

SC34-0313-2

LICENSED
PROGRAM

File No. S1-32

IBM Series/1**Event Driven Executive****Utilities, Operator Commands, Program
Preparation, Messages and Codes**

Program Numbers: 5719-LM5 5719-LM6 5719-AM3
5719-UT3 5719-UT4
5719-XS1 5719-XS2
5719-XX2 5719-XX3
5740-LM2 5740-LM3

SC34-0313-2

LICENSED
PROGRAM

File No. S1-32

IBM Series/1**Event Driven Executive****Utilities, Operator Commands, Program
Preparation, Messages and Codes**

Program Numbers: 5719-LM5 5719-LM6 5719-AM3
5719-UT3 5719-UT4
5719-XS1 5719-XS2
5719-XX2 5719-XX3
5740-LM2 5740-LM3

Third Edition (April 1980)

Use this publication only for the purpose stated.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, refer to the latest IBM Series/1 Graphic Bibliography, GA34-0055, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services which 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.

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

This publication could contain technical inaccuracies or typographical errors. A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Systems Publications, Department 27T, P.O. Box 1328, Boca Raton, Florida 33432. 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.

(C) Copyright IBM Corporation 1979,1980

SUMMARY OF AMENDMENTS

Operator Commands

The following operator commands have been modified to include support for the IBM Series/1 4969 Magnetic Tape Subsystem:

- \$C
- \$VARYOFF
- \$VARYON

Session Manager

The following changes have been made to the Session Manager for the Event Driven Executive Version 2 (5719-UT4):

- \$PL/I (Option 10) has been added to the Program Preparation secondary option menu to support the PL/I compiler.
- Option 3 "Disk Utilities" of the primary option menu has been changed to "Data Management".
- \$TAPEUT1 (Option 10) has been added to the "Data Management" secondary option menu to support tape management.

\$TAPEUT1 Utility

This new utility is described in Chapter 4. The tape READ/WRITE return codes are described in Chapter 6.

\$PREFIND Utility

This utility has been updated to include support for locating tape data sets.

\$JOBUTIL - Job Stream Processor

This utility has been updated to support the PL/I compiler.

Suggested Utility Function Table

This table has been expanded to include all the utility programs within this book for easy reference to their functions and commands.

Glossary

New terms have been added to the glossary.

Reorganization

The book has been reorganized. It is now divided into six chapters with an introduction for each chapter.

Chapter 4 presents the utilities in alphabetic order.

Tabs

Subject tabs have been added to the upper outside corners of Chapters 1 through 6 of the book.

Bibliography

The Bibliography lists the books in the Event Driven Executive Library and a recommended reading sequence. Other publications related to the Event Driven Executive are also listed.

Miscellaneous Changes

This manual has been modified to include new function and to improve technical accuracy and clarity. Additional material and technical changes are indicated by vertical bars in the left margin.

HOW TO USE THIS BOOK

The material in this section is a guide to using this book. It defines the purpose, audience, and content of the book as well as listing aids for using the book and background materials.

PURPOSE

The purpose of this publication is to describe how to use the following:

The operator commands to perform various system control functions

The Session Manager to directly invoke utilities

The Batch Job Stream Processor (\$JOBUTIL) to invoke predefined sequences of utilities

The data management, terminal, graphics, text editing, program preparation, and diagnostic utilities

The messages and codes issued when using the Event Driven Executive

AUDIENCE

This manual is intended for use by:

- System programmers to aid in generating a system to meet the requirements of the installation
- Application programmers to prepare and execute application programs
- Operators to run the system.

HOW THIS BOOK IS ORGANIZED

This book contains six chapters:

"Chapter 1. Overview" provides an overview of the contents of the book.

"Chapter 2. Operator Commands" describes the operator commands and how to use them.

"Chapter 3. Session Manager" describes the method used to interactively access programs from a set of predefined menus and associated procedures.

"Chapter 4. Utilities" describes the utilities and how to use them. This chapter contains a table which cross references the utilities. It shows the utility and command to use to perform the desired function.

"Chapter 5. Application Program Preparation" describes the program preparation utilities and how to use them.

"Chapter 6. Messages and Codes" describes the messages and codes issued by the Event Driven Executive and explains their meaning.

AIDS IN USING THIS PUBLICATION

Illustrations in this book are enclosed in boxes. Many illustrations display output formats printed while using the Event Driven Executive system. In those cases where the actual printer output exceeds the size of the box, the information is illustrated in a compressed format.

Several other aids are provided to assist you in using this book:

- A Summary of Amendments lists the significant changes made to this publication since the last edition
- A Bibliography:
 - Lists the books in the Event Driven Executive library along with a brief description of each book and a recommended reading sequence
 - Lists related publications and materials
- A Glossary defines terms

- A Common Index which includes entries from each book in the Event Driven Executive library

Related Publications

Related publications are listed in the Bibliography.

SUBMITTING AN APAR

If you have a problem with the Series/1 Event Driven Executive services, you are encouraged to fill out an authorized program analysis report (APAR) form as described in the IBM Series/1 Authorized Program Analysis Report (APAR) User's Guide, GC34-0099.

CONTENTS

Chapter 1. Overview	1
Operator Commands	1
Session Manager	2
Utilities	2
Messages and Codes	7
Hardcopy Function for the 4978/4979 Display	7
Chapter 2. The Operator Commands	9
Invoking the operator commands	9
Entering command parameters	10
Operator Commands	11
\$A - List Active Programs	11
\$B - Blank Display Screen	12
\$C - Cancel Program	13
\$CP - Change Terminal's Partition Assignment	14
\$D - Dump Storage	15
\$E - Eject Printer	16
\$L - Load Program	17
\$P - Patch Storage	18
\$T - Set Date and Time	19
\$VARYOFF - Set Device Offline	20
\$VARYON - Set Device Online	22
\$W - Display Date and Time	25
Chapter 3. The Session Manager	27
Invoking the Session Manager	27
Session Manager Program Function Keys	28
Automatic Data Set Allocation/Deletion	29
Session Manager Menus	33
Primary Option Menu	35
Secondary Option Menus	36
Menu Option Combinations	46
Utilities Not Supported by Session Manager Menus	46
Chapter 4. The Utilities	47
Invoking the Utilities	47
Suggested Utility Usage Table	48
\$COMPRES - Compress Library	57
\$COPY - Copy Data Set	59
\$COPY Commands	59
\$COPYUT1 - Copy Data Set with Allocation	64
\$COPYUT1 Commands	64
\$DASDI - Format Disk or Diskette	68
Option 1 - 4964, 4966 Diskette Initialization	68
Option 2 - 4962 Disk Initialization	73
Option 3 - 4963 Disk Initialization	78
\$DEBUG - Debugging Tool	82
Debug Usage Considerations	83
Start and Termination Procedure	85
\$DEBUG Commands	86
\$DEBUG Command Descriptions	90

Tips and Techniques	103
\$DICOMP - Display Composer	105
Invoking \$DICOMP	105
\$DICOMP Commands	105
Composer Subcommands	109
\$DIINTR - Display Interpreter	127
Using \$DIINTR from an Application Program	127
3D Concepts as used by \$DIINTR	129
\$DISKUT1 - Allocate/Delete; List Directory Data	135
\$DISKUT1 Commands	135
\$DISKUT2 - Patch, Dump or Clear Member	142
\$DISKUT2 Commands	143
\$DIUTIL - Display Data Base Utility	150
Invoking \$DIUTIL	150
\$DIUTIL Commands	151
\$DUMP - Format and Display Saved Environment	163
Invoking \$DUMP	163
\$EDIT1 and \$EDIT1N - Line Editors	169
Data Set Requirements	169
Sequence of Operations	170
Special Control Keys	172
Editor Commands	173
Edit Mode Subcommands	182
Line Editing Commands	203
\$FONT - Process 4978 Character Image Tables	205
\$FONT Commands	205
Edit Mode	206
\$FSEDIT - Full Screen Editor	209
Data Set Requirements	209
Scrolling	210
Program Function Keys	211
\$FSEDIT Options and Command Summary	212
Primary Option Menu	213
Primary Commands	218
Edit Line Commands	226
\$IAMUT1 - Build and Maintain Indexed Data Set	235
\$IAMUT1 Commands	235
Building an Indexed Data Set	247
Determining Data Set Size and Format	247
\$IMAGE - Define 4978/4979 Formatted Screen Image	250
\$IMAGE Commands	250
Edit Mode	255
\$INITDSK - Initialize or Verify Volume	256
\$INITDSK Commands	256
Initialization	257
Verification	260
\$IOTEST - Test Sensor I/O; List Configuration	263
Invoking \$IOTEST	263
\$IOTEST Commands	264
\$JOBUTIL - Job Stream Processor	271
Setup Procedure	271
\$JOBUTIL Commands	272
Batch Job Example	290
\$LOG - Log I/O Errors into Data Set	292
Log Data Set	292

