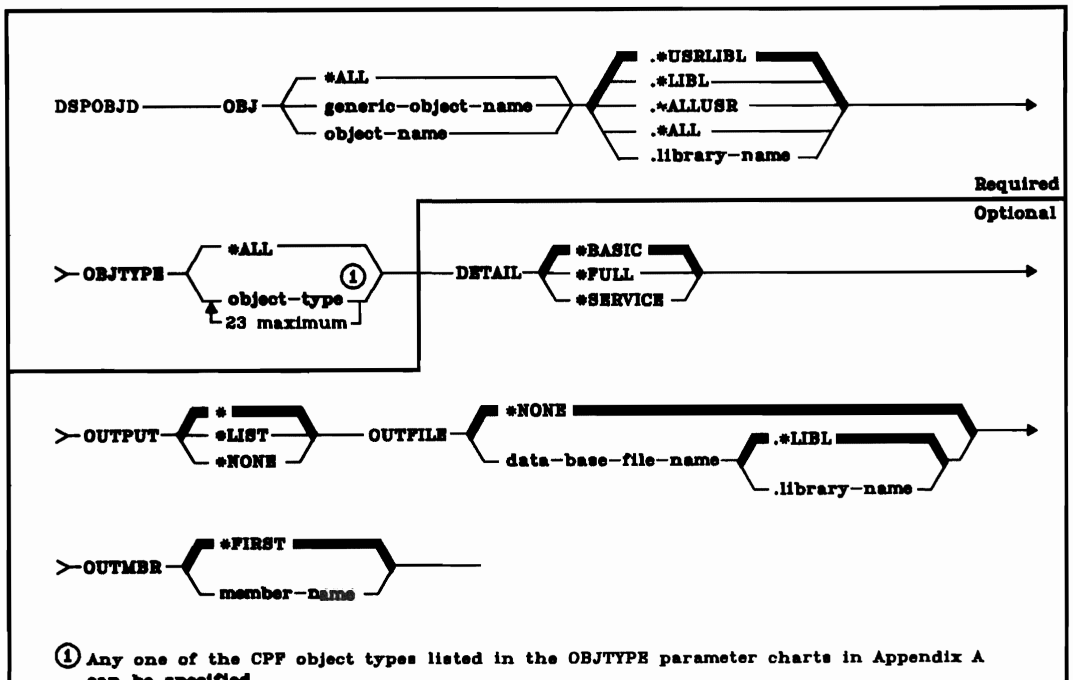
DSPOBJD (Display Object Description) Command

The Display Object Description (DSPOBJD) command displays the names and attributes of specified objects in the specified library or in the libraries of the job's library list. The command can also display the names and attributes of libraries themselves, if specified.

Only the object attributes of each object are displayed; the data attributes of data in the object, and the actual data in the object, are *not* displayed. Also, if an object being displayed has been damaged (possibly by a system failure), the fact that it is damaged is indicated on the display.

Any CPF object for which the user has some authority can be displayed by this command. Objects or libraries for which the user has no operational authority are not displayed, even if specified in the command. If only one object is to be displayed, the user can specify it by entering the object name, object type, and the name of the library that it is in.

Restriction: The user who submits this command must either own the objects or must have read rights for the specified libraries and some rights (meaning any one of the object rights) for each of the objects. If he does not have read rights for a library, none of its objects are displayed.



① Any one of the CPF object types listed in the OBJTYPE parameter charts in Appendix A can be specified.

Job:B,I Pgm:B,I

OBJ Parameter: Specifies which objects in the library or libraries are to have their object attributes displayed. If no library qualifier is specified, *USRLIBL is assumed and all libraries in the user portion of the job's library list are to be searched for the objects. Objects in the library for which the user does not have some authority are not displayed.

Depending on the library qualifier specified or assumed, the following libraries (for which the user has the authority) are to be searched for the specified objects:

- .*USRLIBL (user library list). Only the libraries listed in the user portion of the job's library list.
- *.*LIBL* (library list). All the libraries in the user *and* system portions of the job's library list.
- *.*ALLUSR* (all user libraries). All the nonsystem libraries, which include *all* user-defined libraries and the QGPL library, not just those in the job's library list.
- *.*ALL* (all libraries). All the libraries in the system, including QSYS.
- *.library-name* (one library). Only the library named in this parameter. The user must have operational rights for the specified library.

If the OBJTYPE parameter specifies the value *ALL or any of the values *LIB, *USRPRF, *CUD, *DEVD, or *LIND (these objects appear as though they are stored in QSYS), the library qualifier for the OBJ parameter must be *LIBL, *ALL, or QSYS.

**ALL:* All objects in the libraries identified in the library qualifier that are of the types specified by the OBJTYPE parameter are to be displayed. The user can display only those objects for which he has some authority.

qualified-generic-object-name: Enter the qualified generic object name of the objects that are to be displayed. All objects for which the user has some authority (for example, operational rights) that are in the specified libraries are displayed. To specify a generic name, add an asterisk after the last character in the generic name (ABC*, for example). If an * is not included with the name, the system assumes that the name is a complete object name.

qualified-object-name: Enter the qualified name of the object to be displayed. If the library qualifier is *ALL, *ALLUSR, or a library name, all objects of the specified types for which the user has some authority (for example, operational rights) and that are in the specified libraries are displayed. If the qualifier is *USRLIBL or *LIBL, then only one object type (not *ALL) can be specified, and only the first object found is displayed.

OBJTYPE Parameter: Specifies which types of objects are to be displayed. (For a list of the CPF object types, see the OBJTYPE parameter in Appendix A.)

**ALL:* All object types are displayed that have the specified object name.

object-type: Enter one or more values (up to a maximum of 23) for the types of CPF objects that can be displayed. Any of the CPF object types can be specified; refer to the charts in the expanded description of the OBJTYPE parameter in Appendix A. All objects in the specified libraries, as well as libraries themselves, that have the object type(s) specified will have their object attributes displayed. If the library qualifier specified or assumed for the OBJ parameter is *USRLIBL or *LIBL, then only one object type (not *ALL) can be specified here.

DETAIL Parameter: Specifies which set of attributes is to be displayed for each object. Refer to *Additional Considerations* at the end of this command description for the list of attributes displayed in each set.

**BASIC:* The display or listing is to contain the name and a basic set of object attributes for each object.

**FULL:* The display or listing is to contain the name and a full set of object attributes for each object.

**SERVICE:* The display or listing is to contain the service-related attributes for each object (which includes the basic set of attributes).

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

**:* The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

**NONE:* The output is to be directed to the data base file specified in the OUTFILE parameter.

OUTFILE Parameter: Specifies the name of the data base file to which the output of the display is directed. If the OUTFILE does not exist, this command creates a data base file in the qualified library.

**NONE:* The output is not to be directed to a data base file.

qualified-data-base-file-name: Specifies the name of the data base file that is to receive the output of the display.

OUTMBR Parameter: Specifies the name of the data base file member to which the output of the display is directed. If a member already exists, the system will clear it and add the new records. If the member does not exist and a member name is not specified, the system will create a member with the name of the file specified in the OUTFILE parameter. If a member name is specified, but the member does not exist, the system will create it.

*FIRST: The first member in the file is to receive the output.

member-name: The named file member is to receive the output.

Examples

```
DSPOBJD  OBJ(PAY.X)  OBJTYPE(*ALL)
```

A basic description of all the objects (for which the user has some authority) that are named PAY and are located in library X are displayed. Objects in the library for which the user has no authority of any kind are not displayed.

```
DSPOBJD  OBJ(PAY.X)  OBJTYPE(*PGM)  DETAIL(*FULL)
```

A full description of the program named PAY in library X is displayed. The display includes all the attributes of the program.

```
DSPOBJD  OBJ(PAY.*USRLIBL)  OBJTYPE(*PGM)
```

This command displays information about the *first* program named PAY that is found in the user portion of the job's library list.

```
DSPOBJD  OBJ(ABC*.Z)  OBJTYPE(*FILE)
```

A basic description of all of the files whose names begin with ABC (generic name) located in library Z for which the user has some authority are displayed.

```
DSPOBJD  OBJ(*ALL.*USRLIBL)  OBJTYPE(*PGM)
```

This command displays the attributes of all programs that are in the libraries named in the user's portion of the library list and for which the user has some authority.

```
DSPOBJD  OBJ(*ALL.QSYS)  OBJTYPE(*LIB)
```

This command displays the basic attributes of all the libraries in the system for which the user has some authority.

Additional Considerations

The DSPOBJD command can produce three types of displays. Depending on whether you specify *BASIC, *FULL, or *SERVICE on the DETAIL parameter, you can display the basic attributes, the full set of attributes, or the basic and service-related attributes for one or more objects. (No data attributes of the data in an object are shown.) If DETAIL(*BASIC) is specified, the following display is presented:

```
XX/XX/XX XX:XX:XX    OBJECT DESCRIPTION - *BASIC           +++
Library: XXXXXXXXXX
  OBJECT     TYPE     ATTR       FREED   SIZE      TEXT
  XXXXXXXXXX XXXXXXXX XXXXXXXXXX XXX XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXX XXXXXXXX XXXXXXXXXX XXX XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      .
      .
      .
```

This display shows the basic set of object attributes for several objects at the same time. For each specified library that contains one or more objects meeting the requirements of the OBJ and OBJTYPE parameters, one or more displays are produced. The second line of each display gives the name of the library containing the currently displayed objects. Then, for each object, a separate line is used to display the following:

- The object's name and type.
- Any extended attributes, such as a library being a production or test library. This field lists types of high-level languages or status information of objects not currently available. If the object has been damaged, **DAMAGED* is displayed in this field. Valid values grouped by object type are:

Object Type	**Values**		
File	BSC	DKT	PHY
	CMN	DSP	PRT
	CRD	LGL	TAP
Library	PROD	TEST	
Program	CBL	DFU NOTEXC	
	CL	DFU EXC	
	DFU	QRY NOTEXC	
	QRY	QRY EXC	
	RPG		

- Whether the object has been suspended because the storage for its data has been freed.

- The amount of storage (in bytes) that the object occupies.

- A truncated version (the first 15 or 31 characters, depending on the display device) of the text description that is stored in the object's description. (If the basic attributes are being printed, the entire text description is printed.)

If DETAIL(*FULL) is specified, the following display is presented for each object and type specified in the command:

```
XX/XX/XX XX:XX:XX   OBJECT DESCRIPTION - *FULL           +++
Object: XXXXXXXXXX   Library:   XXXXXXXXXX  Type:  XXXXXXXX
Attr:   XXXXXXXXXX   Freed:     XXX         Size:  XXXXXXXXXX
Text: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Creation date/time: XX/XX/XX XX:XX:XX
Owner:              XXXXXXXXXX
Save date/time:     XX/XX/XX XX:XX:XX  Save command: XXXXXX
Restore date/time:  XX/XX/XX XX:XX:XX  Offline size: XXXXXXXXXX
Starting slot:      XX
XXXXXXXX volumes:   XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX
                    XXXXXX XXXXXX XXXXXX XXXXXX ...
```

This display shows, for one object at a time, the complete set of display attributes of the object; the attributes shown on the basic display are included (on lines 2, 3, and 4), but not in the same order. The text description is displayed in its entirety.

If multiple objects were specified in the OBJ and OBJTYPE parameters of the DSPOBJD command, they are presented one display at a time, in alphabetic order by object name within each object type.

In addition to the basic attributes already described, the following are displayed for each object, beginning on line 5:

- The date and time when the object was created. Note that this time-value will not agree with the time-value shown at the top of a create listing; that value indicates the time of the job.

- The name of the object's owner; that is, the name of the owner's user profile.

- If the object has been saved, the date and time of the most recent save, and the name of the command used to save it.

- If the object has been restored, the date and time of the most recent restore.

- If the object has been saved, the amount of storage (in bytes) that the object occupied at the time of the save.

- If the save was to a magazine, the starting diskette position in the magazine.

- If the object has been saved, the type of offline storage used (magazine, diskette, or tape), and the volume identifiers of the volumes it was saved on. A maximum of 10 save/restore volumes can be identified on this display; if more than 10 were used to save the object, three periods are displayed to the right of the tenth volume identifier.

If the object being displayed has been damaged, a message appears below all the attributes, stating that the object is damaged.

When the full description of objects is *printed*, a different format is used: most of the basic attributes are printed like a header line on the left side, and the other attributes are shown on the right side in generally the same organization as when they are displayed.

If DETAIL(*SERVICE) is specified, the following display is presented for each object specified in the command:

```
 XX/XX/XX  XX:XX:XX      OBJECT DESCRIPTION - *SERVICE       +++
Object:  XXXXXXXXXX       Library:  XXXXXXXXXX  Type:  XXXXXXXX
Attr:    XXXXXXXXXX       Freed:    XXX         Size:  XXXXXXXXXX
Text: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Creation date/time:       XX/XX/XX XX:XX:XX     CPF lvl:  RXXMXX
Src file:                 XXXXXXXXXX.XXXXXXXXXX Member:   XXXXXXXXXX
Src file date/time:       XX/XX/XX XX:XX:XX
Compiler:                 XXXXXXX     RXXMXX
Object control lvl:       XXXXXXXX              User mod: XXX
Program product:          XXXXXXX     RXXMXX
PC number:                XXXXX                 APAR ID:  XXXXXX
```

The information on this display is used by programming support personnel to determine the level of the system on which an object was created and whether or not the object has been modified since it was installed on the system. The information helps isolate problems associated with objects and aids in applying the appropriate fixes to objects that need to be repaired.

This display shows, for one object at a time, the service-related attributes of the object. In addition to the basic attributes already described, the following attributes are displayed for each object (beginning on line 5):

- The release and modification level of CPF at the time the object being displayed was created. This field is defined for all objects on the system. The format of this information is RXXMXX, where R = release, M = modification level, and the Xs indicate the release and modification numbers.

- If the object is a command, program, file, print image, or table, the following information is displayed:
  - Qualified name of the source file that was used to create the object, and the name of the member in the source file.
  - Date and time of the last change to the source file *before* it was used to create the object.
  - Name of the program product compiler used to create the object, and the release and modification level of the compiler.

  These fields are blank for all other object types.

- For IBM-supplied objects, the version level of the object is displayed. (This field is blank for user-created objects.)

- For IBM-supplied objects, a value of YES or NO to indicate whether the object has been modified by a user. The field contains NO for all IBM-supplied objects when the system is shipped. For all user-created objects, the field is always initialized with YES at the time each object is created.

- For objects that are part of an IBM program product, the name, release level, and modification level of the program product to which the object belongs is displayed. (This field is blank for user-created objects.) The format of the release and modification level information is the same as that given earlier in the description of this display.

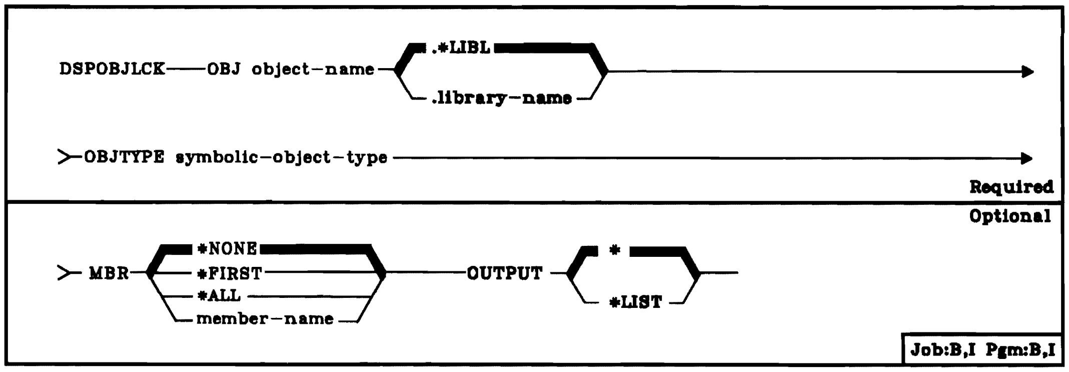
- For objects that are part of a programming change (PC), the number of the PC that resulted in the object being created is displayed. (This field is blank for user-created objects.)

- For program objects that have been patched as a result of an APAR, the APAR number is displayed. (This field is blank for objects that have not been patched.)

DSPOBJLCK (Display Object Lock) Command

The Display Object Lock (DSPOBJLCK) command displays all the object lock requests in the system for a specified object. Both held locks and locks being waited for are displayed.

Note: This command does not show record locks for data base files.



OBJ Parameter: Specifies the qualified name of the CPF object for which locks are being displayed. (If no library name is given, *LIBL is used to find the object.) If a file member is specified for a file, and the file's library value is *LIBL, the first occurrence of the file in the job's library search list will be searched for the member. For object types that exist only in library QSYS (for example, *DEVD), QSYS and *LIBL are the only library names that will be accepted.

OBJTYPE Parameter: Specifies the object type of the CPF object for which locks are being displayed. Enter the predefined value that identifies the object type.

MBR Parameter: Specifies the member name of a CPF data base file. This parameter is valid only when a data base file has been specified on the OBJ parameter.

*NONE: No member locks are displayed, but file level locks are displayed. (The display of member locks for all the members in the file may be requested from the file locks display.)

**FIRST:* Specifies that the first member in the named file is to be displayed.

**ALL:* Specifies that member locks for all the members in the file are to be displayed.

member name: Enter the name of the data base file member for which locks are to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the file(s) used by this command.)

*: The output is to be displayed (if requested by an interactive job) or listed with the job's printer output (if requested by a batch job).

**LIST:* The output is to be listed with the job's printer output. For data base files, all levels of lock information that can be selected on the display are printed.

Example

```
DSPOBJLCK  OBJ(LOCKEDFILE.QGPL)  OBJTYPE(*FILE)  +
     MBR(LOCKEDMBR)  OUTPUT(*LIST)
```

This command causes the lock information for member LOCKEDMBR of file LOCKEDFILE to be written to the job's spooled printer file.

Additional Considerations

A single screen is used to display the locks for all objects except data base files. Because of the complex structure of data base files, several lock display screens are needed. Different screens are used for file locks, file member locks, and shared member locks.

The object locks display has the following format:

```
 XX/XX/XX XX:XX:XX        OBJECT LOCKS
Object:   XXXXXXXXXX  Library:   XXXXXXXXXX  Type:   XXXXXXXXX
   JOB         USER        NBR       LOCK      STS
_ XXXXXXXXXX  XXXXXXXXXX  XXXXXX    XXXXXXX   XXXX  XXXX
_                                   XXXXXXX   XXXX  XXXX
_ XXXXXXXXXX  XXXXXXXXXX  XXXXXX    XXXXXXX   XXXX  XXXX
_ XXXXXXXXXX  XXXXXXXXXX  XXXXXX    XXXXXXX   XXXX  XXXX
        .
        .
        .
 1-DSPJOB  2-Job locks  9-CNLJOB    CF3-Member locks  CF5-Redisplay
```

The first line of the display shows the current job date and time. The second line shows the object name, the library in which it resides, and the object type. For file objects, the TYPE field displays the specific type of file, as follows:

- *FILE-PHY. Physical file.
- *FILE-LGL. Logical file.
- *FILE-xxx. Device file, where xxx is the abbreviation of the specific type of device file.

Beginning with the fourth line of the display, the lock entry information is shown.

All of the DSPOBJLCK screens show lock entries in the same basic format. Each entry contains the qualified job name of the job associated with the lock request, the lock state for the request, and the status of the request. The entries are displayed in alphabetic order by job name. If the same job has more than one lock entry for the object, the job name appears for only the first entry.

The lock state is shown in the LOCK field as one of the following values:

- *SHRRD. Lock shared for read.
- *SHRUPD. Lock shared for update.
- *SHRNUP. Lock shared no update.
- *EXCLRD. Lock exclusive allow read.
- *EXCL. Lock exclusive no read.
- *NONE. Lock entry has a null value and is used to select display of lower level locks.

The status of the lock is shown as one of the following values:

- HELD. The lock is currently held by the job.
- WAIT. The job is waiting for the lock.
- REQ. The job has a lock request outstanding for the object.

The lock may be a single request or part of a multiple lock request for which some other object specified in the request has been identified as unavailable.

An input field exists at the leftmost column of the display. You can use this field to enter one of the option numbers shown along the bottom of the display. The DSPJOB option executes the Display Job command for the job on that line and displays the display job menu (refer to the DSPJOB command for more information). The Job locks option displays the locks held for the job (the display is explained in the *Additional Considerations* section of the DSPJOB command, under the description of the DSPJOB menu option 7). The CNLJOB option executes the Cancel Job command for that job.

The CF5 key can be used to redisplay the object locks display with the most current information. The CF3 key can be used to display member locks for data base files. The member locks display has the following format:

```
 XX/XX/XX XX:XX:XX        MEMBER LOCKS
File:   XXXXXXXXXX   Library:   XXXXXXXXXX   Type:   XXX
   MEMBER     JOB        USER       NBR      LCKTYP LOCK    STS  SHR
_ XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX   XXXXXX XXXXXXX XXXX XXXX XXXX
_                                           XXXXXX XXXXXXX XXXX XXXX XXXX
_                                           XXXXXX XXXXXXX XXXX XXXX XXXX
_            XXXXXXXXXX XXXXXXXXXX XXXXXX   XXXXXX XXXXXXX XXXX XXXX XXXX
_                                           XXXXXX XXXXXXX XXXX XXXX XXXX
_            XXXXXXXXXX XXXXXXXXXX XXXXXX   XXXXXX XXXXXXX XXXX XXXX XXXX
_            XXXXXXXXXX XXXXXXXXXX XXXXXX   XXXXXX XXXXXXX XXXX XXXX XXXX
_ XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX   XXXXXX XXXXXXX XXXX XXXX XXXX
_            XXXXXXXXXX XXXXXXXXXX XXXXXX   XXXXXX XXXXXXX XXXX XXXX XXXX
_            XXXXXXXXXX XXXXXXXXXX XXXXXX   XXXXXX XXXXXXX XXXX XXXX XXXX
                                    .
                                    .
                                    .

1-DSPJOB  2-Job locks  3-Shared member locks  9-CNLJOB    CF5-Redisplay
```

The first two lines of the member locks display contain the date and time of the job, the qualified name of the file for which member locks are being displayed, and the type of the data base file (PHY or LGL).

Beginning with the fifth line, the member lock entries are shown. The entries contain:

- A one character input field
- The name of the member for which the lock was requested
- The qualified name of the job that issued the lock request
- The lock type
- The lock state
- The lock status
- A field that indicates whether shared member locks are associated with the member (this field is not displayed for physical file members)

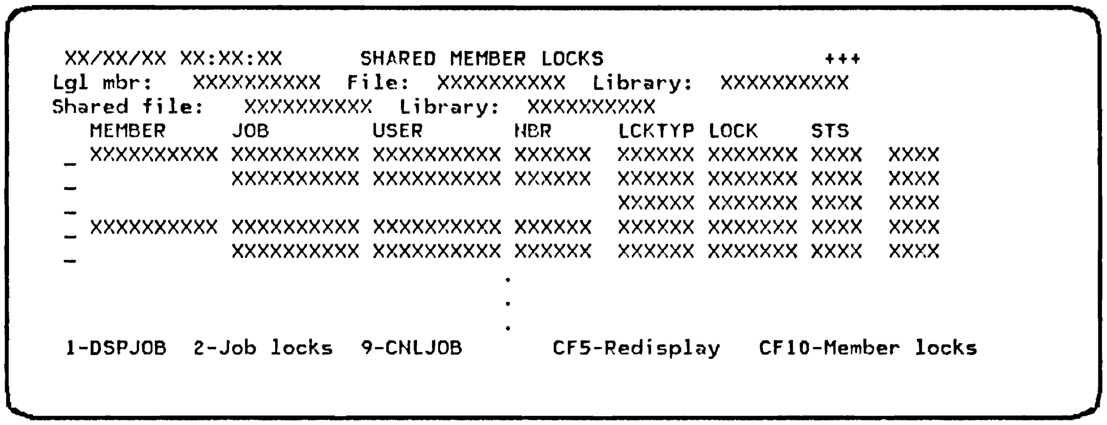
The lock entries are shown alphabetically by file member name and by job name within the member name. The member name is displayed only once for locks by multiple jobs. The job name is shown only once if it has multiple locks on the member.

The lock type (LCKTYP) field may take on any of the following values:

- MBR, for member control block locks
- DATA, for locks on the actual data within a member
- ACCPTH, for locks on the access path used to access a member's data

The shared member locks field (SHR) is displayed for logical member locks only. YES displayed for the SHR field indicates that based-on physical file members or members sharing the access path associated with this logical file are locked. A member lock entry may show no member level locks for the logical file member being displayed, but shared member locks may still exist. In this case, the member name, LOCK(*NONE), and SHR(*YES) are the only fields displayed. All other fields will be blank.

You can enter the Shared member locks option (option 3) to display the shared member locks display:



```
 XX/XX/XX XX:XX:XX      SHARED MEMBER LOCKS              +++
Lgl mbr:   XXXXXXXXXX  File:  XXXXXXXXXX  Library:  XXXXXXXXXX
Shared file:   XXXXXXXXXX  Library:  XXXXXXXXXX
   MEMBER      JOB        USER       NBR      LCKTYP LOCK    STS
 _ XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX  XXXXXX XXXXXXX XXXX  XXXX
 _            XXXXXXXXXX XXXXXXXXXX XXXXXX  XXXXXX XXXXXXX XXXX  XXXX
 _                                          XXXXXX XXXXXXX XXXX  XXXX
 _ XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX  XXXXXX XXXXXXX XXXX  XXXX
 _            XXXXXXXXXX XXXXXXXXXX XXXXXX  XXXXXX XXXXXXX XXXX  XXXX
                                  .
                                  .
                                  .
 1-DSPJOB  2-Job locks  9-CNLJOB       CF5-Redisplay   CF10-Member locks
```

The first line of the display shows the current job date/time, the display title, and, if there are more displays, an additional information indicator (+++). The second line shows the qualified logical file member name that is sharing the displayed file. The third line shows the qualified name of the file whose members are locked.

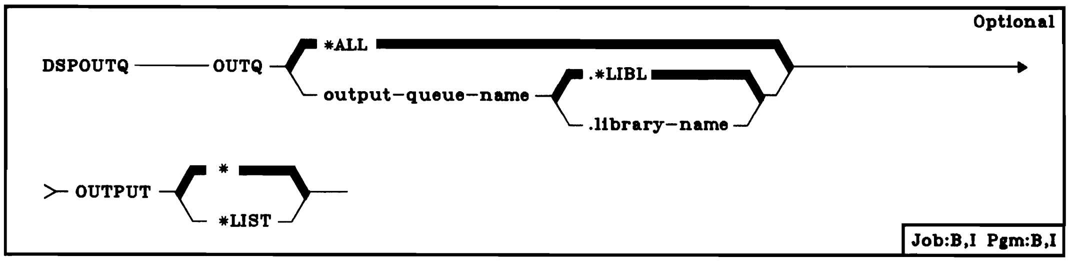
Beginning with the fifth line, the shared member lock entries are shown. The entries are displayed with most of the same fields shown in the member locks display (SHR is not shown).

The logical file being displayed may be based on several physical files, and therefore, several shared member lock displays may be shown (one for each shared file). To scroll through the sequence of displays, press Enter. To return to the previous display, use the CF2 key. To return to the member locks display, use the CF10 key and to exit completely, use the CF1 key.

DSPOUTQ (Display Output Queue) Command

The Display Output Queue (DSPOUTQ) command displays the overall status of all output queues or the detailed status of a specific output queue. The status of the queues may change while the command is being executed.

Restrictions: If only one output queue is to be displayed, the user must have read rights for the queue to be displayed or he must have job control rights in his user profile and the output queue must have the OPRCTL(*YES) attribute. If all the output queues are to be displayed, the user needs authority only for the DSPOUTQ command.



OUTQ Parameter: Specifies that all output queues are to be displayed, or specifies the name of the output queue that is to have its status displayed.

*ALL: The overall status of all output queues is to be displayed. The following information about each output queue is given: qualified output queue name, the queue's status, the name of the writer to which the queue is assigned, and the number of entries on the queue.

qualified-output-queue-name: Enter the qualified name of the output queue about which detailed status information is to be displayed. (If no library qualifier is given, *LIBL is used to find the queue.) The following information about each spooled file entry on the specified queue is given: the job name and number, the user name, the file name and number, the job priority, the number of records in the file, and the file status.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

DSPOUTQ OUTQ(QPRINT.QGPL)

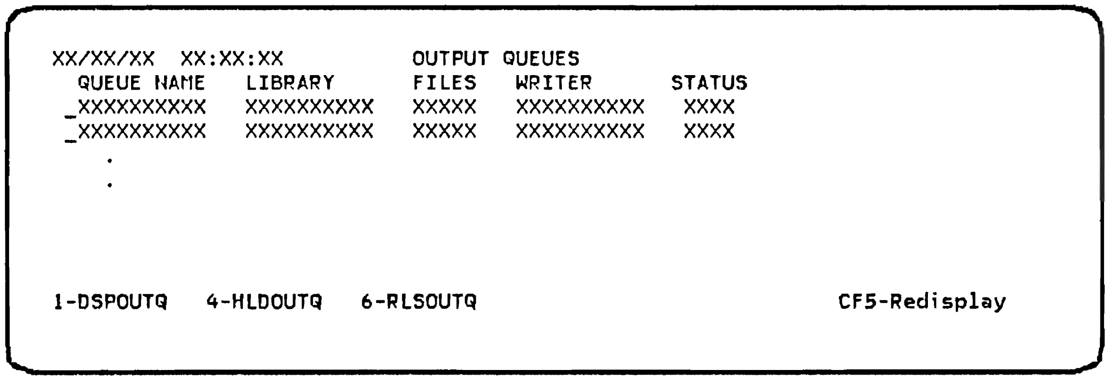
This command displays the detailed status information about the output queue named QPRINT qualified with the name QGPL. Each spooled file entry on the QPRINT output queue is displayed, showing the following information about the spooled file to be printed: the file name and number, the number of file records, and the status of both the file and the job in which the file exists (other job information is also displayed.)

Additional Considerations

One of two display formats is produced by the DSPOUTQ command to display the output queues and their current status. The display presented depends on whether you request the detailed status of a specific output queue or the overall status of all output queues.

Displaying All Output Queues

If OUTQ(*ALL) is specified or assumed on the DSPOUTQ command, the *output queues display* is presented:



```
XX/XX/XX  XX:XX:XX         OUTPUT QUEUES
  QUEUE NAME   LIBRARY      FILES   WRITER       STATUS
 _XXXXXXXXXX   XXXXXXXXXX   XXXXX   XXXXXXXXXX    XXXX
 _XXXXXXXXXX   XXXXXXXXXX   XXXXX   XXXXXXXXXX    XXXX
    .
    .



1-DSPOUTQ   4-HLDOUTQ   6-RLSOUTQ                           CF5-Redisplay
```

The overall status of each of the output queues defined in the system is displayed. For each output queue, a separate line is used to show:

- An input field (to the left of the output queue name) in which a number can be entered. The number can be any one of those shown at the bottom of the display and can be entered in the input field to cause the function (a command) associated with that number to be performed for that output queue when the Enter key is pressed. If numbers are placed in the input fields of several output queues before the Enter key is pressed, the specified functions are performed (one at a time) on the queues in the order in which the queues are shown on the display. The system executes each command using the default values of all its parameters. The following functions can be specified:

  1–DSPOUTQ: Display the names and attributes of all the spooled files on the output queue.
  4–HLDOUTQ: Hold the output queue.
  6–RLSOUTQ: Release the output queue.

  – When all of the commands have been executed, the output queues display is reshown with the status fields of the output queues updated; also shown at the bottom of the display are any error or completion messages that occurred when the commands were executed.
  – If there are more output queues than can be shown on a single display, the Roll Up key can be used to display the rest of them. Numbers can be placed in the input fields on multiple displays before the Enter key is pressed.
  – After the commands have been executed, if there are more messages than can fit on that display, a + is shown at the end of the last (or only) message displayed. To view additional messages, position the cursor anywhere on a message line and press the Roll Up key. If no input values are specified in the input fields and the Enter key is pressed, it causes an exit back to the display from which the current display was requested.
  – The CF1 key can be used to exit from the display shown above, or to exit from a display presented as a result of executing the commands entered on the display shown above. (In both cases, the exit causes a return to the basic working display or menu from which the last primary request was entered.) If, for example, you are viewing a display that resulted from a 1 being entered to execute the DSPOUTQ command, the CF1 key causes all of the functions indicated for the output queues that followed the queue currently being displayed to *not* be executed. That is, all queues further down in the initial output queue list do not have any of their requested functions performed.
  – The CF2 key can be used to back up to a previous display, or if you are viewing the first in the sequence of displays, to return to the caller.
  – The CF5 key can be used to ignore (cancel) any functions not yet executed that were entered in the input fields, and to reshow the output queues display with the updated status of all the queues.

- The name of the output queue and the name of the library in which it is stored.
- The number of file entries on the queue of spooled output files that are waiting to be output to a device.
- The name of the card, diskette, or print writer that is currently associated with the output queue (that is, the writer that produces output from the queue on a device). If no name is shown, the output queue is not associated with a writer and output cannot be produced at the present time.
- The current status of the output queue. If the queue has been held by a HLDOUTQ command, HLD is shown in the status column. If output can be produced from the queue, this field is blank.

Note: You can use the CF5 key to get an update on the status of all the output queues (if viewing the overall status of the queues) or of the spooled files on a single output queue (if viewing the status of a specified queue) without having to reenter the DSPOUTQ command.

Displaying a Single Output Queue

If a single output queue name is specified on the OUTQ parameter of the DSPOUTQ command, the *output queue display* is presented:

```
XX/XX/XX XX:XX:XX  OUTQ - XXXXXXXXXX LIB - XXXXXXXXXX  XXXXXX
  FILE NAME  NBR  JOB NAME   USER       NBR    PTY RCD/PAG STS
_ XXXXXXXXXX XXXX XXXXXXXXXX XXXXXXXXXX XXXXXX X  XXXXXXX XXX
_ XXXXXXXXXX XXXX XXXXXXXXXX XXXXXXXXXX XXXXXX X  XXXXXXX XXX
     .
     .




1-DSPSPLF   2-DSPSPLFA   4-HLDSPLF   6-RLSSPLF   9-CNLSPLF   CF5-Redisplay
```

All the spooled output files that are currently on the output queue identified on line 1 are shown on this display. Also shown on line 1 are the current job date and time, the name of the library in which the output queue is stored, and an output queue status field. If this field indicates WTR, a writer has been started to this queue, but may be inactive. The indication HLD shows that this queue is held, and WTR/HLD indicates that a writer has been started and the queue is held. A blank field indicates that no writer has been started and the queue is not held.

For each spooled output file that has an entry on the output queue, one line per file is used to display:

- An input field (to the left of the file name) in which a number can be entered. The number, which can be any one of the following, causes the function shown with the number to be performed for that file when the Enter key is pressed. The information given in the dashed list describing the input field on the previous display (of all output queues) also applies to the input field on this display for files on a single output queue.

  1–DSPSPLF: Display the data in the spooled file.
  2–DSPSPLFA: Display the attributes of the spooled file.
  4–HLDSPLF: Hold the spooled file.
  6–RLSSPLF: Release the spooled file.
  9–CNLSPLF: Cancel the spooled file.

- The name and number of the spooled file.
  – File name. The file name that was specified by the user program when the file was created.
  – File number. The number of the spooled file, which indicates its sequential order in relation to the other output files produced by the same job.

- The qualified name of the job that produced the spooled file.
  – Job name. Jobs are listed in the order in which they will be selected for execution unless they are held. (For information on where the job name comes from, refer to the expanded description of the JOB parameter in Appendix A.)
  – User name. The name of the user profile under which the job that produced the file ran.
  – Job number. The six-digit number assigned by the system.

- The output priority assigned to the file from a job command or from the job description associated with the job that created the output file.

- The total number of records or pages (for printer output) in the file. If the file is still open, this field is blank. If the number is greater than 999 999, the field is displayed as +++++R (records) or +++++P (pages).

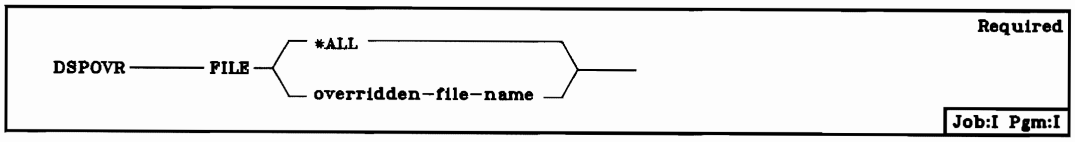
- The status of the file. One of the following values can be specified for the status:
  – RDY (ready). The file is complete and ready to be spooled to an output device.
  – OPN (open). The file has not been completely processed by a program and is not ready to be selected by a spooling writer.
  – CLO (closed). The file has been completely processed by a program, but SCHEDULE(*JOBEND) was specified in the associated device file and the job that produced the file has not yet finished.
  – HLD (held). The file has been held by a HLDOUTQ, HLDJOB, or HLDSPLF command, or SAVE(*YES) was specified in an associated device file description.
  – WTR (writer). The file is currently being produced on an output device by a spooling writer.

DSPOVR (Display Override) Command

The Display Override (DSPOVR) command displays, for one or all file overrides in an invocation, the file override information (for data, source, and message files) that exists in that invocation. The user can request either that all of the file overrides in an invocation are to be displayed or that the file override values in an invocation that affect only one file are to be displayed. Only overrides that are in the invocation in which the DSPOVR command is entered are displayed.

If all file overrides are to be displayed, only the names of the overridden files, overriding files and members are displayed. The member names are displayed only if they are specified on the OVRDBF command. If the name of a single overridden file is specified, the name of the overriding file and all overriding values affecting it are displayed.

Restrictions: This command is valid only when entered interactively.



FILE Parameter: Specifies whether all the file overrides in the invocation are to be displayed, or specifies the name of one overridden file for which all the parameters on the file override command are to be displayed.

**ALL:* The names of all the files that are affected by overrides in the invocation are to be displayed.

overridden-file-name: Enter the name of an overridden file for which all the parameters on the file override command in the invocation are to be displayed.

Examples and Displays

Example 1: Displaying Overrides in a Single Invocation

The following group of five commands demonstrates, for overrides in one invocation, the use and resulting displays of the DSPOVR command. Three file override commands are followed by two DSPOVR commands in the same invocation.

```
OVRDBF  FILE(INPUT) TOFILE(BILLS) MBR(TOPAY)
OVRPRTF  FILE(PRTFX) TOFILE(PRINT) SPOOL(*YES) +
     COPIES(2) SAVE(*YES)
OVRCRDF  FILE(CARD) SPOOL(*NO)
DSPOVR  FILE(*ALL)
DSPOVR  FILE(PRTFX)
```

The display resulting from the first DSPOVR command lists the names of all files that are currently affected by overrides in the invocation. The display lists the names of the overridden files (FILE) in the first column, the names of the overriding files (TOFILE) in the second column, the name of the library (in which the description of the overriding file is stored) in the third column, and, optionally, the name of the member in the fourth column when a member name is specified on an OVRDBF command.

The display resulting from the second DSPOVR command displays all the override information about the override in that invocation affecting the file named PRTFX, which does not have to be a printer device file. It shows that PRTFX is completely overridden by the PRINT file. The values specified in the file description for PRINT file are used, except for the overriding values specified on the OVRPRTF command: SPOOL(*YES), COPIES(2), and SAVE(*YES). Only the overriding values specified on the command are shown; the values in the PRINT file that are not overridden are not shown, even though they are used in the job.

Example 2: Displaying Overrides in Multiple Invocations

This example assumes that commands 1, 2, and 12 are entered in the same invocation, invocation number 1. The rest of the commands are in invocation 2.

```
    1.   OVRDBF  FILE(A) TOFILE(B)
    2.   CALL  PGMA                                         Invocation 1
         ------------------------------------------------------
Program PGMA
         3.   OVRPRTF  FILE(B) TOFILE(C) COPIES(3)
         4.   TFRCTL  PGMB                                  Invocation 2

Program PGMB
         5.   OVRCRDF  FILE(E) TOFILE(F)
         6.   CALL  QCAEXEC ('OVRMSGF  FILE(G) TOFILE(H)' 25)
         7.   DSPOVR  FILE(A)
         8.   MONMSG  MSGID(CPF9842)
         9.   DSPOVR  FILE(B)
        10.   CALL QCAEXEC ('DSPOVR  FILE(*ALL)'  17)
        11.   RETURN
         ------------------------------------------------------
   12.   DSPOVR  FILE(*ALL)                                 Invocation 1
```

Command 1 causes an override in invocation 1 from file A to file B. Command 2 calls PGMA and generates another invocation, invocation 2.

In program PGMA, command 3 causes an override in invocation 2 from file B to file C, and overrides the COPIES parameter in file C with a value of 3. Command 4 transfers control from PGMA to PGMB in the same invocation, invocation 2. Unlike the CALL command, the TFRCTL command does not generate a new invocation.

In program PGMB, command 5 causes an override in invocation 2 from file E to file F. Command 6 calls QCAEXEC and causes an override from file G to file H. When it is called, QCAEXEC executes as though it is just another command in PGMB, rather than executing as a called program. That is, QCAEXEC executes in the same invocation (invocation 2); it does not generate another invocation (invocation 3).

Command 7 would display an override of file A in invocation 2; but there is no override affecting file A in invocation 2, so none is displayed. (The override of file A specified by command 1 is not displayed because it is in invocation 1.) Instead, the escape message CPF9841 (override not found) is sent to PGMB. To prevent a function check, a MONMSG command is needed after the DSPOVR command. In this example, command 8 monitors for CPF9841, but specifies no action to be taken if the message is sent. Therefore, when CPF9841 is received, it is monitored and ignored by command 8, and control is passed to the next command in the program.

Command 9 displays an override of file B in invocation 2; the printer file override specified by command 3 is displayed. Command 10 displays all overrides specified in invocation 2. The overrides specified by commands 3, 5, and 6 are displayed, but the override specified by command 1 is not displayed.

Command 11 causes a return to invocation 1, and invocation 2 is terminated. When invocation 2 terminates, the overrides specified by commands 3, 5, and 6 are implicitly deleted. Command 12 causes all overrides specified in invocation 1 to be displayed; the override specified by command 1 is displayed.

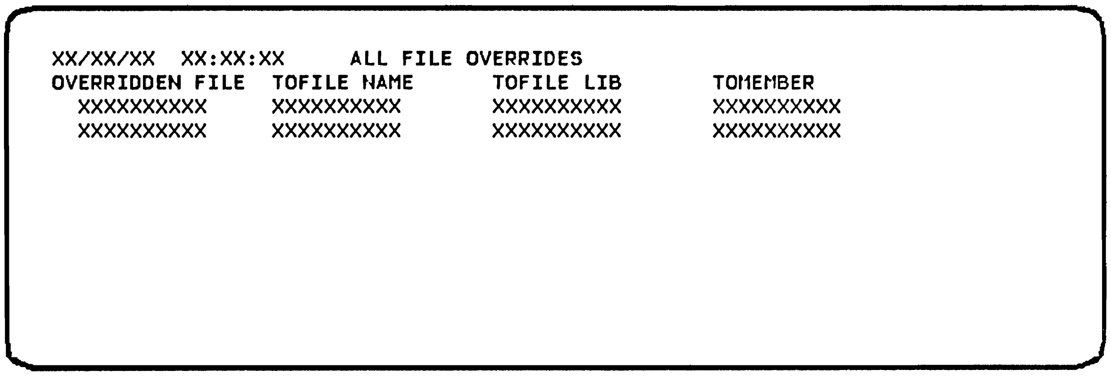
The display resulting from command 9 displays all the information about the override (command 3, OVRPRTF) that affects file B . The information displayed is not the attributes of file B, but of file C. It shows that B is completely overridden by C, and that the number of copies specified in the file description of file C is overridden with a value of 3.

The display resulting from command 10 shows the names of all files in invocation 2 that are affected by overrides in invocation 2. The display lists the names of the overridden files in the first column, the names of the overriding files and their libraries in the second and third columns, and (optionally) the name of the member in the fourth column when a member name is specified on an OVRDBF command.

Additional Considerations

One of several sets of displays can be presented as a result of the DSPOVR command. The display or displays shown depend on the value specified in the FILE parameter, and on the type of device file or data base file that is overriding the file specified in the FILE parameter.

If FILE(*ALL) is specified in the DSPOVR command, a single display is presented that shows all of the overrides that are in the invocation in which the command is entered. The display of all the overrides has the following format:



```
XX/XX/XX  XX:XX:XX     ALL FILE OVERRIDES
OVERRIDDEN FILE  TOFILE NAME      TOFILE LIB       TOMEMBER
  XXXXXXXXXX     XXXXXXXXXX       XXXXXXXXXX       XXXXXXXXXX
  XXXXXXXXXX     XXXXXXXXXX       XXXXXXXXXX       XXXXXXXXXX
```

The first line gives the current job date and the time when the DSPOVR command was entered. The second line shows four column headings below which the following are displayed for each file override in this invocation:

- The name of the file being overridden
- The name of the file being used instead of the overridden file
- The name of the library in which the description of the overriding file is stored
- If the overriding file is a data base file, the name of the member within the file that is used as the overriding member

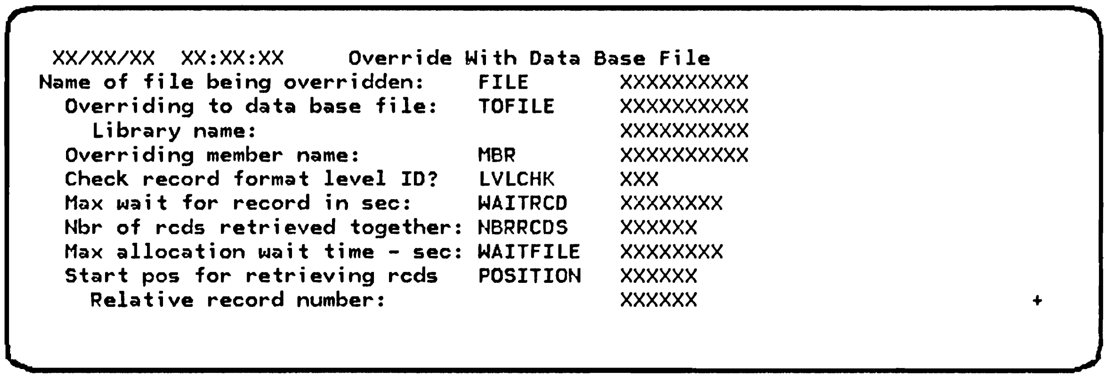
If a specific file name is specified on the FILE parameter of the DSPOVR command, the series of displays presented depends on whether the *overriding* file is a:

- Data base file
- BSC device file
- Communications device file
- Card device file
- Diskette device file
- Display device file
- Printer device file
- Tape device file
- Message file

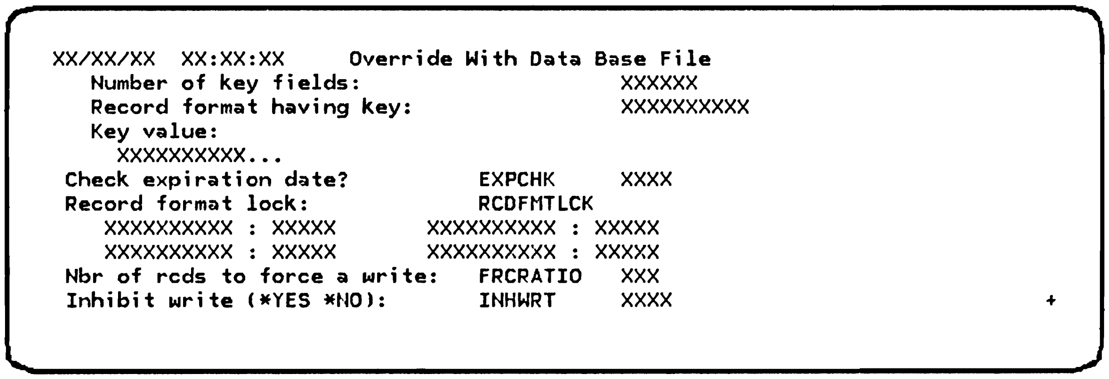
Regardless of the type of overriding file, the following is true for all of the displays of a single overriding file:

- The first line of the display shows the current job date and time, and the *type* of the overriding file (in the title of each display). The names of the overridden and overriding files appear only on the first display of the set.
- The rest of the lines display the attributes of the *overriding* file that are currently in effect; attributes that have no override value specified on the associated override command in the invocation are not shown.
- The explanation of each attribute displayed is to be found in the associated parameter description (whose keyword is also shown on the display) given in the associated override command. For example, in the following data base file override displays, you would refer to the POSITION parameter description given in the OVRDBF command description for information about the *file position* attribute (and about the four following lines, which are also part of the POSITION parameter).

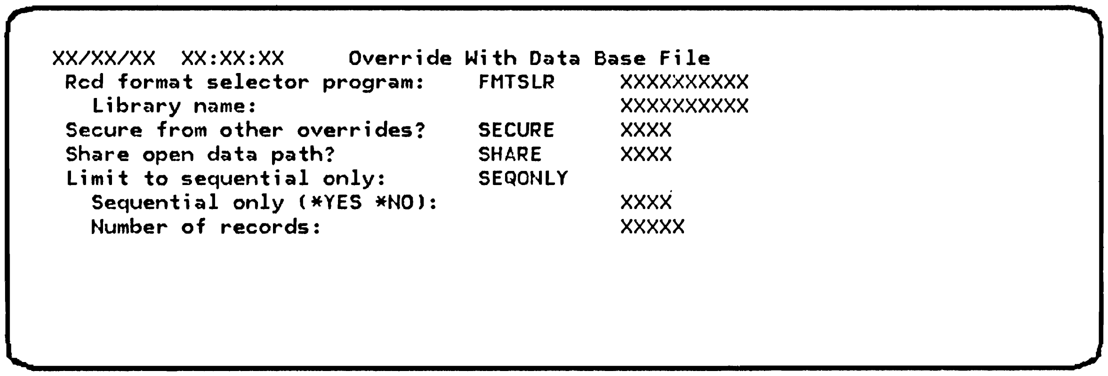
One set of displays is presented here as an example of what all the override displays are like. This set of three displays shows all the displayable attributes of an overriding data base file:



```
 XX/XX/XX  XX:XX:XX     Override With Data Base File
Name of file being overridden:    FILE       XXXXXXXXXX
  Overriding to data base file:   TOFILE     XXXXXXXXXX
    Library name:                            XXXXXXXXXX
  Overriding member name:         MBR        XXXXXXXXXX
  Check record format level ID?   LVLCHK     XXX
  Max wait for record in sec:     WAITRCD    XXXXXXXX
  Nbr of rcds retrieved together: NBRRCDS    XXXXXX
  Max allocation wait time - sec: WAITFILE   XXXXXXXX
  Start pos for retrieving rcds   POSITION   XXXXXX
    Relative record number:                  XXXXXX                          +
```



```
 XX/XX/XX  XX:XX:XX     Override With Data Base File
    Number of key fields:                    XXXXXX
    Record format having key:                XXXXXXXXXX
    Key value:
      XXXXXXXXXX...
  Check expiration date?          EXPCHK     XXXX
  Record format lock:             RCDFMTLCK
     XXXXXXXXXX : XXXXX       XXXXXXXXXX : XXXXX
     XXXXXXXXXX : XXXXX       XXXXXXXXXX : XXXXX
  Nbr of rcds to force a write:   FRCRATIO   XXX
  Inhibit write (*YES *NO):       INHWRT     XXXX                             +
```

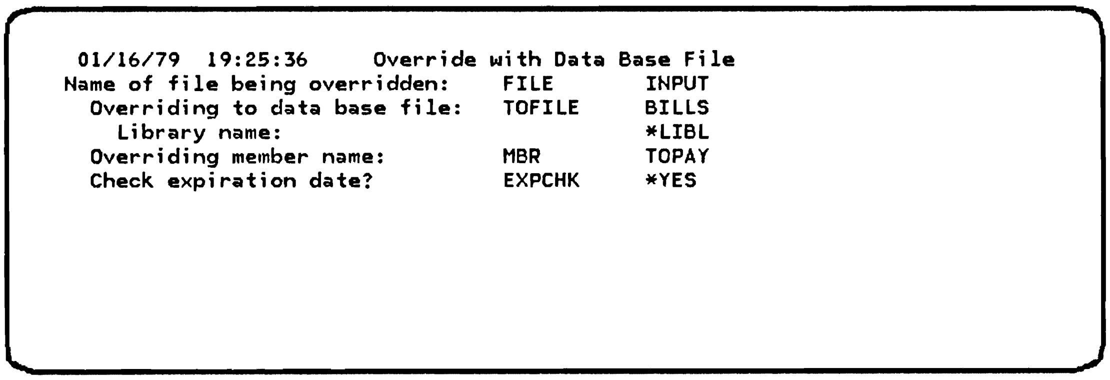


```
 XX/XX/XX  XX:XX:XX     Override With Data Base File
  Rcd format selector program:    FMTSLR     XXXXXXXXXX
    Library name:                            XXXXXXXXXX
  Secure from other overrides?    SECURE     XXXX
  Share open data path?           SHARE      XXXX
  Limit to sequential only:       SEQONLY
    Sequential only (*YES *NO):              XXXX
    Number of records:                       XXXXX
```

Note that, although three displays are shown here for an overriding data base file, in many cases only one display may be needed. Those attributes for which an override value was not specified in the OVRDBF command are not shown, and the space left by them is closed up by those that were specified. For example, if the only override applied to a file named INPUT was:

```
OVRDBF  FILE(INPUT) TOFILE(BILLS) MBR(TOPAY) +
     EXPCHK(*YES)
```

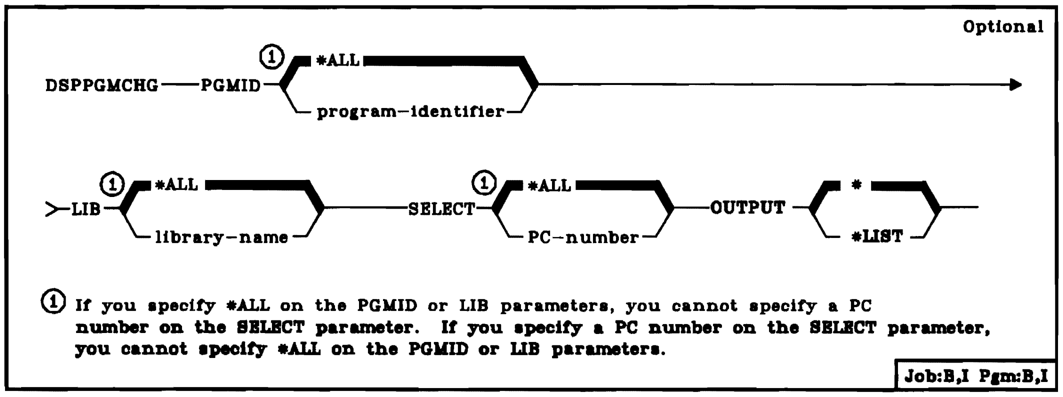
the following display would result if DSPOVR FILE(INPUT) was then entered:



```
 01/16/79  19:25:36     Override with Data Base File
Name of file being overridden:    FILE       INPUT
  Overriding to data base file:   TOFILE     BILLS
    Library name:                            *LIBL
  Overriding member name:         MBR        TOPAY
  Check expiration date?          EXPCHK     *YES
```

DSPPGMCHG (Display Programming Change) Command

The Display Programming Change (DSPPGMCHG) command displays programming changes (PCs) and locally generated programming changes (patches) for a specified program product and library. For a PC or patch to be displayed, there must be an index entry for it in the master programming change index (MPCI) of the library.



① If you specify *ALL on the PGMID or LIB parameters, you cannot specify a PC number on the SELECT parameter. If you specify a PC number on the SELECT parameter, you cannot specify *ALL on the PGMID or LIB parameters.

PGMID Parameter: Specifies the identifier of the program product for which PCs are to be displayed.

<u>*ALL:</u> Specifies that PCs of all program products are to be displayed.

program-identifier: Specifies a particular program product for which PCs are to be displayed.

LIB Parameter: Specifies the name of the library from which PCs are to be displayed.

<u>*ALL:</u> Specifies that all the libraries on the system are to be searched for PCs to display.

library-name: Specifies a particular library from which PCs are to be displayed.

SELECT Parameter: Specifies which PC is to be displayed for the specified program product and library. *ALL cannot be specified for the PGMID or LIB parameter if a PC number is specified on the SELECT parameter.

*ALL: The status of all PCs and patches for the specified program product in the specified library are to be displayed.

PC-number: Enter the PC identification number of the PC that is to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled printer output.

*: The output is to be displayed (if requested by an interactive job) or listed with the job's system output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled printer output.

Examples

```
DSPPGMCHG
```

This command displays the status of all PCs and all patches for every program product in all libraries on the system. The output will be displayed at the work station from which the command was entered.

```
DSPPGMCHG  PGMID(5714SS1)  LIB(QSYS)  OUTPUT(*LIST)
```

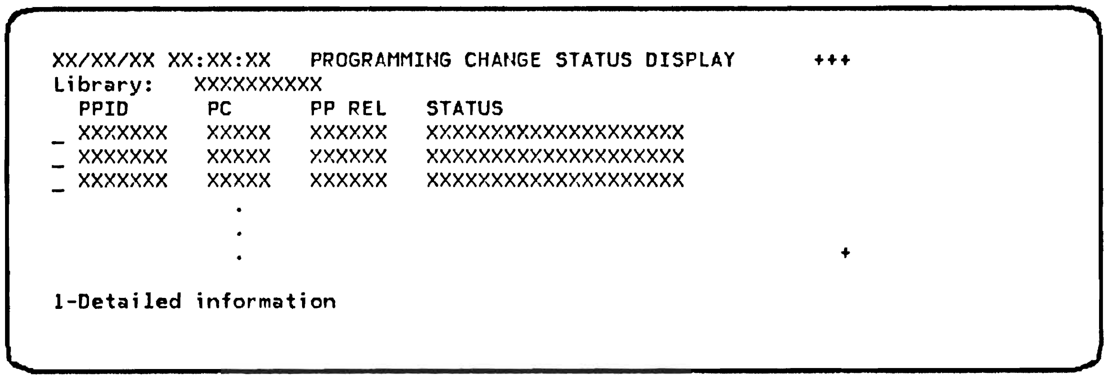
This command will cause a listing to be produced which will contain the status of PCs and patches for the program product 5714SS1 in the QSYS library.

```
DSPPGMCHG  PGMID(5714SS1)  LIB(QSYS) SELECT(00034) OUTPUT(*list)
```

This command will cause a listing to be produced which will contain detailed information about PC 00034 for program product 5714SS1 in the QSYS library.

Additional Considerations

The DSPPGMCHG command uses the following display to show information related to PCs that have been loaded on the system:



```
XX/XX/XX XX:XX:XX   PROGRAMMING CHANGE STATUS DISPLAY      +++
Library:   XXXXXXXXXX
  PPID       PC      PP REL   STATUS
_ XXXXXXX   XXXXX   XXXXXX   XXXXXXXXXXXXXXXXXXXX
_ XXXXXXX   XXXXX   XXXXXX   XXXXXXXXXXXXXXXXXXXX
_ XXXXXXX   XXXXX   XXXXXX   XXXXXXXXXXXXXXXXXXXX
              .
              .
              .                                        +

1-Detailed information
```

The first line of the display shows the date and time the request for the display was made. The second line shows the name of the library in which the PCs are stored. Beginning with the third line, the following information is displayed for each PC:

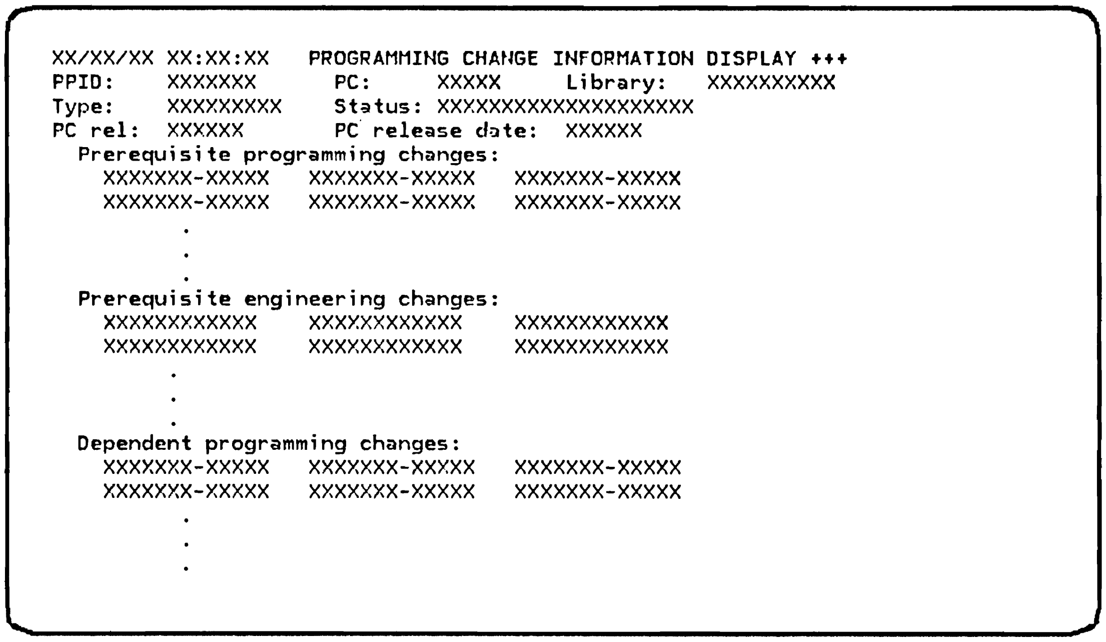
- PPID. The program product identification number of the program product to which the PC applies.

- PC. The number used to identify the PC.

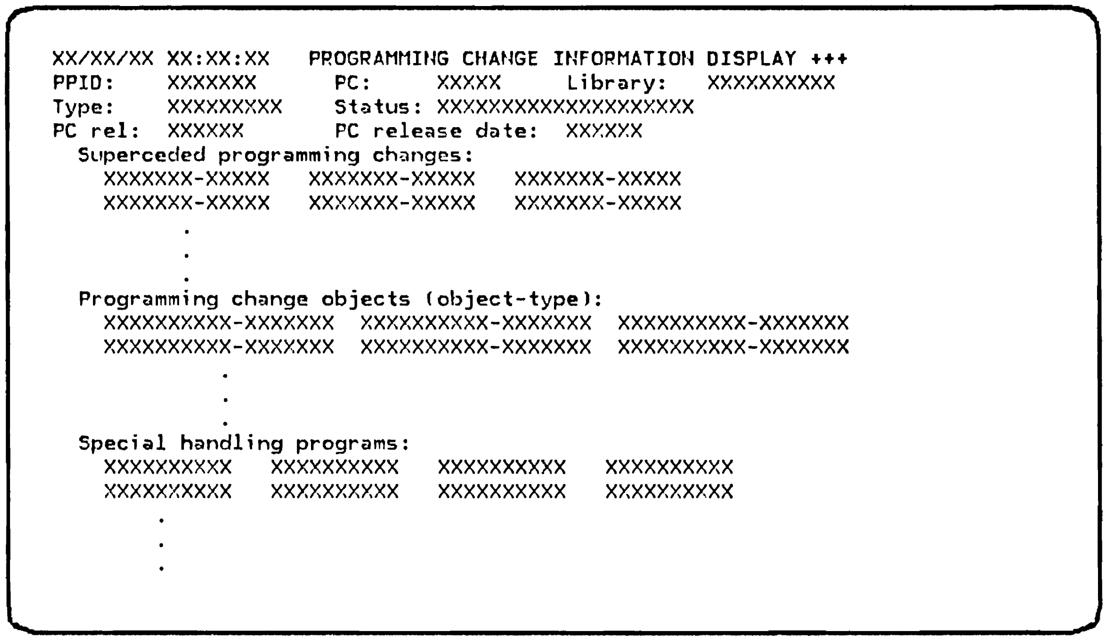
- PP REL. The release level of the installed program.

- STATUS. The release level of the installed program.
  - Permanently applied
  - Temporarily applied
  - Superceded
  - Damaged
  - Not applied

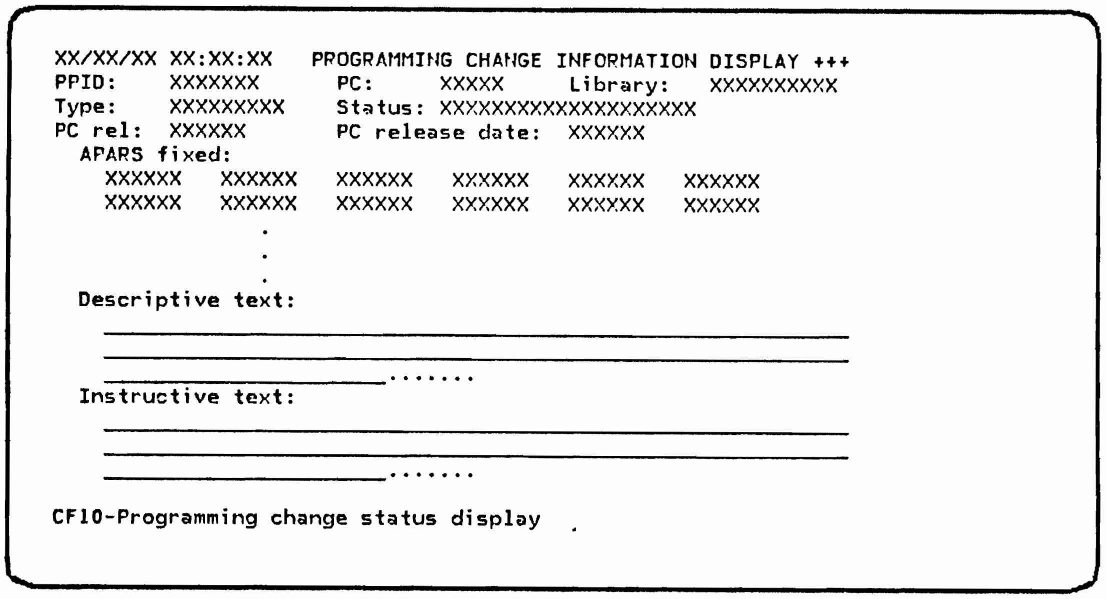
If a 1 is entered before one or more of the program product identifiers shown, the following displays are used to show a detailed description of each PC:



```
XX/XX/XX XX:XX:XX   PROGRAMMING CHANGE INFORMATION DISPLAY +++
PPID:    XXXXXXX     PC:     XXXXX     Library:   XXXXXXXXXX
Type:    XXXXXXXXX   Status: XXXXXXXXXXXXXXXXXXXX
PC rel:  XXXXXX      PC release date:  XXXXXX
  Prerequisite programming changes:
    XXXXXXX-XXXXX   XXXXXXX-XXXXX   XXXXXXX-XXXXX
    XXXXXXX-XXXXX   XXXXXXX-XXXXX   XXXXXXX-XXXXX
         .
         .
         .
  Prerequisite engineering changes:
    XXXXXXXXXXXX    XXXXXXXXXXXX    XXXXXXXXXXXX
    XXXXXXXXXXXX    XXXXXXXXXXXX    XXXXXXXXXXXX
         .
         .
         .
  Dependent programming changes:
    XXXXXXX-XXXXX   XXXXXXX-XXXXX   XXXXXXX-XXXXX
    XXXXXXX-XXXXX   XXXXXXX-XXXXX   XXXXXXX-XXXXX
         .
         .
         .
```



```
XX/XX/XX XX:XX:XX   PROGRAMMING CHANGE INFORMATION DISPLAY +++
PPID:    XXXXXXX     PC:     XXXXX     Library:   XXXXXXXXXX
Type:    XXXXXXXXX   Status: XXXXXXXXXXXXXXXXXXXX
PC rel:  XXXXXX      PC release date:  XXXXXX
  Superceded programming changes:
    XXXXXXX-XXXXX   XXXXXXX-XXXXX   XXXXXXX-XXXXX
    XXXXXXX-XXXXX   XXXXXXX-XXXXX   XXXXXXX-XXXXX
         .
         .
         .
  Programming change objects (object-type):
    XXXXXXXXXX-XXXXXXX  XXXXXXXXXX-XXXXXXX  XXXXXXXXXX-XXXXXXX
    XXXXXXXXXX-XXXXXXX  XXXXXXXXXX-XXXXXXX  XXXXXXXXXX-XXXXXXX
            .
            .
            .
  Special handling programs:
    XXXXXXXXXX   XXXXXXXXXX   XXXXXXXXXX   XXXXXXXXXX
    XXXXXXXXXX   XXXXXXXXXX   XXXXXXXXXX   XXXXXXXXXX
       .
       .
       .
```



```
XX/XX/XX XX:XX:XX   PROGRAMMING CHANGE INFORMATION DISPLAY +++
PPID:    XXXXXXX      PC:      XXXXX     Library:   XXXXXXXXXX
Type:    XXXXXXXXX    Status: XXXXXXXXXXXXXXXXXXXX
PC rel:  XXXXXX       PC release date:  XXXXXX
  APARS fixed:
    XXXXXX   XXXXXX   XXXXXX   XXXXXX   XXXXXX   XXXXXX
    XXXXXX   XXXXXX   XXXXXX   XXXXXX   XXXXXX   XXXXXX
              .
              .
              .
  Descriptive text:
    ____________________________________________________________
    ____________________________________________________________
    _______________________.......
  Instructive text:
    ____________________________________________________________
    ____________________________________________________________
    _______________________.......

CF10-Programming change status display
```

Notes:

1. For PCs that are permanently applied, information from line 4 and down is unavailable for display. A message will appear on the screen stating that no further information is available because the PC is permanently applied.
2. For PCs that are superceded, information from line 4 and down is unavailable for display. A message will appear on the screen stating that no further information is available because the PC is superceded.

The first line of the display shows the date and time the request for the display was made. The second line shows the value used on the first display to identify the PC. In addition to the status of the PC, the third line shows whether the PC is of type deferred or immediate. The fourth line shows the release level of the program product to which the PC applies and the date the PC was released for use.

Beginning with the fifth line, the following information is displayed:

- Prerequisite programming changes: A list of identifiers of PCs that must be applied before this PC can be applied.

- Prerequisite engineering changes: A list of identifiers of engineering changes (ECs) that must be applied before this PC can be applied.

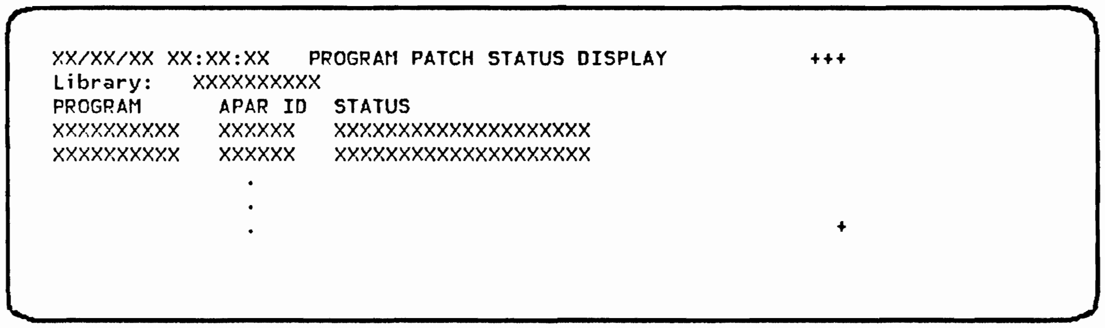
- Dependent programming changes: A list of PCs that are dependent on this PC.

- Superceded programming changes: A list of PCs whose functions have been replaced by this PC.

- Programming change objects: A list of CPF objects that are affected by the function of this PC.
- Special handling programs: A list of exiting programs that are invoked by this PC.
- APARs fixed: A list of identifiers for APARs (authorized program analysis reports) that have been corrected by this PC.

At the bottom of the display text fields are shown that briefly describe this PC and provide instructions on how to use and apply this PC.

Program patches can be shown by pressing Enter on the initial PC display. If any program patches exist, the following display is shown:



```
XX/XX/XX XX:XX:XX   PROGRAM PATCH STATUS DISPLAY          +++
Library:   XXXXXXXXXX
PROGRAM       APAR ID  STATUS
XXXXXXXXXX    XXXXXX   XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXX    XXXXXX   XXXXXXXXXXXXXXXXXXXX
                 .
                 .
                 .                                         +
```

The first line of the display shows the date and time the request for the display was made. The second line shows the name of the library in which the program patches are stored. Beginning with the third line, the following values are shown:

- PROGRAM. The name of the patched program.
- APAR ID. The APAR number assigned to the program patch.
- STATUS. The status of the program patch, which can be:
  - Temporarily applied
  - Damaged
  - Not applied

DSPPGMREF (Display Program References) Command

The Display Program References (DSPPGMREF) command provides a list of the CPF objects referenced by the specified program(s). For CL programs, only files are listed; for RPG programs, only files and data areas are listed. For COBOL programs, referenced files and any programs called that use a non-numeric literal are listed. This information can be displayed, printed, and/or put in a data base file so that a user program can extract data from it as needed.

If the information is printed, a list (by library) of the specified user-authorized programs along with the objects referenced by each program is generated. For files, an indication of program usage of each file (input, output, update, unspecified, or any combination of these four) is also printed.

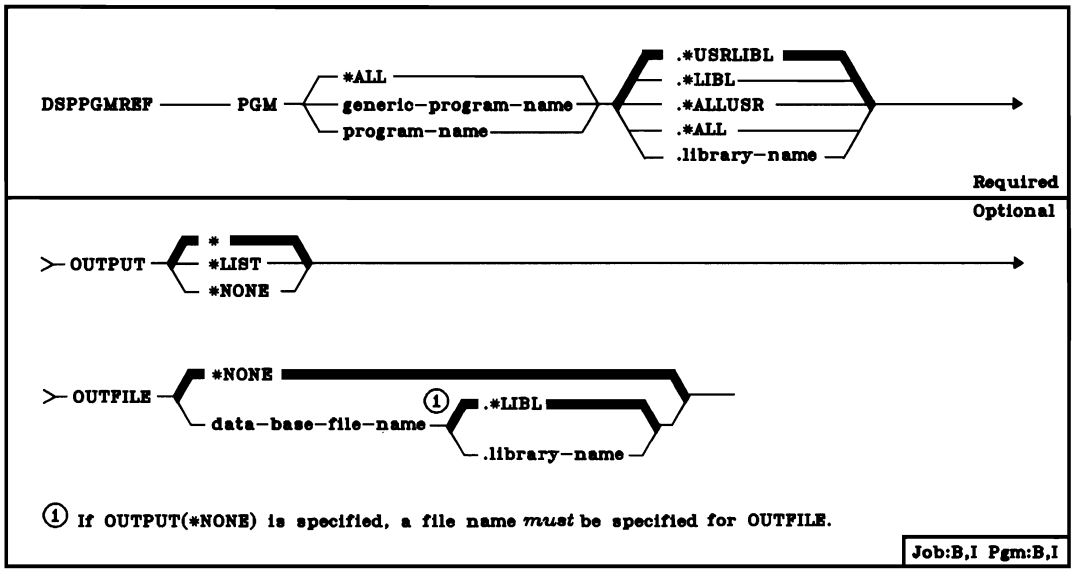
If the information is put in a data base file, a record composed of the following fields is produced. (The format of the record in the data base is *not* related to the format of the printed output.) The data base format is the same as that used in the IBM-supplied data base file QADSPPGM; the format is described in the *Application Documentatation* chapter of the *CPF Programmer's Guide*.

- The name of the program and its text description
- The name of the library containing the program
- The number of objects referenced by the program
- The qualified name of the CPF object
- The information retrieval date(s)

For files, the record contains the following additional fields:

- The name of the file in the program (possibly different from the CPF object name if an override was in effect when the program was created)
- The program usage of the file (input, output, update, unspecified, or a combination of these four)
- The number of record formats referenced, if any
- The name of the record format used by the file and its record format level identifier
- The number of fields referenced for each format

Restrictions: Before you can display the objects in a program, you must have operational authority or read rights for the program. Also, of the libraries specified by the library qualifier, only the libraries for which you have read rights are searched for the programs.



PGM Parameter: Specifies the qualified name of the program (or all programs) that is to have a list of the files and other related CPF objects displayed that are referenced by the program. A specific program name or a generic program name can be specified; either type can be optionally qualified by a library name. If no library qualifier is given, all the libraries in the user portion of the job's library list are assumed (by *.*USRLIBL*). Only the libraries in the specified library qualifier that the user either owns or is authorized to use are searched for the program(s).

Depending on the library qualifier specified or assumed, the following libraries (for which the user has the authority) are to be searched for the programs specified:

- .*USRLIBL (user library list). Only the libraries listed in the user portion of the job's library list. If a specific program name is given, only the first program found by that name is displayed.

- *.*LIBL* (library list). All the libraries in the user *and* system portions of the job's library list. If a specific program name is given, only the first program found by that name is displayed.

- *.*ALLUSR* (all user libraries). All the nonsystem libraries, which include *all* user-defined libraries and the QGPL library, not just those in the job's library list. Libraries other than QGPL that begin with the letter Q are not included.

- *.*ALL* (all libraries). All the libraries in the system, including QSYS.

- *.library-name* (one library). Only the library named in this parameter. The user must have read rights for the specified library.

**ALL:* All programs in the specified library (or all the libraries identified in the library qualifier to which the user has access) are to have file-related information displayed.

qualified-generic-program-name: Enter the qualified name of the program or the generic name of several programs in the specified library (or all libraries identified in the library qualifier to which the user has access) that is to have file-related information displayed. To specify a generic program name, add an asterisk (*) at the end of the characters that are in the names of all the programs desired.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the names of the files used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

**NONE:* The only output is to be to the data base file specified in OUTFILE.

OUTFILE Parameter: Specifies the name of the data base file into which the program-related file information is to be stored. If the specified file does not exist, this command causes a data base file and member to be created.

<u>*NONE:</u> The specified information is not to be put into a data base file.

qualified-data-base-file-name: Enter the qualified name of the data base file into which the program-related information is to be placed. (If no library qualifier is given, *LIBL is used to find the file.) If the specified file name is not found, a file and member by that name is created and stored in the specified library, or in QGPL if not qualified. This file can be reused when other DSPPGMREF commands are entered. Output from the file always starts at the beginning of the file member. (The IBM-supplied data base file QADSPPGM *cannot* be specified.)

Examples

```
DSPPGMREF  PGM(*ALL.LIBRARY1)  OUTFILE(FILE2.LIB2)
```

This command creates a list of all authorized programs found in LIBRARY1, and of the files and other CPF objects that the programs reference. It also causes the list to be displayed or printed (depending on where the command is entered) and stores the list in a data base file named FILE2 in LIB2.

```
DSPPGMREF  PGM(BILLING.LIBRARY1) OUTFILE(*NONE) +
      OUTPUT(*LIST)
```

This command creates a list of all the files that are referenced by the BILLING program, which is stored in LIBRARY1. The output is spooled for printing by a spooling writer.

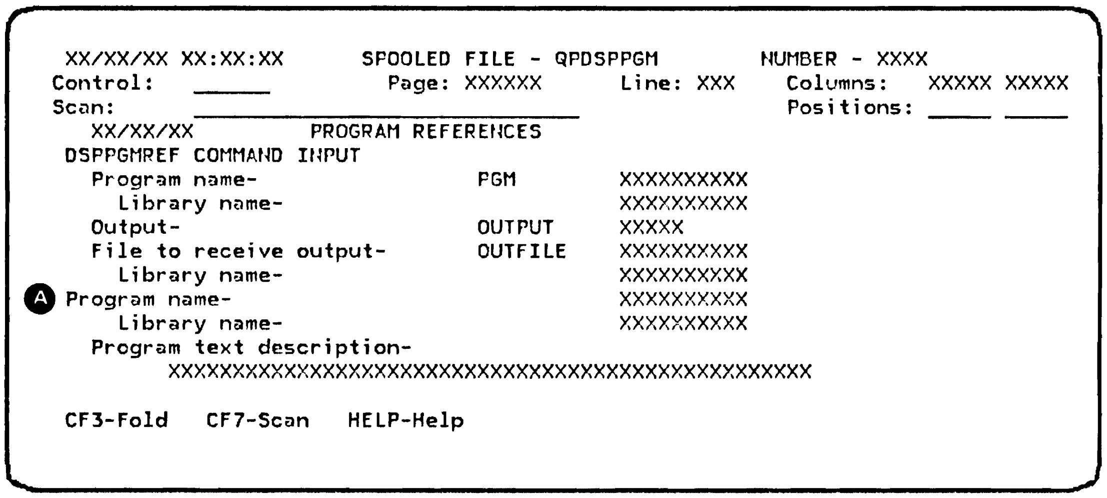
Additional Considerations

When the DSPPGMREF command is entered, the library is searched for the program or programs specified on the PGM parameter; then a group of records that give CPF object-related information about each program is generated. The records are placed in the printer device file named QPDSPPGM. If OUTPUT(*LIST) is specified on the command, the records are listed on the printer in the following order:

- Header information: Lists the DSPPGMREF command input values.
- Program-related information: Lists, for each program identified by the PGM parameter, all of the CPF objects (including files) that are referenced by the program. If more than one program is identified, the beginning of the next one follows the end of the previous one.

If the DSPPGMREF command is entered interactively and OUTPUT(*) is specified or assumed, the records in the printer device file are displayed rather than printed. The first three lines of each display contain information about the spooled file and input fields used for display handling functions. For more information on the uses and meanings of these fileds, refer to the *Additional Considerations* section of the Display Spooled File (DSPSPLF) command. The first three lines of each display contain:

- The current job date
- The name of the spooled printer file (QPDSPPGM) into which all records containing the program-related information are placed after they are generated
- The spooled file number of the display device file created by the DSPPGMREF command
- The page, line, and column numbers of the spooled file being shown



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPPGM       NUMBER - XXXX
Control:  ______          Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
Scan:     ______________________________                 Positions: _____ _____
  XX/XX/XX       PROGRAM REFERENCES
 DSPPGMREF COMMAND INPUT
   Program name-                PGM        XXXXXXXXXX
     Library name-                         XXXXXXXXXX
   Output-                      OUTPUT     XXXXX
   File to receive output-      OUTFILE    XXXXXXXXXX
     Library name-                         XXXXXXXXXX
Ⓐ Program name-                            XXXXXXXXXX
     Library name-                         XXXXXXXXXX
   Program text description-
         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 CF3-Fold   CF7-Scan   HELP-Help
```

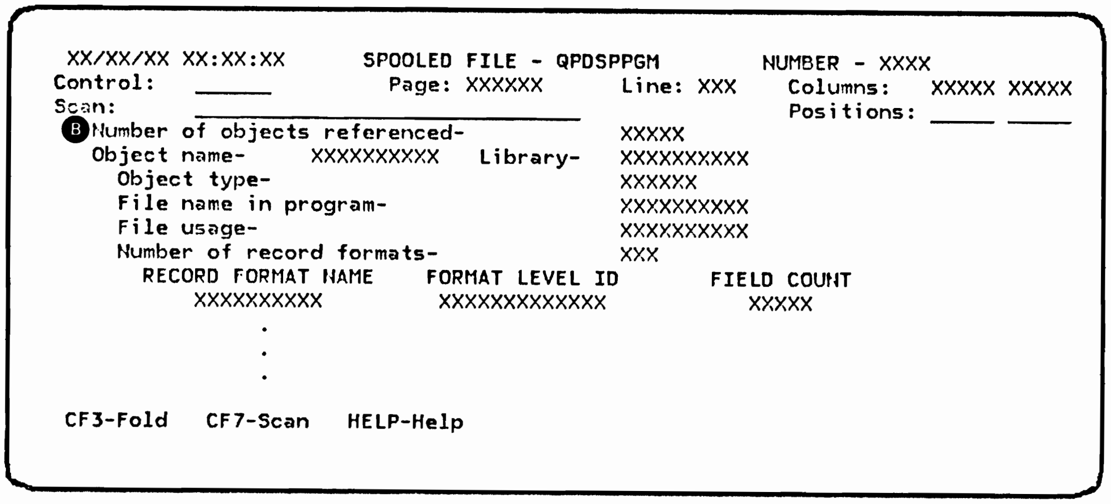
Line 4 of the first display gives the current job date and the name of the displayed information. (This is record 000001 in the printer device file.) The other lines show all of the values specified on the DSPPGMREF command when it was entered.

If more than one program is to be displayed, based on the value specified in the PGM parameter, *all* of the first program's information is shown before the next program is displayed (beginning at Ⓐ on the first display and continuing through the end of the second display). If multiple objects are referenced by the program, all of the attributes for one object are displayed (beginning at Ⓑ on the second display and continuing to the end) before the next object's name is displayed.

For each program having referenced object information displayed, the information is displayed in the order shown. If one or more attributes do not apply to the named program (at A) or object (at B), those lines are not shown, and the lines that would be left blank are filled with the next attributes that do apply.

Ⓐ Beginning with line 11 of the first display, the following attributes of the named program are shown:

- The name of the program having the reference information displayed, and the name of the library in which the program is stored.
- The text describing the program; the text was specified on the command that created the program.



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - QPDSPPGM        NUMBER - XXXX
Control:    _____         Page: XXXXXX       Line: XXX    Columns:   XXXXX XXXXX
Scan:                                                     Positions: _____ _____
ⒷNumber of objects referenced-              XXXXX
  Object name-     XXXXXXXXXX   Library-    XXXXXXXXXX
    Object type-                            XXXXXX
    File name in program-                   XXXXXXXXXX
    File usage-                             XXXXXXXXXX
    Number of record formats-               XXX
      RECORD FORMAT NAME    FORMAT LEVEL ID      FIELD COUNT
          XXXXXXXXXX          XXXXXXXXXXXXX         XXXXX
                .
                .
                .

 CF3-Fold   CF7-Scan   HELP-Help
```

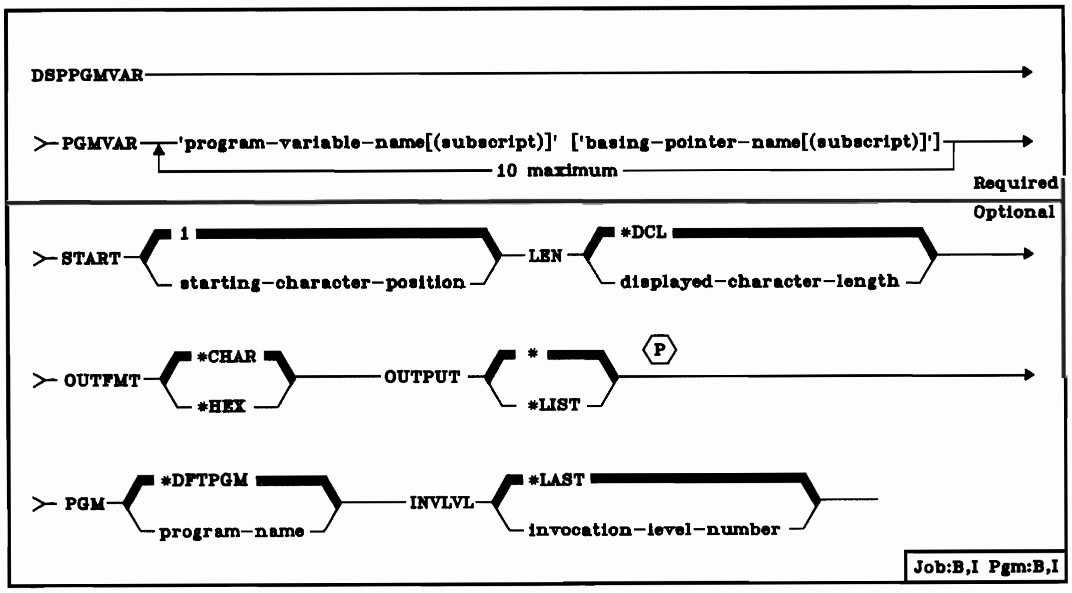
Ⓑ Beginning with line 4 of the second display, the following object-related information about the named CPF object is shown:

- The number of CPF objects that are referenced by the program. (The attributes of each one follow.)
- The name of the object (or program variable) whose attributes follow, and the name of the library in which the object is stored. In the case of a program variable, no other information about it (except its name) is displayed.
- The type of the object.
- If the object is a file, the following file-related information can be displayed:
  - The file name used in the program, which can be different from the actual name of the file referenced by the program.
  - How the file is used by the program. The field can contain INPUT, OUTPUT, UPDATE, UNSPECIFIED, or any combination of the four.
  - The number of record formats in the file that are referenced in the program. If the program references only the file name and not the record format(s) in the file, no record format information is displayed. (The attributes of each one follow.)
  - The name of each record format used in the file, and the most recent identifier assigned to the record format.
  - The number of fields used in the record format.

DSPPGMVAR (Display Program Variable) Command

The Display Program Variable (DSPPGMVAR) command displays the current value of one or more program variables in a program that is being debugged. The variables can be specified either by their variable names or by their MI ODV numbers. A maximum of 10 variables can be specified.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*



PGMVAR Parameter: Specifies the names of one or more program variables whose values are to be displayed. The variables can be in an HLL or MI program.

'program-variable-name': Enter the names of one or more program variables (no more than 10) to be displayed when a breakpoint is reached. If the variable name contains special characters (such as the & in a CL variable name), it must be enclosed in apostrophes. An example is: PGMVAR('&VAR2').

An RPG indicator or an MI ODV (machine interface object definition table directory vector) number can be specified instead of a program variable name. An example of an RPG indicator is: PGMVAR('*IN22'). The ODV number must be preceded by a slash: PGMVAR('/264') for example.

COBOL-qualified program variable names may be specified in this parameter. These names have the following syntax:

```
var-name-1 OF/IN var-name-2 OF/IN varname-3...varname-N
```

where varname-N is the last possible variable name that will fit into the input field of the PGMVAR parameter. The input field length for each variable in the PGMVAR parameter is 98 characters. The subscript specified for a qualified variable name may also be a qualified variable name. A qualified variable name (or one with a subscript), including blanks and parentheses, must be contained within the 98-character limit. The 98-character limit includes the necessary keywords (OF/IN) and blanks, but does not include the enclosing apostrophes.

For COBOL variable names, one combination of variable name length and subscript length which will fit into the 98 character limit is valid. For example, one qualified variable name 98 characters in length (including the keywords OF or IN) can be used with no subscript, or a one-character variable name may be used with a qualified variable name (used as a subscript that uses the other 97 spaces, including parentheses).

'program-variable-name[(subscript)]': For variables in an array, enter the name of the variable and (optionally) the subscript representing the positional element in the array that is to be displayed. If a subscript is not specified, all elements in the array are displayed. The subscript, if specified, must be enclosed in parentheses, and the variable name and subscript number must be enclosed in apostrophes. No more than 10 sets can be specified and blanks must separate each set. An example is: PGMVAR('A(5)' 'B(5)' 'C(5)')

For COBOL, the following apply:

- Variable names used in qualifying strings must be high-level language variable names (qualification with ODVs is not allowed).
- Either keyword (OF or IN) is allowed.
- Each OF/IN keyword must be separated from adjacent variable names by at least one blank.
- A qualified variable name can be used as a variable subscript.
- The order the variable names are specified must be from the lowest to the highest levels in the structure.
- Structure levels may be skipped; enough levels must be specified, however, to uniquely identify the variable.
- Qualified variable names must be enclosed in apostrophes, since they contain blank characters.

Either an integer or another variable name can be specified for each subscript.

['basing-pointer-name[(subscript)]']: This set of values in the PGMVAR parameter applies only to MI or HLL programs that support based-on variables. The values can optionally be used with either of the previous two choices to also specify the value in an array that is based on a pointer. The same description of the coding syntax applies here. An example is:

```
PGMVAR(('VAR1(5)' 'PTR1(5)') ('VAR2(8)' 'PTR2(8)'))
```

This example shows that a different array element in each of two program variables is to be displayed. The fifth element in the array named VAR1, which is based on the fifth element in the pointer array named PTR1, and the eighth element in the VAR2 array, based on the eighth element in the PTR2 pointer array, are to be displayed.

The field length for the basing pointer name is 24 characters.

START Parameter: Specifies, for character variables only, the beginning position in the variable from which its value is to be displayed. If more than one character variable is specified in the PGMVAR parameter, the same starting position is used for each one.

1: The variable is to be displayed from the first position on through the length specified in the LEN parameter.

starting-character-position: Enter the position number from which the variable is to be displayed. The position number (as well as the *combination* of START and LEN) must be no greater than the length of the shortest variable specified in the PGMVAR parameter.

LEN Parameter: Specifies the number of bytes to be displayed from the character variable specified in the PGMVAR parameter, starting at the position specified in the START parameter. If more than one character variable is specified in the PGMVAR parameter, the same length is used for each one.

*DCL: The character variable is to be displayed to the end of the variable or for 200 bytes, whichever is less.

displayed-character-length: Enter the number of characters that are to be displayed. The length (as well as the *combination* of START and LEN) must be no longer than the length of the shortest variable specified in the PGMVAR parameter.

OUTFMT Parameter: Specifies the format in which the values of the program variables are to be displayed. Additional descriptive information for some variable types is also displayed with the variable values in a format predefined by the system.

*CHAR: The program variables are to be displayed in character form, which is the standard character format.

**HEX:* The program variables are to be displayed in hexadecimal form.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

PGM Parameter: Specifies the name of the program that contains the program variables to be displayed.

*DFTPGM: The program previously specified as the default program is to have its variables displayed.

program-name: Enter the name of the program whose program variables are to be displayed.

INVLVL Parameter: Specifies from which invocation level of the program the program variable is to be displayed. Invocation level 1 is the first (or earliest) invocation of the program, invocation level 2 is the second invocation, and so on down to the last (most recent) invocation level in the stack.

*LAST: The last (most recent) invocation of the program is to have its variables displayed.

invocation-level-number: Enter the number of the invocation level of the program that is to have its variables displayed.

Examples

```
DSPPGMVAR  PGMVAR('&QUANT')  PGM(MYPROG)
```

Assuming that the program MYPROG is in debug mode, this command displays the name and current value of the CL variable called &QUANT; its type and length are also displayed.

```
DSPPGMVAR  PGMVAR(TOTSALES  MANHRS)  +
     PGM(REGION)  INVLVL(1)
```

This command displays the program variables TOTSALES and MANHRS of the first invocation of the program REGION.

Additional Considerations

Three types of formats are produced by the DSPPGMVAR command, depending on whether the variables to be displayed are pointer or nonpointer program variables, or whether they identify message monitors. The formats shown in the following three displays may be actually intermixed on one or more displays, depending on the order of the program variable names specified in the PGMVAR parameter. Also, if a series of displays are generated by one DSPPGMVAR command, the information on lines 2 and 3 is displayed only on the first display of the series.

The basic program variable display has the following format:

```
 XX/XX/XX  XX:XX:XX     PROGRAM VARIABLE DISPLAY
ⒶProgram:    XXXXXXXXXX    Inv lvl:   XXXXXXXXXXX
 Output start pos:  XXXXX  Length:    XXXXX  Format:  XXXXX
ⒷVariable:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
             XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
             XXXXXXXXXXX
   Base:     XXXXXXXXXXXXXXXXXXXXXXXX
   Type:     XXXXXXXXX         Length:     XXXXX
   Dimension:       XXXXX
   'XXXXXXXXXXXXXXXXXXXX...'
    .
    .
    .
```

This display shows all of the program variable information for as many as 10 program variables within the specified program invocation (on the PGM and INVLVL parameters).

Ⓐ Lines 2 and 3 contain information that is common to all of the variables displayed (of all three types). If multiple displays occur for one DSPPGMVAR command, only lines 2 and 3 of the *first* display have this information. The fields displayed on line 2 were specified in the PGM and INVLVL parameters, and the fields on line 3 were specified in the START, LEN, and OUTFMT parameters. The following are displayed on lines 2 and 3:

- The name of the program containing the variables to be displayed.
- The invocation level of the program from which the variables are to be displayed.
- The starting character position within each character variable to be displayed. If the variable is not a character variable, the START parameter has no effect on the display.
- The maximum number of characters to be displayed in each character variable.
- Whether the value is to be displayed in character or hexadecimal format.

The remaining lines on the display give the different attributes and the value of the program variable identified on line 4. Only the attributes that apply to the named variable are displayed; the lines that would be left blank are filled with the next attributes that do apply.

If two or more variables are being displayed, all of the attributes and the value for one variable are shown first, and the attributes of the next variable begin on the next line. All applicable lines are repeated for each variable specified on the DSPPGMVAR command.

B Beginning on line 4, the following attributes can be presented for each nonpointer program variable:

- The name of the program variable and, optionally, a subscript shown in parentheses. If the variable is an array and a subscript was specified on the PGMVAR parameter with the variable, the number or variable name representing the positional element in the array is displayed.
- The name of the program pointer, if the variable is based on a program pointer and if its name was given.
- The type of data being displayed in the value for the variable (binary, zoned, packed, or character).
- The length of the variable that was declared in the program.
- The dimension, if the variable is an array; that is, the number of elements in the array. The dimension is *not* shown if the variable was specified with a subscript in the DSPPGMVAR command.
- The actual value of the variable, shown on the last (and succeeding) lines related to that variable, depending on the variable's length and the values specified on the START, LEN, and OUTFMT parameters. In the case of long character variables, a maximum of 32 characters in the string can be displayed on each line. Each 32-character (or less) segment is enclosed in apostrophes on as many lines as necessary.

If a specified variable is used as a program pointer, the following display format is used to display its attributes:

```
XX/XX/XX  XX:XX:XX     PROGRAM VARIABLE DISPLAY
Program:     XXXXXXXXXX    Inv lvl:    XXXXXXXXXXX
Output start pos:  XXXXX  Length:    XXXXX  Format:  XXXXX
Variable:    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
             XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
             XXXXXXXXXXX
  Base:      XXXXXXXXXXXXXXXXXXXXXXXX
  Dimension:        XXXXX
C Pointer type:     XXX     XXXXXXXXXXX:    XXXXXXXXXXXXXXXX
    Object:         XXXXXXXXXX Obj type:     XXXXXXX
    Library:        XXXXXXXXXX
  Type:      XXXXXXXXX         Length:     XXXXX
  'XXXXXXXXXXXXXXXXXXXX...'
      .
      .
      .
```

The display of a program pointer includes all of the attributes presented on the basic display. In addition, the following pointer attributes are also displayed (shown at C), if they apply to the specified program pointer:

- The pointer type is given in the first field on line 7, and additional descriptive information is given in the second field. Depending on the pointer's type, the following information can appear in the two fields:

Type	Descriptive Information
SYP (system pointer)	AUTHORITY: XXXXXXXXXXXXXXXX The 16 bits indicate the types of authority that are valid for the object (named on line 8) pointed to.
SPP (space pointer)	OFFSET: XXXXXXXX The eight-digit hexadecimal number indicates the number of bytes into the space object (named on line 8) that the pointer points to.
IP (instruction pointer)	INSTRUCTION: XXXX The four-digit hexadecimal number of the MI instruction that the pointer points to.
DP (data pointer)	OFFSET: XXXXXXXX The eight-digit hexadecimal number indicates the number of bytes from the beginning of the data object (named on line 8) that the pointer points to.

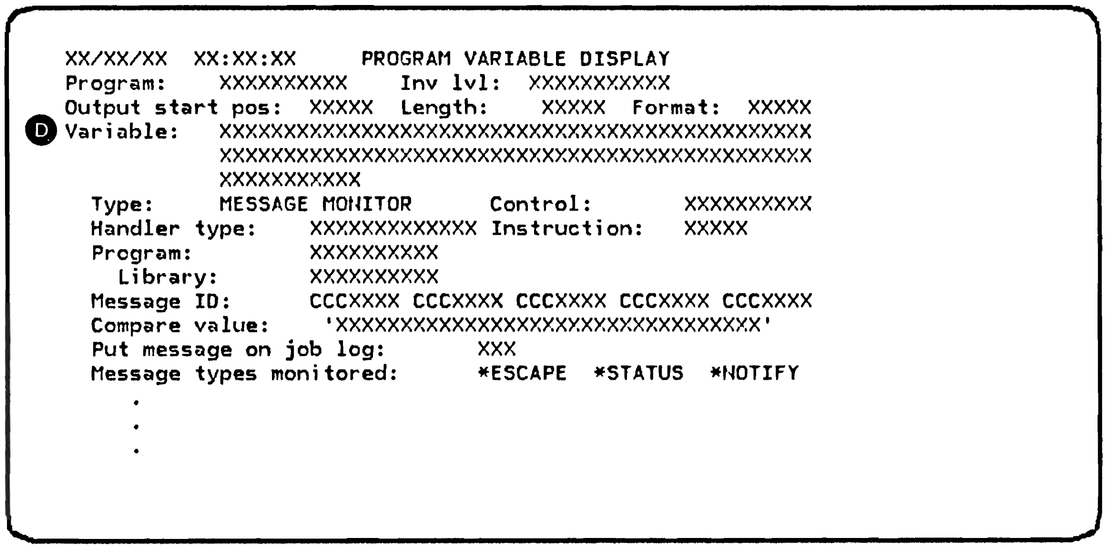
- The name of the object pointed to by SYP—or into by SPP, IP, or DP—and the object type of the object is given (on line 8).

- If applicable, the name of the library in which the object is located is given (on line 9).

- If the pointer is a data pointer (DP), the following is displayed:
  - The data type (character, binary, zoned, or packed) and the declared length of the data object to which it points (line 10)
  - The value in the data object (line 11), depending on the length of the data object and the values specified in the START, LEN, and OUTFMT parameters

Message Monitor Variables

A third display format is used if the specified variable contains the name of a message monitor or the MI ODV number of the monitor. When MI programs are being debugged, MI ODV numbers can be specified in the PGMVAR parameter to reference ODVs for message monitors. ODVs that monitor escape and notify messages are not variables in the normal sense, but they can be displayed by the DSPPGMVAR command. When the command specifies ODV numbers, the following display is presented.



```
  XX/XX/XX  XX:XX:XX     PROGRAM VARIABLE DISPLAY
  Program:    XXXXXXXXXX    Inv lvl:  XXXXXXXXXXX
  Output start pos:  XXXXX  Length:    XXXXX  Format:  XXXXX
D Variable:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              XXXXXXXXXXX
    Type:     MESSAGE MONITOR      Control:       XXXXXXXXXX
    Handler type:    XXXXXXXXXXXXX Instruction:   XXXXX
    Program:         XXXXXXXXXX
      Library:       XXXXXXXXXX
    Message ID:      CCCXXXX CCCXXXX CCCXXXX CCCXXXX CCCXXXX
    Compare value:    'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    Put message on job log:       XXX
    Message types monitored:      *ESCAPE  *STATUS  *NOTIFY
        .
        .
        .
```

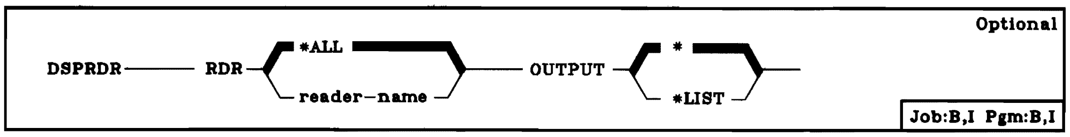
The message monitor information displayed for a variable identified at D includes the following:

- The variable name, which is either the MI ODV number or the name of the message monitor. If the variable is an array, and a subscript representing the positional element in the array is specified, the subscript is shown in parentheses.
- The type of variable being displayed, which (in this case) is a message monitor variable.
- The controlling action taken by the monitor after receiving a monitored message. Examples of the action that can be taken are handle, defer, or default.

- The type of handler invoked for the monitored message.
- If the handler is an internal handler or a branchpoint handler, the hexadecimal number of the instruction (to which control is passed) is displayed.
- If the handler is an external user program, and the name of the program is available at the time the DSPPGMVAR command is executed, the name of the program is given. If the name of the library is available, it is also displayed on the next line.
- The message identifiers of all the messages that are being monitored and that are to be handled by this monitor. Each 7-character identifier gives 3 characters identifying the user program or program product (such as CPF), followed by four digits identifying the message.
- The compare value for the message monitor.
- An indication (YES or NO) of whether the message is to be put to the job log.
- The type of message or the combination of types of messages to be monitored.

DSPRDR (Display Reader) Command

The Display Reader (DSPRDR) command displays or prints the attributes and status of the card, diskette, or data base readers that have been started in the system and that have not yet terminated. Detailed information about a specified reader can be displayed. Or, for all the readers, the name, type, source of input, and status of each one can be displayed. Also, other display or controlling functions can be initiated directly from either type of reader display.



RDR Parameter: Specifies the name of the card, diskette, or data base reader for which information is to be displayed, or specifies that the information for all card, diskette, or data base readers is to be displayed.

<u>*ALL:</u> The status of all card, diskette, or data base readers is to be displayed.

reader-name: Enter the name of the reader for which information is to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

```
DSPRDR
```

This command displays the names of all readers, their types, device files, and status.

```
DSPRDR RDR(DISKREAD)
```

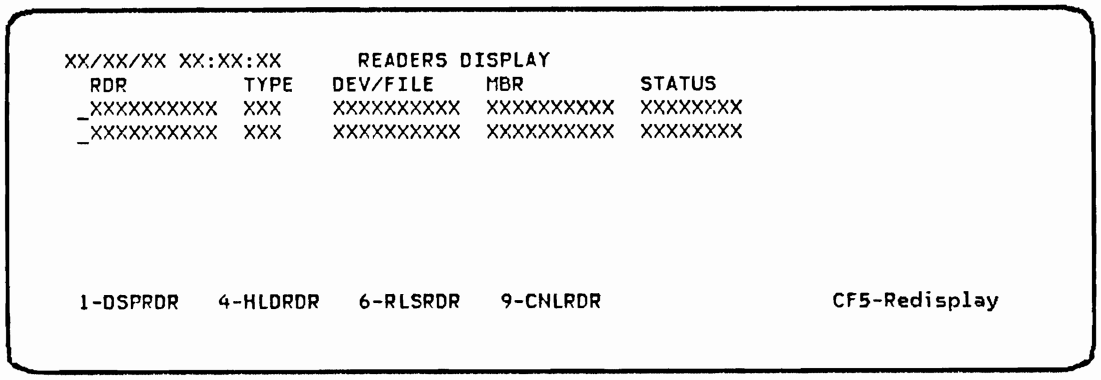
This command displays detailed information about the reader DISKREAD. This information is discussed in the following section.

Additional Considerations

One of two display formats is produced by the DSPRDR command, depending on whether *ALL or a spooling reader name is specified in the RDR parameter.

Displaying All Readers

If RDR(*ALL) is specified or assumed, the *readers display* is presented:



```
XX/XX/XX XX:XX:XX      READERS DISPLAY
  RDR         TYPE   DEV/FILE    MBR          STATUS
 _XXXXXXXXXX  XXX    XXXXXXXXXX  XXXXXXXXXX   XXXXXXXX
 _XXXXXXXXXX  XXX    XXXXXXXXXX  XXXXXXXXXX   XXXXXXXX






 1-DSPRDR   4-HLDRDR   6-RLSRDR   9-CNLRDR                CF5-Redisplay
```

The readers display shows, for each spooling reader in the system, some attributes of the reader (that were specified by its corresponding start reader command) and its current status. For each reader, a separate line is used to show:

- Reader name: The name of the spooling reader, specified on the command that started it.
- Reader type (CRD, DKT, or DB): The type of reader (card, diskette, or data base), determined by the command used to start it (STRCRDRDR, STRDKTRDR, or STRDBRDR).
- Device or file name: The name of the device or file being used or to be used by the reader to read the input stream. (The reader can be currently active or still on the job queue—the reader is a system job in itself, which runs in the QSPL subsystem.) The device name was specified in the DEV parameter of the start reader command.
- Member name: For data base readers only, the name of the member in the file from which the reader is to read the input stream. If the reader has not started, the name displayed is the name specified in the STRDBRDR command; after the reader has started, the name displayed is the actual member being processed.
- Status of the reader: The current status of the reader is indicated by one of the following values:
  - JOBQ. The reader job is still on the job queue; although the reader has been started by a start command, it has not yet begun executing.
  - JOBQ/HLD. The reader job is currently held on the job queue for which it was started.
  - ACT. The reader job is active and is currently reading an input stream from a device or a data base file. It is putting other jobs on their specified job queues by reading the spooled input for each job.
  - ACT/HLD. The reader has been active, but it is currently held because of a HLDRDR command; processing of the input stream has stopped.

Four options are shown at the bottom of the display that are available to the user. An option number can be entered, in the input field preceding the reader name, on one or more lines to cause the associated function to be performed on the reader identified on that line. Options 1, 4, 6, and 9 execute the DSPRDR, HLDRDR, RLSRDR, and CNLRDR commands, respectively.

For a description of the function keys that can be used from this display, refer to *Additional Considerations* in the DSPJOBQ command description.

The information displayed by option 1 is the same as that displayed if the DSPRDR command had been entered with the name of the reader specified in the RDR parameter. The information displayed for a single reader is described in the following section.

Displaying a Specific Reader

If the RDR parameter specifies the name of a reader, the display for that type of reader is shown. All of the attributes for the reader are displayed in one of three formats, depending on the reader type:

CRD	Card reader display
DKT	Diskette reader display
DB	Data base reader display

All of the attributes displayed on lines 2 through 7 are the same for all three types of readers. The title on line 1 indicates the type of reader being displayed. The *diskette reader display* is shown here as an example of all three reader types.

```
XX/XX/XX XX:XX:XX     DISKETTE READER DISPLAY
Reader name:     XXXXXXXXXX   User:  XXXXXXXXXX   Nbr:  XXXXXX
Started by:      XXXXXXXXXX
Status:          XXXXXXXXXXXXXXXXXXXX
Current job:     XXXXXXXXXX   User:  XXXXXXXXXX   Nbr:  XXXXXX
Message queue:   XXXXXXXXXX   Library:  XXXXXXXXXX
Default jobq:    XXXXXXXXXX   Library:  XXXXXXXXXX
Label:           XXXXXXXX     Current volume:   XXXXXX
Location:        XXXXX        Creation date:    XX/XX/XX
Code:            XXXXXXXX
CF3-Reader job display    CF5-Redisplay
```

Briefly, the information presented on the diskette reader display, card reader display, or data base reader display is as follows:

- Line 2 shows the fully qualified job name of the specified reader (which is a system job): reader name, user name, and job number.
- Line 3 shows the name of the user who started the reader.

- Line 4 displays one of the following for the reader's status:
  - JOBQ. The reader job is still on the job queue for which it was started; it has not begun executing.
  - JOBQ/HLD. The reader job is currently held on the job queue for which it was started.
  - ACT/READING. The reader is active and is reading the input stream.
  - ACT/HLD. The reader was executing and is currently held.
  - ACT/WAITING. The reader is active and is waiting for more input.

- Line 5 shows the fully qualified name of the current job being read and placed on the job queue.

- Line 6 shows the qualified name of the message queue used by this reader for operational messages.

- Line 7 shows the qualified name of the job queue specified on the start reader command, or the name of the default job queue on which jobs are placed if the job specifies JOBQ(*RDR).

- Lines 8 through 10 show information that is specifically related to the type of reader being displayed.

  Card Reader (not shown): The *hopper* field on line 8 shows the hopper number in which the card input should be placed.

  Data Base Reader (not shown): The *file name* and *library* fields on line 8 show the qualified name of the data base file containing the input stream. The *member name* field on line 9 shows the member name of this data base file.

  Diskette Reader: The *label* field on line 8 gives the data file identifier of the input stream on the diskette, and the *current volume* field gives the volume identifier of the current volume being processed. The *location* field on line 9 indicates the location of the diskettes containing the diskette data file being processed (the location was specified in the LOC parameter of the STRDKTRDR command). The *creation date* field gives the creation date that was specified on the start command. A value of 00/00/00 is shown if *NONE was specified (or assumed) on the start command and the file has not yet been opened. Once the file is opened, the actual creation date of the data file is shown. The *code* field on line 10 indicates whether the diskette contains EBCDIC or ASCII data.

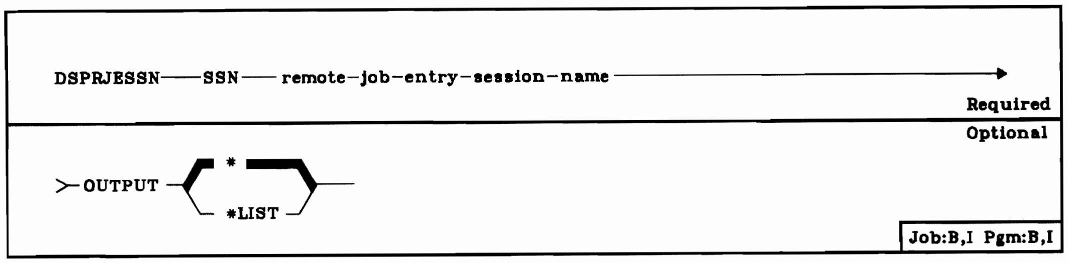
- Line 11 shows that the CF3 key can be used to display additional information about the current reader job's spooled files. CF3 invokes the function of the DSPJOB command and presents the DSPJOB menu, from which you can display any of the reader job attributes and spooled files. Refer to *Additional Considerations* in the DSPJOB command description. Line 11 also shows that the CF5 key can be used to display the most current reader information, if the display information has been updated since you last viewed the display.

DSPRJESSN (Display RJE Session) Command

The Display RJE Session (DSPRJESSN) command displays an active RJEF session status.

Restriction: To use this command, you must have operational rights to the session description and read rights to the library in which the session description is stored.

The Display RJE Session (DSPRJESSN) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSN Parameter: Specifies the name of the RJEF session to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer.

<u>*</u>: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer. The IBM-supplied device file QPRJESTS is used to print the RJEF session status.

Example

```
DSPRJESSN SSN(RJE) +
     OUTPUT(*)
```

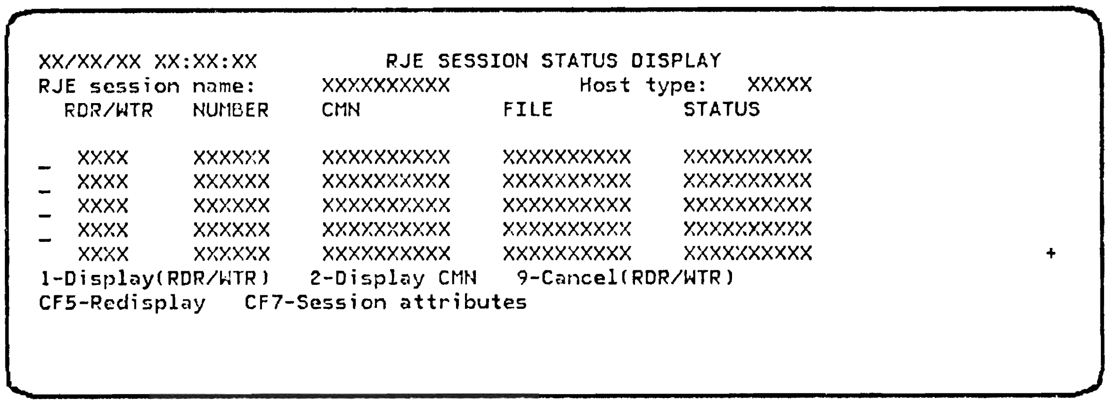
This command (if entered from a batch job) causes a list and detail information, of RJEF devices associated with the active RJEF session named RJE, to be sent to the job's spooling queue to be printed. If the command is entered from an interactive job, a menu of RJEF devices is displayed from which detail selections can be made.

Additional Considerations

The RJE session status display presents information about the activity of the host system device(s) within the requested RJE session.

The host system devices can be controlled to some extent from this display. The Cancel RJE Reader (CNLRJERDR) and the Cancel RJE Writer (CNLRJEWTR) command functions (with OPTION(*CNTRLD) specified) can be performed from this display by entering the appropriate value next to the device to be canceled.

The RJE session status display includes several displays. Entering the Display RJE Session (DSPRJESSN) command causes the following display to appear:



```
XX/XX/XX XX:XX:XX         RJE SESSION STATUS DISPLAY
RJE session name:     XXXXXXXXXX         Host type:   XXXXX
  RDR/WTR   NUMBER    CMN             FILE           STATUS

_  XXXX     XXXXXX    XXXXXXXXXX     XXXXXXXXXX     XXXXXXXXXX
_  XXXX     XXXXXX    XXXXXXXXXX     XXXXXXXXXX     XXXXXXXXXX
_  XXXX     XXXXXX    XXXXXXXXXX     XXXXXXXXXX     XXXXXXXXXX
_  XXXX     XXXXXX    XXXXXXXXXX     XXXXXXXXXX     XXXXXXXXXX
   XXXX     XXXXXX    XXXXXXXXXX     XXXXXXXXXX     XXXXXXXXXX              +
1-Display(RDR/WTR)   2-Display CMN   9-Cancel(RDR/WTR)
CF5-Redisplay   CF7-Session attributes
```

Beginning on line 4, the following session information is displayed:

- RDR/WTR. Identifies the RJEF console, reader, or writer name (for example, CSLO, CSLI, RD1, PR1, or PU1) as specified in the Add RJE Reader Entry (ADDRJERDRE) or Add RJE Writer Entry (ADDRJEWTRE) command. Only those readers and writers defined in the session description are displayed.

- NUMBER. Identifies the job number component of the System/38 fully qualified job name for the RJEF job that is performing the console, reader, or writer function. If the function is not active, the number is blank. The job number of immediate RJEF readers is shown as ******. The fully qualified job name (for example, RD1.QRJE.002538) contains:
  - The RJE reader or writer function identification on this display (for example, RD1)
  - For writer and console jobs, the RJEF profile name (for example, QRJE)
  - For reader jobs, the user identifier of whoever submitted the SBMRJEJOB command
  - The job number (for example, 002538)

- CMN. Identifies the RJEF communications file for which the function identified by RDR/WTR is currently active.

- FILE. Identifies the file being used by the function identified by RDR/WTR. If a reader function is identified, then the file name is the name of the data base file currently being sent to the host system. If a writer function is identified, then the file name is the name of the file being written to. If the function is not currently active or a user program has been called without calling the RJEF writer (QMRSWTR), then FILE is blank.
- STATUS. Indicates the current activity of the function identified by RDR/WTR. The following are the possible STATUS values for this display:

CANCELED	Device was defined in the session description, but is not currently started. The device may have been canceled or never started since the session was started.
STARTED	Device is not currently processing data, but is prepared to process data when it arrives.
ACTIVE	Device is currently processing data.

If a 1 is entered for one or more selections on the RJE session status display, one or more detail status displays are shown consecutively in the order of the entries made on the RJE session status display.

If a 1 is entered for a console entry (CSLO or CSLI) or for an inactive reader or writer entry, a message is issued indicating that there is no additional information for this entry.

If a 2 is entered, the associated RJE communications entry display is shown.

If a 9 is entered for one or more selections, those selections with functions identified under RDR/WTR will have an RJEF-controlled cancel issued against them.

If a 9 is entered for a selection having the console (that is, CSLO or CSLI) identified under RDR/WTR, an error message is displayed indicating that a cancel of a console function is invalid.

Pressing the Enter key starts the sequence of displays (and cancel requests), which ends with a return to the RJE session status display.

Pressing the CF2 key on any display causes a return to the previous display. If the CF2 key is pressed from the RJE session status display, the DSPRJESSN command is terminated.

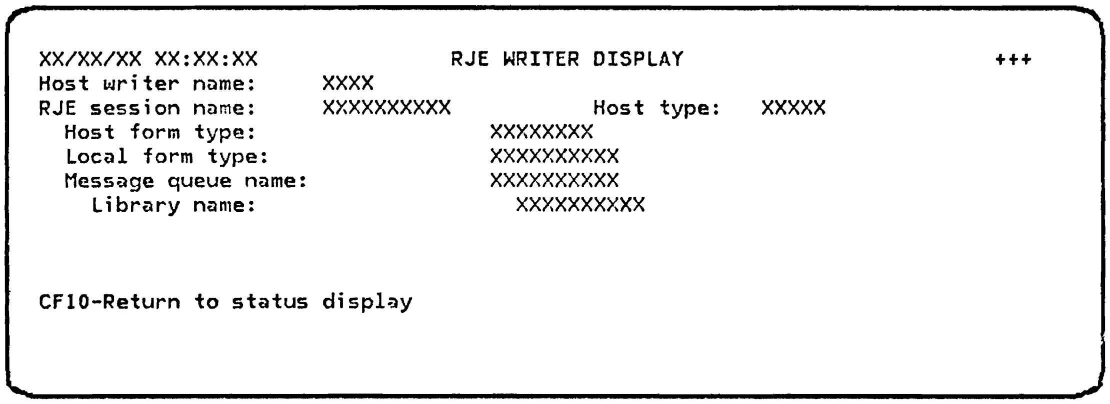
Pressing the CF5 key causes the current display to be reshown with the most current data available.

Pressing the CF7 key causes common session attributes to be displayed.

RJE Writer Displays

The writer displays appear when you enter a 1 for one or more selections on the RJE session status display if the selected RDR/WTR represents a print or punch function. These displays are:

- RJE writer display
- User program RJE writer display
- RJE data base writer display or two RJE print writer displays

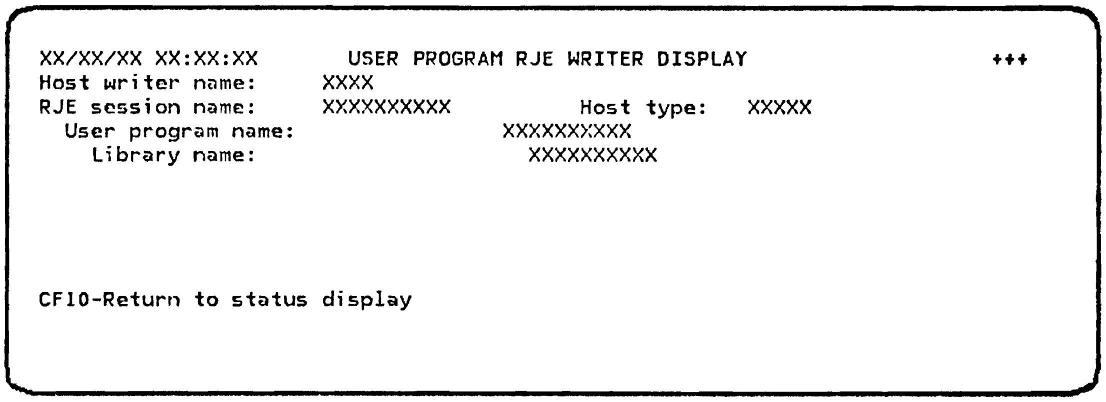


```
XX/XX/XX XX:XX:XX              RJE WRITER DISPLAY                    +++
Host writer name:      XXXX
RJE session name:      XXXXXXXXXX          Host type:   XXXXX
  Host form type:                   XXXXXXXX
  Local form type:                  XXXXXXXXXX
  Message queue name:               XXXXXXXXXX
    Library name:                     XXXXXXXXXX


CF10-Return to status display
```

The RJE writer display appears for each selected writer function. Pressing the Enter key causes one of the following displays to be shown:

- User program RJE writer display
- RJE data base writer display
- RJE print writer display

User Program RJE Writer Display

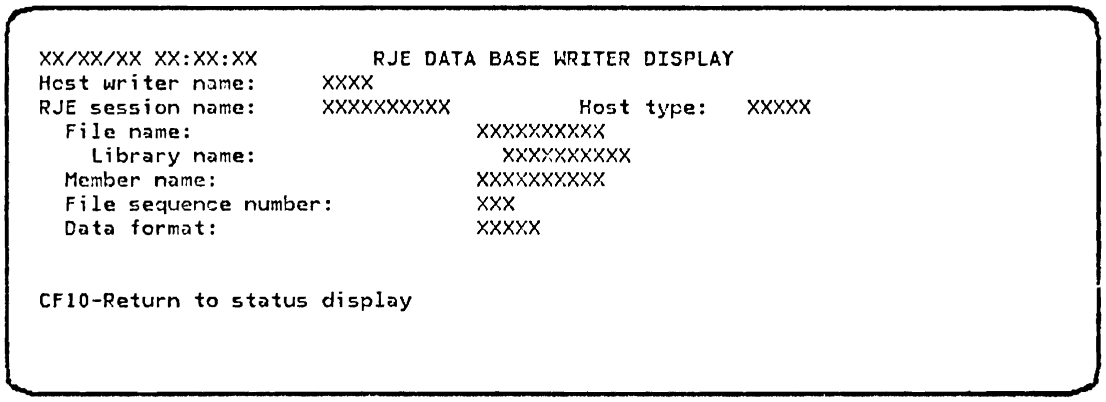


```
XX/XX/XX XX:XX:XX       USER PROGRAM RJE WRITER DISPLAY                    +++
Host writer name:     XXXX
RJE session name:     XXXXXXXXXX          Host type:    XXXXX
  User program name:                XXXXXXXXXX
    Library name:                     XXXXXXXXXX





CF10-Return to status display
```

The user program RJE writer display is shown only if the selected writer function called a user program.

If the user program called the RJEF writer program (QMRSWTR), pressing the Enter key causes either the RJE data base writer display or the RJE print writer display to be shown. Otherwise the next selected RDR/WTR is shown or, if all requests have been serviced, the RJE session status display is shown again.

RJE Data Base Writer Display

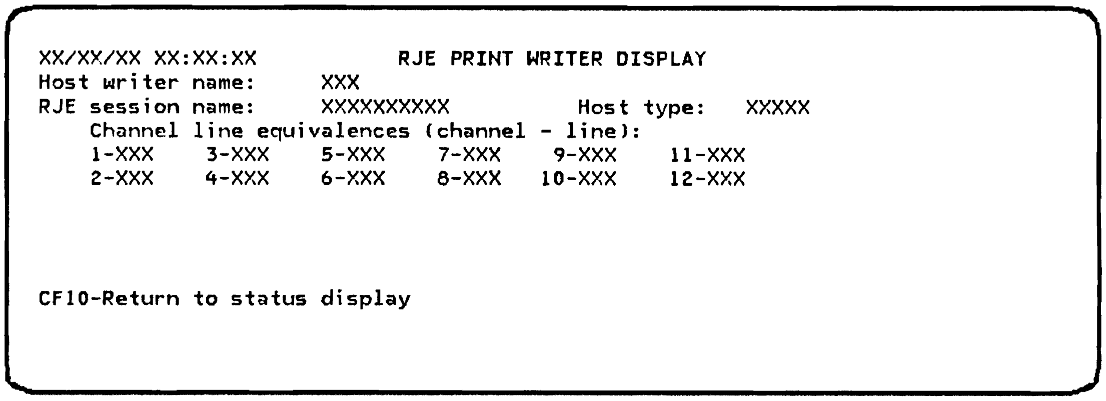


```
XX/XX/XX XX:XX:XX         RJE DATA BASE WRITER DISPLAY
Host writer name:     XXXX
RJE session name:     XXXXXXXXXX          Host type:    XXXXX
  File name:                      XXXXXXXXXX
    Library name:                   XXXXXXXXXX
  Member name:                    XXXXXXXXXX
  File sequence number:           XXX
  Data format:                    XXXXX

CF10-Return to status display
```

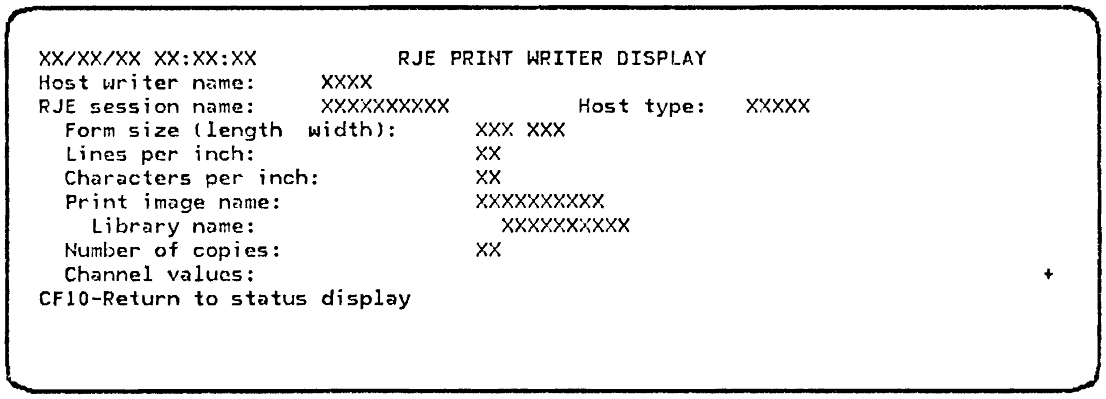
The RJE data base writer display is shown only if the selected writer is writing to a data base file.

Pressing the Enter key causes the next selected RDR/WTR to be shown or, if all requests have been serviced, a return to the RJE session status display.

RJE Print Writer Displays



```
XX/XX/XX XX:XX:XX          RJE PRINT WRITER DISPLAY
Host writer name:     XXX
RJE session name:     XXXXXXXXXX          Host type:   XXXXX
    Channel line equivalences (channel - line):
    1-XXX    3-XXX    5-XXX    7-XXX    9-XXX    11-XXX
    2-XXX    4-XXX    6-XXX    8-XXX   10-XXX    12-XXX




CF10-Return to status display
```

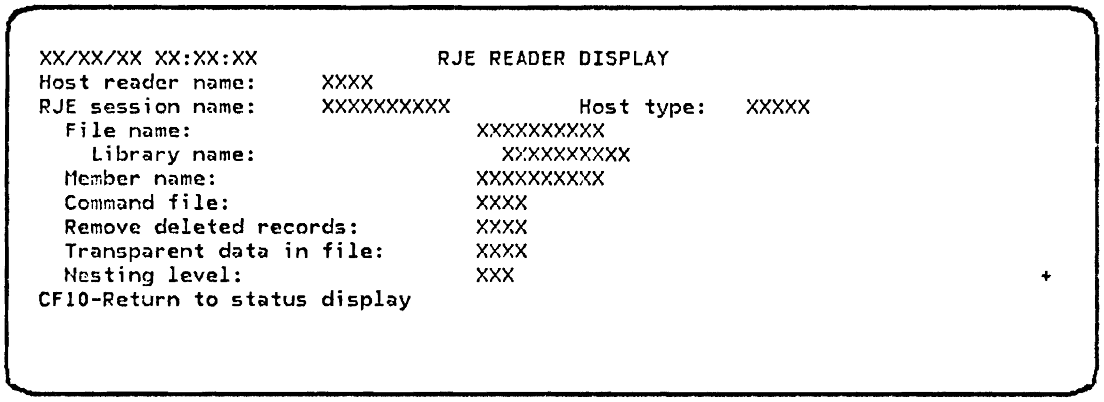


```
XX/XX/XX XX:XX:XX          RJE PRINT WRITER DISPLAY
Host writer name:     XXXX
RJE session name:     XXXXXXXXXX          Host type:   XXXXX
  Form size (length  width):      XXX XXX
  Lines per inch:                 XX
  Characters per inch:            XX
  Print image name:               XXXXXXXXXX
    Library name:                   XXXXXXXXXX
  Number of copies:               XX
  Channel values:                                                 +
CF10-Return to status display
```

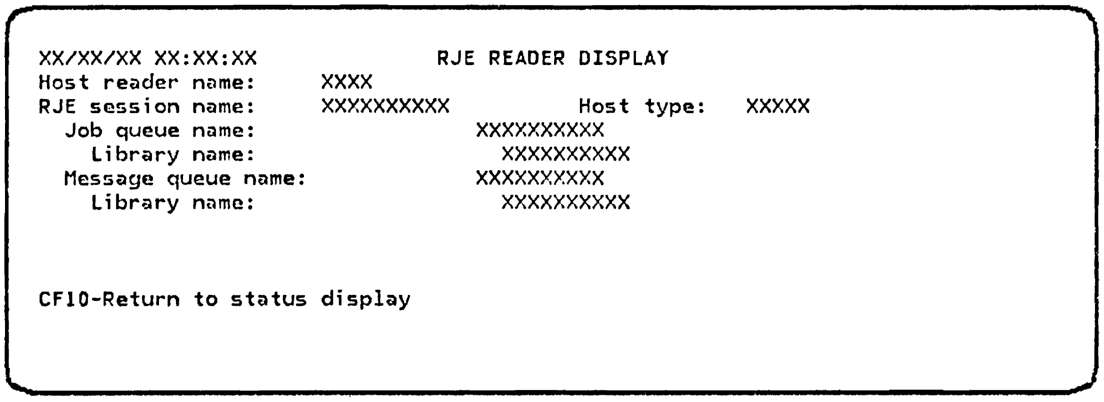
The RJE print writer display is shown only if the selected writer is writing to a printer file.

Pressing the Enter key causes the next selected RDR/WTR to be shown or, if all requests have been serviced, a return to the RJE session status display.

RJE Reader Displays



```
XX/XX/XX XX:XX:XX              RJE READER DISPLAY
Host reader name:     XXXX
RJE session name:     XXXXXXXXXX          Host type:   XXXXX
  File name:                        XXXXXXXXXX
    Library name:                     XXXXXXXXXX
  Member name:                      XXXXXXXXXX
  Command file:                     XXXX
  Remove deleted records:           XXXX
  Transparent data in file:         XXXX
  Nesting level:                    XXX                           +
CF10-Return to status display
```

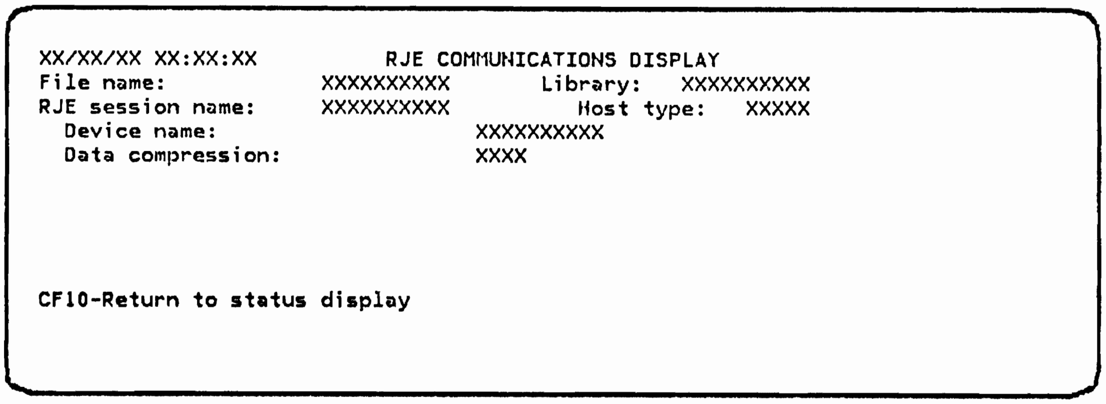


```
XX/XX/XX XX:XX:XX              RJE READER DISPLAY
Host reader name:     XXXX
RJE session name:     XXXXXXXXXX          Host type:   XXXXX
  Job queue name:                   XXXXXXXXXX
    Library name:                     XXXXXXXXXX
  Message queue name:               XXXXXXXXXX
    Library name:                     XXXXXXXXXX



CF10-Return to status display
```

The RJE reader display is shown only if the selected RDR/WTR is a reader function. The first file name displayed is of the highest nested level (that is, the current file being read at the time this display was requested).

Pressing the Enter key causes the next selected RDR/WTR to be shown or, if all requests have been serviced, the RJE session status display is shown again.

RJE Communications Display

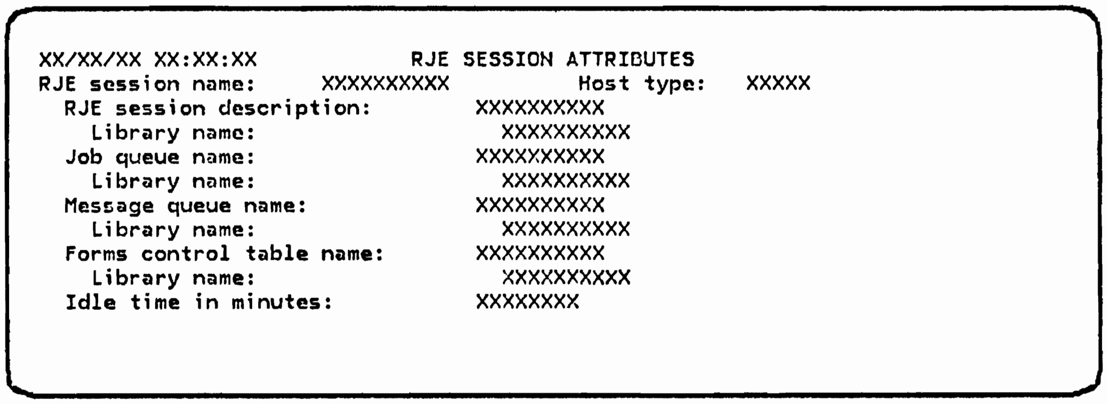


```
XX/XX/XX XX:XX:XX          RJE COMMUNICATIONS DISPLAY
File name:              XXXXXXXXXX       Library:   XXXXXXXXXX
RJE session name:       XXXXXXXXXX         Host type:   XXXXX
  Device name:                XXXXXXXXXX
  Data compression:           XXXX




CF10-Return to status display
```

The RJE communications display is shown as a result of a 2 specified and there is a communications file associated with the request.

Pressing the Enter key or the CF10 key causes the RJE status display to be shown again.

RJE Session Attributes Display



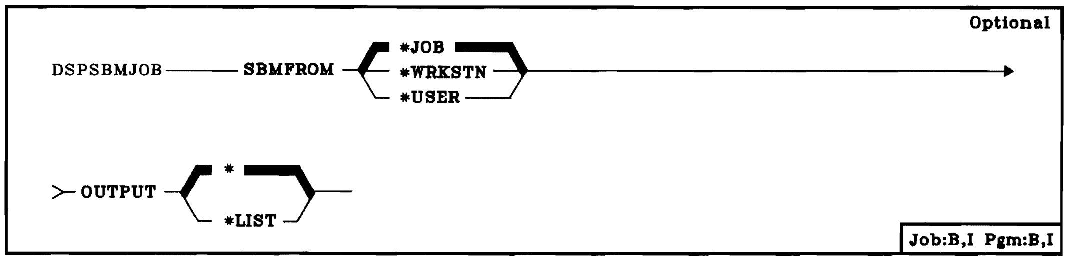
```
XX/XX/XX XX:XX:XX            RJE SESSION ATTRIBUTES
RJE session name:       XXXXXXXXXX         Host type:   XXXXX
  RJE session description:    XXXXXXXXXX
    Library name:               XXXXXXXXXX
  Job queue name:             XXXXXXXXXX
    Library name:               XXXXXXXXXX
  Message queue name:         XXXXXXXXXX
    Library name:               XXXXXXXXXX
  Forms control table name:   XXXXXXXXXX
    Library name:               XXXXXXXXXX
  Idle time in minutes:       XXXXXXXX
```

The RJE session attributes display is shown as a result of pressing the CF7 key.

Pressing the Enter key or the CF10 key causes the RJE session status display to be shown again.

DSPSBMJOB (Display Submitted Jobs) Command

The Display Submitted Jobs (DSPSBMJOB) command displays or prints the status of all jobs submitted at a work station, in a job, or under a user profile. (Jobs submitted with DSPSBMJOB(*NO) specified on the SBMJOB, SBMCRDJOB, SBMDBJOB, or SBMDKTJOB commands are not displayed by this command.)



SBMFROM Parameter: Specifies the type of submitted jobs that are to be displayed. Any jobs of that type that were submitted with the parameter DSPSBMJOB(*NO) specified on the SBMJOB, SBMCRDJOB, SBMDBJOB, or SBMDKTJOB commands are not included.

JOB: The jobs displayed are those that were submitted from the same job in which this command is entered.

**WRKSTN:* The jobs displayed are those that were submitted from the same work station at which this command is entered.

**USER:* The jobs displayed are those that were submitted from a job having the same user profile as the job in which this command is entered.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPSBMJOB  SBMFROM(*USER)
```

This command displays a list of jobs that were submitted by a job running under the same user profile as the job in which this command is executed.

Additional Considerations

The following display format results from the DSPSBMJOB command:

```
XX/XX/XX  XX:XX:XX       SUBMITTED JOBS - XXXXXXXXXX      XXX
  JOB NAME   USER        NBR    TYPE    STATUS
 _XXXXXXXXXX XXXXXXXXXX XXXXXX XXXXXXX XXXXXXX XXXXXXX
 _XXXXXXXXXX XXXXXXXXXX XXXXXX XXXXXXX XXXXXXX XXXXXXX
    .
    .
                                 .



1-DSPJOB   2-Spl files   4-HLDJOB   6-RLSJOB   9-CNLJOB   CF5-Redisplay
```

The first line shows the current job date and time; and, after the display title, the type (specified by the SBMFROM parameter) of submitted jobs being displayed is shown.

For each submitted job, a separate line is used to show:

- Qualified job name, shown in the first three fields (job name, user name, and job number).
- Type of job (BATCH).
- Status of submitted job (in two fields):
  - The first field indicates whether the job has been started (ACTIVE), is currently on an output queue (OUTQ) or on a job queue (JOBQ, or TFRJOB, if the submitted job was transferred to another job), has been suspended by the system request key (SYSREQ), or has finished (FIN).
  - If a job is terminating because a CNLJOB (Cancel Job) *IMMED or a TRMSBS (Terminate Subsystem) *IMMED has been specified, or, if CNLJOB(*CNTRLD) or TRMSBS(*CNTRLD) has been specified and delay time has expired, the first status field will indicate CANCEL.
  - The second field indicates whether a job has been held (HELD) or not held (the field is blank).
  - If the system failed while a job was active, the status JOBLOG PENDING is shown in the two fields until the job log is written.
  - If a job is terminating because a CNLJOB (Cancel Job) *IMMED or a TRMSBS (Terminate Subsystem) *IMMED has been specified, the status field will indicate CANCEL.

Six options are shown at the bottom of the display that are available to the user. An option number can be entered (in the input field preceding the job name) on one or more lines to cause the associated function to be performed. Options 1, 4, 6, and 9 execute the DSPJOB, HLDJOB, RLSJOB, and CNLJOB commands, respectively. Option 2 displays the spooled files that exist for the job (as if option 7 were selected on the programmer menu, and then option 2 were selected on the submitted jobs display).

For additional information about these options, refer to *Additional Considerations* in the DSPSBS command description. The DSPSBS command presents a display that has the same job attributes and options.

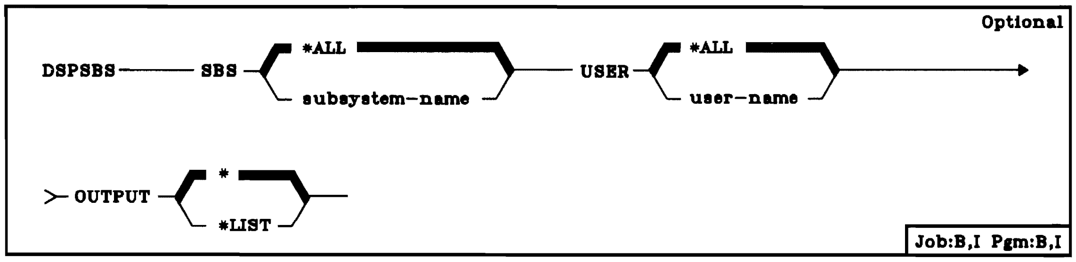
You can use the command function keys as follows:

CF1	Ends the Submitted Jobs Display and returns control to the working display, such as the command entry display, programmer menu, operator menu, and so forth.
CF5	Reshows the display with updated information.

DSPSBS (Display Subsystem) Command

The Display Subsystem (DSPSBS) command displays the qualified job names and the status of all jobs being processed by all subsystems in the system, or by a specified subsystem. If a user name is specified, only those jobs belonging to the specified user are displayed. Also, if one of the user jobs (not a system job) shown on the subsystem display is selected, additional information about that job can be displayed.

The DSPSBS command displays information about *active jobs* in one or all subsystems and, if all subsystems are being displayed, the jobs on job queues and output queues.



SBS Parameter: Specifies the name of the subsystem (or all subsystems) for which the qualified job name and job status of each job currently active in the subsystem are to be displayed.

*ALL: All jobs in all subsystems are to have their job information displayed. In this case, jobs that are on job queues and on output queues are also to be displayed.

subsystem-name: Enter the name of the subsystem. All active jobs in this subsystem are displayed.

USER Parameter: Specifies the name of the user whose jobs are to be displayed.

*ALL: All jobs being processed under all user names are to be displayed.

user-name: Enter a user name. All jobs that are qualified with this user name are displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

```
DSPSBS
```

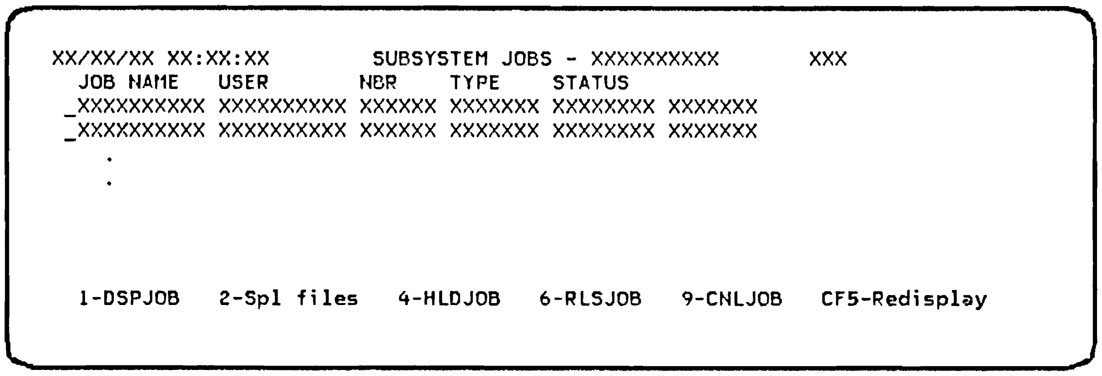
This command, entered from a work station, displays the qualified job names and the status of all jobs in all subsystems, and the jobs on the job queues and output queues. A user can obtain additional information about one of his jobs by entering a 1 before the job's name.

```
DSPSBS SBS(QBATCH) USER(JONES)
```

This command displays all of the jobs in the QBATCH subsystem that belong to the user profile of the user named JONES.

Additional Considerations

The DSPSBS command produces the following *subsystem jobs display* to show the jobs being processed by an active subsystem:



```
XX/XX/XX XX:XX:XX        SUBSYSTEM JOBS - XXXXXXXXXX       XXX
  JOB NAME   USER       NBR    TYPE    STATUS
 _XXXXXXXXXX XXXXXXXXXX XXXXXX XXXXXXX XXXXXXXX XXXXXXX
 _XXXXXXXXXX XXXXXXXXXX XXXXXX XXXXXXX XXXXXXXX XXXXXXX
    .
    .



  1-DSPJOB   2-Spl files   4-HLDJOB   6-RLSJOB   9-CNLJOB   CF5-Redisplay
```

Each active subsystem for which you want job information has its own display. If *ALL was specified or assumed for both the SBS and USER parameters on the DSPSBS command, the Enter key is used to advance from one subsystem display to another. Three plus signs (+++) appear in the upper right corner if the current subsystem is not the last one to be displayed. If a specific subsystem name (rather than *ALL) was specified for the SBS parameter, a single display is presented.

The first line of the subsystem jobs display shows the current job date and time, and, after the display title, shows the name of the active subsystem for which job information is being displayed. Then, for each job shown on the display, the following information is given:

- Job name, user name, and job number: These three attributes are derived from the qualified job name, which is also the name that was specified, or used, when the job was entered on the system. (For additional information about the parts of a qualified job name and where each part comes from, refer to the expanded description of the JOB parameter in Appendix A.)

  The jobs are displayed in order by job number.

- Type of job (INTER, BATCH, AUTO, RDR, or WTR).

- Status of job (in two fields):
  - The first field indicates whether the job has been started (ACTIVE), is currently on an output queue (OUTQ) or on a job queue (JOBQ, or TFRJOB, if the job was transferred to another job), has been suspended by the system request key (SYSREQ), or has finished (FIN).
  - The second field indicates whether the job is being held (HELD) or not held (the field is blank).
  - If the system failed while the job was active, the status JOBLOG PENDING is shown in the two fields until the job log is written.
  - If a job is terminating because a CNLJOB (Cancel Job) *IMMED or a TRMSBS (Terminate Subsystem) *IMMED has been specified, or if CNLJOB(*CNTRLD) or TRMSBS(*CNTRLD) is specified and delay time has expired, the first status field will indicate CANCEL.

If DSPSBS(*ALL) is specified or assumed, those jobs that exist only on a job or output queue are shown after jobs that exist in all active subsystems. The jobs that are only on a job queue are shown first on a separate display; then jobs that are only on an output queue are shown next on another display. (The Enter key is used to display each one.) On these two displays, the subsystem name field on the display title line is blank.

If more jobs exist in a single subsystem than will fit on one display, a single plus sign (+) appears to the right of the last job displayed. Use the Roll keys to view the additional jobs.

An input field (to the left of each job name) can be used to enter any one of the numbers shown at the bottom of the display. When the Enter key is pressed, the function associated with the entered number (a command) is performed for that job. If numbers are placed in the input fields preceding several jobs before the Enter key is pressed, the specified functions are performed (one at a time) on the jobs in the order in which the jobs are shown on the display. The system executes each command using the default values of all its parameters. The following functions can be specified:

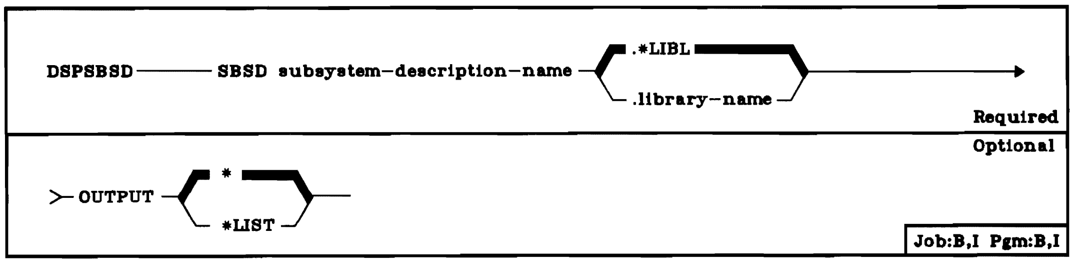
1–DSPJOB:	The display job menu is presented from which several displays can be selected to show the job's definition and execution attributes, the job's status, and the job's spooled input or output files. (Refer to the DSPJOB command description for an explanation of the job displays.)
2–Spl files:	The job's spooled input or output files are displayed if the job is on its job queue or it has any spooled output files. The spooled file displays (for jobs on job queues or output queues) are both described at the end of *Additional Considerations* in the DSPJOB command description.
4–HLDJOB:	The job is held, but its spooled files are not held.
6–RLSJOB:	The job, which must be in the held state, is released. The job could be held because a HLDJOB command was entered for the job, or because it was held when a 4 was specified on a display such as this one.
9–CNLJOB:	The job is canceled, but the spooled files produced by the job are not canceled. A controlled cancel is performed as if the CNLJOB command were entered with all the default parameter values assumed.

- When all of the commands have been executed, the subsystem jobs display is reshown with the status fields of the jobs updated; also shown at the bottom of the display are any error or completion messages that occurred when the commands were executed.
- If there are more jobs than can be shown on a single display, the Roll Up key can be used to display the rest of them. Numbers can be placed in the input fields on multiple displays before the Enter key is pressed.
- After the commands have been executed, if there are more messages than can fit on that display, a + is shown at the end of the last (or only) message displayed. To view additional messages, position the cursor anywhere on a message line and press the Roll Up key. If no input values are specified in the input fields and the Enter key is pressed, it causes an exit back to the display from which the current display was requested. You must position the cursor at the first message and use the Roll Up key to view all of the messages.
- The CF1 key can be used to exit from the display shown above, or to exit from a display presented as a result of executing the commands entered on the display shown above. (In both cases, the exit causes a return to the basic working display or menu from which the last primary request was entered.) If, for example, you are viewing a display that resulted from a 1 being entered to execute the DSPJOB command, the CF1 key causes all of the functions indicated for the jobs that followed the job currently being displayed to *not* be executed. That is, all jobs further down in the initial job list do not have any of their requested functions performed.
- The CF2 key can be used to back up to a previous display, or if you are viewing the first in the sequence of displays, to return to the caller.
- The CF5 key can be used to ignore (cancel) any functions not yet executed that were entered in the input fields, and to reshow the subsystem jobs display with the updated status of all the jobs.

DSPSBSD (Display Subsystem Description) Command

The Display Subsystem Description (DSPSBSD) command displays the information contained in a subsystem description. The types of information (shown on separate displays) include: operational attributes, pool descriptions, autostart job entries, work station entries (by name and type), job queue entries, and routing entries. If the DSPSBSD command is entered in a batch job, the information for *all* the types are printed with the job's spooled output.

Restriction: You must have operational and read rights for the subsystem description before you can display its contents.



SBSD Parameter: Specifies the qualified name of the subsystem description that is to be displayed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPSBSD  SBSD(ORDER.LIB6) OUTPUT(*)
```

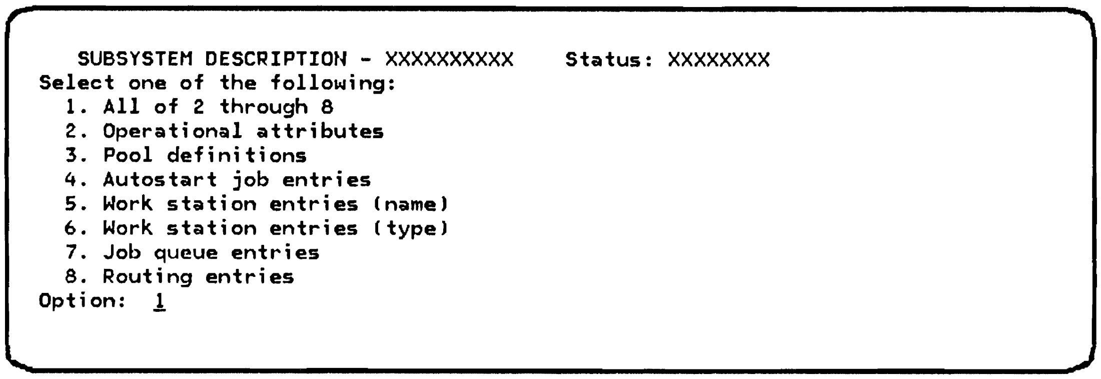
This command (if entered from a batch job) causes a complete set of display information about the subsystem description named ORDER (stored in LIB6 library) to be sent to the job's spooling queue to be printed. If the command is entered in an interactive job, the subsystem description menu is displayed, from which an option can be chosen. The information includes the subsystem's attributes, all of the job entries, and all of the routing entries currently in the subsystem description.

Additional Considerations

When the DSPSBSD command is entered interactively, the *subsystem description menu* is always presented first. Depending on the options selected from the menu, as many as 10 different displays can be presented. There is one summary display for each of the seven types of information and three detailed displays. A detailed display about one of the entries shown on a summary display for options 5, 6, or 8 can be selected from that summary display. (If the command is part of a batch job, *all* of the information about the specified subsystem description is always printed, including the details of every entry shown for options 5, 6, and 8.)

Each of the displays shows (on the first line) the name of the subsystem description and whether that subsystem is active or inactive. The second line of all the displays (except the menu) shows the current job date and time, and the name of the option being displayed.

If the information is to be displayed at a work station, the *subsystem description menu* is presented when the DSPSBSD command is entered.



```
   SUBSYSTEM DESCRIPTION - XXXXXXXXXX    Status: XXXXXXXX
Select one of the following:
  1. All of 2 through 8
  2. Operational attributes
  3. Pool definitions
  4. Autostart job entries
  5. Work station entries (name)
  6. Work station entries (type)
  7. Job queue entries
  8. Routing entries
Option: 1
```

To select one of the eight options from the menu, enter the associated option number (1 through 8) in the *Option* field and press the Enter key. (A blank option field causes an exit from the DSPSBSD function.) If option 1 is selected, all of the displays identified by options 2 through 8 are displayed in the order listed. Briefly, the information displayed is:

2. Operational attributes: Indicates the maximum number of jobs allowed in the subsystem.

3. Pool definitions: Describes all of the storage pools defined in the subsystem description.

4. Autostart job entries: Gives the names of all autostart jobs and their associated job descriptions.

5. Work station entries (by work station name): Lists all the names of work stations that are identified by name in the subsystem description. (A detailed description of one or more of those listed on the display can also be displayed upon request.)

6. Work station entries (by work station type): Lists all the types of work stations that are identified by type in the subsystem description. (A detailed description of one or more of those listed on the display can also be displayed upon request.)

7. Job queue entries: Gives the name of the job queues defined for the subsystem, and the maximum number of jobs that can be concurrently active through them, and the sequence number associated with each.

8. Routing entries: Describes all the routing entries in the subsystem description. (A detailed description of one or more of those listed on the display can also be displayed upon request.)

The Enter key is used to advance from one display to another. If an option (such as option 5) is selected, two display formats are used to present all of the information unless you press the CF10 key to return to the subsystem description menu. If option 1 is selected, you can also use the CF10 key to return to the menu instead of continuing through all of the displays.

Option 2. If option 2 is selected from the subsystem description menu, the *operational attributes* of the subsystem are displayed (not shown here):

- The name of the library containing the subsystem description.
- The maximum number of *all* jobs allowed in the subsystem at any given time.

Option 3. If option 3 is selected, the following are displayed (not shown here) for each *storage pool* definition in the subsystem description:

- The subsystem-related pool identifier.
- The amount of storage (in K-bytes) requested for the storage pool.
- The storage pool activity level.

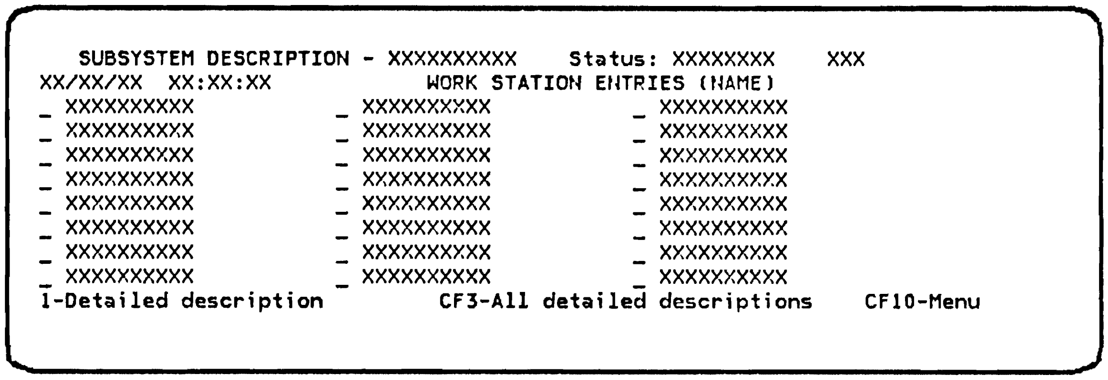
The pool identifiers shown on this display are not the ones shown on the DSPSYS and DSPSYSSTS displays. This display shows the *subsystem*-related pool identifiers that are specified in the subsystem description. The DSPSYS display shows as many as five of the *system*-related pool identifiers of the system storage pools that were obtained for each subsystem that is currently active. The DSPSYSSTS display shows, of the 16 possible *system* storage pools, only those that are active (that is, they contain storage). The only correlation of the pool identifiers is that, beginning with the first available system storage pool, the system assigns the required number (see note) of system pools to the subsystem when it is started and assigns the subsystem pools to the system pools in the order in which the subsystem pool identifiers were specified on the CRTSBSD command or the last CHGSBSD command.

Note: If 16 system pools are already active or if the system pools already active have allocated all the available storage, the subsystem storage pool is assigned to the base storage pool (*BASE).

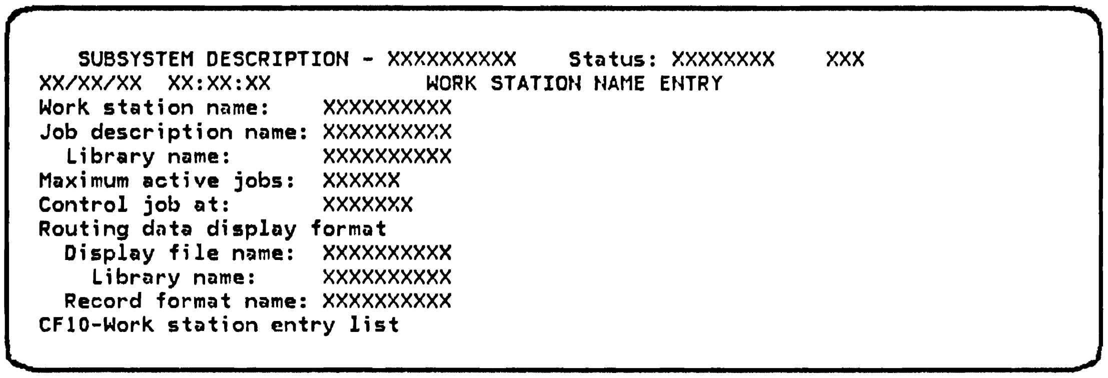
Option 4. If option 4 is selected, the following are displayed (not shown here) for each *autostart job entry* in the subsystem description:

- The name of the autostart job.
- The name of the job description (and its library name) that contains the job attributes given to the autostart job.

Option 5. If option 5 is selected, the first display of the *work station entry (name)* option is presented:


```
   SUBSYSTEM DESCRIPTION - XXXXXXXXXX    Status: XXXXXXXX    XXX
XX/XX/XX  XX:XX:XX           WORK STATION ENTRIES (NAME)
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
_ XXXXXXXXXX            _ XXXXXXXXXX            _ XXXXXXXXXX
1-Detailed description          CF3-All detailed descriptions    CF10-Menu
```

The name of every work station that has a work station name entry in the subsystem description is listed on this display. Each name is the name of the device description describing that work station. If a 1 is entered preceding one or more of the device names shown, a different display presents a detailed description of each entry selected. Each time the Enter key is pressed, the detailed description of the next entry is presented in the following format:


```
   SUBSYSTEM DESCRIPTION - XXXXXXXXXX    Status: XXXXXXXX    XXX
XX/XX/XX  XX:XX:XX           WORK STATION NAME ENTRY
Work station name:     XXXXXXXXXX
Job description name:  XXXXXXXXXX
  Library name:        XXXXXXXXXX
Maximum active jobs:   XXXXXX
Control job at:        XXXXXXX
Routing data display format
  Display file name:   XXXXXXXXXX
    Library name:      XXXXXXXXXX
  Record format name:  XXXXXXXXXX
CF10-Work station entry list
```

Refer to the associated parameter descriptions in the ADDWSE command description for an explanation of each attribute shown. For example, refer to the MAXACT parameter for the description of the maximum active jobs attribute.

If you are displaying multiple entries, you can press the CF10 key once to return to the work station entries summary display, where you can specify additional entries to be displayed in detail. If, while displaying multiple detailed entries, you want to return to a detailed entry display already shown, the CF2 key can be used; each time CF2 is pressed, the next previous display is shown. (If you are at the first detailed display, CF2 returns you to the summary display.)

If you press the CF3 key instead of entering a 1, all of the work station entries can be displayed in detail (one at a time) unless you press the CF10 key to return to the work station entries (name) display. Or, you can press CF10 *twice* to return to the subsystem description menu at any time.

Option 6. If option 6 is selected, the first display of the *work station entry (type)* option is displayed (not shown here). Each different work station type that was specified on the WRKSTNTYPE parameter of the ADDWSE command to create an entry in this subsystem description is listed on this display (similar to the first display shown for option 5).

If a 1 is entered preceding one of the device types and the Enter key is pressed, the same detailed description (except that it is for a work station *type*) is presented that was described for the second display of option 5. The CF3 key can be used to display (one at a time) all of the work station entries in detail.

Option 7. If option 7 is selected, the following are displayed (not shown here) for each *job queue entry* in the subsystem description:

- The job queue entry sequence number.
- The name of the job queue from which jobs are to be taken for each entry, and the name of the library in which the job queue is stored.
- The maximum number of jobs that can be concurrently active through each entry.

Option 8. If option 8 is selected, the first display of the *routing entry* option is presented:

```
   SUBSYSTEM DESCRIPTION - XXXXXXXXXX    Status: XXXXXXXX
XX/XX/XX  XX:XX:XX            ROUTING ENTRIES
SEQUENCE                        START
NUMBER  PROGRAM     LIBRARY     POS COMPARE VALUE
_ XXXX  XXXXXXXXXX  XXXXXXXXXX  XX  XXXXXXXXXXXXXXXXXXXX XXX
_ XXXX  XXXXXXXXXX  XXXXXXXXXX  XX  XXXXXXXXXXXXXXXXXXXX XXX
_ XXXX  XXXXXXXXXX  XXXXXXXXXX  XX  XXXXXXXXXXXXXXXXXXXX XXX
_ XXXX  XXXXXXXXXX  XXXXXXXXXX  XX  XXXXXXXXXXXXXXXXXXXX XXX
_ XXXX  XXXXXXXXXX  XXXXXXXXXX  XX  XXXXXXXXXXXXXXXXXXXX XXX
_ XXXX  XXXXXXXXXX  XXXXXXXXXX  XX  XXXXXXXXXXXXXXXXXXXX XXX
1-Detailed description        CF3-All detailed descriptions    CF10-Menu
```

Every routing entry, identified by its sequence number, that is defined for the subsystem is displayed. For an explanation of what is in each column shown, refer to the ADDRTGE command, under the associated parameter description for each attribute. (If the compare value contains more than 20 characters, including both apostrophes, it is shown in truncated form on the first display and completely on the second display. On the first display, the first 19 characters of the truncated value, preceded by the opening apostrophe, are followed by three periods to show continuation.)

If a 1 is entered before one or more of the sequence numbers shown, a different display presents a detailed description of each routing entry selected. Each time the Enter key is pressed, the detailed description of the next entry is presented in the following format:

```
   SUBSYSTEM DESCRIPTION - XXXXXXXXXX    Status: XXXXXXXX    XXX
XX/XX/XX  XX:XX:XX    ROUTING ENTRY SEQUENCE NUMBER - XXXX
Program name:                XXXXXXXXXX
  Library name:              XXXXXXXXXX
Class name:                  XXXXXXXXXX
  Library name:              XXXXXXXXXX
Pool identifier:             XX
Max active routing steps:    XXXXXX
Compare start position:      XX
Compare value:               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXX
CF10-Routing entry list
```

Again, refer to the ADDRTGE command description for a description of each attribute displayed for the routing entry.

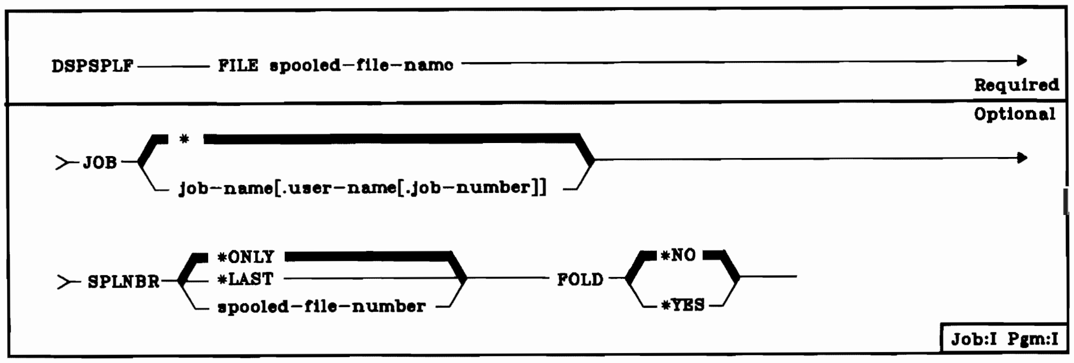
The CF10 and CF2 keys function the same way for displaying routing entries as they do for displaying work station entries, as described under *Option 5*. And, if you use CF3 instead of entering 1s before the entries on the routing entry summary display, you can display every routing entry in detail (one at a time) by using the Enter key.

DSPSPLF (Display Spooled File) Command

The Display Spooled File (DSPSPLF) command displays the data records in the specified spooled output file. The current contents of the file (data records) can be displayed anytime that an entry for the spooled file is on the output queue. The data records can be either folded or truncated (as specified in this command). To position to a particular record in the file, the user can enter a value in the control field on the display.

