

SC21-7731-5

File No. S38-36

IBM System/38

IBM System/38 Control Language Reference Manual

Program Number 5714-SS1



Technical Newsletter

This Newsletter No. SN21-8291

Date 10 September 1982

Base Publication No. SC21-7731-5

File No. S38-36

Previous Newsletters None

IBM System/38 Control Language Reference Manual

©IBM Corp. 1979, 1980, 1981, 1982

This technical newsletter applies to release 4, modification 1 of the IBM System/38 Control Program Facility (Program 5714-SS1) and provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

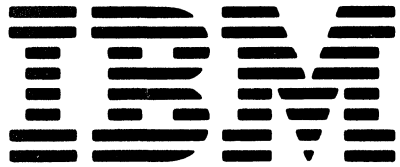
4-61 through 4-64
4-119, 4-120
4-305 through 4-308
4-423 through 4-428
4-431, 4-432
4-791, 4-792
4-792.1, 4-792.2 (added)
4-1211, 4-1212

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

- Support for IBM 5291 Display Station
- Support for IBM 5292 Color Display Station

Note: Please file this cover letter at the back of the manual to provide a record of changes.



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.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual enterprise is entirely coincidental.

Use this publication only for the purpose stated in *About This Manual*.

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Contents

ABOUT THIS MANUAL	ix
Purpose of This Manual	ix
Organization of This Manual	ix
What You Should Know	xi
If You Need More Information	xi
System/38 Overview Information	xi
Control Language Commands	xi
Data Description Specifications	xi
Messages	xi
Languages	xi
Communications	xii
Utilities	xii
System Operation	xii
Installation and Device Configuration	xii
Problem Determination	xii
Content and Use of System/38 Publications	xii

PART 1. CONTROL LANGUAGE FUNCTIONS AND SYNTAX

CHAPTER 1. SUMMARY OF CPF FUNCTIONS AND OBJECT TYPES	1-1
CPF Object Types	1-1
CPF-Provided Libraries	1-4
Commands Operating on CPF Objects	1-5
Commands Operating on Specific Object Types	1-6
Commands Operating on Multiple Object Types	1-7
Command Groups (By Function)	1-8
Object and Library Commands	1-9
Data Base Commands	1-10
Device File Commands	1-11
Device Management Commands	1-12
Programming Commands	1-13
Program Debug Commands	1-14
Message Handling Commands	1-14
Input/Output Spooling Commands	1-15
System and Job Control Commands	1-16
Subsystem Description, Job Description, and Class Commands	1-17
Configuration Commands	1-18
Utility Commands	1-19
Security Commands	1-19
Save/Restore Commands	1-19
Command Definition Commands	1-20
Service Commands	1-20
Remote Job Entry Facility Commands	1-21
Master Command Matrix Chart	1-22
CHAPTER 2. CONTROL LANGUAGE SYNTAX	2-1
Parts of a Command	2-1
Command Label	2-1
Command Name	2-2
Command Parameters	2-2
Command Syntax	2-5
Command Delimiters	2-5
Command Continuation	2-7
Entering Comments	2-7
Control Language Character Set	2-8

Special Characters and Predefined Values	2-9
Summary of Special Character Usage	2-9
Predefined Values	2-11
Rules for Specifying Names	2-12
Identifying CPF Objects	2-13
Simple and Qualified Object Names	2-13
Generic Object Names	2-13
CPF Object Naming Rules	2-14
Parameter Values	2-16
Constant Values	2-17
Variables	2-20
Expressions	2-21
Lists of Values	2-22
Syntax Coding Rules (Summary)	2-23

PART 2. CONTROL LANGUAGE COMMAND DESCRIPTIONS

CHAPTER 3. FORMAT OF COMMAND DESCRIPTIONS	3-1
How Commands Are Described	3-1
Command Description	3-1
Command Syntax	3-1
Parameter Descriptions	3-2
Command Coding Examples	3-2
Additional Command Considerations	3-2
How to Interpret Syntax Diagrams	3-3
Sample Syntax Diagram	3-3
Syntax Diagram Rules	3-5
CHAPTER 4. COMMAND DESCRIPTIONS	4-1
ADDAJE (Add Autostart Job Entry) Command	4-1
ADDDBKP (Add Breakpoint) Command	4-3
ADDFCTE (Add Forms Control Table Entry) Command	4-9
ADDJOBQE (Add Job Queue Entry) Command	4-17
ADDLFM (Add Logical File Member) Command	4-20
ADDMSGD (Add Message Description) Command	4-25
ADDPFM (Add Physical File Member) Command	4-39
ADDPGM (Add Program) Command	4-41
ADDRJECMNE (Add RJE Communications Entry) Command	4-43
ADDRJERDRE (Add RJE Reader Entry) Command	4-45
ADDRJEWTRE (Add RJE Writer Entry) Command	4-47
ADDRTGE (Add Routing Entry) Command	4-52
ADDTRC (Add Trace) Command	4-56
ADDWSE (Add Work Station Entry) Command	4-62
ALCOBJ (Allocate Object) Command	4-66
ANSLIN (Answer Line) Command	4-70
APYJRNCHG (Apply Journalled Changes) Command	4-71
APYPGMCHG (Apply Programming Change) Command	4-77

CRTRPGM (Create RPG Program) Command	4-561	DSNDFUAPP (Design DFU Application) Command	4-678
CRTRPTPGM (Create Auto Report Program) Command	4-568	DSNFMT (Design Format) Command	4-680
CRTSBSD (Create Subsystem Description) Command	4-576	DSNQRYP (Design Query Application) Command	4-682
CRTSRCPF (Create Source Physical File) Command	4-579	DSPACTJOB (Display Active Jobs) Command	4-684
CRTSSND (Create Session Description) Command	4-589	DSPAUTUSR (Display Authorized Users) Command	4-692
CRTTAPF (Create Tape File) Command	4-593	DSPBKP (Display Breakpoints) Command	4-694
CRTTBL (Create Table) Command	4-608	DSPCLS (Display Class) Command	4-698
CRTUSRPRF (Create User Profile) Command	4-610	DSPCMD (Display Command) Command	4-700
		DSPCNPA (Display CSNAP Attributes) Command	4-702
		DSPCTLSTS (Display Control Unit Status) Command	4-704
CVTDAT (Convert Date) Command	4-615	DSPCUD (Display Control Unit Description) Command	4-709
DATA (Data) Command	4-618	DSPDBG (Display Debug) Command	4-712
		DSPDBR (Display Data Base Relations) Command	4-715
DCL (Declare CL Variable) Command	4-620	DSPDEVCFG (Display Device Configuration) Command	4-723
DCLDTAARA (Declare Data Area) Command	4-623	DSPDEVD (Display Device Description) Command	4-726
DCLF (Declare File) Command	4-624	DSPDEVSTS (Display Device Status) Command	4-729
		DSPDKT (Display Diskette) Command	4-734
DLCOBJ (Deallocate Object) Command	4-627	DSPDTA (Display Data) Command	4-746
		DSPDTAARA (Display Data Area) Command	4-747
DLTCLS (Delete Class) Command	4-629	DSPEDTD (Display Edit Description) Command	4-749
DLTCMD (Delete Command) Command	4-630	DSPFCT (Display Forms Control Table) Command	4-751
DLTCUD (Delete Control Unit Description) Command	4-631	DSPFD (Display File Description) Command	4-755
DLTDEVD (Delete Device Description) Command	4-632	DSPFFD (Display File Field Description) Command	4-775
DLTDFUAPP (Delete DFU Application) Command	4-633	DSPJOB (Display Job) Command	4-782
DLTDKTLBL (Delete Diskette Label) Command	4-634	DSPJOBBD (Display Job Description) Command	4-794
DLTDTAARA (Delete Data Area) Command	4-637	DSPJOBQ (Display Job Queue) Command	4-797
DLTEDTD (Delete Edit Description) Command	4-638	DSPJRN (Display Journal) Command	4-802
DLTF (Delete File) Command	4-639	DSPJRNA (Display Journal Attributes) Command	4-813
DLTFCT (Delete Forms Control Table) Command	4-640	DSPJRNRCVA (Display Journal Receiver Attributes) Command	4-817
DLTJOBBD (Delete Job Description) Command	4-641	DSPLIB (Display Library) Command	4-820
DLTJOBQ (Delete Job Queue) Command	4-642	DSPLIBL (Display Library List) Command	4-823
DLTJRN (Delete Journal) Command	4-643	DSPLIND (Display Line Description) Command	4-825
DLTJRNRCV (Delete Journal Receiver) Command	4-644	DSPLNSTS (Display Line Status) Command	4-828
DLTLIB (Delete Library) Command	4-645	DSPLOG (Display Log) Command	4-833
DLTLIND (Delete Line Description) Command	4-646	DSPMSG (Display Messages) Command	4-838
DLTMSGF (Delete Message File) Command	4-647	DSPMSGD (Display Message Description) Command	4-843
DLTMSGQ (Delete Message Queue) Command	4-648	DSPMSGF (Display Message File) Command	4-849
DLTOUTQ (Delete Output Queue) Command	4-649	DSPOBJAUT (Display Object Authority) Command	4-853
DLTOVR (Delete Override) Command	4-650	DSPOBJD (Display Object Description) Command	4-856
DLTPGM (Delete Program) Command	4-653	DSPOBJLCK (Display Object Lock) Command	4-864
DLTPRTIMG (Delete Print Image) Command	4-654	DSPOUTQ (Display Output Queue) Command	4-870
DLTQRYAPP (Delete Query Application) Command	4-655	DSPPOVR (Display Override) Command	4-875
DLTSBSD (Delete Subsystem Description) Command	4-656	DSPPPGMCHG (Display Programming Change) Command	4-882
DLTSSND (Delete Session Description) Command	4-657	DSPPPGMREF (Display Program References) Command	4-888
DLTTBL (Delete Table) Command	4-658	DSPPPGMVAR (Display Program Variable) Command	4-894
DLTUSRPRF (Delete User Profile) Command	4-659	DSPRDR (Display Reader) Command	4-904
		DSPRJESSN (Display RJE Session) Command	4-908
DMPCLPGM (Dump CL Program) Command	4-660		
DMPJOB (Dump Job) Command	4-661		
DMPJOBINT (Dump Job Internal) Command	4-663		
DMPOBJ (Dump Object) Command	4-664		
DMPYSOBY (Dump System Object) Command	4-665		
DMPTAP (Dump Tape) Command	4-671		
DO (Do) Command	4-677		

DSPSBMJOB (Display Submitted Jobs) Command	4-916	LODPGMCHG (Load Programming Change) Command	4-1045
DSPSBS (Display Subsystem) Command	4-919	LOGDBF (Log Data Base File) Command	4-1048
DSPSBSD (Display Subsystem Description) Command	4-923	LSTCMDUSG (List Command Usage) Command	4-1051
DSPSPLF (Display Spooled File) Command	4-929	LSTCNPDTA (List CSNAP Data) Command	4-1053
DSPSPLFA (Display Spooled File Attributes) Command	4-934	LSTCNPHST (List CSNAP History) Command	4-1056
DSPSRVSTS (Display Service Status) Command	4-938	LSTERRLOG (List Error Log) Command	4-1059
DSPSSND (Display Session Description) Command	4-940	LSTINTDTA (List Internal Data) Command	4-1062
DSPSYS (Display System) Command	4-946	MONMSG (Monitor Message) Command	4-1064
DSPSYSSTS (Display System Status) Command	4-949	MOV OBJ (Move Object) Command	4-1068
DSPSYSVAL (Display System Value) Command	4-953	OVRBSCF (Override with BSC File) Command	4-1070
DSPTAP (Display Tape) Command	4-955	OVRMNF (Override with Communications File) Command	4-1078
DSPTRC (Display Trace) Command	4-961	OVRCRDF (Override with Card File) Command	4-1082
DSPTRCDTA (Display Trace Data) Command	4-963	OVRDBF (Override with Data Base File) Command	4-1089
DSPUSRPRF (Display User Profile) Command	4-965	OVRDKTF (Override with Diskette File) Command	4-1098
DSPWTR (Display Writer) Command	4-969	OVRDSPF (Override with Display File) Command	4-1107
DUPDKT (Duplicate Diskette) Command	4-974	OVRMSGF (Override with Message File) Command	4-1111
EDTSRC (Edit Source) Command	4-977	OVRPRTF (Override with Printer File) Command	4-1113
ELSE (Else) Command	4-980	OVRTAPF (Override with Tape File) Command	4-1125
ENDCBLDBG (End COBOL Debug) Command	4-982	PCHPGM (Patch Program) Command	4-1139
ENDDBG (End Debug) Command	4-983	PGM (Program) Command	4-1143
ENDDO (End Do) Command	4-984	PRPAPAR (Prepare APAR) Command	4-1145
ENDINP (End Input) Command	4-985	PWRCTLU (Power Control Unit) Command	4-1149
ENDJOB (End Job) Command	4-986	PWRDEV (Power Device) Command	4-1150
ENDJRNPF (End Journaling Physical File Changes) Command	4-987	PWRDWN SYS (Power Down System) Command	4-1151
ENDLOG (End Logging) Command	4-989	QRYDTA (Query Data) Command	4-1153
ENDPGM (End Program) Command	4-990	RCLRSC (Reclaim Resources) Command	4-1155
ENDSRV (End Service) Command	4-991	RCLSTG (Reclaim Storage) Command	4-1158
ENTCBLDBG (Enter COBOL Debug) Command	4-992	RCVDTAARA (Receive Data Area) Command	4-1161
ENTDBG (Enter Debug) Command	4-993	RCVF (Receive File) Command	4-1162
FMTDTA (Format Data) Command	4-996	RCVMSG (Receive Message) Command	4-1165
FMRJEDTA (Format RJE Data) Command	4-999	RETURN (Return) Command	4-1174
GOTO (Go To) Command	4-1003	RGZPFM (Reorganize Physical File Member) Command	4-1175
GRTOBJAUT (Grant Object Authority) Command	4-1004	RLSJOB (Release Job) Command	4-1179
GRTUSRAUT (Grant User Authority) Command	4-1007	RLSJOBQ (Release Job Queue) Command	4-1180
HLDJOB (Hold Job) Command	4-1009	RLSOUTQ (Release Output Queue) Command	4-1181
HLDJOBQ (Hold Job Queue) Command	4-1011	RLSRDR (Release Reader) Command	4-1182
HLDOUTQ (Hold Output Queue) Command	4-1012	RLSSPLF (Release Spooled File) Command	4-1183
HLD RDR (Hold Reader) Command	4-1013	RLSWTR (Release Writer) Command	4-1185
HLDSPLF (Hold Spooled File) Command	4-1014	RMVAJE (Remove Autostart Job Entry) Command	4-1187
HLDWTR (Hold Writer) Command	4-1016	RMVBKP (Remove Breakpoint) Command	4-1188
IF (If) Command	4-1017		
INZDKT (Initialize Diskette) Command	4-1022		
INZPFM (Initialize Physical File Member) Command	4-1029		
INZTAP (Initialize Tape) Command	4-1031		
JOB (Job) Command	4-1035		
JRNPF (JOURNAL Physical File) Command	4-1043		

RMVFCTE (Remove Forms Control Table Entry) Command	4-1190	SIGNOFF (Sign Off) Command	4-1309
RMVJOBQE (Remove Job Queue Entry) Command	4-1192	SNDBRKMSG (Send Break Message) Command	4-1310
RMVJRNCHG (Remove Journaled Changes) Command	4-1193	SNDDTAARA (Send Data Area) Command	4-1312
RMVM (Remove Member) Command	4-1198	SNDF (Send File) Command	4-1313
RMVMSG (Remove Message) Command	4-1199	SNDJRNE (Send Journal Entry) Command	4-1315
RMVMSGD (Remove Message Description) Command	4-1201	SNDMSG (Send Message) Command	4-1317
RMVPGM (Remove Program) Command	4-1202	SNDPGMMSG (Send Program Message) Command	4-1319
RMVPGMCHG (Remove Programming Change) Command	4-1203	SNDRCVF (Send/Receive File) Command	4-1326
RMVRJECMNE (Remove RJE Communications Entry) Command	4-1205	SNDRPY (Send Reply) Command	4-1329
RMVRJERDRE (Remove RJE Reader Entry) Command	4-1206	SRVJOB (Service Job) Command	4-1331
RMVRJEWTR (Remove RJE Writer Entry) Command	4-1207	STRCNFCHK (Start Confidence Check) Command	4-1332
RMVRTGE (Remove Routing Entry) Command	4-1208	STRCRDRDR (Start Card Reader) Command	4-1334
RMVTRC (Remove Trace) Command	4-1209	STRCRDWTR (Start Card Writer) Command	4-1337
RMVWSE (Remove Work Station Entry) Command	4-1211	STRDBRDR (Start Data Base Reader) Command	4-1341
RNMDKT (Rename Diskette) Command	4-1212	STRDKTRDR (Start Diskette Reader) Command	4-1344
RNMOBJ (Rename Object) Command	4-1215	STRDKTWTR (Start Diskette Writer) Command	4-1348
RPLLIBL (Replace Library List) Command	4-1217	STRPDP (Start Problem Determination Procedure) Command	4-1353
RRTJOB (Reroute Job) Command	4-1218	STRPRTWTR (Start Printer Writer) Command	4-1356
RSMBKP (Resume Breakpoint) Command	4-1220	STRRJECSL (Start RJE Console) Command	4-1360
RSTAUT (Restore Authority) Command	4-1221	STRRJERDR (Start RJE Reader) Command	4-1363
RSTLIB (Restore Library) Command	4-1222	STRRJESSN (Start RJE Session) Command	4-1365
RSTOBJ (Restore Object) Command	4-1229	STRRJEWTR (Start RJE Writer) Command	4-1367
RSTUSRPRF (Restore User Profiles) Command	4-1238	STRSBS (Start Subsystem) Command	4-1373
RTVCLSRC (Retrieve CL Source) Command	4-1242	TFRCTL (Transfer Control) Command	4-1375
RTVDFUSRC (Retrieve DFU Source) Command	4-1243	TFRJOB (Transfer Job) Command	4-1377
RTVDTAARA (Retrieve Data Area) Command	4-1245	TRCINT (Trace Internal) Command	4-1379
RTVJOBA (Retrieve Job Attributes) Command	4-1248	TRCJOB (Trace Job) Command	4-1383
RTVMSG (Retrieve Message) Command	4-1251	TRMCPF (Terminate Control Program Facility) Command	4-1386
RTVQRYSRC (Retrieve Query Source) Command	4-1255	TRMRJESSN (Terminate RJE Session) Command	4-1388
RTVSYVAL (Retrieve System Value) Command	4-1257	TRMSBS (Terminate Subsystem) Command	4-1390
RVKOJAUT (Revoke Object Authority) Command	4-1258	VFYPRT (Verify Printer) Command	4-1393
SAVCHGOBJ (Save Changed Object) Command	4-1261	VRCTLU (Vary Control Unit) Command	4-1394
SAVLIB (Save Library) Command	4-1269	VRYDEV (Vary Device) Command	4-1396
SAVOBJ (Save Object) Command	4-1275	VRYLIN (Vary Line) Command	4-1397
SAVSYS (Save System) Command	4-1282	WAIT (Wait) Command	4-1398
SBMCRDJOB (Submit Card Jobs) Command	4-1286		
SBMDBJOB (Submit Data Base Jobs) Command	4-1289		
SBMDKTJOB (Submit Diskette Jobs) Command	4-1292		
SBMJOB (Submit Job) Command	4-1298		
SBMRJEJOB (Submit RJE Job) Command	4-1305		

CHAPTER 5. COMMAND DEFINITION

STATEMENTS 5-1
Creating User-Defined Commands 5-1
Command Definition Statement Descriptions 5-2
 CMD (Command) Statement 5-2
 PARM (Parameter) Statement 5-3
 ELEM (Element) Statement 5-19
 QUAL (Qualifier) Statement 5-32
 DEP (Dependent) Statement 5-41

PART 3. APPENDIXES

APPENDIX A. EXPANDED PARAMETER

DESCRIPTIONS A-1
CLS Parameter A-2
EXCHTYPE Parameter A-4
FILETYPE Parameter A-5
FRCRATIO Parameter A-7
Operations Using Generic Functions A-8
JOB Parameter A-9
LABEL Parameter A-11
LOC Parameter A-14
MAXACT Parameter A-19
OBJ Parameter A-20
OBJTYPE Parameter A-21
OUTPUT Parameter A-23
PUBAUT Parameter A-24
Scheduling Priority Parameters (JOBPTY,
 OUTPTY, PTYLMT) A-26
SEV Parameter A-28
SPLNBR Parameter A-30
TEXT Parameter A-31
VOL Parameter A-32
WAITFILE Parameter A-36

APPENDIX B. EXPRESSIONS B-1

Operators in Expressions B-2
Priority of Operators When Evaluating
 Expressions B-4
Arithmetic Expressions B-4
Character String Expressions B-5
Relational Expressions B-7
Logical Expressions B-8
%SUBSTRING Built-In Function B-9
%SWITCH Built-In Function B-10

APPENDIX C. USER PROFILE MATRIX CHART C-1

APPENDIX D. FILES USED BY CL COMMANDS D-1

**APPENDIX E. ERROR MESSAGES THAT CAN BE
MONITORED E-1**

**APPENDIX F. COMMAND AND KEYWORD
ABBREVIATIONS F-1**

GLOSSARY OF TERMS AND ABBREVIATIONS G-1

INDEX X-1

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 appendices.

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	LSTCNPST
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	*JOBDD	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 Verbs	Other Associated Commands
			Create	Delete	Change	Override	Display		
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	CHGSQLFA, 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, INZPFM, RGZPFM
	Source physical file	SRCPF	CRT		CHG				
	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	JOB	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, SNDBRMSG, SNDPGMMMSG, SNDRPY
17.	Output queue	OUTQ	CRT	DLT	CHG		DSP	CLR HLD RLS	
18.	Print image	PRTIMG	CRT	DLT					
19.	Program	PGM		DLT				END DMP	CALL, TFRCTL
	CL program	CLPGM	CRT						
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 MOV OBJ 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 MOV OBJ 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)
MOV OBJ	(Move Object)
RNMOBJ	(Rename Object)
RSTOBJ	(Restore Object)
SAVCHGOBJ	(Save Changed Objects)
SAVOBJ	(Save Object)

Library List

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

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)

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)
CRTLFL	(Create Logical File)

Journal

APYJRNCHG	(Apply Journalized 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 Journalized 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)
MOV OBJ (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)
MOV OBJ (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)

CRTCLPGM (Create COBOL Program)

Data Area

CHGDTAARA (Change Data Area)
CRTDTAARA (Create Data Area)
DLDTAARA (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)
MOV OBJ (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)
MOV OBJ (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)
MOV OBJ (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)
HLDSPFL (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

CHGJOB (Change Job Description)
CRTJOB (Create Job Description)
DLTJOB (Delete Job Description)
DSPJOB (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

CHGDEV (Change Device Description)

CRTDEV (Create Device Description)

DLTDEV (Delete Device Description)

DSPDEV (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)

MOV OBJ (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

CRTE (Create Edit Description)

DLTE (Delete Edit Description)

DSPE (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

CRTPTIMG (Create Print Image)

DLTPTIMG (Delete Print Image)

Common Functions for Print Image

CHKOBJ (Check Object)

DSPOBJD (Display Object Description)

DSPOBJLCK (Display Object Locks)

MOV OBJ (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)

MOV OBJ (Move Object)

RNMOBJ (Rename Object)

RSTOBJ (Restore Object)

SAVCHGOBJ (Save Changed Objects)

SAVOBJ (Save Object)

UTILITY COMMANDS¹

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 Utility²

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)

¹These commands are part of the IBM System/38 Interactive Data Base Utilities Program.

²This command is part of the IBM System/38 Conversion Reformat Utility Licensed Program.

COMMAND DEFINITION COMMANDS

Command		Common Functions for Command	
CHGCMD	(Change Command)	CHKOBJ	(Check Object)
CRTCMD	(Create Command)	DSPOBJD	(Display Object Description)
DLTCMD	(Delete Command)	DSPOBJLCK	(Display Object Locks)
DSPCMD	(Display Command)	MOVOBJ	(Move Object)
LSTCMDUSG	(List Command Usage)	RNMOBJ	(Rename Object)
		RSTOBJ	(Restore Object)
		SAVCHGOBJ	(Save Changed Objects)
		SAVOBJ	(Save Object)

SERVICE COMMANDS

Job		Tape Volume	
DMPJOB	(Dump Job)	DMPTAP	(Dump Tape)
DSPACTJOB	(Display Active Jobs)		
DSPSRVSTS	(Display Service Status)	Internal Machine	
ENDSRV	(End Service)	DMPJOBINT	(Dump Job Internal)
SRVJOB	(Service Job)	LSTERRLOG	(List Error Log)
TRCJOB	(Trace Job)	LSTINTDTA	(List Internal Data)
		TRCINT	(Trace Internal)
Object		Problem Reporting	
DMPOBJ	(Dump Object)	PRPAPAR	(Prepare APAR)
DMPSYSOBJ	(Dump System Object)		
Device		Programming Change	
CHGCNPA	(Change CSNAP Attributes)	APYPGMCHG	(Apply Programming Change)
DSPCNPA	(Display CSNAP Attributes)	DSPPGMCHG	(Display Programming Change)
LSTCNPDTA	(List CSNAP Data)	LODPGMCHG	(Load Programming Change)
LSTCNPHST	(List CSNAP History)	PCHPGM	(Patch Program)
LSTERRLOG	(List Error Log)	RMVPGMCHG	(Remove Programming Change)
STRCNFCHK	(Start Confidence Check)		
STRPDP	(Start Problem Determination Procedure)		
Printer			
CLNPRT	(Clean Printer)		
VFYPRT	(Verify Printer)		

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

ADDRJEWTR (Add RJE Writer Entry)
CHGRJEWTR (Change RJE Writer Entry)
RMVRJEWTR (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	DEVDD	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	JOB	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)
Journalled 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					

¹See also Data Base File.

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

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)
Trace	TRC		ADD RMV		DSP		DSP (see DSPFD)
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:

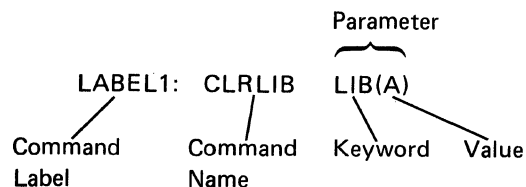
CALL	ELSE	PGM
DATA	GOTO	RETURN
DCL	IF	SIGNOFF
DO	JOB	WAIT

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 MOV OBJ 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 MOV OBJ 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

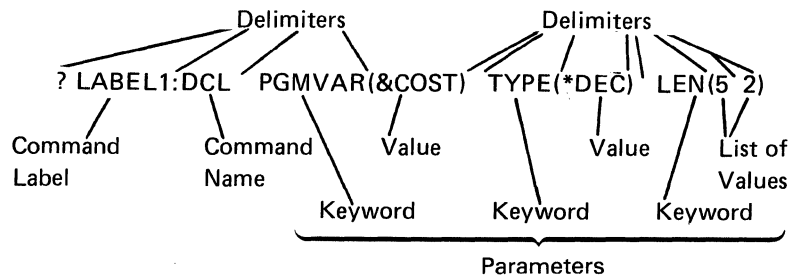
A command has the following general syntax. The brackets indicate that the item within them is optional; however, the parameter set may or may not be optional, depending upon the command.

[//] [?] [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-  
or +  
INUED')
```

{ The minus sign causes the leading blanks on the next line to be entered.

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

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:

```
MOV OBJ 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 <i>Summary of Special Character Usage</i>)

¹Lowercase 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.

²The 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	␣ 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.
<p>¹In this manual, ␣ is used when necessary to represent a blank space. ²Valid only when the QDATSEP system value specifies the same character.</p>		

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 <i>not</i> relational operator.
Equal	=	Symbolic <i>equal</i> relational operator.
Less than	<	Symbolic <i>less than</i> relational operator.
Greater than	>	Symbolic <i>greater than</i> 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.
<p>¹In 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.</p> <p>²In 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.</p> <p>³Valid only when the QDATSEP system value specifies the same character.</p>		

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 alphanumeric 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 alphanumeric (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 alphanumeric 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						
Simple object name	<table border="0"> <tr> <td style="text-align: right;">Object Name</td> <td style="text-align: left;">Library Name</td> </tr> <tr> <td colspan="2">object-name</td> </tr> </table>	Object Name	Library Name	object-name		INVENPGM1		
Object Name	Library Name							
object-name								
Qualified object name	<table border="0"> <tr> <td colspan="2">object-name.library-name</td> </tr> <tr> <td style="text-align: center;">└──────────┘</td> <td style="text-align: center;">└──┘</td> </tr> <tr> <td style="text-align: center;">10 characters maximum</td> <td style="text-align: center;">Connector</td> </tr> </table>	object-name.library-name		└──────────┘	└──┘	10 characters maximum	Connector	INVENPGM2.QGPL
object-name.library-name								
└──────────┘	└──┘							
10 characters maximum	Connector							
Generic name	<table border="0"> <tr> <td colspan="2">generic-name*</td> </tr> <tr> <td style="text-align: center;">Asterisk</td> <td style="text-align: center;">Connector</td> </tr> </table>	generic-name*		Asterisk	Connector	INV*		
generic-name*								
Asterisk	Connector							
Qualified generic name	<table border="0"> <tr> <td colspan="2">generic-name*.library-name</td> </tr> <tr> <td style="text-align: center;">└──────────┘</td> <td style="text-align: center;">└──────────┘</td> </tr> <tr> <td style="text-align: center;">9 characters maximum</td> <td style="text-align: center;">10 characters maximum</td> </tr> </table>	generic-name*.library-name		└──────────┘	└──────────┘	9 characters maximum	10 characters maximum	INV*.QGPL
generic-name*.library-name								
└──────────┘	└──────────┘							
9 characters maximum	10 characters maximum							

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.

OBJA.LIB1	}	three programs	OBJA.LIB1	}	two files
OBJA.LIB2			OBJA.LIB2		
OBJA.LIB3			OBJA.LIB1	}	one command

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 (alphanumeric 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 alphanumeric 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 alphanumeric 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	␣		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.	} Equivalent Values	+ .017
1.23		6278,954374
1,23		-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)) } Nested lists  
((A B)(1 2))
```

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 \textcircled{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.

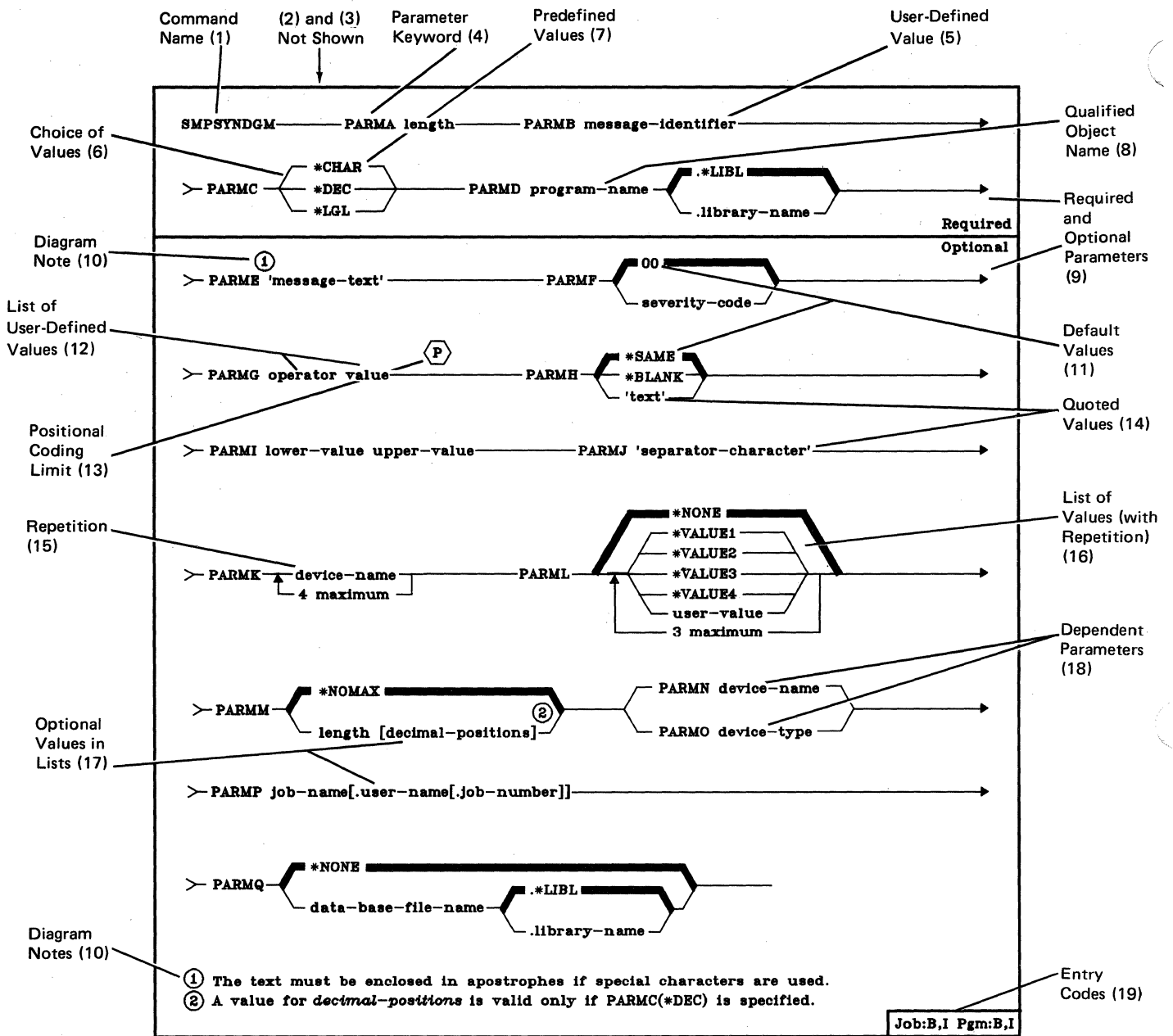
All required parameters precede all optional parameters. The required parameters (if any) are boxed with the command name at the beginning of the diagram. All the other parameters are optional and do not have to be coded; a default value (indicated by heavy branch lines) is assumed for each uncoded parameter,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



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.)

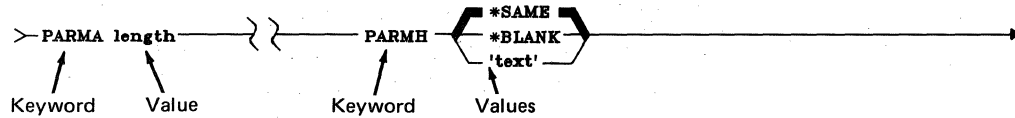
Note: In the few cases where dependent parameter relationships are shown (see rule 18), the positional order of those parameters may not be readily apparent. The order may be specified in a note in the diagram, or the order can be easily determined in the text, because all parameters in the command are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

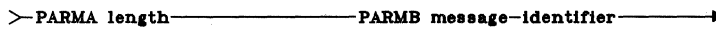


When a parameter is coded in keyword form, its associated values must be enclosed in parentheses. Although parentheses are not shown in the diagram, they must be used when you are coding in keyword form. Examples of PARMA and PARMH are:

```
PARMA(15) PARMH(*BLANK)
```

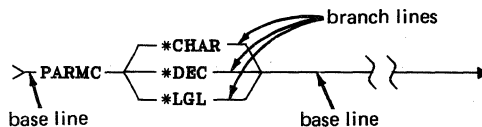
5. User-Defined Values

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:

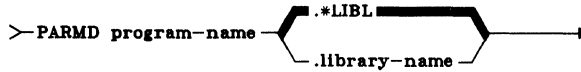
```
PARMC(*DEC)           or as           *DEC
(keyword form)         (positional form)
```

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 PARM shows that a qualified program name can be specified.

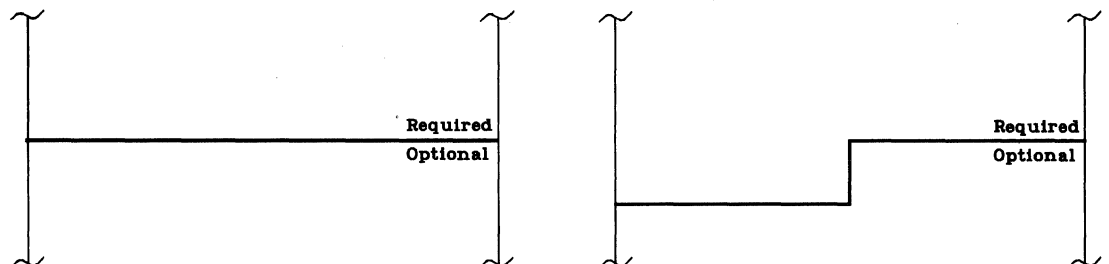
PARMD(PGMX.LIBA)

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.



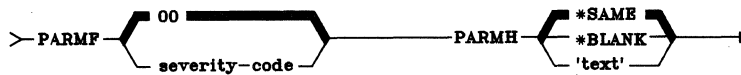
If there are no required parameters, no dividing line is used. If there are required parameters but no optional parameters, *Required* is entered at the top of the diagram.

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 } } PARMJ lower-value upper-value →

Both PARMG and PARMJ 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 PARMJ could be coded as:

PARMG(*EQ 16) PARMJ(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 alphanumeric character is specified.

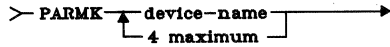
PARME('This is a quoted string')
PARME('10-24-78')
PARME(102478)

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 \leftarrow (n maximum) \rightarrow 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)
```

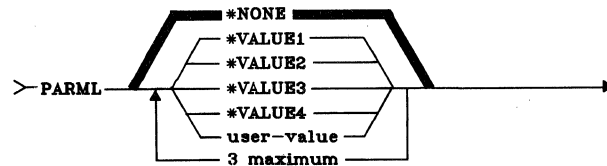
The first example specifies two device names, and the second example specifies the maximum of four device names.

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:



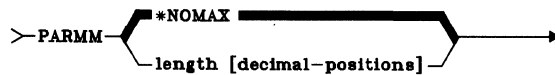
The parameter PARMK can specify one, two, or three of the values shown. Any combination of the four predefined values and user-defined values can be specified. The user-defined value could be any value allowed for that parameter. Any of the following could be coded:

```
PARMK(*VALUE1 *VALUE3)
PARMK(*VALUE3 16)
PARMK(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

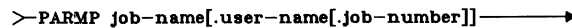
A list of values in which one or more of the elements are optional is shown with brackets ([]). The value within the brackets cannot be coded unless the value outside the brackets is also coded. The brackets themselves are never coded.



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.

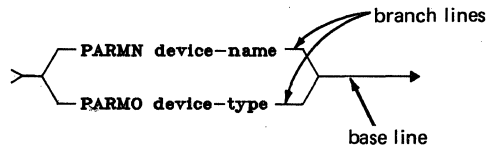


PARMP requires a job name as its value; the name can be optionally qualified. The job name can be specified with only the first part, the first two parts, or with all three parts. A job named JOBX owned by the user RANDY having the job number 210742 can be coded as:

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

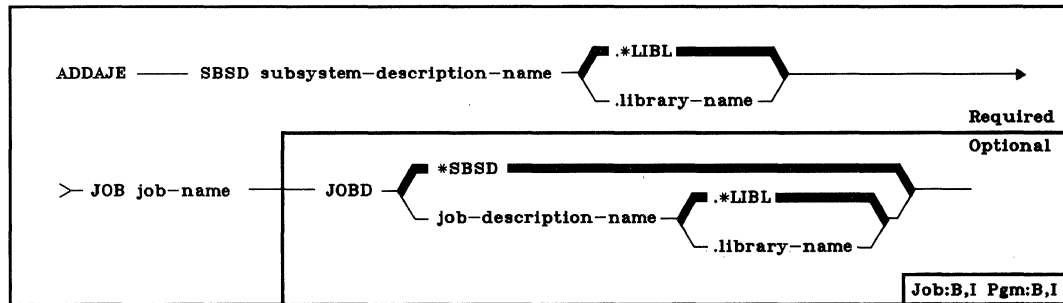
By looking at the entry codes at the bottom of each syntax diagram, you can tell whether the command can be used: only within CL programs (Pgm:B,I), only outside CL programs (Job:B,I), only within interactive jobs (Job:I), or inside or outside a CL program within any batch or interactive job (Job:B,I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

JOB 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.

ADDAJE
(Example)

Example

```
ADDAJE SBS(D(ACCTINT.ACCTLIB) JOB(ACCTINIT) +  
JOB(D(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.

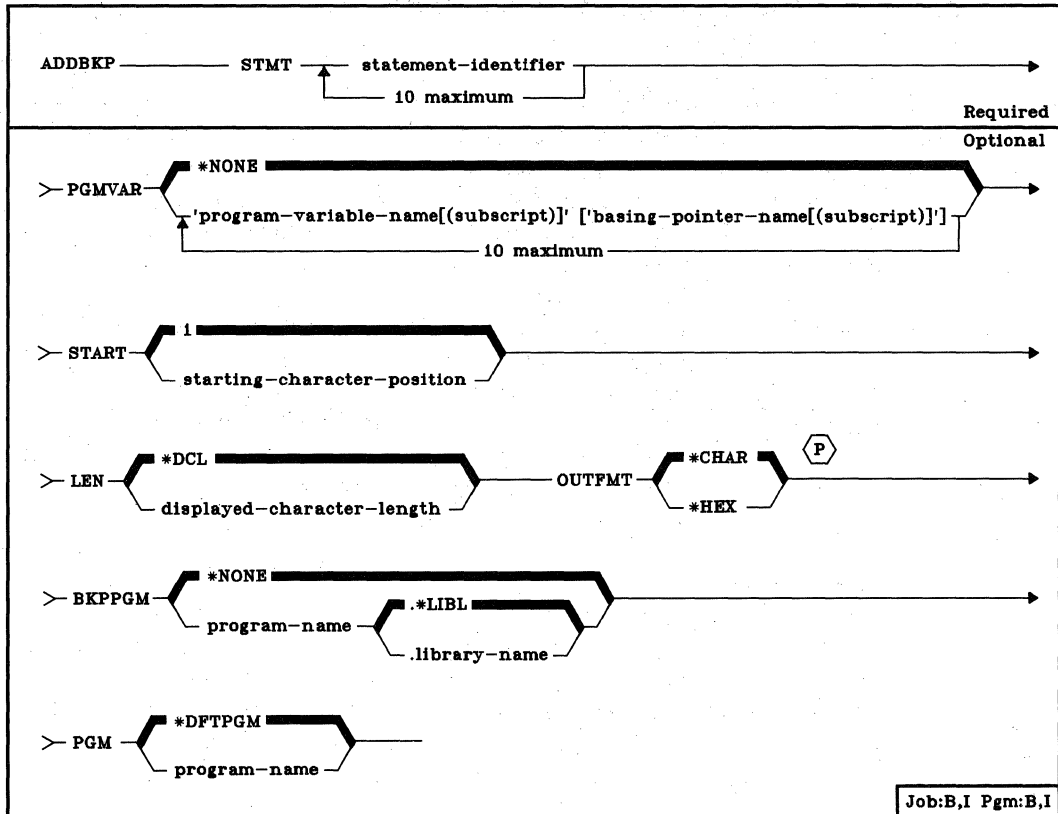
The ADDBKP command can also specify that the values of certain program variables are to be displayed or printed when any breakpoint in the command is reached. As many as 10 variables per breakpoint can be specified, and as many as 10 breakpoints per command can be set. However, the same program variables 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*.

ADDBKP
(Diagram)



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.

ADDBKP
START

- 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.

***CHAR:** 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.

***NONE:** 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.

ADDBKP
PGM

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

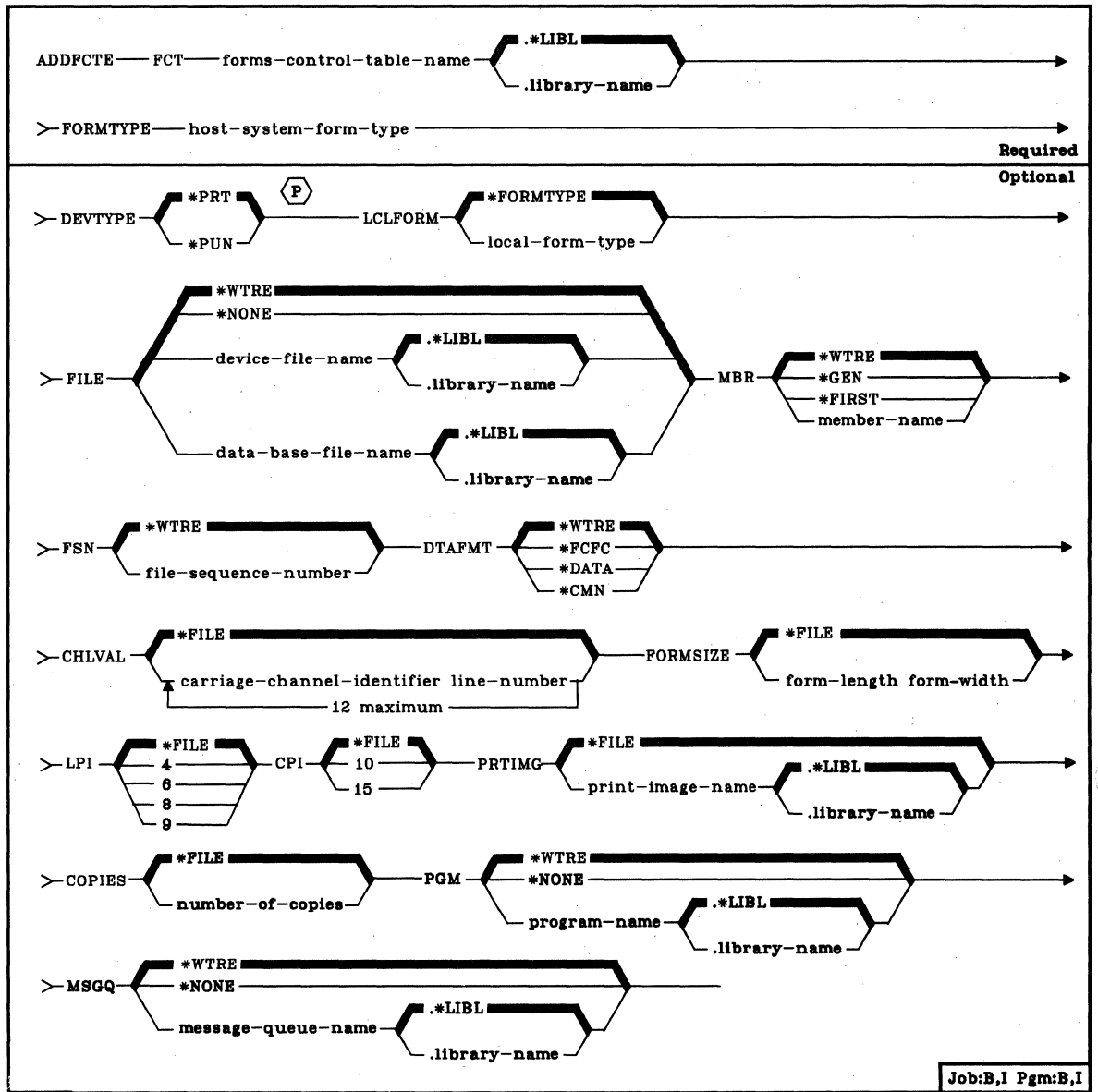
The Add Forms Control Table Entry (ADDFCTE) command adds a new forms entry to an existing forms control table (FCT). The FCT can contain up to 999 entries. Each FCT entry includes such forms-control attributes as:

- 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
(Diagram)



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.

***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.

***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.

***WTRE:** 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.)

ADDFCTE
MBR

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:

Affffffcc or Bffffffcc

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.

***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 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.

ADDFCTE
CHLVAL

CHLVAL Parameter: Specifies the printer carriage channel information.

***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.

***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 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.

***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.

***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.

***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.

***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.

***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.

***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.)

ADDFCTE
(Examples)

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.

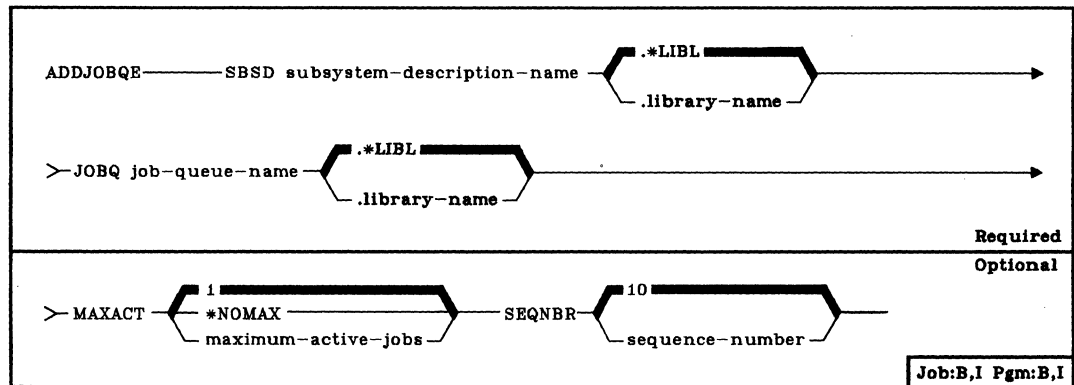
ADDJOBQE (Add Job Queue Entry) Command

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.



SBSID 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.