Invoking \$LOG	292
\$MOVEVOL - Disk Volume Dump/Restore	294
Diskette Usage	294
Data Set Specification	295
Dump Procedure	296
Restoration Procedure	299
\$PFMAP - Identify 4978 Program Function Keys	301
\$PREFIND - Prefind Data Sets and Overlays	302
Program Load Process Overview	302
\$PREFIND Usage Cautions	303
\$PREFIND Commands	304
Invoking \$PREFIND	305
\$TAPEUT1 - Tape Management	311
\$TAPEUT1 Commands	312
\$TERMUT1 - Change Terminal Parameters	334
\$TERMUT1 Commands	334
\$TERMUT2 - Process 4974/4978 Image/Control Store	339
4978 Support	339
4974 Support	340
Data Set Names	340
\$TERMUT2 Commands	340
\$TERMUT3 - Send Message to a Terminal	344
\$TRAP - Save Storage on Error Condition	348
Chapter 5. Application Program Preparation	351
Entering Source Statements	351
Program Assembly/Compilation	352
Linkage Editor	353
Object Module Conversion	353
Prefind of Data Sets and Overlays	353
Summary	354
\$EDXASM - Event Driven Language Compiler	356
Language Control Data Set	357
Required Data Sets	357
Compiler Options	358
\$EDXASM Output	359
Compiler Features Supported	361
Programming Considerations	361
Invoking \$EDXASM	368
\$EDXLIST - Compiler Listing Program	370
\$S1ASM - Series/1 Assembler	372
Required Data Sets	372
Invoking \$S1ASM	373
Host Assembler	382
Invoking the Host Assembler	382
Transferring the Object Module to Series/1	383
EDXLIST - Host Listing Formatter	383
Program Options	385
\$LINK - Linkage Editor	390
Combining Modules	392
Multiple Control Sections	392
Formatting Modules for \$UPDATE	392
Elimination of Duplicate Control Sections	393
Automatic Inclusion (Autocall)	393
Storage Map	393

Building an Event Driven Executive Supervisor	394
The Link Edit Process	394
Input to \$LINK	396
Data Sets Used by \$LINK	400
Autocall Option	401
Output from \$LINK	403
Invoking \$LINK	405
Object Module Record Format	407
\$UPDATE - Object Program Converter	408
\$update Commands	408
Invoking \$UPDATE	414
Creating a Supervisor	417
\$UPDATEH - Object Program Converter (Host)	418
\$updateh Commands	418
Chapter 6. Messages and Codes	421
System Operation Messages	421
IPL Operation	421
Volume Initialization	422
Tape Initialization	423
Storage Map Generation	423
Load Utility Location	424
Sensor I/O Status Check	424
Date and Time Printing	425
Program Load Message	425
Error Messages	427
Program Check Error Message	427
System Program Check Error Message	429
Processor Status Word (PSW)	430
\$update Error Messages	431
\$log Error Message	432
\$rmm Error Messages	433
\$trap Error Messages	435
Utility Completion Codes	436
\$edxasm Completion Codes	436
\$iamut1 Completion Codes	437
\$jobutil Completion Codes	439
\$link Completion Codes	440
\$update Completion Codes	443
Event Driven Language and Function Return Codes	444
\$diskut3 Return Codes	444
\$pds Return Codes	445
BSC Return Codes	446
Data Formatting Return Codes	447
EXIO Return Codes	448
Floating Point Return Codes	450
Formatted Screen Image Return Codes	450
Indexed Access Method Return Codes	451
LOAD Return Codes	452
Multiple Terminal Manager Return Codes	453
READ/WRITE Return Codes	454
SBIO (Sensor-based I/O) Return Codes	457
Terminal I/O Return Codes	458
Terminal I/O - ACCA Return Codes	459
Terminal I/O - Interprocessor Communications Return	

Codes	460
Terminal I/O - Virtual Terminal Communications Return	
Codes	461
TP Return Codes	463
Bibliography	467
Event Driven Executive Library Summary	467
Event Driven Executive Library	467
Summary of Library	468
Reading Sequence	470
Other Event Driven Executive Programming Publications	471
Other Series/1 Programming Publications	471
Other Programming Publications	472
Series/1 System Library Publications	472
Glossary	475
Common Index	487



LIST OF FIGURES

Figure 1.	Session Manager logon menu	27
Figure 2.	.\$SMALLOC contents	30
Figure 3.	Data sets created by the Session Manager	31
Figure 4.	.\$SMDELETE contents	32
Figure 5.	Menu selection hierarchy	34
Figure 6.	Primary option menu	35
Figure 7.	Program preparation utilities option menu	37
Figure 8.	.\$EDXASM parameter selection menu	38
Figure 9.	Data management option menu	39
Figure 10.	Terminal utilities option menu	41
Figure 11.	Graphics utilities option menu	41
Figure 12.	Communications utilities option menu	43
Figure 13.	Diagnostic aids utilities option menu	44
Figure 14.	.\$TRAP parameter selection menu	44
Figure 15.	.\$DUMP parameter selection menu	45
Figure 16.	.\$LOG parameter selection menu	45
Figure 17.	X,Y Coordinate Grid and Viewing Area	110
Figure 18.	X,Y,Z Coordinate Grid and Viewing Area	111
Figure 19.	Viewing Area in 3D Mode.	132
Figure 20.	.\$EDIT1/.\$EDIT1N Commands and Subcommands	171
Figure 21.	.\$MOVEVOL parameter input menu	295
Figure 22.	Event Driven Executive program preparation	355
Figure 23.	Programming with a linkage editor	391

CHAPTER 1. OVERVIEW

The following Event Driven Executive system components are described in this book:

- Operator commands to invoke programs and provide other system control functions
- A session manager to invoke the utilities from option menus
- Data management utilities to maintain disk, diskette, and tape data
- Diagnostic utilities to aid in hardware and software debugging
- Graphics utilities to define, display, and maintain graphic data
- Terminal utilities to define and modify terminal control information
- Text editing utilities to enter and edit source data
- Program preparation utilities for system and application program development
- Messages and codes to aid you in operation of the system

Each of these components is discussed later in the book in detail. A brief description of each follows.

OPERATOR COMMANDS

Twelve operator commands provide functions you can perform at your terminal. Commands that require parameters prompt you for them. Commands are accessed via the ATTN key of the 4978 or 4979 display terminals or the ESC or ALT MODE key on teletypewriter terminals.

The operator commands and the functions they perform are:

- \$A** Display loaded program names and locations
- \$B** Blank a 4978/4979 screen

Overview

\$C	Cancel a program
\$CP	Change a terminal's partition assignment
\$D	Dump storage
\$E	Eject printer page
\$L	Load a program
\$P	Patch storage
\$T	Enter the date and time
\$VARYOFF	Set a device offline
\$VARYON	Set a device online
\$W	Display the date and time

SESSION MANAGER

The session manager is a menu-driven interface used to access both system functions and your applications through a set of predefined full screen menus and their associated procedures. See "Chapter 3. The Session Manager" on page 27 for a detailed description on the session manager.

UTILITIES

The utilities are a set of programs that provide productivity aids for Series/1 application program development and system maintenance.

To aid you in using these utilities, the Event Driven Executive system provides three ways to invoke the utility programs from a terminal:

- The session manager
- The job stream processor utility (\$JOBUTIL)
- \$L command

Most utility programs are used interactively from a terminal. After a utility program is invoked, you can list its defined operations and command by entering a question mark in response to the 'COMMAND (?):' prompt.

In Chapter 4, the utility programs are presented in alphabetic order along with examples of their usage.

The session manager groups the utility programs by function. The following represents the functional groupings of the utilities along with the operations they perform.

Text Editing Utilities

The text editing utilities provide facilities for entering and editing source programs as follows:

- \$EDIT1 A line editor that uses host data sets
- \$EDIT1N A line editor that uses Series/1 data sets
- \$FSEDIT A full screen editor that uses Series/1 or host data sets

Program Preparation Utilities

The program preparation utilities aid in:

- \$COBOL Compiling COBOL programs
- \$EDXASM Compiling Event Driven Language programs
- \$EDXLIST Reformatting \$EDXASM listings
- \$FORT Compiling FORTRAN programs
- \$LINK Link editing more than one program together
- \$PL/I Compiling PL/I programs
- \$PREFIND Prefinding data sets and overlay programs to shorten program loading time
- \$S1ASM Assembling Series/1 assembler language programs

Overview

- \$UPDATE** Converting an object program into an executable load module
- \$UPDATEH** Converting a host object program into an executable load module

Data Management Utilities

The data management utilities aid in:

- \$COMPRES** Compressing disk or diskette libraries
- \$COPY** Copying disk or diskette data sets or volumes
- \$COPYUT1** Copying data sets and volumes with dynamic allocation of the receiving data sets
- \$DASDI** Initializing, formatting, and verifying disks or diskettes
- \$DISKUT1** Allocating and deleting data sets; listing directory data
- \$DISKUT2** Patching and dumping data sets; listing the error log data set
- \$DISKUT3** Performing data management functions from another program. \$DISKUT3 is described in the System Guide.
- \$IAMUT1** Building and maintaining Indexed Access Method data sets
- \$INITDSK** Initializing and verifying a direct access storage volume for use with the Event Driven Executive
- \$MOVEVOL** Transferring volumes of data between systems and creating backup copies of an online data base
- \$PDS** Organizing and accessing partitioned data sets from another program. \$PDS is described in the System Guide.
- \$TAPEUT1** Allocating tape data sets, copying data sets or volumes from disk or diskette to tape, from tape to disk or diskette, or from tape to tape, and changing tape attributes.