For more information on displaying the names of all spooled output files for a job, refer to the *Additional Considerations* section of the DSPJOB (Display Job) command.

Restrictions: The user entering this command must have either (1) created the spooled file, or (2) must have read rights to the output queue that contains the spooled file. The output queue must have had DSPDTA(*YES) specified for its display data attribute.



FILE Parameter: Specifies the name of the spooled output file produced by a job that is to have its records displayed. The file name is the name of the device file that was used by the program to produce the spooled output file.

JOB Parameter: Specifies the name of the job that created (or is creating) the spooled output file whose data records are to be displayed.

<u>*</u>: The job that issued this DSPSPLF command is the job that created the spooled file.

qualified-job-name: Enter the qualified name of the job that created the spooled file. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file from the job whose data records are to be displayed. (For an expanded description of the SPLNBR parameter, see Appendix A.)

*ONLY: Only one spooled output file from the job has the specified file name; therefore, the number of the spooled file is not necessary. If *ONLY is specified and more than one spooled output file has the specified name, an error message is displayed to the user.

**LAST:* The highest numbered spooled output file with the specified file name is the file to be displayed.

spooled-file-number: Enter the number of the spooled file having the specified file name whose data records are to be displayed.

FOLD Parameter: Specifies which display format is to be used initially. Refer to the *Additional Considerations* section of this command for more information on display formats.

*NO: The records are not to be displayed in folded format. This display format allows the scan function.

**YES:* The records are to be displayed in folded format. This display format does not allow the scan function.

Example

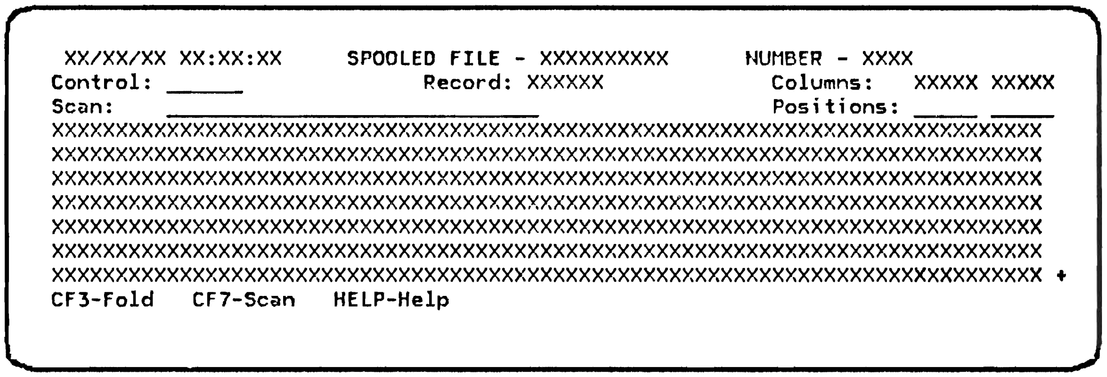
```
DSPSPLF  FILE(QPRINT) JOB(PAYROLL01) SPLNBR(4)
```

This example displays the file QPRINT, which was produced by the job PAYROLL01. The file being displayed is the fourth file produced by that job. The record positions that exceed the length of a displayed line are to be truncated when the record is displayed.

Additional Considerations

Two display formats can be produced by the DSPSPLF command to display the data in a spooled printer, card, or diskette file. These formats can be specified to be either folded or windowed (not folded).

If the display is requested with the default FOLD(*NO), the windowed display is presented:



```
 XX/XX/XX XX:XX:XX     SPOOLED FILE - XXXXXXXXXX      NUMBER - XXXX
Control: ______              Record: XXXXXX             Columns:   XXXXX XXXXX
Scan:    _____________________________                  Positions: _____ _____
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX +
CF3-Fold   CF7-Scan   HELP-Help
```

Once either display is presented (that is, showing records in truncated form or in complete, folded form), the CF3 key can be used to switch between both types of displays–from truncated records to folded records, and vice versa.

The first line of the display gives the current job date and time, and the name and number of the spooled file being displayed. The second line contains the control field, in which a value can be entered to select and control display functions. The second line also contains the page, line, and column numbers of the spooled file being displayed for a printer file. For spooled card and diskette files, the second line shows the record number of the record that begins on the fourth line of the display.

The control characters you can enter in the control field are:

E	Displays the records at the end of the file (if the file is still open, you can view added records by periodically pressing the Enter key)
#n	Displays the data that is n lines or records before or beyond the first one shown
P#n	Displays the data n pages ahead of or behind the current page
W#n	Displays the page n screen widths to the left or right of the currently displayed left column (the column number values indicate the position), where # is the +, -, or no character, and n equals a number of pages.

Examples of the use of the control characters follow:

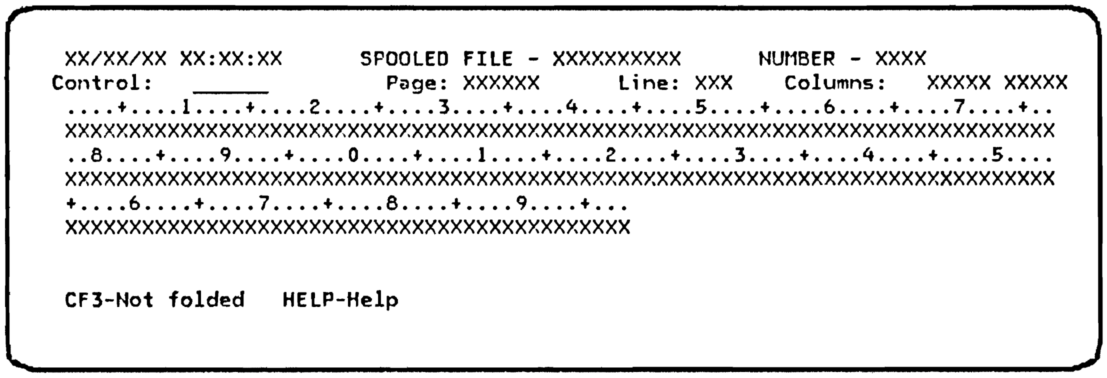
1	Display the first record (or line) of the file.
+50	The data or record that is fifty lines beyond the current first line is moved to and displayed on the first line.
P	Display the next page.
P-13	Display the thirteenth page before the first page shown on the current display.
P4	Display the beginning of page four.
W-	Display the data that is one screen width to the left.
W+13	Display the data that begins thirteen positions after (to the right of) the first position shown on this display.

The Roll keys can also be used to page through the displayed spooled file, forward or backward. If the record format is in folded mode, the roll keys scroll up or down through the file one record at a time. (The folded format and the window format can be interchanged by pressing the CF3 key.) When the roll keys are used, the control field value is ignored.

The third line of a printer display contains the Scan and Position fields. When you enter a character string value (including blanks) in the scan field and press the CF7 key, the system searches the spooled file for the value in the area delimited by the values specified in the position fields. (Refer to the Help text for rules on specifying the string to be used for scanning and the position to specify.) The first occurrence of the value is highlighted on the display. To search for other occurrences of the value, press the CF7 key again. If the value is found in a location outside the current window, the display is shifted to the window that contains the highlighted value (the column numbers in line 2 will reflect the window position). When no other occurrences of the scan value are found, an information message appears at the bottom of the display.

Input fields on the displays are initially set to an appropriate default or blank value, depending on the field. After you enter a different value, that value remains until you delete it. Then, the field assumes the original default value.

If the display is requested with FOLD(*YES) specified, the folded display is presented:



```
 XX/XX/XX XX:XX:XX      SPOOLED FILE - XXXXXXXXXX      NUMBER - XXXX
Control:  ______           Page: XXXXXX      Line: XXX    Columns:   XXXXX XXXXX
 ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+..
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 ..8....+....9....+....0....+....1....+....2....+....3....+....4....+....5....
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 +....6....+....7....+....8....+....9....+...
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


 CF3-Not folded   HELP-Help
```

The first two lines of the display show the same information presented for the windowed spooled file. The control field can be used to position the display the same way it is used for the windowed file.

Beginning with the third line, and continuing on alternating lines for the length of the spooled file, is a position line. This line allows you to determine the positions of characters in the displayed records. The data records are displayed on lines alternating with the position lines.

Refer to the Help text for rules on specifying the string to be used for scanning and the position to specify.

The command function keys available for use are indicated near the bottom of the displays. They are defined as follows:

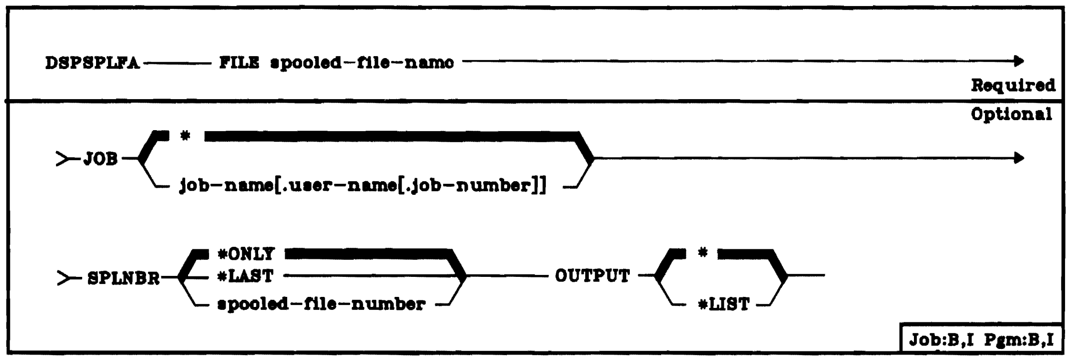
CF1	Return to the working display (command entry display, programmer menu, and so forth).
CF3	Change between folded format and windowed format.
CF7	Searches for the next occurrence of the specified character string (valid only for window format).
Help	Provides a summary of the control field functions and a summary of the scan function.

DSPSPLFA (Display Spooled File Attributes) Command

The Display Spooled File Attributes (DSPSPLFA) command displays the current attributes of the specified spooled output file. The attributes can be displayed after the file is opened and while its file entry remains on the output queue.

For more information on displaying the names of all spooled output files for a job, refer to the *Additional Considerations* section of the DSPJOB (Display Job) command.

Restrictions: The user submitting this command must have either (1) created the spooled file, or (2) must have read rights to the output queue that contains the spooled file, or (3) must have job control rights, SPCAUT(*JOBCTL), specified in his user profile. The output queue must have the OPRCTL(*YES) attribute specified.



FILE Parameter: Specifies the name of the spooled output file produced by a job that is to have its file attributes displayed. The file name is the name of the device file that was used by the program to produce the spooled output file. The name could have been specified in the program that created (opened) the file, or it could have been specified in an override command.

JOB Parameter: Specifies the qualified name of the job that created the spooled file whose attributes are to be displayed.

<u>*</u>: The job that issued this DSPSPLFA command is the job that created the spooled file.

qualified-job-name: Enter the qualified name of the job that created the spooled file. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file from the job whose attributes are to be displayed. (For an expanded description of the SPLNBR parameter, see Appendix A.)

<u>*ONLY:</u> Only one spooled output file from the job has the specified file name; therefore, the number of the spooled file is not necessary. If *ONLY is specified and more than one spooled output file has the specified name, an error message is displayed to the user.

**LAST:* The highest numbered spooled output file with the specified file name is the file to be displayed.

spooled-file-number: Enter the number of the spooled file having the specified file name whose attributes are to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPSPLFA FILE(QPRINT) OUTPUT(*LIST)
```

This command causes a list of the current attributes of the spooled output file QPRINT to be sent to the job's output queue for printing. The job in which this command is entered can produce only one output file named QPRINT.

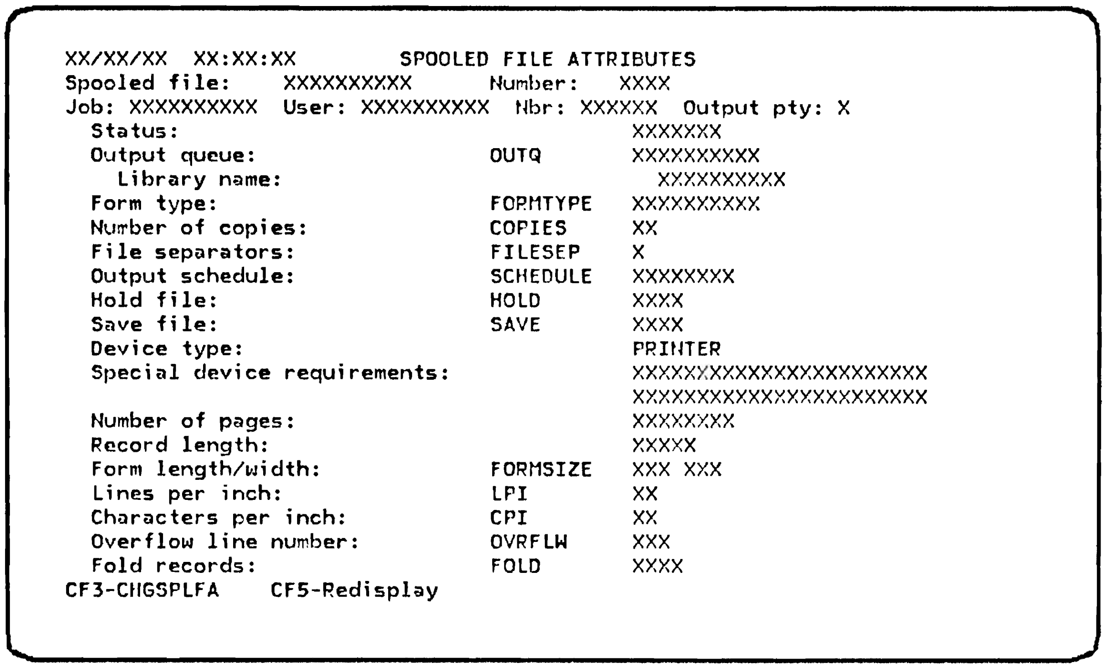
Additional Considerations

Three displays are used here to show the attributes of spooled files that have output on an output queue. A somewhat different display is used to present the attributes of a spooled printer device file, a card device file, and a diskette device file.

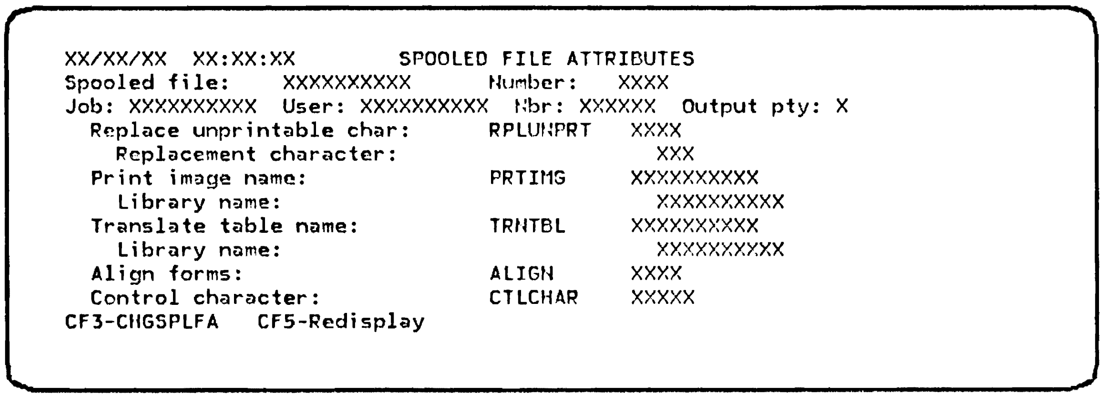
The display presented as a result of the DSPSPLFA command being entered for a spooled printer file displays the following attributes of a file being spooled to an output queue for a printer. Besides identifying the name of the spooled printer file, the command also identifies the name of the job producing the file, and the number of the spooled file that is to be displayed. The same information that is displayed can also be printed if OUTPUT(*LIST) is specified on the command.

The first four lines of each display (for printer, card, or diskette files) contain the same information. The information shown on every display produced by the DSPSPLFA command is:

- The current job date and time.
- The name of the spooled file being displayed.
- The number of the spooled file being displayed.
- The qualified name of the job that produced the spooled file. (For an explanation of the parts of a qualified job name, refer to the expanded description of the JOB parameter in Appendix A.)
- The output priority assigned to the file.



```
XX/XX/XX  XX:XX:XX        SPOOLED FILE ATTRIBUTES
Spooled file:    XXXXXXXXXX       Number:   XXXX
Job: XXXXXXXXXX  User: XXXXXXXXXX  Nbr: XXXXXX  Output pty: X
  Status:                                     XXXXXXX
  Output queue:                    OUTQ       XXXXXXXXXX
    Library name:                               XXXXXXXXXX
  Form type:                       FORMTYPE   XXXXXXXXXX
  Number of copies:                COPIES     XX
  File separators:                 FILESEP    X
  Output schedule:                 SCHEDULE   XXXXXXXX
  Hold file:                       HOLD       XXXX
  Save file:                       SAVE       XXXX
  Device type:                                PRINTER
  Special device requirements:                XXXXXXXXXXXXXXXXXXXXXXX
                                              XXXXXXXXXXXXXXXXXXXXXXX
  Number of pages:                            XXXXXXXX
  Record length:                              XXXXX
  Form length/width:               FORMSIZE   XXX XXX
  Lines per inch:                  LPI        XX
  Characters per inch:             CPI        XX
  Overflow line number:            OVRFLW     XXX
  Fold records:                    FOLD       XXXX
CF3-CHGSPLFA    CF5-Redisplay
```



```
XX/XX/XX  XX:XX:XX        SPOOLED FILE ATTRIBUTES
Spooled file:    XXXXXXXXXX       Number:   XXXX
Job: XXXXXXXXXX  User: XXXXXXXXXX  Nbr: XXXXXX  Output pty: X
  Replace unprintable char:        RPLUNPRT   XXXX
    Replacement character:                      XXX
  Print image name:                PRTIMG     XXXXXXXXXX
    Library name:                               XXXXXXXXXX
  Translate table name:            TRNTBL     XXXXXXXXXX
    Library name:                               XXXXXXXXXX
  Align forms:                     ALIGN      XXXX
  Control character:               CTLCHAR    XXXXX
CF3-CHGSPLFA   CF5-Redisplay
```

Beginning with line 4 of the spooled printer file display, the attributes of the printer device file used are displayed. Because this is a spooled file for the printer, PRINTER is shown as the device type. The attributes shown on this display are those contained in the printer file description, the file overrides that were used, or attributes that were changed by the CHGSPLFA command. The record length for a program-described printer file specifies the number of bytes of data for each printer record; for externally described printer files, the record length will be *RCDFMT. The line for *Special device requirements* shows information about any attributes the printer device file may have had assigned to it through the DDS. A YES shown in this field indicates that the spooled file has not been closed (at the time of this command execution) and requires some special requirements. Refer to the associated parameter descriptions in the CRTPRTF command description for an explanation of the other attributes.

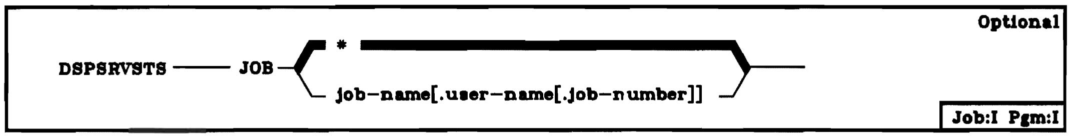
If you want to change any attributes of the displayed spooled file, you can press the CF3 key while any of the spooled attribute displays are shown. The CF3 key causes the prompt for the CHGSPLFA command to be displayed with the name and number of the spooled file, and its job name, user name, and job number already filled in. You can enter values for one or more of the attributes that you want changed. After the Enter key is pressed, the spooled file attributes are again displayed with the changed values.

To reshow the display with the most current attributes, press the CF5 key.

DSPSRVSTS (Display Service Status) Command

The Display Service Status (DSPSRVSTS) command displays information about the current service status of the specified job. This includes the name of the job it is servicing and/or the name of the job servicing the specified job.

Restriction: This command is valid only in interactive jobs.



JOB Parameter: Specifies which job is to have its service status displayed.

*: Status information is displayed about the job in which the DSPSRVSTS command is entered.

qualified-job-name: Enter the qualified name of the job whose status information is displayed. (If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name; the name specified must be unique within the system.)

Example

```
DSPSRVSTS
```

This command displays the service status information for the job in which the command is executed.

Additional Considerations

The display produced by the DSPSRVSTS command has the following format:

```
  XX/XX/XX  XX:XX:XX      SERVICE STATUS DISPLAY
Job: XXXXXXXXXX  XXXXXXXXXX  XXXXXX
  Servicing:    XXXXXXXXXX  XXXXXXXXXX  XXXXXX
  Serviced by: XXXXXXXXXX  XXXXXXXXXX  XXXXXX

Trace storage: XXXXX K
Trace count:   XXXXXX
```

The current job date is shown on line 1. Line 2 shows the qualified name of the job (in three fields) whose current service status is being displayed. If the specified job is servicing another job, the name of the job being serviced is given on line 3. If the specified job is itself being serviced, the name of the job that is servicing it is given on line 4.

For the job being displayed, the status of the trace in the job is displayed by showing:

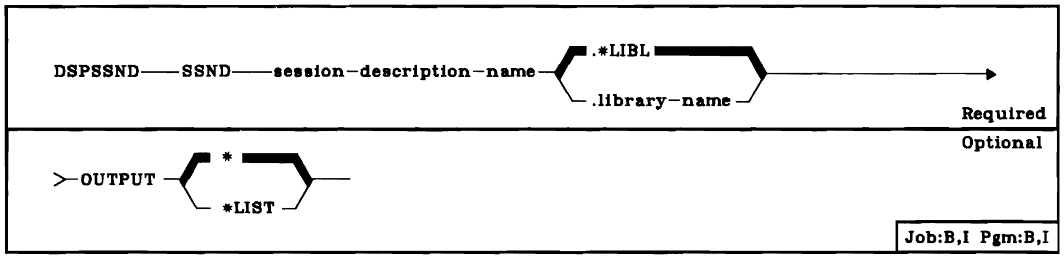
- The current amount of storage (in bytes) that is occupied by trace records for the job
- The current number of trace records for the job

DSPSSND (Display Session Description) Command

The Display Session Description (DSPSSND) command displays the current contents of the active or inactive RJEF session description.

Restriction: To use this command, you must have read rights for the library in which the session description is stored.

The Display Session Description (DSPSSND) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSND Parameter: Specifies the qualified name of the session description that is to be displayed. (If no library qualifier is given, *LIBL is used to find the session description.)

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or printed with the job's spooled output on a printer.

<u>*</u>: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer. The IBM-supplied device file QPRTSSND is used to print the RJEF session description.

Example

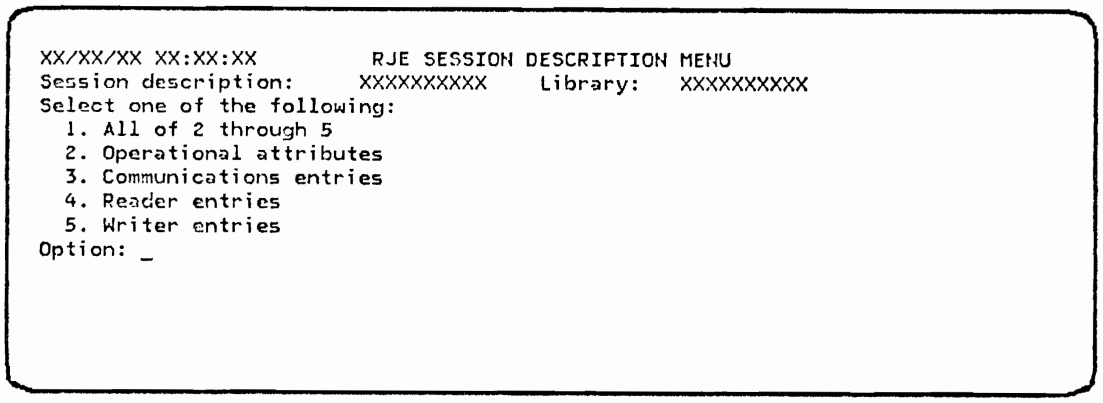
```
DSPSSND SSND(RJE.USERLIB) OUTPUT(*)
```

This command (if entered from a batch job) causes a complete set of display information about the session description named RJE (stored in library USERLIB) to be sent to the job's spooling queue to be printed. If the command is entered from an interactive job, the session description menu is displayed, from which detailed selections can be made. The information includes the session attributes, communications entries, writer entries, and reader entries.

Additional Considerations

The session description displays current information contained in the identified session description.

There are several types of RJE session description displays. The Display Session Description (DSPSSND) command causes the following menu to appear:



```
XX/XX/XX XX:XX:XX          RJE SESSION DESCRIPTION MENU
Session description:     XXXXXXXXXX    Library:   XXXXXXXXXX
Select one of the following:
  1. All of 2 through 5
  2. Operational attributes
  3. Communications entries
  4. Reader entries
  5. Writer entries
Option: _
```

Only the attributes specified in the Create Session Description (CRTSSND) command and the functions specified in the session description through the Add RJE Communications Entry (ADDRJECMNE), Add RJE Reader Entry (ADDRJERDRE), or Add RJE Writer Entry (ADDRJEWTRE) commands are displayed for selection.

Selecting one or all functions from this menu causes one or all of the following displays to appear:

- Session description operational attributes display
- Session description communications entry selection list
- Session description communications entry display
- Session description reader entry selection list
- Session description host reader entry display
- Session description writer entry selection list
- Session description host writer entry display (two displays)

Option 1 starts the sequence of displays, which ends with a return to the RJE session description menu.

Session Description Attributes Display

```
XX/XX/XX XX:XX:XX        SESSION DESCRIPTION ATTRIBUTES                    +++
Session description:     XXXXXXXXXX    Library:   XXXXXXXXXX
  Host RJE subsystem type:        TYPE       XXXXX
  Job queue name:                 JOBQ       XXXXXXXXXX
    Library name:                              XXXXXXXXXX
  Message queue name:             MSGQ       XXXXXXXXXX
    Library name:                              XXXXXXXXXX
  Forms control table name:       FCT        XXXXXXXXXX
    Library name:                              XXXXXXXXXX
  Idle time in minutes:           IDLETIME   XXXXXXXX
  Text description:               TEXT       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

For an explanation of the values shown on this display, refer to the Create Session Description (CRTSSND) command.

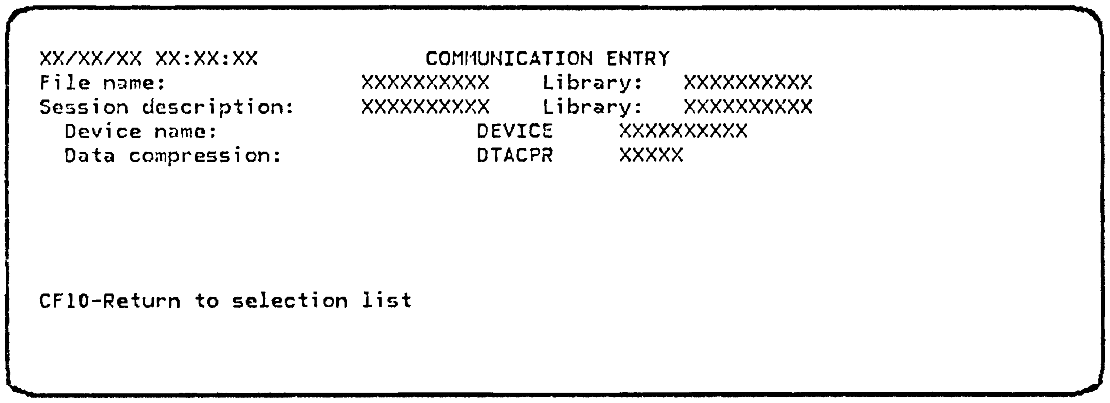
Session Description Communications Entry Selection List

```
XX/XX/XX XX:XX:XX     COMMUNICATIONS ENTRY SELECTION LIST                 +++
Session description:     XXXXXXXXXX    Library:   XXXXXXXXXX
  FILE NAME  LIBRARY     DEVICE NAME  COMPRESS
_ XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX      XXXXX
_ XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX      XXXXX
_ XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX      XXXXX
_ XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX      XXXXX
_ XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX      XXXXX
_ XXXXXXXXXX XXXXXXXXXX  XXXXXXXXXX      XXXXX
1-Detailed description
CF3-All detailed descriptions   CF10-Return to selection menu
```

If a 1 is entered for one or more selections, one or more session description communications entry displays are shown in the order of the entries on the session description communications entry selection list display.

Pressing the CF3 key starts the sequence of displays, which ends with a return to the session description communications entry selection list.

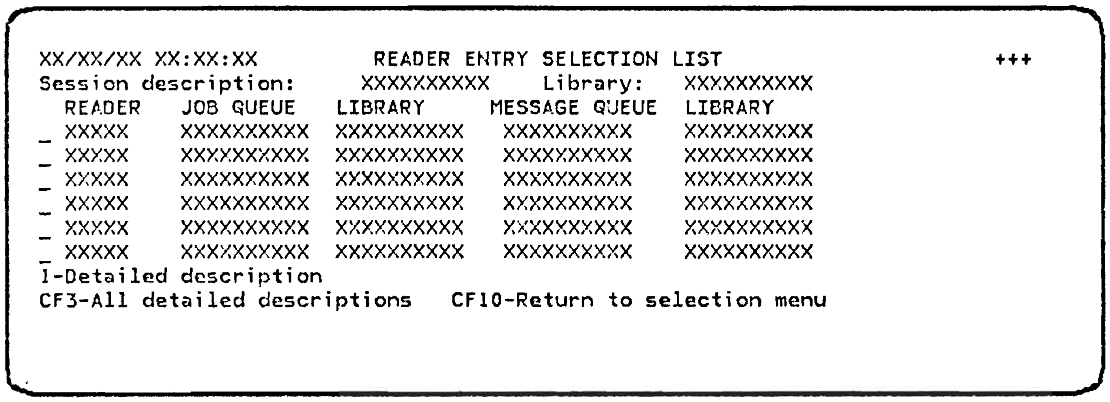
Session Description Communications Entry Display



```
XX/XX/XX XX:XX:XX            COMMUNICATION ENTRY
File name:                 XXXXXXXXXX    Library:    XXXXXXXXXX
Session description:       XXXXXXXXXX    Library:    XXXXXXXXXX
  Device name:                      DEVICE      XXXXXXXXXX
  Data compression:                 DTACPR      XXXXX




CF10-Return to selection list
```

For an explanation of the values shown on this display, refer to the Add RJE Communications Entry (ADDRJECMNE) command.

Session Description Reader Entry Selection List



```
XX/XX/XX XX:XX:XX          READER ENTRY SELECTION LIST                      +++
Session description:       XXXXXXXXXX    Library:    XXXXXXXXXX
  READER    JOB QUEUE    LIBRARY      MESSAGE QUEUE   LIBRARY
_ XXXXX     XXXXXXXXXX  XXXXXXXXXX    XXXXXXXXXX     XXXXXXXXXX
_ XXXXX     XXXXXXXXXX  XXXXXXXXXX    XXXXXXXXXX     XXXXXXXXXX
_ XXXXX     XXXXXXXXXX  XXXXXXXXXX    XXXXXXXXXX     XXXXXXXXXX
_ XXXXX     XXXXXXXXXX  XXXXXXXXXX    XXXXXXXXXX     XXXXXXXXXX
_ XXXXX     XXXXXXXXXX  XXXXXXXXXX    XXXXXXXXXX     XXXXXXXXXX
_ XXXXX     XXXXXXXXXX  XXXXXXXXXX    XXXXXXXXXX     XXXXXXXXXX
1-Detailed description
CF3-All detailed descriptions   CF10-Return to selection menu
```

If a 1 is entered for one or more selections, one or more session description host reader entry displays are shown in the order of the entries on the session description reader entry selection list display.

Pressing the CF3 key starts the sequence of displays, which ends with a return to the RJE session description menu.

Session Description Host Reader Entry Display

```
XX/XX/XX XX:XX:XX                     READER ENTRY
Reader name:               XXXXX
Session description:       XXXXXXXXXX    Library:   XXXXXXXXXX
  Job queue name:                   JOBQ      XXXXXXXXXX
    Library name:                               XXXXXXXXXX
  Message queue name:               MSGQ      XXXXXXXXXX
    Library name:                               XXXXXXXXXX



CF10-Return to selection list
```

For an explanation of the values shown on this display, refer to the Add RJE Reader Entry (ADDRJERDRE) command.

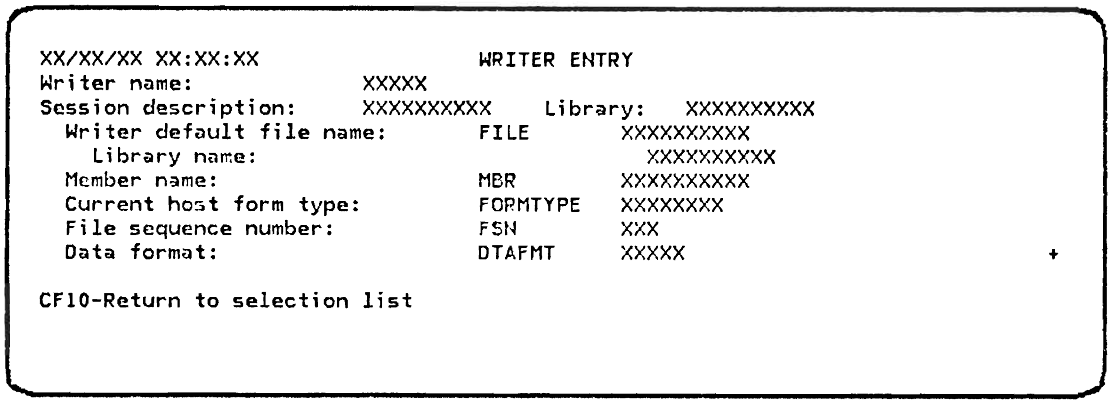
Session Description Writer Entry Selection List

```
XX/XX/XX XX:XX:XX          WRITER ENTRY SELECTION LIST
Session description:       XXXXXXXXXX    Library:   XXXXXXXXXX
  WRITER   FILE NAME  LIBRARY       MEMBER NAME   HOST FORM
_  XXXX    XXXXXXXXXX XXXXXXXXXX    XXXXXXXXXX    XXXXXXXX
_  XXXX    XXXXXXXXXX XXXXXXXXXX    XXXXXXXXXX    XXXXXXXX
_  XXXX    XXXXXXXXXX XXXXXXXXXX    XXXXXXXXXX    XXXXXXXX
_  XXXX    XXXXXXXXXX XXXXXXXXXX    XXXXXXXXXX    XXXXXXXX
_  XXXX    XXXXXXXXXX XXXXXXXXXX    XXXXXXXXXX    XXXXXXXX
_  XXXX    XXXXXXXXXX XXXXXXXXXX    XXXXXXXXXX    XXXXXXXX
1-Detailed description
CF3-All detailed descriptions   CF10-Return to selection menu
```

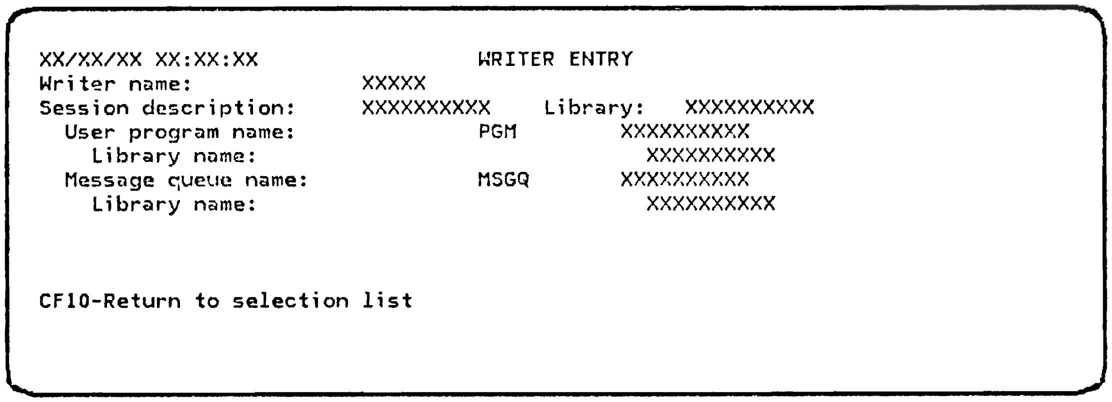
If a 1 is entered for one or more selections, one or more session description host writer entry displays are shown in the order of the entries on the session description writer entry selection list display.

Pressing the CF3 key starts the sequence of displays, which ends with a return to the RJE session description menu.

Session Description Host Writer Entry Display



```
XX/XX/XX XX:XX:XX                 WRITER ENTRY
Writer name:              XXXXX
Session description:      XXXXXXXXXX    Library:   XXXXXXXXXX
  Writer default file name:        FILE       XXXXXXXXXX
    Library name:                               XXXXXXXXXX
  Member name:                     MBR        XXXXXXXXXX
  Current host form type:          FORMTYPE   XXXXXXXX
  File sequence number:            FSN        XXX
  Data format:                     DTAFMT     XXXXX                          +

CF10-Return to selection list
```

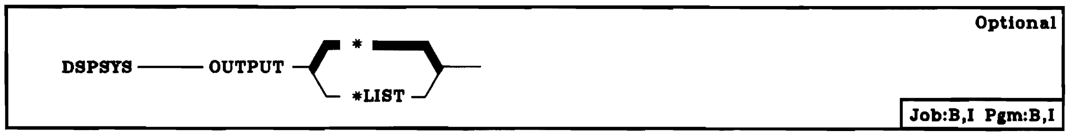


```
XX/XX/XX XX:XX:XX                 WRITER ENTRY
Writer name:              XXXXX
Session description:      XXXXXXXXXX    Library:   XXXXXXXXXX
  User program name:               PGM        XXXXXXXXXX
    Library name:                               XXXXXXXXXX
  Message queue name:              MSGQ       XXXXXXXXXX
    Library name:                               XXXXXXXXXX



CF10-Return to selection list
```

For an explanation of the values shown on these displays, refer to the Add RJE Writer Entry (ADDRJEWTRE) command.

DSPSYS (Display System) Command

The Display System (DSPSYS) command displays the status and information about each subsystem in the system. Also, if one of the subsystems shown on the system display is selected, additional information listing all of the jobs active in that subsystem can be displayed.



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPSYS
```

This command, if entered from a work station, causes information about the system to be displayed. A user can obtain additional information about one of the subsystems by entering a 1 before the subsystem's name on the system display. If the command is entered from a batch job, the information is directed to the job's output spooling queue to be printed.

Additional Considerations

The display produced by the DSPSYS command has the following format:

```
XX/XX/XX  XX:XX:XX          SYSTEM DISPLAY
  SUBSYSTEM    SBS    ACT          TOTAL    SYS POOLS BY SBS POOL ID
  NAME         NUMBER JOBS STATUS  STORAGE   1  2  3  4  5  6  7  8  9 10
_ XXXXXXXXXX   XXXXXX XXXX XXXXXX  XXXX K   XX XX XX XX XX XX XX XX XX XX
_ XXXXXXXXXX   XXXXXX XXXX XXXXXX  XXXX K   XX XX XX XX XX XX XX XX XX XX
    .
    .



1-DSPSBS    2-DSPSBSD    9-TRMSBS              CF3-DSPSYSSTS     CF5-Redisplay
```

The first line of the system display gives the current job date and time. Then, for each subsystem that is active in the system, a separate line is used to display:

- The name of the subsystem that was specified on the STRSBS command.
- The system-assigned number (primarily for use by IBM service personnel) that is associated with the subsystem.
- The number of jobs active in the subsystem. If two jobs are initiated from the same work station (with the Sys Req key), they are counted as only one job on this display, but both jobs are shown on the display produced by the DSPSBS command.
- The status of the subsystem, which can be either ACTIVE, TRM (in the process of terminating), or RSTD (the controlling subsystem is in the restricted state). For more information on the restricted subsystem, refer to the discussion on *Additional Considerations* in the TRMBSB command description.
- The total amount of main storage (in K-bytes) allocated to the storage pools assigned to the subsystem. Storage that has been allocated from the base system pool (pool 2) is not included in this total.
- The system pool identifiers of up to ten of the system storage pools used by the subsystem. These are *not* the identifiers that are specified in the subsystem description; they *are* the same as the identifiers shown on the *system status* display (produced by the DSPSYSSTS command). For example, the base system pool is identified here as pool 2. Refer to the DSPSBSD command description, in the discussion of option 3 under *Additional Considerations*, for information on the difference. Subsystem storage pools 6 through 10 are not shown on the system console.

Additional information about one of the subsystems, or about the system's status, can be requested on this display. If a 1 is entered at the beginning of a line before a subsystem's name, the same *subsystem* display produced by the DSPSBS command is presented for that subsystem.

You can request multiple subsystem displays with one screen, by entering the option 1 for each subsystem required. These requests are processed one at a time; you can terminate individual display processing by pressing the Enter key at the last display in the sequence of displays.

If you entered a 2 at the beginning of a subsystem information line, information dealing with the subsystem description is displayed as if you entered the Display Subsystem Description (DSPSBSD) command. You must have operational rights for the subsystem description before you can use this option to display its contents. Refer to the description of the *DSPSBSD command* for more information.

Option 9 uses the Terminate Subsystem (TRMSBS) command with default parameters [OPTION(*CNTRLD) and DELAY(*NOLIMIT)] to terminate a chosen subsystem. The subsystem will terminate, but only after all jobs in the subsystem have finished. You must have job control authority (SPCAUT(*JOBCTL) must be specified in the Create User Profile (CRTUSRPRF) command) to terminate a subsystem. Refer to *Terminate Subsystem (TRMSBS) Command* for more information on terminating a subsystem.

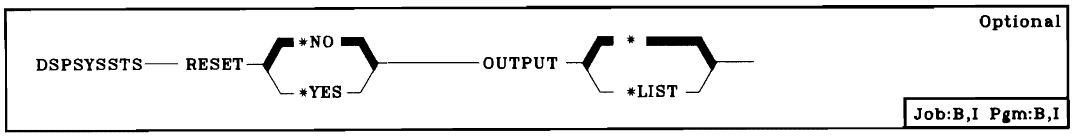
If the CF3 key is pressed, the current status of the entire system is presented on the *system status* display, which is the same as that produced by the DSPSYSSTS command. Refer to the DSPSBS and DSPSYSSTS command descriptions (under *Additional Considerations*) for an explanation of what is on these displays.

The CF2 key can be used to return to the previous display in a sequence or to return to the caller from the first display in the sequence.

Pressing the CF5 key redisplays the system display, showing the most current system information.

DSPSYSSTS (Display System Status) Command

The Display System Status (DSPSYSSTS) command displays a group of statistics that shows the current status of the system. It displays the number of jobs currently in the system, the total capacity of auxiliary storage, the percentage of auxiliary storage currently in use, the percentage of machine addresses used, and statistical information related to each storage pool that currently has main storage allocated to it. The statistical information is gathered during an identified time interval (shown as the elapsed time); the data can either be updated by extending the measurement time interval, or it can be reset and a new time interval can be started to gather a new set of data.



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

***: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

RESET Parameter: Specifies whether system status statistics fields are to be reset to zero, as if this was the first occurrence of the DSPSYSSTS command in this job.

NO: The system status statistics are not to be reset.

**YES:* The system status statistics are to be reset.

Example

```
DSPSYSSTS  OUTPUT(*LIST)
```

This command directs the system status information to the job's output spooling queue to be printed. If OUTPUT(*) is specified instead and the command was entered from a work station, the information about the system is displayed at the work station.

Additional Considerations

The display produced by the DSPSYSSTS command has the following format:

```
XX/XX/XX XX:XX:XX     SYSTEM STATUS DISPLAY    CPU: XXX.X%
Elapsed: XX:XX:XX               Addr segments used: XXX.XXXX%
Aux stg total: XXXXXM  Used: XXX.XXXX%  Jobs in system: XXXXX
 SYS  POOL     DB    DB NONDB NONDB  MAX  ACT-> WAIT-> ACT->
POOL  SIZE   FLTS PAGES  FLTS PAGES  ACT   WAIT  INELG INELG
 XX XXXXXK  XXX.X XXX.X XXX.X XXX.X  XXX  XXX.X  XXX.X XXX.X
 XX XXXXXK  XXX.X XXX.X XXX.X XXX.X  XXX  XXX.X  XXX.X XXX.X
   .
   .
   .

CF3-DSPSYS   CF5-Redisplay   CF6-Reset start   CF8-DSPACTJOB
```

The first line of the system status display gives the current *system* date and time. This is the time either when the display is first presented (after the DSPSYSSTS command is entered) or when it is redisplayed with updated statistics (after the CF5 or CF6 key is pressed). Also shown on the first line is the percentage of the elapsed time the CPU was in use. Beginning with the second line, the following information is displayed:

- Elapsed time: The time that has elapsed between the measurement start time and the current system time.

- Address segments used: The percentage of the maximum possible addresses currently in use. This maximum includes all temporary and permanent addresses that can be used.

- Auxiliary storage total: The capacity of online storage configured for the system.

- Auxiliary storage used: The percentage of the total online auxiliary storage that is currently in use.

- Jobs in system: The total number of user and system jobs that are currently in the system; it includes all jobs on job queues waiting to be processed, all jobs currently active (executing), and all jobs with output on output queues to be produced.

The remainder of the display gives, for each of the 16 possible *system* storage pools that currently has main storage allocated to it: its system-related pool identifier, its storage size, its data base and non-data base paging and page fault rates, its maximum activity level, and its transition rates (between states). This information is displayed under the column headings given on lines 4 and 5.

- System pool identifier: The system-related pool identifier for the pool. The pool identifiers on this display are the same as those given on the system display; the system display can help you to identify to which subsystem (if any) the storage pools are currently assigned. These identifiers are *not* the same as those specified in the subsystem description; refer to the DSPSBSD command description, in the discussion of option 3 under *Additional Considerations*, for information on the difference.
  - Pool identifiers 1 and 2 always represent the machine and base storage pools. The *machine storage pool* (pool 1) contains highly shared machine and CPF programs; it contains the machine's (MI) pageable and nonpageable main storage. The actual size of pool 1 is specified by the system value QMCHPOOL.
  - The *base storage pool* (pool 2) is the shared system pool; many CPF functions and some system jobs are executed in the base pool. The base pool contains all of the main storage not allocated to all the other pools in the system. The minimum size of the base pool is controlled by the system value QBASPOOL.

- Pool size: The amount of storage, in K-bytes, of main storage in the pool.

- Data base faults: The rate, shown in page faults per second, of page faults against pages containing either data base data or access paths.

- Data base pages: The rate, shown in pages per second, at which data base pages are brought into the storage pool.

- Non-data base faults: The rate, shown in page faults per second, of page faults against pages other than those designated as data base pages.

- Non-data base pages: The rate, shown in pages per second, at which non-data base pages are brought into the storage pool.

- Maximum activity level: The maximum number of jobs that can be active in the pool at any time. (This value is determined by the activity level specified in the subsystem description, or by the system value QBASACTLVL if the base system pool is shared with this pool.)

- Active-to-wait state: The rate, in transitions per minute, of transitions of jobs from an active state to a waiting state.

- Wait-to-ineligible state: The rate, in transitions per minute, of transitions of jobs from a waiting state to an ineligible state.

- Active-to-ineligible state: The rate, in transitions per minute, of transitions of jobs from an active state to an ineligible state.

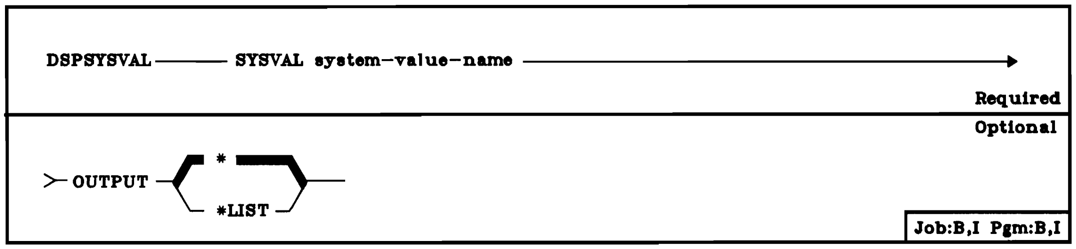
You can update the statistics on this display by pressing the CF5 key; this causes the measurement time interval to be extended and more data is gathered during that time. You can reset the elapsed time and initiate a new measurement interval by pressing the CF6 key. Using the CF3 key, you can go directly to the system display, and by again pressing the CF3 key, you can reshow the system status display, with updated statistics.

The CF2 key can be used to return to the previous display in a sequence, or if you are viewing the first display of the sequence, to return to the caller. The CF8 key can be used to display jobs active in the system.

The data is gathered and stored in counters; because these counters can overflow (wrap around), it is advisable to use the CF6 key to perform occasional resets (setting the counters to zero). Otherwise, if wraparound does occur in a counter, it may appear that very few faults have occurred in the time interval, instead of many. This is most likely to occur on the non-data base faults measurement, when the product of the elapsed time and non-data base faults exceeds 65 535.

DSPSYSVAL (Display System Value) Command

The Display System Value (DSPSYSVAL) command displays the name and the value of the specified system value.



SYSVAL Parameter: Specifies the name of the system value that is to be displayed. Refer to the *CPF Programmer's Guide* for the complete list of system values that are shipped with the system.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

**:* The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPSYSVAL  SYSVAL(QHOUR)
```

This command displays the current value of the system value QHOUR to the user.

Additional Considerations

The display produced by the DSPSYSVAL command has the following format:

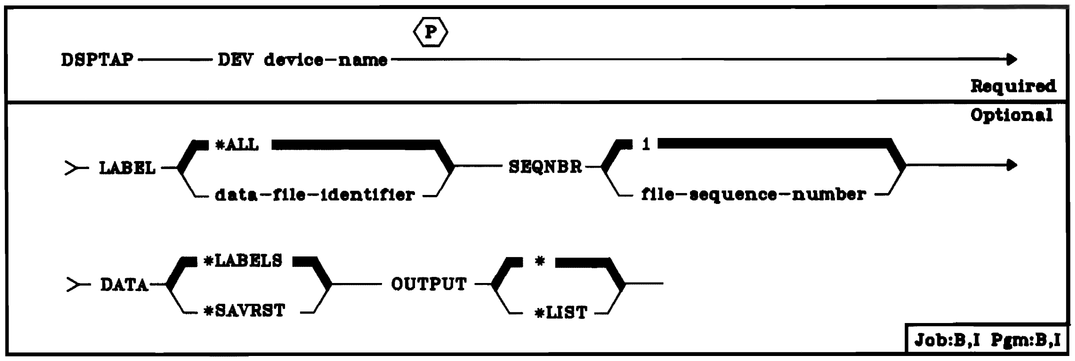
```
XX/XX/XX  XX:XX:XX   SYSTEM VALUE DISPLAY
System value name:     XXXXXXXXXX
  System value:          XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX
                         XXXXXXXXXX  XXXXXXXXXX
```

The first line of the display shows the job date and current time, and the name of the system value being displayed. The remaining line or lines show the current values assigned to the system value. If a CHGSYSVAL command has not been entered to change the value, the IBM-supplied value is displayed.

DSPTAP (Display Tape) Command

The Display Tape (DSPTAP) command displays the volume label and data file label information that is contained on a standard labeled magnetic tape or the volume type and density of a nonlabeled tape. The information can be written on a printer or displayed on a display device. This command allows you to display a single file label or all file labels on a volume. Additional information can be displayed for save/restore files by the use of the DATA parameter. (You should know whether the tape to be displayed contains data in the basic exchange format or in the save/restore format. But, if you do not, you can specify DATA(*LABELS) and check the data file identifiers listed. If the names of libraries are displayed, the data files are probably in the save/restore format.)

The volume whose information is to be displayed must be mounted on the specified device. After the DSPTAP command is entered, as much of the tape as necessary is read before the requested information is displayed.



DEV Parameter: Specifies the name of the device where the volume to be displayed is mounted.

LABEL Parameter: Specifies the data file identifier, or all the identifiers, of the data file(s) on the tape whose label(s) is to be displayed. The data file identifier is stored in the file label that precedes the data in the file.

*ALL: All data file identifiers on the mounted tape are to be displayed.

data-file-identifier: Enter the data file identifier (17 alphameric characters maximum) of the data file whose label information is to be displayed. If this identifier is on a tape in the save/restore format, the identifier can contain a maximum of 10 characters, which identify the library that was saved (or some objects in the library).

SEQNBR Parameter: Specifies, for multifile volumes, the sequence number of the data file on tape whose label information is to be displayed. If LABEL(*ALL) is specified or assumed, all files following the file specified by SEQNBR are also displayed after the specified file.

1: The data file to be displayed is the first file (or the only file) on the tape. If LABEL(*ALL) is specified, all file labels are to be displayed, and no verification of label sequence numbers is performed.

file-sequence-number: Enter the sequence number of the data file on this tape to be displayed. If a specific LABEL identifier is specified, it is compared with the label identifier of the data file with the specified sequence number (SEQNBR). If the identifiers do not match, an error message is issued. If LABEL(*ALL) is specified, all file labels whose sequence numbers are equal to or greater than the value entered are to be displayed.

DATA Parameter: Specifies the type of information that is to be displayed.

*LABELS: The volume label and data file labels are to be displayed.

**SAVRST:* The tape contains save/restore data. For each saved object, the following are displayed: the object's name, its owner's name, the date and time it was saved, its size, and whether or not its storage was already freed when the object was saved. This option is valid only if the data file was created using one of the CL save commands.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the names of the files used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Example

```
DSPTAP  DEV(QTAPE2) LABEL(*ALL)
```

The volume label and file labels on the magnetic tape volume mounted on the tape unit named QTAPE2 are displayed.

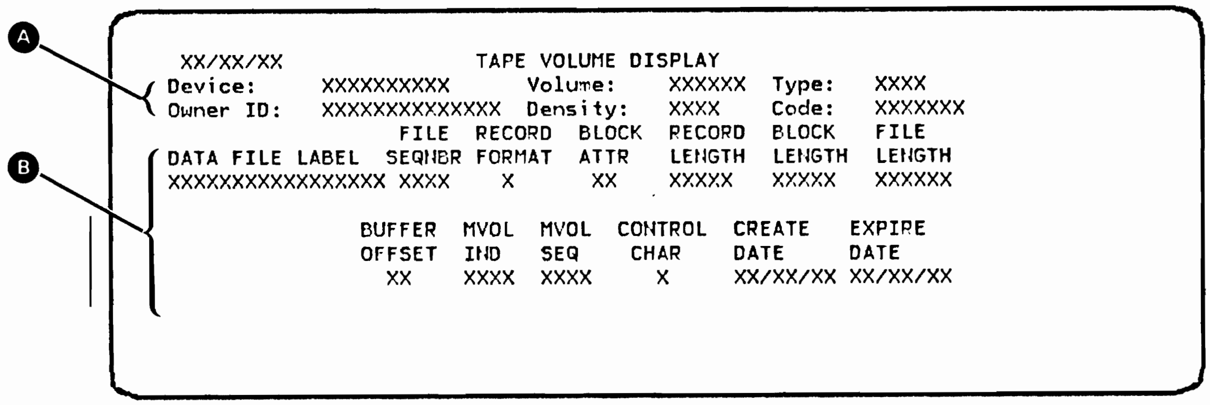
Additional Considerations

The DSPTAP command can produce two types of displays, depending on the value specified in the DATA parameter:

- If *LABELS is specified, the information in the volume label and file label area of a tape is displayed.
- If *SAVRST is specified, a series of displays that identify the libraries, objects, and data base file members saved on a tape can be displayed.

Displaying Tape and Data File Attributes

If DATA(*LABELS) is specified or assumed, the following tape volume display is shown:



```
XX/XX/XX                TAPE VOLUME DISPLAY
Device:     XXXXXXXXXX       Volume:    XXXXXX  Type:   XXXX
Owner ID:   XXXXXXXXXXXXXX  Density:   XXXX    Code:   XXXXXXX
                  FILE   RECORD  BLOCK  RECORD  BLOCK   FILE
DATA FILE LABEL  SEQNBR FORMAT  ATTR   LENGTH  LENGTH  LENGTH
XXXXXXXXXXXXXXXXX XXXX    X       XX    XXXXX   XXXXX   XXXXXX

               BUFFER  MVOL  MVOL  CONTROL  CREATE   EXPIRE
               OFFSET  IND   SEQ    CHAR    DATE     DATE
                 XX    XXXX  XXXX    X      XX/XX/XX XX/XX/XX
```

Each display of this type shows information in the volume label area at the beginning of the tape and all of the information in the file header label associated with one data file identifier. For the same tape, the volume label field information is the same on each display.

A Volume Label Fields

Device: The name of the device on which the volume being displayed is mounted.

Volume: The volume identifier of the volume being displayed. This field is blank if the volume does not have standard labels.

Label Type: The type of labels, if any, that are on the tape. The values that can be displayed are:

*SL	Standard label volume
*NL	Nonlabeled volume with no leading tape mark or initialized nonlabeled volume with no data files
*LTM	Nonlabeled volume with leading tape mark

Owner ID: The owner identifier specified by the user when the tape was initialized. (This is the NEWOWNID value on the INZTAP command.) This field is blank if the volume does not have standard labels.

Density: The density of recorded data on the volume. Either 800 or 1600 is displayed to indicate the density, in bits per inch (BPI), of the information recorded on the tape. (This is the DENSITY value specified on the INZTAP command when the volume was initialized.)

Code: The code used to record information on the volume. Either *EBCDIC or *ASCII is displayed to indicate which coding is used for data files on the tape. (This is the CODE value specified on the INZTAP command when the volume was initialized.) This field is blank if the volume does not have standard labels.

B File Header Label Fields

Data file label information is not displayed for a nonlabeled volume or for a nonlabeled volume with a leading tape mark.

Data File Label: The data file identifier of the file, which was assigned by the user when the file was created.

File Seqnbr (File Sequence Number): Identifies the sequence of the data file on this volume or within the multivolume set.

Record Format: Indicates the format of records in the file.

ƀ = Record format information not available.
F = Fixed length records.
V = Variable length records in EBCDIC or spanned ASCII format.
D = Variable length unspanned records in ASCII format.
U = Undefined (includes tapes written on System/38 in save/restore format).

Note: The following fields may be blank on the display if there is no header label 2 on the tape or the label does not contain all of the fields: record format, block attribute, record length, block length, buffer offset, multivolume indication, and control character.

Block Attribute: Indicates whether the records in the data file are blocked (multiple records are contained in each data block) or unblocked (only a single record is contained in each data block), and whether each record can span multiple blocks or must be contained in a single block. Both fixed length records and variable length records can be blocked. CPF can only process spanned records that are variable length.

ƀ (blank) = Unblocked unspanned records (or block attribute information not available)
B = Blocked unspanned records
S = Spanned unblocked records
BS = Spanned blocked records

Record Length: For fixed length records, this value is the number of bytes in each record. For variable length records, it is the maximum number of bytes in each record. The record length displayed for an undefined format file is zero because each block contains a single record with a length equal to the block length minus any ASCII buffer offset. CPF can process fixed length or undefined format records that have a length of 18 through 32 767 characters, and variable length spanned or unspanned records that have a length of 1 through 32 759 characters.

Block Length: Indicates the number of bytes in each block. (CPF can process only fixed length and variable length records that have a block length of 18 through 32 767 characters per block.)

File Length: Indicates the number of blocks in the part of the data file that is contained on this tape volume. If the entire data file is contained on one volume, then this is the total number of data blocks in the file.

Buffer Offset: Indicates the length of the buffer offset that precedes the first record in each data block. Only ASCII tape files can have a buffer offset. A buffer offset is only shown if the file labels contain the buffer offset value.

MVOL Ind (Multivolume Indication): Indicates whether the data file is complete on the specified volume or is continued on another volume.

Blank = Last or only volume for this file
CONT = Continued on another volume

MVOL Seq (Multivolume Sequence): Displays, for a multivolume file only, a number that indicates the sequential order of this tape relative to the other tapes on which this data file is stored.

Control Character: Indicates the type of control character used as the first data character in each record in this file to provide for carriage control and card punch stacker selection. (This information is ignored by System/38 when it is processing the file.)

A = ANSI standard control characters
M = Machine code control character
ƀ = Records do not contain control characters

Create Date: The date on which the data file was created. The date is displayed in the format specified by the system value QDATFMT. However, the date is stored on the tape in the Julian date format. The Julian date format is *yynnn*, where *yy* is the year and *nnn* is the *n*th day in the year (for example, the Julian date for February 1, 1979 is 79032).

Expire Date: The date on which the data file can be deleted. (This date is also displayed in the format specified by the QDATFMT system value.) No date indicates the data file can be deleted; the expiration date has passed. If *PERM is displayed, the file is a permanent file and should not be overwritten.

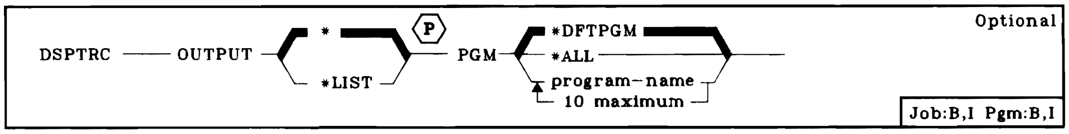
Tapes in Save/Restore Format

The displays for tapes in the save/restore format are nearly identical to those used for diskettes in the save/restore format. Refer to the descriptions of the save/restore displays under *Additional Considerations* in the DSPDKT command.

DSPTRC (Display Trace) Command

The Display Trace (DSPTRC) command displays all of the traces that are currently defined in the program(s) specified in this command. The following tracing information is displayed: the statement ranges or the System/38 instruction ranges in the program, the name or MI ODV numbers of all the program variables associated with the trace statements, and whether the variables are recorded only when a traced statement changes their values.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command*.



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

*LIST: The output is to be listed with the job's spooled output on a printer.

PGM Parameter: Specifies which programs in the debugging environment are to have all of their trace statements and associated program variables displayed.

*DFTPGM: Only the default program is to have its trace statements displayed.

*ALL: All the programs currently in debug mode are to have their trace statements displayed.

program-name: Enter the names of one or more programs whose trace statements are to be displayed. The programs specified must already be in debug mode.

Example

```
DSPTRC
```

This command displays all of the trace statement ranges that are currently specified in the default program of this debugging session. Also displayed are the program variables (but not their values) that are associated with the trace statements.

Additional Considerations

The display produced by the DSPTRC command has the following format:

```
 XX/XX/XX  XX:XX:XX     TRACE DESCRIPTION
ⒶProgram:   XXXXXXXXXX
  ⒷStatement range: XXXXXXXXXX XXXXXXXXXX
     When output variables: XXXXXXX
     Output start pos:  XXXXX  Length:  XXXXX  Format:  XXXXX
    ⒸVariable:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                 XXXXXXXXXXXXXXX
       Base:     XXXXXXXXXXXXXXXXXXXXXXXX
         .
         .
         .
```

This display contains, for each program specified on the PGM parameter, all of the trace ranges specified in the program and the names of all the variables whose values can optionally be recorded when the traced statements are executed.

For each of as many as 10 programs in debug mode, the *descriptions* of all its trace ranges (as many as 5 ranges per program) and the *names* of all the program variables (as many as 10 variables per statement) associated with each statement can be displayed. All of the information for one program is displayed before any information for the next program is shown.

For each program identified by name at Ⓐ, the following information for each trace range identified at Ⓑ is shown:

- The starting and ending statement identifiers (separated by a space) that identify one trace range in the program. If only one statement is to be traced in a group, only one identifier is shown.

- One of two values is displayed in the *output variables* attribute field, depending on whether the variables associated with the trace range are to be displayed every time the trace range is executed (*ALWAYS), or only when one or more of the values in the variables are changed as a result of the trace range being executed (*CHG). In either case, on the resulting trace data display, an * is placed just before the name of each variable whose value was changed during the trace execution.

For each trace range identified at Ⓑ, the following information for each program variable (identified at Ⓒ) associated with the range is displayed.

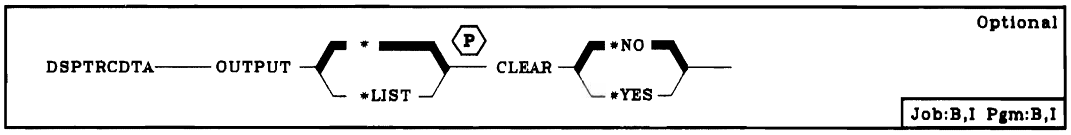
- The start, display length, and output format fields contain the common information about the variables identified on the following lines (beginning at Ⓒ) that are associated with the trace range. The values displayed in these three fields are those that were specified in the START, LEN, and OUTFMT parameters of the ADDTRC command.

- The names of each variable and the program pointer it is based on (if any) are then listed. If the variable is an array, and a subscript representing the positional element in the array is specified, the subscript is shown in parentheses.

DSPTRCDTA (Display Trace Data) Command

The Display Trace Data (DSPTRCDTA) command displays the resulting output of the trace just executed. All of the trace statements and associated program variables within the trace range are displayed. The display shows the sequence in which the traced statements or System/38 instruction numbers were executed and the name or MI ODV number and value of any program variables defined for the trace at each point in the sequence. Note that the display of variable values is controlled by the OUTVAR parameter on the ADDTRC command that defined the trace being displayed.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command*.



OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

CLEAR Parameter: Specifies whether or not the trace data of the trace just executed is to be cleared after it has been displayed.

*NO: The trace data is not to be cleared.

**YES:* The trace data is to be destroyed after it has been displayed.

Example

```
DSPTRCDTA
```

This command displays, at the display station that submitted the command, all of the trace data that was recorded from the trace operation just performed. All of the trace statements in the trace range and the values of the associated program variables are displayed. The trace data is not destroyed after it has been displayed because CLEAR(*NO) is assumed.

Additional Considerations

The display produced by the DSPTRCDTA command has the following format:

```
XX/XX/XX  XX:XX:XX     TRACE DATA DISPLAY
Stmt: XXXXXXXXXX XXXXX Pgm: XXXXXXXXXX Lvl: XXXXX XXXXXXXXXX
  Output start pos:  XXXXX  Length:    XXXXX  Format:  XXXXX
 XVariable:    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
               XXXXXXXXXXX
    Base:      XXXXXXXXXXXXXXXXXXXXXXXX
    Type:      XXXXXXXXX          Length:     XXXXX
    Dimension:        XXXXX
    'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX....'
       .
       .
       .
```

This display presents all of the information gathered for the trace range just executed, which is identified on line 2. The following trace information is shown:

- The statement identifier of the trace just executed in the program. The 10-character field displays the identifier of the HLL statement or the System/38 instruction that was traced, depending on how the trace was defined in an ADDTRC command. However, if traces were requested at both the HLL and System/38 levels for the same point in the program, the System/38 instruction number appears in the 5-character field following the 10-character field.

- The name of the program, and the invocation level, in which the trace was performed.

- The start, display length, and output format fields contain the common information about the variables (which are identified on the lines that follow) that are associated with the trace range. The values displayed in these three fields are those that were specified in the START, LEN, and OUTFMT parameters of the ADDTRC command.

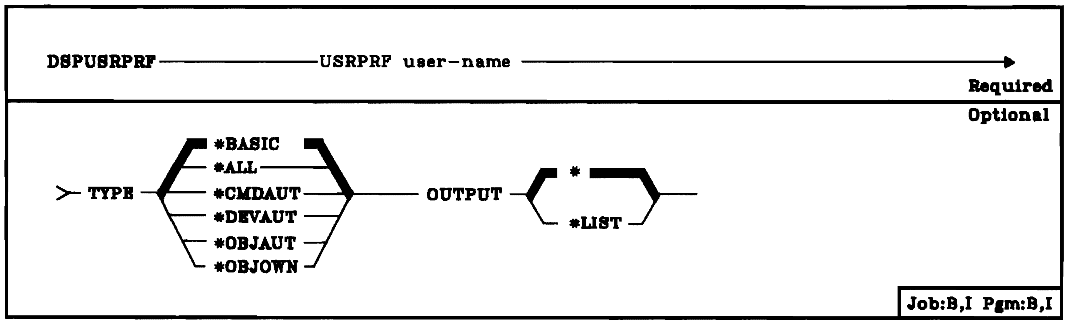
- The values of *all* the variables are displayed if at least one of them has its value changed during the execution of the traced statements. (If OUTVAR(*CHG) was specified on the ADDTRC command for this trace range, *no* variables are displayed if *none* of them changed value. If OUTVAR(*ALWAYS) was specified, all of them are displayed regardless of their values.) For each variable whose value did change during the execution of the traced statements, an * is placed just before the name of that variable. For each program variable associated with the trace range, the following fields are shown if they apply to that variable: the names of the variable and its based-on program pointer (if any), the subscript (shown in parentheses) associated with either the variable or pointer (if any), the type and length of the variable, its dimension (if in an array), and its actual value after the trace was executed.

  The information displayed for each variable in the trace is identical to that shown on the DSPPGMVAR command display. For a complete description of the above fields, refer to the DSPPGMVAR command description, under *Additional Considerations*, beginning with the description of area **B** on that display.

DSPUSRPRF (Display User Profile) Command

The Display User Profile (DSPUSRPRF) command displays the contents of a user profile. The user profile contains the user's operational limits for system resources; the names of the objects, commands, and devices that he has explicit authority to use (not including objects authorized by PUBAUT(*ALL)); and the names of the objects that he owns.

This command does not display the password, nor does it display information about objects that are authorized for public use. The password can be displayed by the DSPAUTUSR command. The use of the DSPUSRPRF command may be authorized for any user on the system, but the user must have operational rights for the user profile to be displayed.



USRPRF Parameter: Enter the name of the user profile to be displayed.

TYPE Parameter: Specifies the type(s) of information that can be displayed from the user profile. All, or one, of the following can be displayed:

- The basic portion of the user profile, describing the profile itself.
- Commands for which the user profile has explicit authority.
- Devices for which the user profile has explicit authority.
- All objects (including commands and devices) for which the user profile has some explicit authority and the rights of use associated with those objects.
- Objects that are owned by the user profile.

*BASIC: Displays the basic portion of the user profile, which includes:
- The user profile name and text describing the user profile
- The user profile's special rights (if any) and the scheduling priority limit
- The maximum amount of storage that can be allocated, and the amount currently being used, for the storage of owned objects
- The total number of objects both owned by the user profile and for which the user profile has some explicit authority
- The user profile's initial sign-on program

The attributes in the basic portion are further described under *Additional Considerations* at the end of this command description.

**ALL:* Displays all of the information in the user profile.

**CMDAUT:* Displays the control language commands that a user operating under this user profile is explicitly authorized to enter.

**DEVAUT:* Displays the system devices that a user operating under this user profile is explicitly authorized to use.

**OBJAUT:* Displays the names of the objects that a user operating under this user profile is explicitly authorized to use (not those authorized for public use), his rights of use for those objects, and the object type for those objects. (Commands and devices are included if *OBJAUT is specified.)

**OBJOWN:* Displays, for each object that the user profile owns, the object name, the type, and the library it is in.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

<u>*:</u> The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

```
DSPUSRPRF  USRPRF(THSMITH)
```

This command displays the basic portion of the user profile named THSMITH because TYPE(*BASIC) is assumed. The commands, devices, and objects that the user is authorized to use are not displayed. Because OUTPUT(*) is also assumed, the operational information will be displayed or printed, depending on where the command is submitted.

```
DSPUSRPRF  USRPRF(RTJOHNSON)  TYPE(*OBJOWN)  OUTPUT(*LIST)
```

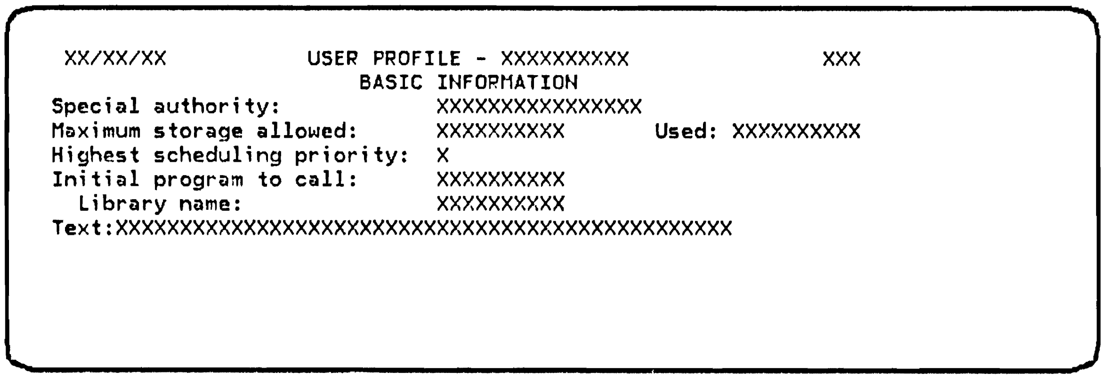
This command causes the list of objects that are owned by the user profile named RTJOHNSON to be printed. The list contains the object names, object types, and the names of the libraries that the objects are in.

Additional Considerations

Five separate display formats are used to display all the information contained in one user profile. One display (or set of displays in the same format) is used for each of the following types of information stored in the user profile: (1) basic information, (2) authorized commands, (3) authorized devices, (4) authorized objects (including commands and devices) and rights, and (5) owned objects. One or all of the five displays can be displayed, depending on what is specified in the TYPE parameter of the DSPUSRPRF command.

Note: The DSPUSRPRF function may be a long-running function (unless only TYPE(*BASIC) is specified), depending upon the number of objects the user profile owns and is authorized to use.

If TYPE(*ALL) is specified, all five sets of displays are presented in the order given in the previous paragraph. If TYPE(*BASIC) is specified, only one display, containing the basic portion of the user profile, is presented:



```
 XX/XX/XX           USER PROFILE - XXXXXXXXXX              XXX
                        BASIC INFORMATION
Special authority:          XXXXXXXXXXXXXXXX
Maximum storage allowed:    XXXXXXXXXX       Used: XXXXXXXXXX
Highest scheduling priority:  X
Initial program to call:    XXXXXXXXXX
  Library name:             XXXXXXXXXX
Text:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

The displayable attributes in the basic portion of the user profile are:

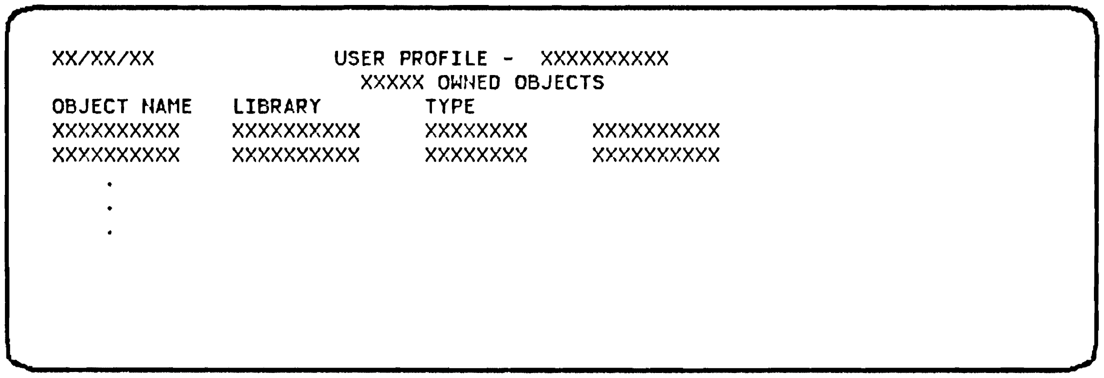
- User profile name (on the first line)
- Special rights granted to the user profile, if any
- Maximum amount of storage (in K-bytes) allowed for storing permanent objects owned by the user profile, and the amount of storage currently being used to store those objects
- Highest scheduling priority authorized for jobs that run under this user profile
- Name of the initial program to be invoked following successful sign-on, and the name of the library in which the program is stored
- User-defined text that describes the user profile

The third set of displays (also not shown) is presented when either *ALL or *DEVAUT is specified on the TYPE parameter. The names of all devices (as given in their device descriptions) that are explicitly authorized for use by the user profile are displayed in alphabetic order in as many as four columns. Any devices that are only authorized for public use are not included; they can be identified on the DSPOBJAUT command display if DEVD is specified as the object type.

The fourth set of displays (not shown here) is presented when either *ALL or *OBJAUT is specified on the TYPE parameter. The objects displayed are not owned by the user profile but are authorized for use by a user operating under its authority; any objects that are only explicitly authorized for public use are *not* included.

For each object authorized for use, the object's name, library, type, and all of the authorized rights are displayed. Each authorized object control right and data right is indicated by an X in the appropriate column. The format of the displayed rights is similar to that produced by the DSPOBJAUT command. Refer to *Additional Considerations* in that command's description for additional information on how the rights are displayed.

The fifth set of displays lists all of the objects that are owned by the user profile and is presented when either *ALL or *OBJOWN is specified on the TYPE parameter.

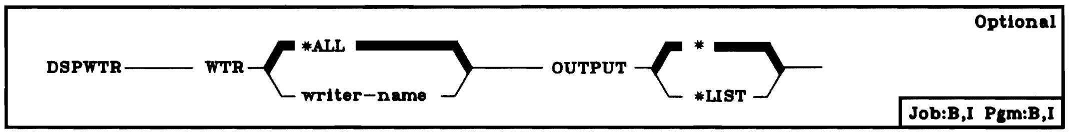


```
XX/XX/XX              USER PROFILE -  XXXXXXXXXX
                        XXXXX OWNED OBJECTS
OBJECT NAME   LIBRARY        TYPE
XXXXXXXXXX    XXXXXXXXXX     XXXXXXXX     XXXXXXXXXX
XXXXXXXXXX    XXXXXXXXXX     XXXXXXXX     XXXXXXXXXX
    .
    .
    .
```

The second line of the display shows the number of objects owned by this user profile. The objects are listed in the order in which they were created (not in alphabetic order). The object's name, the name of the library in which it is stored, and its type are displayed. Also, if the object is being deleted at the same time that this display is being generated, the value *DELETED is shown to the right of the object's type.

DSPWTR (Display Writer) Command

The Display Writer (DSPWTR) command displays or prints the attributes and status of the spooling writers that have been started in the system and that have not yet terminated. Detailed information about a specified writer can be displayed. Or, for all the spooling writers, the name, type, location for output, and status of each one can be displayed. Also, other display or controlling functions can be initiated directly from either type of writer display.



WTR Parameter: Specifies the name of the spooling writer for which detailed information is to be displayed, or specifies that the main attributes and status of *all* spooling writers are to be displayed.

*ALL: The attributes and the current status of all spooling writers are to be displayed.

writer-name: Enter the name of the spooling writer for which its detailed description is to be displayed.

OUTPUT Parameter: Specifies whether the output from the command is to be displayed at the requesting work station or listed with the job's spooled output on a printer. (Refer to Appendix A for an expanded description of the OUTPUT parameter, and to Appendix D for the name of the printer file used by this command.)

*: The output is to be displayed (if requested by an interactive job) or printed with the job's spooled output (if requested by a batch job).

**LIST:* The output is to be listed with the job's spooled output on a printer.

Examples

DSPWTR

This command displays the names of all spooling writers, their types, device files, and status.

DSPWTR WTR(DISKWRITE)

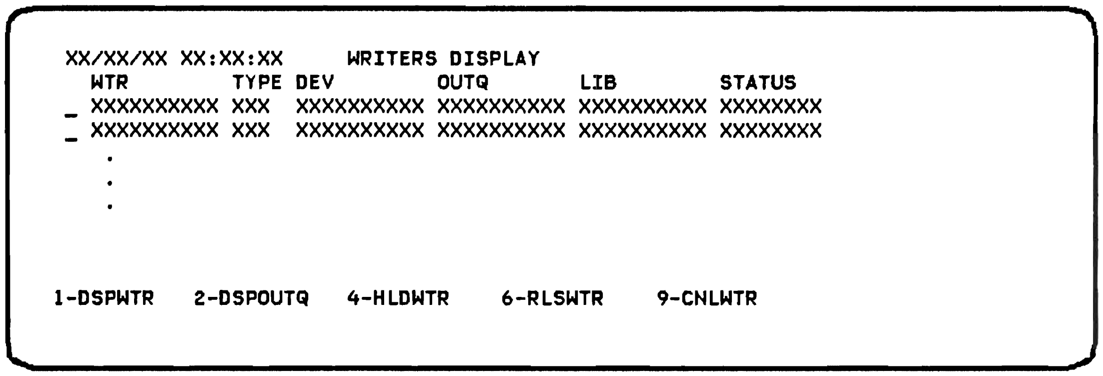
This command displays detailed information about the writer DISKWRITE. This information is described in the next section.

Additional Considerations

One of two display formats is produced by the DSPWTR command, depending on whether *ALL or a spooling writer name is specified in the WTR parameter.

Displaying All Writers

If WTR(*ALL) is specified or assumed, the *writers display* is presented:



```
 XX/XX/XX XX:XX:XX      WRITERS DISPLAY
   WTR        TYPE DEV        OUTQ       LIB        STATUS
 _ XXXXXXXXXX XXX  XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXX
 _ XXXXXXXXXX XXX  XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXX
    .
    .
    .


1-DSPWTR   2-DSPOUTQ   4-HLDWTR    6-RLSWTR    9-CNLWTR
```

The writers display shows, for each spooling writer in the system, the main attributes of the writer (that were specified on its corresponding start writer command) and its current status. For each writer, a separate line is used to show:

- Writer name: The name of the spooling writer, specified on the command that started it.
- Writer type (CRD, DKT, or PRT): The type of writer (card, diskette, or printer), determined by the command used to start it (STRCRDWTR, STRDKTWTR, or STRPRTWTR).
- Device name: The name of the device being used or to be used. (The writer can be currently active or still be on the job queue—the writer is a system job in itself, which runs in the QSPL subsystem.) The device name was specified in the DEV parameter of the start writer command.

- Output queue and library names: The qualified name (in two columns) of the output queue from which the writer is producing spooled output. The output queue was specified in the OUTQ parameter of the start writer command.

- Status of the writer: The current status of the writer is indicated by one of the following values:
  - JOBQ. The writer job is still on the job queue; although the writer has been started by a start command, it has not yet begun executing.
  - JOBQ/HLD. The writer job is currently held on the job queue for which it was started.
  - ACT. The writer job is active and is currently producing spooled output for other jobs in the system.
  - ACT/HLD. The writer has been active, but it is currently held because of a HLDWTR command; no output is being produced.

Five options are shown at the bottom of the display that are available to the user. An option number can be entered, in the input field preceding the writer name, on one or more lines to cause the associated function to be performed on the writer identified on that line. Options 1, 2, 4, 6, and 9 execute the DSPWTR, DSPOUTQ, HLDWTR, RLSWTR, and CNLWTR commands, respectively.

For a description of the function keys that can be used from this display, refer to *Additional Considerations* in the DSPOUTQ command description.

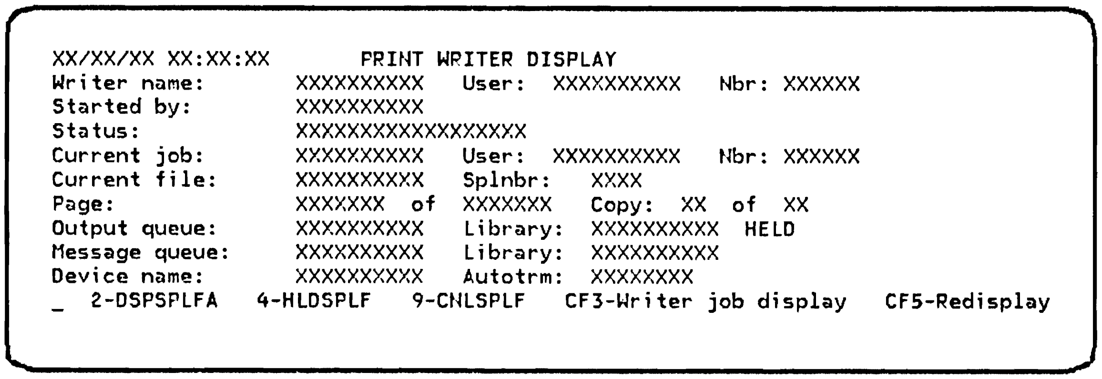
The information displayed by option 1 is the same as that displayed if the DSPWTR command had been entered with the name of the writer specified in the WTR parameter. The information displayed for a single writer is described in the following section.

Displaying a Specific Writer

If the WTR parameter specifies the name of a writer, the display for that type of writer is shown. All of the attributes for the writer are displayed in one of three formats, depending on the writer type:

CRD	Card writer display
DKT	Diskette writer display
PRT	Printer writer display

All of the attributes displayed on lines 2 through 7 are the same for all three types of writers. The title on line 1 indicates the type of writer being displayed. The *printer writer display* is shown here as an example of all three writer types.



```
XX/XX/XX XX:XX:XX        PRINT WRITER DISPLAY
Writer name:       XXXXXXXXXX   User:  XXXXXXXXXX   Nbr: XXXXXX
Started by:        XXXXXXXXXX
Status:            XXXXXXXXXXXXXXXXXX
Current job:       XXXXXXXXXX   User:  XXXXXXXXXX   Nbr: XXXXXX
Current file:      XXXXXXXXXX   Splnbr:   XXXX
Page:              XXXXXXX  of  XXXXXXX   Copy:  XX  of  XX
Output queue:      XXXXXXXXXX   Library:  XXXXXXXXXX  HELD
Message queue:     XXXXXXXXXX   Library:  XXXXXXXXXX
Device name:       XXXXXXXXXX   Autotrm:  XXXXXXXX
_  2-DSPSPLFA   4-HLDSPLF   9-CNLSPLF   CF3-Writer job display   CF5-Redisplay
```

Briefly, the information presented on the printer writer display, card writer display, or diskette writer display is as follows:

- Line 2 shows the fully qualified job name of the specified writer (which is a system job): writer name, user name, and job number.

- Line 3 shows the name of the user who started the writer.

- Line 4 displays one of the following for the writer's status:
  - JOBQ. The writer job is still on the job queue for which it was started; it has not begun executing.
  - JOBQ/HLD. The writer job is currently held on the job queue for which it was started.
  - ACT/WRITING. The writer is active and is producing output.
  - ACT/BETWEEN FILES. The writer is between spooled files.
  - ACT/BETWEEN COPIES. The writer is between copies of a spooled file.
  - ACT/WAITING FOR DATA. The writer is active and is waiting for more spooled output.

- Line 5 shows the fully qualified name of the current job whose spooled output is being produced from the output queue.

- Line 6 shows the name and number of the spooled file currently being produced as output by the writer.

- Line 7 shows information that is specifically related to the type of writer being displayed.

  Printer Writer: The page number of the page in the spooled file that is currently being printed, the total number of pages in the spooled file, the copy number of the copy currently being printed, and the total number of copies to be printed for the file.

  Card Writer: The card number of the record in the spooled file that is currently being punched, the total number of records in the spooled file, the copy number of the copy currently being punched, and the total number of copies to be punched for the file.

  Diskette Writer: The record number of the record in the spooled file that is currently being written to diskette, and the total number of records in the spooled file.

- Line 8 shows the qualified name of the output queue from which spooled output is being produced. If the output queue is being held, the value HELD is shown to the right of the library name.

- Line 9 shows the qualified name of the message queue used by this writer for operational messages.

- Line 10 shows the name of the device being used to produce the output, and whether or not the writer is to terminate automatically after all the output on the output queue has been produced or is to wait for more output. Also shown on line 10 is the number of the hopper being used for card output (for card writers), or the location of the diskettes used for diskette output (for diskette writers).

- Four options are shown on line 11 that are available to the user to control the spooled files being processed by the writer. For the first three options, an option number can be entered in the input field at the beginning of line 11 to cause the associated function to be performed on the writer being displayed. Options 2, 4, and 9 execute the DSPSPLFA, HLDSPLF, and CNLSPLF commands, respectively. For additional information about the use of the function keys, refer to *Additional Considerations* in the DSPOUTQ command description.

  Line 11 also shows that the CF3 key can be used to display additional information about the current writer job. CF3 invokes the function of the DSPJOB command and presents the DSPJOB menu, from which you can display any of the writer job attributes and spooled files. Refer to *Additional Considerations* in the DSPJOB command description.

  The CF5 key can be used to display the most current writer information, if the display information was updated since you last viewed the display.

DUPDKT (Duplicate Diskette) Command

The Duplicate Diskette (DUPDKT) command is used to copy the contents of a single diskette onto one or more diskettes. The VTOC and IPL record information and the data records can be copied to a diskette in a manual slot, or to one or more diskettes in a magazine. Diskette data in either the basic exchange data format or the save/restore format can be duplicated. The volume identifiers of the diskettes do not have to be unique.

If the diskettes being copied on do not have the same sector size as the diskettes being copied from, a message is sent to the system operator, allowing him to terminate the copy operation or to specify that the diskettes to be copied on are to be initialized to the same sector size and then copied on.

Deleted sectors on the input diskette will be ignored when duplicating. The address of the last data record in the file label containing the deleted sector will be adjusted on the output diskette, according to the number of sectors found deleted. Therefore, if the input diskette has deleted sectors, the output diskette will not be an exact copy.

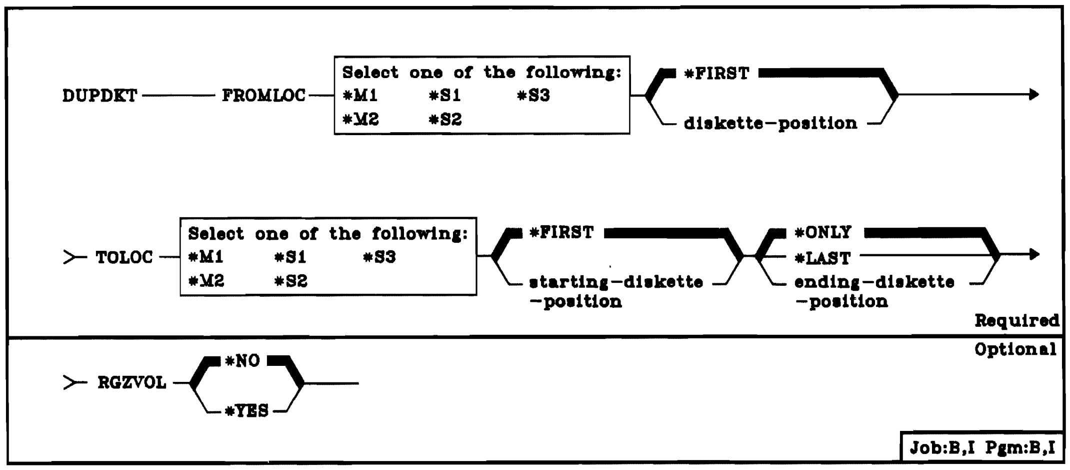
Diskettes that have an extended label area (a maximum of nine cylinders, in addition to cylinder 0, which are allocated as system area for data set labels), can be duplicated if the following restrictions are not violated:

- The RGZVOL option must be *NO.
- No deleted sectors can exist on the input diskette. If deleted sectors are found on the input diskette with an extended label area, a message will be signaled and the duplication function will be terminated.

Once a diskette has been copied, the RNMDKT command can be used to rename the duplicated diskettes so that they have unique volume identifiers.

Restriction: A diskette cannot be duplicated if it contains control records that indicate records have been non-sequentially relocated on it, or sequential sector addresses are not in consecutive, ascending order. If deleted sectors are found on a diskette with an extended label area, the diskette cannot be duplicated.

Note: Results when duplicating to or from a diskette with non-IBM standard labels are unpredictable. The diskette should be initialized by specifying CHECK(*NO) on the Initialize Diskette (INZDKT) command.



FROMLOC Parameter: Specifies the location in the magazine or slot that contains the diskette whose contents are to be duplicated. For magazines, two values are specified: one identifying the magazine and one identifying the diskette position within the magazine. Only one value is needed to identify the manual slot used.

Enter one of the following values to specify which magazine or slot is to be used: *M1, *M2, *S1, *S2, or *S3.

To specify the diskette position in the magazine, one of the following values can be entered:

FIRST: The diskette to be copied is in position 1 of the specified magazine.

diskette-position: Enter the number of the diskette position (1 through 10) in the specified magazine containing the diskette to be copied.

TOLOC Parameter: Specifies which diskette location(s) in the magazines or slots are to contain duplicates of the copied diskette. Three values can be specified: the magazine or manual slot to be used and, if a magazine is specified, the starting and ending diskette positions within the magazine. The first value must be specified; if the last two are not specified for a magazine, *FIRST and *ONLY are assumed by the system.

Unit Type and Location: The first of the three values in the TOLOC parameter specifies which unit and location are to be copied to. Enter one of the following values to specify the unit type and location: *M1, *M2, *S1, *S2, or *S3.

Starting Diskette Position: The second of the three values in the TOLOC parameter specifies which diskette position in the magazine is to be copied on first.

FIRST: The diskette in position 1 of the specified magazine is to be copied on first.

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that is to be copied on first.

Ending Diskette Position: The third of the three values in the TOLOC parameter specifies which diskette position in the magazine is to be copied on last.

*ONLY: Only the diskette position specified by the second value is to be copied on; a single duplicate is to be made.

**LAST:* The diskette in position 10 of the specified magazine is to be copied on last.

ending-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that is to be copied on last.

RGZVOL Parameter: Specifies whether the unused space between files is to be eliminated, to allow space for additional files to be written on the diskette.

*NO: The unused space is to remain as it exists on the input diskette. If no deleted sectors are found, the output diskette will be an exact copy of the input diskette.

**YES:* The unused space between files is to be moved to the end of the last file, making all files on the output diskette contiguous. If unused space between files currently exists, the files on the output diskette will reside at different physical locations.

Examples

```
DUPDKT  FROMLOC(*M1 3)  TOLOC(*M1 4)
```

This command duplicates the entire contents of the diskette in position 3 of magazine 1 onto the diskette in position 4 of the same magazine. Only one copy is made. If TOLOC (*M1 4 6) were specified, three copies would be made, on diskettes 4, 5, and 6 of magazine 1.

```
DUPDKT  FROMLOC(*S1)  TOLOC(*M1 1 10)
```

This command makes 10 copies of the diskette in manual slot 1. The 10 diskettes in magazine 1 are all to be duplicates of the diskette in slot 1.

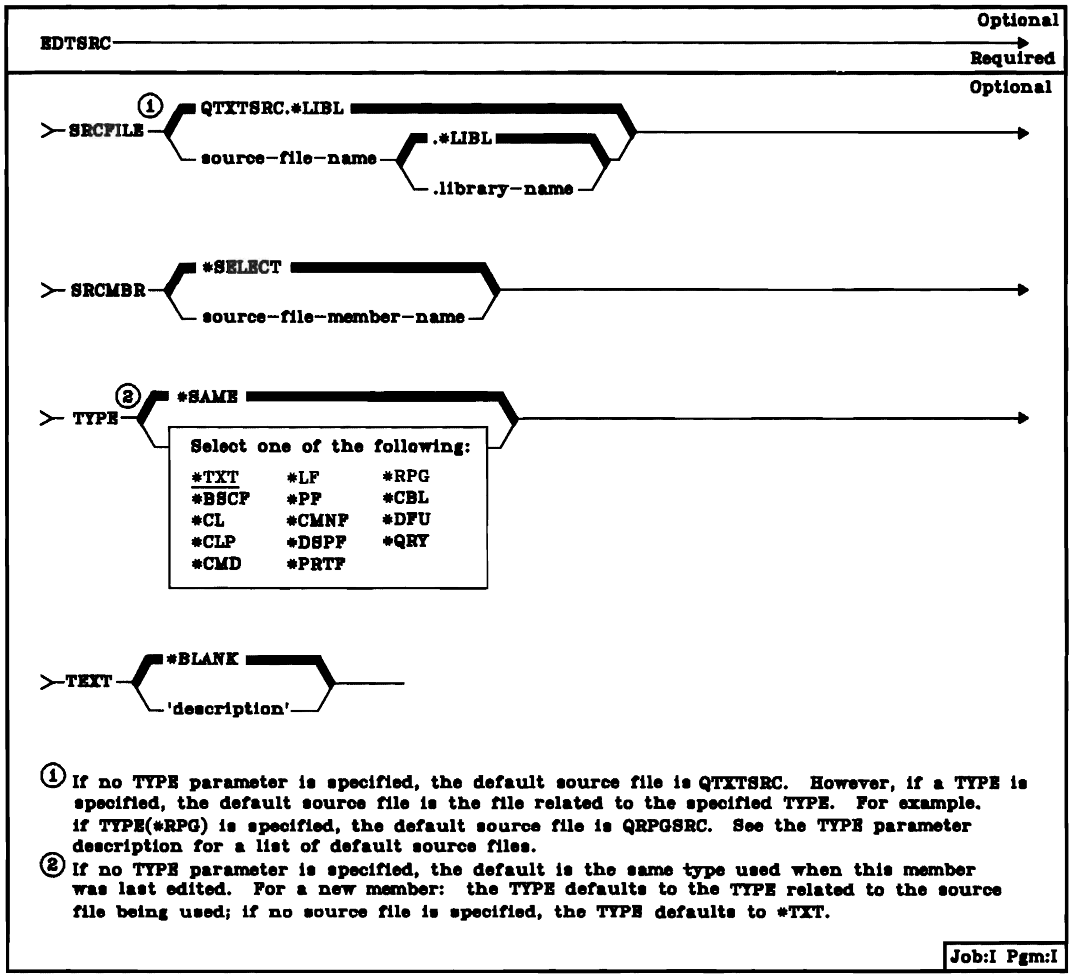
```
DUPDKT FROMLOC(*S2) TOLOC(*S1) RGZVOL(*YES)
```

This command copies the contents of the diskette in slot 2 onto the diskette in slot 1. The unused space of the data area is compressed at the end of the diskette. If unused space exists between files on the input diskette, the files on the output diskette will reside at different physical locations.

EDTSRC (Edit Source) Command

The Edit Source (EDTSRC) command is the sole command used with the Source Entry Utility (SEU); it is used to create, change, display, or remove a source member.

The Source Entry Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Source Entry Utility, refer to the *IBM System/38 SEU Reference Manual and User's Guide,* SC21-7722.



① If no TYPE parameter is specified, the default source file is QTXTSRC. However, if a TYPE is specified, the default source file is the file related to the specified TYPE. For example, if TYPE(*RPG) is specified, the default source file is QRPGSRC. See the TYPE parameter description for a list of default source files.

② If no TYPE parameter is specified, the default is the same type used when this member was last edited. For a new member: the TYPE defaults to the TYPE related to the source file being used; if no source file is specified, the TYPE defaults to *TXT.

SRCFILE Parameter: Specifies the name of the existing source file that contains the member to be edited or that will contain the member that is created.

QTXTSRC: If no SRCFILE parameter and no TYPE parameter are specified, the default source file is QTXTSRC. However, if a TYPE is specified, the default source file is the file related to the specified TYPE. For example, if TYPE(*RPG) is specified, the default soruce file is QRPGSRC. See the TYPE parameter description for a list of default source files.

source-file-name: Specifies the name of an existing source file. The source-file-name can be optionally qualified by the name of the library that contains the file. (If the source-file-name is not qualified, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of the source file member to be edited.

*SELECT: If no SRCMBR parameter is specified, SEU displays a list of all members in the file and allows you to choose the member to be edited or removed. On the list of members, you can key in the name of a member to be created or edited.

source-file-member-name: Specifies the name of the member to be created or edited.

TYPE Parameter: Specifies the type of source that will be edited. (You can change the TYPE during editing by specifying a TYPE on the services display.)

*SAME: If no TYPE parameter is specified, SEU uses the same type used when this member was last edited. For a new member, the TYPE defaults to the TYPE related to the source file being used; if no source file is specified, the TYPE defaults to *TXT.

type: The types that can be specified are shown in the following list. Also in the list is the default source file (the source file that will be used if a TYPE is specified, but no source file is specified) and the source language that the type is used for.

TYPE Specified	Default Source File	Source Language
*TXT	QTXTSRC	80-position free-form text
*CL	QCLSRC	Control Language (CL)
*CLP		CL Program
*CMD	QCMDSRC	CL Command Definition
*PF	QDDSSRC	DDS–Physical File
*LF		DDS–Logical File
*BSCF		DDS–Bisynchronous Communications File
*CMNF		DDS–Communications File
*DSPF		DDS–Display File
*PRTF		DDS–Printer File
*RPG	QRPGSRC	RPG III
*CBL	QCBLSRC	COBOL
*DFU	QUDSSRC	UDS–Data File Utility
*QRY		UDS–Query Utility

Note: On the EDTSRC command, the TYPE value begins with an asterisk (for example, *RPG or *DSPF). On the services display, member list, and programmer menu, the asterisk is not used (for example, RPG or DSPF).

TEXT Parameter: Specifies a character string to be stored in the text field of a member.

*BLANK: If no TEXT parameter is specified for a new member, blanks are put in the text field of the member. If no TEXT parameter is specified for an existing member, the text field of the member is not changed.

'description': Specifies a character string, enclosed in apostrophes, that describes the new member. A maximum of 50 characters can be specified.

Example

```
EDTSRC SRCMBR(MLG105D) TYPE(*DSPF)
```

This command displays the first records of the data description specifications for display device file source member MLG105D.

ELSE (Else) Command

The Else (ELSE) command is used with an IF command to specify another command that is to be conditionally executed. The ELSE command is executed only if the result of evaluating the logical expression on the preceding IF command is false; the ELSE command can specify a CL command to be executed for the false condition. (If the DO command is specified on the ELSE command, a do group can be executed.) If the result of executing the IF command is true, the ELSE command and those commands associated with it are not executed.

An ELSE command does not have to follow each IF command, but each ELSE command that is coded must have an associated IF command preceding it. If nested levels of IF commands are present, a given ELSE is always matched with the innermost IF command that has not already been matched with another ELSE command. Although the ELSE command is optional, coding all of the matching ELSE commands helps clarify where all of the nesting levels start and end.

Restrictions: The ELSE command is valid only in a CL program. It must have an associated IF command preceding it.

```
                                                              Optional
ELSE ——— CMD CL–command ——
                                                              Pgm:B,I
```

CMD Parameter: Specifies the command or commands (in a do group) to be executed if the result of evaluating the expression on the corresponding IF command is false. If the command specified in this parameter is a DO command, the entire do group is considered to be the command specified by the parameter. If no command is specified, no action is taken for a false condition.

If the command specified by the CMD keyword is not coded on the same line as the keyword, the left parenthesis following CMD must be coded on the same line, followed by a + or - to show continuation. The command and the right parenthesis can then be coded on the next line. For example:

```
ELSE  CMD(     +
      GOTO C)
```

If any part of the command continues on the next line, a continuation character (+ or -) must be specified.

If a DO command is specified, only the DO command (not the entire do group) is within the parentheses. For example:

```
ELSE CMD(DO)
     CMD1
     CMD2
       •
       •
     ENDDO
```

The following commands, although valid in CL programs, cannot be specified on the ELSE command: ENDDO, ENDPGM, MONMSG, or another ELSE command. (Of course, neither can the PGM, DCL, DCLF, or DCLDTAARA commands.) In addition, the MONMSG command can not be specified as the next command after the ELSE command.

Examples

```
IF  (&A *GT &B) THEN(CHGVAR  VAR(&A)  VALUE(&B))
ELSE  (CHGVAR &B &A)
```

If the value of &A is greater than the value of &B, &A is set equal to &B. If &A is less than or equal to &B, the test result is false. The CHGVAR command on the ELSE command is executed, and the value of &B is set to the same value as &A. (Refer to the CHGVAR, or Change Variable, command for the description of the command and its parameters.)

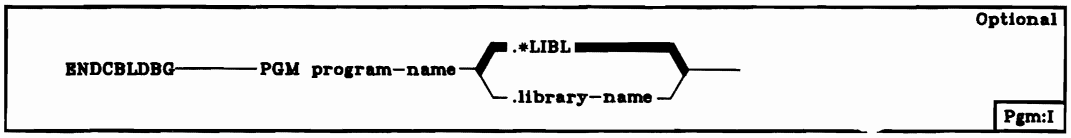
```
IF  COND(&A *EQ &B) +
   THEN(IF  (&C *EQ &D) +
        THEN(IF  (&C *EQ &F)  THEN(DO)))
                           CMD1
                           CMD2
                             •
                             •
                             •
                           ENDDO
              ELSE  CMDX
        ELSE  CMDY
ELSE  DO
```

This example shows the use of nested levels of IF commands where an ELSE command is associated with each IF. The use of the ELSE commands makes the nested levels of IF commands easier to identify.

ENDCBLDBG (End COBOL Debug) Command

The End COBOL Debug (ENDCBLDBG) command de-activates COBOL source language debugging statements. Specifying debugging mode in the COBOL program source code activates all debugging sections and debugging lines compiled into the program. To de-activate the debugging statements, you must have previously executed the Enter COBOL Debug (ENTCBLDBG) command.

The COBOL high-level language is part of the *IBM System/38 COBOL Program Product*, Program 5714-CB1. For more information, refer to the *IBM System/38 COBOL Reference Manual and Programmer's Guide*, SC21-7718.



PGM Parameter: Specifies the qualified name of the debugged program. (If no library name is specified, *LIBL is used to find the program.)

Example

```
ENDCBLDBG PGM(PROG7.JAMES1)
```

This command de-activates the debugging statements compiled into the program PROG7 in library JAMES1.

ENDDBG (End Debug) Command

The End Debug (ENDDBG) command terminates debug mode for a job and removes all the breakpoints and traces that had been set in the job's programs. This command cannot be entered when one or more of the programs in the invocation stack are stopped at a breakpoint. All breakpoints must be terminated by Resume Breakpoint (RSMBKP) or Cancel Request (CNLRQS) commands. If the command is entered interactively, the ENDDBG command causes the system to return the work station to the normal command entry mode. After this command has been entered, all data base files in production libraries can be updated normally.

When debug mode has been terminated, programs that were in debug mode are in normal (production) mode. Also, any breakpoints or traces are removed, and all trace data is cleared from the system.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command*.

```
ENDDBG ———
                                                    Job:B,I Pgm:B,I
```

There are no parameters for this command.

Example

```
ENDDBG
```

Assuming that this command is entered interactively and no program in the invocation stack is stopped at a breakpoint, debug mode for the job is terminated. The work station can then be used for normal command entry. In a batch job, processing continues with the next command.

ENDDO (End Do) Command

The End Do (ENDDO) command is used with the DO command to identify a group of commands that are to be executed together as a group. The ENDDO command specifies the end of the do group that began with an associated DO command. The ENDDO command must be specified after the last command in the do group.

When do groups are nested, each group must have its own ENDDO command at its end. Every ENDDO must be associated with a DO; if extraneous ENDDO commands occur in the CL program source, a message is issued and the program is not created.

Restriction: This command is valid only within a CL program.

```
ENDDO ———
                                                              Pgm:B,I
```

There are no parameters for this command.

Examples

See the examples for the DO and IF commands.

ENDINP (End Input) Command

The End Input (//ENDINP) command is a delimiter that indicates the end of the input in a batch input stream. The //ENDINP command also can delimit an inline data file if the command is detected while the inline file is being processed. If the inline file is using nondefault ending characters, the ENDINP command will be imbedded without being recognized.

Operational Considerations: For card devices, the ENDINP command should be followed by a card with /* in columns 1 and 2 to insure that the ENDINP command is read.

Restrictions: This command cannot be entered at a work station. The ENDINP command must be preceded by two slashes (//) in positions 1 and 2 of the data record: //ENDINP. Blanks can separate the slashes from the command name (// ENDINP).

```
                                                          Required
//ENDINP
                                                     Job:B Pgm:B
```

Example

```
//JOB
  •
  •
  •
//DATA
  •
  •
  •
//ENDINP
```

The ENDINP command indicates the end of the inline data file by indicating the end of all input.

ENDJOB (End Job) Command

The End Job (//ENDJOB) command is a delimiter that indicates the end of a job in a batch job stream. The //ENDJOB command also can delimit an inline data file if the command is detected while the inline file is being processed. Other conditions that can indicate the end of a job are:

- JOB command
- An end-of-file condition while an input stream is being processed

Restrictions: This command cannot be entered at a work station. The ENDJOB command must be preceded by two slashes (//) in positions 1 and 2 of the data record: //ENDJOB. Blanks can separate the slashes from the command name (// ENDJOB).

```
//ENDJOB ——
                                                  Job:B
```

There are no parameters for this command.

Example

```
//ENDJOB
```

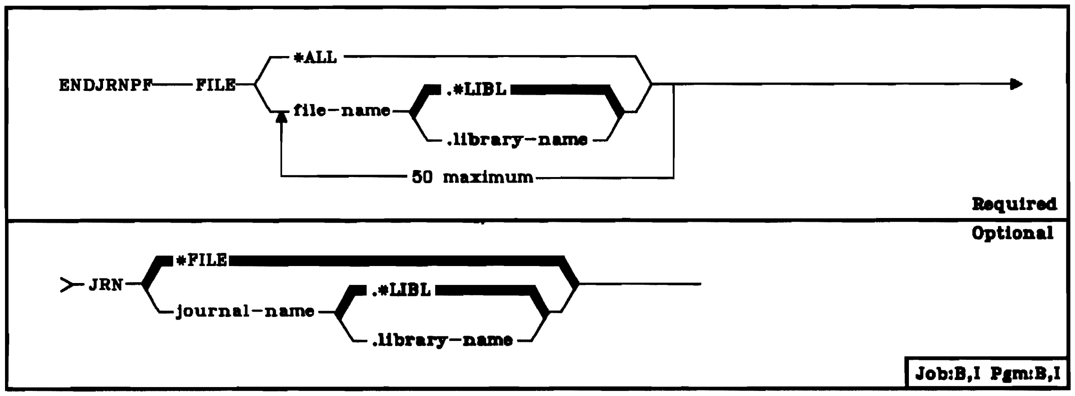
This command indicates the end of a job that began with a JOB command.

ENDJRNPF (End Journaling Physical File Changes) Command

The End Journaling Physical File Changes (ENDJRNPF) command is used to end journaling of changes for a specific physical file and all of its members.

All files currently being journaled to a specific journal may also have journaling stopped.

Restrictions: Members within the files specified on the command cannot be in use for any reason at the time the command is executed. The file names specified must be those that are applicable after overrides have been applied. If FILE(*ALL) is specified, a journal name must be specified. If a journal name and a list of file names are specified, then all files must be currently journaled to the indicated journal.



FILE Parameter: Specifies which physical files are to stop having their changes journaled.

**ALL:* Specifies that all files currently being journaled to the indicated journal will stop having their changes journaled.

file-name: Specifies the qualified name of the physical file that is to stop having its changes journaled. (If no library qualifier is specified, *LIBL is used to find the file.)

JRN Parameter: Specifies the journal to which the indicated files are currently being journaled.

<u>*FILE:</u> Specifies that the journal is to be determined by the system from the specified file names.

journal-name: Specifies the qualified name of the journal to which the files are currently being journaled. (If no library qualifier is specified, *LIBL is used to find the journal.)

Examples

```
ENDJRNPF  FILE(MYFILE.MYLIB)
```

The above command causes the journaling of all changes to all members of file MYFILE in library MYLIB to cease. Changes made subsequent to the execution of the command will not be journaled.

ENDLOG (End Logging) Command

The End Logging (ENDLOG) command ends the logging of all data base operations within the routing step. The logging began when a LOGDBF command was executed in the routing step before the files were opened.

Restriction: This command terminates with an escape message if any files being logged in the routing step are still open, or if logging has already been stopped within the routing step.

```
ENDLOG ———

                                                        Job:B,I Pgm:B,I
```

There are no parameters for this command.

Example

```
ENDLOG
```

This command stops the logging of data base operations that were started when the LOGDBF command was executed earlier in the routing step.

ENDPGM (End Program) Command

The End Program (ENDPGM) command specifies the end of a CL program. When the command is executed, it performs the same function as a RETURN command. That is, control is returned to the command immediately following the CALL command in the last calling invocation.

The ENDPGM command is not required at the end of a CL program. If the last statement in a CL program source file is reached and no ENDPGM command is found, an ENDPGM command is assumed by the compiler.

Restriction: This command is valid only within a CL program.

```
ENDPGM ───

                                                        Pgm:B,I
```

There are no parameters for this command.

Example

```
PGM
 •
 •
 •
ENDPGM
```

This program is identified by a PGM command that contains no parameters and is ended by the ENDPGM command.

ENDSRV (End Service) Command

The End Service (ENDSRV) command terminates remote job service mode. This command terminates the service mode that began when the SRVJOB (Service Job) command was entered.

Restriction: If tracing is active in the serviced job when this command is entered, the remote service mode is *not* ended.

```
ENDSRV ———

                                                    Job:B,I Pgm:B,I
```

There are no parameters for the ENDSRV command.

Example

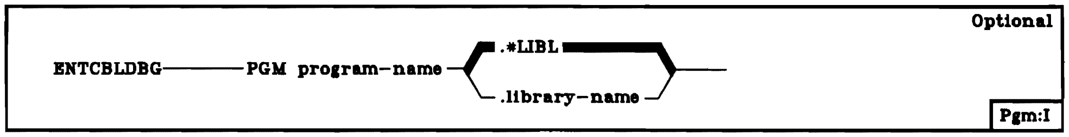
```
ENDSRV
```

This command terminates the service mode of the job currently being serviced.

ENTCBLDBG (Enter COBOL Debug) Command

The Enter COBOL Debug (ENTCBLDBG) command activates COBOL source language debugging statements. Specifying debugging mode in the COBOL program source code activates all debugging sections and debugging lines compiled into the program. To de-activate the debugging statements, you must execute the End COBOL Debug (ENDCBLDBG) command.

The COBOL high-level language is part of the *IBM System/38 COBOL Program Product*, Program 5714-CB1. For more information, refer to the *IBM System/38 COBOL Reference Manual and Programmer's Guide*, SC21-7718.



PGM Parameter: Specifies the qualified name of the program that is to be debugged. (If no library name is specified, *LIBL is used to find the program.)

Example

```
ENTCBLDBG PGM(PROG7.JAMES1)
```

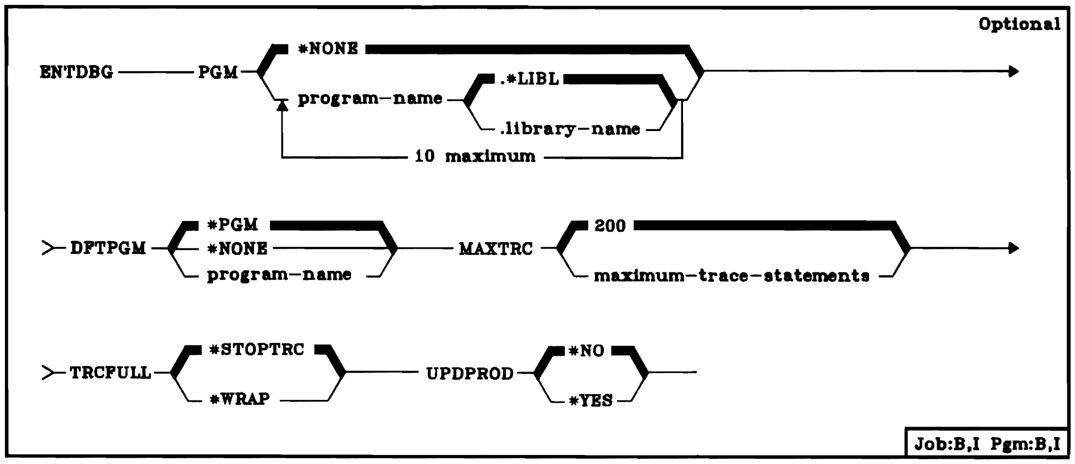
This command activates the debugging statements compiled into the program PROG7 in library JAMES1.

ENTDBG (Enter Debug) Command

The Enter Debug (ENTDBG) command puts a job in debug mode and, optionally, adds as many as 10 programs to the debug environment. It also specifies certain attributes of the debug environment. For example, it can specify whether data base files in production libraries can be updated while in debug mode, or whether breakpoints are to be logged in the job log.

The CHGDBG command can be used later in the job to change the attributes of the debug environment. Also, programs can be added to or removed from the debugging environment if they are specified in the ADDPGM or RMVPGM commands.

Restriction: This command is not valid in debug mode. To end debug mode, refer to *ENDDBG (End Debug) Command*.



PGM Parameter: Specifies the qualified names of one or more programs (if any) to be debugged in the job. (If no library qualifier is given for a program, *LIBL is used to find the program.) Before a program can be debugged, its name must be specified here or in the ADDPGM command.

*NONE: No program names are to be specified at the beginning of debug mode. The ADDPGM command can be used to add programs later.

qualified-program-name: Enter the qualified names of one or more programs that are to be debugged. (If no library qualifier is given for a program, *LIBL is used to find the program.) The program names specified, must always be unique within the job.

DFTPGM Parameter: Specifies the name of the program that is to be the default program in the job's debugging environment. The program specified here is used as the default program for any of the other debug commands that specify *DFTPGM on their PGM parameter.

*PGM: The program named in the PGM parameter of *this* command is to be the default program for the job's debugging environment. If PGM has more than one program name specified, the first program named in the list is the default program. If PGM(*NONE) is specified or is the default, DFTPGM(*NONE) is also assumed when *PGM is specified here.

**NONE:* No program is to be specified as the default program. Either the default program must be named later in the ADDPGM or CHGDBG commands or *DFTPGM cannot be the specified value (or taken as the default) on any of the other debug commands.

program-name: Enter the simple name of the program that is to be the default program for the job's debugging environment. The same name (in qualified form) must also be specified in the PGM parameter of this command.

MAXTRC Parameter: Specifies the maximum number of trace statements that the system can put into the job's trace file before either terminating tracing or wrapping around (overlaying) on the trace file. When the trace file contains the maximum specified, the system performs the action specified in the TRCFULL parameter.

200: Two hundred trace statements can be put into the trace file before tracing is terminated or wrapping on the file occurs.

maximum-trace-statements: Enter the maximum number of trace statements that can be in the trace file.

TRCFULL Parameter: Specifies what is to happen when the job's trace file is full (contains the maximum number of trace statements specified by the MAXTRC parameter).

*STOPTRC: In batch mode, tracing stops but the program continues to execute. In interactive mode, a breakpoint occurs on the next trace statement encountered, and control is given to the user.

**WRAP:* The trace file is overlaid with new trace statements, wrapping from the beginning of the file. The program continues to execute until finished. The trace file will never have more than the maximum specified statements, and they will be the more recently recorded statements. This is useful if only the last occurrences of the traced statements are needed.

UPDPROD Parameter: Specifies whether or not data base files in a production library can be opened for updating records, or for adding new records, while the job is in debug mode. If not, the files must be copied into a test library before an attempt is made to execute a program that uses the files.

*NO: Data base files in production libraries can *not* be updated in debug mode. However, a data base file can be opened for reading only. This protects data base files from erroneous updates while a program is being debugged.

**YES:* Data base files in production libraries can be updated while the job is in debug mode.

Example

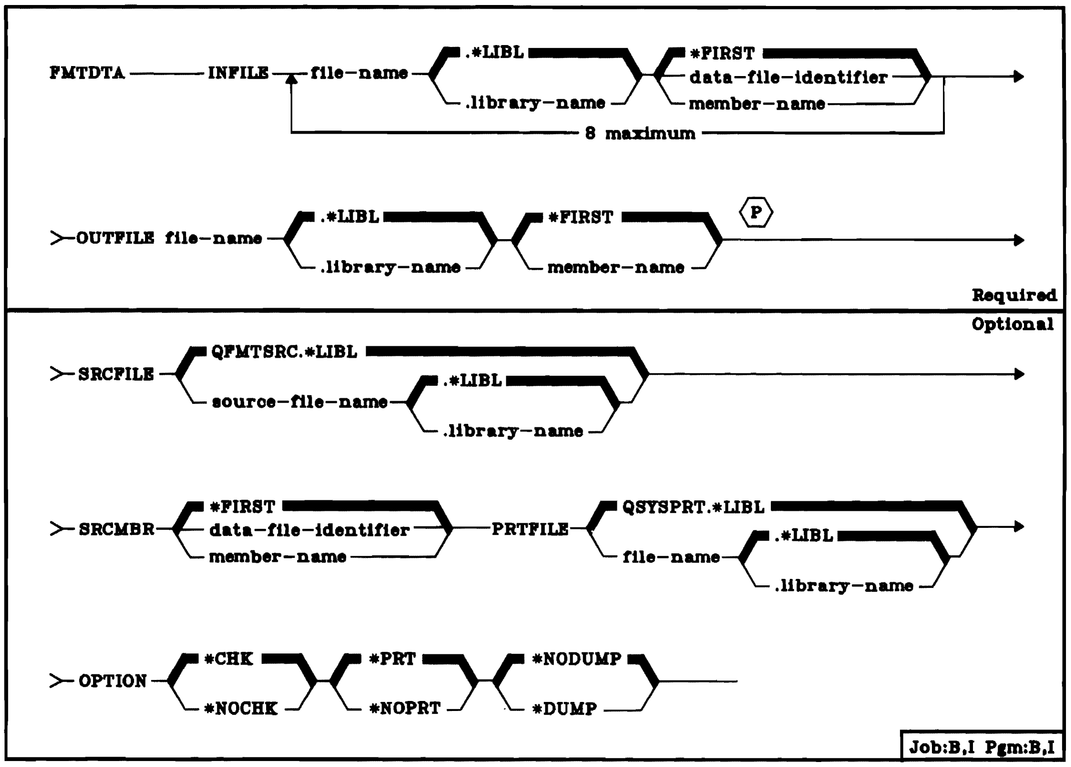
```
ENTDBG PGM(PAYROLL.TESTLIB)
```

This command sets up debug mode to debug the program PAYROLL, which is in the test library TESTLIB. Breakpoint information is put into the job log if breakpoints are used. If tracing is used, a maximum of 200 trace statements can be put in the trace file for the PAYROLL program before tracing stops. Any data base files updated by the PAYROLL program must be in a test library.

FMTDTA (Format Data) Command

The Format Data (FMTDTA) command invokes the *IBM System/38 Conversion Reformat Utility,* Program 5714-CV2. The reformat utility helps you to run System/3-style sort programs on System/38.

For more information on the Conversion Reformat Utility, refer to the *IBM System/38 Conversion Reformat Utility Reference Manual,* SC21-7780.



INFILE Parameter: Specifies up to eight qualified names for files that are to be used as input. For data base files, one member name can be specified for each file name. For diskette files, the diskette identifier can be specified for each device file name.

One of the following can be specified for the second value:

*FIRST: The first member in the file is to be used as input.

data-file-identifier: For diskette files, enter one data file identifier per diskette device file name specified. If more than one diskette data file is to be processed for a device file name, the device file name should be specified as many times as required.

member-name: For data base files, enter one member name per data base file name specified. If more than one member of the same data base file is to be processed, the data base file name should be specified as many times as required.

OUTFILE Parameter: Specifies the qualified name of the file and the name of the member to be used for output. Both the file and member must exist before being named in this parameter.

One of the following can be specified for the second value:

FIRST: The first member in the file is to be used for output.

member-name: Enter the name of the member in the output file to be used for output.

SRCFILE Parameter: Specifies the name of the source file containing the reformat specification statement set to be executed. The source file may be a device or data base file, and it must have the attributes of a source file.

QFMTSRC.*LIBL: Specifies that the IBM-supplied source file QFMTSRC contains the reformat specifications.

source-file-name: Enter the name of the source file that contains the reformat specifications. (If no library qualifier is specified, *LIBL is used to find the file.)

SRCMBR Parameter: Specifies the name of the source file member containing the reformat specification statement set to be executed. The source file may be a device or data base file, and it must have the attributes of a source file.

*FIRST: The first member of the source file containing the reformat specification statement set is to be executed.

data-file-identifier: Enter the name of the diskette data file identifier that contains the reformat specification statements, if the data file resides on diskette.

member-name: Enter the name of the member of the source file containing the reformat specification statement set that is to be executed.

PRTFILE Parameter: Specifies the name of the printer device file to which the print data is to be sent.

QSYSPRT.*LIBL: The data is to be printed by the system printer.

qualified-print-file-name: Enter the name of the printer device file that is to print the data. (If no library qualifier is specified, *LIBL is used to locate the device file.)

OPTION Parameter: Specifies the sequence checking and printing options to be used while the reformat utility is executing.

*CHK: The reformat specification statements are to be sequence checked.

**NOCHK:* The reformat specification statements are not to be sequence checked.

*PRT: The reformat specification statements and any error or informational messages are to be printed.

**NOPRT:* The reformat specification statements and error or informational messages are not to be printed.

*NODUMP: The internal tables used for problem determination are not to be printed.

**DUMP:* The internal tables used for problem determination are to be printed.

Example

```
FMTDTA  INFILE((FILEA.JONES.DATA1)(FILEA.JONES.DATA2)) +
          OUTFILE(FILEA.JONES) SRCFILE(SORT1.JONES)
```

This command sorts the data in the members DATA1 and DATA2 of file FILEA in library JONES and writes the sorted data back into file FILEA in library JONES. The reformat specification statement set is contained in the source file program SORT1. Default values are taken for all other parameters.

FMTRJEDTA (Format RJE Data) Command

The Format RJE Data (FMTRJEDTA) command converts data written in communications format, that is, compressed to a System/38 data base file to decompressed data written to either another data base file or to a System/38 printer device file. The format utility is executed by issuing the Format RJE Data (FMTRJEDTA) command.

If any file in the list of input files causes an error condition, the command issues an error message to the user and skips to the next input file in the list.

Note: If a forms control table (FCT) is specified in this command, the PGM parameter in the FCT entries is ignored.

Restrictions: To use this command, you must have read rights for the input file and the library in which the input file is stored.

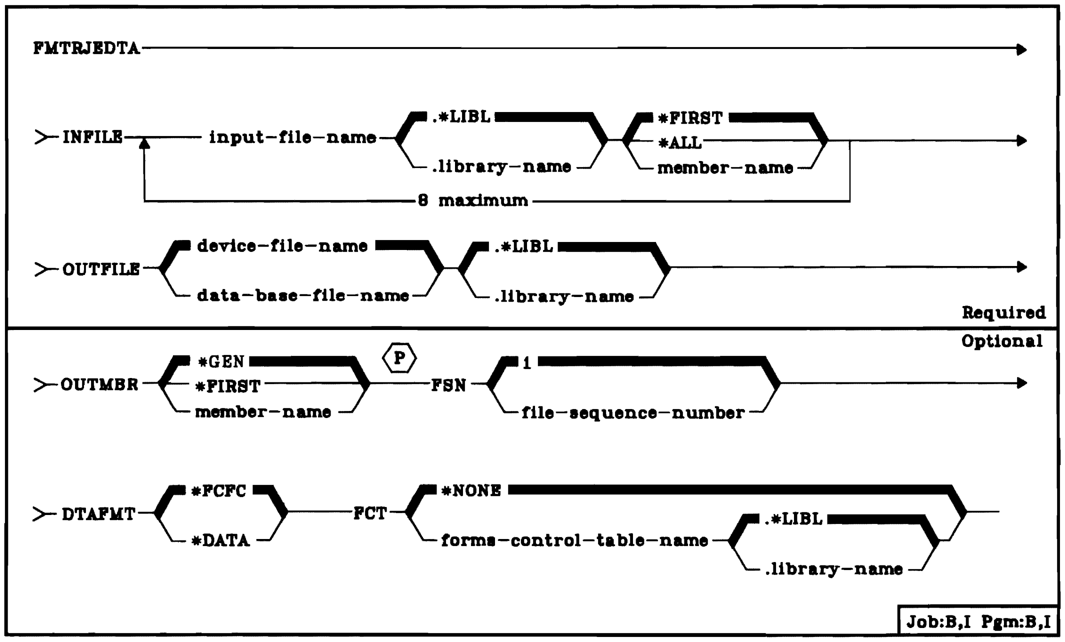
If a data base file name is specified for the output file, you must have add and update rights for that data base file.

If MBR(*GEN) is specified in the FCT entry, you must have object management and operational rights for the data base file as well as read rights for the library in which the data base file is stored.

If a device file name (printer device only) is specified for the output file in this command, you must have operational rights for that device file and read rights for the library in which the device file is stored.

If a FCT is specified in this command, you must have operational rights for the FCT and read rights for the library in which the FCT is stored.

The Format RJE Data (FMTRJEDTA) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



INFILE Parameter: Specifies up to eight qualified names for RJEF-created communications format input files. (If no library qualifier is given, *LIBL is used to find the communications format input file.) If the input file does not contain data in communications format, an error message is issued to the user.

<u>*FIRST:</u> Only the first member of the data base file is to be formatted.

**ALL:* All the members of the data base file are to be formatted.

member-name: Enter the name of the data base file member(s) that is to be formatted.

OUTFILE Parameter: Specifies the qualified name of the output file that is to receive the formatted RJEF data. This file is used only when no matching FCT entry is found.

device-file-name: Enter the qualified name of the program-described printer device file to receive the RJEF formatted data. The device file must be defined to have only a single record format with a single field. (If no library qualifier is given, *LIBL is used to find the device file.)

data-base-file-name: Enter the qualified name of the System/38 physical data base file to receive the RJEF formatted data. (If no library qualifier is given, *LIBL is used to find the data base file.)

OUTMBR Parameter: Specifies the data base file member that the formatted RJEF output data is to be directed if a data base file was specified in the OUTFILE parameter of this command.

<u>*GEN:</u> RJEF creates a member name as follows:

Afffffccc or Bffffffccc

Where:

A	=	file member names beginning with the character A contain print data.
B	=	file member names beginning with the character B contain punch data.
ffffff	=	first six characters of the forms name specified in the FCT or received from the host system. **Note**: Only characters that are valid in a System/38 name are valid in the forms type used to generate data base file member names.
ccc	=	three-digit sequence value controlled by the RJEF session to maintain member uniqueness (refer also to the FSN parameter description of this command).

If a member with this name already exists in the data base file, the three-digit sequence value is incremented by one and another attempt is made to create a member. Incrementing of the sequence value continues until a unique name is generated and a member is created or until all 1000 possibilities have been exhausted without creating a member. If no member is created, an error message is issued.

**FIRST:* The output is to be directed to the first member of the data base file.

member-name: Enter the name of the data base file member to which output is to be directed.

FSN Parameter: Specifies the initial three-digit file sequence number to be used when creating data base file member names. This parameter is ignored unless MBR(*GEN) is specified for this command.

<u>1:</u> The initial file sequence number to be used is one.

file-sequence-number: Enter the initial three-digit file sequence number to be used. Leading zeros are not required for sequence numbers less than 100.

DTAFMT Parameter: Specifies the format of the output data.

*FCFC: The output data is to be in the FCFC data format, with the first character of every record being the ANSI forms control character.

***DATA**: The output data is to be in the normal data format (that is, the first character does not contain forms control information).

FCT Parameter: Specifies the qualified name of the FCT to be used while formatting the data in the RJEF-created data base file member(s).

*NONE: No FCT is to be used while formatting the data.

forms-control-table-name: Enter the qualified name of the FCT that is to be used while formatting the data. (If no library qualifier is given, *LIBL is used to find the FCT.)

Example

```
FMTRJEDTA INFILE(MEDHISTORY.MEDLIB *ALL) +
      OUTFILE(PRTMED.MEDLIB)
```

This command will format all members of file MEDHISTORY in library MEDLIB. All formatted members are written to printer device file PRTMED in library MEDLIB.

```
FMTRJEDTA INFILE((MEDHISTORY.MEDLIB *ALL) (MEDLIST.MEDLIB JAN)) +
      OUTFILE(MEDDB.MEDLIB) +
      MBR(*GEN) +
      DTAFMT(*DATA)
```

This command will format two files. All members of the data base file named MEDHISTORY in library MEDLIB, as well as the member named JAN in data base file called MEDLIST, are formatted. All input files are formatted and written to data base file MEDDB in library MEDLIB. RJEF will create a new data base member for each file formatted. The data format is to be *DATA (no forms control characters embedded in the data).

GOTO (Go To) Command

The Go To (GOTO) command is used in CL programs for branching from one part of the program to another. The branch is to the label on another command that is specified on the GOTO command. The branch can be either forward or backward, but it must not go to a point outside the program.

Restriction: This command is valid only within a CL program.

```
                    ①                                                    Required
GOTO ──────── CMDLBL command-label ────

①A variable cannot be coded on this parameter.                          Pgm:B,I
```

CMDLBL Parameter: Specifies the name of the label on the command to which control is to be transferred; when the GOTO command is executed, control is transferred to the command on which the label is specified. The command with the label is then executed. If the command with the label is not an executable command (for example, if it is a DCL command), control is transferred to the first executable command following the command with the label. The label must be within the same program as the GOTO command. A CL variable name cannot be used to specify the label name.

Examples

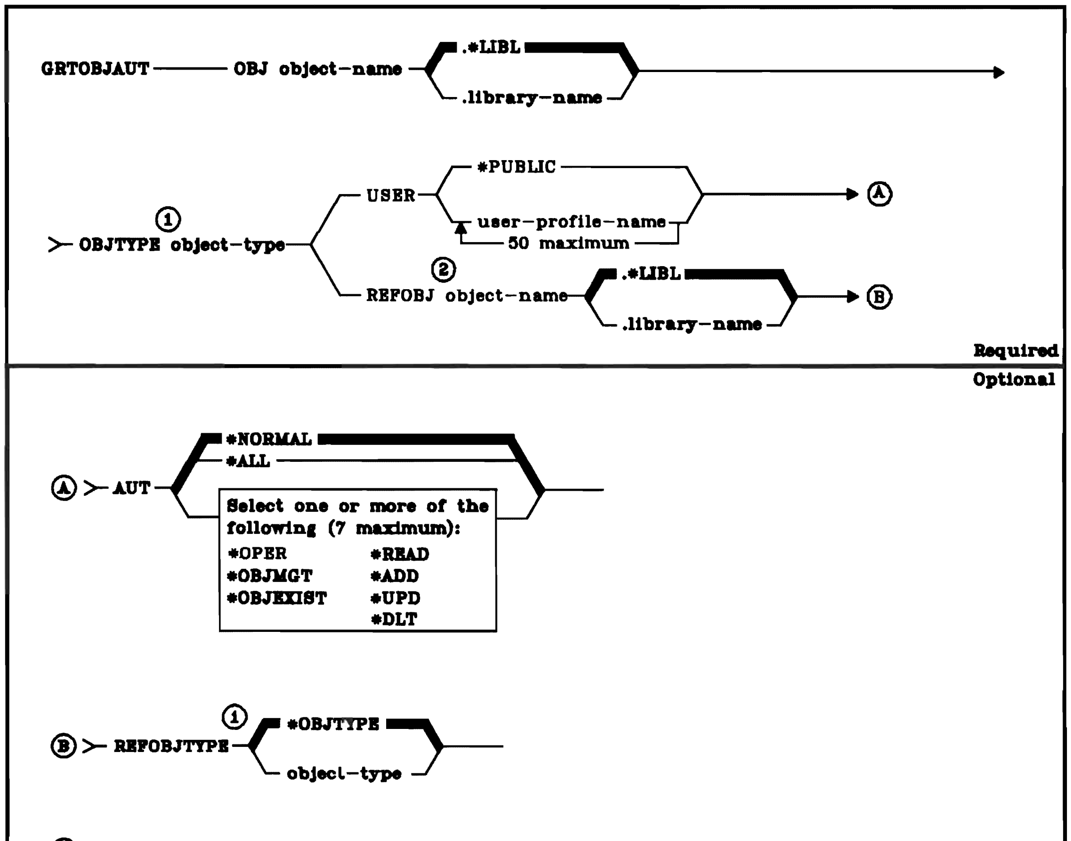
```
LOOP: CHGVAR &A (&A + 1)
      IF (&A *LT 30) THEN(GOTO LOOP)
```

The CHGVAR command is executed to increase the value of &A by 1 until &A is equal to or exceeds 30. The GOTO command is executed each time that the IF command tests the expression and the result is true; the GOTO command following the THEN parameter causes the program to branch back to the label LOOP on the CHGVAR command. (Refer to the descriptions of the CHGVAR, or Change Variable, command and the IF command for additional explanations of their functions.)

GRTOBJAUT (Grant Object Authority) Command

The Grant Object Authority (GRTOBJAUT) command grants, to named users, to all users (*PUBLIC), or to users of the referenced object (REFOBJ parameter), specific rights of use for the object named in the command. (For more information on granting public authority for an object, see the description of the PUBAUT parameter in Appendix A.) This function can be performed by an object's owner, by the security officer, or by a user with object management rights for the object being authorized. A user who has object management rights can grant only the explicit rights that he himself has (except for object management rights, which only the owner and security officer can grant).

Restrictions: Before this command can be used to grant rights of use for a device, control unit, or line description, its associated device, control unit, or line must be varied on. Also, for display work stations, if this command is not entered at the device to which rights are being granted, this command should be preceded by the ALCOBJ command and followed by the DLCOBJ command.



① Any one of the CPF object types listed in the OBJTYPE parameter charts in Appendix A can be specified.
② To code positionally, use *N *N REFOBJ (object-name) REFOBJTYPE (object-type).

Job:B,I Pgm:B,I

OBJ Parameter: Specifies the qualified name of the object that is to have specific rights granted to one or more users. (If no library qualifier is given, *LIBL is used to find the object.)

OBJTYPE Parameter: Specifies the object type of the object that is to have specific rights granted to the specified users. Any one of the CPF object types can be specified; refer to the charts in the expanded description of the OBJTYPE parameter in Appendix A. To grant authority for a library, for example, enter the value *LIB.

USER Parameter: Specifies the names of one or more users to whom rights to the named object are being granted. If USER(*PUBLIC) is specified, the rights are granted to all system users. If user names are specified, the rights are explicitly granted to those users; rights granted by this command can be revoked explicitly by the RVKOBJAUT command.

**PUBLIC:* Specifies that all enrolled users of the system are authorized to use the object in the way specified in the AUT parameter. Public authority is being granted.

user-profile-name: Enter the user profile names of one or more users that are to be explicitly granted rights to the object.

AUT Parameter: Specifies the rights of use that are being granted for the object.

Note: Refer to the chart in the *CPF Programmer's Guide* that shows the applicable rights of use for each object type.

**NORMAL:* Normal rights of use are being granted to the specified users. *NORMAL means that operational rights to the object are being granted as well as data rights that are necessary to perform common functions with the object.

**ALL:* All rights of use applicable to the specified object are being granted.

**OPER:* Operational rights for the specified object are being granted. Operational rights provide the authority to use an object, look at its description, and restore it.

**OBJMGT:* Object management rights, which provide the authority to manage the access and availability of an object, are being granted. A user with object management rights can grant (and revoke) the rights that he has, as well as move and rename objects, and add members to data base files.

**OBJEXIST:* Object existence rights, which provide the authority to control object ownership and existence, are being granted. These rights of use are necessary for a user who wants to delete, free storage, save, restore, or transfer ownership of an object. (If a user has the special save system (*SAVSYS) rights, he does not need object existence rights.)

**READ:* Read rights, which provide the authority needed to retrieve the contents of an object entry, are being granted.

**ADD:* Add rights, which provide the authority needed to add entries to an object, are being granted. (For example, adding job entries to a queue or adding records to a file.)

**UPD:* Update rights, which provide the authority needed to change the entries in an object, are being granted.

**DLT:* Delete rights, which provide the authority needed to remove entries in an object, are being granted. (For example, deleting a program from a library or records from a file.)

REFOBJ Parameter: Specifies the name of the object to be referenced for authorizations. Those authorizations will be granted to the object specified by the OBJ parameter. Any users authorized to the referenced object will be authorized in the same manner to the object being granted authority through reference.

REFOBJTYPE Parameter: Specifies the object type of the object to be referenced (REFOBJ parameter).

*OBJTYPE: Specifies that the object type of the referenced object is to be the same type as the object being granted authority (OBJTYPE parameter).

object-type: Specifies the type of the object. Any one of the CPF object types can be specified; refer to the charts in the expanded description of the OBJTYPE parameter in Appendix A.

Examples

```
GRTOBJAUT  OBJ(PROGRAM1.USERLIB) OBJTYPE(*PGM) +
     USER(*PUBLIC)
```

This command authorizes the object named PROGRAM1 to be used by all enrolled users of the system. The object is a program (*PGM) located in the library named USERLIB. Because the AUT parameter was not specified, the rights granted to all users are the normal rights. These allow all users to execute the program, but not to debug it. To debug the program, users must also have read rights to the program.

```
GRTOBJAUT  OBJ(PROGRAM2.ARLIB) OBJTYPE(*PGM) +
     USER(TMSMITH) AUT(*OBJMGT)
```

This command grants the user named T M Smith the authority to grant the same rights he has to others for the object named PROGRAM2, which is a program located in the library named ARLIB.

GRTUSRAUT (Grant User Authority) Command

The Grant User Authority (GRTUSRAUT) command grants authority to a user profile by referencing another user profile. The authority granted to the user profile depends upon the user profile being referenced and the user profile of the system user issuing the command.

If the security officer issues this command, the user specified in the USER parameter will be granted the same authority for each object that the referenced user profile has, including object management. If the security officer profile is being referenced, only explicit authorities for objects will be granted (no authorities implied in the security officer profile can be granted).

If the system user whose user profile is being referenced issues this command, all authorities for each object he owns will be granted, including object management.

If a system user issues this command to reference another user profile, his user profile must have object management authority and all rights to objects being referenced. In this case, object management authority will not be granted. If the system user does not have object management authority or the rights to be granted for an object being referenced, authority for that object will not be granted.

For objects that the referenced user profile does not own but is authorized to use, the user profile of the system user issuing this command must have object management authority and the rights being granted for the object, or must own the object. Otherwise, no authority will be granted for that object. Object management authority will not be granted unless the user profile of the system user issuing this command owns the object being authorized.

Ownership of an object or authorities held by the referenced user cannot be changed by this command. Authorities to objects granted to a user profile will be added to any authorities that the user profile already had.

User profiles QSYS, QDBSHR, QSPL, or QRJE cannot be specified for either of the parameters on this command.

```
                                                                    Required
GRTUSRAUT────USER user-name──────────────────────────────────────────►

>─REFUSER user-name────────
                                                               Job:B,I Pgm:B,I
```

USER Parameter: Specifies the name of the user to which authority is being granted.

REFUSER Parameter: Specifies the name of the user being referenced for authority.

Example

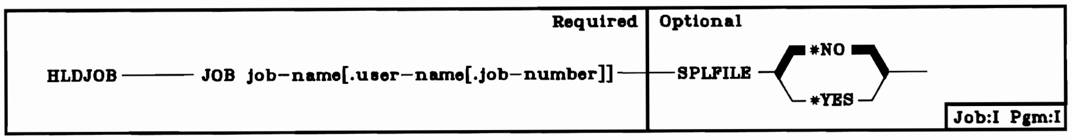
```
GRTUSRAUT  USER(CTB)  REFUSER(your-user-profile-name)
```

This command grants the user profiles CTB and RPB the same authorities that your user profile has for all objects that you own or have rights to (including object management).

HLDJOB (Hold Job) Command

The Hold Job (HLDJOB) command makes a job ineligible for processing by the system. The job is held until it is: released by the Release Job (RLSJOB) command; cleared by the Clear Job Queue (CLRJOBQ) command; canceled by the Cancel Job (CNLJOB) command; or terminated (while the job is active) by the TRMSBS, TRMCPF, or PWRDWNSYS commands. If the job is not executed before CPF is terminated, the queue can be cleared (and the job canceled) when CPF is started again. The specified job to be held can be on the job queue or on the output queue(s), or it can be active in a subsystem. The HLDJOB command also specifies whether the job's spooled output files are to be held.

Restrictions: This command must be submitted from an interactive job. The job must belong to the user issuing the command, or he must have the special job control rights (*JOBCTL).



JOB Parameter: Specifies the qualified name of the job to be held. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. If duplicates of the specified name are found, a qualified job name must be specified. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLFILE Parameter: Specifies whether or not spooled output files created by the job being held are also to be held.

NO: The spooled output files produced by the job are not to be held.

YES: All spooled output files produced by the job are also to be held.

Examples

```
HLDJOB  JOB(PAYROLL)  SPLFILE(*YES)
```

This command makes the job named PAYROLL ineligible for processing. All spooled output files for this job are also to be held.

```
HLDJOB  JOB(WS001)  SPLFILE(*NO)
```

This command holds the interactive job named WS001. If job WS001 produced any spooled output files, they are not to be held.

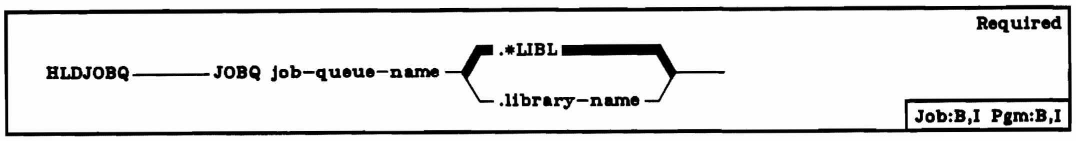
```
HLDJOB  JOB(PAYROLL.DEPTXYZ)
```

This command holds the job named PAYROLL that was submitted by a user operating under the user profile DEPTXYZ. The qualified form of the job name is used when jobs with duplicate names exist in the system. Any spooled output files are not held.

HLDJOBQ (Hold Job Queue) Command

The Hold Job Queue (HLDJOBQ) command prevents the processing of all job entries currently waiting on the job queue and all entries that are added to the queue after the command is issued. The HLDJOBQ command has no effect on executing jobs. Additional entries can be placed on the job queue while it is held, but they will not be processed. The job entries are held until a RLSJOBQ (Release Job Queue) command is issued. When a job queue is held, the job entries can be cleared with the CLRJOBQ command or a specific job can be canceled by the CNLJOB command.

Restriction: The user entering this command must have read, add, and delete rights for the job queue; or his user profile must have job control rights, SPCAUT(*JOBCTL). The job queue must have the OPRCTL(*YES) attribute specified.



JOBQ Parameter: Specifies the qualified name of the job queue that has its current and future entries withheld from further processing. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
HLDJOBQ  JOBQ(QBATCH)
```

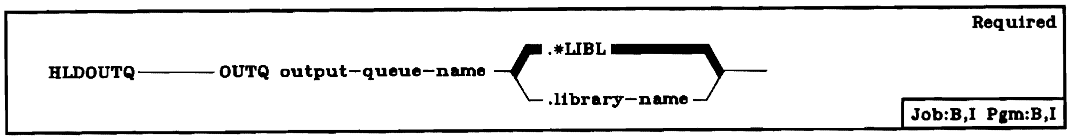
This command prevents the processing of the jobs currently having entries on the QBATCH job queue and any jobs added to the queue. They are held until the queue is released or cleared. Individual jobs can also be canceled with the CNLJOB command, which removes the job's entry from the job queue.

HLDOUTQ (Hold Output Queue) Command

The Hold Output Queue (HLDOUTQ) command prevents all currently waiting output entries, and all entries that are added to the output queue after the command is issued, from being processed by a spooling writer. The HLDOUTQ command has no effect on executing jobs that are adding entries to the queue or on the spooled output file being produced by a spooling writer at the time the command is issued. When the spooling writer completes all copies of the current output file, it cannot begin the output for any other files until the queue is released.

The entries are held until a RLSOUTQ (Release Output Queue) command is issued. Otherwise, a spooled output file can be canceled, the output queue can be cleared with the CLROUTQ command or a specific job can be canceled by the CNLJOB command.

Restriction: The user submitting this command must have read, add, and delete rights for the output queue; or his user profile must have job control rights, SPCAUT(*JOBCTL). The output queue must have the operator control attribute OPRCTL(*YES) specified.



OUTQ Parameter: Specifies the qualified name of the output queue that has its current and future entries withheld from further processing. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
HLDOUTQ  OUTQ(QPRINT)
```

This command prevents the processing of the spooled output files having entries on the QPRINT queue and any entries added to the queue. They are held until the queue is released or cleared. A specific job (with its spooled output files) can also be canceled with the CNLJOB command, which removes the output file entries from the output queue.

HLDRDR (Hold Reader) Command

The Hold Reader (HLDRDR) command immediately halts the activity of the specified spooling reader. The reader is not terminated nor is its associated input device made available to the system. The reader remains inactive until a RLSRDR (Release Reader) or CNLRDR (Cancel Reader) command is issued. Data is not lost when the reader is held.

Restriction: The user must be authorized to use the HLDRDR command; he must also have started the reader or have SPCAUT(*JOBCTL) specified in his user profile.

```
                                                              Required
  HLDRDR ——— RDR reader-name ——

                                                      Job:B,I Pgm:B,I
```

RDR Parameter: Specifies the name of the spooling reader to be held.

Example

```
HLDRDR  RDR(MFCU)
```

This command causes the reader named MFCU to immediately stop reading data. To release the reader, so that it can continue to read data, a RLSRDR (Release Reader) command must be entered. If the CNLRDR (Cancel Reader) command is used, the reader is canceled and the job that was being read in is lost because no job entry was added to the job queue yet.

HLDSPLF (Hold Spooled File) Command

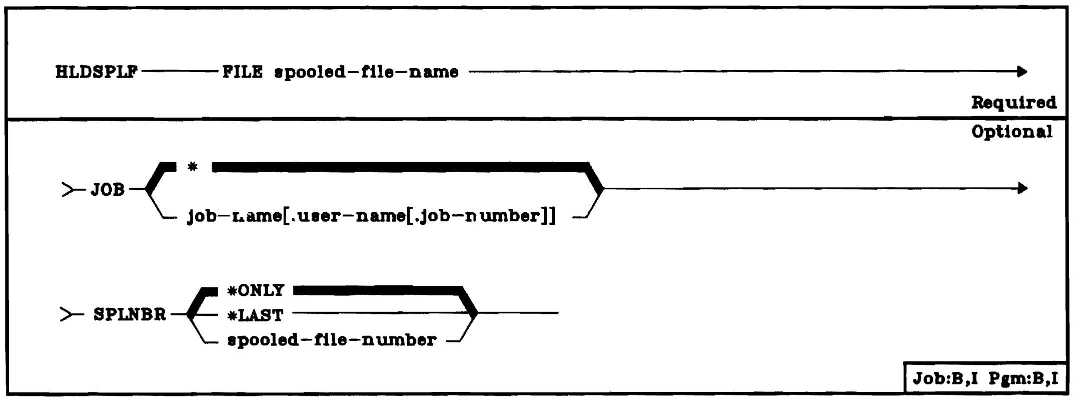
The Hold Spooled File (HLDSPLF) command stops the specified output file from further processing by a spooling writer. If the file is being produced on an output device, the writer stops processing that file and gets the next file to be processed. When the file is released and selected for output, it is again processed starting at the *beginning*. If multiple copies are being produced for the file when it is held, the incomplete copy is produced from the beginning again and any remaining copies follow it.

If the specified file is still receiving records from an executing program when the file is held, the executing program is unaware of the hold state. Also, if the held file is part of a job that produces other spooled output files, they can be processed before the held file is released. The held file remains on the output queue and appears to be the only file produced by the job.

The file is held until one of the following commands is entered: RLSSPLF (Release Spooled File), CNLSPLF (Cancel Spooled File), CNLJOB (Cancel Job), or CLROUTQ (Clear Output Queue) command. If the file is released before the writer begins producing the file, then the file is produced in its normal sequence within the group of files for the job. Production of a job's output is not delayed because some of its files are held.

For more information on holding multiple spooled files for a job, refer to the *Additional Considerations* section of the DSPJOB (Display Job) command.

Restrictions: The user submitting this command must (1) be the owner of the job that created the file being canceled; (2) or have read, add, and delete rights for the output queue containing the file; (3) or have job control rights. The output queue must have OPRCTL(*YES) specified.



FILE Parameter: Specifies the name of the spooled output file to be held. The file name is the name of the device file that was used by the program to create the spooled output file.

JOB Parameter: Specifies the name of the job that created the file to be held.

*: The job that issued this HLDSPLF command is the job that produced the file to be held.

qualified-job-name: Enter the qualified name of the job that created the file to be held. (If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file to be held that was created by the specified job. (For an expanded description of the SPLNBR parameter, see Appendix A.)

*ONLY: Only one spooled output file from the job has the specified file name; therefore, the number of the spooled file is not necessary. If *ONLY is specified and more than one file has the specified name, an error message is displayed to the user or (for batch jobs) sent to the program's message queue.

**LAST:* The highest numbered spooled output file with the specified file name is the file to be held.

spooled-file-number: Enter the number of the spooled file having the specified file name that is to be held.

Example

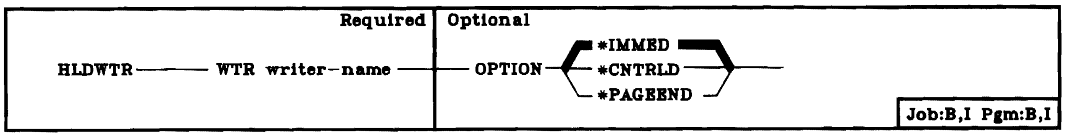
```
HLDSPLF  FILE(SHIPITEMS)  JOB(ORDER.JONES.00009)
```

This command withholds the spooled output file named SHIPITEMS, created by the job ORDER.JONES.000099, from further processing.

HLDWTR (Hold Writer) Command

The Hold Writer (HLDWTR) command stops the specified spooling writer at the end of a record, at the end of a spooled file, or at the end of a printed page. If multiple copies of a file are to be produced, the writer can be held at the end of the copy currently being produced. The writer is not terminated and the device is not made available to the system. The writer remains inactive until a RLSWTR (Release Writer) or CNLWTR (Cancel Writer) command is issued. Data is not lost when the writer is held.

Restriction: To use this command, you must have (1) read, add, and delete rights for the output queue; or (2) you must have job control rights, SPCAUT(*JOBCTL). The output queue must have the operator control attribute OPRCTL(*YES) specified.



WTR Parameter: Specifies the name of the spooling writer to be held.

OPTION Parameter: Specifies when the spooling writer should stop producing output.

*IMMED: The writer stops immediately after it has written the last record, in the current block of records, to the output device. (Each time the writer finishes producing a block of records on a device, it makes another I/O request to get the next block from the file being spooled to the device.) If *IMMED is specified, the writer stops only after it has written the last record in the block being processed, which (for card and diskette output) is a complete card being punched, or a complete diskette record being written on diskette.

When *IMMED is specified for printed output, the writer stops anywhere within or at the end of a print line or at the end of a complete block, which may not be at the end of a line. This is because some data records (which are blocked to improve performance) may be split in two, with the first part of a record at the end of one block and the last part of the record at the beginning of the next block.

**CNTRLD:* The writer stops at the end of the current copy of the file. If only one copy of the file is to be produced or if the last copy is being produced, the entry for the file is removed from the output queue when the output is completed.

**PAGEEND:* The writer is canceled at the end of a page. This value is valid only when the spooling writer is a printer writer.

Example

```
HLDWTR  WTR(PRINTER) OPTION(*CNTRLD)
```

This command stops the writer named PRINTER at the end of the current file. The writer is held until an RLSWTR (Release Writer) or CNLWTR (Cancel Writer) command is issued.

IF (If) Command

The If (IF) command evaluates a logical expression and conditionally executes commands in a CL program based on the result of the expression. If the result of evaluating the logical expression is true (a logical 1), the command or the group of commands (in a do group) specified in the THEN parameter is executed, and the ELSE command with its associated command or do group is not executed. If the result of the logical expression is false (a logical 0), the command specified in the THEN parameter is not executed and control passes to the next command. If that command is an ELSE command, the command or do group specified in that command is executed. If no ELSE is specified, control passes to the next command.

When a DO command is specified in either the THEN parameter on the IF command, or in the CMD parameter of the ELSE command, the entire do group is bypassed if the result of the expression is not the one needed for the group to be executed. That is, control passes to the command following the ENDDO command associated with that DO.

When the command or do group specified by the THEN parameter or the ELSE command is completed (and no GOTO command has been executed), control passes to the next command following the command or do group specified by the ELSE command. If a GOTO command is executed, control is passed to the command with the label specified by the GOTO command, and execution proceeds from that command.

The following command sequence shows this flow. In this example, &TESTSW is a logical variable.

```
IF &TESTSW DO
       .
Group  }.  (group of CL commands)
Ⓐ      .
  ENDDO

ELSE DO
       .
Group  }.  (group of CL commands)
Ⓑ      .
  ENDDO

       .
Group  }.  (continuing CL commands)
Ⓒ      .
```

The IF command tests the logical variable &TESTSW. If a true condition exists (&TESTSW contains a value of '1'), the commands in group Ⓐ are executed, then control passes to the commands in Ⓒ. If a false condition exists, the commands in group A are bypassed, the commands in group Ⓑ are executed, then control passes to the commands in Ⓒ.

Restrictions: The IF command is valid only in CL programs. Up to 10 levels of nested IF and ELSE commands are allowed.

```
                                  Required | Optional

IF ──────── COND logical-expression ───────┼──── THEN CL-command ──────
                                                                     Pgm:B,I
```

COND Parameter: Specifies the logical expression that is to be evaluated to determine a condition in the program and what is to be done next. Refer to *Logical Expressions* in Appendix B for a description of logical expressions. (Note that variables, constants, and the %SUBSTRING and %SWITCH built-in functions can be used within the expression.)

THEN Parameter: Specifies the command or commands (in a do group) that is to be executed if the result of evaluating the expression is true. After the command or do group is executed, control is passed to the next command *after* the ELSE command associated with this IF command. (If the result is true, the ELSE command associated with the IF command is not executed.) If the command specified in this parameter is a DO command, the entire do group is considered to be the command specified by the parameter.

If the command specified by the THEN keyword is not coded on the same line when the keyword is coded, the THEN keyword must be immediately followed (on the same line) either by the left parenthesis or by a + or - to show continuation. (A blank cannot immediately follow any keyword.) The command and the right parenthesis can then be coded on the next line. For example:

```
IF COND(&A *EQ &B)  THEN(    +
      GOTO C)
```

If any part of the command specified by the THEN parameter continues on the next line, a continuation character (+ or -) must be specified.

If a DO command is specified, only the DO command (not the entire do group) is within the parentheses. For example:

```
IF COND(&A *EQ &B)  THEN(DO)
      CMD1
      CMD2
        •
        •
        •
      ENDDO
```

If no command is specified on the THEN parameter (a null THEN) and the ELSE command immediately follows it, the ELSE is executed if the IF expression is false and it is skipped if the expression is true.

Any CL command can be specified on the THEN parameter, except the following commands: ELSE, ENDDO, ENDPGM, MONMSG, PGM, DCL, DCLF, and DCLDTAARA commands. The command can be another IF, unless there are already 10 levels of nested IF and ELSE commands.

Examples

```
IF  COND(&A *EQ &B)  THEN(GOTO X)
IF  (&A *EQ &B)  THEN(GOTO X)
IF  (&A *EQ &B)  (GOTO X)

IF  COND(&A *EQ &B) +
      THEN(GOTO X)
```

The above examples show a number of different ways the IF command can be specified to test a condition and branch to a label. In each of these examples, if &A equals &B, control will pass to a CL command that has a label named X.

```
IF  COND(&TESTSW)  THEN(CHGVAR  VAR(&A)  VALUE(23))
```

If &TESTSW has a logical value of 1 (true), the CHGVAR command is executed to set &A to decimal 23; if not true, CHGVAR is not executed.

```
IF  ((&BAKER *EQ &JOHN) *AND *NOT &CHARLEY) +
      THEN(RETURN)
```

If the value of &BAKER equals the value of &JOHN and &CHARLEY is a logical 0, then return to the program that called this CL program.

```
IF  &LOG1  +
      THEN(IF  (&A *GT 10) +
            THEN(GOTO X))
            ELSE(GOTO Y)
ELSE DO
      •
      • (group of CL commands)
      •
ENDDO
```

This is an example of nested IF commands. If &LOG1 has a logical value of 1 (true) and if &A is greater than 10, a branch is made to label X. If &LOG1 has a logical value of 1 and &A is *not* greater than 10, a branch is made to label Y. If &LOG1 has a logical value of 0 (false), &A is not compared to 10. Instead, the DO group of the second ELSE command is executed.

```
IF  &TEST  THEN(   +
      DO)
         CHGVAR  &A (&A + 1)
         GOTO  X
      ENDDO
ELSE +
      DO
         CHGVAR &B (&B + 1)
         CALL EXTPGM  (&B)
      ENDDO
```

This example shows how the THEN parameter can be continued on the next line. If &TEST has a logical value of 1, the do group specified in the THEN parameter is executed. Otherwise, the do group specified by the ELSE command is executed.

```
IF  (&A *EQ YES) +
DO
   CHGVAR &B 1
   CHGVAR &C 'Z'
ENDDO
```

This example shows a do group as the THEN parameter. The two CHGVAR (Change Variable) commands are executed if, in the relational expression, &A is equal to YES.