ADDJOBQE
MAXACT

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**
(Example)

```
ADDJOBQE SBS(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.)

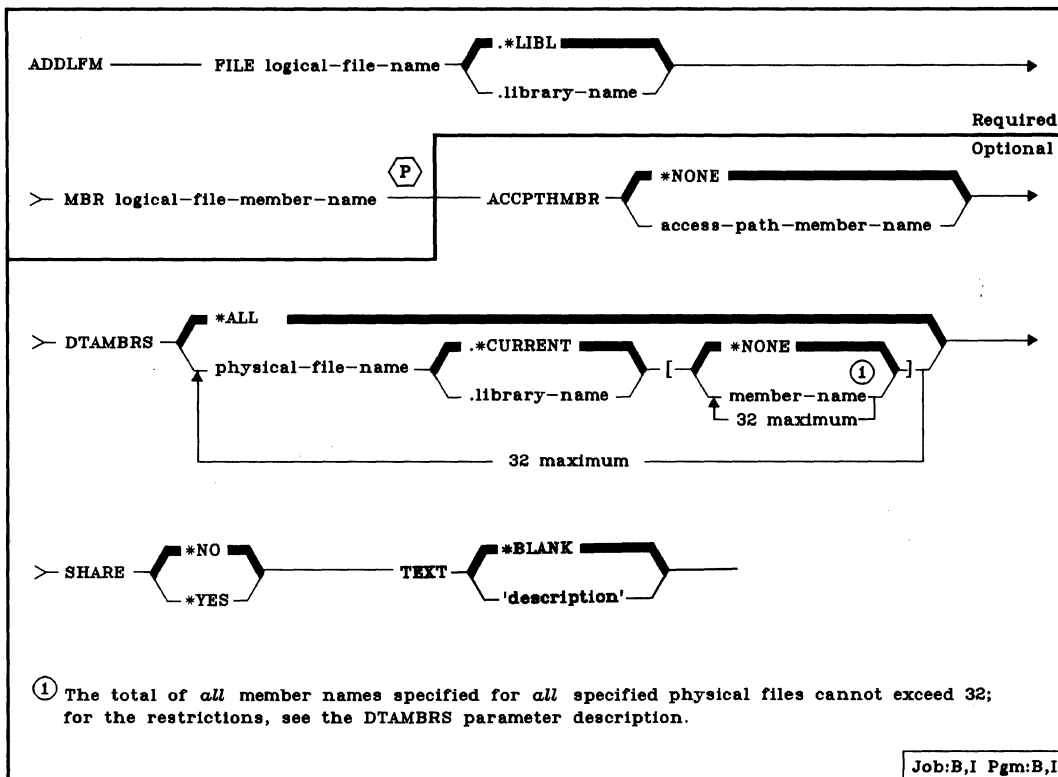
A logical file member can use the data from all, or a subset of, the physical files included in the scope of the logical file. Each member has its own set of data and can have its own access path (or share an access path) that provides an organization to that data.

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.

ADDLFM
(Diagram)



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.

ADDLFM
DTAMBRS

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.

***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 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, ***NONE** 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.

ADDLFM
SHARE

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) +  
DTAMBR((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

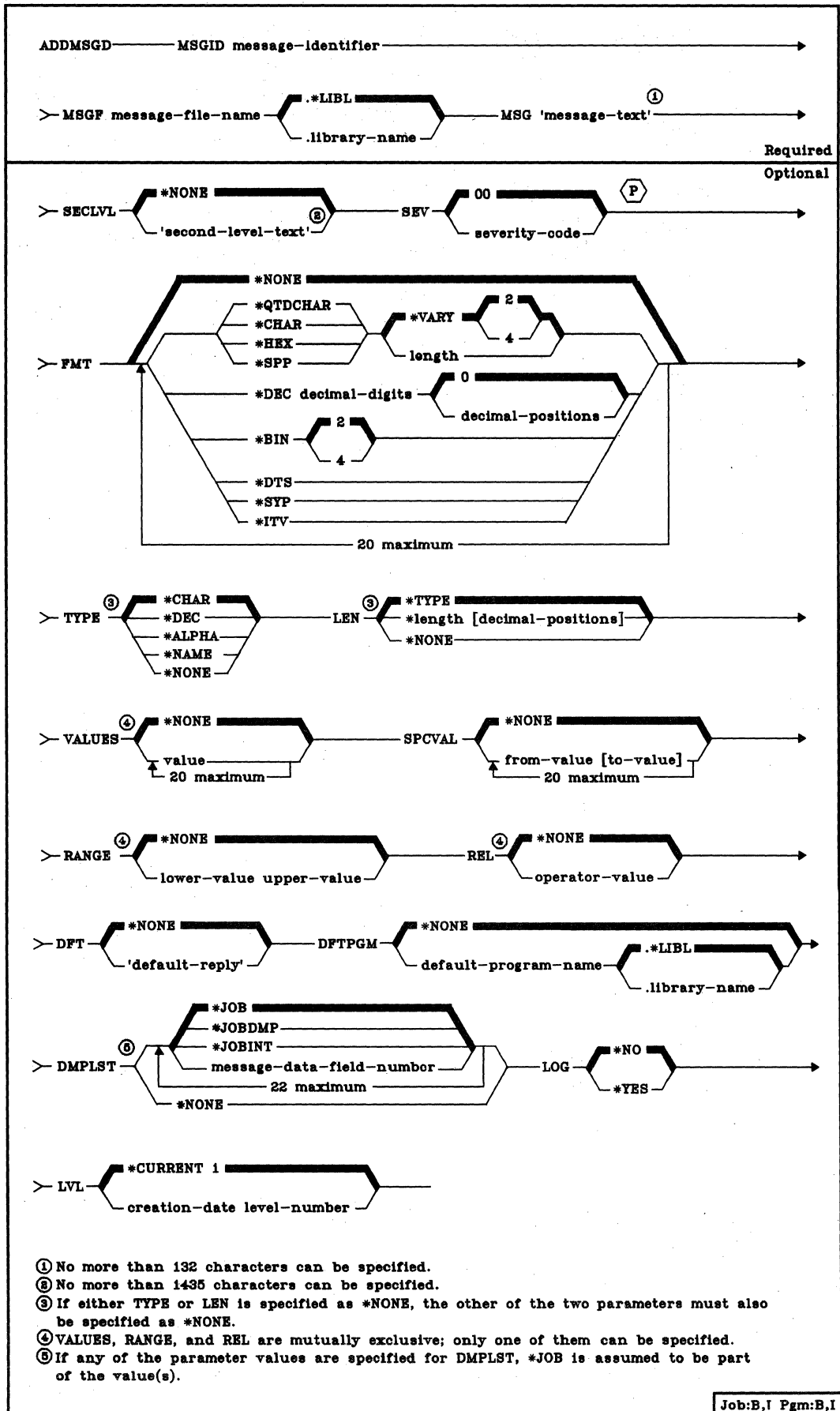
The Add Message Description (ADDMSGD) command describes a message and stores it in a message file for later use. The message description remains in the message file until the file is deleted or until the RMVMSGD command is used to remove it from the file. To change any of the attributes of the message 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 SNDPGMMMSG 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 SNDPGMMMSG 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.

ADDMSGD
(Diagram)



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 alphanumeric (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.

**ADDMSGD
MSGF**

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 QRPMSG, 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.

***NONE:** 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.

**ADDMSGD
FMT**

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 alphanumeric 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'COF4').

***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 ***VARY** 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 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 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 0 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 SNDPGMMMSG 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 alphanumeric.

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 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 alphanumeric 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.

ADDMSGD
SPCVAL

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.

*NONE: 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.

*NONE: 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.)

ADDMSGD
DMPLST

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.

***JOBDM:** 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 ***JOBDM**, ***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 *JOBDM)` results in the equivalent of `DMPLST(*JOB 1 2 *JOBDM)`.

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).

ADDMSGD
(Examples)

If a reply of YES is sent, it is accepted as a Y (SPCVL 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)
```

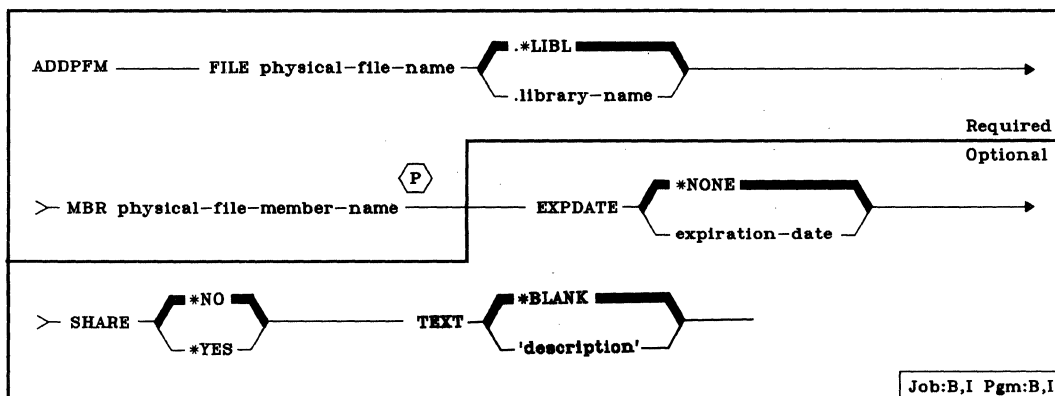
This command defines a message to be sent as an escape message. The sender of the message passes three data values, the first of which (employee serial number) is used as replacement text in the message. If the program to which this message is sent does not monitor for message UPY1234, default system 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.

ADDPFM
EXPDATE

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.)

***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 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.

***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 physical 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.

Examples

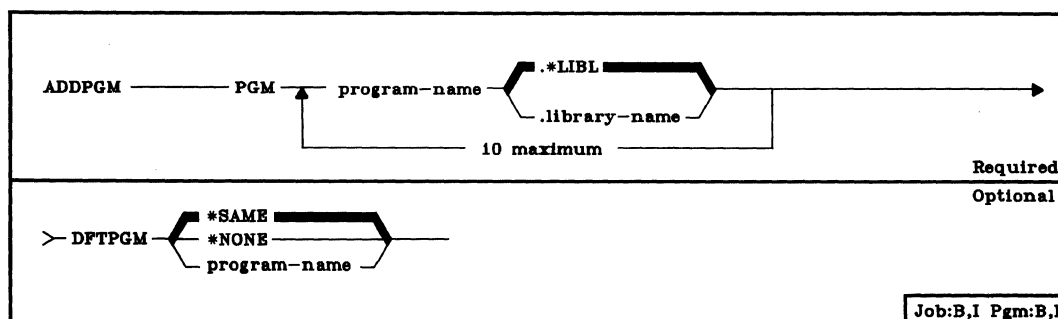
```
ADDPFM FILE(INVENTX) MBR(MONDAYTX) +  
      TEXT('Monday's Inventory Transactions')
```

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.

ADDPGM
(Example)

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:

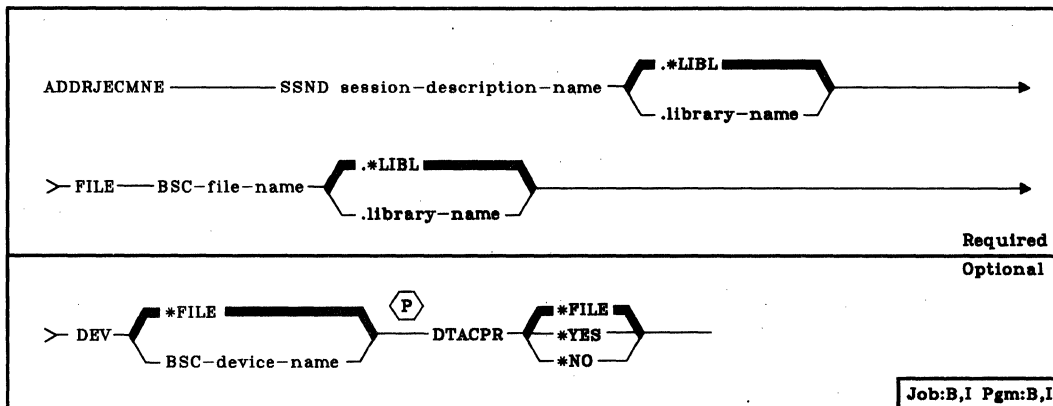
- One entry for an RJEF console input job
- One entry for an RJEF console output job

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.)

ADDRJECMNE
DEV

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

***FILE:** 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.

***FILE:** 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.

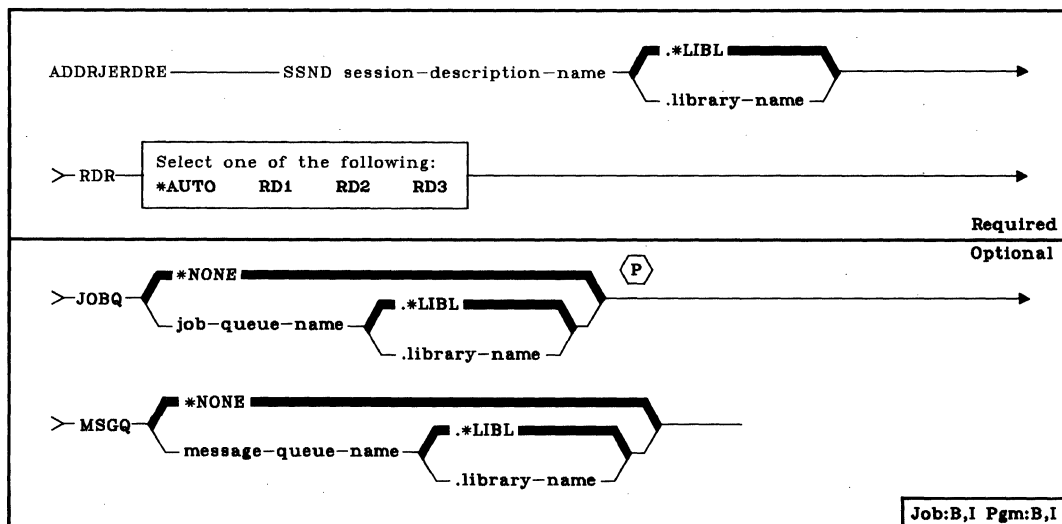
ADDRJERDRE (Add RJE Reader Entry) Command

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.

**ADDRJERDRE
JOBQ**

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.

ADDRJEWTR (Add RJE Writer Entry) Command

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

Each RJE writer entry requires a corresponding communications entry (refer to the Add RJE Communications Entry (ADDRJECMNE) command). A maximum of six RJE 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:

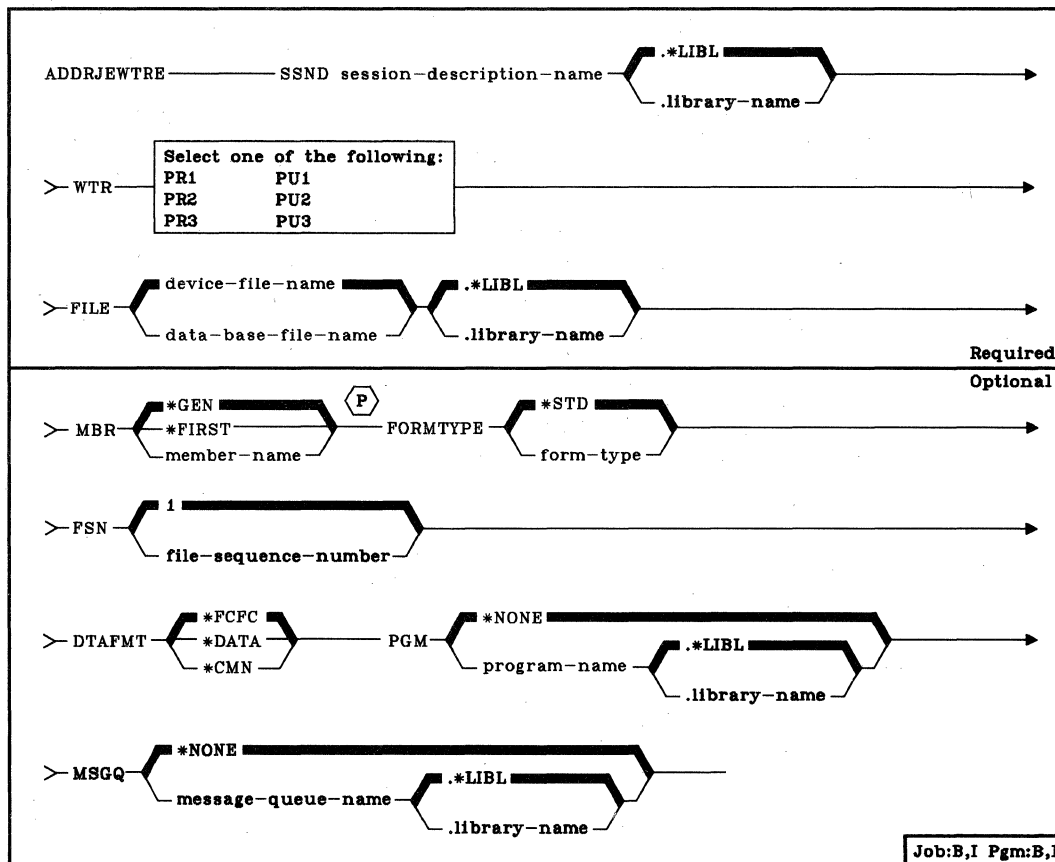
- There is no FCT associated with this session description.
- The forms mount message from the host system specifies a form type that does not exist as an entry in the FCT.
- *NONE is specified on the FILE parameter in the Add Forms Control Table Entry (ADDFCTE) command.
- A writer entry is used for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

ADDRJEWTR
(Diagram)



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.

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

Affffffcc or Bffffffcc

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.

**ADDRJEWTR
FORMTYPE**

***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

```
ADDRJEWTR 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).

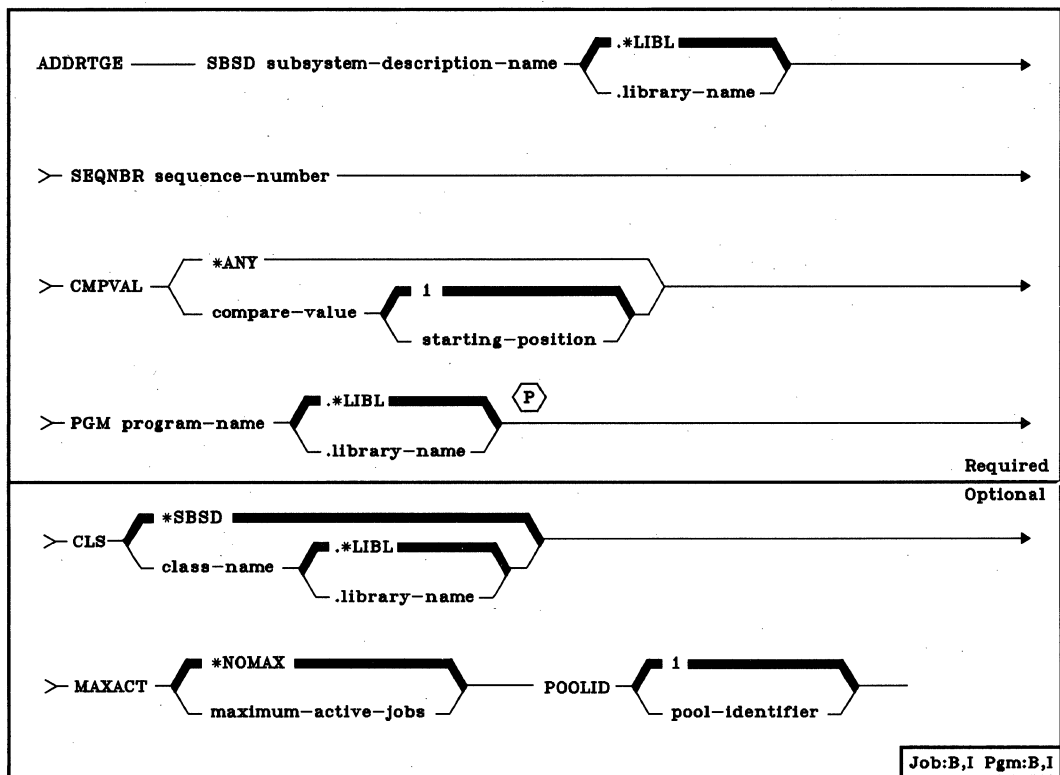
Messages associated with this writer will be written to the user message queue named COMPL in library USERLIB.

ADDRTGE

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.

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, 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.

ADDRTGE
CLS

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 SBSB 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 (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)
```

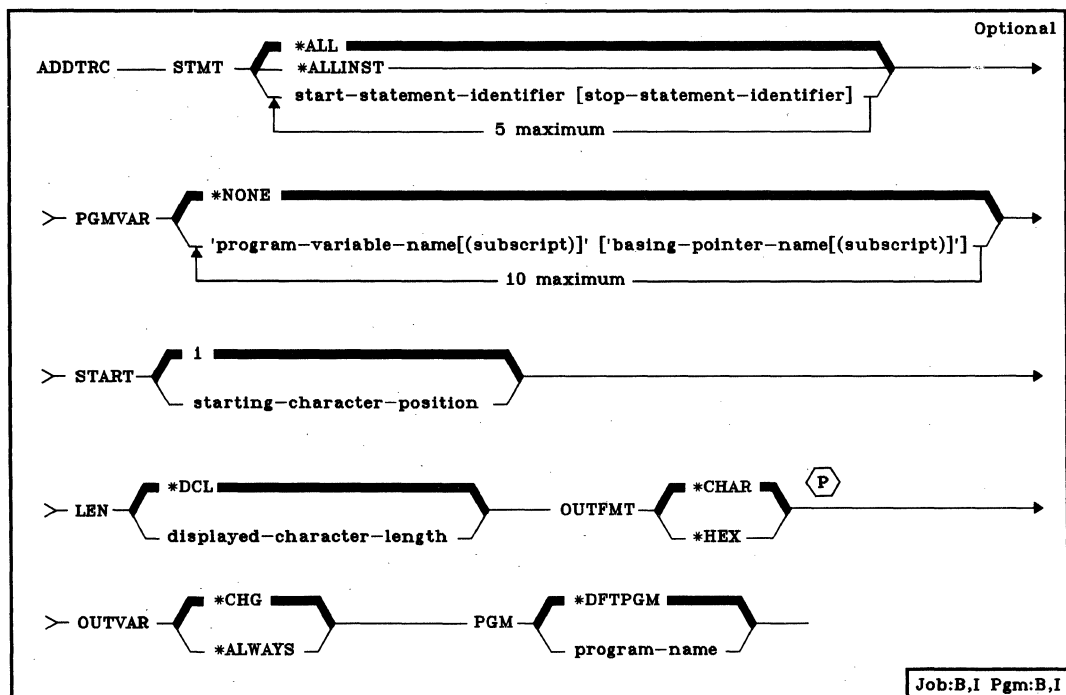
This command adds routing entry 5 to the subsystem description ABLE in the QGPL library. The program REORD in the general purpose library is initiated and uses the class MYCLASS in LIBX when a compare value of XYZ (beginning in position 1) is matched in the routing data. The program runs in storage pool 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.



ADDTRC
STMT

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.

ADDTRC
LEN

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.

***DCL:** 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.

***CHAR:** 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.

***CHG:** 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')
```

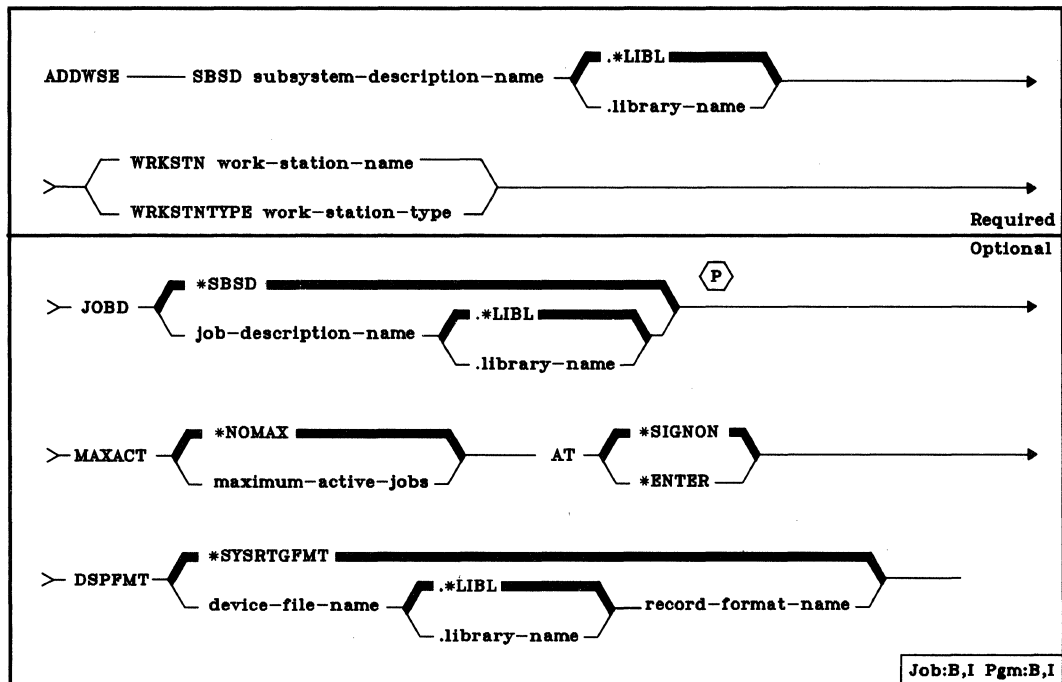
This command traces program statements in the default program between the ranges of statements 100 through 120 and 150 through 200. Also, whenever the values of any of the program variables &CTR, &BRCTR, and &SAM are changed by one of the traced statements within those ranges, the values of all three 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

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 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 device description name that was specified in the CRTDEVD command associated with the work station is the name to be used.

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

ADDWSE
WRKSTNTYPE

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
5251	5251 Display Station
5252	5252 Dual Display Station
5291	5291 Display Station
5292	5292 Color Display Station
*CONS	System console display

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

JOB 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 SBSDB 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.

ADDWSE
AT

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**
(Examples)

```
ADDWSE SBS(D.ORDER.LIB7) WRKSTNTYPE(5251) +  
JOB(D.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 SBS(D.ORDER.LIB7) WRKSTN(A12) +  
JOB(D.ORDENT.LIB7) AT(*SIGNON)
```

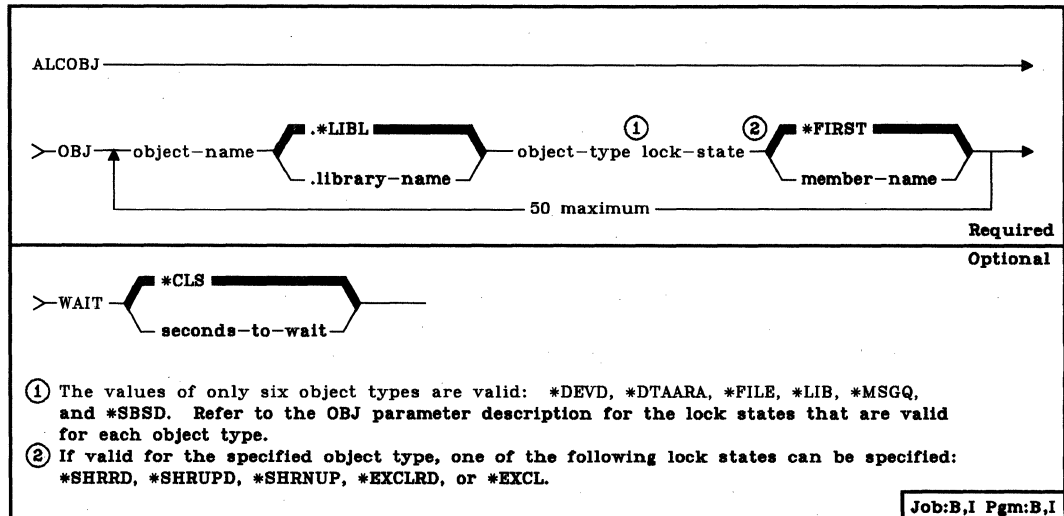
This command adds a work station job entry to a subsystem description named ORDER in library LIB7. Work station A12 is to be signed on at demand. When sign-on is complete, the system-supplied routing data format is displayed at the work station if the job description ORDENT in LIB7 specifies *GET as 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.



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 DEVDD 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.

**ALCOBJ
OBJ**

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).

Object Type	Lock States				
	*SHRRD	*SHRUPD	*SHRNUP	*EXCLRD	*EXCL
*DEV D				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

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 <i>line-description-name</i> _____
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 Journalled Changes) Command

The Apply Journalled Changes (APYJRNCHG) command applies the changes that have been journalled (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 journalled 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 journalled 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 journalled 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.

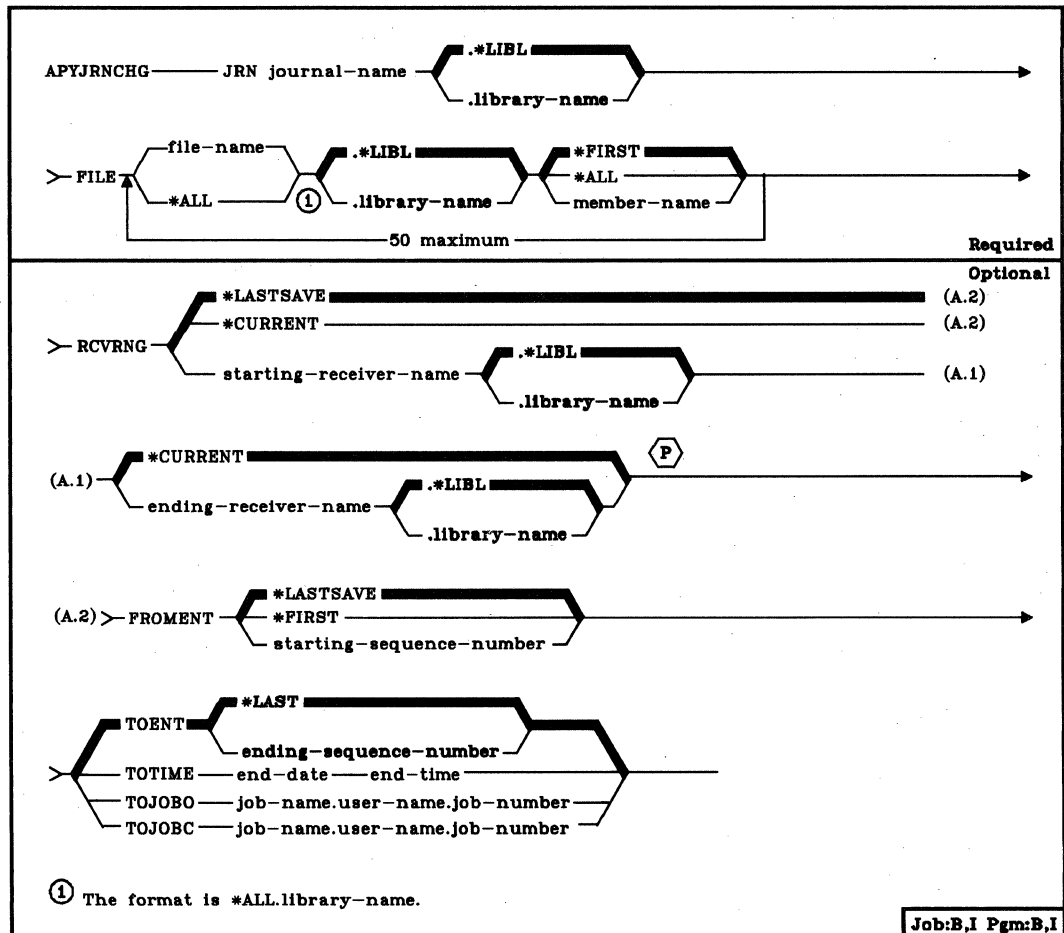
It is possible to apply changes even if the sequence numbers have been reset. The system will handle this condition, send an informational message, and continue to apply the changes. If journal receivers are attached and detached in pairs (dual receivers), the system will always attempt to use the first 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.

APYJRNCHG
(Diagram)

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').

APYJRNCHG
TOJOB0

TOJOB0 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.)

TOJOB0 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.

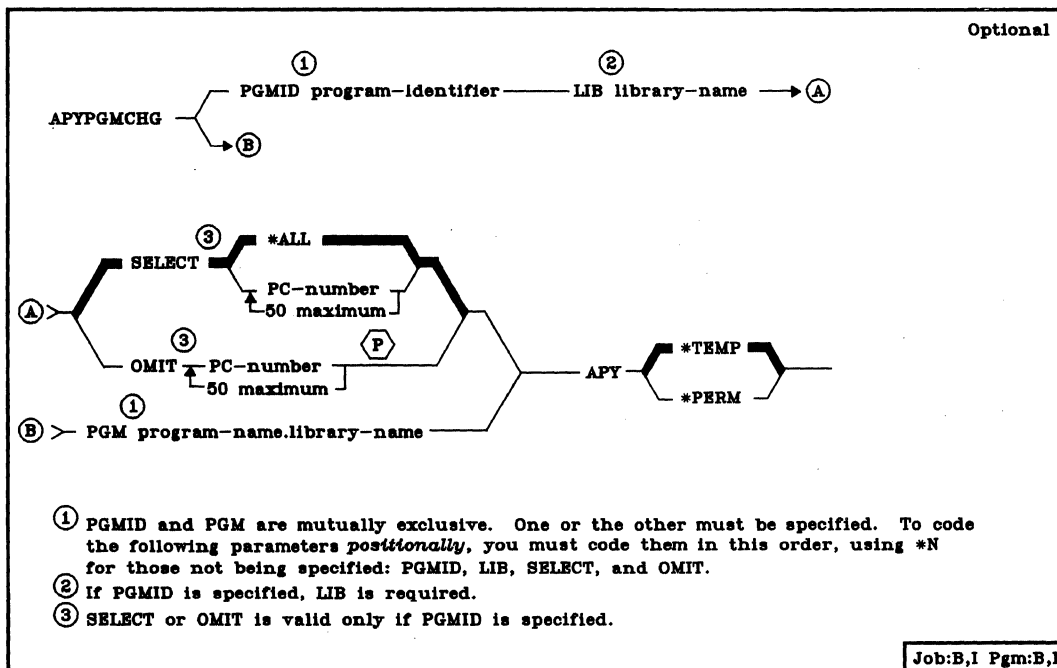
APYPGMCHG (Apply Programming Change) Command

APYPGMCHG

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*.



APYPGMCHG
PGMID

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 (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.

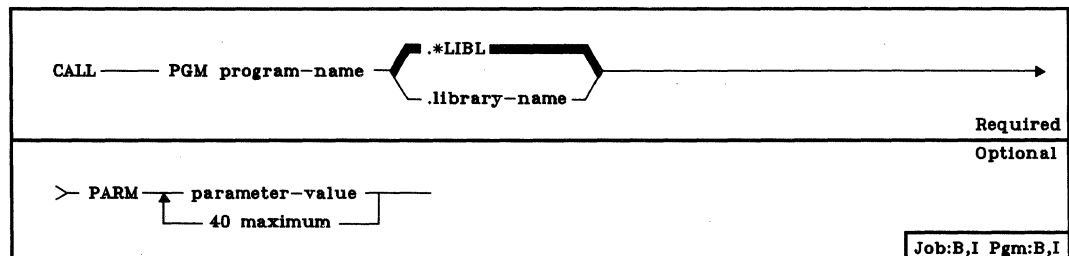
When the CALL command is issued by a CL program, each parameter value passed to the called program can be a character string constant, a numeric constant, a logical constant, or a CL program variable. When parameters are passed, the value of the constant or CL variable is available to the program that 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.

CALL
(Examples)

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.

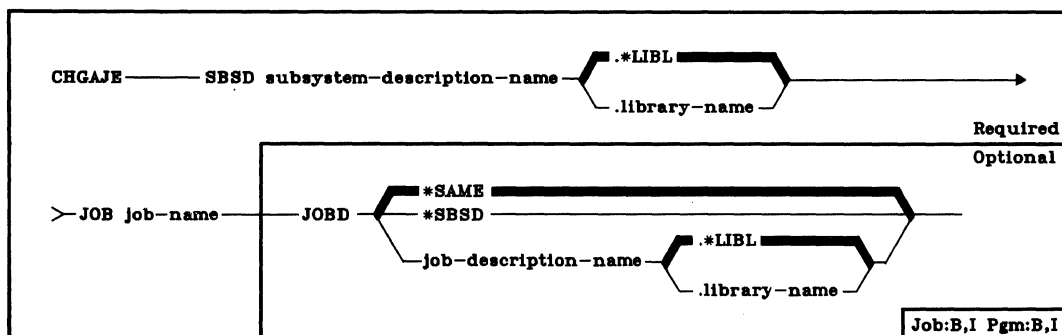
CALL PAYROLL.LIB1 (CHICAGO 1234 &VAR1)

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.

***SAME:** 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 SBS D 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.

CHGAJE
(Example)

Example

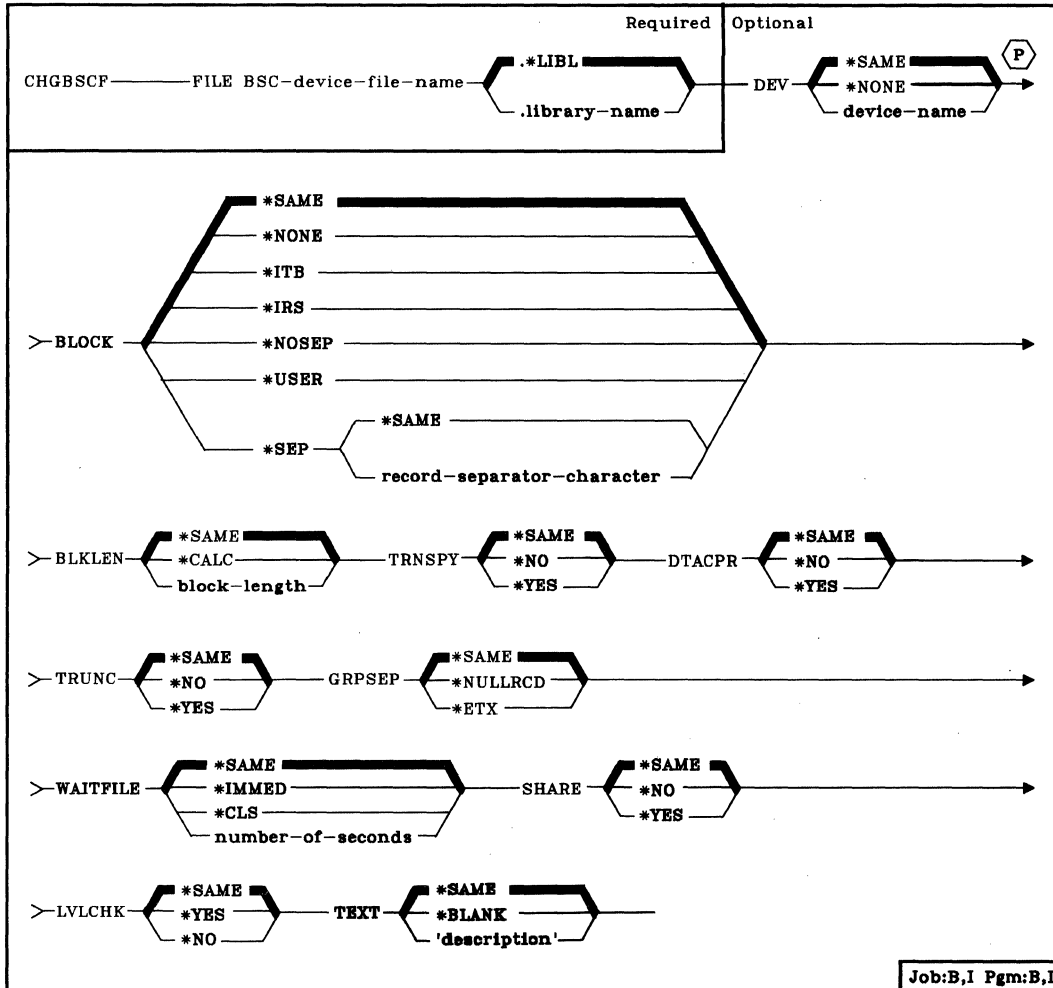
**CHGAJE SBS(D(PAYROLL.QGPL) JOB(INIT) +
JOB(D(MANAGER)**

This command changes the JOB(D 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

CHGBSCF

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.



CHGBSCF
FILE

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 unblock transmitted records. With this parameter, you may specify one of the following conditions of record formatting:

no blocking/unblocking: The record format described in the DDS (data description specifications) is the format for both the record and the block.

user blocking/unblocking: 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/unblocks 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 unblocked 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 unblocks the records, record separator characters and control characters will not be passed to your program as data.

***SAME:** 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 unblock 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.

CHGBSCF
BLKLEN

***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.

***SAME:** 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

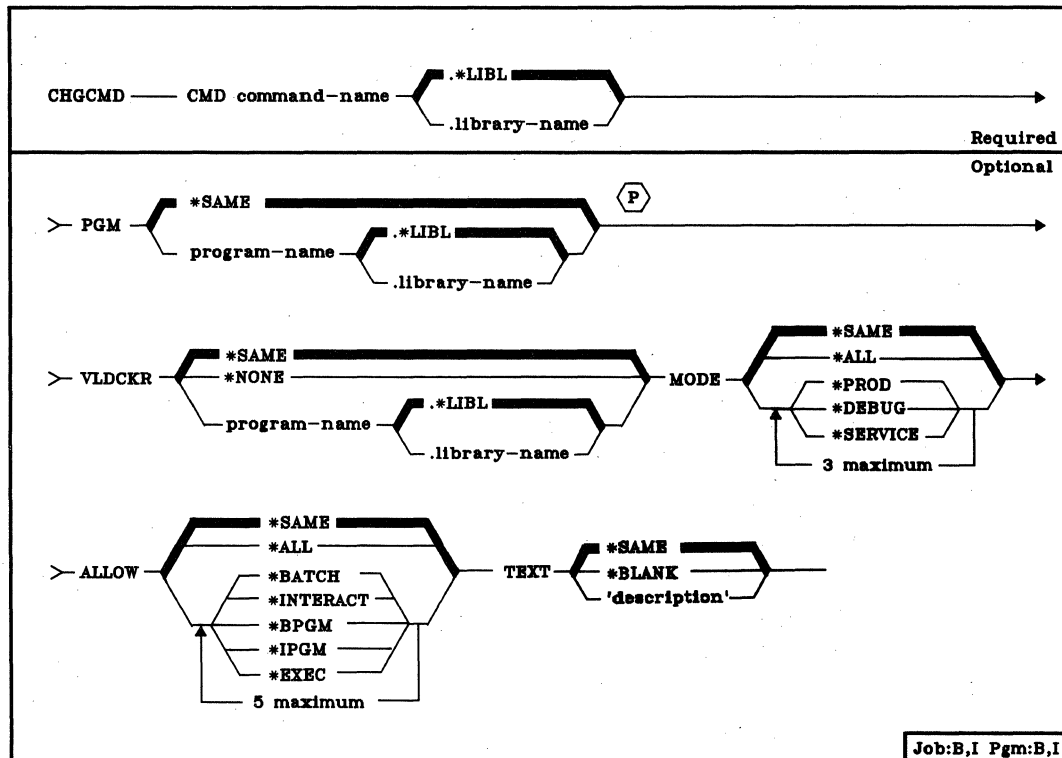
```
CHGBSCF FILE(TRANSD1.COMM1) BLOCK(SEP X'EE') WAITFILE(10)
```

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.)

VLCKR 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.

CHGCMD
ALLOW

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.

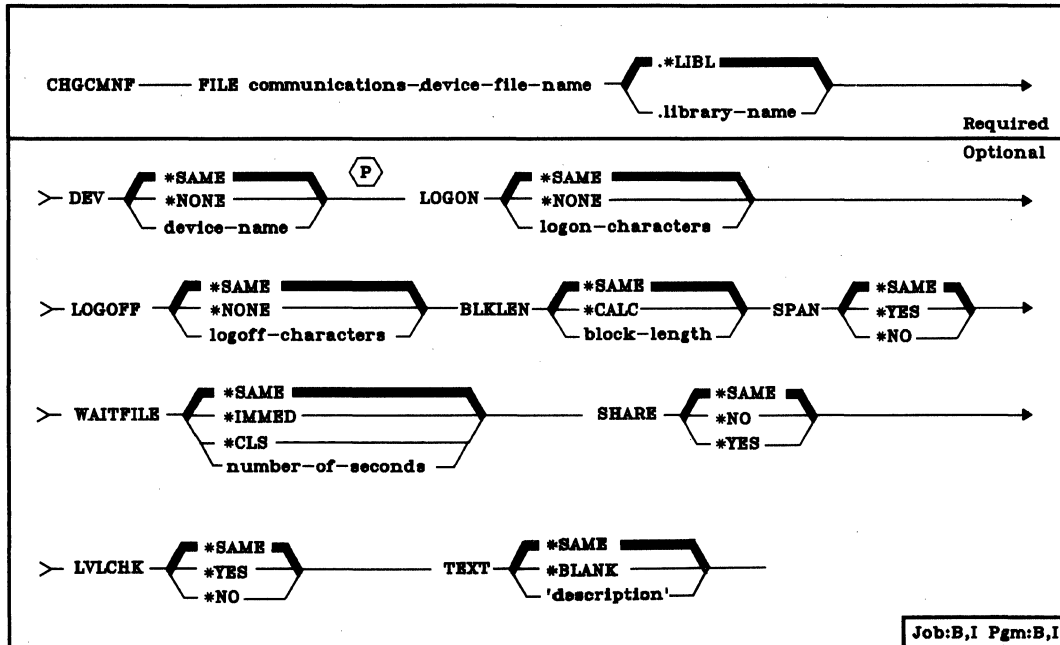
Examples

```
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.

CHGCMNF
LOGON

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.

***SAME:** 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.)

***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.

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).

CHGCMNF
SHARE

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.

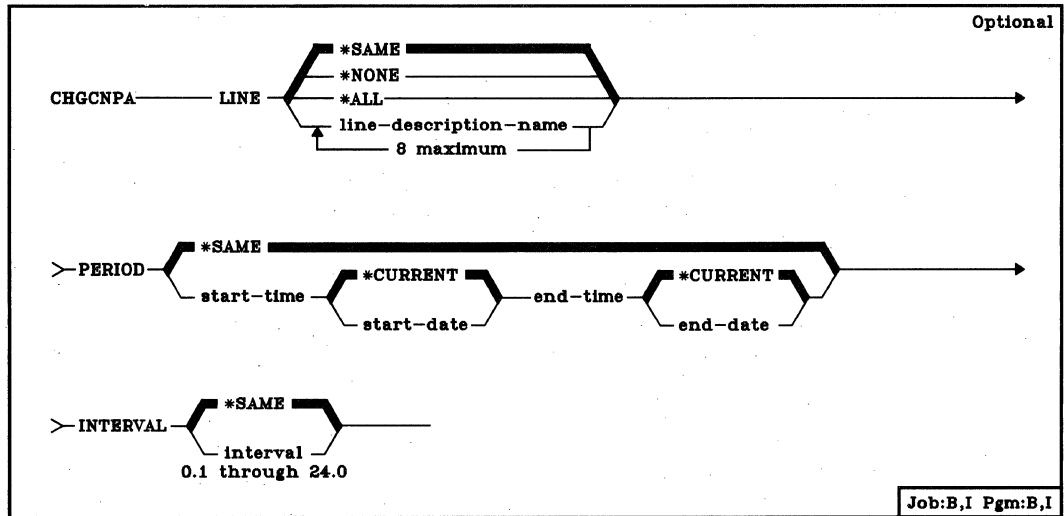
Example

```
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.

CHGCNPA
(Example)

Example

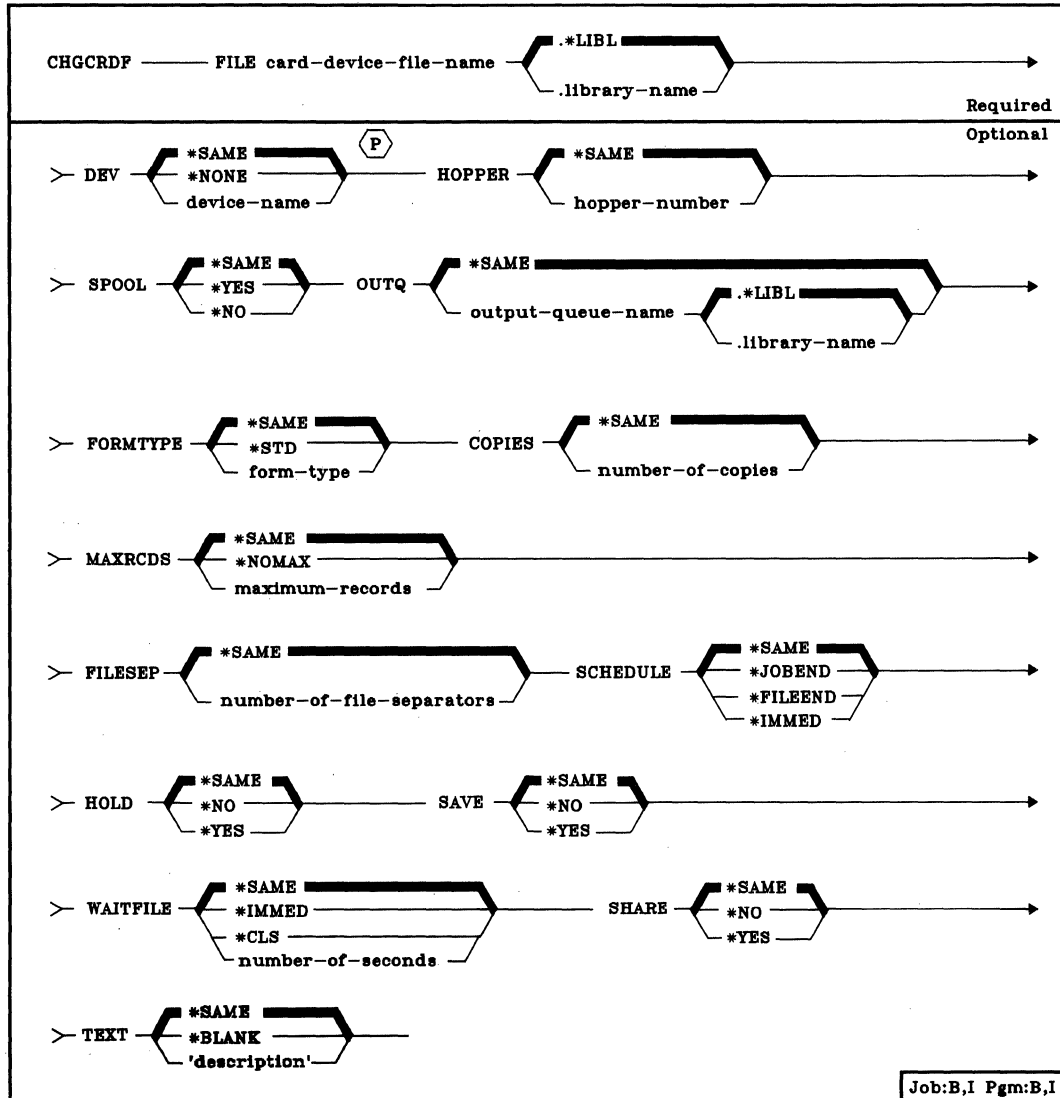
```
CHGCNPA LINE(LN1) PERIOD((133000 *CURRENT) (:153000 *CURRENT)) +  
INTERVAL(.3)
```

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

CHGCRDF

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.)

CHGCRDF
DEV

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.

***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 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.

***SAME:** 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.

***SAME:** 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.

***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 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.

**CHGCRDF
FILESEP**

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.

**CHGCRDF
TEXT**

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.

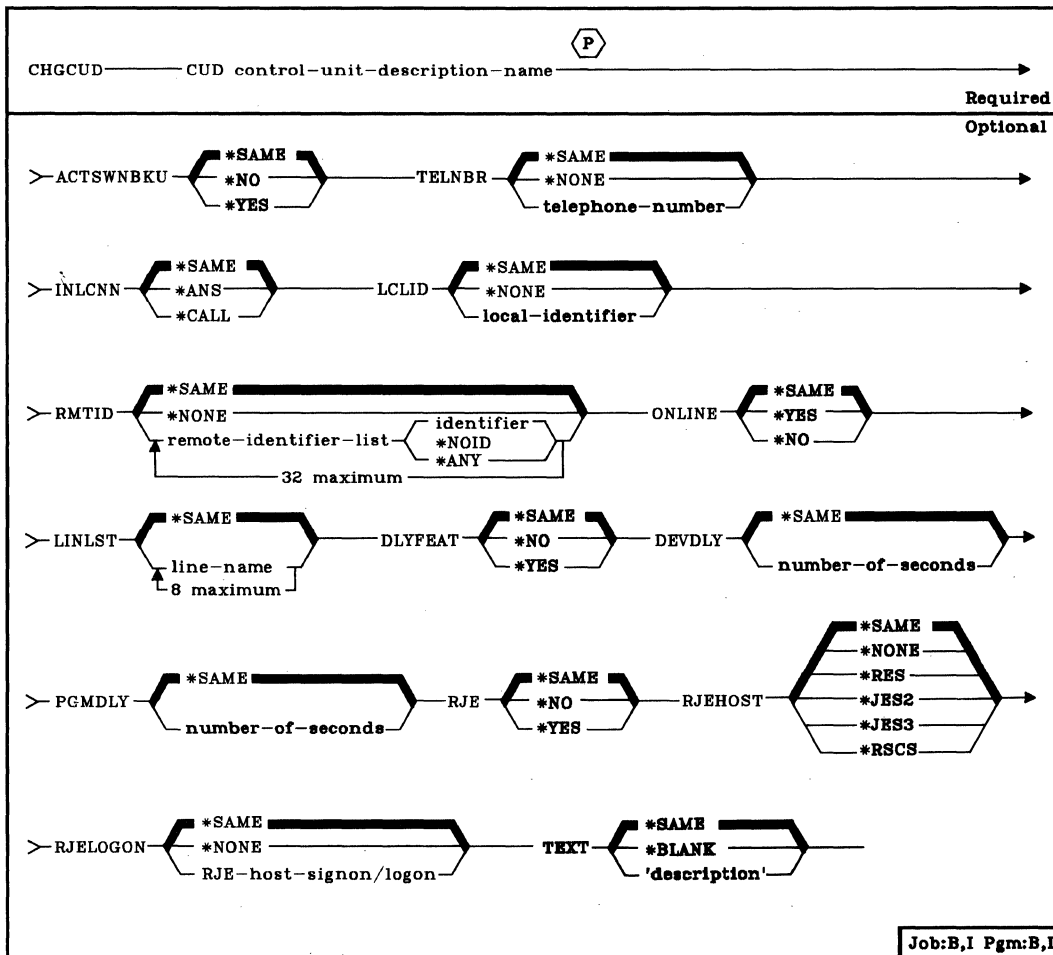
Examples

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

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 autocal 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.

CHGCUD
ONLINE

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 CRTUD 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.

DEVPLY 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.

**CHGCUD
RJELOGON**

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(nnnnnnnnn) +  
LINLST(LINE01)
```

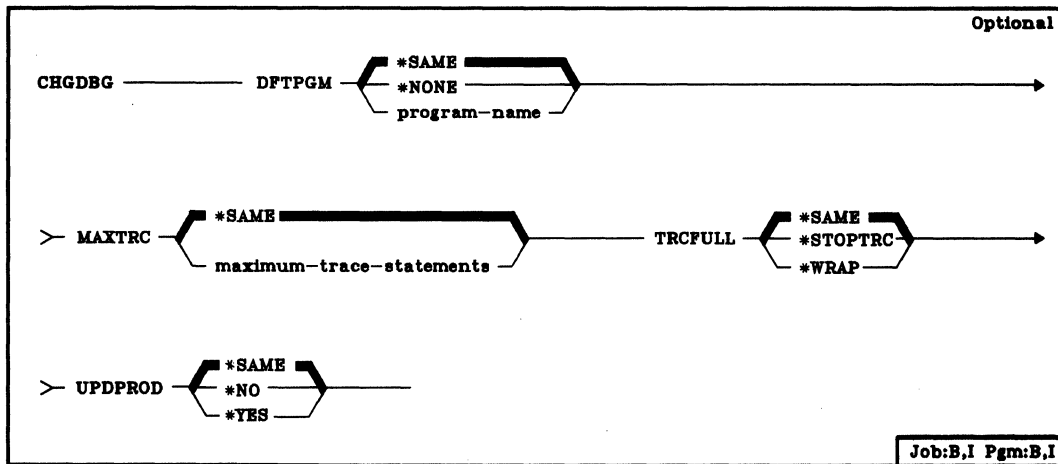
This command changes the control unit description of the control unit named CONTRL01. The line list now contains the line name LINE01, and the telephone number is changed to the number represented here by the letter *n* (nnn-*nnn*-nnnn). Because the line list is always changed from the beginning of the list, LINE01 replaced whatever line name was the first name in the list.