Terminal Utilities

The terminal utilities aid in:

- \$FONT Creating and modifying character image tables for your display terminal
- \$IMAGE Defining formatted screen images
- \$PFMAP Displaying program function key assignments
- \$TERMUT1 Altering logical device names, address assignments, or terminal configurations
- \$TERMUT2 Defining routines and changing key definitions on the 4978 keyboard. Restoring the 4974 printer to the standard character set.
- \$TERMUT3 Sending messages from one terminal to another

The Multiple Terminal Manager utility programs are documented in the Communications and Terminal Applications Guide.

Graphics Utilities

Under the direction of a display processor, three graphics utilities aid in:

- \$DICOMP Generating and modifying displays using an online composer
- \$DIINTR Using an interpreter to display and process the data base
- \$DIUTIL Maintaining the resulting data base

Communications Utilities

The communications utilities aid in:

- \$BSCTRCE Tracing the I/O activities on a given binary synchronous communications line

Overview

\$BSCUT1 Formatting binary synchronous trace files to either a printer or a terminal

\$BSCUT2 Exercising BSCAM capabilities

\$RJE2780 Simulating a 2780 RJE interface

\$RJE3780 Simulating a 3780 RJE interface

\$PRT2780 Printing spool records produced by \$RJE2780

\$PRT3780 Printing spool records produced by \$RJE3780

\$HCFUT1 Interacting with the Host Communications Facility

The communications utilities are documented in the Communications and Terminal Applications Guide.

Diagnostic Utilities

The diagnostic utilities aid in:

\$DEBUG Debugging programs

\$DUMP Formatting and displaying the data saved by \$TRAP on an error condition

\$IOTEST Performing the following functions:

- Testing the operation of sensor based I/O features
- Listing the hardware configuration of the Series/1
- Listing the devices supported by the system
- Listing volume information

\$LOG Logging I/O errors into a data set

\$TRAP Intercepting certain class interrupts and recording the environment on a disk or diskette data set

MESSAGES AND CODES

While using the Event Driven Executive, you may encounter return codes, completion codes, and messages. They are found in Chapter 6. Messages and Codes.

HARDCOPY FUNCTION FOR THE 4978/4979 DISPLAY

Pressing the PF6 key or the assigned hardcopy key on the 4978/4979 keyboard causes the entire display (24 lines) to be transferred to the designated hardcopy device. (During system generation, the TERMINAL statement is used to define the hardcopy device.) If the hardcopy device has not been defined or is currently busy with another operation, then no action is taken. Otherwise, the screen cursor moves to each line as it is printed, returning to its original position after the page is printed. The hardcopy function should not be activated while the screen is being changed.

Operator Commands

CHAPTER 2. THE OPERATOR COMMANDS

Twelve operator commands provide system control functions you can perform at your terminal. The operator commands begin with the character \$ and are directed to the supervisor. (The commands directed to the various utilities are described in "Chapter 4. The Utilities" on page 47 for each utility). Commands that require parameters will prompt you for them.

The operator commands and the functions they perform are:

\$A	Display loaded program names and locations
\$B	Blank a 4978/4979 screen
\$C	Cancel a program
\$CP	Change a terminal's partition number
\$D	Dump storage
\$E	Eject printer page
\$L	Load a program
\$P	Patch storage
\$T	Enter the date and time
\$VARYOFF	Set a device offline
\$VARYON	Set a device online
\$W	Display the date and time

INVOKING THE OPERATOR COMMANDS

To invoke the operation commands, press the ATTN key on the 4978 or 4979 (designated attention key on the teletypewriter terminal). Then enter the desired command in response to the prompting message > from the supervisor.

Note: If the system includes more than 64K bytes of storage, the \$A, \$C, \$D, \$L, and \$P functions operate only within the storage partition assigned to the terminal.

Operator Commands

ENTERING COMMAND PARAMETERS

You are prompted for required parameter information, for example, the storage addresses to be displayed by \$D or the name of the program to be loaded by \$L.

Note: In the syntax definitions in this chapter, the required fields need not be entered on the same line as the command.

An alternate method for entering the operator commands is the single line format. This format allows you to enter successive fields, separated by blanks, as a single entry. This can be done for as many fields as the system can process before it must print an informational response. A possible entry using single line format is:

```
$L $EDXASM CALCSRC ASMWORK ASMJOB
```

OPERATOR COMMANDS

\$A - List Active Programs

Displays the names and load points of all programs that are active within the partition to which the requesting terminal is assigned. Programs that were loaded by operator commands entered at your terminal are identified by an asterisk.

Syntax

```
$A
Required:  None
Default:   None
```

No operands are supported.

Example - Display active programs

```
> $A
PROGRAMS AT 08:14:19
IN PARTITION #5
$SMMAIN 0000 *
$JOBUTIL 0400 *
$DISKUT1 0800 *
$COPYUT1 2600
```


Operator Commands

\$B - Blank Display Screen

Blanks or erases the requesting terminal's (4978/4979) screen, both protected and unprotected areas.

Syntax

```
$B  
Required:  None  
Default:   None
```

No operands are required.

Example - Blank screen

```
> $B
```

Note: Display screen is blanked.

\$C - Cancel Program

Cancels a program and frees the storage that it occupied. When more than one copy of the program is in your partition, you are prompted to specify the load point of the program you wish to cancel. Use \$A to obtain the load point.

\$C will also close, rewind, and set offline any tape data sets defined in the program header of the cancelled program. If a tape drive is online and targetted to receive data, the operation will complete and tape will be set offline.

Caution: Do not use the \$C command as a normal means of terminating program execution. Use it only as an aid when no other way exists to force termination of a program (such as a program to be cancelled is in an endless loop of instructions). \$C can cause unpredictable errors (the task error exit is not taken) and should only be used as a last resort to avoid having to IPL the system again.

Syntax

\$C	program
Required:	program
Default:	None

Operands **Description**

program The name of the program to cancel.

Example - Cancel \$EDIT1

> \$C \$EDIT1
\$EDIT1 CANCELED AT 08:16:24

Operator Commands

\$CP - Change Terminal's Partition Assignment

This command allows you to change the partition number assigned to your terminal. If an invalid partition number is specified, an error message is displayed.

Note: If you are using a 4952 processor, you are limited to partitions 1 and 2.

Syntax

```
$CP          n
Required:    n
Default:     None
```

Operands Description

n	The partition to which the terminal is to be assigned.
---	--

Example - Assign terminal to partition 2

```
> $CP
PARTITION # ? 2
```

\$D - Dump Storage

Dumps the contents of storage in hexadecimal on the terminal.

Syntax

```
$D          origin,address,count
Required:   origin,address,count
Default:    None
```

Operands Description

origin	The hexadecimal origin address (the program load point).
address	The hexadecimal address in the program at which the dump is to start.
count	The decimal number of words to dump.

Example - Dump first 10 words of partition

```
> $D
ENTER ORIGIN: 0000
ENTER ADDRESS,COUNT: 0000,10
0000: 6802 6AF6 0000 0000 6C34 6AF2 6C34 6AF2
0010: 0000 0000
ANOTHER DISPLAY? N
```

Operator Commands

\$E - Eject Printer

Causes the 4974 or 4973 printer (defined as \$SYSPRTR) to advance to the top of the next page a specified number of times.

Syntax

\$E	n
Required:	None
Default:	Ejects one page

<u>Operands</u>	<u>Description</u>
-----------------	--------------------

n	The number of pages to eject.
---	-------------------------------

Example - Eject page on printer

```
> $E 2
```

Note: Printer ejects two pages.

\$L - Load Program

Loads a program from disk or diskette and starts it.

Syntax

```

$L          program,volume,storage data set(s)
Required:   program
Default:    volume defaults to IPL volume

```

<u>Operands</u>	<u>Description</u>
program	The name of the program to load.
volume	The name of the volume which contains the program to load.
storage	The total additional storage to be added to the end of the loaded program.
data set(s)	Data set(s) to be passed to the program being loaded (if required). Specify the data set(s) in the order the program expects.

Example - Load a Program Called PROCESS from EDX003 and Pass a Single Data Set, MYDATA

```
> $L      PROCESS,EDX003 MYDATA
```

Note: Wait until the system is initialized before loading a program. If your system has timers, the system is initialized when the 'SET TIME AND DATE USING \$T' appears (or when the time and date are printed). If your system does not have timers, the system is initialized when it enters the wait state after the storage map has been displayed.

Operator Commands

\$P - Patch Storage

Allows main storage to be patched online. Enter the patch data in response to prompting messages.

Syntax

```
$P          origin,address,count
Required:   origin,address,count
Default:    None
```

Operands Description

origin	The hexadecimal origin address (program load point).
address	The hexadecimal address in the program at which the patch is to start.
count	The decimal number of words to patch.

Example - Patch word X'100' of program loaded at 0 to X'FFFF'

```
> $P
ENTER ORIGIN: 0000
ENTER ADDRESS,COUNT: 0100,1
0100: C462
DATA: FFFF
ANOTHER PATCH? N
```

\$T - Set Date and Time

Enters a new date and time into the system and resets the realtime clock. You can only use \$T from terminals having the label \$SYSLOG and \$SYSLOGA. After entering the time, the timer is started at the instant carriage return/ENTER is pressed. This resets the seconds to zero.