```
IF   %SWITCH(10XXXX10)  THEN(GOTO X)
```

This example illustrates how the built-in function %SWITCH is used to test the eight job switches in a job. (Refer to the end of Appendix B for a complete description of %SWITCH.) In this example, job switches 1, 2, 7, and 8 are tested for the values indicated in the 8-character mask. If switches 1 and 7 contain 1s and switches 2 and 8 contain 0s, then the program branches to the command having the label X. If any of the four switches do not contain the value indicated, the branch does not occur.

INZDKT (Initialize Diskette) Command

The Initialize Diskette (INZDKT) command initializes a diskette by writing identification information on it and setting the format that is to be used to store objects and data on it. Initializing a diskette includes the following:

- Checking for files that are still active and should not be cleared.
- Testing each track for physical defects on the recording surface. A diskette is unusable if more than two defective cylinders are found, cylinder 0 is defective, or the track identifier of a defective track cannot be read.
- Formatting each track to a specified sector size (128, 256, 512 or 1024 bytes) and recording mode (single density or double density). A diskette's format determines what the diskette may be used for in later processing.
- Defining a single (expired) file covering the entire diskette. The file is identified as DATA.

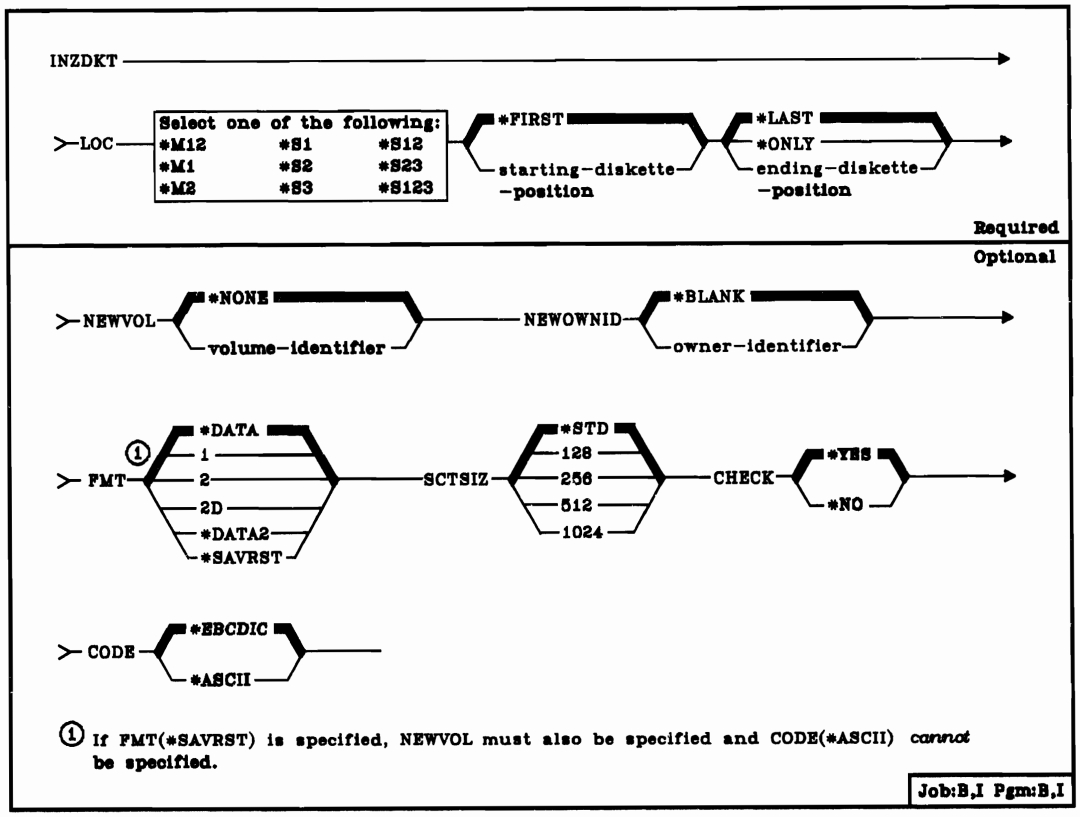
All IBM-supplied diskettes are initialized before they are shipped to a customer. A diskette should be reinitialized when:

- Its format is to be changed.
- The sequence of the records on the diskette is to be changed.
- A defect has occurred in one or two tracks.
- The diskette has been exposed to a strong magnetic field.

For additional information about the care and handling of diskettes, the basic data exchange format, and initialization of the diskette index (cylinder 0), refer to the *IBM Diskette General Information Manual*, GA21-9182.

One INZDKT command can initialize one or more diskettes at a time. The diskettes can be in magazines or in manual slots. If multiple magazines or slots are initialized by one command, they are all given the *same* volume identifier. However, because volumes initialized in the save/restore format should have unique identifiers, multiple magazines or slots cannot be specified in one INZDKT command for that format.

Note: When initializing diskettes with non-IBM standard labels, specify CHECK(*NO).



① If FMT(*SAVRST) is specified, NEWVOL must also be specified and CODE(*ASCII) *cannot* be specified.

LOC Parameter: Specifies which diskette location(s) in the magazines or slots are to be initialized. Three values are needed: (1) the unit type and location (that is, the magazines or slots used), (2) the starting diskette position, and (3) the ending diskette position in the unit. A value must be specified for the first of the three values; if no values are specified for the other two, *FIRST and *LAST are assumed by the system.

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit and location are to be initialized. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be initialized first. Enter one of the following values to specify the starting diskette position:

FIRST: The first diskette position in the location contains the diskette to be initialized first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the first diskette to be initialized. For manual slots, specify the number of the diskette position in the manual slot range specified that is to be the first diskette to be initialized.

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be initialized last. Enter one of the following values to specify the ending diskette position:

*LAST: The last diskette position in the location contains the diskette to be initialized last. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**ONLY:* Only the diskette position specified by the second value is to be initialized.

ending-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the last diskette to be initialized. For manual slots, specify the number of the diskette position in the manual slot range specified that is to be the first diskette to be initialized.

NEWVOL Parameter: Specifies the volume identifier for the diskettes being initialized.

*NONE: No volume identifier is specified; only the system date is written in the volume identifier field of the volume label. *NONE is not valid if FMT(*SAVRST) is specified.

volume-identifier: For diskettes that are not to be used in save/restore operations, enter a maximum of 6 characters for the volume identifier by which all initialized diskettes are to be identified; any combination of characters and digits can be used.

For save/restore diskettes being initialized in manual slots, the volume identifier may be a maximum of 6 characters. Multiple manual slots cannot be initialized to save/restore format; therefore, the system does not append a last digit to the volume identifier.

For save/restore diskettes being initialized in magazines, the volume identifier may be a maximum of 5 characters. The sixth (last) character for each diskette in the magazine is supplied by the system when the diskette is initialized. The system assigns digits 1 through 9 (and 0) to the volume identifier, corresponding to the position of the diskette in the magazine (1 through 10). Diskettes cannot be moved in the magazine after they have been initialized to save/restore format; save/restore requires the last character of the volume identifier to correspond to the position of the diskette in the magazine.

If the identifier has fewer than 6 characters, blanks are added to the right.

For manual slots, specify the number of the diskette position in the manual slot range specified that is to be the last diskette to be initialized.

NEWOWNID Parameter: Specifies the identification of the diskette owner to be written in the volume label. The owner identification contains a maximum of 14 characters (uppercase letters and/or digits in any combination) and is left-justified and padded with blanks on the right if fewer than 14 characters are supplied.

*BLANK: Blanks are to be written in the owner identification field.

owner-identifier: Enter a maximum of 14 uppercase letters and digits that identify the owner of the diskette. Even if enclosed in apostrophes, no lowercase letters, embedded blanks, or special characters can be entered. If fewer than 14 characters are specified, the field is left-justified and padded on the right with blanks.

FMT Parameter: Specifies the format in which the diskettes are to be initialized. Either *DATA, *DATA2, 1, 2, or 2D can be specified if the diskettes are to contain data files that are in the basic, H, or I exchange format. *SAVRST (type E general data exchange) must be specified if the diskettes are to be used in save/restore operations; that is, their data files are to contain saved objects. For additional information on exchange types and diskette types, refer to the *IBM Diskette General Information Manual,* GA21-9182.

*DATA: A one-sided or two-sided diskette is to be formatted with single density recording.

1: A one-sided diskette is to be formatted with single density recording.

2: A two-sided diskette is to be formatted with single density recording.

2D: A two-sided diskette is to be formatted with double density recording.

**DATA2:* A two-sided diskette is to be formatted with double density recording (as if 2D were specified).

**SAVRST:* A two-sided diskette is to be formatted with double density recording. This format is required if the diskette is to be used in a save/restore operation.

Note: Because *DATA2, *SAVRST and 2D specify the diskette will be used for double density recording, it is recommended that a type 2D diskette be used, rather than a type 2. A type 2D diskette is manufactured for double density recording, whereas a type 2 is manufactured for single density recording and would be more prone to media errors if used for double density recording.

Note: If FMT(*SAVRST) is specified:

- The LOC parameter must identify a single magazine or a single slot (*M12 cannot be specified, for example).
- For diskettes in a magazine, the NEWVOL parameter is required; a maximum of 5 characters (the magazine identifier only) can be specified.
- CODE(*ASCII) cannot be specified.

SCTSIZ Parameter: Specifies the number of bytes per sector with which each track is initialized.

STD: Specifies a standard sector size, based on the value of the FMT parameter, as follows:

FMT	SCTSIZ (*STD)
*DATA	128
1	128
2	128
2D	256
*DATA2	256
*SAVRST	1024

128: Specifies that each track is to be initialized with 128 bytes per sector.

256: Specifies that each track is to be initialized with 256 bytes per sector.

512: Specifies that each track is to be initialized with 512 bytes per sector.

1024: Specifies that each track is to be initialized with 1024 bytes per sector.

The following chart chows the valid sector sizes for each FMT value.

SCTSIZ	*DATA	*DATA2	*SAVRST	1	2	2D
*STD	X	X	X	X	X	X
128	X			X	X	
256	X	X		X	X	X
512	X	X		X	X	X
1024		X	X			X

The following chart shows which exchange types can be used for each sector size.

Exchange Type	128	256	512	1024
Basic	X			
H		X[1]		
I	X	X	X	X
SAVRST				X

[1]H will only be used as the exchange type if the diskette is initialized with double density recording (FMT(*DATA2) or FMT(2D)).

For additional information on exchange types, refer to the *IBM Diskette General Information Manual,* GA21-9182.

CHECK Parameter: Specifies whether a check for active files (files having an expiration date greater than the system date) is to be made.

*YES: A check is performed on files whose labels are in cylinder 0 only. File labels in an extended file label area are not checked. If any active files are found, an operator message is issued. The operator can continue initialization (active files are destroyed) or terminate the initialization of that diskette. If more than one diskette is being initialized, initialization continues on the next diskette in sequence.

**NO:* Diskette initialization is to proceed without a check for active files being made.

CODE Parameter: Specifies the code in which the volume label is to be written. All data subsequently written must be in the same code; codes cannot be intermixed on a diskette.

*EBCDIC: The volume label is written in EBCDIC and is an IBM standard label; all subsequent data must also be written in EBCDIC.

**ASCII:* The volume label is written in ASCII and is an IBM standard label in the same format as the EBCDIC label; all subsequent data must also be written in ASCII. If FMT(*SAVRST) is specified, *ASCII cannot be specified.

Examples

```
INZDKT  LOC(*M1) NEWVOL(MASTER) +
        NEWOWNID(DEPT504)
```

This command initializes all diskettes in magazine 1. Each diskette is checked for active files (CHECK(*YES) is assumed). The diskettes are formatted for basic data exchange files (FMT(*DATA) is assumed). Each volume label has MASTER written in the volume identifier field and DEPT504 written in the owner identifier field.

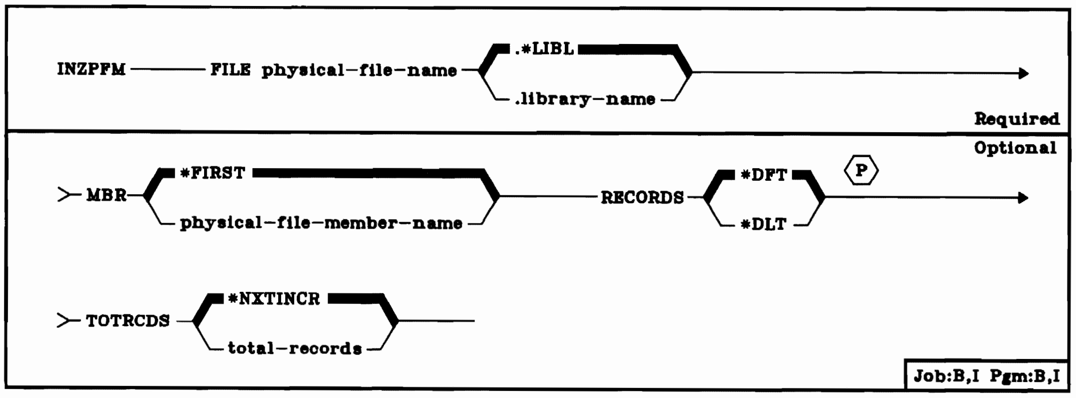
```
INZDKT  LOC(*M2) NEWVOL(SAVE) +
        NEWOWNID(DON) FMT(*SAVRST) CHECK(*NO)
```

This command initializes all diskettes in magazine 2 to the save/restore format. Each diskette is initialized with its location number appended to the NEWVOL parameter of SAVE (for example, diskette 3 has SAVE3 written in its volume identifier field). The owner identifier field has DON written into it.

INZPFM (Initialize Physical File Member) Command

The Initialize Physical File Member (INZPFM) command is used to initialize records in a member of a physical file to the specified type of record (either default or deleted records). The command is typically used for files that have their records in arrival sequence, by relative record numbers. If the member to be initialized is empty, records are added and initialized to the specified type; if the member is not empty, records of the specified type are added to the member. As many records are to be added as necessary to produce the total record count specified.

Restrictions: If any of the access paths to the member are in use when this command is entered, or if any physical file member under any access path is open for updating (this may be through another access path), no records are initialized in the member. To initialize the member with default records, the user must have object management and add rights for the file in which the member exists. To initialize the member with deleted records, the user must additionally have delete rights for the file.



FILE Parameter: Specifies the qualified name of the physical file that contains the member to be initialized. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name of the member, or the first member, that is to be initialized.

<u>*FIRST:</u> The first member of the specified physical file is to be initialized.

physical-file-member-name: Enter the name of the physical file member to be initialized.

RECORDS Parameter: Specifies the type of records that are to be initialized or added to the specified member. The unoccupied record positions in the member are initialized as default records or deleted records.

*DFT: The unoccupied record positions in the member are to be initialized as default records. All numeric fields in each record are initialized to zeros, and all character fields are initialized to blanks.

**DLT:* The unoccupied record positions in the member are to be initialized as deleted records. The records are not eligible for access, but simply hold a place in the file. Deleted records can be updated to reuse the deleted space.

TOTRCDS Parameter: Specifies the total number of records to be in the member after it has been initialized. If the value specified in this parameter causes the size of the file to exceed the size specified when the file was created, a message is sent to the system operator's message queue (QSYSOPR), allowing him to continue or cancel the operation.

*NXTINCR: The number of records in the member is to be increased to extend the file to the next allocation increment. If the member is empty, records are added to meet the initial allocation specified for the member. *NXTINCR is not valid if SIZE(*NOMAX) was specified when the file was created.

total-records: Enter the number of records to which the member is to be increased. If the number of records in the member already meets or exceeds this number, no records are initialized.

Example

```
INZPFM  FILE(INV.QGPL)  TOTRCDS(12000)
```

This command initializes as many as 12 000 records in the first member of the physical file named INV in the QGPL library. Only the number of records are added that brings the total to 12 000 records in the member. Any records that are added are initialized to the default format; numeric fields are initialized to zeros, and character fields are initialized to blanks.

INZTAP (Initialize Tape) Command

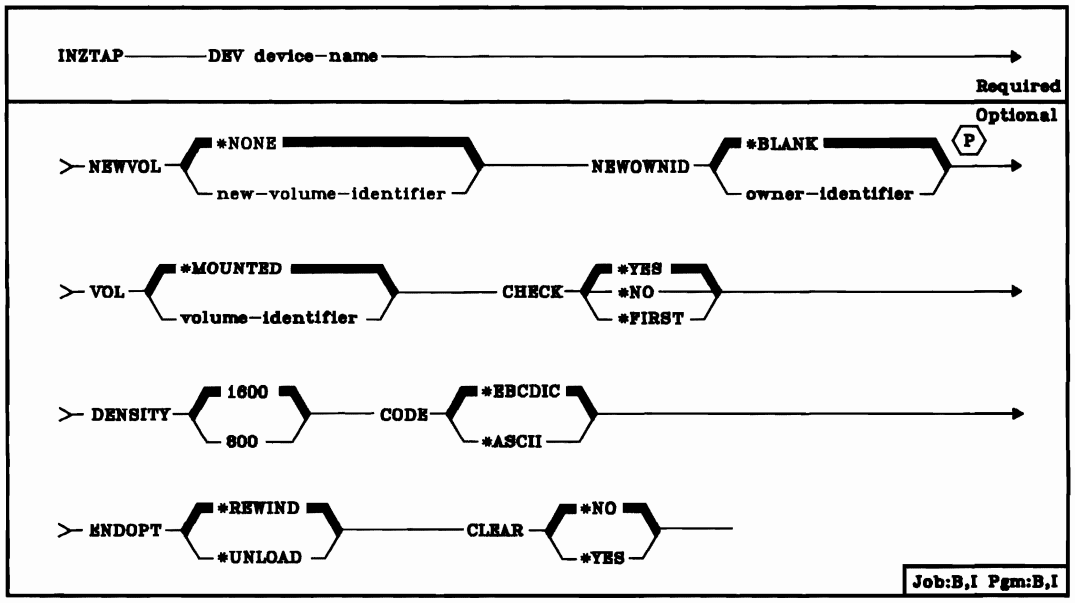
The Initialize Tape (INZTAP) command is used to prepare magnetic tapes for use on the system. This command must be used to write volume labels on standard labeled magnetic tapes so the tape device support can perform standard label processing. Unlabeled tapes must also be initialized by this command or by a similar process on another system before unlabeled tapes can be used on System/38. (Note that this command, unlike the INZDKT command, does not indicate whether the tape is to be used for data written in the basic exchange format or in the save/restore format. That is done when the data files are actually written on the tape.)

Three operations can be performed by this command, depending on the values specified for the CHECK and CLEAR parameters.

- The tape can be checked for active data files (files that have unexpired dates).
- The tape can be initialized either as a standard labeled tape (if NEWVOL specifies a volume identifier) or as an unlabeled tape. Initialization writes only on the beginning of the tape, but makes all data on the reel inaccessible.
- If the tape is being initialized, all previous data (if any) on the tape can also be erased.

If the tape is being initialized as a standard-labeled volume, a volume label followed by two tape marks is written at the beginning of the tape. If it is initialized as an unlabeled tape, only the two tape marks are written.

All tapes must be initialized before use. Tapes that have been initialized need not be reinitialized unless you want to write a new volume label, change the tape type from a standard labeled tape to an unlabeled tape or vice versa, or change the density of a standard labeled tape.



DEV Parameter: Specifies the name of the device on which the volume to be initialized is mounted.

NEWVOL Parameter: Specifies the volume identifier to be written on a tape being initialized as a standard labeled tape. If no volume identifier is specified, the tape is to be initialized as an unlabeled tape; that is, it will have no volume label and no header labels for any data files written on it later.

*NONE: The tape is to be initialized as an unlabeled tape. Only tape marks are to be used to indicate the beginning and end of each data file on it, and the beginning and end of the volume itself.

volume-identifier: Enter a maximum of 6 characters to identify the new volume. The identifier must contain only alphabetic and numeric characters (A-Z, $, #, @, and 0-9), and cannot have a prefix or embedded blanks. Each tape reel should have a unique volume identifier to ensure optimum protection and control of your tape volumes.

NEWOWNID Parameter: Specifies the identification of the tape owner to be written in the volume label. The owner identification contains a maximum of 14 characters (letters and/or digits in any combination) and is left-justified and padded with blanks on the right if fewer than 14 characters are supplied.

*BLANK: Blanks are to be written in the owner identification field.

owner-identifier: Enter a maximum of 14 uppercase letters and digits that identify the owner of the tape. Even if enclosed in apostrophes, no lowercase letters, embedded blanks, or special characters can be entered. If fewer than 14 characters are specified, the field is left-justified and padded on the right with blanks.

VOL Parameter: Specifies the existing volume identifier of the tape to be initialized, or indicates that a mounted tape reel should be initialized.

*MOUNTED: Specifies that any labeled or unlabeled volume mounted on the specified tape device should be initialized. VOL(*MOUNTED) and CHECK(*NO) *must* be used to initialize a new or completely erased volume or CPF will attempt to read labels from the mounted volume and the tape will run completely off the end of the reel.

volume-identifier: Enter the identifier of the labeled volume to be initialized. This parameter value can be used only to reinitialize a tape that is already a labeled volume. If the mounted tape has a different volume identifier than specified or if it is an unlabeled volume, an error message is sent and the tape is not initialized.

CHECK Parameter: Specifies whether a labeled tape volume should be checked for active data files (files with an expiration date greater than the current system date) before it is initialized. If an unlabeled volume is mounted for initialization, this parameter is ignored. If the volume must be checked for active files, as much of the tape is read as necessary before initialization is performed.

<u>*YES:</u> All data file labels on the tape are checked. If any active files are found, the operation is terminated and an error message is sent.

**NO:* Tape initialization should proceed with no checking for active files. To initialize a new or completely erased volume, VOL(*MOUNTED) and CHECK(*NO) must be specified; otherwise, CPF attempts to read labels from the mounted volume until the tape completely unwinds from the reel.

**FIRST:* Only the first data file label on the tape is checked. If there are no data files on the volume or if the first data file has expired, the volume is initialized without checking for any other files on the reel. If the first data file has not expired (active), the operation is terminated and an error message is sent.

If you know there is only one data file on the tape, there are no active files on the volume, or all files have the same expiration date, use CHECK(*FIRST) to perform the fastest initialization of the tape. Note, however, that any data files past the first one will be destroyed, with no expiration check to be made.

DENSITY Parameter: Specifies, in bits per inch, the density in which data is to be written on the tape after it has been initialized. The density used for all data files written to a standard labeled tape is specified when the volume is initialized, and cannot be changed unless the tape is reinitialized. For a nonlabeled tape, the DENSITY parameter specified in the CRTTAPF, CHGTAPF, or OVRTAPF command can change the density of the volume when the first data file on the tape is created.

<u>1600:</u> The data density on this tape volume is to be 1600 bits per inch.

800: The data density on this tape volume is to be 800 bits per inch.

CODE Parameter: Specifies the character code in which the volume label is to be written. All non-save/restore data subsequently written must be in the same code; codes cannot be intermixed on a non-save/restore tape. Save/restore data, however, can be written on any standard labeled tape (initialized in either EBCDIC or ASCII) because save/restore data is written on tape in the same internal format as it was stored in the system (which is in neither EBCDIC nor ASCII). If the tape is being initialized as an unlabeled tape (NEWVOL is specified or assumed to be *NONE), this parameter is ignored.

<u>*EBCDIC:</u> The volume label is written in EBCDIC and is an IBM standard label; all subsequent data must also be written in EBCDIC.

**ASCII:* The volume label is written in ASCII and is an ANSI standard label; all subsequent data must also be written in ASCII.

ENDOPT Parameter: Specifies whether the tape is to be rewound, or rewound and unloaded after it has been initialized.

<u>*REWIND:</u> The tape is to be rewound after it has been initialized.

**UNLOAD:* The tape is to be rewound and unloaded after it has been initialized.

CLEAR Parameter: Specifies whether all previous labels and data are to be erased from the tape if it is initialized. If the volume must be cleared of all data, it will be spaced from the location of the initializing volume label or tape marks to the end of the tape marker.

<u>*NO:</u> The tape is not to be erased. Even though the existing data is not erased, the data on the volume is not accessible after the volume has been initialized.

**YES:* After the beginning of the tape has been initialized, the rest of the tape is to be erased.

Example

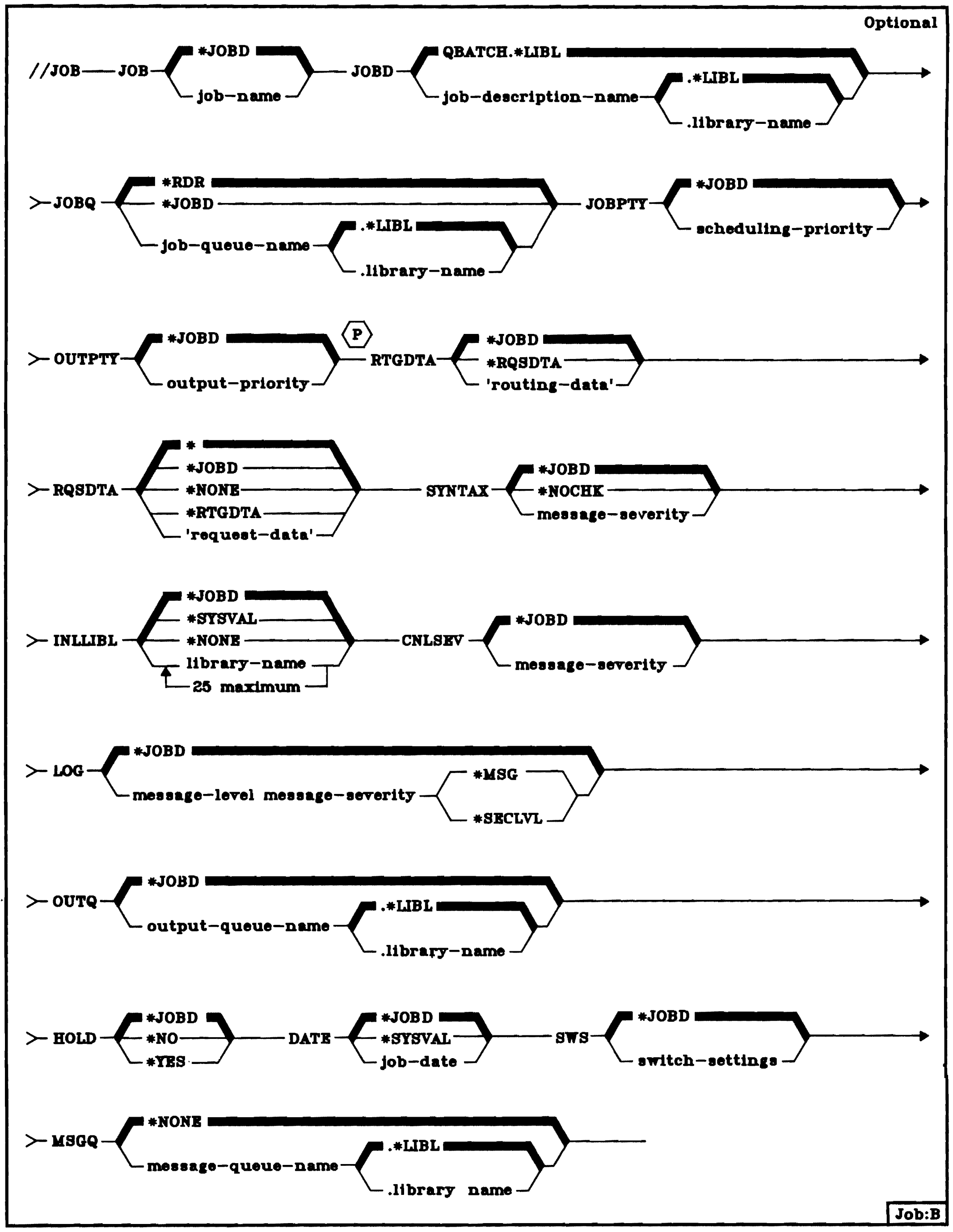
```
INZTAP  DEV(TAPE1)  NEWVOL(T00100)  +
     CHECK(*NO)  CODE(*ASCII)  ENDOPT(*UNLOAD)
```

This command causes the volume mounted on the tape device named TAPE1 to be initialized using the ASCII character code. Its new volume identifier is T00100, regardless of whether it contains a valid volume identifier or unexpired (active) files. Once the volume has been initialized, the tape is rewound and unloaded. (Any previous data beyond the new volume label is not erased, but is no longer accessible.)

JOB (Job) Command

The Job (//JOB) command indicates the beginning of a batch job in a batch input stream. It can also specify different values for the attributes for the job instead of the ones specified in the job description to be used with this job. The values contained in the job description are used for parameters not coded in the JOB command.

The command must be preceded by two slashes in positions 1 and 2; this delimiter makes it possible to locate the beginning of the job in an input stream. Blanks can separate the slashes from the command name (// JOB).


Optional
//JOB
JOB
*JOBD
job-name
JOBD
QBATCH.*LIBL
job-description-name
.*LIBL
.library-name
JOBQ
*RDR
*JOBD
job-queue-name
.*LIBL
.library-name
JOBPTY
*JOBD
scheduling-priority
OUTPTY
*JOBD
output-priority
P
RTGDTA
*JOBD
*RQSDTA
'routing-data'
RQSDTA
*
*JOBD
*NONE
*RTGDTA
'request-data'
SYNTAX
*JOBD
*NOCHK
message-severity
INLLIBL
*JOBD
*SYSVAL
*NONE
library-name
25 maximum
CNLSEV
*JOBD
message-severity
LOG
*JOBD
message-level message-severity
*MSG
*SECLVL
OUTQ
*JOBD
output-queue-name
.*LIBL
.library-name
HOLD
*JOBD
*NO
*YES
DATE
*JOBD
*SYSVAL
job-date
SWS
*JOBD
switch-settings
MSGQ
*NONE
message-queue-name
.*LIBL
.library name
Job:B

JOB Parameter: Specifies the job name to be associated with the job as it is being processed by the system.

*JOBD: The simple name of the job description used with this job is to be the name of the job itself.

job-name: Enter the simple name of the job that is to be used while it is being processed by the system.

JOBD Parameter: Specifies the qualified name of the job description to be used with this job.

QBATCH: The IBM-supplied job description, QBATCH, in the QGPL library is to be used for the job. (The QGPL library must be in the library list used by the spooling reader that reads the job's input.)

job-description-name: Enter the qualified name of the job description that is to be used for the job. (If no library qualifier is given, the job description is found through the library list used by the spooling reader that reads the job's input.)

JOBQ Parameter: Specifies the name of the job queue on which this job is to be placed.

*RDR: The job queue specified in the start reader or submit jobs command that reads this job is the job queue to be used.

**JOBD:* The job queue to be used is named in the job description used with this job.

qualified-job-queue-name: Enter the qualified name of the job queue on which this job is to be placed. (If no library qualifier is given, *LIBL is used to find the queue.)

JOBPTY Parameter: Specifies the scheduling priority that the job is to have. Valid values are 1 through 9, where 1 is the highest priority and 9 is the lowest. (For an expanded description of the JOBPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

*JOBD: The scheduling priority specified in the job description is to be used for this job.

scheduling-priority: Enter a value, 1 through 9, that is to be the scheduling priority for this job.

OUTPTY Parameter: Specifies the output priority for spooled output files that are produced by this job. The highest priority is 1 and the lowest is 9. (For an expanded description of the OUTPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

*JOBD: The output priority specified in the job description is to be used for this job.

output-priority: Enter a value, 1 through 9, for the priority of this job's output files.

RTGDTA Parameter: Specifies the routing data that is to be used to initiate the first routing step in the job. The routing data is compared with the compare values in the routing entries to determine which routing entry is to be used to initiate the routing step. (The routing entry identifies which program is to begin processing the job.) For example, the value QCMDI is the routing data used by the IBM-supplied interactive subsystem (QINTER) to route interactive jobs to the IBM-supplied control language processor, QCL.

*JOBD: The routing data to be used to initiate the first routing step is in the job description used with this job.

**RQSDTA:* The request data (of up to 80 characters) specified in the RQSDTA parameter of this command is to be used as the routing data for this job.

'routing-data': Enter the character string that is to be used as the routing data for initiating the first routing step. A maximum of 80 characters can be entered (enclosed in apostrophes if necessary).

RQSDTA Parameter: Specifies the request data that is to be placed as the last entry in this job's message queue. The request data can be a CL command to be executed or a string of characters to be used by another program. For example, if RTGDTA(QCMDB) is specified, the IBM-supplied batch subsystem QBATCH is being used, and a CL command is supplied, it becomes a message that is read by the control language processor, QCL. Or, if a user program is specified in the routing entry, the request data can specify information such as the record number of the first record in a file to be processed.

Note: If a value other than * is specified for this parameter, then the data that follows the JOB command is ignored (it will not be used as request data).

*: The data following this JOB command is to be inserted into this job's message queue as request data. The request data may be, for example, a group of CL commands that constitute the job.

**JOBD:* The request data specified in the job description used by this job is to be placed as the last entry in this job's message queue.

**NONE:* No request data is to be placed in the job's message queue.

**RTGDTA:* The routing data specified in the RTGDTA parameter of this command is to be placed as the last entry in the job's message queue.

'request-data': Enter the character string that is to be placed as the last entry in the job's message queue. A maximum of 256 characters can be entered (enclosed in apostrophes if necessary). When a CL command is entered, it must be enclosed in single apostrophes, and where apostrophes would normally be used *within* the command, double apostrophes must be used instead.

SYNTAX Parameter: Specifies whether requests placed on the job's message queue are to be syntax checked as CL commands. When syntax checking is specified, the commands are syntax checked when they are submitted rather than when the job is executed, thus providing an earlier diagnosis of syntax errors. If checking is specified, the message severity that causes a syntax error to terminate processing of a command is also specified. This parameter is used only if RQSDTA(*) is specified.

*JOBD: The value specified in the job description used with this job determines whether the request data is to be syntax checked and what message severity is to be used.

**NOCHK:* The request data for this job is not to be syntax checked as CL commands.

message-severity: The request data is to be syntax checked as CL commands, and, if a syntax error occurs that is equal to or greater than the error message severity specified here, the execution of the job containing the erroneous command is suppressed. Enter a value, 00 through 99, that specifies the lowest message severity that causes the job's execution to be suppressed. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

INLLIBL Parameter: Specifies the initial library list that is to be used by this job to search for any CPF object names that were specified without a library qualifier.

*JOBD: The library list specified in the job description used with this job is to be used as the initial library list for the job.

**SYSVAL:* The system default library list is to be used by this job. It contains the library names that were specified in the system values QSYSLIBL and QUSRLIBL at the time that the job is initiated.

**NONE:* The user portion of the initial library list is to be empty; only the system portion is to be used.

library-name: Enter the names of one or more libraries that are to be the user portion of the library list and are to be used by this job. No more than 25 names can be specified; the libraries are searched in the same order as they are listed here.

CNLSEV Parameter: Specifies the message severity level of escape messages that can cause a batch job to be canceled. The batch job is canceled when a request in the batch input stream sends to the request processing program an escape message whose severity code is equal to or greater than that specified. (This type of cancel is considered an abnormal termination.) This parameter value is compared with the severity of any unmonitored escape message that occurs as a result of executing a noncompiled CL command in a batch job.

*JOBD: The severity limit specified in the job description used with this batch job determines when the job is to be canceled.

message-severity: Enter a value, 00 through 99, that specifies the message severity of an escape message that results from a request in the batch input stream and that causes the job to be canceled. Because escape messages have a maximum severity level of 50, a value of 50 or lower may be specified in order for a job to be canceled as a result of an escape message. An unhandled escape message whose severity is equal to or greater than the value specified causes the job to be canceled. (Refer to the expanded description of the SEV parameter in Appendix A for a list of the IBM-defined severity codes.)

LOG Parameter: Specifies the message logging values to be used by this job. They determine the amount and type of information to be logged in the job log. The LOG parameter is made up of a list of three values: the message (or logging) level, the message severity, and the level of message text. If no values are specified for the LOG parameter, the values specified in the job description used with this job are used. If one value is to be changed, all three values in the list must be specified.

*JOBD: The list of values specified for message logging in the job description are to be used for this job.

message-level: Enter a value, 0 through 4, that specifies the message logging level to be used for this job's messages. (For additional information on the message levels, refer to *Message Level* under the CRTJOBD command's LOG parameter.)

message-severity: Enter a value, 00 through 99, that specifies the lowest severity level that causes an error message to be logged in the job's log. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

**MSG:* Only first-level message text is to be written to the job's log or displayed to the user.

**SECLVL:* Both the first-level and second-level text of the error message is to be written to the job's log or displayed to the user.

OUTQ Parameter: Specifies the name of the default output queue that is to be used for spooled output produced by this job.

JOBD: The output queue named in the job description used with this job is to be this job's default output queue.

qualified-output-queue-name: Enter the qualified name of the default output queue that is to be used by this job. (If no library qualifier is given, *LIBL is used to find the queue.)

HOLD Parameter: Specifies whether this job is to be held at the time that it is put on the job queue. A job placed on the job queue in the hold state is held until it is released by the Release Job (RLSJOB) command or canceled, either by the Cancel Job (CNLJOB) command or by the Clear Job Queue (CLRJOBQ) command. If the job is not executed before CPF is terminated, the job queue can be cleared (and the job canceled) when CPF is started again.

JOBD: The value specified in the job description determines whether this job is to be held when it is put on the job queue.

NO: The job is not to be held when it is put on the job queue.

YES: The job is to be held when it is put on the job queue, and held until it is released by a RLSJOB command or canceled by a CNLJOB command.

DATE Parameter: Specifies the date that is to be assigned to the job when it is initiated.

JOBD: The date specified in the job description is to be used as the job date.

SYSVAL: The value in the QDATE system value at the time the job is initiated is to be used as the job date.

job-date: Enter the value that is to be used as the job date when the job is initiated. The value must be entered using the system date format specified by the system value QDATFMT.

SWS Parameter: Specifies the initial settings for a group of eight job switches to be used with this job. These switches can be set or tested in a CL program and used to control the flow of the program. For example, if a certain switch is on, another program could be called. The job switches may also be valid in other HLL programs. Only zeros (off) and ones (on) can be specified in the eight-digit character string.

JOBD: The value specified in the job description is to be the initial settings for this job's switches.

switch-settings: Enter any combination of eight zeros and ones that is to be used as the initial switch setting for this job.

MSGQ Parameter: Specifies the message queue, if any, to which a completion message is to be sent when the submitted job has completed execution, either normally or abnormally. If an abnormal termination occurs, the second-level text of the completion message sent specifies the possible causes.

**NONE:* No completion message is to be sent.

qualified-message-queue-name: Enter the qualified name of the message queue to which the completion message is to be sent. If no library qualifier is given, the job's library list is used to find the queue.

Examples

```
//JOB  JOBD(PAYROLL)
```

This command begins the batch job called PAYROLL. The job name is the same as the name of the job description used with the job. All of the values specified in the job description are used as the attributes of this job.

```
//JOB  JOBD(QBATCH) JOB(PAYROLL) +
     JOBQ(BATCH2) INLLIBL(PAYLIB) +
     SWS(00101100) DATE(010180)
```

This command begins a batch job called PAYROLL, which is executed using attributes from the IBM-supplied job description for batch jobs, QBATCH. The job is placed on the job queue BATCH2. The library PAYLIB is the only library in the user portion of the library list. Switches are set for use in the job, and the date is set at January 1, 1980.

```
// JOB  JOBD(COMPILE) JOBPTY(5) +
     SYNTAX(10) INLLIBL(MYCMDS) CNLSEV(40)
```

This command begins a batch job called COMPILE, which is executed using all of the attributes described in the job description also named COMPILE, except for the parameters that are changed by this command. The library MYCMDS is the only library in the user portion of the library list to be used when the commands are syntax checked or executed. Any syntax errors with a value equal to or greater than 10 will suppress processing of the job. The job is assigned a scheduling priority of 5 and is executed as long as no errors are encountered that cause an escape message to be sent that has a severity of 40 or higher.

JRNPF (Journal Physical File) Command

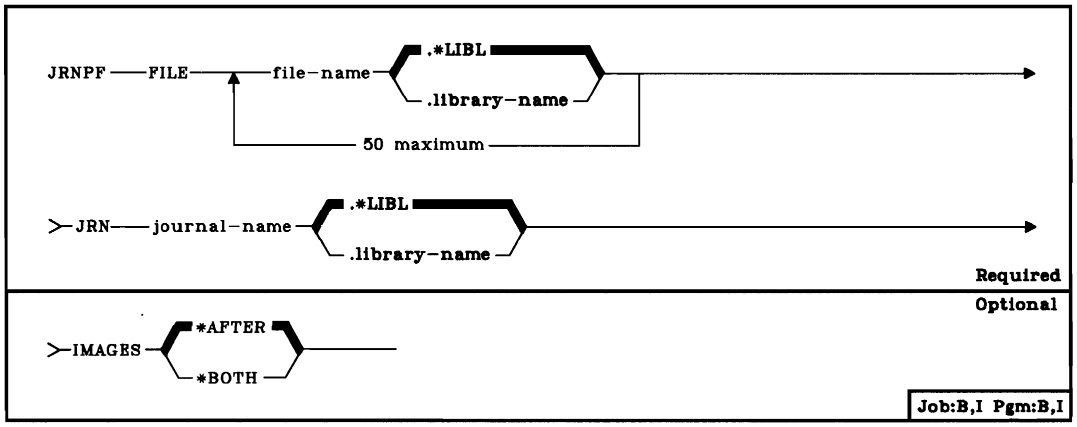
The Journal Physical File (JRNPF) command is used to start journaling changes (made to all members of a specific physical file) into a specific journal. New members added to the file will also have their changes journaled.

You can specify whether you want only the *after* image or both *before* and *after* images of records in the physical file to be recorded in the journal. *Before* images are necessary for use by the IBM-supplied backout procedures, invoked by the Remove Journaled Changes (RMVJRNCHG) command.

After journaling begins for the file, you should execute the Save Changed Object (SAVCHGOBJ) command with OBJTYPE(*FILE) and OBJJRN(*YES) specified. The file must be saved because journaled changes cannot be applied to a version of the file that was saved when journaling was not in effect.

Restrictions: The file must not be journaling changes to another journal when this command is executed to start journaling. There can be no members of the file currently in use for any reason on the system. The file names specified must be those that are applicable after overrides have been applied.

The maximum number of objects that can be associated with one journal is 262 136. This maximum includes physical file members that are currently being journaled, members for which journaling was stopped while the current receiver was attached, and journal receivers that are or were associated with the journal while the current journal receiver is attached. If this maximum is exceeded, journaling will not be started.



FILE Parameter: Specifies the qualified name of the physical file that is to have its changes journaled. (If no library qualifier is specified, *LIBL is used to find the file.) The name specified is the name of the file affected after all overrides have been applied.

JRN Parameter: Specifies the qualified name of the journal that is to receive the journaled changes. (If no library qualifier is specified, *LIBL is used to find the journal.)

IMAGES Parameter: Specifies the record images to be generated for changes to records in the file, when appropriate.

<u>*AFTER:</u> Only 'after' images are to be generated for changes to records in this file.

**BOTH:* The system is to generate both *before* and *after* images for changes to records in this file.

Example

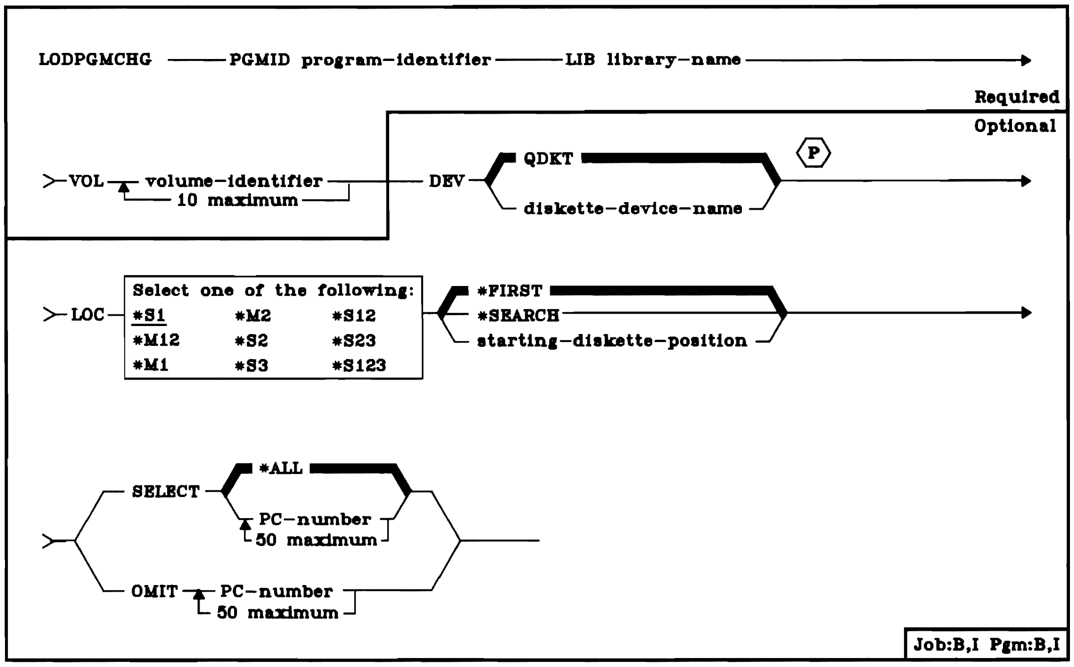
```
JRNPF FILE(MYFILE) JRN(JRNLA.MYLIB)
```

This command journals all changes to all members of file MYFILE (as found using the library search list) to journal JRNLA in library MYLIB. Only the *after* images of the changed record will be available on the journal.

LODPGMCHG (Load Programming Change) Command

The Load Programming Change (LODPGMCHG) command is used to load programming changes (PCs) from diskette into a specified library. Each PC contains one or more objects, including programs, which can be applied to a licensed program by the APYPGMCHG (Apply Programming Change) command.

Only the PCs for a single licensed program, such as CPF, can be loaded at a time. Not all of the PCs for the specified program have to be loaded, because you can select and load specific PCs.



PGMID Parameter: Specifies the identifier of the licensed program for which PCs are to be loaded. The program identifier is used to verify that the PCs are applied to the correct program.

LIB Parameter: Specifies the name of the library into which the programming changes are to be loaded.

VOL Parameter: Specifies the volume identifiers of one or more diskette volumes on which the PCs to be loaded are stored. The volume must be mounted on the diskette magazine drive in the location specified by the LOC parameter. For single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the *magazine* identifier (5 characters maximum) can be specified for each volume. If a magazine is being used, the PCs are *always* loaded starting with the first diskette in the magazine.

DEV Parameter: Specifies the name of the diskette device from which the PCs are to be loaded. The device name must already be known on the system by a device description.

QDKT: The diskette device QDKT is to be the device used to load the PCs.

diskette-device-name: Enter the name of the diskette device that is to be used to load the PCs from diskette.

LOC Parameter: Specifies which diskette locations on the diskette magazine drive are to be used to load the specified programming changes. The PCs can be loaded from diskettes that are in the manual slots or in the magazines. Only one type of unit can be used. If the load operation requires that additional magazines or diskettes be loaded, a message is issued to the operator to mount them. (For an expanded description of the LOC parameter, see Appendix A.)

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.) Note that *S1 (manual slot 1) and *FIRST are the default values.

Unit Type and Location: The first value specifies which unit (magazines or slots) and location are to be used to load the PCs. Enter one of the following values to specify the unit type and location: *S1, *M12, *M1, *M2, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be used first when the PCs are being loaded. Enter one of the following values for the starting diskette position:

FIRST: The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**SEARCH:* The diskettes in the location specified by the first value of this parameter are to be searched for the appropriate programming change file. The search begins with the leftmost diskette in the specified location and ends with the first diskette on which the specified library and version are found.

starting-diskette-position: If *M12, *M1, or *M2 is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10) in the magazine that contains the first diskette to be used. (This option is not valid for the second value if manual slots are specified for the first value.)

SELECT Parameter: Specifies which of the PCs that are on diskette for the specified program are to be loaded. The OMIT parameter cannot be specified if SELECT is specified with PC numbers.

ALL: All the programming changes on diskette for the specified program are to be loaded.

PC-number: Enter the PC identification numbers of the individual programming changes that are to be loaded.

OMIT Parameter: Specifies that all programming changes except for those specified in this parameter are to be loaded from diskette. Enter the PC numbers of the PCs to be omitted when all the rest are loaded. A maximum of 50 PC numbers can be specified. The OMIT parameter cannot be specified if individual PC numbers are specified on the SELECT parameter.

Example

```
LODPGMCHG  OMIT(00003 00008 00014)  LOC(*M1)  VOL(12380)
```

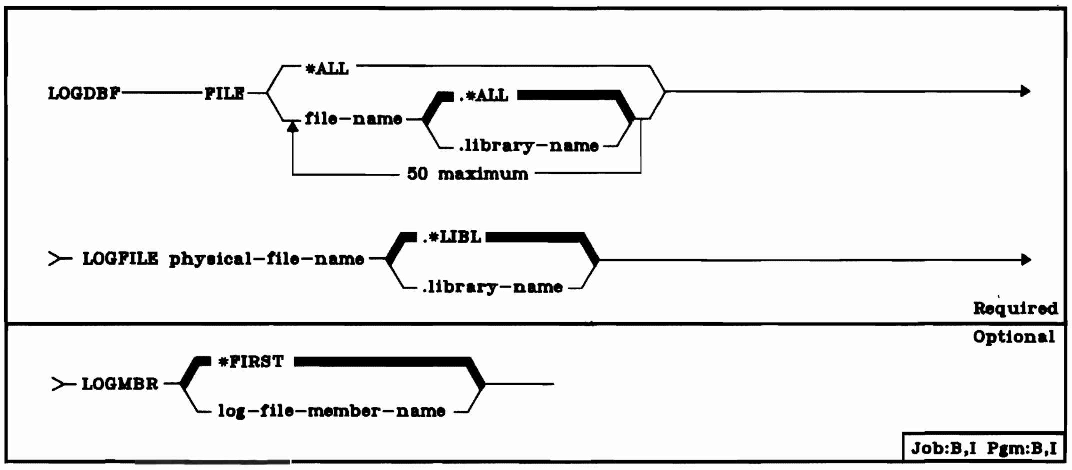
This command loads the PCs written on the diskette located in the volume named 12380, which is mounted in magazine 1. Beginning with the first diskette in magazine 1, all of the PCs except 3, 8, and 14 are loaded. The APYPGMCHG command can then be used to apply these PCs to the CPF licensed program.

LOGDBF (Log Data Base File) Command

The Log Data Base File (LOGDBF) command logs all data base operations for the job or routing step that modify one or more data base files specified by the command. All files specified by the LOGDBF command that are opened in the routing step *after* the command is executed will have written to the log file all changes that are made to the files by this job or routing step. The log file is a physical file that is also identified on the command. The logging (done for all members in the specified files) continues until either the ENDLOG (End Logging) command is executed or the routing step ends. The log produced in the physical file can be used by the user to recover files that may have been affected by a system failure, for example.

Entries are logged in the file in exactly the same order as the operations are performed on the user's file by this and any other job that is also logging changes. This is true for put, update, and delete operations, and other operations such as initializing, reorganizing, or clearing a member in the file. CPF logs each data base operation in the log file before the operation is performed. Three exceptions—put, force end of data (FEOD), and close operations—are each logged after the operation has been successfully completed. For the put operation (adding a new record to the file), this allows the relative record number to be logged. For the FEOD and close operations, this ensures that all file records modified by the routing step are forced to auxiliary storage before the operation is logged. (This does not necessarily force to auxiliary storage records that are modified by other jobs or routing steps.) Therefore, the close and FEOD entries can be used as valid starting points for recovery.

Restrictions: (1) Data base logging does *not* occur for any data base files specified on the LOGDBF command that are already open in the routing step before the command is executed. (2) If logging is already active in the routing step when this command is entered, an escape message is sent to the user.



FILE Parameter: Specifies the qualified names of one or more physical or logical files whose data base operations are to be logged after the files are opened in the routing step. Each file and library name must be the final names that result after all file overrides have been applied to the files at file open time.

*ALL: All data base files that are opened in the routing step after the LOGDBF command is executed are to have all changes to the files logged.

qualified-data-base-file-name: Enter the qualified names of one or more physical and/or logical files for which operations performed on them are to be logged. The names specified must be those that are applicable after file overrides have been applied. If (for each file specified) the library qualifier *ALL is specified or assumed, the operations performed on all files in the routing step having the specified file name are logged. If a library name is specified, only the operations on the specified file in that library are logged.

Note: If (after the LOGDBF command is executed) a MOVOBJ or RNMOBJ command is executed for an unopened file that is to be logged, that file is not logged. If the file is open when either command is issued, logging continues for the file and the entries will contain the old file or library name.

LOGFILE Parameter: Specifies the qualified name of the physical file to which log entries are to be written. (If no library qualifier is given, *LIBL is used to find the file.)

The log file must be a physical file that already exists and must already contain the member specified by LOGMBR. It *must* have a record length of at least 100 bytes for system-supplied data. If the user wants to log the records *after* they are changed (applies to put and update operations only), it *should* contain enough additional space for the longest changed record in the files to be logged. Also, it cannot have a shared ODP (open data path). The user must have operational and add rights for the log file. (For a description of the log file, the format of its logged entries, and a list of the data base operations that generate logged entries, refer to Chapter 5, *Data Base,* in the *CPF Programmer's Guide.*)

LOGMBR Parameter: Specifies the name of the member in the log file that is to contain the logged entries.

*FIRST: The first member of the specified physical file is to contain the logged entries.

log-file-member-name: Enter the name of the member in the physical file that is to contain the logged entries.

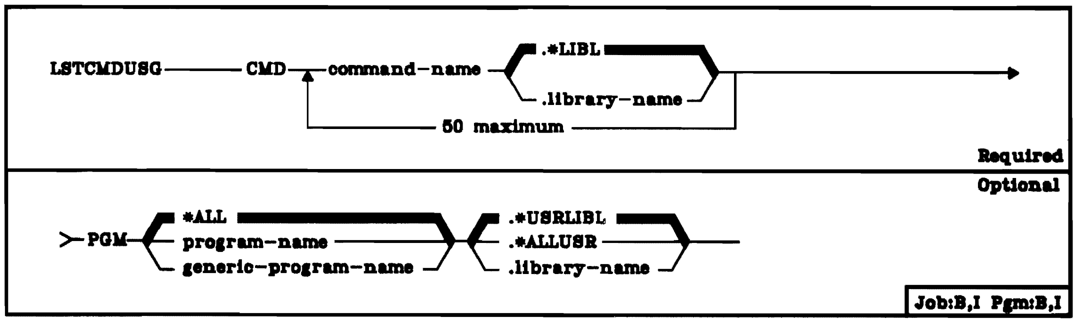
Example

```
LOGDBF  FILE(PAYROLL  ACTEMPL  PAYROLL2.LIB2) +
     LOGFILE(FILEX.QGPL)  LOGMBR(MBR4)
```

This command causes all unopened files named PAYROLL and ACTEMPL in the routing step, and the file PAYROLL2 in library LIB2, to have their data base operations logged to MBR4 of the physical file FILEX in the QGPL library. All unopened files named PAYROLL and ACTEMPL that are in all the libraries in the system will have logging performed for them after they are opened.

LSTCMDUSG (List Command Usage) Command

The List Command Usage (LSTCMDUSG) command produces a cross-reference listing of a specified group of CL commands that are used in a specified group of CL programs. The listing shows, program by program, which of the specified commands are used in each program. The listing can be used to identify which programs need to be recompiled because of changes that have been made to the command definition objects of commands specified on the LSTCMDUSG command. (Note that this command can take a long time to execute and can produce a lot of printed output.)



CMD Parameter: Specifies the qualified names of one or more CL commands for which specified programs are to be searched and listed in a command usage report. The system searches the specified programs for every occurrence of each command-name.library-name character string you specify. If no library qualifier is given, the system searches for command-name.*LIBL. As many as 50 CL command names can be specified.

PGM Parameter: Specifies the qualified name of the program or the generic name of several programs to be searched for the specified commands. Or, this parameter can specify that all programs in the specified library or libraries (*ALL.*USRLIBL for example) are to be searched. Only the programs and libraries for which the user has some (any) authority are included in the report.

The programs are listed in the report in alphabetic order by library. If a program of the specified name contains none of the commands specified, the program name is omitted from the report.

Depending on the library qualifier specified or assumed, the following libraries (for which the user has the authority) are to be searched for the specified program(s):

- .*USRLIBL (user library list). Only the libraries listed in the user portion of the job's library list are to be searched. Note that if the name of a specific program is given and *USRLIBL is specified or assumed, only the first program found that has the specified name is searched for any commands. (All other programs by the same name that are in other libraries in the library list are not searched at all.) If the user does not have authority for the first program found, or if no program is found by that name, an error message is sent and the operation is terminated.
- *.*ALLUSR* (all user libraries). All the nonsystem libraries, which include *all* user-defined libraries and the QGPL library, not just those in the job's library list, are to be searched. Libraries other than QGPL whose names begin with the letter Q are not searched. Every command specified by CMD that occurs in every program specified by PGM that is located in any of the user libraries for which the user has the authority is listed in the report.
- *.library-name* (one library). Only the library named in this parameter is to be searched. The user must have read rights for the specified library.

**ALL:* All CL programs in the specified library (or all libraries identified in the library qualifier) for which the user has some authority are searched.

qualified-generic-program-name: Enter the qualified name of the program or the generic name of several programs in the specified library qualifier that are to be searched for the specified commands. To specify a generic name, add an asterisk (*) at the end of the characters that are in the names of all the programs desired.

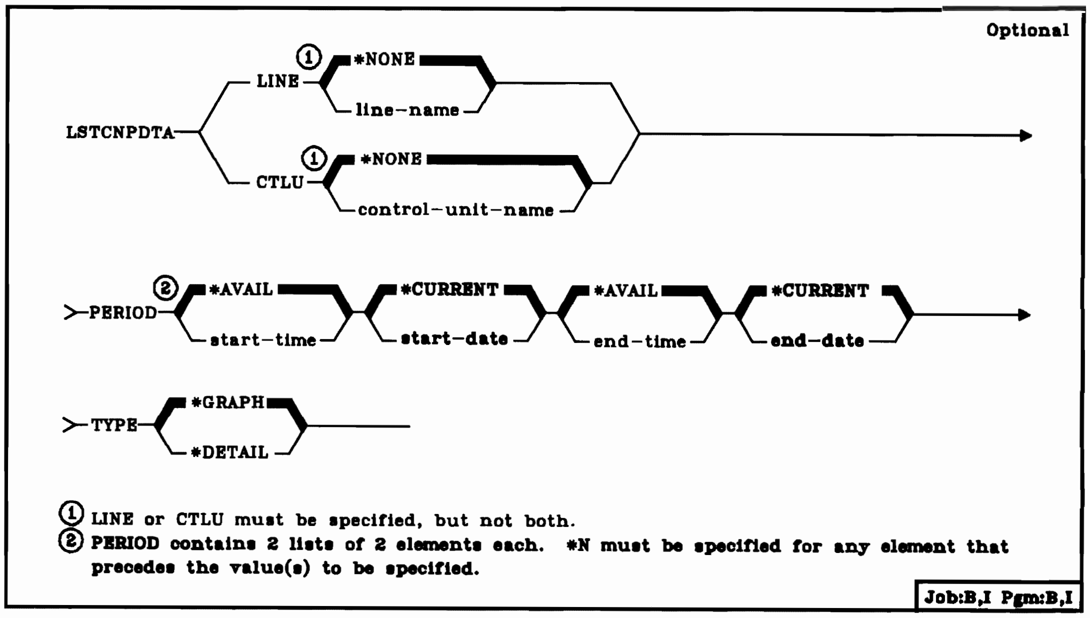
Example

```
LSTCMDUSG  CMD(CPYF)  PGM(*ALL.PAYROLL)
```

This command searches all CL programs in the library PAYROLL for the CPYF command and lists the names of both the command and the program.

LSTCNPDTA (List CSNAP Data) Command

The List CSNAP Data (LSTCNPDTA) command places a formatted listing of the short-term CSNAP (communications statistical network analysis procedure) line and station statistics into a spooled printer device file named QPCSMPRT. This listing can be useful for detecting communications problems. For more information on the use of this command, refer to the *IBM System/38 Problem Determination Guide*, SC21-7726.



① LINE or CTLU must be specified, but not both.
② PERIOD contains 2 lists of 2 elements each. *N must be specified for any element that precedes the value(s) to be specified.

LINE Parameter: Specifies which line is to have its short-term CSNAP data listed.

*NONE: CSNAP data is not to be listed for any lines. If *NONE is specified, a control unit name must be specified (CTLU parameter).

line-name: Enter the name of the line that is to have its short-term CSNAP data listed. The name can be of any line that has a line description. If a line name is specified, CTLU(*NONE) must be specified.

CTLU Parameter: Specifies which control unit is to have its short-term CSNAP data listed.

<u>*NONE:</u> CSNAP data is not to be listed for any control unit. If *NONE is specified, a line name must be specified.

control-unit-name: Enter the name of the control unit that is to have its short-term CSNAP data listed. The name can be of any control unit that has a control unit description. If a control unit name is specified, LINE(*NONE) must be specified.

PERIOD Parameter: Specifies the period of time for which the specified CSNAP data is to be listed. This parameter contains two lists of two values each. Refer to the syntax diagram for the order in which the values are specified.

<u>*AVAIL:</u> The CSNAP data that is available for the specified starting or ending date is to be listed.

<u>*CURRENT:</u> The CSNAP data that is available for the current day and is between the specified starting and ending times (if specified) is to be listed.

start-time: Enter the starting time of the specified starting date for which CSNAP data is to be listed. The time can be entered as four or six digits (hhmm or hhmmss) where hh = hours, mm = minutes, and ss = seconds. If colons are used to separate the time values, the string must be enclosed in apostrophes ('hh:mm:ss').

start-date: Enter the starting date for which CSNAP data is to be listed. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

end-time: Enter the ending time for the specified ending date for which CSNAP data is to be listed. The time can be entered as four or six digits, and colons can be used ('hh:mm:ss').

end-date: Enter the ending date for which CSNAP data is to be listed. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

TYPE Parameter: Specifies the type of report to be generated.

GRAPH: A graph of the short-term CSNAP data is to be generated.

**DETAIL:* A detailed listing of the short-term CSNAP data is to be generated.

Examples

```
LSTCNPDTA  LINE(LN1) PERIOD((*AVAIL 031578)(*N 031578))
```

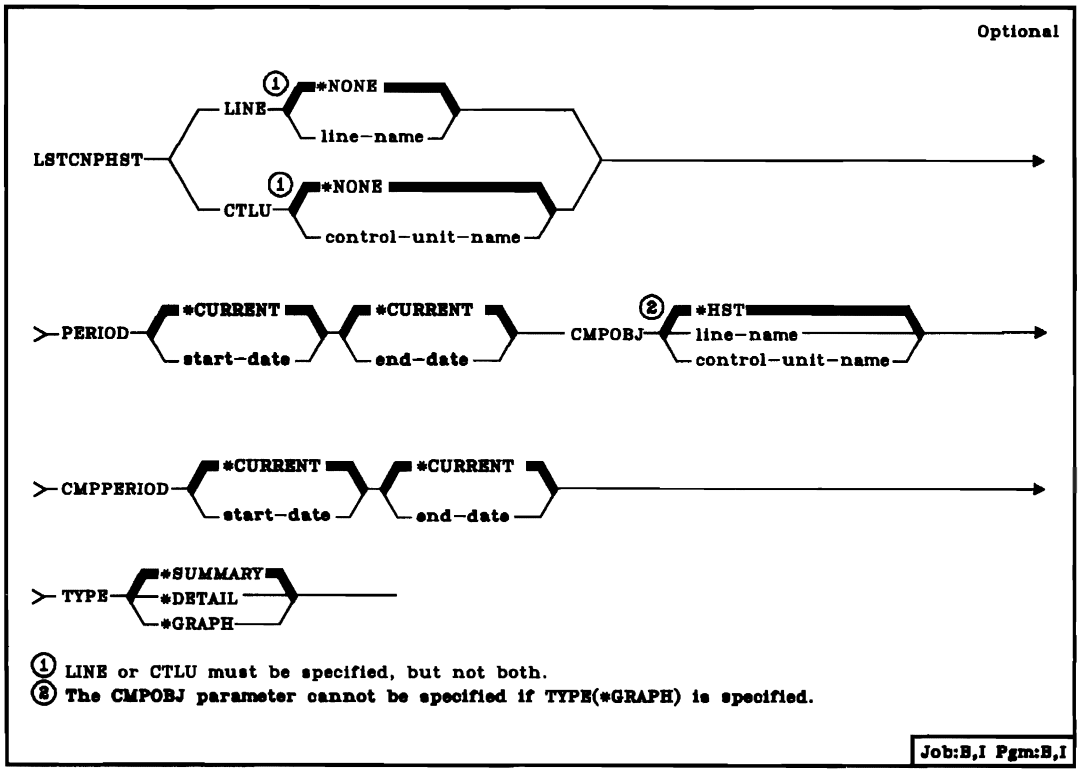
This command lists a graph of the short-term CSNAP data for LN1, that was collected on March 15, 1978.

```
LSTCNPDTA  CTLU(CU1) TYPE(*DETAIL)
```

This command lists a detailed report of short-term CSNAP data collected on the current day for CU1.

LSTCNPHST (List CSNAP History) Command

The List CSNAP History (LSTCNPHST) command places a formatted listing of the CSNAP (communications network statistical analysis procedure) history data in a spooled printer device file named QPCSMPRT. For more information on the use of this command, refer to the *IBM System/38 Problem Determination Guide*, SC21-7726.



LINE Parameter: Specifies which line is to have its CSNAP history data listed.

*NONE: CSNAP data is not to be listed for any lines. If *NONE is specified, then a control unit name must be specified.

line-name: Enter the name of the line that is to have its CSNAP history data listed. The name can be of any line that has a line description. If a line name is specified, then *NONE must be specified on the CTLU parameter.

CTLU Parameter: Specifies which control unit is to have its CSNAP history data listed.

*NONE: CSNAP data is not to be listed for any control unit. If *NONE is specified, then a line name must be specified.

control-unit-name: Enter the name of the control unit that is to have its CSNAP history data listed. The name can be of any control that has a control unit description. If a control unit name is specified, then *NONE must be specified on the LINE parameter.

PERIOD Parameter: Specifies the period of time for which the specified CSNAP data is to be listed. The following values can be coded in this parameter:

*CURRENT: The CSNAP data that is available for the current day is to be listed.

beginning-date: Enter the starting date for which CSNAP data is to be listed. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

ending-date: Enter the ending date for which CSNAP data is to be listed. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

CMPOBJ Parameter: Specifies which line or control unit is to be used to compare the CSNAP data. This parameter is not valid if TYPE(*GRAPH) is specified.

*HST: CSNAP data is to be compared against history.

line-name: Enter the name of the line that is to be used to compare the CSNAP data. The name can be of any line that has a line description. A line-name can be specified only if a line-name was specified on the LINE parameter.

control-unit-name: Enter the name of the control unit that is to be used to compare the CSNAP data. The name can be of any control unit that has a control unit description. A control-unit-name can be specified only if a control-unit-name was specified on the CTLU parameter.

CMPPERIOD Parameter: Specifies the period of time to be used to compare CSNAP data. CMPPERIOD cannot be specified if *HST was specified on CMPOBJ. The following values can be coded in this parameter:

*CURRENT: The CSNAP data that is available for the current day is to be compared.

beginning-date: Enter the starting date for which CSNAP data is to be compared. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

ending-date: Enter the ending date for which CSNAP data is to be compared. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

TYPE Parameter: Specifies the report to be generated.

<u>*SUMMARY:</u> A summary of the CSNAP history data is to be generated.

**DETAIL:* A detail listing of the CSNAP history data is to be generated.

**GRAPH:* A graph of CSNAP history data is to be generated.

Examples

```
LSTCNPHST  LINE(ln1)  PERIOD((031578 031578)
```

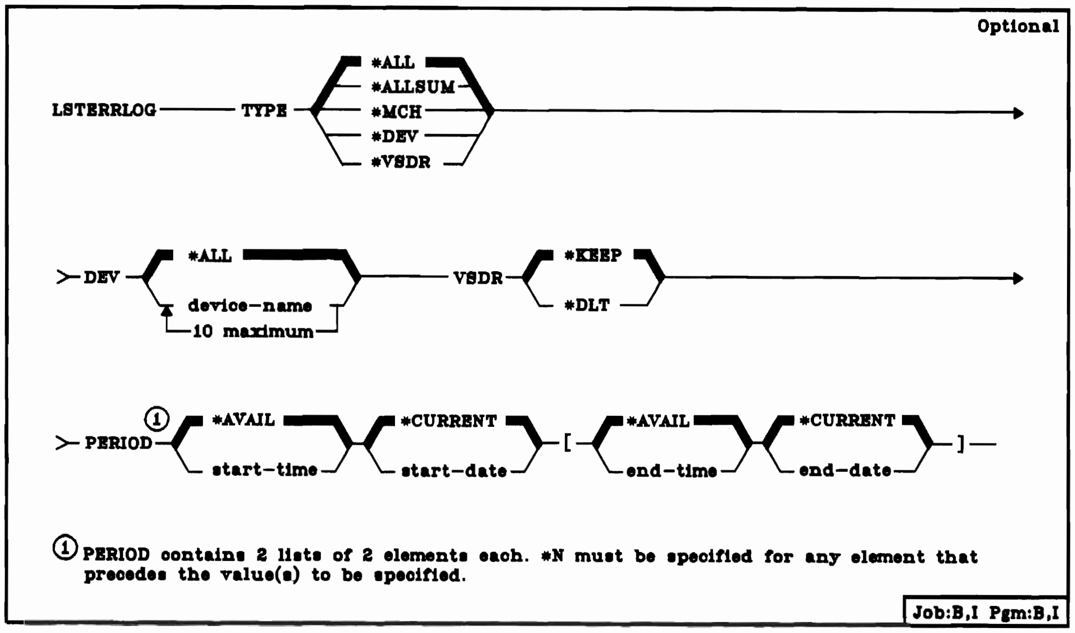
This command lists a summary of the CSNAP history data for LN1 on March 15, 1978 compared with the history of LN1.

```
LSTCNPHST  CTLU(CU1)  CMPCTLU(CU2)  TYPE(*DETAIL)
```

This command lists a detail report of CSNAP history data collected on the current day for CU1 compared with the data collected on the current day for CU2.

LSTERRLOG (List Error Log) Command

The List Error Log (LSTERRLOG) command is used primarily for problem determination. It places a formatted listing of the data in the machine error log in a spooled printer device file named QPCSMPRT. The error log data can be used by the IBM service representatives.



TYPE Parameter: Specifies the type of error log data from the machine error log that is to be listed in the spooled printer file.

<u>*ALL:</u> All the error codes in the machine's error log are to be listed in summary form.

**ALLSUM:* All the data in the error log is to be listed in summary form.

**MCH:* Only the error data produced by machine checks is to be listed.

**DEV:* Only the error data produced by the devices specified in the DEV parameter is to be listed.

**VSDR:* Only the volume statistical data records are to be listed.

DEV Parameter: Specifies which devices are to have their error log data listed from the error log. This parameter is valid only if TYPE(*DEV) is specified.

*ALL: All the error log data for all the devices on the system is to be listed.

device-name: Enter the names of one or more devices that are to have their error log data listed. The names can be of any device for which there is a device description. Also, one or more of the following IBM-defined names of the 62PC and 3370 online storage devices can be entered (the storage units, which have no device descriptions, must be in the system to be valid):

62PC1	62PC4	33701	33704	33707
62PC2	62PC5	33702	33705	33708
62PC3	62PC6	33703	33706	

VSDR Parameter: Specifies whether the volume statistical data records are to be retained or deleted from the machine error log after they are listed.

*KEEP: The volume statistical data records are to be retained in the error log after they are listed.

**DLT:* The volume statistical data records are to be deleted from the error log for nonactive volumes after they are listed.

PERIOD Parameter: Specifies the period of time for which the specified error log data is to be listed. The following values can be coded in this parameter, which contains two lists of two values each. Refer to the syntax diagram for where each value is specified. If PERIOD is not specified, the following values are assumed:

PERIOD((*AVAIL *CURRENT) (*AVAIL *CURRENT))

*AVAIL: The error data that is available for the specified starting or ending date is to be listed.

*CURRENT: The error data that is available for the current day and between the specified starting and ending times (if specified) is to be listed.

start-time: Enter the starting time for the specified starting date that specifies the error data to be listed. The time can be entered as four or six digits (hhmm or hhmmss) where hh = hours, mm = minutes, and ss = seconds. If colons are used to separate the time values, the string must be enclosed in apostrophes ('hh:mm:ss').

start-date: Enter the starting date for which error data is to be listed. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

end-time: Enter the ending time for the specified ending date that specifies the error data to be listed. The time can be entered as four or six digits, and colons can also be used ('hh:mm:ss').

end-date: Enter the ending date for which error data is to be listed. The date must be entered in the format specified by the system values QDATFMT and, if separators are used, QDATSEP.

Examples

```
LSTERRLOG  TYPE(*DEV)  DEV(MFCU)  +
      PERIOD((*AVAIL 031578)(*N 031578))
```

This command lists, from the machine error log, the error log data that was caused by the MFCU device. The MFCU error data recorded on March 15, 1978, is placed in the spooled printer file for printing.

```
LSTERRLOG  PERIOD(('08:00:00' '03/15/78') ('17:00:00' '03/21/78'))
```

This command lists all recorded error log data (of all types) that occurred from 8 a.m. on March 15, 1978 through 5 p.m. on March 21, 1978. The times and dates are enclosed in apostrophes in this example because colons and slashes are used as separators.

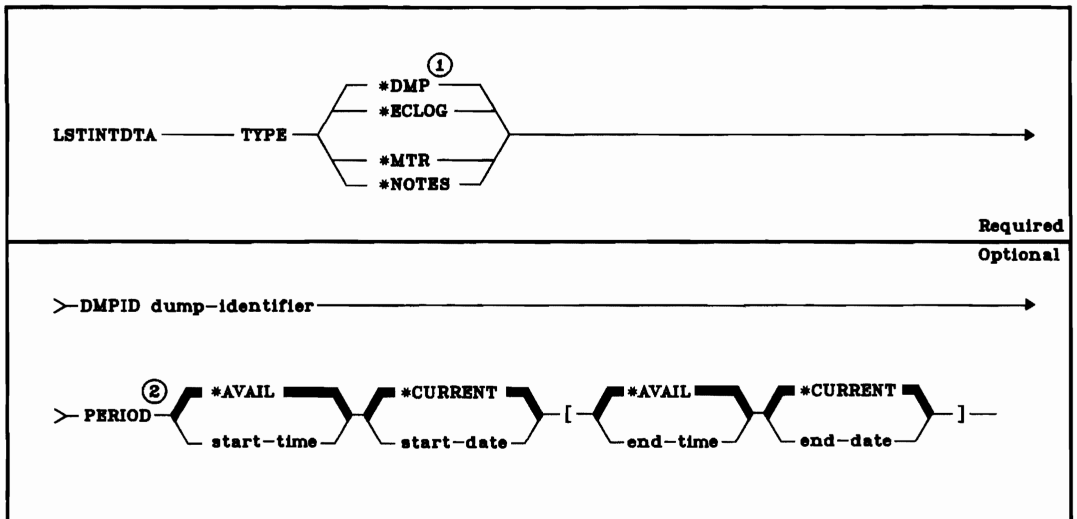
```
LSTERRLOG  TYPE(*VSDR)  VSDR(*DLT)
```

This command lists all the volume statistical data records from the error log to the spooled printer file. After the records are listed, they are deleted from the machine error log.

LSTINTDTA (List Internal Data) Command

The List Internal Data (LSTINTDTA) command is used primarily for problem determination. It writes the machine internal data to a spooled printer file. The data is to be used to service the system. (For the names of files produced by this and other commands, see Appendix D.)

Restriction: This command is intended for use by service representatives only.



① If *DMP is specified, the DMPID parameter must be specified.
② PERIOD contains 2 lists of 2 elements each. *N must be specified for any element that precedes the value(s) to be specified. PERIOD is valid only if TYPE(*NOTES) is specified.

Job:B,I Pgm:B,I

TYPE Parameter: Specifies the type of data that is to be printed.

**DMP:* The data to be printed was dumped by a previously issued DMPJOBINT (Dump Job Internal) command or by the machine when it was processing a machine check or object damage. The dump identifier of the data must be specified in the DMPID parameter.

**ECLOG:* The current log of machine engineering changes is to be printed.

**MTR:* The machine trouble report is to be printed.

**NOTES:* The notes portion of the machine internal data, for the period specified by the PERIOD parameter, is to be printed.

DMPID Parameter: Specifies, for internal dumps only, the dump identifier associated with the machine internal data that is to be printed. This parameter must be specified *only* if TYPE(*DMP) is specified; otherwise, it is ignored. The dump identifier is sent: in a message to the job that issued the DMPJOBINT command that dumped the internal data, in a damage message, or in a machine check or machine function check message. (The message containing the dump identifier may also be sent to the system history log.)

Enter the dump identifier of the dump that is to be printed. The identifier specified must contain 8 characters.

PERIOD Parameter: Specifies the period of time for which the notes portion of the machine internal data is to be printed. This parameter is valid only if TYPE(*NOTES) is specified; otherwise, it is ignored. The following values can be coded in this parameter, which contains two lists of two values each. Refer to the syntax diagram for where each value is specified. If this parameter is not specified, all the available notes for the current date are printed.

*AVAIL: The notes that are available from the starting date to the ending date (or for the current day only) are to be printed.

*CURRENT: The notes that are available for the current day and between the specified starting and ending times (if specified) are to be printed.

start-time: Enter the starting *time* for the specified starting date that specifies the notes to be printed.

start-date: Enter the starting *date* for which notes are to be printed. (The system date format must be used.)

end-time: Enter the ending *time* for the specified ending date that specifies the notes to be printed.

end-date: Enter the ending *date* for which notes are to be printed. (The system date format must be used.)

Example

```
LSTINTDTA  TYPE(*DMP)  DMPID(0102FA3C)
```

This command prints the job internal dump that has a dump identifier of 0102FA3C.

MONMSG (Monitor Message) Command

The Monitor Message (MONMSG) command is used to monitor escape, notify, and status messages sent to the program message queue of the program in which the command is used. Completion and diagnostic messages cannot be monitored.

When the MONMSG command is compiled in a CL program, it establishes a monitor for the arrival of the specified message(s). The command monitors the messages for the condition specified by the comparison data given in the command. If a message meeting the conditions arrives on the message queue, the CL command specified on the MONMSG command is executed.

A maximum of 1000 MONMSG commands can be specified in a program to monitor the arrival of messages for specific conditions or a group of conditions. Specific message identifiers or generic message identifiers can be monitored. See Appendix E, *Error Messages That Can Be Monitored*, for the escape, notify, and status messages, and their identifiers, that can be sent by the CL commands.

The MONMSG command can be coded following most commands in a CL program. A MONMSG command that is not placed at the beginning of the program applies only to the immediately preceding command; this is called a *command-level* MONMSG command. That is, the command-level MONMSG command monitors only those messages sent by the previous command and no others; if the message sent by that command meets the conditions specified in the MONMSG command, the action specified in the same MONMSG command is taken. As many as 100 MONMSG commands, coded immediately after a command, can monitor the messages sent by that command.

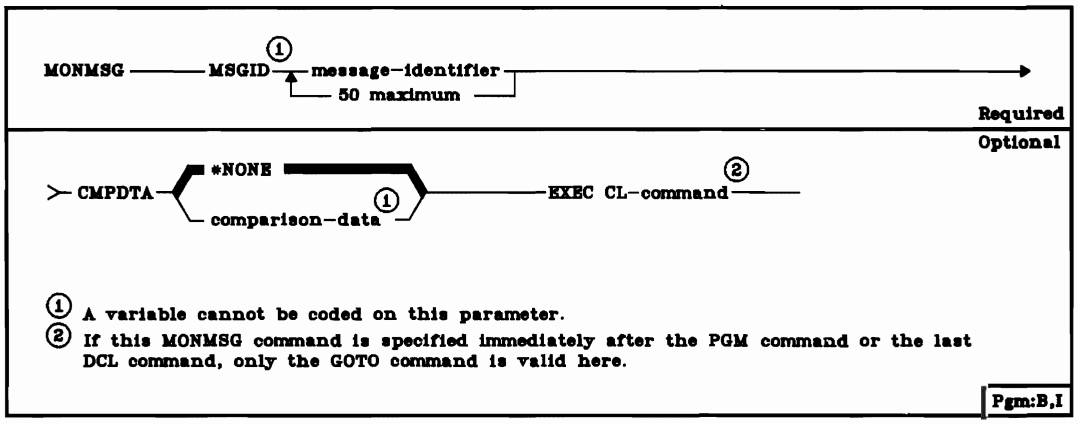
When the action specified in the MONMSG command has been performed, and that action does not end with a GOTO or RETURN command, control returns to the command in the program that follows the command that sent the message. If the action ends with a GOTO command, control branches to the command in the program specified in the GOTO command. If the action ends with a RETURN command, control returns to the program that called the program containing the MONMSG command.

If one or more MONMSG commands occur at the beginning of the program, immediately following the declare commands or the PGM command if there are no declare commands, they monitor all messages sent by all of the commands in the program (maximum of 100). This is called a *program-level* MONMSG command. If any message sent by any command in the program meets the conditions specified in any one of the program-level MONMSG commands, the corresponding action specified in the same command is taken.

The action taken by a command-level MONMSG command overrides a program-level MONMSG command.

If a command is coded for the EXEC parameter on a MONMSG command that is placed at the beginning of a program, *only* the GOTO command may be used, and it must specify the label for the command to which control is to be passed if a monitored message occurs. If a command is not coded for the EXEC parameter, any monitored messages are ignored.

Restrictions: This command is valid only within CL programs. It can be coded after the last declare command (if declare commands are used), following the PGM command that begins the program, or it can be coded following any command allowed in CL programs, except for the following: DO, ELSE, ENDDO, ENDPGM, GOTO, IF, or RETURN. Note that if another program sends a message that is monitored by this command, a return cannot be made to that program.



① A variable cannot be coded on this parameter.
② If this MONMSG command is specified immediately after the PGM command or the last DCL command, only the GOTO command is valid here.

MSGID Parameter: Specifies the message identifiers of one or more escape, notify, or status messages that are to be monitored by this command. As many as 50 specific and/or generic message identifiers can be specified on one command.

Note: Many CL commands issue one escape message for many different error conditions. Details about the error or failure are given in diagnostic messages that precede the escape message. Although diagnostic messages cannot be monitored, they can be received from the job's external message queue after the escape message has activated the user's message monitor.

The message identifiers must each be 7 characters long and in the following format: *pppmmnn*. The first 3 characters must be an alphabetic program code, and the last 4 characters must be a decimal number. If zeros are specified in either two or all four of the rightmost positions, such as ppmm00, a generic message identifier is being specified. For example, if CPF0000 were specified, all the CPF messages would be monitored. (See the *IBM System/38 Messages Guide: CPF, RPG III, IDU.*)

Enter the message identifiers of one or more messages that are to be monitored when they arrive at this program's message queue. (For the identifiers of the escape, notify, and status messages that can be sent by the CL commands, refer to Appendix E in Part 3.) CL variables cannot be used to specify any message identifiers.

CMPDTA Parameter: Specifies the comparison data that is to be used to determine whether the monitored message (having one of the specified message identifiers) received on the program's message queue is to be acted upon by this command. The message data specified in the MSGDTA parameter of the SNDPGMMSG command is compared to this comparison data. If the first part (up through the first 28 characters, or less) of the message's substitution values exactly matches the comparison data specified here, the action specified in the EXEC parameter of this command is taken. The action is also taken if no comparison data is specified.

*NONE: No comparison data is specified; if the message in the program's message queue is from a command that this command is monitoring, and has the specified identifier, the action specified by EXEC is taken.

comparison-data: Enter a character string of no more than 28 characters, enclosed in apostrophes if necessary, that is to be compared with the same number of characters in the message data of the received message, starting with the first character in the message data. If the comparison data matches the first part of the received message data, this command performs the function specified in the EXEC parameter. (A CL variable cannot be specified for the comparison data.)

The comparison data can be displayed by the DSPPGMVAR command. Refer to the third display shown under *Additional Considerations* in that command's description.

EXEC Parameter: Specifies the CL command to be executed when a monitored message sent to the program's message queue meets the conditions specified in this MONMSG command. If no command is specified and a monitored message arrives on the queue, the message is ignored, and control passes to the next command in the program.

If the MONMSG command is placed at the beginning of the program, the EXEC parameter must specify the GOTO command and the label identifying the command that will receive control.

Enter the CL command, including its parameters to be used, that is to be executed when a message meeting the conditions specified in this command is received. The command specified here is not executed if the received message does not meet the specified conditions. A CL variable cannot be specified here in place of the CL command that is to be executed.

Note: If a DO command is specified on EXEC, the entire do group associated with the DO command is executed if the condition is met.

Examples

```
PGM
MONMSG  MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
```

This example shows a MONMSG command at the beginning of a CL program that monitors for the messages CPF0001 and CPF1999; these messages might be sent by any command executed later in the program. When either message is received from any of the commands executing in the program, control branches in the program to the command identified by the label EXIT2.

CPF0001 states that an error was found in the command that is identified in the message itself. CPF1999, which can be sent by many of the debugging commands (like CHGPGMVAR), states that errors occurred on the command, but it does not identify the command in the message. (In Appendix E, CPF0001 can be found under *All Commands* at the beginning of the appendix, and CPF1999 can be found under several of the debugging commands.)

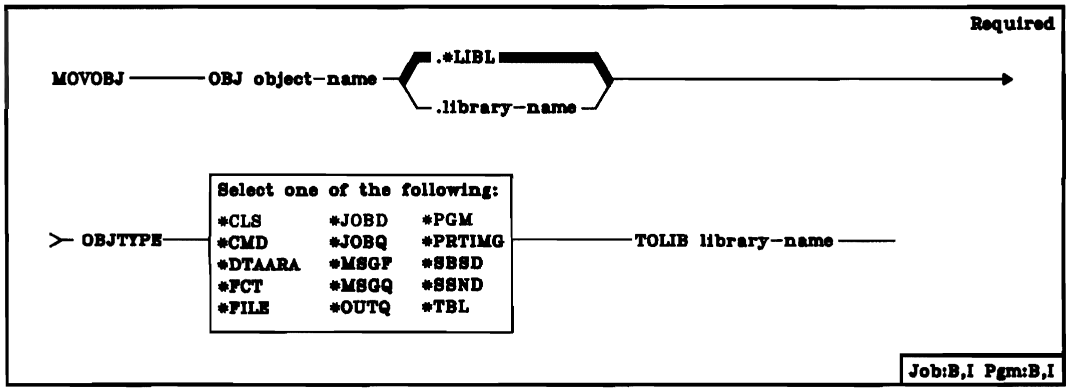
```
CHGVAR  VAR(&A) VALUE(&A/&B)
MONMSG  MSGID(MCH1211) EXEC(CHGVAR VAR(&A) VALUE(1))
```

In this example, the MONMSG command follows a CHGVAR (Change Variable) command and, therefore, is only monitoring messages sent by the CHGVAR command. The MI escape message MCH1211 is sent to this program's message queue when a division by zero is attempted. Because MSGID(MCH1211) is specified, the MONMSG command is monitoring for this condition; when it receives the message, the second CHGVAR command is executed. In this command, the variable &A is set to a value of 1.

MOVOBJ (Move Object) Command

The Move Object (MOVOBJ) command removes an object from its currently assigned library and places it in a different library. The type of the object to be moved is specified in the OBJTYPE parameter.

Restrictions: (1) The user who submits this command must have object management rights for the object to be moved, add rights for the receiving library, and delete rights for the library from which the object is to be moved and operational rights for both libraries. (2) Libraries, user profiles, edit descriptions, line descriptions, control unit descriptions, device descriptions, journals, and journal receivers cannot be moved. (3) The following objects cannot be moved: the system operator message queue QSYSOPR, all work station user message queues, and the system logs QHST, QSRV, and QCHG. (4) The library to which the object is being moved must not already contain an object of the same name and type as the object being moved.



OBJ Parameter: Specifies the qualified name of the object to be moved to another library. (If no library qualifier is given, *LIBL is used to find the object.) The object name should be qualified to ensure that the correct object is moved.

OBJTYPE Parameter: Specifies the type of the object to be moved to another library. Enter one of the following CPF object types:

Value	**Object Type**
*CLS	Class
*CMD	Command
*DTAARA	Data area
*FCT	Forms control table
*FILE	File
*JOBD	Job description
*JOBQ	Job queue
*MSGF	Message file
*MSGQ	Message queue
*OUTQ	Output queue
*PGM	Program
*PRTIMG	Print image
*SBSD	Subsystem description
*SSND	Session description
*TBL	Table

TOLIB Parameter: Specifies the name of the library into which the object is to be moved. The library QTEMP cannot be specified.

Examples

```
MOVOBJ  OBJ(X.QGPL) OBJTYPE(*PGM) TOLIB(MY)
```

The general purpose library is searched for the program (*PGM) object X. Before X is moved to MY library, the user profile of the user submitting the command is checked for (1) object management rights for the object, (2) add authority for MY library, and (3) delete rights for the library from which the object X is to be moved. After this command is executed, object X is no longer in the QGPL library.

```
MOVOBJ  OBJ(Y.*LIBL) OBJTYPE(*FILE) TOLIB(Z)
   or
MOVOBJ  Y *FILE Z
```

The library list (*LIBL) is used to locate the file named Y. If more than one object with the same name exists within the libraries making up the library list, the first object found via the library list for which the user has object management rights is moved to library Z.

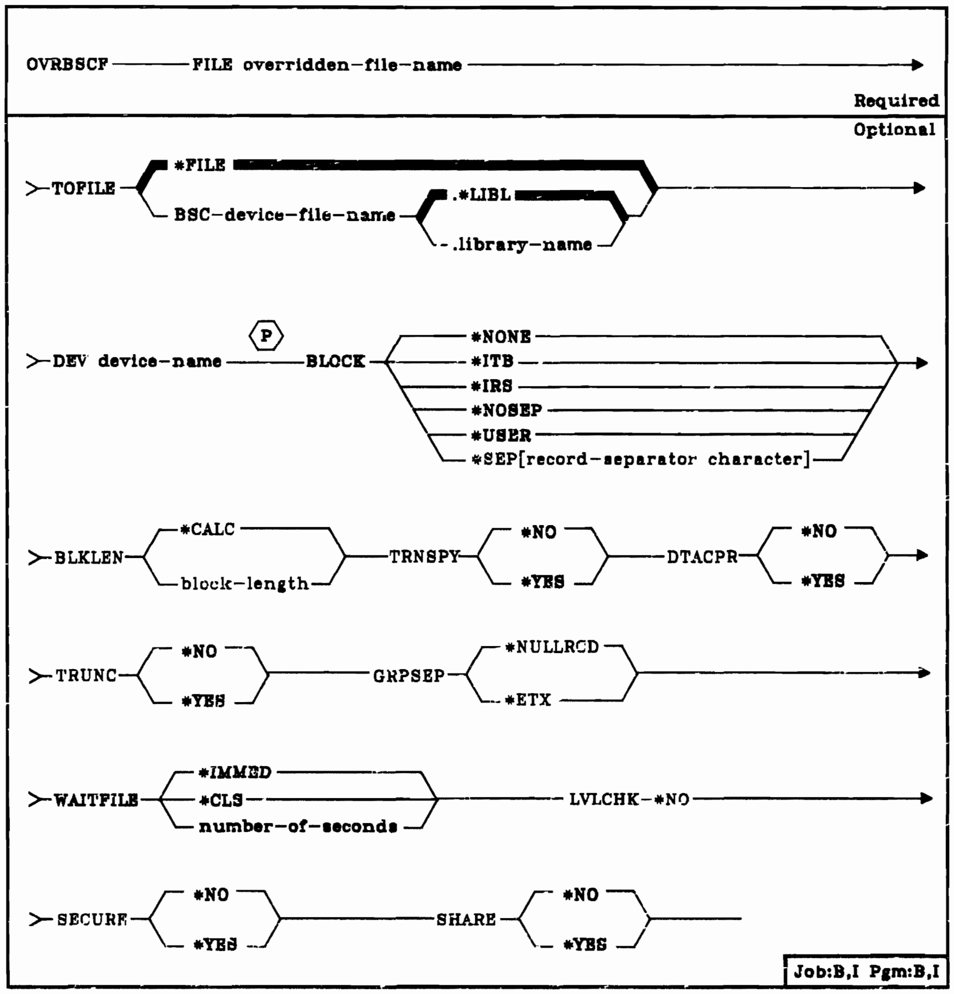
OVRBSCF (Override with BSC File) Command

The Override with BSC File (OVRBSCF) command can be used to (1) override (replace) the file named in the program, (2) override certain parameters of a file that is used by the program, or (3) override the file named in the program *and* override certain parameters of the file to be processed. Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executing in the following invocations. The parameters are selected on the basis of the type of BSC device to be used with the device file.

If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter and the name of the overriding file (the file to be processed) is specified in the TOFILE parameter. The OVRBSCF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRBSCF command. Any parameters that are not specified do not affect the parameters specified in the file description, in the program, and/or in other file override commands executed in the following invocations.

A file override command must be executed before the file to be overridden is used. The file is overridden if it is used in the program containing the file override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in a following invocation. A file override command applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). A file override can be explicitly deleted by a DLTOVR command executed in the same invocation as the override, or it is implicitly deleted when the invocation that issues the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied to it if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed in an invocation following the invocation for this override, before the file is used.



FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a BSC device file when TOFILE specifies *FILE. Otherwise, any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the BSC file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRBSCF command will override the same parameters specified in the BSC device file, in the program, and/or in other OVRBSCF commands executed in following invocations.

<u>*FILE</u>: The BSC device file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-BSC-device-file-name: Enter the qualified name of the BSC device file that is to be used instead of the overridden file. (If no library qualifier is given, *LIBL is used to find the BSC device file description.)

DEV Parameter: Specifies the name of the System/38 BSC device to be used with the BSC device file. Enter the device name that overrides the device name specified in the BSC device file, in the program, and/or in other OVRBSCF commands executed in following invocations.

BLOCK Parameter: Specifies whether the system will block and deblock transmitted records. If a parameter other than *NONE or *USER is selected, records will be blocked as required by the system for output and deblocked on input. Blocking may be done using a fixed length or by means of record separators. If TRNSPY(*YES) is specified, the records must be blocked using the fixed length parameter value BLOCK(*NOSEP); the length the records will be blocked is determined with the device file. If TRNSPY(*NO) is specified, record blocking may be performed using either a fixed length or by means of a record separator character specified by one of the other values of this parameter. This parameter overrides the BLOCK parameter specified in the BSC device file, in the program, and/or in other OVRBSCF commands executed in other invocations.

**NONE:* Specifies that no blocking or deblocking will be done by the system.

**ITB:* Specifies that the records are to be blocked or deblocked based on the location of an ITB (intermediate text block) control character. For input files, a record will be delimited by locating the next ITB character. The ITB character will not be passed to your program. An ETX (end of text) or ETB (end-of-transmission block) character will be used as an ITB character to delimit records. For output files, an ITB character will be inserted after the record. If that character is the last character of the block, the ITB character will be replaced by an ETX or ETB character.

**IRS:* Specifies that the records are to be blocked or deblocked based on the location of an IRS (interrecord separator) character. For input files, a record will be delimited by locating the next IRS character. The IRS character will not be passed to your program. For output files, an IRS character will be inserted after the record.

**NOSEP:* Specifies that no record separator character is contained within the transmission block sent to or received from the device. The system will block and deblock the records according to a fixed-length record, as specified in the DDS format specifications.

**USER:* Specifies that your program is to provide all record separators, BSC framing characters, transparency characters, and so forth necessary to successfully transmit the records.

When transmitting records, BSC device support will scan the buffer for the last non-blank byte to determine the length of the data to be transmitted. For this reason, you must ensure that the unused portion of the buffer contains blanks.

For receiving, you must specify with an ETX control character the end of the received text. BSC device support will pad the remaining buffer space with blanks.

This method of blocking provides you the ability to transmit and receive variable-length data blocks by using a single record format capable of accommodating the maximum block length. Except for the padding and truncating with blanks, BSC device support passes the data to and from the system when user blocking is specified.

Before selecting this option, you should have a good understanding of the device and of the BSC support characteristics. For more information on BSC support characteristics, refer to the *IBM System/38 Data Communications Programmer's Guide,* SC21-7825.

If you are using the Remote Job Entry Facility, BLOCK(*USER) must be specified. For more information on RJEF, refer to the *RJEF Programmer's Guide.*

**SEP:* Specifies that the records are to be blocked or deblocked, based on the location of a record separator character. For input files, a record will be delimited by locating the next record separator character. The record separator character will not be passed to your program. For output files, a record separator character will be inserted after the record.

record-separator-character: Specifies a unique one-byte record separator character. The record separator character may be specified as two hexadecimal characters, as in BLOCK(*SEP X'FD'), or as a single character, as in BLOCK(*SEP @). If a record separator character is not specified, the record separator character from the device file is used.

The following is a list of BSC control characters that must not be used as record separator characters:

EBCDIC	**BSC Control**
X'01'	SOH (Start of header)
X'02'	STX (Start of text)
X'03'	ETX (End of text)
X'10'	DLE (Data link escape)
X'1D'	IGS (Interchange group separator)
X'1F'	ITB (Intermediate text block)
X'26'	ETB (End-of-transmission block)
X'2D'	ENQ (Enquiry)
X'32'	SYN (Synchronization)
X'37'	EOT (End of transmission)
X'3D'	NAK (Negative acknowledgment)

You must be certain that the record separator character does not occur in your data.

BLKLEN Parameter: Specifies the maximum block length (in bytes) for data that is to be transmitted. This parameter overrides the block length specified in the BSC device file, in the programs, and/or in other OVRBSCF commands executed in following invocations.

**CALC:* The block length is to be determined by the system. The length will be 512 or the length of the largest record in the device file, whichever is greater.

block-length: The maximum block length of records to be sent when using this device file. The value must be at least the size of the largest record to be sent. Valid values are 1 through 8192.

TRNSPY Parameter: Specifies whether the text transparency feature is to be used when sending blocked records. This parameter should be specified when transmitting packed or binary data fields. This parameter overrides the TRNSPY value specified in the BSC device file, in the program, and/or in other OVRBSCF commands executed in following invocations.

**NO:* The text transparency feature is not to be used.

**YES:* The text transparency feature is to be used, which permits the use of all 256 EBCDIC character codes. *YES is valid only when BLOCK(*NONE), BLOCK(*NOSEP), or BLOCK(*USER) is specified.

Note: Transparency of received data is determined by the data stream; therefore, this parameter is not relevant for received data. If TRNSPY(*YES) is specified with BLOCK(*USER), BSC ignores the transparency indicator during put operations. You must provide the proper controls with the data in order to get transparent transmission of data.

DTACPR Parameter: Specifies whether blanks in BSC data will be compressed for output and decompressed for input. If TRNSPY(*YES) is specified, or if the line description specifies CODE(*ASCII), DTACPR(*YES) is ignored. This parameter overrides the DTACPR value specified in the BSC device file, in the program, or in other OVRBSCF commands executed in following invocations.

**NO:* No data compression or decompression is to occur.

**YES:* Data is to be compressed on output and decompressed on input.

TRUNC Parameter: Specifies whether trailing blanks are to be removed from output records. TRUNC(*YES) cannot be specified if BLOCK(*USER) or TRNSPY(*YES) is specified. This parameter overrides the TRUNC value specified in the BSC device file, in the program, or in other OVRBSCF commands executed in following invocations.

**NO:* Trailing blanks are not to be removed from output records.

**YES:* Trailing blanks are to be removed from output records.

GRPSEP Parameter: Specifies a separator for groups of data (data sets, documents, and so forth). This parameter overrides the GRPSEP value specified in the BSC device file, in the program, or in other OVRBSCF commands executed in following invocations.

**NULLRCD:* Specifies that a null record (STXETX) is to be used as a data group separator.

**ETX:* A transmission block ending with the BSC control character ETX is to be used as a data group separator.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the BSC device file, in the program, or in other OVRBSCF commands executed in the following invocations. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is to be made.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated to the BSC device file. Valid values are 1 through 32767 (32 767 seconds).

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record is given a unique internal system identifier when the format is created.

This parameter overrides the value specified in the BSC device file, in the program, and/or in other OVRBSCF commands executed in following invocations. Level checking cannot be done unless the program contains the record format identifiers. This command cannot override level checking from *NO to *YES.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previous invocations.

**YES:* This file is protected from the effects of any file override commands executed in previous invocations.

SHARE Parameter: Specifies whether the ODP (open data path) for the BSC device file can be shared with other programs in the same routing step. If so, when the same file is opened more than once, the ODP can be shared with other programs in the same routing step that also specify the share attribute. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a write operation in that program produces the next output record. This parameter overrides the value specified in the BSC device file, in the program, and/or in other OVRBSCF commands executed in following invocations.

**NO:* An ODP created by the program with this attribute is not to be shared with other programs in the routing step. Every time a program opens this file with this attribute, a new ODP to the file is created and activated.

**YES:* An ODP created with this attribute is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file. This includes multiple opens in the same program.

Examples

```
OVRBSCF FILE(BSC370F) DEV(BSC370L) WAITFILE(20)
```

This command overrides the device and file wait time values specified in the BSC370F device file description, in the program, and/or in other OVRBSCF commands executed in following invocations. The program that opens the file named BSC370F uses the device named BSC370L. The program will wait up to 20 seconds, if necessary, for the required file resources to be allocated to the BSC370F file.

```
OVRBSCF FILE(BYSNC1) TOFILE(BYSNC3)
```

This command causes the BSC file named BYSNC3 to be used in place of the BSC file named BYSNC1. No parameters of BYSNC3 are to be overridden; it is used as it is currently described on the system.

```
OVRBSCF FILE(TRANSD1.COMM1)  WAITFILE(*IMMED)
```

This command overrides the file wait time values specified in the TRANSD1 device file description, in the program, and/or in other OVRBSCF commands executed in following invocations. The program in which this command occurs is not to wait for file resources to be allocated; if an immediate allocation of file resources cannot be made, an error message is sent to the program.

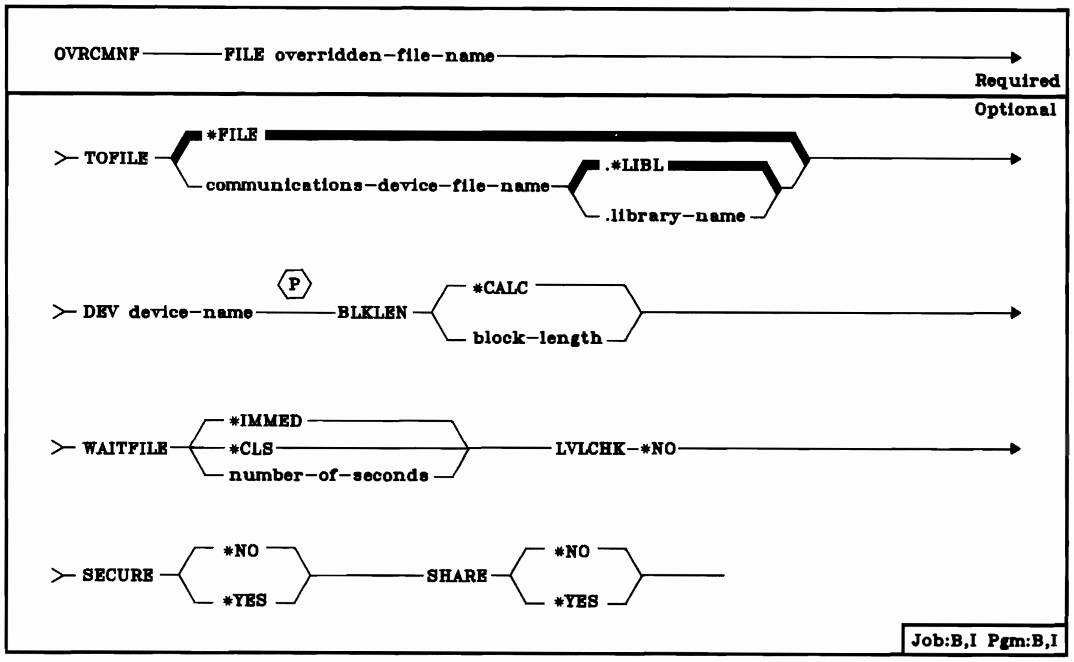
OVRCMNF (Override with Communications File) Command

The Override with Communications File (OVRCMNF) command can be used to (1) override (replace) the file named in the program, (2) override certain parameters of a file that is used by the program, or (3) override the file named in the program *and* override certain parameters of the file to be processed. Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executed in following invocations.

If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter and the name of the overriding file (the file to be processed) is specified in the TOFILE parameter. The OVRCMNF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRCMNF command. Any parameters that are not specified do not affect parameters specified in the file description, in the program, and/or in other file override commands executed in following invocations.

A file override command must be executed before the file to be overridden is used. The file is overridden if it is used in the program containing the override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in following invocations. A file override command applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). An override can be explicitly deleted by a DLTOVR command, executed in the same invocation as the override, or it is implicitly deleted when the invocation that issues the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied to it if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed before the file is opened in an invocation following the invocation for this override.



FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a communications device file when TOFILE specifies *FILE. Otherwise, any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the communications file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRCMNF command will override the same parameters specified in the communications device file or in the program and/or in other OVRCMNF commands executed in following invocations.

FILE: The communications device file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-communications-device-file-name: Enter the qualified name of the communications device file that is to be used instead of the overridden file. (If no library qualifier is given, *LIBL is used to find the communications device file description.)

DEV Parameter: Specifies the name of the System/38 communications device to be used with the communications device file to send and receive data records. Enter the device name that overrides the device name specified in the communications device file, in the program, and/or in other OVRCMNF commands executed in following invocations.

BLKLEN Parameter: Specifies, in bytes, the maximum block length for data that is to be transmitted or received by the communications file. This parameter overrides the block length specified in the device file, in the program, and/or in other OVRCMNF commands executed in following invocations.

**CALC:* The device support chooses an optimum value based on the record sizes in the device file. Device support calculates the smallest multiple of 1792 that is greater than or equal to the largest record in the device file. The calculated value includes the new line (NL) or form feed (FF) characters that follow each record when RCDSEP(*YES) is specified.

block-length: Enter a value (256 through 32767) that specifies the maximum block length of records to be processed by this communications device file. This value must be at least the size of the largest message expected to be transmitted or received. Also, it must include the new line (NL) or form feed (FF) characters that follow each record when RCDSEP(*YES) is specified.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the device file, in the program, and/or in other OVRCMNF commands executed in following invocations. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given a unique internal system identifier when the format is created.

This parameter overrides the value specified in the device file, in the program, and/or in other OVRCMNF commands executed in following invocations. Level checking cannot be done unless the program contains the record format identifiers. This command cannot override level checking from *NO to *YES.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previous invocations.

**YES:* This file is protected from the effects of any file override commands executed in previous invocations.

SHARE Parameter: Specifies whether the ODP (open data path) for the communications device file can be shared with other programs in the same routing step. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next record for the file. A write operation produces the next output record. This parameter overrides the value specified in the device file, in the program, and/or in other OVRCMNF commands executed in following invocations.

**NO:* An ODP created by the program is not shared with other programs in the routing step. Every time a program opens this file, a new ODP to the file is created and activated.

**YES:* If the file is opened more than once, the same ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file. This includes multiple opens in the same program.

Examples

```
OVRCMNF  FILE(CMN370F) DEV(CMN370L) WAITFILE(20)
```

This command overrides the device and file wait time values specified in the CMN370F device file description, in the program, and/or in other OVRCMNF commands executed in following invocations. The program that opens the file named CMN370F uses the device named CMN370L. The program will wait, if necessary, up to 20 seconds for the required file resources to be allocated to the CMN370F file.

```
OVRCMNF  FILE(COMM1) TOFILE(COMM3)
```

This command causes the communications file named COMM3 to be used in place of the communications file COMM1. No parameters of COMM3 are to be overridden; it is to be used as it is already described on the system.

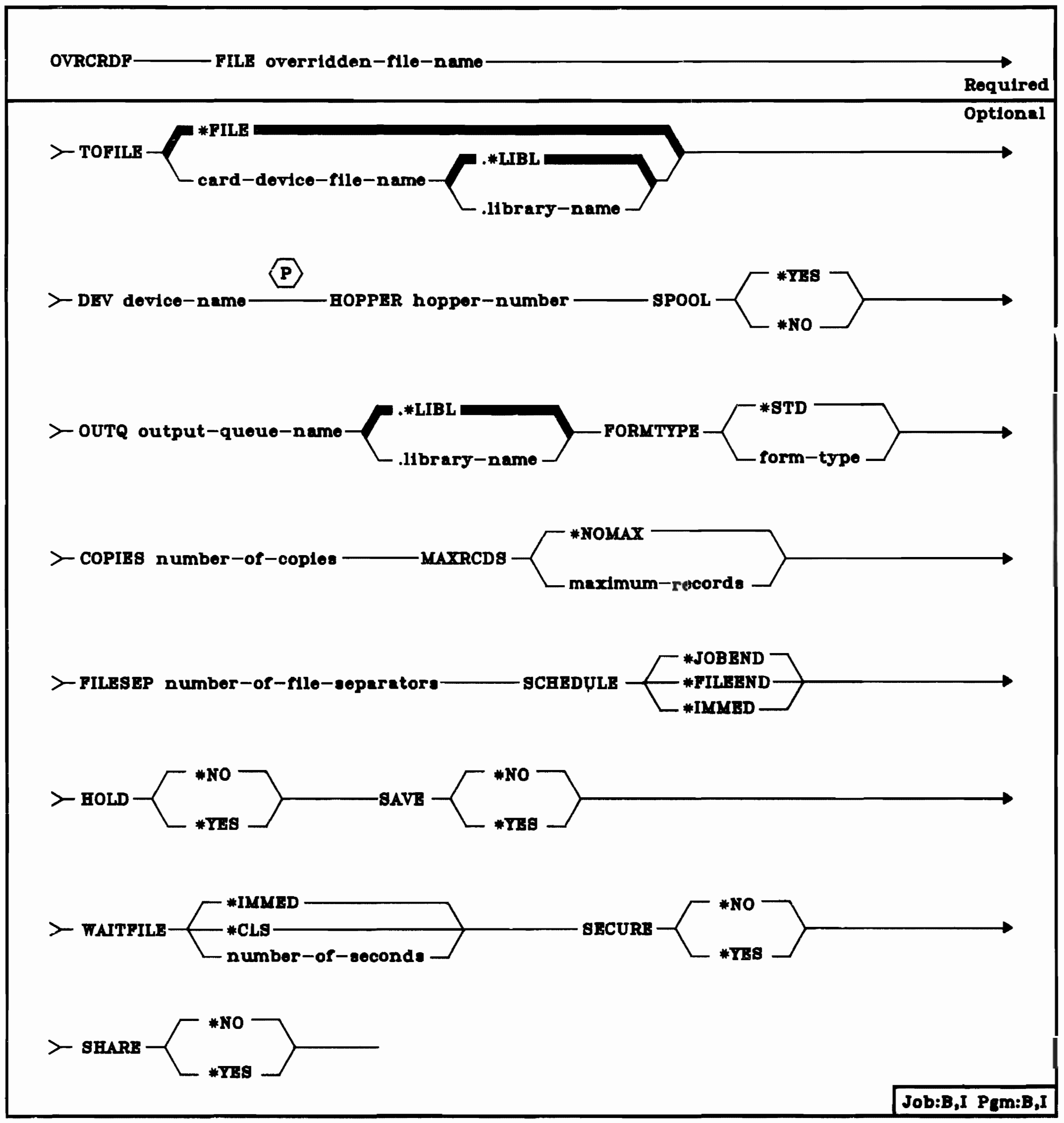
OVRCRDF (Override with Card File) Command

The Override with Card File (OVRCRDF) command can be used to (1) override (replace) the file named in the program, (2) override certain parameters of a file that is used by the program, or (3) override the file named in the program *and* override certain parameters of the file to be processed. Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executed in following invocations.

If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter and the name of the overriding file (the file to be processed) is specified in the TOFILE parameter. The OVRCRDF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRCRDF command. Any parameters that are not specified do not affect parameters specified in the file description, in the program, and/or in other file override commands executed in following invocations.

A file override command must be executed before the file to be overridden is used. The file is overridden if it is used in the program containing the override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in following invocations. A file override applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). An override can be explicitly deleted by a DLTOVR command, executed in the same invocation as the override, or it is implicitly deleted when the invocation that issued the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed before the file is opened in an invocation following the invocation for this override.



FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a card device file when TOFILE specifies *FILE. Otherwise. any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the card device file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRCRDF command will override the same parameters specified in the card device file, in the program, and/or in other OVRCRDF commands executed in following invocations.

*FILE: The card device file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-card-device-file-name: Enter the qualified name of the card device file that is to be used instead of the file specified in the FILE parameter. (If no library qualifier is given, *LIBL is used to find the device file description.)

DEV Parameter: Specifies the name of the card device to be used with the card device file to perform input/output operations. Enter the device name that overrides the device name specified in the card device file, in the program, and/or in other OVRCRDF commands executed in following invocations. The device name of the IBM-supplied card device description is QCARD96. This parameter is ignored if SPOOL(*YES) is in effect for the file when it is opened.

Note: Only 64 of the 256 EBCDIC characters can be read and punched on the MFCU. Therefore, an output file (such as the job log) that might contain characters other than the 64 supported characters should not be redirected to a card device file that uses the MFCU, or punch checks may occur on the MFCU.

HOPPER Parameter: Specifies from which hopper of the MFCU the cards are to be fed. Enter either a 1 or a 2 to indicate the hopper number that overrides the value in the device file, in the program, and/or in other OVRCRDF commands executed in following invocations.

SPOOL Parameter: Specifies whether the input or output data for the card device file is to be spooled. This parameter overrides the value specified in the device file, and/or in other OVRCRDF commands executed in following invocations.

Note: If SPOOL(*NO) is the current value in the card device file, or if SPOOL(*NO) is specified in this or any other OVRCRDF command that is in effect when the file is opened, then the following parameters that apply to spooled output files are ignored: OUTQ, FORMTYPE, COPIES, MAXRCDS, FILESEP, SCHEDULE, HOLD, and SAVE.

**YES:* The data is to be spooled. If the file is opened for input, the named inline data file or the next unnamed inline spooled file is processed. (For information on inline files, refer to the *CPF Programmer's Guide*.) If the file is opened for output, the data is spooled for processing by a spooling writer.

**NO:* The data is not to be spooled. If this file is opened for input, the data is read directly from the card device. If this is an output file, the data is sent directly to the device to be punched or printed as the output becomes available.

OUTQ Parameter: Specifies for spooled output only, the name of the output queue for the spooled output file. Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.) This parameter overrides the output queue name specified in the card device file, and/or in other OVRCRDF commands executed in following invocations.

FORMTYPE Parameter: Specifies, for spooled output card files only, the type of form (cards) on which the card device is to produce the output. Enter either *STD or a user-defined identifier, having 10 alphameric characters or less, that identifies the card type to be used for output. This parameter overrides the form type value specified in the device file, and/or in other OVRCRDF commands executed in following invocations.

COPIES Parameter: Specifies, for spooled output only, the number of copies (card decks) of the output to be produced by the card device. Enter a value, 1 through 99, that indicates the number of identical card decks to be produced. This parameter overrides the copies value specified in the device file, and/or in other OVRCRDF commands executed in following invocations.

MAXRCDS Parameter: Specifies, for spooled output only, the maximum number of records that can be in the spooled output file. This parameter overrides the value specified in the device file.

**NOMAX:* There is no maximum on the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of records that can be in the spooled output file, and/or in other OVRCRDF commands executed in following invocations.

FILESEP Parameter: Specifies, for spooled output only, the number of separator cards to be placed at the beginning of each output card deck. Enter a value, 0 through 9, for the number of cards to be used. If 0 is specified, then at the end of each card output file, the spooling writer sends a message to the message queue specified in the Start Writer (STRWTR) command; the message indicates that the output is to be removed from the device. This parameter overrides the file separator value specified in the device file, and/or in other OVRCRDF commands executed in following invocations.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a spooling writer. This parameter overrides the scheduling value specified in the device file, and/or in other OVRCRDF commands executed in following invocations.

**JOBEND:* The spooled output file is to be made available to the spooling writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the spooling writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the spooling writer as soon as the file is opened in the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The file can be released by the RLSSPLF (Release Spooled File) command. This parameter overrides the hold value specified in the device file, and/or in other OVRCRDF commands executed in following invocations.

**NO:* The spooled output file is not to be held on the output queue. The spooled output is made available to a writer based on the SCHEDULE parameter value.

**YES:* The spooled output file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled output file is to be saved after it has been produced. This parameter overrides the save value specified in the device file, and/or in other OVRCRDF commands executed in following invocations.

**NO:* The spooled file data is not to be retained (saved) on the output queue after it has been produced.

**YES:* The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the device file, in the program, and/or in other OVRCRDF commands executed in following invocations. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is to be made.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previous invocations.

**YES:* This file is protected from the effects of any file override commands executed in previous invocations.

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other opens of the same file in the routing step. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record. This parameter overrides the value specified in the device file, in the program, and/or in other OVRCRDF commands executed in following invocations.

**NO:* An ODP created for this file open is not be be shared. Every time a program opens the file, a new ODP to the file is created and activated.

**YES:* If the file is opened more than once, the same ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file. This includes multiple opens in the same program.

Examples

```
OVRCRDF  FILE(CARD)  SPOOL(*NO)
```

This command overrides only the spooling specification for the card file named CARD. Because of this override, spooling will not be used for the file. If CARD is an input file, records will be read directly from the card reader. If it is an output file, output records will be sent directly to the device.

```
OVRCRDF  FILE(CARD)  TOFILE(CARD1)
```

This command causes the card file CARD1 to be used in place of the card file named CARD. No parameters of CARD1 are overridden; it is used as it is already described on the system.

```
OVRCRDF  FILE(DSKT1)  TOFILE(CARD1)  OUTQ(CARDQ3)
```

This command causes the file CARD1 to be used in place of the diskette file DSKT1. The output queue to be used for the card file is changed to CARDQ3.

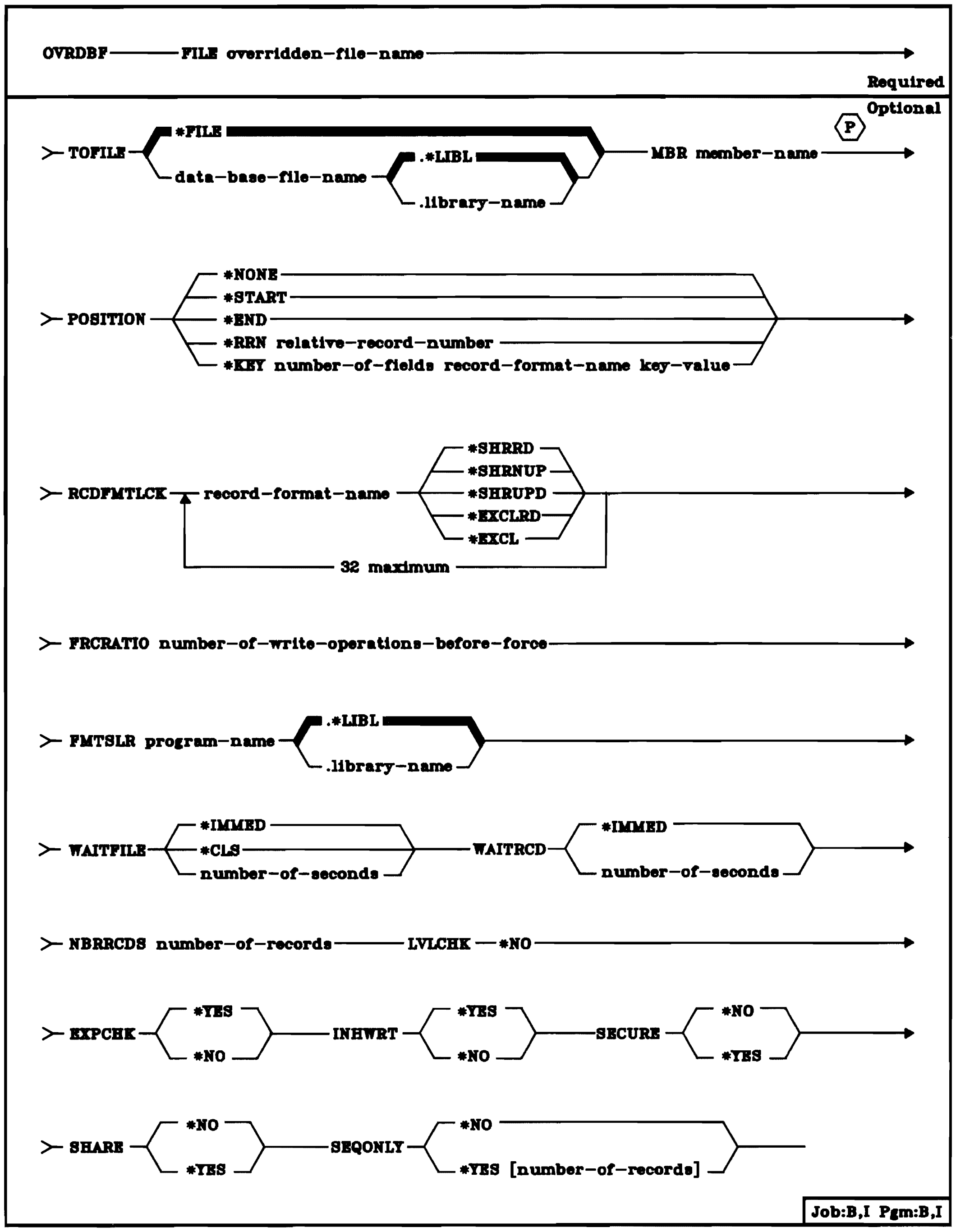
OVRDBF (Override with Data Base File) Command

The Override with Data Base File (OVRDBF) command can be used to (1) override (replace) the file named in the program, (2) override certain parameters of a file that is used by the program, or (3) override the file named in the program *and* override certain parameters of the file to be processed. Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executed in following invocations.

If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter and the name of the overriding file (the file to be processed) is specified in the TOFILE parameter. The OVRDBF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRDBF command. Any parameters that are not specified do not affect parameters specified in the file description, in the program, and/or in other file override commands executed in following invocations.

A file override command must be executed before the file to be overridden is used. The file is overridden if it is used in the program containing the override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in following invocations. A file override applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). An override can be explicitly deleted by a DLTOVR command, executed in the same invocation as the override, or it is implicitly deleted when the invocation that issues the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed before the file is opened in an invocation following the invocation for this override.


OVRDBF
FILE overridden-file-name
Required
Optional
P
TOFILE
*FILE
data-base-file-name
.*LIBL
.library-name
MBR member-name
POSITION
*NONE
*START
*END
*RRN relative-record-number
*KEY number-of-fields record-format-name key-value
RCDFMTLCK
record-format-name
*SHRRD
*SHRNUP
*SHRUPD
*EXCLRD
*EXCL
32 maximum
FRCRATIO number-of-write-operations-before-force
FMTSLR program-name
.*LIBL
.library-name
WAITFILE
*IMMED
*CLS
number-of-seconds
WAITRCD
*IMMED
number-of-seconds
NBRRCDS number-of-records
LVLCHK
*NO
EXPCHK
*YES
*NO
INHWRT
*YES
*NO
SECURE
*NO
*YES
SHARE
*NO
*YES
SEQONLY
*NO
*YES [number-of-records]
Job:B,I Pgm:B,I

FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a data base file when TOFILE specifies *FILE. Otherwise, any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the data base file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRDBF command will override the same parameters specified in the data base file, in the program, and/or in other OVRDBF commands executed in following invocations.

FILE: The data base file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-data-base-file-name: Enter the qualified name of the data base file that is to be used instead of the file specified in the FILE parameter. (If no library qualifier is given, *LIBL is used to find the data base file description.)

MBR Parameter: Specifies the name of the member to be used within the data base file. Enter the member name that overrides (at file open time) the member name specified in the using program, and/or in other OVRDBF commands executed in following invocations. If the member name is not specified here, the first member in the file is used.

POSITION Parameter: Specifies the starting position for retrieving records from the data base file. The first record retrieved can be: at the beginning (*START) or at the end (*END) of the file; the *nth* record in the file (*RRN); or the record indicated by a key field value (*KEY). This parameter overrides the value specified in the program, and/or in other OVRDBF commands executed in following invocations.

NONE: No special positioning is required. The first I/O operation indicates the record to be retrieved.

START: The first record in the file is to be retrieved first.

END: The starting position is the last record in the file. When the next record is read, an end-of-file condition is reached. If a read previous is requested, the last record of the file is read.

RRN relative-record-number: Enter the relative record number (that is, its position from the beginning of the file) of the record that is to be retrieved first. The value *RRN must precede the relative record number. For example, POSITION(*RRN 480) specifies that the record following the 480th record in the file is to be retrieved next.

**KEY number-of-fields record-format-name key-value:* The record identified by the specified key value and record format is the first record to be retrieved. Enter the value *KEY followed by three values that correspond to the following:

- The number of fields that make up the key or composite key. The leftmost key fields in the record format are used.
- The name of the record format in the data base file that contains the key value being specified.
- The key field value, which can be specified as a quoted character string if all are character or positive zoned decimal, or can be specified in hexadecimal form (X'value') that identifies the first record to be retrieved.

For example, POSITION(*KEY 1 FMT2 X'4A11') specifies that the record format FMT2 has a single key field. The record that contains the key with the hexadecimal value 4A11 is to be retrieved next.

RCDFMTLCK Parameter: The record format lock parameter specifies the lock state of the named record format while it is being used by the program. The lock state indicates how the data associated with each format is to be locked. The following chart shows the lock states that can be specified for each record format and the operations allowed to other programs when the lock is in effect:

Lock State	Value	Other Program Operations
*SHRRD	Shared read	Read and update allowed
*SHRNUP	Shared read, no update	Read allowed, update not allowed
*SHRUPD	Shared update	Read and update allowed
*EXCLRD	Exclusive allow read	Read allowed
*EXCL	Exclusive no read	Neither read nor update allowed

For an explanation of each lock state, refer to the *CPF Programmer's Guide.*

For each record format, enter the record format name followed by one lock state value. This parameter overrides the record format locks specified in the program in other OVRDBF commands executed in following invocations, and/or the default locks established when the member was created. If the lock state specified for the file in an ALCOBJ command is more restrictive than the lock state specified here, this parameter is ignored. Thus this parameter can only impose a more restrictive lock state on a record format than that specified for the file.

FRCRATIO Parameter: The force write ratio parameter specifies the number of inserted or updated records that occur before they are forced into auxiliary (permanent) storage. (For an expanded description of the FRCRATIO parameter, see Appendix A.)

If a physical file associated with this data base file is being journaled, a larger force write ratio may be specified. Refer to the *CPF Programmer's Guide* for more information on the Journal Management Facility.

Enter the number of records that are to be accumulated before they are written into permanent storage. This parameter overrides the force ratio specified in the data base member, in the program, and/or in other OVRDBF commands executed in following invocations.

FMTSLR Parameter: Specifies the name of a record format selector program that is to be called when a logical file member contains more than one logical record format. The user-written selector program is called when a record is to be inserted into the data base file and a record format name is not included in the HLL program. The selector program name can be optionally qualified by the name of the library in which the program is stored. If no library qualifier is given, *LIBL is used to find the program. More information about the use of format selector programs is contained in the *CPF Programmer's Guide*. This parameter overrides the value specified in other OVRDBF commands executed in following invocations.

A program specified as the format selector program cannot be created with USRPRF(*OWNER) specified in the CRTCLPGM command.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the data base member, in the program, and/or in other OVRDBF commands executed in following invocations. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is to be made.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

WAITRCD Parameter: Specifies the number of seconds that the program is to wait for a record lock on a specific record when a file is being updated. This delay is needed to prevent another program from updating the record at the same time. If the record cannot be locked in the specified wait time, an error message is sent to the program. This parameter overrides the record wait time specified in the program, and/or in other OVRDBF commands executed in following invocations. (If a value cannot be specified in the HLL program, such as in RPG, a system default value of 60 seconds is used.)

**IMMED:* The program is not to wait; an immediate lock of the record must be obtained when the record is to be retrieved.

number-of-seconds: Enter the number of seconds that the program is to wait for the record lock. Valid values are 1 through 32767 (32 767 seconds).

NBRRCDS Parameter: Specifies the number of records to be retrieved as a unit from auxiliary storage to main storage to be processed. (The amount of data actually retrieved is equal to the number of records times the physical record length, not the logical record length.) The NBRRCDS parameter is valid for sequential or random processing and should be specified only when the data records are known to be physically located in auxiliary storage in the sequence in which they are to be processed. Enter a value for the number of records to be retrieved together from auxiliary storage. This parameter overrides the number of records value specified in the program, and/or in other OVRDBF commands executed in following invocations.

LVLCHK Parameter: Specifies whether the level identifiers for the record formats of the data base file are to be checked when the file is opened by a program. For this check (done while the member is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the data base member.

This parameter overrides the value specified in the data base file member, in the program, and/or in other OVRDBF commands executed in following invocations. Level checking cannot be done unless the program contains the record format identifiers. This command cannot override level checking from *NO to *YES.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

EXPCHK Parameter: Specifies whether the expiration date of the named member is to be checked. This date check is valid only on a physical file member. This parameter overrides the value specified in the program, and/or in other OVRDBF commands executed in following invocations.

**YES:* The expiration date of the physical file member is to be checked. If the current date is greater than the expiration date, an error message is sent to the job, where it can be monitored for. If the program does not handle the message, the message is sent to the job log.

**NO:* The expiration date is not to be checked.

INHWRT Parameter: The inhibit write parameter specifies whether the processed records are to be written, deleted, or updated in the data base file. This parameter allows you to test a program without storing the processed records back in the data base. This parameter overrides the INHWRT parameter in other OVRDBF commands executed in following invocations.

**YES:* Processed records are inhibited from being written into the data base; they can be written only to an output device.

**NO:* All new and changed processed records can be written into the data base, unless the program is in debug mode with UPDPROD(*NO) specified, and the file is in a production library. In that case, an escape message is sent to the program.

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previous invocations.

**YES:* This file is protected from the effects of any file override commands executed in previous invocations.

SHARE Parameter: Specifies whether the ODP (open data path) for the data base file member can be shared with other opens of the same member in the routing step. When an ODP is shared, the programs accessing the file member share such things as the position being accessed, the file status, and the buffer. This parameter overrides the value specified in the data base member, in the program, and/or in other OVRDBF commands executed in following invocations.

**NO:* An ODP created for this member open is not to be shared. Every time a program opens the file member, a new ODP to the member is created and activated.

**YES:* If the member is opened more than once, the same ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the member. This includes multiple opens in the same program.

SEQONLY Parameter: Specifies, for data base files whose records are to be processed in sequential order only, whether sequential only processing is to be used on the file. This parameter also specifies the number of records to be transferred as a group to or from the data base if sequential only processing is to be used. If a number is not specified, a default number is determined by the system. This parameter can be used to improve the performance of programs that process data base files in a sequential manner. This parameter overrides the value specified in the program and/or in other OVRDBF commands executed in following invocations.

For files opened for *input* to a program, the specified number of records are transferred as a group from the data base to an internal data management buffer. The program can perform only sequential reading of the records.

For files opened for *output* from a program, a group of records is transferred to the data base whenever the internal data management buffer receives the specified number of processed records from the program. For output files, sequential only processing is valid for physical file members and for logical file members that are based on one physical file member only.

If SEQONLY(*YES) is specified and the file is opened for *updating*, or for both input and output, the SEQONLY parameter is ignored and a message is sent to the user.

**NO:* The data base file is not restricted to sequential only processing.

**YES:* The data base file is to use sequential only processing. A default value for the number of records to be transferred as a group is determined by the system, based on how the file is used, the type of access path involved, and the file's record length:

- The default is approximately the number of records that will fit in an internal buffer of 2 K-bytes for:
  - All data base files opened for input
  - Physical files opened for output that are processed in either arrival sequence or in nonunique keyed sequence

- The default is 1 record for:
  - All logical files opened for output
  - Physical files opened for output that either have *unique* keyed sequence access paths or have dependent logical files with keyed sequence access paths

**YES number-of-records:* The file is to use sequential only processing, and a value indicating the number of records to be in each group transferred between the data base and the internal buffer is to be specified by the user. Enter *YES followed by a value (1 through 32767 is valid) for the number of records to be transferred each time. The user must ensure that the buffer size specified is *always* available to the program in the storage pool in which the program is running.

While records are in the internal data management buffer, other jobs could make updates to the same records in the data base, and the program performing sequential only *input* processing would not see the updates. To ensure that no other updating is done to records while they are in the buffer, the ALCOBJ command can be used in the program to specify either an *EXCLRD or an *EXCL lock on the records. If a program performs sequential only *output* processing and does not handle output errors (such as duplicate keys and conversion mapping errors) that may occur when the records in the buffer are written back to the data base, records after the first record in error are not written.

If the file is opened for output and the value specified here is not the same as the force write ratio specified for the file, the value used by the system is the smaller of the two; a message stating which value was modified for the job is sent to the user.

When processing SEQONLY(*YES) for putting records into a data base file, feedback information for each record (such as relative record number) is not always updated. If such feedback information is important, specify SEQONLY(*NO) or SEQONLY(*YES 1).

Examples

```
OVRDBF  FILE(ORDERSIN)  MBR(MONDAY)
```

This command causes the member MONDAY to be processed when the file ORDERSIN is opened. No other parameters for the file are overridden.

```
OVRDBF  FILE(ORDERSIN)  SHARE(*YES)
```

This command overrides the share specification for the file ORDERSIN. Because of this override, any subsequent opens of this file within the routing step can share the ODP for the file.

```
OVRDBF  FILE(INPUT)  TOFILE(PAYROLL)  MBR(MBR1) +
      RCDFMTLCK((EMPDATA *EXCL))
```

This command causes the MBR1 member of the PAYROLL file to be processed when the file INPUT is opened. The data records that are in the format specified by EMPDATA have the lock state *EXCL, which prevents another program from reading or updating a record while it is being used by this program.

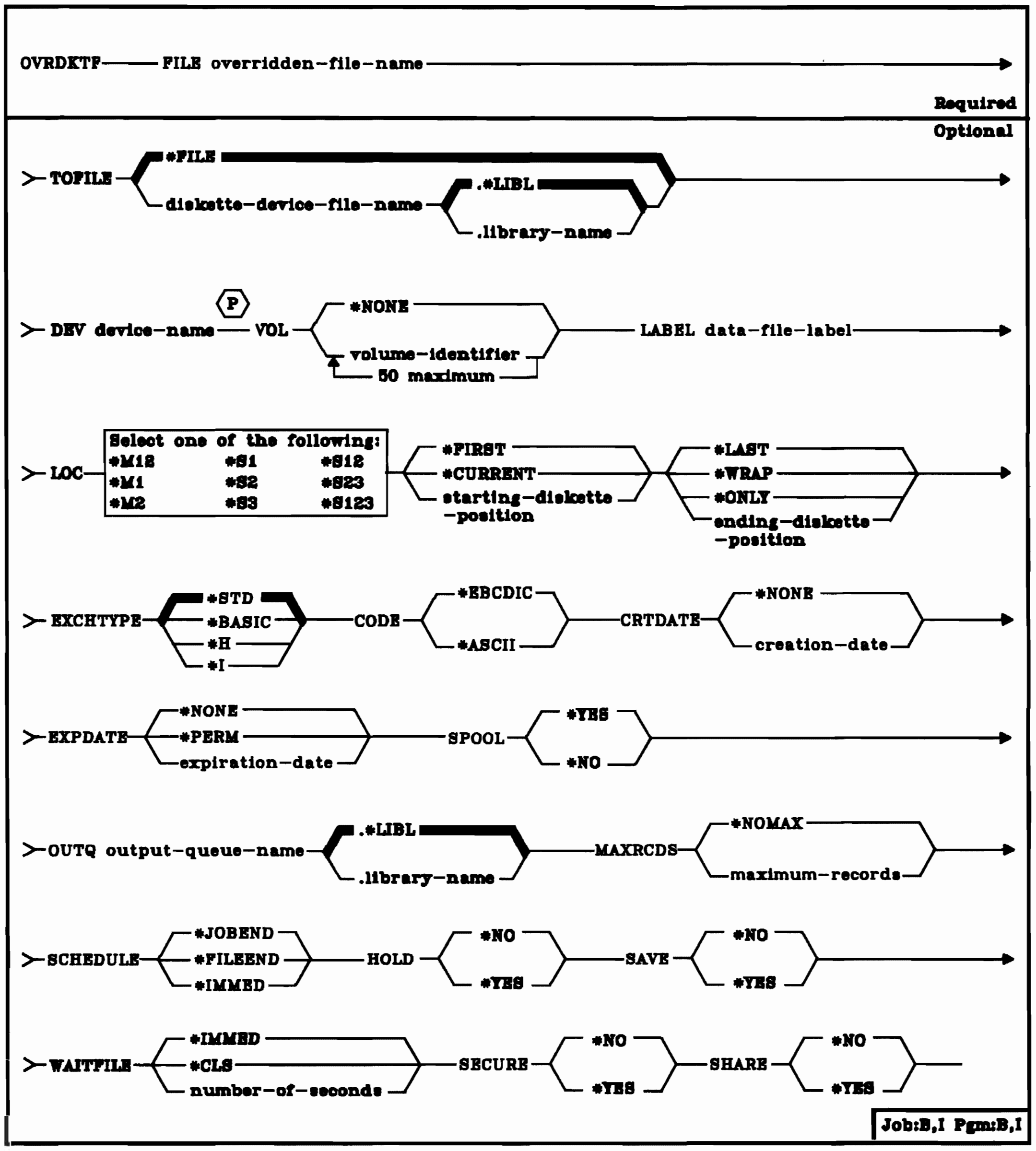
OVRDKTF (Override with Diskette File) Command

The Override with Diskette File (OVRDKTF) command can be used to (1) override (replace) the file named in the program, (2) override certain parameters of a file that is used by the program, or (3) override the file named in the program *and* override certain parameters of the file to be processed. Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executed in following invocations.

If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter and the name of the overriding file (the file to be processed) is specified in the TOFILE parameter. The OVRDKTF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRDKTF command. Any parameters that are not specified do not affect parameters specified in the file description, in the program, and/or in other file override commands executed in following invocations.

A file override command must be executed before the file to be overridden is used. The file is overridden if it is used in the program containing the override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in following invocations. A file override applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). An override can be explicitly deleted by a DLTOVR command, executed in the same invocation as the override, or it is implicitly deleted when the invocation that issues the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed before the file is opened in an invocation following the invocation for this override.



FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a diskette device file when TOFILE specifies *FILE. Otherwise, any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the diskette file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRDKTF command will override the same parameters specified in the diskette device file or in the program, and/or in other OVRDKTF commands executed in following invocations.

*FILE: The diskette device file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-diskette-device-file-name: Enter the qualified name of the diskette device file that is to be used instead of the overridden file. (If no library qualifier is given, *LIBL is used to find the diskette device file description.)

DEV Parameter: Specifies the name of the diskette device to be used with the diskette device file to perform I/O operations. Enter the device name that overrides the device name specified in the diskette device file, in the program, and/or in other OVRDKTF commands executed in following invocations. The device name of the IBM-supplied diskette device description is QDKT. This parameter is ignored if SPOOL(*YES) is in effect for the file when it is opened.

VOL Parameter: Specifies one or more volume identifiers of the diskettes (either in magazines or slots) to be used by the diskette device file. The volumes must be mounted on the device in the same order as their identifiers are specified here. This parameter overrides the volume identifiers specified in the diskette device file, in the program, and/or in other OVRDKTF commands executed in following invocations. (For an expanded description of the VOL parameter, see Appendix A.)

**NONE:* No volume identifiers are specified; no volume identifier checking is to be performed.

volume-identifier: Enter the identifiers (6 alphameric characters or less) in the order in which the volumes are to be mounted.

LABEL Parameter: Specifies the data file label of the data file on diskette that is to be processed or created. For input files (diskette input to the system), this label specifies the identifier of the file that exists on the diskette. For output files (system output to diskette), it specifies the identifier of the file that is to be created on the diskette. (For an expanded description of the LABEL parameter, see Appendix A.)

Enter the identifier (8 characters maximum) of the data file to be used with this diskette device file. This parameter overrides the label specified in the diskette device file, in the program, and/or in other OVRDKTF commands executed in following invocations.

LOC Parameter: Specifies the diskette locations in the magazines or slots that are to be used by the diskette device file. Three values are needed: (1) the unit type and location, (2) the starting diskette position, and (3) the ending diskette position in the unit. (For an expanded description of the LOC parameter, see Appendix A.) This parameter overrides the value specified in the device file, in the program, and/or in other OVRDKTF commands executing in following invocations.

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit and location on the diskette magazine drive are to be used by the device file for diskette input/output.

Enter one of the following values for the unit type and location (the valid starting and ending positions for each unit type are also listed):

Unit Type/Location	**Diskette Starting and Ending Position**
*M12	1 through 10
*M1	1 through 10
*M2	1 through 10
*S1	1
*S2	2
*S3	3
*S12	1 through 2
*S23	2 through 3
*S123	1 through 3

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette used first by the device file. Enter one of the following values as the override value for the starting diskette position:

**FIRST:* The first diskette position in the location contains the diskette to be used first in the read or write operation. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used.

starting-diskette-position: Enter the number of the diskette position in the magazine or the manual slot that contains the first diskette to be used.

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette used last by the device file. Enter one of the following values as the override value for the ending diskette position:

**LAST:* The last diskette position in the location contains the diskette to be used last in the read or write operation. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**WRAP:* If the end of the last diskette in the location is reached before the end of the data file is reached, a message is sent to the system operator to mount another magazine or diskette to continue. (See Appendix A for details.)

**ONLY:* Only the diskette position specified by the second value is to be used, and used only once.

ending-diskette-position: Enter the number of the diskette position in the magazine or the manual slot that contains the last diskette to be used.

EXCHTYPE Parameter: Specifies, for diskette output files only, the exchange type to be used by the device file when the system is writing diskette data. This parameter overrides the value specified in the device file, in the program, and/or in other OVRDKTF commands executed in following invocations. (For an expanded description of the EXCHTYPE parameter, refer to Appendix A).

**STD:* The basic exchange format will be used for a type 1 or a type 2 diskette. The H exchange type will be used for a type 2D diskette.

**BASIC:* The basic exchange type will be used.

**H:* The H exchange type will be used.

**I:* The I exchange type will be used.

CODE Parameter: Specifies the type of character code that is to be used by the device file when the system is reading or writing diskette data. This parameter overrides the value specified in the device file, in the program, and/or in other OVRDKTF commands executed in following invocations.

**EBCDIC:* The EBCDIC character code is to be used with this diskette device file.

**ASCII:* The ASCII character code is to be used with this diskette device file.

CRTDATE Parameter: Specifies, for diskette input data files only, the date when the data file was created on the diskette. If the creation date specified here (if any) does not match the date written on the diskette, an error message is sent to the program. This parameter overrides the value specified in the device file, in the program, and/or in other OVRDKTF commands executed in following invocations.

**NONE:* The creation date of the data file is not to be checked.

creation-date: Enter the creation date of the data file to be used by the device file. The date must be specified in the format defined by the system values QDATFMT and QDATSEP.

EXPDATE Parameter: Specifies, for diskette output files only, the expiration date of the data file used by this device file. The data file will be protected and cannot be written over until the day after the specified expiration date. This parameter overrides the value specified in the device file, in the program, and/or in other OVRDKTF commands executed in following invocations.

**NONE:* The data file is to have no expiration date; it is to be protected only on the day it is created.

**PERM:* The data file is to be permanently protected. An expiration date of 999999 is written on the diskette.

expiration-date: Enter the expiration date after which the data file expires. The date must be specified in the format defined by the system values QDATFMT and QDATSEP.

SPOOL Parameter: Specifies whether the input or output data for the diskette device file is to be spooled. This parameter overrides the spool value specified in the device file, and/or in other OVRDKTF commands executed in following invocations.

Note: If SPOOL(*NO) is the current value in the diskette device file, or if SPOOL(*NO) is specified in this or any other OVRDKTF command that is in effect when the file is opened, then the following parameters that apply to spooled output files are ignored: OUTQ, MAXRCDS, SCHEDULE, HOLD, and SAVE.

**YES:* The data is to be spooled. If this file is opened for input, the named inline data file or the next unnamed inline spooled file is processed. (For information on inline files, refer to the *CPF Programmer's Guide.*) If this is an output file, the data is spooled for processing by a spooling writer.

**NO:* The data is not to be spooled. Depending upon whether this is an input or an output data file, the data is read directly from the diskette, or it is written directly to the diskette as it is processed by the program.

OUTQ Parameter: Specifies, for spooled output only, the name of the output queue for the spooled output file. Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.) This parameter overrides the output queue name specified in the device file, and/or in other OVRDKTF commands executed in following invocations.

MAXRCDS Parameter: Specifies, for spooled output only, the maximum number of records that can be in the spooled output file for spooled jobs using the diskette device file. This parameter overrides the value specified in the diskette device file, and/or in other OVRDKTF commands executed in following invocations.

**NOMAX:* There is no maximum on the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of records that can be in the spooled output file.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a spooling writer. This parameter overrides the scheduling value specified in the device file, and/or in other OVRDKTF commands executed in following invocations.

**JOBEND:* The spooled output file is to be made available to the spooling writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the spooling writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the spooling writer as soon as the first output records are produced by the program.

HOLD Parameter: Specifies, for spooled output files only, whether the spooled file is to be held. The file can be released by the RLSSPLF (Release Spooled File) command. This parameter overrides the hold value specified in the diskette device file, and/or in other OVRDKTF commands executed in following invocations.

**NO:* The spooled output file is not to be held on the output queue. The spooled output is made available to a spooling writer based on the SCHEDULE parameter value.

**YES:* The spooled output file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, for spooled output files only, whether the spooled file is to be saved (left on the output queue) after the output has been produced. This parameter overrides the save value specified in the device file, and/or in other OVRDKTF commands executed in following invocations.

**NO:* The spooled file data is not to be retained (saved) on the output queue after it has been produced.

**YES:* The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the device file, in the program, and/or in other OVRDKTF commands executed in following invocations. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previous invocations.

**YES:* This file is protected from the effects of any file override commands executed in previous invocations.

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other opens of the same file in the routing step. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record. This parameter overrides the value specified in the device file, in the program, and/or in other OVRDKTF commands executed in following invocations.

**NO:* An ODP created for this file open is not to be shared. Every time a program opens the file, a new ODP to the file is created and activated.

**YES:* If the file is opened more than once, the same ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file. This includes multiple opens in the same program.

Examples

```
OVRDKTF  FILE(OUT) VOL(DPT706) LABEL(STATUSR) +
     SPOOL(*YES)
```

This command changes the spooling specification for the output file named OUT. When a program produces output data for the OUT file, the data is spooled for processing by a spooling writer. The writer processes the data by writing it in a data file called STATUSR that is on a diskette whose volume identifier is DPT706.

```
OVRDKTF  FILE(DISK) TOFILE(DISK2)
```

This command causes the diskette device file DISK2 to be used in place of the device file named DISK. No parameters of DISK2 are overridden; it is used as it is described in its device file description.

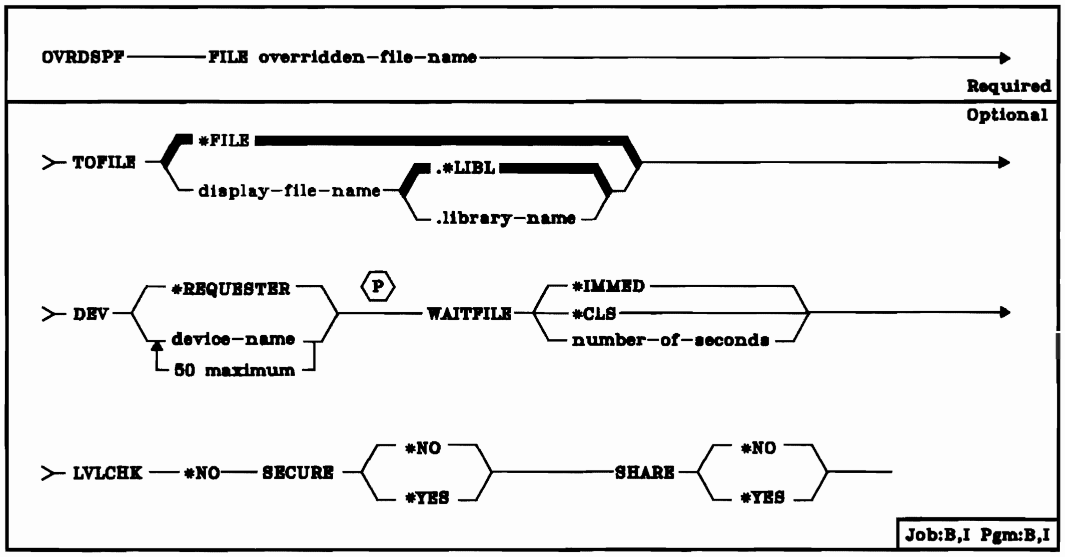
OVRDSPF (Override with Display File) Command

The Override with Display File (OVRDSPF) command can be used to (1) override (replace) the file named in the program, (2) override certain parameters of a file that is used by the program, or (3) override the file named in the program *and* override certain parameters of the file to be processed. Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executed in following invocations.

If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter and the name of the overriding file (the file to be processed) is specified in the TOFILE parameter. The OVRDSPF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRDSPF command. Any parameters that are not specified do not affect parameters specified in the file description, in the program, and/or in other file override commands executed in following invocations.

A file override command must be executed before the file to be overridden is used. The file is overridden if it is used in the program containing the override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in following invocations. A file override applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). An override can be explicitly deleted by a DLTOVR command, executed in the same invocation as the override, or it is implicitly deleted when the invocation that issues the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed before the file is opened in an invocation following the invocation for this override.



FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a display device file when TOFILE specifies *FILE. Otherwise, any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the display file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRDSPF command will override the same parameters specified in the display device file, in the program, and/or in other OVRDSPF commands executed in following invocations.

*FILE: The display device file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-display-device-file-name: Enter the qualified name of the display device file that is to be used instead of the overridden file. (If no library qualifier is given, *LIBL is used to find the display device file description.)

DEV Parameter: Specifies the names of one or more display devices that are to be used with the display device file. This parameter overrides the device name(s) specified in the device file, in the program, and/or in other OVRDSPF commands executed in following invocations. The device name specified in the IBM-supplied display device file is QCONSOLE.

The total number of device names (including *REQUESTER, if it is specified) cannot exceed the number specified in the MAXDEV parameter of this device file.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the device file, in the program, and/or in other OVRDSPF commands executed in following invocations. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is to be made.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

LVLCHK Parameter: Specifies whether the level identifiers of the record formats in this device file are to be checked when the file is opened by a program. For this check (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given a unique internal system identifier when the format is created.

This parameter overrides the value specified in the device file, in the program, and/or in other OVRDSPF commands executed in following invocations. Level checking cannot be done unless the program contains the record format identifiers. This command cannot override level checking from *NO to *YES.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previous invocations.

**YES:* This file is protected from the effects of any file override commands executed in previous invocations.

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other opens of the same file in the routing step. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record to the display. This parameter overrides the value specified in the device file, in the program, and/or in other OVRDSPF commands executed in following invocations.

**NO:* An ODP created for this file open is not to be shared. Every time a program opens the file, a new ODP to the file is created and activated.

**YES:* If the file is opened more than once, the same ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file. This includes multiple opens in the same program.

Example

```
OVRDSPF  FILE(DISPLAY75) WAITFILE(30)
```

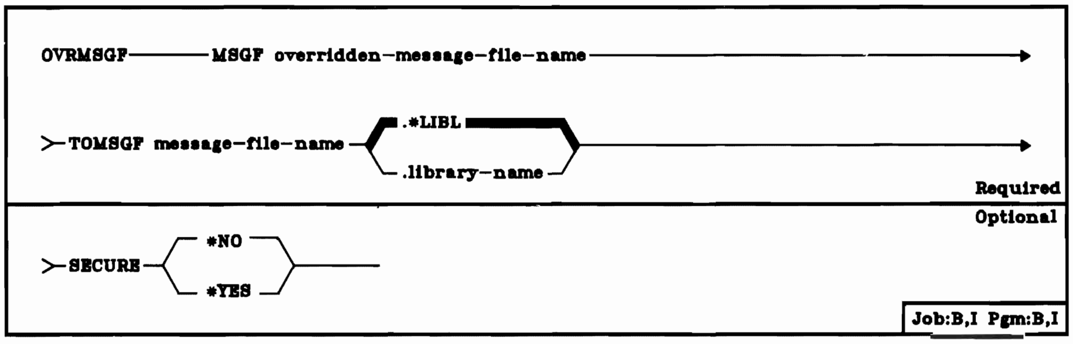
This command overrides the file wait time value specified in the DISPLAY75 device file description, in the program, and/or in other OVRDSPF commands executed in following invocations. The program in which this command occurs is to wait up to 30 seconds (if necessary) for the required file resources to be allocated to the file named DISPLAY75.

OVRMSGF (Override with Message File) Command

The Override with Message File (OVRMSGF) command can be used to override a message file used in a program. The overriding message file (specified in the TOMSGF parameter) is used whenever a message is sent or retrieved and the overridden message file is specified.

The overriding message file need not contain all the messages that the overridden file contains. When a message is received or retrieved and the message identifier cannot be found in the overriding message file, the overridden message file is searched for the identifier. Overriding message files can themselves be overridden, resulting in a chain of overrides. This chain of overrides provides a list of message files to be searched when a message is received or retrieved. A maximum of 30 message file overrides can be specified in a program.

Message file overrides can be deleted by the Delete File Override (DLTOVR) command or displayed by the Display File Override (DSPOVR) command. A message file in a program can be protected from being overridden if SECURE(*YES) is specified.



MSGF Parameter: Specifies the name of the message file in the using program to which this override command is to be applied.

TOMSGF Parameter: Specifies the name of the message file that is to be used instead of the message file specified in the MSGF parameter or, if the names are the same, specifies that the SECURE parameter specified in the command is to be used for the message file. Enter the qualified name of the message file that is to be used in place of the overridden message file. (If no library qualifier is specified, *LIBL is used to find the message file.)

SECURE Parameter: Specifies whether this file is to be secured from the effects of message file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This message file is not protected from other file overrides; its values can be overridden by the effects of any message file overrides executed in previous invocations.

**YES:* This message file is protected from the effects of any message file overrides executed in previous invocations.

Examples

```
OVRMSGF  MSGF(QCPFMSG) TOMSGF(CPFMSGOVR)
```

This override command causes the default replies for messages stored in CPFMSGOVR to be used instead of those stored in QCPFMSG. Only the changed messages need to be stored in CPFMSGOVR. If a message is not found in CPFMSGOVR, the overridden message file (QCPFMSG) is searched.

```
OVRMSGF  MSGF(WSUSRMSG) TOMSGF(ORDENTMSGD)
```

This override command causes the defaults for messages stored in ORDENTMSGD to be used instead of defaults stored in WSUSRMSG (which contains messages designed for work station users). As a result of this command, the messages received by the order entry users are tailored to their own environment.

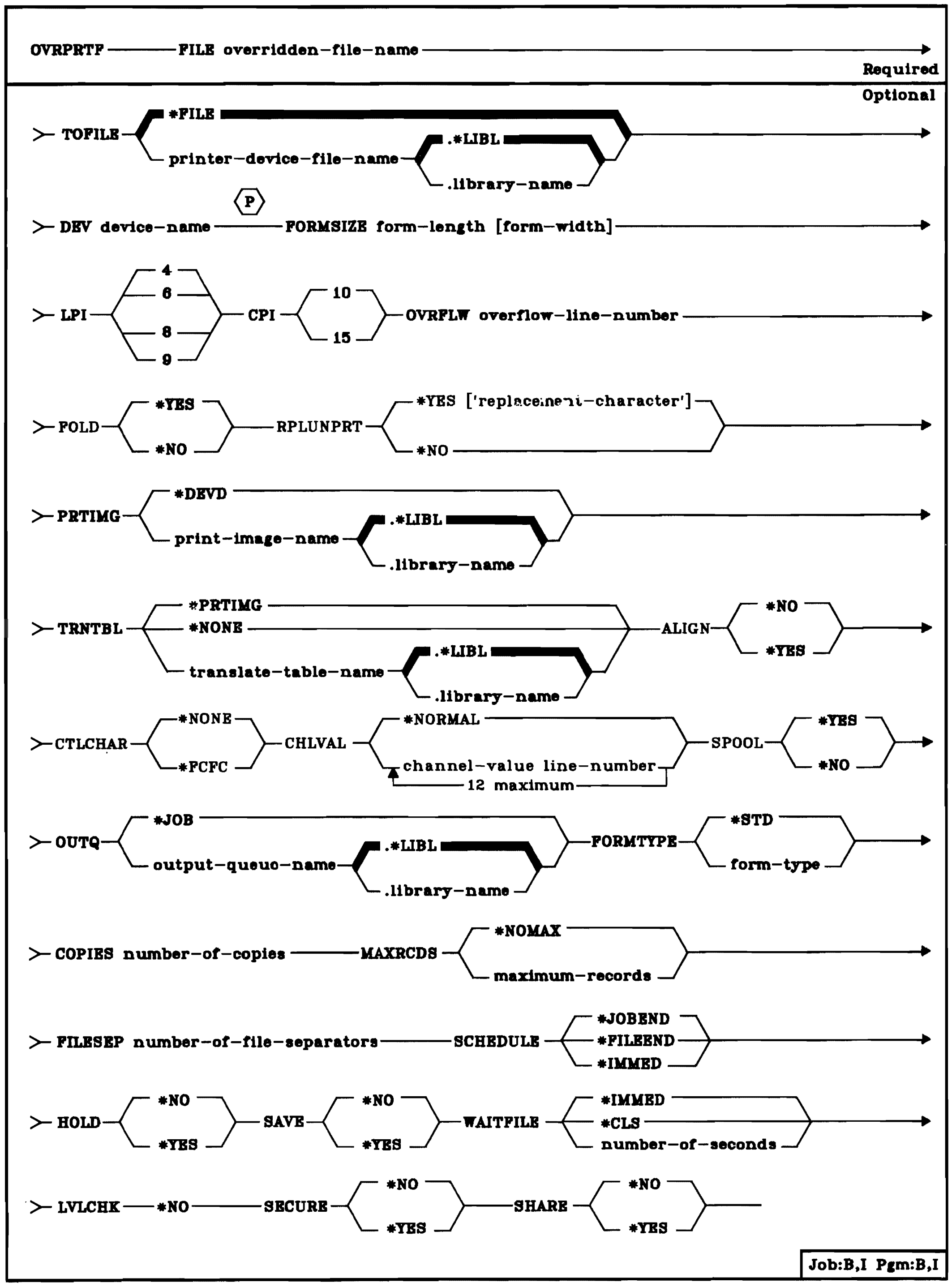
OVRPRTF (Override with Printer File) Command

The Override with Printer File (OVRPRTF) command can be used to (1) override (replace) the file named in the program, (2) override certain parameters of a file that is used by the program, or (3) override the file named in the program *and* override certain parameters of the file to be processed. Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executed in following invocations.

If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter and the name of the overriding file (the file to be processed) is specified in the TOFILE parameter. The OVRPRTF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRPRTF command. Any parameters that are not specified do not affect parameters specified in the file description, in the program, and/or in other file override commands executed in following invocations.

A file override command must be executed before the file to be overridden is used. The file is overridden if it is used in the program containing the override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in following invocations. A file override applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). An override can be explicitly deleted by a DLTOVR command, executed in the same invocation as the override, or it is implicitly deleted when the invocation that issues the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed before the file is opened in an invocation following the invocation for this override.



FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a printer device file when TOFILE specifies *FILE. Otherwise, any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the printer file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRPRTF command will override the same parameters specified in the printer device file, in the program, and/or in other OVRPRTF commands executed in following invocations.

*FILE: The printer device file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-printer-device-file-name: Enter the qualified name of the printer device file that is to be used instead of the overridden file. (If no library qualifier is given, *LIBL is used to find the printer device file description.)

DEV Parameter: Specifies, for *nonspooled* output only, the name of the printer to be used with the printer device file to produce the output. Enter the device name that overrides the device name specified in the printer device file, in the program, and/or in other OVRPRTF commands executed in following invocations. The device names of the IBM-supplied printer device descriptions are QSYSPRT and QSYSPRT2 (provided when two system printers are attached to the System/38). This parameter is ignored if SPOOL(*YES) is in effect for the file when it is opened.

FORMSIZE Parameter: Specifies the length and width of the printer forms to be used by this device file. The length is in lines per page, and the width is in print positions (characters) per line. This parameter overrides the form size values specified in the printer device file, in the program, and/or in other OVRPRTF commands executed in following invocations.

form-length: Enter the form length (in print lines per page) that is to be used by this device file. Although a value of 1 through 255 can be specified as the form length, the value specified should not exceed the actual length of the forms used. The following chart shows the number of lines per page that are valid for each printer type, depending on whether 6 or 8 lines per inch is specified in the LPI parameter for the 3203, 3262, and 5211 Printers, or is manually set on the 5256 Printer. For 5224 and 5225 Printers, 4, 6, 8, or 9 lines per inch can be specified.

	Lines per Page			
Printer	**4 lines/inch**	**6 lines/inch**	**8 lines/inch**	**9 lines/inch**
3203	–	2-144	2-192	–
3262 5211	–	2-84	2-112	–
5224 5225	1-255	1-255	1-255	1-255
5256	–	1-255	1-255	–

form-width: Enter the form width (in characters per printed line) that is to be used by this device file. Valid values for the 3203, 3262, 5211, and 5256 Printers are 1 through 132. Valid values for the 5224 and 5225 Printers are 1 through 198. The value specified should not exceed the actual width of the forms used.

LPI Parameter: Specifies the line spacing setting on the printer, in lines per inch, to be used by this device file. The line spacing on the 5256 work station printer must be set manually. This parameter overrides the overflow value specified in the printer device file, in the program, and/or in other OVRPRTF commands executed in following invocations.

4: The line spacing on the printer is to be 4 lines per inch.

6: The line spacing on the printer is to be 6 lines per inch.

8: The line spacing on the printer is to be 8 lines per inch.

9: The line spacing on the printer is to be 9 lines per inch.

Line spacings of 4 and 9 lines per inch are valid only for 5224 and 5225 Printers.

CPI Parameter: Specifies the printer character density, in characters per inch, to be used by this device file. 15 characters per inch are valid only for the 5224 and 5225 Printers. This parameter overrides the value specified in the device file, in the program, and/or in other OVRPRTF commands executing in following invocations.

<u>10:</u> Character density is to be 10 characters per inch.

15: Character density is to be 15 characters per inch.

OVRFLW Parameter: Specifies the line number on the page when overflow to a new page is to occur. Generally, after the specified line is printed, the printer overflows to the next page before printing continues. (Refer to the *CPF Programmer's Guide* for details about controlling page overflow.) This parameter overrides the overflow value specified in the printer device file, in the program, and/or in other OVRPRTF commands executed in following invocations.

Enter the line number of the line that causes page overflow after the line is printed. The value specified must not exceed the forms length specified in the FORMSIZE parameter for the file.

FOLD Parameter: Specifies whether all positions in a record are to be printed when the record length exceeds the form width (specified by the FORMSIZE parameter). If so, any portion of the record that cannot be printed on the first line is continued (folded) on the next line or lines until the entire record has been printed. This parameter overrides the value specified in the printer device file, in the program, and/or in other OVRPRTF commands executed in following invocations.

**YES:* Records whose length exceeds the form width are to be folded on the following line(s).

**NO:* Records are not folded; if a record is longer than the form width, only the first part of the record that fits on one line is printed.

RPLUNPRT Parameter: The replace unprintable character parameter specifies (1) whether unprintable characters are to be replaced and (2) which substitution character (if any) is to be used. An *unprintable* character is a character that is not on the print belt or train, or in the print image used by the printer.

For 5224, 5225, and 5256 Printers, one of the following occurs when an unprintable character is encountered:

- If you specify RPLUNPRT(*YES), the specified substitution character is printed in place of each unprintable character.
- If you specify RPLUNPRT(*NO) and the value of the unprintable character is hex 00 through hex 3F, or is hex FF, undesirable results may occur. Most characters in this range cause an unrecoverable error to be signaled by the printer, and either the file is held for spooling or it is not processed. Some characters in this range, however, control forms movement and character representation on the printer. If the unprintable character is one of these control characters, additional spacing or skipping may occur. If control characters are specifically placed in the data, other system functions (such as the displaying or copying of a spooled file, or restarting or backing up of a print writer) may cause unpredictable results.
- If you specify RPLUNPRT(*NO) and the value of the unprintable character is in the range of hex 40 through hex FE, a recoverable error is signaled by the device and an inquiry message is sent to the operator, informing him of the error and giving him the chance to cancel the file or to continue processing. If the continue option is selected, subsequent unprintable characters will appear as blanks in the output, and no further inquiry messages will be sent to the operator.

For 3203, 3262, and 5211 Printers, one of the following occurs when an unprintable character is encountered:

- If you specify RPLUNPRT(*YES) and the value of the unprintable character is in the range of hex 00 through hex 3F, or is hex FF, the specified substitution character is printed instead. If no substitution character was specified, the blank is used. if no characters in this range are expected to be in the data to be printed, *NO can be specified for this parameter to gain some performance improvement. However, if *NO *is* specified and an unprintable character in this range does occur, the only recovery is to rerun the job.

- If you specify RPLUNPRT(*YES) and the value of the unprintable character is in the range of hex 40 through hex FE, a translate table should be used to translate unprintable characters to different printable characters; each unprintable hex value can be translated to its own printable character. The translate table, which is specified by the TRNTBL parameter, should also match the print image used by the printer.

- If you specify RPLUNPRT(*NO) and the value of the unprintable character is hex 00 through hex 3F, undesirable results may occur. Most characters in this range cause an unrecoverable error to be signaled by the printer, and either the file is held for spooling or it is not processed. Some characters in this range, however, control forms movement and character representation on the printer. If the unprintable character is one of these control characters, additional spacing or skipping may occur. If control characters are specifically placed in the data, other system functions (such as the displaying or copying of a spooled file, or restarting or backing up of a print writer) may cause unpredictable results.

- If you specify RPLUNPRT(*NO) and the value of the unprintable character is in the range of hex 40 through hex FE, a recoverable error is signaled by the device and a notify message is sent to the program. If you choose to continue processing or if the message is unmonitored, the error will be ignored and processing will continue. Subsequent unprintable characters will appear as blanks in the output, and no further inquiry messages will be sent to the program.

This parameter overrides the value in the printer device file, in the program, and/or in the other OVRPRTF commands executed in following invocations.

**YES:* Unprintable characters are to be replaced. The program is not notified when unprintable characters are detected.

**NO:* Unprintable characters are not to be replaced. When an unprintable character is detected, a message is sent to the program.

'replacement-character': If *YES is also specified in this parameter, enter the substitution character that is to be used each time an unprintable character is detected. Any printable EBCDIC character can be specified.

PRTIMG Parameter: Specifies, for 3203, 3262, and 5211 printers only, the name of the print image to be used by this printer device file. This parameter overrides the print image name specified in the printer device file, in the program, and/or in other OVRPRTF commands executed in following invocations.

**DEVD:* The standard print image for the printer (specified in the device description) is to be used.

qualified-print-image-name: Enter the qualified name of the print image to be used. (If no library qualifier is given, *LIBL is used to find the print image.)

TRNTBL Parameter: Specifies, for 3203, 3262, and 5211 printers only, the name of the translate table (if any) to be used by this device file when the output data is to be translated before it is printed. The translate table is used to convert each unprintable character having a hexadecimal code of hex 40 through FE to the printable character specified in the table that is also on the print belt or train. Each hexadecimal code can specify a different character. This parameter overrides the value specified in the printer device file, in the program, and/or in other OVRPRTF commands executing in following invocations.

For each IBM-supplied print image shipped with the system, a matching translate table is also supplied; the name of the table is the same as the name of the image.

**PRTIMG:* The translate table with the same qualified name as the print image is to be used.

**NONE:* No translation is needed when this device file is used.

qualified-translate-table-name: Enter the qualified name of the translate table to be used by this device file and the 3203, 3262, or 5211 Printer. (If no library qualifier is given, *LIBL is used to find the translate table.)

ALIGN Parameter: Specifies, for *nonspooled* output only, whether the forms must be aligned in the printer before printing is started. If ALIGN(*YES) and SPOOL(*NO) are specified, and forms alignment is required, the system sends a message to the QSYSOPR message queue (or any message queue specified for 5224, 5225, or 5256 Printers), and waits for a reply to the message. This parameter overrides the alignment value specified in the printer device file, in the program, and/or in other OVRPRTF commands executing in following invocations. The parameter is ignored if SPOOL(*YES) is specified. (If the file *is* spooled, the message is sent to the message queue specified on the STRPRTWTR command whenever the writer is started and whenever the forms are to be changed.)

**NO:* No forms alignment is required.

**YES:* The forms are to be aligned before the output is printed.

CTLCHAR Parameter: Specifies whether the printer device file will support input with print control characters. Any invalid control characters that are encountered will be ignored, and single spacing is assumed.

**NONE:* No print control characters will be passed in the data to be printed.

**FCFC:* Specifies that the first character of every record will contain an ANSI forms-control character. If *FCFC is specified, the record length must include one position for the first-character forms-control code. This value is not valid for externally described printer files; that is, SRCFILE(*NONE) was specified on the Create Printer File (CRTPRTF) command.

CHLVAL Parameter: Specifies a list of channel numbers with their assigned line numbers. Use this parameter only if CTLCHAR(*FCFC) has been specified.

**NORMAL:* The default values for skipping to channel identifiers will be used. The following are the default values:

ANSI First-Character Forms-Control Codes

Code	Action Before Printing a Line
' '	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to line 1
2-11	Space one line
12	Skip to overflow line (OVRFLW parameter)

channel-number: Specifies a channel number to be associated with corresponding 'skip to' line number. The only valid values for this parameter are 1 through 12, corresponding to channels 1 through 12. The CHLVAL parameter associates the channel number with a page line number.

If no line number is specified for a channel identifier, and that channel identifier is encountered in the data, a default of 'space one line' before printing is taken. Each channel number may be specified only once per CHGPRTF command invocation.

Note: If one or more channel-number/line-number combinations are overridden, all other combinations must be re-entered.

line-number: The line number assigned for the channel number in the same list. The range of valid line numbers is 1 through 255. If no line number is assigned to a channel number, and that channel number is encountered in the data, a default of 'space one line' before printing is taken. Each line number may be specified only once per CHGPRTF command invocation.

SPOOL Parameter: Specifies whether the output data for the printer device file is to be spooled. This parameter overrides the spool value specified in the device file, and/or in other OVRPRTF commands executed in following invocations.

Note: If SPOOL(*NO) is the current value in the printer device file, or if SPOOL(*NO) is specified in this or any other OVRPRTF command that is in effect when the file is opened, then the following parameters that apply to spooled output files are ignored: OUTQ, FORMTYPE, COPIES, MAXRCDS, FILESEP, SCHEDULE, HOLD, and SAVE.

*YES: The data is to be spooled for processing by a card, diskette, or print writer.

_*NO:_ The data is not to be spooled; it is sent directly to the device to be printed as the output becomes available.

OUTQ Parameter: Specifies, for spooled output only, the name of the output queue for the spooled output file. This parameter overrides the output queue name specified in the printer device file and/or in other OVRPRTF commands executing in following invocations.

*JOB: The output queue specified in the job description associated with this job is to be used for the spooled output.

qualified-output-queue-name: Enter the qualified name of the output queue to which the output data is to be spooled. (If no library qualifier is given, *LIBL is used to find the queue.)

FORMTYPE Parameter: Specifies, for spooled output only, the type of forms to be used in the printer for printed output produced using this device file. Enter either *STD or the user-defined form type identifier, having 10 characters or less, for the printer forms to be used. If a form type other than *STD is specified, the system (when the output is to be produced) sends a message that identifies the form type to the system operator, and requests that the specified type of forms be mounted in the printer. This parameter overrides the form type value specified in the printer device file and/or in other OVRPRTF commands executing in following invocations.

COPIES Parameter: Specifies, for spooled output only, the number of copies of the output to be printed when this printer device file is used. Enter a value, 1 through 99, that indicates the number of identical print runs to be made. This parameter overrides the copy value specified in the device file.

MAXRCDS Parameter: Specifies, for spooled output only, the maximum number of records that can be on the output queue for spooled jobs using the printer device file. This parameter overrides the value specified in the printer device file and/or in other OVRPRTF commands executing in following invocations.

**NOMAX:* There is no maximum on the number of records that can be in the spooled output file.

maximum-records: Enter a value, 1 through 500000 (500 000), that specifies the maximum number of records that can be in the spooled output file.

FILESEP Parameter: Specifies, only if the output is spooled, the number of separator pages to be placed at the beginning of each printed file, including between multiple copies of the same output. Enter a value, 0 through 9, for the number of separator pages between files of the same job. This parameter overrides the separator value specified in the printer device file and/or in other OVRPRTF commands executing in following invocations.

SCHEDULE Parameter: Specifies, for spooled output files only, when the spooled output file is to be made available to a spooling writer. This parameter overrides the scheduling value specified in the printer device file and/or in other OVRPRTF commands executing in following invocations.

**JOBEND:* The spooled output file is to be made available to the spooling writer only after the entire job is completed.

**FILEEND:* The spooled output file is to be made available to the spooling writer as soon as the file is closed in the program.

**IMMED:* The spooled output file is to be made available to the spooling writer as soon as the file is opened in the program.

HOLD Parameter: Specifies, only if the output is spooled, whether the spooled printer file is to be held. The file can be released by the RLSSPLF (Release Spooled File) command. This parameter overrides the hold value specified in the printer device file and/or in other OVRPRTF commands executing in following invocations.

**NO:* The spooled output file is not to be held on the output queue. The spooled output is made available to a spooling writer based on the SCHEDULE parameter value.

**YES:* The spooled output file is to be held until it is released by the RLSSPLF command.

SAVE Parameter: Specifies, only if the output is spooled, whether the spooled file is to be saved after the output has been produced. This parameter overrides the save value specified in the printer device file and/or in other OVRPRTF commands executing in following invocations.

**NO:* The spooled file data is not to be retained (saved) on the output queue after it has been produced.

**YES:* The spooled file data is to be retained on the output queue until the file is deleted.

WAITFILE Parameter: Specifies the number of seconds that the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the printer device file, in the program, and/or in other OVRPRTF commands executing in following invocations. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources is to be made.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

LVLCHK Parameter: Specifies whether the level of the device file is to be checked when the file is opened by a program. For this check, (done while the file is being opened), the system compares the record format identifiers of each record format to be used by the program with the corresponding identifiers in the device file. Because the same record format name can exist in more than one file, each record format is given a unique internal system identifier when the format is created.

This parameter overrides the value specified in the device file, in the program, and/or in other OVRPRTF commands executing in following invocations. Level checking cannot be done unless the program contains the record format identifiers. This command cannot override level checking from *NO to *YES.

**NO:* The level identifiers of the record formats are not to be checked when the file is opened.

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previous invocations. If SECURE is not specified, processing occurs as if SECURE(*NO) had been specified.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previous invocations.

**YES:* This file is protected from the effects of any file override commands executed in previous invocations.

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other opens of the same file in the routing step. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a write operation in that program produces the next output record for the file. This parameter overrides the value specified in the printer device file, in the program, and/or in other OVRPRTF commands executing in following invocations.

**NO:* An ODP created for this file open is not be be shared. Every time a program opens the file, a new ODP to the file is created and activated.

**YES:* If the file is opened more than once the same ODP is to be shared with each program in the routing step that also specifies SHARE(*YES) when it opens the file. This includes multiple opens in the same program.

Examples

```
OVRPRTF  FILE(PRINTOUT) TOFILE(PRINT3) +
     SPOOL(*YES) COPIES(5) OUTQ(CUTPUT1)
```

This command overrides the file named PRINTOUT and uses the printer device file named PRINT3 to produce the spooled output on the printer. The output from the program is sent to the OUTPUT1 output queue. Five copies of the spooled file are to be printed on the printer specified on the Start Printer Writer (STRPRTWTR) command.