CHGDBG (Change Debug Mode) Command

CHGDBG

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.

**CHGDBG
MAXTRC**

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 (overlying) 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.

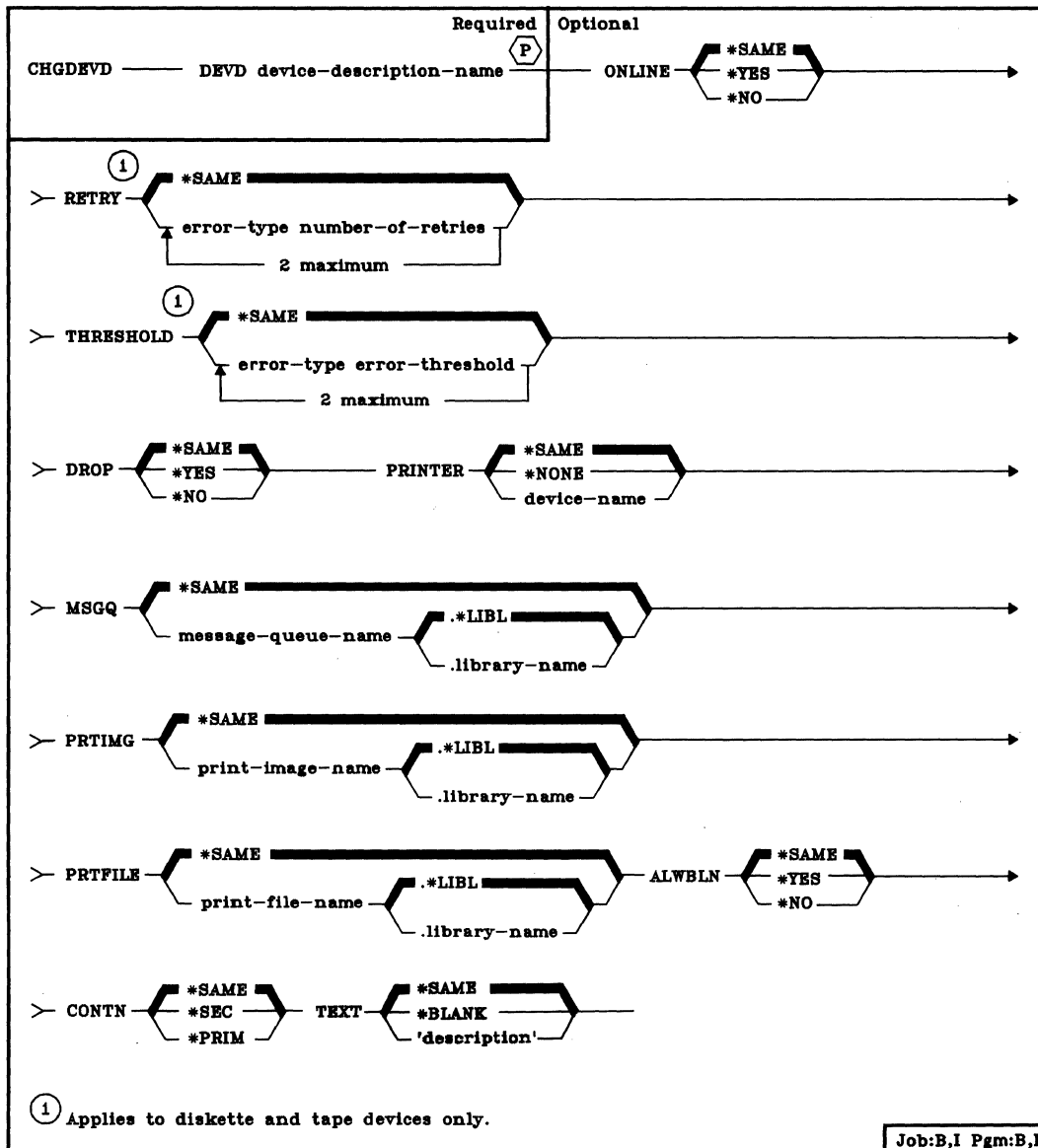
Example

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.



**CHGDEV
DEV**

DEV 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.

***SAME:** 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

***SAME:** 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.

**CHGDEVD
THRESHOLD**

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, for 5251, 5252, 5291, and 5292 display stations 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, 5252, 5291, or 5292 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.

CHGDEV
MSGQ

***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 5251, 5252, 5291, or 5292 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

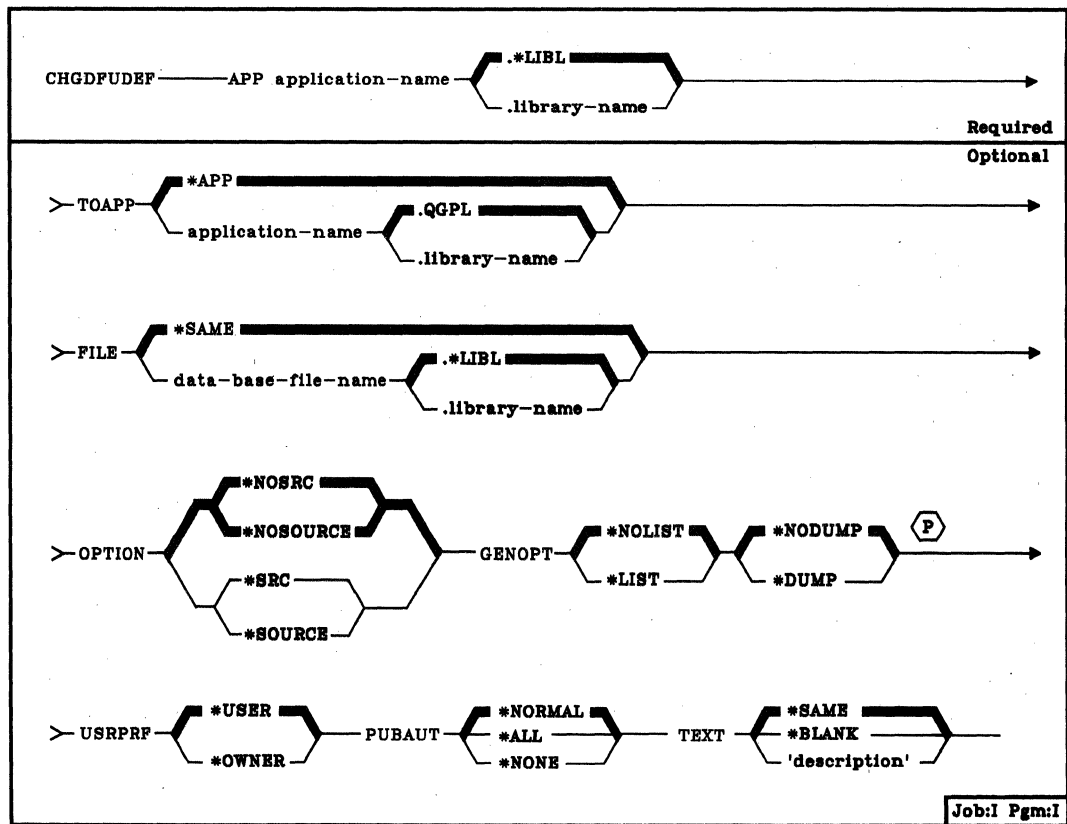
```
CHGDEVD DEVD(DISP01) PRINTER(PRINTMASK1)
```

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.

***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*). The file contains record formats that will be referred to by the application you are changing.

***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, *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 will provide the template only if *LIST has been specified.

CHGDFUDEF
USRPRF

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.

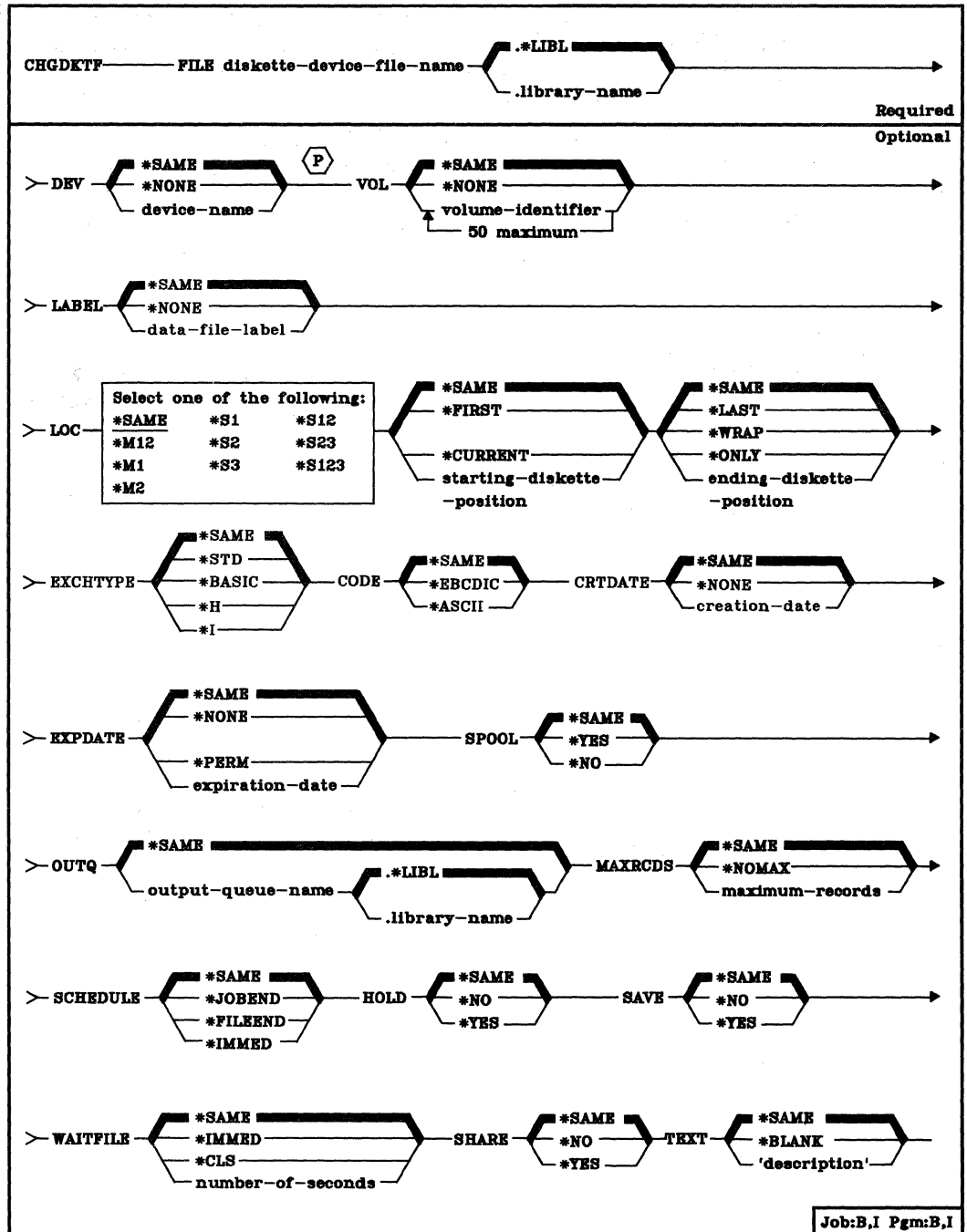
Example**CHGDFUDEF**
(Example)

```
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.

**CHGDKTF
LABEL**

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.

CHGDKTF
TEXT

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.

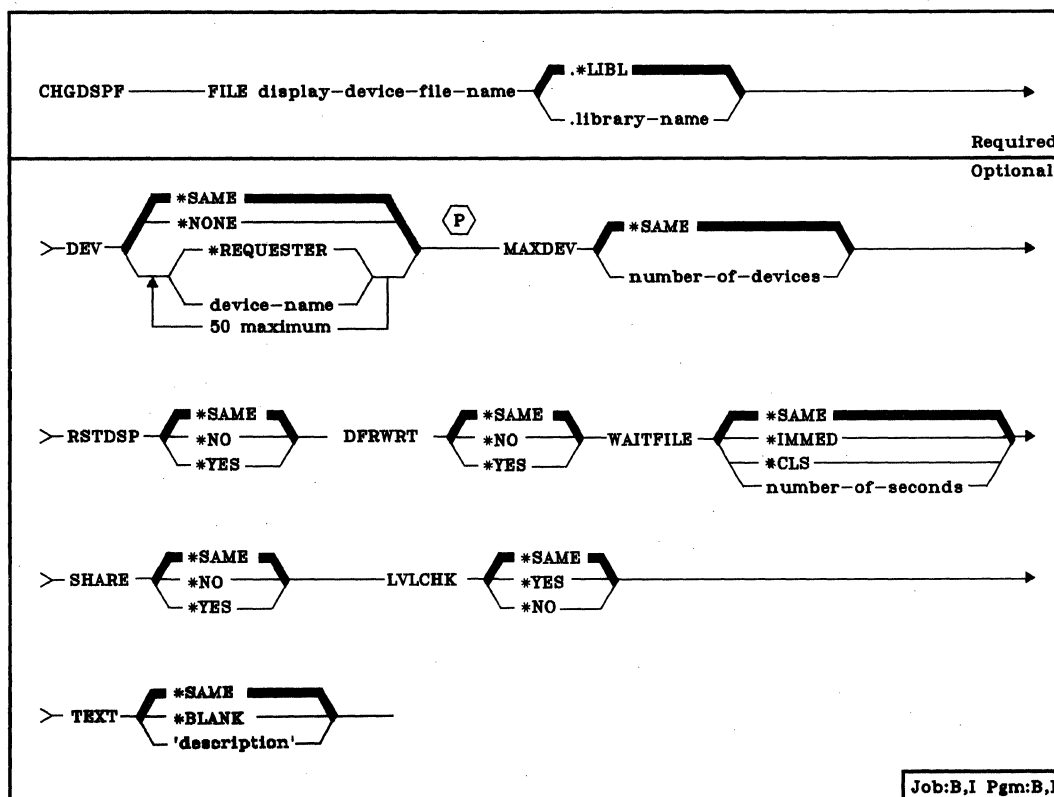
Example

```
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.

CHGDSPF
MAXDEV

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.

***SAME:** 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*.)

***SAME:** 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.

***SAME:** 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.)

***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 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.

***SAME:** The value specified in the device file description is not to be changed.

CHGDSPF
LVLCHK

***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)
```

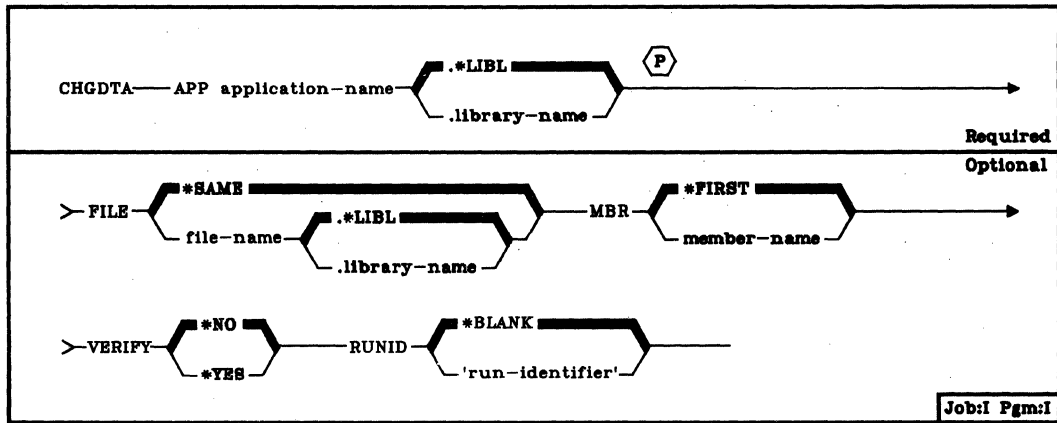
This command changes the description of the display device file named ORDENT. The file is located through the library list. The devices to be used with this file are the work stations WS1, WS2, and WS3. All three of the devices can be used concurrently with this display file.

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.

CHGDTA
VERIFY

VERIFY Parameter: Indicates whether the updates are intended to verify the contents of existing data records.

***NO:** 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.

***BLANK:** 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.

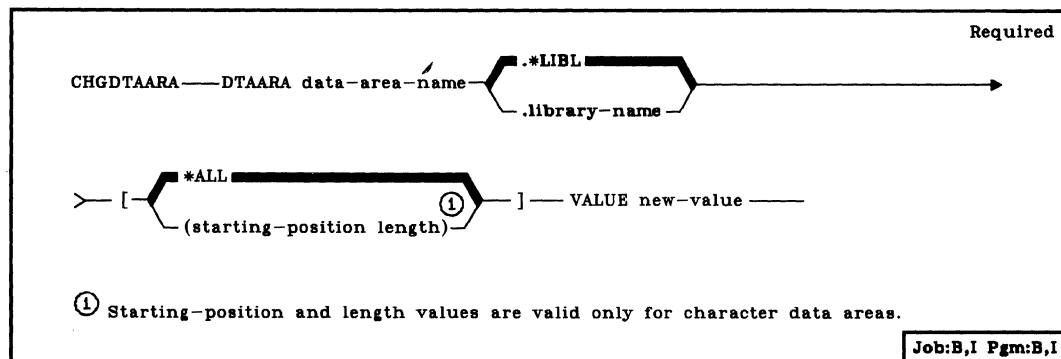
CHGDTAARA (Change Data Area) Command

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.



CHGDTAARA
DTAARA

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.

***ALL:** 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')
```

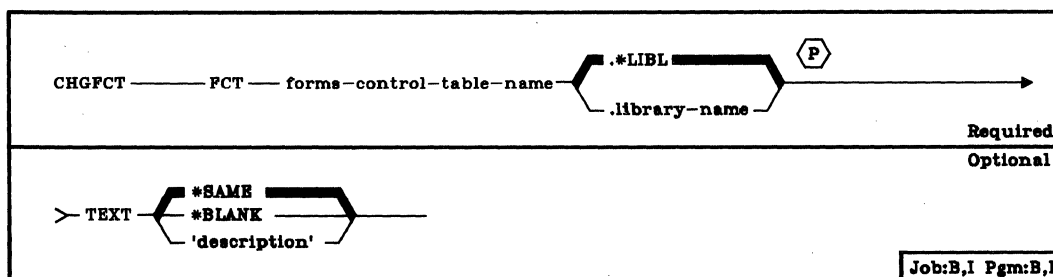
This command changes characters 5 through 8 of the data area MYDATA. Because the new value is shorter than the substring, it will be padded with a blank. If MYDATA is a character data area that previously contained 'ONE TOOTHREE', MYDATA will now contain 'ONE TWO THREE'.

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

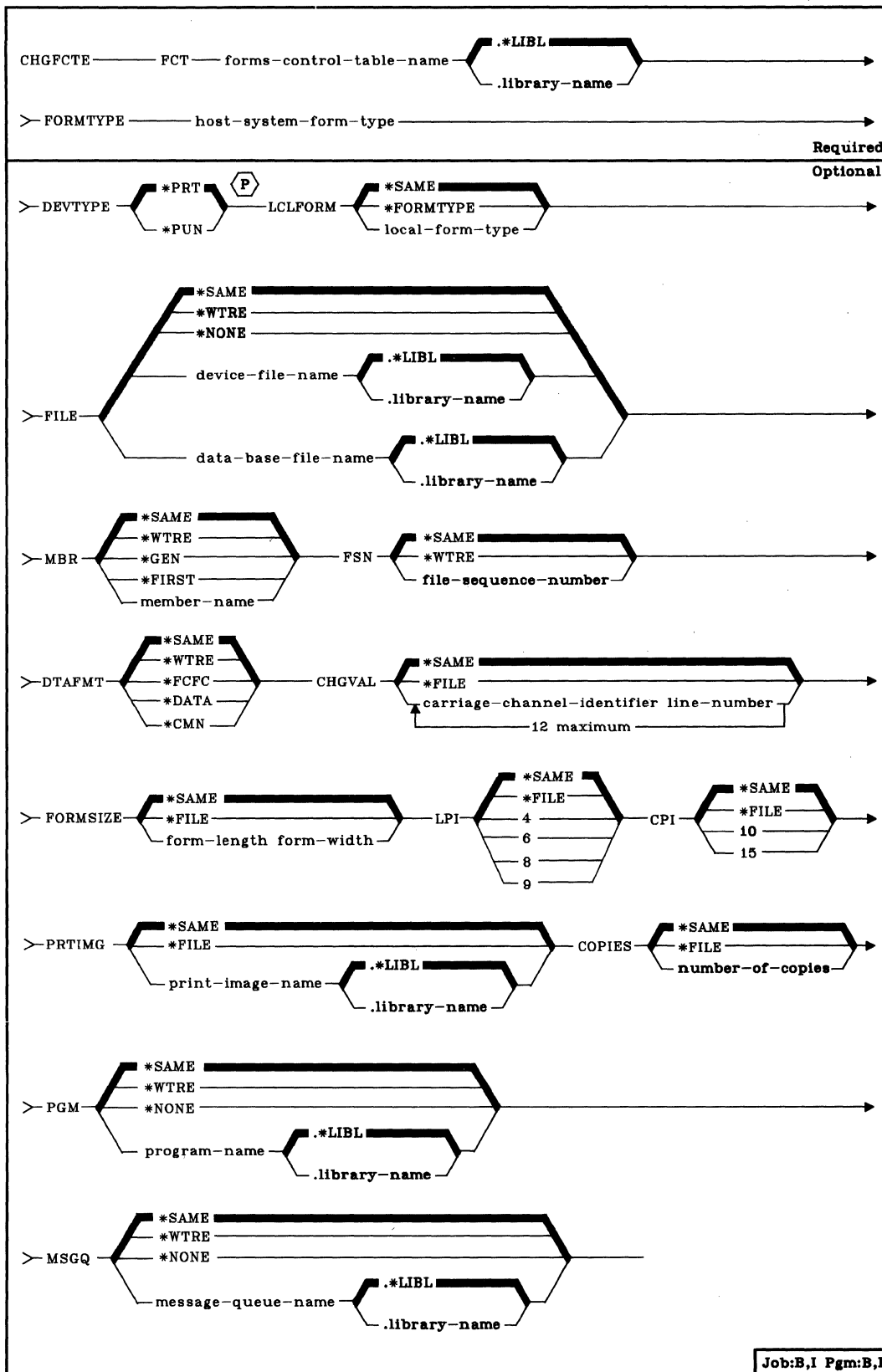
The Change Forms Control Table Entry (CHGFCTE) command changes the attributes in an existing forms control table (FCT) entry.

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
(Diagram)



Job:B,I Pgm:B,I

CHGFCTE
FCT

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).

***SAME**: 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:

Affffffcc or Bffffffcc

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).

CHGFCTE
FSN

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 RJE 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.

***SAME:** 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.

***SAME:** 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.

***SAME:** 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.

CHGFCTE
CPI

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

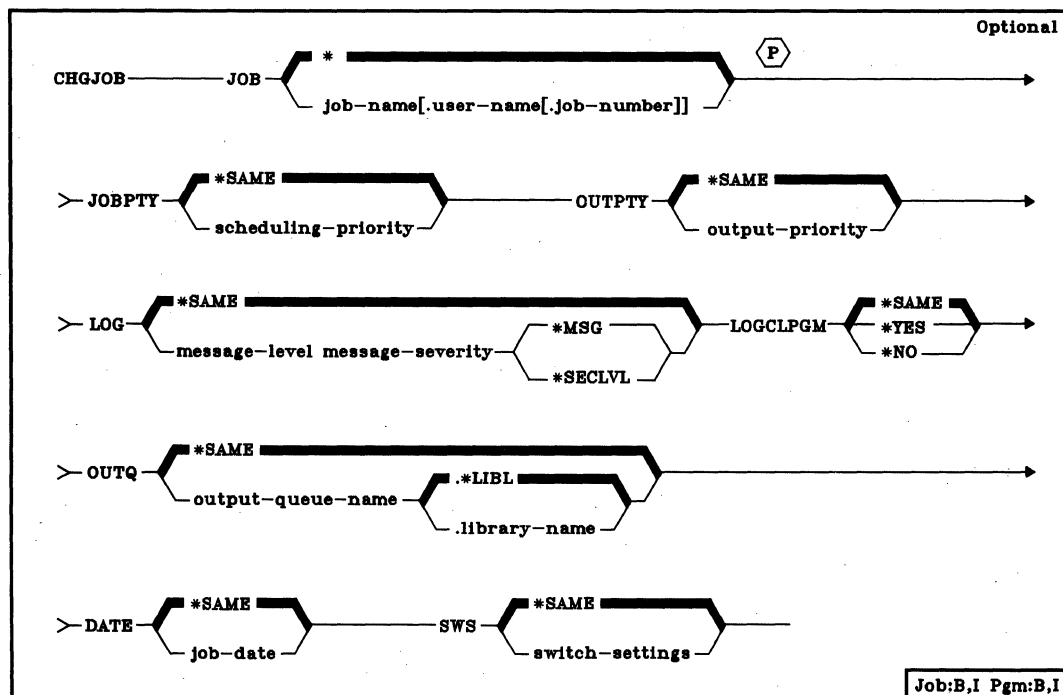
```
CHGFCTE FCT(FORMCTRL.USERLIB) +
        FORMTYPE(MEDICAL) +
        DEVTYPE(*PUN) +
        FSN(200)
```

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 CRTJOB 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.

CHGJOB
LOGCLPGM

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 CRTJOB 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.

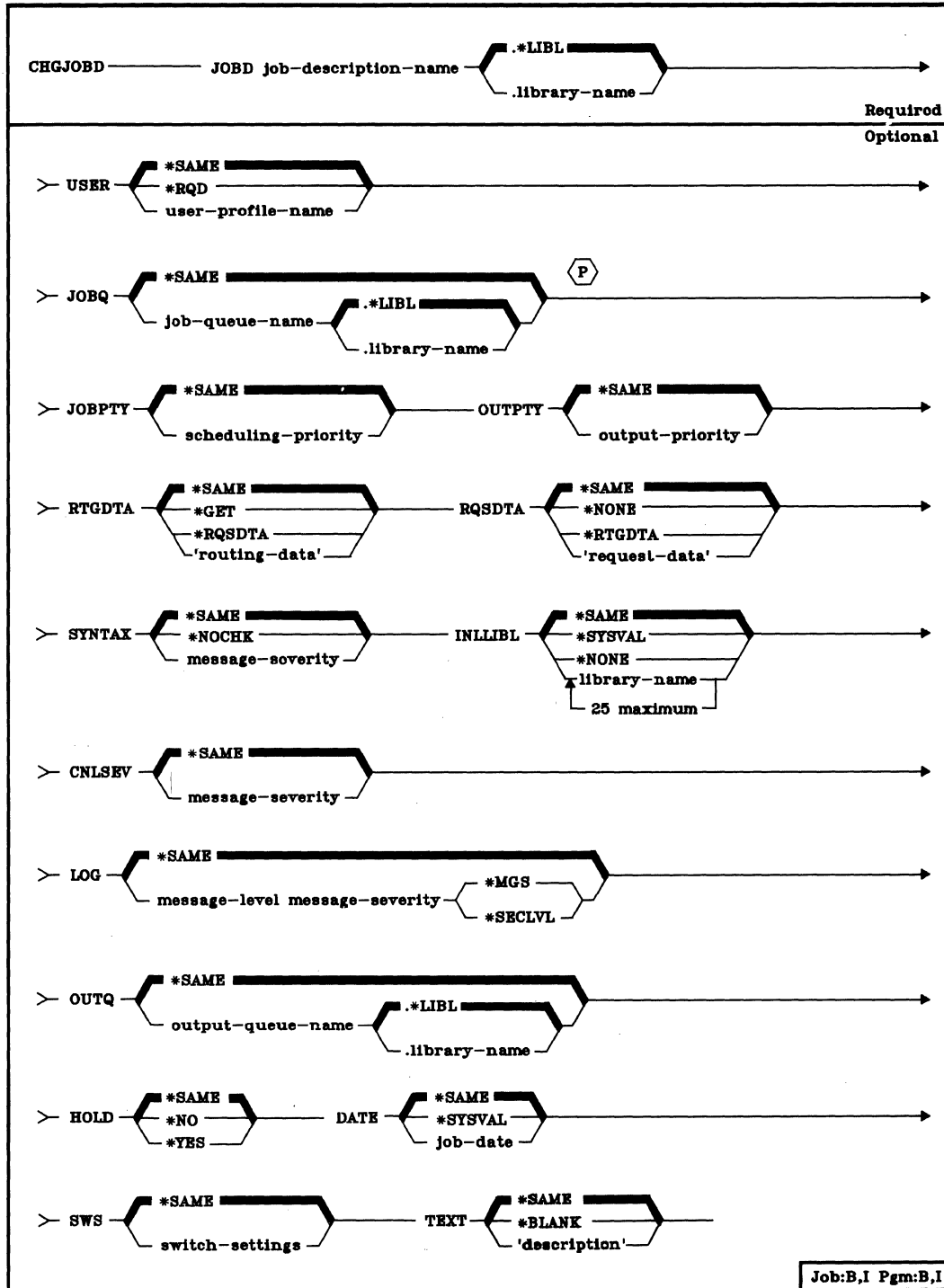
CHGJOB (Change Job Description) Command

The Change Job Description (CHGJOB) command changes the job-related attributes specified for a job description object through the Create Job Description (CRTJOB) 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.

CHGJOB
(Diagram)



Job:B,I Pgm:B,I

CHGJOB
JOB

JOB 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.

*SAME: 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.

*SAME: 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.)

*SAME: 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.)

***SAME:** 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.

***SAME:** 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).

***SAME:** 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.

CHGJOB
SYNTAX

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 CRTJOB 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.

CHGJOB
HOLD

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.

***SAME:** 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.

***SAME:** 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).

***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 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.)

***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

CHGJOB (Examples)

```
CHGJOB JOB(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:

```
CHGJOB JOB(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.

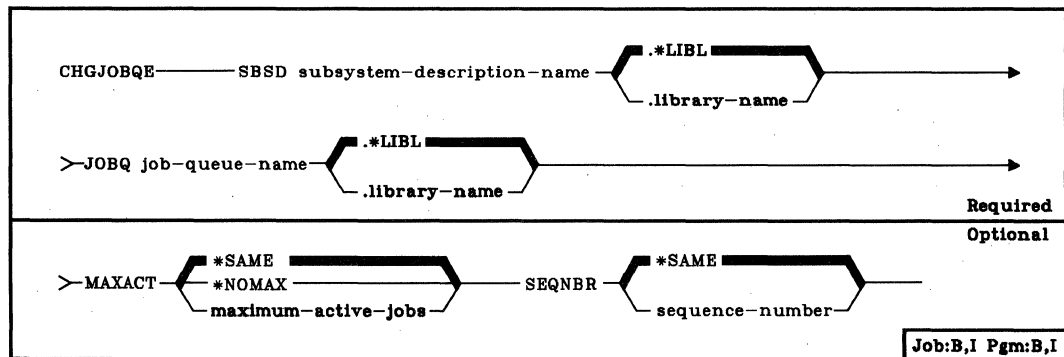
CHGJOBQE (Change Job Queue Entry) Command

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

CHGJOBQE
(Example)

Example

```
CHGJOBQE SBSDB(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

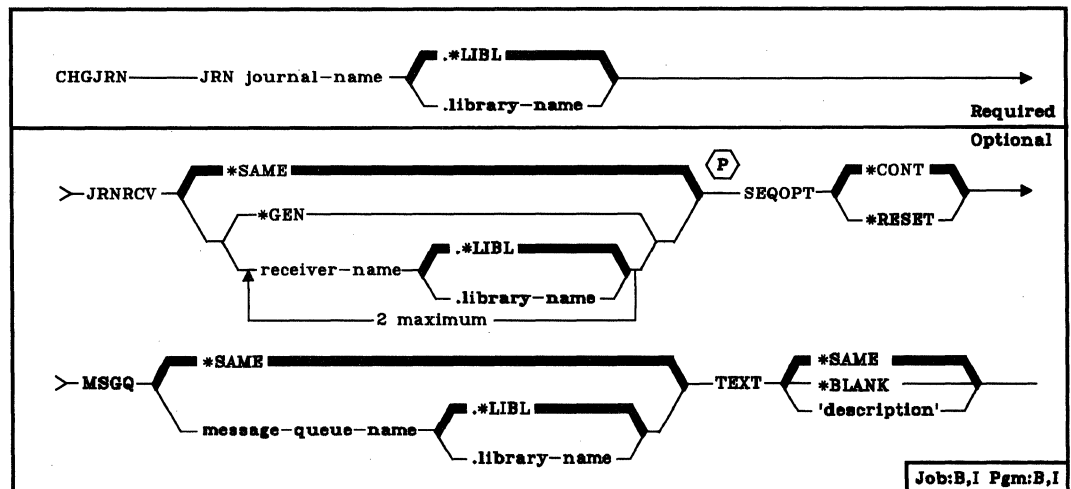
The Change Journal (CHGJRN) command changes the journal receivers, the message queue, or the descriptive text associated with the indicated journal. The command allows up to two journal receivers to be attached to the specified journal. These replace all previously attached journal receivers. The designated 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.

**CHGJRN
MSGQ**

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.

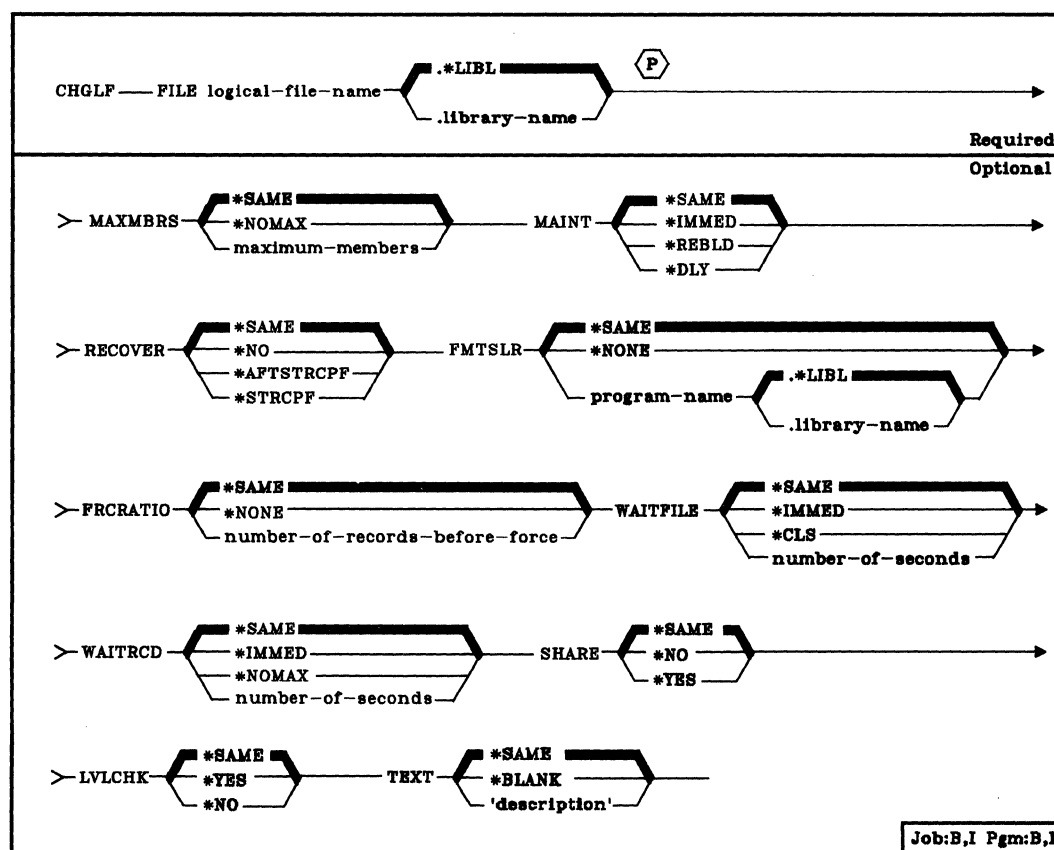
CHGJRN JRN(JRNLA) JRNRCV(*GEN *GEN)

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.



**CHGLF
FILE**

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

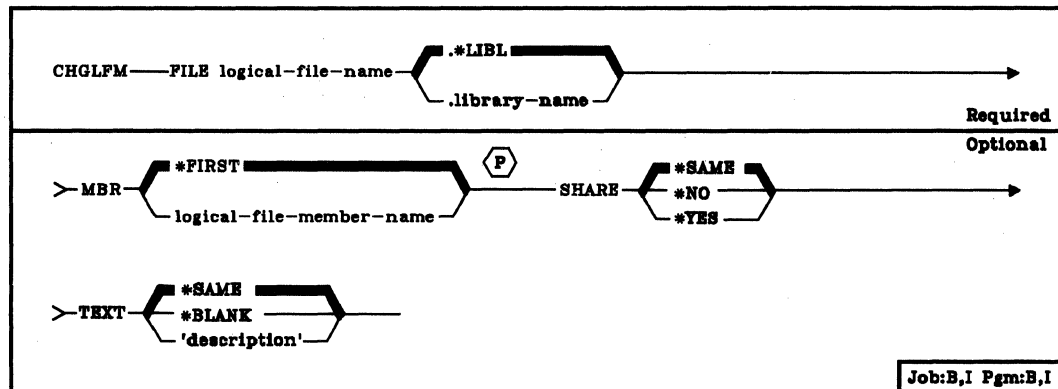
```
CHGLF FILE(INV.QGPL) MBR(FEB) FMTSLR(INVFMTS)
```

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.

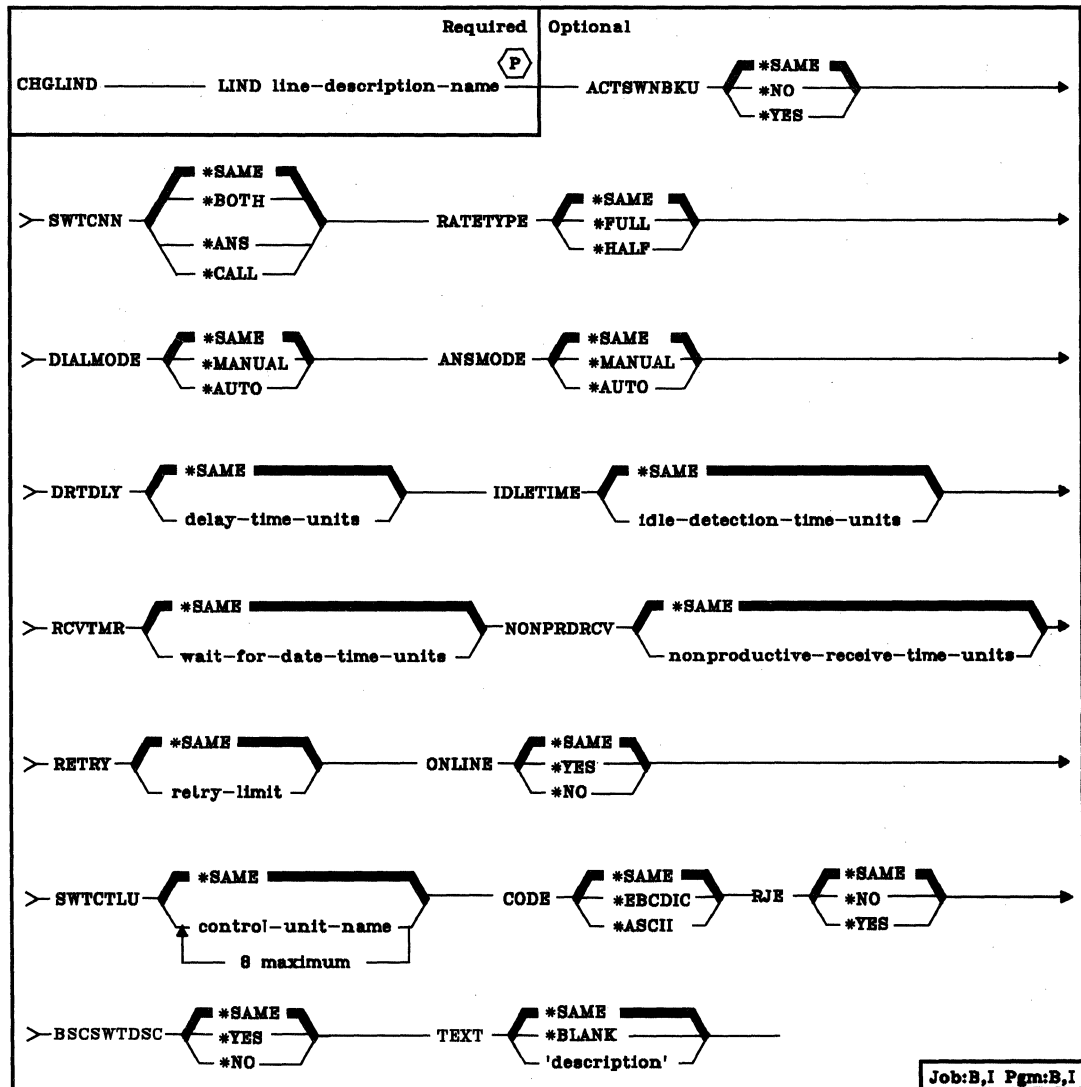
Example

```
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.



Job:B,1 Pgm:B,1

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.

CHGLIND
DIALMODE

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.

CHGLIND
RETRY

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.

CHGLIND
(Example)

Example

```
CHGLIND LIND(LINE01) RETRY(15) ONLINE(*NO)
```

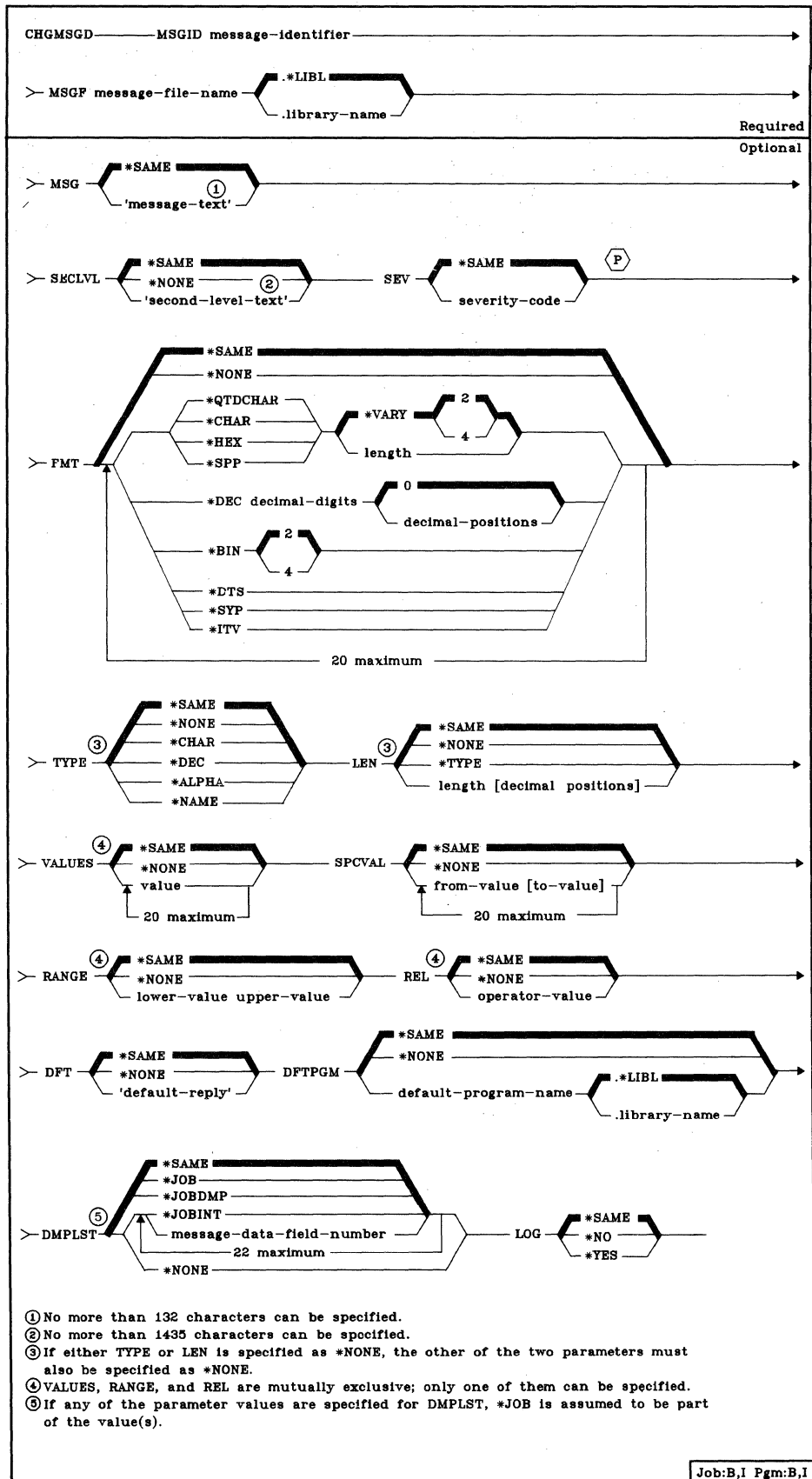
This command changes the line description of the line named LINE01. The RETRY parameter is changed to a maximum of 15 retries for errors occurring while a switched connection is being attempted and a maximum of 21 retries for all other errors associated with the line. The ONLINE parameter is changed 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.

CHGMSGD
(Diagram)



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.

CHGMSGD
SECLVL

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 alphanumeric 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'COF4').

***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).

**CHGMSGD
FMT**

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

CHGMSGD
TYPE

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.

CHGMSGD
LEN

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.

SPCVL 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.

CHGMSGD
DFT

***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 *JOB DMP) results in the equivalent of DMPLST(*JOB 1 2 *JOB DMP).

*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.

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

**CHGMSGD
LOG**

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.