Notes:

1. Make sure your time and date entry is correct as the system does not verify this data.
2. If \$T is entered from other than \$SYSLOG or \$SYSLOGA, it is equivalent to entering \$W.

Syntax

```

$T          date,time
Required:   date,time
Default:    date defaults to 00/00/00
            time defaults to 00:00:00

```

<u>Operands</u>	<u>Description</u>
date	The current date.
time	The current time.

Example - Set date and time

```

> $T
DATE(M.D.T): 8:22:79
TIME(H.M): 11:15

```


Operator Commands

\$VARYOFF - Set Device Offline

Sets the status of a disk, diskette, diskette magazine unit, or tape drive to offline.

On the 4966 diskette magazine unit, each diskette volume in individual diskette slots or either of the diskette magazines can be set to offline.

When you vary a tape device offline, that tape drive is rewound to the load point and set logically offline.

Syntax

```
$VARYOFF    ioda slot
Required:   ioda
Default:    None
```

Operands Description

ioda	The hexadecimal device address of the device to be varied offline.
slot	The slot number (1,2,3,A,B) of the diskette to be varied offline. This parameter applies to the 4966 only.

Examples:

Vary offline the volume in slot 2 of a 4966 device at address 22

```
> $VARYOFF    22 2
IBMIRD OFFLINE
```

Vary offline tape drive at device address 4C on which a standard label tape volume (volume serial 123456) was mounted and is displayed.

```
> $VARYOFF 4C
123456 OFFLINE
```

Vary offline tape drive at device address 4E. In this example, the tape drive was defined for non-labeled (NL) tapes or for bypass label-processing (BLP). Therefore, the tape ID assigned to that device at system generation is displayed.

```
> $VARYOFF 4E
TAPE1 OFFLINE
```

Note: If you vary offline a tape drive that is online and in use, you are prompted as follows:

```
DEVICE MARKED IN USE, CONTINUE? (Y,N):
```

If response is N, the tape is not varied offline. If response is Y, the tape will be put logically offline (closed) and usable (ready to be varied online). This allows an "unclosed" tape drive to be recovered.

Operator Commands

\$VARYON - Set Device Online

Sets the status of a disk, diskette, diskette magazine unit, or tape drive to online.

On the 4966 diskette magazine unit, each diskette volume in the individual slots or either of the diskette magazines can be independently set to online. When a new diskette volume is mounted, the diskette volume must be online for it to become accessible. I/O commands issued to disk or diskette will not operate unless the device and/or the diskette volume is online.

Before I/O commands can be issued to a tape, the tape must be mounted on a tape drive and varied online.

\$VARYON performs special tape functions, depending on the label type that is defined for the tape drive.

- If the drive is defined for a standard label (SL) tape, the VOL1 volume label is read.
- If the drive is defined for a non-labeled (NL) tape, the leading tape mark (if one exists) is automatically bypassed or, if a label is encountered, terminates without setting the tape online.
- If the drive is defined for bypass label-processing (BLP), no initial tape motion occurs.

\$VARYON also allows access to a multiple-file tape volume through a specified file sequence indicator.

The tape drive must be set to the proper density at system generation or by the Change Tape (CT) command of the \$TAPEUT1 utility before you vary a tape online. You can request that the expiration date on an SL tape data set be ignored.

\$VARYON	ioda slot file 'EX'
Required:	ioda
Default:	file defaults to 1, maximum value of 255

Note: The OR symbol (|) indicates mutually exclusive operands.

<u>Operands</u>	<u>Description</u>
ioda	The hexadecimal device address of the device to be varied online.
slot file	The slot number (1,2,3,A,B) of the diskette to be varied online. This parameter applies to the 4966 only.
	The decimal file number on the tape to be accessed. This parameter applies to the tape drive only.
'EX'	This parameter applies to tape only and requests an expiration date override. If a tape data set is initialized with an expiration date, this parameter must be used to write to that tape data set and the file number must be specified.

Examples:

Vary diskette in slot 1 of 4966 at device address 22 online

<pre>> \$VARYON 22 1 IBMIRD ONLINE</pre>

Vary a standard label (SL) tape (volume 123456) at address 4C online and access the first file.

Operator Commands

```
> $VARYON 4C  
123456 ONLINE
```

Vary a non-labeled (NL) tape at address 4C online and access the second file, where TAPE1 was the ID assigned at system generation.

```
> $VARYON 4C 2  
TAPE1 ONLINE
```

Vary a standard label tape at address 4D online. The first file of this tape has an expiration date that has not expired; however, output to this file is allowed.

```
> $VARYON 4D 1 EX  
OVERRIDE EXPIRATION DATE CHECK? (Y,N): Y  
123456 ONLINE
```

\$W - Display Date and Time

Displays date and time.

Syntax

```
$W
Required:  None
Default:   None
```

No operands are supported.

Example - Display date and time

```
> $W
DATE = 08/22/79  TIME = 11:16:54
```

Session Manager

CHAPTER 3. THE SESSION MANAGER

The session manager provides a quick and easy to use method of accessing programs (including the utilities) interactively from a terminal. Predefined full screen menus and associated procedures enable you to invoke the functions you request. Full screen 4978 or 4979 display terminals present prompting messages and gather input parameters for processing system functions or application programs. Input requests are accompanied by prompting messages to help you supply all required information.

The session manager requires a minimum partition of 10K bytes to process menus and your requests, but only 2K bytes remains resident during execution of the functions you request. Each terminal (4978 or 4979) has a dedicated copy of the session manager loaded into the partition to which that terminal is assigned.

INVOKING THE SESSION MANAGER

You can invoke the session either of two ways:

- as part of the IPL procedure
- your request using \$L (Load a Program)

When you IPL the Event Driven Executive system, the session manager can be automatically loaded for each active 4978 or 4979 display terminal and the logon menu (as shown in Figure 1) is displayed.

```

$SMMLOG: THIS TERMINAL IS LOGGED TO THE SESSION MANAGER
                                                17:55:31
ENTER 1-4 CHAR USER ID ==>                05/24/79
(ENTER LOGOFF TO EXIT)

ALTERNATE SESSION MENU ==>
(OPTIONAL)
```

Figure 1. Session Manager logon menu

Session Manager

If you wish to start the session manager at IPL time, you must rename the session manager module \$SMINIT to \$INITIAL, using the following \$DISKUT1 utility command:

```
RE $SMINIT $INITIAL
```

The module \$INITIAL, if present, is part of the IPL stream and is automatically loaded.

If you do not wish the automatic start feature, you can load the session manager using the following command:

```
$L $SMMAIN
```

To begin a session, enter your user ID (one to four characters) and press the ENTER key. Use the ENTER key throughout the session for all data entry operations from the terminal. The user ID should be unique and not be used by more than one person simultaneously. This is because the user ID is used to create temporary work data sets whose names contain the four-character ID. Multiple use of the same ID results in sharing the same work data sets, with unpredictable results.

Note: If you do not wish to use the session manager, enter the word LOGOFF instead of a user ID and the session manager terminates.

SESSION MANAGER PROGRAM FUNCTION KEYS

Four program function (PF) keys enable you to perform the following functions:

PF1 Blanks the current screen image and allows system commands to be entered through prompting messages:

ENTERING SYSTEM COMMAND MODE -
TO REENTER THE SESSION MANAGER,
DEPRESS ATTN KEY AND ENTER '\$SM':

- PF2 Restores the current screen image to its appearance when first displayed. Use this key to erase erroneous entries.
- PF3 Returns the previously displayed screen image. Use this key to back out from the current screen image.
- PF4 Returns you to the primary option menu. Use this key to return from any session manager screen.

AUTOMATIC DATA SET ALLOCATION/DELETION

The session manager allocates and deletes data sets at logon/logoff time.

Data Set Allocation

After you enter your user ID, the session manager allocates work data sets on a disk resident volume. The data set called \$SMALLOC controls the data sets that are to be allocated. Figure 2 on page 30 lists the data set contents which consists of the data set prefix names, the sizes in 256-byte records, and the volumes to be used.

The END statement indicates the end of the list of data sets to be allocated. Six data sets are usually allocated, five of which are temporary. Temporary data sets are deleted at the end of the session manager session. The data sets are allocated on volume EDX003. Figure 3 on page 31 lists the data sets, their sizes, and their functions.

The data sets to be allocated, their volume, or size can be changed by using \$FSEEDIT or \$EDITIN to edit \$SMALLOC. Four other optional data sets (\$SM4, \$SM5, \$SM6, and \$SM7) can also be allocated. To change the number of data sets to be allocated, move the END statement behind the last data set to be allocated. To change a data set size or volume, change the size or

Session Manager

volume field.

The only required data sets are \$SMP and \$SMW. They must be allocated on volume EDX003.

For each user ID, a permanent parameter data set named \$SMPuser is created. The last input parameters entered on a parameter selection menu are displayed, allowing them to be used as is or changed. They are saved in the data set and recalled on the next invocation of the menu.