```
OVRPRTF  FILE(QPRINT) OUTQ(PRINTP)
```

This command overrides the output queue to be used for the printer file QPRINT. All other parameters for the file are unaffected.

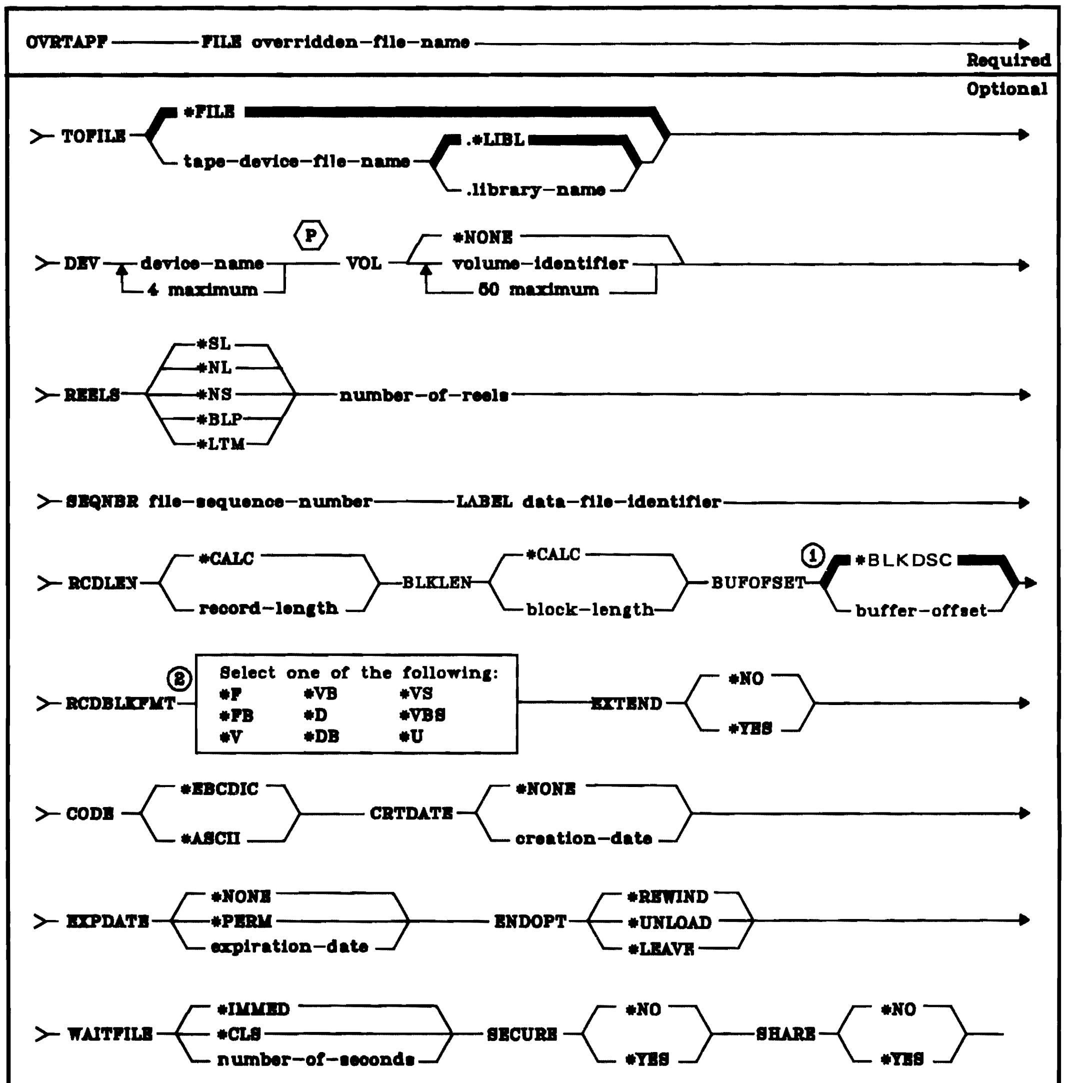
OVRTAPF (Override with Tape File) Command

The Override with Tape File (OVRTAPF) command can be used to (1) override/replace a file named in a program, (2) override certain attributes of a file that is used by a program, or (3) override the file named in a program *and* override certain attributes of the file to be processed.

Parameters overridden by this command can be specified in the file description, in the program, and/or in other file override commands executed in following invocations. If a file named in the program is to be overridden, the name of that file is specified in the FILE parameter; the name of the overriding file is specified in the TOFILE parameter. The OVRTAPF command can also specify parameters to override values contained in the file description of the overriding file. If the file named in the program is not to be replaced, but certain parameters of the file are to be overridden, the name of the file is specified in the FILE parameter and *FILE is specified in the TOFILE parameter. The parameters to be overridden are then specified by the other parameters of the OVRTAPF command. Any parameters that are not specified do not affect the parameters specified in the file description, in the program, and/or in other file override commands executed in following invocations.

A file override command must be executed before the file to be overridden is opened for use. The file is overridden if it is used in the program containing the file override command, if it is used in another program to which control is transferred (by the TFRCTL command), and/or if it is used in a program in a following invocation. A file override applies to all uses of the specified overridden file until the file override is deleted (unless the file is secured). An override can be explicitly deleted by a DLTOVR command, executed in the same invocation as the override, or be implicitly deleted when the invocation that issues the file override command ends. All file overrides are deleted when the routing step ends.

A file in a program is protected (secured) from having file overrides (from this and/or previous invocations) applied to it if SECURE(*YES) is specified. This can be specified either in the program in which the file is used or in a file override command that is executed before the file is opened in an invocation following the invocation for this override.



(1) The value *BLKDSC is valid only for a file with record block format *D or *DB.
(2) The values *F, *FB, *VS, *VBS, and *U are valid for both EBCDIC and ASCII codes, *V and *VB are valid only for EBCDIC, and *D and *DB are valid only for ASCII.

Job:B,I Pgm:B,I

FILE Parameter: Specifies the name of the file in the using program to which this override command is to be applied. The specified file must be a tape device file when TOFILE specifies *FILE. Otherwise, any device file or data base file name can be specified.

TOFILE Parameter: Specifies the name of the tape file that is to be used instead of the file specified in the FILE parameter or, if *FILE is specified, specifies that certain attributes are to be overridden by parameters specified in this command. The parameters specified on this OVRTAPF command will override the other values specified in the tape device file or in the program.

*FILE: The tape device file named in the FILE parameter is to have some of its parameters overridden by values specified in this command.

qualified-tape-device-file-name: Enter the qualified name of the tape device file that is to be used instead of the overridden file. (If no library qualifier is given, *LIBL is used to find the tape device file description.)

DEV Parameter: Specifies the names of one or more tape devices to be used with the tape device file to perform I/O operations. The order in which the device names are specified here is the order in which tapes mounted on the devices are processed. Enter the device names (no more than four) that override the device names specified in the program or in the tape device file. When more volumes are to be processed than the number of devices in the DEV list, the devices are used the same order as specified, wrapping around to the first device as needed.

VOL Parameter: Specifies one or more volume identifiers of the tapes to be used by the tape device file. The tapes (volumes) must be mounted on the devices in the same order as their identifiers are specified here and as the device names are specified in the DEV parameter. If the tape file is opened for read backward, the volume identifiers in the list are processed from last to first (while the devices in the device list are used in first to last order). An inquiry message is sent to the system operator if either *SL or *BLP processing is specified in the REELS parameter and an incorrect volume is mounted, or if no volume is mounted (for any type of label processing). When a list of volume identifiers is provided for the file, operator mount messages indicate the name of the volume which is required. This parameter overrides the volume identifiers specified in the tape device file. (For an expanded description of the VOL parameter, see Appendix A.)

**NONE:* No tape volume identifiers are specified for this file. They can be supplied before the device file is opened in another OVRTAPF command. If no volume identifiers are specified before the device file is opened, no volume checking is performed beyond verifying that the correct label type volume is mounted, and no volume names are provided in operator mount messages.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used by this device file. Each identifier can have 6 alphameric characters or less.

Note: If the VOL parameter value used for the file specifies a list of identifiers rather than VOL(*NONE), the number-of-reels part of the REELS parameter is ignored regardless of where it is specified. (See the *CPF Programmer's Guide* for a description of how the parameter values for the file are determined from the overrides, HLL interface, and the device file when the file is opened.) To ensure that the number-of-reels part of the REELS parameter is used (rather than a VOL identifier list) to control the volumes processed by the tape file, specify VOL(*NONE) in the same OVRTAPF command where the REELS parameter is specified.

REELS Parameter: Specifies the type of labeling used on the tape reels and the maximum number of reels to be processed if there is no list of volume identifiers specified (VOL parameter) and this device file is used with either *NL, *LTM, *NS, or *BLP input files. When the number of reels are specified, the volume identifiers (if processing labeled tapes) on the mounted volumes are ignored; instead; the order in which the reels are mounted must be checked by the operator. This parameter overrides the values specified in the device file, in the program, and/or in other OVRTAPF commands executed in following invocations.

The maximum number of reels specification (the second part of the REELS parameter) is not a limiting value for standard-label or output files. For a standard-label input file, the data file labels limit the number of volumes processed by indicating end-of-file. For an output file, the maximum number of reels value is ignored; the system requests that additional volumes be mounted until the file is closed.

For any type of label processing, the system checks the first record following the load point on the tape to see (1) if it has exactly 80 bytes for EBCDIC or at least 80 bytes for ASCII and (2) if the first 4 bytes contain the values VOL and 1. If so, the reel contains a standard-label tape. *SL and *BLP files require standard-label tape volumes. *NL, *LTM, and *NS tape files cannot process standard-label volumes.

Note: The values *SL, *NL, and *LTM can be specified if the device file is to be used for either reading or writing on tapes. The values *NS and *BLP are valid only if the device file is used to read tapes.

**SL:* The volumes have standard labels. If a list of volume identifiers is specified (with the VOL parameter), the system checks that the correct tape volumes are mounted in the specified sequence. If no volume identifier list is given and the file is opened for output, any standard-label volumes may be mounted. If no volume identifier list is given and the file is opened for input, the first volume may have any volume identifier, but if the file is continued, the system will only allow the correct continuation volumes to be processed (verified by checking the data file labels). For an input file, the end-of-file message will be sent to the using program when the last volume processed contains an end-of-file trailer label (EOF).

**NL:* The volumes have no labels. On a nonlabeled volume, tape marks are used to indicate the end of each data file and the end of the volume. For an input file, the end-of-file message will be sent to the using program when the number of volumes specified in the volume list have been processed or (if no list of volume identifiers is provided) when the number of reels specified in the REELS parameter have been processed.

**NS:* The volumes have nonstandard labels. Each volume must begin with some kind of label information, optionally preceded by a tape mark. This nonstandard label information is ignored, and the system spaces forward to a point beyond the tape mark that follows the nonstandard labels to position the tape at the file's data. Each reel must have a tape mark at the end of the file's data. Any information beyond this ending tape mark is ignored. Only a single data file can exist on a nonstandard tape. Standard-label volumes *cannot* be processed using *NS label processing. For an input file, the end-of-file message will be sent to the using program when the number of volumes specified in the volume list have been processed or, if no list of volume identifiers is provided, when the number of reels specified in the REELS parameter have been processed.

**BLP:* Standard-label processing is to be bypassed. Each reel *must* have standard labels. Although each reel is checked for a standard volume label and each file must have at least one standard header label (HDR1) and one standard trailer label (EOV1 or EOF1), most other label information (such as the data file record length or block length) is ignored. The sequence number of each file on the volume is determined only by the number of tape marks between it and the beginning of tape (in contrast to *SL processing where the file sequence number stored in the header and trailer labels of each file are used to locate a data file).

Most of the information in the data file trailer labels is ignored, but if an end-of-file trailer label (EOF) is found (or the beginning-of-file header label for read backward), the end-of-file message is signaled to the program using the tape file. If no ending labels are encountered by the time the specified number of volumes or reels have been processed (volume identifier list and REELS parameter), the end-of-file message is immediately sent to the program using the tape file. Bypass label processing can be used when you do not know the name of the file to be used or (for example) when some file label information is incorrect.

**LTM:* The volumes have no labels, but have a single leading tape mark before the first data file. REELS(*LTM) is processed the same way as REELS(*NL) except that when SEQNBR(1) is specified for an output file to create the first data file on the tape, a leading tape mark is written at the beginning of the tape before the first data block.

number-of-reels: Enter the maximum number of reels that are to be processed for a *NL, *LTM, *NS, or *BLP input tape operation when there is no list of volume identifiers used (VOL parameter). If the next reel is not already mounted when the end of one tape is reached, a message is sent to the operator requesting that the next tape be mounted on the next tape device. The number-of-reels value is ignored for a standard label (*SL) processing file or for any output file.

SEQNBR Parameter: Specifies the sequence number of the data file on the tape that is to be processed. When standard-label tapes are used, the four-position file sequence number is read from the first header label of the data file. When bypass label processing is used or when standard-label tapes are *not* used, the system counts the tape marks from the beginning of the tape to locate the correct sequence number data file to be processed. (When multifile, multivolume tapes are processed using REELS(*SL), the file sequence numbers continue consecutively through all of the volumes; that is, each new data file has a sequence number that is one greater than the previous file, regardless of which volume it is on). Enter the file sequence number that is to override the sequence number specified in the program or in the device file.

LABEL Parameter: Specifies the data file identifier of the data file that is to be processed by this tape device file. The data file identifier is only defined for standard-label tapes and is stored in the header label immediately preceding the data file that the label describes. If a data file identifier is specified for any type of label processing other than *SL, it is ignored. A label identifier is *required* for a standard label output file, but is optional for an input file (since the sequence number uniquely identifies which data file to process).

For an input file or output file with EXTEND(*YES) specified, this parameter specifies the data file identifier of the file that exists on the tape. The specified identifior must be the same as the one in the labels of the data file that the SEQNBR parameter specifies; otherwise, an error message is sent to the program using this device file. For output files with EXTEND(*NO) specified, the LABEL parameter specifies the identifier of the file that is to be created on the tape. (For an expanded description of the LABEL parameter, see Appendix A.) This parameter overrides the data file identifier specified in the device file, in the program, and/or in other OVRTAPF commands executed in following invocations.

Enter the identifier (17 alphameric characters maximum) of the data file to be used with this tape device file. If this identifier is for a tape that is written in the basic exchange format, and it is to be used on a system other than System/38, a maximum of 8 characters should be used or a qualified identifier having no more than 8 characters per qualifier should be used.

RCDLEN Parameter: Specifies, in bytes, the length of the records contained in the data file that is to be processed with this device file. This parameter overrides the value specified in the device file, in the program, and/or in other OVRTAPF commands executed in following invocations. The system will always use the record length and block length specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified (if a second header label (HDR2) is found on the tape and *BLP label processing has not been specified).

*CALC: No record length is specified for the data file to be processed. If *CALC is specified, the system will attempt to calculate an appropriate record length when the file is opened. RCDLEN(*CALC) can be used for nonlabeled tapes or when there is no HDR2 label if a BLKLEN value other than *CALC is specified for the file and the RCDBLKFMT does not specify spanned or blocked records. In this case, the system calculates an appropriate record length from the block length, record block format, and buffer offset (for an ASCII file) specified for the file. In any other case, the actual record length must be specified by a CHGTAPF or OVRTAPF command, or in the HLL program that opens the device file.

record-length: Enter a value (1 through 32767) that specifies the length of each record in the data file. The minimum and maximum record length that will be allowed for a file is dependent on the record block format, block length, buffer offset (for an ASCII file), and recording code. The following table shows the minimum and maximum record length values allowed for each record block format, assuming the block length value is large enough to support the maximum record length:

Absolute RCDLEN Ranges

		FILETYPE(*DATA)		FILETYPE(*SRC)	
CODE	RCDBLKFMT	Minimum RCDLEN	Maximum RCDLEN	Minimum RCDLEN	Maximum RCDLEN
*EBCDIC	*F *FB *U	18	32767	30	32767
*ASCII	*F *FB *U	18	32767	30	32767
*EBCDIC	*V *VB	1	32759	13	32767
*ASCII	*D *DB	1	9995	13	10007
*EBCDIC	*VS *VBS	1	32759	13	32767
*ASCII	*VS *VBS	1	32759	13	32767

BLKLEN Parameter: Specifies, in bytes, the maximum length of the data blocks that will be transferred to or from the tape for output or input operations. This parameter overrides the value specified in the device file, in the program, or in other OVRTAPF commands executed in following invocations. The system will always use the block length and record length specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified (if a second header label (HDR2) is found on the tape and *BLP label processing has not been specified).

*CALC: No block length is specified for the data file to be processed. If *CALC is specified the system will attempt to calculate an appropriate block length when the file is opened. BLKLEN(*CALC) can be used for nonlabeled tapes or when there is no HDR2 label if a RCDLEN value other than *CALC is specified for the file and the RCDBLKFMT does not specify spanned or blocked records. In this case, the system calculates an appropriate block length from the record length, record block format, and buffer offset (for an ASCII file) specified for the file. In any other case, the actual block length must be specified by a CHGTAPF or OVRTAPF command, or in the HLL program that opens the device file.

block-length: Enter a value, not exceeding 32767 bytes, that specifies the maximum length of each block in the data file to be processed. The minimum block length which can be successfully processed is determined by the tape device hardware and System/38 machine support functions. The minimum value for the 3410/11 tape drive is 18 bytes. The maximum block length is always 32767 for an input file, but is limited to 9999 if block descriptors must be created for an ASCII output file. The following table shows the minimum and maximum block length values allowed for an output file:

Absolute BLKLEN Ranges

CODE	BUFOFSET	Minimum BLKLEN	Maximum BLKLEN
*EBCDIC	ignored	18	32767
*ASCII	0	18	32767
*ASCII	*BLKDSC	18	9999

BUFOFSET Parameter: Specifies the buffer offset value for the start of the first record in each block in the tape data file. A buffer offset value can be used for any record block format ASCII file, and is ignored for an EBCDIC tape file. The system will always use the buffer offset specified in the data file labels for any standard label input file or output file with EXTEND(*YES) specified if a value is contained in the second header label (HDR2) on the tape, and *BLP label processing has not been specified. This parameter overrides the value specified in the device file, in the program, and/or in other OVRTAPF commands executed in following invocations.

The buffer offset parameter specifies the length of any information that precedes the first record in the block. For record block formats *D, *DB, *VS, and *VBS, each record or record segment is preceded by a descriptor that contains the length of the record or segment. A buffer offset value is used to indicate that there is information *ahead* of the descriptor word for the first record in each block, or *ahead* of the data of the first fixed-length or undefined format record in each block.

This parameter is not needed for a standard label file processed for input if the tape includes a second file header label (HDR2) that contains the buffer offset value. A buffer offset must be provided by the CRTTAPF, CHGTAPF, or OVRTAPF command, or by the file labels for an input file that contains any information (such as a block descriptor) ahead of the first record in each block. If you do not specify a buffer offset when a tape file is created, it is not necessary to specify an offset value when the file is read.

The only buffer offset values allowed for an output file are zero and *BLKDSC. An existing standard label data file with a buffer offset value in the HDR2 label can be extended only if the offset value is either zero or four. An offset of zero in the HDR2 label adds data blocks with *no* buffer offset. BUFOFSET(*BLKDSC) must be specified to extend an existing tape data file that contains an offset value of four in the HDR2 label.

**BLKDSC:* Specifies that 4-byte block descriptors are to be created in any tape file created using this device file, and that any input file read using this device file should assume 4-bytes of buffer offset information preceding the first record in each data block. This value is only valid for a record block format *D or *DB file. The contents of the buffer offset part of each output data block when BUFOFSET(*BLKDSC) is specified as the actual length of the data block, in zoned decimal format.

buffer-offset: Enter a value (zero through 99) that specifies the length of the buffer offset information that precedes the first record in each data block.

RCDBLKFMT Parameter: Specifies the type and blocking attribute of records in the tape data file to be processed. This parameter overrides the value specified in the device file, in the program, and/or in other OVRTAPF commands executed in following invocations.

Record block format *V and *VB records can only be processed for an EBCDIC file; *D and *DB records can only be processed for an ASCII file. If a standard label tape (label type *SL or *BLP) is being processed and an inconsistent record block format is specified for the volume code, the correct record type is assumed (V or D) for the volume code and a warning message is sent to the progam that opens the file. If the record type and code are inconsistent for a nonlabeled volume (label type *NL, *LTM, or *NS), an error message is sent and the file is *not* opened, because there are no labels to verify the correct volume code.

If a valid record length, block length, and buffer offset (for an ASCII file) are specified for fixed length records but the block attribute is incorrect, the correct block attribute will be assumed (changing record block format *F to *FB or record block format *FB to *F), and a warning message sent to the program that opens the file.

If a block length is specified that is longer than required to process a maximum length record, then record block format *V, *D, or *VS will be changed to *VB, *DB, or *VBS and a warning message sent to the program which opens the file.

The following chart shows the required relationship between the record length, block length, and buffer offset (for ASCII) file parameters for an output file or an input file where the file parameters are not determined from a second file header label (HDR2):

Required RCDLEN/BLKLEN/BUFOFSET Relation[1]

CODE	RCDBLKFMT	BLKLEN = fcn(RCDLEN,BUFOFSET)
*EBCDIC	*F *U	BLKLEN = RCDLEN
*ASCII	*F *U	BLKLEN = RCDLEN + BUFOFSET
*EBCDIC	*FB	BLKLEN = RCDLEN * n
*ASCII	*FB	BLKLEN = (RCDLEN * n) + BUFOFSET
		n is the number of records in a maximum-length block
*EBCDIC	*V	BLKLEN = RCDLEN + 8
*ASCII	*D	BLKLEN = RCDLEN + 4 + BUFOFSET
*EBCDIC	*VB	BLKLEN >= RCDLEN + 8
*ASCII	*DB	BLKLEN >= RCDLEN + 4 + BUFOFSET
*EBCDIC	*VS *VBS	BLKLEN >= 18
*ASCII	*VS *VBS	BLKLEN >= 6 + BUFOFSET (18 minimum)

[1]When BUFOFSET(*BLKDSC) is specified for the file, a value of 4 should be used for the BUFOFSET part of any BLKLEN calculations, unless existing file labels on the tape specify a different value.

**F:* Fixed length, unblocked, unspanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *FB, based on other file parameters. See the explanation preceding the chart for more information.

**FB:* Fixed length, blocked, unspanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *F, based on other file parameters. See the explanation preceding the chart for more information.

**V:* Variable length, unblocked, unspanned records in EBCDIC type V format are to be processed. The system may change this record block format to *VB, *D, or *DB, based on other file parameters. See the explanation preceding the chart for more information.

**VB:* Variable length, blocked, unspanned records in EBCDIC type V format are to be processed. The system may change this record block format to *DB, based on the volume code. See the explanation preceding the chart for more information.

**D:* Variable length, unblocked, unspanned records in ASCII type D format are to be processed. The system may change this record block format to *DB, *V, or *VB, based on other file parameters. See the explanation preceding the chart for more information.

**DB:* Variable length, blocked, unspanned records in EBCDIC type D format are to be processed. The system may change this record block format to *VB, based on the volume code. See the explanation preceding the chart for more information.

**VS:* Variable length, unblocked, spanned records in either EBCDIC or ASCII code are to be processed. The system may change this record block format to *VBS, based on other file parameters. See the previous explanation for more information. Note that the representation of spanned records on the tape is different for EBCDIC and ASCII files, but the system selects the correct format based on the file code.

**VBS:* Variable length, blocked, spanned records in either EBCDIC or ASCII code are to be processed. Note that the representation of spanned records on the tape is different for EBCDIC and ASCII files, but the system selects the correct format based on the file code.

**U:* Undefined format records in either EBCDIC or ASCII code are to be processed. RCDBLKFMT(*U) records are processed as variable length records, where each record written or read is in a separate tape block. This format can be useful for processing tape files that do not meet the formatting requirements of any other record block format.

EXTEND Parameter: Specifies, for output operations to tape, whether new records are to be added to the end of a data file that is currently on the tape. (The specific data file is identified by the SEQNBR parameter and, for a standard-label file, the LABEL parameter.) If the data file is extended, it becomes the last file on the tape volume; any data files that follow it are overwritten as the specified file is extended. This parameter overrides the extend value specified in the device file, in the program, and/or in other OVRTAPF commands executed in following invocations.

**NO:* Records are not to be added to the end of the specified data file. Regardless of whether there is already a data file with the specified SEQNBR on the tape, a new data file is created (overwriting an existing data file and any files that follow it).

**YES:* New records are to be added to the end of the specified data file on tape when this device file is used.

CODE Parameter: Specifies the type of character code that is to be used by the tape device file when the system is reading or writing tape data. If a labeled volume is recorded in a different code than the value specified for the file, a warning message is sent to the program that opened the file and the volume code is assumed for the file. This parameter overrides the value specified in the program, in the device file, and/or in other OVRTAPF commands executed in following invocations.

**EBCDIC:* The EBCDIC character code is to be used with this tape device file.

**ASCII:* The ASCII character code is to be used with this tape device file.

CRTDATE Parameter: Specifies, for tape input data files and for tape output for which EXTEND(*YES) is specified, the date when the data file was written on tape. The data file creation date is stored in file labels on the tape. If a creation date is specified for any type of label processing other than *SL, it is ignored. If the creation date specified here (if any) does not match the date written on the tape, an inquiry message is sent to the operator when the file is opened. This parameter overrides the value specified in the program, device file, and/or in other OVRTAPF commands executed in following invocations.

**NONE:* The creation date of the data file is not to be checked.

creation-date: Enter the creation date of the data file to be used by the device file. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP.

EXPDATE Parameter: Specifies, for tape output data files only, the expiration date of the data file used by this device file. The data file expiration date is stored in file labels on the tape. If an expiration date is specified for any type of label processing other than *SL, it is ignored. If a date is specified, the data file is protected and cannot be overwritten until the specified expiration date. This parameter overrides the value specified in the program, device file, and/or in other OVRTAPF commands executed in following invocations.

**NONE:* No expiration date for the data file is specified; the file is not to be protected. An expired date is written in the data file labels so the file can be used as a scratch data file.

**PERM:* The data file is to be protected permanently. The date written in the tape data file labels consists of all nines.

expiration-date: Enter the expiration date on which the data file expires. The date must be specified in the format defined by the system values QDATFMT and, if separators are used, QDATSEP.

ENDOPT Parameter: Specifies the positioning operation to be performed automatically on the tape volume when the device file is closed. In the case of a multiple-volume data file, this parameter applies to the *last* reel only; all other reels are always rewound and unloaded when the end of the tape is reached. Unless an ending option is specified by the HLL program when the file is closed, this parameter overrides the ending operation specified in the device file, in the program, and/or in other OVRTAPF commands executed in following invocations.

**REWIND:* The tape is to be rewound, but not unloaded, when the file is closed.

**UNLOAD:* The tape is to be rewound and unloaded when the file is closed.

**LEAVE:* The tape should be left in its current position when the file is closed; it is not to be rewound or unloaded. This option can be used to reduce the time required to position the tape if the next tape file to open to this device uses a data file that is on the same volume.

Note: Even if ENDOPT(*LEAVE) is specified, the next tape file opened to this reel will be positioned at the beginning of some data file on the volume (or end of a data file for either read backward or for output that extends an existing data file on the volume). A tape file is always positioned at the start or end of a data file when it is opened.

WAITFILE Parameter: Specifies the number of seconds the program is to wait for the file resources to be allocated when the file is opened. If the file resources cannot be allocated in the specified wait time, an error message is sent to the program. This parameter overrides the wait time specified in the program or in the device file. (For an expanded description of the WAITFILE parameter, see Appendix A.)

**IMMED:* The program is not to wait; when the file is opened, an immediate allocation of the file resources sources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

SECURE Parameter: Specifies whether this file is to be secured from the effects of file override commands executed in previously invoked programs.

**NO:* This file is not protected from other file overrides; its values can be overridden by the effects of any file override commands executed in previously invoked programs.

**YES:* This file is protected from the effects of any file override commands executed in previously invoked programs.

SHARE Parameter: Specifies whether the ODP (open data path) for the device file can be shared with other opens of the same file in the routing step. When an ODP is shared, the programs accessing the file share such things as the file status and the buffer. When SHARE(*YES) is specified and control is passed to a program, a read operation in that program retrieves the next input record. A write operation produces the next output record. This parameter overrides the value specified in the program or device file.

**NO:* An ODP created for this file open is not to be shared. Every time a program opens the file, a new ODP to the file is created and activated.

**YES:* If the file is opened more than once, the same ODP is to be shared with each program in the routing step that also specifies SHARE (*YES) when it opens the file. This includes multiple opens in the same program.

Example

```
OVRTAPF  FILE(OUT)  VOL(DPT706)  LABEL(STATUSR)
```

This command causes a file named OUT in the using program to use the tape file STATUSR on tape volume DPT706.

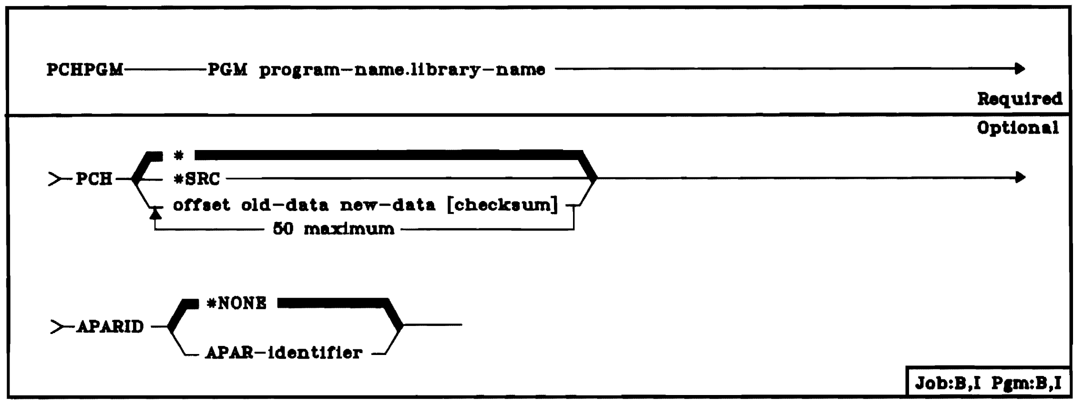
PCHPGM (Patch Program) Command

The Patch Program (PCHPGM) command repairs a program at the MI program template level. A new version of the program is created as a result of this command. The patched program is treated as a locally generated programming change. The same capabilities available for applying and removing PCs are available for applying and removing program patches. Once the program has been patched, the APYPGMCHG command must be used to apply the patch to the system for it to become effective. A patched program can be tested for proper operation before the patch is permanently applied.

The changes to the program can be entered through the PCH parameter, through a source file, or from the work station, if the command is entered interactively. When the changes are entered through the PCH parameter, as many as 50 patches can be made to a program with one PCHPGM command. In this case, each patch specifies the values for the offset location in the program template, the old values to be replaced, the new values, and any checksum values to verify the patches. Each patch can change from 1 to 16 bytes (two hexadecimal digits per byte).

If only the PGM parameter is specified, this command can be used interactively to obtain a display of the program template. Data that is to be used to patch the program can be entered on this display.

Restriction: This command is intended for use by service representatives only.



PGM Parameter: Specifies the qualified name of the program to be patched. Enter the name of the program followed by the name of the library in which the program is stored. (The library name is required.)

PCH Parameter: Specifies either the source from which the patch data is to be obtained or specifies the patch data itself.

*: The patch data is to be obtained from the patch program display for interactive jobs and from the source file QPCHSRC for batch jobs. The format of the source file records is described under *Additional Considerations*.

**SRC:* The patch data is to be obtained from the source file QPCHSRC. The format of the source file records is described under *Additional Considerations*.

offset old-data new-data [checksum]: Three or, optionally, four values are used in each patch that is to be applied to the program. As many as 50 patches can be specified, with each patch consisting of an offset location, the old (current) data at that location, the new data that is to replace the old data, and (optionally) a checksum value. Each set of patch data must be enclosed in parentheses. For example, the following parameter specifies two sets of patch data:

```
PCH((I000A3F 0ABCF3D4 1BCD4E59 1AF2) +
    (I000B2D FEDCBA9876 123456789A 12FE))
```

The values are specified as follows:

- *offset*: Enter the beginning position in the program template, from the beginning location, that specifies where that patch is to be applied. The offset value consists of one character (T, I, E, or D) and six hexadecimal digits. The characters signify that:

T	The offset is from the start of the entire template.
I	The offset is from the start of the instruction stream.
E	The offset is from the start of the OES.
D	The offset is from the start of the ODV.

- *old-data*: Enter the current value that exists in the location to be patched. The value must be specified as an even number of hexadecimal digits. The value must be no less than four and no more than 32 digits. If the value does not equal the current value, the patch is not applied.

- *new-data*: Enter the new value that is to replace the old data. The value must be specified as an even number of hexadecimal digits. The value must be no less than four and no more than 32 digits.

- *checksum (optional)*: If a checksum value is to be compared with a computed checksum, enter a 4-character hexadecimal hash total of the patch. The specified value is compared with the value computed from the old data and new data values. If the checksum value does not equal the computed value, the patch is not applied. If more than one set of patch data is specified in the PCH parameter, a checksum value must either be specified in all the sets or in none of the sets.

APARID Parameter: Specifies the APAR identifier, 6 alphameric characters long, that is to be associated with this program patch.

*NONE: There is no APAR identifier for this patch.

APAR-identifier: Enter the 6-character APAR identifier that is to be associated with this patch.

Examples

```
PCHPGM  PGM(QIBM1234.QSYS) +
     PCH((I000A3F 0ABCDF3D4 1BCD4E59 1AF2)) +
     APARID(Q27312)
```

This command patches the program QIBM1234 in the QSYS library starting at the offset 000A3F from the start of the instruction stream. The current value at that location should be 0ABCF3D4. If it is, and the computed checksum equals 1AF2, the current value is replaced by the value 1BCD4E59. The APAR identifier Q27312 is stored with the new version of the patched program. The APYPGMCHG command must be used to have the patched program replace the current version of the program.

```
PCHPGM  PGM(QIBM2345.QSYS)
```

This command patches the program QIBM2345 in the library QSYS. Assuming that the command is entered interactively, the patch data is to be obtained from the patch program display because PCH(*) is assumed.

```
//JOB JOBD(XYZ.QGPL)
  PCHPGM PGM(QIBM3456.QSYS)
     •
     •
     •
//DATA FILE(QPCHSRC) FILETYPE(*SRC)
I000A3F 0ABCF3D4 1BCD4E59 1AF2
I000B2D FEDCBA9876 123456789A 12FE
I000C3E 0123456789ABCDEF 123456789ABCDEF0 3BDC
END 2DAF
//ENDJOB
```

This batch job patches the program QIBM3456 in the library QSYS. Because PCH(*) is assumed, the patch data is to be obtained from a source file, FILETYPE(*SRC), named QPCHSRC. In the source file, the following conventions are observed.

- The offsets are specified beginning in position 1 and the END record begins in position 1.
- The data fields in each record are separated by one blank.
- An overall checksum is specified on the END record.
-

Additional Considerations

Program patch data can be entered in the PCHPGM command itself, or it can be entered through a source file. This file can be a named inline data file, a data base file, a diskette file, or a card file. If the patch data is entered in the source file, the PCH parameter must specify *SRC. For inline data files, the DATA command must specify FILE(QPCHSRC) and FILETYPE(*SRC). For any other type of file, a file override command must be used to override the file QPCHSRC with the appropriate file name.

Each patch data record in the QPCHSRC file must be in the following format.

- *offset*: The first field specifies the offset as described in the description of the PCH parameter. This field must start in the first position of the record and be followed by a single blank character.

- *old-data*: The second field specifies the current value at the offset location. This field must be an even number of hexadecimal digits (from four to 32 digits long) and must be followed by a single blank character.

- *new-data*: The third field specifies the replacement value. This field must be an even number of hexadecimal digits (from four to 32 digits long). This field must be followed by a single blank character if a checksum value is specified; otherwise, this field is the last field in the record.

- *checksum*: This optional field consists of four hexadecimal digits. If checksum is specified in one patch data record, it must be specified in all the patch data records.

The last record in the file must be the END record. This record specifies END in positions 1 through 3, followed by a blank character. If checksum values are specified in the patch data records, an overall checksum value of four hexadecimal digits must be specified following the blank.

Note: If a program is patched and an attempt is made to patch it again, the second patch is applied to the patched version, not to the original version. Likewise, if a PC is loaded and its associated object is then patched, the PC object is patched, not the original.

Until a PC or patch is permanently applied or permanently removed, two versions of the program exist on the system. When there are two versions, it is always the PC or the patched version that is patched.

PGM (Program) Command

The Program (PGM) command is used in a CL program source file to identify the beginning of a CL program that is to be compiled and to specify what parameters are to be received by the program after it is compiled. If a PGM command is used, it must be the first command in the program source file; if a PGM command is not used, a PGM command without parameters is assumed. (The name of the program is specified in the CRTCLPGM command that is used to create the CL program.)

The PGM command also specifies the parameters to be passed to the program, if any, when it is called for execution by another program. For a description of how constants are passed, refer to the description of the PARM parameter given in the CALL command description.

This program can be invoked by a CALL or TFRCTL (Transfer Control) command, or by a routing entry in a subsystem description. When the program is invoked by a CALL or TFRCTL command, parameters can be passed to it.

If parameters are defined in this command, they must be passed when the program is invoked. The parameters passed must be of the type, length and order specified in this command. Each of the parameter values passed to this program can be a character string, a numeric value, or a CL variable. When received by this program, each value is given a different CL variable name. (Each CL variable name must have been defined in the CL source file by a separate DCL (Declare) command before the program was compiled.) A maximum of 40 parameters can be passed.

Restriction: This command is valid only in a CL program.

```
                                                                    Optional
PGM ──────── PARM ─┬─CL-variable-name─┬───
                   └── 40 maximum ────┘
                                                                    Pgm:B,I
```

PARM Parameter: Specifies the names of one or more CL variables that are to receive the parameter values passed to this program. Enter a CL variable name for each of the values to be received from the calling program; the name must begin with an ampersand (&). (Null values, *N, cannot be specified for any parameter.) The parameter values are associated with the variables in the PARM parameter in the order in which they were specified on the CALL or TFRCTL commands. The type and length of each value passed must have matching attributes in the calling and receiving programs. However, for character constants, the receiving program can specify a shorter length; if so, when a character string is passed, it is truncated to the length declared in the receiving program. For information on how each data type is passed, refer to the description of the PARM parameter in the CALL command.

Examples

```
PGM
  •
  •
  •
ENDPGM
```

This PGM command is the first command in a CL program source file for a program that contains no parameters.

```
PGM  PARM(&X &Y)
```

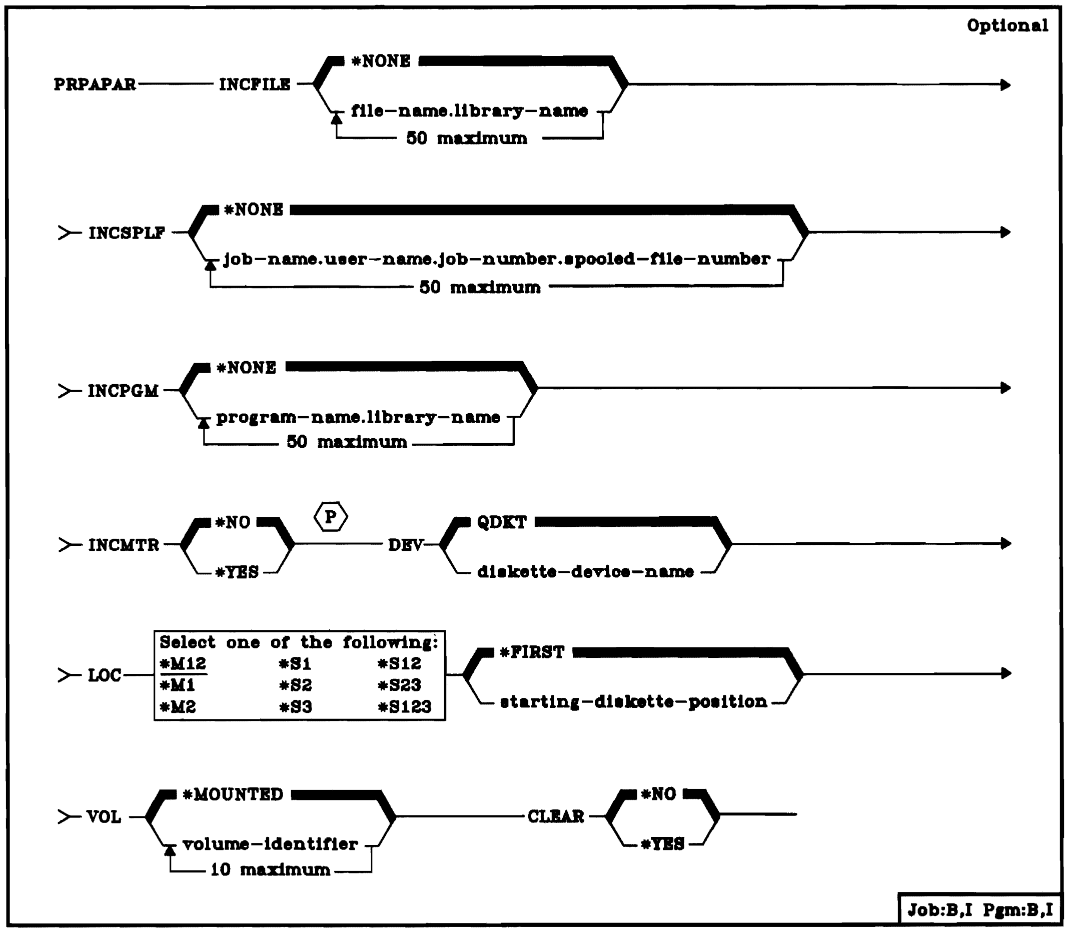
This is the first command in a program source file for a program that contains two parameters, &X and &Y, that are to have values passed to them from the calling program.

```
PGM  (&PARM1 &PARM2)
```

This is the first command in a program source file for a program that specifies two parameters in positional form, &PARM1 and &PARM2. When this program is invoked, the calling program passes, via the CALL or TFRCTL command, the values to be assumed for &PARM1 and &PARM2.

PRPAPAR (Prepare APAR) Command

The Prepare APAR (PRPAPAR) command records data on diskette for possible submission with an APAR. (The diskettes used must be in the save/restore format.) The data is used to diagnose and repair program defects found in IBM-supplied programs. The types of data that can be prepared for APAR submission are device and data base files, spooled output files, programs, and machine trouble report data.



INCFILE Parameter: Specifies the names of one or more data base or device files to be written on diskette and included with the APAR data. If the system history log or the service log is to be included, the name of the required version of the applicable data base file must be specified here.

<u>*NONE:</u> No files are to be included.

qualified-file-name: Enter the qualified name of each data base and device file that is to be written on diskette. The library qualifier *must* be specified with each file name.

INCSPLF Parameter: Specifies the names of one or more spooled output files to be included with the APAR data.

*NONE: No spooled files are to be included with the APAR data.

qualified-spooled-file-name: Enter the qualified names of one or more spooled output files that are to be included with the APAR submission. The identification of the spooled files *must* be in the following form:

job-name.user-name.job-number.spooled-file-number

INCPGM Parameter: Specifies the qualified names of one or more programs that are to be written to diskette and included with the APAR.

*NONE: No programs are to be included with the APAR data.

qualified-program-name: Enter the qualified names of one or more programs that are to be written to diskette and included with the other APAR data. The library qualifier *must* be specified with the program name.

INCMTR Parameter: Specifies whether machine internal data is to be included with the APAR data written to the diskette. This data may be necessary for a machine trouble report, which is used by IBM service representatives.

*NO: Internal data is not to be included with the APAR data.

**YES:* Internal data is to be included with the APAR data.

DEV Parameter: Specifies the name of the diskette device to which the APAR data is to be written. The device name must already be known on the system by a device description.

QDKT: The diskette device QDKT is to be the device used to write the APAR data on diskette.

diskette-device-name: Enter the name of the diskette device to be used to write the APAR data on diskette.

LOC Parameter: Specifies the magazines or slots on the diskette magazine drive that contain the diskettes on which the APAR data is to be written. Only one type of unit (magazines or slots) can be specified. If the APAR data continues past the magazines or slots specified, a message is issued to the operator to mount the additional magazines or diskettes.

If more than one unit is specified and VOL is not specified, the system begins with the leftmost unit. If VOL specifies a volume identifier, the APAR data is written on the volume and diskette specified by the identifier. The diskettes used must be in the save/restore format.

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first value specifies the unit (magazines or slots) and location that contain the diskettes on which the APAR data is to be written. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be written on first. Enter one of the following values for the starting diskette position:

*FIRST: The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

starting-diskette-position: If *M12, *M1, or *M2 is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10) in the magazine that contains the first diskette to be used. (This option is not valid for the second value if manual slots are specified for the first value.)

VOL Parameter: Specifies the volume identifiers of the diskette volumes (either in magazines or slots) on which the APAR data is to be written. The specified volumes must have been initialized in the save/restore format. The volumes must be mounted on the diskette magazine drive in the same order as specified in this parameter.

*MOUNTED: The volume that is mounted in the location specified by the LOC parameter is to contain the APAR data.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used for collecting the APAR data. For single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the *magazine* identifier (5 characters maximum) can be specified for each volume. The write operation always continues with position 1 of every magazine after the first one.

CLEAR Parameter: Specifies whether or not uncleared diskettes encountered while APAR data is being written are to be automatically cleared. (This parameter does not control initializing the diskettes; they must already be initialized in the save/restore format.)

*NO: Uncleared diskettes encountered while APAR data is being written are not to be automatically cleared. If, after the first diskette used, any uncleared diskette is encountered, an inquiry message is sent to the system operator, allowing him to terminate the operation or to specify that the currently selected diskette be cleared so the operation can continue. All diskettes to be used for the APAR data should be cleared before the PRPAPAR command is issued.

**YES:* Uncleared diskettes encountered while APAR data is being written are to be automatically cleared so the operation can continue. If *YES is specified, this parameter applies to all diskettes used in the operation, including the first one.

Examples

```
PRPAPAR  INCFILE(USERFILE.LIBX)  INCPGM(ABC.QGPL)
```

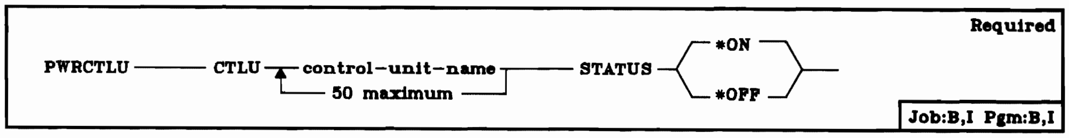
This command writes APAR data to the mounted diskette magazines M1 and M2 that includes: a data base file named USERFILE in library LIBX; a program called ABC, located in the general purpose library, QGPL.

PWRCTLU (Power Control Unit) Command

The Power Control Unit (PWRCTLU) command turns the power supply associated with a control unit on or off. Each control unit specified in this command must first be varied offline (by the VRYCTLU command) before its power can be turned off. This command is usually used by the system operator to power down a control unit that needs servicing or to turn power back on when servicing is complete.

This command applies only to the control units that have the power control feature; therefore, it applies only to the 3411 tape control unit. It does not apply to the 5251 control unit or to the work station controller.

Note: Before this command is used to power down a tape control unit, all the reels mounted on all tape units should first be unloaded to ensure the integrity of data on the tapes.



CTLU Parameter: Specifies the name of one or more control units whose power is to be turned on or off. The name of each control unit must be the same as that specified in its control unit description.

STATUS Parameter: Specifies whether the power supply is to be turned on or off.

**ON:* The power supply is to be turned on.

**OFF:* The power supply is to be turned off.

Examples

```
PWRCTLU  CTLU(CTLR01)  STATUS(*OFF)
```

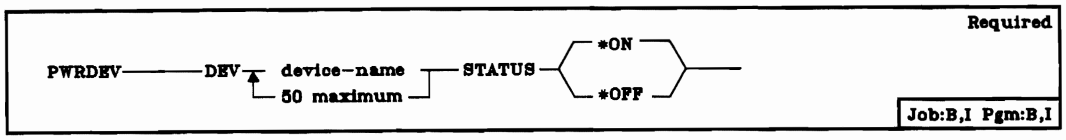
This command turns the power supply off for the control unit named CTLR01.

```
PWRCTLU  CTLU(CTLR02)  STATUS(*ON)
```

This command turns the CTLR02 power supply on.

PWRDEV (Power Device) Command

The Power Device (PWRDEV) command turns the power supply associated with a device on or off. This command applies only to the devices that have the power control feature: the 5424 MFCU, and the 3203, 3262, and 5211 printers. Each device specified in this command must first be varied offline (by the VRYDEV command) before its power can be turned off. This command is usually used by the system operator to power down a device that needs servicing or to turn power back on when servicing is complete.



DEV Parameter: Specifies the name of one or more devices whose power is to be turned on or off. The name of each device must be the same as that specified in its device description. (Only the device names specified for the MFCU and the 3203, 3262, or 5211 standalone printers can be entered.)

STATUS Parameter: Specifies whether the power supply is to be turned on or off.

**ON:* The power supply is to be turned on.

**OFF:* The power supply is to be turned off.

Examples

```
PWRDEV  DEV(READER) STATUS(*OFF)
```

This command turns the power supply off for the device named READER.

```
PWRDEV  DEV(READER) STATUS(*ON)
```

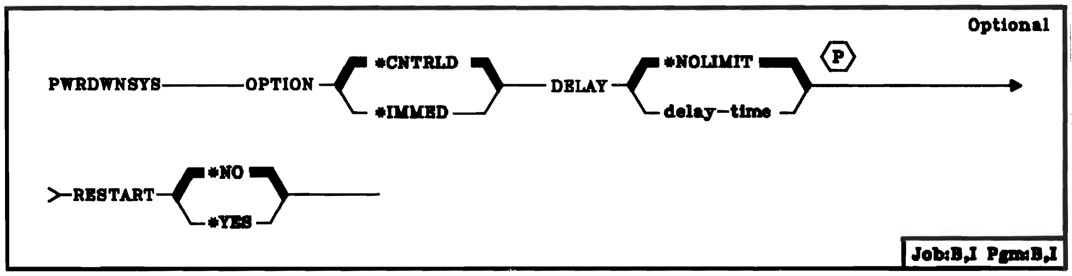
This command turns the READER power supply on.

PWRDWNSYS (Power Down System) Command

The Power Down System (PWRDWNSYS) command prepares the system for termination and then initiates the power-down sequence. All active subsystems are notified that the system is being powered down; no new jobs or routing steps can be initiated by any subsystem. For example, jobs that are on a job queue as a result of a TFRJOB command will not be allowed to be completed. During the subsequent IMPL, they will be removed from the job queue and their job logs will be produced.

Note: If 3410 and 3411 tape units are installed on the system, all tape reels that are mounted should be unloaded *before* the system is powered down; this step ensures the integrity of data on the tapes.

Restriction: To execute this command, you must have job control (*JOBCTL) authority.



OPTION Parameter: Specifies whether the system allows the active subsystem to terminate the processing of active jobs in a controlled manner (which lets the application program perform termination processing), or whether the system terminates the jobs immediately. In either case, the system does perform certain job cleanup functions.

*CNTRLD: The subsystem, within the time specified by the DELAY parameter, is to terminate all active jobs in a controlled manner. During that time, the programs being executed in those jobs are allowed to perform cleanup (termination processing). If the possibility exists that an active job could begin to loop or could send an inquiry message to QSYSOPR, you should specify a time delay with the DELAY parameter (rather than specifying DELAY(*NOLIMIT)).

**IMMED:* The subsystem is to terminate all active jobs immediately, meaning the programs executing in those jobs are not allowed to perform any cleanup. Thus, a minimum amount of time is required when *IMMED is specified. The amount of time allowed for cleanup when *IMMED is specified is controlled by the system value QPWRDWNLMT. (This option might cause undesirable results if data has been partially updated and, therefore should be used only after a controlled cancel has been attempted unsuccessfully.)

When OPTION(*IMMED) is specified while the system is operating under auxiliary power, or if the delay time specified in the DELAY parameter expires while the system is under auxiliary power, the system ignores the QPWRDWNLMT system value and initiates the power-down sequence without further job cleanup activity.

DELAY Parameter: Specifies the amount of time, in seconds, that the system allows a controlled termination to be performed by the active subsystems. If the job termination routines are not finished in the specified delay time, any remaining jobs are canceled immediately.

*NOLIMIT: The amount of time in which to complete a controlled termination process is not limited.

delay-time: Enter the maximum amount of delay time, in seconds, in which a controlled termination can be performed. Valid values are 1 through 99999 (99 999 seconds).

RESTART Parameter: Specifies whether the system terminates and powers down, or whether the system undergoes termination processing and then restarts.

*NO: The system is to terminate and power down.

**YES:* The system undergoes termination processing (but does not power down) and then restarts if the system is using utility power. If the system is on auxiliary power, the system powers down and an auto-IMPL occurs when utility power is restored (if the auto-IMPL feature is activated).

Examples

```
PWRDWNSYS  OPTION(*IMMED)
```

This command causes the system to perform an immediate termination without allowing any active jobs to perform cleanup routines. Once the system completes its termination functions, it initiates the power-down sequence.

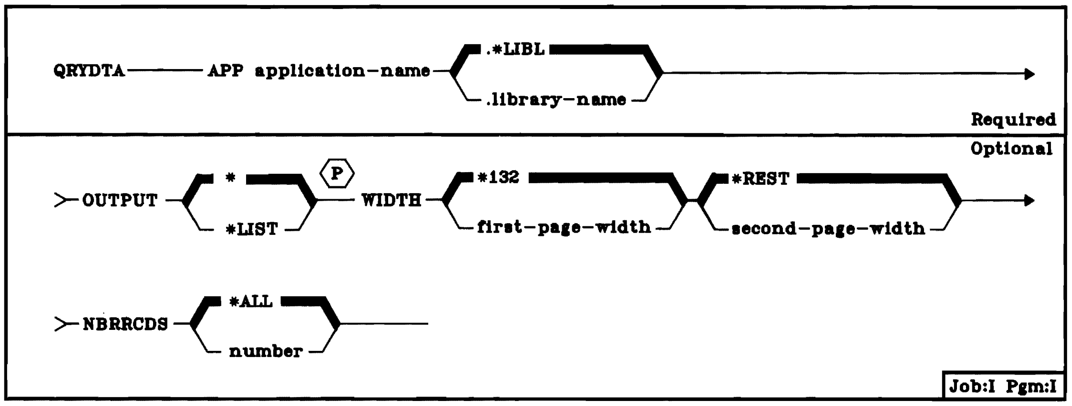
```
SBMJOB  JOB(LASTJOB) JOBD(QBATCH) JOBPTY(9) JOBQ(QBATCH) +
      RQSDTA('PWRDWNSYS *CNTRLD 3600')
```

This command submits a low priority batch job that, when executed, causes the system to perform a controlled termination. The controlled termination is allowed one hour (3600 seconds) for completion before any remaining jobs are canceled. This method of issuing the PWRDWNSYS command could be used to allow other higher priority jobs on job queue QBATCH (including those that are on the queue as a result of a TFRJOB command) to be completed before the PWRDWNSYS command is executed. There must be an active subsystem for which the QBATCH job queue is a source of work.

QRYDTA (Query Data) Command

The Query Data (QRYDTA) command executes a query application to create a report.

The Query Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Query Utility, refer to the *IBM System/38 Query Utility Reference Manual and User's Guide,* SC21-7755.



APP Parameter: Specifies the qualified name of the query application you want to execute, and specifies the library in which it is stored. If no library qualifier is specified, *LIBL is used to find the application.

OUTPUT Parameter: Specifies whether the report is to be displayed before it is printed.

<u>*</u>: If the QRYDTA command is entered from a work station, the report is displayed on the screen and printed at the end of the job. If the QRYDTA command is not entered from a work station, the report is printed by the system printer.

**LIST*: The report is to be printed by the system printer.

WIDTH Parameter: Specifies the number of characters to be printed on each page of your report. When the QRYFLD or TABLE listings produce a column of text that is split between two pages, you can use the parameter to shorten the first page width to force the entire column to the second page of the listing.

When the queried file is overridden to a data base file, this parameter is ignored and the length of the record of the file is used. This parameter does not change the forms width of the printer device file, so you can specify this parameter separately as an override.

<u>132:</u> The width of the first page is to be 132 character positions.

first-page-width: Specifies the width of the first printed page of the query report.

<u>*REST:</u> The pages following the first page are to have the same character width as specified for the first.

second-page-width: Specifies the width of the pages following the first page.

NBRRCDS Parameter: This parameter specifies the number of records you want query to process.

<u>*ALL:</u> Query will process all records in the file.

number: Specify the number of records you want query to process.

Example

```
QRYDTA APP(LOOKUP.BOOK) OUTPUT(*LIST)
```

This command executes the application named LOOKUP in library BOOK, and prints the results on the system printer. The width of the first page is 132 character positions.

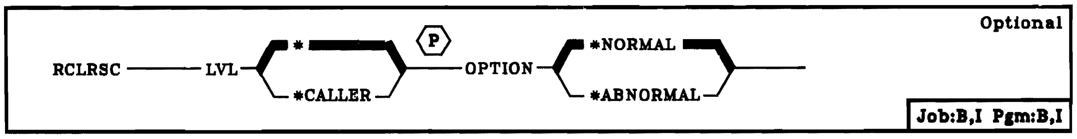
RCLRSC (Reclaim Resources) Command

The Reclaim Resources (RCLRSC) command is used in the controlling program of a set of programs (application) to free the static storage and close any files that were left open by other programs in the application that are no longer active. This command should be used only in the controlling program of the application. When one of the programs in the application (other than the controlling program) terminates abnormally, the RCLRSC command can be used with OPTION(*ABNORMAL) to free the static storage and close the files opened by that program, and to notify, using the communications file, the host system application program of an abnormal termination. The controlling program can then recover from the error and continue. The storage and files can be reused by other programs that are invoked. Default processing will free static storage and close files normally.

The RCLRSC command is not needed to reclaim the resources of RPG programs that have the LR (last record) indicator set on, all CL programs that terminate (return) normally, and COBOL programs that terminate applications with STOP RUN, because, on a normal return, the storage is freed and the files are closed automatically.

For information about how static storage is used, see the *CPF Programmer's Guide* and the publications that apply to the high-level languages.

Restriction: The RCLRSC command should not specify LVL(*CALLER) if it is used in a CL program that also uses the SNDF, RCVF, or SNDRCVF commands. Using RCLRSC LVL(*CALLER) in such a program causes unpredictable results when the SNDF, RCVF, or SNDRCVF commands are used after program execution.



LVL Parameter: Specifies the reference level at which static storage is to be freed.

*: The reference level is the program that contains this RCLRSC command. The static storage is freed and files closed that were used by programs that have finished execution and returned control to this program.

**CALLER:* The reference level is the program that called the program containing this RCLRSC command. The effect is the same as if the command were issued in that program. This parameter value allows controlling programs that are written in a high-level language to call a CL program to free static storage and close files back to the level of the controlling program.

OPTION Parameter: Specifies whether the host systems attached to BSC and communications files that are being closed will be given a normal or abnormal close notification. For other types of files, this parameter is ignored.

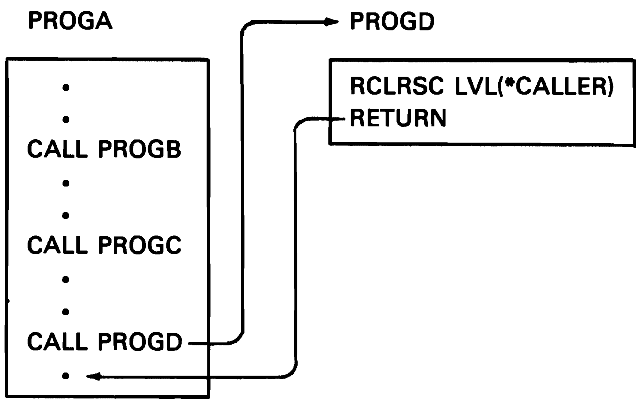
*NORMAL: Host systems attached to BSC and communications files that are being closed will be given a normal notification when the file is closed.

ABNORMAL: Host systems attached to BSC and communications files that are being closed will be given an abnormal notification when the file is closed. Use this value when the controlling program detects error conditions that should be communicated to the host systems (the error condition need not be file-related).

Examples

```
PROGA
  •
  •
CALL PROGB
RCLRSC
  •
  •
CALL PROGC
RCLRSC
  •
  •
  •
```

In this example, PROGA is a controlling program in an application. PROGA calls other programs, which return control to PROGA when they have finished executing. Because control is returned to the next sequential instruction, the RCLRSC command is issued following each CALL command to free the static storage that was used by the called program, and to close the files that were left open.



In this example, PROGA is a controlling program that is written in another high-level language. The RCLRSC command cannot be issued from the high-level language program so PROGD, a CL program, is called to issue the command. When the RCLRSC command is issued in PROGD, the static storage used by PROGB and PROGC is freed; files that were left open are closed.

Any other use of the RCLRSC command may result in files that remain open and storage that remains allocated.

RCLSTG (Reclaim Storage) Command

The Reclaim Storage (RCLSTG) command is used to perform a general cleanup of auxiliary storage. This command should be used after an unexpected failure occurs (such as a power or hardware failure) to correct abnormal conditions on objects in auxiliary storage that were affected by the failure. The command corrects, where possible, objects that were incompletely updated (such as data base files, libraries, and device descriptions) and user profiles containing incorrectly recorded object ownership information. Any unusable objects or fragments are deleted.

Before the RCLSTG command can be executed, all subsystems must be made inactive so that this job, which must be in the controlling subsystem, is the only job active in the system. This can be done either by the TRMCPF command or by TRMSBS *ALL. (If the controlling subsystem is the only one active, it still must be terminated.)

Because the amount of time varies with the number of objects in auxiliary storage, the system periodically sends messages to the work station from which the command was entered. (Note that the RCLSTG command causes the QSYSOPR message queue to be placed in break mode with a severity code of 50; the message queue can be reset, by the CHGMSGQ command, back to notify mode after the auxiliary storage has been reclaimed.)

The RCLSTG command can also be used to reclaim storage when, during a start CPF operation, insufficient storage is available to make CPF fully operational. In this case, the system operator can immediately enter the command after receiving the message about insufficient storage.

If very little additional auxiliary storage is available, the system overhead required to execute the RCLSTG command may use the additional storage; in that case, the RCLSTG command will fail.

Restrictions: All subsystems must be inactive before the RCLSTG command can be entered. This command can be used in an interactive job only.

```
RCLSTG ——
                                                                 Job:I
```

There are no parameters for this command.

Example

```
RCLSTG
```

This command, entered interactively, locates all objects on the system created prior to the last IMPL. Those that have no owners are given default owners, and those that have become lost from their libraries either are inserted into the QRCL library or are deleted. Lost objects that are deleted are certain user objects (user profiles and subsystem descriptions) and certain CPF objects that are damaged and are not usable. The QRCL library, created (when needed) by the RCLSTG command, is a permanent library; but if it contains no objects at the end of the operation because they were all reclaimed, the library is deleted.

Additional Considerations

The following items indicate what happens to objects (including data) that are lost; that is, the object can no longer be addressed by the user through commands because the system cannot determine which library contains the object.

- Objects that cannot be identified as being in any library are placed in the reclaim library QRCL, which (when needed) is automatically created to contain the lost objects. An information message is sent to the QSYSOPR message queue for each object inserted into QRCL.
  - For user-created objects remaining in QRCL after storage has been reclaimed, the user must either move them to the appropriate library or delete them. It may also be necessary to use the GRTOBJAUT command to regrant specific rights to the reclaimed objects.
  - If any IBM-supplied objects are left in QRCL, the user should contact his programming support representative (PSR).
  - If more than one lost object has the same name and type, a unique name is assigned to each of those objects, except the first one, and its original name is retained in the object's text description.
  - For all lost objects, information such as save/restore history information, PC status information, and text descriptions may no longer be retrievable, even though the objects themselves are reclaimed.

- User-created objects whose owners cannot be identified are assigned to the security officer user profile QSECOFR. After storage has been reclaimed, the CHGOBJOWN command can be used to transfer each object back to its original owner or to a new owner. As an object is transferred to an owner, a message, stating that the object has been transferred and giving the name of the owner to whom it is transferred, is sent to the QSYSOPR message queue.

- Lost data base files for which data still exists, and data spaces that contain valid data but are not addressable, are rebuilt as field-level files. These field-level files are identified in the QRCL library in their text descriptions.

- A log of the actions taken by the RCLSTG command (such as objects deleted or renamed, and ownerships transferred) is sent to the QSYSOPR message queue and to the QHST system log. The logged objects can be displayed by the DSPOBJD command for additional information about the damaged or repaired object.

RCVDTAARA (Receive Data Area) Command

The Receive Data Area (RCVDTAARA) command is used in a CL program to copy the value of a data area into a CL variable. The associated CL variable has the same name as the data area except that the variable name is preceded by an &. The data area must have already been created by a CRTDTAARA command and declared in the program by a DCLDTAARA command. The CL variable for the data area is declared automatically when the data area is declared.

When the RCVDTAARA command is executed, the data area is locked to the program during the receive operation so that commands in other jobs cannot change or destroy it until the operation is completed. If the data area is shared with other jobs and it is updated in steps involving more than one command in a job, the data area should be explicitly allocated to that job until all the steps have been performed. The data area can be explicitly allocated with the ALCOBJ command.

Restrictions: (1) This command is valid only in a CL program. (2) To use this command, you must have operational rights to the data area and read rights for both the data area and the library in which it is stored. (3) The attributes of the data area (and the declared CL variable) at execution time must be the same as those of the data area when the program was compiled.

```
                            ①                                          Required
RCVDTAARA ────────DTAARA data-area-name─────

① A variable cannot be coded on this parameter.                        Pgm:B,I
```

DTAARA Parameter: Specifies the name of the data area whose value is to be copied into its corresponding CL variable in the program. A CL variable name cannot be used in this parameter to specify the name of the data area.

Example

```
DCLDTAARA  DTAARA(CHECKNUM.PAYLIB)
   •
   •
   •
RCVDTAARA  DTAARA(CHECKNUM)
CALL PAYROLL &CHECKNUM
```

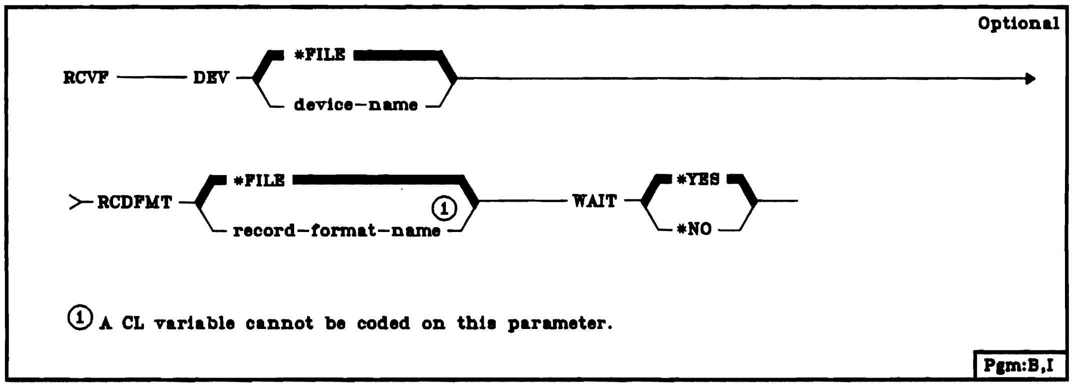
In this example, the RCVDTAARA command causes the value in the data area named CHECKNUM, in the library PAYLIB, to be copied into the CL variable named &CHECKNUM so that it can be passed as a parameter to the program PAYROLL. To copy the data area value into the program, the user must have read rights for both the data area and the library.

RCVF (Receive File) Command

The Receive File (RCVF) command is used by a CL program to receive data from a display device. The command reads a record from the display's device file and puts the data from the record into one or more CL variables. These CL variables were automatically declared in the program when the CL source program was compiled and a DCLF (Declare File) command was processed as part of the source. There is one CL variable for each field in the record format used to receive the data. The data, then, that is entered by a user on the display is copied into CL variables in the program by the RCVF command, where it can be processed by the program.

Only one record format, of those specified in the DCLF command, can be specified in each RCVF command. If the device file has not been opened, it is opened by this command. The file and record format specified in this command can be overridden by an OVRDSPF (Override with Display File) command if that command is entered before the file is opened. However, care should be taken that the fields in the overriding record format correspond to the CL variables declared in the program.

Restriction: This command is valid only within a CL program.



① A CL variable cannot be coded on this parameter.

DEV Parameter: Specifies the name of the display device from which data is to be received. If a CL variable name is used in this parameter, only one RCVF command is needed in the program to receive data from several devices. The device name in the variable can be changed between executions of the command.

*FILE: The user's data is to be received from the device associated with the device file (the device file that was declared in the FILE parameter of the DCLF command). If more than one device name is specified in the device file, *FILE cannot be specified.

device-name: Enter the name of the device or the name of the CL variable that is to contain the name of the device from which the user's data is to be received.

RCDFMT Parameter: Specifies the name of the record format that is to be used to receive data from the user. The format contains all the fields in the record. This parameter must be coded with a record format name if there is more than one record format in the device file; *FILE cannot be coded if there is more than one.

*FILE: There is only one record format in the device file; that is the format in which the user's data is to be received.

record-format-name: Enter the name of the record format in which the data records from the display device are to be received. A CL variable cannot be used here to specify the record format name.

WAIT Parameter: Specifies whether the CL program is to wait for the data to be received from the user's device or continue executing the commands that follow this RCVF command. If WAIT(*NO) is specified, the program must issue a WAIT command later in the program to complete the input operation.

*YES: The program waits until the input operation from the device is completed; the next command is not executed until then.

**NO:* The program does not wait for the input data; commands continue to be executed until a WAIT command is reached later in the program.

Examples

```
DCLF  FILE(MENU1)
  •
  •
  •
RCVF
```

The CL program is to receive data from the user through the file named MENU1. The program waits for the user data before it continues executing.

```
DCLF  FILE(SCREENX) RCDFMT(R1 R2)
  •
  •
  •
RCVF  DEV(DISPLAY2) RCDFMT(R1)
```

The CL program is to receive data from the user at the display device named DISPLAY2. The data is to be received in the record format named R1 in the device file named SCREENX. The program waits for the user data before it continues execution.

```
DCLF  FILE(MSCREEN) RCDFMT(MIN1 MIN2 MIN3)
  •
  •
  •
RCVF  DEV(&DNAME) RCDFMT(MIN2) WAIT(*NO)
WAIT  DEV(&DNAME)
```

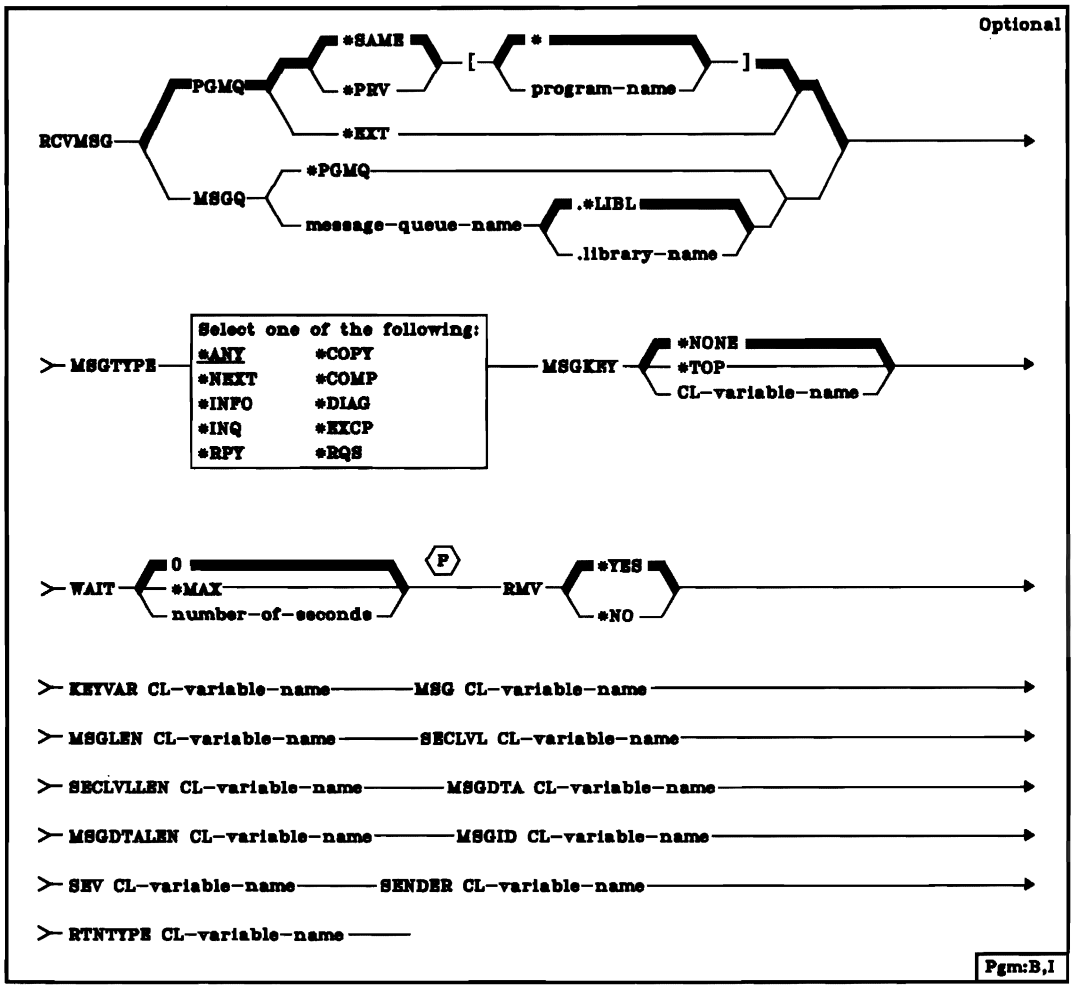
The CL program is to receive user data from several devices one at a time by way of the device file named MSCREEN. The program can receive data from the device named in the variable &DNAME using the record format MIN2, but it does not wait for the data to come in. The same RCVF command can be used to receive data from several devices; because the CL variable &DNAME is used, only the device name in the DEV parameter has to be changed for each execution of the command. A WAIT command for each device must be issued later in the program because the WAIT command actually receives the data. Both the RCVF and the WAIT commands may be executed for each device (one at a time) to send data to the program. (If a user response is delayed, the commands can be executed as many times as necessary until the user responds with the data or a CNLRCV command cancels the request.)

RCVMSG (Receive Message) Command

The Receive Message (RCVMSG) command is used by a program to receive a message sent to a message queue. This command copies a message received in the specified message queue into CL variables within the program. The message and its attributes are copied into the variables specified by the parameters KEYVAR through RTNTYPE.

You can specify the message to be received by indicating the message type, the reference key of the message, or both. The program receiving the message can also specify, on the RCVMSG command, whether a message is to be removed from the message queue or left there as an old message. If the specified message queue is not allocated to the job in which this command is entered, it is implicitly allocated by this command for the duration of the command's execution.

If a message of the specified type has not been received by the queue, the requesting program can either wait for a message to arrive or continue with other processing. This allows a set of message queues to be polled.



PGMQ Parameter: Specifies the program message queue from which the program is to receive a message. (The name of each program message queue is the same as its corresponding program invocation.) If values are specified for the PGMQ parameter, the MSGQ parameter cannot be specified.

A list of two values is used to specify the program message queue from which a message is to be received. The first value specifies the relationship of the sending program invocation with respect to the program invocation specified by the second value.

One of the following values can be specified for the *first* value:

SAME: A message is to be received from the program message queue whose name is specified as the *second* value in this parameter; otherwise, if the value * is specified or assumed, the message is to be received from the program message queue of the program in which this RCVMSG command is used.

**PRV:* A message is to be received from the program message queue of the calling program—that is, the message queue of the program previous to the program named in the second value of this parameter.

One of the following values can be specified for the *second* value:

*: The message is to be received from this program's program message queue (the program in which the RCVMSG command is used).

program-name: The message is to be received either from the message queue of the lowest invocation of the specified program, or from the message queue of the program that called the specified program. A program name can be specified as the second value of this parameter only if *SAME or *PRV is specified as the first value. If *SAME is specified, the message is to be received from the program queue of the lowest level invocation of the program specified here. If *PRV is specified, the message is to be received from the program queue of the caller of the lowest level invocation of the program specified here.

**EXT:* A message is to be received from the external message queue of the job.

MSGQ Parameter: Specifies the name of the message queue (not a program message queue) from which a message is to be received. If MSGQ is specified, the PGMQ parameter cannot be specified.

**PGMQ:* The program message queue specified in the PGMQ parameter is the only queue from which a message is to be received.

qualified-message-queue-name: Enter the qualified name of the message queue from which a message is to be received. (If no library qualifier is given, *LIBL is used to find the queue.)

MSGTYPE Parameter: Specifies the type of message to be received by this program. The message types *COMP, *DIAG, *EXCP, and *RQS can only be received from a program message queue. (For coding relationships between the MSGTYPE and MSGKEY parameters, see *Coding Relationships* in the MSGKEY parameter.)

<u>*ANY:</u> Any type of message (except a sender's copy) is to be received. If a sender's copy is to be received, MSGTYPE(*COPY) must be specified.

**NEXT:* The message that follows the one specified in the MSGKEY parameter is to be received.

**INFO:* An information only message is to be received. This includes all messages that are not one of the following message types.

**INQ:* An inquiry message is to be received. A reply must be sent by the program to an inquiry message.

**RPY:* A reply message is to be received. This program has sent an inquiry message to a message queue and expects a reply.

**COPY:* A copy of an inquiry message that was previously sent is to be received by this program. The sending program's copy is received from the message queue specified as the reply message queue when the inquiry message was sent.

**COMP:* A completion message is to be received; it indicates the status of the work performed that this program requested of another program.

**DIAG:* A diagnostic message is to be received; it provides information about errors detected by another program in the input sent by this program or errors that occurred when the requested function was executed by the other program.

**EXCP:* An exception message is to be received. Exception messages (escape and notify message types) are received by the program in last in, first out (LIFO) order. The receiving program can monitor its program message queue for these messages by using the MONMSG command.

**RQS:* A request message is to be received; the message specifies a request for the execution of a function, specified, for example, by a CL command. The receiving program is to perform the function requested.

MSGKEY Parameter: Specifies the message key of the message that is to be received.

<u>*NONE:</u> No message key is specified.

**TOP:* The top of the message queue is to be used. *TOP can be used only when MSGTYPE(*NEXT) is specified and causes the first message on the message queue to be received. For program message queues, this is the message following the last request message that was received, if any.

CL-variable-name: Enter the name of the CL variable that contains the message reference key of the message to be received by this program. This key is assigned by the system and is not displayable. (To receive an old message, you must specify its message reference key.) The variable must be a character variable having a length of 4 characters.

Coding Relationships: The MSGTYPE and MSGKEY parameters can be used separately in the RCVMSG command, or together.

- If neither MSGTYPE nor MSGKEY is specified, MSGTYPE(*ANY) is assumed and the first *new* message in the queue is to be received; that is, the messages are received in FIFO (first-in, first-out) order.
- If only MSGTYPE is specified, a new message of the specified type is to be received. If the type is *COMP, *DIAG, *INFO, *INQ, *RPY, or *RQS, new messages are received in FIFO order.
- If only MSGKEY is specified with a CL variable name and the message queue contains a message with the specified message reference key, that message is received. Because a reply and the sender's copy of the original message have the same message reference key, the reply to the message is received if it is available. If the reply is not available, no message is returned. If a message is requested by key and the message is not available, an escape message is sent to the requesting program.
- If both MSGTYPE and MSGKEY (CL-variable-name) are specified and the message queue has a message of that type, the message is received by the program. If the reference key is correct and the message type is not, then an error message is sent to the program.
- If MSGTYPE(*NEXT) is specified, MSGKEY must be specified. The message following the message with the specified reference key is to be received. If MSGKEY(*TOP) is specified with MSGTYPE(*NEXT), the first message on the message queue is to be received. For program message queues, this is the first message following the last request message received.
- If MSGTYPE(*COPY) and MSGKEY (CL-variable-name) are specified, the sender's copy of an inquiry message is received.

If MSGTYPE(*RPY) is specified with a MSGKEY value that references either a sender's copy or an inquiry message, any reply to the sender's copy or inquiry message will be returned. If there is no reply to that sender's copy or inquiry message, blanks will be returned. When the MSGKEY value references an inquiry message, the WAIT parameter is ignored.

If MSGTYPE(*ANY) is specified with a KEYVAR variable and the first message type found is a reply message, the KEYVAR variable will return the message reference key of the sender's copy message. Similarly, if MSGTYPE(*RPY) is specified with a KEYVAR variable, the message reference key of the sender's copy message will be returned.

WAIT Parameter: Specifies, in seconds, the length of time that the program is to wait for a message of the specified type to arrive in the message queue if it is not there when this RCVMSG command executes. If the message does not arrive in the specified time, the CL variables named to receive message fields are filled with blanks.

The program cannot wait for a message from a *program* message queue unless it is receiving a reply.

If a wait time is specified (not zero), the message queue is implicitly allocated to the first user whose message is received; it is not deallocated until the request has been handled by the program.

If a message is sent to a message queue within the same job, and the message queue is in break delivery mode, this parameter is ignored.

If the value specified for MSGKEY references an inquiry message, and MSGTYPE(*RPY) has been specified, the program ignores the WAIT parameter.

0: The program is not to wait for the arrival of a message. If the desired message is not in the queue when this command is executed, the CL variables specified are filled with blanks (or zeros, if a decimal variable).

**MAX:* The program is to wait indefinitely for the arrival of the specified message.

number-of-seconds: Enter the number of seconds that the program is to wait for the arrival of a message.

RMV Parameter: Specifies whether the message received by the program is to be removed from the message queue.

*YES: The message is to be removed from the message queue.

**NO:* The message is not to be removed from the message queue. It becomes an old message and can be received again only when the CL variable for the message's reference key is specified.

Parameters for Received Message Fields

All of the following parameters are used to specify the names of the CL variables that are to receive the specified fields and attributes of a message when the message is received by the program. If WAIT (number-of-seconds) is specified, and the time-out occurs, the variables are filled with blanks. (The variables must already be declared in the program.) If the message field to be returned is larger than the CL variable specified, the message field is truncated; if the message field is shorter, it is padded with blanks. If the program does not need the value for a specific message parameter, no CL variable is specified for it. If a parameter is not specified, the corresponding message value is not received in the program.

KEYVAR Parameter: Specifies the name of the CL character variable, if any, that is to contain the message reference key that identifies the message received by the program containing this RCVMSG command. At the time the RCVMSG command is executed, the system returns the message reference key to the variable specified by KEYVAR in this command and changes the received message to an old message. If the message is not found, blanks are returned for the variable. The message key can then be used in the MSGKEY parameter in a subsequent RCVMSG command to receive the *old* message. If the message is not found, blanks are returned for the KEYVAR variable. (For reply type messages, use the MSGKEY parameter on this command in conjunction with the KEYVAR parameter on the SNDPGMMSG command.) The message reference key can also be used by this program for building message subfiles. The CL variable is the name of the field for which the SFLMSGKEY keyword is specified in the DDS for the message subfile.

The variable must be a character variable having a length of 4 characters.

Note: When using the message reference key (obtained from the CL variable specified by the KEYVAR parameter of the Send Program Message (SNDPGMMSG) command) to receive the reply to an inquiry message, note that the message reference key references the sender's copy. The sender's copy message is located on the reply message queue (which defaults to the program message queue that sent the inquiry message), not the message queue to which the inquiry message was sent.

MSG Parameter: Specifies the name of the CL character variable, if any, that is to contain the first-level text of the message when the message is received by the program. This includes the message data fields that were substituted for any substitution variables in the text before the message was sent. (Replies and impromptu messages contain no message data fields.) This is a variable length field, but most messages are designed to be less than 132 characters long.

MSGLEN Parameter: Specifies the name of the CL decimal variable, if any, that is to contain the total length of the first-level message text available to be received. The variable must be a decimal variable having a length of five positions.

SECLVL Parameter: Specifies the name of the CL character variable, if any, that is to contain the second-level text of the message when the message is received by the program. This includes the message data fields that were substituted for any substitution variables in the text before the message was sent. (Replies and impromptu messages do not have second-level text.) This is a variable length field, but most second-level text is designed to be less than 1435 characters long.

SECLVLLEN Parameter: Specifies the name of the CL decimal variable, if any, that is to contain the total length of the second-level message text available to be received. The variable must be a decimal variable having a length of five positions.

MSGDTA Parameter: Specifies the name of the CL character variable, if any, that is to contain the message data record received by the program as part of the message. The message data record contains the substitution values (in a single character string) that were used in the text of the received message. The amount of data returned and its format depend on the message. Any pointers contained in system messages are invalidated.

MSGDTALEN Parameter: Specifies the name of the CL decimal variable, if any, that is to contain the total length of the message data record available to be received. The variable must be a decimal variable having a length of five positions.

MSGID Parameter: Specifies the name of the CL character variable, if any, that is to contain the message identifier of the message received by the program. The minimum length of the variable is 7 characters.

SEV Parameter: Specifies the name of the CL decimal variable, if any, that is to contain the severity code of the message received by the program. The variable must be a decimal variable having a length of two positions.

SENDER Parameter: Specifies the name of the CL character variable, if any, that is to contain the identification of the sender of the message received by the program. The variable should be at least 80 characters long. The data is returned in the variable in the following format:

- The first 26 characters identify the sending job:
  - Job name (10)
  - User name (10)
  - Job number (6)

- The next 16 characters identify the sending program:
  - Program name (12)
  - Statement number (4)

- The next 13 characters are the date time stamp:
  - Date (7 characters, in the format 0yymmdd)
  - Time (6 characters, in the format hhmmss)

- If the message is sent to a program message queue, the next 14 characters identify the receiving program:
  - Program name (10)
  - Statement number (4)

- The last 11 characters are reserved for future use.

Blanks are returned in any of the first three fields that were not specified (in the SENDER parameter of the CRTMSGQ command) when the sending message queue was created, and in the fourth field if the receiving message queue is not a program message queue.

RTNTYPE Parameter: Specifies the name of the CL variable, if any, that is to contain the type code for the message received by the program. The variable must be a character variable having a length of two positions. The following values are returned to indicate the message type:

Value	**Message Type**
01	Completion
02	Diagnostic
04	Information
05	Inquiry
08	Request
10	Request with prompting
14	Notify
15	Escape
21	Reply, not validity checked
22	Reply, validity checked
23	Reply, message default used
24	Reply, system default used

Examples

```
RCVMSG  MSGQ(SMITH)  MSGKEY(&KEY)  MSG(&WORK)
```

This command receives the message having the reference key specified by the program variable &KEY from the message queue SMITH. The text of the message is copied into the CL variable &WORK.

```
RCVMSG  MSGQ(INV)  WAIT(120)  MSG(&WORK)
```

This command receives a new message from the message queue named INV into the CL variable &WORK. The program will wait no more than 120 seconds for the arrival of a new message if there are no new messages in the message queue. If there is more than one new message in the queue, the first message in the queue is the message received by the program.

```
RCVMSG  PGMQ(*)  MSGTYPE(*COMP)  +
      MSGID(&MID)  MSG(&MTEXT)
```

This command receives a completion message from the program's message queue. The message identifier is to be placed in the variable &MID and the message text in &MTEXT.

RETURN (Return) Command

The Return (RETURN) command returns control either to the next higher program invocation in the program invocation stack or to the subsystem monitor that controls the job.

When used outside of a CL program, the command performs the same function as the CF1 key; that is, it returns control from the most recent invocation of QCL. When used in a CL program, the RETURN command returns control to the next command or HLL statement in the calling program after the command or HLL statement that called the returning program. If the RETURN command is used in the highest invocation level in the routing step (either the QCL program, which is the interpretive CL command processor, or a CL program), the routing step is terminated.

Restriction: The RETURN command is valid only within a CL program or in an interactive job.

Note: If the RETURN command is entered interactively from the highest invocation level while the subsystem is undergoing a controlled termination (termination as a result of a TRMSBS, TRMCPF, or PWRDWNSYS command), end-of-job processing will occur.

```
RETURN———
                                                         Pgm:B,I
```

There are no parameters for this command.

Example

```
RETURN
```

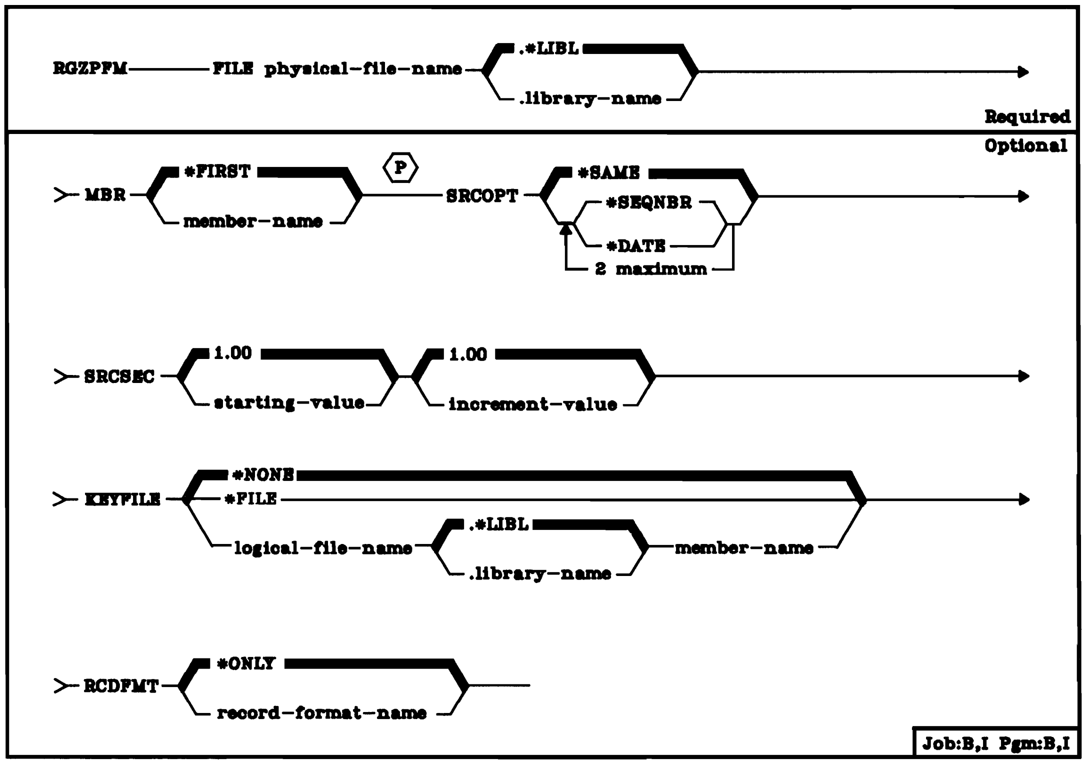
When used in a CL program, this command returns control to the CL command or HLL statement following the point in the last calling program at which this program was called. When used in a an interactive job, this command returns control to the next higher invocation of QCL. If the RETURN command is executed in the highest invocation level program (which could be QCL) in the routing step, the routing step is terminated.

RGZPFM (Reorganize Physical File Member) Command

The Reorganize Physical File Member (RGZPFM) command compresses (purges deleted records in) one member of a physical file in the data base, and it optionally reorganizes that member. If a keyed file is identified (in the KEYFILE parameter), the system reorganizes the member by changing the physical sequence of the records in storage to either match the keyed sequence of the physical file member's access path, or match the access path of a logical file member that is defined over the physical file. Reorganization can decrease file processing time when the program is reading sequentially through a keyed physical file or through a keyed logical file.

When the member is reorganized and the KEYFILE parameter is specified, the sequence in which the records are actually stored is changed and any deleted records are removed from the file. If the KEYFILE parameter is not specified, the sequence of the records is not changed but any deleted records are removed from the member. Optionally, new sequence numbers and zero date fields can be inserted in the source fields of the records. These fields are updated after the member has been compressed or reorganized.

Note: The RGZPFM command ignores all file overrides that are currently in effect for the job. The file names specified in the FILE and KEYFILE parameters identify the files actually involved in the reorganize operation, regardless of any overrides that may exist for these files.



FILE Parameter: Specifies the qualified name of the physical file whose member is to be reorganized. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name of the member in the file that is to be reorganized.

*FIRST: The first, or the only, member in the file is the member to be reorganized.

member-name: Enter the name of the file member that is to be reorganized.

SRCOPT Parameter: Specifies, for physical *source* files only, whether the member is to insert new sequence numbers inserted in the sequence number field, whether to place zeros in the date field, or whether to update both fields. Any updating occurs after the records have been compressed or reorganized.

*SAME: The SRCOPT parameter does not apply; the sequence number field and date field of records are not to be changed.

**SEQNBR:* The records are to have a new sequence number inserted into the sequence number field. The SRCSEQ parameter specifies a start value and an increment value. If *SEQNBR is specified, *DATE can also be specified.

**DATE:* The records are to have a null date (000000) inserted in the date field. If *DATE is specified, *SEQNBR can also be specified.

SRCSEQ Parameter: Specifies, only when SRCOPT(*SEQNBR) is also specified, what sequence number is to be given to the first record in the source file member and what increment value is to be used to renumber all other records in the member. If the member is to be renumbered but SRCSEQ is not specified, SRCSEQ(1.00 1.00) is assumed; the source records are renumbered sequentially beginning with 1.00, and the whole number increment of 1 is used.

1.00: The first source record in the member is to have a sequence number of 0001.00.

starting-value: Enter a value in the range of 0000.01 through 9999.99 that is to be the sequence number of the first source record in the member. A whole number of no more than four digits and/or a fraction of no more than two digits can be specified. If the starting value contains a fraction, a decimal point must be used. Examples are .01 and 3250.4. (If a value has a fraction of .00, such as 5000.00, it can be coded without the fraction; either 5000 or 5000.00 is valid.)

<u>1.00:</u> The source records are to be renumbered in the member with whole number increments of 1. (1.00, 2.00, 3.00...)

increment-value: Enter a value in the range of 0000.01 through 9999.99 that is to be used as the increment value for renumbering all source records after the first one. A whole number of no more than four digits and/or a fraction of no more than two digits can be specified. If the increment value contains a fraction, a decimal point must be used. For example, if SRCSEQ(5000 10) is specified, the first record in the reorganized member is numbered 5000.00, the second is 5010.00, the third is 5020.00, and so on. If SRCSEQ(*N .25) is specified, the records are numbered 1.00, 1.25, 1.50, 1.75, 2.00, and so on. If a starting value of .01 and an increment value of .01 are specified, there are 999,999 unique sequence numbers possible. If the maximum sequence number of 9999.99 is reached, the remaining records will be assigned the sequence number 9999.99 also.

KEYFILE Parameter: Specifies whether the physical file member is to have its arrival sequence changed to match its keyed sequence, is to be reorganized in the sequence of a logical file member, or is *not* to be reorganized at all. If KEYFILE specifies a multiple-format logical file and member, the RCDFMT parameter must also be specified.

<u>*NONE:</u> The member is not to be reorganized; it is only to be compressed by having its deleted records purged.

**FILE:* For a physical file member having a keyed sequence access path, the arrival sequence of the records in the member is to be changed to match their keyed sequence.

qualified-logical-file-name member-name: Enter the qualified name of the logical file and the name of the member within the file whose sequence is to be used to reorganize the physical file member. (If no library qualifier is given, *LIBL is used to find the logical file.)

RCDFMT Parameter: Specifies the record format name if the physical file member is to be reorganized in the sequence of a multiple-format logical file.

<u>*ONLY:</u> The logical file specified by the KEYFILE parameter has only one record format. That format is to be used to reorganize the physical file member.

record-format-name: Enter the name of a record format in the multiple-format logical file that is to be used to reorganize the physical file member.

Note: Compression of a file occurs when the space occupied by a deleted record is freed to hold an undeleted record.

Examples

Example 1: Reorganize a Member of a Physical File

```
RGZPFM  FILE(PAYROLL) MBR(MBR1)
```

This command causes member MBR1 of the PAYROLL file to be compressed by purging the deleted records from the file member.

Example 2: Reorganize a Member of a Source File

```
RGZPFM  FILE(QCLSRC) MBR(CLMBR2) +
    SRCOPT(*SEQNBR *DATE) KEYFILE(*FILE) +
    SRCSEQ(1.25)
```

This command causes the member CLMBR2 of the CL source file QCLSRC to be reorganized in keyed sequence, with the sequence number field used as the key. The reorganized member is to have new sequence numbers (beginning at 1.00 and incrementing by .25) and a null date (000000) inserted in all records when the original member is replaced.

RLSJOB (Release Job) Command

The Release Job (RLSJOB) command makes a job eligible for processing after that job has been held from processing by the HLDJOB (Hold Job) command or if the job was submitted to the system as a held job by the JOB or SBMJOB (Submit Job) commands. The job being released could have been on the job queue, on output queues, or active within a subsystem (competing for system resources) when it was held. Any spooled output files that were held because SPLFILE(*YES) was specified in the HLDJOB command are also released.

Restriction: The job being released must belong to the user issuing the command or he must have the special job control rights (*JOBCTL).

```
                                                                    Required
RLSJOB──── JOB job-name[.user-name[.job-number]]────
                                                           Job:B,I Pgm:B,I
```

JOB Parameter: Specifies the qualified name of the job that is to be released. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. If duplicates of the specified name are found, a qualified job name must be specified. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

Examples

```
RLSJOB  JOB(WSTATION1)
```

This command releases the job WSTATION1 for processing. If the corresponding HLDJOB command had specified SPLFILE(*YES), any spooled output files for WSTATION1 are also released.

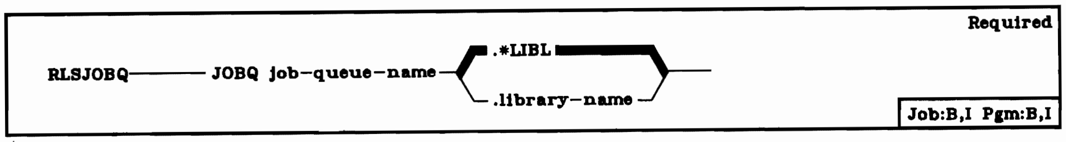
```
RLSJOB  JOB(PAYROLL.DEPTXYZ)
```

This command releases the job named PAYROLL that was submitted by a user via the user profile DEPTXYZ and later held. The qualified form of the job name is used when jobs with duplicate names exist in the system.

RLSJOBQ (Release Job Queue) Command

The Release Job Queue (RLSJOBQ) command releases for further processing the jobs on the specified job queue that were previously held by a HLDJOBQ (Hold Job Queue) command. If the jobs were held by something other than a HLDJOBQ command, they are not released.

Restriction: The user submitting this command must have read, add, and delete rights for the job queue; or his user profile must have job control rights, SPCAUT(*JOBCTL), and the job queue must have the operator control attribute, OPRCTL(*YES), specified.



JOBQ Parameter: Specifies the qualified name of the job queue to be released for further processing. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

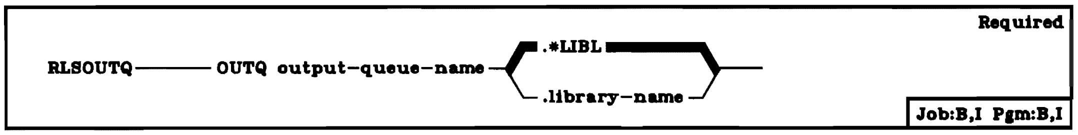
```
RLSJOBQ  JOBQ(QBATCH)
```

Any jobs on the job queue named QBATCH that were held by a HLDJOBQ command become eligible for processing, including jobs that were placed on the queue while it was being held. Specific jobs that were held by the HLDJOB command or that were put on the job queue in the held state are not released.

RLSOUTQ (Release Output Queue) Command

The Release Output Queue (RLSOUTQ) command releases the spooled files on the specified output queue that were previously held by a HLDOUTQ (Hold Output Queue) command. If the files were held by a HLDSPLF (Hold Spooled File) command or created in a held state, or if the files of a job were held by a HLDJOB (Hold Job) command, they are not released.

Restriction: The user submitting this command must have read, add, and delete rights for the output queue; or his user profile must have job control rights, SPCAUT(*JOBCTL), and the output queue must have the operator control attribute OPRCTL(*YES) specified.



OUTQ Parameter: Specifies the qualified name of the output queue to be released for further processing. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
RLSOUTQ  OUTQ(PRINTER)
```

On the output queue named PRINTER, any spooled files that were held by a HLDOUTQ command are released for further processing. This includes all spooled files whose entries were placed on the queue while it was being held, except for specific files that: have been held by the HLDSPLF command, were put on the queue in the held state, or were produced by a job that has been held by the HLDJOB command.

RLSRDR (Release Reader) Command

The Release Reader (RLSRDR) command releases the specified spooling reader for further processing. The specified reader was held by a previously issued HLDRDR (Hold Reader) command. Data is not lost.

Restriction: The user must have originally started the reader or he must have job control authority to release the reader.

```
                                                                  Required
RLSRDR────────RDR reader-name─────

                                                          Job:B,I Pgm:B,I
```

RDR Parameter: Specifies the name of the spooling reader to be released.

Example

```
RLSRDR  RDR(CARD)
```

This command releases the reader named CARD for further processing.

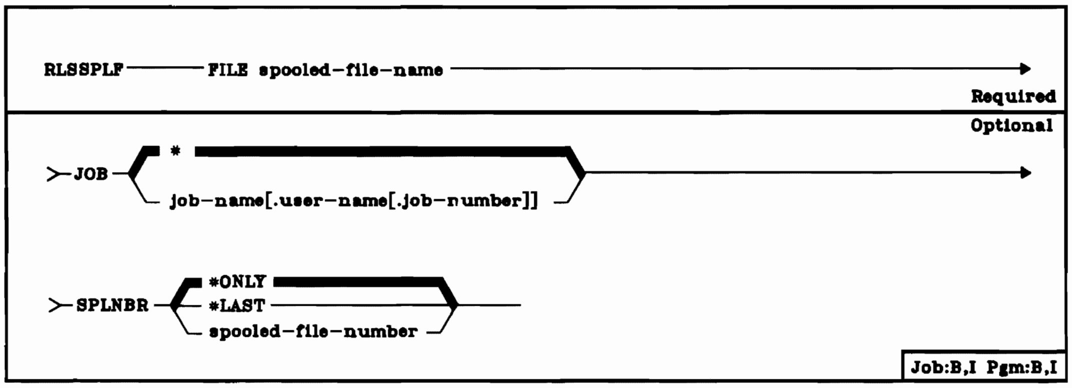
RLSSPLF (Release Spooled File) Command

The Release Spooled File (RLSSPLF) command releases the specified file on an output queue for processing by a spooling writer. The file being released is always produced from the beginning of the file. The RLSSPLF command can release a spooled file that was held by:

- A HLDSPLF command
- HOLD(*YES) being specified in the device file or on an override command
- SAVE(*YES) being specified in the device file, on an override command, or in the CHGSPLFA command
- A HLDWTR command and a RLSWTR command with OPTION(*BYPASS) specified
- The operator canceling a system request to mount forms on the printer or card device

For information on releasing multiple spooled files, refer to the *Additional Considerations* section of the DSPJOB (Display Job) command.

Restrictions: The user submitting this command must (1) be the owner of the job that created the file being released; or (2) have read, add, and delete rights for the output queue containing the file; or (3) have job control rights. The output queue must have OPRCTL(*YES) specified.



FILE Parameter: Specifies the name of the spooled file that is being released to be written to an output device. The file name is the name of the device file that was used by the program to create the spooled output file.

JOB Parameter: Specifies the name of the job that created the file being released for further processing.

*: The job that issued this RLSSPLF command is the job that produced the file being released.

qualified-job-name: Enter the qualified name of the job that created the file being released. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file being released. (For an expanded description of the SPLNBR parameter, see Appendix A.)

*ONLY: Only one spooled output file from the job has the specified file name; therefore, the number of the spooled file is not necessary. If *ONLY is specified and more than one file has the specified name, an error message is displayed to the user or (for batch jobs) sent to the program's message queue.

**LAST:* The highest numbered spooled output file with the specified file name is the file to be released.

spooled-file-number: Enter the number of the spooled file having the specified file name that is to be released.

Example

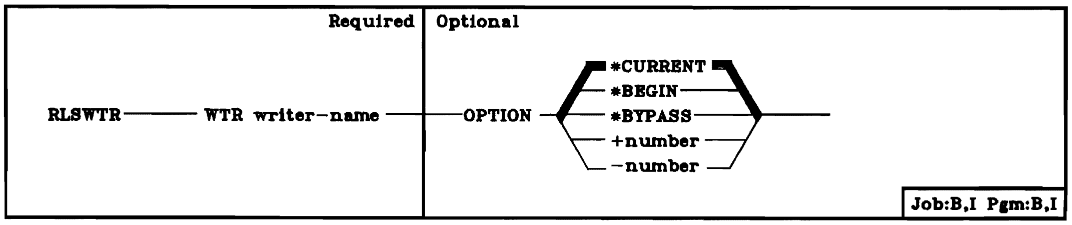
```
RLSSPLF  FILE(STOCK14)  JOB(MASTER.SMITH.000047)
```

This command releases the file named STOCK14 created in the job named MASTER. The file can now be selected for processing by the spooling writer. The job was executed under the user profile named SMITH and was assigned the job number of 000047 by the system.

RLSWTR (Release Writer) Command

The Release Writer (RLSWTR) command releases a held writer for further processing. The specified writer was held by a previously issued HLDWTR (Hold Writer) command. If the writer was writing a file when it was held, the writer can be released to resume writing the file it was writing or start writing the next file. In any case, data from the file that was being written when the HLDWTR command was issued is not lost.

Restriction: The user must have read, add, and delete rights for the output queue, or he must have job control rights, SPCAUT(*JOBCTL), specified in his user profile, and the output queue must have the operator control attribute OPRCTL(*YES) specified.



WTR Parameter: Specifies the name of the spooling writer to release for further processing.

OPTION Parameter: Specifies the point in the file at which the writer is to be released. The last three options (*BYPASS, +number, and -number) can be specified only if the writer was held while producing a file. Also, the diskette writer can be released only at *CURRENT.

CURRENT: The writer is released at the point where it had been held by the HLDWTR (Hold Writer) command.

**BEGIN:* The writer is released at the beginning of the current file.

**BYPASS:* The writer is released at the beginning of the next file. The current file is to be implicitly held on the queue.

+number: The writer is to be released *n* number of pages (for printers) or records (for other devices) past the point where it has been held. An error message is sent to the user if the specified value goes beyond the number of records or printed pages in the file.

-number: The writer is to be released *n* number of pages or records preceding the point where it has been held. An error message is sent to the user if the specified value is greater than the number of records or printed pages previously processed in the file before it was held.

Examples

```
RLSWTR  WTR(PRINTER)  OPTION(*BEGIN)
```

This command releases the writer named PRINTER to begin producing the current file at its beginning.

```
RLSWTR  WTR(QPUNCH)
```

This command releases the writer named QPUNCH to continue punching the current file where it left off when a HLDWTR (Hold Writer) command was issued.

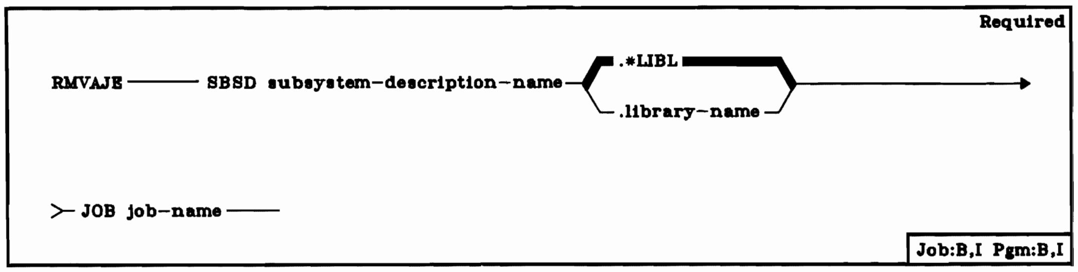
```
RLSWTR  WTR(PRTR)  OPTION(-3)
```

This command releases the writer named PRTR to begin printing again at a point 3 pages preceding the point where the writer was held. That is, the last 3 pages previously printed are the first 3 pages to be printed this time.

RMVAJE (Remove Autostart Job Entry) Command

The Remove Autostart Job Entry (RMVAJE) command removes an autostart job entry from the specified subsystem description; the associated subsystem must be inactive at the time.

Restriction: To use this command, you must have operational and object management rights for the specified subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description from which the autostart job entry is to be removed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

JOB Parameter: Specifies the name of the autostart job entry that is to be removed from the subsystem description.

Example

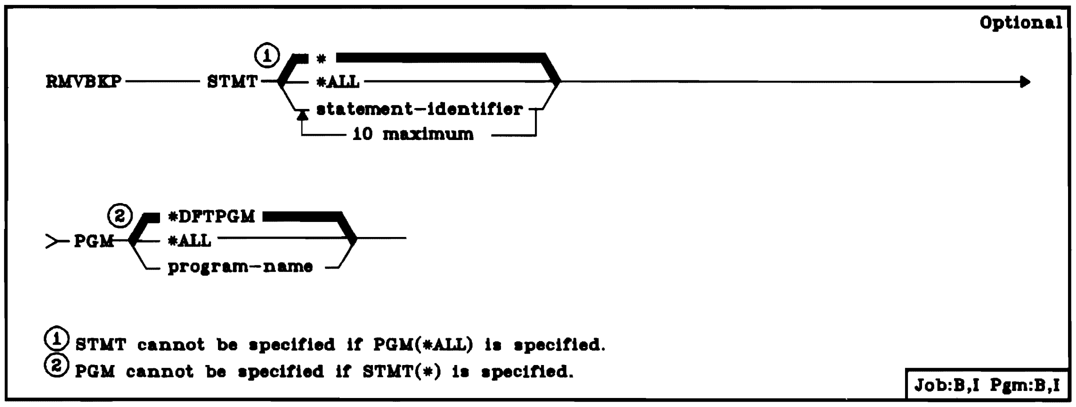
```
RMVAJE  SBSD(PAYROLL.MYLIB) JOB(INITIAL)
```

This command removes the autostart job entry named INITIAL from the PAYROLL subsystem description in the library MYLIB.

RMVBKP (Remove Breakpoint) Command

The Remove Breakpoint (RMVBKP) command removes one or more breakpoints from the specified program being debugged. It can also remove all breakpoints from all the programs currently in debug mode.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*


Optional
RMVBKP
STMT
①
*
*ALL
statement-identifier
10 maximum
PGM
②
*DFTPGM
*ALL
program-name
① STMT cannot be specified if PGM(*ALL) is specified.
② PGM cannot be specified if STMT(*) is specified.
Job:B,I Pgm:B,I

STMT Parameter: Specifies which HLL command or System/38 instruction statements in a program are to have their breakpoints removed. Breakpoints can be removed from a specified program (PGM parameter) or from the most recent program that has reached a breakpoint (PGM is not specified). If a program is specified, one or more statement identifiers can be specified or all the breakpoints can be specified. If a program is not specified, the breakpoint that the most recently halted program has reached is removed. Also, all breakpoints can be removed from all the programs in debug mode.

*: The most recent breakpoint at which a program is currently halted is the breakpoint to be removed.

**ALL:* All breakpoints in the specified program are to be removed.

statement-identifier: Enter the statement identifiers of the HLL commands or System/38 instructions that are to be removed from the program specified by the PGM parameter. No more than 10 identifiers can be specified.

PGM Parameter: Specifies the program from which the specified breakpoints are to be removed. This parameter cannot be specified when STMT (*) is specified.

*DFTPGM: The default program is the program whose breakpoints are to be removed.

**ALL:* All programs currently in debug mode are to have their breakpoints removed. This value can be specified only if the STMT parameter is omitted.

program-name: Enter the name of the program from which the specified breakpoints are to be removed.

Example

```
RMVBKP  STMT(100)
```

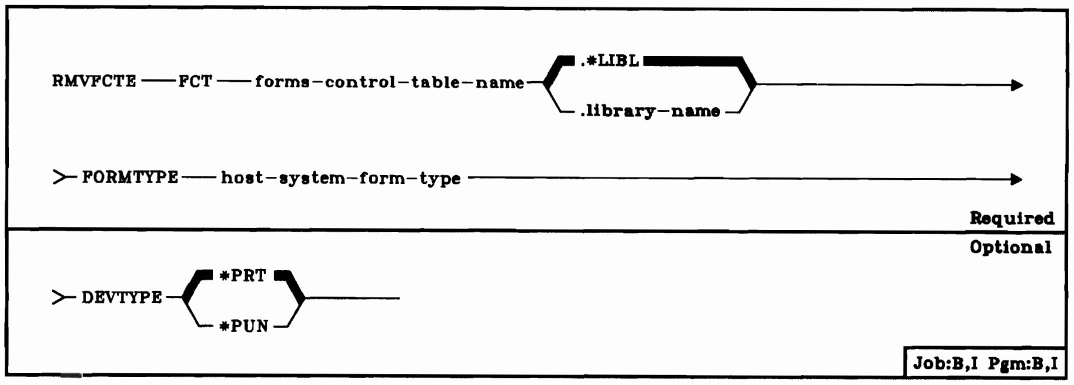
This command removes the breakpoint that is on statement 100 from the default program.

RMVFCTE (Remove Forms Control Table Entry) Command

The Remove Forms Control Table Entry (RMVFCTE) command removes an RJEF FCT (forms control table) entry from the specified inactive FCT.

Restriction: To use this command, you must have operational rights for the FCT and read rights for the library in which the FCT is stored.

The Remove Forms Control Table Entry (RMVFCTE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



FCT Parameter: Specifies the qualified name of the FCT from which the entry is to be removed. If no library qualifier is given, *LIBL is used to find the FCT.

FORMTYPE Parameter: Specifies the form type to be removed.

host-system-form-type: Enter the host system form type of the entry to be removed.

DEVTYPE Parameter: Specifies the device type FCT entry that is to be removed.

*PRT: The FCT entry associated with processing printer output streams is to be removed.

*PUN: The FCT entry associated with processing punch output streams is to be removed.

Example

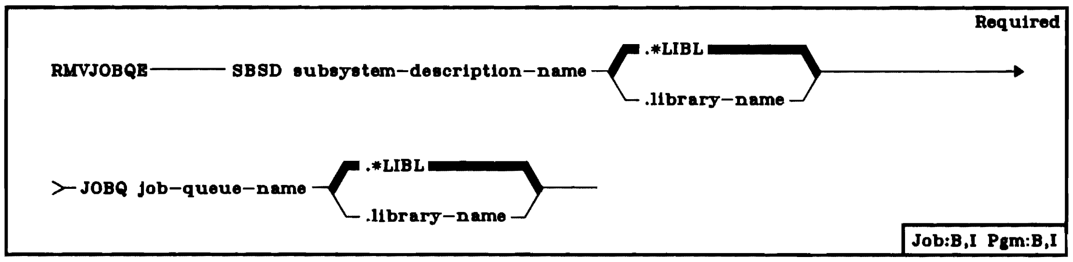
```
RMVFCTE  FCT(FORMCTRL.USERLIB) +
      FORMTYPE(MEDICAL) +
      DEVTYPE(*PUN)
```

This command removes the forms control table entry named MEDICAL, for punch device (*PUN), from forms control table called FORMCTRL in library USERLIB.

RMVJOBQE (Remove Job Queue Entry) Command

The Remove Job Queue Entry (RMVJOBQE) command removes a job queue entry from the specified subsystem description; the associated subsystem must be inactive at the time. This command can be used to remove the current job queue entry so that a different job queue can be assigned. Any jobs on the queue remain on the queue for processing when the queue is reassigned to a subsystem description and the subsystem is started.

Restriction: To use this command, you must have operational and object management rights for the specified subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description from which the job queue entry is to be removed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

JOBQ Parameter: Specifies the qualified name of the job queue that is to have its job queue entry removed from the subsystem description. (If no library qualifier is given, *LIBL is used to find the name of the job queue and the job queue must exist in a library identified in the library list.)

Example

```
RMVJOBQE  SBSD(NIGHTRUN.MYLIB)  JOBQ(BATCH2.MYLIB)
```

This command removes the job queue entry that refers to the BATCH2 job queue in MYLIB from the NIGHTRUN subsystem description stored in library MYLIB.

RMVJRNCHG (Remove Journaled Changes) Command

The Remove Journaled Changes (RMVJRNCHG) command removes the changes that have been journaled for a particular member of a data base file. The journaled changes are removed (backed-out) from the file from the specified starting point until the designated ending point has been reached. (The journal entries are processed in the opposite order from which they were generated, from most recent to oldest.) The starting point can be designated as the last journal entry on the specified journal receiver range or a particular entry on the journal receiver. The ending point can be the point at which the file has had all previous changes removed, a designated entry has been reached, or the OPEN of a file by a job.

Note: The DSPJRN command can be used to help determine the desired starting and ending points.

A list of physical files and members may be specified. The journaled changes for all physical file members are removed in the order that the journal entries are encountered on the journal (the inverse of the order that the changes were originally made to the physical file members).

If an error is encountered at any point while the journaled entries are being removed, the operation is terminated and the file member(s) may be only partially updated from the journal entries. (Termination errors include partial damage to a receiver and any logical error in the file member, such as a duplicate key.) The command also terminates when a journal entry is encountered that indicates that one of the following occurred:

- The member was cleared.
- The member was saved with storage freed.
- The member was reorganized.
- The member was restored.
- Journaling was started for the member.
- The member was deleted.
- Journal IPL synchronization failed.
- The system has already applied the changes (through the APYJRNCHG or RMVJRNCHG command). The user of the command may reissue the command, specifying a new starting sequence number, if a restart is possible.

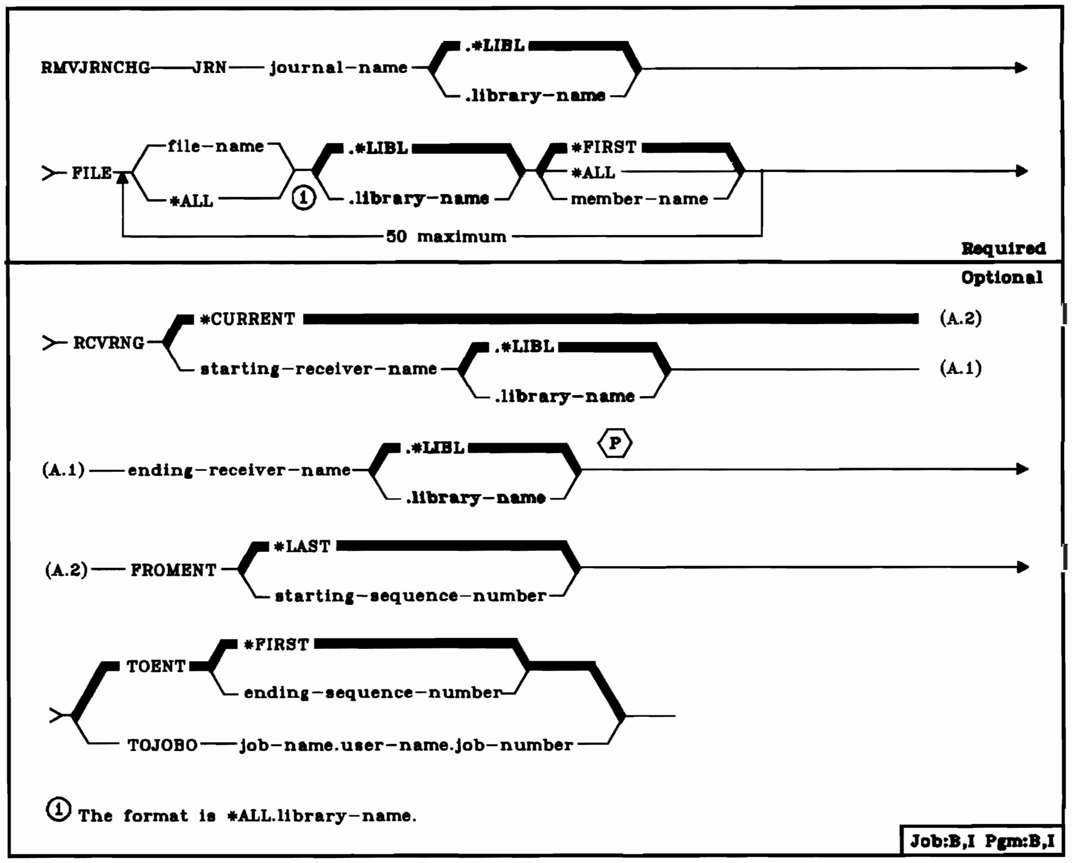
It is possible to remove changes even though the sequence numbers have been reset. The system will handle this condition, send an informational message and continue the backout procedure. If journal receivers are attached and detached in pairs (dual receivers), the system will always attempt to use the first of the two receivers (the first of the two shown in the DSPJRNA receiver directory). When the first receiver is not accessible (for example, damaged or not found), the system will attempt to use the second receiver of the pair. If neither receiver is accessible, the application of changes will terminate.

Restrictions: The files specified on this command must currently be having their changes journaled and they must have been journaled to the specified journal throughout the period indicated on the command. *Before* images are required for the designated files (see the JRNPF command). The files indicated on the command are allocated exclusively while the changes are being applied. If a file cannot be allocated exclusively, the command terminates and no journaled changes are applied.

If there is no journal entry that corresponds to the 'FROM' or 'TO' option, the command is terminated and no journaled changes are removed.

If the sequence number has been reset within the range of receivers specified, the first occurrence of the FROMENT or TOENT parameter will be used, if they are specified.

Note: If the procedure terminates for one of the members specified, it terminates for all of the members specified.



JRN Parameter: Specifies the qualified name of the journal that contains the journal entries that are to be removed. (If no library qualifier is given, *LIBL is used to find the journal.)

FILE Parameter: Specifies the qualified name of the physical data base file that is to have its journal entries backed-out (removed).

file-name: Enter the name of the physical data base file that is to have its journal entries removed. (If no library qualifier is given, *LIBL is used to find the file.)

**ALL:* All physical files within the specified library whose changes are being journaled to the specified journal will have their journal entries removed. The library name *must* be specified. If *ALL is specified and you do not have the required authority to all of the files, an exception is signaled and the command terminates.

The FILE parameter also specifies the name of the member within the file that is to have its journal entries removed.

<u>*FIRST:</u> The first member in the file is to have journal entries removed.

**ALL:* All members in the file are to have their journal entries removed.

member-name: Enter the name of the member within the file that is to have its journal entries removed.

If *ALL is specified for the first part of this parameter, the value specified for the member name is used for all applicable files within the library. For example, if *FIRST is specified, the first member of all applicable files in the library will have the changes removed.

Note: A maximum 256 members can have their changes removed with one invocation of the command. If this maximum is exceeded, an exception will be signaled and no changes will be removed. You must change the values entered on the FILE parameter so that the limit is not exceeded.

RCVRNG Parameter: Specifies the first and last journal receivers to be used in removing the journal entries. The system will begin the backout procedure with the first journal receiver (specified by the first value) and will proceed through receivers until the last receiver (specified by the last value) is processed. (Note that the values specified on the parameter correspond to journal receivers in an inverse order from that in which they were attached to the journal.) If dual receivers were used at any time, the first of the receivers will always be used when chaining through the set of receivers. If any problem is encountered in the receiver chain (such as a damaged receiver or a receiver not online) before the backout of journal entries, the system will attempt to use the second of the dual receivers. If the second of the receivers is damaged or offline, or if the problem is encountered during the backout, the operation will terminate.

**CURRENT:* The currently attached receiver will be used as the only journal receiver with journal entries to be removed.

First Parameter Value

starting-receiver-name: Enter the qualified name of the journal receiver to be used as the first (newest) receiver with journal entries to be removed. (If no library qualifier is given, *LIBL is used to find the receiver.)

Second Parameter Value

ending-receiver-name: Enter the qualified name of the journal receiver to be used as the last (oldest) receiver with the journal entries to be removed. If the end of the receiver chain is reached before encountering a receiver of this name, the operation will terminate. (If no library qualifier is given, *LIBL is used to find the receiver.)

Note: The maximum number of receivers within the range of receivers is 256. If this maximum is exceeded, an exception will be signaled and no changes will be removed.

FROMENT Parameter: Specifies the entry to be used as the starting point for removing changes that have been journaled.

*LAST: Specifies that journal entries are to be removed starting with the last journal entry in the receiver range supplied. If FROMENT is not specified, *LAST is assumed.

starting-sequence-number: Specifies the sequence number of the first journal entry that is to be processed when backing out journal changes from the file (member).

TOENT Parameter: Specifies the entry to be used as the ending point for removing changes that have been journaled.

*FIRST: Specifies that journal entries are to be removed until the first entry in the receiver range supplied has been processed.

ending-sequence-number: Specifies the sequence number of the last journal entry that is to be removed from the file (member).

TOJOBO Parameter: Specifies that the journal entries are to be removed only until the indicated job (fully qualified job name) first opens any physical file member (or logical file member defined over the physical member) in the list of members specified on the FILE parameter that are to have their journal entries removed. (This will be the ending point for all members specified.)

Example

```
RMVJRNCHG  JRN(JRNA)  FILE((PAYROLL.LIB2 JAN)) +
     RCVRNG(RCV25 RCV22)  TOENT(*FIRST)
```

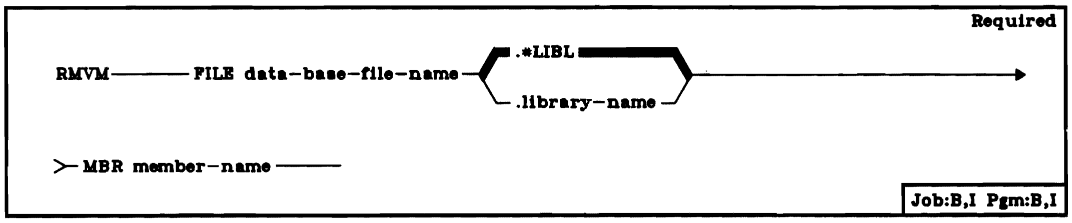
This command will cause the system to backout all changes recorded in journal JRNA to member JAN of file PAYROLL in library LIB2 that are recorded on the journal receiver chain starting with receiver RCV25 and ending with receiver RCV22. (The library search list *LIBL is used to find the journal JRNA and the receivers RCV25 and RCV22).

The backout will begin with the last recorded change on the receiver chain and will end with the first.

RMVM (Remove Member) Command

The Remove Member (RMVM) command removes the specified member from the specified physical or logical file. Removing a physical file member deletes all the data within the member and frees the storage space allocated to that member and its access path. Removing a logical file member deletes its access path to the associated data stored in a physical file member.

Restrictions: If a member of another file is sharing the access path or the data of the member being deleted, the dependent member must be removed first. To remove a member from a file, you must have object existence rights for the file.



FILE Parameter: Specifies the qualified name of the data base file (physical or logical) that contains the member to be deleted. (If no library qualifier is given, *LIBL is used to find the file.)

MBR Parameter: Specifies the name of the physical or logical file member that is to be deleted.

Example

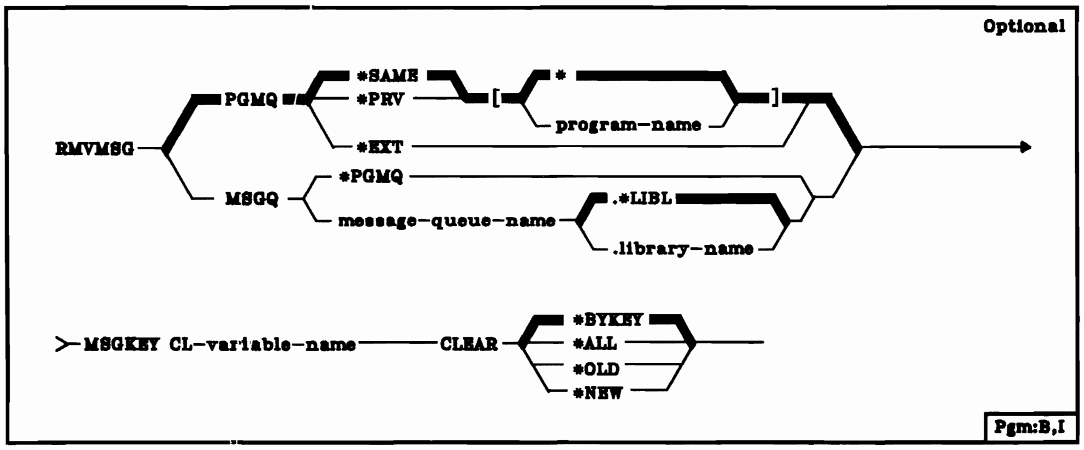
```
RMVM  FILE(JOBHIST1)  MBR(JOBHIST1A)
```

This command deletes the member JOBHIST1A from the file named JOBHIST1. The library list (*LIBL) is used to find the file and member. If JOBHIST1 contains other members, they remain unchanged.

RMVMSG (Remove Message) Command

The Remove Message (RMVMSG) command is used by a program to remove the specified message, or a group of messages, from the specified message queue. If the specified message queue is not allocated to the job in which this command is entered, it is implicitly allocated by this command for the duration of the command.

Restriction: To remove a message from the message queue, you must have operational rights for the queue and delete rights for both the queue and the library in which the queue is stored.



PGMQ Parameter: Specifies the program message queue from which the message is to be removed. (The name of each program message queue is the same as its corresponding program invocation.) If values are specified for the PGMQ parameter, the MSGQ parameter cannot be specified.

A list of two values is used to specify the program message queue from which the message is to be removed. The first value specifies the relationship of the sending program invocation with respect to the program invocation specified by the second value.

One of the following values can be specified for the *first* value:

<u>*SAME:</u> The message is to be removed from the program message queue whose name is specified as the *second* value in this parameter; otherwise, if the value * is specified or assumed, the message is to be removed from the program message queue of the program in which this RMVMSG command is used.

PRV: The message is to be removed from the program message queue of the previous program—that is, the calling program.

One of the following values can be specified for the *second* value:

***: The message is to be removed from this program's program message queue (the program in which the RMVMSG command is used).

program-name: The message is to be removed from the message queue of the lowest invocation of the specified program. (The name of the program message queue is the same as the name of the program.)

**EXT:* The message is to be removed from the external message queue of the job.

MSGQ Parameter: Specifies the name of the message queue from which one or more messages are to be removed. If MSGQ is specified, the PGMQ parameter cannot be specified.

**PGMQ:* The program message queue specified in the PGMQ parameter is the only queue from which the messages are to be removed.

qualified-message-queue-name: Enter the qualified name of the message queue from which one or more messages are to be removed. (If no library qualifier is given, *LIBL is used to find the message queue.)

MSGKEY Parameter: Specifies the name of the CL variable that contains the message reference key of the message to be removed. MSGKEY cannot be specified if *ALL, *OLD, or *NEW is specified for the CLEAR parameter.

CLEAR Parameter: Specifies whether one message, all messages, or only old or new messages in the specified message queue are to be removed from the queue.

*BYKEY: The message identified by the CL variable named in the MSGKEY parameter is to be removed from the message queue.

**ALL:* All messages are to be removed from the specified message queue.

**OLD:* All old messages in the specified message queue are to be removed from the queue.

**NEW:* All new messages in the specified message queue are to be removed from the queue.

Example

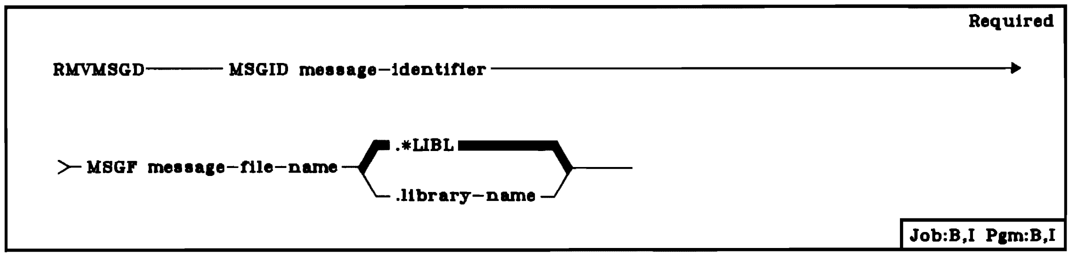
```
RMVMSG  MSGQ(SMITH)  MSGKEY(&KEY)
```

This command removes from the message queue named SMITH the message with the reference key specified in the CL variable &KEY.

RMVMSGD (Remove Message Description) Command

The Remove Message Description (RMVMSGD) command removes a message description from the specified message file.

Restriction: To remove a message description from the message file, you must have delete and operational rights for the message file and operational rights for the library in which the file is stored.



MSGID Parameter: Specifies the message identifier of the message to be removed from the message file.

MSGF Parameter: Specifies the qualified name of the message file containing the message to be removed. (If no library qualifier is given, *LIBL is used to find the message file.) Any message file overrides in effect for the job are ignored by this command; the file specified here is the one from which the message is removed.

Example

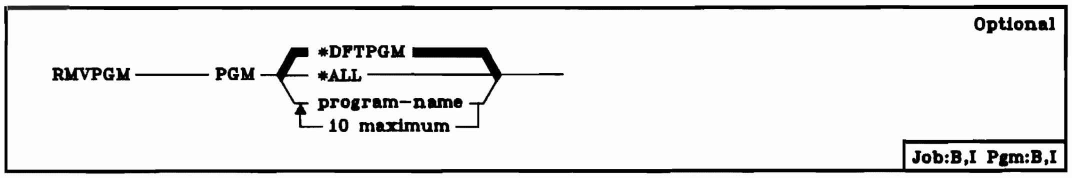
```
RMVMSGD  MSGID(UIN0115)  MSGF(INV)
```

This command removes the message description with the identifier UIN0115 from the message file named INV. The library list is used to find the INV file. Note that if more than one INV message file exists in the libraries searched, the message description could be removed from the wrong file.

RMVPGM (Remove Program) Command

The Remove Program (RMVPGM) command removes one or more programs from the debugging session currently in progress. All breakpoints and traces defined in each program are removed, and the programs are returned to their normal state. If a program is added again, breakpoints and traces must be respecified.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command*.



PGM Parameter: Specifies which programs are to be removed from the current debugging environment.

<u>*DFTPGM:</u> The program currently specified as the default program in the debugging environment is the program to be removed. The debugging session no longer has a default program unless one is specified later.

**ALL:* All the programs that are currently debuggable are to be removed.

program-name: Enter the simple names of one or more programs that are to be removed from the current debugging session.

Example

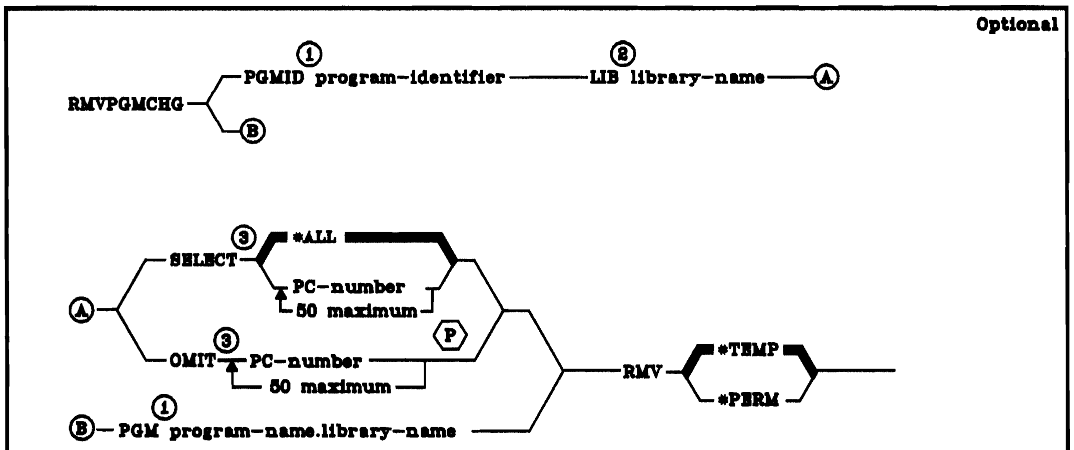
```
RMVPGM  PGM(PGMX  PGMY  PGMZ)
```

This command removes the three programs PGMX, PGMY, and PGMZ from the current debugging session. All breakpoints and traces in the programs are removed.

RMVPGMCHG (Remove Programming Change) Command

The Remove Programming Change (RMVPGMCHG) command removes the specified programming changes (PCs) or program patches from the specified library. If the programming changes or program patches were temporarily applied, the original objects that they had replaced are put back into the library. The PCs and program patches can be temporarily removed, in which case they are retained in the system for reapplication later, if desired.

The RMVPGMCHG command can be used to remove only immediate PCs, not deferred PCs. Deferred PCs must be removed through the deferred programming changes display. This display is explained in the *System/38 Operator's Guide*.



① PGMID and PGM are mutually exclusive. One or the other must be specified. To code the following parameters *positionally*, you must code them in this order, using *N for those not being specified: PGMID, LIB, SELECT, and OMIT.
② If PGMID is specified, LIB is required.
③ SELECT or OMIT is valid only if PGMID is specified.

Job:B,I Pgm:B,I

PGMID Parameter: Specifies the identifier of the program from which PCs are to be removed. The PGM parameter cannot be specified if PGMID is specified. If PGMID is not specified, PGM must be specified.

LIB Parameter: Specifies the name of the library that contains the program specified by the PGMID parameter. If PGMID is specified, LIB must be specified.

SELECT Parameter: Specifies which of the previously applied PCs are to be removed from the specified program. The OMIT parameter cannot be specified if SELECT is specified.

*ALL: All the PCs that were temporarily applied are to be removed from the program. Those that were permanently applied are ignored by this command. If all PCs cannot be removed for some reason, messages indicating the PCs not removed and the reason(s) for not being removed are sent to the operator.

PC-number: Enter the PC identification numbers of the individual programming changes that are to be removed. A maximum of 50 PC numbers can be specified.

OMIT Parameter: Specifies that all temporarily applied programming changes are to be removed except for those specified in this parameter. Enter the PC numbers of the programming changes that are to be omitted (left in the system) when all the rest are removed. A maximum of 50 PC numbers can be specified. The OMIT parameter cannot be specified if individual PC numbers are specified in the SELECT parameter.

PGM Parameter: Specifies the qualified name of the program from which a program patch is to be removed. This parameter is valid only for removing program patches. It cannot be specified if PGMID is specified. If PGM is not specified, PGMID must be specified.

RMV Parameter: Specifies whether the PCs are to be removed temporarily or permanently. Permanently removed PCs are deleted from the system; temporarily removed PCs are retained in the library for later reapplication.

*TEMP: The PCs are to be removed and retained in the library so that they can be reapplied at a later time, if desired.

PERM: The PCs are to be permanently removed and deleted from the library.

Examples

```
RMVPGMCHG  PGMID(5714SS1)  LIB(QSYS)
```

This command temporarily removes all temporarily applied PCs from CPF (program identifier 5714SS1) in the library QSYS. The PCs can be reapplied later, if necessary, through the APYPGMCHG command.

```
RMVPGMCHG  PGMID(5714RG1)  LIB(QRPG) +
     SELECT(00002 00005)  RMV(*PERM)
```

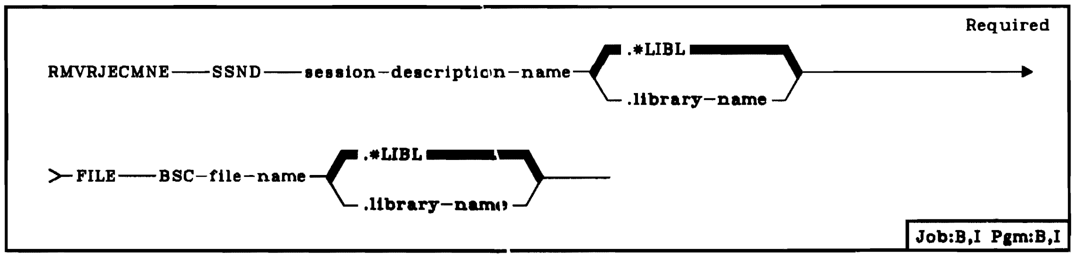
This command permanently removes two PCs (numbers 00002 and 00005) from the RPG compiler (program identifier 5714RG1) in the library QRPG. The two PCs are deleted from the system.

RMVRJECMNE (Remove RJE Communications Entry) Command

The Remove RJE Communications Entry (RMVRJECMNE) command removes a communications entry from the specified inactive session description.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Remove RJE Communications Entry (RMVRJECMNE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description from which the communications file entry is to be removed. If no library qualifier is given, *LIBL is used to find the session description.

FILE Parameter: Specifies the qualified name of the BSC entry that is to be removed. If no library qualifier is given, *LIBL is used to find the file. If the entry was added using *LIBL and you attempt to remove it using the qualified name of the file, a message ENTRY NOT FOUND is issued.

Example

```
RMVRJECMNE  SSND(RJE.USERLIB) +
      FILE(DEVPRT1.USERLIB)
```

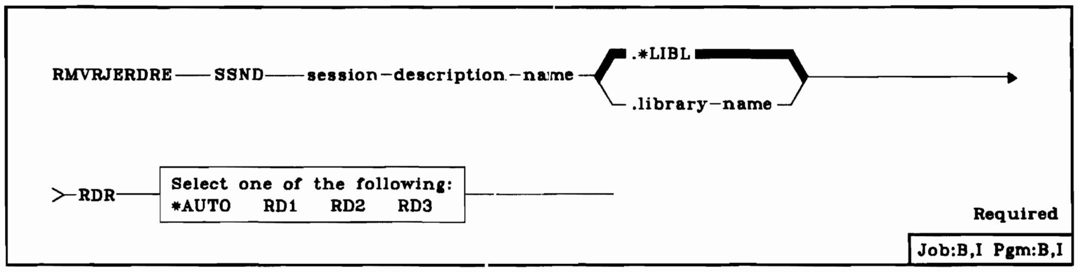
This command removes the communications entry named DEVPRT1.USERLIB from session description named RJE in library USERLIB.

RMVRJERDRE (Remove RJE Reader Entry) Command

The Remove RJE Reader Entry (RMVRJERDRE) command removes an RJEF reader entry from the specified inactive session description.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Remove RJE Reader Entry (RMVRJERDRE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description from which the RJEF reader entry is to be removed. If no library qualifier is given, *LIBL is used to find the session description.

RDR Parameter: Identifies the RJEF reader entry to be removed.

**AUTO:* The *AUTO RJEF reader entry is to be removed.

RD1: RJEF Reader 1 is to be removed.

RD2: RJEF Reader 2 is to be removed.

RD3: RJEF Reader 3 is to be removed.

Example

```
RMVRJERDRE  SSND(RJE.USERLIB) +
      RDR(RD1)
```

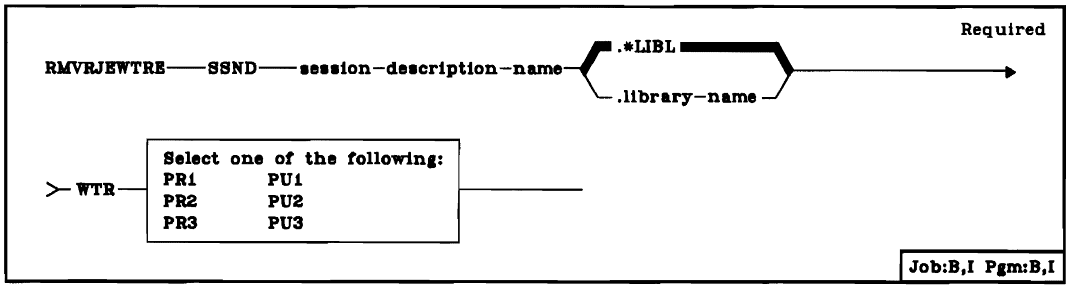
This command removes the reader entry named RD1 from session description named RJE in library USERLIB.

RMVRJEWTRE (Remove RJE Writer Entry) Command

The Remove RJE Writer Entry (RMVRJEWTRE) command removes an RJEF writer entry from the specified inactive session description.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Remove RJE Writer Entry (RMVRJEWTRE) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description from which the RJEF writer entry is to be removed. If no library qualifier is given, *LIBL is used to find the session description.

WTR Parameter: Identifies the RJEF writer entry to be removed.

PR1: RJEF Printer 1 entry is to be removed.

PR2: RJEF Printer 2 entry is to be removed.

PR3: RJEF Printer 3 entry is to be removed.

PU1: RJEF Punch 1 entry is to be removed.

PU2: RJEF Punch 2 entry is to be removed.

PU3: RJEF Punch 3 entry is to be removed.

Example

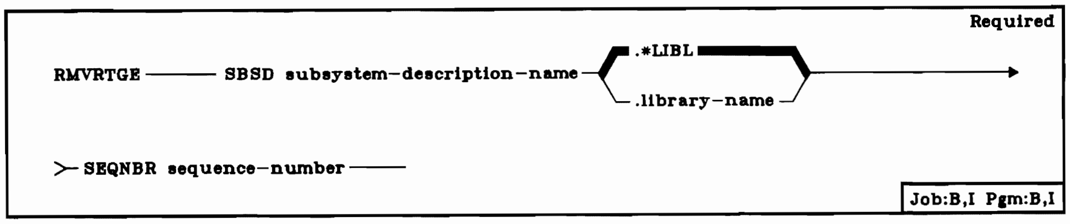
```
RMVRJEWTRE  SSND(RJE.USERLIB) +
     WTR(PR1)
```

This command removes writer entry PR1 from session description named RJE in library USERLIB.

RMVRTGE (Remove Routing Entry) Command

The Remove Routing Entry (RMVRTGE) command removes a routing entry from the specified subsystem description (which must be inactive at the time).

Restriction: To use this command, you must have operational and object management rights for the specified subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description that contains the routing entry to be removed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

SEQNBR Parameter: Specifies the sequence number of the routing entry to be removed.

Example

```
RMVRTGE  SBSD(PERT.OR)  SEQNBR(9912)
```

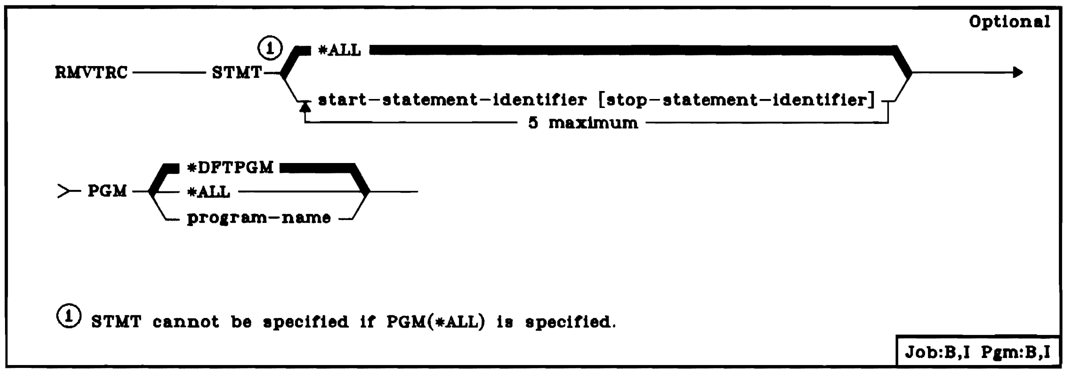
This command removes the routing entry 9912 from subsystem description PERT in library OR.

RMVTRC (Remove Trace) Command

The Remove Trace (RMVTRC) command removes all or part of the traces previously specified in one or more ADDTRC commands for use in debugging the program(s). Any trace data already generated by the traces being removed is not affected by this command. (This data can be removed by the CLRTRCDTA command.) The tracing limits specified in the Change Debug Mode (CHGDBG) command are not changed.

On the RMVTRC command, the user specifies the HLL statement identifiers or the System/38 instruction numbers that correspond to the ranges that he no longer wishes to be traced. To remove a trace, exactly the same range (as specified on the ADDTRC command) must be specified. A maximum of five sets of trace ranges can be specified in one command.

Restriction: This command is valid only in debug mode. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*


Optional
RMVTRC
STMT
①
*ALL
start-statement-identifier [stop-statement-identifier]
5 maximum
PGM
*DFTPGM
*ALL
program-name
① STMT cannot be specified if PGM(*ALL) is specified.
Job:B,I Pgm:B,I

STMT Parameter: Specifies the statement identifiers (or the System/38 instruction numbers) of the trace statements that are to be no longer traced. To remove a trace from a program, the same statement identifiers must be specified here that were specified on the ADDTRC command.

<u>*ALL</u>: All HLL statements and/or System/38 instructions in the specified program(s) are to be no longer traced regardless of how the trace was defined by the ADDTRC command(s).

start-statement-identifier [stop-statement-identifier]: Enter the HLL statement identifier (or the System/38 instruction number) of the first trace statement to be removed and, optionally, the identifier of the last statement to be removed from future tracing. However, if the last statement was specified on the ADDTRC command, the last statement *must* also be specified here. As many as five trace ranges can be specified in the program for each use of this command.

The method used to specify the trace statements on the ADDTRC command (that is, HLL statement identifiers versus System/38 instruction numbers) must also be used here to remove them. If only a starting statement identifier is specified for a range, the single statement specified is the only statement removed for that range.

PGM Parameter: Specifies which program (or all programs) contains the trace statements to be removed from future tracing operations.

*DFTPGM: The program previously specified as the default program contains the statements to be removed from tracing.

**ALL:* All programs that currently have trace ranges in them are to have *all* of their trace ranges removed; no tracing can be done in any of the programs in the debugging session unless more traces are added by the ADDTRC command. *ALL is not valid unless the STMT parameter is omitted.

program-name: Enter the name of the program that is to have the specified trace statements (or all trace statements) removed.

Example

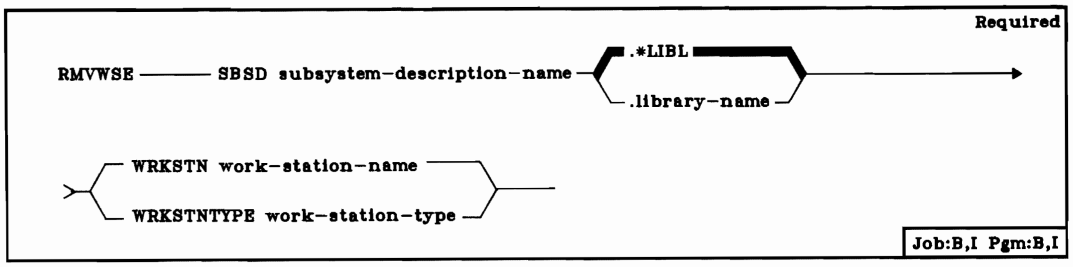
```
RMVTRC
```

This command removes all the trace statements used for tracing in the program currently specified as the default program.

RMVWSE (Remove Work Station Entry) Command

The Remove Work Station Entry (RMVWSE) command removes a work station entry from the specified subsystem description; the associated subsystem must be inactive at the time.

Restriction: To use this command, you must have operational and object management rights for the specified subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description containing the work station job entry to be removed. (If no library qualifier is given, *LIBL is used to find the subsystem description.)

WRKSTN Parameter: Specifies the name of the work station for which the job entry is to be removed. Either WRKSTN or WRKSTNTYPE must be specified.

WRKSTNTYPE Parameter: Specifies the work station type for which the job entry is to be removed. Enter one of the following: CONS for the system console display, 5251 for the 5251 Display Station, or 5252 for the 5252 Dual Display Station.

Example

```
RMVWSE  SBSD(CHARLES.LIB2)  WRKSTN(B53)
```

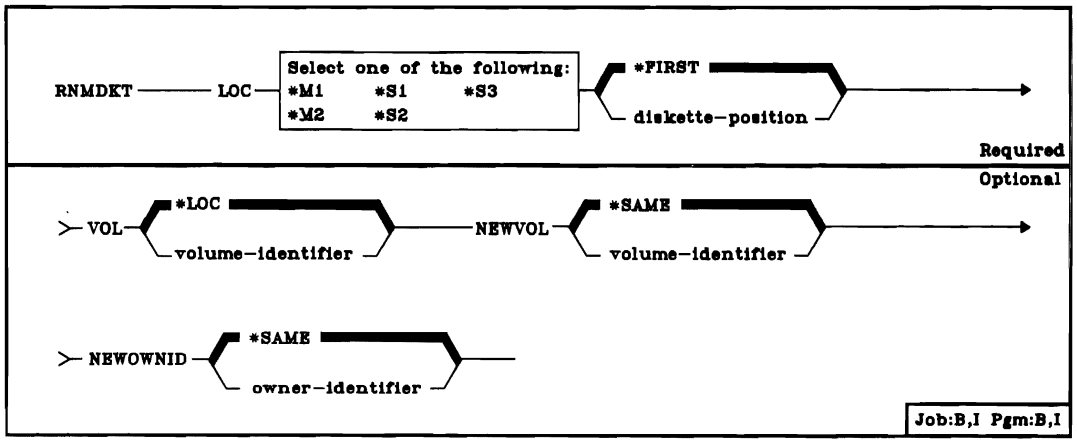
This command removes the work entry for work station B53 from the subsystem description named CHARLES in library LIB2.

RNMDKT (Rename Diskette) Command

The Rename Diskette (RNMDKT) command changes the name of a single diskette or changes the name (or identifier) of its owner. This command can be used to change the contents of the volume identifier field, or the owner identifier field, or both fields.

Diskettes that are in the save/restore format and that contain saved data should not be renamed. If the saved data is no longer needed, the diskettes can be renamed and reinitialized by the INZDKT command. However, if a magazine is to be renamed, refer to the description of the NEWVOL parameter for the requirements.

Note: When processing a diskette with non-IBM standard labels, you may have unpredictable results. To initialize the diskette, execute the Initialize Diskette (INZDKT) command with CHECK(*NO) specified.



LOC Parameter: Specifies the location in the magazine or slot that contains the diskette to be renamed. For magazines, two values must be specified: one for the magazine and one for the first diskette used in the magazine. (Either *FIRST or a diskette position can be specified for the second value.) Only one value is needed to identify the manual slot used. Enter one of the following values to specify which magazine or slot is to be used.

Value	Location Used
*M1	Magazine 1 only
*M2	Magazine 2 only
*S1	Manual slot 1 only
*S2	Manual slot 2 only
*S3	Manual slot 3 only

*FIRST: The first diskette in the specified magazine is the diskette to be renamed. If *FIRST is used and the first value is:

*M1	Rename diskette 1 of magazine 1.
*M2	Rename diskette 1 of magazine 2.

diskette-position: Enter the number of the diskette position (1 through 10) in the specified magazine that contains the only diskette to be renamed. (A value is not valid for manual slots.)

VOL Parameter: Specifies whether a check of the volume identifier field on the diskette should be made before the specified diskette is renamed. If so, the current volume identifier of the volume to be checked must be specified. (For an expanded description of the VOL parameter, see Appendix A.)

*LOC: No volume identifier check is to be made; the diskette currently mounted in the location specified by the LOC parameter is to be renamed without a check.

volume-identifier: Enter a volume identifier that is to be compared with the diskette label volume identifier field on the diskette that is to be renamed. The identifier can have no more than 6 characters; any combination of letters and digits can be used. If a magazine in the save/restore format is to have one or more of its diskettes renamed, each diskette to be renamed must have its full identifier specified (the magazine identifier and the diskette's position within the magazine). If more than one is to be renamed, one RNMDKT command must be used for each diskette, with the same magazine identifier and a different diskette position being specified.

If the volume identifiers do not match, a message is issued to the system operator. The operator can then either insert the correct diskette and try again or terminate the command.

NEWVOL Parameter: Specifies, if the diskette is to be renamed, the new volume identifier of the diskette.

*SAME: The volume identifier is not to be changed.

volume-identifier: Enter no more than 6 characters that are to be the new volume identifier of the diskette(s) being renamed. Any combination of letters and digits can be used. If a magazine in the save/restore format is to be renamed and it contains saved data, *each* diskette in the magazine to be renamed must have its new, full volume identifier specified (that is, 10 volume identifiers must be given to rename all 10 diskettes). Each identifier must specify the same (new) *magazine* identifier and the *current* diskette position in the magazine; for the magazine identifier, a maximum of 5 characters can be specified.

NEWOWNID Parameter: Specifies the owner identification to be written in the volume label. The owner identification contains a maximum of 14 characters (uppercase letters and/or digits in any combination) and is left-justified and padded with blanks on the right if fewer than 14 characters are supplied.

*SAME: The owner identification is not to be changed.

owner-identifier: Enter no more than 14 uppercase letters and digits that identify the owner of the diskette. Even if enclosed in apostrophes, no lowercase letters, embedded blanks, or special characters can be entered. If fewer than 14 characters are entered, the field is left-justified and padded on the right with blanks.

Example

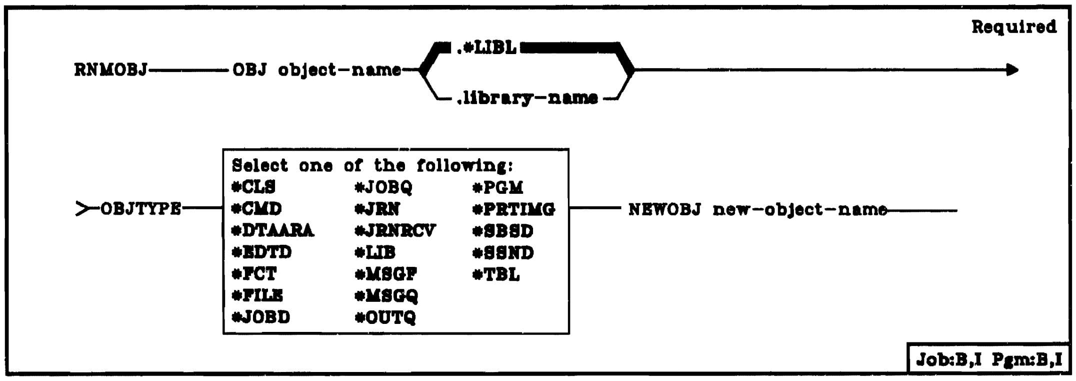
```
RNMDKT  LOC(*S1) VOL(MASTER) NEWVOL(BACKUP)
```

This command changes the name of the diskette in slot 1, if its name is MASTER, to BACKUP. The owner identification is unchanged (NEWOWNID assumed to be *SAME).

RNMOBJ (Rename Object) Command

The Rename Object (RNMOBJ) command changes the name of an object in a library. The new name specified for the object must be unique in the library for the object type. If the object being renamed is in use when the command is entered, the command is not executed and a message is sent to the user who issued the command.

Restrictions: (1) The user must have object management rights for the object that is being renamed; he must also have update rights for the library in which the object is located. (2) User profiles, edit descriptions, line descriptions, device descriptions, control unit descriptions, and journals and journal receivers cannot be renamed. (3) The following objects cannot be renamed: the job's temporary library QTEMP, the system operator message queue QSYSOPR, all work station user message queues, and the system logs QHST, QSRV, and QCHG.



OBJ Parameter: Specifies the current qualified name of the object that is to be renamed. (If no library qualifier is given, *LIBL is used to find the object.) The object name should be qualified to ensure that the right object is renamed.

OBJTYPE Parameter: Specifies the type of the object that is to be renamed. Enter one of the following CPF object types:

Value	Type of CPF Object
*CLS	Class
*CMD	Command
*DTAARA	Data area
*EDTD	Edit description
*FCT	Forms control table
*FILE	File
*JOBD	Job description
*JOBQ	Job queue
*JRN	Journal
*JRNRCV	Journal receiver
*LIB	Library
*MSGF	Message file
*MSGQ	Message queue
*OUTQ	Output queue
*PGM	Program
*PRTIMG	Print image
*SBSD	Subsystem description
*SSND	Session description
*TBL	Table

NEWOBJ Parameter: Specifies the new name of the object being renamed. The object remains in the same library. Enter the new name that identifies the object to the system.

Examples

```
RNMOBJ  OBJ(FILEX.PAYROLL) OBJTYPE(*FILE) NEWOBJ(MSTR)
```

The library named PAYROLL is searched for the file named FILEX. If the file is found and if the user has object management authority for FILEX and update authority for the PAYROLL library, FILEX is renamed MSTR.

```
RNMOBJ  OBJ(PAY.PAYROLL) OBJTYPE(*PGM) NEWOBJ(PAY27)
```

The program PAY in library PAYROLL is renamed PAY27.

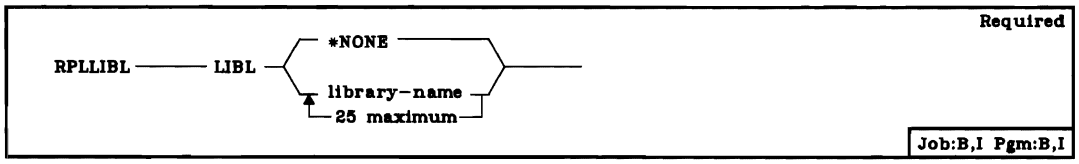
```
RNMOBJ  OBJ(PRINTER.QGPL) OBJTYPE(*OUTQ) +
      NEWOBJ(PRINTER1)
```

The object PRINTER, which is an output queue, is renamed PRINTER1.

RPLLIBL (Replace Library List) Command

The Replace Library List (RPLLIBL) command replaces the user's portion of the current job's library list with the list of libraries specified by the user. This command does not affect the system portion of the library list, nor does it affect any other job's library list.

Restrictions: The user who submits this command must have operational rights for all the specified libraries before the job's current library list is replaced. If the user is not authorized for one of the libraries, the command is not executed.



LIBL Parameter: Specifies the libraries that are to be placed in the user's portion of the current job's library list.

**NONE:* No libraries are to be in the user's portion of the job's library list.

library-name: Enter the names of the libraries in the order in which they are to be searched.

Example

```
RPLLIBL  LIBL(ULIB10 ULIB15 QGPL)
```

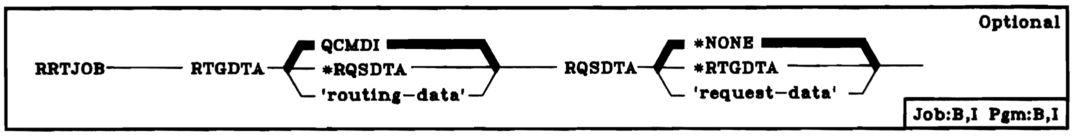
This command replaces the user's portion of the current job's library list that existed before this command was entered. The new library list contains libraries ULIB10, ULIB15, and QGPL. They will be searched in that order.

RRTJOB (Reroute Job) Command

The Reroute Job (RRTJOB) command causes a new routing step to be initiated for a job within the current subsystem. For example, the job may need to be rerouted so that it will execute under a different class or in a different storage pool. The rerouting requires changes in the routing data for the job; the new routing data invokes a different program that is used with the new routing step.

When this command is used, any objects that were allocated in the previous routing step are deallocated and any open files are closed. If the objects or files are needed in the new routing step, they must be allocated or opened again.

Note: The execution of this command within a batch job causes spooled inline files to be lost, because they cannot be accessed in the new routing step. Also, if the RRTJOB command is executed while the system is terminating (by the execution of a TRMSBS, TRMCPF, or PWRDWNSYS command), a new routing step will not be initiated and end-of-job processing will occur.



RTGDTA Parameter: Specifies the routing data that is to be used to initiate the next routing step. The routing data is used to determine the routing entry that identifies the program that is to process the routing step.

<u>QCMDI:</u> This routing data matches a routing entry in the IBM-supplied subsystem description (QINTER), which indicates a routing step that is processed by the IBM-supplied control language processor, QCL, in the QSYS library.

**RQSDTA:* The first 80 characters of the request data specified in the RQSDTA parameter of this command is also to be used as the routing data for the next routing step.

'routing-data': Enter the character string that is to be used as the routing data for initiating the next routing step. A maximum of 80 characters can be entered (enclosed in apostrophes if necessary).

RQSDTA Parameter: Specifies the request data that is to be added on the end of the job's message queue for use by the new routing step. For example, if RTGDTA (QCMDB) is specified, the IBM-supplied batch subsystem QBATCH is being used, and a CL command is supplied here, it becomes a message that is read by the control language processor, QCL (if the submitted job is routed to QCL).

*NONE: No request data is to be placed in the job's message queue.

**RTGDTA:* The routing data specified in the RTGDTA parameter is also to be placed at the end of the job's message queue.

'request-data': Enter the character string that is to be placed at the end of the job's message queue for use by the new routing step or some subsequent routing step in the job. A maximum of 256 characters can be entered (enclosed in apostrophes if necessary). When a CL command is entered, it must be enclosed in single apostrophes, and where apostrophes would normally be used *within* the command, double apostrophes must be used instead.

Example

```
RRTJOB  RTGDTA(INQUIRY)
```

This command reroutes the job in which the command is issued by initiating a new routing step with the routing data INQUIRY. The job remains in the same subsystem.

RSMBKP (Resume Breakpoint) Command

The Resume Breakpoint (RSMBKP) command causes the execution of a program to continue after it has been halted at a breakpoint. The program that continues is the one that most recently stopped at a breakpoint. When more than one program in the job is stopped at a breakpoint, the CNLRQS (Cancel Request) command can be used to return to the command entry display for a previous program invocation that is also stopped at a breakpoint.

(Another way to resume execution is to press the CF1 key, which cancels the request and returns you to the breakpoint display from which you came.)

Restriction: This command is valid only in debug mode and only when the program is stopped at a user-defined breakpoint. That is, this command is not valid at a breakpoint caused by an unmonitored message. To enter debug mode, refer to *ENTDBG (Enter Debug) Command.*

```
RSMBKP────────

                                                        Job:I Pgm:I
```

There are no parameters for this command.

Example

RSMBKP

Assuming that the program having control is stopped at a breakpoint, this command causes the execution of the program to continue, starting from the breakpoint location.

RSTAUT (Restore Authority) Command

The Restore Authority (RSTAUT) command restores the authority of object usage to user profiles when the system is being restored. This command restores the same object usage rights to specified objects in the user profile that each user profile had when all the profiles were saved by the SAVSYS (Save System) command. Authority cannot be restored to the user profiles until the profiles are first restored in the system by the RSTUSRPRF (Restore User Profiles) command and all the objects (for which authority is being given) have been restored to the same libraries by the RSTLIB or RSTOBJ commands.

This command is used only when the entire system is being restored; it must be used as the last step of the following sequence:

1. Start CPF (AIPL): Restores the QSYS library and ensures that the IBM-supplied user profiles are there.

2. RSTUSRPRF command: Restores all the saved user profiles to the system.

3. RSTLIB command: Restores all the user libraries (including the other IBM-supplied libraries). To restore them in one operation, the SAVLIB parameter must specify *NONSYS.

4. RSTAUT command: Restores the object usage rights to user profiles.

Restriction: To use this command, you must have the save system special rights in your user profile, specified by SPCAUT(*SAVSYS).

```
RSTAUT ———

                                                              Job:B,I Pgm:B,I
```

There are no parameters for this command.

Example

RSTAUT

This command restores to each user profile the authority to use each object that the profile had at the time when the system was saved. The user profiles and the libraries and their objects must be restored before the RSTAUT command is issued.

RSTLIB (Restore Library) Command

The Restore Library (RSTLIB) command restores to the system one or all user libraries that were saved on diskette or tape. Any library that has been saved by the SAVLIB command can be restored by this command. The command restores the entire library; this includes the library description, the object descriptions, and contents of the objects in the library. (It also includes status information for PCs that were in the library at the time the library was saved. Note that the restore of a library destroys the PC status information that currently exists in the library being restored to.)