CHGMSGD
(Examples)

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)
```

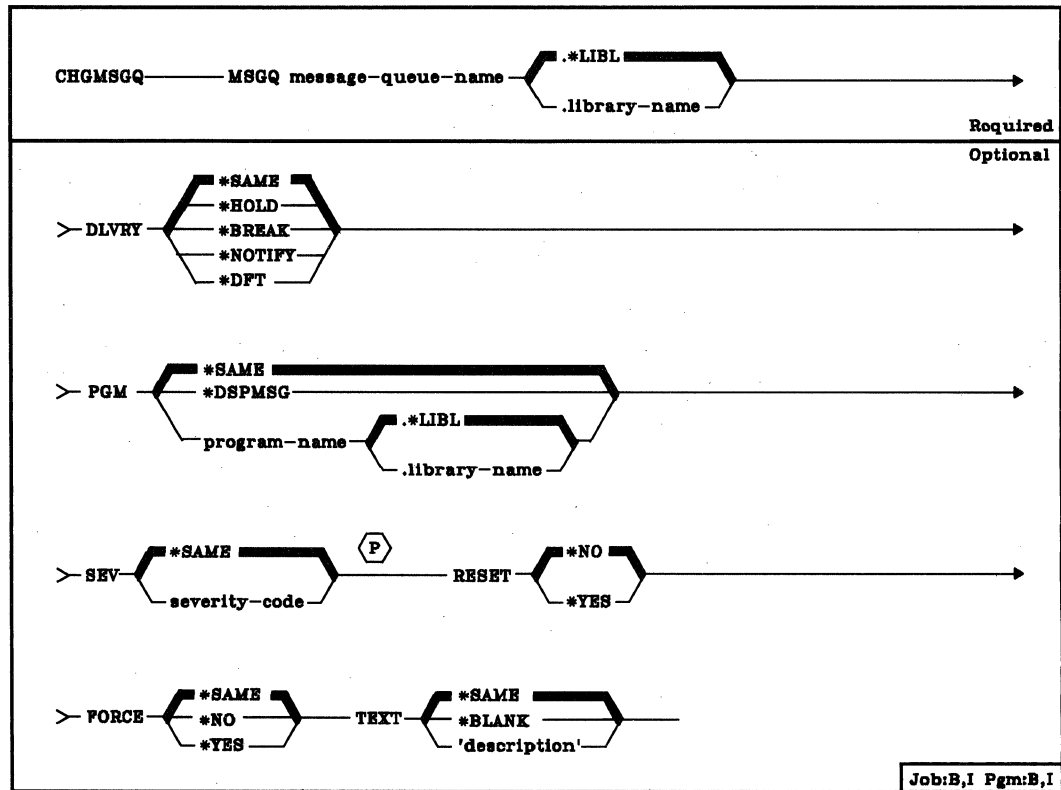
To make the change to the range of reply values valid, you must also change the length (LEN parameter). The correct command coding would be as follows:

```
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.

CHGMSGQ
PGM

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.

CHGMSGQ
(Examples)

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.

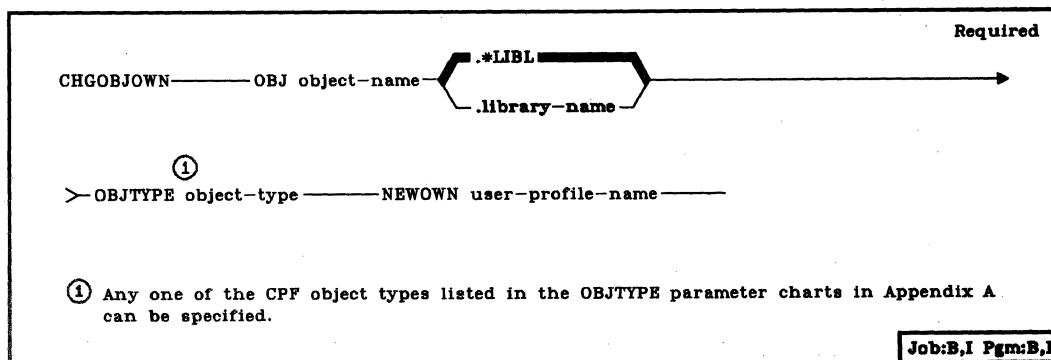
The owner of an object always has all the rights applicable to the object unless they are explicitly revoked. As the owner, he has the authority to grant any rights to any user for his objects. He can also grant to himself any rights that were explicitly revoked previously. The owner may, for example, 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.



CHGOBJOWN
OBJ

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

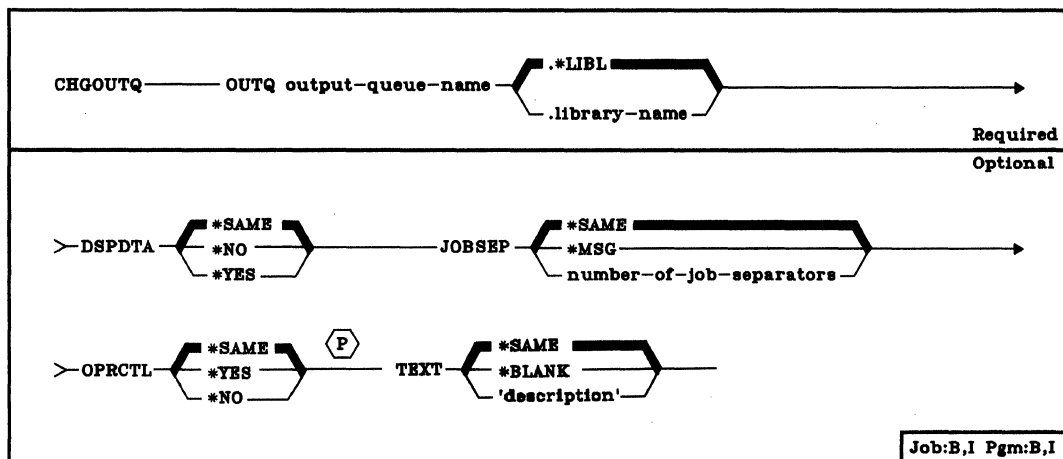
```
CHGOBJOWN OBJ(PROGRAM1.USERLIB) OBJTYPE(*PGM) +  
NEWOWN(ANN)
```

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.

**CHGOUTQ
JOBSEP**

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

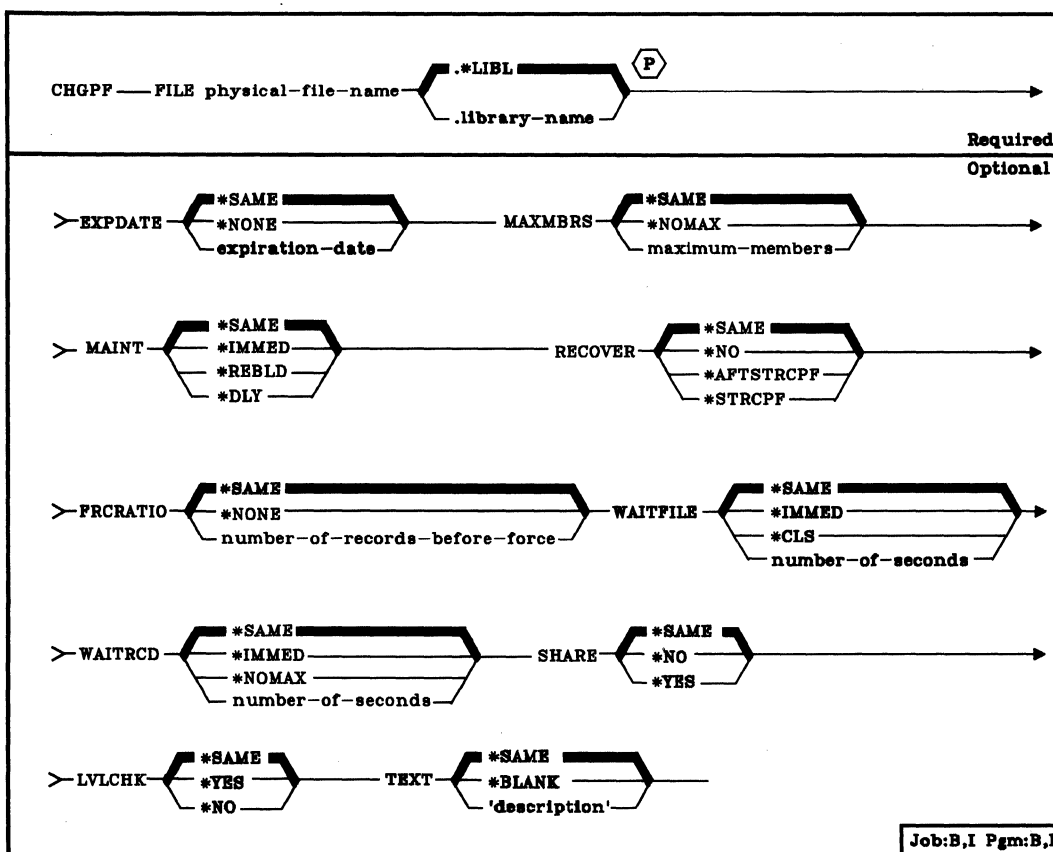
```
CHGOUTQ OUTQ(QPUNCH) JOBSEP(4) TEXT('Default queue +  
for all 96-col cards')
```

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.)

EXPDTE 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 EXPDTE 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 EXPDTE 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).

***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.

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.

CHGPF
LVLCHK

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

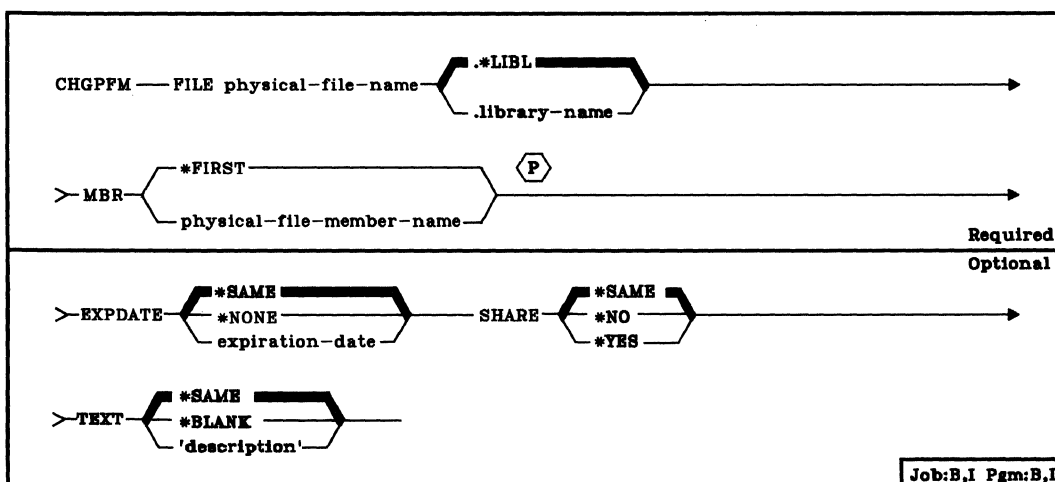
```
CHGPF FILE(INV.QGPL) EXPDATE('10/31/87')
```

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.

***FIRST:** 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.

EXPDTE 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.

***SAME:** 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.

***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 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')
```

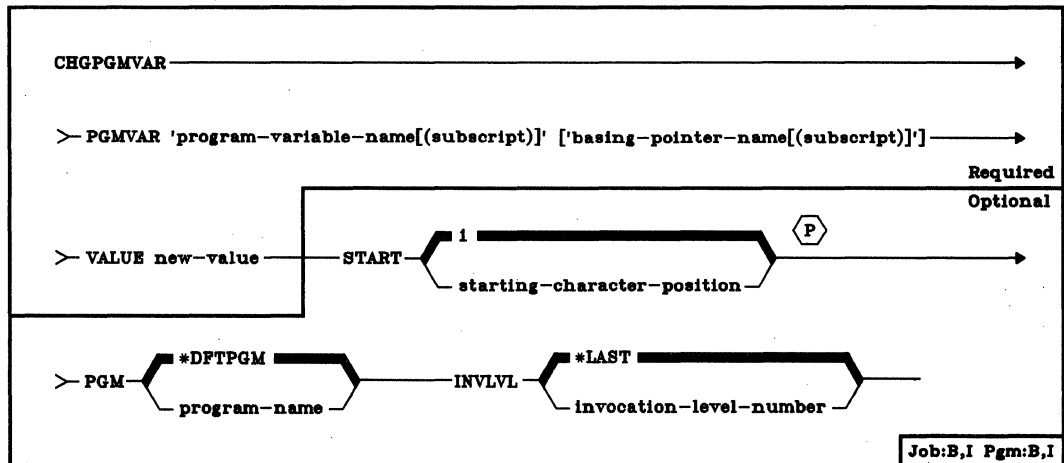
The member named FEB in the physical file INV that is stored in the QGPL library is to be changed so that the expiration date of the member will now be October 31, 1987.

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.

CHGPGMVAR
VALUE

['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.

CHGPGMVAR
(Examples)

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)
```

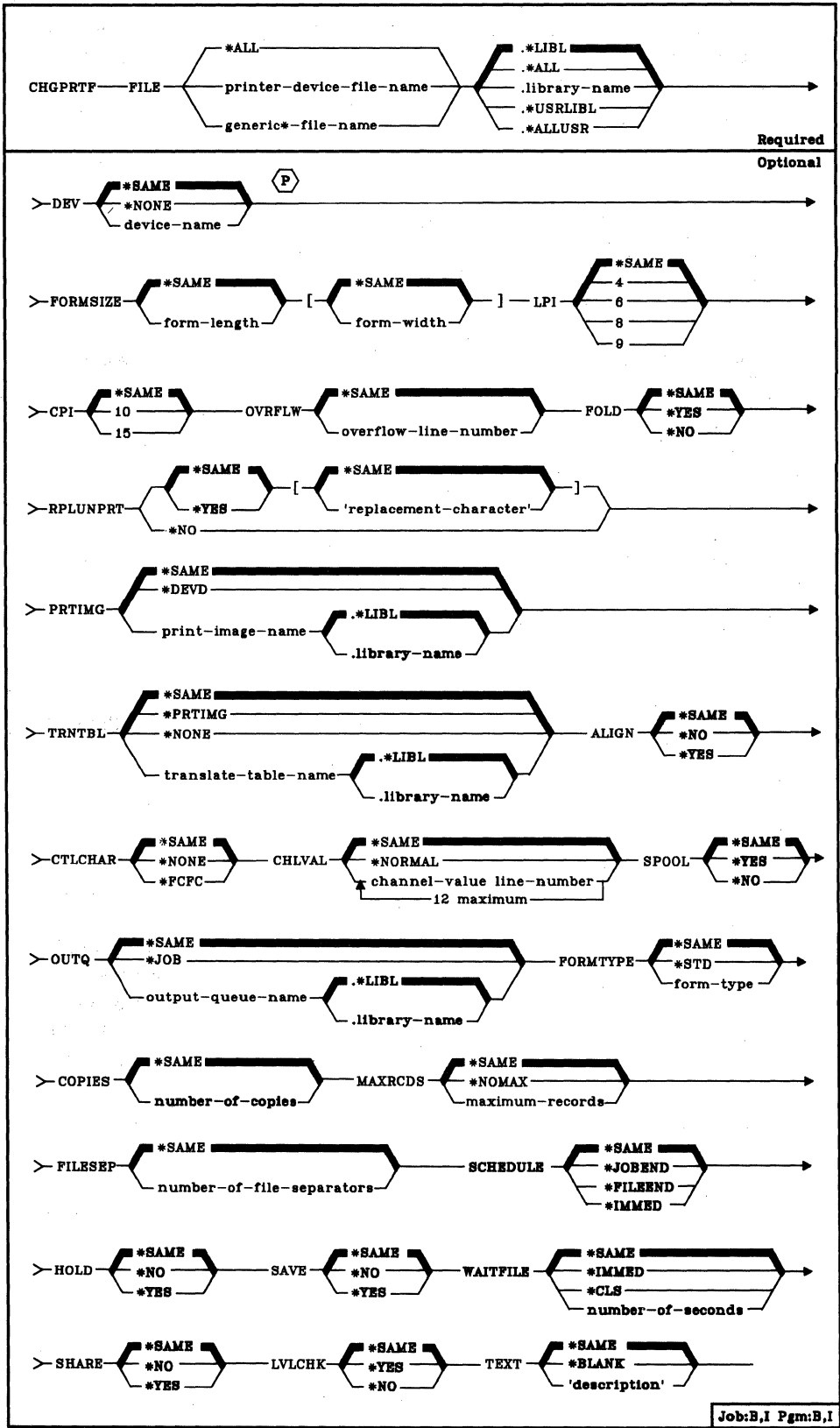
This command changes, starting in position 4, the program variable PARTNO to 56. Because the START parameter is specified, PARTNO must be a character variable. Because PARTNO is a character variable, the numeric value must be enclosed in apostrophes.

CHGPRTF (Change Printer File) Command

CHGPRTF

The Change Printer File (CHGPRTF) command changes, in the file description, one or more of the attributes of the specified printer device file.

CHGPRTF
(Diagram)



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.

Printer	Lines per Page			
	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	—

CHGPRTF
LPI

***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.

CHGPRTF
RPLUNPRT

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.

***DEV D:** 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.)

CHGPRTF
ALIGN

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.

***SAME:** 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.

**CHGPRTF
SPOOL**

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.

CHGPRTF
FILESEP

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.

CHGPRTF
LVLCHK

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)
```

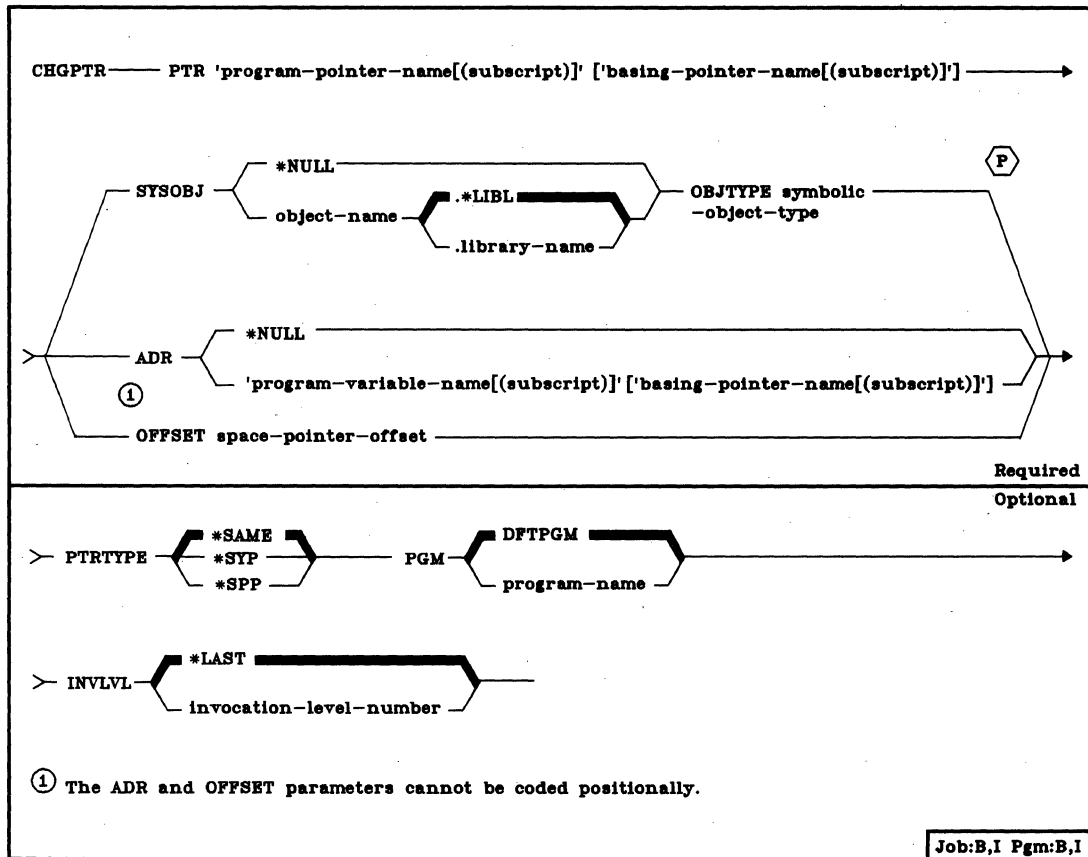
This command changes all IBM-supplied print files in library QSYS to use 88 lines of 132 characters (8 lines per inch), but to skip to the next page after 80 lines.

CHGPTR (Change Pointer) Command

CHGPTR

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

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.

CHGPTR
PTRTYPE

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

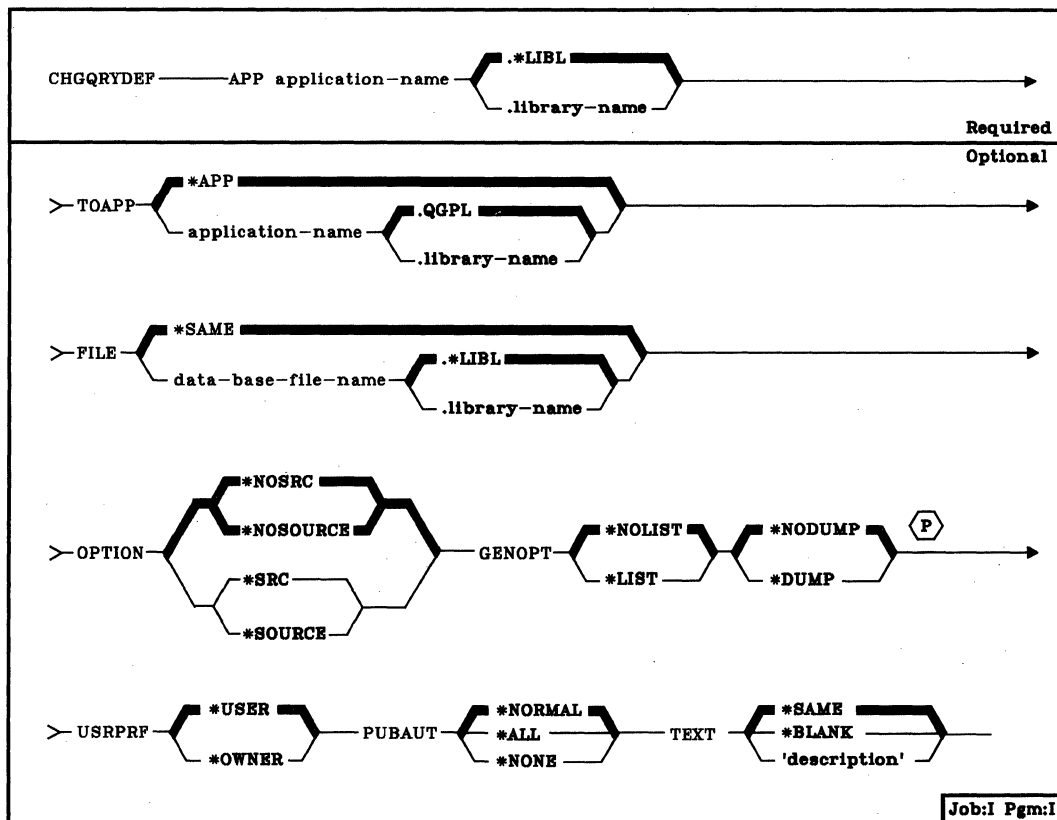
```
CHGPTR PTR(DATAFILPTR) SYSOBJ(MYFILE.QGPL) +  
OBJTYPE(*FILE)
```

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).

**CHGQRYDEF
TOAPP**

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.

CHGQRYDEF
(Example)

Example

```
CHGQRYDEF APP(TEST1) TOAP(TEST2) +  
TEXT('Create application TEST2, based on TEST1')
```

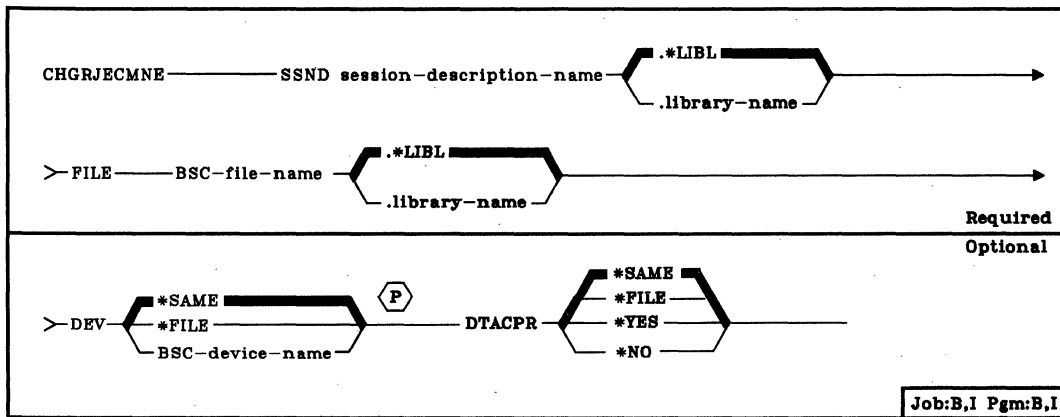
This command begins a prompting sequence that allows you to create an application named TEST2 in library QGPL based on application TEST1 in your library list. Your responses to the prompts can result in changes to the TEST2 application attributes (which differ from the based-on application TEST1). Application 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 RJE 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.)

CHGRJECMNE
DEV

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)
```

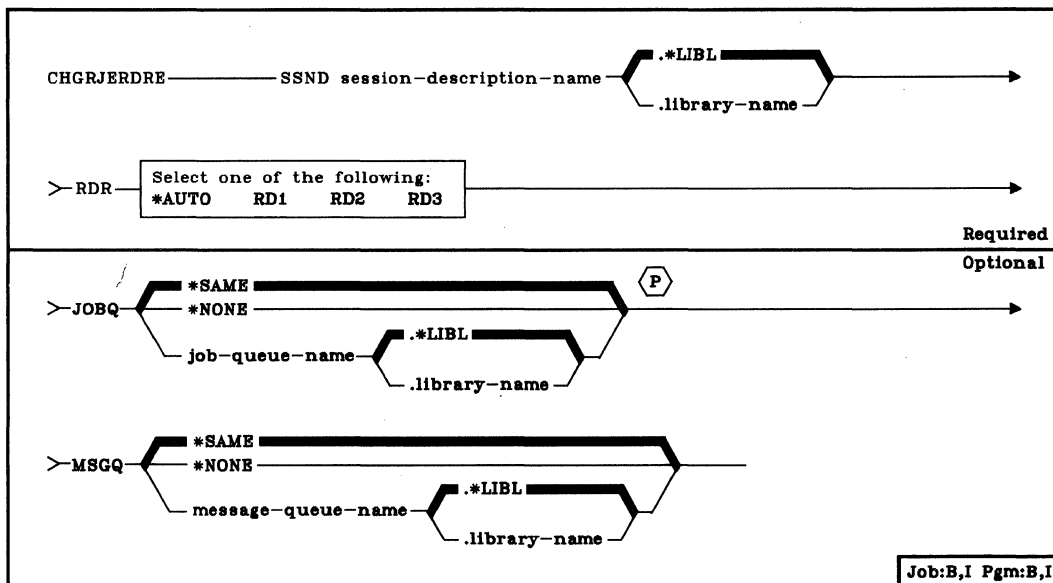
This command changes the communication entry named DEVPRT1.USERLIB in session description named RJE in library USERLIB. The entry is changed to prevent data compression.

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.)

CHGRJERDRE
RDR

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**
(Example)

```
CHGRJERDRE SSND(RJE.USERLIB) +  
RDR(RD1) +  
MSGQ(BROWN.DEPT52)
```

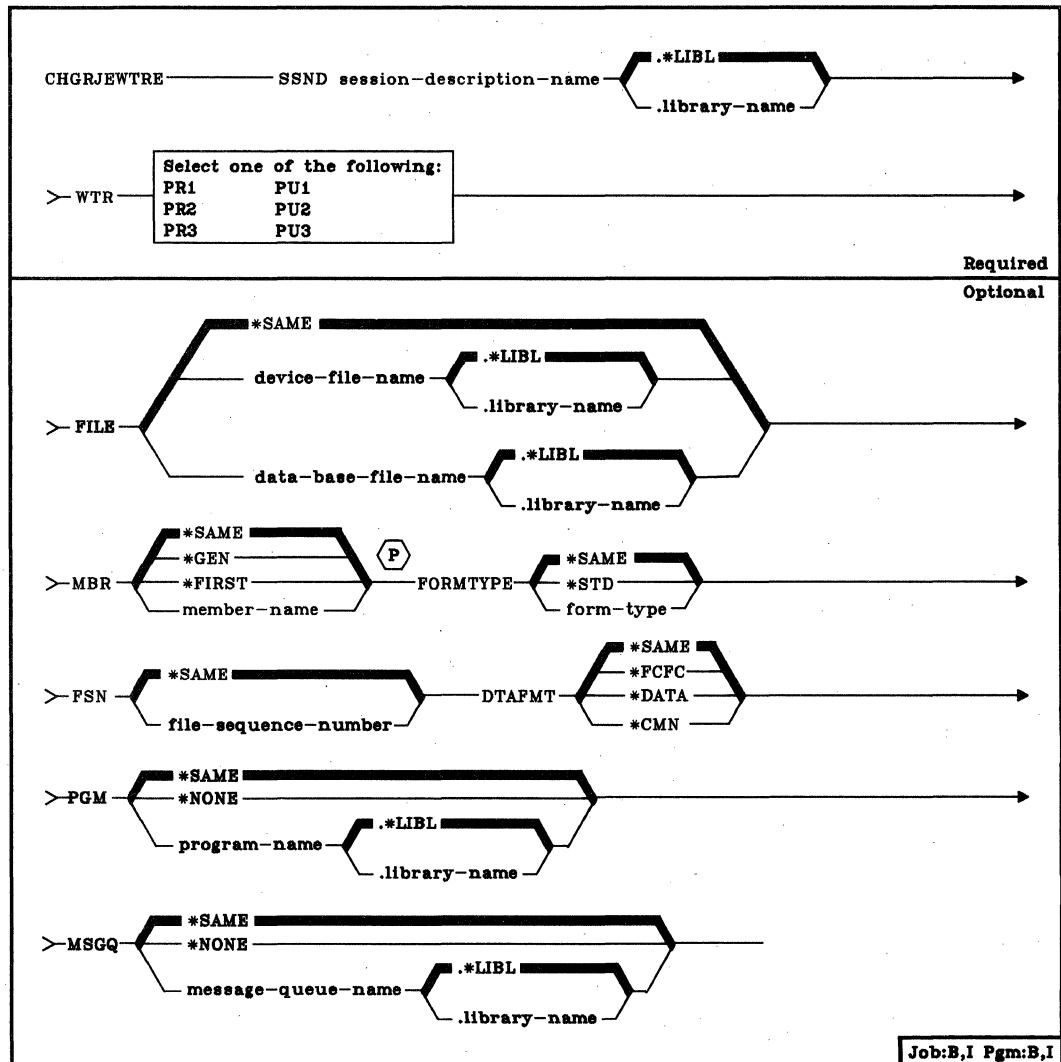
This command changes RD1 reader entry in session description named RJE in library USERLIB. The user message queue is changed to BROWN in library DEPT52. Messages associated with jobs submitted to RD1 will be written to message queue named BROWN in library DEPT52.

CHGRJEWTR (Change RJE Writer Entry) Command

The Change RJE Writer Entry (CHGRJEWTR) command changes the attributes in an existing session description RJE 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 (CHGRJEWTR) 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.)

**CHGRJEWTR
MBR**

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:

Affffffcc or Bffffffcc

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.

*SAME: 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 alphanumeric 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.

*SAME: 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.

*SAME: 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.

**CHGRJEWTR
PGM**

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

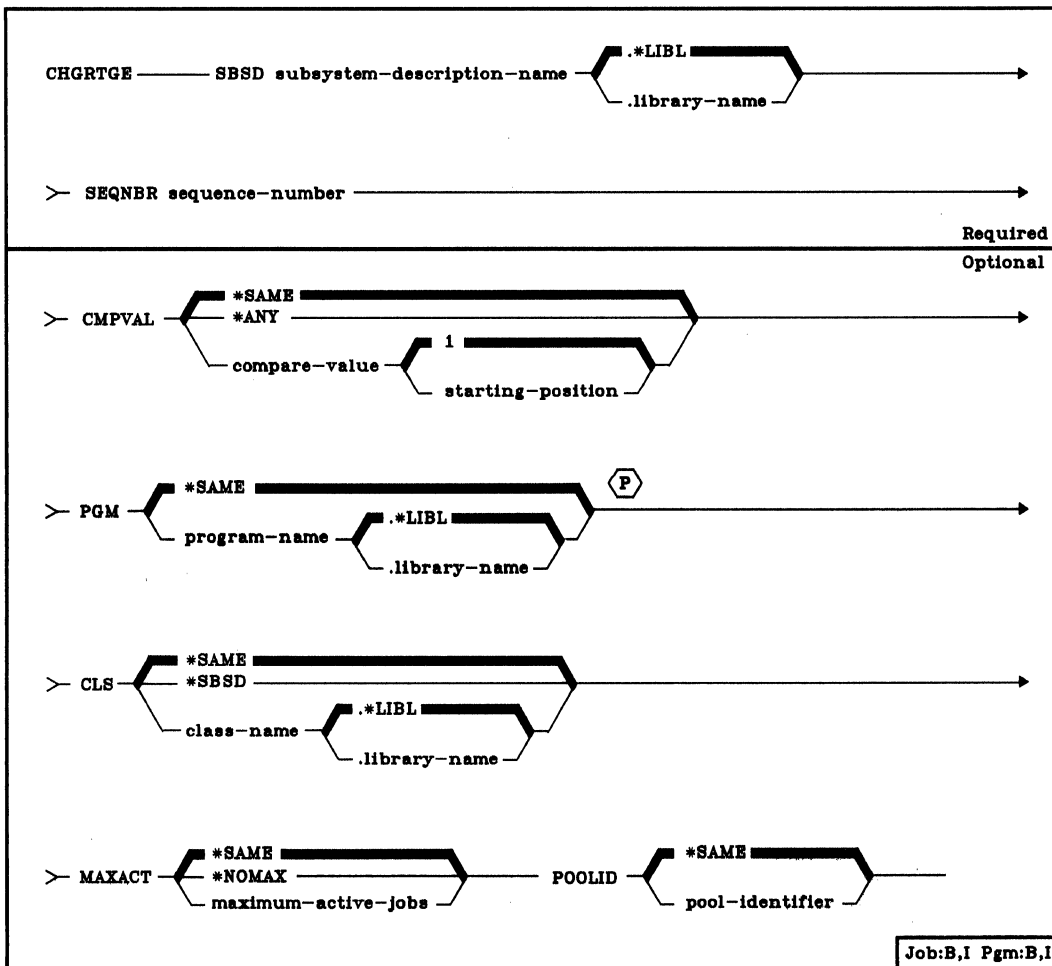
```
CHGRJEWTR SSND(RJE.USERLIB) +  
WTR(PR1) +  
FILE(NIGHTPRT.USERLIB) +  
DTAFMT(*FCFC) +  
MSGQ(*NONE)
```

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.



CHGRTGE
SBSD

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 SBSDB 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.

CHGRTGE
(Examples)

Examples

```
CHGRTGE SBS(DORDER.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.

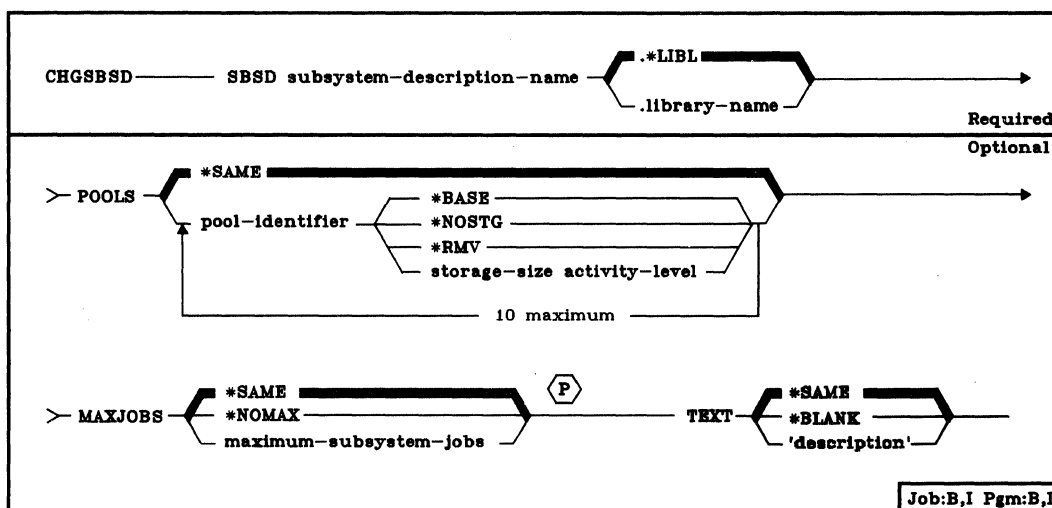
```
CHGRTGE SBS(DPGMR.T7) SEQNBR(157) +  
PGM(INTDEV.T7)
```

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.

**CHGSBSD
MAXJOBS**

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 (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.

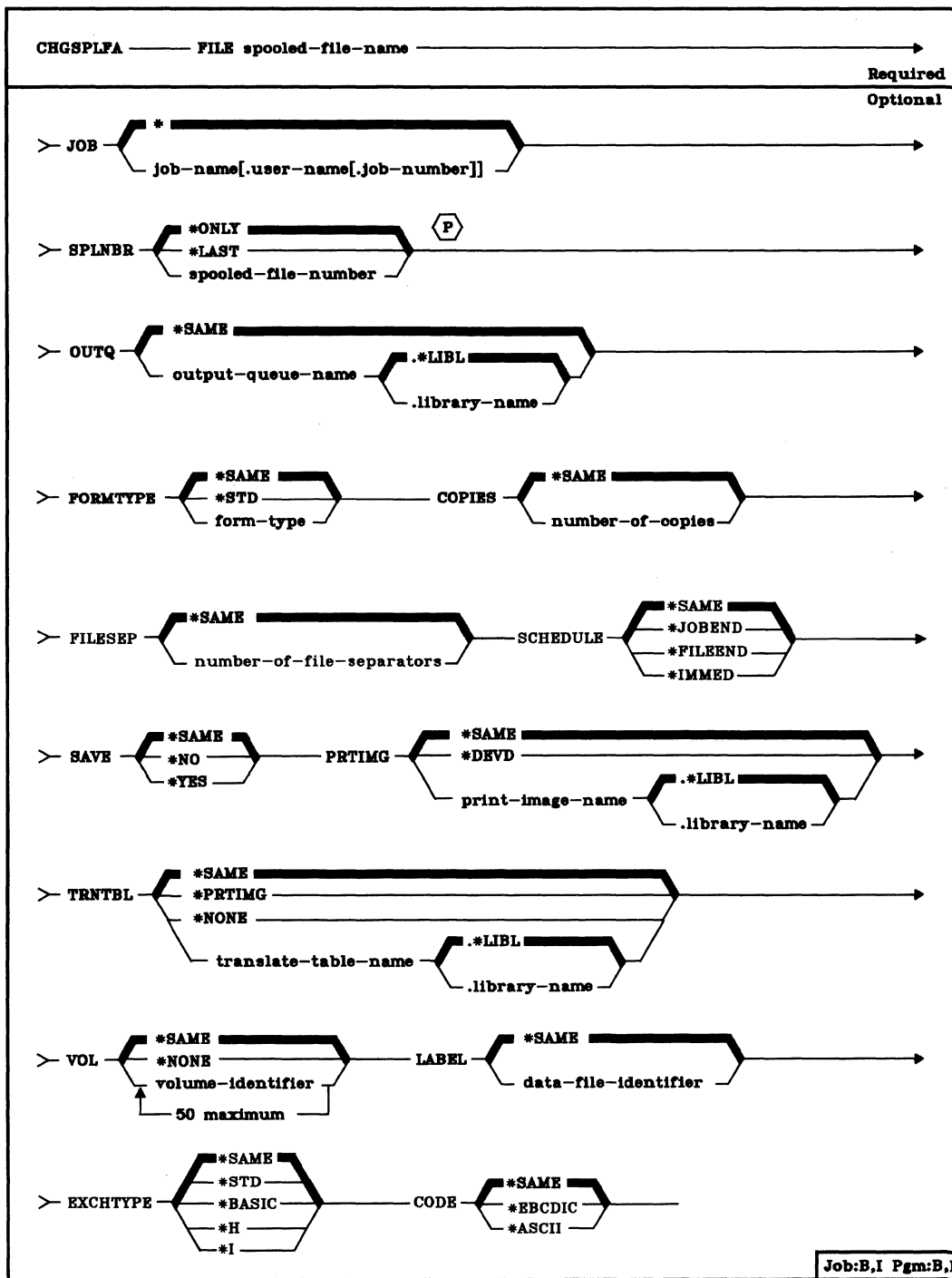
CHGSPLFA (Change Spooled File Attributes) Command

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
(Diagram)



**CHGSPLFA
FILE**

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.

***:** 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.)

***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 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.

***SAME:** 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.

***SAME:** 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 alphanumeric 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.

CHGSPLFA
SAVE

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.

***DEV D:** 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.

CHGSPLFA
EXCHTYPE

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)
```

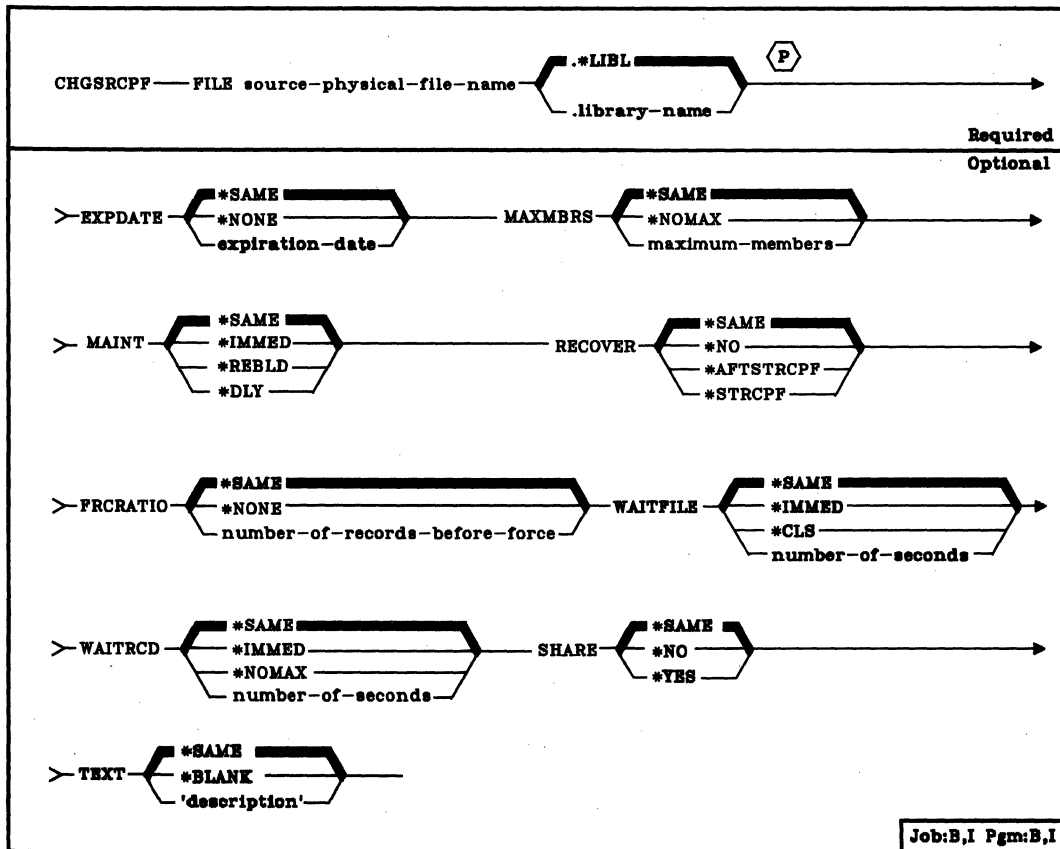
This command changes the attributes of the output file DEPT511 that is produced by the submitter's job. It changes the number of output copies to two and specifies that five separator pages (or cards) are to precede each copy.

CHGSRCPF (Change Source Physical File) Command

CHGSRCPF

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.)

EXPDTE 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 EXPDTE is specified, all members in the file will be changed. An expired member may be changed to non-expired by changing the EXPDTE parameter. The expiration date must be later than or equal to the current day's date.

***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	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.

**CHGSRCPF
RECOVER**

***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.)

***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).

CHGSRCPF
SHARE

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')
```

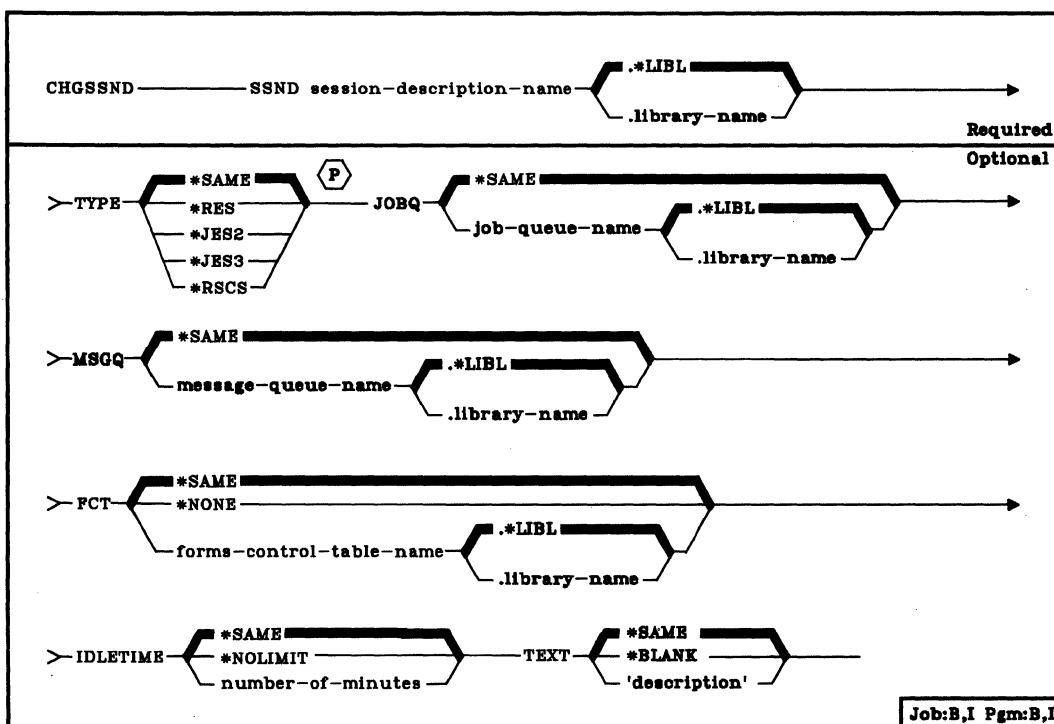
This command changes the expiration date of all members in file INV to October 31, 1987.

CHGSSND (Change Session Description) Command

The Change Session Description (CHGSSND) command changes attributes in an existing RJE 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.)

CHGSSND
TYPE

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 RJE 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, RJE holds all RJE reader job queues defined for this RJE 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 RJE session should remain idle before transmitting the LOGOFF or SIGNOFF command to the host system. Valid values are 0 through 99.

CHGSSND
TEXT

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
<p>CHGSYSVAL _____ SYSVAL system-value-name _____ VALUE new-value _____</p> <p>① Some system values can contain a list of values. They must be enclosed in apostrophes.</p>
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.

CHGSYSVAL
(Examples)

Examples

CHGSYSVAL SYSVAL(QHOUR) VALUE('12')

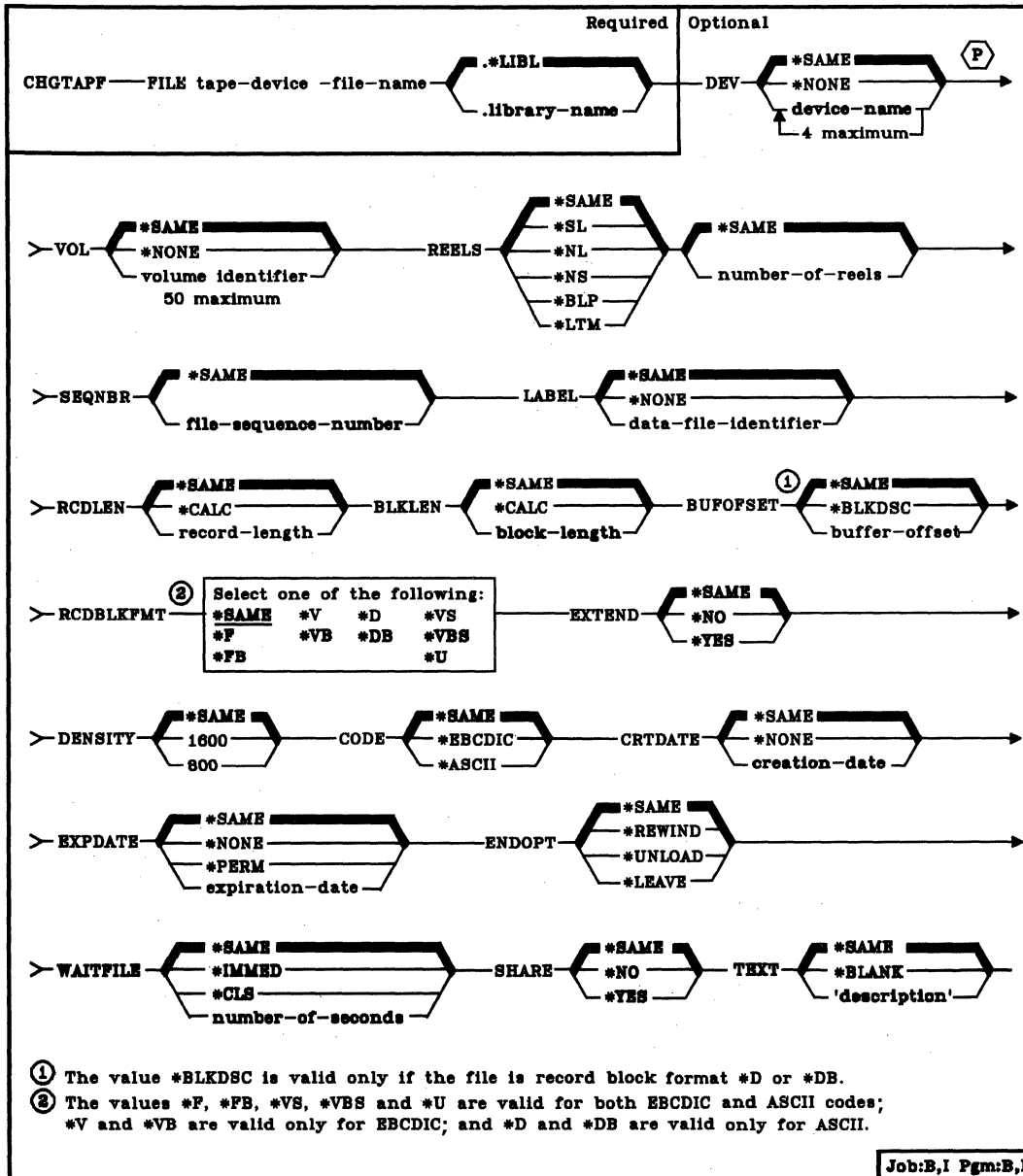
This command changes the value of the system value QHOUR (which is a subvalue of the QTIME system value) to 12. Because QHOUR is a character variable, 2 characters long, the system value is set to the character representation of 12, which is hex F1F2 and, therefore, must be enclosed in apostrophes. Also, 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.



**CHGTAPF
FILE**

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 alphanumeric 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.

CHGTAPF
SEQNBR

***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 alphanumeric 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.)

**CHGTAPF
RCDLEN**

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 RCDBLKFMNT 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					
CODE	RCDFBLKFMT	FILETYPE(*DATA)		FILETYPE(*SRC)	
		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 RCDBLKFMFMT 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

**CHGTAPF
BUFOFSET**

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 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 program 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 *ASCII	*F *U *F *U	BLKLEN = RCDLEN BLKLEN = RCDLEN + BUFOFSET
*EBCDIC *ASCII	*FB *FB	BLKLEN = RCDLEN * n BLKLEN = (RCDLEN * n) + BUFOFSET n is the number of records in a maximum-length block
*EBCDIC *ASCII	*V *D	BLKLEN = RCDLEN + 8 BLKLEN = RCDLEN + 4 + BUFOFSET
*EBCDIC *ASCII	*VB *DB	BLKLEN >= RCDLEN + 8 BLKLEN >= RCDLEN + 4 + BUFOFSET
*EBCDIC *ASCII	*VS *VBS *VS *VBS	BLKLEN >= 18 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. **RCDBLKFM(*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 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.

**CHGTAPF
DENSITY**

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.

**CHGTAPF
WAITFILE**

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
(Example)

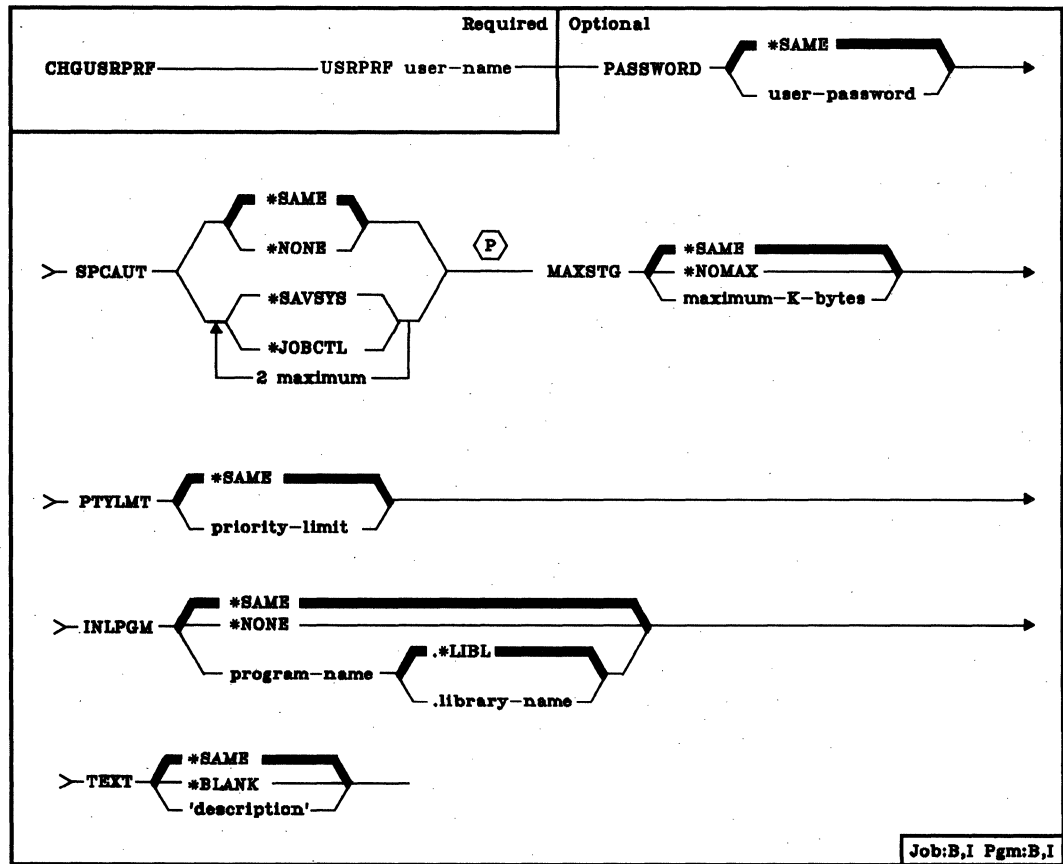
```
CHGTAPF FILE(TAPE01) LABEL(TUESDAY)
```

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.)

CHGUSRPRF
PTYLMT

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.

CHGVAR
VALUE

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.

**CHGVAR
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	' 123.1 '	5,2	123.10
10	' 123.00 '	5,0	123
10	' -123 '	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.

CHGVAR
(Examples)

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: XYZ.