When the data sets have been allocated, the primary option menu is displayed on the screen and you can select the option desired.

```

$SMP      00      EDX003      NAME AND VOLUME FOR OPEN
$SMP      30      EDX003      SIZE AND VOLUME TO ALLOCATE
$SMW      30      EDX003      SIZE AND VOLUME TO ALLOCATE
$SME      400     EDX003      SIZE AND VOLUME TO ALLOCATE
$SM1      400     EDX003      SIZE AND VOLUME TO ALLOCATE
$SM2      400     EDX003      SIZE AND VOLUME TO ALLOCATE
$SM3      250     EDX003      SIZE AND VOLUME TO ALLOCATE
END      *** TERMINATOR - INDICATES END OF ALLOCATED
          DATASETS ***
$SM4      100     EDX003      SIZE AND VOLUME TO ALLOCATE
$SM5      100     EDX003      SIZE AND VOLUME TO ALLOCATE
$SM6      100     EDX003      SIZE AND VOLUME TO ALLOCATE
$SM7      100     EDX003      SIZE AND VOLUME TO ALLOCATE
*****
*****
** $SMLOG WORK DATASET PARAMETER VALUES FOR ALLOCATE **
** FUNCTION **
** NOTE: THE DATASETS $SMW AND $SMP MUST RESIDE ON **
** THE VOLUME EDX003. ALL OTHERS MAY BE **
** REASSIGNED. **
** NOTE: THE FIRST ENTRY IN THIS LIST IS USED TO **
** TEST FOR THE EXISTENCE OF THE $SMP **
** DATASET. DON'T DELETE. **
** DC C'5719-XS1 COPYRIGHT IBM CORP 1979' **
*****
*****
END
*****

```

Figure 2. \$SMALLOC contents

Data set name	Size in 256 byte blocks	Functional usage
\$SMEuser	400	Used by the full screen text editor utility (\$FSEEDIT) as a work data set.
\$SMPuser	30	Used by the session manager to save your input parameters from session to session. This data set is not deleted at the completion of a session.
\$SMWuser	30	Used by the session manager to submit procedures via the job stream utility (\$JOBUTIL).
\$SM1user*	400	Used by the linkage editor (\$LINK), the assembler (\$S1ASM), the compilers (\$EDXASM, COBOL, PL/I, and FORTRAN IV as a work data set.
\$SM2user*	400	Used by the linkage editor (\$LINK), the Series/1 Macro Assembler (\$S1ASM), COBOL, FORTRAN IV, and PL/I as a work data set.
\$SM3user*	250	Used by the Series/1 Macro Assembler (\$S1ASM), COBOL, and PL/I as a work data set.

Notes:

- 'user' in the data set name is replaced by your user ID.
- *Using the session manager to invoke \$S1ASM, COBOL, and PL/I requires that these data sets be deleted and reallocated. Recommended sizes for most programs are 2000 records for \$SM1user and \$SM2user and 800 record for \$SM3user.

Figure 3. Data sets created by the Session Manager

Session Manager

Data Set Deletion

Data sets created by the session manager can be deleted when you return to the logon menu. A prompt message is issued asking if you wish to save the data sets. To reply, enter a Y for yes or an N for no and press the ENTER key. Abnormal termination of the session manager prevents the deletion of the temporary data sets.

The data set \$SMDELETE controls the data sets to be deleted at the end of the session. The data set contains the data set prefix names and the volumes on which they reside. The END statement indicates the last data set to be deleted. The data sets to be deleted should normally be the data sets that were allocated at the start of the session. Figure 4 lists the contents of the \$SMDELETE data set.

```

$SME          EDX003    PREFIX NAME AND VOLUME TO DELETE
$SM1          EDX003    PREFIX NAME AND VOLUME TO DELETE
$SM2          EDX003    PREFIX NAME AND VOLUME TO DELETE
$SM3          EDX003    PREFIX NAME AND VOLUME TO DELETE
$SMW          EDX003    PREFIX NAME AND VOLUME TO DELETE
END          *** TERMINATOR - INDICATES END OF DATA SETS TO
              BE DELETED ***
$SM4          EDX003    PREFIX NAME AND VOLUME TO DELETE
$SM5          EDX003    PREFIX NAME AND VOLUME TO DELETE
$SM6          EDX003    PREFIX NAME AND VOLUME TO DELETE
$SM7          EDX003    PREFIX NAME AND VOLUME TO DELETE
*****
** $SMEND WORK DATASET PARAMETER VALUES FOR DELETE **
** FUNCTION **
** DC C'5719-XS1 COPYRIGHT IBM CORP 1979' **
*****
** END **
*****
```

Figure 4. \$SMDELETE contents

SESSION MANAGER MENUS

The session manager menus enable you to select options, such as text editing or program preparation, and to enter parameters. A unique procedure, associated with each menu, enables you to select option menus or to invoke through the batch job processing utility, \$JOBUTIL, the functions you desire.

Menus and procedures are stored in a library that resides on a direct access storage device. A menu is either a parameter selection menu or an option selection menu. You use a parameter selection menu to pass parameters to the program being invoked. The option selection menu is used to select other menus based on which option is selected.

When you log on to the session manager, the supplied environment can be overridden by specifying a main option menu that you have created. This provides for additional environments tailored to your system. Option selection menus can be modified to add options that provide different environments for different users. (Procedures for modifying and adding new menus are discussed in the System Guide).

Figure 5 on page 34 shows the structure and the various paths that can be used to execute a requested function. When a function completes, the most recently displayed menu is again displayed. You can then change parameters and again request the function, or you can return to previously displayed menus by pressing the PF3 key. Use the PF3 key to return to a previous menu and, finally, to exit the session manager.

Session Manager

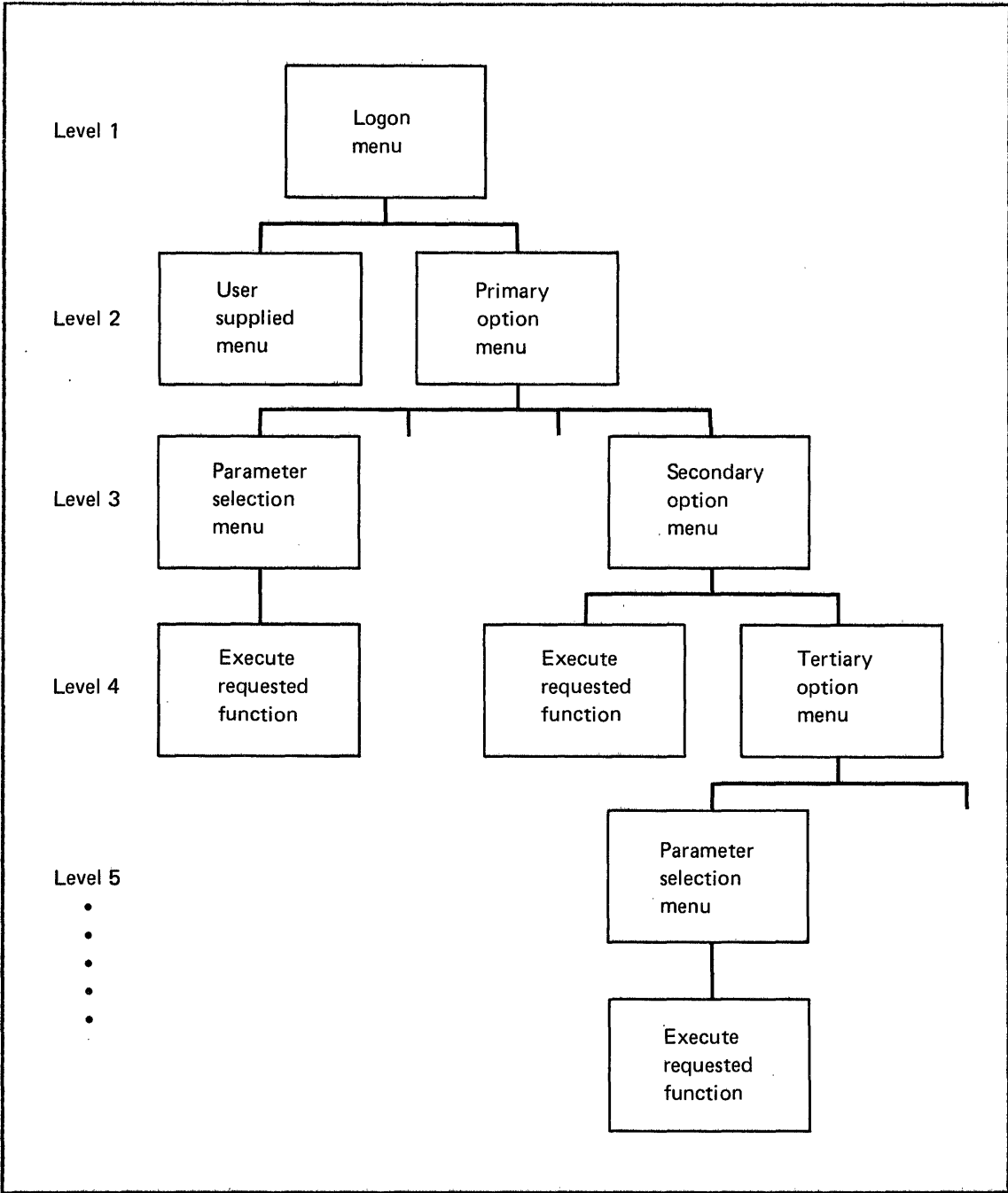


Figure 5. Menu selection hierarchy

Primary Option Menu

The primary option menu provides nine options as shown in Figure 6. The session manager automatically positions the cursor to accept input. Type in the desired option number and press the ENTER key. The function of each primary option is discussed in the following pages.

```

$SMMPRIM: SESSION MANAGER PRIMARY OPTION MENU -----
ENTER/SELECT PARAMETERS:          DEPRESS PF3 TO RETURN
                                   19:42:07
                                   07/13/79
                                   USER

      SELECT OPTION ==>

      1 - TEXT EDITING
      2 - PROGRAM PREPARATION
      3 - DATA MANAGEMENT UTILITIES
      4 - TERMINAL UTILITIES
      5 - GRAPHICS UTILITIES
      6 - EXEC PROGRAM/UTILITY
      7 - EXEC $JOBUTIL PROC
      8 - COMMUNICATION UTILITIES
      9 - DIAGNOSTIC UTILITIES