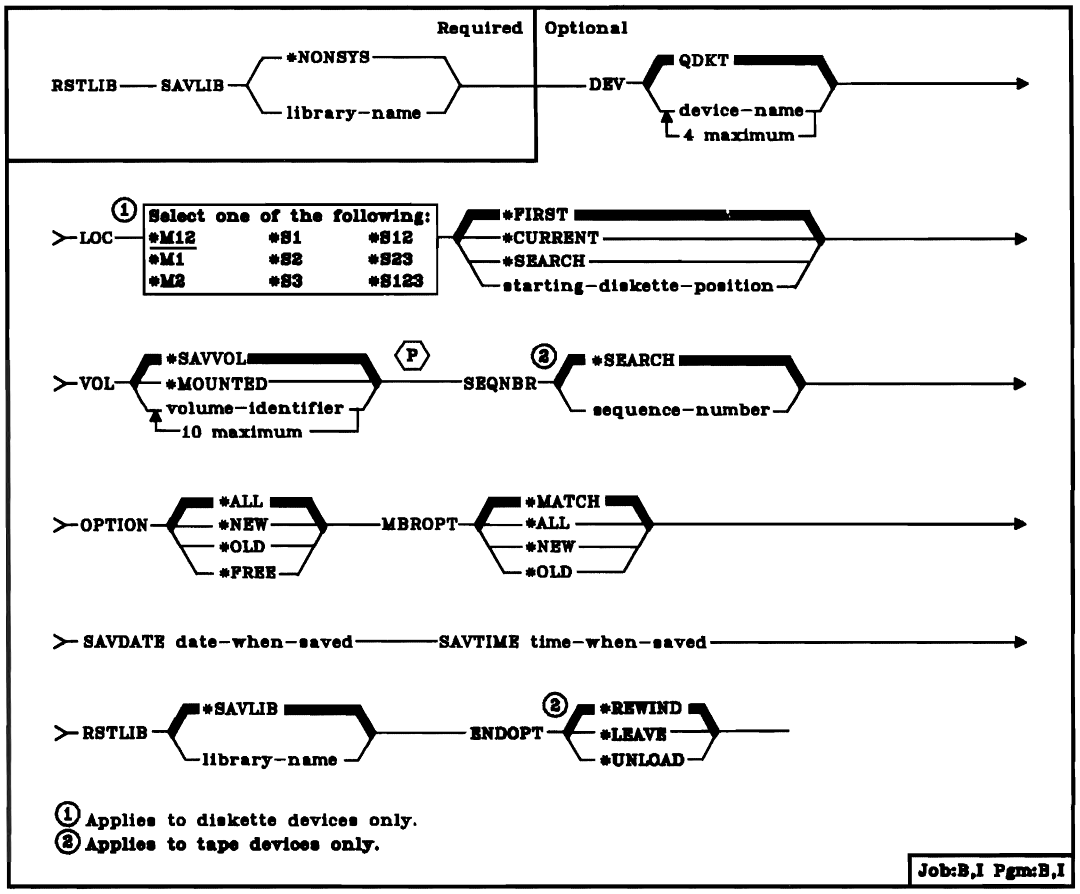
This command can be used to restore libraries whether the library storage was or was not freed by the Save Library (SAVLIB) command, or whether any library was deleted by the Delete Library (DLTLIB) command. If the data portions of the objects in the saved libraries were not freed, each library is copied into the same area of storage that it previously occupied. If the storage was freed, the system finds the needed storage to store the library contents (the object description and data portion of every file and program in the library). If the library is not known to the system because it has been deleted or the library is being restored on a different system, the system must find the storage to store everything that is in the library, including the library description.

The user profile of the security officer (QSECOFR) becomes the default owner of any object restored in the system whose owner is not known to the system.

If an object already exists in the library to which it is being restored, the public and private authorities of the existing object are retained. If the object does not exist in the library, all public authorities are restored, but any private authorities must be granted again.

Restrictions: To use this command, you must have either the special rights *SAVSYS specified in your user profile by the SPCAUT parameter, or you must have: (1) read and add rights for, or be the owner of, each library specified and (2) object existence rights for, or be the owner of, each object in the library. (If you do not have the proper rights for all of the libraries and objects specified, only those for which you do have the rights are restored.)

If a specific library is specified in the SAVLIB parameter, the current versions of programs on the system that are in that library should not be executing while the library is being restored. If any program is executing, it terminates abnormally.



SAVLIB Parameter: Specifies the name of the library, or all libraries, that are to be restored in the system. (You cannot specify *NONSYS if the RSTLIB parameter specifies anything other than *SAVLIB.)

**NONSYS:* All libraries saved by the SAVLIB (Save Library) command with LIB(*NONSYS) specified are to be restored. All other operations on the system must be stopped before this option is specified. This requires terminating all subsystems through the TRMSBS or TRMCPF command.

library-name: Enter the name of the library to be restored. The name of the library being restored must be the same as that used when the library was saved. The names QSYS, QSRV, QTEMP, QSPL, and QRECOVERY cannot be specified.

DEV Parameter: Specifies the name of the device to be used to restore the library. The device name must already be known on the system by a device description.

QDKT: The diskette device QDKT is to be the device used to restore the library.

device-name: Enter the name of the diskette or tape (if installed) device that is to be used to restore the library. If you are using more than one tape device (up to a maximum of four), enter the names of the devices in the order they are to be used. If you are using more than one tape volume, you may want to use more than one tape device, so that one tape volume can rewind/unload while another tape device is processing.

LOC Parameter: Specifies, only when diskettes are used, which diskette locations on the diskette magazine drive are to be used to restore the specified libraries. The data is restored from diskettes that are loaded either in the magazines or in the manual slots. If the restore operation requires that additional magazines or diskettes be loaded, a message is issued to the operator to mount them. (This parameter is ignored if a tape device is specified in the DEV parameter.)

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first value specifies which unit (magazines or slots) and location are to be used for the restore operation. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be used first in the restore operation. Enter one of the following values for the starting diskette position:

*FIRST: The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where a previous restore operation has just ended.) If the currently selected diskette is not in the location specified by the first value, a diagnostic message is sent to the system operator, and the command is terminated.

**SEARCH:* The diskettes in the location specified by the first value are to be searched for the library specified by SAVLIB (and for a specific version, if SAVDATE and SAVTIME are specified). The search begins with the leftmost diskette in the specified location and ends with the first diskette on which the specified library and version are found.

starting-diskette-position: If a multiple diskette location (such as *M1 or *S23) is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10 for magazines, or 1 through 3 for manual slots) that contains the first diskette to be used.

VOL Parameter: Specifies the volume identifiers of the tape volumes (reels) or diskette volumes (either in magazines or slots) from which the library is to be restored. The volumes must be mounted in the same order as they were when the library was saved. If the library is known to the system before the restore operation, the system can check whether the mounted volumes contain the current version of the library. (For an expanded description of the VOL parameter, see Appendix A.)

*SAVVOL: The system, using the save/restore history information, is to determine which tape or diskette volumes contain the most recently saved version of the library. If the device and location specified in the DEV and LOC parameters do not match the device and location of the most recently saved version of the library, an error message is sent to the user, and the function is terminated. If the wrong volume is mounted in the location specified by the command, a message, sent to the system operator, identifies the first volume that must be mounted before the library can be restored. If *SAVVOL is specified, the parameters SAVDATE and SAVTIME cannot be specified.

**MOUNTED:* The library is to be restored from the volumes that are currently mounted on the device specified by the DEV parameter. The version of the library that is restored is the first version found on the media, unless a specific version is identified by the SAVDATE and SAVTIME parameters, or for tape, the SEQNBR parameter.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used to restore the libraries. For tape volumes and single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the *magazine* identifier (5 characters maximum) can be specified for each volume. After the diskettes in the first magazine have been processed, the restore operation always continues with position 1 of the remaining magazines.

SEQNBR Parameter: Specifies, only when tape is used, which sequence number is to be used for the restore process.

*SEARCH: The mounted volume is searched for a data file with an identifier that matches the LABEL parameter value; when a match is found, the data file is restored. If the last operation on the device specifies ENDOPT(*LEAVE) (the tape is positioned at the location at which the last operation terminated), the file search begins with the first data file beyond the current tape position. If ENDOPT(*LEAVE) was not used for the last operation (or if the tape was manually rewound since an ENDOPT(*LEAVE) operation), the search begins with the first data file on the volume.

file-sequence-number: Enter the sequence number of the file to be used for the restore process.

OPTION Parameter: Specifies whether the objects in the saved library to be restored are known to the system and how each object in the library is to be handled.

<u>*ALL:</u> All the objects in the library saved on diskette (or tape) are restored to the library. Old objects on diskette (or tape) replace the current versions in the library in the system, and the objects not having a current version are added to the library in the system. Objects presently in the library remain there.

**NEW:* Only the objects in the library saved on diskette (or tape) that do not exist in the current version of the library in the system are to be added to the library. (Only objects not known to the library in the system are to be restored; known objects are not restored.) This option restores objects that were deleted since they were saved or that are new to this library. If any objects on diskette have a version already in the library in the system, they are not restored and an informational message is issued for each one, but the restore continues.

**OLD:* Only the objects in the library that have an old version on diskette (or tape) are to be restored; that is, the online version of each object is replaced by the version from diskettes. (Only objects known to the library are to be restored.) If any objects on the diskette are no longer part of the online version of the library, they are not added to the library and a warning message is issued for each one, but the restore continues.

**FREE:* The saved objects are to be restored only if they exist in the library in the system with their space freed. The saved version of each object is restored in the system in its previously freed space. (This option restores objects that had their space freed when they were saved.) If any objects on the diskette or tape are no longer part of the current version of the library or if the space is not free for any object, the object is not restored and a warning message is issued for each one. The restore continues, and all of the freed objects are properly restored.

MBROPT Parameter: Specifies, for data base files already known to the system, which members are to be restored. Unless MBROPT(*MATCH) is used, the member list in the saved file need not match, member for member, the current version in the system.

Note: Before restoring a file, the system checks to ensure that the file and member creation dates of the known system objects match the creation dates of the objects to be restored. If this check fails, the file is not restored.

<u>*MATCH:</u> The saved members are to be restored if the lists of the files in which they reside match, member for member, the lists of the current system version.

**ALL:* All members in the saved file are to be restored.

**NEW:* Only new members (members not known to the system) are to be restored.

**OLD:* Only members already known to the system are to be restored.

SAVDATE Parameter: Specifies the date on which the library was saved. If the current version is not to be restored, enter the date that specifies which version of the library is to be restored. (The date must be entered in the system date format specified by the system value QDATFMT; if separators, specified by the system value QDATSEP, are used, the value must be enclosed in apostrophes.) If a volume identifier is specified, but SAVDATE is not, the version of the library to be restored is the first version found on the volume. This parameter is valid only if a volume identifier or VOL(*MOUNTED) is also specified.

SAVTIME Parameter: Specifies the time that the library was saved if the current version is not to be restored. Enter the time as a six-digit value, in the format hours, minutes, and seconds (hhmmss), that specifies which version of the library is to be restored. If colons are used as separators, the value must be enclosed in apostrophes ('hh:mm:ss'). If a volume identifier is specified, but SAVTIME is not, the version of the library to be restored is the first version found on the volume.

This parameter is valid only if SAVDATE is also specified.

RSTLIB Parameter: Specifies whether the library contents are to be restored to the same library from which they were saved, or to a different one. (If a different library is specified, you cannot specify *NONSYS on the SAVLIB parameter.)

SAVLIB: The library contents are restored to the same library or libraries from which they were saved.

library-name: Enter the name of the library to which the saved library contents are to be restored. The names QSYS, QSRV, and QRECOVERY *can* be specified. If *NONSYS is specified on the SAVLIB parameter, a library name cannot be entered in this parameter.

ENDOPT Parameter: Specifies, only when tape is used, what operation is to be automatically performed on the tape volume after the restore operation ends. If more than one reel is involved, this parameter applies only to the last reel used.

REWIND: The tape is to be automatically rewound, but not unloaded, after the restore operation has ended.

**LEAVE:* The tape is not to be rewound; another restore operation can begin at the current position on the tape.

**UNLOAD:* The tape is to be automatically rewound and unloaded after the restore operation has ended.

Examples

```
RSTLIB  SAVLIB(JOE) OPTION(*NEW) VOL(*MOUNTED)
```

This command restores the saved version of library JOE from volumes that must be mounted on magazines 1 and 2. The only objects that are restored in the library are new objects (ones that were in the library when they were saved and then deleted later).

```
RSTLIB  SAVLIB(*NONSYS)
```

This command restores all the saved user-created libraries and QGPL to the system from diskettes read by the QDKT device. The contents of the libraries are restored as they were saved. The diskette volumes are mounted on magazines 1 and 2. New objects (on diskette) are added to the system; old objects in the system are overlaid by the version of the old objects on diskette.

```
RSTLIB  SAVLIB(PAYROLL) LOC(*M1) VOL(PAY) +
      SAVDATE(020180) SAVTIME(103214) RSTLIB(OLDPAY)
```

This command restores the version of the PAYROLL library from the device QDKT, which is on diskettes mounted in magazine 1, whose volume identifier is PAY. The version to be restored was saved at 10:32:14 on the date 02/01/80. All of the objects in the saved PAYROLL library are to be restored to the library OLDPAY.

```
RSTLIB  SAVLIB(QGPL) VOL(QGPL QGPL)
```

This command restores the QGPL library from two diskette magazines that are both named QGPL. Even though the volume identifiers are the same, they must both be specified. (If the magazines are mounted in the wrong order, an error message is sent to the system operator message queue.)

```
RSTLIB  SAVLIB(USRLIB) DEV(TAPE1 TAPE2 TAPE3) +
      VOL(USRA USRB USRC USRD) ENDOPT(*UNLOAD)
```

This command restores library USRLIB from four volumes on three tape devices. Volume USRA is to be mounted on tape device TAPE1, volume USRB on TAPE2, volume USRC on TAPE3, and volume USRD on TAPE1. The operator will rewind volume USRA and remove it from TAPE1, so that TAPE1 can be used by volume USRD.

RSTOBJ (Restore Object) Command

The Restore Object (RSTOBJ) command restores to the system a single object or a group of objects in a single library that were saved on diskette or tape by a single command. The types of objects that can be restored by this command are listed in the OBJTYPE parameter. They could have been saved either as separate objects or as part of the library save. The RSTOBJ command restores the object description and contents of each object specified in the command.

The command can be used to restore the objects regardless of whether the object storage was freed by the Save Object (SAVOBJ) command, or whether the objects were deleted by the associated delete command for that object type. If the storage was not freed, each object is restored in the same area of storage that it previously occupied. If the version of the object being restored is larger than the version in the system (for example, data records that have been deleted from the system still exist in the saved version of a file), the additional storage needed for the object is acquired. If the saved version of the object is smaller (for example, data records have been added to the system), the space that was acquired for the object remains assigned to that object and is available for use by the object.

If the storage was freed, the system finds the needed storage to store the contents of each file and program (only the data portion). If the objects are not known to the system because they have been deleted or they are being restored in a different system, the system must find the storage to store everything about each unknown object (its description and data portion).

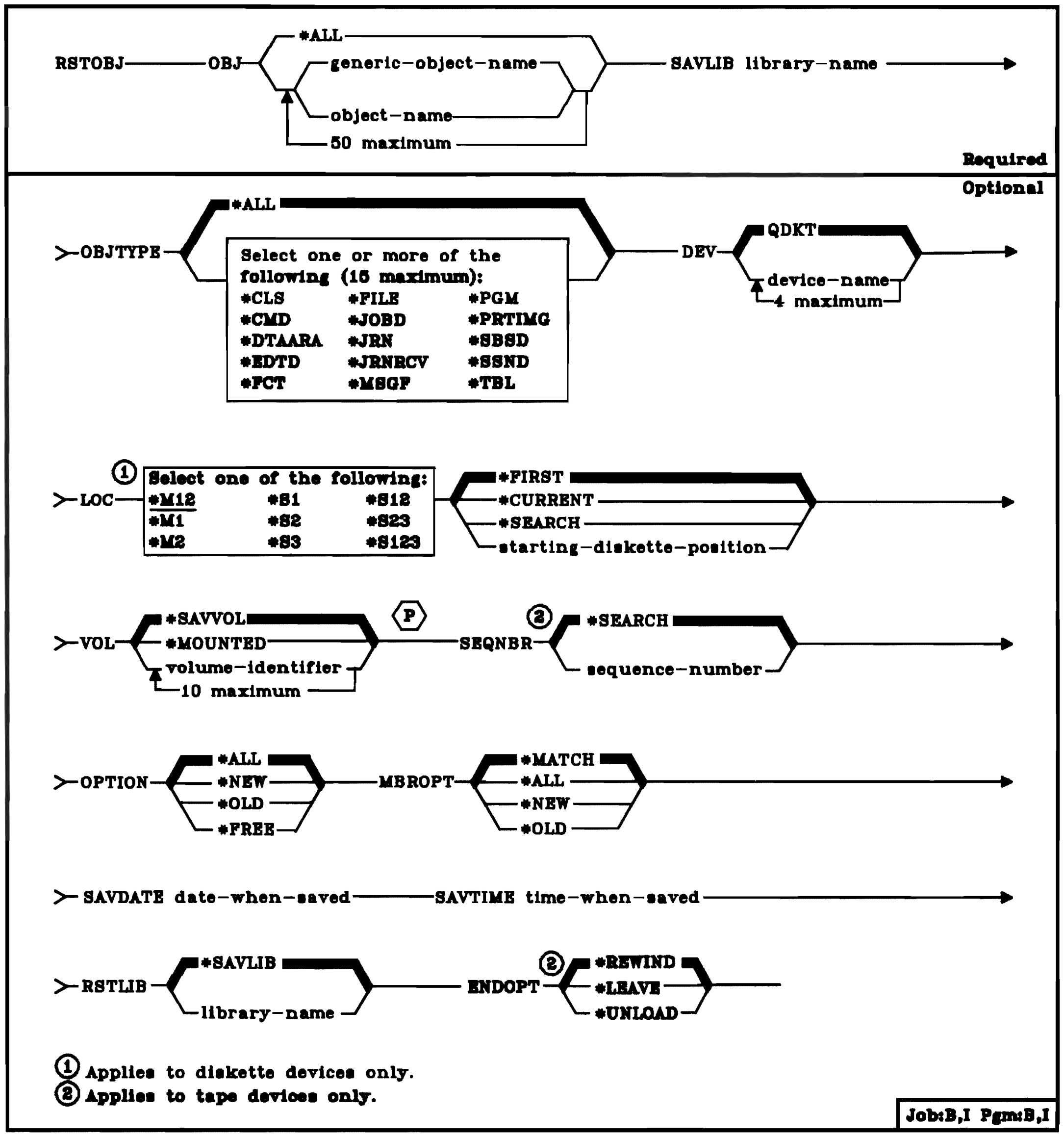
The user profile of the security officer (QSECOFR) becomes the default owner of any objects restored in the system whose owner is not known to the system.

If an object already exists in the library to which it is being restored, the public and private authorities of the existing object are retained. If the object does not exist in the library, all public authorities are restored, but any private authorities must be granted again.

Restrictions: To use this command, you must have either the special rights *SAVSYS specified in your user profile by the SPCAUT parameter, or you must have: (1) read and add rights for, or be the owner of, the specified library and (2) object existence rights for, or be the owner of, each object specified.

If this command is used to restore a program, the copy of that program that is currently in the system should not be executing while the program is being restored. If this occurs, the executing program is abnormally terminated.

Objects saved by separate commands must also be restored by separate commands. If a single command is used, some of the objects are not restored.



OBJ Parameter: Specifies the names of one or more objects that are to be restored. All the objects must be in the library specified in the SAVLIB parameter. Enter the name of each object or the generic name of each group of objects to be restored. (Add an asterisk after the last character in the generic name; for example, ABC*. If an * is not included with the name, the system assumes that the name is a complete object name.)

If the OBJTYPE parameter is not specified when the command is entered, all the object types listed in the description of the OBJTYPE parameter are restored, provided they are in the specified library and have the specified names.

**ALL:* All the objects in the specified library saved on diskette are to be restored, depending on the values specified for OBJTYPE and OPTION.

generic-object-name: Enter one or more generic names of groups of objects in the specified library that are to be restored.

object-name: Enter one or more names of specific objects that are to be restored. (Both generic names and specific names can be specified in the same command.)

SAVLIB Parameter: Specifies the name of the library that contained the objects when they were saved. If RSTLIB is not specified, this is also the name of the library to which the objects are restored. Enter the name of the library.

OBJTYPE Parameter: Specifies the types of CPF objects that are to be restored.

*ALL: All object types that are specified by name and are in the specified library saved on diskette are to be restored. If *ALL is also specified for the OBJ parameter, then all objects saved for that library are restored.

object-type: Enter the value for each of the types of objects that are to be restored. The following types can be specified.

Value	Type of CPF Object
*CLS	Class
*CMD	Command
*DTAARA	Data area
*EDTD	Edit description
*FCT	Forms control table
*FILE	File
*JOBD	Job description
*JRN	Journal
*JRNRCV	Journal receiver
*MSGF	Message file
*PGM	Program
*PRTIMG	Print image
*SBSD	Subsystem description
*SSND	Session description
*TBL	Table

The object types shown are also the ones that are saved and restored by the SAVLIB and the RSTLIB commands. Other object types can be saved only by the SAVSYS command and restored by the CPF install operation.

DEV Parameter: Specifies the name of the device to be used to restore the objects. The device name must already be known on the system by a device description.

QDKT: The diskette device QDKT is to be the device used to restore the objects.

device-name: Enter the name of the diskette or tape (if installed) device that is to be used to restore the objects. If you are using more than one tape device (up to a maximum of four), enter the names of the devices in the order they are to be used. If using more than one tape volume, you may want to use more than one tape device, so that one tape volume can rewind/unload while another tape device is processing.

LOC Parameter: Specifies, only when diskettes are used, which diskette locations on the diskette magazine drive are to be used to restore the specified objects. The data is restored from diskettes that are loaded either in the magazines or in the manual slots. If the restore operation requires that additional magazines or diskettes be loaded, a message is issued to the operator to mount them. (This parameter is ignored if a tape device is specified in the DEV parameter.)

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first value specifies which unit (magazines or slots) and location are to be used for the restore operation. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be used first in the restore operation. Enter one of the following values for the starting diskette position:

*FIRST: The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where a previous restore operation has just ended.) If the currently selected diskette is not anywhere in the location specified by the first value, a diagnostic message is sent to the system operator, and the command is terminated.

**SEARCH:* The diskettes in the location specified by the first value are to be searched for the objects specified by OBJ and SAVLIB (and for a specific version, if SAVDATE and SAVTIME are specified). The search begins with the leftmost diskette in the specified location and ends with the first diskette on which one of the objects is found.

starting-diskette-position: If a multiple diskette location (such as *M1 or *S23) is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10 for magazines, or 1 through 3 for manual slots) that contains the first diskette to be used.

VOL Parameter: Specifies the volume identifiers of the tape volumes (reels) or diskette volumes (either in magazines or slots) from which the objects are to be restored. The volumes must be mounted in the same order as they were when the objects were saved. If the objects are known to the system before the restore operation, the system can check whether the mounted volumes contain the current version of each object. (For an expanded description of the VOL parameter, see Appendix A.)

*SAVVOL: The system, using the save/restore history information, is to determine which tape or diskette volumes contain the most recently saved version of the objects. When *SAVVOL is specified or assumed, the following operational characteristics and restrictions apply:

- If the characteristics of the device and location specified in the DEV and LOC parameters do not match the device and location of the most recently saved version of the library, an error message is sent to the *user*, and the function is terminated.
- If the wrong volume is mounted in the location specified by the command, a message is sent to the *system operator* that identifies the first volume that must be mounted before the objects can be restored.
- When multiple objects are to be restored, they are considered in the order in which they would appear in a display produced by the DSPLIB command. (That is, the object names and types specified in the RSTOBJ command are used to determine which file of saved objects is to be used in the restore operation. One file is produced for each SAVLIB or SAVOBJ command executed.) The file chosen is the one in which the first considered object was last saved. Each object considered that either was not saved in the file chosen to be processed, or that has been more recently saved, is not restored; for each object not restored, an error message is sent to the user.
- If *SAVVOL is specified, the parameters SAVDATE, SAVTIME, and OPTION(*NEW) cannot be specified.

**MOUNTED:* The objects are to be restored from the volumes that are currently mounted on the device specified by the DEV parameter. The version of the objects that is restored is the first version found in the specified location, unless a specific version is identified by the SAVDATE and SAVTIME parameters.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used to restore the objects. For tape volumes and single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the *magazine* identifier (5 characters maximum) can be specified for each volume. After the diskettes in the first magazine have been processed, the restore operation always continues with position 1 of the remaining magazines.

SEQNBR Parameter: Specifies, only when tape is used, which sequence number is to be used for the restore process.

*SEARCH: The mounted volume is searched for a data file with an identifier that matches the LABEL parameter value; when a match is found, the object is restored. If the last operation on the device specifies ENDOPT(*LEAVE) (the tape is positioned at the location at which the last operation terminated), the file search begins with the first data file beyond the current tape position. If ENDOPT(*LEAVE) was not used for the last operation (or if the tape was manually rewound since an ENDOPT(*LEAVE) operation), the search begins with the first data file on the volume.

file-sequence-number: Enter the sequence number of the file to be used for the restore process.

OPTION Parameter: Specifies whether each saved object to be restored is known to the library in the system that the objects are being restored to and how the objects are to be handled. In each case, objects already in the system are not affected.

*ALL: All the objects in the library saved on diskette (or tape) are restored in the system. Old objects on diskette or tape replace the current versions in the library in the system, and the objects not having a current version are added to the library in the system. Objects presently in the library remain there.

**NEW:* Only the objects in the library saved on diskette (or tape) that do not exist in the current version of the library in the system are to be added to the library. (Only objects not known to the library in the system are to be restored; known objects are not restored.) This option adds objects that were deleted since they were saved or that are new to this library. If any objects on diskette or tape have a current version in the library in the system, they are not restored and an informational message is issued for each one, but the restore continues.

**OLD:* Only the objects in the library that have an old version on diskette (or tape) are to be restored; that is, the current version of each object is replaced by the old version from diskettes. (Only objects known to the library are to be restored.) If any objects on the diskette or tape are no longer part of the current version of the library, they are not added to the library and an informational message is issued for each one, but the restore continues.

**FREE:* The saved objects are to be restored only if they exist in the library in the system with their space freed. The saved version of each object is restored in the system in its previously freed space. (This option restores objects that had their space freed when they were saved.) If any objects on the diskette or tape are no longer part of the current version of the library or if the space is not free for any object, the object is not restored and an informational message is issued for each one. The restore continues and all of the freed objects are properly restored.

MBROPT Parameter: Specifies, for data base files already known to the system, which members are to be restored. Unless MBROPT(*MATCH) is used, the member list in the saved file need not match, member for member, the current version in the system.

Note: Before restoring a file, the system checks to ensure that the file and member creation dates of the known system objects match the creation dates of the objects to be restored. If this check fails, the file is not restored.

<u>*MATCH:</u> The saved members are to be restored if the lists of the files in which they reside match, member for member, the lists of the current system version.

**ALL:* All members in the saved file are to be restored.

**NEW:* Only new members (members not known to the system) are to be restored.

**OLD:* Only members already known to the system are to be restored.

SAVDATE Parameter: Specifies the date on which the objects were saved. If the current version is not be to restored, enter the date that specifies which version of the objects are to be restored. (The date must be entered in the system date format specified by the system value QDATFMT; if separators, specified by the system value QDATSEP, are used, the value must be enclosed in apostrophes.) If a volume identifier is specified, but SAVDATE is not, the version of the objects to be restored is the first version found on the volume. This parameter is valid only if a volume identifier or VOL(*MOUNTED) is also specified.

SAVTIME Parameter: Specifies the time that the objects were saved. Enter the time as a six-digit value, in the format hours, minutes, and seconds (hhmmss), that specifies which version of the objects are to be restored if the current version is not to be restored. If colons are used as separators, the value must be enclosed in apostrophes ('hh:mm:ss'). If a volume identifier is specified, but SAVTIME is not, the version of the objects to be restored is the first version found on the volume.

This parameter is valid only if SAVDATE is also specified.

RSTLIB Parameter: Specifies whether the objects are to be restored to a different library or to the same library from which they were saved.

*SAVLIB: The objects are restored to the same library from which they were saved.

library-name: Enter the name of the library to which the saved objects are to be restored.

ENDOPT Parameter: Specifies, only when tape is used, what operation is to be automatically performed on the tape volume after the restore operation ends. If more than one reel is involved, this parameter applies only to the last reel used.

*REWIND: The tape is to be automatically rewound, but not unloaded, after the restore operation has ended.

**LEAVE:* The tape is not to be rewound; another restore operation can begin at the current position on the tape.

**UNLOAD:* The tape is to be automatically rewound and unloaded after the restore operation has ended.

Examples

```
RSTOBJ  OBJ(PAYROLL) SAVLIB(LIBX) OBJTYPE(*PGM)
```

This command restores to LIBX the program named PAYROLL that was saved from LIBX. Because the DEV, LOC, and VOL parameters are not specified, the diskette magazine drive is to be used to restore the most recently saved version of the program from magazines that must be currently mounted as magazines 1 and 2.

```
RSTOBJ  OBJ(PAY*) SAVLIB(LIBX) LOC(*M1 4) VOL(ABCD) +
      OPTION(*OLD) SAVDATE(102279) SAVTIME(143000) +
      RSTLIB(LIBY)
```

All objects whose names begin with PAY and that were saved from library LIBX on magazine volume ABCD at 14:30:00 on 10/22/79 are restored to LIBY. Volume ABCD must be mounted in magazine 1, and diskette position 4 contains the first diskette to be used. Because OPTION(*OLD) is specified, the only objects restored are those having the same object name and type both in LIBY in the system and LIBX on diskette.

```
RSTOBJ  OBJ(NEWPROG) SAVLIB(QGPL) OBJTYPE(*PGM) +
      LOC(*S3) VOL(PGMS) OPTION(*NEW)
```

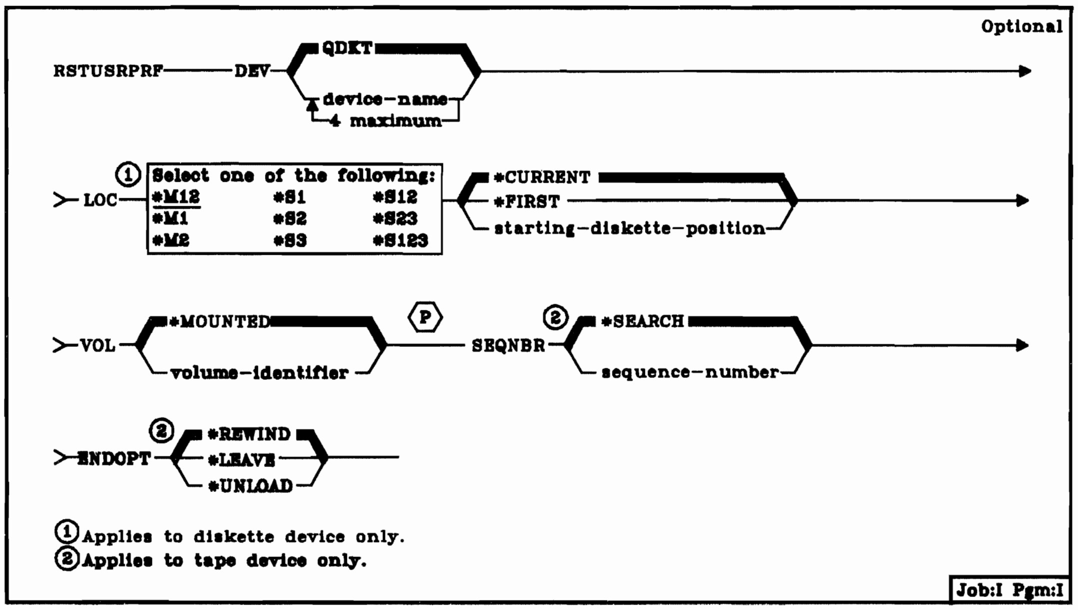
A new program named NEWPROG is added to the general purpose library, QGPL. It is to be restored from a diskette labeled PGMS that is inserted in manual slot S3.

RSTUSRPRF (Restore User Profiles) Command

The Restore User Profiles (RSTUSRPRF) command restores the basic parts of all user profiles that were saved by the Save System (SAVSYS) command. The RSTUSRPRF command restores only the special rights granted in the Create User Profile (CRTUSRPRF) command; it does not restore the authority for the named objects (owned by other users). To do that, the Restore Authority (RSTAUT) command must be used after the profiles, libraries, and objects have been restored.

Before this command is entered, all other operations on the system must be stopped. This requires terminating all subsystems through the TRMSBS command or entering this command when CPF is started. The RSTUSRPRF command is normally used after the install operation but before the user libraries are restored. (The user profiles must be restored before any libraries or objects belonging to them can be restored.) After the libraries and their objects are restored, the authority for the objects is restored to the user profiles by the RSTAUT command. See the *System/38 Operator's Guide* for information on installing (restoring) the system.

Restriction: To use this command, you must have the save system rights in your user profile, specified by SPCAUT(*SAVSYS).



DEV Parameter: Specifies the name of the diskette device or tape device(s) to be used to restore the user profiles. The device names must already be known on the system by a device description. For tape devices only, a maximum of four devices may be specified.

QDKT: The diskette device QDKT is to be used to restore the user profiles.

device-name: Enter the name of the diskette device or tape devices (maximum of four) to be used to restore the user profiles.

LOC Parameter: Specifies, only when diskettes are used, which diskette locations on the diskette magazine drive are to be used to restore the user profiles. The data is restored from diskettes that are loaded either in the magazines or in the manual slots. If the restore operation requires that additional magazines or diskettes be loaded, a message is issued to the operator to mount them. (For an expanded description of the LOC parameter, see Appendix A.)

To determine which of the diskettes contain the user profiles, use the information given in *Restoring User Profiles* in the *System/38 Operator's Guide*.

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first value specifies which unit (magazines or slots) and location are to be used for the restore operation. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be used first in the restore operation. Enter one of the following values for the starting diskette position:

*CURRENT: The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where a previous restore operation has just ended.) If the currently selected diskette is not anywhere in the location specified by the first value, a diagnostic message is sent to the system operator, and the command is terminated.

**FIRST:* The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

starting-diskette-position: If a multiple diskette location (such as *M1 or *S123) is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10 for magazines, or 1 through 3 for manual slots) that contains the first diskette to be used.

VOL Parameter: Specifies the volume identifiers of the tape volumes or diskette volumes (either in magazines or slots) from which the user profiles are to be restored. The volumes must be mounted on the device(s) in the same order as they were when the user profiles were saved by the SAVSYS command. (For an expanded description of the VOL parameter, see Appendix A.)

*MOUNTED: For diskette, the user profiles are to be restored from the volumes that are currently mounted in the locations specified by LOC. For tape, the user profiles are to be restored from the volumes mounted on the tape devices, in the order that those devices were specified. This operation is normally performed immediately after an installation has been performed and the user profiles are on the same volume.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used to restore the user profiles. For tape volumes and single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the *magazine* identifier (5 characters maximum) can be specified for each volume. After the diskettes in the first magazine have been processed, the restore operation always continues with position 1 of the remaining magazines.

SEQNBR Parameter: Specifies the sequence number of the user profile to be used for the restore process.

*SEARCH: The mounted volume is searched for a file with an identifier that matches the LABEL parameter value; when a match is found, the user profile is restored. If the last operation on the device specifies ENDOPT(*LEAVE) (the tape is positioned at the location at which the last operation terminated), the file search begins with the first file beyond the current tape position. If ENDOPT(*LEAVE) was not used for the last operation (or if the tape was manually rewound since an ENDOPT(*LEAVE) operation), the search begins with the first file on the volume.

file-sequence-number: Enter the sequence number of the file to be used to restore user profiles.

ENDOPT Parameter: Specifies the positioning operation to be performed automatically on the tape volume when the save/restore file is closed. In the case of a multiple-volume save/restore file, this parameter applies to the *last* reel only; all other reels are rewound and unloaded when the end of the tape is reached.

**REWIND:* The tape is to be rewound, but not unloaded, when the file is closed.

**UNLOAD:* The tape is to be rewound and unloaded when the file is closed.

**LEAVE:* The tape should be left in its current position when the file is closed; it is not to be rewound or unloaded.

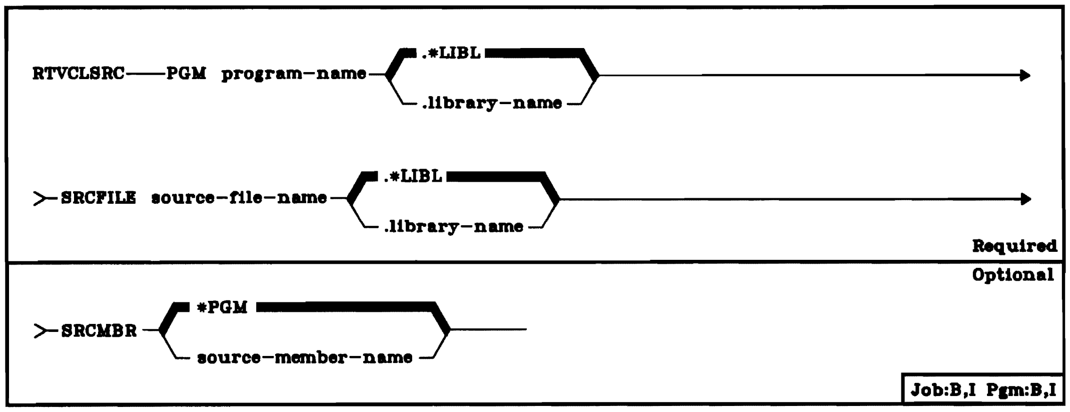
Example

```
RSTUSRPRF
```

The saved version of all user profiles contained on the currently mounted diskette volume is to be restored in the system. The version must have been saved on one or more magazines, which must be mounted on M1 and M2 for the restore.

RTVCLSRC (Retrieve CL Source) Command

The Retrieve CL Source (RTVCLSRC) command is used to retrieve the source statements from a CL program used to compile that program. These source statements are placed into a source file member, which can be used as input when recompiling the CL program.



PGM Parameter: Specifies the qualified name of the CL program whose source is to be retrieved. If the program name is not qualified with a library name, *LIBL is used to find the program.

SRCFILE Parameter: Specifies the qualified name of the previously created data base source file into which the CL source statements are to be written. If the source file name is not qualified with a library name, *LIBL is used to find the source file.

SRCMBR Parameter: Specifies the qualified name of the data base source file member into which the CL source statements are to be written. If not specified, the CL program name will be assumed. If the member existed prior to the execution of this command, it will be cleared before any source statements are written into it. If the member did not exist, it will be created.

*PGM: The name of the CL program is to be used as the member name.

source-member-name: Enter the name of the source file member that will contain the CL source statements.

Example

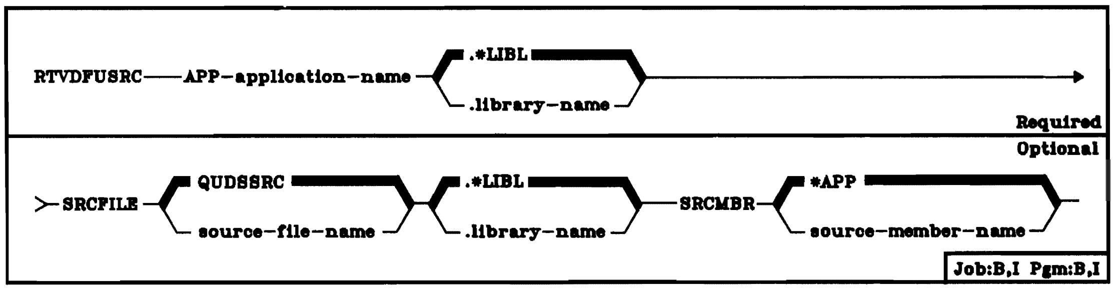
```
RTVCLSRC  PGM(TEXT1.JOHN1) SRCFILE(JOHN2) SRCMBR(JOHN3)
```

This command retrieves the source statements from the CL program named TEXT1 in library JOHN1. The retrieved source statements are placed into the file named JOHN2, and are named as member JOHN3.

RTVDFUSRC (Retrieve DFU Source) Command

The Retrieve DFU Source (RTVDFUSRC) command retrieves UDS (utility definition statements) used by the DFU (Data File Utility) to define a DFU application. The UDS that are retrieved are functionally equivalent if not identical in format to those that were originally generated. The UDS are placed, in correct sequence, into the source file and source member identified by the SRCFILE and SRCMBR parameters. After this command is executed, the source file and member can be used as input to the Create DFU Application (CRTDFUAPP) command to create an application.

The Data File Utility is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Data File Utility, refer to the *IBM System/38 DFU Reference Manual and User's Guide,* SC21-7714.



APP Parameter: Specifies the qualified name of the DFU application whose UDS are to be retrieved. If no library qualifier is specified, *LIBL is used to find the application.

SRCFILE Parameter: Specifies the name of an existing data base file as the location to which the retrieved UDS are to be written. If no library qualifier is specified, *LIBL is used to find the data base file.

QUDDSRC: The retrieved UDS are to be stored in the IBM-supplied source file.

source-file-name: Enter the name of the source file.

SRCMBR Parameter: Specifies the name of an existing source file member as the location to which the retrieved UDS are to be sent. If no name is specified, the application name (specified in the APP parameter) is used.

APP: The name of the DFU application is to be used as the member name.

source-member-name: Enter the name of the source member.

Example

```
RTVDFUSRC APP(JAMES.JAMES1) SRCFILE(JAMES2) SRCMBR(JAMES3)
```

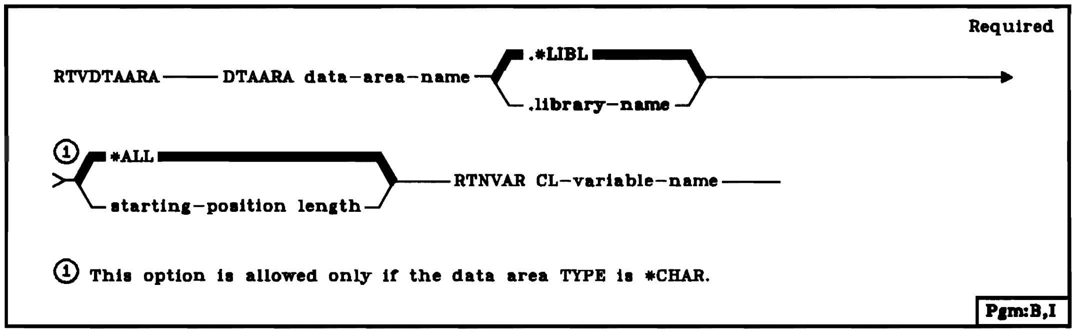
This command retrieves the utility definition source statements from the application named JAMES in library JAMES1. The retrieved source statements are placed into the file named JAMES2, and are named as a member JAMES3.

RTVDTAARA (Retrieve Data Area) Command

The Retrieve Data Area (RTVDTAARA) command is used in a CL program to retrieve all or part of a specified data area and copy the portion of the data area being retrieved into a CL variable within the CL program. RTVDTAARA does not retrieve any other attributes of the data area. The data area is not required to exist at the time the CL program is compiled.

When the RTVDTAARA command is executed, the data area is locked during the retrieval operation so that commands in other jobs cannot change or destroy the data area until the operation is complete. If the data area is shared with other jobs and is updated in steps involving more than one command in a job, the data area should be explicitly allocated to that job until all the steps have been performed. The data area can be explicitly allocated with the ALCOBJ command.

Restrictions: (1) This command is valid only in compiled CL programs. (2) To use this command, you must have operational rights to the data area and read rights to the library in which the data area is located.



DTAARA Parameter: Specifies the qualified name of the data area whose value is to be retrieved. If no library qualifier is given, *LIBL is used to find the data area.

The substring retrieval option, valid for character data areas only, specifies the starting position and the length of the portion of the data area that is to be retrieved into the CL character variable.

*ALL: Specifies that the entire data area will be retrieved.

Starting position element: Specifies the starting position of the data area to be retrieved.

Length element: Specifies the length of the data area substring to be retrieved.

Note that it is not possible to retrieve data outside the bounds of the data area. The combination of starting position and length must always specify positions within the data area.

RTNVAR Parameter: Specifies the name of the CL program variable that is to receive the contents of the data area being returned.

No type conversion is performed by the RTVDTAARA command. If RTNVAR is declared as TYPE(*DEC), the data area retrieved must be TYPE(*DEC). If RTNVAR is declared as TYPE(*CHAR), the data area retrieved must be either TYPE(*CHAR) or TYPE(*LGL). If RTNVAR is declared as TYPE(*LGL), the data area retrieved must be either TYPE(*LGL) or TYPE(*CHAR) with a value of either '0' or '1'.

If a retrieved character string is shorter than the length of the variable specified by the RTNVAR parameter, the value will be padded to the right with blanks. The retrieved string length must be less than or equal to the CL variable length.

When decimal data areas are retrieved, the decimals are aligned. The value of the integer portion of the data area must be able to be placed into the integer portions of the CL variable. Fractional data will be truncated if the fraction contains more digits than the CL variable.

Character Data Area Example

Assume data area DA1 has been created by the following command:

```
CRTDTAARA  DTAARA(DA1) TYPE(*CHAR) LEN(3) VALUE(ABC)
```

and variable &CLVAR1 has been declared as follows:

```
DCL  VAR(&CLVAR1) TYPE(*CHAR) LEN(5) VALUE(VWXYZ)
```

The execution of the command:

```
RTVDTAARA  DTAARA(DA1) RTNVAR(&CLVAR1)
```

will yield the following:

```
&CLVAR1 = 'ABC  '
```

The execution of the command:

```
RTVDTAARA  DTAARA(DA1 (2 1)) RTNVAR(&CLVAR1)
```

will yield the following:

```
&CLVAR1 = 'B    '
```

Decimal Data Area Example

Assume data area DA2 has been created with the following attributes:

```
CRTDTAARA  DTAARA(DA2) TYPE(*DEC) LEN(5 2) VALUE(12.39)
```

and variable &CLVAR2 has been declared as follows:

```
DCL  VAR(&CLVAR2) TYPE(*DEC) LEN(5 1) VALUE(4567.8)
```

The execution of this command:

```
RTVDTAARA  DTAARA(&DTAARA.MYLIB) RTNVAR(&CLVAR2)
```

will yield the following:

```
CLVAR2 = 0012.3
```

Note that fractional digits are truncated rather than rounded.

RTVJOBA (Retrieve Job Attributes) Command

The Retrieve Job Attributes (RTVJOBA) command is used in a CL program to retrieve the values of one or more job attributes and place the values into the specified CL variable. The attributes are retrieved for the job in which this command is used. The following attributes can be retrieved:

- The job name (JOB), user profile name (USER), and job number (NBR)
- The job type (TYPE), which can be batch or interactive
- The date when the job was started (DATE)
- The values of the eight job switches (SWS) associated with the job
- The return code (RTNCDE) set by the last RPG, COBOL, DFU, or conversion reformat utility program that has finished execution in the job
- The cancellation status (CNLSTS), which indicates whether a controlled cancel operation affecting the job is in progress

Restriction: This command is valid only within a CL program.

```
                                                                              Optional
RTVJOBA────────JOB CL-variable-name────────USER CL-variable-name──────────────────────►
                                     (P)
>─NBR CL-variable-name────────TYPE CL-variable-name──────────────────────────────────►

>─DATE CL-variable-name────────SWS CL-variable-name──────────────────────────────────►

>─RTNCDE CL-variable-name────────CNLSTS CL-variable-name────────
                                                                               Pgm:B,I
```

JOB Parameter: Specifies, if the job name is to be returned, the name of the CL variable that is to receive the name of the job. The variable must be a character variable with a minimum length of 10 characters. (If the job name has fewer characters than the variable allows, the value is padded on the right with blanks.)

USER Parameter: Specifies, if the user name is to be returned, the name of the CL variable that is to receive the name of the user profile associated with the job when the job was started. The user name is the second part of the qualified job name. The variable must be a character variable with a minimum length of 10 characters. (If the user name has fewer characters than the variable allows, the value is padded on the right with blanks.)

NBR Parameter: Specifies, if the job number is to be returned, the name of the CL variable that is to receive the unique 6-character number assigned to the job by the system. The job number is the third part of the qualified job name. The variable must be a character variable with a minimum length of 6 characters.

TYPE Parameter: Specifies, if the type of job environment is to be returned, the name of the CL variable that is to receive the 1-character value representing the environment of the job. A character value of 0 indicates the job is executing as a batch job, and a 1 indicates an interactive job. The variable must be a character variable with a minimum length of 1 character.

DATE Parameter: Specifies, if the job date is to be returned, the name of the CL variable that is to receive the date assigned to the job by the system when the job was started. The variable must be a character variable with a minimum length of 6 characters. (The job date is returned in the system date format specified by the system value QDATFMT.)

SWS Parameter: Specifies, if the job switches are to be returned, the name of the CL variable that is to receive the value of the eight job switches used by the job. The job switches are retrieved as a single 8-character value with each of the characters specifying a 1 or 0 as the value of the associated switch. The CL variable must be a character variable with a minimum length of 8 characters.

RTNCDE Parameter: Specifies, if the completion status of an RPG, COBOL, DFU, or conversion reformat utility program is to be returned, the name of the CL variable that is to receive the five-digit decimal return code. The return code is set by these programs before they return to the programs that called them. The return code indicates the completion status of the last program (of these types) that has finished execution within the job, as follows:

0	Normal return (RPG, COBOL, DFU, or Conversion Reformat Utility)
1	LR (last record) indicator on (RPG)
2	Error–no halt indicator set (RPG, COBOL, DFU, or Conversion Reformat Utility)
3	Halt indicator set on (one of the RPG indicators H1 through H9)

The CL variable must be a five-position decimal variable with no decimal positions.

CNLSTS Parameter: Specifies, if checking for a controlled cancel operation, the name of the CL variable that is to receive the cancellation status. The single-character value indicates whether a controlled cancel that affects the job is currently being performed. A value of 1 indicates that the system, the subsystem in which the job is running, or the job itself is being canceled; a 0 indicates no controlled cancel is being performed. The CL variable must be a character variable with a minimum length of 1 character.

Example

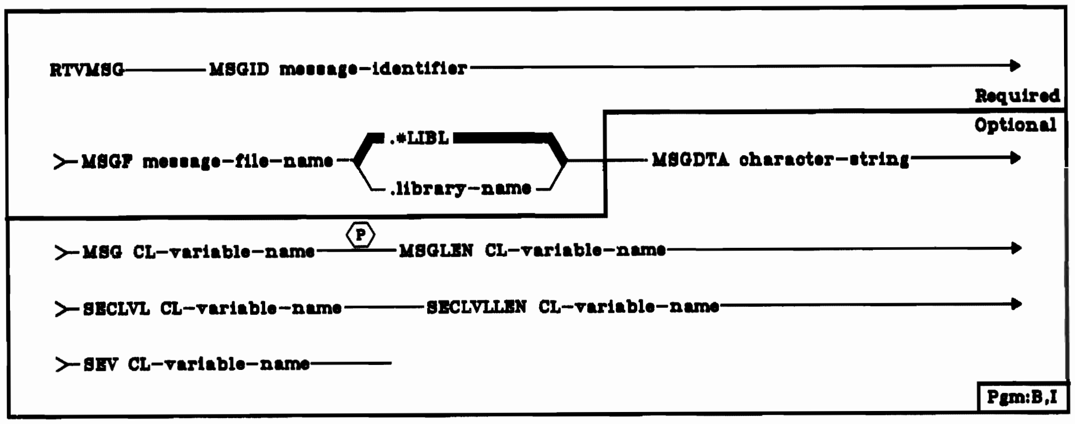
```
RTVJOBA  NBR(&JOBNBR) DATE(&JOBDATE)
```

This command retrieves the job number and job date attributes from the job that this command is in. The six-digit job number is to be copied into the CL variable &JOBNBR. The job date is to be copied into the CL variable &JOBDATE; both variables must be character variables with a length of six. The format of the date is determined by the contents of the system value QDATFMT, which controls the system date format.

RTVMSG (Retrieve Message) Command

The Retrieve Message (RTVMSG) command is used by a program to retrieve a specified predefined message from a message file and copy it into CL variables within the program. Substitution values can be specified in the MSGDTA parameter (as a single character string containing one or more concatenated message data fields) to replace the substitution variables in the predefined message text. The program can later write the message to an output device file to be printed, for example.

Restrictions: This command is valid only in compiled CL programs. To use this command, you must have read rights for the message file, and read and operational rights for the library in which the message is stored.



MSGID Parameter: Specifies the message identifier of the predefined message that is to be retrieved from the specified message file.

MSGF Parameter: Specifies the qualified name of the message file that contains the predefined message to be retrieved. (If no library qualifier is given, *LIBL is used to find the file.)

MSGDTA Parameter: Specifies, if the predefined message contains substitution variables, the substitution values that are to be used as the message data fields within the retrieved message. Either a character string or a CL variable containing the character string can be specified. As many substitution values can be specified in the character string as there are substitution variables in the predefined message. The values, which take the place of the substitution variables defined in the message text when the message was defined, must be specified according to the following rules:

- Multiple values must be concatenated together to form a single character string. The length of the entire character string of concatenated substitution values cannot exceed 132 characters. (See the *CPF Programmer's Guide* for more details.)
- Multiple values must be specified in the same order in the character string as the substitution variables were defined in the MSG and FMT parameters of the ADDMSGD command.
- Each value must be specified as long as its associated variable was defined, and the specified character string must be the same length as the sum of the message data fields defined in the message.
- For multiple values, each substitution value can be specified as a CL program constant or CL variable; that is, any combination of constants and variables can be specified if they are first concatenated into one character string and they are in the same format and sequence expected.
- The length of the substitution value should be the same length as the length defined for the substitution variables. If the substitution value length is longer than the substitution variable length, the message data will be truncated. If the substitution value is shorter, it becomes a null field.

Enter the character string that is to be used as the substitution value in the specified predefined message that is to be sent by the program, or enter the name of the CL variable that contains the character string. For more information on coding this parameter, refer to the description of the MSGDTA parameter in the SNDPGMMSG command.

MSG Parameter: Specifies the name of the CL character variable in the program into which the first-level text of the retrieved message is to be copied. If a CL variable name is not specified, the first-level text is not copied into the program.

The specified variable must be a character variable. If the retrieved first-level text is longer than the variable's field length, the text is truncated. If the text is shorter, it is padded to the right with blanks. Although this is a variable length field, most messages are designed to be less than 132 characters.

MSGLEN Parameter: Specifies the name of the CL decimal variable in the program into which the total length of the first-level text available to be retrieved is to be copied. The length specified is the total length after the substitution values (specified in the MSGDTA parameter) have been placed in the first-level text.

The specified variable must be a decimal variable that has a length of five digits. If a CL variable name is not specified, the length is not copied into the program.

SECLVL Parameter: Specifies the name of the CL character variable in the program into which the second-level text of the retrieved message is to be copied. If a variable name is not specified, the second-level text is not copied into the program.

If the retrieved second-level text is longer than the variable's field length, the text is truncated. If the text is shorter, it is padded to the right with blanks. Although this is a variable length field, most second-level text is designed to be less than 1435 characters.

SECLVLLEN Parameter: Specifies the name of the CL decimal variable in the program into which the total length of the second-level text being retrieved is to be copied. The length specified is the total length after the substitution values (specified in the MSGDTA parameter) have been placed in the second-level text.

The specified variable must be a decimal variable that has a length of five positions. If a CL variable name is not specified, the length is not copied into the program.

SEV Parameter: Specifies the name of the CL decimal variable into which the severity code of the retrieved message is to be copied. The specified variable must be a decimal variable that has a length of two positions. If a variable name is not specified, the severity code of the retrieved message is not copied into the program.

Examples

```
RTVMSG  MSGID(UIN0145) MSGF(INVN) MSG(&WORK) +
        MSGDTA('any old time')
```

This command retrieves the message text of the message UIN0145 stored in the INVN message file. The retrieved text is copied into the CL variable &WORK after the substitution variables are replaced with the values *any*, *old*, and *time*. This example assumes that the substitution variables &1, &2, and &3 have been declared in the program as character variables, each 4 characters long.

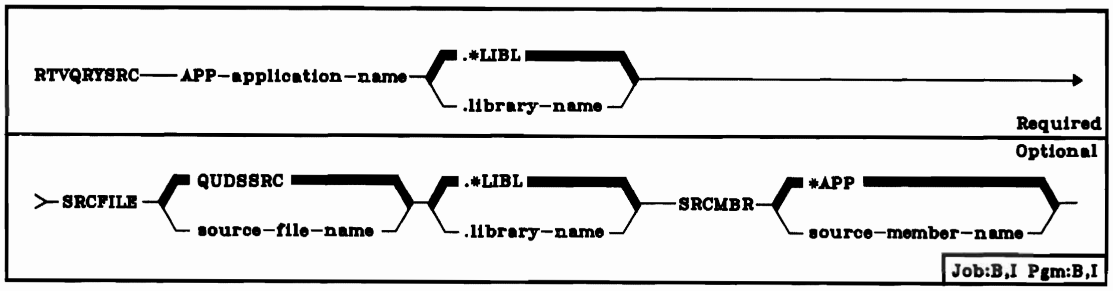
```
RTVMSG  MSGID(UIN0150) MSGF(INV) MSG(&MSG) +
        SECLVL(&SECLVL)
```

This command retrieves the first- and second-level text of the message UIN0150, which is stored in message file INV, and moves it into the CL variables &MSG and &SECLVL.

RTVQRYSRC (Retrieve Query Source) Command

The Retrieve Query Source (RTVQRYSRC) command retrieves UDS (utility definition statements) used by Query to define an application. The UDS that are retrieved are functionally equivalent if not identical in format to those that were originally generated. The UDS are placed, in correct sequence, into the source file and source member identified by the SRCFILE and SRCMBR parameters. After this command is executed, the source file and member can be used as input to the Design Query Application (DSNQRYAPP) command to create an application.

Query is part of the *IBM System/38 Interactive Data Base Utilities Licensed Program,* Program 5714-UT1. For more information on the Query utility, refer to the *IBM System/38 Query Utility Reference Manual and User's Guide,* SC21-7724.



APP Parameter: Specifies the qualified name of the Query application whose UDS are to be retrieved. If no library qualifier is specified, *LIBL is used to find the application.

SRCFILE Parameter: Specifies the qualified name of an existing data base file as the location to which the retrieved UDS are to be written. If no library qualifier is specified, *LIBL is used to find the data base file.

QUDDSRC: The retrieved UDS are to be stored in the IBM-supplied source file.

source-file-name: Enter the name of the source file.

SRCMBR Parameter: Specifies the name of an existing source file member as the location to which the retrieved UDS are to be sent. If no name is specified, the application name (specified in the APP parameter) is used.

APP: The name of the query application is to be used as the member name.

source-member-name: Enter the name of the source member.

Example

```
RTVQRYSRC  APP(QSCAN.MARY1)  SRCFILE(FILE10)  SRCMBR(QRYMBR)
```

This command retrieves the Query utility definition statements from the application named QSCAN in library MARY1. The retrieved source statements are placed into the file named FILE10 and are named as a member QRYMBR.

RTVSYSVAL (Retrieve System Value) Command

The Retrieve System Value (RTVSYSVAL) command is used in a CL program to retrieve the value from the specified system value so that it can be used in the program. The value is returned (copied) to the specified CL variable in the program.

Restrictions: This command is valid only in compiled CL programs. The attributes of the system value and the receiving CL variable must be compatible.

```
                                                                                  Required
RTVSYSVAL——— SYSVAL system-value-name———RTNVAR  CL-variable-name———
                                                                                  Pgm:B,I
```

SYSVAL Parameter: Specifies the name of the system value whose value is to be retrieved and returned for use in the program. For the names and descriptions of the system values that can be specified, refer to the *CPF Programmer's Guide.*

RTNVAR Parameter: Specifies the name of the CL program variable that is to receive the value of the system value being returned. The type and length for the CL variable when it was declared must be compatible with that of the system value whose value is to be received. (See the *CPF Programmer's Guide* for the attributes of individual system values.)

In general, the type of the return variable must match the type of the system value. For character system values that are 1 character long, the CL variable can be a character or logical variable. For character and logical system values, the length of the CL variable must equal the length of the system value. For decimal system values, the CL variable must have a length that is greater than or equal to the length of the system value.

Example

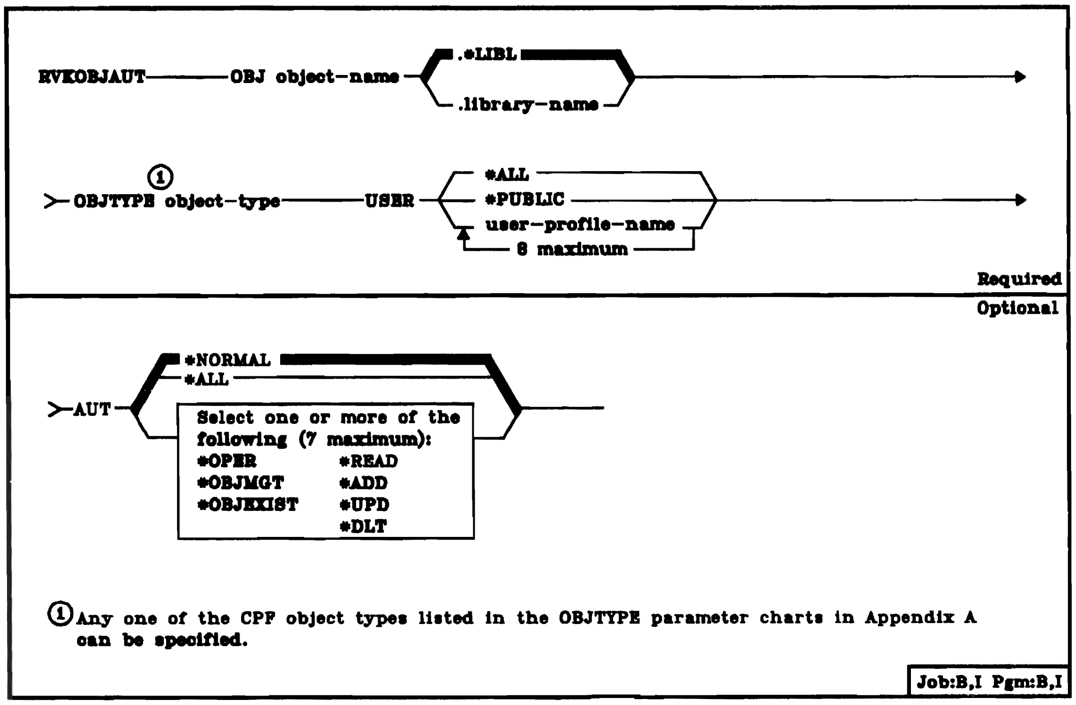
```
RTVSYSVAL  SYSVAL(QTIME) RTNVAR(&TIME)
```

This command retrieves the time value from the system value QTIME and copies it into the CL variable &TIME. The CL variable must be declared as a 6-character variable to match the attributes of the system value.

RVKOBJAUT (Revoke Object Authority) Command

The Revoke Object Authority (RVKOBJAUT) command is used to withdraw explicit (or all) rights of use for the named object from one or more users also named in the command. This command can be entered by the security officer, by an object's owner, or by a user who has object management rights for the object being revoked. A user who has object management rights can revoke only the explicit rights that he himself has. Also, a user may not be able to grant or revoke rights for an object that has been allocated (locked) to another job.

Restrictions: Before this command can be used to revoke rights of use for a device, control unit, or line description, its associated device, control unit, or line must be varied on. Also, for display work stations, if this command is not entered at the device for which rights are being revoked, this command should be preceded by the ALCOBJ command and followed by the DLCOBJ command.



OBJ Parameter: Specifies the qualified name of the object for which the rights of use are being withdrawn from one or more users. (If no library qualifier is given, *LIBL is used to find the object.)

OBJTYPE Parameter: Specifies the object type of the object that is to have specific rights revoked. Any one of the CPF object types can be specified; refer to the charts in the expanded description of the OBJTYPE parameter in Appendix A. To revoke authority for a library, for example, enter the value *LIB.

USER Parameter: Specifies the names of one or more users from whom specific rights to the named object are being withdrawn. If a user was granted the rights by USER(*PUBLIC) being specified in the GRTOBJAUT command, the same rights can be revoked by *PUBLIC being specified in this parameter. If he was explicitly authorized (he was granted the rights by having his name specified), his name must be specified in this parameter to revoke the same rights.

**ALL:* The rights specified in the AUT parameter are being withdrawn from all enrolled users of the system except the owner, whether they were publicly or explicitly authorized.

**PUBLIC:* The specified rights are being withdrawn from users who were publicly authorized. Any users who were explicitly authorized still retain their rights to the object.

user-profile-name: Enter the user profile names of one or more users that are having the specified rights revoked. The rights specified in the AUT parameter are being explicitly withdrawn from each user specified. This parameter cannot be used to revoke public authority from specific users; only rights that were explicitly granted to them can be explicitly revoked.

AUT Parameter: The authority parameter specifies the rights of use that are being withdrawn from the users named for the specified object.

Note: Refer to the chart in the *CPF Programmer's Guide* that shows the applicable rights of use for each object type.

<u>*NORMAL:</u> Normal rights of use are being withdrawn from the specified users. *NORMAL means that operational rights to the object and common data rights are being revoked.

**ALL:* All rights of use applicable to the specified object are being withdrawn.

**OPER:* Operational rights for the specified object are being revoked. Operational rights provide the authority to use an object, look at its description, and restore it.

**OBJMGT:* Object management rights, which provide the authority to manage the access and availability of an object, are being revoked. A user with object management rights can grant (and revoke) the rights that he has, as well as move and rename objects, and add members to data base files.

**OBJEXIST:* Object existence rights, which provide the authority to control object ownership and existence, are being revoked. This right of use allows the user to delete, free storage, save, restore, or transfer ownership of an object.

***READ:** Read rights, which provide the authority needed to retrieve the contents of an object entry, are being revoked.

***ADD:** Add rights, which provide the authority needed to add entries to an object, are being revoked. (For example, adding job entries to a queue or adding records to a file.)

***UPD:** Update rights, which provide the authority needed to change the entries in an object, are being revoked.

***DLT:** Delete rights, which provide the authority needed to remove entries in an object, are being revoked. (For example, deleting a program from a library or records from a file.)

Examples

```
RVKOBJAUT  OBJ(PROG1.ARLIB) OBJTYPE(*PGM) USER(*ALL)
```

This command withdraws from *all* users except the owner (who were either explicitly or publicly authorized) the normal rights of use (AUT was not specified; *NORMAL is assumed) for the program (*PGM) named PROG1, located in the library named ARLIB.

```
RVKOBJAUT  OBJ(TSMITHPGM.SMITHLIB) +
      OBJTYPE(*PGM) USER(TMSMITH) +
      AUT(*OBJEXIST)
```

This command withdraws the object owner's (T M Smith) authority to delete a program (TSMITHPGM) in his library (SMITHLIB). (The object owner might do this to ensure that he does not delete the object inadvertently.) If the owner ever wants to delete the object, he can grant himself object existence rights (using the GRTOBJAUT command) for the object.

```
RVKOBJAUT  OBJ(FILEX) OBJTYPE(*FILE) +
      USER(HEANDERSON) AUT(*DLT *UPD)
```

This command withdraws delete and update rights for the file named FILEX from the user H E Anderson.

```
RVKOBJAUT  OBJ(ARJOBD.ARLIB) OBJTYPE(*JOBD) +
      USER(RLJOHNSON) AUT(*OBJEXIST)
```

The command withdraws the object existence authority for the object named ARJOBD from the user R L Johnson. ARJOBD is a job description that is located in the library named ARLIB.

SAVCHGOBJ (Save Changed Object) Command

The Save Changed Object (SAVCHGOBJ) command saves a copy of each changed object or group of objects located in the same library. For data base files, only the changed members are saved. You can specify the date and time from which the system will save the objects or members.

Objects changed since the date and time specified will be saved with the following exceptions:

- Data base files currently being journaled will not be saved, unless journaling was started after the specified date and time. This ensures that changes made to a physical file before journaling begins are not lost (since they have not been recorded in a journal receiver).
- Data base file members that have been added or changed will be saved.
- Freed objects (programs, files, journal receivers, and so forth) will not be saved.
- Message, job, and output queues will not be saved.

Specified objects are locked (shared, no update) while their change date and time is being checked, and the objects that have been changed and the library in which they reside remain locked for the duration of the save operation.

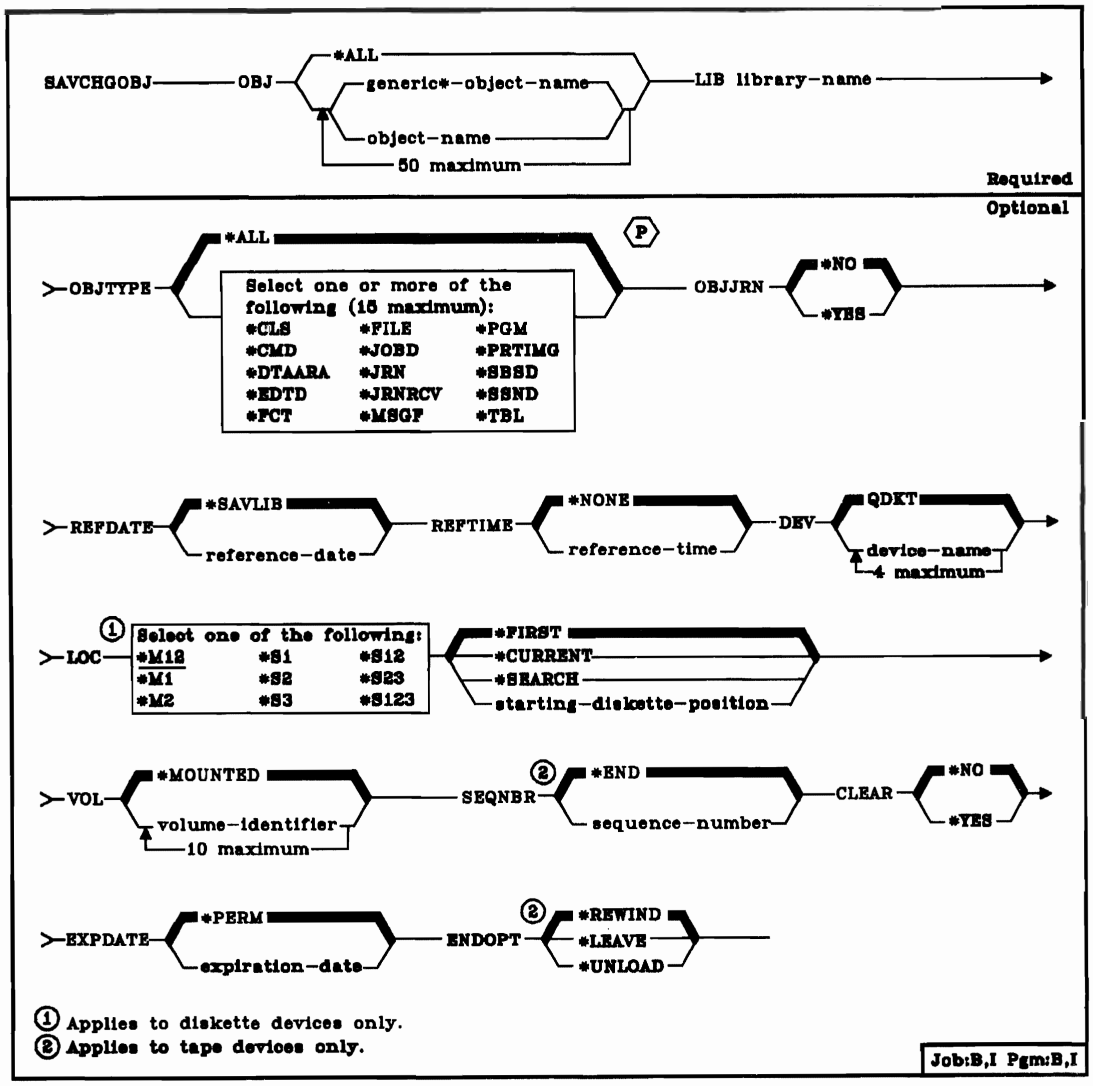
Saved objects can be restored with the Restore Object (RSTOBJ) command.

To determine the date and time that an object was changed, execute the Display Object Description (DSPOBJD) command with DETAIL(*FULL) specified. For data base file members that have been changed, execute the Display File Description (DSPFD) command. Refer to the *Additional Considerations* sections of those commands for more information.

The types of objects that can be saved by this command are listed in the OBJTYPE parameter. The system saves the changed objects on offline storage by writing a copy of each one onto diskettes or tapes. The objects are not affected in the system unless the command specifies that the storage is to be freed. However, the description of each object is updated with the date, time, and place when it was last saved.

Restrictions: To use this command, you must have the special authority *SAVSYS specified in your user profile by the SPCAUT parameter. Otherwise, you must have: (1) object existence rights for each object specified, and (2) read rights for the specified library. (If you do not have the necessary rights to a specified object, all changed objects except that one are saved.)

All diskettes in the magazines or slots to be used to save the changed objects must have been initialized in the save/restore format. If tape is to be used, a standard labeled volume must be mounted. Also, no changed object being saved can be updated by a job that is executing at the time the save operation occurs.



OBJ Parameter: Specifies the names of one or more changed objects that are to be saved. All the objects must be in the library specified in the LIB parameter. Enter the name of each object or the generic name of each group of objects to be saved.

If the OBJTYPE parameter is not specified, all the object types listed in the description of the OBJTYPE parameter are saved, provided they are in the specified library and have the specified names.

ALL: All changed objects in the specified library are to be saved, depending on the values specified for OBJTYPE.

generic-object-name:* Enter one or more generic names of groups of objects in the specified library whose changes are to be saved.

object-name: Enter as many as 50 names of specific changed objects that are to be saved. (Both generic names and specific names can be specified in the same command.)

LIB Parameter: Enter the name of the library that contains the changed objects that are to be saved on offline storage.

OBJTYPE Parameter: Specifies the types of CPF objects whose changes are to be saved. (Changes to *LIB, *USRPRF, *LIND, *CUD, and *DEVD objects are not monitored by the SAVCHGOBJ command for later display.)

<u>*ALL:</u> Changes to all object types that are specified by name and are in the specified library are to be saved. If *ALL is also specified on the OBJ parameter, then changes to all objects in the library that are of the types shown in the following table are monitored.

object-type: Enter the value for each of the types of objects that are to be saved. The following types can be specified:

Value	**Type of CPF Object**
*CLS	Class
*CMD	Command
*DTAARA	Data area
*EDTD	Edit description
*FCT	Forms control table
*FILE	File
*JOBD	Job description
*JRN	Journal
*JRNRCV	Journal receiver
*MSGF	Message file
*PGM	Program
*PRTIMG	Print image
*SBSD	Subsystem description
*SSND	Session description
*TBL	Table

OBJJRN Parameter: Specifies whether changes to objects currently being journaled (as specified in the Journal Physical File (JRNPF) command) are to be saved.

*NO: Changes to journaled objects (for which journaling was begun before the date and time specified in the REFDATE and REFTIME parameters) will not be saved, but the journal receivers are saved. If the JRNPF command was executed after the specified data and time, all changed members will be saved.

**YES:* Changes to journaled objects will be saved.

REFDATE Parameter: Specifies the date after which changed objects in the library are to be saved.

*SAVLIB: The changed objects are to be saved, starting from the date of the last execution of the Save Library (SAVLIB) command. If the specified library has never been saved, a message is issued and the operation terminates.

reference-date: Enter the date from which changed objects are to be saved. If this date is later than the date of the execution of this command, a message is issued and the operation terminates.

REFTIME Parameter: Specifies a time from which the changed objects of the specified library are to be saved.

*NONE: No explicit time is to be specified. The changed objects will be saved from the beginning of the date specified by the REFDATE parameter.

reference-time: Enter the time after which changed objects are to be saved. If REFDATE(*SAVLIB) is specified, no reference time can be specified. If this time is greater than the time of the execution of this command, a message is issued and the operation terminates.

DEV Parameter: Specifies the name of the device on which the changed objects are to be saved. The device name must already be known on the system by a device description.

QDKT: The diskette device QDKT is to be the device used to save the objects.

device-name: Enter the name of the diskette or tape device on which the library is to be saved. If you are using more than one tape device (up to a maximum of four), enter the names of the devices in the order they are to be used. If you are using more than one tape volume, you may want to use more than one tape device, so that one tape volume can rewind/unload while another tape device is processing.

LOC Parameter: Specifies, only when diskettes are used, which diskette locations on the diskette magazine drive are to be used to save the specified objects. The data is saved (stored) on diskettes that are loaded either in the magazines or in the manual slots. If magazines are used, all the diskette positions to be used must contain diskettes initialized in the save/restore format. If all of the available space in the specified magazines or slots is filled before the save operation is finished, a message is issued to the operator to mount additional magazines or diskettes. (This parameter is ignored if a tape device is specified in the DEV parameter.)

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first value specifies which unit (magazines or slots) and location are to be used for the save operation. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be used first in the save operation. Enter one of the following values for the starting diskette position:

*FIRST: The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where a previous save or restore operation has just ended.) If the currently selected diskette is not in the location specified by the first value, a diagnostic message is sent to the system operator, and the command is terminated.

**SEARCH:* The diskettes in the location specified by the first value are to be searched for the first diskette having cleared space, where the save operation can begin. The search begins with the leftmost diskette in the specified location and ends with the first diskette on which cleared space is found.

position-in-range: If a multiple diskette location (such as *M1 or *S23) is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10 for magazines, or 1 through 3 for manual slots) that contains the first diskette to be used.

VOL Parameter: Specifies the volume identifiers of the tape volumes (reels) or diskette volumes (either magazines or slots) on which the object data is to be saved. The volumes must be mounted in the same order as the volume identifiers are specified in this parameter. (For an expanded description of the VOL parameter, see Appendix A.)

*MOUNTED: The objects to be saved are copied on whatever volumes are mounted on the device. For diskettes, the volumes are those mounted on the diskette magazine drive in the location specified by LOC. All diskettes to be used in the mounted volumes specified by LOC should already be cleared, unless CLEAR(*YES) is specified.

volume-identifier: Enter the identifiers of one or more volumes (10 maximum) in the order in which they are to be mounted and used to save the objects. For tape volumes and single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the magazine identifier (5 characters maximum) can be specified for each volume. The save operation always continues with position 1 of every magazine after the first one.

SEQNBR Parameter: Specifies, only when tape is used, which sequence number is to be used as the starting point for saving changed objects.

*END: The system will save the objects after the last sequence number on the tape volume.

file-sequence-number: Enter the sequence number of the file to be used as a starting point to save changed objects.

CLEAR Parameter: Specifies, only when diskettes are used, whether uncleared diskettes encountered during the save operation are to be automatically cleared. (This parameter does not control initializing the diskettes; they must already be initialized in the save/restore format.) If tapes are used and a sequence number is specified, the tape will be cleared starting with that sequence number; all tapes following the first tape will be cleared.

*NO: Uncleared diskettes encountered during the save operation are not to be automatically cleared. If, after the first diskette used, any uncleared diskette is encountered, an inquiry message is sent to the system operator, allowing him to terminate the save operation or to specify that the currently selected diskette be cleared so the operation can continue. All diskettes to be used to save the objects should be cleared before the SAVOBJ command is issued.

YES: Uncleared diskettes encountered during the save operation are to be automatically cleared so the save can continue. If *CURRENT or *SEARCH was specified on the LOC parameter, this parameter applies only to uncleared diskettes following the diskette on which the save operation began. In these cases, the first diskette used was already clear (or had sufficient space to begin the save operation) because it was the diskette at which the *last* save operation ended (*CURRENT), or it was found to be clear (or to have sufficient space) by *SEARCH in this operation. If *FIRST or a starting diskette position was specified for LOC, this parameter applies to all diskettes (including the first one) used in the operation.

EXPDATE Parameter: Specifies the expiration date of the saved data files. The data file expiration date is used only for standard-label files and is stored on the tape in the header label immediately preceding the data file that the label describes. If a date is specified, the data file is protected and cannot be overwritten until the specified expiration date.

*PERM: The data file is to be protected permanently. The date written in the tape data file labels consists of all nines.

expiration-date: Enter the expiration date on which the data file expires.

ENDOPT Parameter: Specifies, only when tape is used, what operation is to be automatically performed on the tape volume after the save operation ends. If more than one reel is involved, this parameter applies only to the last reel used.

REWIND: The tape is to be automatically rewound, but not unloaded, after the save operation has ended.

**LEAVE:* The tape is not to be rewound; another save operation can begin at the current position on the tape.

**UNLOAD:* The tape is to be automatically rewound and unloaded after the save operation has ended.

Example

```
SAVCHGOBJ  OBJ(ORD*)  LIB(DSTPRODLB)  OBJTYPE(*FILE)  +
      REFDATE(122281)
```

This command saves all changed files with names that begin with the characters ORD in library DSTPRODLB that have been changed since the 22nd of December, 1981. The files are to be copied on diskettes that are loaded in the magazines on the diskette magazine drive.

SAVLIB (Save Library) Command

The Save Library (SAVLIB) command saves a copy of a single library or all libraries, except for QSYS and other CPF-required libraries. The QGPL library and all user-created libraries can be saved. To save the specified libraries on offline storage, a copy of each library is written onto diskettes or tapes.

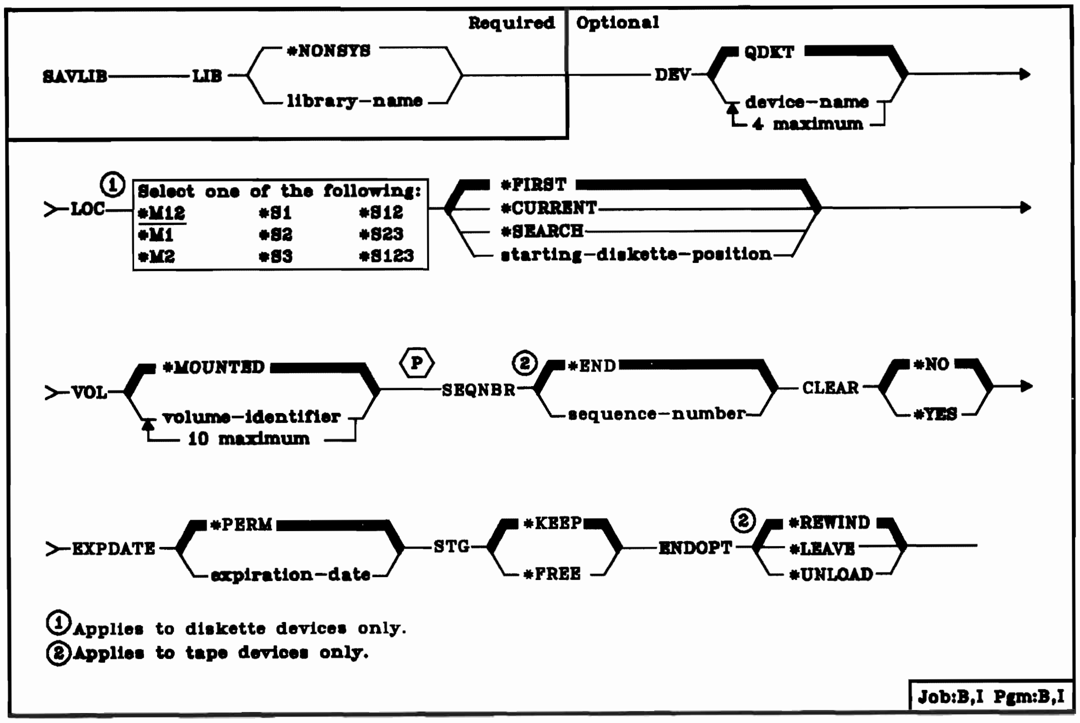
The SAVLIB command saves the entire library; this includes the library description, the object descriptions, and the contents of the objects in the library. It also includes PC (programming change) status information for all PCs in the library. The libraries and their objects are not affected in the system unless the command specifies that the storage is to be freed. However, the description of each library and each object is updated with the date, place, and time when it was last saved. If all user libraries are saved, the date, time, and place are updated in the history information for the data area QSAVLIBALL in QSYS.

The types of objects that are saved by this command are the same as those listed in the OBJTYPE parameter of the SAVOBJ command. Certain CPF objects that are not contained in user libraries (such as user profiles) are not saved by this command. They can be saved by the SAVSYS command.

Restrictions: To use this command, you must have either the special authority *SAVSYS specified in your user profile by the SPCAUT parameter, or you must have: (1) read rights for, or be the owner of, each library specified and (2) object existence rights for each object in the library. (If you do not have the proper rights for all of the libraries and objects specified, only those for which you do have the rights are saved.)

All diskettes in the magazines or slots to be used to save the libraries must have been initialized in the save/restore format. If tape is to be used, a standard labeled tape volume must be mounted. Also, no library being saved, nor the objects in it, can be updated by an executing job at the time the save operation occurs.

If a library being saved contains more than 8000 internal objects, the save operation terminates before any objects are saved, and an error message is sent to the system operator. For information on how many internal objects are saved for each CPF object type, refer to the *CPF Programmer's Guide*.



LIB Parameter: Specifies which library, or all user libraries, is to be saved on offline storage.

**NONSYS:* All libraries not saved by the SAVSYS (Save System) command are to be saved. *NONSYS specifies that all user-created libraries and the QGPL library are to be saved. All subsystems must be terminated, by the TRMSBS or TRMCPF command, before this option is specified.

library-name: Enter the name of the library that is to be saved. The libraries QSYS, QSRV, QTEMP, QSPL, and QRECOVERY cannot be specified.

DEV Parameter: Specifies the name of the device on which the library is to be saved. The device name must already be known on the system by a device description.

QDKT: The diskette device QDKT is to be the device used to save the library.

device-name: Enter the name of the diskette or tape device on which the library is to be saved. If you are using more than one tape device (up to a maximum of four), enter the names of the devices in the order they are to be used. If you are using more than one tape volume, you may want to use more than one tape device, so that one tape volume can rewind/unload while another tape device is processing.

LOC Parameter: Specifies, only when diskettes are used, which diskette locations on the diskette magazine drive are to be used to save the specified libraries. The data is saved (stored) on diskettes that are loaded either in the magazines or in the manual slots. If magazines are used, all the diskette positions to be used must contain diskettes initialized in the save/restore format. If all of the available space in the specified magazines or slots is used up before the save operation is finished, a message is issued to the operator to mount additional magazines or diskettes. (This parameter is ignored if a tape device is specified in the DEV parameter.)

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first value specifies which unit (magazines or slots) and location are to be used for the save operation. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be used first in the save operation. Enter one of the following values for the starting diskette position:

*FIRST: The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where a previous save or restore operation has just ended.) If the currently selected diskette is not in the location specified by the first value, a diagnostic message is sent to the system operator, and the command is terminated.

**SEARCH:* The diskettes in the location specified by the first value are to be searched for the first diskette having cleared space, where the save operation can begin. The search begins with the leftmost diskette in the specified location and ends with the first diskette on which cleared space is found.

starting-diskette-position: If a multiple diskette location (such as *M1 or *S23) is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10 for magazines, or 1 through 3 for manual slots) that contains the first diskette to be used.

VOL Parameter: Specifies the volume identifiers of the tape volumes (reels) or diskette volumes (either magazines or slots) on which the library data is to be saved. The volumes must be mounted in the same order as the volume identifiers are specified in this parameter. (For an expanded description of the VOL parameter, see Appendix A.)

*MOUNTED: The libraries are to be saved on whatever volumes are mounted on the device. For diskette, the volumes are those mounted on the diskette magazine drive in the locations specified by LOC. All diskettes or tapes to be used should already be cleared, unless CLEAR(*YES) is specified.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used to save the libraries. For tape volumes and single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the *magazine* identifier (5 characters maximum) can be specified for each volume. The save operation always continues with position 1 of every magazine after the first one.

SEQNBR Parameter: Specifies, only when tape is used, which sequence number is to be used as the starting point for the save/restore process.

*END: The system will save the file to the sequence number after the last sequence number on the tape.

file-sequence-number: Enter the sequence number of the file to be used as a starting point to save the library file(s).

CLEAR Parameter: Specifies whether or not uncleared diskettes or tapes encountered during the save operation are to be automatically cleared. (This parameter does not control initializing the diskettes or tapes; diskettes must already be initialized in the save/restore format; tapes must be standard labeled.)

If tapes are used and a sequence number is specified, the tape will be cleared starting with that sequence number; all tapes following the first tape will be cleared.

*NO: Uncleared diskettes or tapes encountered during the save operation are not to be automatically cleared. If, after the first diskette or tape used, any uncleared diskette or tape is encountered, an inquiry message is sent to the system operator, allowing him to terminate the save operation or to specify that the currently selected diskette or tape be cleared so the operation can continue. All diskettes or tapes to be used to save the libraries should be cleared before the SAVLIB command is issued.

**YES:* Uncleared diskettes or tapes encountered during the save operation are to be automatically cleared so the save can continue. For diskettes, *CURRENT or *SEARCH was specified on the LOC parameter for diskettes, this parameter applies only to uncleared diskettes following the diskette on which the save operation began. In these cases, the first diskette used was already clear (or had sufficient space to begin the save operation), because it was the diskette at which the *last* save operation ended (*CURRENT), or it was found to be clear (or to have sufficient space) by *SEARCH in this operation. If *FIRST or a starting diskette position was specified for LOC, this parameter applies to all diskettes (including the first one) used in the operation.

EXPDATE Parameter: Specifies the expiration date of the save/restore file. If a date is specified, the save/restore file is protected and cannot be overwritten until the specified expiration date. This parameter is valid for tape and diskette for all save/restore commands.

*PERM: The save/restore file is to be protected permanently.

expiration-date: Enter the date on which the save/restore file expires.

STG Parameter: Specifies whether the system storage that is occupied by the data portion of files and programs in the library being saved is to be freed as part of the save operation. Only the data portion of the objects is ever freed, not the descriptions of the objects.

*KEEP: The storage occupied by the data portion of the objects being saved is not to be freed.

**FREE:* The storage occupied by the data portion of the files and programs being saved is to be freed as part of the save operation. The storage for all the objects in a library is freed only after all the objects in that library have been saved successfully.

ENDOPT Parameter: Specifies, only when tape is used, what operation is to be automatically performed on the tape volume after the save operation ends. If more than one reel is involved, this parameter applies only to the last reel used.

*REWIND: The tape is to be automatically rewound, but not unloaded, after the save operation has ended.

**LEAVE:* The tape is not to be rewound; another save operation can begin at the current position on the tape.

**UNLOAD:* The tape is to be automatically rewound and unloaded after the save operation has ended.

Examples

```
SAVLIB  LIB(JOE) CLEAR(*YES)
```

This command saves the library named JOE on whatever diskette magazine volume is mounted as magazine 1 (and 2, if needed) on the magazine diskette drive, because the DEV, LOC, and VOL parameters were not specified. The storage occupied by JOE in the system will not be freed.

```
SAVLIB  LIB(QGPL) LOC(*M12 4) VOL(ABC DEF GHI)
```

The general purpose library (QGPL) is to be saved on diskettes mounted in magazines 1 and 2. Volume ABC must be mounted as magazine 1, and DEF must be mounted as magazine 2. The save operation is to begin on diskette 4 in volume ABC and continue on diskettes 5 through 10; then diskettes 1 through 10 of volume DEF are used. If the save operation is not finished when diskette 10 of volume DEF is full, a message is issued to the operator. Volume GHI must be mounted as magazine 1 to complete the operation.

```
SAVLIB  LIB(CUSTDATA) LOC(*S23) VOL(CUSNM CUSAD) +
          STG(*FREE)
```

The library CUSTDATA is to be saved on volumes CUSNM and CUSAD, which are mounted in manual slots 2 and 3 of the diskette magazine drive. The storage occupied by the files and programs in the CUSTDATA library is to be freed after it is saved.

```
SAVLIB  LIB(QGPL) DEV(QTAPE1) VOL(BKUP14) +
          ENDOPT(*LEAVE)
```

This command uses the tape device named QTAPE1 to save the QGPL library on tape. The tape volume named BKUP14, used to store the contents of the library, is not rewound after the library has been saved; another save command could be entered to save more data on the tape.

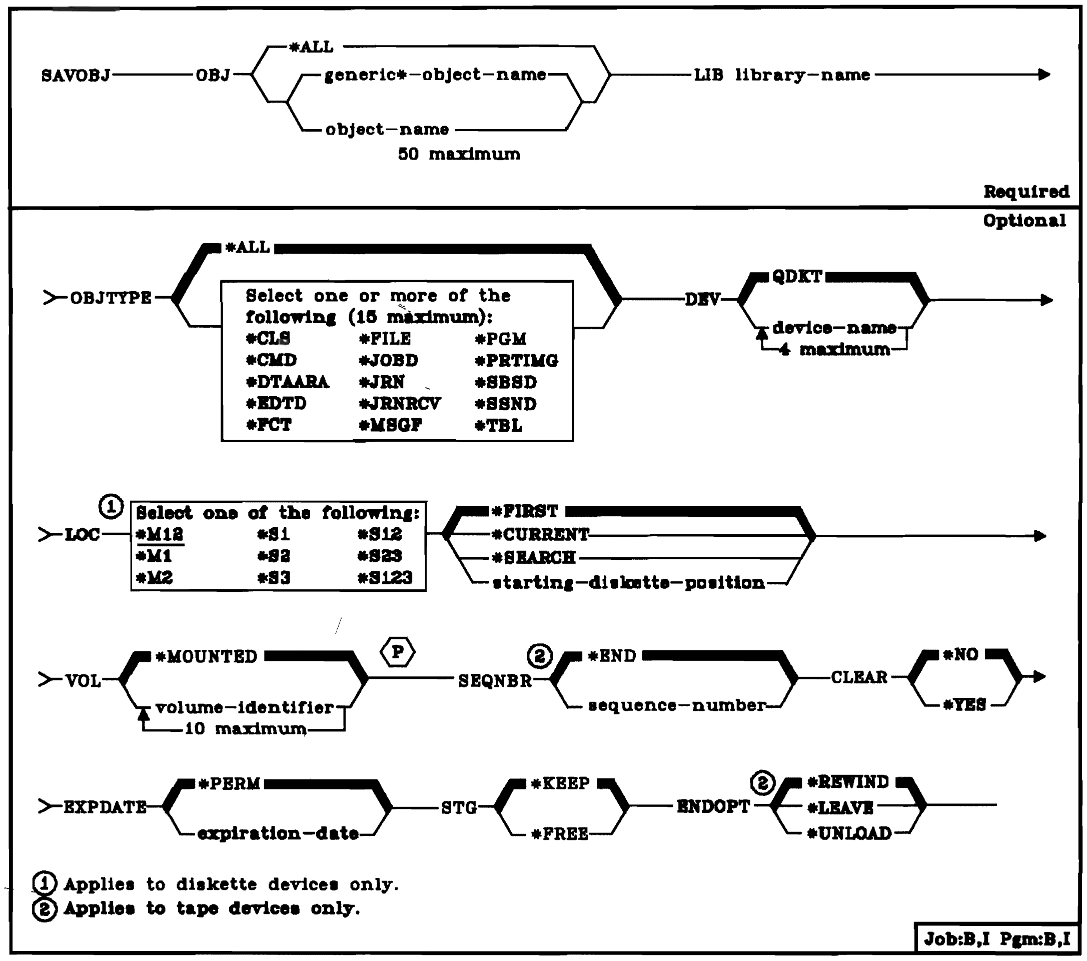
SAVOBJ (Save Object) Command

The Save Object (SAVOBJ) command saves a copy of a single object or a group of objects located in the same library. The types of objects that can be saved by this command are listed in the OBJTYPE parameter. The system saves the specified objects on offline storage by writing a copy of each one onto diskettes or tapes. The objects are not affected in the system unless the command specifies that the storage is to be freed. However, the description of each object is updated with the date, time, and place when it was last saved.

Restrictions: To use this command, you must have either the special authority *SAVSYS specified in your user profile by the SPCAUT parameter, or you must have: (1) object existence rights for each object specified, and (2) read rights for the specified library. (If you do not have the necessary rights to a specified object, all objects except that one are saved.)

All diskettes in the magazines or slots to be used to save the objects must have been initialized in the save/restore format. If tape is to be used, a standard labeled volume must be mounted. Also, no object being saved can be updated by an executing job that is executing at the time the save operation occurs.

If more than 8000 internal objects are to be saved from the specified library, the save operation terminates before any objects are saved, and an error message is sent to the system operator. This can occur if the library contains more than 8000 internal objects and OBJ(*ALL) is specified, or if the library contains more than 8000 objects whose generic names and object types are specified by the OBJ and OBJTYPE parameters. For information on how many internal objects are saved for each CPF object type, refer to the *CPF Programmer's Guide.*



OBJ Parameter: Specifies the names of one or more objects that are to be saved. All the objects must be in the library specified in the LIB parameter. Enter the name of each object or the generic name of each group of objects to be saved.

If the OBJTYPE parameter is not specified, all the object types listed in the description of the OBJTYPE parameter are saved, provided they are in the specified library and have the specified names.

**ALL:* All the objects in the specified library are to be saved, depending on the values specified for OBJTYPE.

generic-object-name: Enter one or more generic names of groups of objects in the specified library that are to be saved.

object-name: Enter one or more names of specific objects that are to be saved. (Both generic names and specific names can be specified in the same command.)

LIB Parameter: Specifies which library contains the objects that are to be saved on offline storage. Enter the name of the library.

OBJTYPE Parameter: Specifies the types of CPF objects that are to be saved.

*ALL: All object types that are specified by name and are in the specified library are to be saved. If *ALL is also specified on the OBJ parameter, then all the objects in the library that are of the types shown in the following table are saved.

object-type: Enter the value for each of the types of objects that are to be saved. The following types can be specified.

Value	Type of CPF Object
*CLS	Class
*CMD	Command
*DTAARA	Data area
*EDTD	Edit description
*FCT	Forms control table
*FILE	File
*JOBD	Job description
*JRN	Journal
*JRNRCV	Journal receiver
*MSGF	Message file
*PGM	Program
*PRTIMG	Print image
*SBSD	Subsystem description
*SSND	Session description
*TBL	Table

The object types shown are also the ones that are saved and restored by the SAVLIB and the RSTLIB commands.

DEV Parameter: Specifies the name of the device on which the objects are to be saved. The device name must already be known on the system by a device description.

QDKT: The diskette device QDKT is to be the device used to save the objects.

device-name: Enter the name of the diskette or tape device on which the library is to be saved. If you are using more than one tape device (up to a maximum of four), enter the names of the devices in the order they are to be used. If you are using more than one tape volume, you may want to use more than one tape device, so that one tape volume can rewind/unload while another tape device is processing.

LOC Parameter: Specifies, only when diskettes are used, which diskette locations on the diskette magazine drive are to be used to save the specified objects. The data is saved (stored) on diskettes that are loaded either in the magazines or in the manual slots. If magazines are used, all the diskette positions to be used must contain diskettes initialized in the save/restore format. If all of the available space in the specified magazines or slots is filled before the save operation is finished, a message is issued to the operator to mount additional magazines or diskettes. (This parameter is ignored if a tape device is specified in the DEV parameter.)

Two values can be specified for the LOC parameter: (1) the unit type and location, and (2) the starting diskette position. (For an expanded description of the LOC parameter, see Appendix A.)

Unit Type and Location: The first value specifies which unit (magazines or slots) and location are to be used for the save operation. Enter one of the following values to specify the unit type and location: *M12*, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second value in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be used first in the save operation. Enter one of the following values for the starting diskette position:

FIRST: The first diskette position in the specified location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where a previous save or restore operation has just ended.) If the currently selected diskette is not in the location specified by the first value, a diagnostic message is sent to the system operator, and the command is terminated.

**SEARCH:* The diskettes in the location specified by the first value are to be searched for the first diskette having cleared space, where the save operation can begin. The search begins with the leftmost diskette in the specified location and ends with the first diskette on which cleared space is found.

starting-diskette-position: If a multiple diskette location (such as *M1 or *S23) is specified as the first value in the LOC parameter, enter the number of the diskette position (1 through 10 for magazines, or 1 through 3 for manual slots) that contains the first diskette to be used.

VOL Parameter: Specifies the volume identifiers of the tape volumes (reels) or diskette volumes (either magazines or slots) on which the object data is to be saved. The volumes must be mounted in the same order as the volume identifiers are specified in this parameter. (For an expanded description of the VOL parameter, see Appendix A.)

*MOUNTED: The objects to be saved are copied on whatever volumes are mounted on the device. For diskettes, the volumes are those mounted on the diskette magazine drive in the location specified by LOC. All diskettes to be used in the mounted volumes specified by LOC should already be cleared, unless CLEAR(*YES) is specified.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used to save the objects. For tape volumes and single-diskette volumes (placed in manual slots), a maximum of 6 characters identify each volume. For magazine volumes, only the *magazine* identifier (5 characters maximum) can be specified for each volume. The save operation always continues with position 1 of every magazine after the first one.

SEQNBR Parameter: Specifies, only when tape is used, which sequence number is to be used for saving the objects.

*END: The system will save the objects at the sequence number immediately following the last position on the tape.

file-sequence-number: Enter the sequence number of the file to be used to save the objects.

CLEAR Parameter: Specifies, only when diskettes are used, whether or not uncleared diskettes encountered during the save operation are to be automatically cleared. (This parameter does not control initializing the diskettes; they must already be initialized in the save/restore format.)

If tapes are used and a sequence number is specified, the tape will be cleared starting with that sequence number; all tapes following the first tape will be cleared.

*NO: Uncleared diskettes encountered during the save operation are not to be automatically cleared. If, after the first diskette used, any uncleared diskette is encountered, an inquiry message is sent to the system operator, allowing him to terminate the save operation or to specify that the currently selected diskette be cleared so the operation can continue. All diskettes to be used to save the objects should be cleared before the SAVOBJ command is issued.

**YES:* Uncleared diskettes encountered during the save operation are to be automatically cleared so the save can continue. If *CURRENT or *SEARCH was specified on the LOC parameter, this parameter applies only to uncleared diskettes following the diskette on which the save operation began. In these cases, the first diskette used was already clear (or had sufficient space to begin the save operation), because it was the diskette at which the *last* save operation ended (*CURRENT), or it was found to be clear (or to have sufficient space) by *SEARCH in this operation. If *FIRST or a starting diskette position was specified for LOC, this parameter applies to all diskettes (including the first one) used in the operation.

EXPDATE Parameter: Specifies the expiration date of the save/restore files. If a date is specified, the save/restore file is protected and cannot be overwritten until the specified expiration date.

*PERM: The save/retore file is to be protected permanently.

expiration-date: Enter the date on which the save/restore file expires.

STG Parameter: Specifies whether the system storage that is occupied by the data portion of the file and program objects being saved is to be freed after the save operation is finished. Only the data portion of the object is freed, not the object's description.

*KEEP: The storage occupied by the data portion of the objects being saved is not to be freed.

**FREE:* The storage occupied by the data portion of the file and program objects is to be freed as part of the save operation after the objects have been saved on diskette.

ENDOPT Parameter: Specifies, only when tape is used, what operation is to be automatically performed on the tape volume after the save operation ends. If more than one reel is involved, this parameter applies only to the last reel used.

*REWIND: The tape is to be automatically rewound, but not unloaded, after the save operation has ended.

**LEAVE:* The tape is not to be rewound; another save operation can begin at the current position on the tape.

**UNLOAD:* The tape is to be automatically rewound and unloaded after the save operation has ended.

Examples

```
SAVOBJ  OBJ(PETE)  LIB(LIBX)  LOC(*S1)
```

This command saves the object named PETE located in LIBX on the diskette placed in manual slot 1 of the diskette magazine drive. If, for example, LIBX contains both a program and a file named PETE, both objects are saved. The storage occupied by the object is not freed because the STG parameter default (*KEEP) was assumed.

```
SAVOBJ  OBJ(MSTRPAY PAY*)  LIB(QGPL)  STG(*FREE)
```

The object named MSTRPAY and all the objects whose names start with the characters PAY, located in the general purpose library (QGPL), are to be saved. The objects are to be copied on diskettes that are loaded in the magazines on the diskette magazine drive. As part of the save operation, the system storage that was occupied by the data portion of the saved file and program objects is freed.

```
SAVOBJ  OBJ(FILEA)  OBJTYPE(*FILE)  LIB(LIBY)  LOC(*S1)  +
      VOL(TOM)  CLEAR(*YES)
```

The file named FILEA in library LIBY is to be saved on the diskette placed in slot S1; the diskette must have the volume identifier TOM on it. If the diskette has not been cleared, it is to be cleared automatically before FILEA is saved on it.

SAVSYS (Save System) Command

The Save System (SAVSYS) command saves a copy of the CPF-required libraries, including QSYS. (It does not save the QGPL library or utility libraries such as QRPG, QIDU, or QS3E.) It also saves all of the following objects: user profiles (including the object authorizations granted to each profile), edit descriptions, and all the descriptions of devices, control units, and lines.

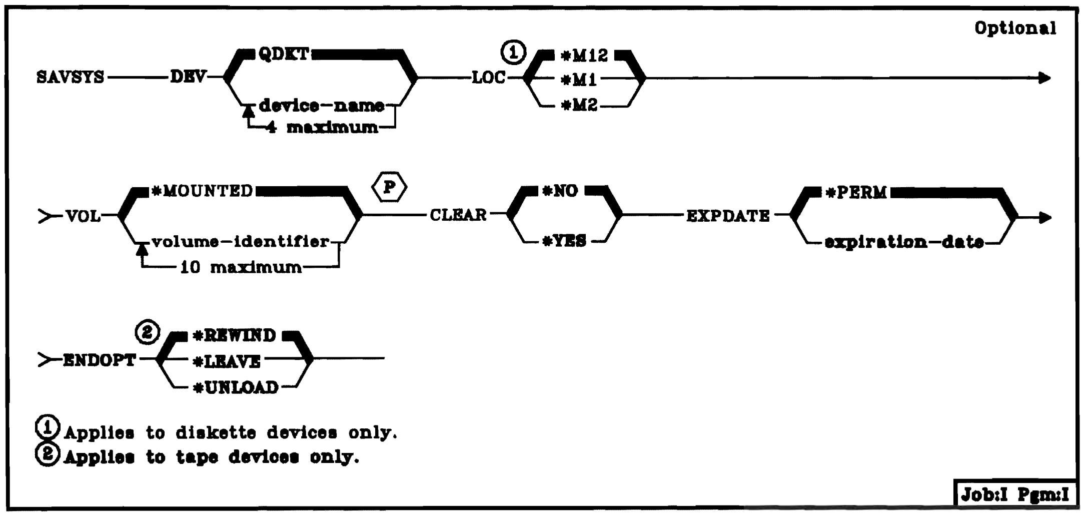
To save the system data on offline storage, the system writes a copy of all objects to be saved onto diskettes or tapes. (When saving the system using tape, the initial CPF installation programs are written onto a diskette. This diskette must be cleared or must contain only expired files before being used for a save system operation, and it must be mounted in slot *S1.) The libraries and objects are not affected in the system; this command cannot be used to free any of the space occupied by these objects. However, the history information for the data area QSAVUSRPRF in QSYS is updated with the date, time, and place where user profiles were saved, and the history information for the data area QSAVSYS in QSYS is updated with the date, time, and place where the system was saved. To display the information in these data areas, enter the DSPOBJD command and specify DETAIL(*FULL). Save the information from the display of QSAVUSRPRF for the location (volume name, slot number of diskette, or sequence number for tape) where the user profiles were saved.

All subsystems must be terminated, through the TRMSBS command, before this command is used.

The QSYSOPR message queue should be in break delivery mode when saving the system. For a sample CL program that will reset this queue, refer to the *Programmer's Guide*.

For information on the procedures used to restore a saved system, refer to the *IBM System/38 Operator's Guide*, SC21-7735.

Restrictions: (1) To use this command, you must have the save system rights in your user profile, specified by SPCAUT(*SAVSYS). (2) If diskettes are to be used, all diskettes in the magazine(s) used to save the system must have been initialized in the save/restore format. (The system save always begins with the first diskette position.) (3) If tape is to be used, a standard-labeled volume must be mounted, and a clear diskette initialized in the save/restore format must be mounted in manual slot *S1 of the diskette device. (4) The system cannot be executing any other jobs during a save system operation.



DEV Parameter: Specifies the name of the diskette or tape device on which the system data is to be saved. The device name must already be known on the system by a device description. For tape devices only, a maximum of four devices may be specified.

QDKT: The diskette device QDKT is to be the device used to save the system data.

device-name: Enter the name of the diskette device or tape devices (maximum of four) on which the system data is to be saved.

LOC Parameter: Specifies, when using diskettes for saving system data, whether magazine 1, magazine 2, or both are to be used to save the system. The data is saved (stored) on diskettes that are loaded in the magazines. All the diskette positions in the magazines must contain initialized diskettes. If all available space in the specified magazine(s) is filled before the save operation is finished, a message is issued to the operator to mount another magazine. (For an expanded description of the LOC parameter, see Appendix A.)

One of the following values can be entered to specify which location is to be used in the save operation.

Value	**Location Used**
*M12	Both magazines 1 and 2
*M1	Magazine 1 only
*M2	Magazine 2 only

If both magazines are to be used, the system begins with the first diskette in magazine 1.

VOL Parameter: Specifies the volume identifiers of the tape volumes (reels) or diskette volumes on which the system data is to be saved. The volumes must be mounted in the same order as the volume identifiers are specified in this parameter. All diskettes to be used in the mounted volumes specified by LOC should already be cleared, unless CLEAR(*YES) is specified. (For an expanded description of the VOL parameter, see Appendix A.)

MOUNTED: The system data is to be saved on whatever volumes are mounted on the device. For diskette, the volumes are those mounted on the diskette magazine drive in the locations specified by LOC.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used to save the system data. For tape, a maximum of 6 characters identify each volume. For magazine volumes, only the magazine identifier (5 characters maximum) can be specified for each volume. The save operation always continues with position 1 of every magazine after the first one.

CLEAR Parameter: Specifies whether uncleared diskettes or tape volumes encountered during the save operation are to be automatically cleared. (This parameter does not control initializing diskettes or tapes; diskettes must already be initialized in the save/restore format; tapes must be standard labeled.)

NO: Uncleared diskettes or tape volumes encountered during the save operation are not to be automatically cleared. If any uncleared tape volume or diskette is encountered, an inquiry message is sent to the system operator, allowing him to terminate the save operation or to specify that the currently selected tape or diskette be cleared so the operation can continue. All diskettes or tapes to be used to save the system should be cleared before the SAVSYS command is issued.

**YES:* Uncleared diskettes or tape volumes encountered during the save operation are to be automatically cleared so the save can continue. If *YES is specified, this parameter applies to all diskettes or tape volumes used in the save operation, including the first one. CLEAR(*YES) does not apply for the initial-program diskette used when saving the system on tape.

EXPDATE Parameter: Specifies the expiration date of the save/restore files. If a date is specified, the file is protected and cannot be overwritten until the specified expiration date.

PERM: The save/restore file is to be protected permanently.

expiration-date: Enter the date on which the save/restore file expires.

ENDOPT Parameter: Specifies the positioning operation to be performed automatically on the tape volume when the save operation has been completed. In the case of a multiple-volume save operation, this parameter applies to the *last* reel only; all other reels are rewound and unloaded when the end of the tape is reached.

REWIND: The tape is to be rewound, but not unloaded, when the save operation has been completed.

**LEAVE:* The tape should be left in its current position when the operation has been completed; it is not to be rewound or unloaded.

**UNLOAD:* The tape is to be rewound and unloaded when the save operation has been completed.

Examples

```
SAVSYS  CLEAR(*YES)
```

This command saves the system libraries, all user profiles (including private authority for objects), and all line, control unit, and device descriptions. They are saved on the diskette device named QDKT, starting with diskette position 1 of the magazine volume that is mounted as magazine 1. Each uncleared diskette is cleared automatically when it is encountered, and the save operation continues without operator intervention.

```
SAVSYS  LOC(*M2) VOL(ABCDE)
```

The system libraries and the system objects not contained in libraries are saved on the diskette device QDKT. The save begins on the volume labeled ABCDE, which must be the first diskette mounted in magazine 2. If the save operation exceeds the storage capacity of one magazine, a message requesting that another volume be mounted at magazine 2 is displayed to the operator.

```
SAVSYS  DEV(QTAPE1 QTAPE2) VOL(SYSTPA SYSTPB)
```

The initial-program diskette required by the installation procedure is saved to a diskette mounted in manual slot *S1. The save of CPF-required libraries and objects will continue to the tape volume SYSTPA mounted on tape device QTAPE1. If this tape volume is filled, the tape will be rewound and unloaded; the save of the system will continue will volume SYSTPB mounted on tape device QTAPE2. When the system has been saved, volume SYSTPB will be rewound (but not unloaded).

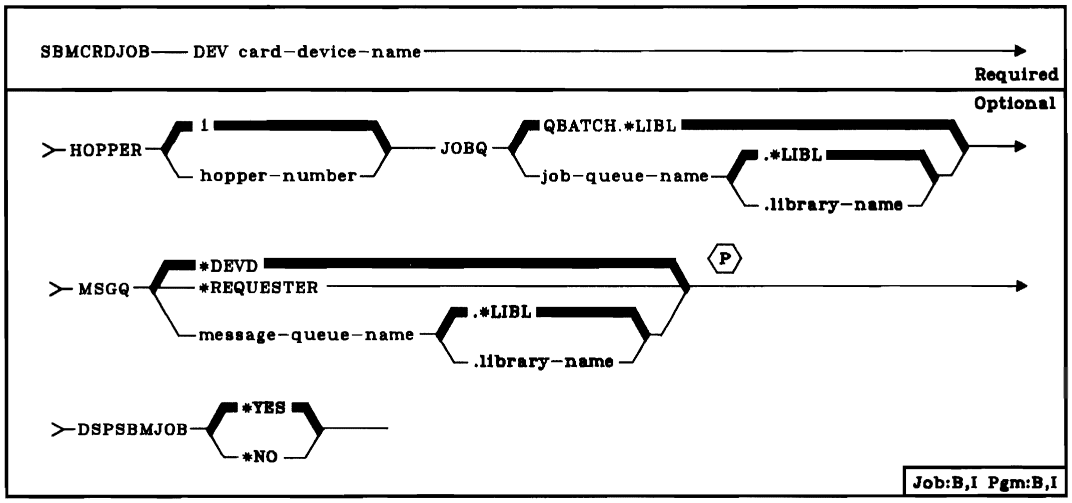
SBMCRDJOB (Submit Card Jobs) Command

The Submit Card Jobs (SBMCRDJOB) command allows an executing job to submit card job streams to job queues to be executed as batch jobs. This command specifies the name of the card device from which the cards are read, the hopper number, the names of the job queue and message queue to be used, and whether jobs being submitted can be displayed by the Display Submitted Jobs (DSPSBMJOB) command.

The Submit Card Jobs operation reads cards from the card device until an ENDINP command (delimiter) is encountered. The ENDINP command (delimiter) is not recognized if it is within an inline file that ends with nondefault ending characters (as specified in the ENDCHAR parameter of the DATA command). A /* card should follow the //ENDINP statement to ensure that the command is read. The SBMCRDJOB operation can be canceled either by canceling the request from the system request menu or by canceling the job in which the operation is running.

In contrast to a spool reader started with the STRCRDRDR (Start Card Reader) command, the SBMCRDJOB command operates in the same process as the requesting function and does not do syntax checking on the job stream.

Restrictions: To use this command you must have operational authority for the job descriptions (specified by the JOB commands) read by the submit-jobs function. You must also be authorized to use the device, message queue and job queue.



DEV Parameter: Specifies the name of the card device to be used to read cards in a job stream.

HOPPER Parameter: Specifies which hopper of the MFCU is to be used for input data. The value for the primary hopper is 1 and the value for the secondary hopper is 2.

1: The primary hopper is to be used for input data.

hopper-number: Enter either a 1 or a 2 to indicate which MFCU hopper is to be used.

JOBQ Parameter: Specifies the name of the job queue on which the job entries are to be placed. An entry is placed on this job queue for each job in the job stream that specifies JOBQ(*RDR) on its JOB command. If *RDR is not specified on the JOB command, the job queue specified in the JOB command or in the job description is used. (Note that the job queue for each job in the job stream can be different.)

Restriction: If the user identified in the job description of the job being read is not authorized to use the job queue on which the job should be placed, that job terminates, and a diagnostic message that describes the cause for termination is placed in the job log. The job stream, however, continues to be processed, starting with the next job.

QBATCH.*LIBL: The job entry is to be placed on the QBATCH job queue, which is the default job queue if JOBQ(*RDR) is specified on the JOB command.

job-queue-name: The qualified name of the job queue to which each job in the job stream is to be sent if JOBQ(*RDR) is specified on the JOB command. If no library qualifier is given, *LIBL is used to find the queue.

MSGQ Parameter: Specifies the name of the message queue to which operational messages are to be sent.

*DEVD: The messages are to be sent to the message queue indicated in the device description.

**REQUESTER:* The messages are to be sent to the message queue of the user who started the process. This value is ignored for batch jobs.

message-queue-name: Enter the qualified name of the message queue to which operational messages are to be sent. If no library qualifier is given, *LIBL is used to find the queue.

DSPSBMJOB Parameter: Specifies whether the jobs being submitted are allowed to be displayed on the submitted jobs display. Any submitted job of the type specified by the SBMFROM parameter of the DSPSBMJOB command can be displayed if *YES is specified on this parameter.

YES: This job can be displayed by the DSPSBMJOB command.

**NO:* This job is not to be displayed on any display produced by the DSPSBMJOB command.

Example

```
SBMCRDJOB  DEV(MFCU1)
```

This command submits card jobs using data cards from the device named MFCU1. The cards containing the job stream are read from hopper 1 of the MFCU. The job entries are placed on the default job queue QBATCH for jobs whose JOB command specifies *RDR for the job queue name. Operational messages are sent to the message queue defined by the device.

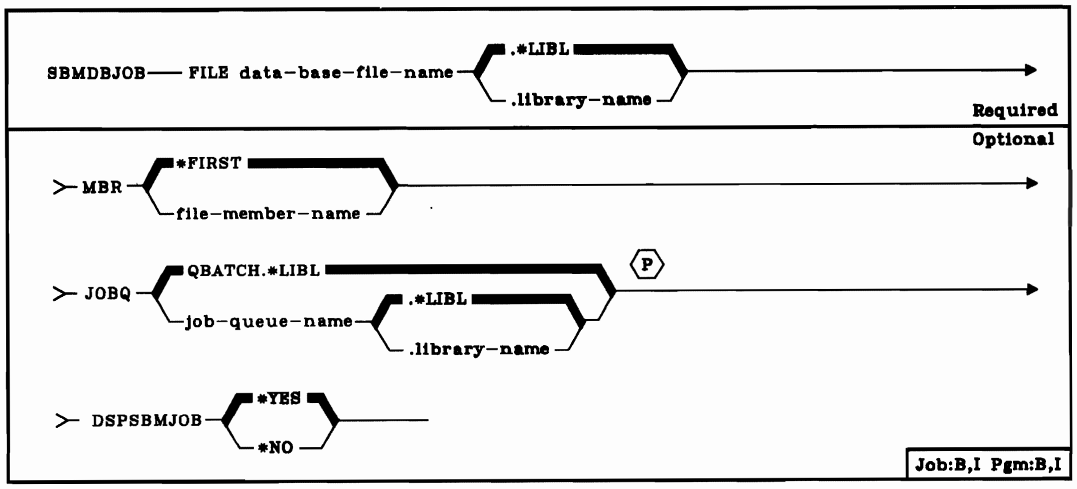
SBMDBJOB (Submit Data Base Jobs) Command

The Submit Data Base Jobs (SBMDBJOB) command allows an executing job to submit other jobs to job queues to be executed as batch jobs. The job stream is read either from a physical data base file or from a logical data base file in single-record format. This command specifies the name of the data base file and member from which the jobs are read, the name of the job queue to be used if the JOB command has JOBQ(*RDR) specified, and whether jobs being submitted can be displayed by the Display Submitted Jobs (DSPSBMJOB) command.

A Submit Data Base Jobs operation will read the file once and terminate when the end-of-file is read or when an ENDINP (End Input) command (delimiter) is encountered. The ENDINP command (delimiter) is not recognized if it is within an inline file that ends with nondefault ending characters (as specified in the ENDCHAR parameter of the DATA command). The SBMDBJOB operation can be canceled either by canceling the request from the system request menu or by canceling the job in which the process is running.

In contrast to a spool reader started with the STRDBRDR (Start Data Base Reader) command, the SBMDBJOB command operates in the same process as the requesting function and does not do syntax checking on the job stream.

Restrictions: The specified data base file must consist of single field records and must have an arrival sequence access path, or it must be a standard data base source file. To read the specified data base file, you must have read rights for the file. You must be authorized to use the job queue, and you must also have operational authority for the job descriptions (specified by JOB commands) read by the submit jobs function.



FILE Parameter: Specifies the qualified name of the data base file from which the job stream is to be read. If no library qualifier is specified, *LIBL is used to find the file.

MBR Parameter: Specifies the name of the member in the specified file that contains the job stream to be read.

*FIRST: No member name is specified; the first member in the file is to be used.

file-member-name: Enter the name of the member that contains the job stream to be read.

JOBQ Parameter: Specifies the name of the job queue on which the entries are to be placed. An entry is placed on this queue for each job in the job stream that specifies JOBQ(*RDR) on its JOB command. If *RDR is not specified on the JOB command, the job queue specified in the JOB command or in the job description is used. (Note that the job queue for each job in the job stream can be different.)

Restriction: If the user identified in the job description of the job being read is not authorized to the job queue on which the job should be placed, that job terminates, and a diagnostic message that describes the cause for termination is placed in the job log. The job stream, however, continues to be processed, starting with the next job.

QBATCH.*LIBL: The job entry is to be placed on the QBATCH job queue, which is the default job queue if JOBQ(*RDR) is specified on the JOB command.

job-queue-name: The qualified name of the job queue to which each job in the job stream is to be sent if JOBQ(*RDR) is specified on the JOB command. If no library qualifier is given, *LIBL is used to find the queue.

DSPSBMJOB Parameter: Specifies whether the jobs being submitted are to be displayed on the submitted jobs display. Any submitted job of the type specified by the SBMFROM parameter of the DSPSBMJOB command can be displayed if *YES is specified on this parameter.

YES: This job can be displayed by the DSPSBMJOB command.

**NO:* This job is not to be displayed on any display produced by the DSPSBMJOB command.

Example

```
SBMDBJOB  FILE(BILLING.QGPL)
```

This command submits jobs using input from the data base file named BILLING, which is in the QGPL library. The first member in the BILLING file contains the job stream to be processed. The default job queue QBATCH is used.

SBMDKTJOB (Submit Diskette Jobs) Command

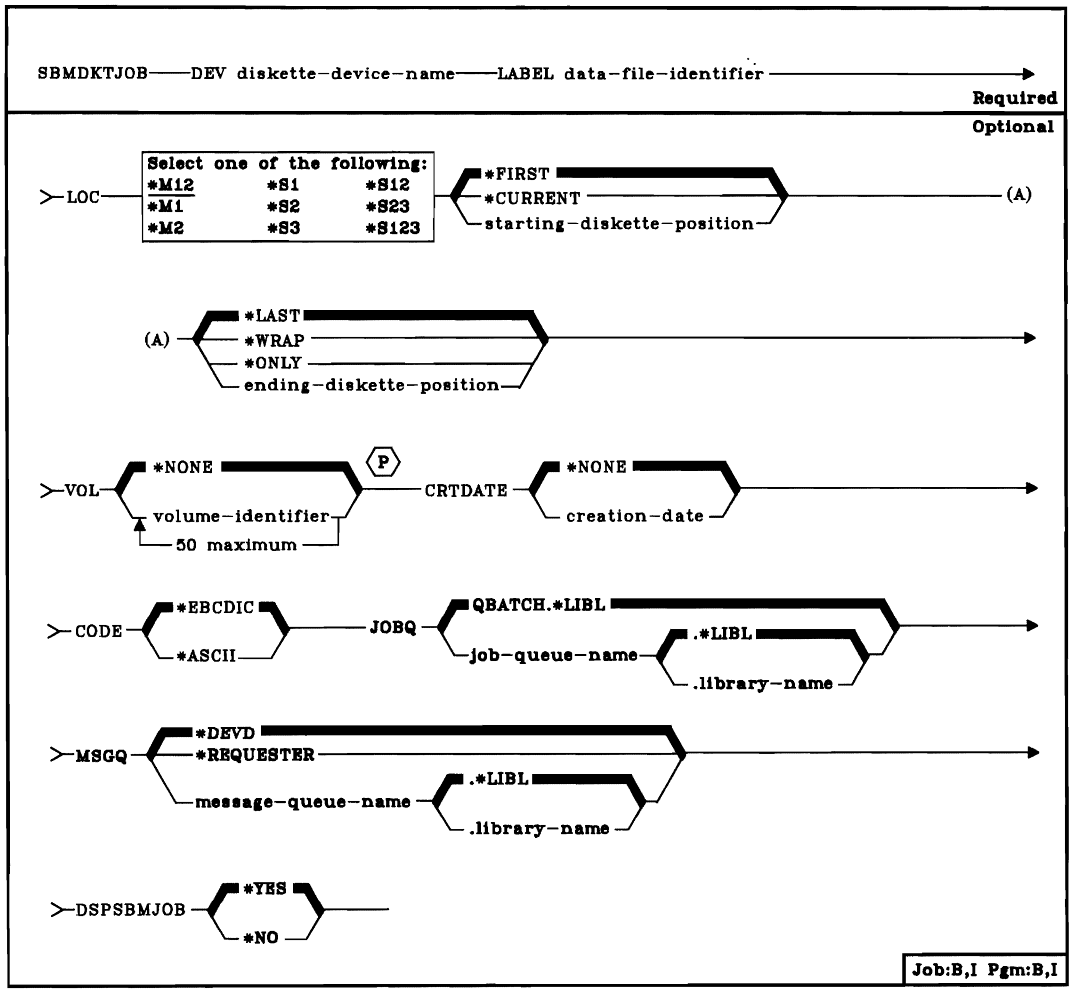
The Submit Diskette Jobs (SBMDKTJOB) command allows an executing job to submit other jobs to job queues to be executed as batch jobs. The job stream is read from a diskette. This command specifies the name of the diskette device, the label identifier, the location, volume, creation date, and interchange code type of the file containing the job stream, the names of the job queue and message queue to be used, and whether jobs being submitted can be displayed by the Display Submitted Jobs (DSPSBMJOB) command.

A Submit Diskette Jobs operation will read the file once and terminate when the end of file is read or when an ENDINP (End Input) command (delimiter) is encountered. The ENDINP command (delimiter) is not recognized if it is within an inline file that ends with nondefault ending characters (as specified in the ENDCHAR parameter of the DATA command). The SBMDKTJOB operation can be canceled by either canceling the request from the system request menu or by canceling the job in which the operation is running.

In contrast to a spool reader started with the STRDKTRDR (Start Diskette Reader) command, the SBMDKTJOB command operates in the same process as the requesting function and does not do syntax checking on the job stream.

Restrictions: To use this command, you must have operational authority for the job descriptions (specified by JOB commands) read by this function. You must also be authorized to the device, message queue and job queue.

Note: This command cannot be used to read data files of diskettes that are in the save/restore format.



DEV Parameter: Specifies the name of the diskette device to be used to read the job stream.

LOC Parameter: Specifies which diskette locations in the magazines or slots of the diskette magazine drive are to be used in diskette input/output operations. Only one of the two types of units can be used in any diskette I/O operation. One or more of the diskette locations in either unit type can be used, depending on the values specified in this parameter. For an expanded description of the LOC parameter, refer to Appendix A.

Note: In the LOC parameter, the tenth diskette position of a magazine is specified as 10.

Unit Type and Location: One of the following values can be entered as the first element to specify which unit and location is to be used in the submit diskette jobs function.

Value	**Location Used**
*M12	Both magazines 1 and 2
*M1	Magazine 1 only
*M2	Magazine 2 only
*S1	Manual slot 1 only
*S2	Manual slot 2 only
*S3	Manual slot 3 only
*S12	Manual slots 1 and 2
*S23	Manual slots 2 and 3
*S123	Manual slots 1, 2, and 3

Starting Diskette Position: One of the following values can be entered as the second element to specify in which diskette position (in the location specified by the first element) the submit jobs function is to start. The starting position contains the diskette that has the beginning of the data file on it.

For magazines, the starting diskette position can be specified for the first magazine only. For all following magazines, the operation always begins with the leftmost diskette position of the units specified.

Note: The first diskette that is read must contain the data file.

*FIRST: The first diskette position in the location contains the diskette to be used first. It is the leftmost diskette in the magazine(s) or slots specified.

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where the previous operation has just ended.) If the currently selected diskette is not in any slot within the range specified by the unit identified in the first value, *FIRST is used instead of *CURRENT.

starting-diskette-position: The number of the diskette position (1 through 10) in the magazine that contains the first diskette to be used.

Ending Diskette Position: The third element in the list specifies (1) the last diskette to be used and (2) what action is to occur if the end of the volume on that diskette is reached before the end of the data file is reached.

One of the following values can be entered to specify the ending diskette to be used in the operation.

**LAST:* The last diskette position in the location contains the diskette to be read last. It is the rightmost diskette position in the magazine(s) or slots specified. If the end of the volume (of the last diskette) is reached before the end of the data file, an error message is sent to the system operator and the command terminates.

**WRAP:* If the end of the last diskette in the location is reached before the end of the data file is reached, a message is sent to the system operator requesting that another volume be mounted. The message sent depends on which location was specified by the first element, as shown in the following chart:

First List Element	Meaning of *WRAP
*M12	Mount a new pair of magazines and continue processing diskette 1 of magazine 1.
*M1	Mount a new magazine for magazine 1 and continue processing diskette 1.
*M2	Mount a new magazine for magazine 2 and continue processing diskette 1.
*S1	Insert a new diskette in slot 1 and continue processing.
*S2	Insert a new diskette in slot 2 and continue processing.
*S3	Insert a new diskette in slot 3 and continue processing.
*S12	Insert new diskettes in slots 1 and 2 and continue processing at slot 1.
*S23	Insert new diskettes in slots 2 and 3 and continue processing at slot 2.
*S123	Insert new diskettes in slots 1, 2, and 3 and continue processing at slot 1.

**ONLY:* The diskette position specified by the second element in the list is to be used once. A second diskette cannot be mounted in that position to continue the operation.

ending-diskette-position: Enter the number of the diskette position (within the range specified by the first element) that contains the last diskette to be used. If both magazines (*M12) are being used, the ending diskette position is in magazine 2. For manual slots, this number may be 1 through 3, depending on what is specified for the first element. For a magazine, this number may be 1 through 10.

VOL Parameter: The volume (VOL) parameter specifies the volume identifiers of the volumes to be used in a diskette operation. A volume may consist of a single diskette within a magazine or slot, or multiple diskettes within the same magazine. Normally, a volume is a single diskette placed in a manual slot or 10 diskettes placed in a magazine. For an expanded description of the VOL parameter, refer to Appendix A.

*NONE: No volume identifiers are to be specified; the current volume is to be used.

volume-identifier: The identifiers of one or more volumes are in the order in which they are to be mounted and used. Each volume identifier contains a maximum of six characters. A blank is used as the separator character when listing multiple identifiers.

CRTDATE Parameter: Specifies when the diskette data file was created on diskette; the creation date should not be specified if the date is not to be checked. If the date written on the diskette containing the data file does not match the date specified here, an error message is sent to the message queue named in the MSGQ parameter.

*NONE: The creation date is not specified; no check is to be made.

creation-date: Enter the creation date of the data file to be read. The date should be in the format specified by the QDATFMT system value.

CODE Parameter: Specifies the type of character code that is to be used by the diskette device.

*EBCDIC: The EBCDIC character code is to be used.

ASCII: The ASCII character code is to be used.

JOBQ Parameter: Specifies the name of the job queue on which the job entries are to be placed. An entry is placed on this job queue for each job in the job stream that specifies JOBQ(*RDR) on its JOB command. If *RDR is not specified on the JOB command, the job queue specified in the JOB command or in the job description is used. (Note that the job queue for each job in the job stream can be different.)

Restriction: If the user identified in the job description of the job being read is not authorized to the job queue on which the job should be placed, that job terminates, and a diagnostic message that describes the cause for termination is placed in the job log. The job stream, however, continues to be processed, starting with the next job.

QBATCH.*LIBL: The job entry is to be placed on the QBATCH job queue, which is the default job queue if JOBQ(*RDR) is specified on the JOB command.

job-queue-name: The qualified name of the job queue to which each job in the job stream is to be sent if JOBQ(*RDR) is specified on its JOB command. If no library qualifier is given, *LIBL is used to find the queue.

MSGQ Parameter: Specifies the name of the message queue to which operational messages are to be sent.

*DEVD: The messages are to be sent to the message queue indicated by the device description of the device being used.

**REQUESTER*: The messages are to be sent to the message queue of the user who started the process. This value is not valid for batch jobs.

message-queue-name: Enter the qualified name of the message queue to which operational messages are to be sent. If no library qualifier is given, *LIBL is used to find the queue.

DSPSBMJOB Parameter: Specifies whether the jobs being submitted are to be displayed on the submitted jobs display. Any submitted job of the type specified by the SBMFROM parameter of the DSPSBMJOB command can be displayed if *YES is specified.

*YES: This job can be displayed by the DSPSBMJOB command.

**NO:* This job is not to be displayed on any display produced by the DSPSBMJOB command.

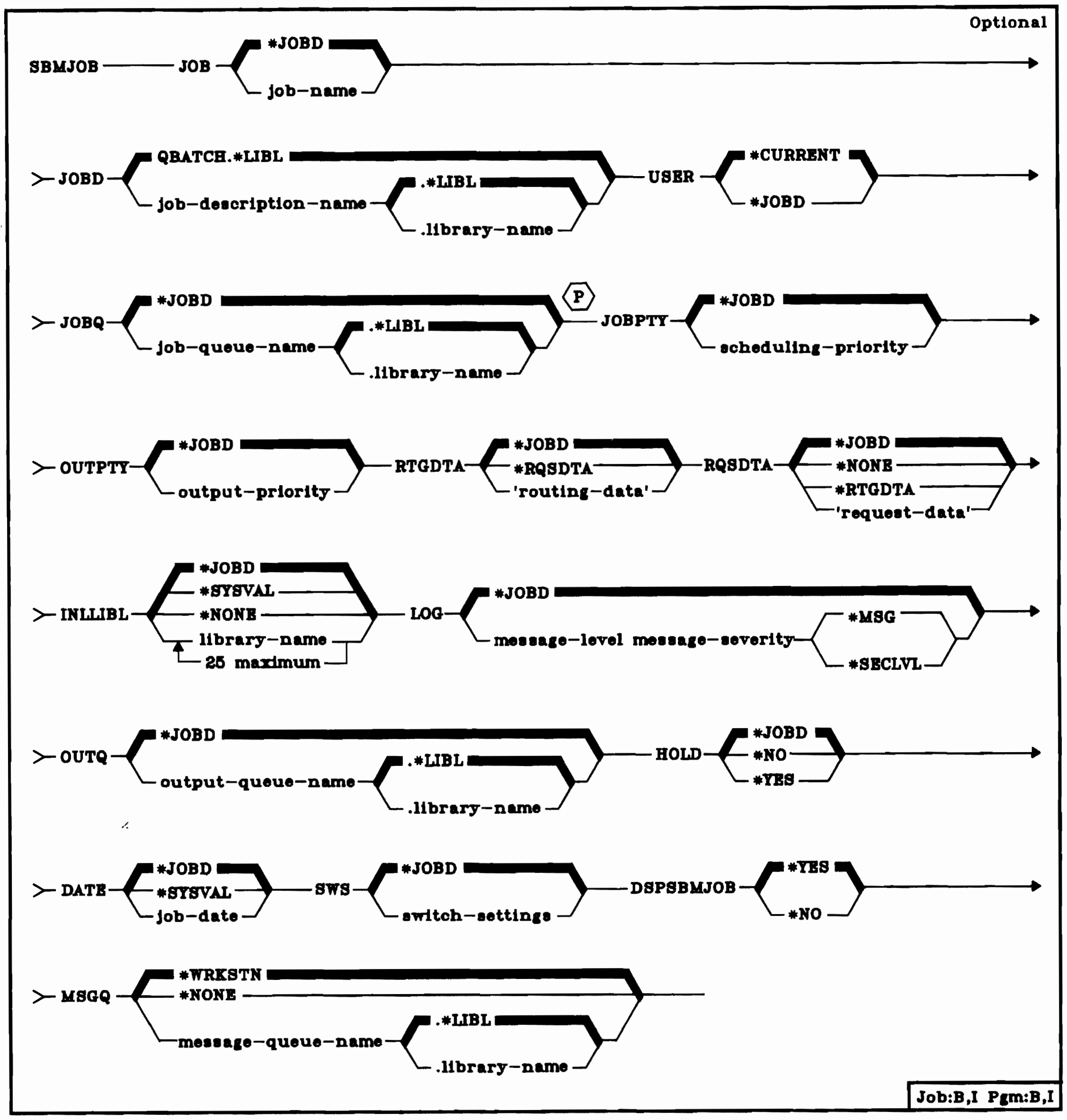
Example

```
SBMDKTJOB  DEV(QDKT) LABEL(OCT24) LOC(*S12) +
      VOL(SALES)
```

This command submits diskette jobs using diskette input from the device QDKT. The submit diskette jobs function gets its input from the data file named OCT24 that is located on the single diskettes placed in manual slots S1 and S2, with the volume identifiers SALES. The default job queue QBATCH is used as the receiving job queue when JOBQ(*RDR) is found in the job description. Operational messages are sent to message queue defined by the device.

SBMJOB (Submit Job) Command

The Submit Job (SBMJOB) command allows an executing job to submit another job to a job queue to be executed later as a batch job. Only one element of request data can be placed on the new job's message queue. The request data can be a CL command if the routing entry used for the job specifies a CL command processing program.



JOB Parameter: Specifies the job name to be associated with the submitted job as it is being processed by the system.

<u>*JOBD:</u> The simple name of the job description used with this job is to be the name of the job itself.

job-name: Enter the simple name of the job that is to be used while it is being processed by the system.

JOBD Parameter: Specifies the qualified name of the job description to be used with the job being submitted.

<u>QBATCH:</u> The IBM-supplied job description, QBATCH, in the QGPL library is to be used for the job. (The QGPL library must be in the library list used by the job in which the SBMJOB command is entered.)

job-description-name: Enter the qualified name of the job description that is to be used for the job. (If no library qualifier is given, the job description is found through the library list used by the job in which the SBMJOB command is entered.)

USER Parameter: Specifies the name of the user profile to be associated with the job being submitted. If USER(*RQD) is specified in the job description, *JOBD cannot be specified here; *CURRENT must be specified instead.

<u>*CURRENT:</u> The same user profile being used by the currently executing job is to be used for the submitted job.

**JOBD:* The user profile that was named in the specified job description is to be used for the job being submitted.

JOBQ Parameter: Specifies the name of the job queue in which the submitted job is to be placed.

<u>*JOBD:</u> The submitted job is to be placed in the job queue named in the specified job description.

qualified-job-queue-name: Enter the qualified name of the job queue on which the submitted job is to be placed. (If no library qualifier is given, *LIBL is used to find the queue.)

JOBPTY Parameter: Specifies the scheduling priority for the submitted job. Valid values are 1 through 9, where 1 is the highest priority and 9 is the lowest. (For an expanded description of the JOBPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

*JOBD: The scheduling priority specified in the job description is to be used for the job.

scheduling-priority: Enter a value, 1 through 9, that is to be the scheduling priority for the job.

OUTPTY Parameter: Specifies the output priority for spooled output files that are produced by the submitted job. The highest priority is 1 and the lowest is 9. (For an expanded description of the OUTPTY parameter, see *Scheduling Priority Parameters* in Appendix A.)

*JOBD: The output priority specified in the job description is to be used for the job.

output-priority: Enter a value, 1 through 9, for the priority that the submitted job's output files are to have.

RTGDTA Parameter: Specifies the routing data that is to be used to initiate the routing step for the submitted job. The routing data is used to determine the routing entry that identifies the program that is to process the routing step. For example, the value QCMDI is the routing data used by the IBM-supplied interactive subsystem QINTER to route interactive jobs to the IBM-supplied control language processor QCL.

*JOBD: The routing data to be used to initiate the routing step is in the job description used with the job.

**RQSDTA:* Up to 80 characters of the request data specified in the RQSDTA parameter of this command is to be used as the routing data for the job.

'routing-data': Enter the character string that is to be used as the routing data for initiating the job. A maximum of 80 characters can be entered (enclosed in apostrophes if necessary).

RQSDTA Parameter: Specifies the request data that is to be placed in the submitted job's message queue. For example, if RTGDTA (QCMDB) is specified, the IBM-supplied batch subsystem QBATCH is being used, and a CL command is supplied, it becomes a message that is read by the control language processor, QCL.

*JOBD: The request data specified in the job description used by the job is to be placed in this job's message queue.

**NONE:* No request data is to be placed in the job's message queue.

**RTGDTA:* The routing data specified in the RTGDTA parameter of this command is to be placed as the last entry in the job's message queue.

'request-data': Enter the character string that is to be placed as the last entry in the submitted job's message queue. A maximum of 256 characters can be entered (enclosed in apostrophes if necessary). When a CL command is entered, it must be enclosed in single apostrophes, and where apostrophes would normally be used *within* the command, double apostrophes must be used instead.

INLLIBL Parameter: Specifies the initial library list that is to be used by the submitted job to search for any object names that were specified without a library qualifier.

*JOBD: The library list specified in the job description used with the job is to be used as the initial library list for the job.

**SYSVAL:* The system default library list is to be used by the job. It contains the library names that were specified in the system values QSYSLIBL and QUSRLIBL at the time that the job is initiated.

**NONE:* The user portion of the initial library list is to be empty; only the system portion is to be used.

library-name: Enter the names of one or more libraries that are to be in the user portion of the library list and are to be used by the submitted job. No more than 25 names can be specified; the libraries are searched in the same order as they are listed here.

LOG Parameter: Specifies the message logging values to be used by the submitted job. They determine the amount and type of information to be logged in the job log. The LOG parameter is made up of a list of three values: the message (or logging) level, the message severity, and the level of message text. If no values are specified for the LOG parameter, the values specified in the job description used with the job are used. If one value is to be changed, all three values in the list must be specified.

*JOBD: The list of values specified for message logging in the job description are to be used for the submitted job.

message-level: Enter a value, 0 through 4, that specifies the message logging level to be used for the submitted job's messages. (For additional information on the message levels, refer to *Message Level* under the CRTJOBD command's LOG parameter.)

message-severity: Enter a value, 00 through 99, that specifies the lowest severity level that causes an error message to be logged in the job's log. (For an expanded description of severity codes, see the SEV parameter in Appendix A.)

**MSG:* Only first-level message text is to be written to the job's log or displayed to the user.

**SECLVL:* Both the first-level and second-level text of the error message is to be written to the job's log or displayed to the user.

OUTQ Parameter: Specifies the name of the default output queue that is to be used for spooled output produced by the submitted job.

*JOBD: The output queue named in the job description used with the submitted job is to be the job's default output queue.

qualified-output-queue-name: Enter the qualified name of the output queue that is to be used as the default output queue by the submitted job. (If no library qualifier is given, *LIBL is used to find the queue.)

HOLD Parameter: Specifies whether the submitted job is to be held at the time that it is put on the job queue. A job placed on the job queue in the hold state is held until it is released by the Release Job (RLSJOB) command, canceled by the Cancel Job (CNLJOB) command, or removed from the job queue by the Clear Job Queue (CLRJOBQ) command. If the job is not executed before CPF is terminated, the job queue can be cleared (and the job canceled) when CPF is started again.

*JOBD: The value specified in the job description determines whether the job is to be held when it is put on the job queue.

**NO:* The job is not to be held when it is put on the job queue.

**YES:* The job is to be held when it is put on the job queue, and held until it is released by a RLSJOB command or canceled by a CNLJOB command.

DATE Parameter: Specifies the date that is to be assigned to the submitted job when it is initiated.

*JOBD: The date specified in the job description is to be used as the job date.

**SYSVAL:* The value in the QDATE system value at the time the job is initiated is to be used as the job date.

job-date: Enter the value that is to be used as the job date when the job is initiated; the date must be in the format specified by the system value QDATFMT. (Refer to the *CPF Programmer's Guide* for the formats.)

SWS Parameter: Specifies the initial settings for a group of eight job switches to be used with the submitted job. These switches can be set or tested in a CL program and used to control the flow of the program. For example, if a certain switch is on, another program could be called. The job switches may also be valid in other HLL programs. Only zeros (off) and ones (on) can be specified in the eight-digit character string.

*JOBD: The value specified in the job description is to be the initial settings for the job's switches.

switch-settings: Enter any combination of eight zeros and ones that is to be used as the initial switch setting for the submitted job.

DSPSBMJOB Parameter: Specifies whether the job being submitted is allowed to be displayed on the *submitted jobs display*. Any submitted job of the type specified by the SBMFROM parameter of the DSPSBMJOB command can be displayed if the job is not prevented by this parameter.

*YES: This job can be displayed by the DSPSBMJOB command.

**NO:* This job is not to be displayed on any display produced by the DSPSBMJOB command.

MSGQ Parameter: Specifies the message queue, if any, to which a completion message is to be sent when the submitted job has completed execution, either normally or abnormally. If an abnormal termination occurs, the second-level text of the completion message sent specifies the possible causes.

*WRKSTN: A completion message is to be sent to the work station message queue of the work station from which the job was submitted. If the job is submitted by a batch job, no completion message is sent.

**NONE:* No completion message is to be sent.

qualified-message-queue-name: Enter the qualified name of the message queue to which the completion message is to be sent. If no library qualifier is given, the library list of the job issuing the SBMJOB command is used to find the queue.

Examples

```
SBMJOB  JOB(SPECIAL) JOBD(MYJOBD.MYLIB) +
      RQSDTA('CALL MYPROG')
```

This command causes the job named SPECIAL to be submitted. Most of the attributes for the job are taken from the job description MYJOBD, except for the request data attribute. The CALL command is placed on the submitted job's message queue so that the program MYPROG can be called and executed later.

```
SBMJOB  JOB(PAYROLL) JOBD(PAYROLL)
```

This command submits a job named PAYROLL to the system. All the information needed for this job (such as the job queue and routing data) is contained in the job description PAYROLL. The library list in effect for the job issuing this command is used to find the job description.

```
SBMJOB  JOBD(QBATCH) JOB(COPY12) JOBQ(NIGHTQ) +
      RQSDTA('CPYF FILEA FILEB')
```

This command submits the job COPY12, which uses the job description QBATCH, to the job queue NIGHTQ. The request data provides the CL command necessary for job execution. A command such as this might be used to copy the file at night while the system is unattended.

SBMRJEJOB (Submit RJE Job) Command

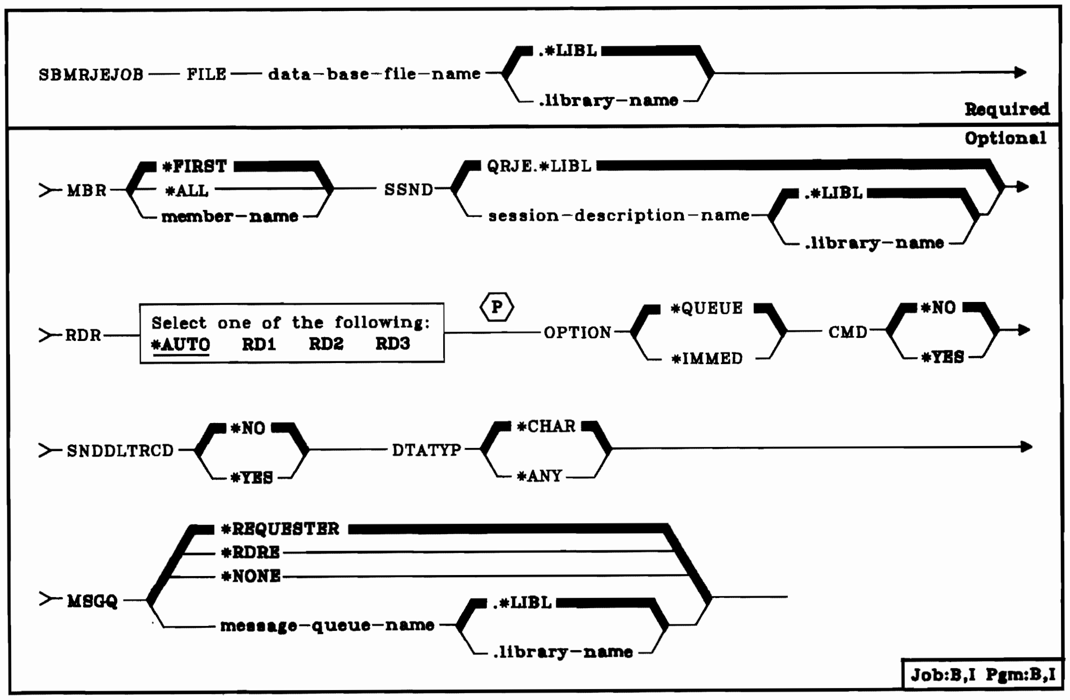
The Submit RJE Job (SBMRJEJOB) command requests that a specific RJEF remote job input stream (System/38 data base file) be transmitted to the host system.

Through the OPTION parameter of this command, you can have an RJEF reader job sent to the host system immediately, or placed on an RJEF reader job queue to be sent to the host system as a batch job.

Restrictions: To use this command, you must have:

- Operational rights for the session description.
- Operational and read rights for the input data base file and any files named on embedded READFILE control statements.
- Read rights for the library in which the session description and data base file are stored.

The Submit RJE Job (SBMRJEJOB) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



FILE Parameter: Specifies the qualified name of the data base file that contains the remote job input stream to be submitted. (If no library qualifier is given, *LIBL is used to find the data base file.)

MBR Parameter: Specifies the member in the data base file that is to be sent to the host system.

FIRST: Only the first member in the data base file is to be sent to the host system.

**ALL:* All the members in the data base file are to be sent to the host system.

member-name: Enter the name of the member of the data base file that is to be sent to the host system.

SSND Parameter: Specifies the qualified name of the session description that is to be used to send an RJEF job to the host system.

QRJE: The RJEF session description name in the QRJE library is to be used for this session description. (If no library qualifier is given, *LIBL is used to find the session description.)

session-description-name: Enter the qualified name of the session description that is to be used to send an RJEF job to the host system. (If no library qualifier is given, *LIBL is used to find the session description.)

RDR Parameter: Identifies the RJEF reader that is associated with this RJEF reader function.

*AUTO: Any RJEF reader input stream that is available at the time this command executes is to be used.

RD1: RJEF Reader 1 is to be used.

RD2: RJEF Reader 2 is to be used.

RD3: RJEF Reader 3 is to be used.

OPTION Parameter: Specifies whether the RJEF reader job is sent to an RJEF reader job queue or started immediately.

QUEUE: The job is submitted to the job queue identified by the reader specified in the RDR parameter in this command. The RJEF session does not have to be active. The RJEF reader job queue is released when the session is started and the STRRJERDR command is issued.

**IMMED:* The job is started immediately and control is not returned to the work station until the job is complete. The RJEF session must be active. The specified RJEF reader must be started and not in use.

CMD Parameter: Specifies whether the file specified in the FILE parameter in this command is to be searched for embedded RJE control statements (READFILE or EOF).

NO: The file is not to be searched for embedded control statements.

**YES:* The file is to be searched for embedded control statements.

SNDDLTRCD Parameter: Specifies whether blank records are to be sent to the host system for deleted records encountered in the data base file member.

NO: Blank records are not to be sent to the host system.

**YES:* Blank records are to be sent to the host system.

DTATYP Parameter: Specifies whether the file specified in the FILE parameter in this command contains any data characters less than X'40'.

CHAR: The file does not contain any data characters that are less than X'40'.

**ANY:* The file contains data characters that are less than X'40'.

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF reader are to be recorded.

Note: Messages for RJEF readers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands. If inquiry messages are issued by RJEF, they are sent to the user message queue (if specified) where they must receive a response.

REQUESTER: The RJEF reader messages are to be sent to the requester's work station message queue as well as to the RJEF message queue.

If this command is executed in a batch job, *REQUESTER is treated as *NONE (inquiry messages cannot be issued to a batch job log).

**RDRE:* The message queue name is to be retrieved from the session description RJEF reader entry associated with this RJEF reader function.

**NONE:* No user message queue exists on which the messages for these RJEF reader jobs are to be recorded.

message-queue-name: Enter the qualified name of the user message queue on which this RJEF reader job's messages are to be recorded. (If no library qualifier is given, *LIBL is used to find the message queue.)

Example

```
SBMRJEJOB FILE(JCL.USERLIB) +
      MBR(*ALL) +
      SSND(RJE.USERLIB) +
      RDR(*AUTO) +
      OPTION(QUEUE) +
      CMD(*YES) +
      MSGQ(BROWN.MEDLIB)
```

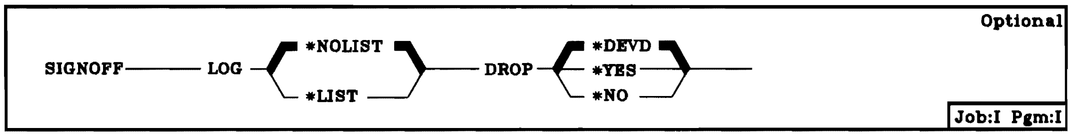
This command sends data from the data base file named JCL in library USERLIB to the host system. All members in the data base file are to be sent. The job is submitted to an RJE reader job queue (the job queue is specified in the *AUTO reader entry in the session description named RJE in library USERLIB). All members of the data base file are command files, meaning they are checked for READFILE and EOF control statements.

RJEF messages associated with RD1, while sending this file, are written to user message queue named BROWN in library MEDLIB.

SIGNOFF (Sign Off) Command

The Sign Off (SIGNOFF) command terminates an interactive job. A user enters this command to sign off at a work station or at the console.

Restriction: This command is valid only in an interactive job.



LOG Parameter: Specifies whether the job log for this interactive job is to be deleted or is to be included in the job's spooled output for printing.

*NOLIST: The information in the job log, which has already been displayed throughout the job, is no longer needed and is to be deleted.

*LIST: The job log is to be spooled for printing, along with the rest of the job's spooled output, if any.

DROP Parameter: Specifies, for switched (dial-up) lines only, whether or not the switched line attached to the work station is to be disconnected (dropped) if no other work stations on the same line are signed on. This parameter is ignored if the work station is attached to a nonswitched line.

*DEVD: The value specified in the DROP parameter of the work station's device description is to be assumed.

*YES: The switched line is to be disconnected when the job is terminated if no other work stations are signed on the line.

_*NO:_ The switched line is not to be disconnected when the job is terminated.

Examples

```
SIGNOFF
```

This command signs off the user of the work station and terminates the interactive job. The switched line is dropped only if so specified in the device description of this work station and if no other work station on this line is active. A job termination message that gives the job start and stop times is written in the job's log.

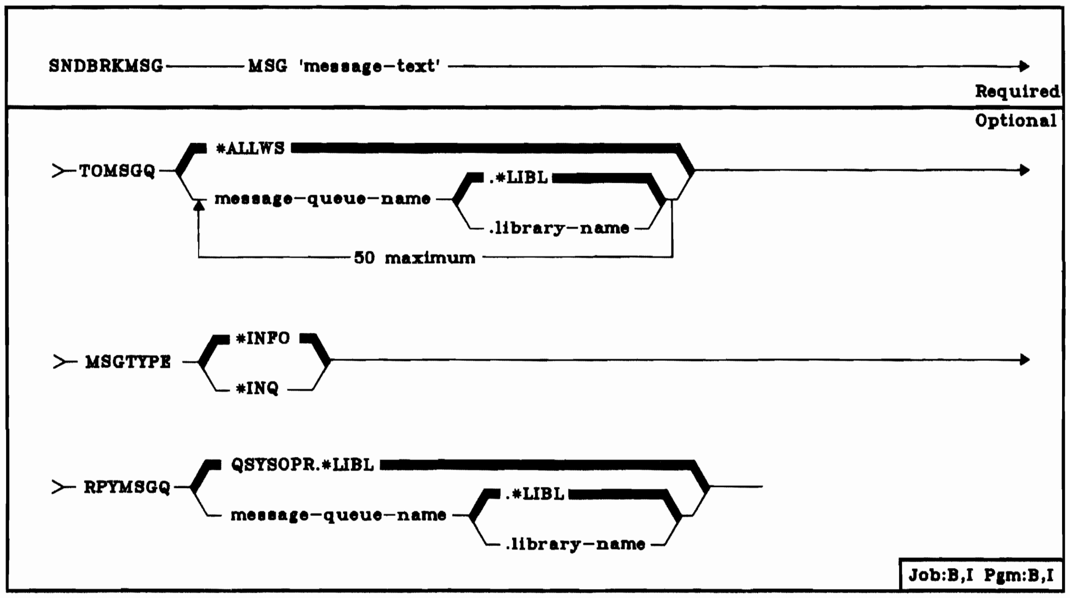
```
SIGNOFF  LOG(*LIST) DROP(*NO)
```

This command terminates the interactive job, but the switched line is not released. The job log is to be included with the job's spooled output.

SNDBRKMSG (Send Break Message) Command

The Send Break Message (SNDBRKMSG) command is used by the system operator to send an impromptu message to one or more work station message queues. The command causes the message to be delivered in break mode, regardless of how the receiving message queue's delivery attribute is set. (An impromptu message is a message that is not predefined and is not stored in a message file.) This command is primarily intended for the system operator's use.

Restrictions: This command can be used to send break messages to work station message queues only. The user sending the message must have operational rights for the specified work station message queues and the QSYS library in which the message queues are stored.



MSG Parameter: Specifies the message text of the impromptu message to be sent. The message must be enclosed in apostrophes if it contains blanks or other special characters. A maximum of 132 characters can be specified.

TOMSGQ Parameter: Specifies the names of one or more work station message queues to which the break message is to be sent. Only the names of work station message queues can be specified.

*ALLWS: The break message is to be sent to all work station message queues.

qualified-message-queue-name: Enter the qualified names of one or more work station message queues to which the break message is to be sent. The only library name that can be specified is QSYS. (If no library qualifiers are given, *LIBL is used to find the message queues.)

MSGTYPE Parameter: Specifies the type of message that is to be sent in break mode. Only informational or inquiry message types can be specified.

<u>*INFO:</u> An information only message is to be sent in break mode.

**INQ:* An inquiry message is to be sent in break mode; the work station receiving the message is expected to reply to it. (Inquiry messages can be sent to only one work station at a time.)

RPYMSGQ Parameter: Specifies, only if an inquiry message is sent, the name of the message queue that the work station user's reply is to be sent to.

<u>QSYSOPR:</u> The replies to the break message are to be sent to the system operator's message queue, QSYSOPR, which is in the QSYS library.

qualified-message-queue-name: Enter the qualified name of the message queue to which a reply to the break message is to be sent. Only a user message queue or a work station message queue can be specified. (If no library qualifier is given, *LIBL is used to find the queue.)

Examples

```
SNDBRKMSG  MSG('The inventory application +
      shuts down at 4:00 PM today.')
```

This command sends, in break delivery mode, the message 'The inventory application shuts down at 4:00 PM today.' to the message queues of all signed-on work stations. The message is delivered regardless of the delivery attribute setting of those message queues.

```
SNDBRKMSG  MSG ('Your printed output is ready.') +
      TOMSGQ(GEORGEMSGQ)
```

This example shows a typical use of the SNDBRKMSG command by the system operator to send a message to a work station user.

SNDDTAARA (Send Data Area) Command

The Send Data Area (SNDDTAARA) command is used in a CL program to copy the value of a CL variable in the program to a data area outside the program. The value is copied into the data area that has the same name as the CL variable, but without the &. The data area must already have been created by a CRTDTAARA command and declared in the program by a DCLDTAARA command. The CL variable for the data area is declared automatically for the user when the data area is declared. If constant data or data from a CL variable with a different name is to be copied to the data area, the Change Data Area (CHGDTAARA) command must be used.

When the SNDDTAARA command is executed, the data area is locked to the program during the send operation so that commands in other jobs cannot change or destroy it until the operation is completed. If the data area is shared with other jobs and it is updated in steps involving more than one command in a job, the data area should be explicitly allocated to that job until all the steps have been performed. The data area can be explicitly allocated with the ALCOBJ command. If the data area is stored in a library other than QTEMP, the SNDDTAARA command ensures that the update is made in permanent storage as a precaution against a possible system failure.

Restrictions: (1) This command is valid only in CL programs. (2) To use this command, you must have update rights for the data area and read rights for the library in which it is stored. (3) The attributes of the data area at execution time must be the same as the attributes of the data area at compile time.

```
                         (1)                                        Required
SNDDTAARA─────────DTAARA data-area-name────

(1) A variable cannot be coded on this parameter.                   Pgm:B,I
```

DTAARA Parameter: Specifies the name of the data area to which the value in the corresponding CL variable is sent. A CL variable name cannot be used in this parameter to specify the name of the data area.

Example

```
DCLDTAARA  DTAARA(CHECKNUM)
  •
  •
  •
SNDDTAARA  DTAARA(CHECKNUM)
```

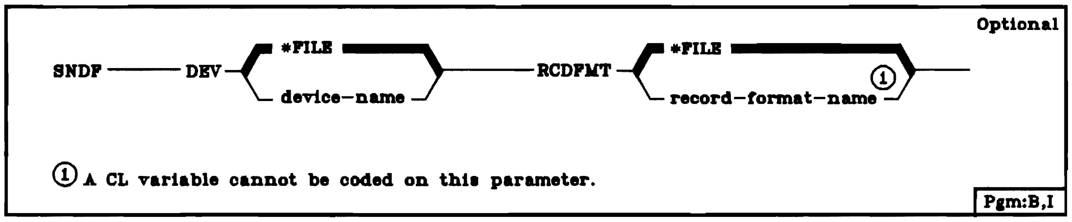
This command causes the value in the CL program variable &CHECKNUM to be copied into the data area named CHECKNUM. The updated data area value can then be used by other programs and jobs.

SNDF (Send File) Command

The Send File (SNDF) command is used by a CL program to send a record to a display device that is being used by an interactive user. The device can be any display station including the system console. The command sends the data from the program's CL variables to the display's device file in the specified record format. These variables were automatically declared in the program (one for each field in the record format) when the CL source program was compiled and a DCLF command was processed as part of the source.

Of the record formats specified in the DCLF command, only one can be specified in each SNDF command. If the device file has not been opened, it is opened by this command. The file and record format specified in this command can be overridden by an OVRDSPF (Override with Display File) command if that command is entered before the file is opened. However, care should be taken that the fields in the overriding record format correspond to the CL variables declared in the program.

Restriction: This command is valid only within a CL program.



① A CL variable cannot be coded on this parameter.

DEV Parameter: Specifies the name of the display device to which the data in the CL variables for the specified record format is to be sent.

*FILE: The program's data is to be sent to the device associated with the device file (the device file that was declared in the FILE parameter of the DCLF command). If more than one device name is specified in the device file, *FILE cannot be specified.

device-name: Enter the name of the device or the name of the CL variable that is to contain the name of the device to which the program's data is to be sent.

RCDFMT Parameter: Specifies the name of the record format that is to be used to send data to the user. The format contains all the fields in the record. This parameter must be coded with a record format name if there is more than one record format name in the device file; *FILE cannot be coded if there is more than one.

<u>*FILE:</u> There is only one record format in the device file; that is the format in which the program's data is sent to the user.

record-format-name: Enter the name of the record format in which the program's data is to be sent to the user. A CL variable cannot be used here to specify the record format name.

Examples

```
DCLF  FILE(MENU1)
   •
   •
   •
SNDF
```

The record format in the device file MENU1 is to be sent to the device specified in the file. There is only one record format in the file.

```
DCLF  FILE(SCREEN1) RCDFMT(REC1 REC2)
   •
   •
   •
SNDF  DEV(DISP3) RCDFMT(REC1)
```

The device file named SCREEN1 causes the display work station named DISP3 to display the data sent by the CL program to the user. The data is displayed in the format specified by the REC1 record format.

```
DCLF  SCREEN1 (REC1 REC2)
   •
   •
   •
SNDF  *N REC2
```

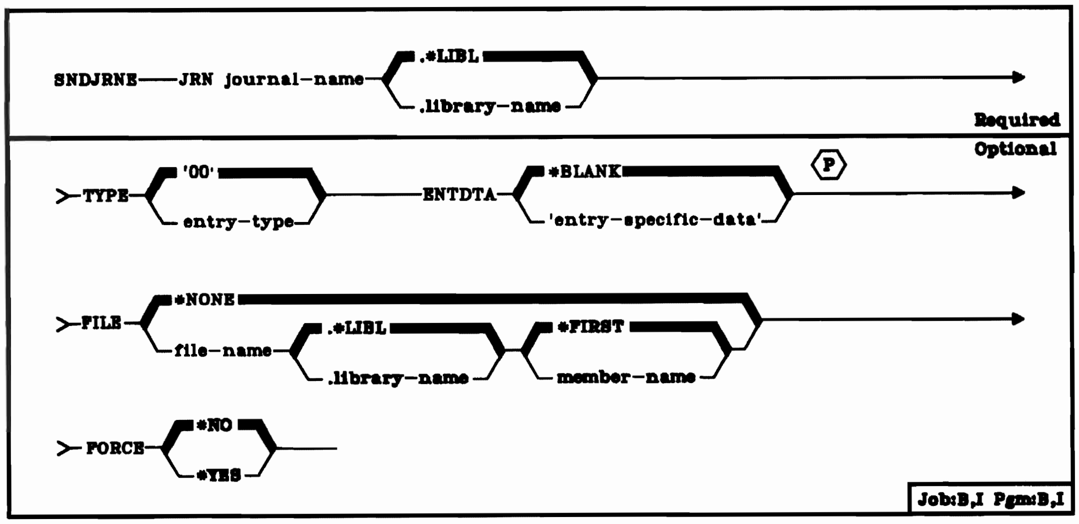
The device file named SCREEN1 is to send data to the device named in the same device file; *N is the null value for the DEV parameter coded positionally, which defaults to *FILE. The data is presented to the user in the format specified by REC2.

SNDJRNE (Send Journal Entry) Command

The Send Journal Entry (SNDJRNE) command is used to place a single journal entry onto a specific journal. The entry can contain any information. The user may assign an entry type to the journal entry and also may associate the journal entry with a particular physical file member.

Note: The journal code for the entry will be 'U', indicating a user-specified journal entry.

Restriction: If a file is specified, it must be currently journaled or must have been last journaled through the designated journal.



JRN Parameter: Specifies the qualified name of the journal that is to contain the new journal entry. (If a library qualifier is not specified, *LIBL is used to find the journal.)

TYPE Parameter: Specifies the journal entry type of this journal entry.

00: The journal entry type is to be a '00' (hex 'F0F0').

entry-type: Enter a two-character value or hex value that is to be used for the journal entry type. This value must be greater than or equal to hex 'C000'.

If a hex value is specified that does not represent characters, that value will not be shown on the DSPJRN display or the printer listing when the entry is converted.

ENTDTA Parameter: Specifies the user-specified data that is to be placed in the variable portion of the journal entry.

*BLANK: Specifies that no user-specified data is to be placed in the journal entry.

entry-specific-data: Enter no more than 2000 characters, enclosed in apostrophes.

FILE Parameter: Specifies the name of a data base physical file and member with which this entry will be associated.

*NONE: Specifies that there is no associated physical file for this entry.

file-name: Specifies the qualified name of the physical file with which this entry will be associated.

member-name: Specifies the name of the physical file member with which this entry will be associated. If the member name is specified as *FIRST, then the first member in the file will be used.

FORCE Parameter: Specifies whether the journal receiver(s) is to be forced to auxiliary storage after the user entry is placed on it.

*NO: The journal receiver(s) is not to be forced to auxiliary storage.

**YES:* The journal receiver(s) is to be forced to auxiliary storage.

Examples

```
SNDJRNE  JRN(JRNLA) TYPE(AB) ENTDTA('PROGRAM COMPLETE') +
      FILE(ORDERENT.MYLIB MBR1) FORCE(*YES)
```

The above command causes a journal entry of type AB (hex 'C1C2') with the designated user-generated journal entry data to be placed on the current journal receivers attached to journal JRNLA as found using the library search list. The entry will be associated with member MBR1 of file ORDERENT in library MYLIB. The journal receivers will be forced to auxiliary storage after the entry has been placed on them.