```
CHGVAR &VAR1 '12'
```

Assuming &VAR1 is a character variable that is 6 characters long, the result is again padded on the right with blanks: 12. The apostrophes are required in this example.

```
CHGVAR VAR(%SUBSTRING(&A 4 3)) VALUE(REP)
or
CHGVAR VAR(%SST(&A 4 3)) VALUE(REP)
```

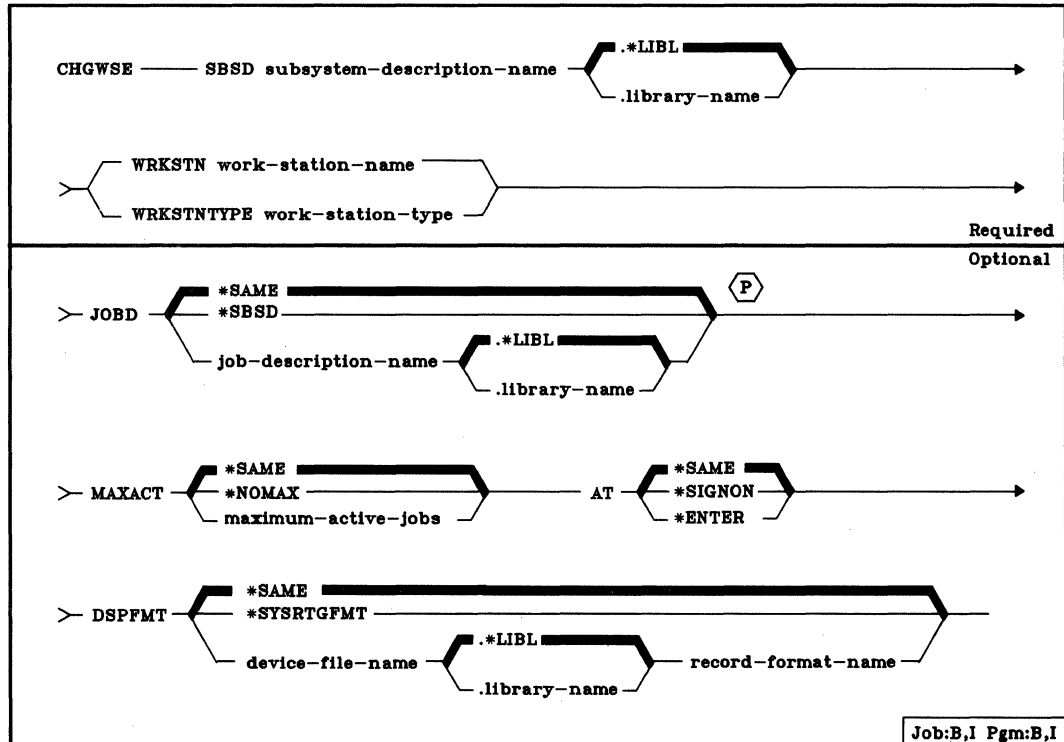
The substring built-in function is used to change 3 characters of the character constant in the variable named &A. If &A has a value of ABCDEFGH, the fourth, fifth, and sixth characters in &A are set to REP, and the result is ABCREPGH.

CHGWSE

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 device description name of the work station whose work station entry is to be changed.

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

**CHGWSE
WRKSTNTYPE**

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
5251	5251 Display Station
5252	5252 Dual Display Station
5291	5291 Display Station
5292	5292 Color Display Station
*CONS	System console display

A value must be specified for either the WRKSTN or the WRKSTNTYPE parameter, but not both.

JOBID 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.

*SAME: The same job description is to be used.

*SBSD: The job description having the same qualified name as the subsystem description, specified by the SBSID 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.)

*SAME: 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.

CHGWSE
AT

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 (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.

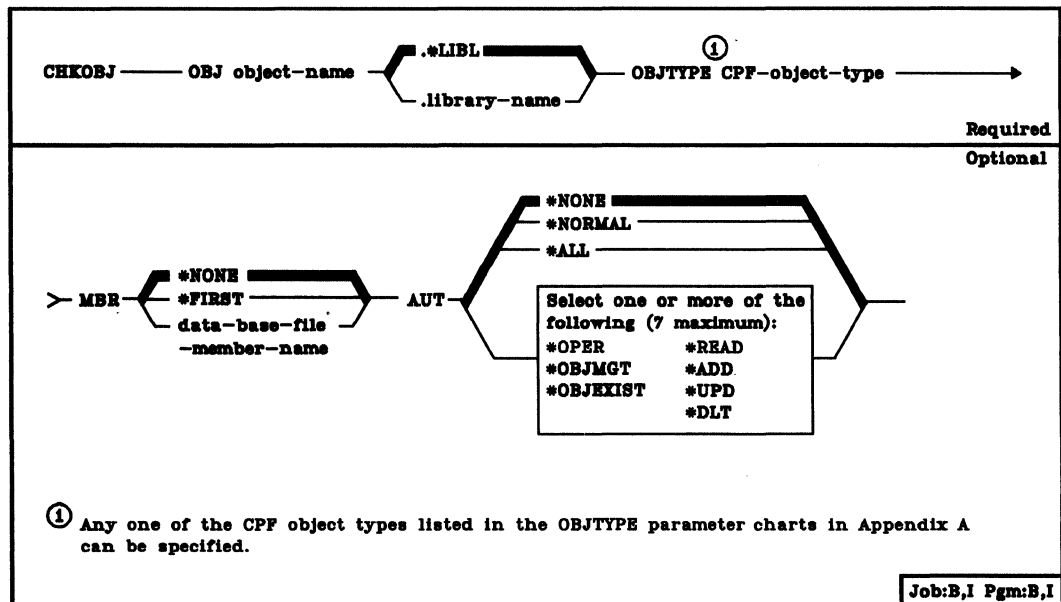
CHKOBJ (Check Object) Command

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.)



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.

CHKOBJ
(Examples)

***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

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.

CLNPRT _____, DEV 3203-device-name _____	Required
	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

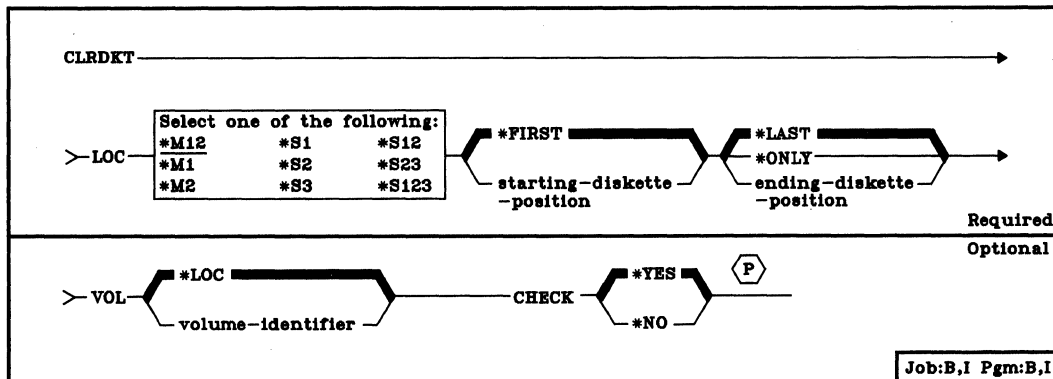
The Clear Diskette (CLRDKT) command deletes all files, active and inactive, from one or more diskettes by deleting the data file identifiers from the diskette label area on each diskette. A single (expired) file is defined, covering the entire diskette, and is identified as DATA. The data contained in 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.



**CLRDKT
LOC**

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.

CLRJOBQ	JOBQ job-queue-name	.*LIBL ┌───────────┐ │ │ └───────────┘ .library-name	Required
			Job:B,I Pgm:B,I

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

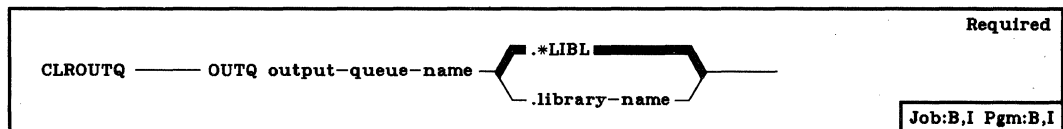
```
CLRLIB LIB(A)
```

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)
```

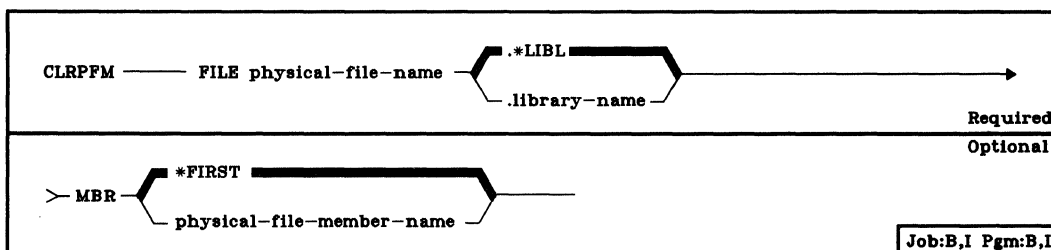
This command removes from the output queue, QPUNCH, the entries for all spooled files that are waiting to be punched or are being held. The entries for the file currently being punched and those files still receiving records from executing programs are not affected.

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.

***FIRST:** 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.

CLRPFM
(Example)

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*.

CLRTRCDTA _____	Optional
	Job:B,I Pgm:B,I

Example

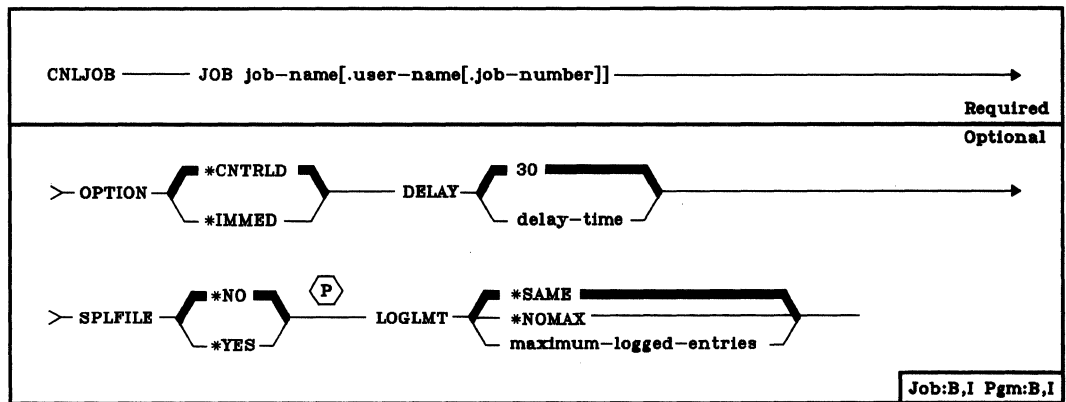
CLRTRCDTA

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.

CNLJOB
(Examples)

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.

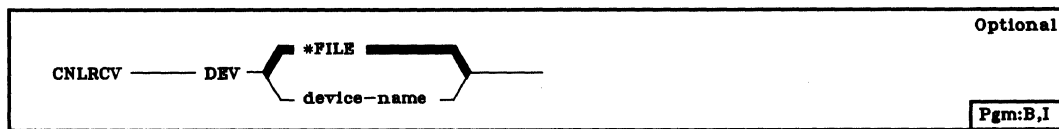
```
CNLJOB JOB(WSTATION2) OPTION(*CNTRLD) +  
      DELAY(50) SPLFILE(*NO)
```

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

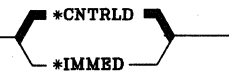
```
CNLRCV DEV(MYDISPLAY)
```

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.

Required	Optional
CNLRDR — RDR reader-name	OPTION —  *CNTRLD *IMMED
Job:B,I Pgm:B,I	

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)

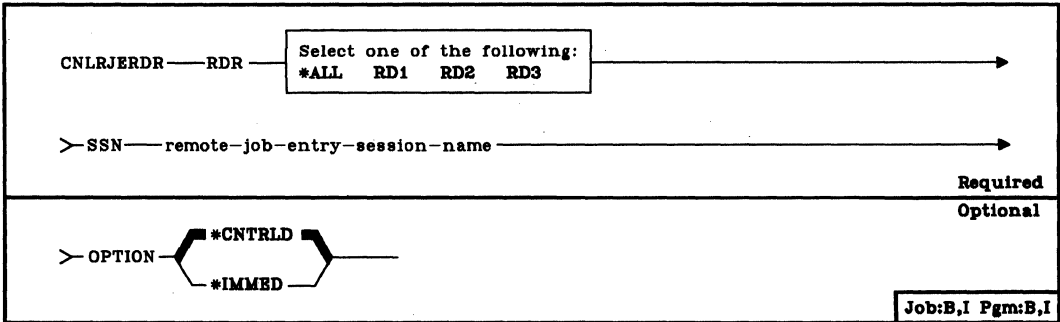
This command stops the reader CARD as soon as the current job is completely read in and releases that device to the system. To process any jobs that remain in the input stream, another reader can be started, but the system operator may have to put the job card and other cards needed for the next job 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.

**CNLRJERDR
OPTION**

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.

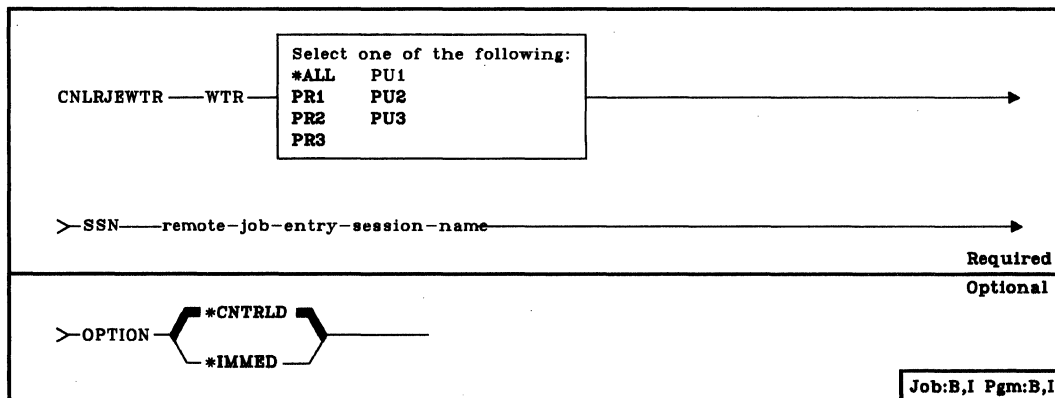
CNLRJEWTR (Cancel RJE Writer) Command

CNLRJEWTR

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.

CNLRJEWTR
OPTION

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.

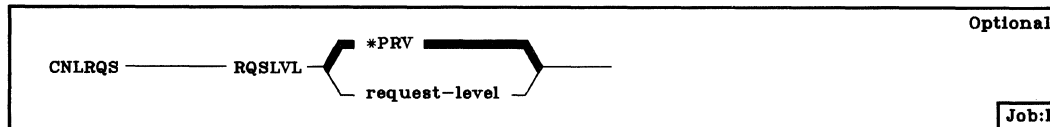
CNLRQS (Cancel Request) Command

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.

CNLRQS
(Examples)

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)

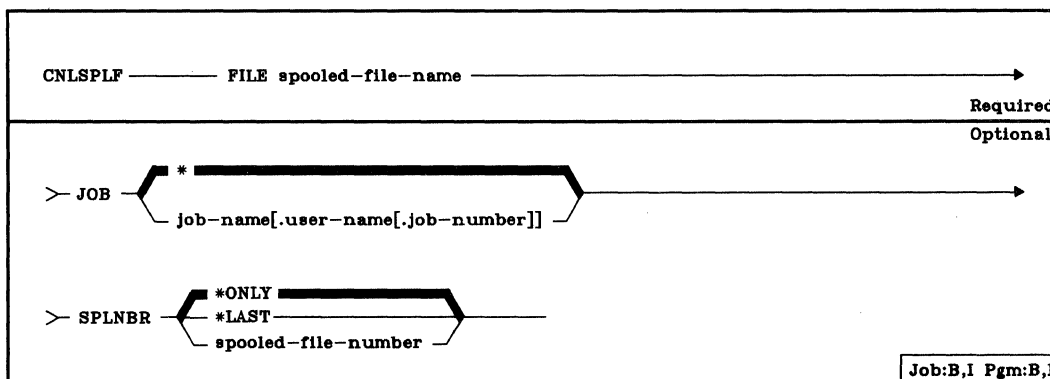
In this example, the request made at the highest level (CALL PROGA) is canceled. Consequently, any requests made between level 1 and level 3 are also canceled.

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.)

**CNLSPLF
SPLNBR**

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.

Required	Optional
CNLWTR — WTR writer-name	OPTION { *CNTRLD *IMMED *PAGEEND }
	Job:B,I Pgm:B,I

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.

From File	To 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

¹An 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.

²An 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 (NBRRCD 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).

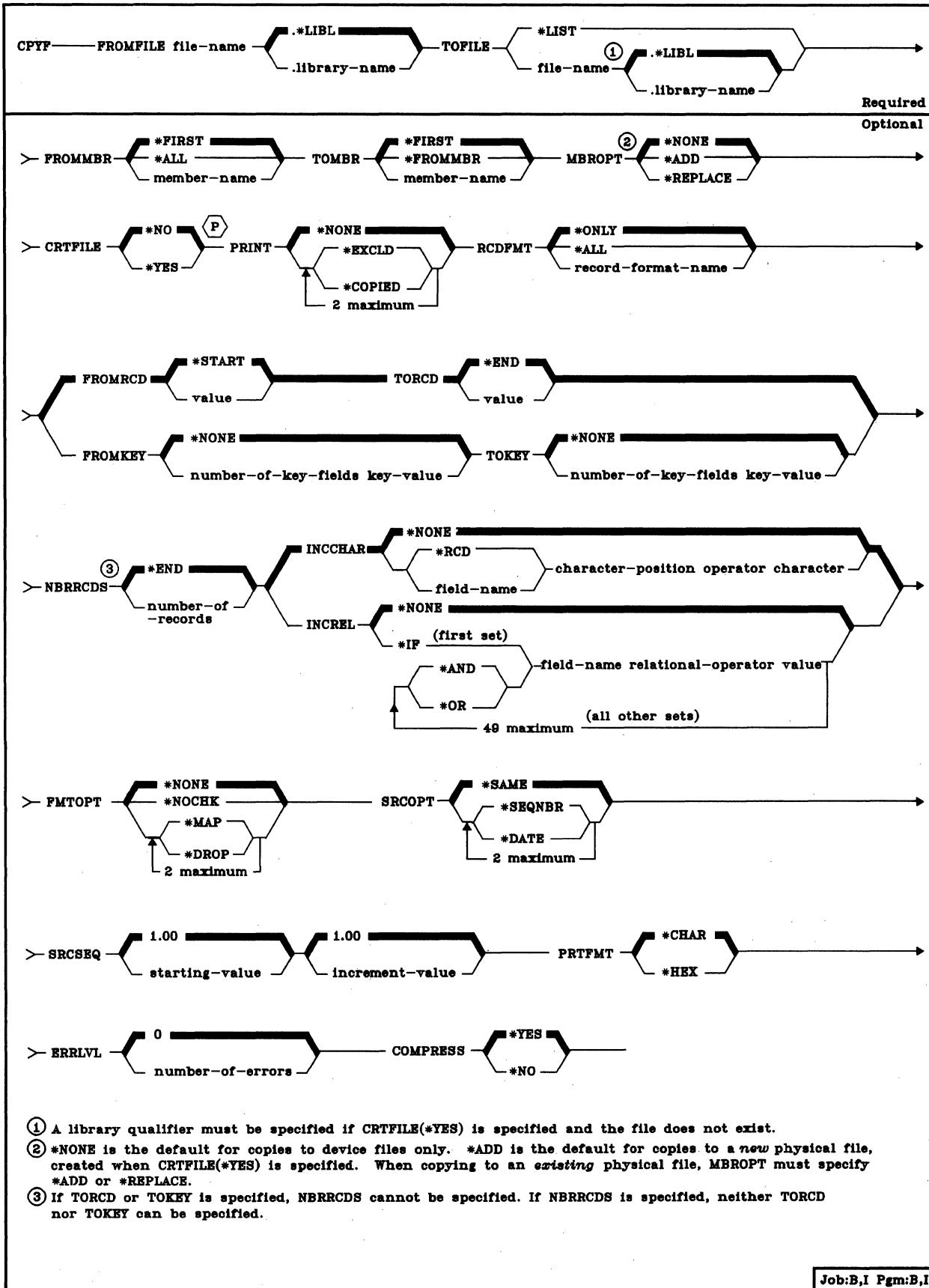
CPYF

- 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.

CPYF
(Diagram)



Job:B,I Pgm:B,I

CPYF
(Chart)

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).

Parameter	Data Base Files				Device Files									
	Physical		Logical		Diskette		Tape		Card		Printer		Inline Data	
	From	To	From	To	From	To	From	To	From	To	From	To	From	To
FROMFILE	X		X		X ¹		X		X					X
TOFILE		X				X ¹		X		X		X ²		
FROMMBR	X		X		X ³		X ³							
TOMBR		X				X ³		X ³						
MBROPT		X												
CRTFILE	X		X											
PRINT	X	X	X		X	X	X	X	X	X		X	X	
RCDFMT			X											
FROMRCD	X ⁴		X ⁵		X		X		X				X	
TORCD	X ⁴		X ⁵		X		X		X				X	
FROMKEY	X ⁶		X ⁶											
TOKEY	X ⁶		X ⁶											
NBRRCDS	X		X		X		X		X				X	
INCCHAR	X		X		X		X		X				X	
INCREL	X		X											
FMTOPT	X ⁷	X	X ⁷											
SRCOPT	X ⁷	X	X ⁷											
SRCSEQ	X ⁷	X	X ⁷											
PRTFMT ⁸												X		
ERRLVL	X	X	X											
COMPRESS	X	X												

¹Valid only if the other file (being copied to, or from) is not another diskette file.
²Valid only if *LIST or a program-described printer file is specified. An externally described printer file *cannot* be specified.
³Valid only if a data file identifier is not previously provided on a file override.
⁴Valid only if FROMKEY and TOKEY are not specified.
⁵Valid only if the logical file has an arrival sequence access path.
⁶Valid only if FROMRCD and TORCD are not specified.
⁷Valid only if copy is to a physical file.
⁸Valid 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.

***FIRST:** 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.

CPYF
TOMBR

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).

**CPYF
PRINT**

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 INCCCHAR 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.

**CPYF
FROMKEY**

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 NBRRCD 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.

NBRRCD 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 NBRRCD.

***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.

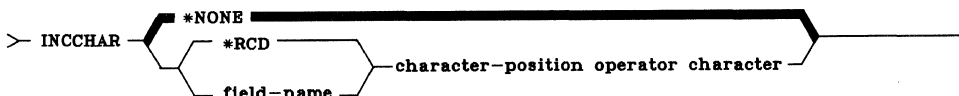
number-of-records: Enter a value, no more than nine digits in length, that specifies the number of records to be copied from the file.

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.

CPYF
INCREL

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:

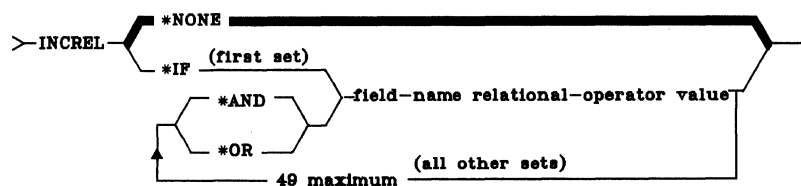
1. One of the logical operators *IF, *AND, or *OR
2. The name of the field to be compared
3. One of the relational operators (from the list that follows)
4. The comparison value

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.

***NONE:** 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.

***SAME:** 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.

**CPYF
SRCSEQ**

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, ERRLLVL, 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) ERRVL(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) +
FROMMMBR(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) +
FROMMMBR(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) +
FROMMMBR(TOTSALES) RCD FMT(AA) +
NBRR CDS(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.

CPYF
(Examples)

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			Existing PAYROLL.FINANCE File, MBR1		
Name	Emp#	Position	Name	Emp#	Position
Jane P	15678	Clerk	Mary P	13467	Sec
Tom J	23451	Clerk	Tommy J	21457	Sr Clerk
Bob H	85712	Sec	Jake T	22444	Jr Clerk
Joe P	71567	Cashier	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	
Bob H	85712	Sec	
Joe P	71567	Cashier	

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)
```

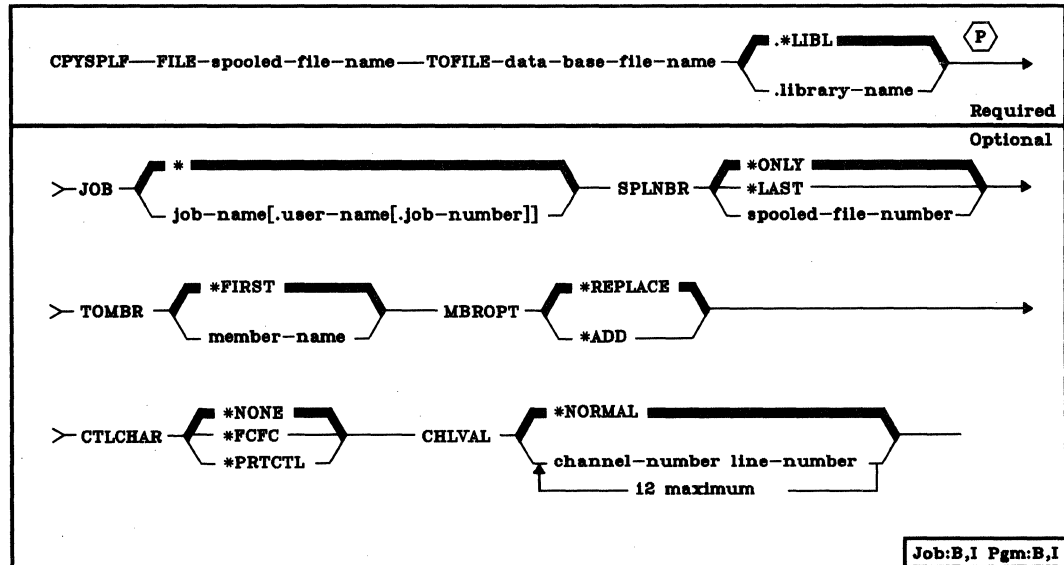
This command copies all card records with an X in column 80 to the tape device. The device file name for the card reader is CARDIN. The device file name for the tape device is TAPEOUT. All cards that are not copied are to be printed. The TOMBR parameter has specified the tape label to which the records 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.

*: 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.)

*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 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.

*FIRST: 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.

*REPLACE: 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.

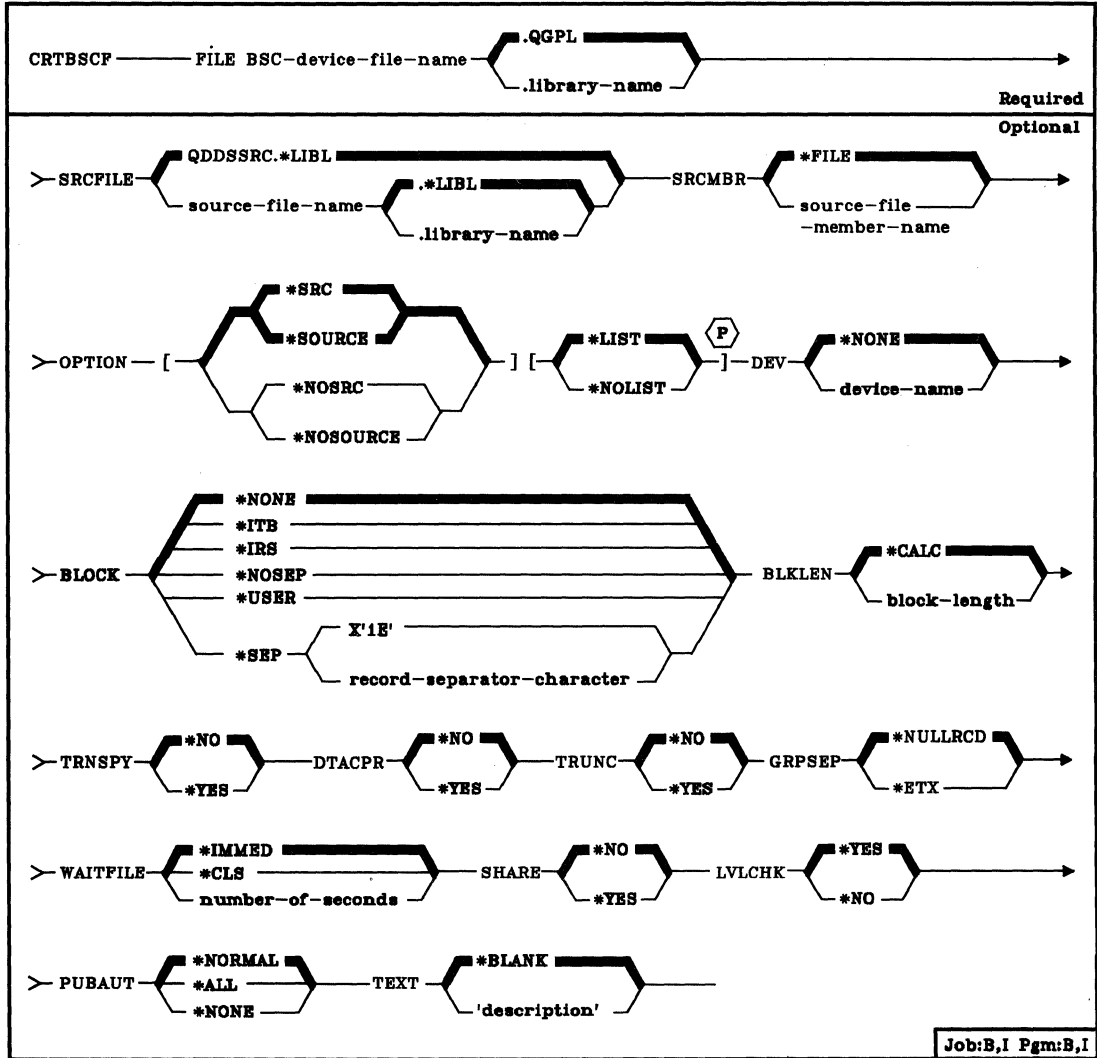
```
CPYSPLF FILE(QPRINT) TOFILE(MYFILE.MYLIB) JOB(PAYROLL02) +
MBROPT(*ADD) CTLCHAR(*FCFC) CHLVAL( (1 3) (4 15) )
```

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.

CRTBSCF
BLKLEN

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.

**CRTBSCF
WAITFILE**

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.

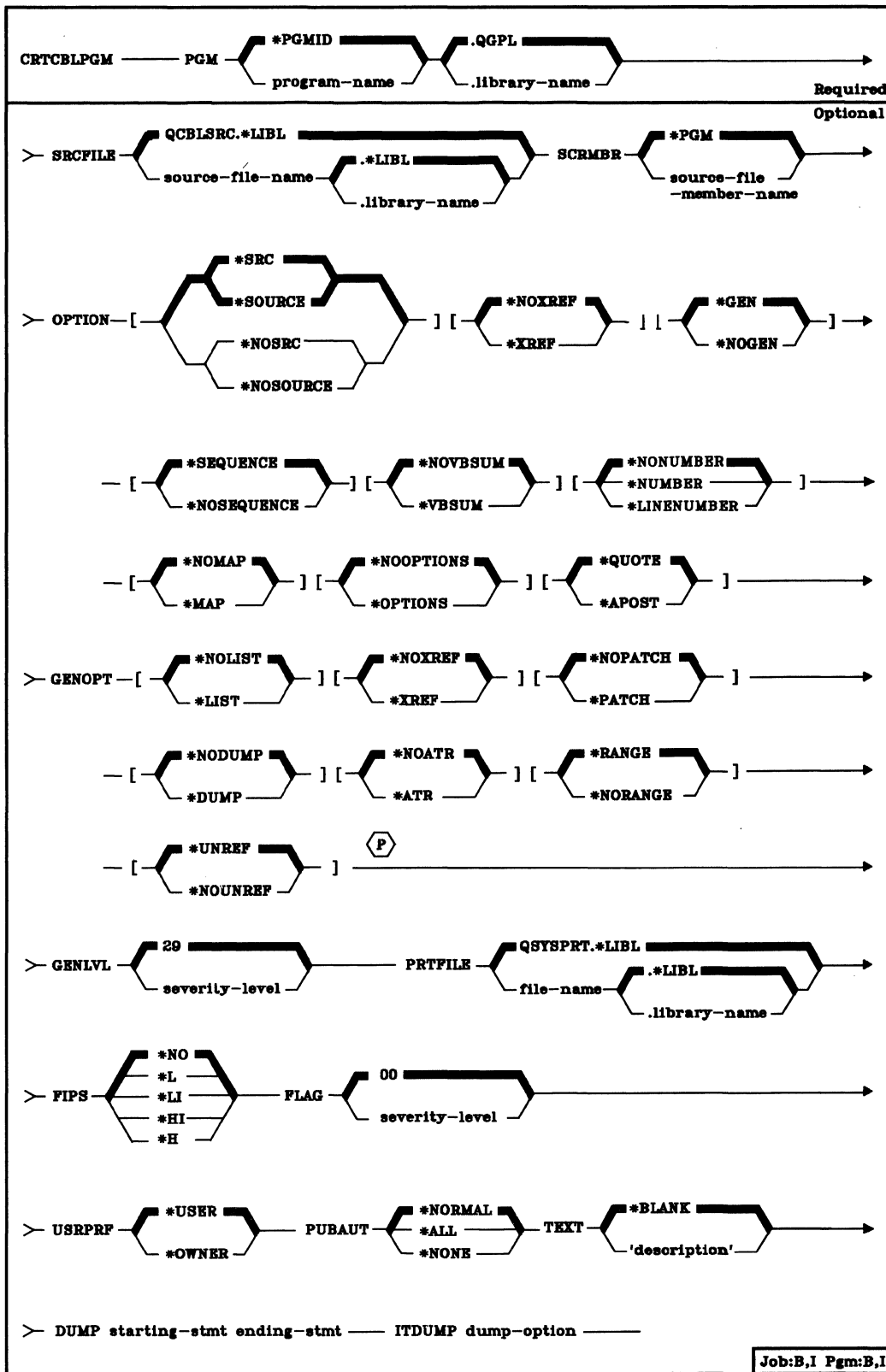
CRTCBLPGM (Create COBOL Program) Command

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 alphanumeric characters, the first of which must be alphabetic.

CRTCBLPGM
(Diagram)



CRTCBLPGM
PGM

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.

***SOURCE** 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.

***NOXREF:** 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.

***GEN:** 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.

***SEQUENCE:** 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.

***NOVBSUM:** Verb usage counts are not printed.

***VBSUM:** Verb usage counts are printed.

***NONUMBER:** 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.

***NOMAP:** The compiler does not list the Data Division map.

***MAP:** The compiler lists the Data Division map.

CRTCBLPGM
GENOPT

***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**
(Example)

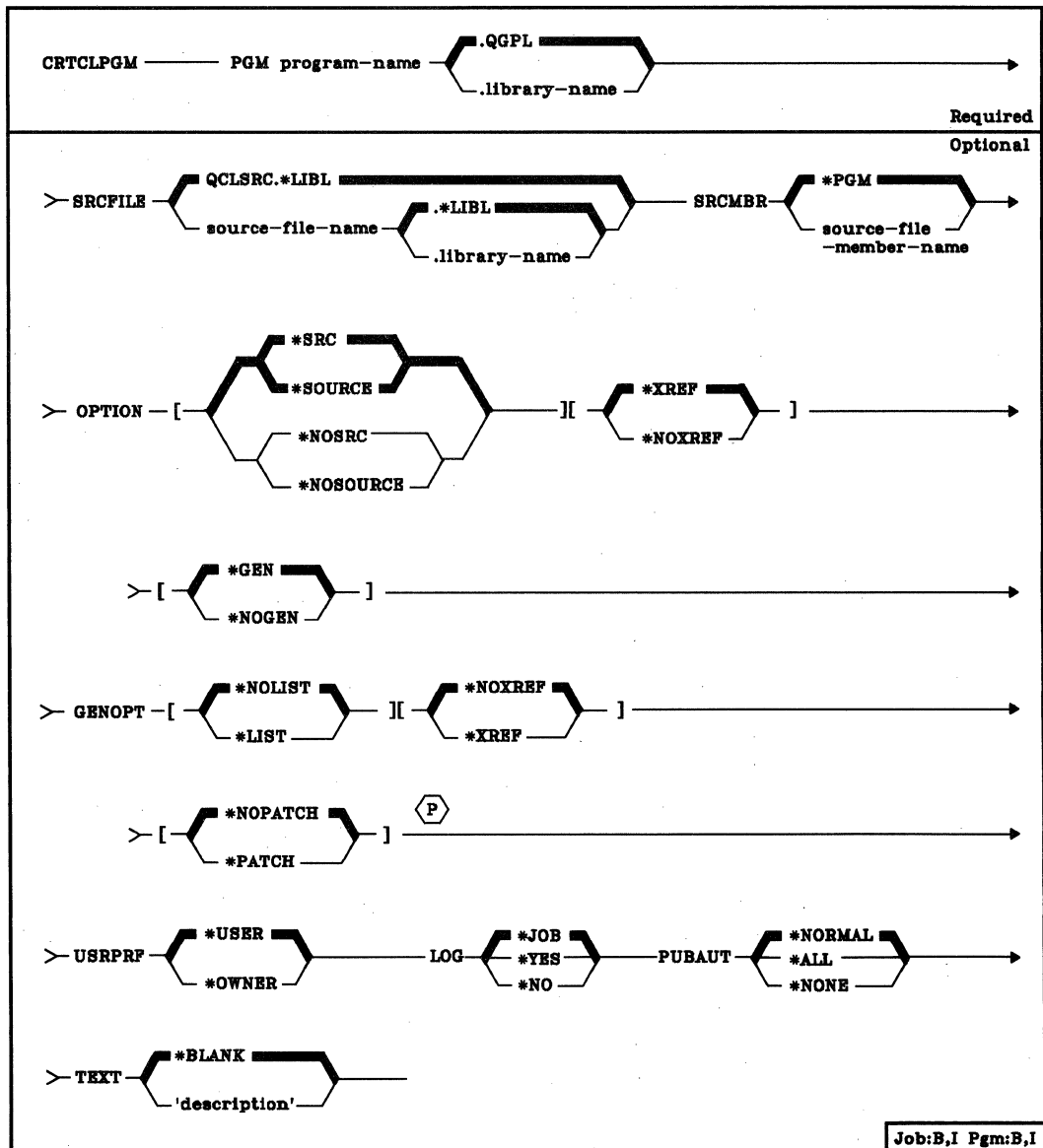
```
CRTCBLPGM PGM(STATS.ACCTS) SRCFILE(ACTIVE.ACCTS) GENOPT(*PATCH) +  
GENLVL(19) FLAG(19) TEXT('Statistical analysis program for active +  
accounts')
```

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.



Job:B,I Pgm:B,I

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.

CRTCLPGM
GENOPT

***GEN:** 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.)

***NOLIST:** 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.

***NOXREF:** 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.

***NOPATCH:** 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).

***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.

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.

CRTCLPGM
(Examples)

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.

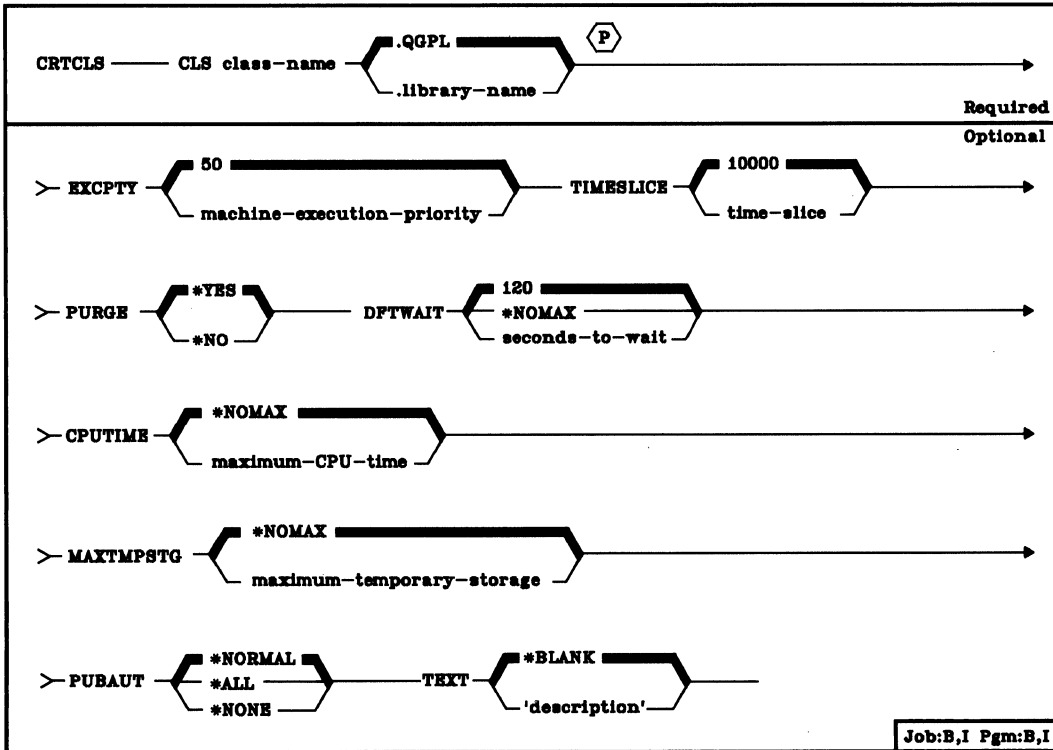
```
CRTCLPGM PGM(PARTS) SRCFILE(PARTDATA.MYLIB) PUBAUT(*NONE) +  
TEXT('This program displays all parts data')
```

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

CRTCLS

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.

CRTCLS
TIMESLICE

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.

CRTCLS
(Example)

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

CRTCMD

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.

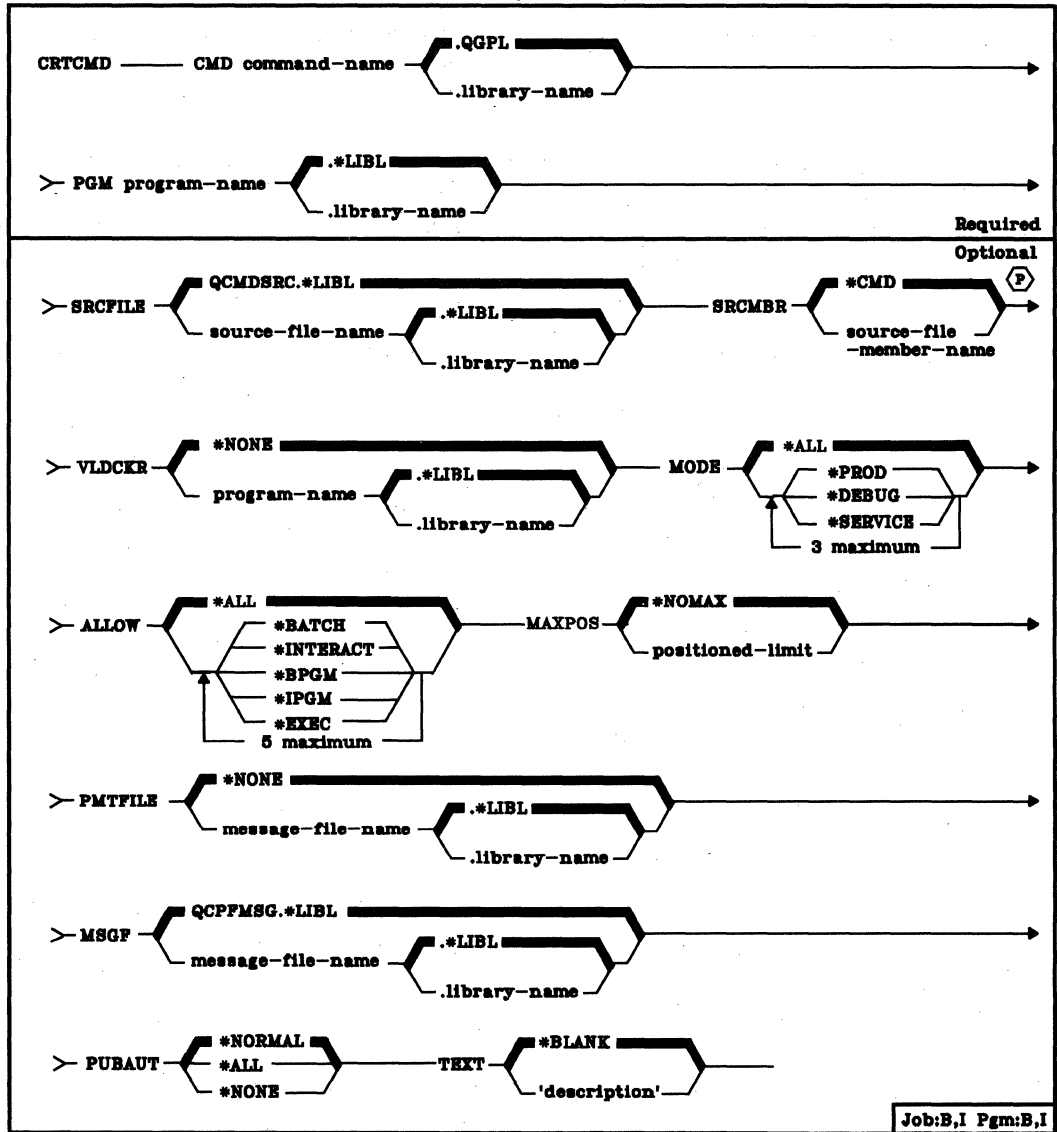
To create a command, a set of command definition statements are entered into a source file. The CRTCMD command is used to process the source file and create a command definition object. The following command definition statements are used as needed:

- 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
(Diagram)



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.

QCMSRC: The IBM-supplied source file, QCMSRC, 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.

***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 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.

***ALL:** 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.

***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.

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).

***NOMAX:** 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.

***NONE:** 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.)

QCPFMSG: 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.)

**CRTCMD
PUBAUT**

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.

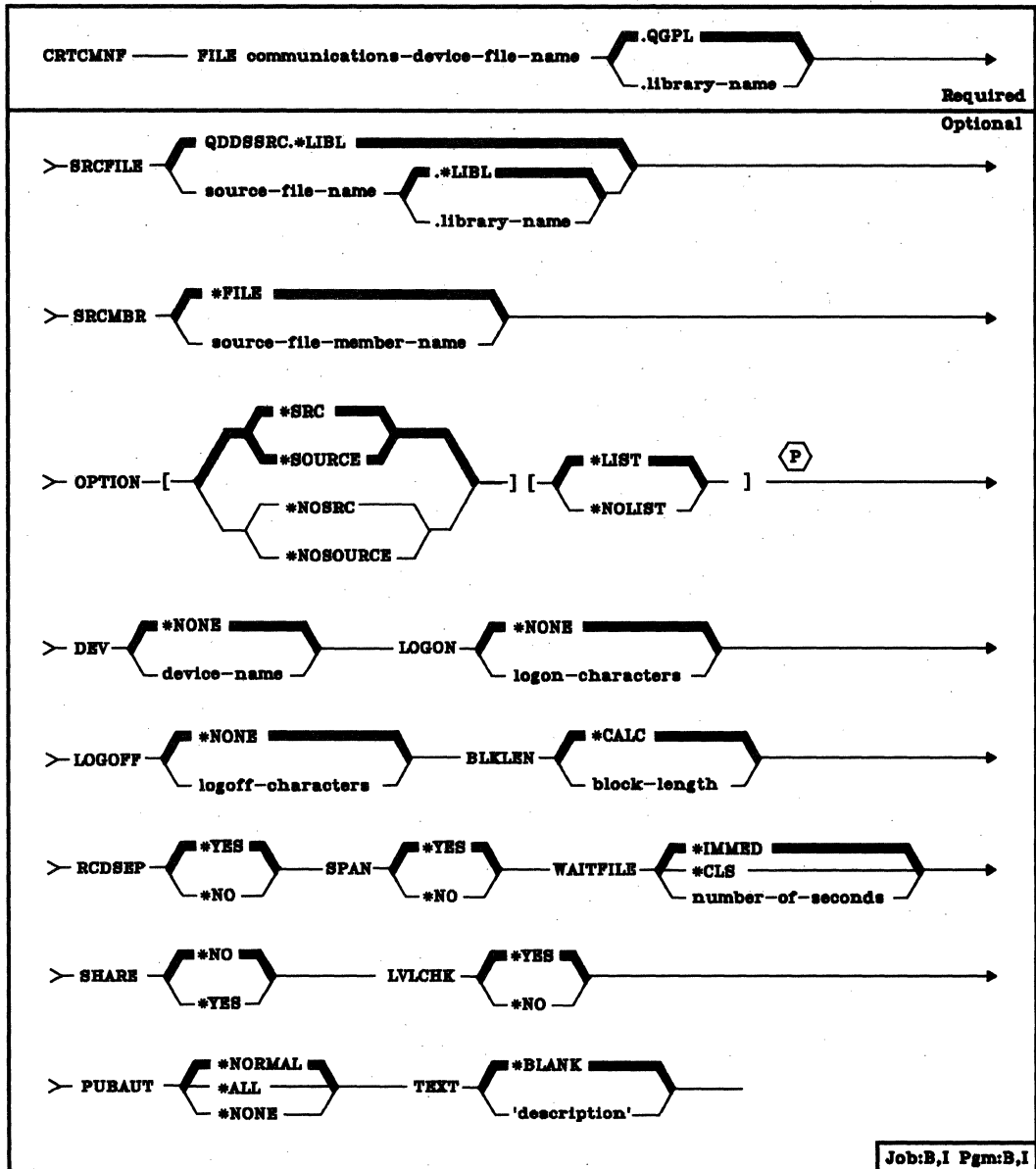
CRTCMNF (Create Communications File) Command

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.

CRTC MNF
(Diagram)



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.

CRTC MNF
DEV

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 OVR CMNF 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.

CRTC MNF
SHARE

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**CRTC MNF**
(Example)

```
CRTC MNF 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.