```

Figure 6. Primary option menu

Session Manager

Secondary Option Menus

Option 1 - Text Editing

This option requires no further parameter input. When you select this option, the session manager loads the full screen text editor utility program \$FSEDIT and passes control to it. The edit work data set used is the one automatically preallocated by the Session Manager. All further communication is directly between you and \$FSEDIT. Commands for \$FSEDIT can be found under "Primary Commands" on page 218.

When \$FSEDIT terminates, control returns to the session manager, the primary option menu is displayed, and another option can be selected.

Option 2 - Program Preparation

This option allows you to prepare programs for execution. By further specification on a secondary option menu, programs can be assembled, compiled, linked, updated, or listed. A secondary option menu, as shown in Figure 7 on page 37, is displayed on the screen after you select the program preparation utilities option from the primary option menu. You can select one of the listed utility programs from it.

A parameter selection menu is displayed based on which secondary option you select. The menu allows entry of required parameters such as: source input data set name, object output data set name, and assembler or compiler options. The selected parameters are saved from one session logon to the next. An example of the \$EDXASM parameter menu is shown in Figure 8 on page 38. Parameter menus for other program preparation utilities are similar and self explanatory. The required work data set for assemblers, compilers, and the linkage editor are preallocated by the session manager. You must allocate the object data set.

The following programs can be invoked from this option:

\$EDXASM	Compiles Event Driven Language programs
\$S1ASM	Assembles macro assembler language programs
\$COBOL	Compiles COBOL programs

\$FORT Compiles FORTRAN IV programs
 \$LINK Link-edits program object modules
 \$UPDATE Converts an object program to an executable program
 \$UPDATEH Transmits and converts host assembled modules
 \$PREFIND Locates data sets/overlays prior to loading a program
 \$PL/I Compiles PL/I programs

Option 9 of the program preparation utilities option menu invokes the utilities referenced in the sequence shown. You can pass parameters to those programs as needed.

```

$SMM02  SESSION MANAGER PROGRAM PREPARATION OPTION MENU--
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN

      SELECT OPTION ==>

          1 - $EDXASM COMPILER
          2 - $S1ASM ASSEMBLER
          3 - $COBOL COMPILER
          4 - $FORT FORTRAN COMPILER
          5 - $LINK LINKAGE EDITOR
          6 - $UPDATE
          7 - $UPDATEH (HOST)
          8 - $PREFIND
          9 - $EDXASM/$LINK/$UPDATE
         10 - $PL/I COMPILER/$LINK/$UPDATE
  
```

Figure 7. Program preparation utilities option menu

Session Manager

```
$SMM0201: SESSION MANAGER $EDXASM PARAMETER INPUT MENU  
ENTER/SELECT PARAMETERS:          DEPRESS PF3 TO RETURN
```

```
SOURCE INPUT  (NAME,VOLUME) ==>
```

```
OBJECT OUTPUT (NAME,VOLUME) ==>
```

```
ENTER OPTIONAL PARAMETERS  
BY POSITION==>
```

```
1-----2-----  
LIST      PRINTER NAME  
NOLIST  
ERRORS
```

```
DEFAULTS ARE: LIST $SYSPRTR
```

Figure 8. \$EDXASM parameter selection menu

Option 3 - Data Management

This option allows you to invoke the following utilities:

```
$DISKUT1  Allocates and deletes data sets; lists directory  
          data  
$DISKUT2  Patches and dumps data sets  
$COPYUT1  Copies data sets with output data set allocation  
$COMPRES  Compresses libraries  
$COPY     Copies disk/diskette data sets and volumes  
$DASDI    Initializes, formats, and verifies disks or  
          diskettes  
$INITDSK  Initializes and read-verifies a direct access  
          storage volume  
$MOVEVOL  Saves direct access volumes whose size requires  
          multiple diskettes
```

\$IAMUT1 Builds and maintains Indexed Access Method data sets

\$TAPEUT1 Allocates tape data sets, copies data sets or volumes from disk/diskette to tape, from tape to disk/diskette, or tape to tape, and changes tape attributes.

A secondary option menu, as shown in Figure 9, is displayed on the screen after you select the data management option from the primary option menu. You can select one of the listed utilities from it.

```

$SMM03 SESSION MANAGER DATA MANAGEMENT OPTION MENU-----
ENTER/SELECT PARAMETERS:                                DEPRESS PF3 TO RETURN
  
```

```

SELECT OPTION ==>
  
```

- 1 - \$DISKUT1 (ALLOCATE, LIST DIRECTORY)
- 2 - \$DISKUT2 (DUMP/LIST DATA SETS)
- 3 - \$COPYUT1 (COPY DATASETS/VOLUMES)
- 4 - \$COMPRES (COMPRESS A VOLUME)
- 5 - \$COPY (COPY DISK/DISKETTE DATASETS/VOLUMES)
- 6 - \$DASDI (DISK(ETTE) SURFACE INITIALIZATION)
- 7 - \$INITDSK (INITIALIZE/VERIFY DISK/DISKETTES)
- 8 - \$MOVEVOL (COPY DISK VOLUME TO MULTI-DISKETTES)
- 9 - \$IAMUT1 (MAINTAIN INDEXED DATA SETS)
- 10 - \$TAPEUT1 (TAPE ALLOCATE, CHANGE, COPY)

```

WHEN ENTERING THESE UTILITIES, THE USER IS EXPECTED
TO ENTER A COMMAND. IF A QUESTION MARK (?) IS ENTERED
INSTEAD OF A COMMAND, THE USER WILL BE PRESENTED WITH
A LIST OF AVAILABLE COMMANDS.
  
```

Figure 9. Data management option menu

The \$TAPEUT1 utility has been added to the secondary option menu of the Data Management utilities for the Event Driven Executive Version 2 (5719-UT4).

Session Manager

Option 4 - Terminal Utilities

This option allows you to invoke the following utility programs:

- \$TERMUT1 Alters logical device names, address assignments, or terminal configuration parameters
- \$TERMUT2 Defines routines and changes key definitions on the 4978 keyboard. Restores the 4974 printer to the standard character set.
- \$TERMUT3 Sends a single line message from one terminal to another
- \$IMAGE Defines formatted screen images for the 4978 or 4979 display terminals
- \$FONT Modifies character image tables for the 4978 terminal
- \$PFMAP Identifies program function keys on the 4978 terminal

A secondary option menu, as shown in Figure 10 on page 41, is displayed on the screen after you select the terminal utilities option from the primary option menu. You can select one of the listed utility programs from it.

Option 5 - Graphics Utilities

This option allows you to invoke utilities to generate, store, and display information graphically or in reports. The following utility programs can be invoked from this option:

- \$DIUTIL Performs utility functions on a display data base
- \$DICOMP Composes existing display profiles and adds new ones
- \$DIINTR Generates the requested display

A secondary option menu, as shown in Figure 11 on page 41, is displayed on the screen after you select the graphic utilities option from the primary option menu. You can select one of the listed utility programs from it.

```

$SMM04 SESSION MANAGER TERMINAL UTILITIES OPTION MENU---
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN

```

```

SELECT OPTION ==>

```

- 1 - \$TERMUT1 (TERMINAL CONFIGURATOR)
- 2 - \$TERMUT2 (4978 KEYBOARD DEFINE)
- 3 - \$TERMUT3 (TERMINAL MESSAGE SENDER)
- 4 - \$IMAGE (SCREEN FORMAT BUILDER)
- 5 - \$FONT (CREATE/MODIFY CHARACTER IMAGES)
- 6 - \$PFMAP (DISPLAY PF KEY CODES)

WHEN ENTERING THESE UTILITIES, THE USER IS EXPECTED TO ENTER A COMMAND. IF A QUESTION MARK (?) IS ENTERED INSTEAD OF A COMMAND, THE USER WILL BE PRESENTED WITH A LIST OF AVAILABLE COMMANDS.