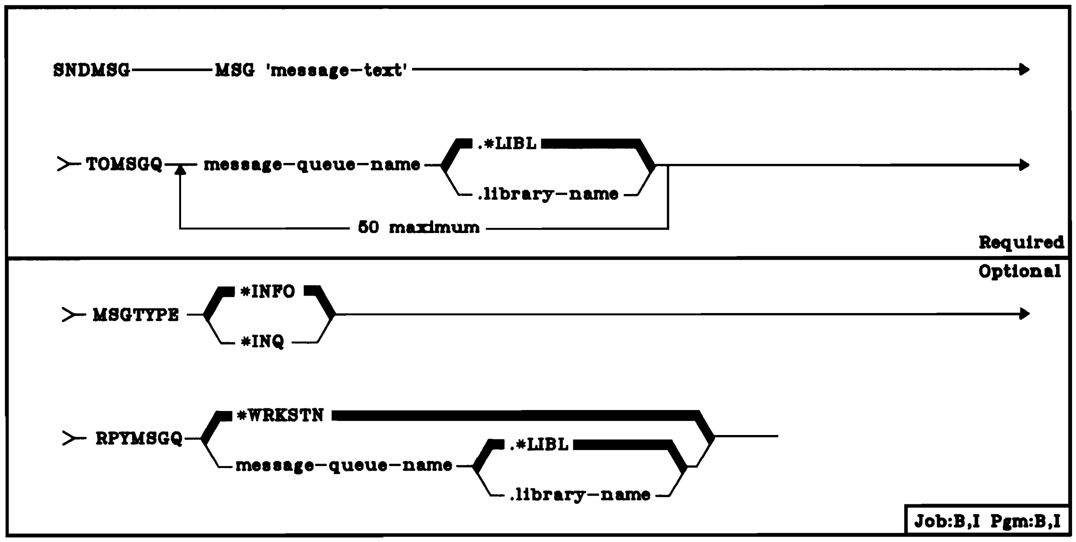
```
SNDJRNE  JRN(JRNLA) TYPE(x'C1F1')
```

The above command causes a journal entry of type 'A1' (hexadecimal 'C1F1') with no user-generated journal entry data to be placed on the current journal receivers attached to journal JRNLA as found using the library search list. The entry will not be associated with any physical file member.

SNDMSG (Send Message) Command

The Send Message (SNDMSG) command is used by a work station user to send an impromptu message from his work station to one or more message queues. (An impromptu message is a message that is not predefined and is not stored in a message file.) The message can be sent to the system operator, to other work station users, to a user's message queue, or to one of the IBM-supplied logs, such as QHST. The sender can require a reply from the message receiver. The primary users of this command are work station operators and the system operator.

Restriction: To use this command, you must have operational rights for the specified message queues and for the libraries in which they are stored.



MSG Parameter: Specifies the message text of the impromptu message that is to be sent. The text must be enclosed in apostrophes if it contains blanks or other special characters. A maximum of 132 characters can be specified.

TOMSGQ Parameter: Specifies the qualified names of one or more message queues to which the message is to be sent. A message may also be sent to one of the IBM-supplied logs (such as QHST). (If no library qualifier is given, *LIBL is used to find the queue.) To send the message to the system operator, enter the value QSYSOPR, which is the name of the system operator's message queue.

MSGTYPE Parameter: Specifies the type of message to be sent. Only informational and inquiry message types can be specified.

INFO: An informational message is to be sent.

**INQ:* An inquiry message is to be sent; the message queue receiving the message must reply to it. Inquiry messages can be sent to only one message queue at a time.

RPYMSGQ Parameter: Specifies, only if an inquiry message is sent, the name of the message queue to which a reply is to be sent.

*WRKSTN: The reply to the message is sent to the work station message queue associated with the sender's work station.

qualified-message-queue-name: Enter the qualified name of the message queue to which a reply is to be sent. Only a user message queue, a work station message queue, or the system operator message queue can be specified. (If no library qualifier is given, *LIBL is used to find the queue.)

Examples

```
SNDMSG  MSG('Do you want to update INV now?') +
      TOMSGQ(JONES) MSGTYPE(*INQ) RPYMSGQ(SMITH)
```

This command sends a message to the user message queue JONES, requiring that a reply be sent to user message queue SMITH.

```
SNDMSG  MSG('Input errors on PAYROLL cost +
      me 1 hour of run time.') TOMSGQ(QHST)
```

This command is used by the system operator to send an informational message to the system's history log, QHST, via the log's message queue, which has the same name.

```
SNDMSG  MSG('Please make 2 copies of printer +
      file LABORSTAT.') TOMSGQ(QSYSOPR)
```

This example shows a typical use of the SNDMSG command by a work station user. The user is sending the message 'please make 2 copies of printer file LABORSTAT.' to the system operator.

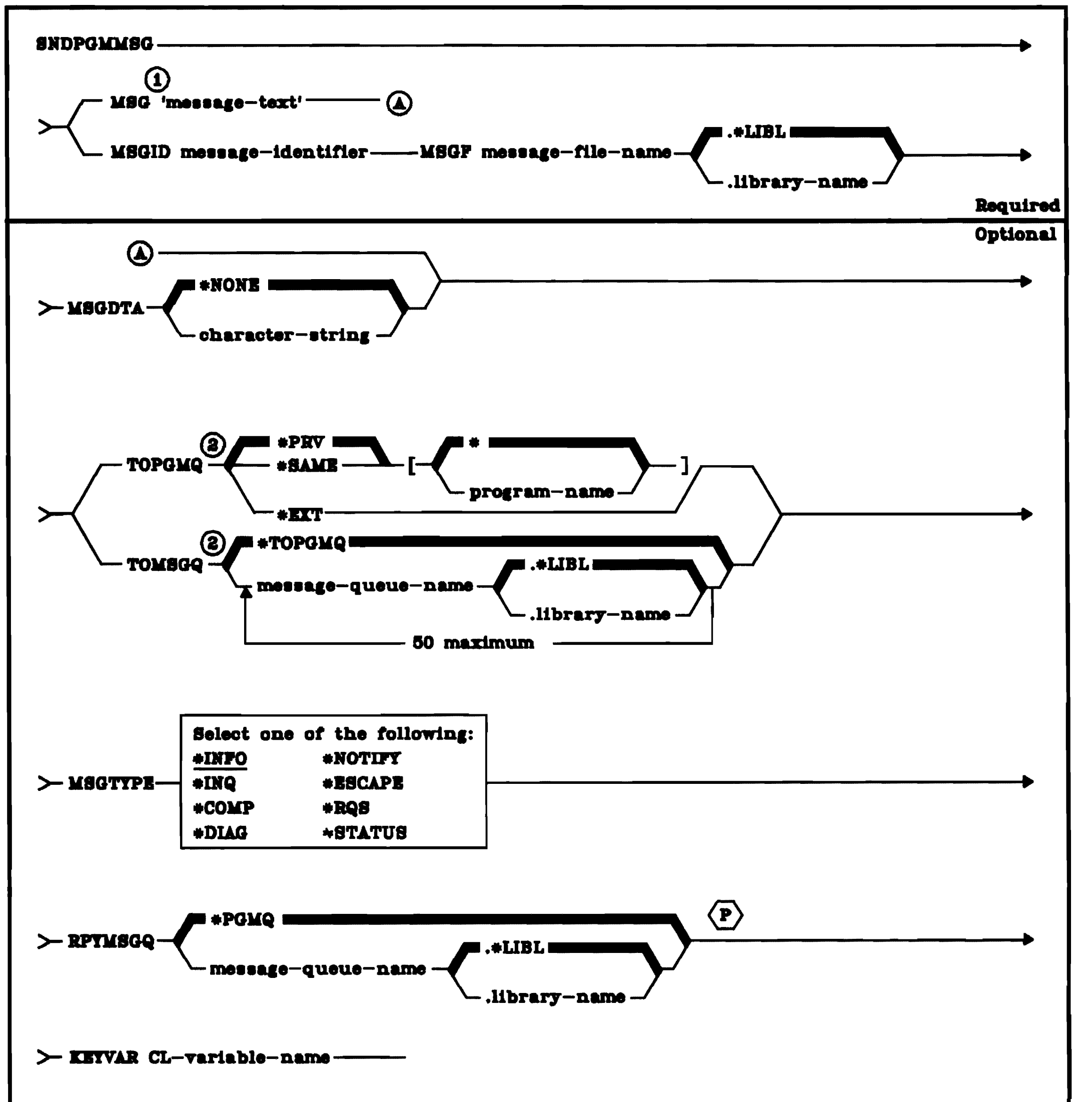
```
SNDMSG  MSG ('How long will the work stations be +
      online today?') TOMSGQ(QSYSOPR) MSGTYPE(*INQ)
```

This inquiry message from a work station user requires a reply. The system operator displays the message by using the DSPMSG command and enters the reply on the display. The work station user enters another DSPMSG command to display the reply.

SNDPGMMSG (Send Program Message) Command

The Send Program Message (SNDPGMMSG) command is used by a program to send a program message to the named message queue. The message can be an impromptu message, user-defined data, or a predefined message stored in a message file. Substitution values can be specified in the MSGDTA parameter (as a single character string containing one or more concatenated message data fields) to replace the substitution variables in the message. The message can be sent to the job's external message queue, to the program message queue of a program invoked by the job, to one or more nonprogram message queues, such as the system operator message queue, or to one of the IBM-supplied logs, such as QHST.

Restriction: This command is valid only in CL programs.



① To code the following parameters *positionally*, you must code them in this order, using *N for those not being specified: MSG, MSGID, MSGF, and MSGDTA.
② If both TOPGMQ and TOMSGQ are specified, the message is sent to the specified program message queue and to the specified nonprogram message queues.

Pgm:B,I

MSG Parameter: Specifies the message text of an impromptu message that is to be sent by the program to a message queue. A maximum of 132 characters can be specified. The text string must be enclosed in apostrophes if special characters (including blanks) are used. If MSG is specified, MSGID cannot be specified. If MSG is specified, *ESCAPE, *NOTIFY, or *STATUS cannot be specified for the MSGTYPE parameter, because these types require that a message identifier must also be specified.

MSGID Parameter: Specifies the message identifier of a message description whose predefined message is to be sent by the program to a message queue. If MSGID is specified, the MSG parameter cannot be specified.

MSGF Parameter: Specifies the qualified name of the message file that contains the predefined message to be sent. (If no library qualifier is given, *LIBL is used to find the file. If the message is sent to the QSYSOPR message queue, the library list used is the one that is in effect for the console or work station at which the message is actually displayed.) This parameter is required if MSGID is specified.

MSGDTA Parameter: Specifies the character string, or a CL variable that contains a character string, containing one or more substitution values that are to be used as message data fields within the predefined message. The substitution values take the place of the substitution variables that were defined in the message text when the message was defined.

Multiple values must be concatenated to form a single character string (see the *CPF Programmer's Guide* for more details). If more than one substitution variable exists in the message, the specified character string must be the same length as the sum of the message data fields defined in the message. The length of the entire character string of concatenated substitution values cannot exceed 132 characters. (For information about the length of individual message data fields, see the FMT parameter description in the ADDMSGD command.) Each substitution value, specified in the same order in the string as defined in the MSG and FMT parameters of the ADDMSGD command, must be specified as long as its associated variable was defined. Also, if the length of the message data that is passed to a substitution variable is shorter than the length specified for ADDMSGD FMT, the substitution value becomes a null field.

*NONE: There are no program-supplied substitution values to be used in the specified message. The predefined message has no substitution variables in its text.

character-string: Enter the character string that gives the substitution values in the specified predefined message that is to be sent by the program, or enter the name of the CL variable that contains the character string.

If the message does contain more than one substitution variable, the character string containing the replacement values must be coded with the proper fill characters (zeros for numeric values, and blanks for character values) to position the values properly in the string to match the concatenated fields used to create the single string. If the string contains special characters (including blanks), it must be enclosed in apostrophes. The null value *N cannot be specified for a substitution value.

For example, if a predefined message contains the following three substitution variables:

Variable	Format Type	Length	Expected Values
&1	*CHAR	3 characters	Yes or no
&2	*CHAR	4 digits	Four digits
&3	*CHAR	8 characters	One or two words

the following valid character strings can be specified:

```
MSGDTA('Yes0007not done')
MSGDTA('No 0285finished')
MSGDTA(&DTA)
```

Note that the variable &2 was defined as a character variable, not as a decimal variable. If a character string is to be used instead of a CL variable to specify the substitution *values* in the MSGDTA parameter, the associated substitution *variables* must be defined (in the FMT parameter of the ADDMSGD command) as character variables to prevent invalid results in the text of the sent message.

The last example shows, instead of the actual character string, the CL variable &DTA, whose value is to be used as the message data. &DTA might have been declared in the sending program as:

```
DCL VAR(&DTA) TYPE(*CHAR) LEN(15) VALUE('YES0175not done')
```

The character string, specified as the initial value for &DTA, contains the three concatenated substitution values *YES*, *0175*, and *not done*, which are to be sent as message data.

TOPGMQ Parameter: Specifies the name of the program message queue to which the specified message is to be sent. The named program message queue must be the queue of a program invocation in the sending job.

A list of two values is used to specify the program message queue to which the message is to be sent. The first value specifies the relationship of the receiving program invocation with respect to the program invocation specified by the second value.

One of the following values can be specified for the first value:

*PRV: The program message queue of the program invoked before the specified invocation (that is, the caller of the invocation) is the queue to which the message is to be sent.

**SAME:* The program message queue of the same program specified in the second value is the queue to which the message is to be sent.

One of the following values can be specified for the second value:

*: Entered as the second value in the list of values, it specifies the program message queue of the sending program.

program-name: Entered as the second value in the list of values, it specifies the message queue of the lowest invocation of the named program. (The name of the program message queue is the same as the name of its corresponding program invocation.)

**EXT:* The external message queue of the job is the queue to which the message is to be sent. The external message queue is used to communicate with the external requester of the job, such as a work station user.

TOMSGQ Parameter: Specifies the names of one or more message queues (nonprogram message queues) to which the message is to be sent by the program.

**TOPGMQ:* The program message queue that is specified in the TOPGMQ parameter is the only queue to which the message is to be sent.

qualified-message-queue-name: Enter the qualified names of the message queues to which the message is to be sent. (If no library qualifier is given, *LIBL is used to find the message queue.) The system operator message queue, QSYSOPR (in the QSYS library), can also be specified.

MSGTYPE Parameter: Specifies which message type is to be assigned to this message when it is sent by this program. Inquiry messages cannot be sent to program message queues. Completion, diagnostic, escape, notify, and status messages should be sent *only* to program message queues.

INFO: The message is to be sent as an informational message; it is to be sent to a program or nonprogram message queue.

**INQ:* The message is to be sent as an inquiry message; it is to be sent to a nonprogram message queue or to the external message queue of the job, *EXT. Normally, a reply must be returned to the program.

**COMP:* The message is to be sent as a completion message; it is to be sent to the program message queue of the program that requested work to be done by this program (sending the message). A completion message indicates the status of the work that is *successfully* performed. (Compare with **STATUS.*)

**DIAG:* The message is to be sent as a diagnostic message; it is to be sent to the program message queue of the program that requested work to be done by this program. Diagnostic messages provide information about errors detected by this program in the input sent to it or errors that occurred while it was executing the requested function. Normally, an escape or notify message should be sent to inform the receiving program of the diagnostic messages that were sent to its program message queue.

**NOTIFY:* The message is to be sent as a notify message; it is to be sent to a program message queue. A notify message describes a condition for which the sending program requires corrective action or a reply from the program to which the message is sent before the sending program can continue processing the requested function. If control returns to the sending program, the reply can be received from its program message queue. *NOTIFY cannot be specified if the MSG parameter is specified.

**ESCAPE:* The message is to be sent as an escape message; it is to be sent to a program message queue. An escape message describes the error condition for which the sending program is terminating. Sending an escape message terminates processing in the sending program. *ESCAPE cannot be specified if the MSG parameter is specified.

**RQS:* The message is to be sent as a request message; it is being sent to the program message queue of the program that requested work to be done by this program. A request message allows request data received from device files to be passed from this program to another. An impromptu message, specified by the MSG parameter, must be used to send the request.

**STATUS:* The message is to be sent as a status message; it is to be sent to a program message queue. A status message describes the status of work performed by the program sending the message. The first 28 characters of message data in the MSGDTA parameter are used as the comparison data for message monitors (established by the MONMSG command). If the program to which the status message is sent is not monitoring for that message identifier, control is returned to the sending program. If a status message is sent to the external message queue of an interactive job, the message is displayed and processing continues; no response is required.

RPYMSGQ Parameter: Specifies, for inquiry and notify messages only, the name of the program or nonprogram message queue to which the reply message is to be sent.

*PGMQ: The reply to an inquiry or notify message (sent by this program) is to be sent to this program's own program message queue.

qualified-message-queue-name: Enter the qualified name of the message queue to which the reply to an inquiry is to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

KEYVAR Parameter: Specifies the name of the CL character variable, if any, that is to contain the message reference key that identifies the message sent by the program containing this SNDPGMMSG command. The message reference key is assigned by the system when the message is sent and is placed in the variable specified by KEYVAR. The message key can then, for example, be used in the MSGKEY parameter of the RCVMSG command to receive the reply to an inquiry message being sent by this command. The message reference key can be used by the program specified on the TOPGMQ parameter for building message subfiles. The CL variable is the name of the field for which the SFLMSGKEY keyword is specified in the DDS for the message subfile.

If a message is being sent to a program message queue, KEYVAR refers to that message queue (specified by the TOPGMQ parameter). If *INQ or *NOTIFY is specified for the MSGTYPE parameter, KEYVAR refers to the message queue specified in the RPYMSGQ parameter. In all other cases, KEYVAR refers to the message queue specified in the TOMSGQ parameter.

If an inquiry message has been sent to the *EXT program message queue with a value specified for the KEYVAR parameter, the message reference key returned will be that of the sender's copy message, which is located on the reply message queue. The reply message queue defaults to the program message queue of the program that sent the inquiry message.

Any type of message can be assigned a key when it is being sent to a program message queue. For messages sent to a nonprogram message queue, message reference keys are available for inquiry (*INQ) messages only. If another message type is sent to a nonprogram queue, no message key is available and blanks are returned for KEYVAR.

The variable must be a character variable having a length of 4 characters. If KEYVAR is not specified and a reply is required, it can be received by the program in FIFO order.

Examples

```
SNDPGMMSG  MSGID(UIN0023) MSGF(INV) +
     MSGDTA(' 50100') TOPGMQ(*EXT)
```

This command sends the message identified as UIN0023, which is stored in message file INV, to the external message queue of the job. The data, containing two substitution values specified in the MSGDTA parameter is sent with the message. This data can then be used as substitution values when the message is received, or it can be used as data to be dumped, depending on how the message UIN0023 is defined in the message file. Assuming that the variables &1 and &2 have been defined in the message file as character variables, each 3 characters long, and that the message UIN0023 is:

'Requested item decreased by &1; current balance &2.'

the message text received is:

'Requested item decreased by 50; current balance 100.'

```
SNDPGMMSG  MSG('Mount payroll checks in printer +
     before continuing') MSGTYPE(*INQ) +
     TOMSGQ(QSYSOPR)
```

This command sends an inquiry message to the system operator. The operator displays the message by using the DSPMSG command and responds to the message on that display. A RCVMSG command is used in the program to accept the operator's response.

```
SNDPGMMSG  MSGID(USR0001) MSGF(USRMSGR) +
     TOPGMQ(*PRV *) MSGTYPE(*ESCAPE)
```

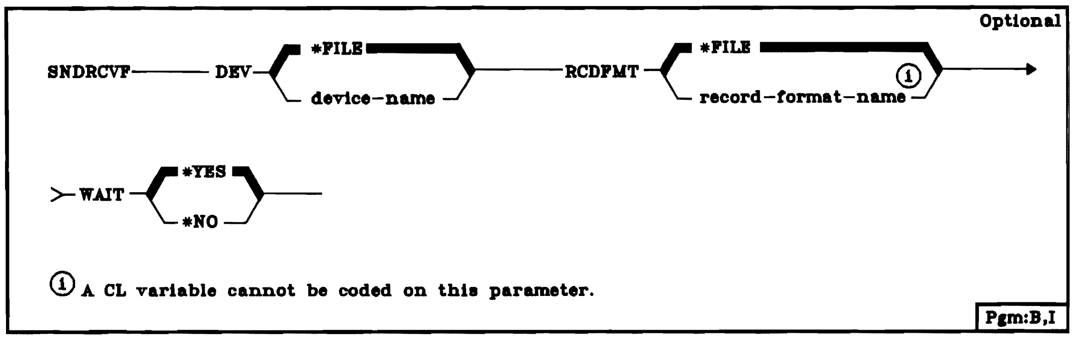
This command is an example of how a message could be sent to a program to cause an abnormal termination. The message USR0001 could indicate that an invalid code was passed (such as a nonnumeric code when numeric is required). Because the message being sent is an escape message, the program sending the message is terminated. The values *PRV and * did not have to be coded on this command because they are the default values on the TOPGMQ parameter.

SNDRCVF (Send/Receive File) Command

The Send/Receive File (SNDRCVF) command is used by a CL program to send to and receive data from a device that is being used interactively by a user. The data is passed between the device file and the program in records that are in the format specified in this command. The data sent by the program to the user is taken from CL variables that were automatically declared in the program when it was compiled and a DCLF command was processed as part of the source. There is one CL variable for each field in the record format used to send and receive the data.

Of the record formats specified in the DCLF command, only one can be specified in each SNDRCVF command. If the device file has not been opened, it is opened by this command. The file and record format specified in this command can be overridden by an Override with Display File (OVRDSPF) command if that command is entered before the file is opened. However, care should be taken that the fields in the overriding record format correspond to the CL variables declared in the program.

Restriction: This command is valid only within a CL program.



① A CL variable cannot be coded on this parameter.

DEV Parameter: Specifies the name of the display device that the CL program's data is to be sent to and the user's data is to be received from. The device name in the variable can be changed between executions of the command.

<u>*FILE:</u> The data is to be sent to and received from the device associated with the device file (the device file that was declared in the FILE parameter of the DCLF command). If more than one device name is specified in the device file, *FILE cannot be specified.

device-name: Enter the name of the device or the name of the CL variable that is to contain the name of the device that the CL program is to send data to and receive data from. If a CL variable name is used in this parameter, only one SNDRCVF command is needed in the program to receive data from several devices.

RCDFMT Parameter: Specifies the name of the record format that is to be used to pass the data between the CL program and the user. The format contains all the fields in the record. This parameter must be coded with a record format name if there is more than one record format in the device file; *FILE cannot be coded if there is more than one.

<u>*FILE:</u> There is only one record format in the device file; that format is used to send the data to and receive the data from the user.

record-format-name: Enter the name of the record format in which the data is to be sent to and received from the user. A CL variable name cannot be used here to specify the record format name.

WAIT Parameter: Specifies whether the CL program is to wait for the data to be received from the user's device or continue executing the commands that follow this SNDRCVF command. If WAIT(*NO) is specified, the program must issue a WAIT command later in the program to complete the input operation.

<u>*YES:</u> The program waits until the input operation from the device is completed; the next command is not executed until then.

**NO:* The program does not wait for the input data; it continues to execute commands until a WAIT command is reached later in the program.

Examples

```
DCLF  FILE(MENU1)
  •
  •
  •
SNDRCVF
```

This command sends and receives user data by way of the device file MENU1. Only one record format exists in the file. The device used is specified in the file.

```
DCLF  FILE(SCR)  RCDFMT(REC8)
  •
  •
  •
SNDRCVF  RCDFMT(REC8)
```

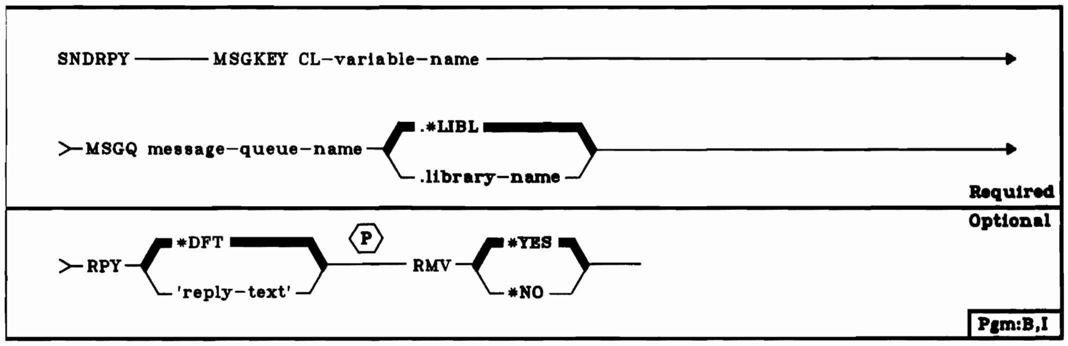
The CL program is to send data to a user and receive data from the user who is using the device named in the device file (*FILE was assumed because DEV was not specified). The data is to be passed in the format specified by REC8 record format in the device file named SCR. The CL program will wait for the user data before it continues execution.

```
DCLF  FILE(DF1)  RCDFMT(REC8)
  •
  •
  •
SNDRCVF  DEV(&DN)  RCDFMT(REC8)  WAIT(*NO)
  •
  •
  •
WAIT  DEV(&DN)
```

This command sends and receives user data by way of the device file named DF1. Using the record format REC8, the CL program can pass data between itself and the user who is at the device named in the variable &DN, but it does not wait for a response to come back. If the program is to send and receive data from several devices (users), the same SNDRCVF command can be used. Because the CL variable &DN is used, only the value in the DEV parameter has to be changed between executions of the command. A WAIT command for each device must be issued later in the program to ensure that all the devices do respond.

SNDRPY (Send Reply) Command

The Send Reply (SNDRPY) command sends a reply message to the sender of an inquiry message. The message that is answered is the one having the specified message reference key that was received at the specified message queue. If the specified message queue is not allocated to the job in which this command is entered, it is implicitly allocated by this command for the duration of the command.



MSGKEY Parameter: Specifies the name of the CL variable in the program that contains the message reference key of the message that the reply answers.

MSGQ Parameter: Specifies the qualified name of the message queue that received the inquiry message being answered. (If no library qualifier is given, *LIBL is used to find the queue.)

RPY Parameter: Specifies the reply that the program is to send as a response to the inquiry message.

<u>*DFT:</u> The default reply stored in the message description of the inquiry message that was sent is to be sent as the reply. (For impromptu inquiry messages, *DFT is the same as *N.)

'reply-text': Enter the text (enclosed in apostrophes if it contains blanks or special characters) or a CL variable that contains the text that is to be sent as the program's reply to the inquiry message. The number of characters and their format are defined by the validity specifications given in the ADDMSGD command for the specified inquiry message. However, if no validity specifications are specified for replies in the ADDMSGD command, as many as 132 characters can be used in the reply text.

RMV Parameter: Specifies whether the inquiry message and its reply are to be removed from the specified message queue.

*YES: The message and its reply are to be removed from the message queue when the reply is sent.

NO: The message and its reply are to be retained in the message queue. The inquiry message cannot be replied to more than once, but it can be received or displayed multiple times.

Example

```
SNDRPY  MSGKEY(&KEY)  MSGQ(SMITH)  RPY(YES)
```

This command sends a reply of YES to the message whose reference key is specified by &KEY, which was received at message queue SMITH. Because the reply contains only one word, the reply does not have to be enclosed in apostrophes.

SRVJOB (Service Job) Command

The Service Job (SRVJOB) command activates the remote service mode for a specified job (other than the job issuing the command) so that other service commands can be entered to service the specified job. Any dump and trace commands can be executed in that job until service mode ends. Service mode exists until the ENDSRV (End Service) command is executed.

Restriction: To service a job, you must have operational authority for the user profile under which that job is running. However, a programming support representative (PSR) operating from the console may service any job.

```
                                                                    Required
SRVJOB────── JOB job-name[.user-name[.job-number]]─────
                                                            Job:B,I Pgm:B,I
```

JOB Parameter: Specifies the qualified name of the job to be serviced. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. If duplicates of the specified name are found, messages are sent to the user, and a qualified job name must be specified. The job name entered cannot be the name of the job issuing the command. (For an expanded description of the JOB parameter, see Appendix A.)

Example

```
SRVJOB JOB(ABCD)
```

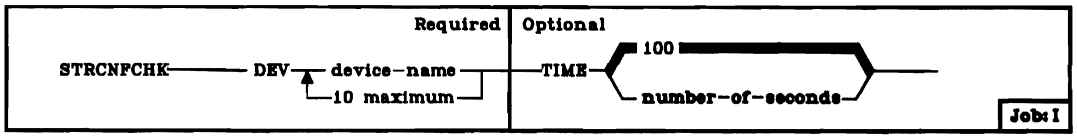
This command activates the remote service mode so that any trace or dump commands entered in this job will be applied to the job named ABCD.

STRCNFCHK (Start Confidence Check) Command

The Start Confidence Check (STRCNFCHK) command is used to exercise the 5424 MFCU, the 3203, 3262, or 5211 printers, the 3410/3411 tape units, and the diskette magazine drive on the system to ensure that they are functioning properly with the rest of the system. Any combination of the devices can be exercised with I/O operations as long as they are not being used by another job.

This command can only be used interactively. While it is executing, it presents a display at the work station through which the user can hold, release, or cancel any device that is being exercised. When the command has completed execution, it presents a message on the display. The user can then terminate the command by pressing the CF1 key.

Restriction: You must be authorized to use every device specified.



DEV Parameter: Specifies the names of one or more of the MFCU, 3203, 3262, or 5211 printers, 3410 tape units, 3411 tape control unit, and diskette magazine drive on the system. Enter the names specified in their respective device descriptions.

TIME Parameter: Specifies the number of seconds that the specified devices are to be exercised.

<u>100:</u> Each device specified is to be exercised for 100 seconds.

number-of-seconds: Enter a value, in seconds, that indicates how long each device is to be exercised. The specified value cannot be less than 60.

Examples

```
STRCNFCHK  DEV(QSYSPRT QCARD96) TIME(240)
```

This command exercises the 5211 Printer named QSYSPRT and the MFCU named QCARD96. The 5211 prints lines continuously. The MFCU punches and prints on blank cards from hopper 1. The punched cards are routed in stacker 1. The punched cards are removed from stacker 1 and placed in hopper 2 from which they are read and verified. The cards are then routed to stacker 4. The data printed and punched represents a pattern. The pattern is repeated on both devices until time runs out at 240 seconds.

```
STRCNFCHK  DEV(QDKT QTAPE1)
```

This command exercises the diskette magazine drive named QDKT and a tape unit named QTAPE1. A diskette initialized on the System/38 in basic exchange format is placed in manual slot *S1. A tape, initialized as an unlabeled tape on the System/38, is loaded on the tape unit QTAPE1. Data records are written to a file on the diskette and (independently) to a file on tape. After several records have been written, the records are read back and the data is verified. This process is repeated until time runs out (after 100 seconds).

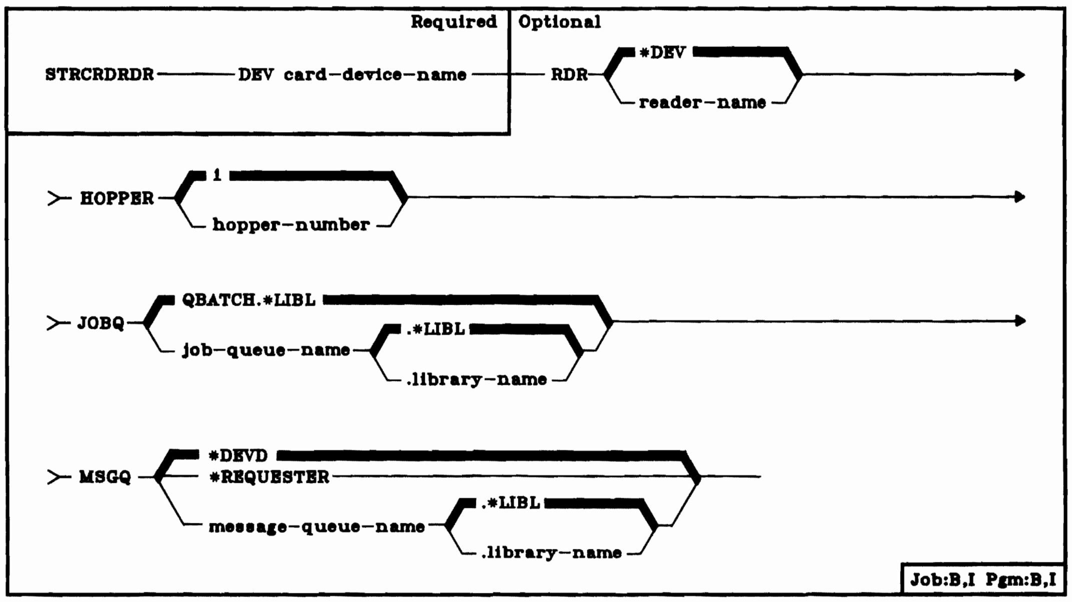
STRCRDRDR (Start Card Reader) Command

The Start Card Reader (STRCRDRDR) command starts a spooling reader to the specified card device to read a batch input stream into the system. This command specifies the name of the card device from which the cards are read, the name of the reader, the hopper number, and the names of the job queue and message queue to be used.

More than one reader, which is a system job, can be active concurrently (as determined by the spooling subsystem description). Each reader must have a unique reader name and have its own input device assigned to it. A reader that has been started can be actively reading input or waiting for device input. The reader can also be held or canceled if the HLDRDR or CNLRDR command is used. The card reader can be terminated only by the CNLRDR command. The batch input stream should end with a /* in positions 1 and 2 of the last card to ensure that the last job in the input stream is processed.

Because each reader runs independently of the job that started it, the user can continue doing other work on the system after he has started a reader. The reader is owned by the user who issues the STRCRDRDR command.

Restriction: To use this command you must have operational authority for the job descriptions associated with the jobs read by the reader.



DEV Parameter: Specifies the name of the card device to be used to read in an input stream.

RDR Parameter: Specifies the name of the spooling reader being started. Each reader name must be unique.

<u>*DEV:</u> The name of the reader is to be the same as the name of the card device specified in the DEV parameter.

reader-name: Enter the name by which the reader being started is to be identified.

HOPPER Parameter: Specifies from which hopper of the MFCU the cards are to be read by this spooling reader. Valid entries are 1 (for the primary hopper) and 2 (for the secondary hopper).

<u>1:</u> The primary hopper is to be used with this reader.

hopper-number: Enter either a 1 or a 2 to indicate which MFCU hopper is to be used.

JOBQ Parameter: Specifies the name of the job queue on which the spooling reader places job entries. An entry is placed on this queue for each job in the input stream that specifies JOBQ(*RDR) on its JOB command. If *RDR is not specified on the JOB command, the job queue specified here is ignored and the job queue named in the JOB command or in the job description is used. (Note that, if *RDR is not specified, the job queue for each job in the input stream can be different.)

Restriction: If the user identified in the job description of the job being read is not authorized for the job queue used by the reader for a job in the input stream, an error message is sent to the message queue specified by MSGQ and that job is not executed. The reader, however, continues processing the input stream starting with the next job.

<u>QBATCH:</u> The job entry is to be placed on the QBATCH job queue in the QGPL library. QBATCH is the default if JOBQ(*RDR) is specified on the JOB command.

qualified-job-queue-name: Enter the qualified name of the job queue to which each job read by this reader is to be sent if JOBQ(*RDR) is specified on its JOB command. (If no library qualifier is given, *LIBL is used to find the queue.)

MSGQ Parameter: Specifies the name of the message queue to which messages generated by the reader are to be sent.

<u>*DEVD:</u> The messages are to be sent to the message queue specified in the device description of the device named in the DEV parameter.

**REQUESTER:* The messages are to be sent to the message queue of the user who started the reader. This value is not valid for batch jobs.

qualified-message-queue-name: Enter the qualified name of the message queue to which messages generated by the reader are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
STRCRDRDR  DEV(MFCU1)
```

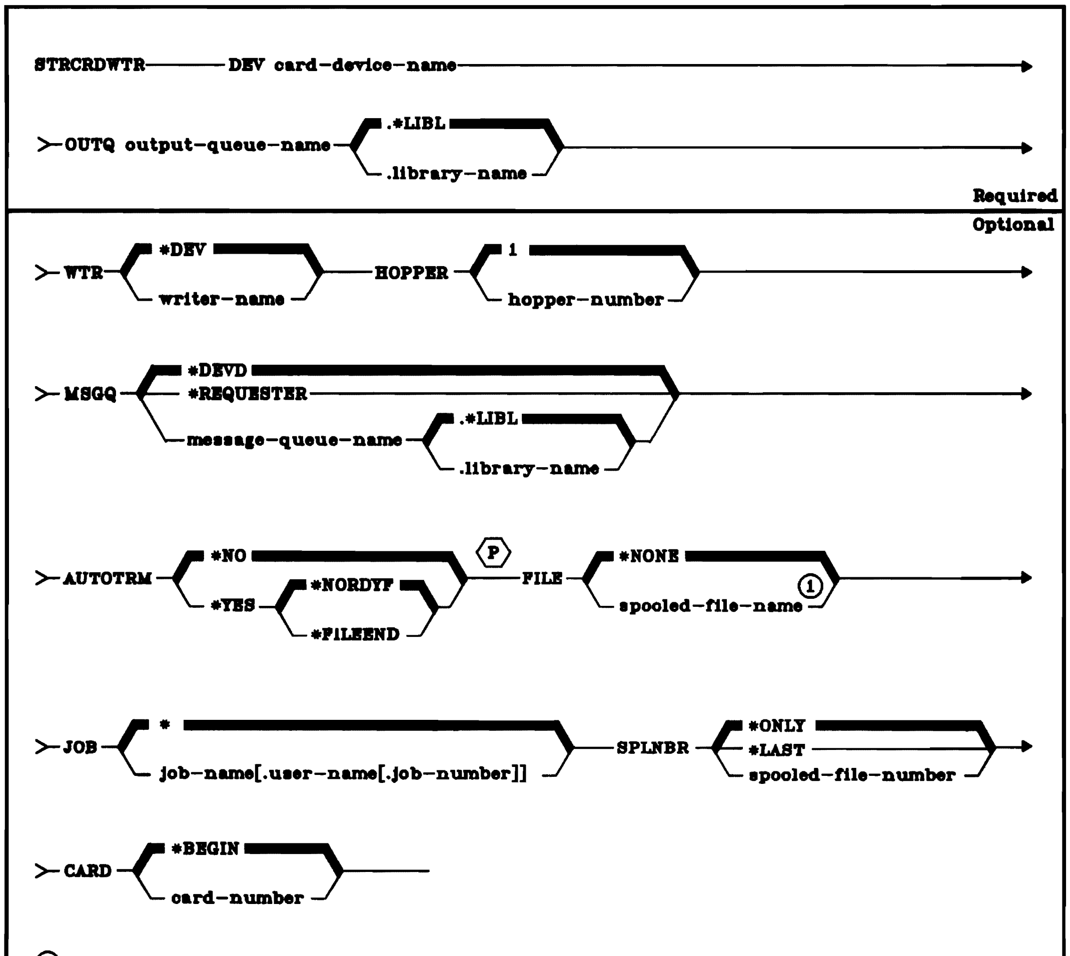
This command starts a spooling reader to the device named MFCU1. The reader is named MFCU1 because the RDR parameter was not specified. The cards containing the input stream are read from hopper 1 of the MFCU, and the records are sent to the QBATCH job queue unless the JOB card for any job, or a job's job description, specifies otherwise. The job entries are placed on the default job queue QBATCH for jobs whose JOB command specifies *RDR for the job queue name. Messages generated by the reader are sent to the system operator message queue.

STRCRDWTR (Start Card Writer) Command

The Start Card Writer (STRCRDWTR) command starts a spooling writer to a card device. The writer, which is a system job, takes spooled output files from an output queue and produces (writes) the output on the specified card device. This command specifies the name of the writer, the names of the output and message queues used, and the name of the card device used.

More than one writer can be active concurrently (as determined by the spooling subsystem description). Each writer must have a unique writer name, its own output queue, and its own device. A writer that has been started can be actively producing output or waiting for a file entry to be put on the output queue. Optionally, the writer can be automatically terminated when it has written all the files having entries on the output queue. The writer can also be held or canceled if the HLDWTR or CNLWTR command is used.

Because each writer runs independently of the job that started it, the user can continue doing other work on the system after he has started a writer. The writer is owned by the user who issued the STRCRDWTR command.



① The JOB, SPLNBR, and CARD parameters are valid only if a spooled file name is specified for FILE.

Job:B,I Pgm:B,I

DEV Parameter: Specifies the name of the card device to be used to punch and/or print the spooled output.

OUTQ Parameter: Specifies the qualified name of the output queue from which the writer is to process spooled output files. (If no library qualifier is given, *LIBL is used to find the queue.) The output queue must be available before the writer can be started.

WTR Parameter: Specifies the name of the spooling writer being started. Each writer name must be unique.

<u>*DEV:</u> The name of the writer is to be the same as the name of the card device specified in the DEV parameter.

writer-name: Enter the name by which the writer being started is to be identified.

HOPPER Parameter: Specifies which hopper of the MFCU contains the cards to be punched or printed. Valid entries are 1 (for the primary hopper) and 2 (for the secondary hopper).

<u>1:</u> The primary hopper is to be used with this writer.

hopper-number: Enter either a 1 or a 2 to indicate which MFCU hopper is to be used.

MSGQ Parameter: Specifies the name of the message queue to which messages generated by the writer are to be sent.

<u>*DEVD:</u> The messages are to be sent to the message queue specified in the device description of the device named in the DEV parameter.

**REQUESTER:* The messages are to be sent to the message queue of the user who started the writer. This value is not valid for batch jobs.

qualified-message-queue-name: Enter the qualified name of the message queue to which messages generated by the writer are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

AUTOTRM Parameter: Specifies whether the writer is to be terminated automatically when the output queue has no more available entries for spooled output files to be written to the card device.

*NO: The writer is not to be terminated when the last available entry has been removed from the output queue; it is to wait for another spooled file entry to be put on the queue.

**YES:* If *NORDYF (no ready file) is specified, the writer is to be automatically terminated when all the available entries have been removed from the output queue. If *FILEEND is specified, the writer is terminated after it has finished processing one spooled output file.

FILE Parameter: Specifies the name of the first (or only) spooled output file to be processed by the spooling writer and written to the card device. If several files are available on the output queue, the next file produced is the first one available with the highest priority.

*NONE: No spooled file name is to be specified; the first spooled file that becomes available on the output queue is to be processed first.

spooled-file-name: Enter the name of the spooled output file that is to be the first (or only) output file to be written to the card device. The file name is the same as the name of the device file used by the program to produce the spooled file.

JOB Parameter: Specifies the qualified name of the job that created the spooled file to be written to the card device. This parameter is valid only if a spooled file name is specified in the FILE parameter.

*: The job in which this STRCRDWTR command is entered is the job that created the spooled file.

qualified-job-name: Enter the qualified name of the job that created the spooled file. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file to be processed first. This parameter is valid only if a spooled file name is specified in the FILE parameter.

*ONLY: Only one spooled output file from the job on the specified output queue has the specified file name; no spooled file number is needed. If *ONLY is specified and more than one spooled file on the output queue has the specified name, an error message is displayed to the user.

**LAST:* The highest numbered spooled output file with the specified file name on the specified output queue is the file to be processed first.

spooled-file-number: Enter the number of the specified file on the specified output queue that is to be processed first.

CARD Parameter: Specifies the card number of the first card to be punched (or printed) by the spooling writer. The card number is actually the relative record number of records within the output file. For example, if CARD(100) is specified, the 100th record (from the beginning of the file) will be the first record to be punched in a card when the card writer is started.

This parameter is valid only if a spooled file name is specified in the FILE parameter.

*BEGIN: The first record in the file is to be punched in the first card punched by the card device.

card-number: Enter the relative record number of the record in the file that is to be punched in the first card by the card device.

Example

```
STRCRDWTR  DEV(QCARD96) OUTQ(QPUNCH)
```

This command starts a spooling writer to a card device named QCARD96. The name of the writer is also QCARD96 because WTR(*DEV) is the default and assumes the name of the card device. Blank cards must be placed in hopper 1 of the MFCU to be punched. The spooled output files identified on the QPUNCH output queue are processed by the writer. Any writer messages are sent to the system operator's message queue, and the writer is to wait for more output when the queue is emptied.

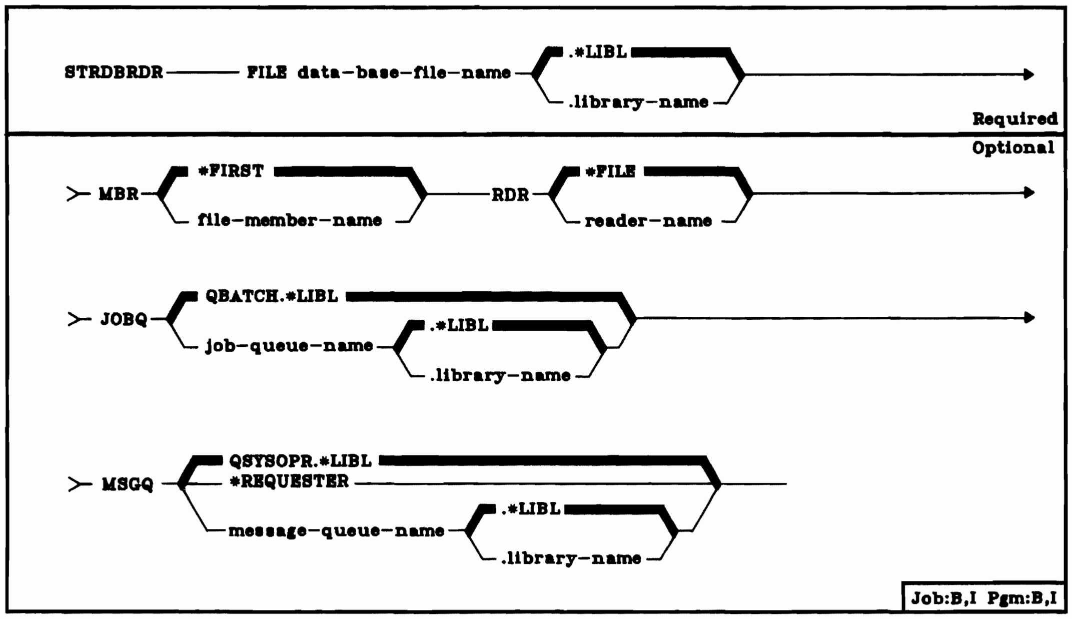
STRDBRDR (Start Data Base Reader) Command

The Start Data Base Reader (STRDBRDR) command starts a spooling reader to a data base file; the reader reads a batch input stream from the data base and places the jobs onto one or more job queues. This command specifies the name of the data base file and member from which the input stream is read, the name of the reader, and the names of the job queue and message queue to be used.

More than one reader can be active concurrently (as determined by the spooling subsystem description). Each data base reader must have a unique reader name, and the specified file or member must be available. A data base reader that has been started can be actively reading input from the data base or waiting for more input from the same file. The reader can also be held or canceled if the HLDRDR or CNLRDR command is used.

Because each reader runs independently of the job that started it, the user can continue doing other work on the system after he has started a reader. The reader is owned by the user who issues the STRDBRDR command.

Restrictions: The specified data base file must consist of single field records and must have an arrival sequence access path, or it must be a standard data base source file. To read the specified data base file, you must have read rights for the file and operational authority for the job descriptions associated with the jobs read by the reader.



FILE Parameter: Specifies the qualified name of the data base file from which the input stream is to be read. (If no library qualifier is given, *LIBL is used to find the file.) The file must be available for allocation to the spooling reader before the reader can be started.

MBR Parameter: Specifies the name of the member in the specified file that contains the input stream to be read.

*FIRST: No member name is specified; the first member in the file is to be used.

member-name: Enter the name of the member that contains the input stream to be read by the reader.

RDR Parameter: Specifies the name of the spooling reader being started. Each reader name must be unique.

*FILE: The name of the reader is to be the same as the name of the data base file specified by the FILE parameter.

reader-name: Enter the name by which the reader being started is to be identified.

JOBQ Parameter: Specifies the name of the job queue on which the spooling reader places job entries. An entry is placed on this job queue for each job in the input stream that specifies JOBQ(*RDR) on its JOB command. If *RDR is not specified on the JOB command, the job queue specified in the JOB command or in the job description is used. (Note that the job queue for each job in the input stream can be different.)

Restriction: If the user starting the reader is not authorized for the job queue used by the reader for a job in the input stream, an error message is sent to the message queue specified by MSGQ and the job is not executed.

QBATCH: The job entry is to be placed on the QBATCH job queue in the QGPL library.

qualified-job-queue-name: Enter the qualified name of the job queue to which each job read by this reader is to be sent if JOBQ(*RDR) is specified on its JOB command. (If no library qualifier is given, *LIBL is used to find the queue.)

MSGQ Parameter: Specifies the name of the message queue to which messages generated by the reader are to be sent.

QSYSOPR: The messages are to be sent to the system operator's message queue, QSYSOPR.

**REQUESTER:* The messages are to be sent to the message queue of the user who started the reader. This value is not valid for batch jobs.

qualified-message-queue-name: Enter the qualified name of the message queue to which any messages generated by the reader are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
STRDBRDR  FILE(BILLING.QGPL)
```

This command starts a spooling reader that reads its input from the data base file named BILLING, which is in the QGPL library. The reader name is also BILLING because the RDR parameter was not specified. The first member in the BILLING file contains the input stream to be processed. The default job queue QBATCH and the message queue QSYSOPR are used by the data base reader.

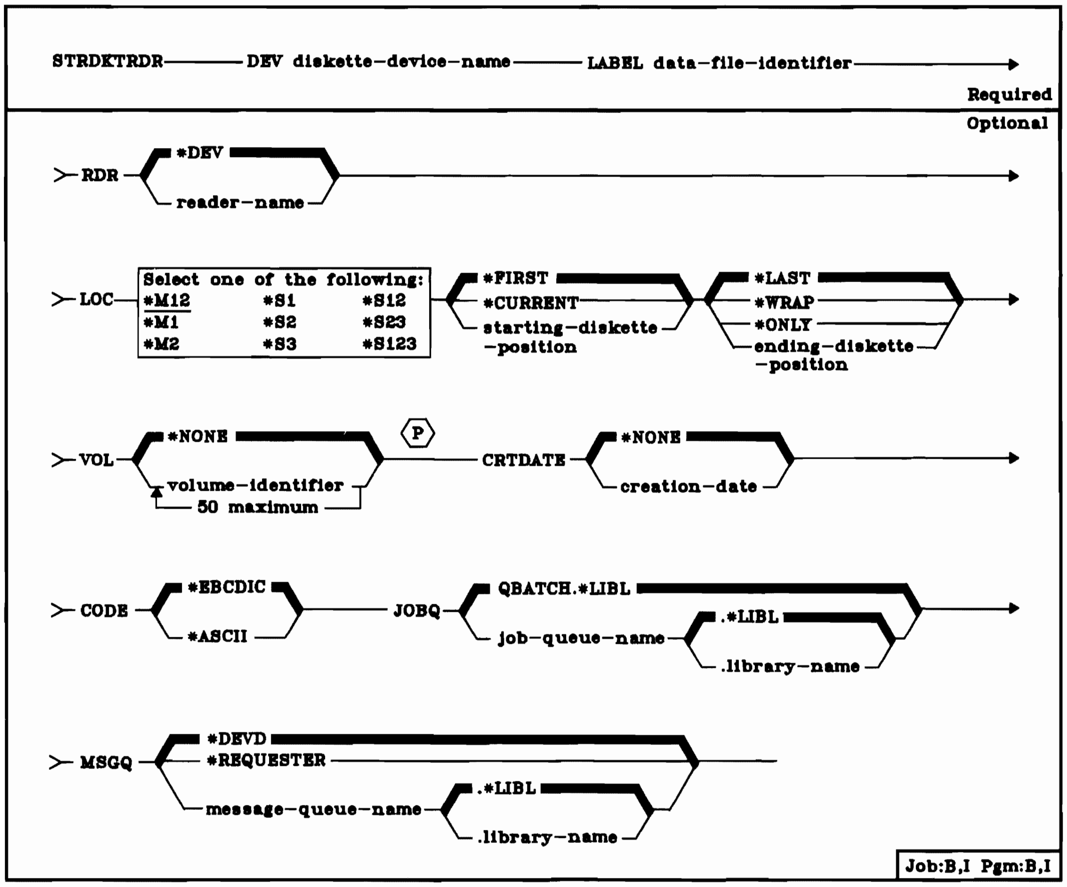
STRDKTRDR (Start Diskette Reader) Command

The Start Diskette Reader (STRDKTRDR) command starts a spooling reader to the specified diskette device to read a batch input stream into the system. This command specifies the name of the diskette device from which the input stream is read; the location, volume, and data file that the input stream is on; and the names of the job queue and message queue to be used.

More than one reader, which is a system job, can be active concurrently (as determined by the spooling subsystem description). Each reader must have a unique reader name and have its own input device assigned to it. A reader that has been started can be actively reading input or waiting for device input. The reader can also be held or canceled if the HLDRDR or CNLRDR command is used. The reader is terminated at end-of-file.

Because each reader runs independently of the job that started it, the user can continue doing other work on the system after he has started a reader. The reader is owned by the user who issues the STRDKTRDR command.

Restrictions: To use this command, you must have operational authority for the job descriptions associated with the jobs read by the reader. This command cannot be used to read data files of diskettes that are in the save/restore format.



DEV Parameter: Specifies the name of the diskette device to be used to read in an input stream.

LABEL Parameter: Specifies the data file identifier of the file to be processed as an input stream. (For an expanded description of the LABEL parameter, see Appendix A.) Enter the data file identifier, which cannot exceed 8 alphameric characters, of the file on diskette.

RDR Parameter: Specifies the name of the spooling reader being started. Each reader name must be unique.

*DEV: The name of the reader is to be the same as the name of the diskette device specified in the DEV parameter.

reader-name: Enter the name by which the reader being started is to be identified.

LOC Parameter: Specifies which diskette locations in the magazines or slots are to be read from. Three values are needed: (1) the unit type and location (that is, the magazines or slots used), (2) the starting diskette position, and (3) the ending diskette position in the unit. (For an expanded description of the LOC parameter, see Appendix A.) If LOC is not specified, *M12, *FIRST, and *LAST are assumed by the system.

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit and location are to be read from. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be read first. Enter one of the following values to specify the starting diskette position:

*FIRST: The first diskette position in the location contains the diskette to be read first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be read first.

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the first diskette to be read. (A value is not valid for manual slots.)

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be read last. Enter one of the following values to specify the ending diskette position:

*LAST: The last diskette position in the location contains the diskette to be read last. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**WRAP:* If the end of the last diskette in the location is reached before the end of the data file is reached, a message is sent to the system operator to mount another magazine or diskette to continue. (See Appendix A for details and restrictions on using *WRAP.)

**ONLY:* Only the diskette position specified by the second value is to be read, and read only once.

ending-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the last diskette to be read. (A value is not valid for manual slots.)

VOL Parameter: Specifies the volume identifiers of the diskette volumes (either magazines or slots) that contain the input stream to be processed. The volumes must be mounted in the same order that the volume identifiers are specified in this parameter. (For an expanded description of the VOL parameter, see Appendix A.)

*NONE: No volume identifier is specified. The input stream begins on the first diskette in the location specified in the LOC parameter.

volume-identifier: Enter the identifiers of one or more volumes in the order that they are to be mounted and read. Each identifier can have 6 alphameric characters or less.

CRTDATE Parameter: Specifies when the diskette data file was created on diskette; the creation date should not be specified if the date is not to be checked. If the date written on the diskette containing the data file does not match the date specified here, an error message is sent to the message queue named in the MSGQ parameter.

*NONE: The creation date is not specified; no check is to be made.

creation-date: Enter the creation date of the data file to be read. The date should be in the format specified by the QDATFMT system value.

CODE Parameter: Specifies the type of character code to be used to read the diskette data into the job queue.

*EBCDIC: The diskette data file is written in the EBCDIC character code.

**ASCII:* The diskette data file is written in the ASCII character code.

JOBQ Parameter: Specifies the name of the job queue on which the spooling reader places job entries. An entry is placed on the queue specified here only for each job in the input stream that specifies JOBQ(*RDR) on its JOB command. If *RDR is not specified on the JOB command, the job queue specified here is ignored and the job queue named in the JOB command or in the job description is used. (Note that, if *RDR is not specified, the job queue for each job in the input stream can be different.)

Restriction: If the user starting the diskette reader is not authorized for the job queue used by the reader for a job in the input stream, an error message is sent to the message queue specified by MSGQ and the job is not executed.

QBATCH: The job entry is to be placed on the QBATCH job queue in the QGPL library.

qualified-job-queue-name: Enter the qualified name of the job queue to which each job read by this reader is to be sent if JOBQ(*RDR) is specified on its JOB command. (If no library qualifier is given, *LIBL is used to find the queue.)

MSGQ Parameter: Specifies the name of the message queue to which messages generated by the diskette reader are to be sent.

*DEVD: The messages are to be sent to the message queue specified in the device description of the device named in the DEV parameter.

**REQUESTER:* The messages are to be sent to the message queue of the user who started the reader. This value is not valid for batch jobs.

qualified-message-queue-name: Enter the qualified name of the message queue to which diskette reader messages generated by the reader are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

Example

```
STRDKTRDR  DEV(QDKT) LABEL(OCT24) LOC(*S12) +
      VOL(SALES)
```

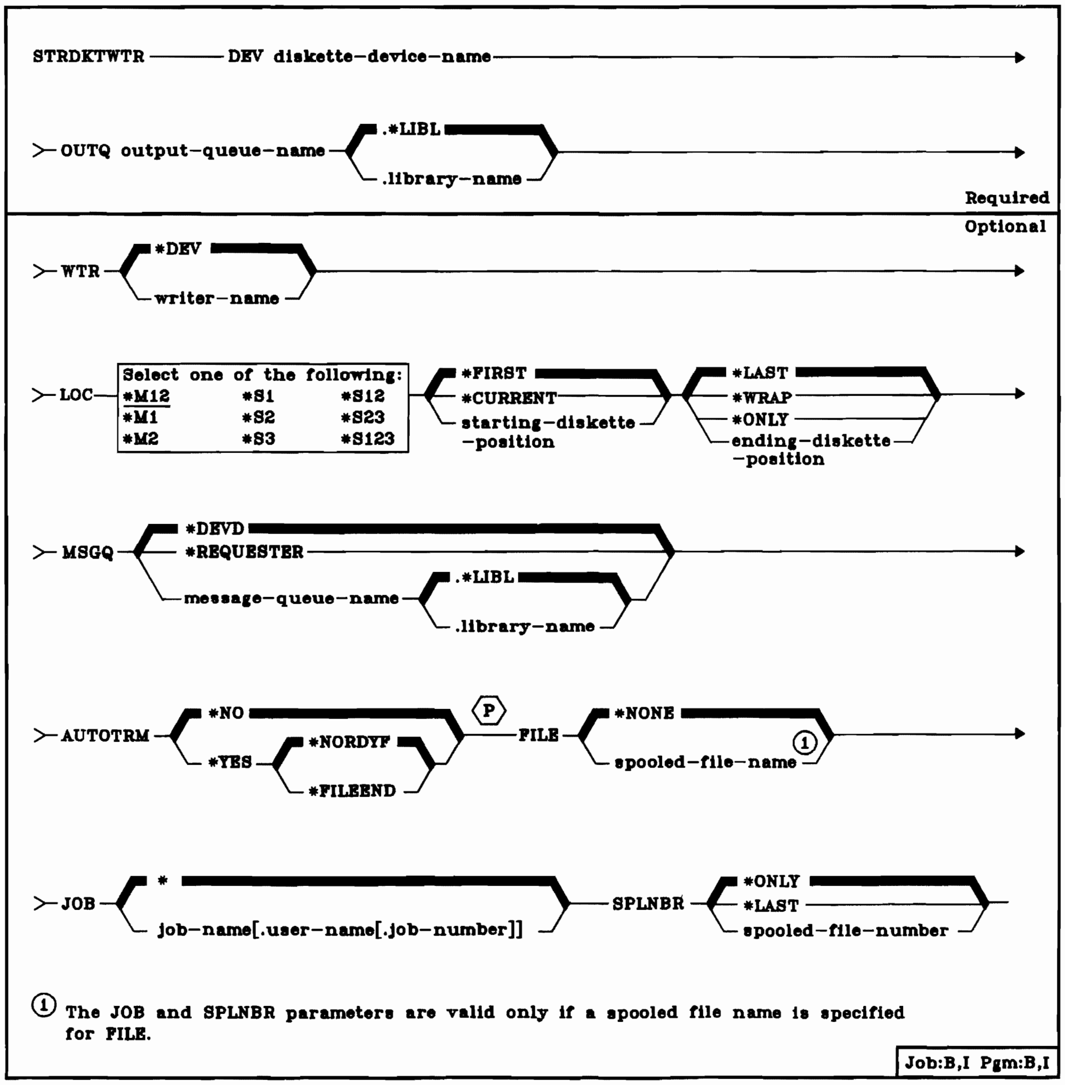
This command starts the spooling reader named QDKT, which reads diskette input from the device QDKT. (Because *DEV was the default on the unspecified RDR parameter, the device name QDKT is also used as the reader name.) The reader reads its input from the data file named OCT24 that is located on the single diskettes placed in manual slots S1 and S2, whose volume identifiers must be SALES. The default job queue QBATCH and the message queue QSYSOPR are used by the diskette reader.

STRDKTWTR (Start Diskette Writer) Command

The Start Diskette Writer (STRDKTWTR) command starts a spooling writer to the specified diskette device. The writer, which is a system job, takes spooled output files from an output queue and produces (writes) the output on the diskette device. This command specifies the names of the diskette device and the writer, the names of the output and message queues used, and the location on diskette(s) where the output is to be written.

More than one writer can be active concurrently (as determined by the spooling subsystem description). Each writer must have a unique writer name, its own output queue, and its own device. A writer that has been started can be actively producing output or waiting for a file entry to be put on the output queue. Optionally, the writer can be automatically terminated when it has processed all the files having entries on the output queue. The writer can also be held or canceled if the HLDWTR or CNLWTR command is used.

Because each writer runs independently of the job that started it, the user can continue doing other work on the system after he has started a writer.



① The JOB and SPLNBR parameters are valid only if a spooled file name is specified for FILE.

DEV Parameter: Specifies the name of the diskette device on which the spooled output is to be written.

OUTQ Parameter: Specifies the qualified name of the output queue from which the writer is to process spooled output files. (If no library qualifier is given, *LIBL is used to find the queue.) The output queue must be available before the writer can be started.

WTR Parameter: Specifies the name of the spooling writer being started. Each writer name must be unique.

*DEV: The name of the writer is to be the same as the name of the diskette device specified in the DEV parameter.

writer-name: Enter the name by which the writer being started is to be identified.

LOC Parameter: Specifies which diskette locations in the magazines or slots are to be written on. Three values are needed: (1) the unit type and location (that is, the magazines or slots used), (2) the starting diskette position, and (3) the ending diskette position in the unit. (For an expanded description of the LOC parameter, see Appendix A.) If LOC is not specified, *M12, *FIRST, and *LAST are assumed by the system.

Unit Type and Location: The first of the three values in the LOC parameter specifies which unit and location are to be written on. Enter one of the following values to specify the unit type and location: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, or *S123.

Starting Diskette Position: The second of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be written on first. Enter one of the following values to specify the starting diskette position:

*FIRST: The first diskette position in the location contains the diskette to be written on first. It is the leftmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be written on.

starting-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the first diskette to be written on. (A value is not valid for manual slots.)

Ending Diskette Position: The third of the three values in the LOC parameter specifies which diskette position, in a location having more than one diskette, contains the diskette to be written on last. Enter one of the following values to specify the ending diskette position:

*LAST: The last diskette position in the location contains the diskette to be written on last. It is the rightmost diskette in the magazine(s) or slots specified. (See Appendix A for details.)

**WRAP:* If the end of the last diskette in the location is reached before the end of the data file is reached, a message is sent to the system operator to mount another magazine or diskette to continue. (See Appendix A for details and restrictions on using *WRAP.)

**ONLY:* Only the diskette position specified by the second value is to be written on, and only once.

ending-diskette-position: Enter the number of the diskette position (1 through 10) in the magazine that contains the last diskette to be written on. (A value is not valid for manual slots.)

MSGQ Parameter: Specifies the name of the message queue to which messages generated by the diskette writer are to be sent.

*DEVD: The messages are to be sent to the message queue specified in the device description of the device named in the DEV parameter.

**REQUESTER:* The messages are to be sent to the message queue of the user who started the reader. This value is not valid for batch jobs.

qualified-message-queue-name: Enter the qualified name of the message queue to which any diskette writer messages are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

AUTOTRM Parameter: Specifies whether the writer is to be terminated automatically when the output queue has no more available entries for spooled output files to be written to the diskette device.

*NO: The writer is not to be terminated when the last available entry has been removed from the output queue; it is to wait for another spooled file entry to be put on the queue.

**YES:* If *NORDYF (no ready file) is specified, the writer is to be automatically terminated when all the available entries have been removed from the output queue. If *FILEEND is specified, the writer is terminated after it has finished processing one spooled output file.

FILE Parameter: Specifies the name of the first (or only) spooled output file to be processed by the spooling writer and written to diskette. If several files are available on the output queue, the next file produced is the first one available with the highest priority.

*NONE: No spooled file name is to be specified; the first spooled file that becomes available on the output queue is to be processed first.

spooled-file-name: Enter the name of the spooled output file that is to be the first (or only) output file to be written to diskette. The file name is the same as the name of the device file used by the program to produce the spooled file.

JOB Parameter: Specifies the qualified name of the job that created the spooled file to be written to diskette. This parameter is valid only if a spooled file name is specified in the FILE parameter.

*: The job in which this STRDKTWTR command is entered is the job that created the spooled file.

qualified-job-name: Enter the qualified name of the job that created the spooled file. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file to be processed first. This parameter is valid only if a spooled file name is specified in the FILE parameter.

*ONLY: Only one spooled output file from the job on the specified output queue has the specified name; no spooled file number is needed. If *ONLY is specified and more than one spooled file on the output queue has the specified name, an error message is displayed to the user.

**LAST:* The highest numbered spooled output file with the specified file name on the specified output queue is the file to be processed first.

spooled-file-number: Enter the number of the specified file on the specified output queue that is to be processed first.

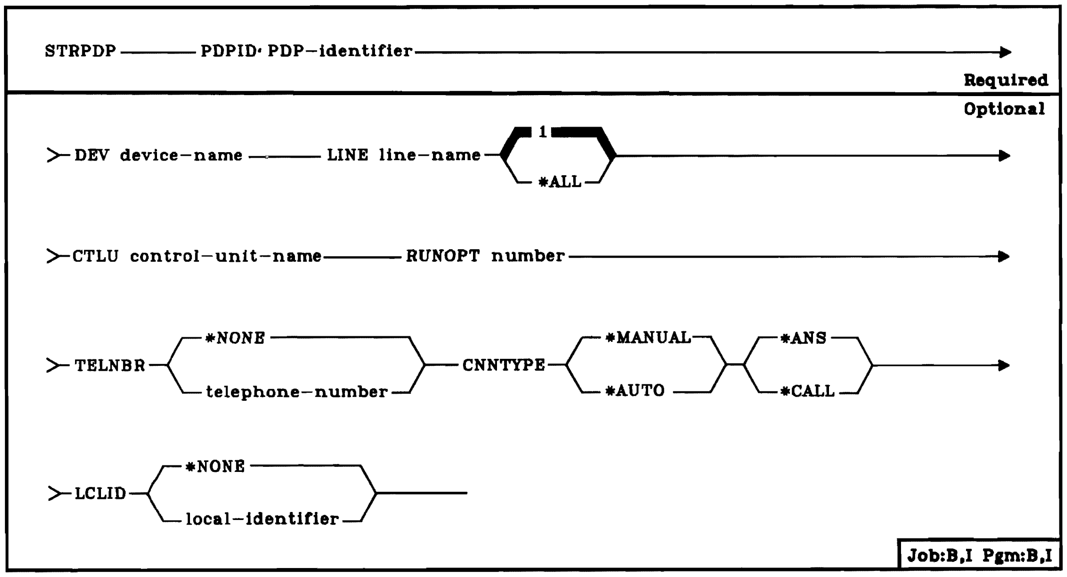
Example

```
STRDKTWTR  DEV(QDKT)  OUTQ(QDKT)  +
      LOC(*M1  *N  *WRAP)  AUTOTRM(*YES)
```

This command starts a spooling writer to the diskette magazine drive. The files written on the diskettes are on the IBM-supplied output queue QDKT. The files are to be written on the diskettes in magazine 1, starting with the first diskette in the magazine. (*N specified for the second value in the location parameter causes the default, *FIRST, to be assumed.) When all the diskettes in the magazine have been used, the system operator is notified to load another magazine. When all the files have been written (no more entries are on the QDKT output queue), the writer is automatically terminated and the diskette magazine drive is available for other uses.

STRPDP (Start Problem Determination Procedure) Command

The Start Problem Determination Procedure (STRPDP) command starts testing procedures that run below the machine interface (MI). These problem determination procedures (PDPs) are used to diagnose the source of problems in the machine product. The *System/38 Problem Determination Guide* describes the procedures and provides the PDP identifiers and examples of their use.



PDPID Parameter: Specifies the identifier of the PDP to be run. Enter the six-digit identifier of the PDP. Refer to the *System/38 Problem Determination Guide* for all of the valid identifiers.

DEV Parameter: Specifies the name of the device to be tested by the specified PDP. The name of the device must be the same one that is in the associated device description.

LINE Parameter: Specifies the name of the line to be tested by the specified PDP. Also, a value for all other lines directly associated with this line can be optionally specified to allow all the lines to be tested at once. The name of the line specified must be the same as the one that is in the associated line description.

line-name 1: Enter the name of the line to be tested. Only the primary line is to be tested if there are other lines associated with it.

*line-name *ALL:* Enter the line name identifying a line. This line and all its associated lines are to be tested by the specified PDP.

CTLU Parameter: Specifies the name of the control unit to be tested by the specified PDP. The name of the control unit must be the same as the one that is in the associated control unit description.

RUNOPT Parameter: Specifies, depending on the PDP identifier specified, either the length of time the specified PDP is to run or the number of times (from 01 to 99) that it is to be run.

TELNBR Parameter: Specifies the telephone number of the remote control unit. The number is dialed at the System/38 location to establish a switched connection with the control unit.

**NONE:* No telephone number is specified for the control unit.

telephone-number: Enter the control unit telephone number. The number can be from 1 to 16 digits in length; the allowable set of values consists of the digits 0 through 9, and two special characters, the separator character (apostrophe) and the end-of-number character (asterisk). Refer to the *Guide to Program Product Installation and Device Configuration* for information and user considerations for these two special characters with autocall equipment.

CNNTYPE Parameter: Specifies the method and location to be used to make the initial switched connection between the System/38 and the remote control unit.

Either of the following two parameter values must be specified:

**MANUAL:* The connection is to be made by dialing the telephone number.

**AUTO:* The connection is to be made by using autocall equipment.

Either of the following two parameter values must be specified:

**ANS:* The connection is to be initiated at the remote location.

**CALL:* The connection is to be initiated locally (the site at which this command is being entered).

LCLID Parameter: Specifies the local identifier for identifying the System/38 to the remote BSC control unit.

**NONE:* The local identifier is to be made null.

local-identifier: Enter a string of from 2 to 15 characters for identifying the System/38 to a remote BSC control unit. If a two-character identifier is specified, both characters must be the same. The identifier cannot contain BSC control characters.

Examples

```
STRPDP  PDPID(MP5821)  LINE(LINE1)
```

The problem determination procedure MP5821 is to be executed on the modem for the line named LINE1.

```
STRPDP  PDPID(LT0001)  CTLU(CU1)  RUNOPT(20)
```

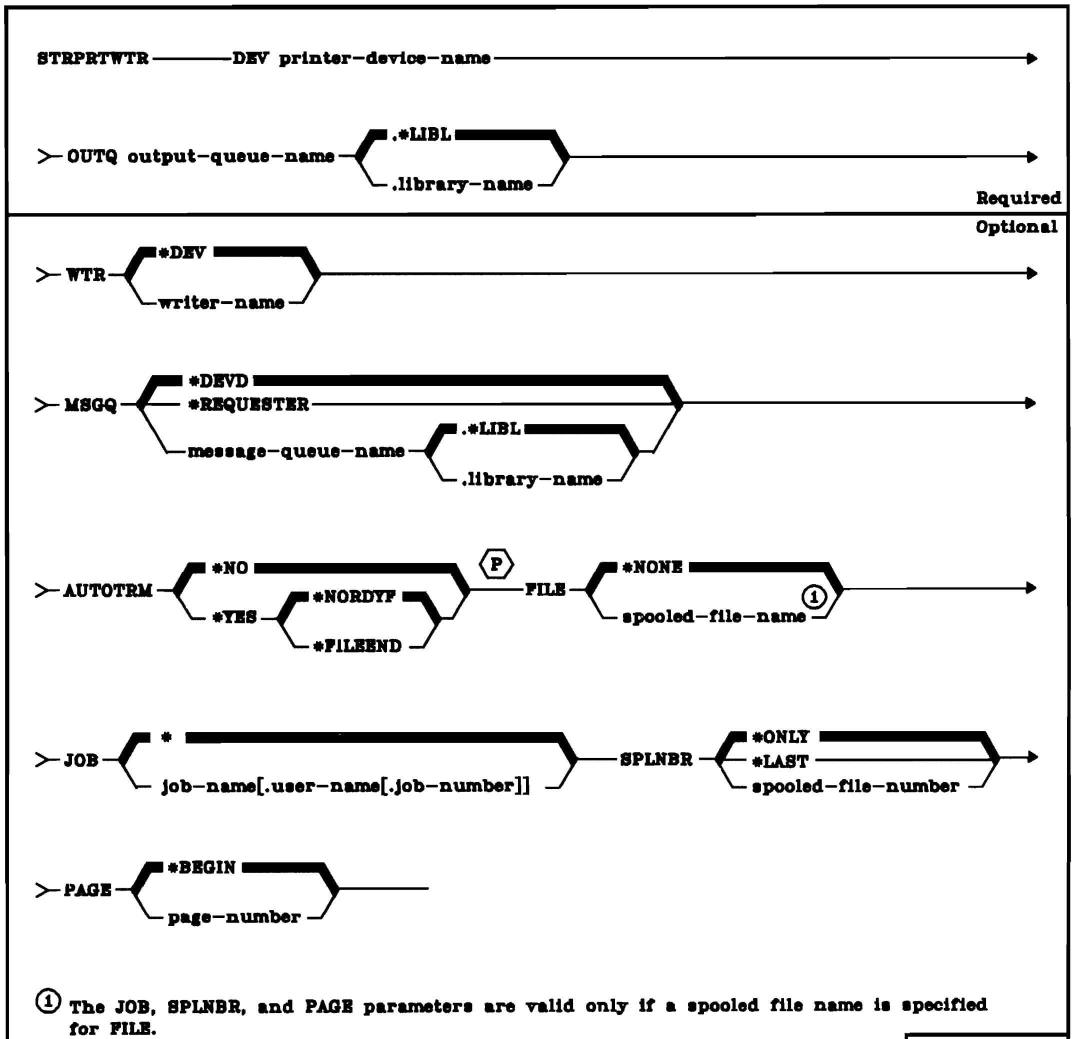
This command uses procedure LT0001 to check the line while an application program is running.

STRPRTWTR (Start Printer Writer) Command

The Start Printer Writer (STRPRTWTR) command starts a spooling writer to the specified printer. The writer, which is a system job, takes spooled output files from an output queue and produces (writes) the output on the printer device. This command specifies the name of the printer, the names of the output and message queues used, and the name of the writer.

More than one writer can be active concurrently (as determined by the spooling subsystem description). Each writer must have a unique writer name, its own output queue, and its own device. A writer that has been started can be actively producing output or waiting for a file entry to be put on the output queue. Optionally, the writer can be automatically terminated when it has processed all the files having entries on the output queue. The writer can also be held or canceled if the HLDWTR or CNLWTR command is used.

Because each writer runs independently of the job that started it, the user can continue doing other work on the system after he has started a writer. The writer is owned by the user who issued the STRPRTWTR command.



① The JOB, SPLNBR, and PAGE parameters are valid only if a spooled file name is specified for FILE.

Job:B,I Pgm:B,I

DEV Parameter: Specifies the name of the printer device to be used to print the spooled output.

OUTQ Parameter: Specifies the qualified name of the output queue from which the writer is to process spooled output files. (If no library qualifier is given, *LIBL is used to find the queue.) The output queue must be available before the writer can be started.

WTR Parameter: Specifies the name of the spooling writer being started. Each writer name must be unique.

*DEV: The name of the writer is to be the same as the name of the printer device specified in the DEV parameter.

writer-name: Enter the name by which the writer being started is to be identified.

MSGQ Parameter: Specifies the name of the message queue to which messages generated by the writer are to be sent.

*DEVD: The messages are to be sent to the message queue specified in the device description of the device named in the DEV parameter.

**REQUESTER:* The messages are to be sent to the message queue of the user who started the writer. This value is not valid for batch jobs.

qualified-message-queue-name: Enter the qualified name of the message queue to which messages generated by the writer are to be sent. (If no library qualifier is given, *LIBL is used to find the queue.)

AUTOTRM Parameter: Specifies whether the writer is to be terminated automatically when the output queue has no more available entries for spooled output files to be written to the printer device.

*NO: The writer is not to be terminated when the last available entry has been removed from the output queue; it is to wait for another spooled file entry to be put on the queue.

**YES:* If *NORDYF (no ready file) is specified, the writer is to be automatically terminated when all the available entries have been removed from the output queue. If *FILEEND is specified, the writer is terminated after it has finished processing one spooled output file.

FILE Parameter: Specifies the name of the first (or only) spooled output file to be processed by the spooling writer and printed on the printer. If several files are available on the output queue, the next file produced is the first one available with the highest priority.

*NONE: No spooled file name is to be specified; the first spooled file that becomes available on the output queue is to be processed first.

spooled-file-name: Enter the name of the spooled output file that is to be the first (or only) output file to be written to the printer. The file name is the same as the name of the device file used by the program to produce the spooled file.

JOB Parameter: Specifies the qualified name of the job that created the spooled file to be written to the printer. This parameter is valid only if a spooled file name is specified in the FILE parameter.

*: The job in which this STRPRTWTR command is entered is the job that created the spooled file.

qualified-job-name: Enter the qualified name of the job that created the spooled file. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name. (For an expanded description of the JOB parameter and duplicate job names, see Appendix A.)

SPLNBR Parameter: Specifies the number of the spooled output file to be processed first. This parameter is valid only if a spooled file name is specified in the FILE parameter.

*ONLY: Only one spooled output file from the job on the specified output queue has the specified file name; no spooled file number is needed. If *ONLY is specified and more than one spooled file on the output queue has the specified name, an error message is displayed to the user.

**LAST:* The highest numbered spooled output file with the specified file name on the specified output queue is the file to be processed first.

spooled-file-number: Enter the number of the specified file on the specified output queue that is to be processed first.

PAGE Parameter: Specifies the page number of the first page to be printed by the spooling writer. (The relative record number of the first record in the output file to be printed on the specified page is determined by the system, based on such factors as heading lines, skipped lines, and records per page.)

This parameter is valid only if a spooled file name is specified in the FILE parameter.

***BEGIN:** The first page of records in the file is the first page to be printed.

page-number: Enter the number of the first page to be printed.

Example

```
STRPRTWTR  DEV(QSYSPRT) OUTQ(QPRINTS) +
      WTR(TOM)
```

This command starts a spooling writer named TOM that is to print the output on the printer named QSYSPRT that is taken from the output queue named QPRINTS. Any writer messages are sent to the system operator's message queue, and the writer is to wait for more output when the queue is emptied.

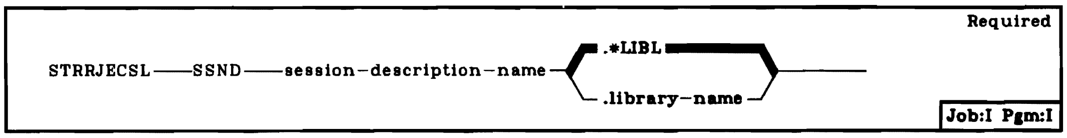
STRRJECSL (Start RJE Console) Command

The Start RJE Console (STRRJECSL) command causes the RJEF console messages to be displayed at the work station where the STRRJECSL command was issued. This command can be issued interactively only from either the System/38 console or a System/38 work station.

The associated RJEF session does not need to be active when this command is issued. However, if the RJEF session is active, you can enter host system commands.

Restriction: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The Start RJE Console (STRRJECSL) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



SSND Parameter: Specifies the qualified name of the session description from which you want to get the RJEF message queue to interactively display host system messages. If the RJEF session is active, you can enter host system commands. If no library qualifier is given, *LIBL is used to find the session description.

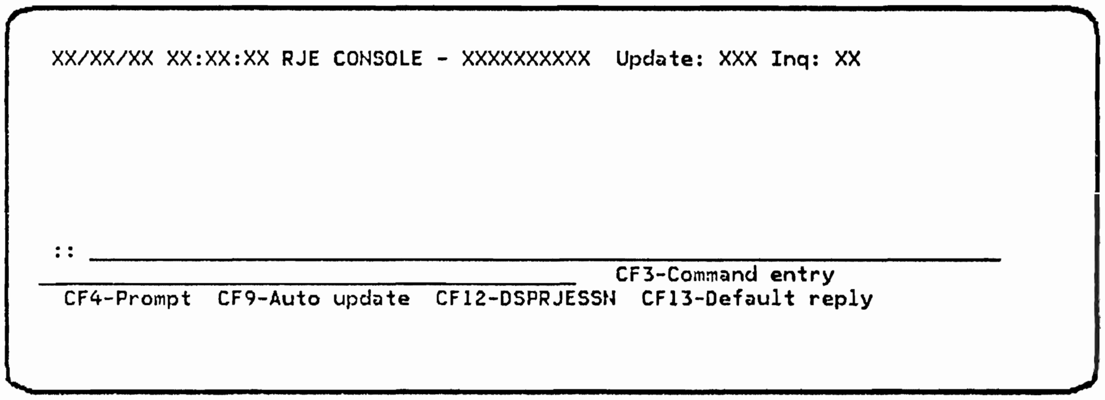
Example

```
STRRJECSL SSND(RJE.USERLIB)
```

This command starts an RJEF console. Messages from the RJEF message queue specified in the session description named RJE in library USERLIB are displayed. If the RJEF session is active, commands can be sent to the host system.

Additional Considerations

The start RJE console display appears when a System/38 work station user issues the Start RJE Console (STRRJECSL) command.



```
XX/XX/XX XX:XX:XX RJE CONSOLE - XXXXXXXXXX  Update: XXX Inq: XX




:: ________________________________________________________________________
___________________________________________ CF3-Command entry
 CF4-Prompt  CF9-Auto update  CF12-DSPRJESSN  CF13-Default reply
```

The middle portion of the display contains messages from the host system and those generated by RJEF. New messages are displayed at the bottom of the middle portion of the display and roll upward until they roll off the the top of the display. Messages that have rolled off the top of the display can be retrieved by using the Roll Down key. Using the roll keys causes automatic update to be turned off (refer to the CF9 description of this display for additional information).

If a message has been truncated on the display, the entire message can be displayed by placing the cursor on the line of the truncated message and pressing the Help key.

To respond from the RJEF console display to a System/38 inquiry message, perform the following steps:

1. Press the CF9 key to set the automatic update to no.
2. Position the cursor to the inquiry message reply field.
3. Enter the reply or press the CF12 key.
4. Press the Enter key.
5. Press the CF9 key to set the automatic update back to yes.

Note: Steps 4 and 5 can be combined by just pressing the CF9 key.

Commands that have been used are also displayed. The messages and commands shown on the middle portion of the display are in chronological order.

The bottom portion of the display is the input area to be used for entering commands and responding to messages. Up to 120 characters can be entered as console data to be sent to the host system. Commands entered at the work station (or console) keyboard retain the double colon prefix when displayed on the middle portion of the display.

The following command function (CF) keys are used with the start RJE console display:

CF1	Causes an exit from the RJEF console display.
CF3	Causes the CL command entered to be executed. If the input line is blank, the command entry display appears.
CF4	Used to request command prompting assistance.
CF9	Turns the automatic update on or off. When the automatic update function is turned on, you are returned to the display containing the last newest message. The top line of the display indicates whether the display is in automatic update mode (for example, UPDATE: YES indicates that the display is in automatic update mode and UPDATE: NO indicates it is not). When automatic update is turned off, the plus sign (+) in the lower righthand portion of the middle of the display indicates whether there are more messages to be displayed.
CF12	Causes the Display RJE Session (DSPRJESSN) command to be executed, after which the display being shown when the CF12 key was pressed appears again.
CF13	Causes the default reply to the inquiry message identified by you to be sent. You identify the inquiry message by placing the cursor in the corresponding inquiry reply field for that message. The top line of the display (Inq: XX) indicates how many unanswered inquiry messages are outstanding.

STRRJERDR (Start RJE Reader) Command

The Start RJE Reader (STRRJERDR) command causes all or a specific RJEF reader to be started in order to send data to the host system.

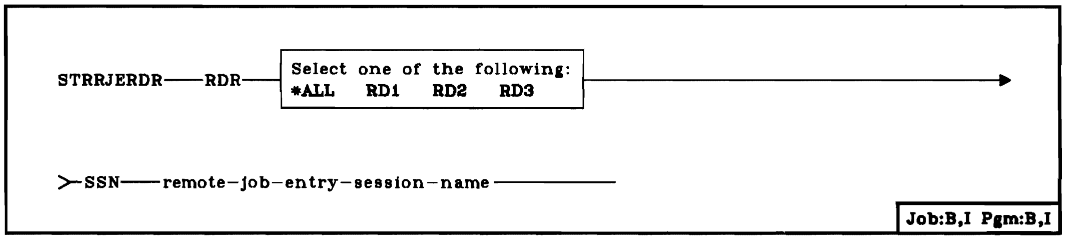
If an RJEF job queue is specified in the session description RJEF reader entry, the job queue is released.

If there is no RJEF job queue specified in the session description RJEF reader entry, the RJEF reader is reserved for interactive jobs.

This command sends a start command to the host to start the associated host reader. If the host reader is already started, some hosts return error messages. RJEF ignores these messages.

Restriction: To use this command, you must have operational rights for the session description.

The Start RJE Reader (STRRJERDR) command is part of the *IBM System/38 Remote Job Entry Facility Program Product,* Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide,* SC21-7914.



RDR Parameter: Identifies the RJEF reader that is associated with this RJEF reader function.

**ALL:* All the RJEF readers associated with the specified RJEF session are to be started.

RD1: RJEF Reader 1 is to be started.

RD2: RJEF Reader 2 is to be started.

RD3: RJEF Reader 3 is to be started.

Note: *AUTO is started (its job queue released, when specified) when a reader is started that allows *AUTO to be used.

SSN Parameter: Specifies the name of the active RJEF session in which the RJEF reader is to be used to send an RJEF reader job to the host system.

Example

```
STRRJERDR RDR(RD1) +
     SSN(RJE)
```

This command starts reader 1 (RD1) in the active RJEF session named RJE. Starting a reader allows it to be used on the Submit RJE Job (SBMRJEJOB) command to send files to the host system.

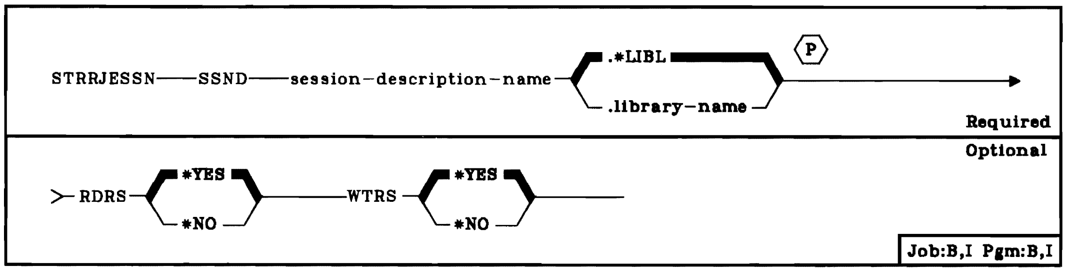
STRRJESSN (Start RJE Session) Command

The Start RJE Session (STRRJESSN) command starts an RJEF session. Each session must have a unique name.

Note: When RJEF readers and writers are started, start commands are sent to the host to start the associated host functions. If the host reader or writer is already started, some hosts return error messages. RJEF ignores these messages.

Restriction: To use this command, you must have operational rights to the session description and read rights to the library in which the session description is stored.

The Start RJE Session (STRRJESSN) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSND Parameter: Specifies the qualified name of the session description for the session that is to be started. If no library qualifier is given, *LIBL is used to find the session description.

RDRS Parameter: Specifies whether the RJEF readers defined in the session description are to be started.

<u>*YES:</u> All the RJEF readers defined in the session description for this RJEF session are initially started (that is, all the reader job queues defined in the session description are released by RJEF and a START command is sent to the host system). Only readers with corresponding communications entries are started. Also, the corresponding communications devices must be varied on or the session is not started.

**NO:* No RJEF readers are started. The STRRJERDR command can be used to start the readers after the session is active.

WTRS Parameter: Specifies whether the RJEF writers defined in the session description are to be started.

*YES: All the RJEF writers defined in the session description for this RJEF session are started. Corresponding communications devices must be varied on or the session is not started.

**NO:* No RJEF writers are started. If a request is received from the host system for an RJEF writer that is not started, an informational message is issued to the RJEF message queue. The data received is held until the RJEF writer is started. The STRRJEWTR command can be used to start the writers after the session is active.

Example

```
STRRJESSN SSND(RJE.USERLIB) +
     RDRS(*YES) +
     WTRS(*YES)
```

This command starts an RJEF session. The session description named RJE in library USERLIB describes the RJEF session environment. In addition to the required RJEF jobs (session control, console input and console output), all readers and writers (printers and punches) defined in the session description are started. When the RJEF startup completes, read files can be sent to the host system and host system files can be received by the writers. Host system commands can be sent by starting the RJE console.

STRRJEWTR (Start RJE Writer) Command

The Start RJE Writer (STRRJEWTR) command causes all or a specific RJEF writer to be started in order to receive writer output from the host system.

This command sends a start command to the host to start the associated host writer. If the host writer is already started, some hosts return error messages. RJEF ignores these messages.

Restrictions: To use this command, you must have operational rights for the session description and read rights for the library in which the session description is stored.

The RJEF user profile (QRJE) must have operational rights for any program specified in this command and read rights for the library in which the specified program is stored.

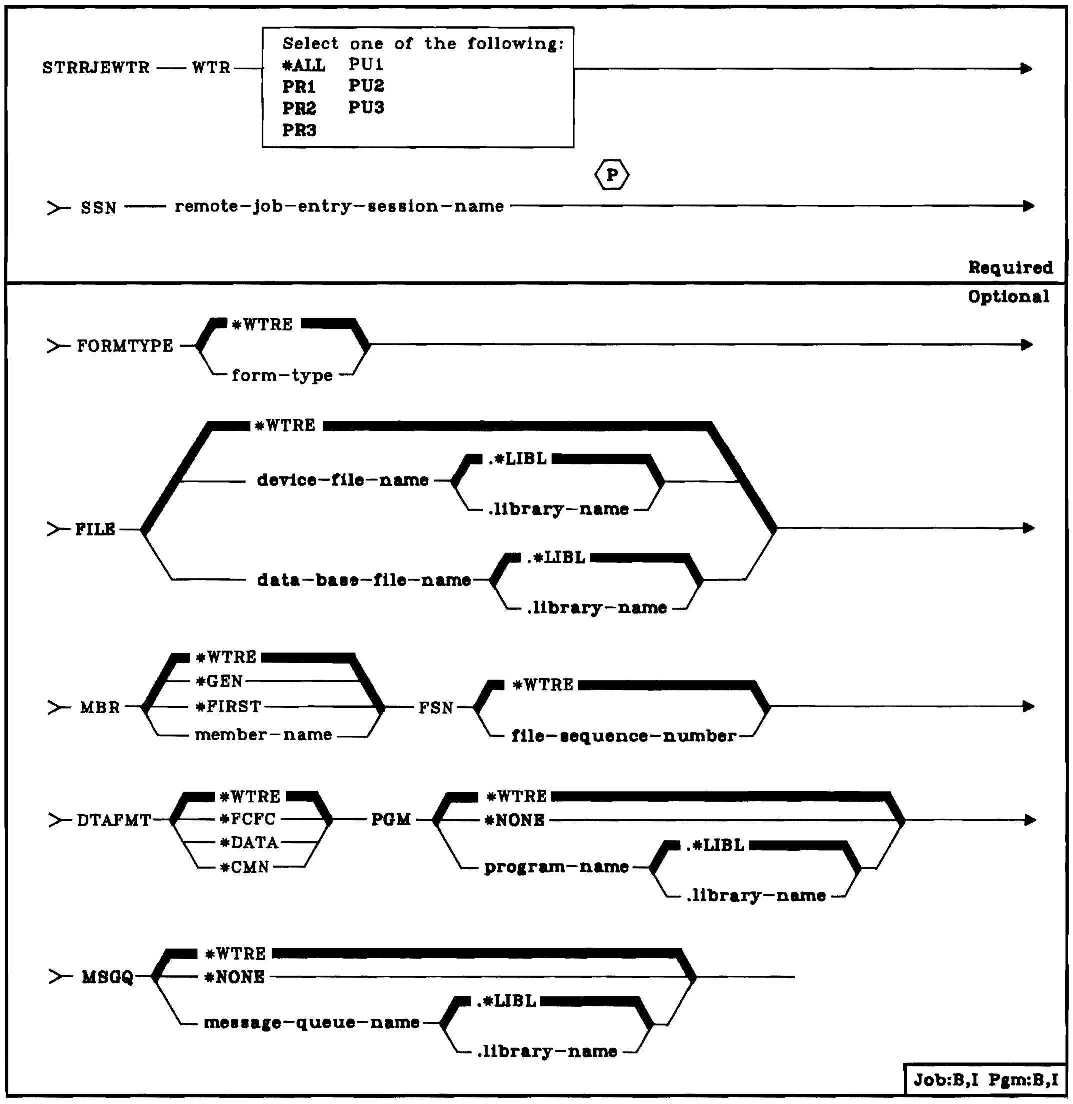
If a data base file name is specified in this command or in the session description RJEF writer entry (that is, if this command's WTR parameter defaults to the value specified in the RJEF writer entry), the RJEF user profile must have add rights to that data base file.

If MBR(*GEN) is specified in this command or in the session description RJEF writer entry, the RJEF user profile must have object management and operational rights to the data base file as well as read rights for the library in which the data base file is stored.

If a device file name (printer device only) is specified in this command or in the session description RJEF writer entry (that is, if this command's FILE parameter defaults to the value specified in the RJEF writer entry), the RJEF user profile must have operational rights for that device file and read rights for the library in which the device file is stored.

If a message queue is specified in this command or in the session description RJEF writer entry (if this command's MSGQ parameter defaults to the value specified in the RJEF writer entry), the RJEF user profile must have operational rights for that message queue and read rights for the library in which the message queue is stored.

The Start RJE Writer (STRRJEWTR) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.


STRRJEWTR — WTR —
Select one of the following:
*ALL PU1
PR1 PU2
PR2 PU3
PR3
P
SSN — remote-job-entry-session-name
Required
Optional
FORMTYPE
*WTRE
form-type
FILE
*WTRE
device-file-name
.*LIBL
.library-name
data-base-file-name
.*LIBL
.library-name
MBR
*WTRE
*GEN
*FIRST
member-name
FSN
*WTRE
file-sequence-number
DTAFMT
*WTRE
*FCFC
*DATA
*CMN
PGM
*WTRE
*NONE
program-name
.*LIBL
.library-name
MSGQ
*WTRE
*NONE
message-queue-name
.*LIBL
.library-name
Job:B,I Pgm:B,I

WTR Parameter: Identifies the RJEF writer that is to be started.

**ALL:* All the RJEF writers associated with the specified RJEF session are to be started.

PR1: RJEF Printer 1 is to be started.

PR2: RJEF Printer 2 is to be started.

PR3: RJEF Printer 3 is to be started.

PU1: RJEF Punch 1 is to be started.

PU2: RJEF Punch 2 is to be started.

PU3: RJEF Punch 3 is to be started.

SSN Parameter: Specifies the name of the active RJEF session in which the RJEF writer is to be started.

FORMTYPE Parameter: Specifies the initial form type to be used for the RJEF writer.

*WTRE: The form type specified in the session description writer entry is to be associated with the RJEF writer.

form-type: Enter the initial form type. Valid values can be one through eight alphameric characters in length. The value should correspond to a valid entry in the forms control table (FCT) specified in the Create Session Description (CRTSSND) command.

FILE Parameter: Specifies the qualified name of the RJEF writer device file (printer only) or the System/38 data base file that is to receive output data from the host system.

*WTRE: The device file specified in the session description RJEF writer entry is to be associated with the RJEF writer.

device-file-name: Enter the qualified name of the program-described device file to receive data for the RJEF writer. If no library qualifier is given, *LIBL is used to find the device file.

data-base-file-name: Enter the qualified name of the System/38 physical data base file to receive the data. If no library qualifier is given, *LIBL is used to find the data base file.

MBR Parameter: Specifies the data base file member to which the output is to be directed (if a data base file was specified in the FILE parameter of this command or in the associated session description writer entry).

**WTRE:* The data base file member is to be generated according to the method specified in the associated session description writer entry.

**GEN:* RJEF creates a member name as follows:

Afffffccc or Bffffffccc

Where:

A	=	file member names beginning with the character A contain print data.
B	=	file member names beginning with the character B contain punch data.
ffffff	=	first six characters of the forms name specified in the FCT or received from the host system. **Note:** Only characters that are valid in a System/38 name are valid in the forms type used to generate data base file member names.
ccc	=	three-digit sequence value controlled by the RJEF session to maintain member uniqueness (refer also to the FSN parameter description of this command).

If a member with this name already exists in the data base file, the three-digit sequence value is incremented by one and another attempt is made to create a member. Incrementing of the sequence value continues until a unique name is generated and a member is created or until all 1000 possibilities have been exhausted without creating a member. If no member is created, the RJEF operator receives a message indicating the failure and a request to retry or cancel of this file.

**FIRST:* The output is to be directed to the first member of the data base file (if a data base file is specified in the FILE parameter of this command or in the associated session description writer entry).

member-name: Enter the name of the data base file member to which output is to be directed (if a data base file is specified in the FILE parameter of this command or the associated session description writer entry). This member is not created by RJEF. If the member does not exist when it is needed, an inquiry message is sent to the RJEF message queue.

FSN Parameter: Specifies the initial three-digit file sequence number to be used when creating data base file member names. This parameter is ignored unless MBR(*GEN) is specified for this command or in the associated session description writer entry.

WTRE: The initial file sequence number to be used is the same as the number specified in the session description writer entry.

file-sequence-number: Enter the initial three-digit file sequence number to be used. Leading zeros are not required for sequence numbers less than 100.

DTAFMT Parameter: Specifies the format of the output data.

WTRE: The output data is to be in the format specified in the session description writer entry.

**FCFC:* The output data is to be in the FCFC data format, with the first character of every record being the ANSI forms control character. Specify *FCFC if the data is to be printed.

The data can be written to a data base file in the FCFC data format and be printed later by using the Copy File (CPYF) command and specifying an FCFC printer file on the TOFILE parameter.

**DATA:* The output data is to be in the normal data format (that is, no FCFC characters are embedded in the data).

**CMN:* The output data is to be in the communications data format (that is, still compressed or truncated). *CMN should be used to decrease communications time. However, before the data can be used, the Format RJE Data (FMTRJEDTA) command must be used to change the data to *FCFC or *DATA. If *CMN is specified, the output file must be a data base file with a length of 256.

PGM Parameter: Specifies the qualified name of a user-supplied program to be used for the RJEF writer.

WTRE: The associated session description writer entry is to be used for the RJEF writer.

**NONE:* No user-supplied program is to be used for the RJEF writer.

program-name: Enter the qualified name of the user-supplied program to be used for the RJEF writer. (If no library qualifier is given, *LIBL is used to find the user-supplied program.)

MSGQ Parameter: Specifies the qualified name for the user message queue on which messages for this RJEF writer are to be recorded.

Note: Messages for RJEF writers are always recorded in the RJEF message queue associated with the named RJEF session. The RJEF message queue name depends upon the name specified in the MSGQ parameter in the Create Session Description (CRTSSND) or Change Session Description (CHGSSND) commands. If inquiry messages are issued by RJEF, they are sent to the user message queue (if specified) where they must receive a response.

If the message queue does not exist when the RJEF writer is ready to use it, a warning message is issued to the RJEF operator message queue indicating the MSGQ parameter of this command has specified a value of *NONE.

**WTRE:* The message queue specified in the session description writer entry is to be used for the RJEF writer.

**NONE:* No user message queue exists on which the messages for the RJEF writer are to be recorded.

message-queue-name: Enter the qualified name of the message queue on which the messages for the RJEF writer are to be recorded. (If no library qualifier is given, *LIBL is used to find the message queue.)

Example

```
STRRJEWTR WTR(*ALL) +
      SSN(RJE) +
      FORMTYPE(COMMON) +
      MSGF(BROWN.USERLIB)
```

This command will start all RJEF writers not already started in the active RJEF session named RJE.

For each writer started, the forms type specified in the session description writer entry is overridden and changes to forms type COMMON. This forms control entry is used by each writer if the host system does not send a forms mount message referencing a valid forms control entry (assuming the session description specified a forms control table name).

The user message queue specified in the session description writer entry, for each writer started, is overridden and changes to user message queue named BROWN in library USERLIB.

STRSBS (Start Subsystem) Command

The Start Subsystem (STRSBS) command starts a subsystem based on the subsystem description specified in the command. Once started, the subsystem is known by the unqualified name of the subsystem description (the subsystem description name without the library name). When the subsystem is started, CPF allocates the necessary and available resources (storage, work stations, and job queue) that are specified in the subsystem description.

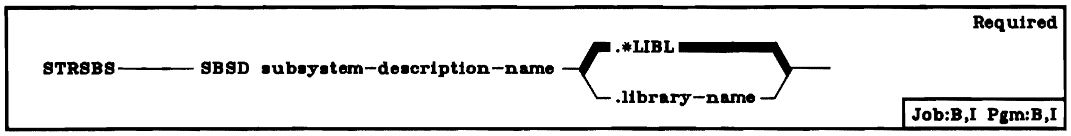
Storage is allocated to the subsystem according to the storage pool definitions specified in the subsystem description, starting with the lower numbered storage pool definitions. If all the pool definitions cannot be allocated, because the maximum number of storage pools on the system is reached or because insufficient storage is available, messages indicating which pools could not be allocated are sent to the system operator. If storage becomes available later, or if the number of active storage pools is reduced, the available resources are automatically allocated to the subsystem to satisfy its unfulfilled requirements. Any routing steps that would normally execute in a storage pool that is not allocated are executed in the shared storage pool (*BASE).

Allocating Work Stations: Work stations are allocated to the subsystem according to the work station entries in the subsystem description. Each work station whose name (or type, if not specified by name) is contained in one of the subsystem description's work station entries, and whose entry specifies AT(*SIGNON), is allocated to this subsystem unless it is currently signed on to another subsystem. The sign-on prompt is displayed on each work station that is allocated. Work stations that are already signed on in another subsystem remain allocated to that subsystem until the subsystem that allowed the sign-on is terminated, or until a user transfers his job to this subsystem. (Messages indicating the names of the work stations that could not be allocated are sent to the system operator.)

If multiple subsystems specify the same work station in their work station entries, each subsystem, as it is started, attempts to allocate that work station. Each successive subsystem does allocate that work station unless a user signs on while the work station is allocated to one of the previously started subsystems. When a signed-on work station is signed off, it still remains allocated to the same subsystem until another subsystem is started that specifies that work station. If, however, a work station is varied offline and several active subsystems specify that work station, the subsystem to which the work station will be allocated when it is varied online is unpredictable.

Allocating Job Queues: If a job queue is specified in the work entries of the subsystem description, the job queue is allocated to the subsystem. If the job queue does not exist or if it is already allocated to an active subsystem, no job queue is allocated to the subsystem and a message is sent to the system operator. If the job queue later becomes available, it is *not* automatically allocated to the subsystem. To allocate the job queue to the subsystem, the subsystem must be terminated, then started again.

Restriction: To start a subsystem, you must have operational rights to use the subsystem description.



SBSD Parameter: Specifies the qualified name of the subsystem description that defines the operational environment (subsystem) being started. (If no library qualifier is given, *LIBL is used to find the subsystem description.) The unqualified name of the subsystem description cannot be the same as the name of a subsystem that is currently active, even though the subsystem descriptions are in different libraries.

Examples

```
STRSBS  SBSD(QBATCH)
```

This command starts the batch subsystem named QBATCH.

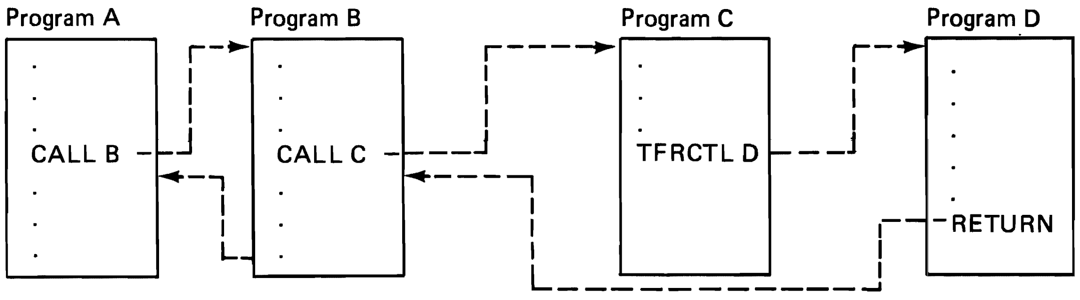
```
STRSBS  SBSD(TELLER.QGPL)
```

This command starts the subsystem that is associated with the TELLER subsystem description in library QGPL. The subsystem name is TELLER.

TFRCTL (Transfer Control) Command

The Transfer Control (TFRCTL) command invokes the program specified on the command, passes control to it, and removes the transferring program from the return stack. When the transferring program is removed from the program invocation stack, control does not return to the program transferring control when the invoked program returns control. Instead, control is returned to the next command after the last call executed before the transferring program is invoked.

For example, in the following drawing, program B returns control to program A. However, because program C transferred control to program D, the RETURN command in program D returns control to program B, which called program C. When program D gets control from program C, program C is removed from the program invocation stack.



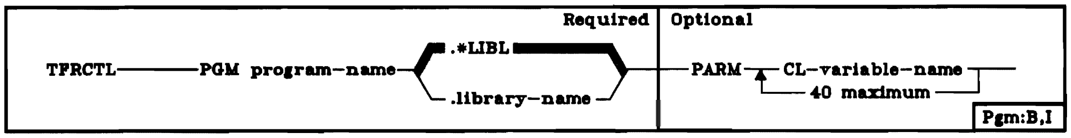
If the transferring program was created with USRPRF(*OWNER), the authority does not transfer. However, if a program created with USRPRF(*OWNER) calls a program that transfers control, the authority *does* transfer because the program that called the transferring program remains in the invocation stack.

Optionally, the transferring program can pass parameters to the program being invoked. The storage space used by the CL variables in the transferring program is freed and made available for use by the program receiving control.

The parameter values must be passed in CL variables. Values cannot be passed as constants, as null parameters (that is, parameters whose values are null, specified by *N), as lists of values, or as CL variables that were not specified as parameters on the PGM command that identified the beginning of the transferring program.

The transferring program can only pass CL variables that were previously passed to it. No CL variable can be passed that exists within the transferring program itself. A maximum of 40 parameters can be passed to the invoked program. The parameters passed must agree in type, length, number, and order with those expected by the receiving program, as specified in the PARM parameter of the PGM command.

Restrictions: This command is valid only within CL programs. The user must have operational rights or one of the data rights for the program to which control is being transferred.



PGM Parameter: Specifies the qualified name of the program that is to receive control from the program transferring control. (If no library qualifier is given, *LIBL is used to find the program.)

PARM Parameter: Specifies the names of one or more CL variables that are to be passed to the program receiving control. The variables passed can only be parameters that were themselves passed to the program currently transferring control. A maximum of 40 variables can be specified. Enter a CL variable name for each of the values to be passed by the program transferring control. The parameter values are received in the receiving program in the order in which they were specified on the TFRCTL command that invokes the program.

The value *N cannot be specified as a value to be passed because a null value cannot be passed to another program.

Example

```
TFRCTL  PROGA &PARM1
```

This command transfers control to the program PROGA and passes the parameter &PARM1 to it. The parameter &PARM1 must previously have been passed to the program issuing this command.

TFRJOB (Transfer Job) Command

The Transfer Job (TFRJOB) command transfers a job to the specified job queue to be executed in the subsystem in which that queue is active. The job that is transferred is the one in which this TFRJOB command is issued. The specified job queue is normally in a different subsystem than the one the job is currently in. If the job being transferred is an interactive job, it is given the highest priority on the job queue. New routing data and request data can be specified for the job when it is transferred. If objects that were allocated to the previous routing step or files that were open in the previous routing step are needed in the new routing step, they must be allocated or opened again.

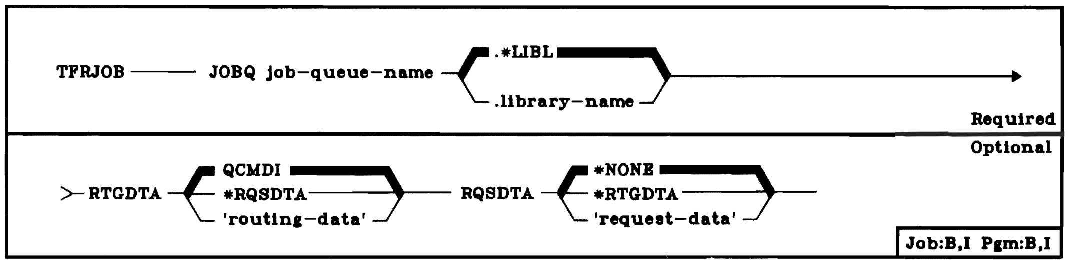
Note: The execution of this command within a batch job causes spooled inline files to be lost, because they cannot be accessed in the new routing step.

Also, if the target subsystem is terminated (by the execution of a TRMSBS, TRMCPF, or PWRDWNSYS command) while an interactive transferring job is on a job queue, the job is canceled as part of subsystem termination.

Because a PWRDWNSYS command inhibits new jobs or routing steps from being initiated by any subsystem, a batch job transferred to a job queue (by the TFRJOB command) will not be completed before the system is powered down. The temporary objects associated with a transferring job (such as the library list, the QTEMP library, and all objects within it) are destroyed during PWRDWNSYS, so that during a re-IMPL the system is unable to restore the job to its previous state. During re-IMPL the system removes the job from the job queue and produces its job log.

Restrictions: The user must have read and add authority for the job queue and for the subsystem that the job queue is allocated to. If the job being transferred is an interactive job, the following restrictions apply:

- The job queue on which the job is to be placed must be associated with an active subsystem.
- The work station associated with the job must have a corresponding work station entry in the subsystem description associated with the new subsystem.
- The work station associated with the job must not have another job associated with it that has been suspended by means of the Sys Req (system request) key. The suspended job must be canceled before the Transfer Job command can be executed.



JOBQ Parameter: Specifies the qualified name of the job queue to which the job is to be transferred. (If no library qualifier is given, *LIBL is used to find the queue.)

RTGDTA Parameter: Specifies the routing data that is to be used to initiate the routing step for the transferred job. The routing data is used to determine the routing entry that identifies the program that is to process the routing step.

QCMDI: This routing data matches a routing entry in the IBM-supplied QINTER subsystem description, which initiates a routing step that is processed by the IBM-supplied control language processor QCL, in the QSYS library.

**RQSDTA:* Up to 80 characters of the request data specified in the RQSDTA parameter of this command are also to be used as the routing data for the routing step.

'routing-data': Enter the character string that is to be used as the routing data for initiating the routing step. A maximum of 80 characters can be entered (enclosed in apostrophes if necessary).

RQSDTA Parameter: Specifies the request data that is to be added to the end of the job's message queue for use by the new routing step.

*NONE: No request data is to be placed in the job's message queue.

**RTGDTA:* The routing data specified in the RTGDTA is also to be placed at the end of the job's message queue.

'request-data': Enter the character string that is to be placed at the end of the job's message queue for use by the new routing step or some subsequent routing step in the job. A maximum of 256 characters can be entered (enclosed in apostrophes if necessary). When a CL command is entered, it must be enclosed in single apostrophes, and where apostrophes would normally be used *within* the command, double apostrophes must be used instead.

Example

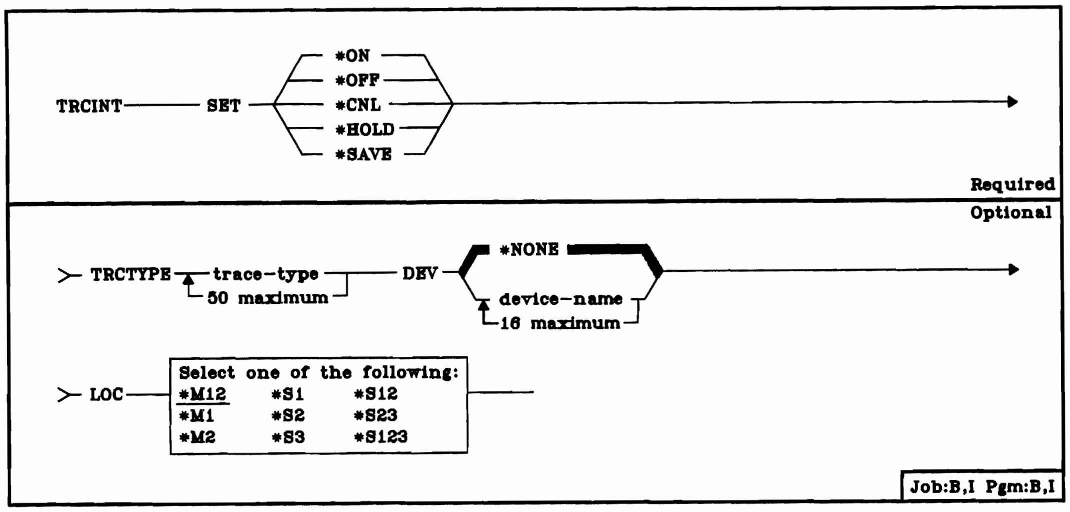
```
TFRJOB  JOBQ(QCTL.QGPL) RTGDTA(APPLICS)
```

This command transfers the job in which the command is entered to the QCTL job queue that is in the QGPL library. The job is to be routed using the routing data APPLICS. If the job is an interactive job, the job queue must be allocated by an active subsystem.

TRCINT (Trace Internal) Command

The Trace Internal (TRCINT) command is used primarily for problem determination. It controls traces of internal events associated with the current job that occur at a level below the machine interface. Specific types of traces can be started and stopped by this command. Those that apply to devices can be limited to a particular device.

While previously started internal traces are being performed, additional internal traces can be started through this command. The output generated by the trace is placed in internal storage used by the internal trace command. The records from the internal storage can be written to a spooled printer file or written to a diskette.



SET Parameter: Specifies whether the generation of internal trace records is to be initiated or terminated.

**ON:* The generation of internal trace records is to be initiated for the trace types specified in the TRCTYPE parameter. If the trace file already contains trace records, the new trace records are added to the file. If the file becomes full, wraparound occurs and the oldest records in the file are overlaid by new ones.

**OFF:* All generation of internal trace records requested through previous TRCINT commands is to be terminated, and the records are to be written to the spooled printer file QPCSMPRT.

**CNL:* All internal traces are to be canceled. All internal trace records are destroyed.

**HOLD:* All internal traces are to be terminated, and the internal trace records that were generated are to be held in internal storage. Held records can be printed later if another TRCINT command is entered that specifies SET(*OFF), or they can be put on diskette if SET(*SAVE) is specified.

**SAVE:* All internal traces are to be terminated, and the internal trace records are to be written to a diskette file named QSYSDKT. The diskettes on which the records are to be written must have been initialized in the basic exchange format (*DATA or *DATA2). The location of the diskette to which the file is to be written is specified in the LOC parameter.

TRCTYPE Parameter: Specifies the type or types of traces that are to be started. If any types are specified, SET(*ON) must also be specified. If any value other than *ON is specified for the SET parameter, TRCTYPE is ignored. Each trace type is identified by a six-digit code; all six digits must be specified. Enter up to 50 of the following types:

Code	**Type of Trace**
030000	System/38 instruction supervisor linkage (SVL) trace
040000	Multiprogramming level (MPL) trace
060000	Task switch trace

Code	**Component Data Trace**
010100	Resource management
010200	Exception management
010300	Program management
010400	Data base management (events for all data base files will be traced)
010500	Process management
010600	Event management
010700	Queue management
010801	Load/dump (save/restore)
010802	Machine services control point
010803	Source/sink device (device support)
010900	Authority management
011000	Context management
011101	Auxiliary storage management
011102	Main storage management–invocations
011103	Main storage management–details
011104	Storage management–ERP
011105	All storage management
011200	Common functions
011300	Machine observation
011400	Independent index management
011500	Space object management
011600	Journal management

Code	Component Call Trace
050100	Common functions
050200	Authority management
050400	Context management
050500	Data base management
050600	Exception management
050700	Event management
050800	Independent index management
050900	Journal management
051000	Process management
051100	Program management
051200	Resource management
051300	Source/sink (device support) management
051400	Space object management
051500	Machine services control point
051600	Machine observation
051700	Recovery management
051800	Queue management

DEV Parameter: Specifies the names of the devices for which the associated internal events are to be traced. This parameter can only be specified if TRCTYPE(010803) is specified; otherwise, it is ignored.

*NONE: No devices are to be traced by this command.

device-name: Enter the names of up to 16 devices for which the internal trace is to be started. The device names must be the same as the names specified in the associated device descriptions.

LOC Parameter: Specifies which diskette locations on the diskette magazine drive are to be used to save the internal trace records. The records are saved (stored) on diskettes that are loaded either in the magazines or in the manual slots. The diskettes must be initialized in the basic exchange format (*DATA or *DATA2). If all of the available space in the specified magazines or slots is used before all the records have been saved, a message is issued to the operator to mount additional magazines or diskettes. This parameter can be specified only if SET(*SAVE) is specified; otherwise, it is ignored.

Enter one of the following values to specify which location is to be used: *M12, *M1, *M2, *S1, *S2, *S3, *S12, *S23, *S123.

If a multiple location is specified (for example, *M12 or *S23) the system begins with the first location specified.

Examples

```
TRCINT  SET(*ON)  TRCTYPE(010100 010400 050500 051200) +
      FILE(*ALL)
```

This command starts component data traces and call traces for resource management and data base. All data base operations associated with all data base files are to be used in generating component data trace records.

```
TRCINT  SET(*ON)  TRCTYPE(010803)  DEV(WS1 WS2 WS3)
```

This command starts component data traces for source/sink management (device support) operations involving the devices WS1, WS2, and WS3.

```
TRCINT  SET(*SAVE)  LOC(*S1)
```

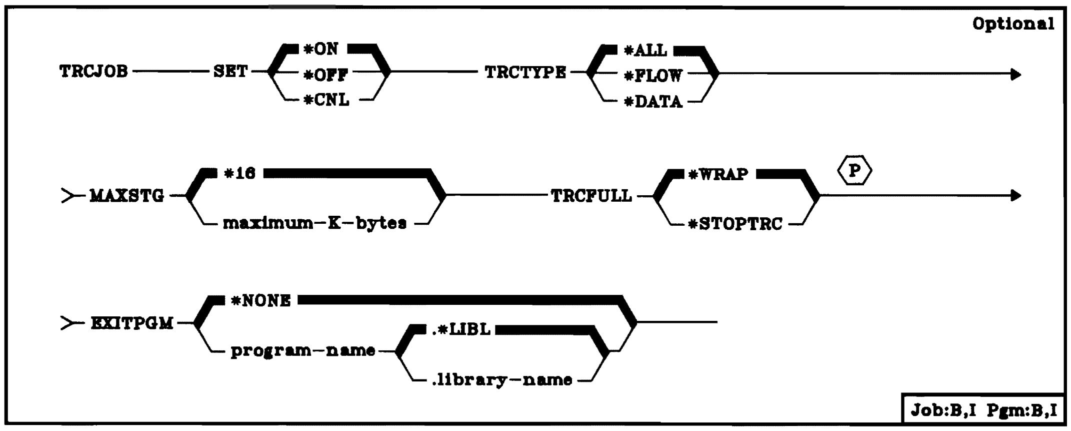
This command terminates all internal traces and saves the internal trace records on the diskette that is mounted at manual slot 1.

TRCJOB (Trace Job) Command

The Trace Job (TRCJOB) command controls traces of program calls and returns that occur in the current job or in the job being serviced as a result of the SRVJOB command directed to that job. The command, which sets a trace on or off, can trace module flow, CPF data acquisition (including CL command traces), or both.

As the trace records are generated, they are stored in an internal trace storage area. When the trace is terminated, the trace records can be written to a spooled printer file, QPSRVTRC. The format of the records is described in the *CPF Programmer's Guide.*

If the SRVJOB command is entered before the TRCJOB command, the job that is traced is the one identified by the SRVJOB command. The trace output from the serviced job is returned to the servicing job after the trace is set off or after the serviced job has terminated.



SET Parameter: Specifies whether the generation of trace records is to be initiated or terminated.

<u>*ON</u>: The generation of trace records is to be initiated.

**OFF:* The generation of trace records is to be terminated, and the trace records are to be written to the spooled printer file.

**CNL:* The generation of trace records is to be terminated, and all existing trace records are to be deleted. No spooled printer file is generated.

TRCTYPE Parameter: Specifies the type of trace data to be stored in a trace file.

<u>*ALL:</u> All the trace data generated is to be stored in trace records. This includes tracing the flow of control and the trace data itself.

**FLOW:* The flow of control is to be traced when programs are called and when they return control.

**DATA:* The data that is provided at predefined trace points within the CPF is to be stored in trace records. This includes trace records for the CL commands that have been executed.

MAXSTG Parameter: Specifies the maximum amount of storage to be used for generated trace records.

<u>16:</u> A maximum of 16 K-bytes of storage is to be used.

maximum-K-bytes: Enter the maximum amount of storage, in K-bytes, to be used to store trace records. (One K equals 1024 bytes.)

TRCFULL Parameter: Specifies whether the trace records are to wrap (replace oldest records with new records) or to set trace off when all of the storage (specified by MAXSTG) has been used.

<u>*WRAP:</u> When the trace file is full, the trace is to wrap to the beginning. The oldest trace records are written over by new ones as they are generated.

**STOPTRC:* Tracing is to be terminated when the trace file is full of trace records.

EXITPGM Parameter: Specifies the name of a user-written program that is to be given control just prior to the generation of each trace record.

<u>*NONE:</u> No user-written program is to be called.

qualified-program-name: Enter the qualified name of the user-written program to be called before each trace record is generated. (If no library qualifier is given, the exit program is found by the library list (*LIBL) that is in effect for the job in which the program is called.) This program can examine and alter the 78-character trace record passed to it as a parameter. (The format of this record is described in the *CPF Programmer's Guide.*) If it replaces the first 2 characters in the trace record with blanks or binary zeros, the entire trace record is suppressed. This program can also call any of the service dump commands and must return control when it completes its task. If the user-written program (which *cannot* be a CL program) executes other CPF commands, the output from those commands is associated with the job in which they are executed. The dump and trace commands associate their output data either with the job that enters the dump and trace commands or with the job being serviced as the result of the SRVJOB command.

Example

```
TRCJOB  TRCTYPE(*FLOW)  MAXSTG(40)
```

This command causes the module flow of the current job to be traced. Trace records are generated for each program call and return that occurs in the job. The trace file contains 40K of storage and wraps if that amount of storage is filled with trace records.

TRMCPF (Terminate Control Program Facility) Command

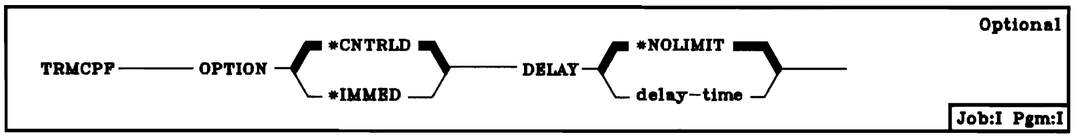
The Terminate Control Program Facility (TRMCPF) command prepares the system for CPF termination and leaves the system in a condition in which only the console is active in the controlling subsystem. (This is normally done so that hardware servicing diagnostics can be performed.)

If two jobs are active in the controlling subsystem at the console (through the use of the system request key), neither of the jobs is forced to terminate. The TRMCPF command cannot complete its execution until you terminate one of the jobs (either by signing off in one job or by canceling one job from the other).

All active subsystems are notified that CPF termination is in process. No new jobs or routing steps can be accepted by the subsystems. The TRMCPF command also specifies what happens to all active work.

Interactive jobs that have been transferred to a job queue by the TFRJOB command will be canceled as part of subsystem termination. If an IMPL occurs while either a batch or interactive job is on a job queue (because of the TFRJOB command), that job will be removed from the job queue during IMPL and its job log will be produced.

Restriction: This command can be entered only in an interactive job in the controlling subsystem. To use this command, you must have job control (*JOBCTL) authority.



OPTION Parameter: Specifies whether all active jobs are to be canceled in a controlled manner (which lets the application programs perform termination processing) or immediately. In either case, the system does perform certain job cleanup functions.

*CNTRLD: The jobs are to be terminated in a controlled manner. This allows the executing programs to perform cleanup (termination processing).

**IMMED:* The jobs are to be terminated immediately, meaning the executing programs are not allowed to perform any cleanup. (This option might cause undesirable results if data has been partially updated and, therefore, should be used only after a controlled cancel has been attempted unsuccessfully.)

DELAY Parameter: Specifies the amount of time, in seconds, that the controlled CPF termination is to be allowed. If this amount of time is exceeded and the termination process is not complete, any jobs still being processed are canceled immediately, except for those executing long-running System/38 instructions.

*NOLIMIT: The amount of time in which to complete a controlled termination process is not limited.

delay-time: Enter a value, in seconds, for the time in which the termination process is to be completed.

Examples

```
TRMCPF
```

This command terminates the Control Program Facility after all active jobs in the system are allowed to perform their own termination processes. The amount of time the termination processes can take is not limited.

```
TRMCPF  OPTION(*IMMED)
```

This command terminates the CPF after all active jobs are immediately canceled.

Additional Considerations

The system may be brought into a *restricted* state by doing one of the following:

- Terminating all subsystems
- Terminating CPF
- Initiating an IMPL under the QCE user profile (Concurrent Service Monitor)
- Failing to allocate sufficient storage during IMPL

During the restricted state, the controlling subsystem is the only active subsystem, with only one active work station (which can be the system console). Only one active job is allowed at a time in the system.

If you sign off the active work station, the sign-on prompt will appear; if you sign on, the controlling subsystem remains in the restricted state.

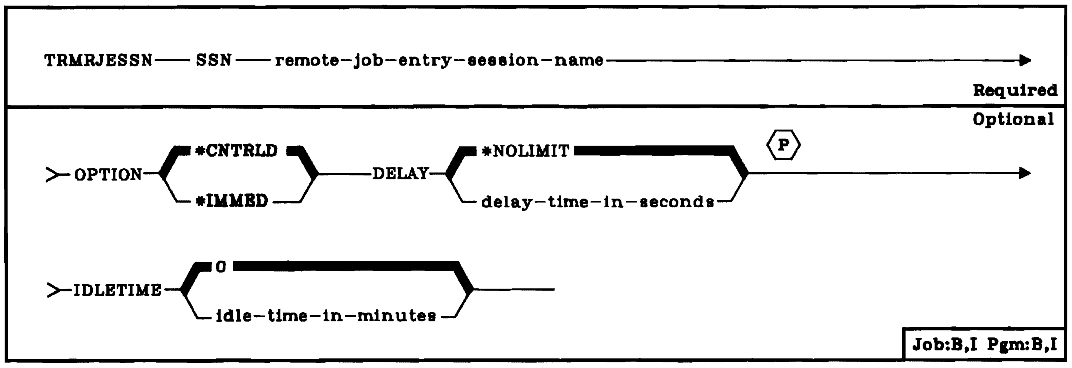
To exit from the restricted state, you must start a subsystem. If the subsystem being started is not the controlling subsystem; the controlling subsystem remains in the restricted state.

TRMRJESSN (Terminate RJE Session) Command

The Terminate RJE Session (TRMRJESSN) command terminates the specified RJEF session.

Restriction: To use this command, you must have operational rights to the session description and read rights to the library in which the session description is stored.

The Terminate RJE Session (TRMRJESSN) command is part of the *IBM System/38 Remote Job Entry Facility Program Product*, Program 5714-RC1. For more information on the Remote Job Entry Facility, refer to the *IBM System/38 Remote Job Entry Facility Programmer's Guide*, SC21-7914.



SSN Parameter: Specifies the name of the RJEF session that is to be terminated.

OPTION Parameter: Specifies whether an RJEF session is to be canceled in a controlled manner (which lets the RJEF programs perform termination processing) or canceled immediately.

**CNTRLD:* The RJEF session is to be terminated in a controlled manner. Controlled termination is initiated with a LOGOFF or SIGNOFF command sent to the host system. When an end-of-file is received from the host system for each active printer and punch and an end-of-file is sent to the host system for each active reader job, RJEF terminates communications. The transmitting and receiving of all printer, punch, readers, and message data continues until the output and input streams are completed. All RJEF reader job queues for the session are held.

**IMMED:* The RJEF session is to be terminated immediately. No LOGOFF or SIGNOFF command is sent to the host system. All RJEF reader job queues for the session are held.

DELAY Parameter: Specifies the amount of time, in seconds, that the controlled RJEF session termination is allowed. If this amount of time is exceeded and the termination process is not complete, the session still being processed is canceled immediately. This parameter is ignored if OPTION(*IMMED) is specified for this command.

*NOLIMIT: The amount of time in which to complete a controlled termination process is not limited.

delay-time-in-seconds: Enter a value, in seconds, for the time in which the RJEF session must begin termination processing. When the specified time elapses, the termination request is upgraded to an immediate termination request. The delay time countdown begins when the command is executed. Valid values are 1 through 99999.

IDLETIME Parameter: Specifies the minimum number of minutes that the RJEF session should remain idle before the transmission of a LOGOFF or SIGNOFF command. This parameter is ignored unless OPTION(*IMMED) is specified for this command.

0: The idle time value is zero. The LOGOFF or SIGNOFF command is transmitted immediately and RJEF holds all the RJEF reader job queues specified for this RJEF session.

idle-time-in-minutes: Enter a value, in minutes, that the RJEF session should remain idle after all files are closed. The idle time countdown begins following the end-of-file of the last file transmitted or received and is reset each time data becomes available for transmitting or receiving. If there are any input streams that have started but have not ended (that is, have not received end-of-file), except for the console input streams, the idle time countdown does not begin. Valid values are 1 through 99.

Example

```
TRMRJESSN SSN(RJE) +
     OPTION(*CNTRLD) +
     IDLETIME(3)
```

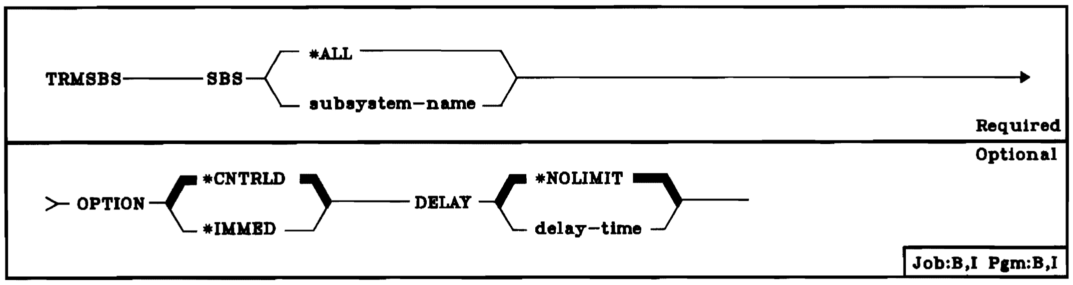
This command terminates an active RJEF session. The termination is to be a controlled termination. Files currently being processed are allowed to complete. IDLETIME specifies that after 3 minutes of no activity (no files being sent or received) LOGOFF or SIGNOFF (host dependent) is sent to the host system and the RJEF session terminates.

TRMSBS (Terminate Subsystem) Command

The Terminate Subsystem (TRMSBS) command terminates the specified subsystem (or all active subsystems) and specifies what happens to active work being processed by that subsystem or all subsystems. No new jobs or routing steps are initiated in the subsystem or subsystems.

Interactive jobs that have been transferred to a job queue by the TFRJOB command will be canceled as part of subsystem termination. If an IMPL occurs while either a batch or interactive job is on a job queue (because of the TFRJOB command), that job will be removed from the job queue during IMPL and its job log will be produced.

Restriction: If the controlling subsystem is being terminated, because either its name or *ALL is specified by the SBS parameter, this command can be entered only in an interactive job that is in the controlling subsystem and only from a work station (associated with the interactive job) whose work station entry in the controlling subsystem description specifies AT(*SIGNON). To use this command, you must have job control (*JOBCTL) authority.



SBS Parameter: Specifies the name of the subsystem that is to be terminated, or it specifies that all active subsystems are to be terminated.

**ALL:* All the subsystems that are currently active are to be terminated. All jobs are terminated except the job in which this command is entered. When this value is specified, the QSYSOPR message queue should be in break delivery mode.

subsystem-name: Enter the simple name of the subsystem that is to be terminated. If the subsystem specified is the controlling subsystem, the interactive job from which the command was issued remains active. Also, if the subsystem specified is the controlling subsystem and the job that issues this command is one of two jobs that are active at the work station (through the use of the system request key), neither of the jobs is forced to terminate. The TRMSBS command cannot complete its execution until you terminate one of the jobs (either by signing off in one job or by canceling one job from the other).

OPTION Parameter: Specifies whether jobs in the subsystem are to be canceled in a controlled manner (which lets the application programs perform termination processing) or immediately. In either case, the system does perform certain job cleanup functions.

*CNTRLD: The jobs are to be terminated in a controlled manner. This allows the executing programs to perform cleanup (termination processing).

**IMMED:* The jobs are to be terminated immediately, meaning the executing programs are not allowed to perform any cleanup. (This option might cause undesirable results if data has been partially updated and, therefore, should be used only after a controlled cancel has been attempted unsuccessfully.)

DELAY Parameter: Specifies the amount of time, in seconds, that the controlled subsystem termination is allowed. If this amount of time is exceeded and the termination process is not complete, any jobs still being processed in the subsystem are canceled immediately.

*NOLIMIT: The amount of time in which to complete a controlled termination process is not limited.

delay-time: Enter a value, in seconds, for the time in which the termination process is to be completed.

Example

```
TRMSBS  SBS(QBATCH)  OPTION(*CNTRLD)  DELAY(60)
```

This command cancels all active jobs in the QBATCH subsystem and terminates the subsystem. The active jobs are allowed 60 seconds to perform application-provided termination processing.

Additional Considerations

The controlling subsystem may be brought into a *restricted* state by doing one of the following:

- Terminating the controlling subsystem
- Terminating all subsystems
- Terminating CPF
- Initiating an IMPL under the QCE user profile (Concurrent Service Monitor)
- Failing to allocate sufficient storage during IMPL

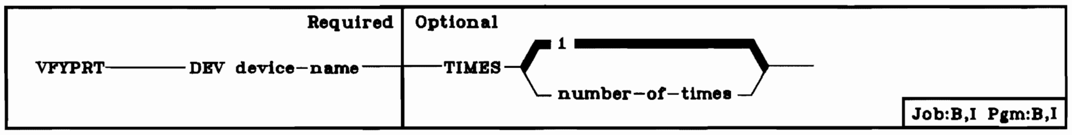
During the restricted state, the controlling subsystem is the only active subsystem, with only one active work station (which can be the system console). Only one active job is allowed at a time in the controlling subsystem, but there may be other jobs running in other nonrestricted subsystems.

If you sign off the active work station, the sign-on prompt will appear; if you sign on, the controlling subsystem remains in the restricted state.

To exit from the restricted state, you must restart the controlling subsystem. If the subsystem being started is not the controlling subsystem; the controlling subsystem remains in the restricted state.

VFYPRT (Verify Printer) Command

The Verify Printer (VFYPRT) command is used to exercise the 5224/5225/5256 Printer by causing it to print a test pattern a specified number of times.



DEV Parameter: Specifies the name of the 5224/5225/5256 Printer that is to be exercised. The device name must be the same as that specified in the device description for the printer.

TIMES Parameter: Specifies the number of times that the specified printer is to print the test pattern.

1: The test pattern is to be printed one time.

number-of-times: Enter a value for the number of times that the test pattern is to be printed.

Example

```
VFYPRT  DEV(PRTR3) TIMES(15)
```

This command causes the printer named PRTR3 to print a test pattern 15 times.

VRYCTLU (Vary Control Unit) Command

The Vary Control Unit (VRYCTLU) command is used to bring (vary) one or more control units online with the system, or offline. This command applies to all control units on the system: the 3411 tape control unit, the 5251 work station control units, the work station controllers (WSC), the physical unit (type 2), and BSC control units.

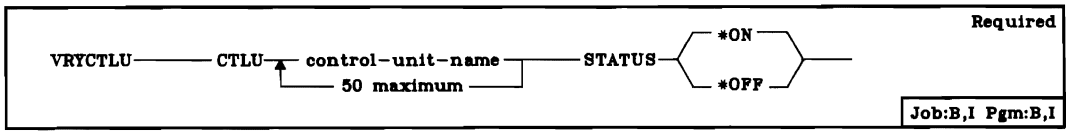
A control unit cannot be varied online:

- If its power is turned off. (Its power switch must be turned on first, or the PWRCTLU command must be used, if applicable.)
- If the line to which it is attached is varied offline, in the case of leased (nonswitched) lines. (The VRYLIN command must be used to put the line online.)
- Until a dialed connection has been completed, in the case of switched lines.

A control unit cannot be varied offline:

- If it is being used or allocated for use.
- If any of the devices attached to the control unit are still online. (The VRYDEV command must be used to vary all the devices offline.)
- If a tape device has not completed rewind or unload.

Note that, although the VRYCTLU command can be executed at any time, the control unit is not actually online or offline (changed status) until the above conditions are satisfied. This command is normally used to vary a control unit offline when it needs servicing or to vary the control unit online when servicing is complete.



CTLU Parameter: Specifies the names of one or more control units that are to be varied online or offline. The name of each control unit must be the same as that specified in its control unit description.

STATUS Parameter: Specifies whether one or more control units are to be varied online or offline.

**ON:* Each control unit specified is to be varied online.

**OFF:* Each control unit specified is to be varied offline.

Examples

```
VRYCTLU  CTLU(CONTROLU2) STATUS(*OFF)
```

This command places the control unit named CONTROLU2 offline. All the devices attached to CONTROLU2 must already be offline before this command is entered.

```
VRYCTLU  CTLU(CONTROLU3) STATUS(*ON)
```

This command places the control unit named CONTROLU3 online. The communication line to which CONTROLU3 is attached must already be online before this command is entered.

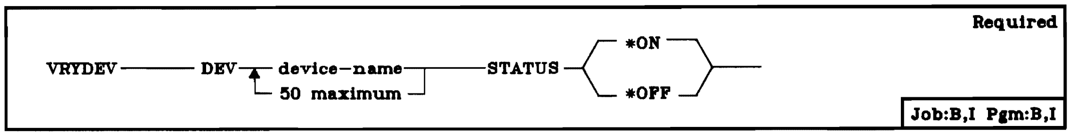
VRYDEV (Vary Device) Command

The Vary Device (VRYDEV) command is used to vary one or more devices online with the system, or offline. This command applies to all devices that can be on the system, which includes the MFCU, diskette magazine drive, console, primary logical unit (PLU1), BSC devices, and all of the printers, work stations, and tape units.

A device cannot be varied offline if it is being used (or allocated for use). A device cannot be varied online:

- If its power is turned off. (The PWRDEV command must be used to turn the device on.)
- If the control unit to which it is attached is varied offline. (The VRYCTLU command must be used to vary the control unit online.)
- If the device is a tape drive that has not completed rewind or unload.

Note that, although the VRYDEV command can be executed before a switched connection is made, the device is not actually online until the connection is made. This command is normally used by the system operator to vary a device offline when it needs servicing or to vary the device online when servicing is completed.



DEV Parameter: Specifies the names of one or more devices that are to be varied online or offline. The name of each device must be the same as that specified in its device description.

STATUS Parameter: Specifies whether one or more devices are to be varied online or offline.

**ON:* Each device specified is to be varied online.

**OFF:* Each device specified is to be varied offline.

Example

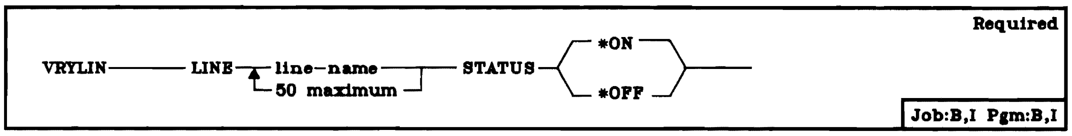
```
VRYDEV  DEV(PRINTMASK)  STATUS(*ON)
```

This command places the device named PRINTMASK online. The power for this device must be turned on before this command is entered.

VRYLIN (Vary Line) Command

The Vary Line (VRYLIN) command is used to vary one or more communication lines online with the system, or offline. A line cannot be varied offline until the devices and control units attached to the line are varied offline first. Similarly, a leased (nonswitched) line must be varied online before its control units and devices can be varied online.

This command is normally used by the system operator to vary a line offline when it needs servicing or to vary the line online when servicing is completed.



LINE Parameter: Specifies the names of one or more communication lines to be varied online or offline. The name of each line must be the same as that specified in its line description.

STATUS Parameter: Specifies whether one or more communication lines are to be varied online or offline.

**ON:* Each communication line specified is to be varied online.

**OFF:* Each communication line specified is to be varied offline.

Examples

```
VRYLIN  LINE(LINEA)  STATUS(*ON)
```

This command places the communication line named LINEA online.

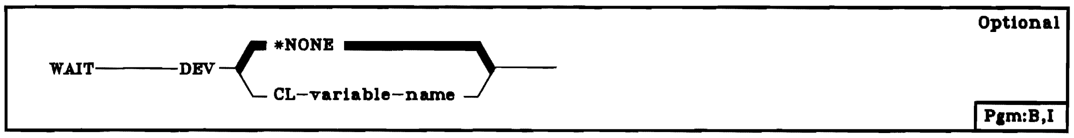
```
VRYLIN  LINE(LINEB)  STATUS(*OFF)
```

This command places the communication line named LINEB offline. All the control units and devices attached to LINEB must be offline before this command is entered.

WAIT (Wait) Command

The Wait (WAIT) command accepts input from any display device from which user data was invited by one or more previous RCVF or SNDRCVF commands that did not want to wait for the input data to be received. Those commands had *NO specified in the WAIT parameter and specified a particular device file to receive and transfer the data to the CL program. Only one input request *per device* can be outstanding at any given time. If there are multiple outstanding input requests, the first device to respond to the specified device file has its user data sent to the CL program. When the data is received, the wait operation ends and the next command in the program is executed.

Restriction: This command is valid only within a CL program.



DEV Parameter: Specifies the name of the CL variable that is to receive the name of the display device that responds with user data for the CL program.

<u>*NONE:</u> No CL variable name is to be specified; the name of the responding device is not needed.

CL-variable-name: Enter the name of the CL variable that is to receive the name of the responding device. The variable must be a character variable with a maximum length of 10. A device name cannot be specified in this parameter.

Examples

```
DCLF  FILE(MSCREEN)
   •
   •
   •
RCVF  DEV(DEV1) WAIT(*NO)
   •
   •
   •
RCVF  DEV(DEV2) WAIT(*NO)
   •
   •
   •
WAIT  DEV(&DEVNAM)
```

In this example, the device file MSCREEN is to be used to receive user data; the RCVF commands specify that the program is not to wait for the data. The WAIT command causes the program to wait for the display device file MSCREEN to pass input data to it from one of its devices. The name of the responding display device is to be placed into the CL variable &DEVNAM. The received data is to be placed in the program variables associated with the record format of the declared file.

```
DCLF  FILE(DF1)
   •
   •
   •
RCVF  DEV(DEV1) WAIT(*NO)
   •
   •
   •
RCVF  DEV(DEV2) WAIT(*NO)
   •
   •
   •
WAIT  DEV(&DN)
```

In this example, the RCVF commands specify that the program is not to wait for user data. When the WAIT command is issued, the name of the responding device is to be placed in the CL variable &DN. The data received from the device file DF1 is to be placed in the program variable of the associated record format in the file DF1.

Chapter 5. Command Definition Statements

The command definition facility lets you define a command that will invoke a program to perform some function. The defined command can include:

- Keywords to pass parameters to programs
- Default values for omitted parameters
- Parameter validity checking so that the program performing the function is assured of correct input
- Prompt text for prompting interactive users

CREATING USER-DEFINED COMMANDS

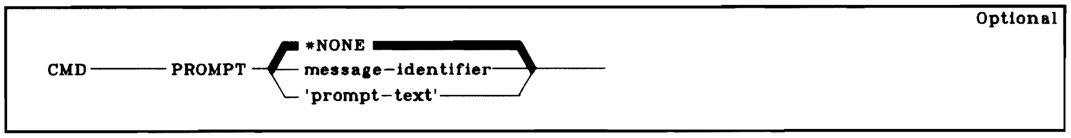
You can define a command by entering command definition statements into a source file and executing a Create Command (CRTCMD) command using the source file as input. One Command (CMD) statement must be somewhere in the source file. A Parameter (PARM) statement must be provided for each parameter that appears on the command being created. If any special keyword relationships need checking, the Dependent (DEP) statement is used to define the relationships. The DEP statement can only reference parameters that have been previously defined. These statements can appear in any order. See the *CPF Programmer's Guide* for a complete description of how to use these statements to define a command.

Only one command can be defined in each source member in the source file. The CRTCMD command is executed to create the command definition object from the command definition statements in one source file member. Other users can then be authorized to use the new command by the Grant Object Authority (GRTOBJAUT) command.

COMMAND DEFINITION STATEMENT DESCRIPTIONS

CMD (Command) Statement

The Command (CMD) statement specifies the prompt text for the command being created. The prompt text is displayed at a work station when a user requests prompting while entering the command that is being defined. The CMD statement can be anywhere in the source file referenced by the CRTCMD (Create Command) command; one and only one CMD statement must be used in the source file even if no prompt text is to be specified for the created command.



PROMPT Parameter: Specifies what prompt text, if any, is to be included in the heading (title) of the prompt display for the command being defined. The prompt text further describes the name of the command. For example, in the CRTLIB prompt heading, *Create Library (CRTLIB) Prompt*, the words *Create Library* would be specified as the prompt text in this PROMPT parameter.

Note: Prompt text for each of the *parameters* in this command can be specified in the PROMPT parameters of the PARM, ELEM, and QUAL command definition statements. They specify the prompt text for the parameters, just as the PROMPT parameter in the CMD statement specifies the prompt text for the command (in the heading).

*NONE: No prompt text is to be included in the displayed heading of the prompt when the command is being prompted.

message-identifier: Enter the message identifier that specifies the message, containing no more than 30 characters, for the prompt text to be displayed when the command is being prompted. If a message having the specified identifier cannot be found in the message file specified in the PMTFILE parameter of the Create Command (CRTCMD) command, the message identifier itself is used as the prompt text.

'prompt-text': Enter the prompt text that is to be displayed during the command prompting. It must be a character string of no more than 30 characters enclosed in apostrophes.

Example

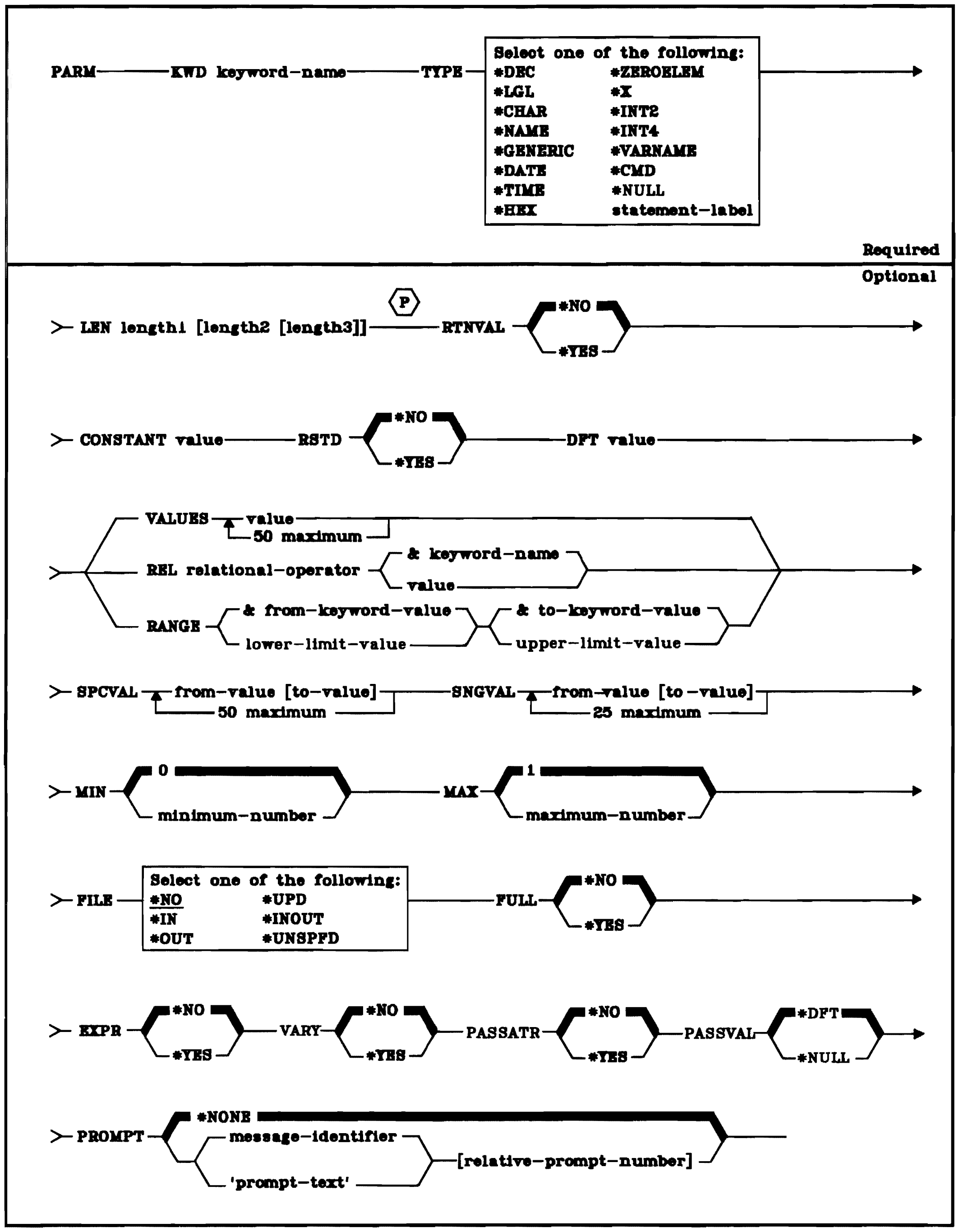
```
CMD  PROMPT(UCD0001)
```

This statement describes a command that will be prompted with additional text in the display heading; the prompt text comes from the message identified by UCD0001.

PARM (Parameter) Statement

The Parameter (PARM) statement defines a parameter of a command being created. A parameter is the means by which a value is passed to the command processing program (CPP). One PARM statement must be used for each parameter that appears in the command being defined. The order in which the PARM statements are entered into the source file determines the order in which the parameters must be specified when the command is entered in positional form and the order in which they are passed to the validity checker and the CPP. A maximum of 75 parameters can be defined for one command. Note, however, that commands having a large number of parameters take longer to execute, regardless of how many parameters are actually coded.

Note: The PARM statement contains certain parameters and predefined values that can be used only when IBM-supplied CPPs are to be invoked by the command being defined. Because of limitations in some high-level languages, these values may not be useful in the definition statements of user-defined commands. These parameters and values are identified by the phrase (*For IBM-supplied commands*) that immediately follows the parameter keyword (if the entire parameter is for IBM-supplied commands only) or the predefined value to which it applies.


PARM
KWD keyword-name
TYPE
Select one of the following:
*DEC
*ZEROELEM
*LGL
*X
*CHAR
*INT2
*NAME
*INT4
*GENERIC
*VARNAME
*DATE
*CMD
*TIME
*NULL
*HEX
statement-label
Required
Optional
LEN length1 [length2 [length3]]
P
RTNVAL
*NO
*YES
CONSTANT value
RSTD
*NO
*YES
DFT value
VALUES
value
50 maximum
REL relational-operator
& keyword-name
value
RANGE
& from-keyword-value
lower-limit-value
& to-keyword-value
upper-limit-value
SPCVAL
from-value [to-value]
50 maximum
SNGVAL
from-value [to-value]
25 maximum
MIN
0
minimum-number
MAX
1
maximum-number
FILE
Select one of the following:
*NO
*UPD
*IN
*INOUT
*OUT
*UNSPFD
FULL
*NO
*YES
EXPR
*NO
*YES
VARY
*NO
*YES
PASSATR
*NO
*YES
PASSVAL
*DFT
*NULL
PROMPT
*NONE
message-identifier
'prompt-text'
[relative-prompt-number]

KWD Parameter: Specifies the keyword name of the parameter being created. The keyword identifies the parameter in the command and is used when the parameter is entered in keyword form. Enter the keyword name using no more than 10 alphameric characters, the first being alphabetic.

TYPE Parameter: Specifies the type of the value that can be specified for the parameter named in KWD. The value can be an integer, a decimal, hexadecimal, or logical value, or a quoted or unquoted character string that can be a name, date, or time; the value can also be a command. Enter one of the following options to specify the parameter type:

******DEC:* The parameter value is a packed decimal number.

******LGL:* The parameter value is a logical value of one ('1') or zero ('0').

******CHAR:* The parameter value is a character string that can (optionally) be enclosed in apostrophes. If the character string contains any special characters (not including an *), it *must* be enclosed in apostrophes. The maximum length of the character string is 2000 characters.

******NAME:* The parameter value is a character string, representing a name. The maximum length of the name is 256 characters; the first character must be alphabetic, and the remaining characters must be alphameric or an underscore. If a special value is to be used, it should be specified in the SPCVAL parameter.

******GENERIC:* The parameter value is a character string, representing a generic name. A generic name contains a maximum of 255 characters followed by an asterisk (*) or 256 characters without an asterisk; the name identifies a group of objects whose names all begin with the characters preceding the *. (If an * is not included, the system assumes that the generic name is a complete object name.)

******DATE:* The parameter value is a character string that represents a date. When it is passed to the CPP, it is always passed in the format *Cyymmdd*, where C = century, yy = year, mm = month, and dd = day. The century digit is set to 0 (zero) if yy is equal to or greater than 40 (1940 through 1999); it is set to 1 (one) if yy is less than 40 (2000 through 2039). When a date value is specified in this PARM statement, it must be specified in the format *mmddyy*. When a user enters a date in the command at execution time, it must be entered in the format specified by the system value QDATFMT. The system value QDATSEP specifies the optional separator character that can be used when the date is entered. If the separator character is used, the date must be enclosed in apostrophes.

******TIME:* The parameter value is a character string that represents a time. It is entered in the format *hhmmss* or, if enclosed in apostrophes, in the format *'hh:mm:ss'*, where hh = hours, mm = minutes, and ss = seconds. It is passed to the CPP in a 6-byte character string as hhmmss.

**HEX:* The parameter value is hexadecimal in form. The specified characters must be 0 through F. They will be converted to hexadecimal (EBCDIC) characters (2 hex digits per byte), right adjusted, and padded with zeros. If the value is enclosed in apostrophes, an even number of digits is required. If the value is not enclosed in apostrophes, an even number of digits is *not* required.

**ZEROELEM:* The parameter is always to be considered as a list of zero elements, for which no value can be specified in the command. It is used to prevent a value from being entered for a parameter that is a list even though the CPP expects one. For example, if two commands use the same CPP, one command could pass a list for a parameter and the other command may not have *any* values to pass. The second command would be coded with TYPE(*ZEROELEM).

**X:* (For IBM-supplied commands) The parameter value is a character string, variable name, or numeric value. The value is passed as a numeric value if it contains only digits, a + or - sign and/or a decimal point; otherwise, it is passed as a character string.

**INT2:* The parameter value is an integer that is to be passed as a 2-byte signed binary number. CL programs do not support binary values in variables.

**INT4:* The parameter value is an integer that is to be passed as a 4-byte signed binary number. CL programs do not support binary values in variables.

**VARNAME:* (For IBM-supplied commands) The parameter value is a CL variable name that is to be passed as a character string. The name can contain a maximum of 11 characters, including the ampersand (&).

**CMD:* (For IBM-supplied commands) The parameter value is a command. For example, the IF command has a parameter called THEN whose value must be another command. The command will be validity checked by the command analyzer.

statement-label: The parameter accepts a qualified name or a mixed list of values. The statement label specified here by the TYPE parameter is the statement label that identifies the first of a series of QUAL or ELEM statements that further describe the qualified name or the mixed list being defined by this PARM statement.

**NULL:* The parameter value is to be a null pointer, which can be used as a constant placeholder. A DEP statement or the REL and RANGE keywords of other PARM statements may not reference the value of a parameter defined with TYPE(*NULL).

LEN Parameter: Specifies the length of the parameter value that is to be passed to the CPP. If TYPE was specified as *INT2, *INT4, *DATE, *TIME, *CMD, *ZEROELEM, *NULL, or statement-label, LEN is not allowed. With other TYPE specifications, this parameter has the following applications:

- If TYPE(*DEC) was specified, the decimal length is specified in the form (length1 length2), where length1 specifies the total number of digits in the value (including the decimal portion), and length2 specifies the number of allowable decimal digits to the right of the decimal point. (The value for length2 is optional. Zero is assumed if it is not entered.)

- If TYPE(*CHAR), TYPE(*NAME), or TYPE(*VARNAME) was specified, only length1 is specified. It identifies the number of characters passed.

- If TYPE(*HEX) was specified, only length1 is specified. This length specifies the number of characters passed after the hexadecimal digits have been converted to character digits. Because two hexadecimal digits are converted to one decimal digit, the number of hexadecimal digits converted is twice the value of this length.

- If TYPE(*X) was specified, the LEN parameter is used as follows:
  - For character data, length1 specifies the minimum length to be passed. If a longer value is entered, the entire value is passed.
  - For decimal data, length2 and length3 specify the length and decimal positions for a constant value. If a variable is entered, it is passed according to the variable attributes.
  - For a logical value, length1 specifies the length of the value, which is always 1.

The default length that is assumed by the system and the maximum length for each type of parameter value are shown in the following table:

Data Type	Default Length	Maximum Length
*DEC	(15 5)	(24 9)
*LGL	1	1
*CHAR	32	2000 (see note)
*NAME	10	256 (see note)
*GENERIC	10	256 (see note)
*HEX	1	256 (see note)
*X	(1 15 5)	(256 24 9)
*VARNAME	11	11

Note: Whereas the maximum shown here is the maximum for values used in the command when it is executed, the maximum length of character constants specified in the command definition is limited to 32 characters. This restriction applies to the following parameters: CONSTANT, DFT, VALUES, REL, RANGE, SPCVAL, and SNGVAL.

LEN

For those data types for which length cannot be coded, the following are the maximum lengths and the lengths passed.

Data Type	Maximum Length	Length Passed
*DATE (Note 1)	8	7
*TIME (Note 2)	8	6
*ZEROELEM	0	0
*INT2 (Note 3)	6	2
*INT4 (Note 4)	11	4
*CMD	Any	Length needed
statement-label (Note 5)	–	–

Notes:

1. If a date is specified, the value is passed as 7 characters.
2. If a time is specified, the value is passed as 6 characters.
3. The value must meet the following condition: $-2^{15} \leq \text{value} \leq 2^{15}-1$. The value is passed as a 2-byte signed binary number.
4. The value must meet the following condition: $-2^{31} \leq \text{value} \leq 2^{31}-1$. The value is passed as a 4-byte signed binary number.
5. The length of the data accepted and passed is defined by the ELEM or QUAL statement that the label identifies.

RTNVAL Parameter: Specifies whether a value is to be returned by the CPP via the parameter being defined in this PARM statement.

*NO: No value can be returned in the parameter being defined. The parameter is an input parameter only.

**YES:* A value is to be returned by the CPP in the parameter. A CL variable name must be specified (on the CALL command) to receive the value. *YES is valid only if the TYPE parameter was specified as *DEC, *CHAR, *LGL, or *X. Also, *YES is valid only on commands that are limited to CL programs. (That is, if either *BPGM or *IPGM is specified in the CRTCMD command that uses the source file containing this PARM statement, RTNVAL(*YES) can be specified.) *YES is not valid with MAX(>1), CONSTANT, DFT, RSTD, VALUES, REL, RANGE, SPCVAL, SNGVAL, FILE, FULL, or EXPR. VARY(*YES) must be specified if PASSATR(*YES) and RTNVAL(*YES) have been specified.

CONSTANT Parameter: Specifies that a value is to be passed to the CPP as a constant when the command being defined is processed; the parameter is not to appear externally on the command. The value specified in this parameter (if any) must satisfy the requirements specified by the TYPE, LEN, VALUES, REL, RANGE, SPCVAL, and FULL parameters. (As noted in the LEN parameter chart, if a character constant is specified in this parameter, it can be no longer than 32 characters.)

If a constant is specified for the parameter being defined, no prompt text can be specified for the PROMPT parameter, because the parameter will not be prompted for.

CONSTANT is not valid for TYPE(*CMD), TYPE(*NULL), TYPE(*ZEROELEM), MAX(>1), DFT, RTNVAL(*YES), or EXPR(*YES).

RSTD Parameter: Specifies whether the value entered for the parameter (specified in the PARM statement) is restricted to only one of the values given in the VALUES, SPCVAL, or SNGVAL parameters, or whether the value can be any value that satisfies the requirements specified by the TYPE, LEN, REL, RANGE, SPCVAL, SNGVAL and FULL parameters.

*NO: The value entered for the parameter specified by KWD in this PARM statement can be anything that matches the requirement specified by the following parameters in this PARM statement: TYPE, LEN, REL, RANGE, SPCVAL, SNGVAL and FULL.

**YES:* The value entered for the parameter specified by KWD in this PARM statement is restricted to one of the values in the VALUES parameter, or to one of the from-values in the SPCVAL or SNGVAL parameters. *YES cannot be specified if TYPE(statement-label), TYPE(*CMD), TYPE(*NULL), TYPE(*ZEROELEM), TYPE(*X), or RTNVAL(*YES) is specified.

DFT Parameter: Specifies the default value that is to be assigned to the parameter if a value is not specified by the user. That is, the default value is used as the value of the parameter if the user omits the parameter while entering the command or if he enters *N as the parameter value. The default value must satisfy one of the following:

- It must match the requirements specified by TYPE, LEN, REL, RANGE, and FULL.
- It must be one of the from-values in the SPCVAL or SNGVAL parameters.
- If the default is to be a character constant, it can have no more than 32 characters (as noted in the LEN parameter chart).
- If RSTD(*YES) is specified, it must be in the list of values in the VALUES parameter or in the list of from-values of the SPCVAL or SNGVAL parameters.
- It must be a from-value in the SNGVAL parameter if the parameter being defined is a list of unlike values or it is a qualified name (this is true when a statement label is specified for TYPE; the label is used to identify a QUAL or ELEM statement).

The DFT parameter is not valid if CONSTANT is specified. The DFT parameter is valid only if MIN is 0, meaning the parameter named in KWD is optional. No default can be specified if RTNVAL(*YES) is specified; instead a null pointer is passed for the default. A default cannot be specified if TYPE(*CMD), TYPE(*ZEROELEM), or TYPE(*NULL) is specified. If TYPE(*VARNAME) is specified, a default special value can be specified; a default variable name cannot.

If DFT is not specified and MIN(0) and RTNVAL(*NO) are specified, then the default assumed is as follows, depending on the specified type:

Assumed Default	**Types**
0	*DEC, *INT2, *INT4
'0'	*LGL
zeros	*DATE, *TIME, *HEX
blanks	*CHAR, *NAME, *GENERIC, *VARNAME
null	*CMD, *X, *NULL

An *assumed* default value is not displayed by the command prompter; a blank input field is shown instead. If a default is specified in the DFT parameter, it is displayed by the prompter exactly as specified.

value: Enter the default value that meets the specified requirements or that is one of the values specified in the VALUES, SPCVAL, or SNGVAL parameters.

VALUES Parameter: Specifies a list of 1 to 50 constants (fixed values) from which one constant can be entered as the value of the parameter named in KWD. The VALUES parameter is valid only if all of the following are true: RSTD(*YES) is specified, both RANGE and REL are *not* specified, and each constant matches the attributes specified by the TYPE, LEN, and FULL parameters. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.) Enter the constants (not more than 50) that can be entered as the value of the parameter. The VALUES parameter is not valid if TYPE(*CMD), TYPE(*X), TYPE(*NULL), TYPE(statement-label), TYPE(*VARNAME), TYPE(*ZEROELEM), or if RTNVAL(*YES) is specified.

REL Parameter: Specifies the relationship between the parameter value of this parameter and the value of a constant or another parameter. If a keyword is specified, it must be preceded by an ampersand (&) to indicate that it is the value of the keyword that is to be tested. The value associated with the referenced keyword is the value passed to the CPP, not the user-specified value. If the relationship is with another parameter whose value is a list of values or a qualified name, the first value only is used in the comparison.

To specify the relationship, enter one of the following relational operators followed by either a constant or the keyword name (KWD) of the other parameter (which must be preceded by an &).

*LT	(less than)
*LE	(less than or equal to)
*EQ	(equal to)
*GE	(greater than or equal to)
*GT	(greater than)
*NL	(not less than)
*NE	(not equal to)
*NG	(not greater than)

The REL parameter is not valid if RTNVAL(*YES) is specified, if either RANGE or VALUES is specified, or if TYPE is specified as *LGL, *VARNAME, *CMD, *X, *ZEROELEM, *NULL, or a statement label.

If a character type is specified by TYPE(*CHAR), the EBCDIC value of the character string is used as an unsigned integer in the comparison. (As noted in the LEN parameter chart, if a character constant is specified in this parameter, it can be no longer than 32 characters.)

RANGE Parameter: Specifies the range, or limits, within which the value of the parameter must be. The parameter value must be greater than or equal to the lower limit value specified, and it must be less than or equal to the upper limit value specified. For example, 15 would be valid if RANGE was specified as (0 16).

The upper and lower limits of the range can be specified either by a keyword representing the value or by the value itself. If a keyword is specified, it must be preceded by an ampersand (&) to indicate that it is the value of the keyword that is to be tested. The value of its parameter at the time of the check is used to determine the range. The value that is tested is the value passed to the CPP, not the user-specified value. If the keyword identifies a list of values or a qualified name, only the first value is used as the range limit. A keyword may not reference a parameter that is defined with PASSVAL(*NULL), and RANGE is not valid with PASSVAL(*NULL).

The RANGE parameter is not valid if RTNVAL(*YES) is specified, if either REL or VALUES is specified, or if TYPE is specified as *LGL, *VARNAME, *CMD, *X, *ZEROELEM, *NULL, or statement label. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.)

SPCVAL Parameter: Specifies a list of 1 to 50 entries that define special values that can be entered on the parameter named in KWD. Each entry specifies a character string (from value) that can be entered even though it may not meet all validity checking criteria. If the entered character string matches the from-value of one of the entries, and the to-value is specified, the string is replaced with the to-value and is then passed to the CPP without further checking. If the to-value is omitted, the from-value is passed to the CPP. SPCVAL is not valid if TYPE is specified as *CMD, *X, *NULL, statement-label, *ZEROELEM, or if RTNVAL(*YES) is specified.