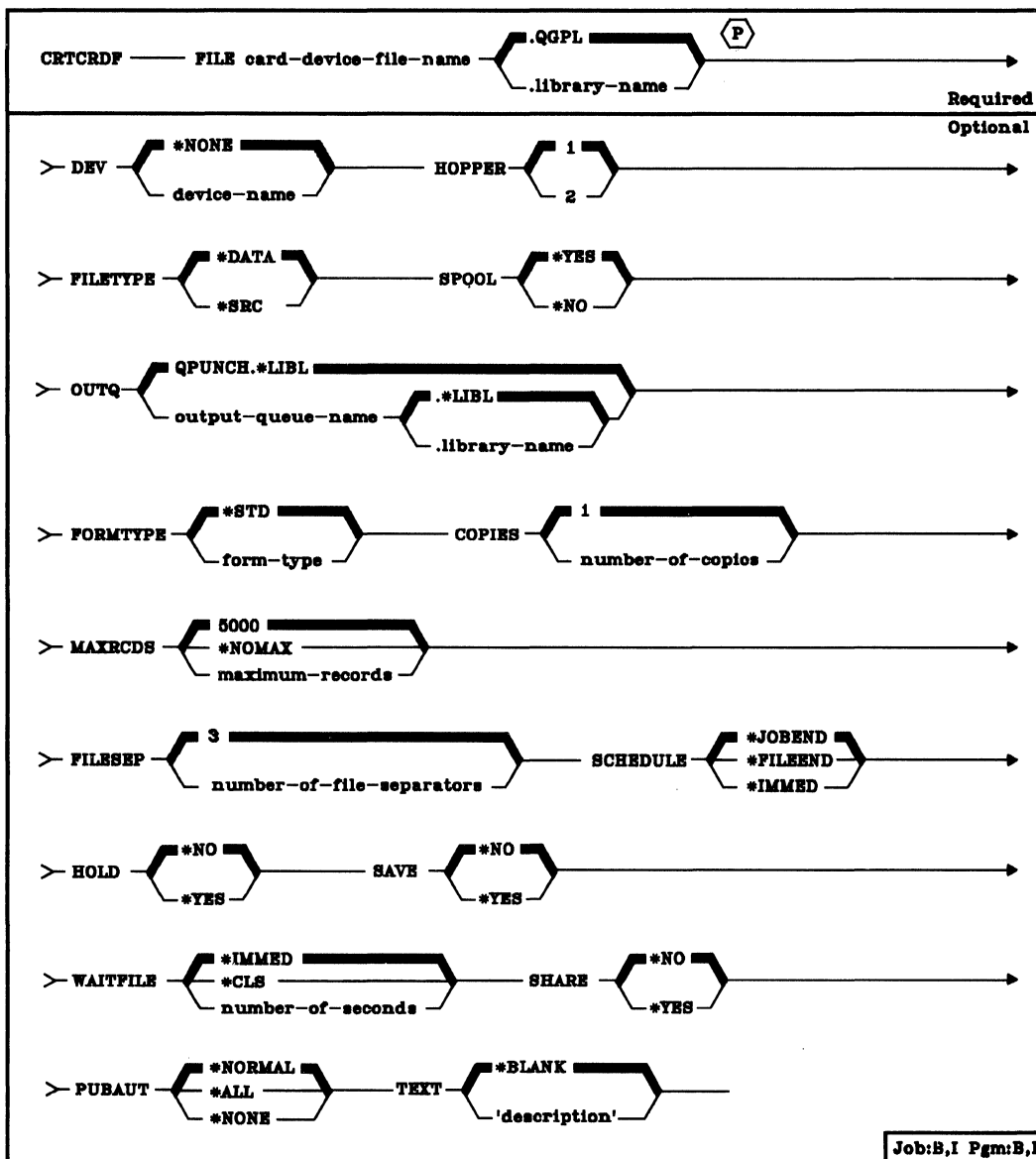
CRTCRDF (Create Card File) Command

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.

CRTCRDF
(Diagram)



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.)

CRTCRDF
DEV

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.

CRTCRDF
FILESEP

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.

**CRTCRDF
PUBAUT**

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.

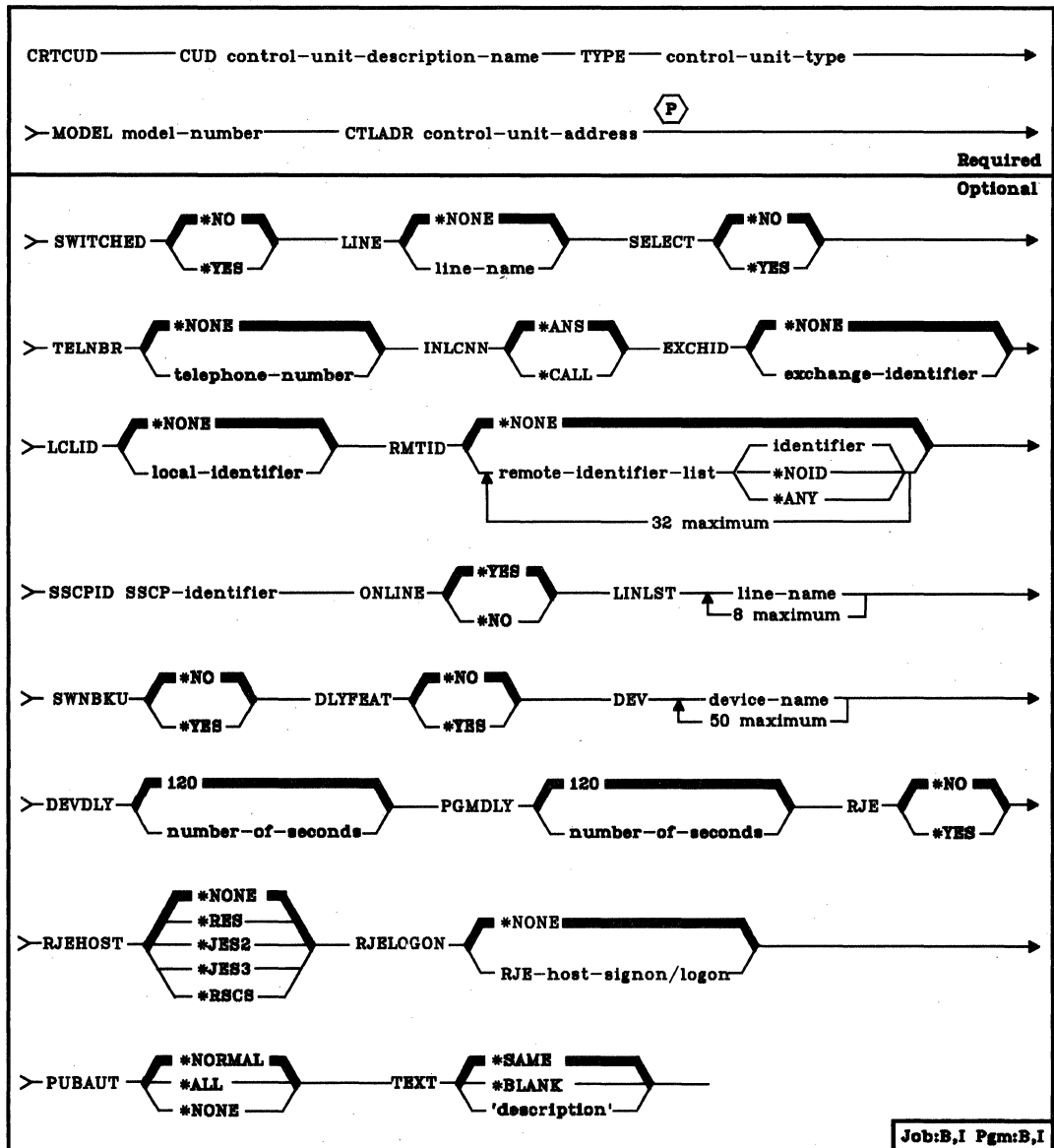
CRTCUD (Create Control Unit Description) Command

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
(Diagram)



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 **CRTDEV** (*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.)

**CRTCUD
SWITCHED**

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.)

***NO:** 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 autocal 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 (autocal 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.

CRTCUD
LINLST

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

¹Maximum 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 CRTDEV commands). Use this parameter only when the associated device descriptions have already been created before this control unit description.

DEV DLY 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.

CRTCUD
RJE

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 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.)

***BLANK:** 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.

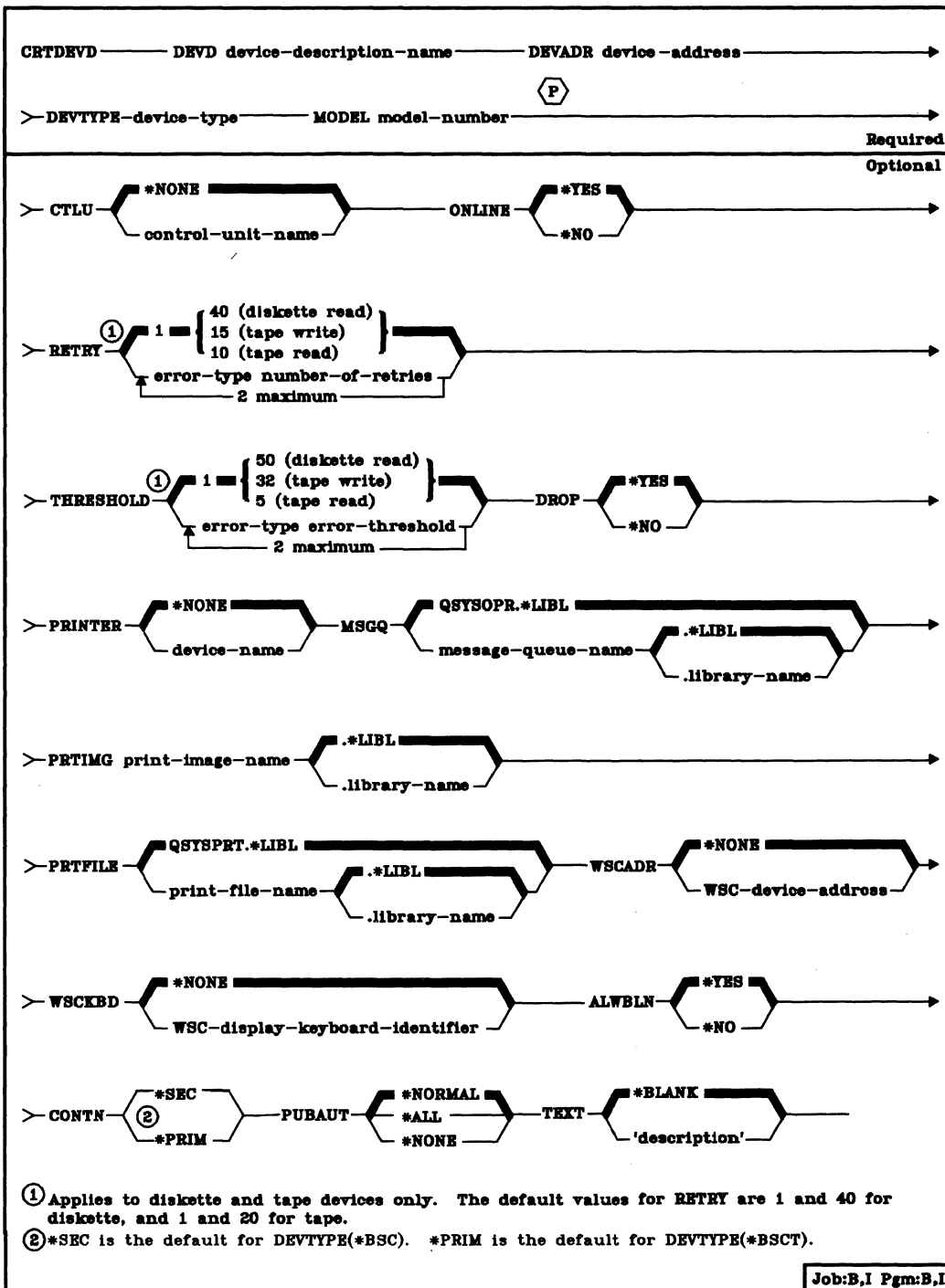
For devices that require a line description and/or control unit description, this command should be used to create each device description after the associated line description and control unit description are created for the line and control unit associated with the device. Also, the device description 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

CRTDEV D
(Diagram)



CRTDEVD
DEVD

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, or 20-23, 60-63
	For RJE devices:		00, or 20-23, 60-63
	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	
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

¹Any 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).

²If 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.

³Any 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).

⁴For 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.

CRTDEVD
DEVTYPE

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)	5291	Display Station
3262	Printer (system)	5292	Color Display 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
5252	Display Station		
5256	Printer (work station)		

**CRTDEV
 MODEL**

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)
5291	Display Station	1 (1920 characters)
5292	Color Display Station	1 (1920 characters)
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:

- 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.
- All 3410 tape units associated with a 3411 tape control unit must have the same model number as that of the control unit.

CRTDEVD
CTLU

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 } (standard (63 bytes/mm) } 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, 5252, 5291, or 5292 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.

***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 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 1 (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.

CRTDEVD
DROP

DROP Parameter: Specifies, for 5251, 5252, 5291, and 5292 display stations 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, 5252, 5291, or 5292 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 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.

QSYSVRT: 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 (xyyyzz), 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

¹Typewriter and data entry with proof feature keyboards only.
²Typewriter keyboard only.

CRTDEV
ALWBLN

ALWBLN Parameter: Allows users to suppress the (software-controlled) blinking cursor. (For 5291 and 5292 display devices, allowing the cursor to blink may distract the operator.)

***YES:** Allows the cursor to blink for the 5251, 5252, 5291, and 5292 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.)

***NORMAL:** 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.)

***BLANK:** No text is to be specified.

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

Examples

CRTDEVD (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.

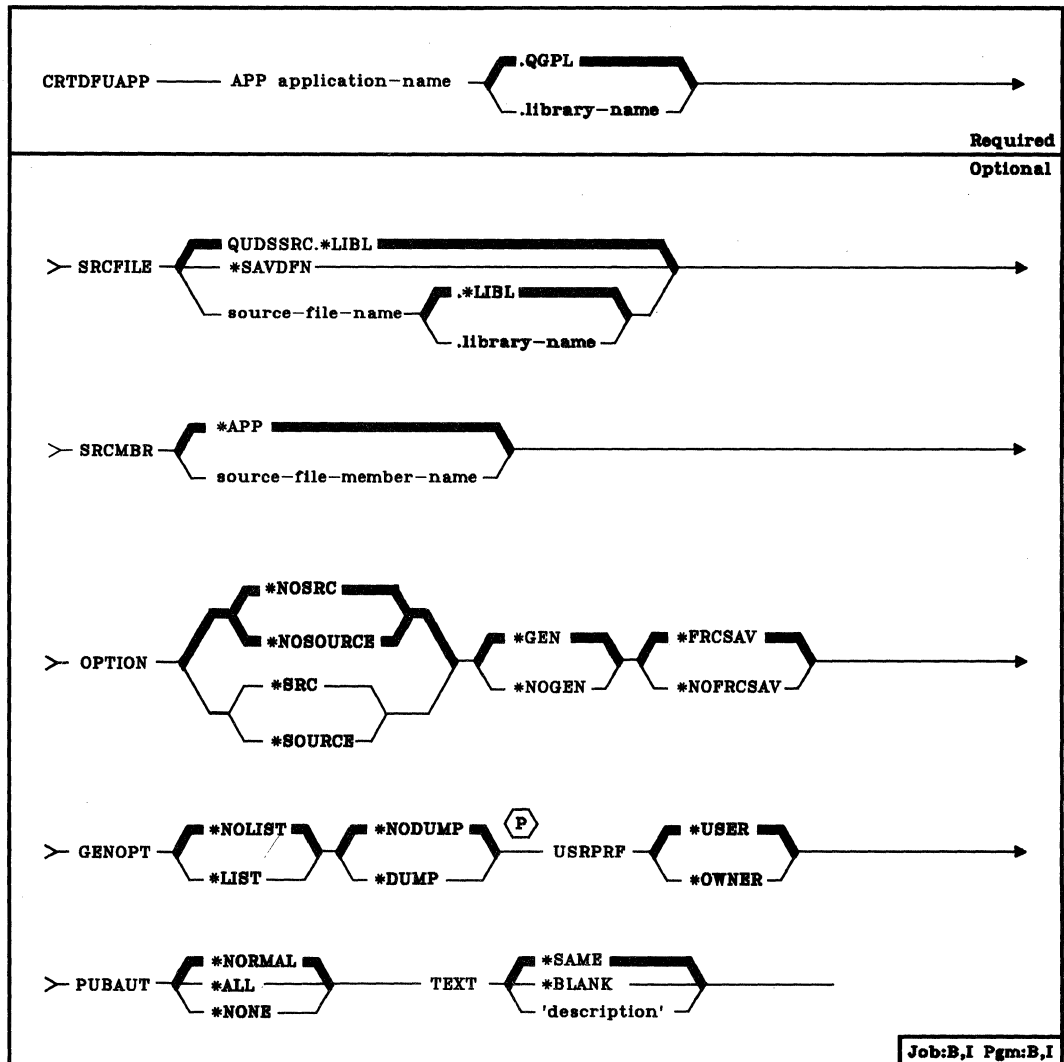
```
CRTDEVD DEVD(PRTR1) DEVADR(000040) DEVTYPE(3203) +  
MODEL(5) PRTIMG(HN)
```

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.

CRTDFUAPP
GENOPT

***FRCSAV:** 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.

***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.

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.)

***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. 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

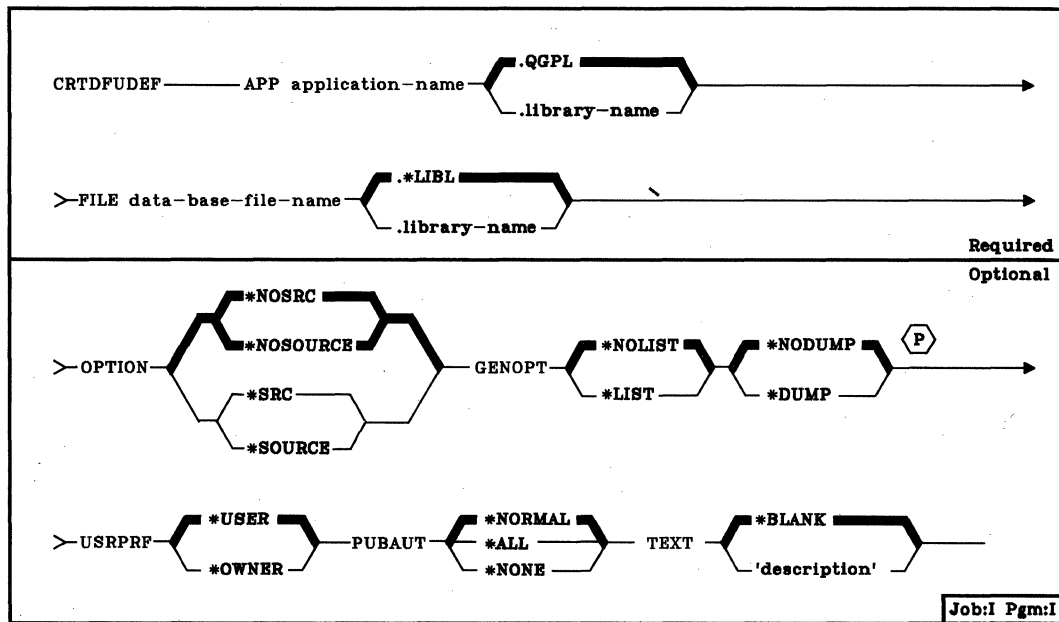
```
CRTDFUAPP APP(TEST1) SRCFILE(FILE1) SRCMBR(TEST2) +  
TEXT('Test application for TEST1')
```

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.

**CRTDFUDEF
TEXT**

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

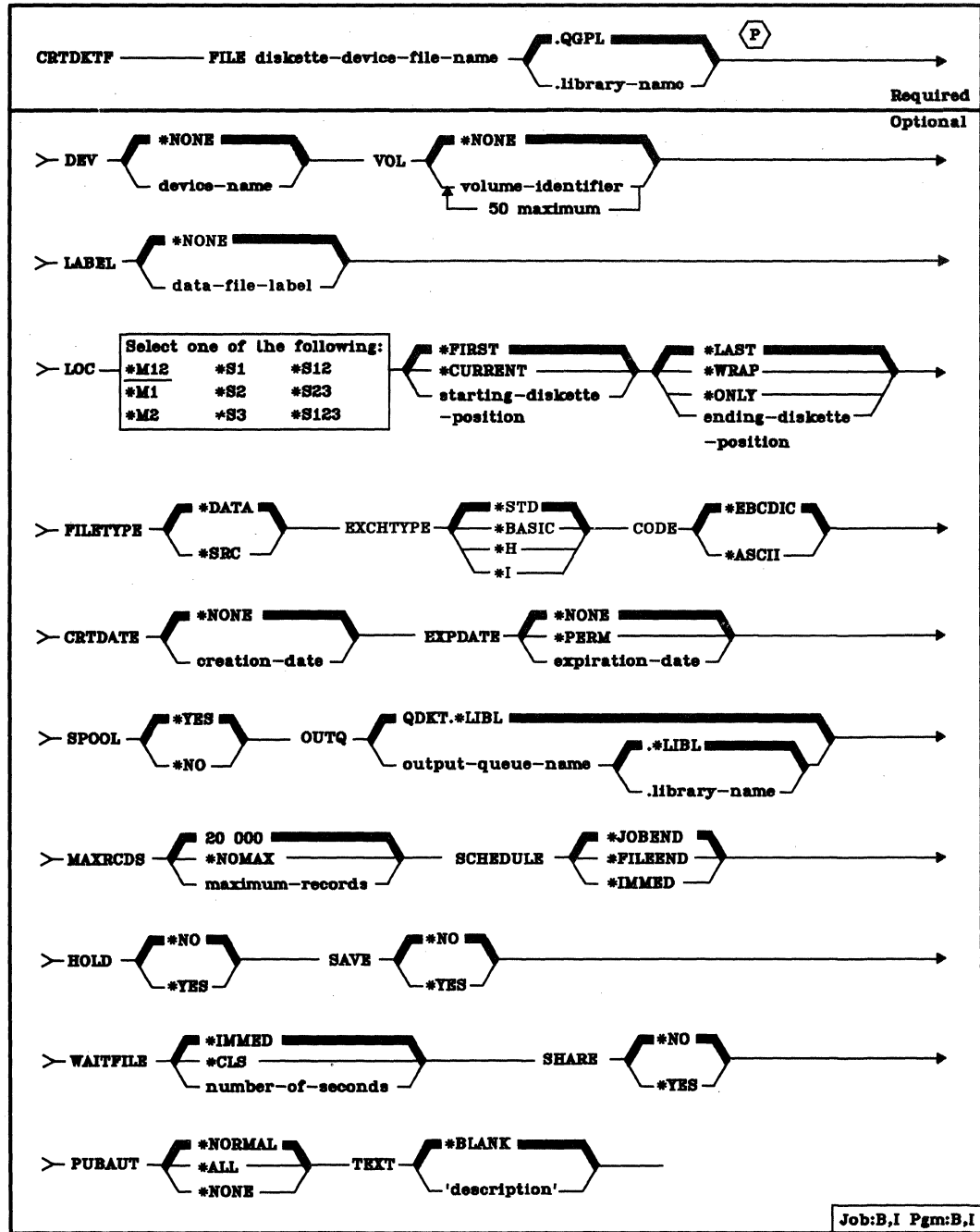
The Create Diskette File (CRTDKTF) command creates a diskette device file. The device file contains the file description, which identifies the device to be used and specifies the spooling requirements; the device file does not contain data. The diskette device file is used to read and write records on 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
(Diagram)



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.)

CRTDKTF
LOC

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*.

CRTDKTF
EXPDATE

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.

CRTDKTF
WAITFILE

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.)

***BLANK:** 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.

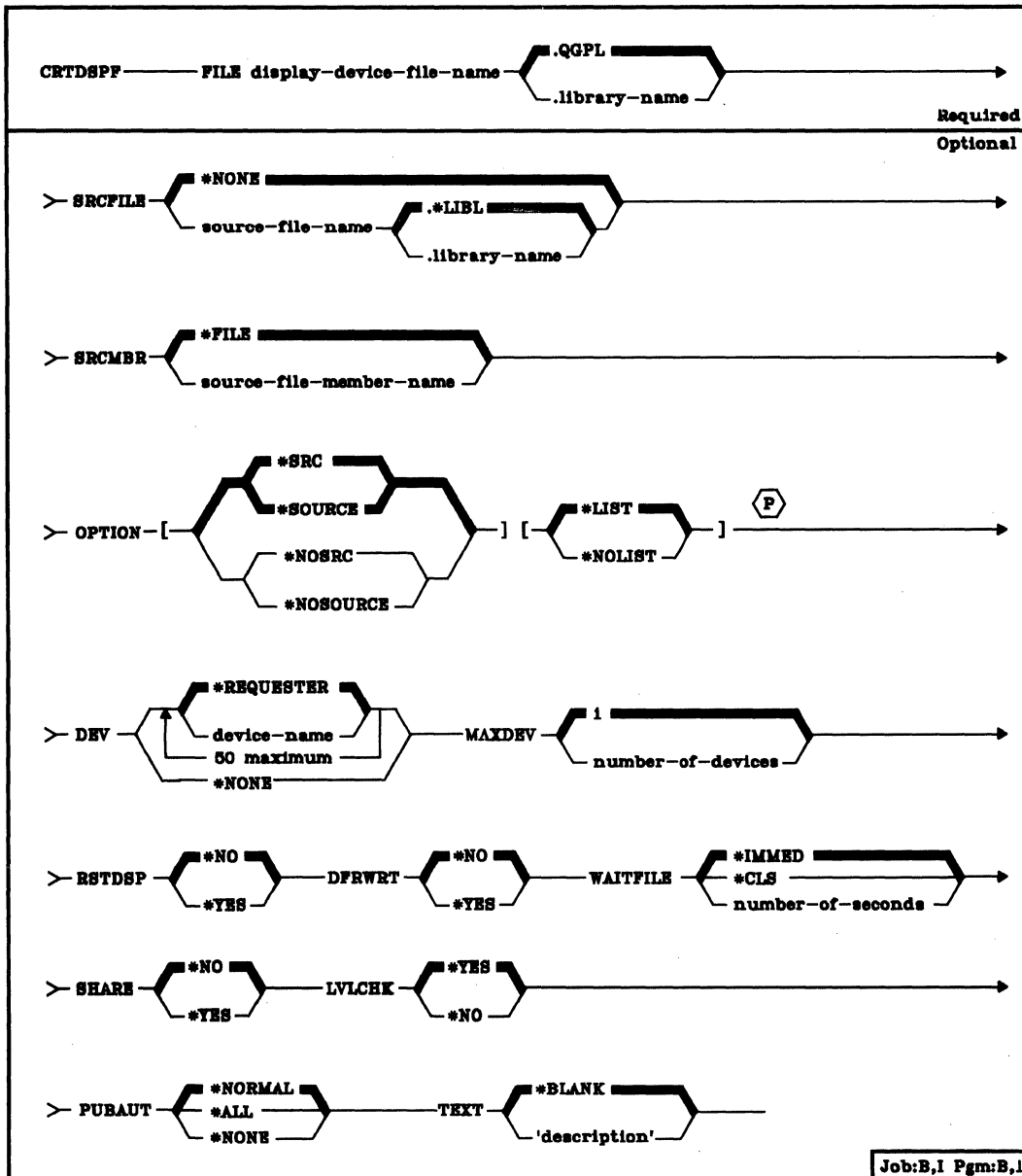
CRTDSPF (Create Display File) Command

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.

CRTDSPF
(Diagram)



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.)

CRTDSPF
SRCFILE

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*.)

***NONE:** 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.

***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 display 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 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 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.

CRTDSPF
RSTDSP

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.

CRTDSPF
PUBAUT

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

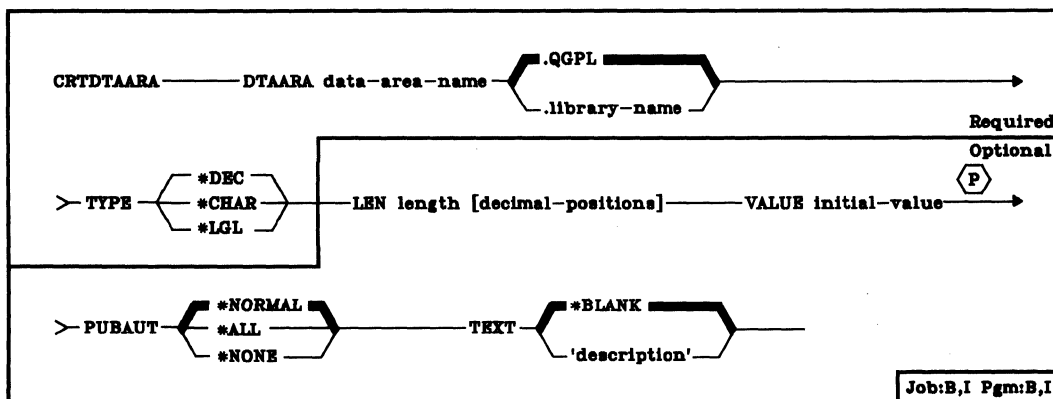
The Create Data Area (CRTDTAARA) command creates a data area and stores it in a specified library. It also specifies the attributes of the data. The data area can optionally be initialized to a specific value.

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.)

CRTDTAARA
TYPE

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.

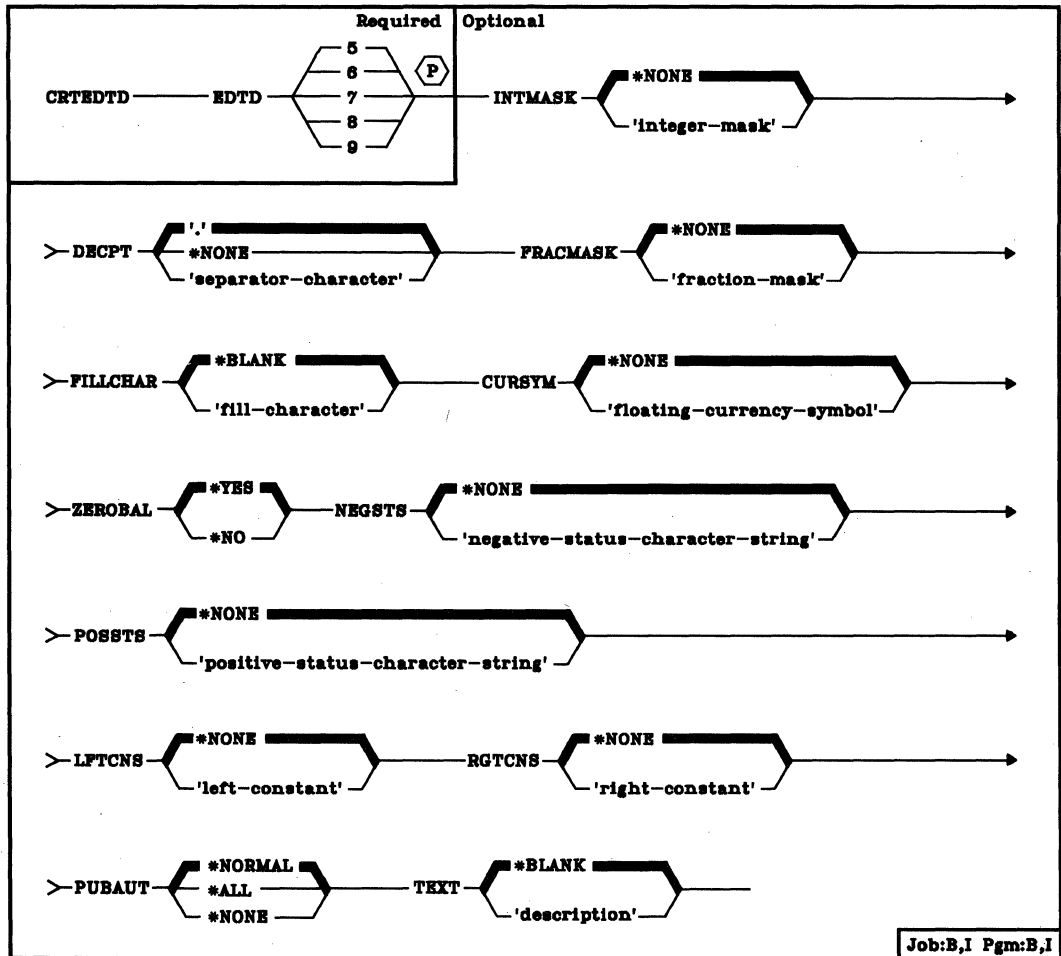
```
CRTDTAARA DTAARA(CUSTOMER) TYPE(*CHAR) LEN(148) +
          TEXT('Customer name area')
```

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.

**CRTEDTD
FRACMASK**

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.)

***NORMAL:** 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.)

***BLANK:** 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:

$$\begin{aligned} &(\text{length of LFTCNS}) + (\text{length of CURSYM}) + \\ &(\text{length of truncated INTMASK}) + \\ &(1 \text{ (or 0 if DECPNT equals *NONE)}) + \\ &(\text{length of truncated FRACMASK}) + \\ &(\text{length of NEGSTS or POSSTS}) + \\ &(\text{length of RGTCNS}) = \text{width of edited result} \end{aligned}$$

- 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:

- FIELD A – Six digits (four integer and two decimal positions) with a value of 001234
- FIELD B – Same as FIELD A but with a negative value (-001234)
- FIELD C – Same as FIELD A but with a zero value (000000)
- DATE – Six digits (0 decimal positions) with a value of 091878

The character `␣` is used to represent blank spaces.

Example 1

```
CRTEDTD EDTD(5) INTMASK('␣␣␣␣␣␣') +
FRACMASK('␣␣␣␣') +
NEGSTS('DB␣␣␣␣') POSSTS('CREDIT') +
LFTCNS('$') RUTCNS('␣**')
```

- FIELD A – Logical mask is '\$␣␣␣0.␣␣ DB␣␣␣␣ ␣**' for a negative value or '\$␣␣␣0.␣␣ CREDIT ␣**' for a positive value
 - Edited result is \$␣␣␣12.34CREDIT␣**
- FIELD B – Same logical mask
 - Edited result is \$␣␣␣12.34DB␣␣␣␣␣**
- FIELD C – Same logical mask
 - Edited result is \$␣␣␣␣␣.00CREDIT␣** or, if ZEROBAL(*NO) had been specified, \$␣␣␣␣␣␣␣CREDIT␣**

CRTEDTD
(Examples)

Example 2

CRTEDTD EDTD(6) INTMASK('###.###') DECPNT(',') +
FRACMASK('###') CURSYM('DM') NEGSTS('-***')

- FIELD A - Logical mask is '###.###,##-***' with floating DM
- Edited result is ###DM12,34###
- FIELD B - Same logical mask
- Edited result is ###DM12,34-***
- FIELD C - Same logical mask
- Edited result is ###DM0,00### or, if ZEROBAL(*NO) had been specified, #####

Example 3

CRTEDTD EDTD(7) INTMASK('0#MONTH##DAY&#YEAR') +
LFTCNS('DATE#IS#')

- DATE - Logical mask is equal to the INTMASK parameter value
- Edited result is DATE#IS##9MONTH18DAY#78YEAR'

Example 4

CRTEDTD EDTD(9) INTMASK('##,##0') DECPNT('.') +
FRACMASK('##') FILLCHAR('*') NEGSTS('#ERROR##')

- FIELD A - Logical mask is '##,##0.#####' or '##,##0.###ERROR##' (Both use the * as the fill character)
- Edited result is ***12.34#####
- FIELD B - Same logical mask
- Edited result is ***12.34#ERROR##
- FIELD C - Same logical mask
- Edited result is *****.00##### or, if ZEROBAL(*NO) had been specified, *****#####

CRTFCT
TEXT

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.

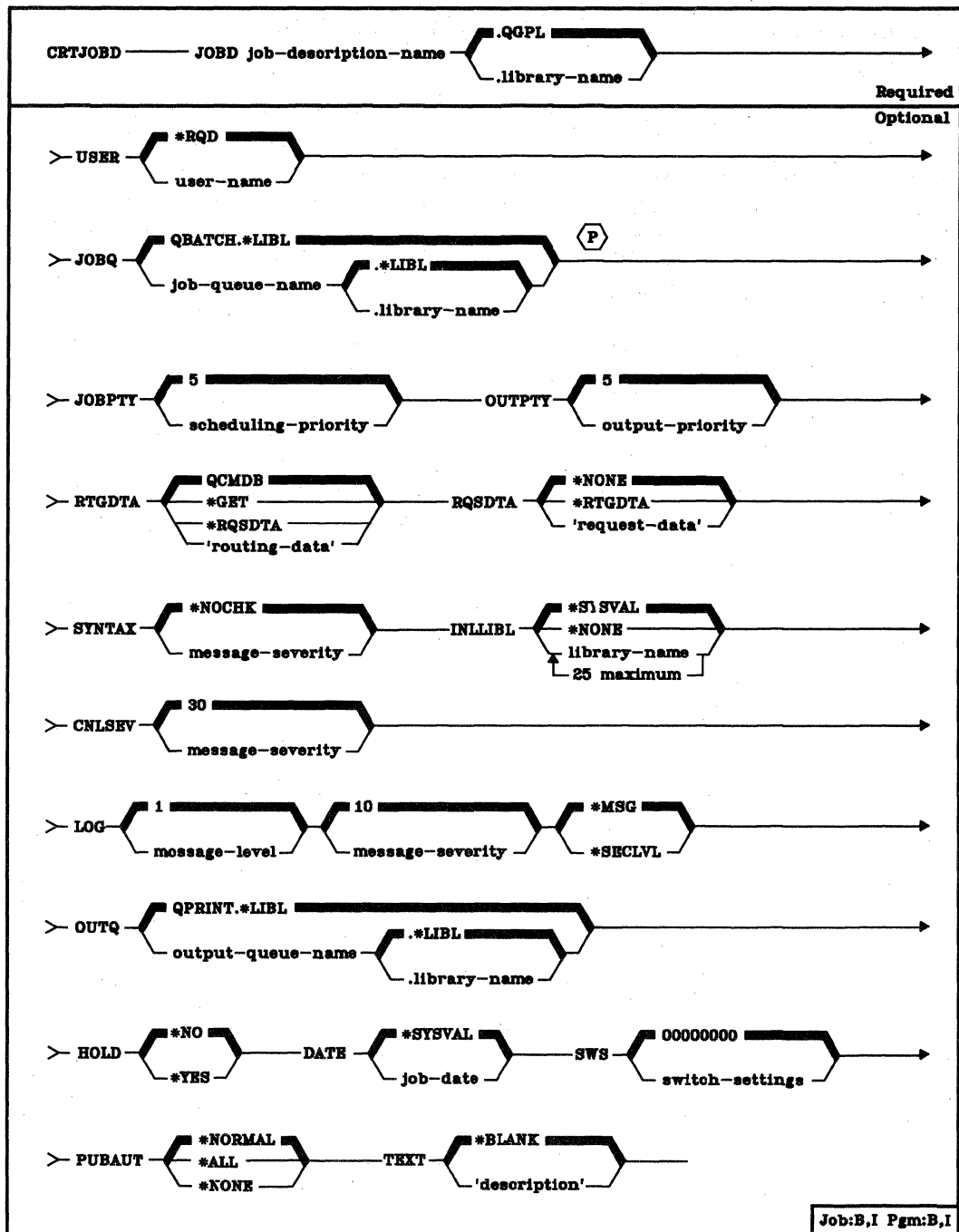
CRTJOB (Create Job Description) Command

The Create Job Description (CRTJOB) 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.

CRTJOBDB
(Diagram)



Job:B,I Pgm:B,I

JOB 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.

CRTJOB
OUTPTY

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.

CRTJOB
HOLD

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

```
CRTJOB JOB(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.

```
CRTJOB JOB(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.

```
CRTJOB JOB(PAYWK) USER(QPGMR) RTGDTA(QCMDB) +
      RQSDTA('CALL PAY025 PARM(WEEKLY UNION)')
```

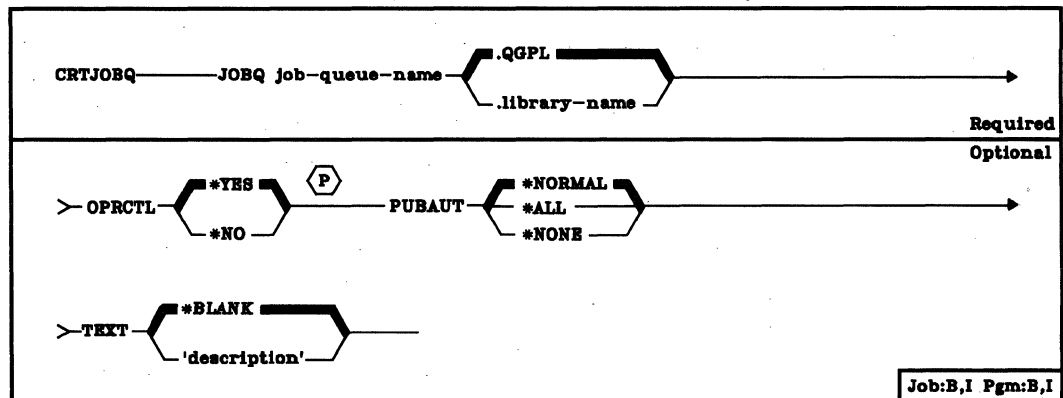
This command creates a job description named PAYWK in the general purpose library (QGPL). Jobs using this job description execute under the IBM-supplied user profile for the programmer, QPGMR. The routing data QCMDB is to be compared with entries in the routing table of the subsystem in which the job 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')
```

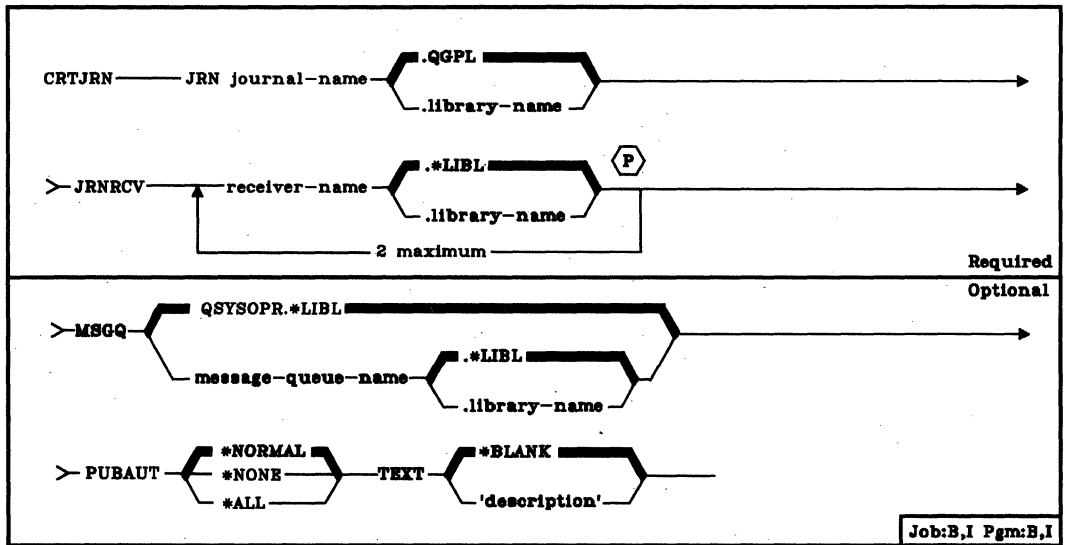
This command creates a job queue named DEPTA and puts it in the QGPL library. Because PUBAUT(*NONE) is specified and OPRCTL(*YES) is assumed, the job queue can be used and controlled only by the user who created the queue and by users with job control rights (*JOBCTL).

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

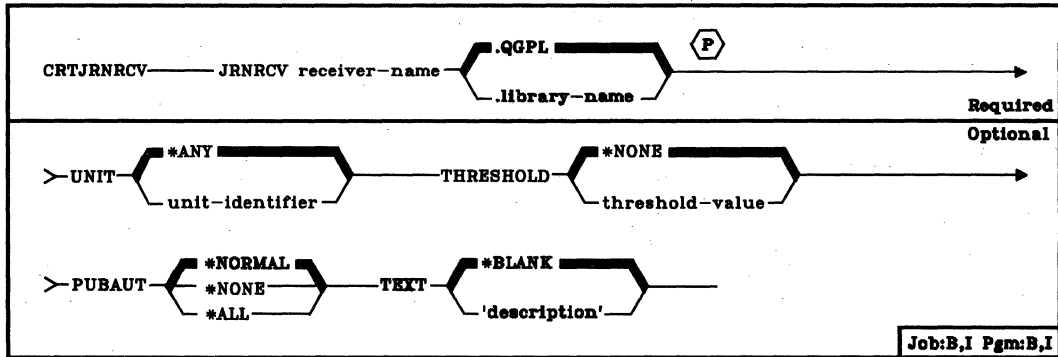
```
CRTJRN JRN(JRNLA.MYLIB) JRNRCV(RCV01.MYLIB RCV01A.MYLIB)
```

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).

CRTJNRVCV
PUBAUT

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

```
CRTJNRVCV JNRVCV(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.

CRTLFL (Create Logical File) Command

The Create Logical File (CRTLFL) command creates a logical file in the System/38 data base. The logical file is created from the file description parameters in this CRTLFL 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.