Figure 10. Terminal utilities option menu

```

$SMM05 SESSION MANAGER GRAPHICS UTILITIES OPTION MENU--
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN

```

```

SELECT OPTION ==>

```

- 1 - \$DIUTIL (GRAPHICS ORGANIZOR)
- 2 - \$DICOMP (GRAPHICS COMPOSER)
- 3 - \$DIINTR (GRAPHICS INTERPRETER)

WHEN ENTERING THESE UTILITIES, THE USER IS EXPECTED TO ENTER A COMMAND. IF A QUESTION MARK (?) IS ENTERED INSTEAD OF A COMMAND, THE USER WILL BE PRESENTED WITH A LIST OF AVAILABLE COMMANDS.

Figure 11. Graphics utilities option menu

Option 6 - Execute Program

This option allows you to execute any program. The program can be a system program or utility, or one of your application programs. The parameter selection menu has provision for specifying the parameters and data sets that the program may need.

Session Manager

Option 7 - Execute \$JOBUTIL Procedure

This option allows the submission of previously built procedures to the job stream processor utility \$JOBUTIL. A parameter selection menu allows entry of the procedure name and volume name it resides on. The procedure cannot invoke the name of another procedure.

Option 8 - Communications Utilities

This option allows you to invoke the following utility programs:

- \$BSCTRCE Traces the I/O activities on a given binary synchronous communications line
- \$BSCUT1 Formats binary synchronous trace files to either a printer or a terminal
- \$BSCUT2 Exercises BSCAM capabilities
- \$RJE2780 Simulates a 2780 RJE interface
- \$RJE3780 Simulates a 3780 RJE interface
- \$PRT2780 Prints spool records produced by \$RJE2780
- \$PRT3780 Prints spool records produced by \$RJE3780
- \$HCFUT1 Interacts with the Host Communications Facility

A secondary option menu, as shown in Figure 12 on page 43 is displayed on the screen after you select the communications utility option from the primary option menu. You can select one of the listed utility programs from it.

```

$SMM08  SESSION MANAGER COMMUNICATION UTILITIES OPTION
        MENU
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN

SELECT OPTION ==>

      1 - $BSCTRACE (TRACE BSCAM LINES)
      2 - $BSCUT1  (PRINT TRACE FILE)
      3 - $BSCUT2  (BSC EXERCISER)
      4 - $RJE2780 (2780 RJE TO HOST)
      5 - $RJE3780 (3780 RJE TO HOST)
      6 - $PRT2780 (2780 SPOOLED RJE FILE PRINTER)
      7 - $PRT3780 (3780 SPOOLED RJE FILE PRINTER)
      8 - $HCFUT1  (HOST COMMUNICATIONS FACILITY)

WHEN ENTERING THESE UTILITIES, THE USER IS EXPECTED
TO ENTER A COMMAND. IF A QUESTION MARK (?) IS ENTERED
INSTEAD OF A COMMAND, THE USER WILL BE PRESENTED WITH
A LIST OF AVAILABLE COMMANDS.

```

Figure 12. Communications utilities option menu

Option 9 - Diagnostic Utilities

This option allows you to invoke the following utility programs:

\$TRAP Saves the environment in a data set in case of error

\$DUMP Formats and displays the data saved by \$TRAP

\$LOG Logs I/O errors into a data set

\$DISKUT2 Formats and displays the log data set on a printer or terminal

\$IOTEST Tests sensor-based operations; lists system configuration and volumes

A secondary option menu, as shown in Figure 13 on page 44 is displayed on the screen after you select the diagnostic utility option from the primary option menu. You can select one of the listed utility programs from it.

Session Manager

```

$SMM09  SESSION MANAGER DIAGNOSTIC AIDS OPTION MENU --
ENTER/SELECT PARAMETERS:          DEPRESS PF3 TO RETURN

```

```

SELECT OPTION ==>

```

- 1 - \$TRAP (CAPTURE PROGRAM INFORMATION TO
DATA SET)
- 2 - \$DUMP (FORMATTED STORAGE/REGISTER
- 3 - \$LOG (I/O ERROR LOGGING)
- 4 - \$DISKUT2 (DUMP/PATCH DISK(ETTE) UTILITY
- 5 - \$IOTEST (SENSOR I/O DEVICE EXERCISOR)

```

WHEN ENTERING THESE UTILITIES, THE USER IS EXPECTED
TO ENTER A COMMAND. IF A QUESTION MARK (?) IS ENTERED
INSTEAD OF A COMMAND, THE USER WILL BE PRESENTED WITH
A LIST OF AVAILABLE COMMANDS.

```

Figure 13. Diagnostic aids utilities option menu

A parameter selection menu, as shown in Figure 14 is displayed on the screen after you select the \$TRAP utility. Use the menu to enter the name of the data set to be used by \$TRAP and the name of the volume containing the data set.

```

$SMM0901: SESSION MANAGER $TRAP PARAMETER INPUT MENU -
ENTER/SELECT PARAMETERS:          DEPRESS PF3 TO RETURN

```

```

$TRAP

```

```

DUMP DATASET (NAME,VOLUME)==>

```

Figure 14. \$TRAP parameter selection menu

A parameter selection menu, as shown in Figure 15 on page 45 is displayed on the screen after you select the \$DUMP utility. Use the menu to enter the name of the data set to be used for the dump and the name of the volume containing the data set.

```
⌘SMM0902: SESSION MANAGER ⌘DUMP PARAMETER INPUT MENU ---  
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN
```

```
⌘DUMP
```

```
DUMP DATASET (NAME,VOLUME)==>
```

Figure 15. ⌘DUMP parameter selection menu

A parameter selection menu, as shown in Figure 16 is displayed on the screen after you select the ⌘LOG utility. Use the menu to enter the name of the data set to be used by ⌘LOG and the name of the volume containing the data set.

```
⌘SMM0903: SESSION MANAGER ⌘LOG PARAMETER INPUT MENU --  
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN
```

```
⌘LOG
```

```
LOG DATASET (NAME,VOLUME)==>
```

Figure 16. ⌘LOG parameter selection menu

Session Manager

Menu Option Combinations

When selecting a primary option, the secondary option, if applicable, can also be entered. This results in the secondary option being invoked without display of the secondary option menu. The secondary option is separated from the primary option by a period.

Example: To request an assembly from the primary option menu, enter 2.1 as the selected option. You have selected primary option 2 (program preparation) and its secondary option 1 (\$EDXASM - compile Event Driven Language program). The next menu displayed is the parameter selection menu (\$SMM0201) for the compiler \$EDXASM.

Utilities Not Supported by Session Manager Menus

The following utility programs are not supported by the Session Manager menus:

- \$DEBUG
- \$DISKUT3
- \$EDIT1
- \$EDIT1N
- \$EDXLIST
- \$PDS
- The Multiple Terminal Manager utilities
- The Remote Management Utility (\$RMU)

With the exception of \$DISKUT3 and \$PDS, these utility programs can be invoked by entering the operator command \$L (Load a Program) and the utility name. \$DISKUT3 and \$PDS can only be used from a program using the LOAD instruction and are documented in the System Guide. \$RMU is documented in the Communications and Terminal Applications Guide.

CHAPTER 4. THE UTILITIES

The utilities are a set of programs supplied with the Event Driven Executive system that allow you to interactively communicate with the system and perform many functions necessary for Series/1 application program development and system maintenance.

INVOKING THE UTILITIES

To aid you in performing utility functions, the Event Driven Executive system provides three ways to invoke the utility programs from a terminal:

- The session manager - You choose the desired utility program from a predefined option menu provided. Most utilities can be invoked in this manner. This is the easiest to use for interactive utilities because you need only enter option numbers (not program names) to access the function needed.
- \$JOBUTIL - The job stream processor utility can be used to invoke a predefined sequence of program preparation utilities and pass parameters to them. \$JOBUTIL can itself be invoked by the session manager.
- \$L command - Enter the operator command \$L (Load program), followed by the name of the utility of your choice. All utilities described in this book can be invoked in this manner.

Most utility programs are used interactively from a terminal. After a utility program is invoked, you can list its defined operations and command codes by entering a question mark in response to the 'COMMAND (?):' prompt.

Utilities

SUGGESTED UTILITY USAGE TABLE

The following table is intended to help you find the appropriate utility program and command to perform the function that you want to accomplish. To use it, find the activity and function that you want to do in the left columns; the corresponding utility and command to accomplish the function are in the right columns. The program name indicated can be used on the \$L command to load that utility program. When using the session manager, the menu option corresponding to the program name on the secondary option menu can be selected to access the program. The command indicated is used to direct the utility to perform the desired function.