The from-value is a character string, but the to-value can be anything that is passable. (If a CL variable is used for the from-value, its type must be *CHAR.) The to-value must be no longer than LEN specifies and, if TYPE is *DEC, *INT2, or *INT4, the type of the to-value must be the same; if TYPE is a character type (such as *CHAR, *LGL or *DATE), the to-value must be a character string. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.) If a to-value is not specified, the from-value must be passable.

SNGVAL Parameter: Specifies a list of 1 to 25 single values that can be specified for a parameter being defined as a mixed list or as a qualified name (when a statement label is specified for TYPE); or specifies that it is to accept two or more values (defined by the MAX parameter). Any one of the single values can be used instead of a list of values or a qualified name that the parameter is defined to accept. Each entry specifies a character string (from-value) that can be entered. If an entered character string matches the from-value of one of the entries and the to-value is specified, the data is replaced with the to-value and is then passed to the CPP without further checking. If the to-value is omitted, the from-value is passed to the CPP.

The to-value (or the from-value, if the to-value is omitted) must be passable, as specified in the SPCVAL parameter. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.) SNGVAL can be specified only if MAX is greater than one or TYPE is specified as a statement label of a QUAL or ELEM statement. Each single value can only substitute for a list of values or a qualified name; it cannot be a list element or qualifier. It is passed as the first and only element of the list.

SNGVAL is not valid if TYPE is specified as *CMD, *NULL, *ZEROELEM, *X, or if RTNVAL(*YES) is specified.

MIN Parameter: Specifies the minimum number of values that must be entered for the parameter being defined. For a parameter that does not allow multiple like values, only zero (0) for optional and one (1) for required can be specified as the minimum number of values.

For a parameter that allows multiple like values (because a value greater than one is specified for the MAX parameter), zero (0) indicates that no values need be entered; therefore, it is an *optional* parameter. A value of one (1) or greater than one indicates the minimum number of values that must be entered for the parameter, and, therefore, it is a *required* parameter. The value specified for MIN cannot exceed the value specified for the MAX parameter and cannot exceed 1 for TYPE(*NULL).

0: The parameter is optional; it does not have to be entered.

minimum-number: Enter the minimum number of elements that must be specified for this parameter (named by KWD). If 1 is assigned as the value, it specifies that one value is required for the parameter. If a number greater than 1 is specified, the parameter is a simple list that must have at least as many elements as the number specified.

MAX Parameter: Specifies, if this PARM statement is defining a simple list parameter, the maximum number of list elements that this list parameter can contain. If a value greater than 1 is specified, the parameter is capable of accepting multiple like values (that is, a simple list). This support is primarily intended for IBM-supplied commands. All values entered for this parameter (at the time the command is to be executed) must satisfy the validity checking requirements specified by the other parameter values on this PARM statement.

Note: The values for a list parameter are passed consecutively, preceded by a 2-byte binary value that indicates the number of values entered in the parameter by the user. CL programs do not support the handling of binary values in variables.

1: The parameter accepts only one value; the parameter is not a list parameter.

maximum-number: Enter the maximum number of elements that the list parameter can accept. The specified maximum must be greater than or equal to the value specified in MIN, and less than or equal to 50. If the maximum is greater than 1 and TYPE is not a statement label that identifies a QUAL or ELEM statement, the parameter is a simple list of like elements (that is, each element in the list has the same requirements, such as type and length). If TYPE(statement-label) is specified and it points to the label of an ELEM or QUAL statement, MAX should only be specified greater than 1 if a list of lists or a list of qualified names is to be accepted. A maximum greater than 1 is not valid if TYPE is specified as *CMD or *NULL, or if RTNVAL(*YES) or CONSTANT is specified.

FILE Parameter: Specifies whether the parameter is a file name and what the expected use of the file is. The parameter can be specified as the name of a file that has a specific use so that, at compile time, the names can be used to get file reference information about where the files are used. The specification in the FILE parameter does not have any effect on the operation of the parameter being defined; it only indicates to the compiler that the value for this parameter is a file name and what type of file it is. This information is stored so that it can be included in the output of the DSPPGMREF (Display Program References) command. FILE is valid only for TYPEs *CHAR, *NAME, *GENERIC, and statement-label. FILE is not valid with RTNVAL(*YES). One of the following types of files can be specified:

*NO: The parameter (named by KWD) is not a file name.

**IN:* The parameter value is an input file name.

**OUT:* The parameter value is an output file name.

**UPD:* The parameter value is an update file name.

**INOUT:* The parameter value is the name of a file that is to be used for both input and output.

**UNSPFD:* The parameter value is the name of a file, but its use cannot be specified.

The use of the file must match the type of file specified. For example, if *IN is specified, the file can be used only for input; if *UPD is specified, it can be used only to update existing records.

FULL Parameter: Specifies whether the number of characters in the parameter must be exactly the same as the number specified in the LEN parameter (if specified) or its default length (if LEN is not specified).

<u>*NO:</u> The number of characters in the parameter can be less than that specified by LEN.

**YES:* The number of characters in the parameter must equal the number specified by LEN or the default length for that type. The exact length is valid only for the following parameter types: *LGL, *CHAR, *NAME, *GENERIC, *VARNAME, and *HEX. FULL(*YES) is valid with RTNVAL(*YES).

EXPR Parameter: Specifies whether the parameter (named in the KWD parameter) can accept an expression containing a character concatenation. Valid character concatenation operators are:

*CAT	or	||	Concatenation
*BCAT	or	|>	Blank insertion with concatenation
*TCAT	or	|<	Blank truncation with concatenation

For examples of the use of these operators in expressions, refer to the topic *Character String Expressions* in Appendix B.

Restrictions: Expressions are not allowed on parameters where the TYPE parameter specifies *CMD, *ZEROELEM, *NULL, or statement-label.

<u>*NO:</u> The parameter value cannot be a concatenation expression.

**YES:* The parameter value can be a concatenation expression.

VARY Parameter: Specifies whether the value that is to be passed to the CPP is to be preceded by a length value that indicates the number of characters entered for the command parameter.

*NO: The parameter value is not to be preceded by a length value.

**YES:* The parameter value passed to the CPP is preceded by a 2-byte binary length field that indicates the number of characters actually specified for the parameter. CL programs do not support binary values in variables. The data is passed in a field of the length specified by LEN or by the default length. *YES is valid only for the following parameter types: *CHAR, *NAME, *GENERIC, *LGL, *VARNAME, *CMD, and *X. *YES must be specified if PASSATR(*YES) and RTNVAL(*YES) are specified.

PASSATR Parameter: (For IBM-supplied commands) Specifies whether an attribute byte is to be passed to the CPP with the parameter data.

*NO: No attribute byte is to be passed with the parameter.

**YES:* An attribute byte is passed with the parameter; the attribute byte indicates whether the data value came from the default, the data type of the value, and, if TYPE(*CHAR) was specified, whether the character string was enclosed in apostrophes.

PASSVAL Parameter: Specifies whether a value is to be passed to the command processing program for this parameter. *NULL is not valid if the parameter is a constant parameter (a parameter in which a value has been specified for the CONSTANT parameter on the PARM statement, a parameter for which TYPE(*ZEROELEM) or TYPE(*NULL) has been specified, or is a list/qualified name defined by all constant ELEM or QUAL statements). *NULL is also not valid if RTNVAL(*YES) has been specified or if the value specified for the MIN parameter is greater than zero. A DEP statement or the REL and RANGE keywords of other PARM statements may not reference the value of a parameter defined with PASSVAL(*NULL).

*DFT: The default value is always passed to the CPP (command processing program).

**NULL:* A null pointer is passed to the CPP if the parameter is not specified.

PROMPT Parameter: Specifies what prompt text, if any, is to be used for the parameter named by KWD. The prompt text further describes the keyword and input field to the user, who may enter a response to the information displayed. For example, the prompt text for the TYPE parameter on the Create Library (CRTLIB) command is:

Library Type (*PROD *TEST):

When the prompt for the command is displayed, the prompt for the TYPE parameter is:

Library Type (*PROD *TEST): TYPE <u>*PROD</u>

The underscored field is the prompt input field where the user enters the value for the TYPE parameter. When the prompt is displayed, the prompt input field contains the default value (*PROD in this example). Prompt text cannot be specified if TYPE(*ZEROELEM) or TYPE(*NULL) is specified or if a constant value is specified in the CONSTANT parameter.

<u>*NONE:</u> No prompt text is to be displayed for the parameter defined by this PARM statement. This parameter will still be prompted for by its keyword name, but no text will be displayed with it.

message-identifier [relative-prompt-number]: Enter the message identifier that specifies the message containing the prompt text of up to 30 characters that is to be displayed when the parameter is being prompted. If a message having the specified identifier cannot be found in the message file specified in the PMTFILE parameter of the Create Command (CRTCMD) command, the message identifier itself is used as the prompt text.

Optionally, a relative prompt number can be specified with the message identifier. The relative prompt number specifies the order in which parameter keywords are to be prompted for. This order affects only the order of prompting, not the order in which the parameters are passed to the CPP. Parameters having prompt numbers are prompted before those parameters having no prompt numbers.

'prompt-text' [relative-prompt-number]: Enter the prompt text that is to be displayed when the parameter is being prompted. The text must be a character string of no more than 30 characters, enclosed in apostrophes. An optional relative prompt number can be specified with the prompt text, the same as for the message identifier option.

Examples

```
PARM  KWD(X)  TYPE(*DEC)  LEN(2)  MIN(1)  REL(*GT 5)
```

The value for the parameter named X, a two-digit decimal number, must be entered. The value must be greater than 5.

```
PARM  KWD(CLASS)  TYPE(*CHAR)  LEN(1)  +
      DFT(A)  VALUES(A B C)  RSTD(*YES)
```

The value of the parameter named CLASS must be A, B, or.C, if entered. If it is not present, A is assumed.

```
PARM  KWD(MAXREC)  TYPE(*DEC)  LEN(3)  +
      MIN(1)  RANGE(&MINREC 999)
```

The value of the MAXREC parameter must be entered as a decimal number of three digits or less, with no digits to the right of the decimal point. The value must be greater than or equal to the value entered for parameter MINREC and also must be less than or equal to 999.

```
PARM  KWD(FILES)  TYPE(*NAME)  MIN(2)  MAX(5)
```

The FILES parameter is a homogeneous list of from two to five names.

```
PARM  KWD(INVFNAME)  TYPE(*NAME)  DFT(*ALL)  +
      SNGVAL((*ALL XXX))  VALUES(DEPT1 DEPT2 DEPT3)  +
      FILE(*UPD)  MIN(0)  MAX(3)  RSTD(*YES)  +
      PROMPT(USR0002 1)
```

The value of the parameter named INVFNAME can be a list of up to three file names of which DEPT1, DEPT2, DEPT3, and *ALL are the valid choices. If *ALL is entered, no other values can be entered for the parameter. If this parameter is omitted, file name XXX is passed to the command processing program. If this parameter is entered through the prompter, the prompt text, as stored in the message file, is displayed and *ALL is listed as the default.

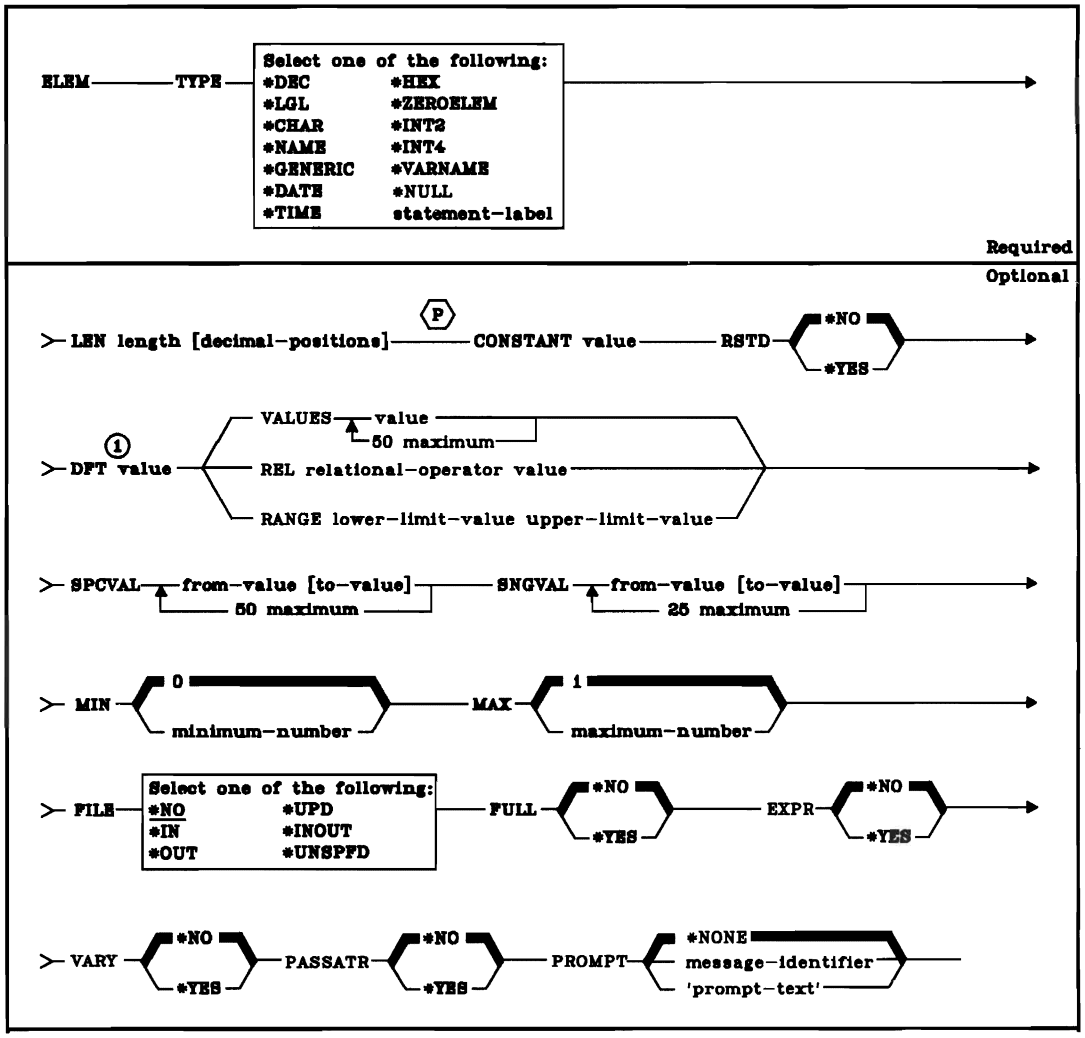
ELEM (Element) Statement

Element (ELEM) statements are used to define the elements of a mixed list parameter on a command. A list parameter is a parameter that accepts multiple values that are passed together as consecutive values pointed to by a single parameter. The values are preceded by a 2-byte binary value that indicates the number of elements defined for the parameter. CL programs do not support binary values in variables.

A list element is the value that represents one of a group of values organized in a specific order in a list. One ELEM statement must be used for each element that appears in the list being defined, if all of the list elements are not the same type. If all the elements are of the same type (a simple list), ELEM statements are not required. (For a simple list, all that is necessary is to specify the number of elements in the list on the MAX parameter of the PARM statement.)

The order in which the ELEM statements are entered into the source file determines the positional order of occurrence in the list. The first ELEM statement (for the first list element) must have a statement label that matches TYPE (statement-label) on the PARM or ELEM statements for the same list. The remaining ELEM statements in the list must be unlabeled. Lists of different values can be nested to the depth of three levels including the highest level. A maximum of 50 elements can be included in one list.

Note: The ELEM statement contains certain parameters and predefined values that can be used only when IBM-supplied CPPs are to be invoked by the command being defined. Because of limitations in some high-level languages, these values may not be useful in the definition statements of user-defined commands. These parameters and values are identified by the phrase (*For IBM-supplied commands*) that immediately follows the parameter keyword (if the entire parameter is for IBM-supplied commands only) or the predefined value to which it applies.



TYPE Parameter: Specifies the type of list element being defined. The element can be an integer, a decimal or logical value, or a quoted or unquoted character string that can be name, label, date, or time. Enter one of the following options to specify the type of element:

**DEC:* The list element is a packed decimal number.

**LGL:* The list element is a logical value of one ('1') or zero ('0').

**CHAR:* The list element is a character string that can (optionally) be enclosed in apostrophes. If the character string contains any special characters (not including an *), it *must* be enclosed in apostrophes. The maximum number of characters that can be in the character string is 2000.

**NAME:* The list element is a character string, representing a name. The maximum length of the name is 256 characters; the first character must be alphabetic, and the remaining characters must be alphameric or an underscore. If a special value is used (as in *LIBL or *NONE), it should be specified in the SPCVAL parameter.

**GENERIC:* The list element is a character string, representing a generic name. A generic name contains a maximum of 255 characters followed by an asterisk (*) or 256 characters without an asterisk; the name identifies a group of objects whose names all begin with the characters preceding the *. (If an * is not included, the system assumes that the generic name is a complete object name.)

**DATE:* The list element is a character string that represents a date. When it is passed to the CPP, it is always passed in the format *Cyymmdd*, where C = century, yy = year, mm = month, and dd = day. The century digit is set to 0 (zero) if yy is equal to or greater than 40 (1940 through 1999); it is set to 1 (one) if yy is less than 40 (2000 through 2039). When a date value is specified in this ELEM statement, it must be specified in the format *mmyydd*. When a user enters a date in the command at execution time, it must be entered in the format specified by the system value QDATFMT. The system value QDATSEP specifies the optional separator character that can be used when the date is entered. If the separator character is used, the date must be enclosed in apostrophes.

**TIME:* The list element is a character string that represents a time. It is in the format *hh:mm:ss*, where hh = hours, mm = minutes, and ss = seconds. It is passed to the CPP in a 6-byte character string as *hhmmss*. Any values specified in this statement must be in the format *hhmmss*.

**HEX:* The list element value is hexadecimal in form. The specified characters must be 0 through F. They will be converted to hexadecimal (EBCDIC) characters (two hex digits per byte), right adjusted, and padded with zeros. If the value is enclosed in apostrophes, an even number of digits is required. If the value is not enclosed in apostrophes, an even number of digits is *not* required.

**ZEROELEM:* The list element is always to be considered as a list of zero elements, for which no value can be specified in the command. It is used to prevent a value from being entered for an element that is a list even though the CPP expects one. An element for which *ZEROELEM is specified is not prompted for, although the other elements in the parameter are prompted and are passed to the CPP as a list.

**INT2:* The list element is an integer that is to be passed as a 2-byte signed binary number. CL programs do not support binary values in variables.

**INT4:* The list element is an integer that is to be passed as a 4-byte signed binary number. CL programs do not support binary values in variables.

**VARNAME:* (For IBM-supplied commands) The list element is a variable name that is to be passed as a character string. The name can contain a maximum of 11 characters, including the ampersand (&).

statement-label: The list element accepts a qualified name or a mixed list of values. The statement label specified here by the TYPE parameter is the statement label that identifies the first of a series of QUAL or ELEM statements that further describe the qualified name or the mixed list being defined. The label must be the same as the label specified by TYPE(statement-label) on the PARM statement for this list.

LEN Parameter: Specifies the length of the list element value. If TYPE was specified as *INT2, *INT4, *DATE, *TIME, *ZEROELEM, or statement-label, LEN is not allowed. If TYPE(*DEC) was specified, the decimal length is specified in the form (n1 n2), where n1 specifies.the total number of digits in the value (including the decimal portion), and n2 specifies the number of allowable decimal digits to the right of the decimal point. (The value for n2 is optional. Zero is assumed if n2 is not entered.) If TYPE is other than *DEC, the decimal portion (n2) must be omitted and simply the number of characters must be specified.

The default length that is assumed by the system and the maximum length for each type of parameter are shown in the following table:

Data Type	**Default Length**	**Maximum Length**
*DEC	(15 5)	(24 9)
*LGL	1	1
*CHAR	32	2000 (see note)
*NAME	10	256 (see note)
*GENERIC	10	256 (see note)
*HEX	1	256 (see note)
*VARNAME	11	11

Note: Whereas the maximum shown here is the maximum for values used in the command when it is executed, the maximum length of character constants specified in the command definition is limited to 32 characters. This restriction applies to the following parameters: CONSTANT, DFT, VALUES, REL, RANGE, SPCVAL, and SNGVAL.

For those data types for which length cannot be coded, the following are the maximum lengths and the lengths passed.

Data Type	Maximum Length	Length Passed
*DATE (Note 1)	8	7
*TIME (Note 2)	8	6
*ZEROELEM	0	0
*INT2 (Note 3)	6	2
*INT4 (Note 4)	11	4
*CMD	Any	Length needed
statement-label (Note 5)	–	–

Notes:

1. If a date is specified, the value is passed as 7 characters.
2. If a time is specified, the value is passed as 6 characters.
3. The value must meet the following condition: $-2^{15} \leq$ value $\leq 2^{15}-1$. The value is passed as a 2-byte signed binary number.
4. The value must meet the following condition: $-2^{31} \leq$ value $\leq 2^{31}-1$. The value is passed as a 4-byte signed binary number.
5. The length of the data accepted and passed is defined by the ELEM or QUAL statement that the label identifies.

CONSTANT Parameter: Specifies that a value is to be passed to the CPP as a constant for the list element when the command being defined is processed; the element is not to appear externally on the command. If specified, the value must satisfy the requirements specified by the TYPE, LEN, VALUES, REL, RANGE, SPCVAL, and FULL parameters. (As noted in the LEN parameter chart, if a character constant is specified in this parameter, it can be no longer than 32 characters). CONSTANT is not valid with TYPE(*ZEROELEM), EXPR(*YES), MAX(>1), or if DFT was coded for ELEM.

If a constant is specified for the element being defined, no prompt text can be specified for the PROMPT parameter of this ELEM statement. However, the other elements of the list parameter (of which this list element is a part) are still prompted for, and their values along with this constant value are still passed to the CPP as a list.

RSTD Parameter: Specifies whether the value entered for the list element (specified in the ELEM statement) is restricted to only one of the values given in the VALUES, SPCVAL, or SNGVAL parameters, or whether the value can be any value that satisfies the requirements specified by the TYPE, LEN, REL, RANGE, SPCVAL, SNGVAL and FULL parameters.

*NO: The value entered for the list element defined by this ELEM statement can be anything that matches the requirements specified by the following parameters in this ELEM statement: TYPE, LEN, REL, RANGE, SPCVAL, SNGVAL and FULL.

*YES: The value entered for the list element in this ELEM statement is restricted to one of the values in the VALUES parameter, or to one of the from-values in the SPCVAL or SNGVAL parameter. *YES cannot be specified if TYPE(statement-label) or TYPE(*ZEROELEM) is specified.

DFT Parameter: Specifies the default that is to be assigned to the list element if a value is not specified by the user. That is, the default value is used as the value of the list element if the user omits the parameter that represents this list element, or specifies *N for the element, while coding or entering the command. The default value must satisfy one of the following:

- It must match the element requirements specified by TYPE, LEN, REL, RANGE, and FULL.
- It must be one of the from-values in the SPCVAL or SNGVAL parameters.
- If the default is to be a character constant; it can have no more than 32 characters (as noted in the LEN parameter chart).
- If RSTD(*YES) is specified, it must be in the list of values in the VALUES parameter or in the list of from-values of the SPCVAL or SNGVAL parameters.
- If this ELEM statement itself defines a list, the default value must be specified in the SNGVAL parameter.

The DFT parameter is valid only if MIN is 0, meaning the element defined by this ELEM statement for this list is optional. The DFT parameter is invalid if CONSTANT is specified. A default cannot be specified if TYPE(*ZEROELEM) is specified; in that case, an assumed default is passed.

If DFT is not specified, the default assumed is as follows, depending on the specified type:

Assumed Default	Types
0	*DEC, *INT2, *INT4, *ZEROELEM
'0'	*LGL
zeros	*DATE, *TIME, *HEX
blanks	*CHAR, *NAME, *GENERIC, *VARNAME

An *assumed* default value is not displayed by the command prompter; a blank input field is shown instead. If a default is specified in the DFT parameter, it is displayed by the prompter exactly as specified.

value: Enter the default value that meets the specified requirements or that is one of the values specified in the SPCVAL, SNGVAL, or VALUES parameters.

VALUES Parameter: Specifies a list of 1 to 50 constants (fixed values) from which one constant can be entered as the value of the list element. The VALUES parameter is valid only if all of the following are true: RSTD(*YES) is specified, both RANGE and REL are *not* specified, and each constant matches the attributes specified by the TYPE, LEN, and FULL parameters in this ELEM statement. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.) Enter the constants (not more than 50) that can be entered as the value of the list element. The VALUES parameter is not valid for TYPE(statement-label) or for TYPE(*ZEROELEM).

If this ELEM statement is defining the first element in a list, the value specified for this parameter cannot be the same as the value specified in the SNGVAL parameter on either the PARM or ELEM statement that points to this ELEM statement.

REL Parameter: Specifies the relationship between the list element value and the value of another parameter or constant. The value associated with the referenced keyword is the value passed to the CPP, not the user-specified value. To specify the relationship, enter one of the following relational operators followed by a constant or the value of another parameter.

*LT	(less than)
*LE	(less than or equal to)
*EQ	(equal to)
*GE	(greater than or equal to)
*GT	(greater than)
*NL	(not less than)
*NE	(not equal to)
*NG	(not greater than)

The REL parameter is not valid if TYPE is specified as *LGL, *VARNAME, *ZEROELEM, or statement label; or if either RANGE or VALUES is specified.

If a character type is specified by TYPE(*CHAR), the EBCDIC value of the character string is used as an unsigned integer in the comparison. (As noted in the LEN parameter chart, if a character constant is specified in this parameter, it can be no longer than 32 characters.)

RANGE Parameter: Specifies the range, or the limits, within which the value of the list element must be. The list element value must be greater than or equal to the lower limit value specified, and it must be less than or equal to the upper limit value specified. The value tested is the value sent to the CPP, not the user-specified value. For example, 15 would be valid if RANGE was specified as (0 16).

The RANGE parameter is not valid if either REL or VALUES is specified, or if TYPE is specified as *LGL, *VARNAME, *ZEROELEM, or statement label. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.)

SPCVAL Parameter: Specifies a list of 1 to 50 entries that define special values that can be entered for the element defined by this ELEM statement. Each entry specifies a character string (from value) that can be entered even though it may not meet all validity checking criteria. If the entered character string matches the from-value of one of the entries, and the to-value is specified, the string is replaced with the to-value and is then passed to the CPP without further checking. If the to-value is omitted, the from-value is passed to the CPP. SPCVAL is not valid for TYPE(statement-label) or for TYPE(*ZEROELEM).

The from-value is a character string, but the to-value can be anything that is passable. (If a CL variable is used for the from-value, its type must be *CHAR.) If this ELEM statement is defining the first element in a list, the value specified for the from-value cannot be the same as the value specified in the SNGVAL parameter on either the PARM or ELEM statement that points to this ELEM statement.

The to-value must be no longer than LEN specifies and, if TYPE is *DEC, *INT2, or *INT4, the type of the to-value must be the same; if TYPE is a character type (such as *CHAR, *LGL, or *DATE), the to-value must be a character string. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters). If a to-value is not specified, the from-value must be passable.

SNGVAL Parameter: Specifies a list of 1 to 25 single values that can be specified for an element being defined as a statement label, or that is to have two or more list elements in its nested list (defined by the MAX parameter). Any one of the single values can be used instead of a nested list of values or a qualified name that the element is defined to accept. Each entry specifies a character string (from-value) that can be entered. If an entered character string matches the from-value of one of the entries and the to-value is specified, the data is replaced with the to-value and is then passed to the CPP without further checking. If the to-value is omitted, the from-value is passed to the CPP.

If this ELEM statement is defining the first element in a list, the value specified for the from-value cannot be the same as the value specified in the SNGVAL parameter on either the PARM or ELEM statement that points to this ELEM statement.

The to-value (or the from-value, if the to-value is omitted) must be passable, as specified in the SPCVAL parameter. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.) SNGVAL can be specified only if MAX is greater than one or TYPE is specified as a statement label; it is not valid for TYPE(*ZEROELEM). Each single value can only substitute for a list of values or a qualified name; it cannot be a list element or qualifier. It is passed as the first element of the list.

MIN Parameter: Specifies the minimum number of values that must be entered for the element being defined. For an element that does not allow multiple like values, only zero (0) for optional and one (1) for required can be specified as the minimum number of values.

For an element that allows multiple like values (because a value greater than one is specified in the MAX parameter), zero (0) indicates that no values need be entered; therefore, it is an *optional* element. A value of one (1) or greater than one indicates the minimum number of values that must be entered for the element, and therefore it is a *required* element. The value specified for MIN cannot exceed the value specified for the MAX parameter.

(The number specified tells how many list elements are required in another list.) If MIN is not specified, zero (0) is assumed, meaning that the element is optional.

0: The list element is optional; it does not have to be entered.

minimum-number: Enter the minimum number of elements that must be specified in the nested list. If 1 is assigned as the value, it specifies that one value is required for the element. If a number greater than 1 is specified, the element contains a simple list that must have at least as many elements as the number specified.

MAX Parameter: Specifies, if this ELEM statement is defining a simple list element, the maximum number of elements that this list element can have in its nested list. If a value greater than 1 is specified, the element is capable of accepting multiple like values (that is, a simple nested list). All values entered for this element (at the time the command is to be executed) must satisfy the validity checking requirements specified by the other parameter values on this ELEM statement.

Note: The values for a nested list are passed consecutively, preceded by a 2-byte binary value that indicates the number of values entered in the list element by the user. CL programs do not support the handling of binary values in variables.

1: The list element accepts only one value; there is no nested list.

maximum-number: Enter the maximum number of elements that the list element can accept. The specified maximum must be greater than or equal to the value specified in MIN, and less than or equal to 50. If the maximum is greater than 1 and TYPE is not a statement label that identifies a QUAL statement or another ELEM statement, the parameter—which is an element itself—is a simple list of like values (that is, each element in the list has the same requirements, such as type and length). If TYPE(statement-label) is specified and it points to the label of a QUAL statement or another ELEM statement, MAX should only be specified greater than 1 if a list of lists or a list of qualified names is to be accepted. A maximum greater than 1 is not valid if CONSTANT is also specified.

FILE Parameter: Specifies whether the list element is a file name and what the expected use of the file is. The element can be specified as the name of a file that has a specific use so that, at compile time, the names can be used to get file reference information about where the files are used. FILE is valid only if TYPE is statement-label, *CHAR, *NAME, or *GENERIC. The specification in the FILE parameter does not have any effect on the list element being defined by the ELEM statement; it only indicates to the compiler that the value for this element is a file name and what type of file it is. This information is stored so that it can be included in the output of the DSPPGMREF (Display Program References) command. One of the following types of files can be specified:

*NO: The list element (defined in this ELEM statement) is not a file name.

**IN:* The list element is an input file name.

**OUT:* The list element is an output file name.

**UPD:* The list element is an update file name.

**INOUT:* The list element value is the name of a file that is to be used for both input and output.

**UNSPFD:* The list element value is the name of a file, but its use cannot be specified.

FULL Parameter: Specifies whether the number of characters in the list element must be exactly the same as the number specified in the LEN parameter (if specified) or its default length (if LEN is not specified).

*NO: The number of characters in the list element can be less than that specified by LEN.

*YES: The number of characters in the list element must equal the number specified by LEN or the default length for that type. The exact length is valid only for the following element types: *LGL, *CHAR, *NAME, *GENERIC, *VARNAME, and *HEX.

EXPR Parameter: Specifies whether the element can accept an expression containing a character concatenation. Valid character concatenation operators are:

*CAT	or	||	Concatenation
*BCAT	or	|>	Blank insertion with concatenation
*TCAT	or	|<	Blank truncation with concatenation

For examples of using these operators in expressions, refer to the topic *Character String Expressions* in Appendix B.

*NO: The element value cannot be a concatenation expression.

*YES: The element value can be a concatenation expression. *YES is not valid if a value is specified for CONSTANT.

VARY Parameter: Specifies whether the list element value that is to be passed to the CPP is to be preceded by a length value that indicates the number of characters entered for the element's value.

*NO: The element value is not to be preceded by a length value.

*YES: The element value passed to the CPP is preceded by a 2-byte binary length field that indicates the number of characters actually specified for the list element. CL programs do not support binary values in variables. The data is passed in a field of the length specified by LEN or by the default length. *YES is valid only for the following element types: *CHAR, *NAME, *GENERIC, and *VARNAME.

PASSATR Parameter: (For IBM-supplied commands) Specifies whether an attribute byte is to be passed to the CPP with the list element data. PASSATR is not valid for TYPE(statement-label) or for TYPE(*ZEROELEM).

*NO: No attribute byte is to be passed with the list element.

*YES: An attribute byte is passed with the list element; the attribute byte indicates whether the data value came from the default, the data type of the value, and, if TYPE(*CHAR) was specified, whether the character string was enclosed in apostrophes.

PROMPT Parameter: Specifies what prompt text, if any, is to be used for the list element (defined in this ELEM statement). The prompt text describes the element to the user, who may enter a response to the information displayed. Prompt text cannot be specified if TYPE(*ZEROELEM) is specified or if a constant value is specified in the CONSTANT parameter.

<u>*NONE:</u> No prompt text is to be displayed for the list element defined by this ELEM statement. This list element will still be prompted for by an input field, but no text will be displayed with it.

message-identifier: Enter the message identifier that specifies the message containing the prompt text of up to 30 characters that is to be displayed when the program is prompting for the list element. If a message having the specified identifier cannot be found in the message file specified in the PMTFILE parameter of the Create Command (CRTCMD) command, the message identifier itself is used as the prompt text.

'prompt-text': Enter the prompt text that is to be displayed when the program is prompting for the list element. The text must be a character string of no more than 30 characters, enclosed in apostrophes.

Examples

Example 1. Defining a JOBDESC Parameter

Parameter syntax:

```
——JOBDESC job-name number——
```

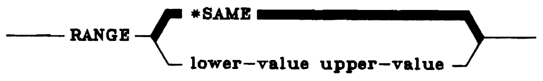
Command definition statements:

```
     PARM  KWD(JOBDESC) TYPE(L1)
L1:  ELEM  TYPE(*NAME) MIN(1)
     ELEM  TYPE(*DEC) LEN(2) MIN(1) REL(*LE 60)
```

The parameter named JOBDESC can be omitted, but, if present, it *must* be a list of two elements. The first element is a name, and the second is a two-digit number that is less than or equal to 60.

Example 2. Defining a RANGE Parameter

Parameter syntax:



Command definition statements:

```
      PARM  KWD(RANGE) TYPE(L1) DFT(*SAME) +
              SNGVAL((*SAME 101))
L1:   ELEM  TYPE(*DEC) MIN(1) REL(*LE 100)
      ELEM  TYPE(*DEC) MIN(1) REL(*LE 100)
```

The parameter named RANGE can be omitted, but, if present, it must be a list of two numbers, neither of which can be greater than 100. Validity checking is performed to determine that the lower-limit value is less than the upper-limit value. To allow the CPP to determine whether the value passed is a user-specified value or the default, *SAME is mapped to 101.

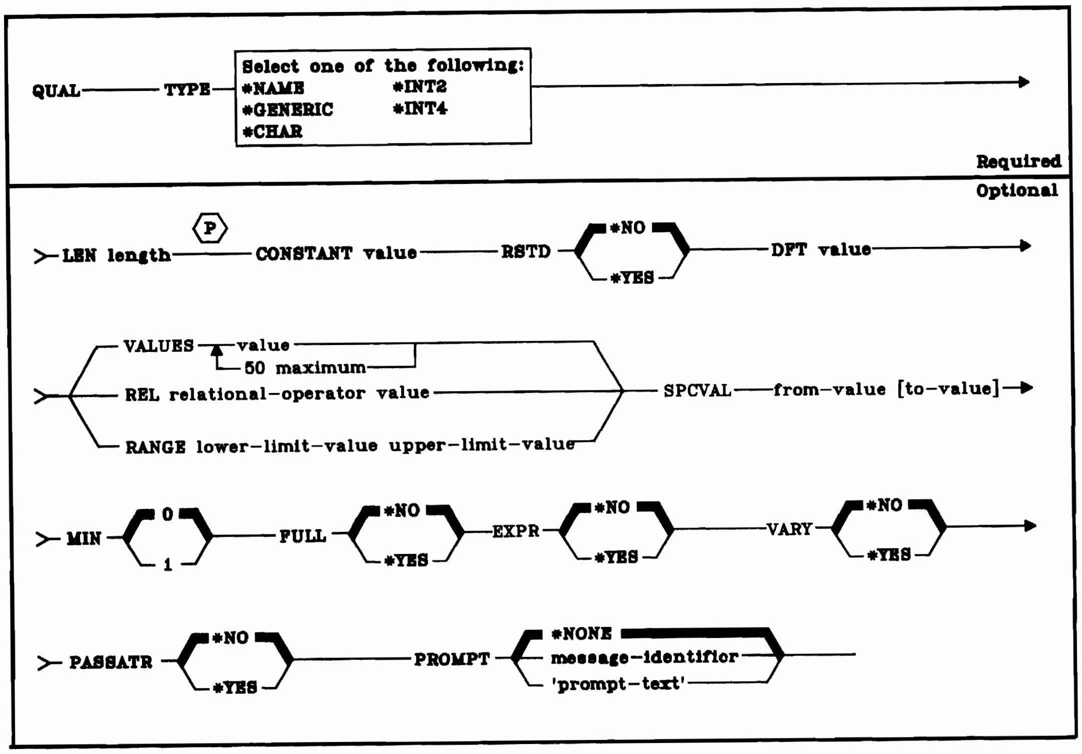
QUAL (Qualifier) Statement

The Qualifier (QUAL) statement describes one part of a qualified name. If a name is the allowable value of a parameter or list element defined in a PARM or ELEM statement, it can be made a qualified name by means of a QUAL statement for each qualifier used to qualify the name.

A maximum of 50 qualifiers can be used to make up the qualified name. The order in which the QUAL statements are entered into the source file determines the positional order in which the qualifiers must be specified and passed to the validity checker and the CPP. The first qualification of a qualified name must be either a simple name, a generic name, or a defined special value.

The QUAL statement (or only the *first* QUAL statement if there are more than one) *must* have a statement label that matches the statement label value that must be specified in a PARM or ELEM statement for which the qualifier is being defined. The qualifiers for the parameter or list element are then entered on the command in the form: value1[.value2[.value3]], where values 1 through 3 are qualifiers that are each described by a QUAL statement. The values are passed to the CPP in the same order, with the periods removed, and with each value padded to its maximum length.

Note: The QUAL statement contains certain parameters and predefined values that can be used only when IBM-supplied CPPs are to be invoked by the command being defined. Because of limitations in some high-level languages, these values may not be useful in the definition statements of user-defined commands. These parameters and values are identified by the phrase (*For IBM-supplied commands*) that immediately follows the parameter keyword (if the entire parameter is for IBM-supplied commands only) or the predefined value to which it applies.



TYPE Parameter: Specifies the type of qualifier used to qualify a parameter name or list element name. The qualifier can be a name or generic name, a quoted or unquoted character string, or an integer. Enter one of the following options to specify the type of qualifier. The first qualifier for any qualified name must have a type of name (*NAME) or generic name (*GENERIC).

**NAME:* The qualifier is a name. The maximum length of the name is 256 characters; the first character must be alphabetic, and the remaining characters must be alphameric or an underscore. If special value is to be used, it should be specified in the SPCVAL parameter.

**GENERIC:* The qualifier is a generic name. A generic name contains a maximum of 255 characters followed by an asterisk (*) or 256 characters without an asterisk; the name identifies a group of objects whose names all begin with the characters preceding the *. (If an * is not included, the system assumes that the generic name is a complete object name.)

**CHAR:* The qualifier is a character string that can (optionally) be enclosed in apostrophes. If the character string contains any special characters (not including an *), it *must* be enclosed in apostrophes. The maximum length of the character string is 2000 characters.

**INT2:* The qualifier is an integer that is to be passed as a 2-byte signed binary number. CL programs do not support binary values in variables.

**INT4:* The qualifier is an integer that is to be passed as a 4-byte signed binary number. CL programs do not support binary values in variables.

LEN Parameter: Specifies the length of the qualifier, if its data type is *NAME, *GENERIC, or *CHAR. The default length that is assumed by the system and the maximum length for each type of qualifier are shown in the following table:

Data Type	Default Length	Maximum Length
*NAME	10	256 (Note 3)
*GENERIC	10	256 (Note 3)
*CHAR	32	2000 (Note 3)
*INT2 (Note 1)	6	
*INT4 (Note 2)	11	

Notes:
1. For *INT2, length cannot be specified; its assumed length is 6. The value must meet the following condition: $-2^{15} \leq$ value $\leq 2^{15}-1$. The value is passed as a 2-byte signed binary number.
2. For *INT4, length cannot be specified; its assumed length is 11. The value must meet the following condition: $-2^{31} \leq$ value $\leq 2^{31}-1$. The value is passed as a 4-byte signed binary number.
3. Whereas the maximum shown here is the maximum for values used in the command when it is executed, the maximum length of character constants specified in the command definition is limited to 32 characters. This restriction applies to the following parameters: CONSTANT, DFT, VALUES, REL, RANGE, and SPCVAL.

CONSTANT Parameter: Specifies that a value is to be passed to the CPP as a constant for the qualifier when the command being defined is processed; the qualifier is not to appear externally on the command. If specified, the value must satisfy the requirements specified by the TYPE, LEN, VALUES, REL, RANGE, and SPCVAL parameters. (As noted in the LEN parameter chart, if a character constant is specified in this parameter, it can be no longer than 32 characters.)

If a constant is specified in this QUAL statement and other QUAL statements immediately follow it, they must also be defined as constants, unless a label precedes one of them. (A label indicates the beginning of a new group of QUAL statements, which can be defined differently.)

Also, if a constant is specified for the qualifier being defined, no prompt text can be specified for the PROMPT parameter of this QUAL statement. However, any other qualifiers or groups of qualifiers are still prompted for, and their values are still passed to the CPP as a qualified name.

CONSTANT is not valid if DFT or EXPR(*YES) is specified.

RSTD Parameter: Specifies whether the value entered for the qualifier is restricted to only one of the values given in the VALUES or SPCVAL parameters, or whether the value can be any value that satisfies the requirements specified by the TYPE, LEN, REL, RANGE, and SPCVAL parameters.

**NO:* The value entered for the qualifier defined by this QUAL statement can be anything that matches the requirements specified by the following parameters in this QUAL statement: TYPE, LEN, REL, RANGE, and SPCVAL.

**YES:* The value entered for the qualifier in this QUAL statement is restricted to one of the values in the VALUES parameter, or to one of the from-values in the SPCVAL parameters.

DFT Parameter: Specifies the default value that is to be assigned to the qualifier if a value is not specified by the user. The default value must satisfy one of the following:

- It must match the qualifier requirements specified by TYPE, LEN, REL, and RANGE.
- It must be one of the from-values in the SPCVAL parameter.
- If RSTD(*YES) is specified, it must be in the list of values in the VALUES parameter or in the list of from-values in the SPCVAL parameter.
- If the default is to be a character constant, it can have no more than 32 characters (as noted in the LEN parameter chart).

The DFT parameter is valid only if MIN is 0, meaning the qualifier defined by this QUAL statement for this list is optional. A default is not meaningful on this QUAL statement if it is the first one (defining the first part) for a qualified name and if a default is specified on the PARM or ELEM statement that this QUAL statement further defines.

If DFT is not specified, it has a default of its own: a blank (ƀ) if TYPE was specified as *CHAR, *NAME, or *GENERIC; or a zero (0) if TYPE was specified as *INT2 or *INT4. An *assumed* default value is not displayed by the command prompter; a blank input field is shown instead. If a default is specified in the DFT parameter, it is displayed by the prompter exactly as specified.

The DFT parameter is not valid if CONSTANT is specified.

value: Enter the default value that meets the specified requirements or that is one of the values specified in the SPCVAL or VALUES parameters.

VALUES Parameter: Specifies a list of 1 to 50 constants (fixed values) from which one constant can be entered as the value of the qualifier. The VALUES parameter is valid only if all of the following are true: RSTD(*YES) is specified, both RANGE and REL are *not* specified, and the constant matches the attributes specified by the TYPE and LEN parameters in this QUAL statement. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.) Enter the constants (not more than 50) that can be entered as the value of the qualifier.

REL Parameter: Specifies the relationship between the qualifier value and the value of another parameter or constant. To specify the relationship, enter one of the following relational operators followed by a constant or the value of another parameter.

*LT	(less than)
*LE	(less than or equal to)
*EQ	(equal to)
*GE	(greater than or equal to)
*GT	(greater than)
*NL	(not less than)
*NE	(not equal to)
*NG	(not greater than)

The REL parameter is not valid if either RANGE or VALUES is specified. If a character type is specified by TYPE(*CHAR), the EBCDIC value of the character string is used as an unsigned integer in the comparison. (As noted in the LEN parameter chart, if a character constant is specified in this parameter, it can be no longer than 32 characters.)

RANGE Parameter: Specifies the range, or limits, within which the value of the qualifier must be. The qualifier value must be greater than or equal to the lower limit value specified, and it must be less than or equal to the upper limit value specified. For example, 15 would be valid if RANGE was specified as (0 16). The RANGE parameter is not valid if either REL or VALUES is specified. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.)

SPCVAL Parameter: Specifies a list of 1 to 50 entries that define special values that can be entered on the parameter named in KWD on the PARM statement. Each entry specifies a character string (from-value) that can be entered even though it may not meet all validity checking criteria. If the entered character string matches the from-value of one of the entries, and the to-value is specified, the string is replaced with the to-value and is then passed to the CPP without further checking. If the to-value is omitted, the from-value is passed to the CPP. The from-value is a character string, but the to-value can be anything that is passable. (If a CL variable is used for the from-value, its type must be *CHAR.)

However, the first qualifier can only have special to-values with the from-values that are a name, a generic name, or an asterisk followed by a name (such as *ALL).

Each to-value must be passable to the CPP. The to-value must be no longer than LEN specifies and, if TYPE is *INT2 or *INT4, the type of the to-value must be the same; if TYPE is a character type (such as *CHAR or *NAME), the to-value must be a character string. (As noted in the LEN parameter chart, character constants specified in this parameter can be no longer than 32 characters.) If a to-value is not specified, the from-value must be passable.

MIN Parameter: Specifies whether the qualifier being defined in this QUAL statement is required or optional. If MIN is not specified, 0 is assumed, meaning the qualifier is optional. If a required qualified name is needed, MIN(1) must be coded on both the first QUAL and on the PARM or ELEM that references it.

_0:_ The qualifier is optional on the name being qualified.

1: The qualifier is required on the name being qualified; it must be entered.

FULL Parameter: Specifies whether the number of characters in the qualifier value must be exactly the same as the number specified in the LEN parameter (if specified) or its default length (if LEN is not specified).

_*NO:_ The number of characters in the qualifier value can be less than that specified by LEN.

**YES:* The number of characters in the qualifier value must equal the number specified by LEN or the default length for that type. The exact length is valid only for the following qualifier types: *CHAR, *NAME, and *GENERIC.

EXPR Parameter: Specifies whether the qualifier can accept an expression containing a character concatenation. Valid character concatenation operators are:

*CAT	or	||	Concatenation
*BCAT	or	|>	Blank insertion with concatenation
*TCAT	or	|<	Blank truncation with concatenation

For examples of using these operators in expressions, refer to the topic *Character String Expressions* in Appendix B.

NO: The qualifier value cannot be a concatenation expression.

**YES:* The qualifier value can be a concatenation expression.

VARY Parameter: Specifies whether the qualifier value that is to be passed to the CPP is to be preceded by a length value that indicates the number of characters entered for the qualifier's value.

NO: The qualifier value is not to be preceded by a length value.

**YES:* The qualifier value passed to the CPP is preceded by a 2-byte binary length field that indicates the number of characters actually specified for the qualifier. CL programs do not support binary values in variables. The data is passed in a field of the length specified by LEN or by the default length. *YES is valid only for the following qualifier types: *CHAR, *NAME, or *GENERIC.

PASSATR Parameter: (For IBM-supplied commands) Specifies whether an attribute byte is to be passed to the CPP with the qualifier.

NO: No attribute byte is to be passed with the qualifier.

**YES:* An attribute byte is passed with the qualifier; the attribute byte indicates whether the data value came from the default, the data type of the value, and, if TYPE(*CHAR) was specified, whether the character string was enclosed in apostrophes.

PROMPT Parameter: Specifies what prompt text, if any, is to be used for the qualifier (defined in this QUAL statement). The PROMPT parameter is not allowed for the first qualifier or for a qualifier for which CONSTANT is specified. (The prompt text for the first qualifier comes from the PROMPT parameter of the PARM or ELEM statement pointing to the qualifier.) The prompt text describes the input field for the qualifier to the user, who may enter a response to the information displayed.

*NONE: No prompt text is to be displayed for the qualifier defined by this QUAL statement. This qualifier will still be prompted for by an input field, but no text will be displayed with it.

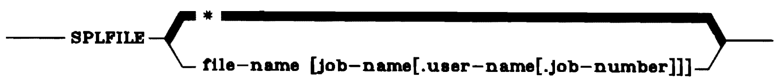
message-identifier: Enter the message identifier that specifies the message containing the prompt text of up to 30 characters that is to be displayed when the program is prompting for the qualifier. If a message having the specified identifier cannot be found in the message file specified in the PMTFILE parameter of the Create Command (CRTCMD) command, the message identifier itself is used as the prompt text.

'prompt-text': Enter the prompt text that is to be displayed when the program is prompting for the qualifier. The text must be a character string of no more than 30 characters, enclosed in apostrophes.

Examples

Example 1. Defining a SPLFILE Parameter

Parameter syntax:



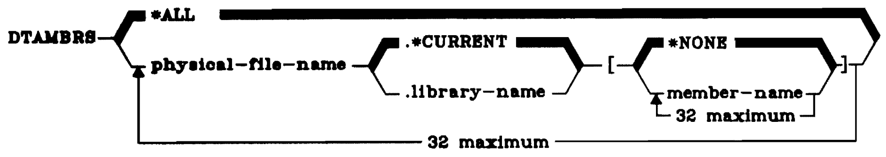
Command definition statements:

```
    PARM  KWD(SPLFILE) TYPE(L1) DFT(*) SNGVAL(*)
L1: ELEM  TYPE(*NAME) MIN(1)    /* For file name */
    ELEM  TYPE(Q1)              /* For job name */
Q1: QUAL  TYPE(*NAME) MIN(1)    /* For job name */
    QUAL  TYPE(*NAME)           /* For user name */
    QUAL  TYPE(*CHAR) LEN(6)    /* For job number */
```

The SPLFILE parameter is optional and, if not specified, defaults to *. Otherwise, the value consists of a two-element list. The first element is a file name and is required. The second element is a qualified job name. The first qualifier is required; the last two qualifiers are optional.

Example 2. Defining a DTAMBRS Parameter

Parameter syntax:



Command definition statements:

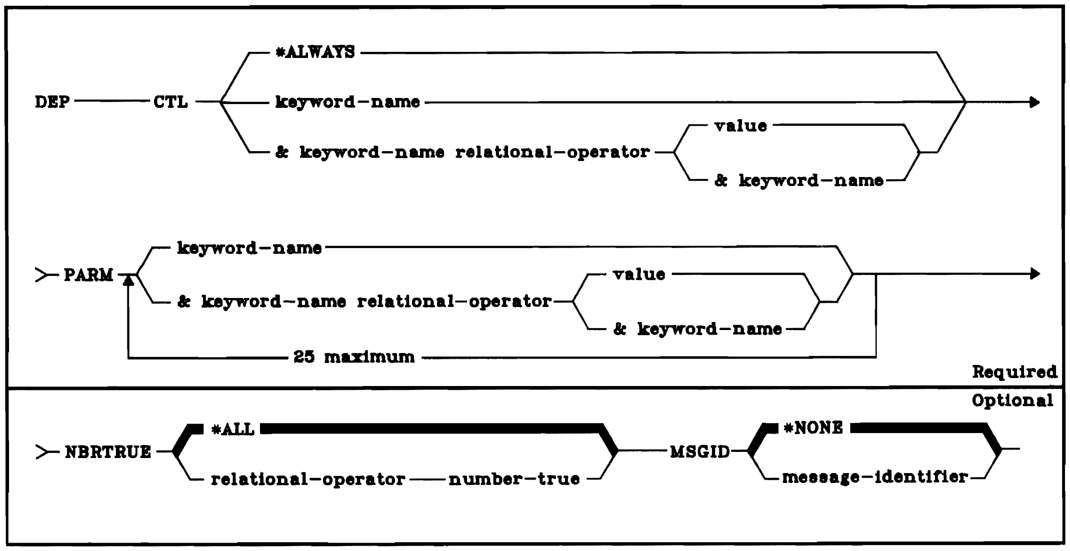
```
     PARM  KWD(DTAMBRS) TYPE(L1) DFT(*ALL) MAX(32) +
             SNGVAL(*ALL)
L1:  ELEM  TYPE(Q1) MIN(1)
     ELEM  TYPE(*NAME) MIN(0) MAX(32) SNGVAL(*NONE) +
             DFT(*NONE)
Q1:  QUAL  TYPE(*NAME) MIN(1)
     QUAL  TYPE(*NAME) DFT(*CURRENT) SPCVAL(*CURRENT)
```

The parameter named DTAMBRS is optional and, if not specified, defaults to *ALL. Otherwise, the value consists of a list, each element of which is itself a list. Each sublist consists of a qualified file name optionally followed by one or more member names. If no member name is specified, *NONE is taken as the default. If no library qualifier is specified for the physical file, *CURRENT is taken as the default, meaning the library is the one currently indicated by the qualified physical file name saved in the description of the logical file (to which this DTAMBRS parameter applies). Each sublist can contain one file name and up to 32 member names. Up to 32 such sublists can appear as the value of DTAMBRS.

DEP (Dependent) Statement

The Dependent (DEP) statement defines a required relationship between parameters and parameter values that must be checked. This relationship can relate to the specific value of a parameter or parameters, or the required presence of parameters.

If a parameter has a default value and the parameter is not specified, the following occurs, depending on whether the DEP statement is performing a specification check or a relational check. If a *specification* check is made on an unspecified parameter (checking for the presence of a value for that parameter), the system assumes that no value was specified, and the default value is *not* used. If a *relational* check is made on an unspecified parameter, the default value is used as the parameter value in the relational check.



CTL Parameter: Specifies the controlling conditions that must be true before the parameter dependencies defined in the PARM statement need to be true. The first keyword specified identifies the controlling parameter. The controlling condition can be specified by a keyword name only, or by a keyword name and a test relationship that determines whether the controlling condition requires the presence of the parameters it is dependent upon. The relationship between the controlling parameter and a specified value can be tested to determine whether the condition specified is met. If it is, the parameters that the controlling parameter is dependent upon must meet the requirements specified in the PARM and NBRTRUE keywords.

**ALWAYS:* The parameter dependency is to be always checked, regardless of the form of the command.

keyword-name: Enter the keyword name of the parameter for which a value must be specified to control dependency. (The keyword name is the name of the parameter that was specified by the KWD parameter on the PARM statement defining it.) If the keyword was specified, the parameter dependency will be checked. The keyword must not be defined as TYPE(*NULL).

(&keyword-name relational-operator value): Enter the keyword name of the controlling parameter followed by a relational operator (such as *LE or *EQ) and a value to be tested for. If the condition tested for is met, the parameters that the controlling parameter is dependent upon must meet the requirements specified for the PARM keyword. The value must be no longer than 32 characters. The keyword must not be defined as TYPE(*NULL).

The keyword name must be preceded by an ampersand (&) to indicate that the value of the keyword is to be tested if the relational operator and value are specified; the ampersand must not be used if they are not specified.

(&keyword-name relational-operator &keyword-name): Enter the keyword name of the controlling parameter followed by a relational operator (such as *EQ) and the keyword name of another parameter whose value is to be compared with the value of the controlling parameter. The keywords must not be defined as TYPE(*NULL) or with PASSVAL(*NULL) specified.

PARM Parameter: Specifies the parameter dependencies that must be tested if the controlling conditions defined by CTL are true. The dependencies can be the names of one or more parameters that are to be tested for their presence and/or one or more test relationships of keyword values to other keyword values or constant values. A maximum of 25 parameters can be specified for the PARM parameter. Keywords specified in this parameter must not be defined as TYPE(*NULL).

keyword-name: Enter the keyword name of each parameter that must have a value specified for it.

(&keyword-name relational-operator value): Enter the keyword name of each parameter followed by a relational operator and a value to be tested for. (An ampersand must precede the keyword name to indicate that the value of the keyword is to be tested.) The value must be no longer than 32 characters.

(&keyword-name relational-operator &keyword-name): Enter the keyword name of one parameter followed by a relational operator and the keyword name of another parameter whose value is to be compared with the value of the first parameter. A keyword defined with PASSVAL(*NULL) may not be specified.

NBRTRUE Parameter: Specifies the number of parameter dependencies defined in the associated PARM statement that must be true if the control condition is satisfied.

*ALL: All the parameter dependencies specified in the PARM statement must be true to satisfy the control condition.

(relational-operator number-true): Enter a relational operator and a number that specifies the number of parameter dependencies that must be true to satisfy the relation specified here. For example, if (*EQ 2) is specified for NBRTRUE, then two, and only two, of the parameter dependencies must be true for the PARM statement.

MSGID Parameter: Specifies the message identifier of an error message in a message file that is to be sent to the user if all the parameter dependencies that are specified have not been satisfied.

*NONE: No special message is to be sent; instead, a message generated by the command analyzer is to be sent to the user.

message-identifier: Enter the message identifier of the error message to be sent to the user if all the dependencies for this statement are not met. Messages whose identifiers begin with the 3-character prefix of *CPF* or *CPD* are retrieved from the IBM-supplied message file QCPFMSG. All other messages specified here are retrieved from the message file identified by the MSGF parameter on the CRTCMD command used to create the command being defined with these dependencies.

Examples

```
DEP  CTL(&TYPE *EQ LIST)  PARM(ELEMLIST)
```

If TYPE(LIST) is specified, the ELEMLIST parameter must be specified.

```
DEP  CTL(FILE)  PARM(VOL LABEL)  NBRTRUE(*EQ 2)
```

If the FILE parameter is specified, both the VOL and LABEL parameters must be specified.

```
DEP  CTL(GLOOP)  PARM(J1 D J2)  NBRTRUE(*EQ 1)
```

If the GLOOP parameter is specified, a value must be specified for one (and only one) of the J1, D, and J2 parameters.

Part 3. Appendixes

Part 3 contains the following:

- Appendix A, *Expanded Parameter Descriptions*, contains the expanded descriptions of parameters that are common to several commands and that contain additional information not essential in the command descriptions themselves.
- Appendix B, *Expressions*, contains a description of expressions and how they can be used in the control language.
- Appendix C, *User Profile Matrix Chart*, contains a description of the IBM-supplied user profiles and a chart showing all the CL commands and to which of the user profiles each command is initially assigned when the system is shipped.
- Appendix D, *Files Used by CL Commands*, contains a list of the IBM-supplied data base files and device files, shipped with the system, that are used by CL commands.
- Appendix E, *Error Messages That Can Be Monitored*, lists, by command, all the escape, notify, and status messages associated with each command.
- Appendix F, *Command and Keyword Abbreviations*, contains a list of all the abbreviations used in the command names and parameter keywords.
- The *Glossary of Terms and Abbreviations* defines the terms and abbreviations used in this manual.

Appendix A. Expanded Parameter Descriptions

This appendix contains the expanded descriptions of some of the parameters commonly used in the CL commands. The parameters that are included in this appendix meet one or both of these conditions:

- They have extensive information about how they are used (such as the LOC parameter on diskette commands).
- They are used in many of the CL commands (such as the PUBAUT parameter), and the description of the parameter in the individual commands can be written in a more concise form, giving only the essential information.

The expanded descriptions of the applicable command parameters have been placed in this appendix to:

- Reduce the amount of material needed in the individual commands, which is normally not needed by the programmer who is familiar with the parameter's main function.
- Provide a place where additional information can be supplied that would be useful to the programmer in some instances.

The format for this appendix is designed for easy reference.

- The common parameters are organized in alphabetic order by their keywords.
- Marginal keys that identify the first new parameter on the page are placed in the outer margin of each page.
- A general description of each parameter is given that explains its function, states the rules for its use, and provides other helpful information.
- The values that can be specified for each parameter are listed. Each value contains an explanation of what it means and possibly in which commands it is used. (Not all of the values will appear in every command. Refer to the individual commands for the specific use of the value in that command parameter.)

CLS Parameter

The class (CLS) parameter identifies the attributes that define the execution time environment of a routing step. The following attributes are defined in each class:

- Execution priority: A number that specifies the priority level assigned to all routing steps executed using the class. The priority level is used to determine which routing step, of all the routing steps competing for system resources, is to be executed next. The value can be 1 through 99, where 1 is the highest priority (all routing steps having a 1 priority are executed first).

- Time slice: The maximum amount of processor time that the system will allow the routing step to execute when it is allowed to begin. The time slice indicates the amount of time needed by the routing step to accomplish a meaningful amount of work. (The time used by the system for accessing auxiliary storage is not charged against the time slice.) When the time slice expires, the routing step waits while other routing steps of the same or higher priority are allowed to execute (up to the time specified in their time slices); then the routing step is given another time slice.

- Purge value: Indicates whether the routing step is to be made eligible to be moved out of main storage to auxiliary storage when the routing step is waiting for some resource before it can continue, or when its time slice has been used up (and higher priority routing steps are waiting).

- Default wait time: The default amount of time that the system is to wait for the completion of a System/38 instruction that performs a wait. (This wait time applies to times when an instruction is waiting for a system action, not to the time an instruction is waiting for a response from a user. Normally, this would be the amount of time the system user would be willing to wait for the system before canceling the request.) If the wait time is exceeded, an error message is passed to the job. This default wait time applies only when a wait time is not specified in the CL command that causes the wait.

  The wait time used for allocating file resources is specified in the file description and can be overridden by an override command. It can specify that the wait time specified in the class object is to be used. If file resources are not available when the file is opened, the system waits for them until the wait time expires.

- Maximum CPU time: The maximum amount of processor time (time slice period multiplied by number of time slices) allowed for a routing step to complete execution. If the routing step is not completed in this amount of time, it is terminated, and a message is written to the job log.

- Maximum temporary storage: The maximum amount of temporary storage that can be used by a routing step. This temporary storage is used for the programs that execute in the routing step, for the system objects used to support the routing step, and for temporary objects created by the routing step.

The system is shipped with a set of classes that define the attributes for several job execution environments. Other classes can be created by the CRTCLS command; any class can be displayed or deleted by the respective DSPCLS and DLTCLS commands.

The following classes (by name) are supplied with the system:

QBATCH.QGPL	For use by batch jobs
QCTL.QSYS	For use by the controlling subsystem
QINTER.QGPL	For use by interactive jobs
QPGMR.QGPL	For use by programming subsystem
QSPL.QGPL	For use by the spooling subsystem printer writer
QSPL2.QGPL	For general spooling use in base system pool

Values Allowed

qualified-class-name: Enter the name of the class, optionally qualified by the name of the library in which the class is stored. If the class name is not qualified and the CLS parameter is in the CRTCLS command, the class object is stored in QGPL; otherwise, the library list (*LIBL) is used to find the class name.

EXCHTYPE Parameter

The Exchange Type (EXCHTYPE) parameter specifies which one of the three diskette exchange types (basic, H, or I) is to be used when the system is writing diskette files. The exchange type is stored in the volume label area on the diskette. There is one label for each data file on the diskette.

The exchange type to be used can be specified in one of the diskette device file commands (CRTDKTF, CHGDKTF, or OVRDKTF), or it can be passed as a parameter when the device file is opened by the high-level language (HLL) program (if supported).

The exchange type specified in the diskette device file or HLL program is not used by the system when processing a diskette input file. Instead, the system uses the exchange type from the file label on the diskette.

It is possible for one diskette to contain both basic and I exchange format files or H and I format files, but it is not possible for one diskette to contain both basic and H format files.

Values Allowed

One of the following values can be specified for the EXCHTYPE parameter, depending on the diskette type (1, 2, or 2D) and the diskette sector size (128, 256, 512, or 1024):

******STD:* The exchange type is determined by the system, based on the diskette type and sector size. A basic exchange type is used if the diskette type is 1 or 2, and the diskette sector size is 128 bytes. A H exchange type will be used if the diskette type is 2D and the diskette sector size is 256 bytes. *STD is not valid for any other combinations of type and sector size.

**Basic:* The basic exchange type will be used. The diskette type must be 1 or 2, and the diskette sector size must be 128 bytes.

**H:* The H exchange type will be used. The diskette type must be 2D, and the diskette sector size must be 256 bytes.

**I:* The I exchange type will be used. The diskette type and sector size may be any of the following:

Diskette Type	Sector Size
1	128, 256, or 512
2	128, 256, or 512
2D	256, 512, 1024

FILETYPE Parameter

The FILETYPE parameter specifies (1) whether the device file description describes data or source records, or (2) whether each member of a data base file being created is to contain data records, or source records (statements). The file could contain, for example, RPG source statements for an RPG program or data description source statements for another device or data base file.

Note: If you are creating a source type *physical* data base file and are not providing field-level descriptions of the file (via DDS), you can use either the CRTPF or the CRTSRCPF command. However, the CRTSRCPF command is usually more convenient and efficient, because it is designed to be used to create source physical files. If DDS is to be provided when you are creating a source type data base file, you should use the CRTPF or CRTLF command, which both have the SRCFILE and SRCMBR parameters for specifying source input.

Records in a source file must have at least three fields: the first two are the source sequence number field and the date field; the third field contains the source statement. These three fields are automatically provided by CPF when a source file is created for which no DDS is provided. (Additional source fields can be defined in data description specifications.) The length of the sequence number field must be six zoned digits with two decimal places. The length of the date field must be six zoned digits with no decimal places.

The sequence number and date field are added to the source record when:

- The records are read into the system.
- The records are created by the Source Entry Utility (which is part of the Interactive Data Base Utilities licensed program).

The fields are added when an inline data file (specified as the standard source file format) is read from the card or diskette device or when an inline data file is read from a data base file that is not specified as a source file. The spooling reader places a sequence number in the sequence number field and zeros in the date field.

If the fields already exist in records read from the card or diskette device, the fields are not changed. If the records in a data base file are in source format and are being read as an inline data file in data format, the sequence number and date fields are removed.

For more information about data and source files, see the *CPF Programmer's Guide.*

FILETYPE
Parameter

Values Allowed

**DATA:* The file being created is to contain, or describe, data records.

**SRC:* The file being created is to contain, or describe, source records. If the source file is a data base file, the file must have a keyed sequence access path. The six-digit source sequence number field is used as the key field.

FRCRATIO Parameter

The force write ratio (FRCRATIO) parameter specifies the maximum number of records that can be inserted or updated before they are forced into auxiliary (permanent) storage. The force ratio ensures that all new or changed records are written into auxiliary storage at least as often as this parameter specifies. In the event of system failure, the only records that might be lost would be those that were entered or updated since the last force write operation occurred.

The force ratio is applied to all records inserted or updated in the file through the open data path (ODP) to which the force ratio applies. If two programs are sharing the file, SHARE(*YES), the force ratio is not applied separately to the set of records inserted or updated by each program, but is applied to any combination of records (from both programs) that equals the specified force ratio parameter value. For example, if a force ratio of 5 was specified for the file, any combination of five records from the two programs (such as four from one program and one from the other) forces the records to be written to auxiliary storage. If two or more programs are using the file through separate ODPs, a separate force ratio can be specified for each ODP. The updates and insertions from each program are accumulated individually for each ODP.

Each data base file can have a force ratio assigned to it. Logical files, which can access data from more than one physical file, can specify a more restrictive force ratio (a smaller number of records) than that specified for the based-on physical files. However, a logical file cannot specify a less restrictive force ratio. If a logical file specifies a less restrictive force ratio than that specified for any of the physical files, the most restrictive force ratio from the physical files is used for the logical file. For example, if the force ratios of three physical files are 2, 6, and 8, the force ratio of a logical file based on these physical files cannot be greater than 2. If no force ratio is specified for the logical file, 2 is assumed. Thus, each time a program updates two records in the logical file (regardless of which physical files are affected), those records are forced into auxiliary storage.

Access paths associated with the updated or inserted records are written to auxiliary storage only when all the records covered by the access path have been written to auxiliary storage. If only one ODP exists for the file, the access path is forced to auxiliary storage whenever a forced write occurs. If two or more ODPs to the file exist, the access path is written to auxiliary storage whenever all the updated records for all the ODPs have been forced.

Note: These rules apply only when a force ratio of 2 or higher is specified. When a force ratio of 1 is specified, the access path is not written to auxiliary storage until all the ODPs have been closed.

Values Allowed

**NONE:* There is no specified ratio; the system determines when the records are written in auxiliary storage.

number-of-records-before-force: Enter the number of new or changed records that are processed before they are explicitly forced into auxiliary storage.

Operations Using Generic Functions

Certain commands allow you to perform operations on objects identified by *generic* names. A generic name is one that contains a series of initial characters that are common to a group of objects, followed by an asterisk. (The asterisk identifies the series of common characters as a generic name; otherwise, the system interprets the series of characters as a name of a specific object).

When you specify a generic name, the system performs the required function on all objects whose names begin with the specified series of characters. You must have the authority required to perform that function on every object the generic name identifies; if you do not, the function is not performed for that object and a diagnostic message is issued for each instance in which the generic function was attempted but failed. A completion message is issued for each object the generic function has operated on successfully (CF7 must be used to view the completion messages). Once the entire generic function is completed, a completion message is issued that states that all objects were operated on successfully, or an escape message is issued that states that one or more objects could not be successfully operated on.

The library qualifier specified with the generic name limits the scope of the operation. For example, a CHGPRTF command with FILE(PRT*.LIB1) will perform the desired operation on printer files beginning with PRT in library LIB1 only; printer files in other libraries are not affected. The limitations associated with the various library qualifiers are as follows:

- library-name: The operation will be performed on generic object names in the specified library only.
- *LIBL: The operation will be performed on generic object names in the library list associated with the job that requested the generic operation.
- *ALL: The operation will be performed on generic object names in all libraries on the system for which you are authorized.
- *USRLIBL: The operation will be performed on generic object names in the user part of the library list (*LIBL) for the job.
- *ALLUSR: The operation will be performed on generic object names in all user libraries for which you are authorized, including QGPL.

JOB Parameter

The JOB parameter specifies the name of the job to which the command is to be applied. The job name identifies all types of jobs on the system. Each job is identified by a qualified job name, which has the following format:

job-name.user-name.job-number

Note that, although the syntax is similar, job names are qualified in a manner different from CPF object names.

- Job name: The job name can contain a maximum of 10 alphameric characters, of which the first must be alphabetic. The name can come from one of three sources, depending on the type of job:
  - Batch job: The job name is specified on the JOB or SBMJOB commands or, if not specified there, the unqualified name of the job description is assumed.
  - Interactive job: The job name is the same as the name of the device (work station) from which the sign-on was performed.
  - Autostart job: The job name is provided in the autostart job entry in the subsystem description under which the job runs. (The job name was specified in the ADDAJE command.)

- User name: The user name identifies the user who submits the job and the user profile under which the job is to be executed. The user name is the same as the name of the user profile and contains a maximum of 10 alphameric characters. The name can come from one of several sources, again, depending on the type of job:
  - Batch job: The user name is specified on the SBMJOB command, or it is specified in the job description referenced by the JOB or SBMJOB commands.
  - Interactive job: The user name is associated with the user password entered at sign-on, or the user name is provided from the default in the job description referenced by the work station's job entry.
  - Autostart job: The user name is specified in the job description referenced by the job entry for the autostart job.

- Job number: The job number is a unique six-digit number that is assigned to each job by the system. The job number provides a unique qualifier if the job name is not otherwise unique. The job number can be determined by means of the DSPSBS (Display Subsystem) command or the DSPJOB (Display Job) command. If specified, the job number must have exactly six digits.

Commands only require that the simple name be used to identify the job. However, additional qualification must be used if the simple job name is not unique.

Duplicate Job Names

If a duplicate job name is specified in an *interactive* job, the system displays all of the duplicates of the specified job name to the user in qualified form. The job names are displayed in qualified form along with the user name and job number so that the user can further identify the job that is to be specified in a command. The correct qualified job name can then be entered.

If a duplicate job name is used in a command that is in a *batch* job, the command is not executed. Instead, an error message is written to the job's job log.

Values Allowed

The JOB parameter can have one or more of the following values, depending upon the command:

**:* The job is the one in which the command is entered; that is, the command with JOB(*) specified on it.

**JOBD:* The simple job name is the unqualified name of the job description.

**NONE:* No job name is specified (DSPLOG command).

job-name: A simple job name is to be specified.

qualified-job-name: A qualified job name is to be specified. If no job qualifier (user name and job number) is given, all of the jobs currently in the system are searched for the job name. If duplicates of the specified name are found, a qualified job name must be specified.

LABEL Parameter

The LABEL parameter specifies the data file identifier of the data file on diskette or tape that is to be used in input/output operations. The data file can be in either the basic exchange format or the save/restore format. Note, however, that the device file commands are used for diskettes and tapes that are in the basic exchange format only, *not* in the save/restore format; user-defined device files are not used in save/restore operations. Each data file that is on a diskette or tape has its data file identifier stored in its own file label.

On diskette, all of the data file labels are in one place—in the volume label area of that diskette. In addition to the data file identifier, each label contains other information about the file, such as where the file is stored on the diskette (track and sector) and whether the file continues on another diskette (in the case of a multivolume data file).

On tape, the data file label (or header label) of each data file is stored on the tape just before the data in the file. (That is, each file on the tape has its own header label and its own data records together as a unit, and one file follows another.) In addition to the data file identifier, each label also contains other information about the file, such as the file sequence number, record and block attributes, and whether it is a multivolume data file.

Generally, the data file identifier is an alphameric character string that contains no more than 8 characters. However, the maximum length actually depends on several things: what data format is used for the files, whether the files are on diskette or on tape, and which CL commands the identifiers are specified in. (Any of the unused positions in the identifier field are left blank.)

The first character of the data file identifier must be alphabetic (A-Z, $, #, or @) and the rest of the characters *should* be alphameric (A-Z, 0-9, $, #, @, and _). Special characters can be used if the identifier is enclosed in apostrophes. However, if the diskette or tape is to be used on a system other than System/38, the requirements for specifying identifiers on that system must be considered.

Diskette and Tape Data File Identifiers

For *diskettes* in the basic exchange format, the data file identifier cannot exceed 8 characters (the same limit that RPG file names have). This limitation applies to the following commands: CRTDKTF, CHGDKTF, CHGSPLFA, OVRDKTF, DLTDKTLBL, DSPDKT, and STRDKTRDR.

For *tapes*, the identifier can have as many as 17 characters. However, if a tape is to be used on a system other than System/38, a maximum of 8 characters, or a qualified identifier of no more than 17 characters, should be used. If more than 8 characters are used, the identifier should be qualified and enclosed in apostrophes so that no more than 8 characters occur in either part, and the parts are separated by a period; for example, LABEL('TAXES.JAN1980'). This limitation applies to the following commands: CRTTAPF, CHGTAPF, OVRTAPF, and DSPTAP.

Duplicate data file identifiers are not allowed on the same volume, for either diskette or tape. However, the identifier can be the same as the name of the data base file being written to diskette or tape, if the file name contains no more than 8 characters. (The diskette and tape data files contain only data, not file descriptions like those of data base files.) On diskette, the identifiers ERRORSET and SYSAREA cannot be used; they are reserved for special use.

The data file identifier is put on the volume when the data file is put on the volume. For input/output operations, the identifier can be specified in one of the diskette or tape device file commands, or it can be passed as a parameter when the device file is opened by the HLL program that uses the file.

Save/Restore Format

For *diskettes* in the save/restore format, the data file identifier can have a maximum of 15 characters. The identifier consists of a library name of up to 10 characters followed by a period, a Q, and a 3-digit sequence number; for example, LABEL('PAYLIB.Q014'). The 15-character limit applies to identifiers of save/restore data files displayed by the DSPDKT command.

For *tapes* in the save/restore format, the identifier cannot exceed 10 characters. Only the library name is used in the identifier. This limitation applies to identifiers of save/restore data files displayed by the DSPTAP command.

Values Allowed

One of the following values can be specified for the LABEL parameter, depending upon the command.

**ALL:* All the data file identifiers on the specified diskette or tape volumes are to have their labels displayed.

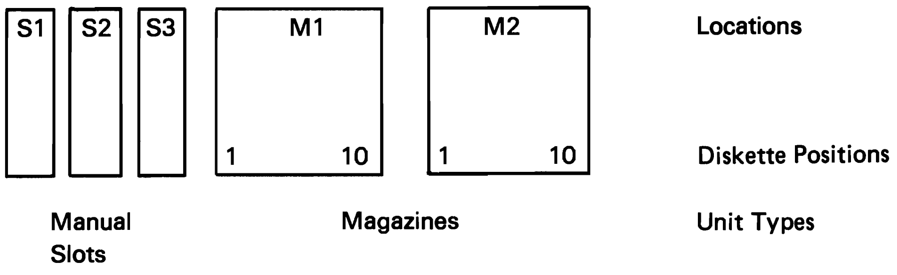
**NONE:* The data file identifier is not specified. It must be supplied before the device file (and/or data base file) is opened to be used in the diskette or tape operation.

**SAME:* The data file identifier that is already present in the diskette or tape device file is not to be changed.

data-file-identifier: Enter the identifier of the data file on diskette or tape to be used with the device file description, or to be displayed.

LOC Parameter

The location (LOC) parameter specifies which diskette locations in the magazines or slots of the diskette magazine drive are to be used in diskette input/output operations. Only one of the two types of units, either magazines or manual slots, can be used in any diskette I/O operation. One or more of the diskette locations in either unit type can be involved, depending on the values specified in this parameter.



Note: In the LOC parameter, the tenth diskette position is specified as 10. However, when a diskette is initialized in the save/restore format in position 10 of a magazine, the system assigns a single digit of 0 as the last character in that diskette's volume identifier (stored on the diskette).

For data that is in the basic data exchange format, this parameter has a list of three elements (three values) that specify which diskettes are involved in the operation: (1) the unit location (that is, the magazines or slots used); (2) the starting diskette position; and (3) the ending diskette position. For data in the save/restore format, only the unit location and starting position are specified in this parameter.

If the operation requires that additional magazines or diskettes be loaded, a message is issued to the operator to mount them. If LOC is not specified, *M12 is assumed (except for the LODPGMCHG command).

If more than one unit is specified in the LOC parameter, the volumes must be mounted in the specified units in the same order as the volume identifiers are specified in the VOL parameter. If VOL is not specified, the system begins the operation with the first unit specified.

Values Allowed

Unit Type and Location: One of the following values can be entered as the first element (in the list of three elements) to specify which unit and location is to be used in the diskette read or write operation. The first column contains the values that are coded to indicate which units, in the second column, are to be used.

Value	Location Used
*M12	Both magazines 1 and 2
*M1	Magazine 1 only
*M2	Magazine 2 only
*S1	Manual slot 1 only
*S2	Manual slot 2 only
*S3	Manual slot 3 only
*S12	Manual slots 1 and 2
*S23	Manual slots 2 and 3
*S123	Manual slots 1, 2, and 3

The second and third elements specify which of the diskettes are the starting and ending diskettes.

Starting Diskette Position: One of the following values can be entered as the second element to specify which diskette position (in the location specified by the first element) the operation is to start with. The starting position contains the diskette that has the beginning of the data file on it.

Notes:
1. For read operations involving multiple diskettes (specified by the first element), the *first* diskette specified or assumed by the second element must contain the desired data file (or saved object) because only the first diskette is searched. The only time the other diskettes in the location are searched is when *SEARCH is specified as the second element on save/restore commands.
2. For magazines, the starting diskette position can be specified for the first magazine only. For all following magazines, the operation always begins with the leftmost diskette position of the units specified. (The SAVSYS command *always* begins with the leftmost position.)

**FIRST:* The first diskette position in the location specified by the first element contains the diskette to begin the operation with. If *FIRST is used for the second element and the first element is:

*M12	Start at diskette 1 of magazine 1.
*M1	Start at diskette 1 of magazine 1.
*M2	Start at diskette 1 of magazine 2.
*S1	Start at slot 1.
*S2	Start at slot 2.
*S3	Start at slot 3.
*S12	Start at slot 1.
*S23	Start at slot 2.
*S123	Start at slot 1.

**CURRENT:* The diskette in the location at which the diskette magazine drive is currently positioned is to be used. (This is normally where the previous operation has just ended.) If the currently selected diskette is not anywhere in the location specified by the first value, a diagnostic message is sent to the system operator, and the command is terminated.

**SEARCH:* The diskettes in the location specified by the first value are to be searched for the first diskette having cleared space (on save commands) or the specified objects (on restore commands), where the operation can then begin. The search starts with the leftmost diskette in the specified location and ends with the first diskette having the desired item.

starting-diskette-position: Enter the number of the diskette position (within the range specified by the first element) that contains the first diskette to be used. If both magazines (*M12) are being used, the starting diskette position is in magazine 1. For the manual slots, this number may be 1 through 3, depending on what is specified by the first element. For a magazine, this number may be 1 through 10.

Ending Diskette Position: The third element in the list specifies (1) the last diskette to be used and (2) what action is to occur if the end of the volume on that diskette is reached before the end of the data file is reached. One of the following values can be entered as the third element to specify the ending diskette to be used in the operation.

Note: An ending diskette position cannot be specified within a unit on the save or restore commands; only the units and starting position can be specified.

**LAST:* The last diskette position in the location specified by the first element contains the last diskette to be used. If *LAST is used for the third element, and the first element value is:

*M12	End with diskette 10 of magazine 1.
*M1	End with diskette 10 of magazine 1.
*M2	End with diskette 10 of magazine 2.
*S1	End with slot 1.
*S2	End with slot 2.
*S3	End with slot 1.
*S12	End with slot 2.
*S23	End with slot 3.
*S123	End with slot 3.

If the end of the volume (of the last diskette) is reached before the end of the data file, an error message is sent to the system operator.

**WRAP:* If the end of the last diskette in the location is reached before the end of the data file is reached, a message is sent to the system operator requesting that another volume be mounted. The message sent depends on which location was specified by the first element, as shown in the following chart:

First List Element	**Meaning of *WRAP**
*M12	Mount a new pair of magazines and continue processing on diskette 1 of magazine 1.
*M1	Mount a new magazine for magazine 1 and continue processing on diskette 1.
*M2	Mount a new magazine for magazine 2 and continue processing on diskette 1.
*S1	Insert a new diskette in slot 1 and continue processing.
*S2	Insert a new diskette in slot 2 and continue processing.
*S3	Insert a new diskette in slot 3 and continue processing.
*S12	Insert new diskettes in slots 1 and 2 and continue processing at slot 1.
*S23	Insert new diskettes in slots 2 and 3 and continue processing at slot 2.
*S123	Insert new diskettes in slots 1, 2, and 3 and continue processing at slot 1.

When *WRAP is specified, the locations specified by the first element in the LOC parameter must all contain diskettes from the specified starting diskette position to the last possible diskette in the location(s). When the last diskette has been used and new diskettes have been mounted, then the wraparound continues on the *leftmost* diskette position in the location(s), *not* at the starting position specified by the second element.

**ONLY:* Only the diskette position specified by the second element in the list is to be used, and used only once; a second diskette cannot be mounted in that position to continue the operation.

ending-diskette-position: Enter the number of the diskette position (within the range specified by the first element) that contains the last diskette to be used. If both magazines (*M12) are being used, the ending diskette position is in magazine 2. For the manual slots, this number may be 1 through 3, depending on what is specified by the first element. For a magazine, this number may be 1 through 10.

MAXACT Parameter

The maximum activity level (MAXACT) parameter specifies the maximum number of jobs that can be concurrently initiated and active through a job queue entry, routing entry, or work station entry. A job is considered active from the time execution is initiated until execution is terminated. This includes time when:

- The job is actually using the processor to execute.
- The job is waiting for a response from a work station user.
- The job is started and available for processing but is not actually using the processor. (For example, it might have used up its time slice and is waiting for another time slice.)
- The job is started but is not available for processing. (For example, it could be waiting for a message to arrive on its message queue.)

Values Allowed

**NOMAX:* There is no limit to the number of jobs that can be concurrently active through this entry.

maximum-active-jobs: Enter a value that specifies the maximum number of jobs that can be concurrently active through this entry.

For a description of activity level controls, refer to the *CPF Programmer's Guide.*

OBJ Parameter

The object (OBJ) parameter specifies the names of one or more objects that are to be affected by the command in which this parameter is used. All the objects must be in the library specified in the LIB parameter, SAVLIB parameter, or the library qualifier in the OBJ parameter, depending upon which command is used.

On some commands, the generic name of a group of objects can be specified. (To form a generic name, add an asterisk after the last character in the common group of characters; for example, ABC*. If an * is not included with the name, the system assumes that the name is a complete object name.)

Values Allowed

Depending on the command, the following types of values can be specified on the OBJ parameter:

- *ALL
- Simple object name
- Qualified object name
- Generic object name
- Qualified generic object name

OBJTYPE Parameter

The object type (OBJTYPE) parameter specifies the types of CPF objects that can be operated on by the command in which they are specified. The object type(s) that can be specified in the OBJTYPE parameter varies by command.

The predefined values for the 23 CPF object types are listed below. When an object is created and a library qualifier can be specified and is not, the object is stored in the QGPL library, as shown in the last column. The other objects, identified by N/A in the last column, cannot be stored in user-provided libraries.

Value	Object Type	Default User Library
*CLS	Class	QGPL
*CMD	Command	QGPL
*CUD	Control unit description	N/A
*DEVD	Device description	N/A
*DTAARA	Data area	QGPL
*EDTD	Edit description	N/A
*FCT	Forms control table	QGPL
*FILE	File	QGPL
*JOBD	Job description	QGPL
*JOBQ	Job queue	QGPL
*JRN	Journal	QGPL
*JRNRCV	Journal receiver	QGPL
*LIB	Library	N/A
*LIND	Line description	N/A
*MSGF	Message file	QGPL
*MSGQ	Message queue	QGPL
*OUTQ	Output queue	QGPL
*PGM	Program	QGPL
*PRTIMG	Print image	QGPL
*SBSD	Subsystem description	QGPL
*SSND	Session description	QGPL
*TBL	Table	QGPL
*USRPRF	User profile	N/A

OBJTYPE Parameter

The following chart shows all the commands, other than the create, delete, change, and display commands, that operate on the objects.

	Commands							
Value	**DSPOBJ**	**SAVOBJ**	**SAVCHGOBJ**	**RSTOBJ**	**MOVOBJ**	**RNMOBJ**	**DMPOBJ**	**DMPSYSOBJ**
*CLS	X	X	X	X	X	X	X	X
*CMD	X	X	X	X	X	X	X	X
*CUD	X						X	X
*DEVD	X					X	X	
*DTAARA	X	X	X	X	X	X	X	X
*EDTD	X	X	X	X		X	X	X
*FCT	X	X	X	X	X	X	X	X
*FILE	X	X	X	X	X	X	X	X
*JOBD	X	X	X	X	X	X	X	X
*JOBQ	X				X	X	X	X
*JRN	X	X	X	X			X	X
*JRNRCV	X	X	X	X			X	X
*LIB	X					X	X	X
*LIND	X						X	X
*MSGF	X	X	X	X	X	X	X	X
*MSGQ	X				X	X	X	X
*OUTQ	X				X	X	X	X
*PGM	X	X	X	X	X	X	X	X
*PRTIMG	X	X	X	X	X	X	X	X
*SBSD	X	X	X	X	X	X	X	X
*SSND	X	X	X	X	X	X	X	X
*TBL	X	X	X	X	X	X	X	X
*USRPRF	X						X	X

Values Allowed

**ALL:* All the object types that are allowed in the command and that are specified by name and are in the specified library are to be operated on by the command in which they are specified. *ALL refers only to the object types that apply to that command; refer to the individual command descriptions of the OBJTYPE parameter to see which of the CPF object types can be specified.

object-type: Enter the predefined values for the types of objects that are to be operated on by the command.

OUTPUT Parameter

The OUTPUT parameter specifies whether the output from the display command is to be displayed or printed. Basically, the same information is provided in either form; only the format is changed as necessary to present the information in the best format for the device. (For example, because there are more lines on a printed page than on a display, column headings are not repeated as often in printed output.)

If the output is to be displayed, it is sent to the work station that issued the command. It is displayed in the format specified in the display device file used by that display command. A different device file is used for the output of each display command, and the file is different for displayed or printed output. In most cases, the name of the command is part of the file names of either type of device file. Refer to Appendix D for the names of the display device files and the spooled printer files used by each command.

If the output is to be printed, it is spooled and an entry is placed on the job's output queue. The output can be printed on a system printer or on a 5224/5225/5256 Printer, depending on which device is specified in the Start Printer Writer (STRPRTWTR) command.

Note: Although the IBM-supplied printer files are shipped with SPOOL(*YES) specified, they can be changed to SPOOL(*NO) by the OVRPRTF and CHGPRTF commands.

If the OUTPUT parameter is not specified in the display command, the default value of * is assumed. The output resulting from this value depends on the type of job that entered the command. The following chart shows how the output is produced for interactive and batch jobs.

Output	Interactive Job	Batch Job
*	Displayed	Printed
*LIST	Printed	Printed

For an explanation of the displayed data that is produced, refer to the *Additional Considerations* section of the appropriate display command description.

Values Allowed

**:* The output is to be displayed (if requested by an interactive job) or spooled for printing after job completion (if requested by a batch job).

**LIST:* The output is to be spooled for printing after job completion.

**NONE:* The only output is to be to the specified data base file. (This value is used only by the file reference commands DSPDBR and DSPFFD.)

PUBAUT Parameter

The public authority (PUBAUT) parameter is used only in create commands; it specifies what authority the public (that is, all users that have access to the system) can have for the object being created. This is the initial authority that all users have in common for the created object. The owner of the object initially has all rights to the object.

The owner can create the object giving all users (public) complete authority, no authority, or limited authority for its use.

- Complete authority (*ALL): The object is a *public* object, meaning that *all* applicable object and data rights are available to every user on the system.
- No authority (*NONE): The object is a *private* object, meaning that *no* object rights or data rights are available to all users.
- Limited authority (*NORMAL): The object can be used by all users for *normal* operations only. The users are limited to operational rights and data rights (read, add, delete, and update as defined by *NORMAL for each object type). The users do not have any object control rights that affect the object's existence or description.

Note: Refer to the GRTOBJAUT (Grant Object Authority) command for a description of all of the object control rights and data rights. However, not all rights apply to all object types.

If the object is created as a private object or with limited authority given to all users, the owner can specifically give more, or complete, authority to specific users by naming the users and the rights in the GRTOBJAUT command. He can also withdraw specific rights from specific users, or from all users (either publicly authorized and/or explicitly authorized), by using the RVKOBJAUT command.

Refer to the *CPF Programmer's Guide* for a more complete description of security and for the applicable rights of use by object type. Also, see the individual PUBAUT parameter descriptions in each create command for the specific rights that are granted when *NORMAL is specified. If PUBAUT is not specified when the create command is entered, *NORMAL is always assumed.

Values Allowed

**NORMAL:* *Normal* authority includes the authority necessary for a user to perform usual functions associated with a specific CPF object type. In most cases, this means that all users have only operational rights for the object being created, which allows all users to use the object and display its description. Some objects, such as data base files, also have data rights (add, delete, read, and update) as part of this authority.

**ALL:* All users have complete authority for the object being created, the same as its owner has: operational, object existence, and object management rights, plus the read, add, delete and update rights (data rights), as applicable.

**NONE:* No user (other than its owner) can use the object, unless specific rights to one or more users are granted in the GRTOBJAUT command; the rights specified are granted to the users whose names are also specified in the command.

Scheduling Priority Parameters (JOBPTY, OUTPTY, PTYLMT)

The scheduling priority parameters specify the priority values to be used by the system to determine the order in which the jobs and spooled files will be selected for processing. Each job is given a scheduling priority that is used for both job selection and spooled file output. The job scheduling priority is specified by the JOBPTY parameter in commands like the JOB, SBMJOB, CRTJOBD, and CHGJOBD commands. The priority for producing the spooled output from a job is specified by the OUTPTY parameter in the same commands.

In addition, because every job is processed under a specific user profile, the priority for jobs can be limited by the PTYLMT parameter specified on the CRTUSRPRF and CHGUSRPRF commands. This parameter value controls the maximum job scheduling priority and output priority that *any* job running under a user profile can have; that is, the priority specified in the JOBPTY and OUTPTY parameters of any job command cannot exceed the priority specified in the PTYLMT parameter for that user profile. The scheduling priority is used to determine the order in which jobs are selected for execution and is not related to the execution priority specified in the class object.

The three scheduling priority parameters specify the following:

- The PTYLMT parameter specifies the *highest* scheduling priority for *any* job that the user submits. In the commands that affect the user's user profile, the PTYLMT parameter specifies the highest priority that can be specified in another JOBPTY parameter on commands relating to each specific job. (The user can specify a lower priority for a job on the command used to submit the job. If a higher priority is specified for JOBPTY in the JOB or SBMJOB command than is specified for PTYLMT in the associated user profile, an error message is displayed and the maximum priority specified in PTYLMT is assumed. If a higher job priority is specified in the CHGJOB or CHGJOBD command, an error message is displayed and the job's attributes are not changed.)

- The JOBPTY parameter specifies the priority value to be used for a *specific* job being submitted. In the commands relating to a specific job being submitted, the JOBPTY parameter specifies the actual scheduling priority for the job.

- The OUTPTY parameter specifies the priority for producing the output from all spooled output files from the job. The priority value specified in the OUTPTY parameter determines the order in which spooled files are handled for output. The same value is applied to all the spooled files produced by the job.

The scheduling priority can have a value of 1 through 9, where 1 is the highest priority. Any job with a priority of 1 is scheduled for execution before all other jobs that are waiting and that have priorities of 2 through 9.

The three priority parameters can be specified in the following commands:

JOBPTY & OUTPTY	**PTYLMT**
CHGJOB	CHGUSRPRF
CHGJOBD	CRTUSRPRF
CRTJOBD	
JOB	
SBMJOB	

Values Allowed

Depending upon the command, one or more of the following values apply to the parameter.

5: If not specified in the CRTUSRPRF command, 5 is the default value that is assumed for the priority limit for the user profile. That would be the highest priority that the user could specify for any job he submits for execution. If not specified in the CRTJOBD command, 5 is the default value for both the job scheduling priority and the output priority.

**SAME:* The priority assigned, or the highest priority that can be assigned, remains the same.

**JOBD:* The scheduling priority for the job is to be obtained from the job description under which the job will run.

scheduling-priority: A priority value of 1 through 9 can be entered, where 1 is the highest and 9 is the lowest priority.

SEV Parameter

The severity (SEV) parameter specifies the severity code that:

- Describes the level of severity that is to be associated with a message
- Indicates the minimum severity level that causes a message to be sent to a user or program
- Causes a batch job to be canceled
- Causes execution of a command to be terminated if a syntax error of sufficient severity occurs

Note: The LOG parameter on some commands also uses these severity codes for logging purposes (to control which job activity messages and error messages are logged in the job log).

The severity code is a two-digit number that can be 00 through 99. The higher the value, the more severe or important is the condition. The severity code of a message that is sent to a user indicates the severity of the condition described by the message. More than one message can have the same severity code. If a severity code is not specified for a predefined message, it is assumed to be 00 (information only).

The user can specify a severity code for any message when it is defined by the ADDMSGD (Add Message Description) command. To change the severity code of a message, execute the CHGMSGD (Change Message Description) command.

IBM-defined severity codes are used in all of the IBM-supplied messages that are shipped with the system. These severity codes (which are included with all the message descriptions in the *System/38 Messages Guide: CPF, RPG III, IDU*) are listed below with their meanings.

00 – Information: A message of this severity is for information purposes only; no error was detected and no reply is needed. The message could indicate that a function is in progress or it has reached a successful completion.

10 – Warning: A message of this severity indicates a potential error condition. The program may have taken a default, such as supplying missing input. The results of the operation are assumed to be what was intended.

20 – Error: An error has been detected, but it is one for which automatic recovery procedures probably were applied; processing has continued. A default may have been taken to replace erroneous input. The results of the operation may not be valid. The function may have been only partially completed; for example, some items in a list may have been processed correctly while others may have failed.

30 – Severe Error: The error detected is too severe for automatic recovery, and no defaults are possible. If the error was in source data, the entire input record was skipped. If the error occurred during program execution, it will lead to an abnormal termination of the program (severity 40). The results of the operation are not valid.

40 – Abnormal Termination of Program or Function: The operation has terminated, possibly because it was unable to handle invalid data, or possibly because the user has canceled it.

50 – Abnormal Termination of Job: The job was terminated or was not started. A routing step may have terminated abnormally or failed to start, a job-level function may not have been performed as required, or the job may have been canceled.

60 – System Status: A message of this severity is issued only to the system operator. It gives either the status of or a warning about a device, a subsystem, or the system.

70 – Device Integrity: A message of this severity is issued only to the system operator. It indicates that a device is malfunctioning or in some way is no longer operational. The user may be able to recover from the failure, or the assistance of a service representative may be required.

80 – System Alert: A message of this severity is issued only to the system operator. It warns of a condition that, although not severe enough to stop the system now, could become more severe unless preventive measures are taken.

90 – System Integrity: A message of this severity is issued only to the system operator. It describes a condition that renders either a subsystem or the system inoperative.

99 – Action: A message of this severity indicates that some manual action is required, such as entering a reply, changing printer forms, or replacing diskettes.

SPLNBR Parameter

The spooled file number (SPLNBR) parameter is used when more than one spooled file is created by a job and the files all have the same name. The files are numbered, starting with 1, in the order that they are opened by the job. The job log is always the last file for a job.

A file number is generated at the time that each file is opened within a job (when output records are produced) and can be used by the system as long as the job and/or the files are on the system. If the files are not uniquely named because they were opened more than once, this file number is used to specify which file (or group of records, if the complete file has not been produced yet) is to be acted upon by a CL command.

TEXT Parameter

The TEXT parameter specifies the user-defined description that briefly describes the object being created or changed. The description can include up to 50 characters; if it is a quoted string (that is, enclosed in apostrophes), any of the 256 EBCDIC characters can be used. The apostrophes are not required if the string does not contain any blanks or other special characters. Any of the 50 character positions not filled by the specified description are padded with blanks.

The description is used to describe any of the CPF objects when the named object is displayed by the DSPOBJD command. (A user can display only those objects for which he has operational authority.) Refer to the OBJTYPE parameter here in Appendix A for a list of the CPF object types.

Values Allowed

Depending upon the command, one or more of the following values apply to the TEXT parameter.

**BLANK:* The user description of the object being created or changed is to be left blank.

**SAME:* The user-defined description, if any, is not to be changed.

'description': This is the user's description of the object being created or changed. A maximum of 50 characters, enclosed in apostrophes (required for blanks and other special characters), can be entered to describe the object. If an apostrophe is to be one of the 50 characters, you must use two apostrophes ('') instead of one.

VOL Parameter

The volume (VOL) parameter specifies the volume identifiers of the volumes to be used in a diskette or tape operation. For diskettes, a *volume* may consist of a single diskette within a magazine or slot, or multiple diskettes within the same magazine. Normally, a volume is a single diskette placed in a manual slot or 10 diskettes placed in a magazine. For tape, a volume consists of a single reel of tape; each reel is a separate volume.

The volume identifier is the identifier stored on each diskette or tape (in the volume label area) that it identifies. The diskettes (volumes) must be mounted on the diskette magazine drive and within the magazine(s) in the same order as the identifiers are specified here. The identifiers are matched, one by one, with the diskette locations specified in the LOC parameter. An inquiry message is sent to the system operator if a volume identifier is missing or out of order.

Tape volumes must be mounted on the tape units in the same order as their identifiers are specified in the VOL parameter and as the device names are specified in the DEV parameter of the tape device file commands. However, if the tapes are to be read backward (a function supported in COBOL), the volumes must be mounted in reverse order to that specified in the VOL parameter. (But the device names are still specified in forward order in the DEV parameter.)

The general rule for specifying diskette and tape volume identifiers is that as many as 6 characters, containing any combination of letters and digits, can be used. (Special characters can be used if the identifier is enclosed in apostrophes. However, if the diskette or tape is to be used on a system other than System/38, the requirements for specifying identifiers on that system must be considered.) The exception to this general rule applies to diskette magazines when they are in the save/restore format (see below).

Basic Exchange Format (Diskettes)

For diskettes in the basic data exchange format and for labeled tapes, the following rules apply:

- Characters: A maximum of 6 characters, or fewer, can be specified for each volume identifier. Alphabetic and numeric characters can be used in any order.

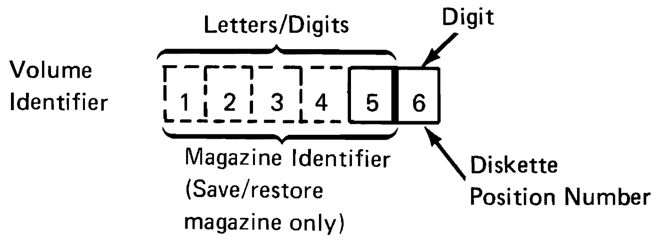
- Uniqueness: More than one volume can have the same identifier. (You may have a file using the same identifier for several diskettes; in this case, the system keeps track of the order internally with a sequence number written on the diskettes.)

- Order: Within a magazine, the volumes can be in any order, and have no relationship to one another. However, when multiple volumes (with different identifiers) exist in the magazine and more than one is to be used in an operation, they must be in the same order as the volume identifiers that are specified in the VOL parameter.

Save/Restore Format (Diskettes)

For diskettes in the save/restore (S/R) format, the following rules apply:

- Characters: The number of characters in the volume identifier depends on whether the S/R volume is a single diskette in a manual slot or is a diskette magazine.
  - Slots: The identifiers of S/R diskettes used in slots can have as many as 6 characters in any combination of letters and digits. (However, because no more than 5 characters can be specified on the INZDKT command for diskettes initialized in slots in the S/R format, 6 characters can occur only if the diskettes are first initialized in a magazine and then used in manual slots.)
  - Magazines: The volume identifiers of S/R magazines are made up of two parts: the *magazine* identifier and the diskette position number. The magazine identifier identifies *all* the diskettes in that magazine.



The magazine identifier (which is user-specified) can contain as many as 5 characters. Of those 5 characters, any combination of letters and digits can be specified.

When diskettes in a magazine are initialized in the S/R format (by the INZDKT command), the sixth or last character (a digit) in the volume identifier is assigned by the system; it indicates the diskette position in which the diskette was initialized and where the system expects it to be for any S/R operation. (Diskettes initialized in positions 1 through 9 are assigned a digit of 1 through 9, respectively, and the diskette initialized in position 10 is assigned the single digit of 0.) The only commands in which the last character (digit) can be specified as part of a S/R volume identifier are the CLRDKT (Clear Diskette) and the DLTDKTLBL (Delete Diskette Label) commands.

For any S/R operation, only the 5-character magazine identifier of each magazine to be used in the operation can be specified in the VOL parameter. (The position number only in the *first* magazine can be specified in the LOC parameter. The system always begins with diskette position 1 of every magazine after the first one.)

- Uniqueness: Volume identifiers should be unique. That is, each magazine volume identifier should be different from all other magazines, and each diskette within the same magazine *must* have the same *magazine* identifier. The volume identifiers of single diskette volumes should be different from all other single diskettes and magazines. They should also be unique from all non-S/R volume identifiers.

- Order: Within a magazine, the volumes must be in the same order as when they were initialized to the S/R format.

Multivolume Files

If a multivolume file on diskette (that is, a data file on several diskettes, all having the same name) does not have all the volumes used by the file mounted on the diskette magazine drive, a message is sent to the system operator. If (for S/R only) more than 100 diskettes are used for the same file, duplicate diskette sequence numbers occur for each additional diskette used after the first hundred (01-99, and 00). For each 100 diskettes written for the file, a message is sent to the system operator indicating the total number written. When the diskettes are to be read, the operator must determine the order in which the diskette magazines are to be mounted.

If multiple volumes (magazines, tapes, or single diskettes) to be used in an operation all have the same volume identifier, that identifier must be specified in the VOL parameter once for each volume used. For example, if three magazines named QGPL are to be used in a save operation, VOL(QGPL QGPL QGPL) must be specified.

When a multivolume file on *tape* is to be processed and multiple tape units are to be used, the tape volumes must be mounted on the tape devices in the same order as they are specified in the VOL parameter. (A save/restore operation cannot use more than one tape unit.) For example, if five volumes and three tape units are to be used, they are mounted as follows: VOL1 on unit 1, VOL2 on unit 2, VOL3 on unit 3, VOL4 on unit 1, and VOL5 on unit 2.

Values Allowed

**LOC:* The diskette volume mounted in the unit and location specified by the LOC parameter is to be used for the operation.

**MOUNTED:* The diskette volumes that are mounted in the unit and location specified by the LOC parameter are to be used for the operation. (This option is normally used when multiple save or restore operations are being done consecutively.)

**NONE:* No volume identifier is to be specified.

**SAME:* The previously specified volume identifiers, if any, are to be used.

**SAVVOL:* The system, using the save/restore history information, is to determine which tape or diskette volumes contain the most recently saved version. If the device and location specified in the DEV and LOC parameters of the restore command do not match the device and location of the most recently saved version of the object, an error message is sent to the user, and the function is terminated. If the wrong volume is mounted in the location specified by the command, a message is sent to the system operator that identifies the first volume that must be mounted before the restore operation can begin.

volume-identifier: Enter the identifiers of one or more volumes in the order in which they are to be mounted and used. Each volume identifier contains a maximum of 6 characters. For S/R operations using magazines, only the magazine identifier (5 characters maximum) can be specified for each volume.

WAITFILE Parameter

The WAITFILE parameter specifies the maximum number of seconds that a program is to wait for file resources to be allocated when the file is opened. If the program must wait, it is placed in a wait state until the resources are available or the wait time expires. If two or more file resources are needed and are not available because they are being used by different system users, the acquisition of each resource might require a wait. This maximum is applied to each wait.

The length of the wait can be specified in this parameter, or the default wait time of the class that applies to the object can be used. If the file resources cannot be allocated in the specified number of seconds, an error message is sent to the program to indicate that the file cannot be opened.

The file resources that must be allocated depend on the type of file being opened. File resources consist of the following.

- For device files that are not spooled (SPOOL(*NO)), the file resources include the file description and device description. Because the device description must be allocated, the device itself must also be available.
- For device files that are spooled (SPOOL(*YES)), the file resources include the file description, the specified output queue, and storage in the system for the spooled data. Because the data is spooled, the device description (and thus the device itself) need not be available.
- For data base files, the file resources consist of the entire file, including the file description and member description, as well as internal system objects that are referenced by the file and the associated access paths.

  The ALCOBJ command can be used to allocate specific file resources before the file is opened.

Values Allowed

**IMMED:* This value specifies that no wait time is allowed. When the file is opened, an immediate allocation of the file resources is required.

**CLS:* The default wait time specified in the class description is to be used as the wait time for the file resources to be allocated.

number-of-seconds: Enter the maximum number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

Appendix B. Expressions

A character string expression can be used for any parameter, element, or qualifier defined with EXPR(*YES) in the command definition object. Any expression can be used as a single parameter in the CHGVAR and IF commands. An expression in its simple form is a single constant, a variable, or a built-in function. An expression usually contains two operands and an operator that indicates how the expression is to be evaluated. Two or more expressions can be combined to make a complex expression.

The following types of expressions are supported in CL programs:

- Arithmetic (&VAR + 15)
- Character string (SIX || TEEN)
- Relational (&VAR > 15)
- Logical (&VAR & &TEST)

Each type is discussed on the following pages.

A *complex* expression contains multiple operands, operators that indicate what operation is to be performed on the operands, and parentheses to group them. Only one operator is allowed between operands, except for the + and - signs when they immediately precede a decimal value (as a signed value), and the *NOT operator when it is used in a logical expression.

No complex expression can have more than five nested levels of parentheses, including the outermost (required) level.

Arithmetic and character string expressions can be used together in a complex expression if they are used with relational and logical operators; for example: (A=B&(1+2)=3). A pair of arithmetic expressions or a pair of character string expressions can be compared within a relational expression. Also, relational expressions can be used within a logical expression.

Operators in Expressions

Operators are used in expressions to indicate (1) an action to be performed on the operands in the expression or (2) the relationship between the operands. There are four kinds of operators, one for each of the four types of expressions:

- Arithmetic operators (+, -, *, /)
- Character operator (||, |>, |<)
- Relational operators (=, >, <, >=, <=, ¬=, ¬>, ¬<)
- Logical operators (&, |, ¬)

Each operator must be between the operands of the expression in which it is used; for example, (&A + 4). Operators can be specified as a predefined value (like *EQ) or as a symbol (like =).

- All predefined value operators must have a blank on each side of the operator: (&VAR *EQ 7).
- All symbolic operators need *no* blanks on either side: (&VAR=7) or (&VAR = 7) is valid.

The following character combinations are the predefined values and symbols that represent the four kinds of operators; they should not be used in unquoted strings for any other purpose.

Predefined Value and Symbol		Meaning	Type
	+	Addition	Arithmetic operator
	-	Subtraction	Arithmetic operator
	*	Multiplication	Arithmetic operator
	/	Division	Arithmetic operator
*BCAT	|>[1]	Blank insertion with concatenation	Character string operator
*TCAT	|<[1]	Blank truncation with concatenation	Character string operator
*CAT	||[1]	Concatenation	Character string operator
*EQ	=	Equal	Relational operator
*GT	>	Greater than	Relational operator
*LT	<	Less than	Relational operator
*GE	>=	Greater or equal	Relational operator
*LE	<=	Less or equal	Relational operator
*NE	¬=	Not equal	Relational operator
*NG	¬>	Not greater	Relational operator
*NL	¬<	Not less	Relational operator
*AND	&	And	Logical operator
*OR	|[1]	Or	Logical operator
*NOT	¬[2]	Not	Logical operator

[1]In some national character sets and in the multinational character set, the character | (hex 4F) is replaced by the character ! (exclamation point). Either ! or *OR can be used as the OR operator and either !! or *CAT, !> or *BCAT, and !< or TCAT can be used for concatenation in those character sets.

[2]In some national character sets and in the multinational character set, the character ¬ (hex 5F) is replaced by the character ^. Either ^ or *NOT can be used as the NOT operator in those character sets.

Priority of Operators When Evaluating Expressions

When multiple operators occur in an expression, the expression is evaluated in a specific order depending upon the operators in the expression. The following table shows the priority of all the operators used in expressions, including signed decimal values.

Priority	Operators
1	+, -, signed (+ and -) decimal values, *NOT, ¬
2	*, /
3	+, - (when used between two operands)
4	*CAT, ||, *BCAT, |>, *TCAT, |<
5	*GT, *LT, *EQ, *GE, *LE, *NE, *NG, *NL, >, <, =, >=, <=, ¬=, ¬>, ¬<
6	*AND, &
7	*OR, |

A priority of 1 is the highest priority; signed values are evaluated first. A priority of 7 is the lowest; OR relationships are evaluated last. When operators with different priority levels appear in an expression, operations are performed according to priorities.

When operators of the *same* priority appear in an expression, operations are performed from left to right within the expression. Parentheses can always be used to control the order in which operations are performed.

Arithmetic Expressions

The operands in an arithmetic expression must be decimal values or CL variables. An arithmetic operator (only in symbolic form) must be between the operands. The results of all arithmetic expressions are decimal values, which may be stored in a CL variable.

Arithmetic operands can have signed or unsigned values; that is, each operand (whether it is a numeric value or a symbolic variable) can be immediately preceded by a plus (+) or minus (-) sign, but a sign is not required. When used as a sign, no blanks can occur between the + or - and its value. For example, a decimal value of 23.7 can be expressed as +23.7 or -23.7 (signed values) or as 23.7 (unsigned value).

The following are examples of arithmetic expressions:

```
(&A + 1)           (&A + &B -15)
(&A - &F)          (&A+&B-15)
(&A + (-&B))
```

If the last nonblank character on a line is a + or -, it is treated as a continuation character and not as an arithmetic operator.

Character String Expressions

The operands in a character string expression must be quoted or unquoted character strings, variables, or the substring (%SUBSTRING or %SST) built-in function. (The value associated with each variable or built-in function must be a character string.) The result of concatenation is a character string.

There are three operators that can be used in character string expressions. They are *CAT (concatenation symbol | |), *BCAT (concatenation with blank insertion symbol |>), and *TCAT (concatenation with trailing blank truncation symbol |<). These operators concatenate (or join) two character strings, but each has a slightly different function.

The *CAT operator specifies that two character strings are to be concatenated; for example, ABC *CAT DEF becomes ABCDEF. Blanks are included in the concatenation; for example, 'ABC ' *CAT 'DEF ' becomes ABC DEF.

In the case of *BCAT, any trailing blanks in the first character string are truncated; one blank is inserted, then the two character strings are concatenated. Any leading blanks on the second operand are not truncated. For example, ABC *BCAT DEF becomes ABC DEF; 'ABC ' *BCAT DEF becomes ABC DEF, and ABC *BCAT ' DEF' becomes ABC DEF.

In the case of *TCAT, all trailing blanks in the first character string are truncated, then the two character strings are concatenated. Any leading blanks on the second operand are not truncated. For example, ABC *TCAT DEF becomes ABCDEF; 'ABC ' *TCAT DEF becomes ABCDEF, and ABC *TCAT ' DEF' becomes ABC DEF.

Any blanks that surround the concatenation operator are ignored, but at least one blank must be on each side of the reserved value operator (*CAT, *BCAT, or *TCAT). If multiple blanks are wanted in the expression, a quoted character string must be used.

The following are examples of string expressions. (Assume the contents of &AA are 'GOOD ' and of &BB are 'REPLACEMENT'; the contents of &CC are 'ALSO GOOD' and of &DD are 'METHOD'.)

Expression		Result
(&AA || &BB)		
(&AA||&BB)	Equivalent	GOOD REPLACEMENT
(&AA *CAT &BB)	values	
(&CC |> &DD)		ALSO GOOD METHOD
(&CC *BCAT &DD)		
(A *CAT MOUSE)		AMOUSE
('FAST ' *CAT MOUSE)		FAST MOUSE
('FAST ' *TCAT MOUSE)		FASTMOUSE
('FAST ' *CAT MOUSE)		
(FAST *BCAT MOUSE)	Equivalent	FAST MOUSE
('FAST' *TCAT ' MOUSE')	values	
('AB' *CAT 'CD')		ABCD
(%SUBSTRING(&AA 1 5) *CAT %SUBSTRING(&BB 3 5))		GOOD PLACE
(%SUBSTRING(&CC 1 9) *BCAT %SUBSTRING(&BB 3 5))		ALSO GOOD PLACE
(&AA *CAT ' TIME')		GOOD TIME
(&CC *BCAT TIME)		ALSO GOOD TIME

The following example shows how several character variables and character strings can be concatenated to produce a message for a work station operator. (The example assumes that the variables &DAYS and &CUSNUM were declared as character, not decimal variables.)

```
DCL  VAR(&MSG)  TYPE(*CHAR)  LEN(100)
  •
  •
CHGVAR  &MSG  ('Customer'  *BCAT  &CUSNAME  *BCAT  +
                'Account Number'  *BCAT  &CUSNUM  *BCAT  +
                'is overdue by'  *BCAT  &DAYS  *BCAT  'days.')
```

After the appropriate variables have been substituted, the resulting message might be:

```
Customer ABC COMPANY Account Number 12345 is overdue by 4 days.
```

If the variables &DAYS and &CUSNUM had been declared as decimal variables, then two other CHGVAR commands would have to be executed to change the decimal variables to character variables before the concatenation could be performed. If, for example, two character variables named &DAYSALPH and &CUSNUMALPH were also declared in the program, the CHGVAR commands would be:

```
CHGVAR  &DAYSALPH  &DAYS
CHGVAR  &CUSNUMALPH  &CUSNUM
```

Then instead of &DAYS and &CUSNUM, the new variables &DAYSALPH and &CUSNUMALPH would be specified in the CHGVAR command used to concatenate all the variables and character strings for &MSG.

Relational Expressions

The operands in a relational expression can be arithmetic or character string expressions; they can also be logical constants and logical variables. Only two operands can be used with each relational operator. The data type (arithmetic, character string, or logical) must be the same for the pair of operands. The result of a relational expression is a logical value '0' or '1'.

Refer to the chart under *Operators in Expressions* for the meanings of the relational operators, which can be specified by symbols (=, >, <, >=, <=, ¬=, ¬>, ¬<) or their reserved values (*EQ, *GT, for example).

If an operation involves character fields of unequal length, the shorter field is extended by blanks added to the right.

Arithmetic fields are compared algebraically; character fields are compared according to the EBCDIC collating sequence.

When logical fields are compared, a logical one is greater than logical zero. ('1' > '0').

The following are examples of relational expressions:

```
(&X *GT 25) )                 (&NAME *EQ GSD)
(&X > 25)   }  Equivalent     (&NAME *EQ &GSD)
(&X>25)     )  Values         (&NAME *EQ 'GSD')
                              (&BLANK *EQ ' ')
```

Logical Expressions

The operands in a logical expression consist of relational expressions, logical variables or constants, separated by logical operators. Two or more of these types can be used in combinations, making up two or more expressions within expressions, up to the maximum of five nested levels of parentheses. The result of a logical expression is a '0' or '1' that can be used as part of another expression or saved in logical variables.

The logical operators that are used to specify the relationship between the operands are *AND and *OR (as reserved values), and & and | (as symbols). The AND operator indicates that both operands (on either side of the operator) have to be a certain value to produce a particular result. The OR operator indicates that one or the other of its operands can determine the result.

The logical operator *NOT (or ¬) is used to negate logical variables or logical constants. Any *NOT operators will be evaluated before the *AND or *OR operators are evaluated. Any values that follow *NOT operators will be evaluated before the logical relationship between the operands is evaluated.

The following are examples of logical expressions:

```
((&C *LT 1) *AND (&TIME *GT 1430))   }
(&C *LT 1 *AND &TIME *GT 1430)       }  Equivalent Values
((&C < 1) & (&TIME > 1430))          }
((&C<1)&(&TIME>1430))                }

(&A *OR *NOT &B)
(&TOWN *EQ CHICAGO *AND &ZIP *EQ 60605)
```

Two examples of logical expressions used in the IF command are:

```
IF &A CALL PROG1
IF (&A *OR &B) CALL PROG1
```

Substring Built-In Function

The substring built-in function operates on a portion of a character string that is contained in a CL character variable. %SUBSTRING or %SST can be used in expressions and as the first operand (receiver) of the CHGVAR command. (Refer to the description of the CHGVAR command.) This built-in function can be coded as either %SUBSTRING or %SST.

The syntax of the substring built-in function is:

```
%SUBSTRING(character-variable-name starting-position length)
                    or
%SST(character-variable-name starting-position length)
```

This built-in function produces a substring from the contents of the specified CL character variable. The substring begins at the specified starting position in the value and continues for the length specified. For example:

```
%SUBSTRING(&TEST 5 3)
```

In this example, the value of a field is changed. A substring is produced from the value contained in the variable named &TEST; the substring is 3 characters long and begins with the fifth character position in the string. If &TEST contained ABCDEFGHIJ, the resulting substring is EFG.

CL variables can also be used to specify the starting position and the length values in the function. For example:

```
CHGVAR &X %SUBSTRING(&A &B &C)
```

The value of the character variable named &X is to be replaced by the value in the variable &A, starting at the position obtained from variable &B and continuing for the length specified by the value in &C.

```
RTVJOBA SWS(&JOBSWS)
CHGVAR VAR(&CURSW4) VALUE(%SUBSTRING(&JOBSWS 4 1))
```

In this example, the RTVJOBA (Retrieve Job Attributes) command is used to retrieve the current value of the job's eight job switches. The CHGVAR command is then used to extract the current value of the fourth job switch only and store it in the variable &CURSW4. If the value of the eight job switches retrieved in &JOBSWS is 10010000, the second 1 would be stored in &CURSW4.

%SWITCH Built-In Function

The built-in function %SWITCH tests one or more of the eight job switches in the current job and returns a logical value of 1 or 0. If every job switch tested by %SWITCH has the value indicated, the result is a 1 (true); if any switch tested does not have the value indicated, the result is a 0 (false).

The syntax of the %SWITCH built-in function is:

```
%SWITCH(8-character-mask)
```

The 8-character mask is used to indicate which job switches are to be tested, and what value each switch is to be tested for. Each position in the mask corresponds with one of the eight job switches in a job. Position 1 corresponds with job switch 1, position 2 with switch 2, and so on. Each position in the mask can be specified as one of three values: 0, 1, or X.

0	The corresponding job switch is to be tested for a 0 (off).
1	The corresponding job switch is to be tested for a 1 (on).
X	The corresponding job switch is not to be tested. The value in the switch does not affect the result of %SWITCH.

If %SWITCH(0X111XX0) is specified, job switches 1 and 8 are tested for 0s, switches 3, 4, and 5 are tested for 1s, and switches 2, 6, and 7 are not tested. If each job switch contains the value (1 or 0 only) shown in the mask, the result of %SWITCH is true (1).

%SWITCH can be used in the CHGVAR and IF commands. On the CHGVAR command, it can be used in place of a logical variable in the VALUE parameter. On the IF command, it can be used in the COND parameter as the logical expression to be tested.

The following two examples show how the same mask can be used to control a branch in a program (the IF command), or to set the value of a variable (the CHGVAR command).

```
IF  COND(%SWITCH(0X111XX0)) THEN(GOTO C)
CHGVAR  VAR(&A) VALUE(%SWITCH(0X111XX0))
```

If job switches 1, 3, 4, 5, and 8 respectively contain 0, 1, 1, 1, and 0 respectively when %SWITCH(0X111XX0) is specified in the IF command, the result is true and the program branches to the command having the label C. If one or more of the switches tested do not have the values indicated in the mask, the result is false and the branch does not occur. If the same mask is used in the CHGVAR command and the result is true, the variable &A is set to a '1'; if the result is false, &A is set to a '0'. (&A must be declared as a logical variable.)

Appendix C. User Profile Matrix Chart

The chart shown on the following pages identifies which IBM-supplied user profiles are authorized to use each command. The chart shows the command authorizations that exist when the system is shipped to the user's installation.

All of the CL commands are listed in alphabetic order in the user profile chart. All of the IBM-supplied user profiles are listed by their user-profile names across the top of the chart.

The following list describes briefly the user profiles that are shipped with the system.

- QSECOFR. The security officer user profile has complete authority for all the CL commands. Because the security officer is responsible for the security of the commands, he is the only one who can authorize other users on the system to use the commands. By using the security commands, the security officer can also grant the special rights to other users or revoke them. (The special rights are described in the CRTUSRPRF command description, on the SPCAUT parameter.)

- QPGMR. The programmer user profile has authority for the commands that are normally used by a system or application programmer. The QPGMR user profile is authorized (when the system is shipped) to use most of the commands, except for those used only by the system operator, or service personnel, and those *restricted* to the security officer.

- QSYSOPR. The system operator user profile has authority for the commands that are normally used by the system operator. The QSYSOPR user profile is authorized to use those commands that control the system's operation, control the jobs that are active in the system, and perform save/restore operations. When the system is shipped, QSYSOPR has the job control rights and save system rights assigned to it.

- QUSER. The work station user profile is shipped with no commands explicitly authorized for it. Only the commands that are identified as public commands can be used by users operating under this user profile.

- QPSR. The PSR user profile has authority for the commands that are used by the programming support representative (PSR) who services the system programming.

- QCE. The CE user profile has authority for the commands that are used by the customer engineer (CE) who services the system hardware using the concurrent service monitor.

The security officer can change the command authorizations for any of the user profiles; he controls which commands are public and which users can use a command. Each command can be explicitly authorized for one or more users, except for the few commands that are restricted to the security officer only. Note that some authority is usually needed for the CPF objects affected by the commands, as well as for the commands themselves.

The following chart shows the commands that are public (indicated by a P in the QUSER profile column), those that are explicitly authorized for specific user profiles (indicated by an E under the profile name for which they are authorized), and those that are restricted to the security officer only (indicated by an R in the QSECOFR profile column).

All of the commands that are public are shown by a P in the QUSER column only, for clarity of presentation, but they are public for *all* of the other user profiles as well. Therefore, for example, the QPGMR user profile has authority for all of the commands in both the QUSER and the QPGMR columns. Also, because the security officer has the authority to use *every* command, only the commands that he cannot authorize for any other profile are indicated in the QSECOFR column (with an R).

The security officer may wish to use this chart to change some of the command authorizations and indicate the changes on the chart.

	User Profile Names					
Command Name	**QUSER (P)**	**QPGMR (E)**	**QSYSOPR (E)**	**QPSR (E)**	**QCE (E)**	**QSECOFR (R)**
ADDAJE	P					
ADDBKP	P					
ADDFCTE	P					
ADDJOBQE	P					
ADDLFM	P					
ADDMSGD	P					
ADDPFM	P					
ADDPGM	P					
ADDRJECMNE	P					
ADDRJERDRE	P					
ADDRJEWTRE	P					
ADDRTGE	P					
ADDTRC	P					
ADDWSE	P					
ALCOBJ	P					
ANSLIN	P					
APYJRNCHG		E		E	E	
APYPGMCHG		E	E	E	E	
CALL	P					
CHGAJE	P					

Command Name	User Profile Names					
	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
CHGBSCF	P					
CHGCMD	P					
CHGCMNF	P					
CHGCNPA		E	E	E		
CHGCRDF	P					
CHGCUD	P					
CHGDBG	P					
CHGDEVD	P					
CHGDFUDEF	P					
CHGDKTF	P					
CHGDSPF	P					
CHGDTA	P					
CHGDTAARA	P					
CHGFCT	P					
CHGFCTE	P					
CHGJOB	P					
CHGJOBD	P					
CHGJOBQE	P					
CHGJRN		E	E	E	E	
CHGLF	P					
CHGLFM	P					
CHGLIND	P					
CHGMSGD	P					
CHGMSGQ	P					
CHGOBJOWN	P					
CHGOUTQ	P					
CHGPF		E		E	E	
CHGPFM		E		E	E	
CHGPGMVAR	P					
CHGPRTF	P					
CHGPTR				E	E	
CHGQRYDEF	P					
CHGRJECMNE	P					
CHGRJERDRE	P					
CHGRJEWTRE	P					
CHGRTGE	P					
CHGSBSD	P					
CHGSPLFA	P					
CHGSRCPF		E		E	E	
CHGSSND	P					
CHGSYSVAL		E	E	E	E	
CHGTAPF	P					
CHGUSRPRF						R
CHGVAR	P					

Command Name	User Profile Names					
	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
CHGWSE	P					
CHKOBJ	P					
CLNPRT		E	E	E	E	
CLRDKT		E	E	E	E	
CLRJOBQ	P					
CLRLIB	P					
CLROUTQ	P					
CLRPFM	P					
CLRTRCDTA	P					
CNLJOB	P					
CNLRCV	P					
CNLRDR	P					
CNLRJERDR	P					
CNLRJEWTR	P					
CNLRQS	P					
CNLSPLF	P					
CNLWTR	P					
CPYF	P					
CPYSPLF	P					
CRTBSCF		E		E	E	
CRTCBLPGM		E		E	E	
CRTCLPGM		E		E	E	
CRTCLS		E		E	E	
CRTCMD		E		E	E	
CRTCMNF		E		E	E	
CRTCRDF		E		E	E	
CRTCUD		E		E	E	
CRTDEVD		E		E	E	
CRTDFUAPP	P					
CRTDFUDEF	P					
CRTDKTF		E		E	E	
CRTDSPF		E		E	E	
CRTDTAARA		E		E	E	
CRTEDTD		E		E	E	
CRTFCT	P					
CRTJOBD		E		E	E	
CRTJOBQ		E		E	E	
CRTJRN		E		E	E	
CRTJRNRCV		E		E	E	
CRTLF	P					
CRTLIB		E		E	E	
CRTLIND		E		E	E	
CRTMSGF		E		E	E	
CRTMSGQ		E		E	E	

Command Name	User Profile Names					
	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
CRTOUTQ		E		E	E	
CRTPF		E		E	E	
CRTPRTF		E		E	E	
CRTPRTIMG		E		E	E	
CRTQRYAPP	P					
CRTQRYDEF	P					
CRTRPGPGM	P					
CRTRPTPGM	P					
CRTSBSD		E		E	E	
CRTSRCPF		E		E	E	
CRTSSND	P					
CRTTAPF		E		E	E	
CRTTBL		E		E	E	
CRTUSRPRF						R
CVTDAT	P					
DATA	P					
DCL	P					
DCLDTAARA	P					
DCLF	P					
DLCOBJ	P					
DLTCLS	P					
DLTCMD	P					
DLTCUD	P					
DLTDEVD	P					
DLTDKTLBL		E	E	E	E	
DLTDTAARA	P					
DLTEDTD	P					
DLTDFUAPP	P					
DLTF	P					
DLTFCT	P					
DLTJOBD	P					
DLTJOBQ	P					
DLTJRN	P					
DLTJRNRCV	P					
DLTLIB	P					
DLTLIND	P					
DLTMSGF	P					
DLTMSGQ	P					
DLTOUTQ	P					
DLTOVR	P					
DLTPGM	P					
DLTPRTIMG	P					
DLTQRYAPP	P					
DLTSBSD	P					

Command Name	User Profile Names					
	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
DLTSSND	P					
DLTTBL	P					
DLTUSRPRF						R
DMPCLPGM	P					
DMPJOB		E	E	E	E	
DMPJOBINT		E	E	E	E	
DMPOBJ		E	E	E	E	
DMPSYSOBJ		E	E	E	E	
DMPTAP	P					
DO	P					
DSNDFUAPP	P					
DSNFMT	P					
DSNQRYAPP	P					
DSPACTJOB	P					
DSPAUTUSR						R
DSPBKP	P					
DSPCLS	P					
DSPCMD	P					
DSPCNPA		E	E	E		
DSPCTLSTS		E	E	E	E	
DSPCUD	P					
DSPDBG	P					
DSPDBR	P					
DSPDEVCFG	P					
DSPDEVD	P					
DSPDEVSTS		E	E	E	E	
DSPDKT	P					
DSPDTA	P					
DSPDTAARA	P					
DSPEDTD	P					
DSPFCT	P					
DSPFD	P					
DSPFFD	P					
DSPJOB	P					
DSPJOBD	P					
DSPJOBQ	P					
DSPJRN	P					
DSPJRNA	P					
DSPJRNRCVA	P					
DSPLIB	P					
DSPLIBL	P					
DSPLIND	P					
DSPLINSTS		E	E	E	E	
DSPLOG		E	E	E	E	

Command Name	User Profile Names					
	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
DSPMSG	P					
DSPMSGD	P					
DSPMSGF	P					
DSPOBJAUT	P					
DSPOBJD	P					
DSPOBJLCK	P					
DSPOUTQ	P					
DSPOVR	P					
DSPPGMCHG		E	E	E	E	
DSPPGMREF	P					
DSPPGMVAR	P					
DSPRDR	P					
DSPRJESSN	P					
DSPSBMJOB	P					
DSPSBS	P					
DSPSBSD	P					
DSPSPLF	P					
DSPSPLFA	P					
DSPSRVSTS		E	E	E	E	
DSPSSND	P					
DSPSYS	P					
DSPSYSSTS	P					
DSPSYSVAL	P					
DSPTAP	P					
DSPTRC	P					
DSPTRCDTA	P					
DSPUSRPRF	P					
DSPWTR	P					
DUPDKT		E	E	E	E	
EDTSRC	P					
ELSE	P					
ENDCBLDBG	P					
ENDDBG	P					
ENDDO	P					
ENDINP	P					
ENDJOB	P					
ENDJRNPF	P					
ENDLOG	P					
ENDPGM	P					
ENDSRV		E	E	E	E	
ENTCBLDBG	P					
ENTDBG		E		E	E	
FMTDTA	P					
FMTRJEDTA	P					

Command Name	User Profile Names					
	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
GOTO	P					
GRTOBJAUT	P					
GRTUSRAUT	P					
HLDJOB	P					
HLDJOBQ	P					
HLDOUTQ	P					
HLDRDR	P					
HLDSPLF	P					
HLDWTR	P					
IF	P					
INZDKT		E	E	E	E	
INZPFM	P					
INZTAP		E	E	E	E	
JOB	P					
JRNPF	P					
LODPGMCHG		E	E	E	E	
LOGDBF	P					
LSTCMDUSG	P					
LSTCNPDTA		E	E	E		
LSTCNPHST		E	E	E		
LSTERRLOG		E	E	E		
LSTINTDTA		E	E	E		
MONMSG	P					
MOVOBJ	P					
OVRBSCF	P					
OVRCMNF	P					
OVRCRDF	P					
OVRDBF	P					
OVRDKTF	P					
OVRDSPF	P					
OVRMSGF	P					
OVRPRTF	P					
OVRTAPF	P					
PCHPGM				E	E	
PGM	P					
PRPAPAR		E	E	E	E	
PWRCTLU		E	E	E	E	
PWRDEV		E	E	E	E	
PWRDWNSYS	P					
QRYDTA	P					
RCLRSC	P					
RCLSTG		E	E	E	E	
RCVDTAARA	P					
RCVF	P					

Command Name	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
RCVMSG	P					
RETURN	P					
RGZPFM	P					
RLSJOB	P					
RLSJOBQ	P					
RLSOUTQ	P					
RLSRDR	P					
RLSSPLF	P					
RLSWTR	P					
RMVAJE	P					
RMVBKP	P					
RMVFCTE	P					
RMVJOBQE	P					
RMVJRNCHG		E		E		E
RMVM	P					
RMVMSG	P					
RMVMSGD	P					
RMVPGM	P					
RMVPGMCHG		E	E	E	E	
RMVRJECMNE	P					
RMVRJERDRE	P					
RMVRJEWTRE	P					
RMVRTGE	P					
RMVTRC	P					
RMVWSE	P					
RNMDKT		E	E	E	E	
RNMOBJ	P					
RPLLIBL	P					
RRTJOB	P					
RSMBKP	P					
RSTAUT	P					
RSTLIB		E	E	E	E	
RSTOBJ		E	E	E	E	
RSTUSRPRF	P					
RTVCLSRC	P					
RTVDFUSRC	P					
RTVDTAARA	P					
RTVJOBA	P					
RTVMSG	P					
RTVQRYSRC	P					
RTVSYSVAL	P					
RVKOBJAUT	P					
SAVCHGOBJ		E	E	E	E	
SAVLIB		E	E	E	E	

Command Name	User Profile Names					
	QUSER (P)	QPGMR (E)	QSYSOPR (E)	QPSR (E)	QCE (E)	QSECOFR (R)
SAVOBJ		E	E	E	E	
SAVSYS	P					
SBMCRDJOB	P					
SBMDBJOB	P					
SBMDKTJOB	P					
SBMJOB	P					
SBMRJEJOB	P					
SIGNOFF	P					
SNDBRKMSG		E	E	E	E	
SNDDTAARA	P					
SNDF	P					
SNDJRNE	P					
SNDMSG	P					
SNDPGMMSG	P					
SNDRCVF	P					
SNDRPY	P					
SRVJOB		E	E	E	E	
STRCNFCHK		E	E	E	E	
STRCRDRDR	P					
STRCRDWTR	P					
STRDBRDR	P					
STRDKTRDR	P					
STRDKTWTR	P					
STRPDP		E	E	E		
STRPRTWTR	P					
STRRJECSL	P					
STRRJERDR	P					
STRRJESSN	P					
STRRJEWTR	P					
STRSBS		E	E	E	E	
TFRCTL	P					
TFRJOB	P					
TRCINT		E		E		
TRCJOB		E	E	E	E	
TRMCPF	P					
TRMRJESSN	P					
TRMSBS	P					
VFYPRT		E	E	E	E	
VRYCTLU		E	E	E	E	
VRYDEV		E	E	E	E	
VRYLIN		E	E	E	E	
WAIT	P					

Appendix D. Files Used by CL Commands

Many of the IBM-supplied CL commands use data base and device files during their execution. The attributes of many of these files, which are provided as part of CPF when the system is shipped, are given in the *CPF Programmer's Guide*.

The lists in this appendix provide a cross-reference between the commands and the CPF-provided files used by the commands. Only the data base files and BSC, card, diskette, tape, and printer device files used by CL commands are included here. The display device files are not listed because they should not be changed or overridden. Other IBM-supplied files *not* used by the commands (such as QPRINT, for example) can be found in the *CPF Programmer's Guide*.

The first list contains the CL commands that use CPF-provided files. The list is in alphabetic order by command name and by file name, when there is more than one file listed for a command.

An asterisk (*) following the description of a file indicates that the file is used only when the output from a display command is to be printed (when OUTPUT(*) is specified in a batch job, or when OUTPUT(*LIST) is specified in a batch or interactive job).

Command Name	File Name	File Type	Description of File
CHGDTA	QDTALOG	PRT	Audit control log printer file
	QDTAPRT	PRT	Record and accumulator total printer file
CLRDKT	QSYSDKT	DKT	Diskette device file used for output
CPYF	QSYSPRT	PRT	Copy file output listing
CRTBSCF	QDDSSRC	DBF	DDS source default input file
	QPDDSSRC	PRT	DDS source output file
CRTCBLPGM	QCBLSRC	DBF	COBOL source default input file
	QSYSPRT	PRT	COBOL source output file
CRTCLPGM	QCLSRC	DBF	CL source default input file
	QSYSPRT	PRT	CL source listing output file
CRTCMD	QCMDSRC	DBF	Default source input file
	QSYSPRT	PRT	Command definition output file
CRTCMNF	QDDSSRC	DBF	DDS source default input file
	QPDDSSRC	PRT	DDS source output file
CRTDFUAPP	QUDSSRC	DBF	DFU source default input file
	QSYSPRT	PRT	DFU source output file

Command Name	File Name	File Type	Description of File
CRTDFUDEF	QUDSSRC	DBF	DFU source default input file
	QSYSPRT	PRT	DFU source output file
CRTDSPF	QDDSSRC	DBF	DDS source default input file
	QPDDSSRC	PRT	DDS source output file
CRTLF	QDDSSRC	DBF	DDS source default input file
	QPDDSSRC	PRT	DDS source output file
CRTPF	QDDSSRC	DBF	DDS source default input file
	QPDDSSRC	PRT	DDS source output file
CRTPRTF	QDDSSRC	DBF	DDS source default input file
	QPDDSSRC	PRT	DDS source output file
CRTPRTIMG	QIMGSRC	DBF	Print image source default input file
CRTQRYAPP	QQRYSRC	DBF	Query source default input file
	QSYSPRT	PRT	Query source output file
CRTQRYDEF	QQRYSRC	DBF	Query source default input file
	QSYSPRT	PRT	Query source output file
CRTRPGPGM	QRPGSRC	DBF	RPG source default input file
	QSYSPRT	PRT	RPG source output file
CRTRPTPGM	QRPGSRC	DBF	RPG source default input file
	QSYSPRT	PRT	RPG source output file
CRTSRCPF	QDDSSRC	DBF	DDS source default input file
	QPDDSSRC	PRT	DDS source output file
CRTTBL	QTBLSRC	DBF	Table source default input file
DMPCLPGM	QPPGMDMP	PRT	CL program dump printer file
DMPJOB	QPSRVDMP	PRT	Service dump printer file
DMPOBJ	QPSRVDMP	PRT	Service dump printer file
DMPSYSOBJ	QPSRVDMP	PRT	Service dump printer file
DMPTAP	QPTAPDMP	PRT	Tape dump printer file
DSNFMT	QDDSSRC	DBF	DDS source default input file
	QSYSPRT	PRT	SDA source output file
DSPACTJOB	QPDSPAJB	PRT	Active jobs display printer file*
DSPAUTUSR	QPAUTUSR	PRT	Authorized users printer file*
DSPBKP	QPDBGDSP	PRT	Breakpoint printer file (debug mode)*
DSPCLS	QPDSPCLS	PRT	Class printer file*
DSPCMD	QSYSPRT	PRT	Command definition output file
DSPCNPA	QPCNPDSP	PRT	CSNAP printer file
	QDCNPDSP	DSP	CSNAP display file
DSPCTLSTS	QPNETSTS	PRT	Control unit status printer file (device configuration)*

Command Name	File Name	File Type	Description of File
DSPCUD	QPDVCDSP	PRT	Control unit description printer file (device configuration)*
DSPDBG	QPDBGDSP	PRT	Debug display printer file (debug mode)*
DSPDBR	QADSPDBR	DBF	Data base file that defines the record format of the created file used to store data base relationships. Not an actual output file.
	QPDSPDBR	PRT	Data base relations printer file*
DSPDEVCFG	QPDVCDSP	PRT	Device configuration printer file*
DSPDEVD	QPDVCDSP	PRT	Device description printer file (device configuration)*
DSPDEVSTS	QPNETSTS	PRT	Device status printer file (device configuration)*
DSPDKT	QPDSPDKT	PRT	Printer file for diskettes in basic data exchange format*
	QPSRODSP	PRT	Printer file for diskettes in save/restore format*
	QSYSDKT	DKT	Diskette device file for input
DSPDTA	QDTAPRT	PRT	DFU audit control print file
DSPDTAARA	QPDSPDTA	PRT	Data area printer file*
DSPEDTD	QPDVCDSP	PRT	Edit description printer file (device configuration)*
DSPFCT	QPDSPFCT	PRT	Forms control table printer file*
DSPFD	QPDSPFD	PRT	File description printer file
DSPFFD	QADSPFFD	DBF	Data base file that defines the record format of the created file used to store file field descriptions. Not an actual output file.
	QPDSPFFD	PRT	File field description printer file*
DSPJOB	QPDSPJOB	PRT	Job display printer file*
DSPJOBD	QPRTJOBD	PRT	Job description printer file*
DSPJOBQ	QPRTSPLQ	PRT	Job queue printer file (spooling queue)*
DSPJRN	QADSPJRN	DBF	Data base file that defines the record format of the created file used for the converted journal entries
	QPDSPJRN	PRT	Journal display printer file*
DSPJRNA	QPDSPJRNA	PRT	Journal attributes display printer file*
DSPJRNRCVA	QPDSPRCV	PRT	Journal receiver attributes display printer file*
DSPLIB	QPDSPLIB	PRT	Library display printer file*
DSPLIBL	QPRTLIBL	PRT	Library list printer file*

Command Name	File Name	File Type	Description of File
DSPLIND	QPDVCDSP	PRT	Line description printer file (device configuration)*
DSPLINSTS	QPNETSTS	PRT	Line status printer file (device configuration)*
DSPLOG	QPDSPLOG	PRT	Log display printer file*
DSPMSG	QPDSPMSG	PRT	Message display printer file*
DSPMSGD	QPMSGD	PRT	Message description printer file*
DSPMSGF	QPMSGD	PRT	Message file printer file*
DSPOBJAUT	QPOBJAUT	PRT	Object authority printer file*
DSPOBJD	QPRTOBJD	PRT	Object description printer file*
DSPOBJLCK	QPDSPOLK	PRT	Object locks display printer file*
DSPOUTQ	QPRTSPLQ	PRT	Output queue printer file (spooling queue)*
DSPPGMCHG	QPDSPPC	PRT	Programming change display printer file*
DSPPGMREF	QADSPPGM	DBF	Data base file that defines the record format of the created file used to store program references. Not an actual output file.
	QPDSPPGM	PRT	Program references printer file*
DSPPGMVAR	QPDBGDSP	PRT	Program variable printer file (debug mode)*
DSPRDR	QPRTRDWT	PRT	Reader display printer file*
DSPRJESSN	QPRJESTS	PRT	RJE session display printer file*
DSPSBMJOB	QPDSPSBS	PRT	Submitted jobs printer file*
DSPSBS	QPDSPSBS	PRT	Subsystem display printer file*
DSPSBSD	QPRTSBSD	PRT	Subsystem description printer file*
DSPSPLFA	QPDSPSFA	PRT	Spooled file attributes printer file*
DSPSSND	QPRTSSND	PRT	Session description printer file*
DSPSYS	QPDSPSYS	PRT	System display printer file*
DSPSYSSTS	QPDSPSTS	PRT	System status printer file*
DSPSYSVAL	QPDSPSVL	PRT	System value printer file*
DSPTAP	QPTAPDSP	PRT	Printer file for tape output
	QPSRODSP	PRT	Printer file for tapes in save/restore format*
	QSYSTAP	TAP	Tape device file for input
DSPTRC	QPDBGDSP	PRT	Trace printer file (debug mode)*
DSPTRCDTA	QPDBGDSP	PRT	Trace data printer file (debug mode)*
DSPUSRPRF	QPUSRPRF	PRT	User profile printer file*
DSPWTR	QPRTRDWT	PRT	Writer display printer file*
EDTSRC	QTXTSRC	DBF	SEU source default input file
	QPSUPRTF	PRT	SEU source output file

Command Name	File Name	File Type	Description of File
FMTDTA	QSYSPRT	PRT	Data format printer file
INZDKT	QSYSDKT	DKT	Diskette device file used for output
INZTAP	QSYSTAP	TAP	Tape device file used for output
LODPGMCHG	QSYSDKT	DKT	Diskette device file used for input
LSTCMDUSG	QSYSPRT	PRT	Command usage printer file
LSTERRLOG	QPCSMPRT	PRT	Error log printer file (concurrent service monitor)
LSTINTDTA	QPCSMPRT	PRT	Internal data printer file (concurrent service monitor)
PCHPGM	QPPCHPGM	PRT	Patch data printer output file
PRPAPAR	QSYSDKT	DKT	Diskette device file used for output
QRYDTA	QQRYPRT	PRT	Query application output file
	QQRYPRT2	PRT	Secondary query application output file
	QSYSPRT	PRT	Query source output file
RCLSTG	QPRCLDMP	PRT	Reclaim dump output listing
RSTLIB	QSYSDKT	DKT	Diskette device file used for input
	QSYSTAP	TAP	Tape device file used for input
RSTOBJ	QSYSDKT	DKT	Diskette device file used for input
	QSYSTAP	TAP	Tape device file used for input
RSTUSRPRF	QSYSDKT	DKT	Diskette device file used for input
SAVLIB	QSYSDKT	DKT	Diskette device file used for output
	QSYSTAP	TAP	Tape device file used for output
SAVOBJ	QSYSDKT	DKT	Diskette device file used for output
	QSYSTAP	TAP	Tape device file used for output
SAVSYS	QSYSDKT	DKT	Diskette device file used for output
SBMCRDJOB	QCSPLCRD	CRD	Card device file used for all card reading
SBMDKTJOB	QKSPLDKT	DKT	Diskette device file used for all diskette reading
STRCRDRDR	QCSPLCRD	CRD	Card device file used for all card reading
STRCRDWTR	QCSPLCRD	CRD	Card device file used for all card writing
STRDKTRDR	QKSPLDKT	DKT	Diskette device file used for all diskette reading
STRDKTWTR	QKSPLDKT	DKT	Diskette device file used for all diskette writing
STRPRTWTR	QPSPLPRT	PRT	Printer device file used for all printer writing

File Chart

Command Name	File Name	File Type	Description of File
TRCINT	QPCSMPRT	PRT	Internal trace printer output file (concurrent service monitor)
	QSYSDKT	DKT	Diskette device file for output
TRCJOB	QPSRVTRC	PRT	Job trace printer output file

The following list contains the files above that are used by more than one command.

File Name	Commands Using the File
QCSPLCRD	SBMCRDJOB, STRCRDRDR, STRCRDWTR
QDDSSRC	CRTBSCF, CRTCMNF, CRTDSPF, CRTLF, CRTPF, CRTPRTF, DSNFMT
QKSPLDKT	SBMDKTJOB, STRDKTRDR, STRDKTWTR
QPCSMPRT	LSTERRLOG, LSTINTDTA, TRCINT
QPDBGDSP	DSPBKP, DSPDBG, DSPPGMVAR, DSPTRC, DSPTRCDTA
QPDDSSRC	CRTBSCF, CRTCMNF, CRTDSPF, CRTLF, CRTPF, CRTPRTF
QPDSPSBS	DSPSBMJOB, DSPSBS
QPDVCDSP	DSPCTLSTS, DSPCUD, DSPDEVCFG, DSPDEVSTS, DSPDEVD, DSPEDTD, DSPLIND, DSPLINSTS
QPMSGD	DSPMSGD, DSPMSGF
QPRTRDWT	DSPRDR, DSPWTR
QPRTSPLQ	DSPJOBQ, DSPOUTQ
QPSRODSP	DSPDKT, DSPTAP
QPSRVDMP	DMPJOB, DMPOBJ, DMPSYSOBJ
QSYSDKT	CLRDKT, DSPDKT, INZDKT, LODPGMCHG, PRPAPAR, RSTLIB, RSTOBJ, RSTUSRPRF, SAVLIB, SAVOBJ, SAVSYS, TRCINT
QSYSPRT	CPYF, CPYSPLF, CRTCBLPGM, CRTCLPGM, CRTCMD, CRTDFUAPP, CRTQRYAPP, CRTRPGPGM, CRTRPTPGM, DSNFMT, DSPCMD, DSPDTA, FMTDTA, LSTCMDUSG, QRYDTA
QSYSTAP	DSPTAP, INZTAP, RSTLIB, RSTOBJ, SAVLIB, SAVOBJ

Appendix E. Error Messages That Can Be Monitored

This appendix contains the escape, notify, and status messages that can be generated by each CL command. The commands are listed in alphabetic order. The following information is given for each message:

- The message identifier
- The message type code (E for escape messages, N for notify messages, and S for status messages)
- The message text

For a more detailed description of any CPF message, including its severity code, see the *System/38 Messages Guide: CPF, RPG III, IDU.*

***ALL**

CPF0001 E	Error found on &1 command.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9805 E	Object &2.&3 type *&5 destroyed.
CPF9807 E	One or more libraries in library list previously deleted.
CPF9808 E	Unable to allocate one or more libraries on library list.
CPF9830 E	Unable to allocate library &1.
CPF9901 S	Request check. &1 unmonitored by &2 at &3
CPF9999 E	Function check. &1 unmonitored by &2 at stmt &5, inst &3.

ADDAJE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

ADDBKP

CPF1924 E	ODV number from symbol table not valid for program &1.
CPF1925 E	MI instruction number from BOM or symbol table not valid for pgm &1.
CPF1999 E	Errors occurred on command.

ADDFCTE

RJE0024 E	Errors found, cannot continue.

ADDJOBQE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

ADDLFM

CPF7306 E	Member &1 not added to file &2.&3.

ADDMSGD

CPF2401 E	Not authorized to library &1.
CPF2407 E	Message file &1.&2 not found.
CPF2411 E	Not authorized to message file &1.&2.
CPF2412 E	Msg ID &1 already exists in msg file &2.&3.
CPF2430 E	Message description not added to message file
CPF2461 E	Message file &1 not extended because already extended max of &2 times
CPF2483 E	Message file currently in use.
CPF2510 E	Message file &1.&2 logically damaged.
CPF9830 E	Unable to allocate library &1.

ADDPFM

CPF7306 E	Member &1 not added to file &2.&3.

ADDPGM

CPF1999 E	Errors occurred on command.

ADDRJECMNE

RJE0036 E	Maximum of &1 &2 entries in session desc exceeded.
RJE0082 E	Cmn entry &1.&2 already exists in session desc &3.&4.

ADDRJERDRE

RJE0018 E	&1 already exists in session desc &2.&3.

ADDRJEWTRE

RJE0018 E	&1 already exists in session desc &2.&3.
RJE0067 E	No members found in file &1.&2

ADDRTGE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

ADDTRC

CPF1924 E	ODV number from symbol table not valid for program &1.
CPF1925 E	MI instruction number from BOM or symbol table not valid for pgm &1.
CPF1999 E	Errors occurred on command.

ADDWSE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

ALCOBJ

CPF1002 E	Unable to allocate object(s).
CPF1040 E	Maximum number of objects allocated on system reached.
CPF1085 E	Object(s) not allocated.

ANSLIN

CPF2704 E	Line &1 not found.
CPF5917 E	Not authorized to line description &1.
CPF5919 E	Line &1 not available.
CPF5945 E	ANSLIN command not valid for line &1.

APYJRNCHG

CPF7002 E	Object &1.&2 not physical file.
CPF7003 E	Entry not journaled to journal &1.&2, reason &3.
CPF7006 E	Mbr &3 not found in file &1.&2.
CPF7007 E	Unable to allocate mbr &3 of file &1.&2.
CPF7041 E	Entry for job &1.&2.&3 not found on RCVRNG.
CPF7042 E	File &1.&2 not being journaled or being journaled through different journal.
CPF7044 E	Apply or remove of journaled entries failed, reason code &7.
CPF7045 E	Partial damage detected on journal receiver &1.&2.
CPF7046 E	Duplicate key not allowed for file &1.&2 mbr &3.
CPF7047 E	File &1.&2 mbr &3 is full.
CPF7048 E	Unique access path problems prevent processing journaled change to file &1.&2 mbr &3.
CPF7049 E	Operation cannot be performed across journal entry &6.
CPF7050 E	Date of LASTSAVE does not match restored version of file &1.&2 mbr &3.
CPF7051 E	Save entry for file &1.&2 mbr &3 not found in RCVRNG.
CPF7052 E	Select/omit failure on rcd in logical file over file &1.&2 mbr &3.
CPF7053 E	Error on RCVRNG specifications, reason &1.
CPF7054 E	Invalid FROM and TO values.
CPF7055 E	Maximum number of members exceeded.
CPF7057 E	*LIBL invalid for file library if FILE(*ALL) specified.
CPF7058 E	Apply or remove of journaled entries failed.
CPF7059 E	Entry for &1 not found on RCVRNG.
CPF7068 E	Entry required to perform apply or remove not found on journal.
CPF7069 E	No entries applied or removed from journal &1.&2.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9809 E	Library &1 not accessible.
CPF9810 E	Library &1 not found.
CPF9812 E	File &1.&2 not found.
CPF9820 E	Not authorized to library &1.
CPF9822 E	Not authorized to file &1.&2.

APYPGMCHG

CPF3551 E	PGMID parameter not valid
CPF3558 E	Unable to allocate &1.&3 type *&2.
CPF3564 E	Programming change &1-&2 damaged.
CPF3583 E	Programming change not applied because error occured.
CPF3590 E	Patch already applied temporarily
CPF3591 E	Programming change not removed or applied because program not patched
CPF3596 E	Programming change numbers in select/omit list not valid
CPF3597 E	Prerequisite for programming changes to be applied not met
CPF3598 E	Programming change function already in process.
CPF3612 E	Library &1 not found.
CPF3614 E	Data area &1 not found
CPF3640 E	No immediate PCs applied.
CPF3677 E	Not authorized to library &1.
CPF3931 E	Required programs not found. PC incomplete.
CPF3945 E	Rcds of PC activity for lib &1 deleted.
CPF3956 E	Error occurred during PC processing.
CPF3964 E	Patch permanently removed because patched object not found.
CPF3966 E	Patched program cannot be applied because it is part of PC.

CALL

CPF0006 E	Errors occurred in command.
CPF0805 E	Error found when control language program &1.&2 invoked.

CHGAJE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

CHGBSCF

CPF7304 E	File &1.&2 not changed.

CHGCMD

CPF6209 E	Library &1 not found.
CPF6210 E	Command &1.&2 not found.
CPF6211 E	Not authorized to change command &1.&2.
CPF6212 E	Command &1 not changed because type not *CMD.
CPF6213 E	Unable to allocate command &1.&2.
CPF6214 E	Errors detected while changing command &1.&2
CPF6219 E	Not authorized to library &2.

CHGCMNF

CPF7304 E	File &1.&2 not changed.

CHGCNPA

CPF2704 E	Line &1 not found.
CPF2704 E	Line &1 not found.
CPF3693 E	Service function terminated because error occurred.
CPF3983 E	Period parameter start-time less than current system time.
CPF3984 E	Period attribute not set in system.
CPF3985 E	PERIOD parm end-time and end-date earlier than start-time and start-date.
CPF3986 E	Unable to start CSNAP service function.
CPF3987 E	Period parameter exceeds specified limit.
CPF3988 E	Period and interval parameters exceed specified limits.
CPF3989 E	Interval attribute not set in system.
CPF3990 E	LINE parameter *SAME not valid when no lines previously set in system.
CPF3991 E	LINE parameter contains invalid line(s).
CPF3997 E	No communication lines defined on system.
MCH3401E	Cannot resolve to obj &3. Type/subtype &1 auth &4.

CHGCRDF

CPF7304 E	File &1.&2 not changed.

CHGCUD

CPF2602 E	Control unit &1 not found.
CPF2605 E	Unable to allocate &1.
CPF2610 E	Parameter value not valid.
CPF2612 E	*NONE not valid value for RJEHOST or RJELOGON if ctlu in RJE mode.
CPF2615 E	Ctlu &1 must be offline or powered off for this operation.
CPF2626 E	Line previously deleted.
CPF2627 E	Control unit previously deleted.
CPF2640 E	Command failed because error occurred.
CPF2644 E	Not authorized to change &1.
CPF2656 E	Parameter value not valid.
CPF2657 E	Parameter value not valid.
CPF2658 E	Parameter value not valid.
CPF2672 E	Control unit description &1 not changed. Return code is &2.
CPF2701 E	RMTID entry found following *ANY.

CHGDBG

CPF1999 E	Errors occurred on command.

CHGDEVD

CPF2603 E	Device &1 not found.
CPF2604 E	Printer &1 not found.
CPF2628 E	Device previously deleted.
CPF2629 E	Parameter not valid. Parameter code is &1.
CPF2635 E	Command not executed because device &1 in use.
CPF2640 E	Command failed because error occurred.
CPF2641 E	Device &1 not remote printer.
CPF2644 E	Not authorized to change &1.
CPF2653 E	Printer &1 not attached to same control unit as display station &2.
CPF2697 E	Command timed out and may not have completed.
CPF2745 E	Error type &1 not valid for this device.
CPF2782 E	Message &1 not monitored. Function check occurred.
CPF2799 E	&1 &2.&3 not found.

CHGDKTF

CPF7304 E	File &1.&2 not changed.

CHGDSPF

CPF7304 E	File &1.&2 not changed.

CHGDTA

IDU9001 E	Error found on &1 command.

CHGDTAARA

CPF1015 E	Data area &1.&2 not found.
CPF1018 E	Not authorized to change data area &1.&2.
CPF1019 E	VALUE parameter invalid type for data area &1.&2.
CPF1020 E	VALUE parameter length too long for data area &1.&2.
CPF1021 E	Library &1 not found for data area &2.
CPF1022 E	Not authorized to library &1 for data area &2.
CPF1026 E	VALUE parameter must be '0' or '1' for data area &1.&2.
CPF1062 E	Null string not valid as character value.
CPF1063 E	Unable to allocate data area &1.&2.
CPF1067 E	Unable to allocate library &1.
CPF1087 E	Substring not allowed for decimal or logical data area &1.&2.
CPF1088 E	Starting position outside of data area &1.&2.
CPF1089 E	Invalid substring specified for data area &1.&2.

CHGFCTE

RJE0024 E	Errors found, cannot continue.
RJE0028 E	Form type &1 dev type &2 does not exist in FCT &3.&4.

CHGJOB

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1321 E	Job &1.&2.&3 not found.
CPF1332 E	End of duplicate job names
CPF1336 E	Errors on CHGJOB cmd for job &1.&2.&3.
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1341 E	Reader or writer &1.&2.&3 not allowed as job name.
CPF1343 E	System job &1.&2.&3 not allowed as job name for this command.
CPF1344 E	Not authorized to control job &1.&2.&3.
CPF1351 E	Function check occurred in subsystem for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.

CHGJOBD

CPF1618 E	Job description &1.&2 damaged.
CPF1625 E	Job description &1.&2 not changed.

CHGJOBQE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

CHGJRN

CPF7003 E	Entry not journaled to journal &1.&2, reason &3.
CPF7004 E	Maximum number of objects journaled to journal &1.&2.
CPF7011 E	Storage limit exceeded.
CPF7013 E	Journal receiver not created.
CPF7015 E	Error on JRNRCV specifications.
CPF7018 E	Resetting sequence numbers not allowed.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9804 E	Object &2.&3 type *&5 damaged.
CPF9806 E	Object &2.&3 type *&5 saved with storage freed.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.

CHGLF

CPF7304 E	File &1.&2 not changed.

CHGLFM

CPF3288 E	Member &3 in file &1.&2 not changed.

CHGLIND

CPF2601 E	Line &1 not found.
CPF2605 E	Unable to allocate &1.
CPF2626 E	Line previously deleted.
CPF2633 E	Line description &1 must be offline for this operation.
CPF2644 E	Not authorized to change &1.
CPF2645 E	Invalid parameters on CHGLIND command for &1.
CPF2673 E	Line description &1 not compatible with control unit. Return code is &2.

CHGMSGD

CPF2401 E	Not authorized to library &1.
CPF2407 E	Message file &1.&2 not found.
CPF2411 E	Not authorized to message file &1.&2.
CPF2419 E	Message ID &1 not found in message file &2.&3.
CPF2461 E	Message file &1 not extended because already extended max of &2 times
CPF2483 E	Message file currently in use.
CPF2499 E	Message identifier &1 not valid.
CPF2510 E	Message file &1.&2 logically damaged.
CPF2542 E	Message description not changed for &1.
CPF9830 E	Unable to allocate library &1.

CHGMSGQ

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2405 E	Break pgm &1.&2 not found for msg queue &3.&4.
CPF2406 E	Not authorized to break pgm &1.&2 for msg queue &3.&4.
CPF2408 E	Not authorized to message queue &1.
CPF2446 E	Delivery mode not valid for system log message queue
CPF2450 E	Work station message queue &1 not allocated to job.
CPF2451 E	Message queue &1 not allocated to job.
CPF2467 E	&3 msg queue &1.&2 logically damaged.
CPF2477 E	Message queue &1 currently in use.
CPF2485 E	Nbr parms for break pgm &1.&2 msg queue &3.&4 not valid.
CPF2507 E	MODE(*NOTIFY) not allowed in batch mode.
CPF8127 E	&7 damage on msg queue &4.&8. VLOG-&6.
CPF8176 E	Message queue for logical unit description &4 damaged.

CHGOBJOWN

CPF0601 E	Not authorized to perform operation to file &1.&2.
CPF0605 E	Dev file &1.&2 saved with storage freed or previously deleted.
CPF0608 E	Unable to allocate user profile specified on command.
CPF0609 E	Not authorized to user profile specified on command.
CPF0610 E	Unable to allocate file &1.&2.
CPF2204 E	User profile &1 not found.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF2208 E	Object &1.&3 type *&2 not found.
CPF2209 E	Library &1 not found.
CPF2210 E	Operation not allowed for object type *&1.
CPF2211 E	Unable to allocate object &1.&3 type *&2.
CPF2213 E	Unable to allocate user profile &1.
CPF2216 E	Not authorized to library &1.
CPF2217 E	Not authorized to user profile &1.
CPF2222 E	Storage limit exceeded for user profile &1.
CPF2230 E	Not authorized to object &1.&3 type *&2.
CPF2231 E	Not authorized to use CHGOBJOWN command for program &1.
CPF2232 E	Not authorized to user profile &1.
CPF2233 E	User does not have *DLT authority to owning user profile &1.
CPF3202 E	File &1.&2 in use.
CPF3203 E	Unable to allocate object for file &1.&2.
CPF3245 E	Logical damage of file &1.&2 prevents current operation on file &3.&4.

CHGOUTQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3319 E	Outq &1.&2 active. Nbr of separators unchanged.
CPF3330 E	Unable to allocate necessary resource.
CPF3357 E	Output queue &1.&2 not found.

CHGPF

CPF7304 E	File &1.&2 not changed.

CHGPFM

CPF3288 E	Member &3 in file &1.&2 not changed.

CHGPGMVAR

CPF1924 E	ODV number from symbol table not valid for program &1.
CPF1926 E	Data from obj mapping tbl of template for pgm &1 not valid. Internal failure.
CPF1999 E	Errors occurred on command.

CHGPRTF

CPF7304 E	File &1.&2 not changed.
CPF7308 E	&5 files not changed for &1.&2. &4 files changed.

CHGPTR

CPF1924 E	ODV number from symbol table not valid for program &1.
CPF1926 E	Data from obj mapping tbl of template for pgm &1 not valid. Internal failure.
CPF1999 E	Errors occurred on command.

CHGRJECMNE

RJE0111 E	Cmn entry &1.&2 does not exist in session desc &3.&4.

CHGRJERDRE

RJE0017 E	&1 not found in session desc &2.&3.

CHGRJEWTRE

RJE0017 E	&1 not found in session desc &2.&3.
RJE0067 E	No members found in file &1.&2

CHGRTGE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

CHGSBSD

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1691 E	Active sbsd &1.&2 may or may not be changed.
CPF1697 E	Subsystem description &1.&2 not changed.

CHGSPLFA

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3309 E	File &1 job &3.&4.&5 complete.
CPF3330 E	Unable to allocate necessary resource.
CPF3335 E	File &1 number &2 attributes not changed.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3341 E	File &1 number &2 attributes not changed. File being written.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3344 E	File &1 number &2 already processed.
CPF3492 E	File access limited to owner.

CHGSRCPF

CPF7304 E	File &1.&2 not changed.

CHGSYSVAL

CPF1001 E	Time expired on wait for system response.
CPF1028 E	&1 not valid for parameter SYSVAL.
CPF1030 E	System value &1 cannot be changed.
CPF1058 E	VALUE parameter data type incorrect for sys val &1.
CPF1059 E	VALUE parameter invalid length for system value &1.
CPF1074 E	SYSVAL(QMONTH) not valid for Julian date format.
CPF1076 E	VALUE parameter invalid for system value &1.
CPF1077 E	Device specified for QSYSOPRDEV not display device.
CPF1078 E	System value &1 not changed.
CPF1079 E	Too many values listed for system value &1.

CHGTAPF

CPF7304 E	File &1.&2 not changed.

CHGUSRPRF

CPF2201 E	User password &1 already exists.
CPF2203 E	User profile &1 in error.
CPF2204 E	User profile &1 not found.
CPF2209 E	Library &1 not found.
CPF2213 E	Unable to allocate user profile &1.
CPF2225 E	Unable to allocate internal system object.
CPF2228 E	Not authorized to change user profile.

CHGVAR

CPF0816 E	%SWITCH mask &1 not valid.
CPF0817 E	Decimal variable too small to hold result.
CPF0818 E	Value cannot be converted to type implied by receiver.
CPF0819 E	Variable or substring of variable too small to hold result.
MCH0601E	Space offset &2 is outside current extent for object &1
MCH0603E	Range of subscript value or character string error
MCH1202E	Decimal data error.
MCH1210E	Receiver operand too small to hold result
MCH1211E	Attempt made to divide by zero

CHGWSE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

CHKOBJ

CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9810 E	Library &1 not found.
CPF9815 E	Member &5 in file &2.&3 not found.
CPF9820 E	Not authorized to library &1.
CPF9830 E	Unable to allocate library &1.
CPF9899 E	Error(s) occurred during execution of command.

CLNPRT

CPF3977 E	Device not 3203 printer.
CPF3980 E	System operator replied C to message CPA3975.
CPF9814 E	Device &1 not found.
CPF9825 E	Not authorized to device &1.
CPF9831 E	Unable to allocate device &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

CLRDKT

CPF6156 E	Cancel reply received for &6 file &2.&3 dev &4.
CPF6159 E	Clear diskette terminated because previous error occurred.

CLRJOBQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3307 E	Job queue &1.&2 not found.
CPF3330 E	Unable to allocate necessary resource.

CLRLIB

CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2161 E	Unable to delete some objects in library &1.
CPF2182 E	Not authorized to library &1.
CPF8122 E	&7 damage on library &4. VLOG-&6.

CLROUTQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3330 E	Unable to allocate necessary resource.
CPF3357 E	Output queue &1.&2 not found.

CLRPFM

CPF3130 E	Member &2 file &1.&3 in use.
CPF3133 E	File &1.&3 contains no members.
CPF3136 E	Logical or device file &1.&3 not allowed on command.
CPF3137 E	Not authorized to clear/initialize/copy mbr &2 file &1.&3.
CPF3141 E	Mbr &2 file &1.&3 not found.
CPF3142 E	File &1.&3 not found.
CPF3144 E	Clear or initialize not performed because UPDPROD(*NO).
CPF3156 E	File &1.&3 in use.
CPF3158 E	Operation not performed on mbr &2 file &1.&3.
CPF3159 E	Mbr &2 file &1.&3 saved with STG(*FREE).
CPF3160 E	Operation on mbr &2 file &1.&3 not performed because entry cannot be journaled.