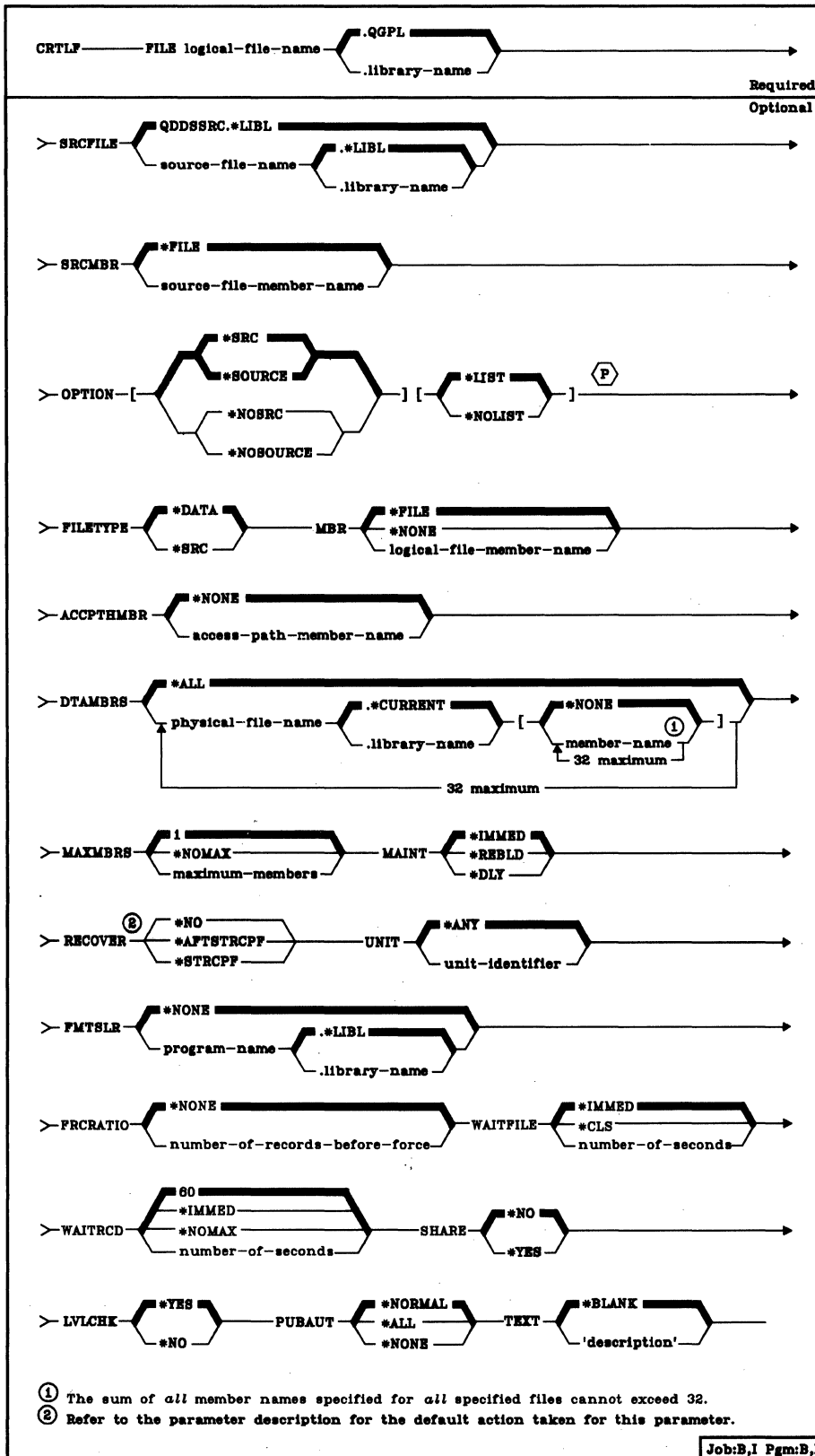
CRTLF

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.

CRTL
(Diagram)



**CRTL
FILE**

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.

CRTL
ACCPHMBR

ACCPHMBR 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 **ACCPH** keyword in the *CPF Reference Manual—DDS*. The **ACCPH** 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 **ACCPH** keyword.

DTAMBR 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 **DTAMBR(*ALL)**; or the member can contain a subset of the total files and members, specified by **DTAMBR(qualified-file-name(s) [member-name(s)])**. **DTAMBR** 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 **ACCPHMBR** 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).

CRTLF
MAXMBRS

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:

DTAMBR((PFA M1))

Single file with multiple members:

DTAMBR((PFA (M1 M2 M3)))

Multiple files with single members and no members:

DTAMBR((PFA M1) (PFB M4) (PFE.NONE))

Multiple files with multiple members:

DTAMBR((PFA (M1 M3 M4)) (PFB (M1 M2 M4)))

Multiple files with the same name in different libraries:

DTAMBR(PFA.LIBX M1) (PFA.LIBY (M1 M2)))

Multiple files with the same name in the same library:

DTAMBR((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 DTAMBR 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.

**CRTL
RECOVER**

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.

**CRTL
WAITFILE**

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

```
CRTL 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 CRTL command to complete successfully, the user must have object management authority for all specified based-on physical files.)

CRTL
(Considerations)

```
CRTL 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(*ACCPH) 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.

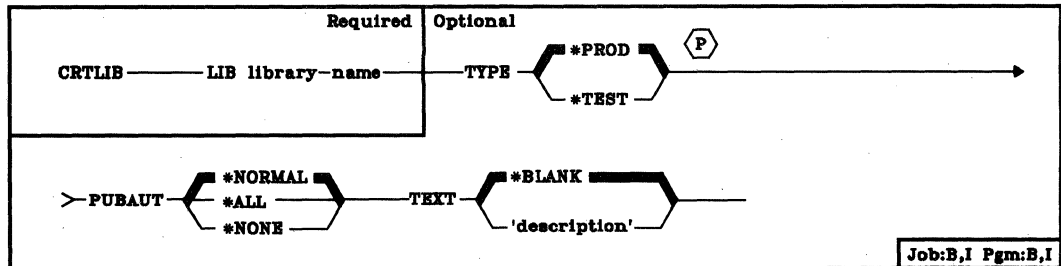
For example, assume that two record formats (FMT1 and FMT2) in this logical file are based on one physical file (PFA) and FMT1 is specified before FMT2 in the DDS. If the logical file member being added will use only FMT2 from member 2 (M2) of the physical file, the following DTAMBRS parameter would be specified: DTAMBRS((PFA)(PFA M2))

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.

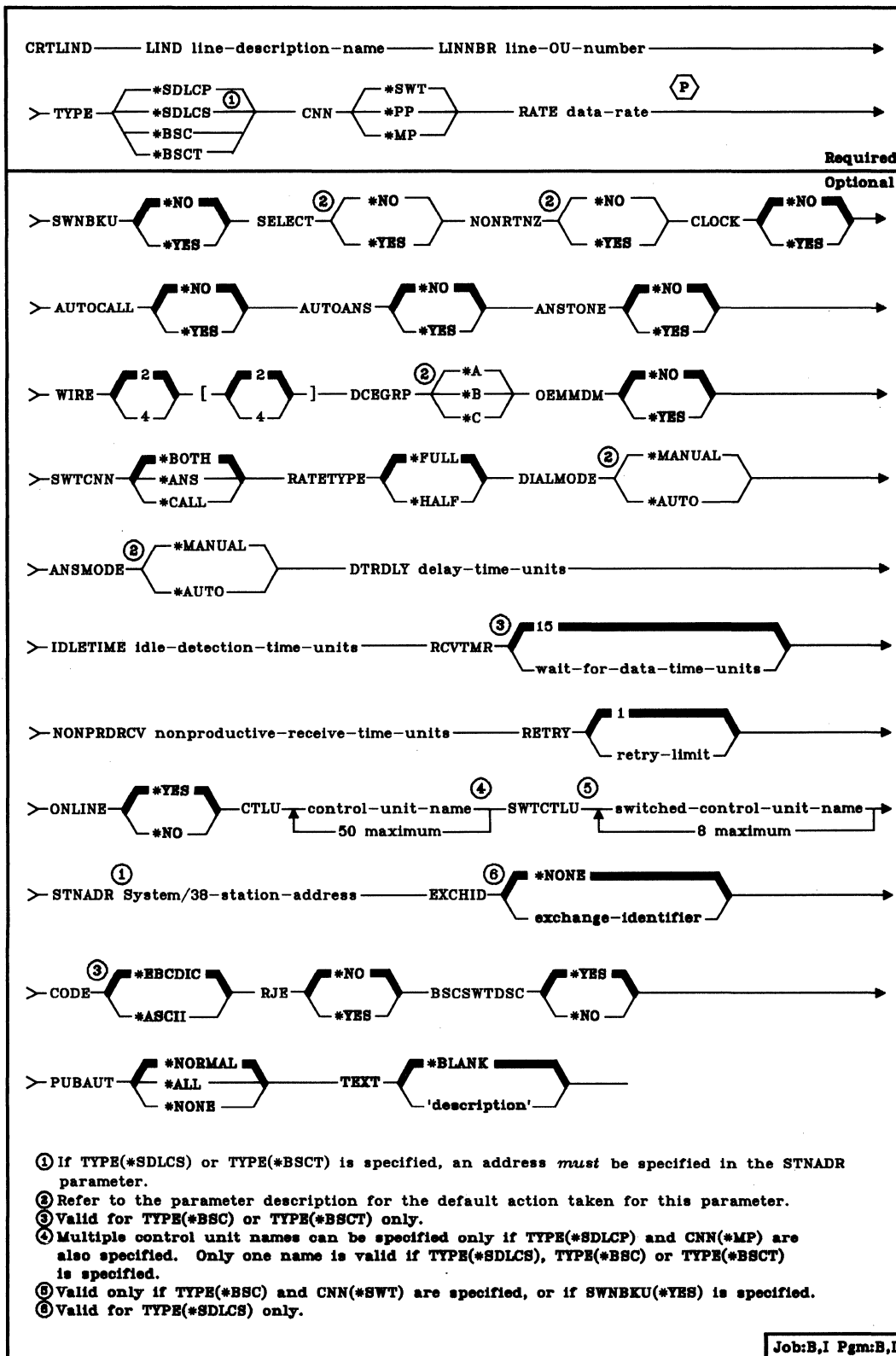
One CRTLIND command must be entered for each communications line on the system. Up to 10 line descriptions can be created for each line number. Only one line description and its associated network can be varied on at one time. This enables you to change the line description or communications protocol 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.

CRTLIND
(Diagram)



CRTLIND
(Chart)

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
NONPRDCV	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

¹Dependent on OEMMDM parameter.

²If TYPE(*SDLCP) or TYPE(*SDLCS) is specified, NONRTNZ(*YES) must be specified (default).

³Dependent on CNN(*SWT) parameter.

⁴Dependent on AUTOANS(*YES) parameter.

⁵Dependent on SWNBKU(*YES) parameter.

⁶SELECT(*YES) must be specified for RATETYPE(*HALF) selection.

⁷Dependent on AUTOCALL(*YES) parameter.

⁸Must be a valid EBCDIC value or a translatable ASCII value.

⁹CODE(*ASCII) and RJE(*YES) are mutually exclusive.

¹⁰At 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.

*SDLC: 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.

**CRTLIND
CNN**

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)
AUTOCALL	SWNBKU	SWNBKU
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

¹Valid only if SWNBKU(*YES) is specified.
²If 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 autocal feature is not installed.

***YES:** The autocal feature is installed.

CRTLIND
AUTOANS

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.

CRTLIND
DIALMODE

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

¹If 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.

CRTLIND
RETRY

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 '011111'B)	7
CPU buffer overflow (on input)	7
Idle state detected	7
Nonproductive receive timeout	7
Data overrun (receive)	7
Data underrun (transmit)	7

**CRTLIND
ONLINE**

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.

***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.

*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 (RJE). RJE(*YES) is not valid for TYPE(*BSCT).

*NO: This line description is not to be used by RJE.

*YES: This line description is to be used by RJE.

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.

*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.

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.

*NORMAL: 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.

CRTLIND
TEXT

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.

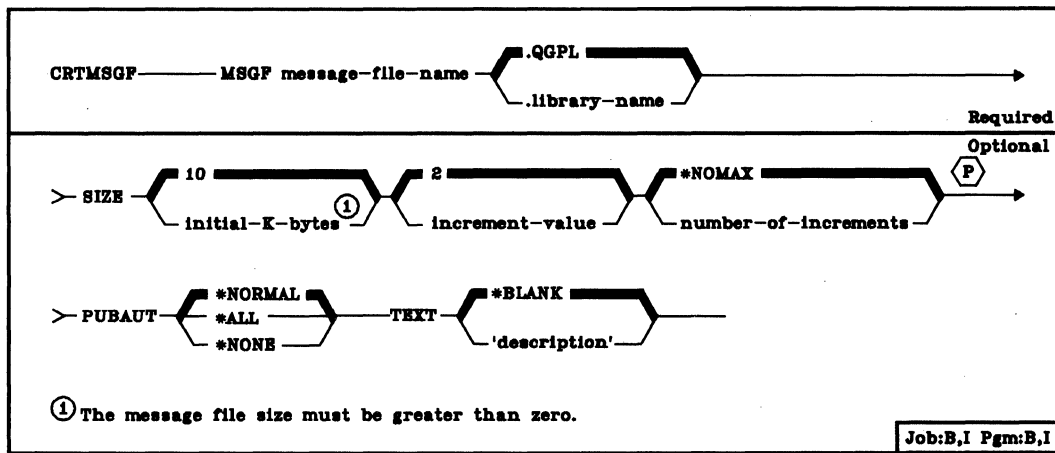
```
CRTLIND LIND(MIL__MAD) LINNBR(21) TYPE(*SDLCP) +  
      CNN(*MP) RATE(4800) WIRE(4) +  
      DTRDLY(1) IDLETIME(15) +  
      NONPRDRCV(2) TEXT('MP line between +  
      Chicago, Milwaukee, & Madison')
```

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

CRTMSGF

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 QRPMSG (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).

CRTMSGF
PUBAUT

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

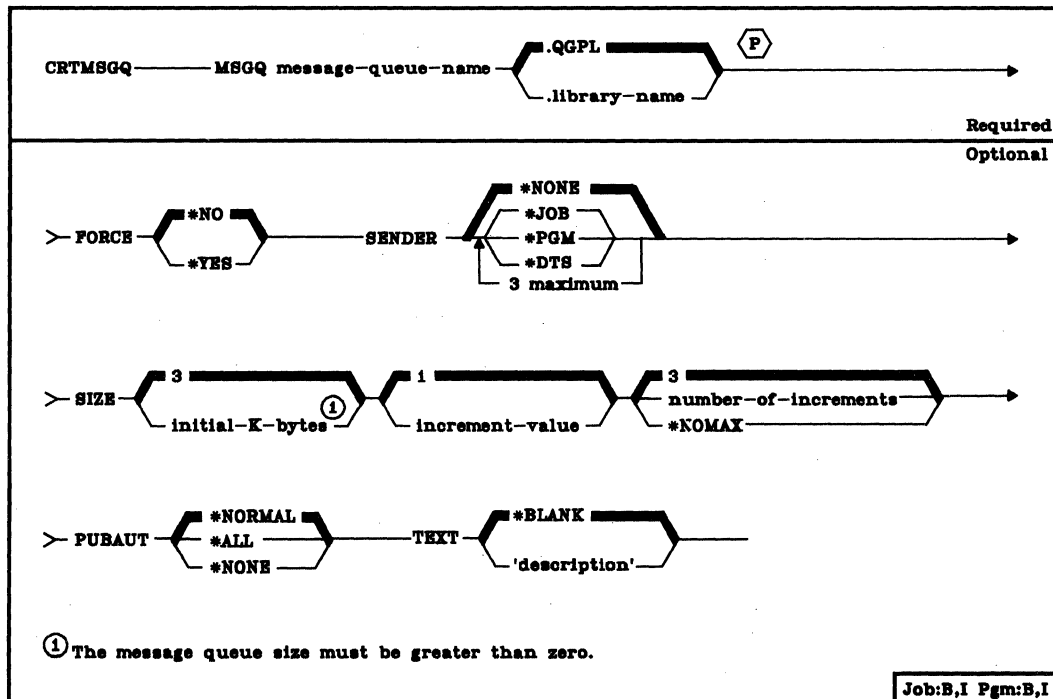
```
CRTMSGF MSGF(INVMSG.S.INVLIB) +  
TEXT('Inventory Application Messages')
```

This command creates a message file named INVMSG.S 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

CRTMSGQ

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.

CRTMSGQ
SENDER

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)
```

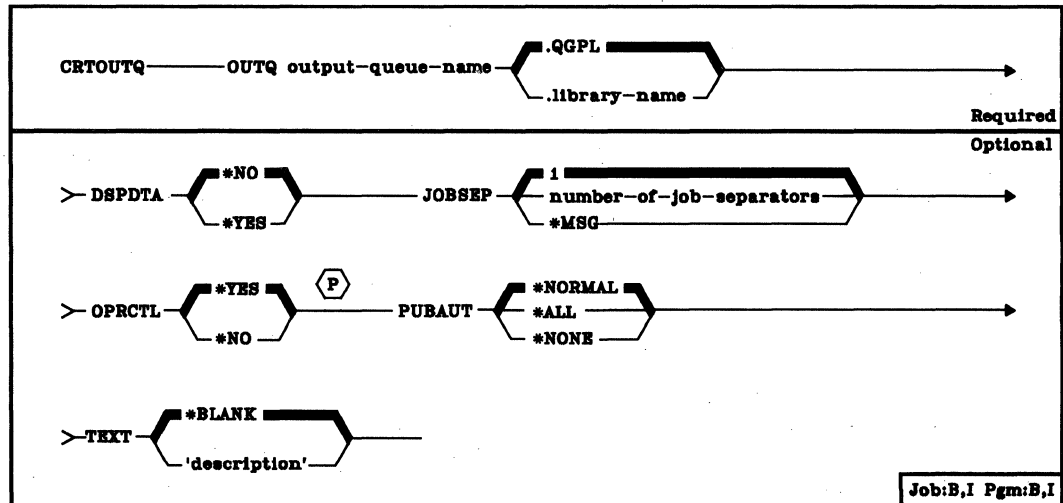
This command creates a message queue named MYQ and stores it in the general purpose library, QGPL, by default. All users are authorized to send messages to the queue and to read its description. Each message sent will be identified by the name of the job sending the message.

```
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.

***NO:** 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.

CRTOUTQ
TEXT

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.

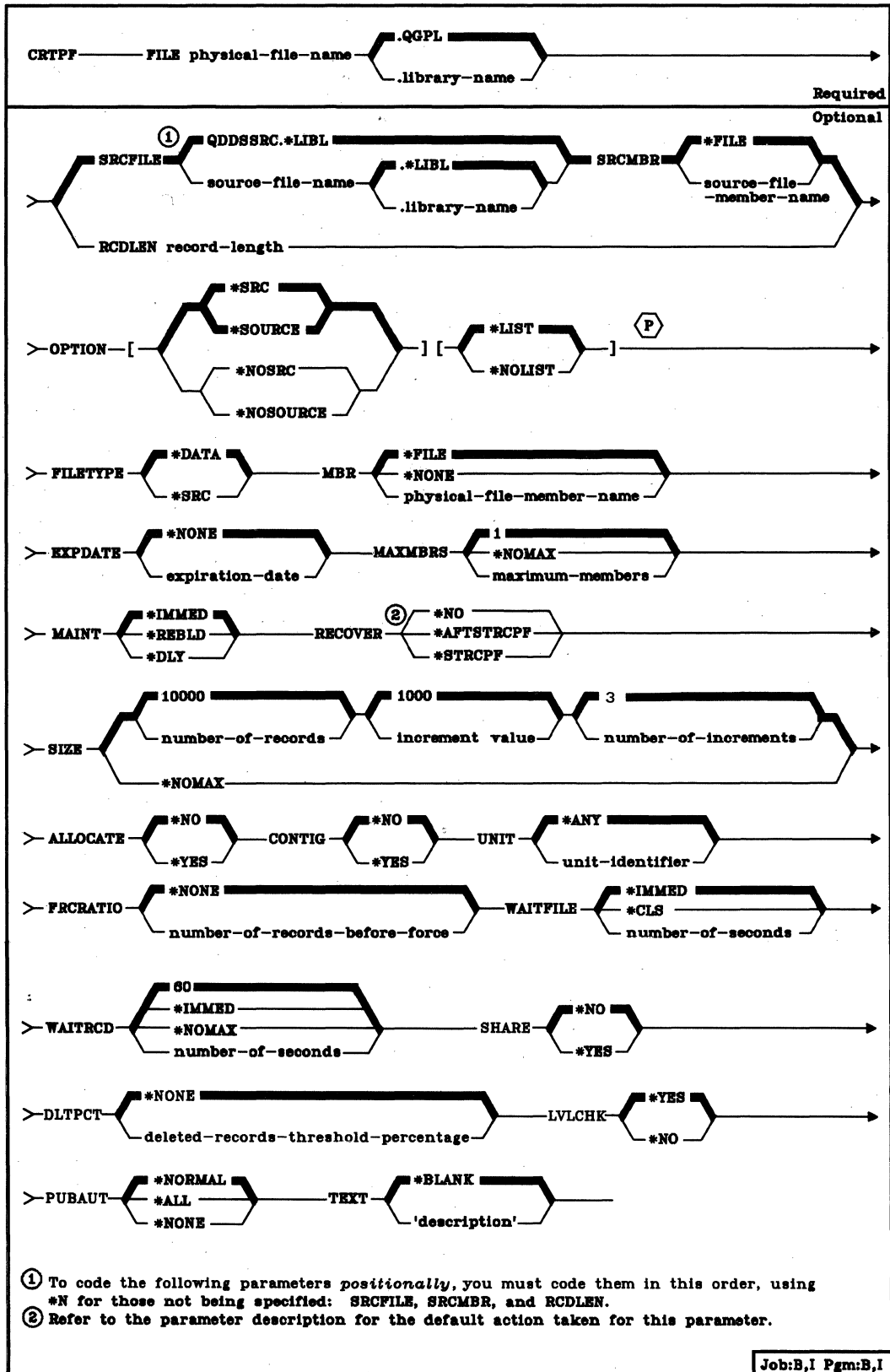
Data is stored and referenced as fields within records, and the records are physically stored in the order in which they are written to the file (arrival sequence). However, the records can be logically processed in any order (arrival sequence or keyed sequence); the processing order is established by 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.

CRTPF
(Diagram)



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.

CRTPF
RCDLEN

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. ***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 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.

10000: 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.

CRTPF
ALLOCATE

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.

CRTPF
FRCRATIO

***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
(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.

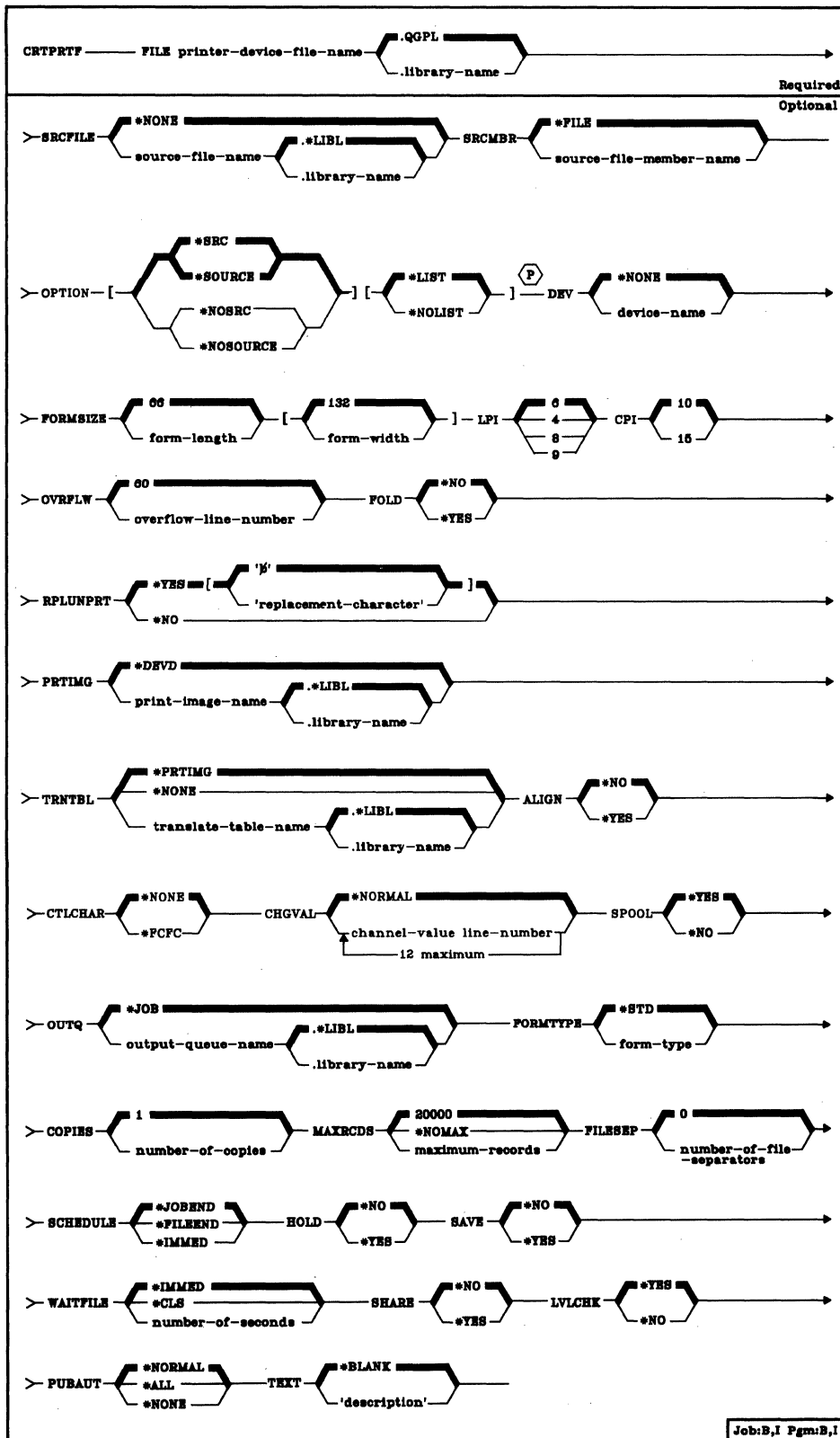
CRTPRTF (Create Printer File) Command

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
(Diagram)



**CRTPRTF
FILE**

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.

Printer	Lines per Page			
	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.

CRTPRTF
LPI

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 ␣).

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.

**CRTPRTF
PRTIMG**

- 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.

'b': 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.)

***DEV D:** 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 alphanumeric 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.

**CRTPRTF
FILESEP**

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.

CRTPRTF
PUBAUT

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.

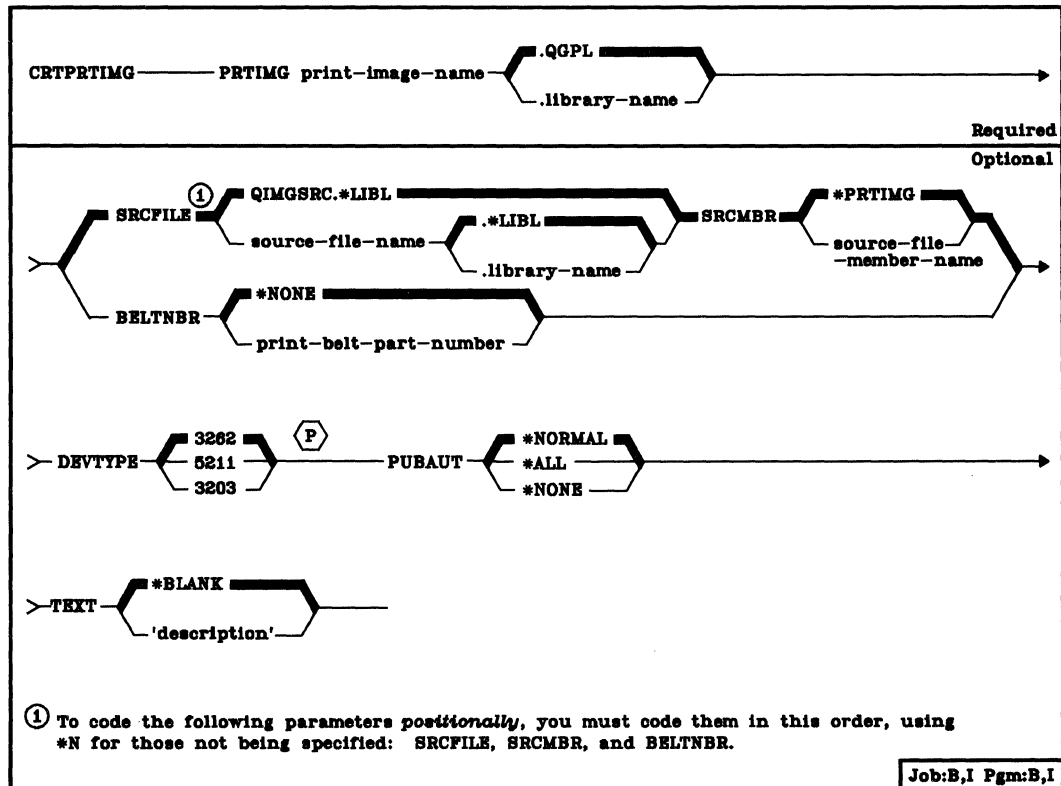
The printer uses standard forms for that installation that are 66 lines long and 132 print positions wide. It prints 6 lines per inch and overflows to a new page after line 60 is printed. The print image specified in the device description is used. Output is spooled to the output queue specified for 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

CRTPRTIMG

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.



CRTPRTIMG
PRTIMG

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

```
CRTPRIMG 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

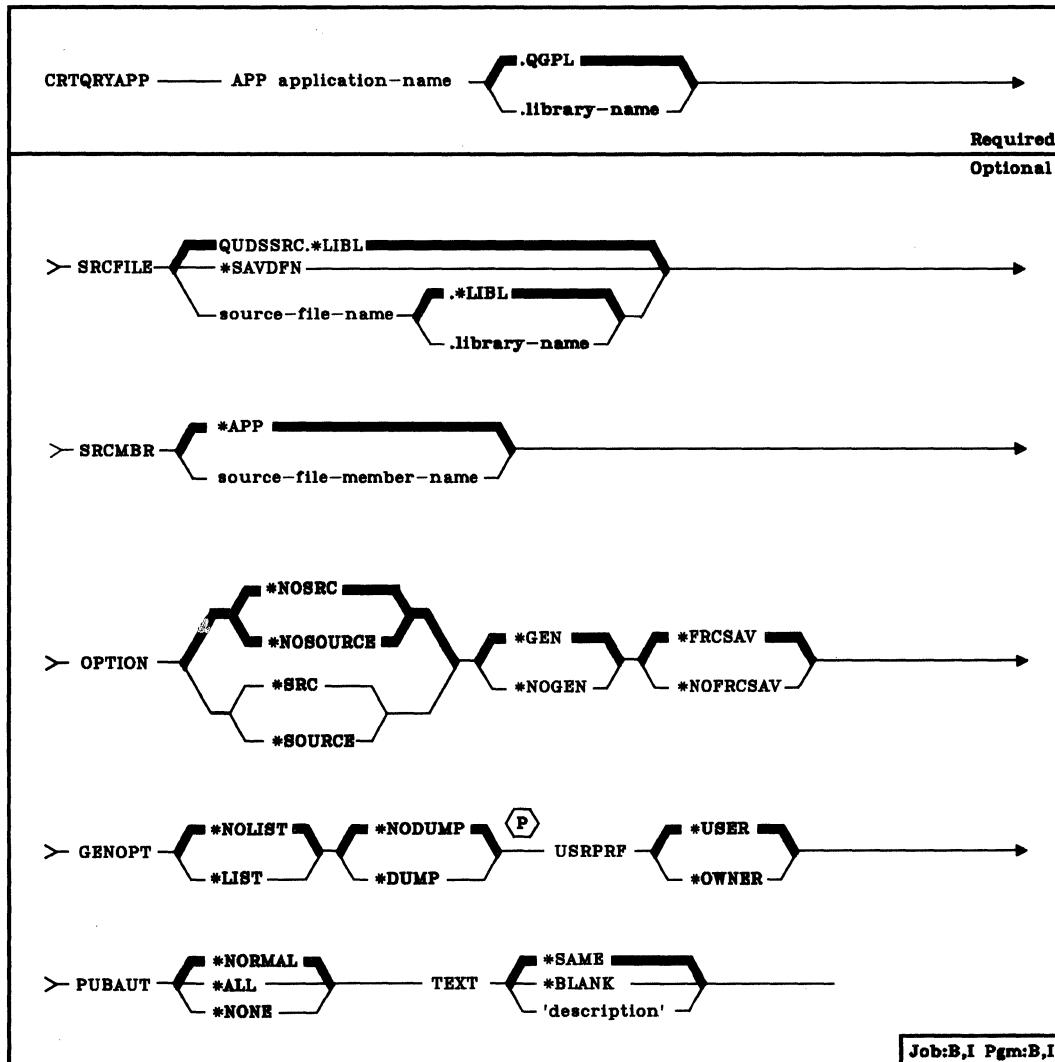
Note that, in the hex format, the total number of characters entered in the source input records is always double that of the character count (identical to that specified in *Size of Character Set*) for the character format. It takes two source records in the HEX format for each source record in the CHAR format. More information about print image source records is contained in the *CPF Programmer's Guide*.

CRTQRYAPP (Create Query Application) Command

CRTQRYAPP

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

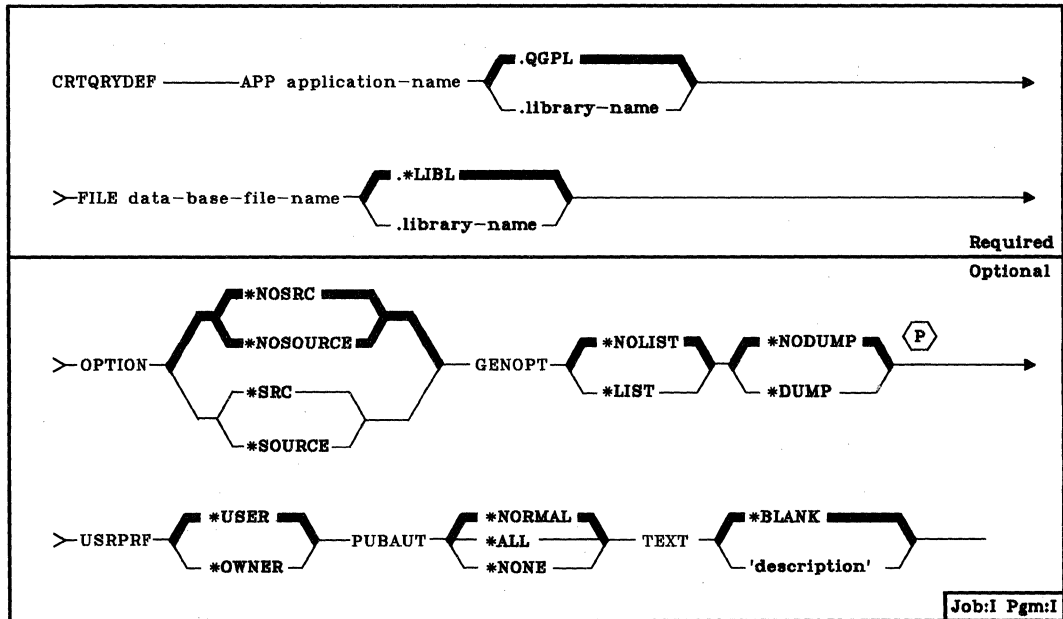
```
CRTQRYAPP APP(TEST1) SRCFILE(FILE1) SRCMBR(TEST2) +  
TEXT('Test application for TEST1')
```

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.

CRTQRYDEF
PUBAUT

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

```
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

CRTRPGPGM

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 alphanumeric characters, the first of which must be alphabetic. The length of the names cannot exceed 10 characters.

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.

CRTRPGPGM
OPTION

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**
(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 QRPGRSRC. 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.

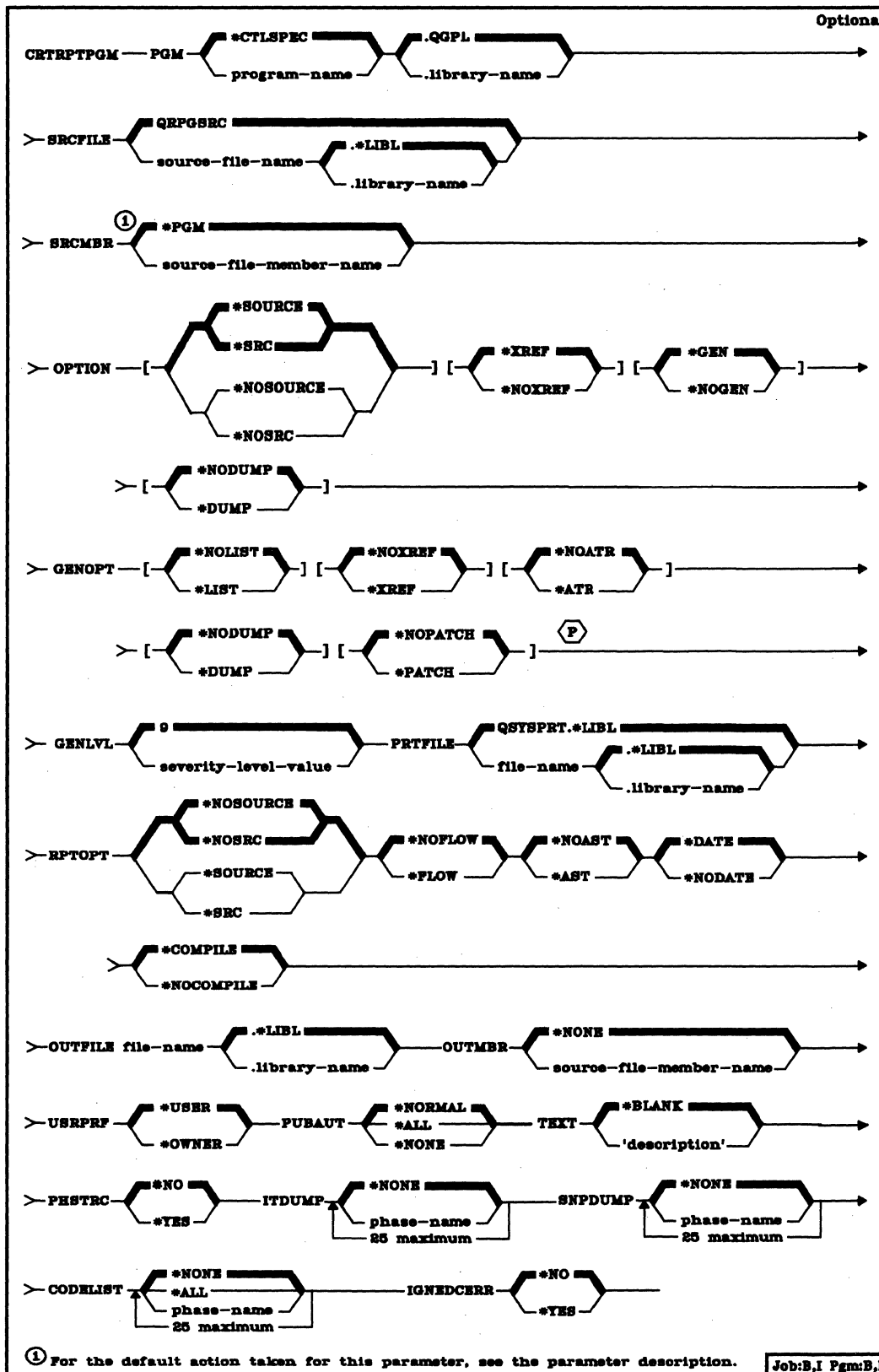
CRTRPTPGM (Create Auto Report Program) Command

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.

CRRTPGM
(Diagram)



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 RPJOB.

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.

CRTTRPTGM
PRTFILE

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.

*NONE: 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).

*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**CRTRPTPGM**
(Example)

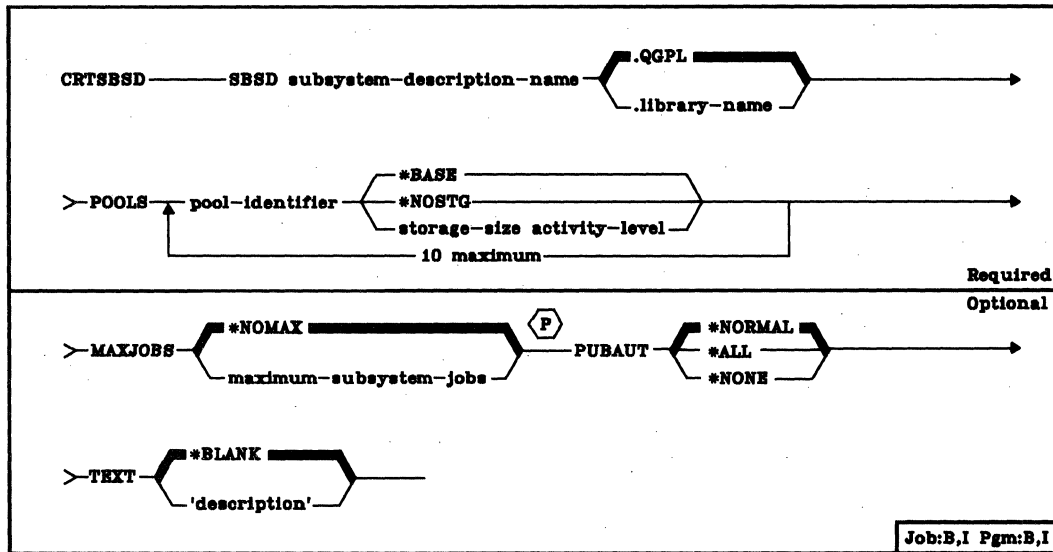
```
CRTRPTPGM PGM(ARINQ5.JONES) GENLVL(29) TEXT('Accounts Receivable +  
Inquiry, Branch 5')
```

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 QRPGRSRC. 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.

*NOMAX: 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

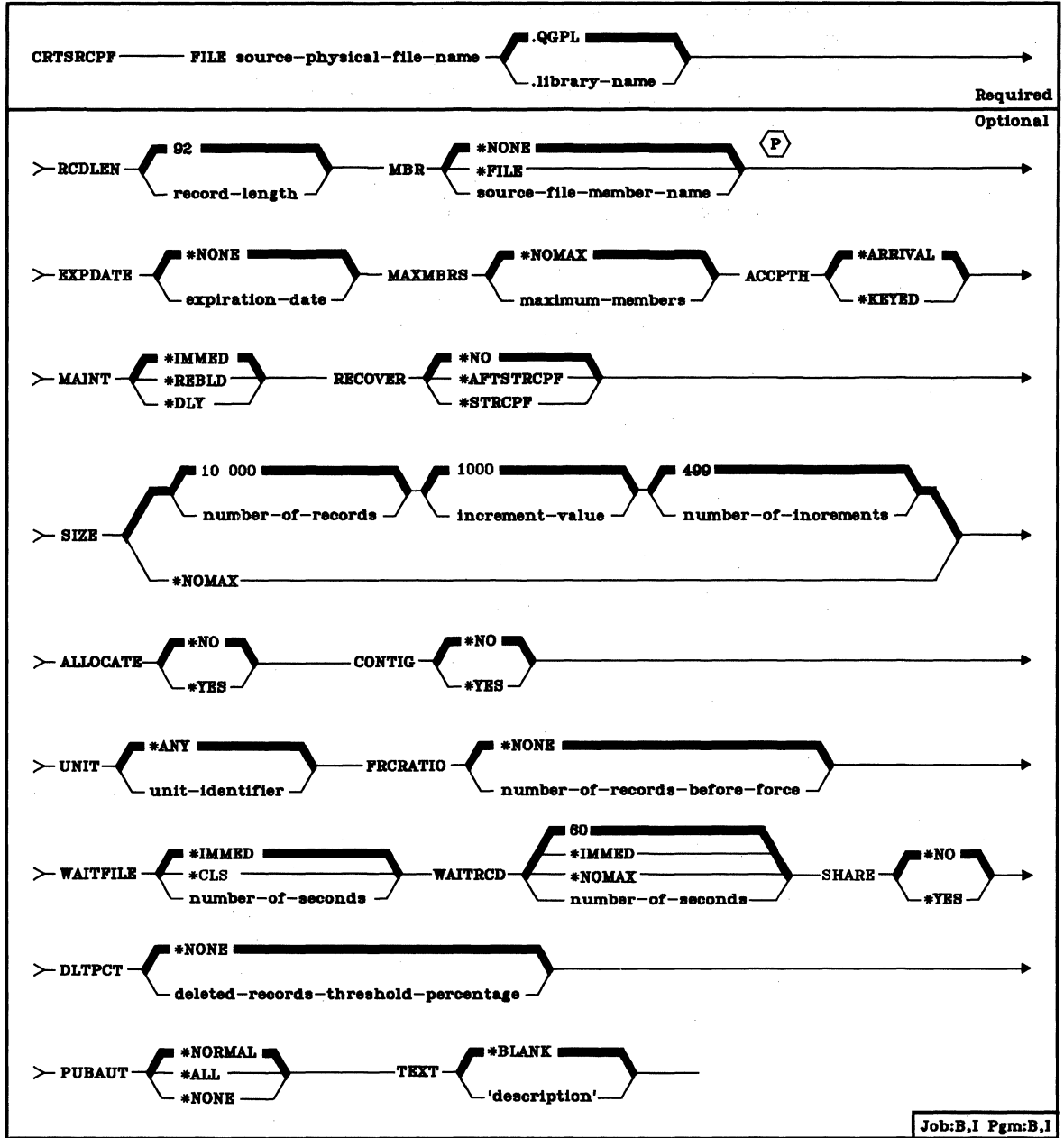
The Create Source Physical File (CRTSRCPF) command creates a named source physical file in the data base. A source file is created from the file description parameters in the CRTSRCPF command; it is used to store source records to be used as input to IBM-supplied source processors, such as the data description 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
(Diagram)



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.

CRTSRCPF
EXPDATE

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.

ACCPH 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.

**CRTSRCPF
SIZE**

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.

**CRTSRCPF
FRCRATIO**

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.)

***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 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.

CRTSRCPF
PUBAUT

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.

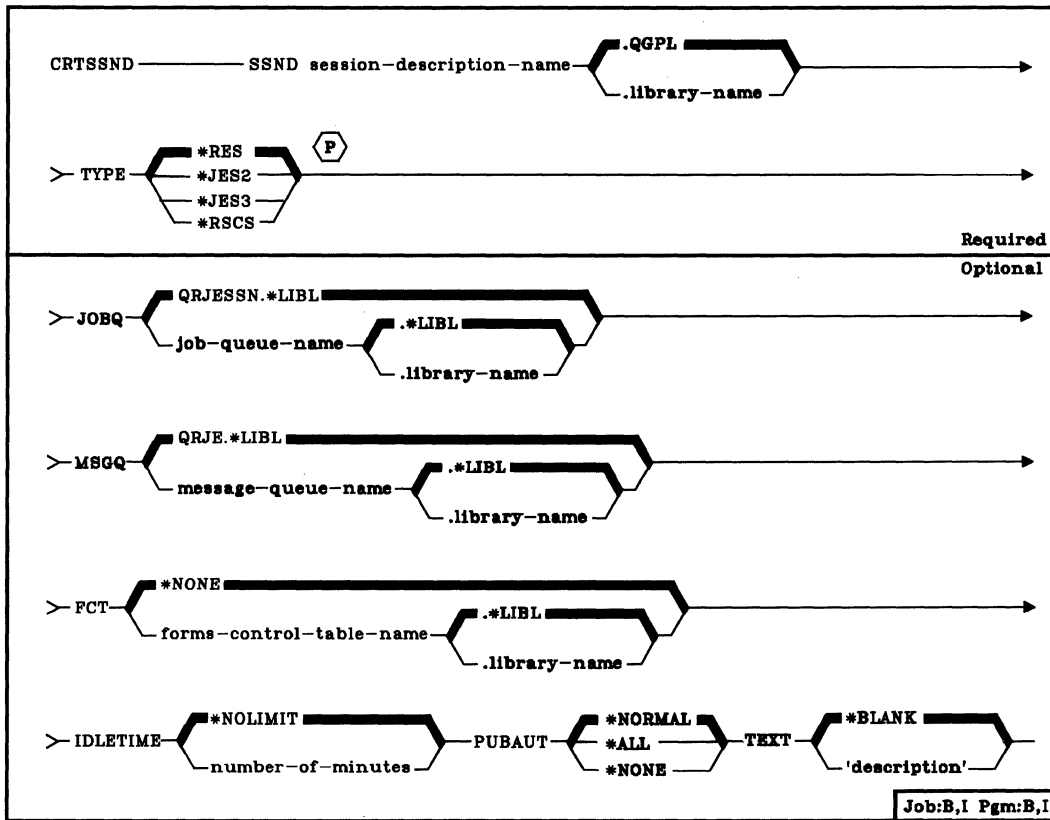
CRTSSND (Create Session Description) Command

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 (ADDRJEWTR), 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 RJE 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 RJE job queue on which all the RJE session jobs are to be started.

QRJESSN.*LIBL: The RJE 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 RJE 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 RJE message queue in which all the RJE 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 RJE jobs. In addition, this message queue serves as a job log for all RJE jobs in the active RJE session. The message queue can be displayed by issuing the STRRJECSL command.

QRJE.*LIBL: The RJE 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 RJE 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.