CLRTRCDTA

CPF1999 E	Errors occurred on command.

CNLJOB

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1321 E	Job &1.&2.&3 not found.
CPF1332 E	End of duplicate job names
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1341 E	Reader or writer &1.&2.&3 not allowed as job name.
CPF1342 E	Current job not allowed as job name on this command.
CPF1343 E	System job &1.&2.&3 not allowed as job name for this command.
CPF1344 E	Not authorized to control job &1.&2.&3.
CPF1351 E	Function check occurred in subsystem for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.
CPF1361 E	&1.&2.&3 already canceling with *IMMED option.
CPF1362 E	&1.&2.&3 completed execution.
CPF1363 E	&1.&2.&3 already canceling with *CNTRLD option.
CPF8172 E	Spool control block for job &7.&8.&9 damaged.

CNLRCV

CPF0883 E	Value of *FILE not valid in DEV parameter for file &1
CPF2200 E	Authorization Violations (Generic)
CPF4100 E	Open Exceptions (Generic)
CPF4200 E	Open Exceptions (Generic)
CPF4300 E	Open Exceptions (Generic)
CPF5100 E	I/O Exceptions (Generic)
CPF5200 E	I/O Exceptions (Generic)
CPF5300 E	I/O Exceptions (Generic)
CPF5700 E	File Manipulation Errors (Generic)

CNLRDR

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.
CPF3312 E	Reader &1 not active nor on job queue.
CPF3330 E	Unable to allocate necessary resource.
CPF3490 E	Not authorized to specified reader.

CNLRJERDR

RJE0003 E	RJE session &1 not active.
RJE0004 E	User not authorized to RJE session desc &1 or library &2.
RJE0024 E	Errors found, cannot continue.
RJE0063 E	RJE session &1 already being terminated.
RJE0143 E	No RJE readers defined for this RJE session.
RJE0144 E	RJE device &3 not defined in RJE session &1.
RJE0145 E	Not authorized to library QRECOVERY.
RJE0146 E	Not authorized to message queue &1.QRECOVERY.
RJE0147 E	Message file QRJEMSG not found.

CNLRJEWTR

RJE0003 E	RJE session &1 not active.
RJE0004 E	User not authorized to RJE session desc &1 or library &2.
RJE0004 E	User not authorized to RJE session desc &1 or library &2.
RJE0024 E	Errors found, cannot continue.
RJE0034 E	No writers defined in RJE session &1.
RJE0063 E	RJE session &1 already being terminated.
RJE0145 E	Not authorized to library QRECOVERY.
RJE0146 E	Not authorized to message queue &1.QRECOVERY.
RJE0147 E	Message file QRJEMSG not found.

CNLRQS

CPF1907 E	Last request at level &1 canceled.
CPF1998 E	Cancel to request level &2 not valid at this time.
CPF1999 E	Errors occurred on command.

CNLSPLF

CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3309 E	File &1 job &3.&4.&5 complete.
CPF3330 E	Unable to allocate necessary resource.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3344 E	File &1 number &2 already processed.
CPF3492 E	File access limited to owner.

CNLWTR

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.
CPF3313 E	Writer &1 not active nor on job queue.
CPF3330 E	Unable to allocate necessary resource.
CPF3331 E	Not authorized to control writer &1.&2.&3.
CPF3339 E	Previous cancel to wtr &1.&2.&3 pending.
CPF3438 E	Hold or cancel *PAGEEND invalid for wtr &1.&2.&3.

CPYF

CPF2817 E	Command not executed because error occurred
CPF2850 E	Diskette label not allowed on OVRDKTF command when copying all diskette datasets
CPF2854 E	&2.&3 is externally described printer file.
CPF2857 E	Override of TOMBR with LABEL parm not allowed when copying all mbrs to dkt.
CPF2858 E	Attributes of QSYSPRT not allowed for copy function.
CPF2875 E	FROMFILE &2 must be qualified.
CPF2908 E	Diskette not basic exchange format
CPF2909 E	Mbr &4 file &2.&3 not cleared because mbr not available.
CPF2949 E	Error occurred closing &2.&3. Data may be lost.
CPF2952 E	No data copied because &2.&3 not opened
CPF2968 E	FROMKEY or FROMRCD parameter value is negative value or is past end of file
CPF2970 E	Cannot perform copy with COMPRESS(*NO) to &4 &2.&3.
CPF2971 E	Error on GET from FROMFILE.
CPF2972 E	Error on PUT to TOFILE or printer.
CPF2974 E	&4 &2.&3 has unique index. COMPRESS(*NO) not allowed.
CPF2975 E	File has keyed access path and GET I/O error occured.
CPF2976 E	Maximum number errors (ERRLVL parameter) exceeded.
CPF2985 N	Source seq nbrs for &4 &2.&3 exceeds max allowed.
CPF2999 E	Copy terminated. Printer errors occurred.
CPF3137 E	Not authorized to clear/initialize/copy mbr &2 file &1.&3.
CPF3140 E	Initialize/copy of mbr &2 file &1.&3 canceled.
CPF3143 E	Increments not allowed for mbr &2 in &1.&3.
CPF3148 E	New records require too much space for mbr &2 file &1.&3.
CPF3154 E	UFCB specified for data base copy not valid.

CPYSPLF

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3207 E	Member not added because error occurred.
CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3309 E	File &1 job &3.&4.&5 complete.
CPF3330 E	Unable to allocate necessary resource.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3344 E	File &1 number &2 already processed.
CPF3429 E	Unable to display or copy file &1 nbr &2.
CPF3430 E	No punch data for file &1 number &2.
CPF3482 E	Spool file &1 open. Copy request failed.
CPF3483 E	Invalid file type for file &6.&7. Copy request failed.
CPF3486 E	CHLVAL parameter value not valid.
CPF3492 E	File access limited to owner.
CPF3493 E	CTLCHAR parameter value invalid for spool file &1.
CPF3499 E	Records in file &1 preceded all assigned channel values.
CPF9812 E	File &1.&2 not found.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

CRTBSCF

CPF7302 E	File &1 not created in library &2.

CRTCBLPGM

CBL9001 E	Compile failed. Program not created.

CRTCLPGM

CPF0801 E	Program &1 not created in library &2 because error occurred.
CPF0802 E	Combined IF and DO nest depth exceeds 50.
CPF0807 E	File containing compiler listing not opened
CPF0808 E	Error in compiler-generated code for program resolution monitor. Probable compiler error
CPF0849 E	Space addressing violation with negative offset occurred in CL compiler. Internal failure in system.

CRTCLS

CPF1027 E	Not authorized to library &1 to create class &2.
CPF1039 E	Class library &1 not found.
CPF1064 E	Class &1 already exists in library &2.
CPF1067 E	Unable to allocate library &1.

CRTCMD

CPF0201 E	Command &2 not created in library &3.
CPF0210 E	Unable to open printer file.
CPF0212 E	Unable to open source file.

CRTCMNF

CPF7302 E	File &1 not created in library &2.

CRTCRDF

CPF7302 E	File &1 not created in library &2.

CRTCUD

CPF2612 E	*NONE not valid value for RJEHOST or RJELOGON if ctlu in RJE mode.
CPF2622 E	Valid TELNBR required for switched or SWNBKU control unit description.
CPF2630 E	Address specified does not match address of object.
CPF2640 E	Command failed because error occurred.
CPF2642 E	Valid RMTID required for switched or SWNBKU BSC control unit description.
CPF2650 E	Attempting to create more than &1 &2s attached to &3.
CPF2716 E	Errors occurred on CRTCUD command.

CRTDEVD

CPF2640 E	Command failed because error occurred.
CPF2650 E	Attempting to create more than &1 &2s attached to &3.
CPF2787 E	User message queue &1 exists with same name as work station.

CRTDFUAPP

IDU9001 E	Error found on &1 command.

CRTDKTF

CPF7302 E	File &1 not created in library &2.

CRTDSPF

CPF7302 E	File &1 not created in library &2.

CRTDTAARA

CPF1008 E	Data area &2 not created. Library &1 not available.
CPF1015 E	Data area &1.&2 not found.
CPF1021 E	Library &1 not found for data area &2.
CPF1022 E	Not authorized to library &1 for data area &2.
CPF1023 E	Data area &1 already exists in library &2.
CPF1024 E	TYPE and VALUE parameters not compatible.
CPF1025 E	LEN and VALUE parameters not compatible.
CPF1026 E	VALUE parameter must be '0' or '1' for data area &1.&2.
CPF1047 E	LEN parameter value not valid for data area &1.&2.
CPF1062 E	Null string not valid as character value.
CPF1092 E	Library &1 damaged.

CRTEDTD

CPF9805 E	Object &2.&3 type *&5 destroyed.

CRTFCT

RJE0015 E	Forms control table &1 already exists in lib &2.

CRTJOBD

CPF1621 E	Job description &1.&2 not created.

CRTJOBQ

CPF2182 E	Not authorized to library &1.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3323 E	Job queue &1.&2 already exists.
CPF3351 E	Temporary library &1 invalid for jobq &2.
CPF3354 E	Library &1 not found.
CPF3356 E	Unable to allocate library &1.
CPF3371 E	Spool user profile QSPL damaged or not found.

CRTJRN

CPF7003 E	Entry not journaled to journal &1.&2, reason &3.
CPF7010 E	Object &1.&2 type *&3 already exists.
CPF7011 E	Storage limit exceeded.
CPF7015 E	Error on JRNRCV specifications.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9806 E	Object &2.&3 type *&5 saved with storage freed.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.
CPF9830 E	Unable to allocate library &1.

CRTJRNRCV

CPF7010 E	Object &1.&2 type *&3 already exists.
CPF7011 E	Storage limit exceeded.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.
CPF9830 E	Unable to allocate library &1.

CRTLF

CPF7302 E	File &1 not created in library &2.

CRTLIB

CPF2111 E	Library &1 already exists.

CRTLIND

CPF2606 E	Line name &1 already exists.
CPF2625 E	Unable to allocate &1.
CPF2630 E	Address specified does not match address of object.
CPF2632 E	LINNBR parameter value not valid. Return code is &1.
CPF2637 E	Address contains invalid hexadecimal digits.
CPF2649 E	Line not created. Maximum of 10 lines for specified LINNBR already exist.
CPF2662 E	Parameter not valid. Parameter code is &1.
CPF2673 E	Line description &1 not compatible with control unit. Return code is &2.
CPF2718 E	Errors occurred on CRTLIND command.
CPF2776 E	Invalid data in EXCHID.

CRTMSGF

CPF2112 E	Object &1.&2 of type *&3 already exists.
CPF2113 E	Unable to allocate library &1.
CPF2182 E	Not authorized to library &1.
CPF2402 E	Library &1 not found
CPF2497 E	Size for &1.&2 exceeds machine limit.

CRTMSGQ

CPF2112 E	Object &1.&2 of type *&3 already exists.
CPF2113 E	Unable to allocate library &1.
CPF2182 E	Not authorized to library &1.
CPF2402 E	Library &1 not found
CPF2497 E	Size for &1.&2 exceeds machine limit.

CRTOUTQ

CPF2182 E	Not authorized to library &1.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3352 E	Temporary library &1 invalid for outq &2.
CPF3353 E	Output queue &1.&2 already exists.
CPF3354 E	Library &1 not found.
CPF3356 E	Unable to allocate library &1.
CPF3371 E	Spool user profile QSPL damaged or not found.

CRTPF

CPF7302 E	File &1 not created in library &2.

CRTPRTF

CPF7302 E	File &1 not created in library &2.

CRTPRTIMG

CPF2113 E	Unable to allocate library &1.
CPF2182 E	Not authorized to library &1.
CPF2613 E	Print image &1.&2 already exists.
CPF2621 E	Format of header record for print image source file not correct.
CPF2623 E	Library &1 not found.
CPF2659 E	Belt number &1 not found on IBM SLV Diskettes.
CPF2678 E	Data in source file &1.&2 not valid.
CPF2695 E	Cancel reply received for &1.
CPF2773 E	Invalid device type &2 specified for belt number &1.

CRTQRYAPP

IDU9001 E	Error found on &1 command.

CRTRJECFG

RJE0013 E	RJE configuration for session &1 not created.
RJE0024 E	Errors found, cannot continue.

CRTRPGPGM

QRG9001E	Compile failed. Program not created.

CRTRPTPGM

RPT9001 E	Auto report failed.

CRTSBSD

CPF1696 E	Subsystem description &1.&2 not created.

CRTSRCPF

CPF7302 E	File &1 not created in library &2.

CRTSSND

RJE0001 E	Session desc &1 already exists in lib &2.

CRTTAPF

CPF7302 E	File &1 not created in library &2.

CRTTBL

CPF2614 E	Table &1.&2 already exists.
CPF2623 E	Library &1 not found.
CPF2678 E	Data in source file &1.&2 not valid.

CRTUSRPRF

CPF2201 E	User password &1 already exists.
CPF2202 E	Not authorized to create user profile.
CPF2209 E	Library &1 not found.
CPF2212 E	Unable to allocate library &1.
CPF2213 E	Unable to allocate user profile &1.
CPF2214 E	User profile &1 already exists.
CPF2225 E	Unable to allocate internal system object.

CVTDAT

CPF0550 E	Date passed too short for format specified.
CPF0551 E	Invalid separators in date.
CPF0552 E	Date contains extraneous or misplaced separators.
CPF0553 E	Date contains too many or too few numeric characters for format specified.
CPF0554 E	Variable for converted date too short for format specified.
CPF0555 E	Date not passed in format specified or date not valid.
CPF0556 E	Date passed contains unlike separators.

DATA

CPF1753 E	Command not executable.

DLCOBJ

CPF1005 E	Object(s) not deallocated.

DLTCLS

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted. .

DLTCMD

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTCUD

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2615 E	Ctlu &1 must be offline or powered off for this operation.
CPF2633 E	Line description &1 must be offline for this operation.
CPF2634 E	Not authorized to object &1.
CPF2636 E	Command not executed because control unit &1 in use.
CPF2655 E	Command not executed because control unit &1 not powered off.
CPF2697 E	Command timed out and may not have completed.
CPF2705 E	Device description previously deleted.
CPF2716 E	Errors occurred on CRTCUD command.
CPF2780 E	&1 in diagnostic mode.
CPF2782 E	Message &1 not monitored. Function check occurred.

DLTDEVD

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2603 E	Device &1 not found.
CPF2605 E	Unable to allocate &1.
CPF2615 E	Ctlu &1 must be offline or powered off for this operation.
CPF2616 E	Device description &1 not deleted because device online (in use).
CPF2634 E	Not authorized to object &1.
CPF2640 E	Command failed because error occurred.
CPF2697 E	Command timed out and may not have completed.
CPF2740 E	Device &1 currently powered on.
CPF2780 E	&1 in diagnostic mode.
CPF2782 E	Message &1 not monitored. Function check occurred.

DLTDKTLBL

CPF6155 E	File labeled &1 at diskette location &2 not found
CPF6156 E	Cancel reply received for &6 file &2.&3 dev &4.
CPF6158 E	Delete dkt label terminated because previous error occurred.

DLTDTAARA

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTEDTD

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.
CPF2625 E	Unable to allocate &1.

DLTF

CPF0601 E	Not authorized to perform operation to file &1.&2.
CPF0605 E	Dev file &1.&2 saved with storage freed or previously deleted.
CPF0607 E	File destroyed during command execution.
CPF0610 E	Unable to allocate file &1.&2.
CPF0675 E	Device file &1.&2 currently open.
CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.
CPF3273 E	File or member not deleted or changed because UPDPROD(*NO) specified.

DLTFCT

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTJOBD

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTJOBQ

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2182 E	Not authorized to library &1.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3324 E	Jobq &1.&2 not eligible for deletion.
CPF3330 E	Unable to allocate necessary resource.

DLTJRN

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.
CPF7021 E	Cannot delete journal &1.&2 while files being journaled.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9830 E	Unable to allocate library &1.

DLTJRNRCV

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.
CPF7022 E	Journal receiver &1.&2 cannot be deleted while attached.
CPF7023 E	Cannot delete journal receiver &1.&2 from middle of chain.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.
CPF9830 E	Unable to allocate library &1.

DLTLIB

CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2161 E	Unable to delete some objects in library &1.
CPF2167 E	Library &1 not deleted because library on *LIBL.
CPF2182 E	Not authorized to library &1.

DLTLIND

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2625 E	Unable to allocate &1.
CPF2627 E	Control unit previously deleted.
CPF2633 E	Line description &1 must be offline for this operation.
CPF2634 E	Not authorized to object &1.
CPF2780 E	&1 in diagnostic mode.

DLTMSGF

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTMSGQ

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2182 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2450 E	Work station message queue &1 not allocated to job.
CPF2451 E	Message queue &1 not allocated to job.
CPF2477 E	Message queue &1 currently in use.
CPF2505 E	Deleting work station msg queue not allowed.
CPF9830 E	Unable to allocate library &1.

DLTOUTQ

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2182 E	Not authorized to library &1.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3330 E	Unable to allocate necessary resource.
CPF3360 E	Outq &1.&2 not eligible for deletion.

DLTOVR

CPF9841 E	Override not found at current invocation for &2 &3 &4 &5 &6

DLTPGM

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTPRTIMG

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.
CPF2625 E	Unable to allocate &1.

DLTRJECFG

RJE0024 E	Errors found, cannot continue.
RJE0026 E	RJE configuration for session &1 not completely deleted.
RJE0029 E	RJE configuration for session &1 not found.

DLTSBSD

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTSSND

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.

DLTTBL

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2195 E	Object information lost but object deleted.
CPF2625 E	Unable to allocate &1.

DLTUSRPRF

CPF2204 E	User profile &1 not found.
CPF2213 E	Unable to allocate user profile &1.
CPF2215 E	User profile &1 not deleted because it owns objects.
CPF2225 E	Unable to allocate internal system object.
CPF2227 E	Error(s) during execution of command.
CPF2229 E	Not authorized to delete user profile.

DMPCLPGM

CPF0570 E	Unable to dump CL program &1.&2.

DMPJOB

CPF3546 E	Program parameters specified were not found.
CPF3560 E	Job being serviced not executing.
CPF3585 E	Library name *ALL and invocation level are mutually exclusive.
CPF3909 E	Previous service request not complete.
CPF3918 E	User canceled service request.
CPF3925 E	Unable to open file &1.
CPF3935 E	Job being serviced terminated during dump
CPF3950 E	I/O error message &2 received for file &1. Request terminated.
CPF3951 E	File &1 overridden to invalid file name &2.
CPF3967 E	Dump cannot be taken because serviced job not executing.
CPF3968 E	Dump cannot be taken because serviced job completed execution.
CPF3969 E	Error during close of file &1. Output may be incomplete.

DMPJOBINT

CPF3560 E	Job being serviced not executing.
CPF3636 E	Internal job dump not taken
CPF3909 E	Previous service request not complete.
CPF3918 E	User canceled service request.
CPF3935 E	Job being serviced terminated during dump
CPF3950 E	I/O error message &2 received for file &1. Request terminated.
CPF3967 E	Dump cannot be taken because serviced job not executing.
CPF3968 E	Dump cannot be taken because serviced job completed execution.

DMPOBJ

CPF3560 E	Job being serviced not executing.
CPF3561 E	Context &8 &9 &7 not found.
CPF3562 E	Object &8 &9 &7 not found.
CPF3673 E	Not authorized to library &7.
CPF3909 E	Previous service request not complete.
CPF3918 E	User canceled service request.
CPF3925 E	Unable to open file &1.
CPF3935 E	Job being serviced terminated during dump
CPF3946 E	Machine context damaged.
CPF3947 E	Library &7 not available.
CPF3948 E	Library &3 previously deleted.
CPF3949 E	Library &7 damaged.
CPF3950 E	I/O error message &2 received for file &1. Request terminated.
CPF3951 E	File &1 overridden to invalid file name &2.
CPF3967 E	Dump cannot be taken because serviced job not executing.
CPF3968 E	Dump cannot be taken because serviced job completed execution.
CPF3969 E	Error during close of file &1. Output may be incomplete.

DMPSYSOBJ

CPF3502 E	No objects dumped because no objects found.
CPF3508 E	SUBTYPE (&5) not valid.
CPF3523 E	Starting offset &8 greater than size of space.
CPF3534 E	Not authorized to object &2 type &3 subtype &4 in context &5.
CPF3537 E	Object &2 type &3 subtype &4 in context &5 damaged.
CPF3538 E	Unable to allocate object &2 type &3 subtype &4 in context &5.
CPF3539 E	Object &2 type &4 subtype &5 in context &3 destroyed while object being dumped.
CPF3560 E	Job being serviced not executing.
CPF3561 E	Context &8 &9 &7 not found.
CPF3562 E	Object &8 &9 &7 not found.
CPF3566 E	No objects dumped because no objects found.
CPF3577 E	Data object &7 specified by final pointer not found.
CPF3578 E	Number &9 base data object &7 not found.
CPF3642 E	Address of chain pointer &7 not valid.
CPF3643 E	Address for chain pointer &7 not 16-byte aligned.
CPF3644 E	Base object &7 has no associated space.
CPF3645 E	Not authorized to base object &7.
CPF3646 E	Base object &2 type &3 subtype &4 in context &5 damaged.
CPF3647 E	Base object number &8 or previous base object destroyed.
CPF3648 E	Base object &2 type &3 subtype &4 in context &5 saved with storage freed.
CPF3649 E	Chaining pointer &7 does not exist at location specified.
CPF3650 E	Chaining pointer &7 is instruction pointer.
CPF3651 E	Offset too large for base object &7.
CPF3652 E	Offset to last chaining pointer too large.
CPF3653 E	Location specified for last chaining pointer not 16-byte aligned.
CPF3654 E	Object &2 type &3 subtype &4 in context &5 damaged.
CPF3655 E	Last base object or final object previously deleted.
CPF3656 E	Object &2 type &3 subtype &4 in context &5 saved with storage freed.
CPF3663 E	Base object &7 not found.
CPF3664 E	Object &2 type &3 subtype &4 in context &5 has no associated space.
CPF3665 E	Not authorized to dump object &2 type &3 subtype &4 in context &5.
CPF3666 E	Object &2 type &3 subtype &4 in context &5 damaged.
CPF3667 E	Object to be dumped was destroyed.
CPF3668 E	Object &2 type &3 subtype &4 in context &5 saved with storage freed.
CPF3669 E	Final pointer does not exist at location specified.
CPF3670 E	Final pointer is instruction pointer.
CPF3671 E	Starting offset &8 too large.
CPF3672 E	Object specified by final pointer not found.
CPF3673 E	Not authorized to library &7.
CPF3909 E	Previous service request not complete.
CPF3913 E	Context &7 previously deleted.
CPF3914 E	Context &7 saved with storage freed.
CPF3915 E	Context &7 damaged.
CPF3916 E	Context &7 not available.
CPF3918 E	User canceled service request.
CPF3925 E	Unable to open file &1.
CPF3935 E	Job being serviced terminated during dump
CPF3941 E	CONTEXT(*MCHCTX) and TYPE(&4) are mutually exclusive.
CPF3942 E	CONTEXT(*MCHCTX) and OBJTYPE(*&6) are mutually exclusive.
CPF3946 E	Machine context damaged.
CPF3947 E	Library &7 not available.
CPF3948 E	Library &3 previously deleted.
CPF3949 E	Library &7 damaged.
CPF3950 E	I/O error message &2 received for file &1. Request terminated.
CPF3951 E	File &1 overridden to invalid file name &2.
CPF3967 E	Dump cannot be taken because serviced job not executing.
CPF3968 E	Dump cannot be taken because serviced job completed execution.
CPF3969 E	Error during close of file &1. Output may be incomplete.

DMPTAP

CPF6708 E	Command terminated due to error.
CPF6718 E	Unable to allocate device &1.
CPF6720 E	Incorrect vol &2 found on dev &1. Volume not processed.
CPF6721 E	Device &1 not a tape device.
CPF6723 E	File not found on vol &2 on dev &1 at SEQNBR(&3 &4).
CPF6724 E	File label &5 not found on vol &2 dev &1 at SEQNBR(&3 &4).
CPF6725 E	Ending file seqnbr &4 less than starting seqnbr &3.
CPF6726 E	Ending data block &4 less than starting block &3.
CPF6727 E	Invalid dump TYPE for nonlabeled vol on dev &1.
CPF6728 E	LABEL(&5) invalid for nonlabeled vol on dev &1.
CPF6729 E	Not authorized to file data on vol &2 dev &1.
CPF6730 E	Not authorized to data file seqnbr &3 label &4 on vol &2 dev &1.
CPF6731 E	File label &5 not found on vol &2 dev &1.
CPF6760 E	Device &1 not ready.
CPF6772 E	Device &1 cannot process mounted volume. ERR &2-&3-&4.
CPF9814 E	Device &1 not found.
CPF9825 E	Not authorized to device &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.
CPF9850 E	Override of printer file &1 not valid.

DSNDFUAPP

IDU9001 E	Error found on &1 command.

DSNFMT

SDA0002E	Work station not supported by SDA.
SDA0004E	SDA ended in error.

DSNQRYAPP

IDU9001 E	Error found on &1 command.

DSPACTJOB

CPF1007 E	Override of printer file &1 to file &2 not valid
CPF1093 E	Override of file device type is not valid.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.
CPF9851 E	Overflow value for file &1.&2 too small.

DSPAUTUSR

CPF2225 E	Unable to allocate internal system object.
CPF2237 E	Not authorized to display authorized users.

DSPBKP

CPF1999 E	Errors occurred on command.

DSPCLS

CPF1029 E	Not authorized to library &1 to display class &2.
CPF1039 E	Class library &1 not found.
CPF1065 E	Class &1.&2 not found.
CPF1067 E	Unable to allocate library &1.
CPF1068 E	Unable to allocate class &1.&2.
CPF1098 E	Not authorized to class &1.&2.

DSPCMD

CPF6210 E	Command &1.&2 not found.
CPF6250 E	Cannot display command &1.&2.
CPF6251 E	Forms length of file &1.&2 too small.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9805 E	Object &2.&3 type *&5 destroyed.
CPF9807 E	One or more libraries in library list previously deleted.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.
CPF9824 E	Not authorized to command &1.&2.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.
CPF9850 E	Override of printer file &1 not valid.

DSPCNPA

CPF3986 E	Unable to start CSNAP service function.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

DSPCTLSTS

CPF2601 E	Line &1 not found.
CPF2602 E	Control unit &1 not found.
CPF2603 E	Device &1 not found.
CPF2628 E	Device previously deleted.
CPF2702 E	Device &1 not found.
CPF2703 E	Control unit &1 not found.
CPF2704 E	Line &1 not found.
CPF2777 E	Device description &1 damaged.
CPF2778 E	Control unit description &1 damaged.
CPF2779 E	Line description &1 damaged.

DSPCUD

CPF2602 E	Control unit &1 not found.
CPF2625 E	Unable to allocate &1.
CPF2627 E	Control unit previously deleted.

DSPDBR

CPF3010 E	Only device files present in &1.&2.
CPF3012 E	File &1.&2 not found.
CPF3014 E	File &1.&2 not accessible.
CPF3052 E	File descr for file &1.&2 not valid or not available.
CPF3060 E	Not authorized to create file or command CRTPF not found.
CPF3061 E	Record format &1 not found in file &2.&3.
CPF3062 E	Output file &1.&2 not arrival sequence file.
CPF3063 E	Output file &1.&2 not physical file.
CPF3064 E	Library &1 not found.
CPF3066 E	Error occurred when creating output file &1.&2.
CPF3067 E	Error occurred when opening file &1.&2.
CPF3068 E	Error occurred when writing information to file &1.&2.
CPF3069 E	Error occurred when closing file &1.&2.
CPF3070 E	Error occurred when creating mbr &1 file &2.&3.
CPF3071 E	Mbr names for outfile &1.&2 not available.
CPF3072 E	File &1.&2 is where-used system file.
CPF3074 E	Not authorized to library &1.
CPF3076 E	Error occurred when display attempted.
CPF3084 E	Error occurred when clearing mbr &1 file &2.&3.

DSPDEVD

CPF2603 E	Device &1 not found.
CPF2625 E	Unable to allocate &1.
CPF2628 E	Device previously deleted.
CPF2777 E	Device description &1 damaged.

DSPDEVSTS

CPF2601 E	Line &1 not found.
CPF2602 E	Control unit &1 not found.
CPF2603 E	Device &1 not found.
CPF2628 E	Device previously deleted.
CPF2702 E	Device &1 not found.
CPF2703 E	Control unit &1 not found.
CPF2704 E	Line &1 not found.
CPF2777 E	Device description &1 damaged.
CPF2778 E	Control unit description &1 damaged.
CPF2779 E	Line description &1 damaged.

DSPDKT

CPF3704 E	Display terminated because error occurred.
CPF3738 E	Device &1 damaged.
CPF3743 E	Save/restore file incompatible with system release level.
CPF3769 E	File found on media not save/restore file.
CPF3791 E	End of save/restore file &4 encountered while processing &2 &1.&3.
CPF3793 E	Machine storage limit reached.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3796 E	Storage limit for user profile &4 exceeded.
CPF6017 E	Display diskette terminated because previous error occurred.
CPF9850 E	Override of printer file &1 not valid.

DSPDTA

IDU9001 E	Error found on &1 command.

DSPDTAARA

CPF1015 E	Data area &1.&2 not found.
CPF1016 E	Not authorized to data area &1.&2.
CPF1021 E	Library &1 not found for data area &2.
CPF1022 E	Not authorized to library &1 for data area &2.
CPF1063 E	Unable to allocate data area &1.&2.
CPF1067 E	Unable to allocate library &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.

DSPEDTD

CPF2617 E	Edit code description not found.
CPF2624 E	&1.&2 already deleted.
CPF2625 E	Unable to allocate &1.

DSPFCT

RJE0028 E	Form type &1 dev type &2 does not exist in FCT &3.&4.
RJE0074 E	Forms control table &1.&2 empty.

DSPFD

CPF3011 E	Valid type not specified for file &1.&2.
CPF3012 E	File &1.&2 not found.
CPF3014 E	File &1.&2 not accessible.
CPF3064 E	Library &1 not found.
CPF3067 E	Error occurred when opening file &1.&2.
CPF3068 E	Error occurred when writing information to file &1.&2.
CPF3069 E	Error occurred when closing file &1.&2.
CPF3074 E	Not authorized to library &1.
CPF3076 E	Error occurred when display attempted.

DSPFFD

CPF3012 E	File &1.&2 not found.
CPF3014 E	File &1.&2 not accessible.
CPF3052 E	File descr for file &1.&2 not valid or not available.
CPF3060 E	Not authorized to create file or command CRTPF not found.
CPF3061 E	Record format &1 not found in file &2.&3.
CPF3062 E	Output file &1.&2 not arrival sequence file.
CPF3063 E	Output file &1.&2 not physical file.
CPF3064 E	Library &1 not found.
CPF3066 E	Error occurred when creating output file &1.&2.
CPF3067 E	Error occurred when opening file &1.&2.
CPF3068 E	Error occurred when writing information to file &1.&2.
CPF3069 E	Error occurred when closing file &1.&2.
CPF3070 E	Error occurred when creating mbr &1 file &2.&3.
CPF3071 E	Mbr names for outfile &1.&2 not available.
CPF3072 E	File &1.&2 is where-used system file.
CPF3074 E	Not authorized to library &1.
CPF3076 E	Error occurred when display attempted.
CPF3084 E	Error occurred when clearing mbr &1 file &2.&3.

DSPJOB

CPF0941 E	Job &1.&2.&3 no longer in system.
CPF1007 E	Override of printer file &1 to file &2 not valid
CPF1069 E	End of duplicate names.
CPF1070 E	Job &1.&2.&3 not found.
CPF1071 E	Not authorized to job &1.&2.&3.
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1343 E	System job &1.&2.&3 not allowed as job name for this command.
CPF3330 E	Unable to allocate necessary resource.
CPF3336 E	Job &3.&4.&5 no longer in system.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.
CPF9851 E	Overflow value for file &1.&2 too small.

DSPJOBD

CPF1618 E	Job description &1.&2 damaged.
CPF1623 E	Job description &1.&2 not displayed.
CPF9850 E	Override of printer file &1 not valid.

DSPJOBQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3302 E	Override of print file &1 to file &2 invalid.
CPF3307 E	Job queue &1.&2 not found.
CPF3330 E	Unable to allocate necessary resource.

DSPJRN

CPF7002 E	Object &1.&2 not physical file.
CPF7006 E	Mbr &3 not found in file &1.&2.
CPF7053 E	Error on RCVRNG specifications, reason &1.
CPF7054 E	Invalid FROM and TO values.
CPF7055 E	Maximum number of members exceeded.
CPF7057 E	*LIBL invalid for file library if FILE(*ALL) specified.
CPF7060 E	File &1.&2 never journaled to journal &3.&4.
CPF7061 E	Conversion of journal entries failed. Ending seq nbr &1.
CPF7062 E	No entries converted from journal &1.&2.
CPF7065 E	ENTTYP not valid for specified JRNCDE.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9809 E	Library &1 not accessible.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.
CPF9822 E	Not authorized to file &1.&2.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9850 E	Override of printer file &1 not valid.
CPF9860 E	Error occurred during outfile processing.

DSPJRNA

CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

DSPJRNRCVA

CPF7071 E	Journal receiver &1.&2 not same as requested.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9804 E	Object &2.&3 type *&5 damaged.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to library &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

DSPLIB

CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2150 E	Object information manipulation failed.
CPF2151 E	Object information in library &2 inaccessible.
CPF2176 E	Library &1 damaged.
CPF2179 E	Library(s) not displayed because error occurred.
CPF2182 E	Not authorized to library &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.

DSPLIBL

CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.

DSPLIND

CPF2601 E	Line &1 not found.
CPF2625 E	Unable to allocate &1.
CPF2626 E	Line previously deleted.

DSPLINSTS

CPF2601 E	Line &1 not found.
CPF2602 E	Control unit &1 not found.
CPF2603 E	Device &1 not found.
CPF2628 E	Device previously deleted.
CPF2702 E	Device &1 not found.
CPF2703 E	Control unit &1 not found.
CPF2704 E	Line &1 not found.
CPF2777 E	Device description &1 damaged.
CPF2778 E	Control unit description &1 damaged.
CPF2779 E	Line description &1 damaged.

DSPLOG

CPF2403 E	Message queue &1.&2 not found.
CPF2447 E	No entries exist in current version of log.
CPF2478 E	Not authorized to requested version of log
CPF2480 E	Requested version of log damaged
CPF2499 E	Message identifier &1 not valid.
CPF2519 E	Invalid message ID list.
CPF2537 E	Number of records written by &1 to file &2.&3 exceeds &4.

DSPMSG

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2433 E	Function not allowed for system log message queue &1.
CPF2450 E	Work station message queue &1 not allocated to job.
CPF2451 E	Message queue &1 not allocated to job.
CPF2467 E	&3 msg queue &1.&2 logically damaged.
CPF2477 E	Message queue &1 currently in use.
CPF2537 E	Number of records written by &1 to file &2.&3 exceeds &4.
CPF8127 E	&7 damage on msg queue &4.&8. VLOG-&6.
CPF8176 E	Message queue for logical unit description &4 damaged.
CPF9830 E	Unable to allocate library &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.

DSPMSGD

CPF2401 E	Not authorized to library &1.
CPF2407 E	Message file &1.&2 not found.
CPF2411 E	Not authorized to message file &1.&2.
CPF2483 E	Message file currently in use.
CPF2515 E	Invalid message ID range.
CPF2516 E	Unable to open display file &1.&2.
CPF2519 E	Invalid message ID list.
CPF2537 E	Number of records written by &1 to file &2.&3 exceeds &4.
CPF9807 E	One or more libraries in library list previously deleted.
CPF9810 E	Library &1 not found.
CPF9830 E	Unable to allocate library &1.

DSPMSGF

CPF2401 E	Not authorized to library &1.
CPF2407 E	Message file &1.&2 not found.
CPF2411 E	Not authorized to message file &1.&2.
CPF2483 E	Message file currently in use.
CPF2515 E	Invalid message ID range.
CPF2516 E	Unable to open display file &1.&2.
CPF2520 S	Message used internally by message handler.
CPF2537 E	Number of records written by &1 to file &2.&3 exceeds &4.
CPF9807 E	One or more libraries in library list previously deleted.
CPF9810 E	Library &1 not found.
CPF9830 E	Unable to allocate library &1.

DSPOBJAUT

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF2208 E	Object &1.&3 type *&2 not found.
CPF2209 E	Library &1 not found.
CPF2211 E	Unable to allocate object &1.&3 type *&2.
CPF2216 E	Not authorized to library &1.

DSPOBJD

CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2115 E	Object &1.&2 type *&3 damaged.
CPF2121 E	One or more libraries not accessible.
CPF2123 E	No objects of specified name or type(s) exist in &2.
CPF2124 E	No objects of specified name or types can be displayed from &2.
CPF2150 E	Object information manipulation failed.
CPF2151 E	Object information in library &2 inaccessible.
CPF2176 E	Library &1 damaged.
CPF2177 E	OBJTYPE value not compatible with OBJ value.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF9850 E	Override of printer file &1 not valid.
CPF9860 E	Error occurred during outfile processing.

DSPOBJLCK

CPF1007 E	Override of printer file &1 to file &2 not valid
CPF1093 E	Override of file device type is not valid.
CPF3285 E	Damage encountered on file &1.&3.
CPF9804 E	Object &2.&3 type *&5 damaged.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.
CPF9851 E	Overflow value for file &1.&2 too small.

DSPOUTQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3302 E	Override of print file &1 to file &2 invalid.
CPF3330 E	Unable to allocate necessary resource.
CPF3357 E	Output queue &1.&2 not found.

DSPOVR

CPF9842 E	Override not found at current invocation for file &1.

DSPPGMCHG

CPF3598 E	Programming change function already in process.
CPF3612 E	Library &1 not found.
CPF3677 E	Not authorized to library &1.
CPF3925 E	Unable to open file &1.
CPF3945 E	Rcds of PC activity for lib &1 deleted.
CPF3950 E	I/O error message &2 received for file &1. Request terminated.
CPF3951 E	File &1 overridden to invalid file name &2.
CPF6601 E	No PC activity exists in library &1.
CPF6602 E	PC &1-&2 in library &3 not found.
CPF6603 E	Authorized libs do not contain PC activity for requested pgm products.
CPF6604 E	Unable to obtain access to PCs in library &1.
CPF6605 E	Master Programming Change Index for lib &1 deleted.
CPF9851 E	Overflow value for file &1.&2 too small.

DSPPGMREF

CPF3033 E	Program &1.&2 not found.
CPF3034 E	Program &1.&2 not accessible.
CPF3052 E	File descr for file &1.&2 not valid or not available.
CPF3060 E	Not authorized to create file or command CRTPF not found.
CPF3061 E	Record format &1 not found in file &2.&3.
CPF3062 E	Output file &1.&2 not arrival sequence file.
CPF3063 E	Output file &1.&2 not physical file.
CPF3064 E	Library &1 not found.
CPF3066 E	Error occurred when creating output file &1.&2.
CPF3067 E	Error occurred when opening file &1.&2.
CPF3068 E	Error occurred when writing information to file &1.&2.
CPF3069 E	Error occurred when closing file &1.&2.
CPF3070 E	Error occurred when creating mbr &1 file &2.&3.
CPF3071 E	Mbr names for outfile &1.&2 not available.
CPF3072 E	File &1.&2 is where-used system file.
CPF3074 E	Not authorized to library &1.
CPF3076 E	Error occurred when display attempted.
CPF3084 E	Error occurred when clearing mbr &1 file &2.&3.

DSPPGMVAR

CPF1915 E	ODV not valid for msg mon &2 pgm &1. Internal failure in system.
CPF1924 E	ODV number from symbol table not valid for program &1.
CPF1926 E	Data from obj mapping tbl of template for pgm &1 not valid. Internal failure.
CPF1999 E	Errors occurred on command.

DSPRDR

CPF0941 E	Job &1.&2.&3 no longer in system.
CPF1070 E	Job &1.&2.&3 not found.
CPF1071 E	Not authorized to job &1.&2.&3.
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF3302 E	Override of print file &1 to file &2 invalid.
CPF3312 E	Reader &1 not active nor on job queue.
CPF3330 E	Unable to allocate necessary resource.
CPF3336 E	Job &3.&4.&5 no longer in system.

DSPRJESSN

RJE0003 E	RJE session &1 not active.
RJE0004 E	User not authorized to RJE session desc &1 or library &2.
RJE0126 E	RJE session terminated while DSPRJESSN command active.

DSPSBMJOB

CPF1007 E	Override of printer file &1 to file &2 not valid

DSPSBS

CPF1003 E	Subsystem &1 not active
CPF1007 E	Override of printer file &1 to file &2 not valid

DSPSBSD

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1692 E	Subsystem description &1.&2 not displayed.
CPF9850 E	Override of printer file &1 not valid.

DSPSPLF

CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3308 E	Unable to convert spooled file data for display.
CPF3309 E	File &1 job &3.&4.&5 complete.
CPF3330 E	Unable to allocate necessary resource.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3344 E	File &1 number &2 already processed.
CPF3427 E	Job &3.&4.&5 not interactive job.
CPF3428 E	File &1 number &2 too large to display.
CPF3429 E	Unable to display or copy file &1 nbr &2.
CPF3430 E	No punch data for file &1 number &2.
CPF3492 E	File access limited to owner.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

DSPSPLFA

CPF3302 E	Override of print file &1 to file &2 invalid.
CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3309 E	File &1 job &3.&4.&5 complete.
CPF3330 E	Unable to allocate necessary resource.
CPF3336 E	Job &3.&4.&5 no longer in system.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3344 E	File &1 number &2 already processed.
CPF3492 E	File access limited to owner.

DSPSRVSTS

CPF3520 E	Job not found
CPF3524 E	More than one job with specified name found
CPF3925 E	Unable to open file &1.
CPF3950 E	I/O error message &2 received for file &1. Request terminated.

DSPSYSVAL

CPF1028 E	&1 not valid for parameter SYSVAL.
CPF1074 E	SYSVAL(QMONTH) not valid for Julian date format.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.
CPF9847 E	Error occured while closing file &1.&2.
CPF9850 E	Override of printer file &1 not valid.

DSPTAP

CPF6708 E	Command terminated due to error.
CPF6718 E	Unable to allocate device &1.
CPF6721 E	Device &1 not a tape device.

ENDJOB

CPF1753 E	Command not executable.

ENDJRNPF

CPF7002 E	Object &1.&2 not physical file.
CPF7031 E	Unable to allocate mbr &3 of file &1.&2.
CPF7032 E	File &1.&2 not being journaled.
CPF7033 E	Start or end journaling failed for file &1.&2 mbr &3.
CPF7034 E	Logical damage of file &1.&2.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9810 E	Library &1 not found.
CPF9812 E	File &1.&2 not found.
CPF9820 E	Not authorized to library &1.
CPF9822 E	Not authorized to file &1.&2.

ENDLOG

CPF6905 E	Log entry insert for file &1.&2 mbr &3 failed.
CPF6911 E	Logging not active.
CPF6912 E	Files actively being logged.

ENDSRV

CPF3503 E	Service not ended because trace active
CPF3909 E	Previous service request not complete.
CPF3918 E	User canceled service request.

ENTCBLDBG

CBE7018 E	Program &1 not found.
CBE7019 E	Library &1 not found.

ENTDBG

CPF1999 E	Errors occurred on command.

FMTDTA

FMT9001 E	*** ABNORMAL END *** &1.

FMTRJEDTA

RJE0067 E	No members found in file &1.&2
RJE0095 S	Formatting host file &4 from input file &1.&2 mbr &3.
RJE0109 E	Unrecoverable error occurred during execution of command.
RJE0110 E	Error(s) occurred during execution of command.

GRTOBJAUT

CPF0601 E	Not authorized to perform operation to file &1.&2.
CPF0605 E	Dev file &1.&2 saved with storage freed or previously deleted.
CPF0608 E	Unable to allocate user profile specified on command.
CPF0610 E	Unable to allocate file &1.&2.
CPF2160 E	Object type *&1 not eligible for requested function.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF2208 E	Object &1.&3 type *&2 not found.
CPF2209 E	Library &1 not found.
CPF2210 E	Operation not allowed for object type *&1.
CPF2211 E	Unable to allocate object &1.&3 type *&2.
CPF2216 E	Not authorized to library &1.
CPF2223 E	Not authorized to grant authority to object &1.&3 type *&2.
CPF2227 E	Error(s) during execution of command.
CPF2235 E	AUT parameter value not valid for object type *&1.
CPF2236 E	AUT parameter contains unsupported value.
CPF2245 E	Not owner of object &1.&3 type *&2.
CPF3202 E	File &1.&2 in use.
CPF3203 E	Unable to allocate object for file &1.&2.
CPF3245 E	Logical damage of file &1.&2 prevents current operation on file &3.&4.

GRTUSRAUT

CPF2204 E	User profile &1 not found.
CPF2213 E	Unable to allocate user profile &1.
CPF2217 E	Not authorized to user profile &1.
CPF2252 E	Authorities granted except for those objects listed in previous messages.

HLDJOB

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1321 E	Job &1.&2.&3 not found.
CPF1332 E	End of duplicate job names
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1341 E	Reader or writer &1.&2.&3 not allowed as job name.
CPF1342 E	Current job not allowed as job name on this command.
CPF1343 E	System job &1.&2.&3 not allowed as job name for this command.
CPF1344 E	Not authorized to control job &1.&2.&3.
CPF1345 E	Unable to hold job &1.&2.&3 because job being canceled or SBS terminating.
CPF1346 E	Job &1.&2.&3 already held.
CPF1347 E	Cannot hold job &1.&2.&3.
CPF1348 E	Job &1.&2.&3 held but unable to hold its files.
CPF1350 E	Job &1.&2.&3 only on OUTQ and SPLFILE(*NO) specified.
CPF1351 E	Function check occurred in subsystem for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.

HLDJOBQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3307 E	Job queue &1.&2 not found.
CPF3330 E	Unable to allocate necessary resource.
CPF3425 E	Job queue &1.&2 already held.

HLDOUTQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3330 E	Unable to allocate necessary resource.
CPF3357 E	Output queue &1.&2 not found.
CPF3426 E	Output queue &1.&2 already held.

HLDRDR

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1345 E	Unable to hold job &1.&2.&3 because job being canceled or SBS terminating.
CPF1347 E	Cannot hold job &1.&2.&3.
CPF1350 E	Job &1.&2.&3 only on OUTQ and SPLFILE(*NO) specified.
CPF1351 E	Function check occurred in subsystem for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.
CPF3312 E	Reader &1 not active nor on job queue.
CPF3330 E	Unable to allocate necessary resource.
CPF3333 E	Reader &1.&2.&3 already held.
CPF3490 E	Not authorized to specified reader.

HLDSPLF

CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3309 E	File &1 job &3.&4.&5 complete.
CPF3330 E	Unable to allocate necessary resource.
CPF3337 E	File &1 number &2 already held.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3344 E	File &1 number &2 already processed.
CPF3492 E	File access limited to owner.

HLDWTR

CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF3313 E	Writer &1 not active nor on job queue.
CPF3330 E	Unable to allocate necessary resource.
CPF3331 E	Not authorized to control writer &1.&2.&3.
CPF3332 E	Writer &1.&2.&3 already held.
CPF3334 E	Previous hold to writer &1.&2.&3 pending.
CPF3438 E	Hold or cancel *PAGEEND invalid for wtr &1.&2.&3.

IF

CPF0816 E	%SWITCH mask &1 not valid.
MCH0601E	Space offset &2 is outside current extent for object &1
MCH0603E	Range of subscript value or character string error
MCH1202E	Decimal data error.
MCH1210E	Receiver operand too small to hold result
MCH1211E	Attempt made to divide by zero

INZDKT

CPF6156 E	Cancel reply received for &6 file &2.&3 dev &4.
CPF6716 E	Command terminated because NEWVOL parameter specified more than five characters.
CPF6717 E	Initialize dkt terminated because previous error occurred.
CPF6757 E	Owner identifier &1 contains invalid characters.
CPF6758 E	Volume identifier &1 contains invalid characters.

INZPFM

CPF3130 E	Member &2 file &1.&3 in use.
CPF3131 E	Cannot initialize mbr &2 file &1.&3 at default rcds.
CPF3132 E	TOTRCDS missing or too small for mbr &2 file &1.&3.
CPF3133 E	File &1.&3 contains no members.
CPF3136 E	Logical or device file &1.&3 not allowed on command.
CPF3137 E	Not authorized to clear/initialize/copy mbr &2 file &1.&3.
CPF3140 E	Initialize/copy of mbr &2 file &1.&3 canceled.
CPF3141 E	Mbr &2 file &1.&3 not found.
CPF3142 E	File &1.&3 not found.
CPF3143 E	Increments not allowed for mbr &2 in &1.&3.
CPF3144 E	Clear or initialize not performed because UPDPROD(*NO).
CPF3148 E	New records require too much space for mbr &2 file &1.&3.
CPF3156 E	File &1.&3 in use.
CPF3158 E	Operation not performed on mbr &2 file &1.&3.
CPF3159 E	Mbr &2 file &1.&3 saved with STG(*FREE).
CPF3160 E	Operation on mbr &2 file &1.&3 not performed because entry cannot be journaled.

INZTAP

CPF6702 E	Media error processing vol on dev &1. ERR &2-&3-&4.
CPF6708 E	Command terminated due to error.
CPF6715 E	Media error at beginning of tape on dev &1. ERR &2-&3-&4.
CPF6718 E	Unable to allocate device &1.
CPF6720 E	Incorrect vol &2 found on dev &1. Volume not processed.
CPF6721 E	Device &1 not a tape device.
CPF6722 E	Initialize tape ended due to invalid tape mounted on dev &1.
CPF6754 E	Active file &4 seqnbr &3 found on vol &2 dev &1.
CPF6760 E	Device &1 not ready.
CPF6768 E	Missing write ring on vol on dev &1.
CPF6772 E	Device &1 cannot process mounted volume. ERR &2-&3-&4.
CPF6774 E	NEWVOL(&2) is nonstandard identifier. Volume not initialized.
CPF9814 E	Device &1 not found.
CPF9825 E	Not authorized to device &1.

JOB

CPF1374 E	JOB command not valid in current environment.

JRNPF

CPF7002 E	Object &1.&2 not physical file.
CPF7003 E	Entry not journaled to journal &1.&2, reason &3.
CPF7004 E	Maximum number of objects journaled to journal &1.&2.
CPF7011 E	Storage limit exceeded.
CPF7030 E	File &1.&2 already being journaled.
CPF7031 E	Unable to allocate mbr &3 of file &1.&2.
CPF7033 E	Start or end journaling failed for file &1.&2 mbr &3.
CPF7034 E	Logical damage of file &1.&2.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9810 E	Library &1 not found.
CPF9812 E	File &1.&2 not found.
CPF9820 E	Not authorized to library &1.
CPF9822 E	Not authorized to file &1.&2.

LODPGMCHG

CPF3558 E	Unable to allocate &1.&3 type *&2.
CPF3559 E	Prerequisite engineering changes not made.
CPF3586 E	List of programming changes not valid. Changes not loaded.
CPF3587 E	Prerequisites for programming changes not loaded or supersede requirements not met.
CPF3598 E	Programming change function already in process.
CPF3611 E	Programming changes not loaded because &1 has local temporary patch
CPF3612 E	Library &1 not found.
CPF3614 E	Data area &1 not found
CPF3616 E	All programming changes already loaded.
CPF3619 E	Programming change for release &1, but release &2 is installed.
CPF3657 E	Programming changes not loaded because error occurred
CPF3677 E	Not authorized to library &1.
CPF3924 E	PC not loaded. Some of its objects contained in temporarily applied PC.
CPF3945 E	Rcds of PC activity for lib &1 deleted.
CPF3956 E	Error occurred during PC processing.

LOGDBF

CPF6901 E	Logging already active.
CPF6902 E	Attributes of file &1.&2 mbr &3 invalid for use as LOGFILE.
CPF6904 E	Cannot open log file &1.&2 log mbr &3.
CPF6905 E	Log entry insert for file &1.&2 mbr &3 failed.
CPF6906 E	Not authorized to file &1.&2 mbr &3.

LSTCMDUSG

CPF0578 E	Internal space problems on extension of space.
CPF0579 E	Internal space problems.
CPF0593 E	LSTCMDUSG terminated by controlled cancel.
CPF0595 E	LSTCMDUSG terminated.
CPF0596 E	LSTCMDUSG terminated. Cannot open print file.

LSTCNPDTA

CPF2703 E	Control unit &1 not found.
CPF2704 E	Line &1 not found.
CPF3571 E	Line name &1 not valid.
CPF3690 E	No statistics available for &1.
CPF3926 E	Control unit name &1 not valid.
CPF3985 E	PERIOD parm end-time and end-date earlier than start-time and start-date.
CPF3986 E	Unable to start CSNAP service function.

LSTCNPHST

CPF2703 E	Control unit &1 not found.
CPF2704 E	Line &1 not found.
CPF3571 E	Line name &1 not valid.
CPF3690 E	No statistics available for &1.
CPF3691 E	CMPOBJ parameter contains invalid line or control unit name.
CPF3902 E	PERIOD parameter end-date earlier than start-date.
CPF3903 E	CMPOBJ parm not same type as name specified in LINE or CTLU parm.
CPF3905 E	CMPPERIOD parameter end-date earlier than start-date.
CPF3926 E	Control unit name &1 not valid.
CPF3986 E	Unable to start CSNAP service function.

LSTERRLOG

CPF3535 E	Error log not available for listing.
CPF3593 E	PERIOD parameter start time exceeds end time.
CPF3693 E	Service function terminated because error occurred.
CPF9814 E	Device &1 not found.

LSTINTDTA

CPF3519 E	Unable to start service function
CPF3533 E	Data not listed because error occurred

MOVOBJ

CPF0601 E	Not authorized to perform operation to file &1.&2.
CPF0602 E	File &1 already exists in library &2.
CPF0605 E	Dev file &1.&2 saved with storage freed or previously deleted.
CPF0610 E	Unable to allocate file &1.&2.
CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2112 E	Object &1.&2 of type *&3 already exists.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2150 E	Object information manipulation failed.
CPF2151 E	Object information in library &2 inaccessible.
CPF2160 E	Object type *&1 not eligible for requested function.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2512 E	Operation not allowed for message queue &1.
CPF3201 E	File &1.&2 already exists.
CPF3202 E	File &1.&2 in use.
CPF3203 E	Unable to allocate object for file &1.&2.
CPF3208 E	File &1 same as NEWOBJ name or library &2 same as TOLIB name.
CPF3231 E	Unable to move file &1 from lib &2.
CPF3245 E	Logical damage of file &1.&2 prevents current operation on file &3.&4.
CPF3323 E	Job queue &1.&2 already exists.
CPF3330 E	Unable to allocate necessary resource.
CPF3353 E	Output queue &1.&2 already exists.
CPF3373 E	Job queue &1.&2 not eligible for move.
CPF3374 E	Output queue &1.&2 not eligible for move.
CPF9807 E	One or more libraries in library list previously deleted.
CPF9808 E	Unable to allocate one or more libraries on library list.

PCHPGM

CPF3505 E	Old data at offset &1 does not match old data value.
CPF3506 E	Checksum for offset &1 does not match computed checksum.
CPF3507 E	Offset value &1 not valid.
CPF3514 E	Program not patched because program not found.
CPF3517 E	Hexadecimal digits in offset of PCH parameter not valid.
CPF3525 E	Program not observable. Program cannot be patched.
CPF3527 E	Offset, old-data, or new-data must be even number of hex digits.
CPF3541 E	Offset, old-data, and new-data required for each patch.
CPF3547 E	Offset &1 plus new data length exceeds program segment size.
CPF3555 E	Patched pgm not created. Message &1 issued. Scalar data is &2.
CPF3598 E	Programming change function already in process.
CPF3603 E	Hexadecimal digits in new-data, old-data, or checksum not valid.
CPF3612 E	Library &1 not found.
CPF3660 E	Checksum required for all elements in PCH parameter.
CPF3677 E	Not authorized to library &1.
CPF3907 E	Not authorized to patch program.
CPF3910 E	Program cannot be patched while patch temporarily applied.
CPF3911 E	Program not patched because it is part of temporarily applied PC &1-&2.
CPF3925 E	Unable to open file &1.
CPF3929 E	Program not patched because it is saved with storage freed.
CPF3932 E	Old-data and new-data must be no less than 4 and no more than 32 digits.
CPF3945 E	Rcds of PC activity for lib &1 deleted.
CPF3950 E	I/O error message &2 received for file &1. Request terminated.
CPF3951 E	File &1 overridden to invalid file name &2.
CPF3962 E	Unable to patch program because it is locked.
CPF3970 E	I/O error msg &2 received for file &1. Patch complete, but printout may not be complete.
CPF3971 E	Open error msg &2 received for file &1. Patch complete, but printout may not be complete.
CPF3972 E	Close error message &2 received for file &1.

PRPAPAR

CPF3540 E	Unable to retrieve MTR data.
CPF3545 E	Unable to allocate diskette device.
CPF3553 E	Spooled file &1 job &2 user &3 number &4 not found
CPF3554 E	Unable to allocate spooled file &1 job &2 user &3 number &4
CPF3557 E	Error occurred when spooled file being copied to data base file
CPF3940 E	Error occurred when preparing APAR diskette.
CPF3953 E	Diskette specified contains APAR previously created.
CPF3955 E	Unable to prepare APAR. Library QSRV not allocated.
CPF3960 E	Invalid DEV parameter specified. Must be valid diskette device.
CPF3965 E	Error occurred during APAR processing.
CPF3973 E	File QPCSMPRT containing MTR not found.

PWRCTLU

CPF2640 E	Command failed because error occurred.

PWRDEV

CPF2640 E	Command failed because error occurred.

PWRDWNSYS

CPF1001 E	Time expired on wait for system response.
CPF1036 E	System currently powering down with *CNTRLD option.
CPF1037 E	System currently powering down with *IMMED option.
CPF1038 E	Not authorized to perform function.
CPF1091 E	Function check occurred in system arbiter.

QRYDTA

IDU9001 E	Error found on &1 command.

RCLSTG

CPF8201 E	User profile critical to RCLSTG damaged.
CPF8209 E	System not in proper state to reclaim storage.
CPF8211 E	RCLSTG command terminated because library &1 damaged.
CPF8224 E	Duplicate object encountered during move or rename member.
CPF8251 E	RCLSTG command terminated because library &1 damaged.
CPF8252 E	Error occured during rebuild of damaged library &1.

RCVDTAARA

CPF0875 E	Data area &1 previously declared with different attributes.
CPF0876 E	Not authorized to data area &1
CPF0877 E	Unable to allocate data area &1.
CPF0878 E	Data area &1 previously deleted
CPF0879 E	Library &2 for data area &1 not found.
CPF0880 E	Data area &1 not found
CPF0881 E	Not authorized to data area &1
CPF1022 E	Not authorized to library &1 for data area &2.
CPF1067 E	Unable to allocate library &1.

RCVF

CPF0859 E	File override caused I/O buffer size to be exceeded
CPF0861 E	File &1.&2 not display file.
CPF0883 E	Value of *FILE not valid in DEV parameter for file &1
CPF0886 E	Character data not valid. Specify decimal data.
CPF2200 E	Authorization Violations (Generic)
CPF4100 E	Open Exceptions (Generic)
CPF4200 E	Open Exceptions (Generic)
CPF4300 E	Open Exceptions (Generic)
CPF5100 E	I/O Exceptions (Generic)
CPF5200 E	I/O Exceptions (Generic)
CPF5300 E	I/O Exceptions (Generic)
CPF5700 E	File Manipulation Errors (Generic)

RCVMSG

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2407 E	Message file &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2410 E	Message key not found in message queue &1.
CPF2415 E	End of requests
CPF2419 E	Message ID &1 not found in message file &2.&3.
CPF2423 E	Variable specified in SENDER parameter less than 80 bytes.
CPF2433 E	Function not allowed for system log message queue &1.
CPF2450 E	Work station message queue &1 not allocated to job.
CPF2451 E	Message queue &1 not allocated to job.
CPF2458 E	Message CPF2457 not found in message file QCPFMSG
CPF2471 E	Length of field not valid.
CPF2477 E	Message queue &1 currently in use.
CPF2479 E	No invocation found for pgm msg queue &1. Program not active.
CPF2482 E	Message type &1 not valid. Internal failure in system.
CPF2547 E	Damage on message file QCPFMSG.
CPF2548 E	Damage on message file &1.&2.
CPF2551 E	Invalid MRK/MSGTYPE combination.

RETURN

CPF2415 E	End of requests

RGZPFM

CPF2981 E	Physical file member not reorganized because error occurred
CPF2985 N	Source seq nbrs for &4 &2.&3 exceeds max allowed.
CPF3158 E	Operation not performed on mbr &2 file &1.&3.
CPF3160 E	Operation on mbr &2 file &1.&3 not performed because entry cannot be journaled.

RLSJOB

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1321 E	Job &1.&2.&3 not found.
CPF1332 E	End of duplicate job names
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1341 E	Reader or writer &1.&2.&3 not allowed as job name.
CPF1343 E	System job &1.&2.&3 not allowed as job name for this command.
CPF1344 E	Not authorized to control job &1.&2.&3.
CPF1349 E	Job &1.&2.&3 not released because not held.
CPF1351 E	Function check occurred in subsystem for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.

RLSJOBQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3307 E	Job queue &1.&2 not found.
CPF3330 E	Unable to allocate necessary resource.
CPF3423 E	Jobq &1.&2 not released because not held.

RLSOUTQ

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3330 E	Unable to allocate necessary resource.
CPF3357 E	Output queue &1.&2 not found.
CPF3424 E	Outq &1.&2 not released because not held.

RLSRDR

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1351 E	Function check occurred in subsystem for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.
CPF3312 E	Reader &1 not active nor on job queue.
CPF3315 E	Reader &1.&2.&3 not released because reader not held.
CPF3330 E	Unable to allocate necessary resource.
CPF3490 E	Not authorized to specified reader.

RLSSPLF

CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3309 E	File &1 job &3.&4.&5 complete.
CPF3322 E	File &1 number &2 not released.
CPF3330 E	Unable to allocate necessary resource.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3344 E	File &1 number &2 already processed.
CPF3492 E	File access limited to owner.

RLSWTR

CPF1317 E	No response from subsystem for job &1.&2.&3.
CPF1340 E	Unable to allocate required spooling object for job &1.&2.&3.
CPF1352 E	Function not performed. &1.&2.&3 in transition state.
CPF3313 E	Writer &1 not active nor on job queue.
CPF3316 E	Writer &1.&2.&3 not released because writer not held.
CPF3317 E	OPTION parm value not allowed for diskette writer &1.&2.&3.
CPF3330 E	Unable to allocate necessary resource.
CPF3331 E	Not authorized to control writer &1.&2.&3.
CPF3334 E	Previous hold to writer &1.&2.&3 pending.

RMVAJE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

RMVBKP

CPF1999 E	Errors occurred on command.

RMVFCTE

RJE0024 E	Errors found, cannot continue.
RJE0028 E	Form type &1 dev type &2 does not exist in FCT &3.&4.

RMVJOBQE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

RMVJRNCHG

CPF7002 E	Object &1.&2 not physical file.
CPF7003 E	Entry not journaled to journal &1.&2, reason &3.
CPF7006 E	Mbr &3 not found in file &1.&2.
CPF7007 E	Unable to allocate mbr &3 of file &1.&2.
CPF7041 E	Entry for job &1.&2.&3 not found on RCVRNG.
CPF7042 E	File &1.&2 not being journaled or being journaled through different journal.
CPF7044 E	Apply or remove of journaled entries failed, reason code &7.
CPF7045 E	Partial damage detected on journal receiver &1.&2.
CPF7046 E	Duplicate key not allowed for file &1.&2 mbr &3.
CPF7048 E	Unique access path problems prevent processing journaled change to file &1.&2 mbr &3.
CPF7049 E	Operation cannot be performed across journal entry &6.
CPF7052 E	Select/omit failure on rcd in logical file over file &1.&2 mbr &3.
CPF7053 E	Error on RCVRNG specifications, reason &1.
CPF7054 E	Invalid FROM and TO values.
CPF7055 E	Maximum number of members exceeded.
CPF7056 E	File &1.&2 not being journaled with before-images.
CPF7057 E	*LIBL invalid for file library if FILE(*ALL) specified.
CPF7058 E	Apply or remove of journaled entries failed.
CPF7068 E	Entry required to perform apply or remove not found on journal.
CPF7069 E	No entries applied or removed from journal &1.&2.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9809 E	Library &1 not accessible.
CPF9810 E	Library &1 not found.
CPF9812 E	File &1.&2 not found.
CPF9820 E	Not authorized to library &1.
CPF9822 E	Not authorized to file &1.&2.

RMVM

CPF7310 E	Member &1 not removed from file &2.&3.

RMVMSG

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2410 E	Message key not found in message queue &1.
CPF2433 E	Function not allowed for system log message queue &1.
CPF2450 E	Work station message queue &1 not allocated to job.
CPF2477 E	Message queue &1 currently in use.
CPF2479 E	No invocation found for pgm msg queue &1. Program not active.
CPF8127 E	&7 damage on msg queue &4.&8. VLOG-&6.

RMVMSGD

CPF2401 E	Not authorized to library &1.
CPF2407 E	Message file &1.&2 not found.
CPF2411 E	Not authorized to message file &1.&2.
CPF2419 E	Message ID &1 not found in message file &2.&3.
CPF2483 E	Message file currently in use.
CPF2499 E	Message identifier &1 not valid.
CPF9830 E	Unable to allocate library &1.

RMVPGM

CPF1999 E	Errors occurred on command.

RMVPGMCHG

CPF3551 E	PGMID parameter not valid
CPF3558 E	Unable to allocate &1.&3 type *&2.
CPF3564 E	Programming change &1-&2 damaged.
CPF3591 E	Programming change not removed or applied because program not patched
CPF3596 E	Programming change numbers in select/omit list not valid
CPF3598 E	Programming change function already in process.
CPF3604 E	Programming change not removed because error occurred
CPF3612 E	Library &1 not found.
CPF3614 E	Data area &1 not found
CPF3633 E	Program patch already removed temporarily
CPF3641 E	No immediate PCs removed.
CPF3677 E	Not authorized to library &1.
CPF3931 E	Required programs not found. PC incomplete.
CPF3945 E	Rcds of PC activity for lib &1 deleted.
CPF3956 E	Error occurred during PC processing.
CPF3963 E	Program not removed because it is part of PC.
CPF3964 E	Patch permanently removed because patched object not found.

RMVRJECMNE

RJE0111 E	Cmn entry &1.&2 does not exist in session desc &3.&4.

RMVRJERDRE

RJE0017 E	&1 not found in session desc &2.&3.

RMVRJEWTRE

RJE0017 E	&1 not found in session desc &2.&3.

RMVRTGE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

RMVTRC

CPF1925 E	MI instruction number from BOM or symbol table not valid for pgm &1.
CPF1999 E	Errors occurred on command.

RMVWSE

CPF1619 E	Subsystem description &1.&2 damaged.
CPF1697 E	Subsystem description &1.&2 not changed.

RNMDKT

CPF6153 E	NEWOWNID parameter value not valid.
CPF6154 E	NEWVOL parameter value not valid.
CPF6156 E	Cancel reply received for &6 file &2.&3 dev &4.
CPF6160 E	Rename diskette terminated because previous error occurred.

RNMOBJ

CPF0601 E	Not authorized to perform operation to file &1.&2.
CPF0602 E	File &1 already exists in library &2.
CPF0605 E	Dev file &1.&2 saved with storage freed or previously deleted.
CPF0610 E	Unable to allocate file &1.&2.
CPF2105 E	Object &1.&2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2111 E	Library &1 already exists.
CPF2112 E	Object &1.&2 of type *&3 already exists.
CPF2113 E	Unable to allocate library &1.
CPF2114 E	Unable to allocate object &1.&2 type *&3.
CPF2150 E	Object information manipulation failed.
CPF2151 E	Object information in library &2 inaccessible.
CPF2160 E	Object type *&1 not eligible for requested function.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1.&2 type *&3.
CPF2512 E	Operation not allowed for message queue &1.
CPF3201 E	File &1.&2 already exists.
CPF3202 E	File &1.&2 in use.
CPF3203 E	Unable to allocate object for file &1.&2.
CPF3208 E	File &1 same as NEWOBJ name or library &2 same as TOLIB name.
CPF3245 E	Logical damage of file &1.&2 prevents current operation on file &3.&4.
CPF3323 E	Job queue &1.&2 already exists.
CPF3330 E	Unable to allocate necessary resource.
CPF3353 E	Output queue &1.&2 already exists.
CPF3375 E	Job queue &1.&2 not eligible for rename.
CPF3376 E	Output queue &1.&2 not eligible for rename.
CPF9807 E	One or more libraries in library list previously deleted.
CPF9808 E	Unable to allocate one or more libraries on library list.

RPLLIBL

CPF2184 E	Library list not replaced because error occurred.

RSMBKP

CPF1999 E	Errors occurred on command.

RSTAUT

CPF2206 E	Not authorized to &1.
CPF3776 E	Restored authorities at &3. &2 authorities not restored.
CPF3785 E	System not available for save or restore because other jobs running.
CPF3787 E	RSTAUT command already submitted or RSTUSRPRF command not submitted.

RSTLIB

CPF3705 E	Restore terminated. Journal entry failed for &2 &1.&3.
CPF3706 E	Restore terminated. Saved version of &2 &1.&3 incompatible with current system version.
CPF3727 E	Duplicate device &1 specified in device name list.
CPF3728 E	Diskette device &1 included in multiple device specification.
CPF3730 E	Not authorized to save or restore &2 &1.&3.
CPF3731 E	Unable to allocate &2 &1.&3.
CPF3732 E	Status of &2 &1 changed while restore operation in progress.
CPF3733 E	&2 &1.&3 previously damaged.
CPF3738 E	Device &1 damaged.
CPF3739 E	Member of data base file &1.&3 damaged.
CPF3740 E	&2 &1.&3 not found.
CPF3743 E	Save/restore file incompatible with system release level.
CPF3752 E	No record of save operation exists for &2 &1.
CPF3767 E	Device &1 not found.
CPF3768 E	Device &1 not valid for command.
CPF3769 E	File found on media not save/restore file.
CPF3770 E	No objs saved/restored for lib &1 at &2.
CPF3773 E	Restored &1 objs from &3 to &4, &2 not restored, at &5.
CPF3779 E	Restored &1 libs at &3. &2 not restored.
CPF3780 E	File for lib &1 with specified objs not found.
CPF3781 E	Library &1 not found.
CPF3782 E	Volume &2 found. Volume &1 expected.
CPF3783 E	Unable to determine *SAVVOL location. No objects restored.
CPF3784 E	Restore location (LOC or DEV parameter) does not match *SAVVOL location.
CPF3785 E	System not available for save or restore because other jobs running.
CPF3791 E	End of save/restore file &4 encountered while processing &2 &1.&3.
CPF3793 E	Machine storage limit reached.
CPF3794 E	Save/restore terminated because error occurred.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3796 E	Storage limit for user profile &4 exceeded.

RSTOBJ

CPF3705 E	Restore terminated. Journal entry failed for &2 &1.&3.
CPF3706 E	Restore terminated. Saved version of &2 &1.&3 incompatible with current system version.
CPF3727 E	Duplicate device &1 specified in device name list.
CPF3728 E	Diskette device &1 included in multiple device specification.
CPF3730 E	Not authorized to save or restore &2 &1.&3.
CPF3731 E	Unable to allocate &2 &1.&3.
CPF3733 E	&2 &1.&3 previously damaged.
CPF3738 E	Device &1 damaged.
CPF3739 E	Member of data base file &1.&3 damaged.
CPF3743 E	Save/restore file incompatible with system release level.
CPF3767 E	Device &1 not found.
CPF3768 E	Device &1 not valid for command.
CPF3769 E	File found on media not save/restore file.
CPF3770 E	No objs saved/restored for lib &1 at &2.
CPF3773 E	Restored &1 objs from &3 to &4, &2 not restored, at &5.
CPF3780 E	File for lib &1 with specified objs not found.
CPF3781 E	Library &1 not found.
CPF3782 E	Volume &2 found. Volume &1 expected.
CPF3783 E	Unable to determine *SAVVOL location. No objects restored.
CPF3784 E	Restore location (LOC or DEV parameter) does not match *SAVVOL location.
CPF3791 E	End of save/restore file &4 encountered while processing &2 &1.&3.
CPF3793 E	Machine storage limit reached.
CPF3794 E	Save/restore terminated because error occurred.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3796 E	Storage limit for user profile &4 exceeded.

RSTUSRPRF

CPF2206 E	Not authorized to &1.
CPF3738 E	Device &1 damaged.
CPF3743 E	Save/restore file incompatible with system release level.
CPF3767 E	Device &1 not found.
CPF3768 E	Device &1 not valid for command.
CPF3775 E	Restored &1 user profiles at &3. &2 not restored.
CPF3780 E	File for lib &1 with specified objs not found.
CPF3782 E	Volume &2 found. Volume &1 expected.
CPF3785 E	System not available for save or restore because other jobs running.
CPF3793 E	Machine storage limit reached.
CPF3794 E	Save/restore terminated because error occurred.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3796 E	Storage limit for user profile &4 exceeded.

RTVCLSRC

CPF0560 E	Program &1.&2 not CL program.
CPF0561 E	Unable to retrieve CL source from CL program &2.&3.
CPF0562 E	File &1.&2 not data base source file.
CPF0563 E	Rcd len too small for data base source file &1.&2 mbr &3.
CPF0564 E	Unable to add data base member &3 to file &1.&2.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9806 E	Object &2.&3 type *&5 saved with storage freed.
CPF9807 E	One or more libraries in library list previously deleted.
CPF9808 E	Unable to allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1.&2 not found.
CPF9820 E	Not authorized to library &1.
CPF9821 E	Not authorized to program &1.&2.
CPF9830 E	Unable to allocate library &1.
CPF9848 E	Unable to open file &1.&2 member &3.
CPF9848 E	Unable to open file &1.&2 member &3.
CPF9849 E	Error while processing file &1.&2 member &3.

RTVDFUSRC

IDU9001 E	Error found on &1 command.

RTVDTAARA

CPF0811 E	RTNVAR parameter length too small for data area &1.&2.
CPF0812 E	RTNVAR parameter invalid type for data area &1.&2.
CPF0813 E	Value in data area &1.&2 not logical value.
CPF1015 E	Data area &1.&2 not found.
CPF1016 E	Not authorized to data area &1.&2.
CPF1021 E	Library &1 not found for data area &2.
CPF1022 E	Not authorized to library &1 for data area &2.
CPF1063 E	Unable to allocate data area &1.&2.
CPF1067 E	Unable to allocate library &1.
CPF1087 E	Substring not allowed for decimal or logical data area &1.&2.
CPF1088 E	Starting position outside of data area &1.&2.
CPF1089 E	Invalid substring specified for data area &1.&2.

RTVMSG

CPF2401 E	Not authorized to library &1.
CPF2407 E	Message file &1.&2 not found.
CPF2411 E	Not authorized to message file &1.&2.
CPF2419 E	Message ID &1 not found in message file &2.&3.
CPF2458 E	Message CPF2457 not found in message file QCPFMS.
CPF2465 N	Replacement text of msg &1 in &2.&3 not valid for fmt specified.
CPF2471 E	Length of field not valid.
CPF2499 E	Message identifier &1 not valid.
CPF2547 E	Damage on message file QCPFMSG.
CPF2548 E	Damage on message file &1.&2.

RTVQRYSRC

IDU9001 E	Error found on &1 command.

RTVSYSVAL

CPF1028 E	&1 not valid for parameter SYSVAL.
CPF1074 E	SYSVAL(QMONTH) not valid for Julian date format.
CPF1094 E	CL variable not same type as system value &1.
CPF1095 E	CL variable value length not valid for system value &1.

RVKOBJAUT

CPF0601 E	Not authorized to perform operation to file &1.&2.
CPF0605 E	Dev file &1.&2 saved with storage freed or previously deleted.
CPF0608 E	Unable to allocate user profile specified on command.
CPF0610 E	Unable to allocate file &1.&2.
CPF2160 E	Object type *&1 not eligible for requested function.
CPF2204 E	User profile &1 not found.
CPF2208 E	Object &1.&3 type *&2 not found.
CPF2209 E	Library &1 not found.
CPF2210 E	Operation not allowed for object type *&1.
CPF2211 E	Unable to allocate object &1.&3 type *&2.
CPF2224 E	Not authorized to revoke authority for object &1.&3 type *&2.
CPF2227 E	Error(s) during execution of command.
CPF2235 E	AUT parameter value not valid for object type *&1.
CPF2236 E	AUT parameter contains unsupported value.
CPF3202 E	File &1.&2 in use.
CPF3203 E	Unable to allocate object for file &1.&2.
CPF3245 E	Logical damage of file &1.&2 prevents current operation on file &3.&4.
CPF3381 E	Revoke authority from spool user profile QSPL invalid.

SAVCHGOBJ

CPF3703 E	&2 &1.&3 incompatible with engineering change level on system.
CPF3727 E	Duplicate device &1 specified in device name list.
CPF3728 E	Diskette device &1 included in multiple device specification.
CPF3730 E	Not authorized to save or restore &2 &1.&3.
CPF3731 E	Unable to allocate &2 &1.&3.
CPF3733 E	&2 &1.&3 previously damaged.
CPF3735 E	Storage limit for user profile &1 exceeded.
CPF3738 E	Device &1 damaged.
CPF3745 E	No record of SAVLIB operation exists for &1.
CPF3746 E	Reference date/time &1 later than system date/time.
CPF3767 E	Device &1 not found.
CPF3768 E	Device &1 not valid for command.
CPF3770 E	No objs saved/restored for lib &1 at &2.
CPF3774 E	Saved &1 objs from &3, &2 not saved, &8 excluded.
CPF3781 E	Library &1 not found.
CPF3789 E	Name of volume &1 not valid. Name should be &2.
CPF3790 E	No available space on diskette(s) specified.
CPF3793 E	Machine storage limit reached.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3797 E	Save limit exceeded. No objects saved from library &1.

SAVLIB

CPF3703 E	&2 &1.&3 incompatible with engineering change level on system.
CPF3727 E	Duplicate device &1 specified in device name list.
CPF3728 E	Diskette device &1 included in multiple device specification.
CPF3730 E	Not authorized to save or restore &2 &1.&3.
CPF3731 E	Unable to allocate &2 &1.&3.
CPF3733 E	&2 &1.&3 previously damaged.
CPF3735 E	Storage limit for user profile &1 exceeded.
CPF3738 E	Device &1 damaged.
CPF3767 E	Device &1 not found.
CPF3768 E	Device &1 not valid for command.
CPF3771 E	Saved &1 objects from &3. &2 not saved.
CPF3777 E	Saved &1 libraries at &3. &2 not saved.
CPF3781 E	Library &1 not found.
CPF3785 E	System not available for save or restore because other jobs running.
CPF3789 E	Name of volume &1 not valid. Name should be &2.
CPF3790 E	No available space on diskette(s) specified.
CPF3793 E	Machine storage limit reached.
CPF3794 E	Save/restore terminated because error occurred.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3797 E	Save limit exceeded. No objects saved from library &1.

SAVOBJ

CPF3703 E	&2 &1.&3 incompatible with engineering change level on system.
CPF3727 E	Duplicate device &1 specified in device name list.
CPF3728 E	Diskette device &1 included in multiple device specification.
CPF3730 E	Not authorized to save or restore &2 &1.&3.
CPF3731 E	Unable to allocate &2 &1.&3.
CPF3733 E	&2 &1.&3 previously damaged.
CPF3735 E	Storage limit for user profile &1 exceeded.
CPF3738 E	Device &1 damaged.
CPF3767 E	Device &1 not found.
CPF3768 E	Device &1 not valid for command.
CPF3770 E	No objs saved/restored for lib &1 at &2.
CPF3771 E	Saved &1 objects from &3. &2 not saved.
CPF3781 E	Library &1 not found.
CPF3789 E	Name of volume &1 not valid. Name should be &2.
CPF3790 E	No available space on diskette(s) specified.
CPF3793 E	Machine storage limit reached.
CPF3794 E	Save/restore terminated because error occurred.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3797 E	Save limit exceeded. No objects saved from library &1.

SAVSYS

CPF2206 E	Not authorized to &1.
CPF3703 E	&2 &1.&3 incompatible with engineering change level on system.
CPF3727 E	Duplicate device &1 specified in device name list.
CPF3728 E	Diskette device &1 included in multiple device specification.
CPF3729 E	No diskette device found.
CPF3733 E	&2 &1.&3 previously damaged.
CPF3735 E	Storage limit for user profile &1 exceeded.
CPF3738 E	Device &1 damaged.
CPF3767 E	Device &1 not found.
CPF3768 E	Device &1 not valid for command.
CPF3772 E	SAVSYS incomplete. One or more objects not saved.
CPF3785 E	System not available for save or restore because other jobs running.
CPF3786 E	System not saved because diskette(s) not clear.
CPF3788 E	Error occurred on diskette device.
CPF3789 E	Name of volume &1 not valid. Name should be &2.
CPF3793 E	Machine storage limit reached.
CPF3794 E	Save/restore terminated because error occurred.
CPF3795 E	Error occurred while processing &2 &1.&3. ERR &4-&5-&6.
CPF3797 E	Save limit exceeded. No objects saved from library &1.
CPF3798 E	Installation &2 &1.&3 not found.

SBMCRDJOB

CPF1751 E	Error while processing job &1.&2.&3.
CPF1752 E	Device &1 is incorrect device type.
CPF1760 E	Submit jobs command not allowed in this environment.
CPF1762 S	Reading job &1.&2.&3.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3307 E	Job queue &1.&2 not found.
CPF3363 E	Message queue &1.&2 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9814 E	Device &1 not found.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

SBMDBJOB

CPF1751 E	Error while processing job &1.&2.&3.
CPF1754 E	File &1.&2 not data base file.
CPF1760 E	Submit jobs command not allowed in this environment.
CPF1762 S	Reading job &1.&2.&3.
CPF3307 E	Job queue &1.&2 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9812 E	File &1.&2 not found.
CPF9815 E	Member &5 in file &2.&3 not found.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

SBMDKTJOB

CPF1751 E	Error while processing job &1.&2.&3.
CPF1752 E	Device &1 is incorrect device type.
CPF1755 S	Reading job &1.&2.&3 from vol &4.
CPF1760 E	Submit jobs command not allowed in this environment.
CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3307 E	Job queue &1.&2 not found.
CPF3363 E	Message queue &1.&2 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9814 E	Device &1 not found.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

SBMJOB

CPF1338 E	Errors occurred on SBMJOB command.

SBMRJEJOB

RJE0003 E	RJE session &1 not active.
RJE0017 E	&1 not found in session desc &2.&3.
RJE0041 E	Remote job not submitted.
RJE0067 E	No members found in file &1.&2
RJE0080 E	No batch readers defined in session description &1.&2.
RJE0112 E	RJE reader &1 not available.
RJE0113 E	No RJE readers available.

SNDBRKMSG

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2428 E	Only one message queue allowed for *INQ and *NOTIFY type messages.
CPF2460 E	Message queue &1 not extended because already extended max of &2 times.
CPF2467 E	&3 msg queue &1.&2 logically damaged.
CPF2469 E	Error occurred when sending message&1.
CPF2471 E	Length of field not valid.
CPF2477 E	Message queue &1 currently in use.

SNDDTAARA

CPF0875 E	Data area &1 previously declared with different attributes.
CPF0876 E	Not authorized to data area &1
CPF0877 E	Unable to allocate data area &1.
CPF0878 E	Data area &1 previously deleted
CPF0879 E	Library &2 for data area &1 not found.
CPF0880 E	Data area &1 not found
CPF0881 E	Not authorized to data area &1
CPF1022 E	Not authorized to library &1 for data area &2.
CPF1067 E	Unable to allocate library &1.

SNDF

CPF0859 E	File override caused I/O buffer size to be exceeded
CPF0861 E	File &1.&2 not display file.
CPF0883 E	Value of *FILE not valid in DEV parameter for file &1
CPF2200 E	Authorization Violations (Generic)
CPF4100 E	Open Exceptions (Generic)
CPF4200 E	Open Exceptions (Generic)
CPF4300 E	Open Exceptions (Generic)
CPF5100 E	I/O Exceptions (Generic)
CPF5200 E	I/O Exceptions (Generic)
CPF5300 E	I/O Exceptions (Generic)
CPF5700 E	File Manipulation Errors (Generic)

SNDJRNE

CPF7002 E	Object &1.&2 not physical file.
CPF7003 E	Entry not journaled to journal &1.&2, reason &3.
CPF7037 E	File &1.&2 is not being or was not last journaled to journal &3.&4.
CPF9801 E	Object &2.&3 type *&5 not found.
CPF9802 E	Not authorized to object &2.&3 type *&5.
CPF9803 E	Unable to allocate object &2.&3 type *&5.
CPF9810 E	Library &1 not found.
CPF9812 E	File &1.&2 not found.
CPF9815 E	Member &5 in file &2.&3 not found.
CPF9820 E	Not authorized to library &1.
CPF9822 E	Not authorized to file &1.&2.

SNDMSG

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2428 E	Only one message queue allowed for *INQ and *NOTIFY type messages.
CPF2460 E	Message queue &1 not extended because already extended max of &2 times.
CPF2467 E	&3 msg queue &1.&2 logically damaged.
CPF2469 E	Error occurred when sending message&1.
CPF2471 E	Length of field not valid.
CPF2477 E	Message queue &1 currently in use.
CPF2488 E	RPYMSGQ(*WRKSTN) not valid for batch job.

SNDPGMMSG

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2409 E	Specified message type not valid with specified program message queue.
CPF2428 E	Only one message queue allowed for *INQ and *NOTIFY type messages.
CPF2453 E	Reply queue not sender's program message queue.
CPF2460 E	Message queue &1 not extended because already extended max of &2 times.
CPF2467 E	&3 msg queue &1.&2 logically damaged.
CPF2469 E	Error occurred when sending message&1.
CPF2471 E	Length of field not valid.
CPF2477 E	Message queue &1 currently in use.
CPF2479 E	No invocation found for pgm msg queue &1. Program not active.
CPF2499 E	Message identifier &1 not valid.
CPF2525 E	Status messages can only be sent to program message queues.
CPF2547 E	Damage on message file QCPFMSG.
CPF2548 E	Damage on message file &1.&2.

SNDRCVF

CPF0859 E	File override caused I/O buffer size to be exceeded
CPF0861 E	File &1.&2 not display file.
CPF0883 E	Value of *FILE not valid in DEV parameter for file &1
CPF0886 E	Character data not valid. Specify decimal data.
CPF2200 E	Authorization Violations (Generic)
CPF4100 E	Open Exceptions (Generic)
CPF4200 E	Open Exceptions (Generic)
CPF4300 E	Open Exceptions (Generic)
CPF5100 E	I/O Exceptions (Generic)
CPF5200 E	I/O Exceptions (Generic)
CPF5300 E	I/O Exceptions (Generic)
CPF5700 E	File Manipulation Errors (Generic)

SNDRPY

CPF2401 E	Not authorized to library &1.
CPF2403 E	Message queue &1.&2 not found.
CPF2408 E	Not authorized to message queue &1.
CPF2410 E	Message key not found in message queue &1.
CPF2411 E	Not authorized to message file &1.&2.
CPF2420 E	Reply already sent for inquiry or notify message.
CPF2422 E	Reply not valid
CPF2432 E	Cannot send reply to message type other than *INQ or *NOTIFY
CPF2433 E	Function not allowed for system log message queue &1.
CPF2460 E	Message queue &1 not extended because already extended max of &2 times.
CPF2471 E	Length of field not valid.
CPF2477 E	Message queue &1 currently in use.
CPF2547 E	Damage on message file QCPFMSG.
CPF2548 E	Damage on message file &1.&2.
CPF9830 E	Unable to allocate library &1.

SRVJOB

CPF3501 E	Job already being serviced
CPF3520 E	Job not found
CPF3524 E	More than one job with specified name found
CPF3531 E	Job cannot be serviced because it is your own
CPF3536 E	Job cannot be serviced because job completed execution
CPF3675 E	Unable to allocate QSYS library.
CPF3676 E	Not authorized to service job
CPF3909 E	Previous service request not complete.
CPF3918 E	User canceled service request.
CPF3938 E	Already servicing a job

STRCRDRDR

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3301 E	Reader &1 already started.
CPF3307 E	Job queue &1.&2 not found.
CPF3330 E	Unable to allocate necessary resource.
CPF3347 E	Device &1 not found. Reader or writer not started.
CPF3362 E	Objects in QTEMP invalid for parameter values.
CPF3363 E	Message queue &1.&2 not found.
CPF3366 E	Device &1 not card device.

STRCRDWTR

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3305 E	Outq &1.&2 assigned to another writer.
CPF3310 E	Writer &1 already started.
CPF3330 E	Unable to allocate necessary resource.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3347 E	Device &1 not found. Reader or writer not started.
CPF3357 E	Output queue &1.&2 not found.
CPF3362 E	Objects in QTEMP invalid for parameter values.
CPF3363 E	Message queue &1.&2 not found.
CPF3366 E	Device &1 not card device.
CPF3478 E	File &1 not found in job &3.&4.&5 on outq &6.&7.

STRDBRDR

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3301 E	Reader &1 already started.
CPF3307 E	Job queue &1.&2 not found.
CPF3330 E	Unable to allocate necessary resource.
CPF3362 E	Objects in QTEMP invalid for parameter values.
CPF3363 E	Message queue &1.&2 not found.
CPF3364 E	File &1.&2 not data base file.
CPF9812 E	File &1.&2 not found.
CPF9815 E	Member &5 in file &2.&3 not found.

STRDKTRDR

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3301 E	Reader &1 already started.
CPF3307 E	Job queue &1.&2 not found.
CPF3330 E	Unable to allocate necessary resource.
CPF3347 E	Device &1 not found. Reader or writer not started.
CPF3362 E	Objects in QTEMP invalid for parameter values.
CPF3363 E	Message queue &1.&2 not found.
CPF3367 E	Device &1 not diskette device.

STRDKTWTR

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3305 E	Outq &1.&2 assigned to another writer.
CPF3310 E	Writer &1 already started.
CPF3330 E	Unable to allocate necessary resource.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3347 E	Device &1 not found. Reader or writer not started.
CPF3357 E	Output queue &1.&2 not found.
CPF3362 E	Objects in QTEMP invalid for parameter values.
CPF3363 E	Message queue &1.&2 not found.
CPF3367 E	Device &1 not diskette device.
CPF3478 E	File &1 not found in job &3.&4.&5 on outq &6.&7.

STRPDP

CPF3550 E	PDP message qualifier &3 not valid.
CPF3568 E	PDP identifier &1 not valid.
CPF3569 E	Device name &1 not valid.
CPF3571 E	Line name &1 not valid.
CPF3606 E	Invalid parameter for STRPDP command.
CPF3607 E	Invalid parameter for STRPDP command.
CPF3608 E	Invalid parameter for STRPDP command.
CPF3613 E	LINE parameter required for PDPID.
CPF3615 E	LINE(*ALL) invalid for specified PDPID.
CPF3926 E	Control unit name &1 not valid.

STRPRTWTR

CPF2207 E	Not authorized to object &1.&3 type *&2.
CPF3303 E	File &1 not found in job &3.&4.&5.
CPF3305 E	Outq &1.&2 assigned to another writer.
CPF3310 E	Writer &1 already started.
CPF3330 E	Unable to allocate necessary resource.
CPF3340 E	Multiple files with specified name found in job &3.&4.&5.
CPF3342 E	Job &3.&4.&5 not found.
CPF3343 E	Multiple jobs found with specified name.
CPF3347 E	Device &1 not found. Reader or writer not started.
CPF3357 E	Output queue &1.&2 not found.
CPF3362 E	Objects in QTEMP invalid for parameter values.
CPF3363 E	Message queue &1.&2 not found.
CPF3369 E	Device &1 not printer device.
CPF3478 E	File &1 not found in job &3.&4.&5 on outq &6.&7.

STRRJECSL

RJE0003 E	RJE session &1 not active.

STRRJERDR

RJE0035 E	&1.&2 not data base file.
RJE0041 E	Remote job not submitted.
RJE0059 E	Cannot send host command to complete the request.
RJE0065 E	RJE reader &1 for interactive jobs only.
RJE0067 E	No members found in file &1.&2
RJE0143 E	No RJE readers defined for this RJE session.
RJE0144 E	RJE device &3 not defined in RJE session &1.

STRRJESSN

RJE0012 E	RJE session &1 already active with session desc &1.&2.
RJE0024 E	Errors found, cannot continue.
RJE0045 E	No cmn entries in &1.&2 session desc.

STRRJEWTR

RJE0004 E	User not authorized to RJE session desc &1 or library &2.
RJE0034 E	No writers defined in RJE session &1.
RJE0059 E	Cannot send host command to complete the request.
RJE0144 E	RJE device &3 not defined in RJE session &1.
RJE0147 E	Message file QRJEMSG not found.

STRSBS

CPF1001 E	Time expired on wait for system response.
CPF1004 E	Function check occurred during start subsystem.
CPF1010 E	Subsystem with name of &1 already started.
CPF1011 E	Start subsystem failed for SBSD &1.&2
CPF1012 E	Not authorized to start subsystem description &1.&2.
CPF1013 E	Subsystem description &1.&2 not found.
CPF1014 E	Subsystem description &1 not started because it is in QTEMP lib.
CPF1031 E	Not authorized to library &1.
CPF1049 E	Unable to allocate subsystem description &1.&2.
CPF1050 E	Insufficient storage to start subsystem.
CPF1057 E	Subsystem description &1.&2 damaged.
CPF1067 E	Unable to allocate library &1.
CPF1080 E	Library &1 not found.
CPF1086 E	SBSD &1.&2 allocated to your job.
CPF1099 E	Subsystem not started because system terminating.

TFRJOB

CPF1364 E	Job not transferred. Job queue &1.&2 not active.
CPF1365 E	Job not transferred. Subsystem &1 terminating.
CPF1366 E	SBS &1 has no usable work station entry for &2.
CPF1367 E	User &1 not authorized to subsystem &2
CPF1368 E	&1 not authorized to job queue &2.&3.
CPF1369 E	Job queue &1.&2 not found.
CPF1370 E	Job queue &1.&2 not accessible.
CPF1372 E	Job not transferred. Job currently being canceled.
CPF1373 E	Job not transferred. System request in effect for job.
CPF1375 E	Job not transferred. Single active device not allowed to transfer.

TRCINT

CPF3515 E	Number of trace requests exceeds maximum allowed
CPF3518 E	End time &3 date &4 earlier than start time &1 date &2.
CPF3679 E	Service function returned completion code &1 qualifier &2.
CPF3683 E	Error occurred while attempting to open print file. Completion qualifier is &2.
CPF3684 E	Error occurred while attempting to close print file. Completion qualifier is &2.
CPF3685 E	Error occurred while data being put to print file. Completion qualifier is &2.
CPF3686 E	Service function returned completion code &1 qualifier &2
CPF3687 E	Error occurred while attempting to open diskette file. Qualifier is &2.
CPF3688 E	Error occurred while diskette file being closed. Completion qualifier is &2.
CPF3689 E	Error occurred while writing date to diskette. Qualifier is &2.
CPF3692 E	Service function returned completion code &1 qualifier &2
CPF3693 E	Service function terminated because error occurred.
CPF3694 E	Unable to start service function
CPF3695 E	No trace tables exist.
CPF3696 E	No traces recorded
CPF3697 E	TRCTYPE parameter value missing

TRCJOB

CPF3501 E	Job already being serviced
CPF3510 E	User exit program not found in specified library
CPF3511 E	Trace already active
CPF3512 E	Trace already off
CPF3521 E	Not enough storage for trace table
CPF3542 E	Job not traced because it is being serviced
CPF3548 E	Serviced job completed execution while trace being turned on
CPF3675 E	Unable to allocate QSYS library.
CPF3909 E	Previous service request not complete.
CPF3918 E	User canceled service request.
CPF3925 E	Unable to open file &1.
CPF3936 E	Job being serviced terminated before trace set on
CPF3950 E	I/O error message &2 received for file &1. Request terminated.
CPF3951 E	File &1 overridden to invalid file name &2.
CPF3957 E	Not authorized to exit program library &2.
CPF3958 E	Not authorized to pgm &1.&2.
CPF3969 E	Error during close of file &1. Output may be incomplete.

TRMCPF

CPF1001 E	Time expired on wait for system response.
CPF1017 E	TRMCPF not allowed when console powered or varied off.
CPF1032 E	CPF currently terminating with *CNTRLD option.
CPF1033 E	CPF currently terminating with *IMMED option.
CPF1034 E	All subsystems currently terminating with *CNTRLD option.
CPF1035 E	All subsystems currently terminating with *IMMED option.
CPF1036 E	System currently powering down with *CNTRLD option.
CPF1037 E	System currently powering down with *IMMED option.
CPF1038 E	Not authorized to perform function.
CPF1051 E	Command can only be entered in controlling subsystem.
CPF1082 E	Controlling sbs being terminated to a single jo b (other than console).
CPF1091 E	Function check occurred in system arbiter.

TRMRJESSN

RJE0003 E	RJE session &1 not active.
RJE0063 E	RJE session &1 already being terminated.

TRMSBS

CPF1001 E	Time expired on wait for system response.
CPF1032 E	CPF currently terminating with *CNTRLD option.
CPF1033 E	CPF currently terminating with *IMMED option.
CPF1034 E	All subsystems currently terminating with *CNTRLD option.
CPF1035 E	All subsystems currently terminating with *IMMED option.
CPF1036 E	System currently powering down with *CNTRLD option.
CPF1037 E	System currently powering down with *IMMED option.
CPF1038 E	Not authorized to perform function.
CPF1052 E	TRMSBS *ALL not allowed in current environment.
CPF1053 E	Terminating controlling subsystem &1 not allowed.
CPF1054 E	No subsystem &1 active.
CPF1055 E	Subsystem &1 already terminating with *CNTRLD option.
CPF1056 E	Subsystem &1 already terminating with *IMMED option.
CPF1081 E	Controlling subsystem being terminated to a single job.
CPF1091 E	Function check occurred in system arbiter.

VFYPRT

CPF3943 E	Device parameter not valid.
CPF9814 E	Device &1 not found.
CPF9825 E	Not authorized to device &1.
CPF9831 E	Unable to allocate device &1.
CPF9845 E	Error occurred while opening file &1.&2.
CPF9846 E	Error while processing file &1.&2.

VRYCTLU

CPF2640 E	Command failed because error occurred.

VRYDEV

CPF2640 E	Command failed because error occurred.
CPF2787 E	User message queue &1 exists with same name as work station.

VRYLIN

CPF2640 E	Command failed because error occurred.

WAIT

CPF0859 E	File override caused I/O buffer size to be exceeded
CPF0882 E	No corresponding RCVF or SNDRCVF command for WAIT command
CPF0886 E	Character data not valid. Specify decimal data.
CPF2200 E	Authorization Violations (Generic)
CPF4100 E	Open Exceptions (Generic)
CPF4200 E	Open Exceptions (Generic)
CPF4300 E	Open Exceptions (Generic)
CPF5100 E	I/O Exceptions (Generic)
CPF5200 E	I/O Exceptions (Generic)
CPF5300 E	I/O Exceptions (Generic)
CPF5700 E	File Manipulation Errors (Generic)

Appendix F. Command and Keyword Abbreviations

This appendix contains an alphabetic list of all the abbreviations used in the CL command names and their parameter keywords. The list is in alphabetic order by abbreviation and the associated meaning(s) is given next to the abbreviation.

A	attributes
ACC	access
ACT	active
ADD	add
ADR	address
AJE	autostart job entry
ALC	allocate
ANS	answer
APAR	authorized program analysis report
APP	application
APY	apply
ARA	area
AUT	authority, authorized
AUTO	automatic
BAL	balance
BKP	breakpoint
BKU	backup
BLK	block
BRK	break
BSC	Binary Synchronous Communications
CBL	COBOL
CFG	configuration
CFK	command function key
CHAR	character
CHG	change
CHK	check
CKR	checker
CL	control language
CLN	clean
CLR	clear
CLS	class
CMD	command
CMN	communications
CMNE	communications entry
CMP	compare
CNF	confidence
CNL	cancel
CNN	connection
CNS	constant
COND	condition
CONTIG	contiguous
CPI	characters per inch
CPF	Control Program Facility
CPU	central processing unit
CPY	copy
CRD	card
CRT	create
CSL	console
CSNAP	Communications Statistics Network Analysis Procedure
CTL	control
CTLU	control unit
CTR	controller
CUD	control unit description
CUR	currency
CVT	convert
D	description
DAT	date
DB	data base
DBF	data base file
DBG	debug
DBR	data base relations
DCE	data communication equipment
DCL	declare
DEC	decimal
DEF	definition
DEP	dependent
DEV	device
DFT	default
DFU	Data File Utility
DKT	diskette
DLC	deallocate
DLT	delete
DLVRY	delivery
DLY	delay
DMP	dump
DSN	design
DSP	display
DTA	data
DTR	data terminal ready
DUP	duplicate
DWN	down
E	entry
EDT	edit
ELEM	element

END	end
ENT	enter
ERR	error
EXC	execution
EXCH	exchange
EXEC	execute
EXP	expiration
EXPR	expression
F	file
FA	file attributes
FCT	forms control table
FCTE	forms control table entry
FD	file description
FEAT	feature
FF	file field
FFD	file field description
FI	file interactive
FMT	format
FRAC	fraction
FRC	force
GEN	generate
GRP	group
GRT	grant
HLD	hold
ID	identifier
IMG	image
INC	include
INH	inhibit
INL	initial
INP	input
INT	integer, internal
INV	invocation
INZ	initialize
ITV	interval
J	job
JOB	job, jobs
KBD	keyboard
KWD	keyword
L	list
LBL	label
LCK	lock
LEN	length
LF	logical file
LFM	logical file member
LFT	left
LIB	library
LIN	line
LMT	limit
LOC	location
LOD	load
LOG	log
LPI	lines per inch
LST	list
LVL	level
M	member
MAINT	maintenance
MAX	maximum
MBR	member
MBRS	members
MDM	modem
MI	machine interface
MIN	minimum
MON	monitor
MOV	move
MSG	message
NBR	number
NEG	negative
NONPRDRCV	nonproductive receive
NONRTNZ	nonreturn to zero
OBJ	object
OEM	original equipment manufacturer
OPR	operator
OPT	option
OUT	output
OVR	override
OVRFLW	overflow
OWN	owner
PARM	parameter
PCH	patch
PCT	percent
PDP	problem determination procedure
PF	physical file
PFM	physical file member
PGM	program
PMT	prompt
PNT	point
POS	positive
PRD	productive
PRF	profile
PROD	production
PRP	prepare
PRT	print, printer
PTH	path
PTR	pointer
PTY	priority

PUB	public
PWR	power
Q	queue
QRY	query
QUAL	qualifier
R	relations
RCD	record
RCL	reclaim
RCV	receive
RDR	reader
RDRE	reader entry
REF	references
REL	relation
RGT	right
RGZ	reorganize
RJE	Remote Job Entry Facility
RLS	release
RMV	remove
RNM	rename
RPG	RPG III
RPL	replace, replacement
RPT	report
RPY	reply
RQS	request
RRT	reroute
RSC	resources
RSM	resume
RST	restore
RSTD	restricted
RTG	routing
RTN	return
RTV	retrieve
RTY	retry
RVK	revoke
SAV	save
SBM	submit, submitted
SBS	subsystem
SDLC	synchronous data link control
SEC	second
SEP	separator
SEQ	sequence
SEV	severity
SLR	selector
SNA	systems network architecture
SND	send
SNG	single
SPC	special
SPL	spooled, spooling
SRC	source
SRV	service
SSCP	system services control point
SSN	session
SSND	session description
STG	storage
STMT	statement
STN	station
STR	start
STS	status
SUM	summary for CHKSUM
SWN	switched network
SWS	switches
SYM	symbol
SYNC	synchronous
SYS	system
TAP	tape
TBL	table
TEL	telephone
TFR	transfer
TMP	temporary
TOT	total
TRC	trace
TRM	terminate
TRN	translate
U	unit
UNPRT	unprintable
UPD	update
USG	usage
USR	user
VAL	value
VAR	variable
VFY	verify
VLD	validity
VOL	volume
VRY	vary
VSDR	volume statistical data records
WRK	work
WRT	writer
WRTE	writer entry
WS	work station
WSC	work station controller
WSE	work station entry
WTR	writer
Z	zero

Glossary

abnormal termination: System termination by a means other than the successful execution of the Power Down System (PWRDWNSYS) command. See also *system termination* and *normal termination.*

access path: The means by which CPF provides a logical organization to the data in a data base file so that the data can be processed by a program. See also *arrival sequence access path* and *keyed sequence access path.*

acknowledgment character: (1) In BSC, a transmission control character that is sent as a positive response to a data transmission. (2) In System/38 RJEF, a transmission control character *sequence* that is sent as a positive response to a data transmission.

active file: A diskette file, or tape file whose expiration date is greater than the system date.

activity level: An attribute of a storage pool or the system that specifies the maximum number of jobs that can execute concurrently in the storage pool or in the system.

activity trail: A record of operations that is used to identify what activities have been done, the order in which they were done, and who performed the activities.

add rights: The authority to add an entry to an object. Contrast with *delete rights, read rights,* and *update rights.*

addressability: The ability to locate an object in online storage.

after-image: The image of a record in a physical file member after the data has been modified by a write or an update operation. Contrast with *before-image.*

allocate: To assign a resource for use in performing a specific task. Contrast with *deallocate.*

alphabetic character: (1) Any one of the letters A through Z (uppercase and lowercase) or one of the characters #, $, or @. (2) In COBOL, a character that is one of the 26 uppercase characters of the alphabet, or a space.

alphameric character: Any one of the alphabetic characters, one of the digits 0 through 9, or the character _ (underscore) as defined in CPF.

alternate collating sequence: A collating sequence that differs from the normal collating sequence or that allows two or more characters to be considered equal. See *collating sequence.*

American National Standard Code for Information Interchange: The standard code used for information interchange between data processing systems, data communications systems, and associated equipment. The code uses a coded character set consisting of 7-bit coded characters (8 bits including parity check). The set consists of control characters and graphic characters. Abbreviated ASCII.

APAR: See *authorized program analysis report.*

application: (1) A particular data processing task, such as an inventory control application or a payroll application. (2) In IDU, specialized program created by IDU from user input.

application program: A program used to perform a particular data processing task such as inventory control or payroll.

apply: In journaling, to place after-images of records into a physical file member. The after-images are recorded as entries in a journal. Contrast with *remove.*

arithmetic operator: A symbol that indicates the arithmetic operation to be performed. In CL, the arithmetic operators are + (addition), - (subtraction), * (multiplication), and / (division). In COBOL, the arithmetic operators include: + (addition), - (subtraction), * (multiplication), / (division), and ** (exponentiation).

arrival sequence access path: An access path that is based on the order in which records are stored in a physical file. See also *keyed sequence access path* and *access path.*

ascending key sequence: The arrangement of data in an order from the lowest value of the key field to the highest value of the key field. Contrast with *descending key sequence.*

ascending sequence: The arrangement of data in an order from low to high based on the contents of a specific field or fields. Contrast with *descending sequence.*

ASCII: See *American National Standard Code for Information Interchange.*

attached: Pertaining to a journal receiver that is connected to a journal and is receiving journal entries for that journal. Contrast with *detached.*

attribute: A characteristic; for example, attributes of a field include its length and data type, and attributes of a job include its user name and job date.

authority: The right to access objects, resources, or functions.

authorization: The process of giving a user either complete or restricted access to an object, resource, or function.

authorized program analysis report: A request for correction of a problem caused by a defect in a current unaltered release of a program. Abbreviated APAR.

auto report: A function of the RPG III licensed program that uses simplified specifications and standard RPG specifications to generate a complete RPG source program. See also *auto report program.*

auto report program: A set of instructions (program) that use the RPG auto report function. See also *auto report.*

autoanswer: See *automatic answer.*

autocall: See *automatic call.*

automatic answer: A machine feature that permits a station to respond to a call it receives over a switched line without operator action. Abbreviated autoanswer.

automatic call: A machine feature that permits a station to initiate a connection with another station over a switched line without operator action. Abbreviated autocall.

autostart job: A job that is automatically initiated when a subsystem is started.

autostart job entry: A work entry in a subsystem description that specifies a job to be automatically initiated each time the subsystem is started.

auxiliary storage: All addressable storage other than main storage. Auxiliary storage is located in the system's nonremovable disk enclosures.

back out: To remove changes from a physical file member in the inverse chronological order from which the changes were originally made.

base storage pool: A storage pool that contains all unassigned main storage on the system and whose minimum size is specified in the system value QBASPOOL. The system-recognized identifier is *BASE.

basic data exchange: A format for exchanging data on diskettes between systems or devices.

basic working display: The display that serves as the base from which you make requests of the system at a work station. When the request is completed, you return to the display. It is usually the display you receive when you sign on.

batch device: Any device that can read serial input or write serial output, or both, but cannot communicate interactively with the system. Examples of batch devices are card devices, printers, and diskette units.

batch job: A group of processing actions submitted as a predefined series of actions to be performed with little or no interaction between the user and the system.

batch processing: A method of executing a program or a series of programs in which one or more records (a batch) is processed with little or no interaction with the user or operator. Contrast with *interactive processing.*

batch subsystem: A subsystem in which batch jobs are to be processed. IBM supplies one batch subsystem: QBATCH.

before-image: The image of a record in a physical file member before the data has been modified by a write, an update, or a delete operation. Contrast with *after-image*.

binary: Relating to, being, or belonging to a numbering system with a base of 2. Valid digits are 0 and 1.

binary format: Representation of a decimal value in which each field must be 2 or 4 bytes long. The sign (+ or -) is in the leftmost bit of the field, and the integer value is in the remaining bits of the field. Positive numbers have a 0 in the sign bit and are in true form. Negative numbers have a 1 in the sign bit and are in twos complement form.

binary synchronous communications: A form of communications line control that uses transmission control characters to control the transfer of data over a communications line. Abbreviated BSC. Contrast with *Synchronous Data Link Control.*

block: (1) A set of adjacent logical records recorded as a unit on a diskette or magnetic tape. (2) In COBOL, a unit of data that is moved into or out of the computer.

break delivery: The method of delivering messages to a message queue in which the job associated with that message queue is interrupted as soon as the message arrives.

breakpoint: A place in a program (specified by a command or a condition) where the system halts execution and gives control to the work station user or to a specified program.

breakpoint program: For a batch job, a user program that can be invoked when a breakpoint is reached.

BSC: See *binary synchronous communications.*

BSC file: A device file created by the user to support BSC. Contrast with *communications file.*

byte: A group of eight adjacent binary digits that represents one EBCDIC character.

call: (1) To instruct that a program is to begin execution. (2) An instruction to a program to begin execution. (3) In data communications, the action necessary to make a connection between two stations on a switched line.

called program: A program whose execution is requested by another program (a calling program) or by a command.

calling program: A program that requests the execution of another program (a called program).

card file: A device file created by the user to support a card device.

CF key: See *command function key.*

character: Any letter, digit, or other symbol in the data character set that is part of the organization, control, or representation of data.

character field: An area that is reserved for a particular unit of information and that can contain any of the characters in the data character set. Contrast with *numeric field.*

character literal: A symbol, quantity, or constant in a source program that is itself data, instead of a reference to data. Contrast with *numeric literal.*

character string: (1) A string consisting of any of the 256 EBCDIC characters that are used as a value. (2) (ANSI) In COBOL, a sequence of characters that form a COBOL word, a literal, a PICTURE character-string, or a comment-entry.

CL: See *control language.*

class: An object that contains the execution parameters for a routing step. The system-recognized identifier for the object type is *CLS.

close: A data manipulation function that ends the connection between a file and a program. Contrast with *open.*

CNP: See *communications statistical network analysis procedure.*

command: (1) A statement used to request a function of the system. A command consists of the command name, which identifies the requested function, and parameters. (2) In SNA, any field set in the transmission header (TH), request header (RH), and sometimes portions of a request unit that initiates an action or that begins a protocol.

command analyzer: An IBM-supplied program that processes commands. Command processing includes validity checking, transferring control to a command processing program (CPP), and returning to the caller of the command analyzer.

command definition: An object that contains the definition of a command (including the command name, parameter definitions, and validity checking information) and identifies the program that performs the function requested by the command. The system-recognized identifier for the object type is *CMD.

command definition statement: A source statement used in creating a command definition. Command definition statements define keywords and parameter values, qualified names, elements in a list, parameter dependencies and interrelationships, and prompt text for a command.

command file: In System/38 RJEF, a remote job input stream that can contain host system commands and job control language (JCL), data, and RJEF control statements (READFILE or EOF). Contrast with *data file.*

command function key: At a work station, a keyboard key that is used with the command (CMD) function control key to request preassigned functions. At the system console, a keyboard key, called a CF key, that is used to request preassigned functions.

command level: Pertaining to an operation that is performed for a specific command in a program. For example, a Monitor Message (MONMSG) command that immediately follows a specific command in a CL program is a command-level MONMSG command. Contrast with *program level.*

command processing program: A program that processes a command. This program performs some validity checking and executes the command so that the requested function is performed. Abbreviated CPP.

comment: A word or statement in a program, command, or file that serves as documentation instead of as instructions. A comment is ignored by a compiler.

communications adapter: A hardware feature that enables System/38 to become part of a data communications network.

communications file: A device file created by the user to support LU1 SDLC communications. Contrast with *BSC file.*

communications line: The physical link (such as a wire or a telephone circuit) that connects one or more work stations to a communications control unit, or connects one control unit to another. Contrast with *data link.*

communications statistical network analysis procedure: A procedure that allows the service personnel to obtain statistics on communications line activity. Abbreviated CSNAP or CNP.

compile: To translate a source program into an executable program (an object).

compile time: The time during which a source program is translated by a compiler into an executable program.

compiler: A program that translates a source program into an executable program.

compiler listing: A printout that is produced by compiling a program or creating a file and that optionally includes, for example, a line-by-line source listing, a cross-reference list, diagnostic information, and for programs, the description of externally described files.

completion message: A message that conveys completion status of work.

compress: To save storage space by eliminating gaps, empty fields, redundancies, or unnecessary data to shorten the length of records or files.

concatenated field: Two or more fields from a physical file record format that have been combined to make one field in a logical file record format.

constant: Data that has an unchanging, predefined value to be used in processing. A constant does not change during the execution of a program, but the contents of a field or variable can. See also *literal.*

constant field: In an externally described display or printer file, an unnamed field that contains actual data that is passed to the display or printer but is unknown to the program passing it.

contention state: In data communications, a type of half-duplex line or link control in which either user may transmit any time the line/link is available. In the event that both users attempt to transmit a request simultaneously, the protocols or the hardware determines who wins the contention.

control language: The set of all commands with which a user requests functions. Abbreviated CL.

control language program: An executable object that is created from source consisting entirely of control language commands.

control language variable: A program variable that is declared in a control language program and is available only to the program.

Control Program Facility: The system support licensed program for System/38. It provides many functions that are fully integrated in the system such as work management, data base data management, job control, message handling, security, programming aids, and service. Abbreviated CPF.

control station: The primary or controlling system on a multipoint line. The control station controls the sending and receiving of data.

control unit: Circuitry or a device that coordinates and controls the operation of one or more input/output devices (such as work stations) and synchronizes the operation of such devices with the operation of the system as a whole. Same as *controller*. Abbreviated CTL or CTLU.

control unit description: An object that contains a description of the features of a control unit that is either directly attached to the system or attached to a communications line. The system-recognized identifier for the object type is *CUD. Abbreviated CUD.

controller: See *control unit*.

controlling subsystem: An interactive subsystem that is started automatically when the system is started and through which the system operator controls the system. IBM supplies one controlling subsystem: QCTL.

converted journal entry: The version of a journal entry that can be displayed, printed, or written to a data base output file.

CPF: See *Control Program Facility*.

CPP: See *command processing program*.

CPU: Central processing unit. See *processor*.

create: (1) The function used to bring an object into existence in the system. (2) To bring an object into existence in the system.

cross-reference listing: The portion of the compiler listing that contains information on where files, fields, and indicators are defined, referenced, and modified in a program.

CSNAP: See *communications statistical network analysis procedure*.

CTL: See *control unit*.

CTLU: See *control unit*.

CUD: See *control unit description*.

customer engineer user profile: The CPF-supplied user profile that has the authority necessary for the customer engineer to perform diagnostics and service the machine, and the special authority of job control rights. Named QCE.

data area: An object that is used to communicate data such as CL variable values between the programs within a job and between jobs. The system-recognized identifier for the object type is *DTAARA.

data base: The collection of all data base files stored in the system.

data base file: An object that contains descriptions of how input data is to be presented to a program from internal storage and how output data is to be presented to internal storage from a program. See also *physical file* and *logical file*.

data base logging: A method of data base recovery in which changes to records in a data base file are also written to a data base log file.

data communications: The transmission of data between systems and/or remote devices over a communications line.

data description specifications: A description of the user's data base or device files that is entered into the system using a fixed-form syntax. The description is then used to create files. Abbreviated DDS.

data file: (1) Any nonsource file. A data file is created by the specification of FILETYPE(*DATA) on a create file command. (2) In System/38 RJEF, a remote job input stream that can contain host system commands and job control language (JCL) as well as data. Contrast with *command file*.

data file utility: The utility of the Interactive Data Base Utilities licensed program that is used to create, maintain, and display records in a data base file. Abbreviated DFU.

data link: The communications lines, modems, control units, work stations, and other communications equipment used for the transmission of data between a receiving station and a transmitting station in a data network. Contrast with *communications line*.

data rights: The authority to read, add, update (modify), or delete data contained in an object. See also *add rights*, *delete rights*, *read rights*, and *update rights*.

data stream: (1) In BSC, all data transmitted over a data link in a single read or write operation. (2) For System/38 RJEF, see *input stream* and *output stream*.

data type: An attribute used for defining data as numeric or character.

DDS: See *data description specifications*.

deallocate: To release a resource that is assigned to a specific task. Contrast with *allocate*.

debug mode: An environment in which programs can be tested.

default delivery: The method of delivering messages to a message queue in which messages are placed on the queue without interrupting the job, and the default reply is sent for any messages requiring a reply.

default program: A user-specified program that is assumed when no other program is specifically named on a debug command, or a special program defined for handling error messages.

default record: A record in which numeric fields are initialized with zeros and character fields are initialized with blanks.

default reply: A system-assigned reply to an inquiry or notify message that is used when the message queue at which the message arrives is in default delivery mode.

default user name: A CPF-provided name for user identification for an installation that does not want to require separate user identifications.

default value: A value assumed when no value has been specified.

delay maintenance: A method of maintaining keyed access paths for data base files. This method does not update an access path when the file is closed, but it retains updates in a *delayed* form so that they can be quickly applied at the next open, avoiding a complete rebuild. Contrast with *rebuild maintenance* and *immediate maintenance*.

delete: (1) To remove an object or a unit of data (such as character, a field, or a record). (2) The SEU operation in which existing records can be removed from a source member.

delete rights: The authority to delete an entry from an object or to delete the object itself. Contrast with *add rights*, *read rights*, and *update rights*.

deleted record: A record that has been initialized or removed so that it is not eligible for access. A deleted record holds a place in a physical file member and can be replaced with a data record by an update operation.

delimiter: A character or a sequence of contiguous characters that identifies the end of a string of characters. A delimiter separates a string of characters from the following string of characters. A delimiter is not part of the string of characters that it delimits.

descending key: The values by which data is ordered from the highest value to the lowest value of the key, in accordance with the rules for comparing data items. Contrast with *ascending key*.

descending key sequence: The arrangement of data in order from the highest value of the key field to the lowest value of the key field. Contrast with *ascending key sequence*.

descending sequence: The arrangement of data in an order from high to low based on the contents of a specific field. Contrast with *ascending sequence*.

detached: Pertaining to a journal receiver that is not connected to a journal and is not receiving journal entries for that journal. Contrast with *attached*.

DEVD: See *device description*.

device description: An object that contains information describing a particular device that is attached to the system. The system-recognized identifier for the object type is *DEVD. Abbreviated DEVD.

device file: An object that contains a description of how input data is to be presented to a program from an external device and/or how output data is to be presented to the external device from the program. External devices can be work stations, card devices, printers, diskette magazine drives, magazine tape drives, or a communications line.

device name: The symbolic name of an individual device. The name is specified when the device is defined to the system by the Create Device Description (CRTDEVD) command.

device type: The generic name for a group of device names. All the devices in a device type must have the same physical attributes and functions.

DFU: See *data file utility.*

DFU application: See *application.*

diagnostic message: A message that contains information about errors in the execution of an application program or a system function.

digit: Any of the numerals from 0 through 9.

disconnect time-out: A time-out that indicates a loss of communications with the BSC device or work station.

diskette drive: The mechanism used to seek, read, and write data on diskettes. See also *diskette magazine drive.*

diskette file: A device file created by the user to support a diskette device.

diskette location: The slot into which the diskette is inserted before being read or written.

diskette magazine drive: A diskette drive that can hold two magazines, each containing 10 diskettes, plus individual diskettes in three separate slots. It is used to transfer information between system internal storage and removable diskettes.

display: A visual presentation of information on a work station screen, usually in a specific format. Display is often used as a shortened version of information display. Sometimes called a screen.

display file: A device file created by the user to support a display work station or console.

display format: The name of the device file and the name of the record format to be used when the subsystem obtains routing data from the user.

dump: To copy data in a readable format from main or auxiliary storage onto an external medium such as tape, diskette, or printer.

duplicate key value: The occurrence of the same value in a key field or in a composite key in more than one record in a file.

EBCDIC: Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

EC log: See *engineering change log.*

edit: (1) To modify a numeric field for output by suppressing zeros and inserting commas, periods, currency symbols, the sign status, or other constant information. (2) The process of using SEU to key in new source records and update existing source records in a source member.

edit code: A letter or number indicating what kind of editing should be done before a field is displayed or printed.

edit description: An object that contains a description of a user-defined edit code. The system-recognized identifier for the object type is *EDTD.

edit display: The SEU display from which frequently performed operations, such as delete, copy, and insert, are requested.

edit word: A user-defined word with a specific format that indicates how editing should be done.

element: (1) A parameter value in a list of parameter values. (2) In RPG, the smallest addressable unit of an array or table.

embedded blank: A blank that appears between characters.

embedded command: A command that is specified as a value in a parameter of another command.

end-of-transmission character: In BSC, the transmission control character that is usually used to end transmission with the remote system. Abbreviated EOT.

engineering change log: A list of engineering changes that have been installed in the machine; the list is useful for preparing APARs. Abbreviated EC log.

error log: A record of machine checks, device errors, and volume statistical data.

escape message: A message that can be monitored for and that describes a condition for which a program terminates without completing the requested function.

exclusive-allow-read lock state: The allocation that a routing step has for an object in which other routing steps can read the object if they request a shared for read lock state for the same object. The predefined value for this lock state is *EXCLRD.

exclusive lock state: The allocation that a routing step has for an object in which no other routing steps can use the object. The predefined value for this lock state is *EXCL.

executable program: The set of machine language instructions that is the output from the compilation of a source program. The actual processing of data is done by the executable program.

execute: To cause a program, command, utility, or other machine function to be performed.

execution: The carrying out of the instructions of a computer program by a processing unit.

execution time: The time during which the instructions of a computer program are executed by a processing unit.

external message queue: A message queue that is part of the job message queue and is used to send messages between an interactive job and the work station user. For batch jobs, messages sent to the external message queue only appear in the job log.

external storage: Data storage other than main or auxiliary storage.

externally described data: Data contained in a file for which the fields in the records are described to CPF, by using data description specifications, when the file is created. The field descriptions can be used by the program when the file is processed. Contrast with *program-described data.*

externally described file: A file for which the fields in the records are described to CPF, through data description specifications, when the file is created. The field descriptions can be used by the program when the file is processed. Contrast with *program-described file.*

FCFC: See *first-character forms control.*

FCS: See *function control sequence.*

FCT: See *forms control table.*

field: An area that is reserved and used for a particular item of information.

field reference file: A physical file that contains no members and whose record format describes the fields used by a group of files.

file: A generic term for the object type that refers to a data base file, a device file, or a set of related records treated as a unit. The system-recognized identifier for the object type is *FILE.

file description: (1) The information contained in the file that describes the file and its contents. (2) In COBOL, an entry in the File Section of the Data Division that provides information about the identification and physical structure of a file.

file overrides: The file attributes specified at execution time that will override the attributes specified in the file description or in the program.

file reference function: A CPF function that lets the user track file usage on the system.

file separator: The pages or cards to be produced at the beginning of each output file to separate the file from the other files being spooled to an output device.

first-character forms control: A method for controlling the format of printed output. The first character of each record determines the format. Abbreviated FCFC.

first-level message: The initial message that is presented to the user. The initial message contains general information or designates an error. Contrast with *second-level message.*

fold: To continue data for a line on the following printed or displayed line. Contrast with *truncate.*

form: The area between perforations on continuous printer paper.

forms control table: An object that designates the special processing requirements for specific printer or punch output streams received by an RJEF session from a host system. The system-recognized identifier for the object type is *FCT. Abbreviated FCT.

format selector: A user-defined program (either a CL or an HLL program) that determines where a record should be placed in the data base when an application program does not pass a record name for a record being added to a logical file member.

forward recovery: The process of reconstructing a file from a particular point by restoring a saved version of the file and then applying changes to that file in the same order in which they were originally made.

function check: A notification (by a message) that an unexpected condition has stopped the execution of a program.

function control sequence: In System/38 (RJEF) MTAM, a control character used to control the flow of individual function streams. Abbreviated FCS.

function key: A keyboard key that is used to request a specific system function. See also *command function key.*

general-purpose library: The library provided by CPF to contain user-oriented, IBM-provided objects and user-created objects that are not explicitly placed in a different library when they are created. Named QGPL.

generic name: The initial characters common to object names that can be used to identify a group of objects. A generic name ends with an * (asterisk). For example, ORD* identifies all objects whose names begin with the characters ORD.

get operation: An input operation that obtains a record from an input file and passes it to a program. Also called input operation or read operation.

GFT: See *grant functional transmission.*

grant functional transmission: In System/38 (RJEF) MTAM, a control character indicating that the host system grants permission to System/38 to send reader data, or that System/38 grants permission to the host system to send writer data. Abbreviated GFT. Contrast with *request functional transmission.*

heading: A constant, or field, usually at the top of a page or display, that identifies the information on the page or display.

help text: Information that is associated with an information display, a menu, or a prompt that explains options or values displayed. Help text is requested by pressing the Help key.

hexadecimal: Pertaining to a numbering system with a base of 16. Valid numbers are the digits 0 through 9 and the characters A through F, where A represents 10 and F represents 15.

hexadecimal number: The 1-byte hexadecimal equivalent of an EBCDIC character.

high-level language: A programming language that relieves the programmer from the rigors of machine level or assembler level programming; for example, RPG III, CL, and COBOL. Abbreviated HLL.

high-level message: A message that is sent to the program message queue of the program receiving the request. The message is displayed or provided for the user who entered the request. Contrast with *low-level message.*

history log: A log of information about system status and events. Named QHST.

HLL: See *high-level language.*

hold delivery: The method of delivering messages to a message queue that holds the messages until the user requests them. The user is not notified when a message arrives at a message queue that is in hold delivery.

host system: The controlling or highest level system in a data communications configuration. For example, a System/38 is the host system for the work stations connected to it.

IDU: See *Interactive Data Base Utilities.*

immediate maintenance: A method of maintaining keyed access paths for data base files. This method updates the access path whenever changes are made to the data in the access path. Contrast with *rebuild maintenance* and *delay maintenance.*

impromptu message: A message that is created when it is sent. Contrast with *predefined message.*

informational message: A message that conveys information about the normal condition of a function.

initial program: A program, specified in a user profile, that is to be executed when the user signs on and the command processor program QCL is invoked. QCL invokes the initial program.

initialize: To set to a starting position or value.

inline data file: A file described by a //DATA command that is included as part of a job when the job is read from an input device by a reader program.

input: Information (or data) to be processed.

input-capable field: Any field in a display file that can receive input from a user.

input field: A field in a display file into which a work station user can key in data. An input field is passed from the device to the program when the program reads the record containing that field.

input file: (1) A data base or device file that has been opened with the option to allow records to be read. (2) (ANSI) In COBOL, a file that is opened in the input mode.

input stream: (1) A group of records submitted to the system as batch input that contains CL commands for one or more jobs and/or the data records for one or more inline data files. (2) In RJEF, data sent to the host system. Contrast with *output streams.*

inquiry: A request for information from a data file usually made against one record.

inquiry message: A message that conveys information and that requests a reply.

insert: The SEU operation during which source statements are keyed in and added as new records in a source member.

integrity: The protection of data and programs from inadvertent destruction or alteration.

interactive: Pertaining to a program or system that alternately accepts input and then responds. An interactive system is conversational; that is, a continuous dialog exists between the user and the system.

Interactive Data Base Utilities: A System/38 licensed program that consists of DFU, SEU, query, and SDA. Abbreviated IDU.

interactive job: A job in which the processing actions are performed in response to input provided by a work station user. During a job, a dialog exists between the user and the system.

interactive processing: Pertaining to a program or procedure that alternately accepts input and then responds to the input. Contrast with *batch processing.*

interactive subsystem: A subsystem in which interactive jobs are to be processed. IBM supplies three interactive subsystems: QCTL, QINTER, and QPGMR.

internal storage: All main and auxiliary storage in the system.

invocation: An instance of the execution of a program.

invocation level: Identifies the occurrence of the same program in the job's invocation stack. An invocation level is used in debug mode only. The first occurrence of a program in a job has an invocation level of 1.

invocation nesting: The situation in which more than one invocation of the same program exists in an invocation stack.

invocation number: The number that identifies each program invocation in an invocation stack. The highest level program has an invocation number of 1.

invocation stack: A series of invocations linked together as a result of programs invoking other programs.

invoke: To instruct a specific program to start executing. Same as *call.*

I/O slot: One of three locations in the diskette magazine drive where individual diskettes can be inserted for input/output operations. Same as *manual slot.*

JES: See *Job Entry Subsystem.*

job: A single identifiable sequence of processing actions that represents a single use of the system. A job is the basic unit by which work is identified on the system.

job control rights: The authority to change, cancel, display, hold, and release all jobs and, optionally, job and output queues and entries on them.

job date: The date associated with a job. The job date usually defaults to the system date.

job description: An object that contains information defining the attributes of a job. The system-recognized identifier for the object type is *JOBD.

Job Entry Subsystem: A host system (non-System/38) subsystem that receives jobs into the system and processes all output data produced by the jobs. Abbreviated JES.

job log: A record of requests submitted to the system by a job, the messages related to the requests, and the actions performed by the system on the job. The job log is maintained by CPF.

job message queue: A message queue that is created for each job. A job message queue is used for receiving requests to be processed (such as commands) and for sending messages that result from processing the requests. A job message queue consists of an external message queue and a set of program message queues. See also *external message queue* and *program message queue.*

job name: The name of a job as identified to the system. For an interactive job, the job name is the name of the work station at which the job was initiated; for a batch job, the job name is specified in the command used to submit the job. Contrast with *qualified job name.*

job number: A number assigned to a job as it enters the system to distinguish the job from other jobs.

job priority: The order in which batch jobs on a job queue are selected for execution by CPF. More than one job can have the same priority.

job queue: An object that contains a list of batch jobs submitted to the system for execution and from which the batch jobs are selected for execution by CPF. The system-recognized identifier for the object type is *JOBQ.

job queue entry: A work entry in a subsystem description that specifies the job queue from which the subsystem can accept batch jobs and transferred jobs.

job separator: The pages or cards placed at the beginning of the output for each job that has spooled file entries on the output queue. Each separator contains information that identifies the job such as its name, the job user's name, the job number, and the time and date the job was executed.

job stream: See *input stream.*

journal: (1) An object through which entries are placed in a journal receiver when a change is made to a data base file. The system uses the journal to record information about the journal receivers and data base files that are associated with the journal. The system-recognized identifier for the object type is *JRN. See also *journal entry* and *journal receiver.* (2) To place entries in a journal.

journal code: A 1-character code in a journal entry that identifies the primary category of the journal entry; for example, F identifies a file-level entry.

journal entry: A record in a journal receiver that contains information about data base files being journaled. See also *journal code, journal entry identifier, journal entry qualifier, journal entry type,* and *journal entry-specific data.*

journal entry identifier: The portion of a journal entry that identifies the category of the journal entry, the type of journal entry, the date and time of the entry, the job name, the user name, and the program name.

journal entry qualifier: The portion of a journal entry that identifies the name of the object for which the journal entry was generated.

journal entry-specific data: The user-generated or system-generated data in a journal entry. This data is unique to the operation that generated the journal entry.

journal entry type: A 2-character field in a journal entry that identifies the type of user-generated or system-generated journal entry; for example, PT is the entry type for a put operation.

journal receiver: An object that contains journal entries that are generated when a change is made to a data base file being journaled. The system-recognized identifier for the object type is *JRNRCV. See also *journal*.

journaling: The process of recording changes made to a physical file member in a journal. Journaling allows the programmer to reconstruct a physical member by applying the changes in the journal to a saved version of the physical file member.

key field: A field in a record whose contents are used to sequence the records of a particular type within a file member.

keyed sequence: The order in which records appear in an access path. The access path is based on the contents of one or more key fields contained in the records.

keyed sequence access path: An access path to a data base file that is ordered on the contents of key fields contained in the individual records. See also *arrival sequence access path* and *access path*.

keyword: (1) A name that identifies a parameter. Keywords are used in CL commands and in DDS. (2) In RPG, a word whose use is essential to the meaning and structure of a statement in a programming language.

label: (1) The name of a file on a diskette or tape. (2) An identifier of a command generally used for branching. (3) In RPG, a symbolic name that represents a specific location in a program. A label can serve as the destination point for one or more branching operations.

level checking: A function that compares the record format level identifiers of a file to be opened with the file description that is part of a compiled program to determine if the file record format has changed since the program was compiled.

library: An object that serves as a directory to other objects. A library is used to group related objects and to find objects by name when they are used. The system-recognized identifier for the object type is *LIB.

library list: An ordered list of library names used to find an object. The library list indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is *LIBL. *LIBL specifies to the system that a job's current library list is to be used to find the object.

licensed program: An IBM-written program that performs functions related to processing user data.

LIND: See *line description*.

line: See *communications line*, *multipoint line*, *nonswitched line*, *point-to-point line*, and *switched line*.

line control characters: See *transmission control characters*.

line description: An object that contains a description of a communications line to the system. The system-recognized identifier for the object type is *LIND. Abbreviated LIND.

list element: One of several values specified in a list parameter.

list parameter: A parameter defined to accept a list of multiple like values or unlike values.

listing: A printout usually containing the input and output of the compilation of a program, the creation (compilation) of an object, or the execution of a program. See also *compiler listing*.

literal: A character string whose value is given by the characters themselves. For example, the numeric literal 7 has the value 7, and the character literal 'CHARACTERS' has the value CHARACTERS. See also *character literal*, *constant*, and *numeric literal*.

local work station: A work station that is connected directly to System/38 without need for data transmission facilities. Contrast with *remote work station*.

lock state: The definition of how an object is allocated, how it is used (read or update), and whether the object can be shared (used by more than one job).

log: See *error log*, *history log*, *job log*, *programming change log*, and *service log*.

logical file: A description of how data is to be presented to or received from a program. This type of data base file contains no data, but it provides an ordering and format for one or more physical files. Contrast with *physical file.*

logical file member: A named logical grouping of data records from one or more physical file members. See also *member*.

logical unit: In SNA, one of three types of network addressable units. It is a port through which a user accesses the SNA network in order to communicate with another user and through which the user accesses the functions provided by the system services control point. Abbreviated LU. See also *physical unit, system services control point, primary logical unit,* and *secondary logical unit.*

logical unit description: An MI object that is created as the result of executing the Create Device Description (CRTDEVD) command. Abbreviated LUD.

low-level message: A message that is sent to the program message queue of the lower-level program invocation. A low-level message is normally not displayed. Contrast with *high-level message.*

LU: See *logical unit.*

LUD: See *logical unit description.*

machine execution priority: The priority of a routing step when competing with other routing steps for machine resources.

machine interface: The instruction set and interface to the machine. The instruction set is called the System/38 instruction set. Abbreviated MI.

machine storage pool: A storage pool used by the machine and certain highly shared CPF programs and whose size is specified in the system value QMCHPOOL.

magazine: A container that holds up to 10 diskettes and is inserted into a diskette magazine drive.

main storage: All storage in a computer from which instructions can be executed directly.

manual answer: Operator actions to make a station ready when a station receives a call on a switched line.

manual call: Operator actions to make a connection with a station on a switched line.

manual slot: See *I/O slot.*

member: A description of a named subset of records in a physical or logical file. Each member conforms to the characteristics of the file and has its own access path. All I/O requests are directed to a specific member of a data base file.

member list display: A display that lists the names of the members in a file and allows you to select a member to process.

menu: A display in which a list of options is shown.

merge: To intersperse records throughout a single output file.

message: A communication sent from one person or program to another person or program.

message description: The information describing a particular message. A message description is stored in a message file.

message field: In a display file, an output field that is treated as a message.

message file: An object that contains message descriptions. The system-recognized identifier for the object type is *MSGF.

message identifier: A 7-character code that identifies a predefined message and is used to retrieve its message description from a message file.

message queue: An object on which messages are placed when they are sent to a person or program. The system-recognized identifier for the object type is *MSGQ.

message reference key: A key assigned to every message on a message queue. This key is used to remove a message from a message queue, to receive a message, and to reply to a message.

MI: See *machine interface.*

mixed list: A series of unlike values for a parameter that accepts a set of separately defined values. Contrast with *simple list.*

modem: A mechanism that modulates and demodulates signals transmitted over data communications facilities.

MRJE: See *multi-leaving remote job entry.*

MTAM: See *multi-leaving telecommunications access method.*

multi-leaving remote job entry: The fully synchronized, two-directional transmission of a variable number of data streams between two computers using BSC facilities.

multi-leaving telecommunications access method: An access method that supports System/38 MRJE functions.

multiple device file: A device file that was created with the maximum number of devices greater than 1. When the file is opened, one or more devices can be connected to the data path. I/O requests are directed to each device by specifying the device name on the request.

multipoint line: A line or circuit interconnecting several stations. Contrast with *point-to-point line.*

multivolume file: A file that is contained on more than one diskette or tape.

nested command: A command or group of commands whose execution is conditioned on the evaluation of a preceding or associated command. Nesting is a structured form of branching. In CL programs, the nested command is embedded in an associated command. If the nested command is a DO command, the entire do group is nested.

nested do group: A do group that is contained within another do group.

network: A configuration in which two or more stations can communicate.

nonswitched line: A connection between systems or devices that does not have to be made by dialing. Contrast with *switched line.*

normal termination: System termination that results from the successful execution of the Power Down System (PWRDWNSYS) command. See also *abnormal termination* and *system termination.*

notify delivery: The method of delivering messages to a message queue in which the work station user is notified that a message is on the queue. The notification is by means of an attention light or an audible alarm.

notify message: A message that describes a condition for which a program requires a reply from its caller, or a default reply is sent to the program.

numeric character: Any one of the digits 0 through 9.

numeric field: An area that is reserved for a particular unit of information and that can contain only the digits 0 through 9. Contrast with *character field.*

numeric literal: The actual numeric value to be used in processing, instead of the name of a field containing the data. A numeric literal can contain any of the numeric digits 0 through 9, a sign (plus or minus), and a decimal point. Contrast with *character literal.*

object: A named unit that consists of a set of attributes (that describe the object) and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed. Some examples of objects are programs, files, and libraries.

object authority: The right to use or control an object. See *object rights* and *data rights.*

object definition table: A part of the definition of a program that defines the program objects associated with the instructions in its instruction stream. Operands of an instruction refer to entries in this table. Abbreviated ODT.

object description: The attributes (such as name, type, and owner name) that describe an object.

object existence rights: The authority to delete, save, free the storage of, restore, and transfer ownership of an object.

object management rights: The authority to move, rename, grant authority to, revoke authority from, and change the attributes of an object.

object name: The name of an object. Contrast with *qualified object name.*

object owner: A user who creates an object or to whom the ownership of an object has been transferred. The object owner has complete control over the object.

object rights: The authority that controls what a system user can do to an entire object. For example, object rights include deleting, moving, or renaming an object. There are three types of object rights: object existence, object management, and operational.

object type: The attributes that define the purpose of an object within the system. Each object type has associated with it a set of commands with which to process that type of object.

object user: A user who has been authorized by the object owner, the security officer, or a user with object existence rights to perform certain functions on an object.

ODP: See *open data path.*

ODT: See *object definition table.*

offline: Pertaining to the operation of a functional unit that is not under the continual control of the system. Contrast with *online.*

omit function: A CPF function that determines which records from a physical file are to be omitted from a logical file's access path. Contrast with *select function.*

online: Pertaining to the operation of a functional unit that is under the continual control of the system. Contrast with *offline.*

online backup: The method of using the Copy File (CPYF) command to internally copy a data base file to another data base file for backup.

online test: A standardized set of tests for BSC. The tests are used to ensure the proper operation and integrity of the communications link (lines and modems) portion of the total system.

open: The function that connects a file to a program for processing. Contrast with *close.*

open data path: The path through which all I/O operations for the file are performed. Abbreviated ODP.

operational rights: The authority to use an object and to look at its description.

output: Data transferred from storage to an output device.

output field: A field in a device file in which data can be modified by the program and sent to the device during an output operation.

output file: (1) A data base or device file that has been opened with the option to allow records to be written. (2) In COBOL, a file that is opened in either output mode or extend mode.

output priority: The priority used to determine the order in which spooled output files produced by the job are to be written. More than one file can have the same priority.

output queue: An object that contains a list of output files to be written to an output device by a writer. The system-recognized identifier for the object type is *OUTQ.

output stream: In RJEF, data received from the host system (for example, control characters, data files, and messages). Contrast with *input stream.*

overflow: The condition that occurs when the last line specified as the overflow line to be printed on a page has been passed.

packed decimal format: Representation of a decimal value in which each byte within a field represents two numeric digits except the rightmost byte, which contains one digit in bits 0 through 3 and the sign in bits 4 through 7. For all other bytes, bits 0 through 3 represent one digit; bits 4 through 7 represent one digit. For example, the decimal value +123 is represented as 0001 0010 0011 1111. Contrast with *zoned decimal format.*

packed field: A field that contains data in the packed decimal format.

page: (1) A 512-byte block of information that can be transferred between auxiliary storage and main storage. (2) Each group of records in a subfile that are displayed concurrently. (3) One printer form.

page fault: A program notification that occurs when a page that is marked as not in main storage is referred to by an active page.

page frame: A 512-byte block of main storage that contains a page.

page-in: The process of transferring a page from auxiliary storage to main storage.

page-out: The process of transferring a page from main storage to auxiliary storage.

parameter: (1) Data passed to or received from another program. (2) In CPF, an argument that identifies an individual value or group of values to be used by a command to tailor a function requested through the command.

parameter list: A list of values that provides a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

partial journal receiver: A journal receiver that was saved while it was attached to a journal. Therefore, the saved version of a partial journal receiver does not contain all the journal entries that are in the attached journal receiver.

password: A unique string of characters that a system user enters to identify himself to the system.

PC: See *programming change*.

physical file: A description of how data is to be presented to or received from a program and how data is actually stored in the data base. A physical file contains one record format and one or more members. Contrast with *logical file*.

physical file member: A named subset of the data records in a physical file. See also *member*.

physical unit: In SNA, one of three types of network addressable units. A physical unit exists in each node of an SNA network to manage and monitor the resources (such as attached links and adjacent link stations) of a node, as requested by an SSCP-LU session. Abbreviated PU.

physical unit type: In SNA, the classification of a physical unit according to the type of node in which it resides. The physical unit type is the same as its node type; that is, a type 1 physical unit resides in a type 1 node, and so on.

PLU: See *primary logical unit*.

point-to-point line: A data link that connects a single remote station to a data processing system; it can be either switched or nonswitched. Contrast with *multipoint line*.

predefined message: A message whose description is created independently of when it is sent and is stored in a message file. Contrast with *impromptu message*.

predefined value: An IBM-defined fixed value that has a special use in the control language and is reserved in CPF. A predefined value has an asterisk (*) as the first character in the value.

primary logical unit: In SNA, the logical unit that contains the primary half-session for a particular LU-LU session. Abbreviated PLU. See also *logical unit*. Contrast with *secondary logical unit*.

print image: An object that contains a description of the print belt or train on a printer. The system-recognized identifier for the object type is *PRTIMG.

printer file: A device file created by the user to support a printer device.

priority: The relative significance of one job to other jobs in competing for allocation of resources.

problem determination: The process of determining the source of a problem as a component problem, a machine failure, a common carrier link, a user-supplied element, or a user error.

problem determination procedure: A prescribed sequence of steps taken to identify the source of a problem.

processing: The action of performing operations on input data.

production library: A library containing objects needed for normal processing. Contrast with *test library*.

program: An object that contains a set of instructions that tell a computer where to get input, how to process it, and where to put the results. A program is created as a result of a compilation. The system-recognized identifier for the object type is *PGM.

program-described data: Data contained in a file for which the fields in the records are described in the program that processes the file. Contrast with *externally described data*.

program-described file: A file for which the fields in the records are described only in the program that processes the file. To CPF, the record is viewed as a character string. Contrast with *externally described file*.

program level: Pertaining to an operation that is performed for an entire program. For example, a Monitor Message (MONMSG) command that immediately follows the last declare command in a CL program is a program-level MONMSG command. Contrast with *command level*.

program message queue: A message queue used to hold messages that are sent between program invocations of a routing step. The program message queue is part of the job message queue.

program object: One of two MI object classifications. It includes those objects used in programs that get their definition from ODT entries. Contrast with *system object*.

program patch: A method of repairing a program at the MI program template level.

program variable: A named changeable value that can exist only within programs. Its value cannot be obtained or used when the program that contains it is no longer invoked.

programmer subsystem: An interactive subsystem in which programmers can perform online programming through 5250 Display Stations. IBM supplies one programmer subsystem: QPGMR.

programmer user profile: The CPF-supplied user profile that has the authority necessary for system and application programmers and the special authorities of save system rights and job control rights. Named QPGMR.

programming change: A modification to an IBM-supplied program. Abbreviated PC.

programming change log: A log of information about the application of program changes and patches to IBM products. Named QCHG.

programming service representative user profile: The CPF-supplied user profile that has the authority necessary for the programming service representative to service the system's programming and the special authorities of save system rights and job control rights. Named QPSR.

prompt: A displayed request for information or user action. The user must respond to allow the program to proceed.

protected field: A field in a display file in which data cannot be keyed, changed, or erased.

PU: See *physical unit*.

public authority: The authority to an object granted to all users.

purge: An attribute that specifies whether the job is to be marked eligible to be moved out of main storage and put into auxiliary storage at the end of a time slice or upon entering a long wait.

put operation: An output operation that writes a record to an output file. Also called an output operation or a write operation.

put-get operation: A combination of an output operation (put) followed by an input operation (get) to the same record format.

QCL: The IBM-supplied control language processor that accepts CL commands so that they can be interpreted and executed by the system.

QGPL: See *general-purpose library*.

qualified job name: A job name and its associated user name and a system-assigned job number. Contrast with *job name*.

qualified object name: An object name and the name of the library containing the object. Contrast with *object name*.

qualifier: A name used to uniquely identify another name. In CPF, for example, a library name can be used to qualify an object name. In COBOL, group data-names, section-names, and library-names can be used as qualifiers to form qualified names.

query: (1) A utility that is part of the Interactive Data Base Utilities licensed program. (2) A request to extract, from a file, one or more records based upon some combination of data.

query application: See *application.*

queue: A line or list formed by items in the system waiting for service; for example, work to be performed or messages to be displayed.

random processing: A method of file processing in which a program does not read records from a file in any prespecified order. Instead, the program uses a key field or a relative record number to access a specific record.

RCB: See *record control byte.*

read rights: The authority to read the entries in an object. Contrast with *add rights, delete rights,* and *update rights.*

reader: (1) A program that reads jobs from an input device or a data base file and places them on a job queue. (2) In RJEF, a program that reads jobs from a data base file and sends them to the host system.

rebuild maintenance: A method of maintaining keyed access paths for data base files. This method updates the access path only while the file is open, not when the file is closed; the access path is rebuilt when the file is opened. Contrast with *immediate maintenance* and *delay maintenance.*

receive mode: A time during which the BSC adapter looks for synchronization characters and then stores the data characters in main storage.

receive time-out: For BSC, an indication that no data has been received by this communications adapter in a given period of time.

receiver: See *journal receiver.*

receiver directory: A display that contains summary information about the journal receivers that are or were attached to the specified journal and that are still known to the system.

record: (1) An ordered set of fields that make up a single occurrence of the basic unit of data transferred between a file and a program. (2) In COBOL, a set of one or more related data items that are grouped for processing. Records can be defined for an input/output device or for internal processing. See also *logical record.*

record control byte: In System/38 (RJEF) MTAM, a control character used to identify each record type within a transmission block. Abbreviated RCB.

record format: The definition of how data is structured in the records contained in a file. The definition includes the record name, field names, and field descriptions (such as length and data type). The record formats used in a file are contained in the file's description.

record format level identifier: An identifier placed on a record format that uniquely identifies the record description. See also *level checking.*

recovery: The act of resetting the system, or data stored in the system, to an operable state following damage.

recovery library: The library containing information related to recovery of data base operations from system failures. Named QRECOVERY.

relational operator: (1) In CL, an operator that can be used in an arithmetic, character, or logical relation to indicate the comparison to be performed between the terms in the relation. The relational operators are *EQ or = (equal to), *GT or > (greater than), *LT or < (less than), *GE or >= (greater than or equal to), *LE or <= (less than or equal to), *NE or ¬= (not equal to), *NG or ¬> (not greater than), *NL or ¬< (not less than). (2) In COBOL, a reserved word, a relation character, a group of consecutive reserved words, or a group of consecutive reserved words and relation characters used to construct a relation condition.

relative record number: A number that specifies the location of a record in relation to the beginning of a data base file member or subfile. For example, the first record in a data base file member or subfile has a relative record number of 1.

remote device: A device whose control unit is connected to a System/38 through a data link.

remote entry services: In OS/VS1, the set of functions added to the Job Entry Subsystem (JES) that allows jobs and their associated data to be entered from remote devices (System/38), processed at the central system, and then transmitted back to the remote devices. Abbreviated RES.

remote equipment: The modem and control unit equipment that provides the communications connection between a communications line and a remote device or station. This remote equipment is at the other end of a data link from the host System/38. For System/38, the remote equipment could be partially or totally contained within a 5251 Model 2 or Model 12 work station/control unit.

Remote Job Entry Facility: A System/38 licensed program that provides a data link with a remote host system. Abbreviated RJEF.

Remote Spooling Communications Subsystem: The component of VM/370 that transfers spooled files between VM/370 users, remote stations (System/38), and remote and local batch stations through HASP-compatible telecommunications facilities. Abbreviated RSCS.

remote terminal access method: A non-System/38 facility that controls operations between the Job Entry Subsystems (JES2 and JES3) and remote work stations (System/38). Abbreviated RTAM.

remote work station: A work station whose connection to the processing system uses modems and common carrier or private data transmission facilities. Contrast with *local work station.*

remove: In journaling, to remove the after-images of records from a physical file member. The file then contains the before-images of the records that are recorded in a journal. Contrast with *apply.*

reply message: A message that is sent as a response to a received inquiry or notify message.

request: (1) A CL command, the selection of an option on a menu, or the entering of data that instructs the system to perform a function. A CL command can be entered interactively or in a batch job. A request is identified as RQS on the job log. (2) In SNA, a request unit that asks for a particular action or protocol; or a message unit that acknowledges receipt of a request. A request consists of a request unit, a segment of a request unit, or both.

request data: Data to be put in a job message queue that is used by a job. For example, a single command or group of commands.

request for test: In BSC, a request to perform an online test function. Abbreviated RFT.

request functional transmission: In System/38 (RJEF) MTAM, a control character indicating a request for permission to send reader data or writer data. Abbreviated RFT. Contrast with *grant functional transmission.*

request message: A message that requests a function from the receiving program.

RES: See *remote entry services.*

response indicator: A 1-character field passed with an input record from CPF to a program to provide information about the data record or actions taken by the work station user.

restore: To transfer specific objects or libraries from magnetic media such as diskettes or tape to internal storage by reconstructing them in internal storage. Contrast with *save.*

RFT: (1) See *request for test.* (2) See *request functional transmission.*

right-adjust: To place an entry in a field or to move the contents of a field so that the rightmost character of the data is in the rightmost position of the field.

RJEF: See *Remote Job Entry Facility.*

routing data: A character string that CPF compares with character strings in the subsystem description routing entries to select the routing entry that is to be used to initiate a routing step. Routing data can be provided by a work station user, specified in a command, or provided through the job description for the job.

routing entry: An entry in a subsystem description that specifies the program to be invoked to control a routing step that executes in the subsystem.

routing step: The processing performed as a result of invoking a program specified in a routing entry.

RQS: See *request.*

RSCS: See *Remote Spooling Communications Subsystem.*

RTAM: See *remote terminal access method.*

save: To duplicate specific objects or libraries by transferring them from internal storage to magnetic media such as diskettes or tape. Contrast with *restore.*

save system rights: The authority to save all objects.

screen design aid: The utility of the Interactive Data Base Utilities licensed program that is used to interactively design, create, and maintain display record formats and menus. Abbreviated SDA.

SDA: See *screen design aid.*

SDLC: See *Synchronous Data Link Control.*

secondary logical unit: In SNA, the logical unit that contains the secondary half-session for a particular LU-LU session. Abbreviated SLU. See also *logical unit.* Contrast with *primary logical unit.*

second-level message: A message that provides additional information to that already provided in a first-level message. See also *second-level message display.*

second-level message display: A display containing the second-level message text (if any) and additional message information. This display is obtained by pressing the Help key while a first-level message is displayed.

sector: The addressable unit into which each track on a diskette is divided.

secured file: A file that is protected from being overridden by an override file command.

security: The control of access to, or use of, data or functions.

security officer: The individual at an installation who is designated to control the authorization of functions and data in System/38.

security officer user profile: The CPF-supplied user profile that has authority to control the authorization of functions and data used in the installation. Named QSECOFR.

select function: A CPF function that determines which records from a physical file are to be selected for a logical file's access path. Contrast with *omit function.*

select/omit field: A field in a logical file record format whose value is compared with a constant, the contents of another field, a range of values, or a list of values to determine if a record is to be omitted from the access path of the logical file or selected for use by the logical file. See also *omit function* and *select function.*

separator: A punctuation character used to delimit character strings. See also *file separator* and *job separator.*

sequence number: (1) The number of a record that identifies the record within the source member. (2) A field in a journal entry that contains a number assigned by journal management. This number is initially 1 and is incremented by 1 until the journal is deleted or the sequence number is reset by the user.

sequential-by-key processing: A method of file processing that reads records from a keyed sequence file in the order in which the keys are arranged in the access path.

sequential file: A file in which records are processed in the order that they are stored in the file.

service library: The library provided in CPF that is used temporarily for loading IBM-supplied programming changes and assembling data for APAR submission. Named QSRV.

service log: A log of information about errors detected in IBM program products. Named QSRV.

session description: An object that contains a description of the operating characteristics of an RJEF session. The system-recognized identifier for the object type is *SSND.

SEU: See *source entry utility.*

severity code: A code that indicates how important a message is. The higher the code, the more serious the condition is.

shared access path: An access path used by more than one file to provide access to data common to the files.

shared file: A file whose open data path can be shared between two or more programs executing in the same routing step.

shared-for-read lock state: The allocation that a routing step has for an object in which the object can be shared with another routing step if the routing step does not request exclusive use of the object. The predefined value for this lock state is *SHRRD.

shared-for-update lock state: The allocation that a routing step has for an object in which the object can be shared either for update or for read with another routing step. The predefined value for this lock state is *SHRUPD.

shared-no-update lock state: The allocation that a routing step has for an object in which the object can be shared with another routing step if the routing step requests either a shared-no-update lock state or a shared-for-read lock state. The predefined value for this lock state is *SHRNUP.

shared record format: A record format that is used in more than one externally described file.

sign off: To enter a command or to select an option from a menu at a work station that instructs the system to end an interactive job.

sign on: To enter a password that identifies the user to the system and instructs the system to establish an interactive job at a work station.

simple list: A list of like values, for example, a list of user names. Contrast with *mixed list*.

simple object name: Same as *object name*.

single value: A value that can be specified in place of multiple values for a list parameter in a CL command.

skip: To cause the printer to move the paper to a specified line before or after it prints a line.

slot: See *I/O slot*.

SLU: See *secondary logical unit*.

source entry utility: The utility of the Interactive Data Base Utilities licensed program that is used to create and change source members. Abbreviated SEU.

source file: A file created by the specification of FILETYPE(*SRC). A source file can contain source statements for such items as high-level language programs and data description specifications.

source listing: A portion of a compiler listing that contains source statements and optionally diagnostics. See also *compiler listing*.

source member: A member of a data base source file that contains source statements such as RPG, COBOL, or DDS specifications. See also *member*.

source program: A set of instructions, written in a programming language such as RPG or COBOL, that represents a particular job as defined by a programmer. A source program is used as input to the compiler to create an executable program.

source statement: A statement written in symbols of a programming language. For example, RPG, COBOL, or DDS specifications are source statements.

space: To cause the printer to move the paper a specified number of lines before or after it prints a line.

spanned record: A logical record stored in more than one block on a diskette.

special authority: The right to perform certain system control operations, such as save system and job control operations.

special character: (1) A character other than a digit, a letter, or #, $, or @. For example, *, +, and % are special characters. (2) In COBOL, a character that is neither numeric nor alphabetic. Special characters in COBOL include: + - * / = $, . ") (; < >

special value: A value that can be specified in a CL command or in a message and that does not have to satisfy validity checking criteria.

spooled file: A generic term for three types of files: a device file that provides access to an inline data file or that creates a spooled output file, an inline data file, or a spooled output file.

spooled input file: See *inline data file*.

spooled output file: A device file that causes output data to be saved for later processing by a writer.

spooling: The CPF-provided execution-time support that reads and writes input and output streams on an intermediate device in a format convenient for later processing or output.

spooling subsystem: A subsystem that provides the operating environment needed by the CPF programs that read jobs onto job queues and write files from the output queues. IBM supplies one spooling subsystem: QSPL.

standby line: A modem feature that allows a point-to-point nonswitched line modem to also function on a point-to-point switched line.

station: A system or device that can send or receive data over a communications line.

status message: A message that describes the status of the work done by a program.

storage pool: A logical segment of main storage reserved for executing a group of jobs.

subfile: A group of records of the same record format that can be displayed concurrently at a work station. The system sends the entire group of records to the work station in a single operation and receives the group in another operation.

substitution variable: A variable used to pass information such as a file name for use in a message.

subsystem: An operating environment, defined by a subsystem description, through which CPF coordinates work flow and resource usage.

subsystem attributes: Specifications in a subsystem description that specify the amount of main storage available to the subsystem and the number of jobs that can execute concurrently in the subsystem.

subsystem description: An object that contains information defining a subsystem and that CPF uses to control the subsystem. The system-recognized identifier for the object type is *SBSD.

switched line: A connection between two stations that is established by dialing. Contrast with *nonswitched line*.

Synchronous Data Link Control: A discipline conforming to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute (ANSI) and High-level Data Link Control (HDLC) of the International Standards Organization (ISO), for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop. Abbreviated SDLC. Contrast with *binary synchronous communications*.

syntax checking: A function of the command analyzer, a compiler, or SEU that checks single statements for violations of the rules governing the structure of the statement.

system date: The date established for the system when it is started.

system library: The library provided by CPF to contain system-oriented objects provided as part of CPF. Named QSYS.

system log message queue: A message queue used for sending information to the system history log, service log, or programming change log, from any job in the system.

system object: One of two MI object classifications. It includes those MI objects whose formats are not visible above MI and all objects that are defined during execution time from attribute template operands on create instructions. These objects are referred to through system pointers. Contrast with *program object*.

system operator: The person who operates the system and looks after the peripheral equipment necessary to initiate computer runs or finalize the computer output in the form of completed reports and documents.

system operator message queue: The message queue used by the system operator to receive and reply to messages from the system, work station users, and application programs. Named QSYSOPR.

system operator user profile: The CPF-supplied user profile that has the authority necessary for the system operator and the special authorities of save system rights and job control rights. Named QSYSOPR.

system pointer: A pointer that contains addressability to an MI system object.

system resources: Those items owned by the system and allocated or deallocated for use in jobs.

system services control point: In SNA, a network addressable unit that provides configuration, maintenance management, and session services through sessions with physical units, logical units, and other system services control points. Abbreviated SSCP.

system termination: The state in which all processing on the system is stopped. Depending on the cause of the termination, system power could be shut off (such as by a power interruption or by entering the Power Down System (PWRDWNSYS) command) or could remain on (such as caused by a machine error condition). See also *abnormal termination* and *normal termination.*

system time: The elapsed time from the point where the system was started to the current time. If the system time is changed to the local time when the system is started, the current system time is the local time of day.

system unit: The main unit of the system, which contains the processing unit, the system console keyboard/display, the operator/service panel, the diskette magazine drive, main storage, auxiliary storage, the work station controller, and the communications subsystem.

system value: A value that contains control information for the operation of certain parts of the system. A user can change the system default value to tailor the system to his working environment. System date and library list are examples of system values.

Systems Network Architecture: The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of Systems Network Architecture networks. Abbreviated SNA.
Note: The layered structure of SNA allows the ultimate origins and destinations of information (that is, the end users) to be independent of, and unaffected by, the specific SNA network services and facilities used for information exchange.

tape file: A device file created by the user to support a tape device.

telecommunications access method: A non-System/38 access method used to transfer data between main storage and terminals (local or remote). Abbreviated TCAM.

temporary library: A library that is automatically created for each job to contain temporary objects that are created by that job. The objects in the temporary library are deleted when the job ends. Named QTEMP.

terminal: In data communications, same as *work station.*

termination: The act of putting the system or an element of the system (such as CPF or a subsystem) in the state where it no longer performs its normal function. See also *system termination.*

test library: A library to be used in debug mode and that does not contain objects needed for normal processing. Contrast with *production library.*

text transparency: A provision that allows BSC to send and receive messages containing any or all of the 256 character combinations in EBCDIC, including transmission control characters. Transmission control characters sent in a message are treated as data, unless they are preceded by the DLE control character.

TH: See *transmission header.*

time slice: The quantity of processor time (specified in milliseconds) allowed for a routing step before other waiting routing steps are given the opportunity to execute.

time stamp: (1) To apply the current system time. (2) The value on an object that is an indication of the system time at some critical point in the object's history. (3) In query, the identification of the day and time a query report was created that query automatically provides on each report.

trace: The process of recording the sequence in which the statements in a program are executed and, optionally, the values of the program variables used in the statements.

track: That portion of the diskette recording surface available to one read/write head at each access position.

translate table: An object that contains a set of hexadecimal characters used to translate one or more characters of data. For example, unprintable characters can be translated to blanks, and lowercase alphabetic characters can be translated to uppercase characters. The system-recognized identifier for the object type is *TBL.

transmission control characters: Special BSC characters that are included in a message to control communications over a data link.

transmission header: In SNA, control information, optionally followed by a basic information unit or a basic information unit segment, that is created and used by path control to route message units and to control their flow within the network. Abbreviated TH.

transparent text mode: A method of binary synchronous transmission in which only transmission control characters preceded by the DLE control character are processed as transmission control characters.

tributary station: A secondary device on a multipoint line.

truncate: To drop data that cannot be printed or displayed in the line width specified or available. Contrast with *fold*.

UDS: See *utility definition specifications*.

unary operator: (ANSI) In COBOL, a plus sign (+) or a minus sign (-), which precedes a variable or a left parenthesis in an arithmetic expression and which has the effect of mulitiplying the expression by +1 or -1, respectively.

uninterruptible power supply: A buffer between the utility power (or other power source) and a load that requires uninterrupted, precise power. Abbreviated UPS.

update operation: An I/O operation that modifies the information in a file.

update rights: The authority to change the entries in an object. Contrast with *add rights*, *delete rights*, and *read rights*.

UPS: See *uninterruptible power supply*.

user-defined edit code: A number (5 through 9) indicating that editing should be done on a numeric output field according to a pattern predefined to CPF. User-defined edit codes can take the place of edit words, so that repetitive coding of the same edit word is not necessary.

user identification: System recognition of a system user so that only the facilities and data he is authorized to use are made available to him.

user message queue: A user-created message queue used to send messages to system users and between application programs.

user name: The name by which a particular user is known to the system.

user password: A unique string of characters that a system user enters to identify himself to the system.

user profile: An object that contains a description of a particular user or group of users. A user profile contains a list of authorizations to objects and functions. The system-recognized identifier for the object type is *USRPRF.

utility definition specification: A group of source statements, which have the same syntax as CL commands, from which a DFU or query application is created. Abbreviated UDS.

validity checker: A user-written program that tests commands for errors in the parameter values. Validity checking is done in addition to the checking done by the command analyzer.

validity checking: Operations performed against a field value to ensure that the field contains appropriate data. Checking can be done on a single field (for example, the field must be plus) or on multiple fields (for example, if FLDA contains a 1, FLDB can contain only a 2 or 3).

variable: A named modifiable value. The value can be accessed or modified by referring to the name of the variable.

vary off: To make a device, control unit, or line unavailable for its normal intended use.

vary on: To make a device, control unit, or line available for its normal intended use.

volume: A storage medium that is mounted and demounted as a unit; for example, magnetic tape or diskette.

work entry: An entry in a subsystem description that specifies a source from which jobs can be accepted to be executed in the subsystem.

work station: A device that lets a person transmit information to or receive information from a computer as needed to perform his job.

work station controller: A device in the system unit that provides for a direct connection of local work stations to the system.

work station entry: A work entry in a subsystem description that specifies the work stations from which users can sign on to the subsystem or from which interactive jobs can transfer to the subsystem.

work station message queue: A message queue that is associated with a particular work station and that is used for sending and receiving messages sent to the work station. The name of the message queue is the same as the name of the work station.

work station user: A person who uses a work station to communicate with System/38.

work station user profile: The CPF-supplied user profile that has the authority necessary for work station users. Named QUSER.

working display: See *basic working display.*

wrap test: For BSC, a test that checks attachment or control unit circuitry (without checking the mechanism itself) by returning the output of the mechanism as input. For example, when unrecoverable, communications adapter, machine errors occur, the wrap test transmits a specific character pattern to or through the modem in a loop and then compares the character pattern received to what was transmitted.

writer: (1) A CPF program that writes spooled output files from an output queue to an external device, such as a printer. (2) In RJEF, a program that receives output data (files) from the host system.

zoned decimal format: Representation of a decimal value by 1 byte per digit. Bits 0 through 3 of the rightmost byte represent the sign; bits 0 through 3 of all other bytes represent the zone portion; bits 4 through 7 of all bytes represent the numeric portion. For example, in zoned decimal format, the decimal value of +123 is represented as 1111 0001 1111 0010 1111 0011. Contrast with *packed decimal format.*

zoned field: A field that contains data in the zoned decimal format.

Index

This index contains all of the CL commands in alphabetic order by their descriptive (English-form) names; the command name (mnemonic) follows the descriptive name. You can retrieve the commands alphabetically *by mnemonic* by using either the table of contents or the marginal keys in the corner of each page of the command descriptions.

Certain index entries contain lists of commands that the entry or subentry describes. In some cases, the list of command names is not complete; it contains only a representative group of the commands identified by the entry. An asterisk (*) preceding an entry or subentry indicates that only a representative listing follows.

READER'S COMMENT FORM

IBM System/38
Control Language
Reference Manual

SC21-7731-5

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your nearest IBM branch office to request additional publications.

Name ______________________

Company or
Organization ______________________

Address ______________________

____

City State Zip Code

SC21-7731-5

Cut Along Line

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N. Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM CORPORATION
Publications, Department 245
Rochester, Minnesota 55901

Fold and tape

Please do not staple

Fold and tape

IBM

IBM System/38 Control Language Reference Manual (File No. S38-36) Printed in U.S.A. SC21-7731-5