CRTSSND
PUBAUT

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

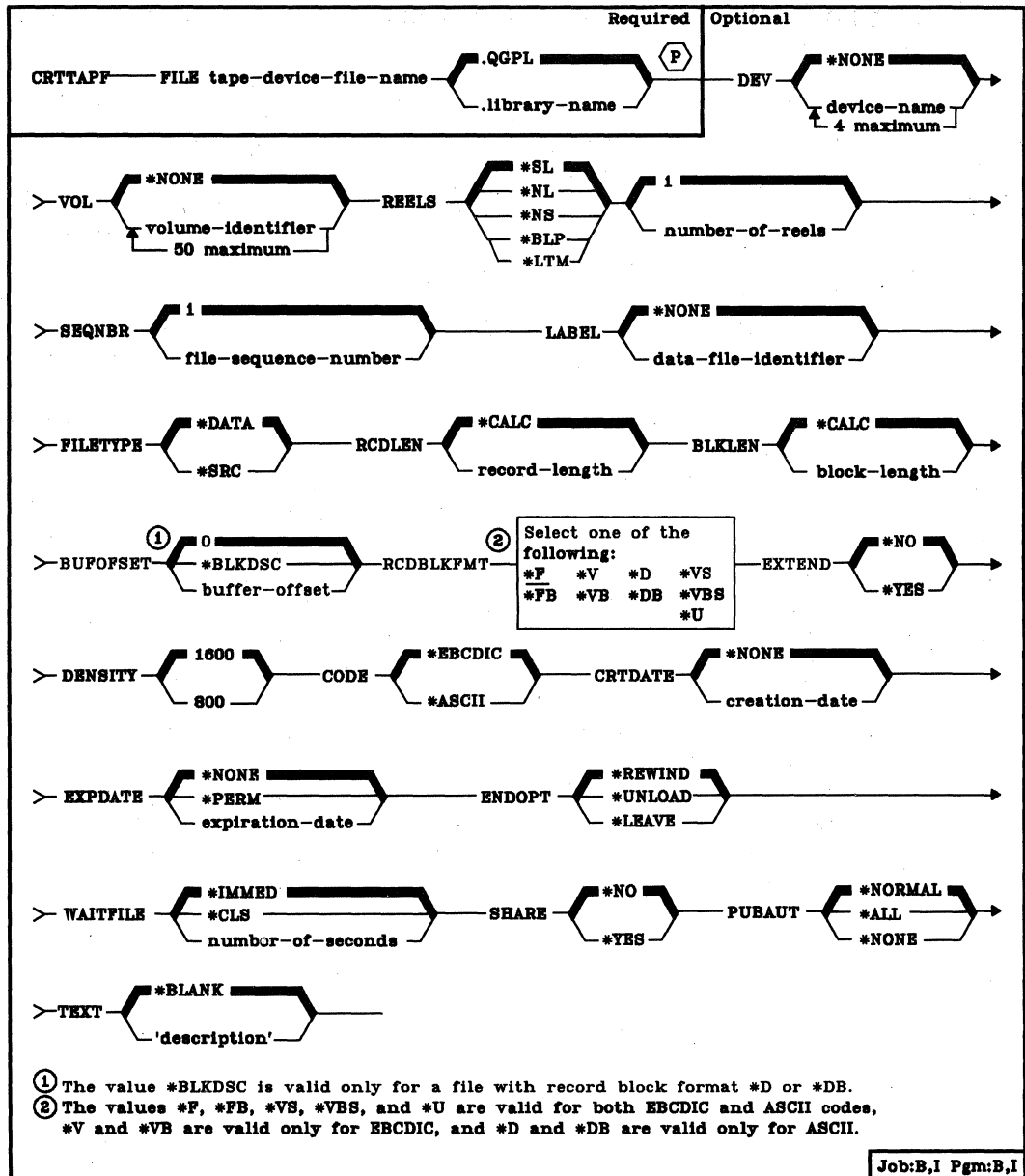
The Create Tape File (CRTTAPF) command creates a tape device file. The device file contains the file description, which identifies the device to be used; the device file does not contain data. The tape device file is used to read and write records on tape. The same device file can be used for both input 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.

CRTTAPF
(Diagram)



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.

**CRTTAPF
REELS**

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.

CRTTAPF
SEQNBR

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 alphanumeric 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					
CODE	RCDFBLKFMT	FILETYPE(*DATA)		FILETYPE(*SRC)	
		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

CRTTAPF
BLKLEN

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 RCDBLKFMFMT 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 CRRAPF, 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 program 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 *ASCII	*F *U *F *U	BLKLEN = RCDLEN BLKLEN = RCDLEN + BUFOFSET
*EBCDIC *ASCII	*FB *FB	BLKLEN = RCDLEN * n BLKLEN = (RCDLEN * n) + BUFOFSET n is the number of records in a maximum-length block
*EBCDIC *ASCII	*V *D	BLKLEN = RCDLEN + 8 BLKLEN = RCDLEN + 4 + BUFOFSET
*EBCDIC *ASCII	*VB *DB	BLKLEN >= RCDLEN + 8 BLKLEN >= RCDLEN + 4 + BUFOFSET
*EBCDIC *ASCII	*VS *VBS *VS *VBS	BLKLEN >= 18 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.

CRTTAPF
EXTEND

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.

***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 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.

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. 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.

***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.

***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.

CRTTAPF
WAITFILE

***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.)

*BLANK: 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)
```

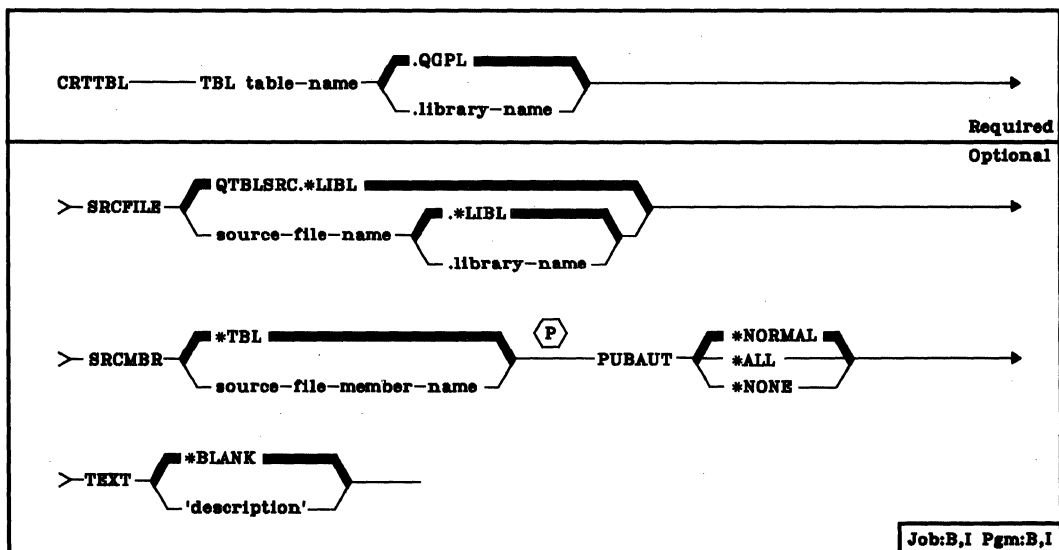
This command creates a description of the tape device file named BACKHST in library QGPL, to be used with the tape devices QTAPE1, QTAPE2, and QTAPE3. All volumes processed on these devices with this device file must have standard labels. Each block of data (EBCDIC character code) on the tape volumes 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*.

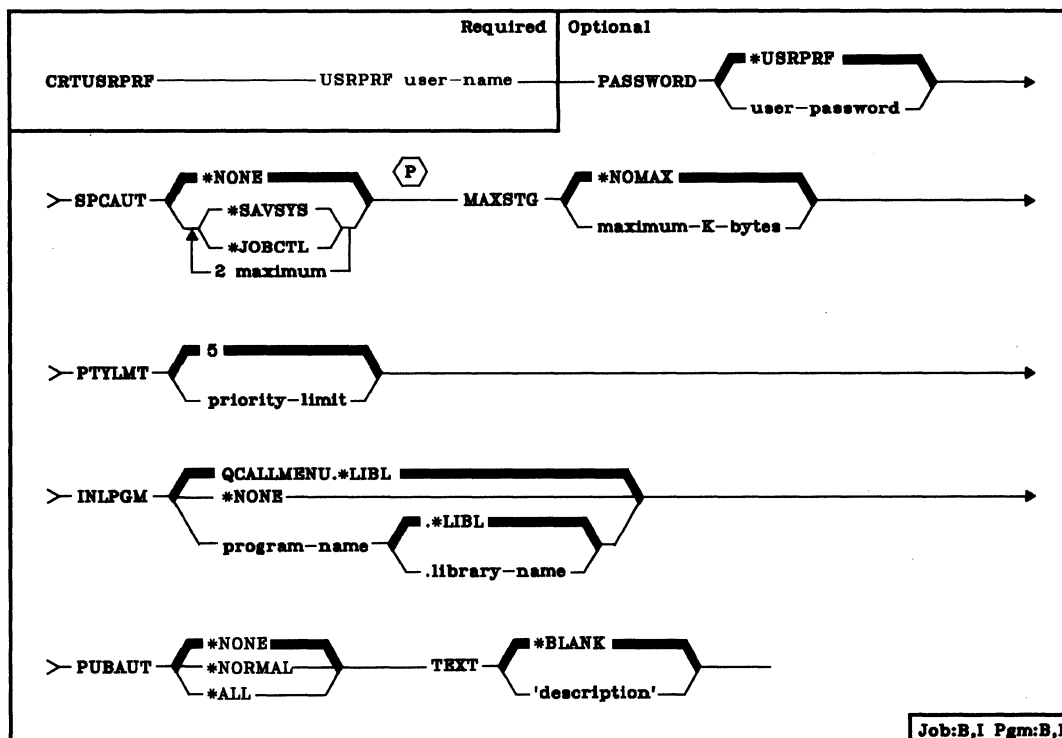
CRTUSRPRF (Create User Profile) Command

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.

CRTUSRPRF
(Examples)

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.

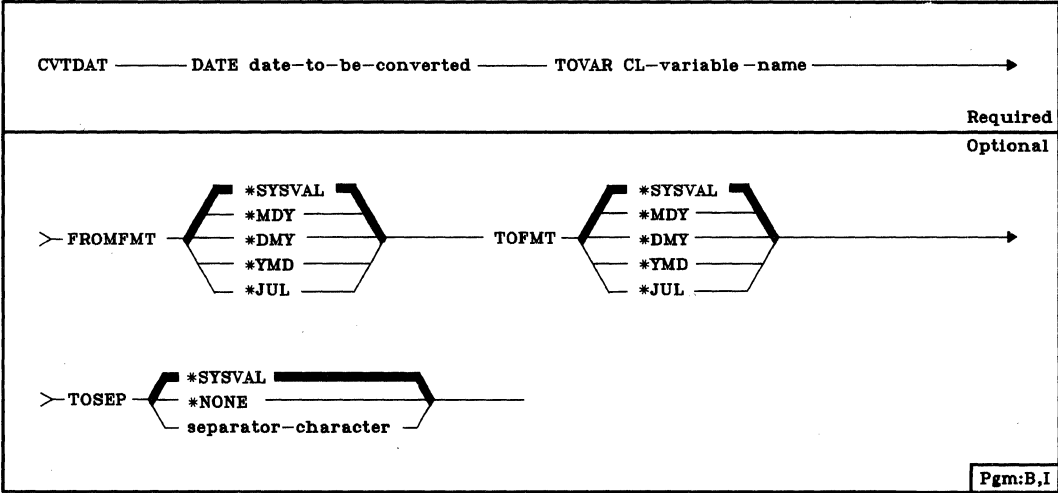
```
CRTUSRPRF USRPRF(TMSMITH) MAXSTG(10) +  
          INLPGM(CALC.PROGMR) +  
          TEXT('Ted M. Smith, Dept 41, +  
          Application Programs')
```

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.

CVTDAT
FROMFMT

FROMFMT Parameter: Specifies the current format of the date being converted.

***SYSVAL:** 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.

***SYSVAL:** 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.

***SYSVAL:** 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

CVTDAT (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.

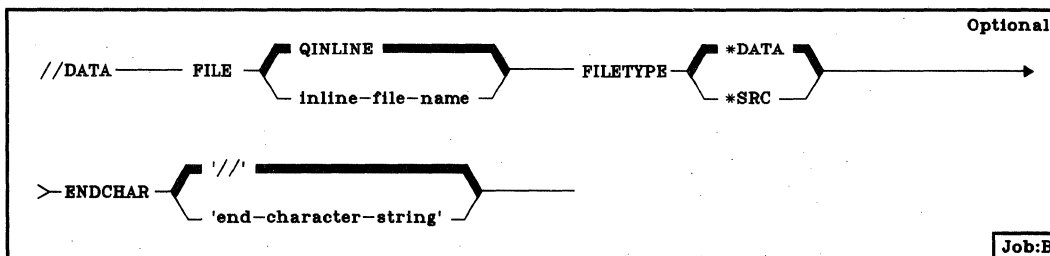
```
DCL &PAYDAY *CHAR 6
DCL &NEWPDAY *CHAR 6
.
.
.
CVTDAT DATE(&PAYDAY) TOVAR(&NEWPDAY) +
FROMFMT(*YMD) TOSEP(*NONE)
```

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.

QINLINE: 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.)

***DATA:** 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')
```

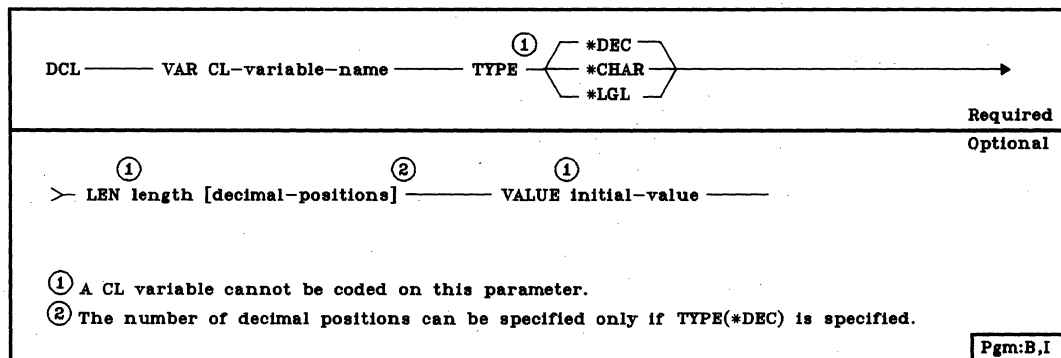
This command assigns the name FILE2 to the data following it; the file continues until a record is found that contains the characters STOPIT in positions 1 through 6. This delimiter allows the //JOB, //ENDJOB, and //DATA commands and records with /* in positions 1 and 2 to be embedded in an inline file.

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.



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

¹For 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.

**DCL
VALUE**

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.

Required
<p>DCLDTAARA ——— DTAARA ^① data-area-name ——— *.LIBL ——— .library-name</p>
<p>① A CL variable cannot be coded on this parameter.</p>
Pgm:B,I

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.

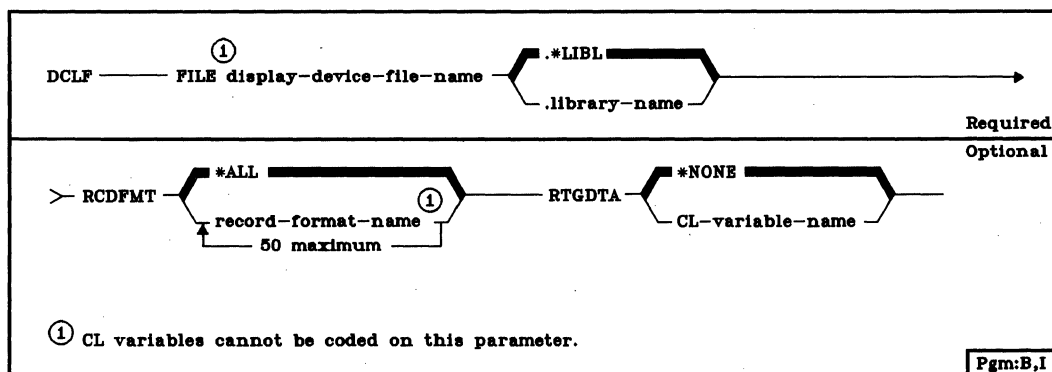
(The contents of the variables, not the variable names, are seen by the user; the display shows one, some, or all of the fields in the record format that the user can fill in. DDS determines the display format.)

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.

DCLF
RTGDTA

***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.

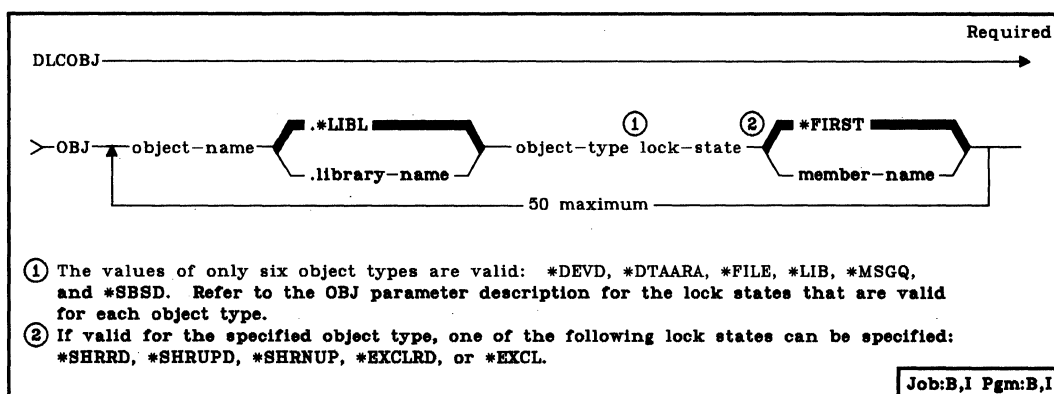
DCLF FILE(BAKER) RCDFMT(REC2 REC6) RTGDTA(&FB)

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.



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 DEV D 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*.

DLCOBJ
(Example)

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

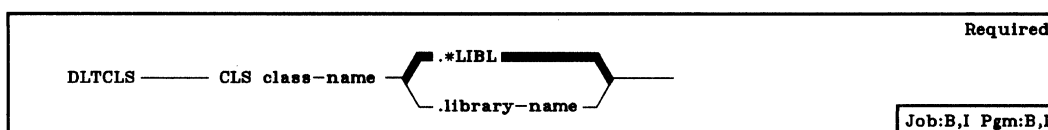
```
DLCOBJ OBJ((FILEA.LIBB *FILE *SHRRD))
```

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

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.

DLTCMD	_____	CMD command-name	┌	.*LIBL	┐	Required
			└	.library-name	┘	
						Job:B,I Pgm:B,I

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 (CRTUD) 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 (CRTDEV) 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

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

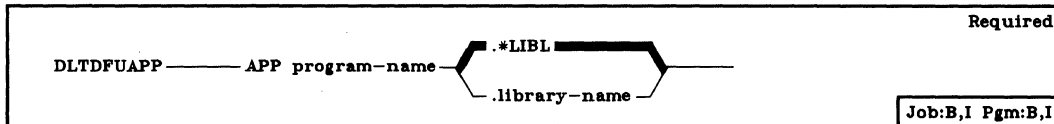
DLTDEVD DEVD(ARCTIC01)

This command deletes the device description of the device named ARCTIC01 from the system.

DLTDFUAPP (Delete DFU Application) Command

DLTDFUAPP

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)
```

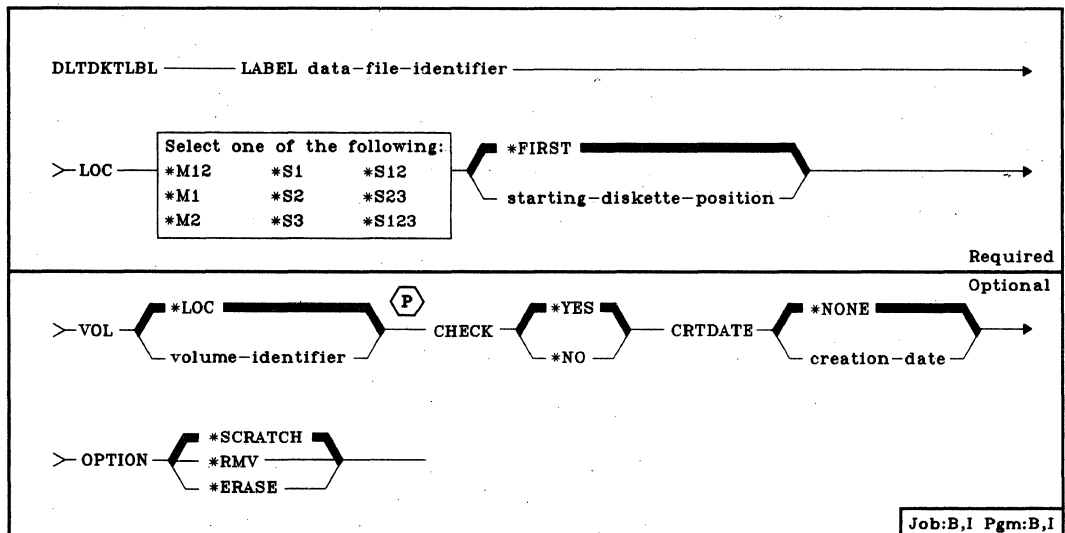
This command deletes the application DATA from the library named LIB1.

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.

***FIRST:** 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.

***LOC:** 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.

**DLTDKTLBL
CHECK**

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

DLTDTAARA

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.

DLTDTAARA	DTAARA	^① data-area-name	. *LIBL	Required
			.library-name	
^① A variable cannot be coded on this parameter.				Job:B,I Pgm:B,I

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

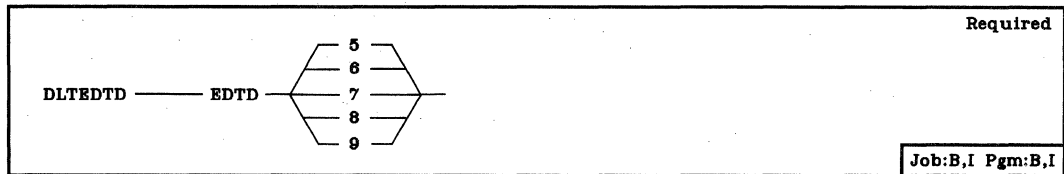
```
DLTDTAARA DTAARA(MYDATA.MYLIB)
```

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.

DLTF	FILE file-name	<pre> . *LIBL /-----\ / \ / \ / \ / \ / \ \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ </pre>	Required
			Job:B,I Pgm:B,I

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)
```

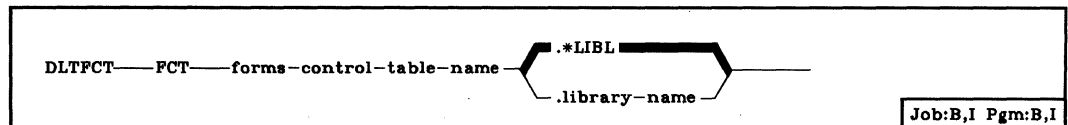
This command deletes the file named ORDERS, stored in the BILLING library. Only the BILLING library is searched for the file.

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

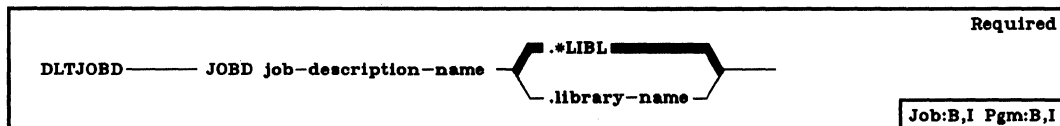
```
DLTFCT FCT(FORMCTRL.USERLIB)
```

This command deletes the forms control table named FORMCTRL in library USERLIB.

DLTJOB (Delete Job Description) Command

The Delete Job Description (DLTJOB) 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.



JOB 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

```
DLTJOB JOB(MYJOB.MYLIB)
```

This command deletes the job description named MYJOB 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.

DLTJOBQ ——— JOBQ job-queue-name *.LIBL .library-name	Required Job:B,I Pgm:B,I
--	--

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 ENDJRNP (End Journaling Physical File Changes) command to terminate the journaling.

DLTJRN _____ JRN journal-name _____ <div style="display: inline-block; vertical-align: middle; border-left: 1px solid black; border-right: 1px solid black; padding: 0 5px;"> .*LIBL .library-name </div>	Required Job:B,I Pgm:B,I
--	--

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.

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 CRTUD 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 QRPMSG), cannot be deleted (unless authorized by the security officer).

DLTMSGF	MSGF message-file-name	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> *.LIBL .library-name </div>	Required
			Job:B,I Pgm:B,I

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.

DLTMSGQ	MSGQ	message-queue-name	*LIBL	Required
			.library-name	Job:B,I Pgm:B,I

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

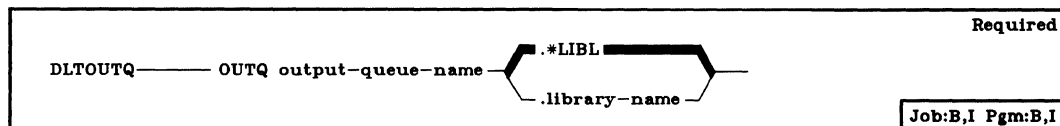
```
DLTMSGQ MSGQ(JONES)
```

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

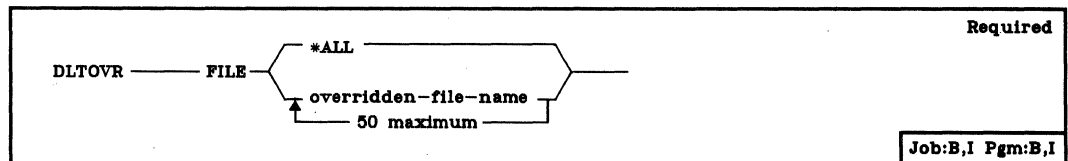
```
DLTOUTQ OUTQ(PUNCH2)
```

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

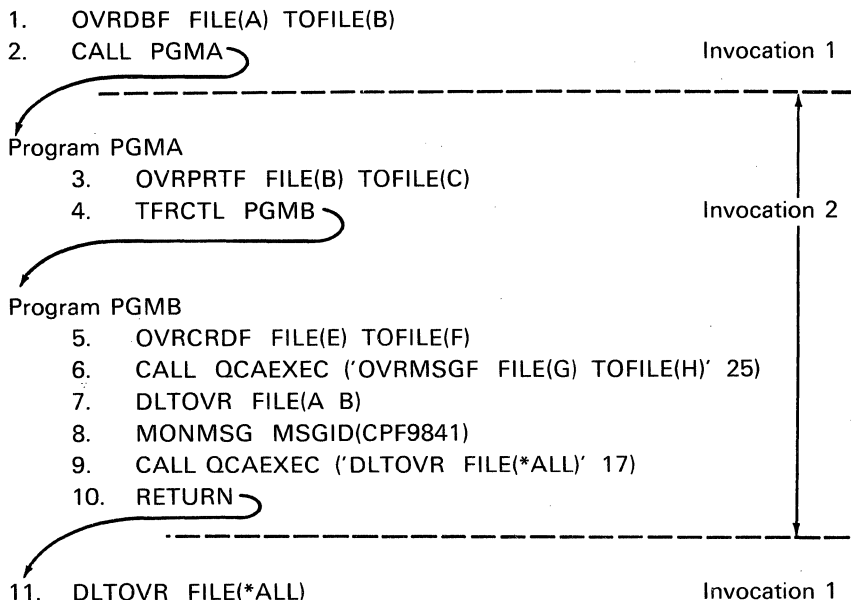
Example 1: Deleting Overrides in a Single Invocation

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.



DLTOVR
(Examples)

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.

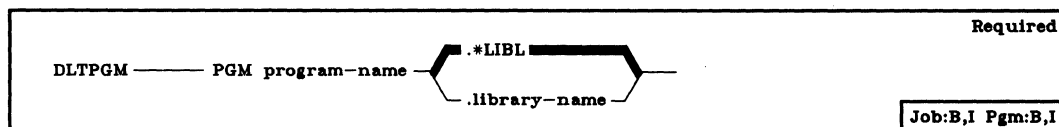
Command 9 deletes all remaining overrides in invocation 2. The overrides specified by commands 5 and 6 are deleted, but the override specified by command 1 is not.

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

DLTPRTIMG (Delete Print Image) Command

The Delete Print Image (DLTPRTIMG) command deletes the specified print image.

DLTPRTIMG	PRTIMG	print-image-name	.LIBL	Required
			.library-name	Job:B,I Pgm:B,I

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

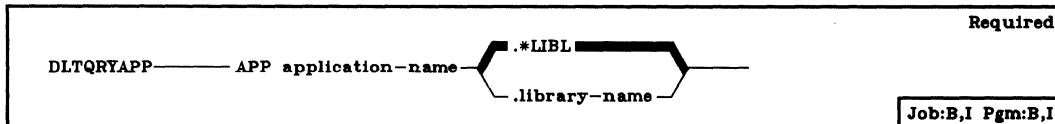
```
DLTPRTIMG PRTIMG(CHAR48PI)
```

This command deletes the print image named CHAR48PI from the system.

DLTQRYAPP (Delete Query Application) Command

DLTQRYAPP

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

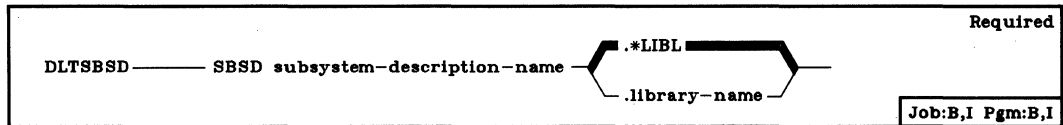
```
DLTQRYAPP APP(QDATA.LIB1)
```

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)
```

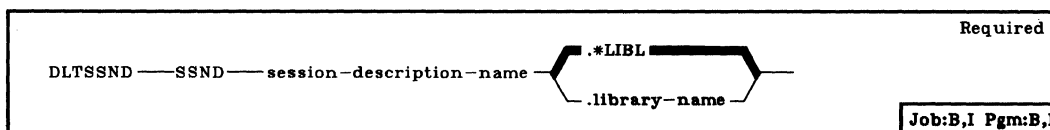
This command deletes the inactive subsystem description called BAKER from library LIB1.

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

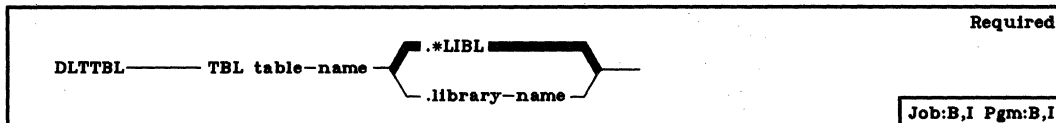
```
DLTSSND SSND(RJE.USERLIB)
```

This command deletes the inactive session description called RJE from library USERLIB.

DLTTBL

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.

DMPCLPGM _____	Required
	Pgm:B,I

There are no parameters for this command.

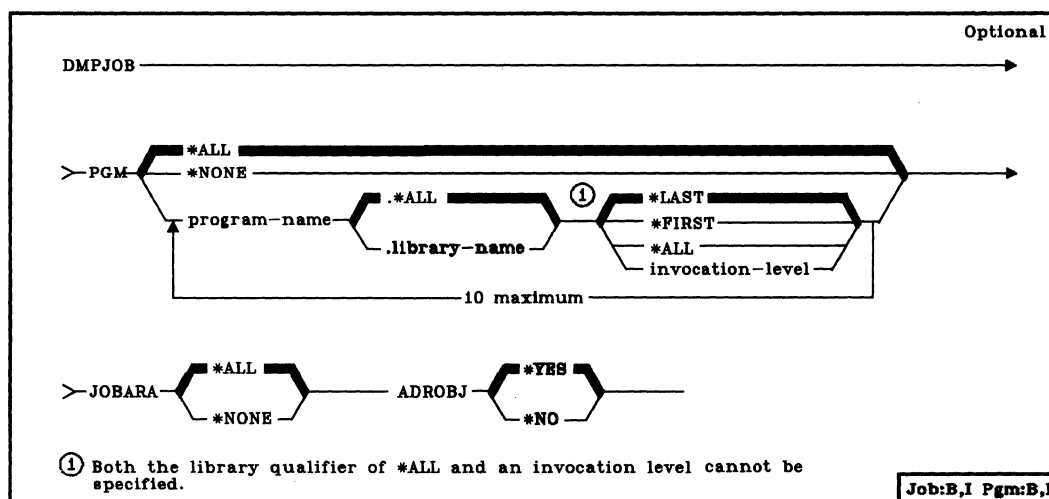
Example

```
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

DMPJOBINT

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.

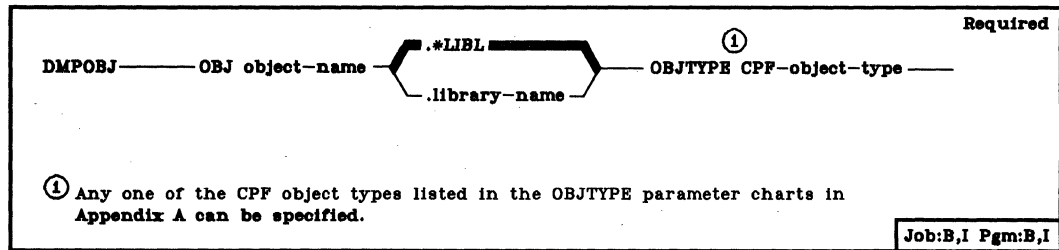
Example

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.



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)
```

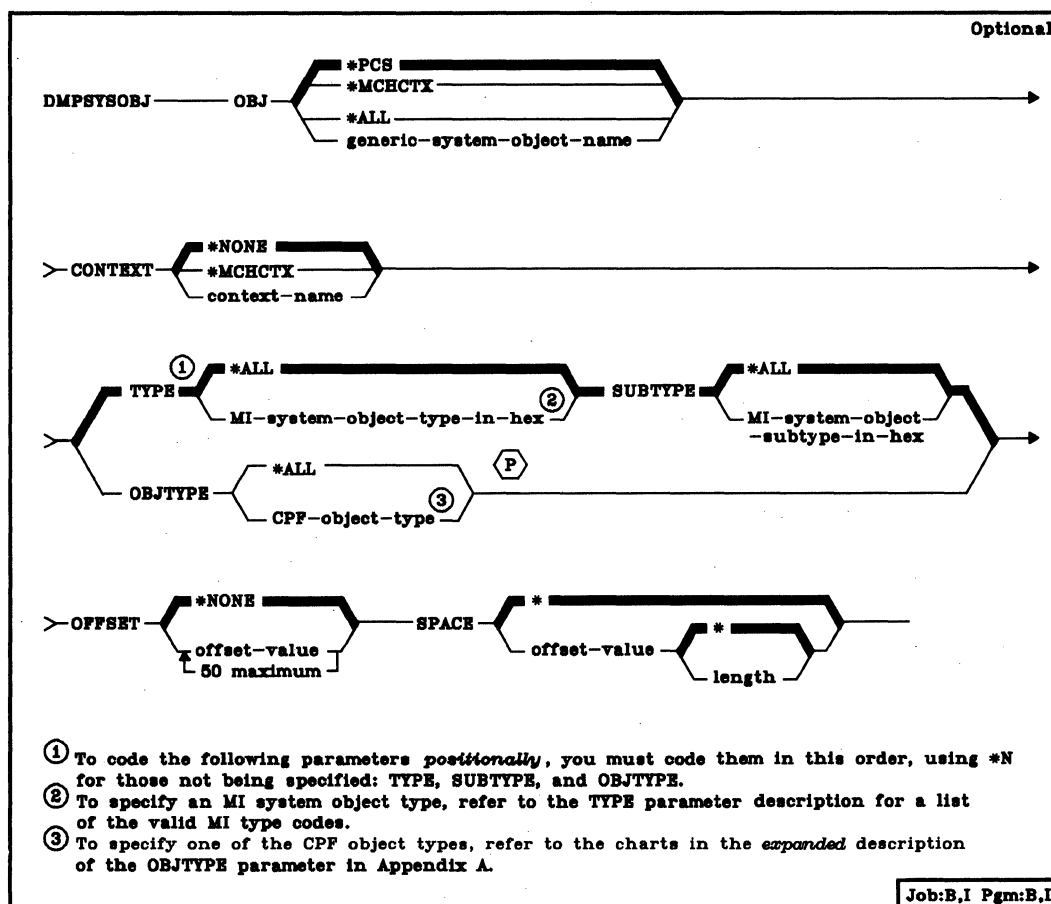
This command dumps the contents of the file named ORDERIN that is stored in the ORDENT library.

```
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.



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.

DMPYSOBJ
CONTEXT

***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

DMPYSOJB
SPACE

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

DMPYSOJB CONTEXT(QTEMP) TYPE(OE)

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 OE.

DMPYSOJB OBJ(WS1) CONTEXT(*MCHCTX) OBJTYPE(*DEVD)

This command dumps the device description for work station WS1, which is stored in the machine context.

DMPYSOJB OBJ(*PCS) SPACE(0 2A0)

This command dumps the work control block from the space associated with the process control space for the job.

DMPYSOJB 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) would be specified.

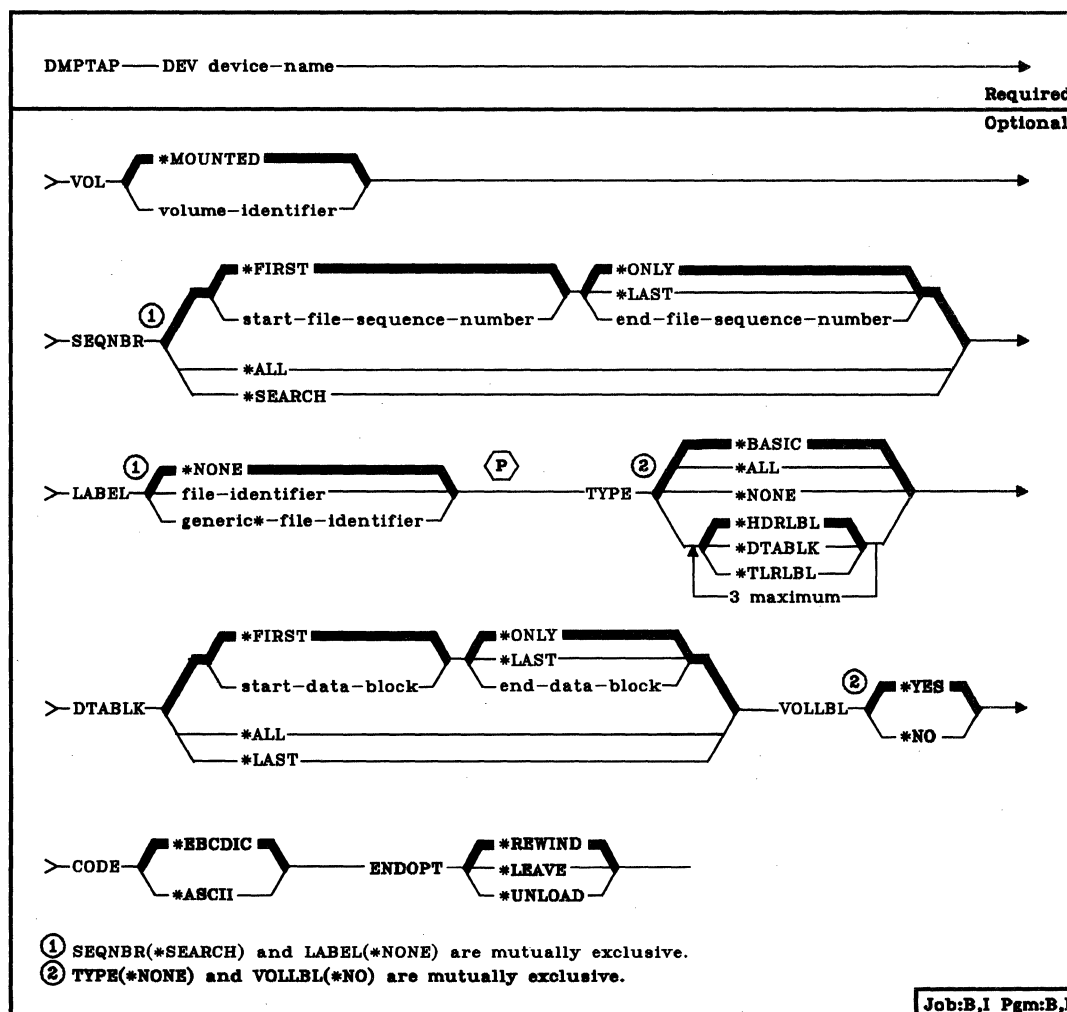
DMPTAP (Dump Tape) Command

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.



DMPTAP
DEV

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.

***ONLY:** 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.

***NONE:** 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 alphanumeric 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 alphanumeric 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.

DMPTAP
TYPE

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.

***HDLRBL:** 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(*HDLRBL) 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.

***FIRST:** 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).

***ONLY:** 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.

DMPTAP
VOLLBL

VOLLBL Parameter: Specifies whether volume labels are to be dumped. This parameter is ignored for nonlabeled volumes.

***YES:** 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.

***EBCDIC:** 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.

***REWIND:** 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.

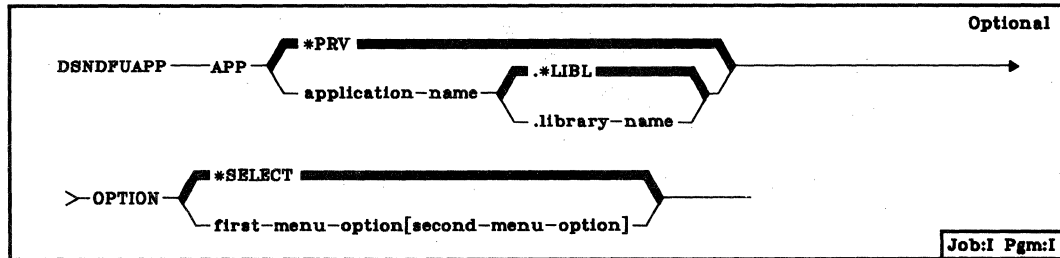
```

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.

***PRV:** 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.

***SELECT:** 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

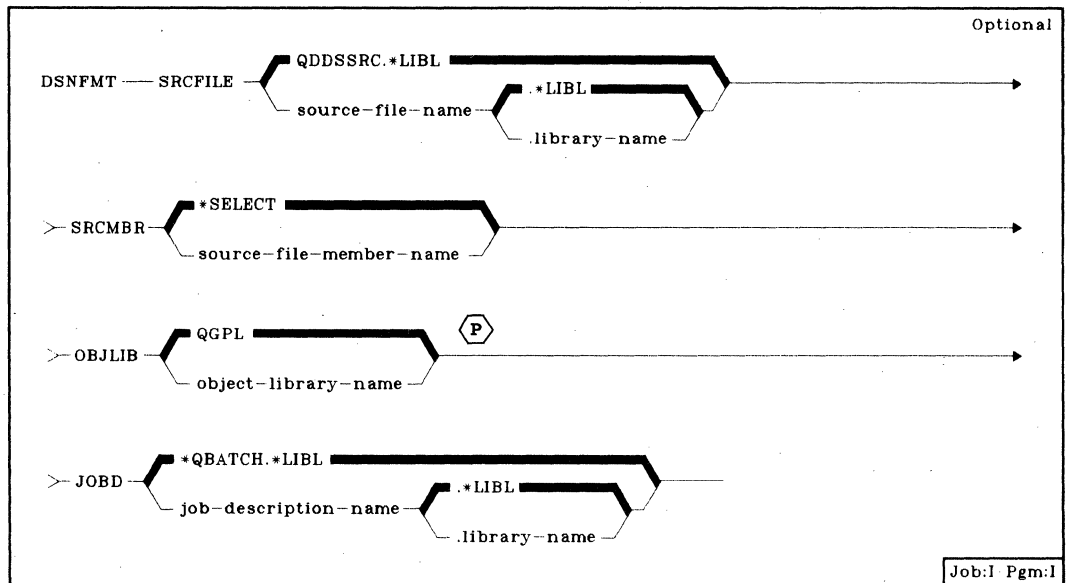
```
DSNDFUAPP APP(DATA.LIB1)
```

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.

JOBID Parameter: Specifies the qualified name of the job description to be used with jobs being submitted to SDA.

QBATCH.*LIBL: If the JOBID 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*.

[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

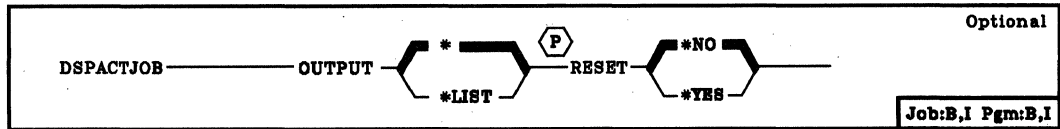
Example

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

DSPACTJOB (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-Redispl  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).

DSPACTJOB
(Considerations)

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 (QMNSYSRO, 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.

DSPACTJOB
(Considerations)

- * – 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.

DSPACTJOB
(Considerations)

- 4-HLDJOB: The job is held, but its spooled files are not held. The HLRDR 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 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.

DSPACTJOB
(Considerations)

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).

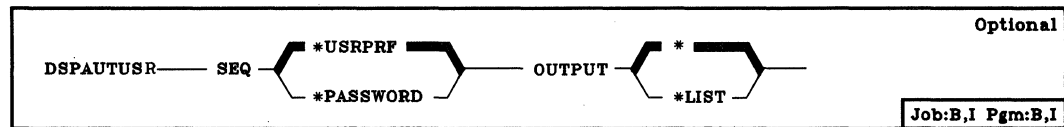
If there are more jobs than can be shown on a single display, the roll keys can be used and options can be placed in the input fields on multiple displays before the enter key is pressed.

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.

***USRPRF:** 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.)

***:** 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

DSPAUTUSR
(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.

Additional Considerations

DSPBKP (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
A Program: XXXXXXXXXXXX
B Statement: XXXXXXXXXXXX
  Breakpoint program: XXXXXXXXXXXX
  Library name: XXXXXXXXXXXX
  Breakpoint at invocation level: XXXXX XXXXX
  Output start pos: XXXXX Length: XXXXX Format: XXXXX
C Variable: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           XXXXXXXXXXXXXXXXXXXX
  Base: XXXXXXXXXXXXXXXXXXXXXXXXXXXX
      .
      .
      .
```

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 **A**, all of its breakpoints identified at **B** and all of its variables named at **C** 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 **A**) whose breakpoints are displayed on the following lines.
- The label name, the System/38 instruction number, or the statement number of the statement (at **B**) 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.)

DSPBKP
(Considerations)

- 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:  XXXXXXXXXXX XXXXX
Program:    XXXXXXXXXX   Inv lvl:  XXXXX
Output start pos: XXXXX Length:  XXXXX Format: XXXXX
Variable:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           XXXXXXXXXXX
Base:       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Type:      XXXXXXXXXX   Length:   XXXXX
Dimension:  XXXXX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXX...'
.
.
.
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:  XXXXXXXXXXX XXXXX
Program:    XXXXXXXXXXX   Inv lvl:  XXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
CF1-Cancel request  CF3-Command Entry  ENTER-Cont function chk
```

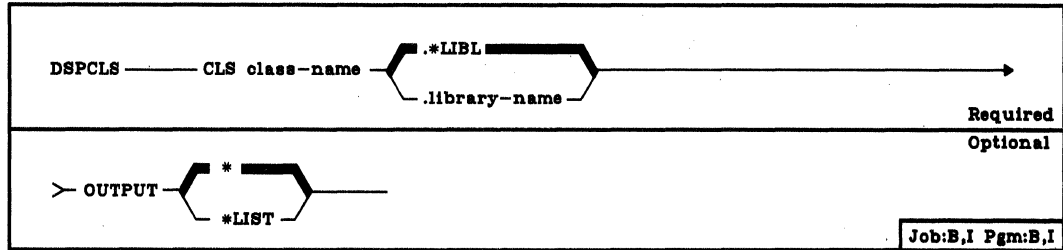
Again, line 2 identifies the statement/machine instruction number causing the error, and line 3 indicates the program and invocation level of the program in which the error occurred. The error message itself (first-level text only) is shown on the following lines. The second-level text can be displayed 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

DSPCLS
(Considerations)

The display produced by the DSPCLS command has the following format:

```
XX/XX/XX XX:XX:XX          CLASS DISPLAY
Class name:                 XXXXXXXXXXXX
Library name:               XXXXXXXXXXXX
Execution priority:         XX
Time slice in millisec:    XXXXXXXX
Eligible for purge:        XXXX
Default wait time in sec:  XXXXXXXX
Max CPU time in millisec:  XXXXXXXX
Max temp storage in K-bytes: XXXXXXXX
Text: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

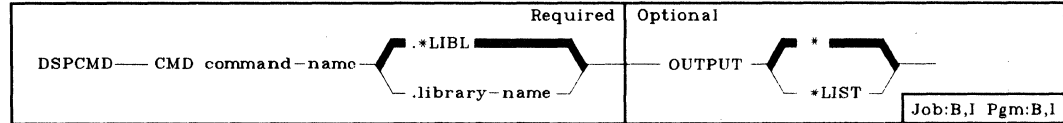
Line 1 of the class display shows the current date and time for the job. Lines 3 and 4 identify the class being displayed and the remaining lines show the attributes of the class.

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.

***:** 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

**DSPCMD
(Considerations)**

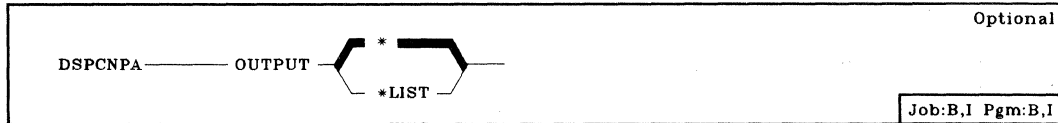
The display produced by the DSPCMD command has the following format:

```
XX/XX/XX XX:XX:XX          COMMAND DISPLAY
Command name: XXXXXXXXXXXX  Library: XXXXXXXXXXXX
  Program to execute command: PGM          XXXXXXXXXXXX
    Library name:                XXXXXXXXXXXX
  Source file name:           SRCFILE      XXXXXXXXXXXX
    Library name:                XXXXXXXXXXXX
  Source member name:        SRCMBR       XXXXXXXXXXXX
  Validity checking program: VLDCKR      XXXXXXXXXXXX
    Library name:                XXXXXXXXXXXX
  Mode in which valid:       MODE         XXXXXXXXXXXXXXXXX
                                         XXXXXXXXXXXXXXXXX
  Where allowed to execute:  ALLOW        XXXXXXXXXXXXXXXXX
                                         XXXXXXXXXXXXXXXXX
  Max positional parameters: MAXPOS      XXXXXXXXXXXX
  Message file for prompt text: PMTFILE  XXXXXXXXXXXX
    Library name:                XXXXXXXXXXXX
  Message file name:        MSGF         XXXXXXXXXXXX
    Library name:                XXXXXXXXXXXX
  Text description:         TEXT         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
```

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 (DSPCNPA) 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

**DSPCNPA
(Considerations)**

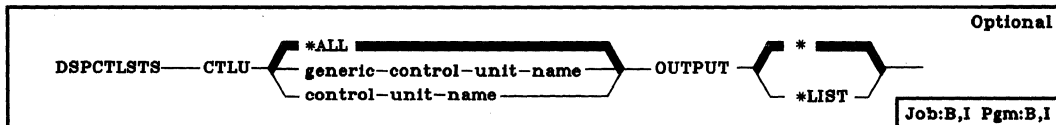
The DSPCNPA command produces the following display:

```
XX/XX/XX XX:XX:XX      CSNAP (CNP) ATTRIBUTES
Line names:             LINE      XXXXXXXXXXXX
XXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX
XXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX
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 - XXXXXXXXXXXX
LINE/CTLU/DEV STATUS JOB NAME USER NBR
-XXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
- XXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
- XXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX XXXXXX
- XXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX XXXXXX
- XXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
- XXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
-
1-DSPJOB 2-DSP desc 3-CHG desc 4-Vary on 5-Vary off 9-CHLJOB 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 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.

DSPCTLSTS
(Considerations)

- **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.