ACTIVITY	FUNCTION	UTILITY	COMMAND
Activate	Error logging	\$LOG	\$LOGON
	Stopped task	\$DEBUG	GO
	TRAP function of storage dump	\$TRAP	TRAPON
Allocate	Data set	\$DISKUT1 \$DIUTIL \$IAMUT1 \$TAPEUT1	AL AL CR TA
	A new data set in a data base for a graphics display profile	\$DICOMP	AD
Assign	Alternate for defective 4963 sector	\$DASDI	3,1
	DEFINE key in a 4978 control store	\$TERMUT2	AD
Cancel	Dump	\$DUMP	CA
	List option	\$FSEDIT	CA
	Multiple member copy	\$COPYUT1	CA
	Multiple member list	\$DISKUT1	CA
	Multiple record dump	\$DISKUT2	CA
	Multiple record list	\$DISKUT2	CA
Change	\$EDXASM assembly of EDL	\$EDXASM	CA
	Address assignment of a terminal	\$TERMUT1	RA
	Base address	\$DEBUG	QUALIFY
	Execution sequence	\$DEBUG	GOTO
	Graphic or report display profile	\$DICOMP	AL,IN
	Hardcopy device associated with 4978/4979	\$TERMUT1	RH
	Host library	\$UPDATEH	CH
	Key definition in a 4978 control store	\$TERMUT2	C
	Name of logical device	\$TERMUT1	RE
	Page formatting parameters of a terminal	\$TERMUT1	CT

Utilities

ACTIVITY	FUNCTION	UTILITY	COMMAND	
Change	Series/1 library	\$UPDATEH	CV	
	Tape drive attributes	\$TAPEUT1	CT	
	Trap function of storage dump	\$TRAP	\$STRAPEND	
	Volume		\$DISKUT1	CV
			\$DISKUT2	CV
			\$UPDATE	CV
Clear	Data set to zero	\$DISKUT2	CD	
Compress	Graphics data base	\$DIUTIL	CP	
	Library	\$COMPRES	None	
Copy	All of data set to existing data set	\$COPY	CD	
	Part of data set to existing data set	\$COPY	CD	
	All of disk or diskette data set to tape data set	\$TAPEUT1	CD	
	A tape data set to disk or diskette data set	\$TAPEUT1	CD	
	A tape data set to another tape data set	\$TAPEUT1	CD	
	Absolute	\$COPY	CD	
	Entire disk or diskette volume	\$COPY	CV	
	Member with allocation	\$COPYUT1	CM	
	Member from graphic source data base to target data base	\$DIUTIL	CM	
	All members with allocation	\$COPYUT1	CALL	
	Data members with allocation	\$COPYUT1	CAD	
	Program members with allocation	\$COPYUT1	CAP	
	Generic members with allocation	\$COPYUT1	CG	
	Non-generic members with allocation	\$COPYUT1	CNG	

ACTIVITY	FUNCTION	UTILITY	COMMAND
Define	Indexed Access Method data set	\$IAMUT1	DF
	Formatted screen:		
	- image dimensions	\$IMAGE	DIMS
	- horizontal tab	\$IMAGE	HTAB
	- vertical tab	\$IMAGE	VTAB
	- null representation	\$IMAGE	NULL
Delete	Member	\$DISKUT1 \$DIUTIL	DE DE
	Prefound status of data sets and overlays	\$PREFIND	DE
Direct	Output to \$SYSPRNT	\$DISKUT1	LISTP
		\$DISKUT2	PL
	Output to terminal	\$DISKUT1	LISTT
		\$DISKUT2	LL
Display	4978 program function keys	\$PFMAP	Each key
	Character image table	\$FONT	DISP
	Contents of storage or registers	\$DEBUG	LIST
	Graphics or report display profile	\$DIINTR	None
	Header of a data member in graphics data base	\$DIUTIL	LH
	Indexed Access Method SE command parameters	\$IAMUT1	DI
	Members in graphics data base	\$DIUTIL	LA
	Status of all tasks	\$DEBUG	WHERE
	Status of current graphics data base	\$DIUTIL	ST
	Volume information	\$IOTEST	VI

Utilities

ACTIVITY	FUNCTION	PROGRAM	COMMAND
Dump	Data set or program on printer	\$DISKUT2 \$DICOMP \$DUMP	DP PR None
	Data set or program on console	\$DISKUT2 \$DICOMP \$DUMP	DU PR None
	Tape on printer or terminal	\$TAPEUT1	DP
Exercise	Tape to verify it is executing correctly and surface is free of defects	\$TAPEUT1	EX
Generate	Graphic or report display profile	\$DIINTR	None
Help	With use of \$COPY	\$COPY	?
	With use of \$COPYUT1	\$COPYUT1	?
	With use of \$DICOMP	\$DICOMP	?
	With use of \$DISKUT1	\$DISKUT1	?
	With use of \$DISKUT2	\$DISKUT2	?
	With use of \$DIUTIL	\$DIUTIL	?
	With use of \$FONT	\$FONT	?
	With use of \$FSEDIT	\$FSEDIT	?
	With use of \$IAMUT1	\$IAMUT1	?
	With use of \$IMAGE	\$IMAGE	?
	With use of \$INITDSK	\$INITDSK	?
	With use of \$IOTEST	\$IOTEST	?
	With use of \$PREFIND	\$PREFIND	?
	With use of \$TAPEUT1	\$TAPEUT1	?
	With use of \$TERMUT1	\$TERMUT1	?
With use of \$TERMUT2	\$TERMUT2	?	
With use of \$UPDATE	\$UPDATE	?	
With use of \$UPDATEH	\$UPDATEH	?	

ACTIVITY	FUNCTION	UTILITY	COMMAND	
Initialize	Disk	\$DASDI	2	
	Diskette	\$DASDI	1	
	For use with Event Driven Executive	\$INITDSK	I	
	Graphics data base	\$DIUTIL	IN	
	Log data set	\$LOG	\$LOGINIT	
	Tape	\$TAPEUT1	IT	
List	All members in volume	\$DISKUT1	LA	
	All members (CTS mode)	\$DISKUT1	LACTS	
	Breakpoints/trace ranges	\$DEBUG	BP	
	Data members (CTS mode)	\$DISKUT1	LDCTS	
	Data members in volume	\$DISKUT1	LD	
	Data set on printer	\$DISKUT2	LP	
	Data set on console	\$DISKUT2	LU	
	Devices supported by system	\$IOTEST	LS	
	Hardware configuration	\$IOTEST	LD	
	Log on printer	\$DISKUT2	PL	
	Log on console	\$DISKUT2	LL	
	One member	\$DISKUT1	LM	
	Program function keys	\$IMAGE	KEYS	
	Program members (CTS mode)	\$DISKUT1	LPCTS	
	Program members in volume	\$DISKUT1	LP	
	Space available	\$DISKUT1	LS	
	Terminal names/types/addresses	\$TERMUT1	LA	
	Through volumes	\$DISKUT1	LV	
	Load	A 4978 control store from a direct access data set	\$TERMUT2	LC
		Character image table into a 4978	\$FONT	PUT
Indexed Access Method data set		\$IAMUT1	LO	
Log	I/O errors into disk or diskette data set	\$LOG	\$LOGON	

Utilities

ACTIVITY	FUNCTION	UTILITY	COMMAND
Move	Graphics data base from one volume to another	\$DIUTIL	MD
	Tape forward or back	\$TAPEUT1	MT
Patch	Data set or program	\$DISKUT2	PA
	Storage or registers	\$DEBUG	PATCH
Preallocate	Data sets and overlays	\$PREFIND	PF
Post	An event or process interrupt	\$DEBUG	POST
Print	Map of 4963 alternate sectors	\$DASDI	3,3
Pulse	A digital output address	\$IOTEST	PD
Read	Character image table from 4978	\$FONT	GET
	Analog input	\$IOTEST	AI
	Program	\$UPDATE	RP
		\$UPDATEH	RP
	Digital input	\$IOTEST	DI
	Digital input using external sync	\$IOTEST	XI
Rename	Member	\$DISKUT1	RE
	Display profile identification name	\$DIUTIL	RE
Remove	Breakpoints/trace ranges	\$DEBUG	OFF
Reset	Indexed Access Method ECHO mode	\$IAMUT1	EC
	Indexed Access Method SE command parameters	\$IAMUT1	RE
Reorganize	Indexed Access Method data set	\$IAMUT1	RO

ACTIVITY	FUNCTION	UTILITY	COMMAND
Restore	Assigned 4963 alternate sector	\$DASDI	3,2
	Disk volume from diskettes	\$MOVEVOL	None
	Disk or diskette volume from tape	\$TAPEUT1	RT
	4974 printer to the standard character set	\$TERMUT2	RE
Rewind	Tape	\$TAPEUT1	MT
	Tape and place offline	\$TAPEUT1	MT
Save	A 4978 control store into a direct access data set	\$TERMUT2	SC
	Character image table	\$FONT	SAVE
	Contents of hardware registers and main storage in disk or diskette data set	\$STRAP	None
	Disk volume on diskettes	\$MOVEVOL	None
	Disk or diskette volumes on tape	\$TAPEUT1	ST
	Formatted screen image on disk	\$IMAGE	SAVE
Send	A single line message to another terminal	\$TERMUT3	None
Set	Breakpoints/trace ranges	\$DEBUG	AT
	Indexed Access Method ECHO mode	\$IAMUT1	EC
	Indexed Access Method SE command parameters	\$IAMUT1	SE
	Tape offline	\$TAPEUT1	MT
Terminate	Logging	\$LOG	\$LOGTERM
Test	Generated report or graphic display profile member	\$DICOMP	TD
	Process interrupt for occurrence of an event	\$IOTEST	PI,SG,SB

Utilities

ACTIVITY	FUNCTION	UTILITY	COMMAND
Unload	Indexed Access Method data set	\$IAMUT1	UN
Verify	Readability of disk or diskette data set	\$INITDSK	V
	Tape executing correctly	\$TAPEUT1	EX
	Tape surface free of defects	\$TAPEUT1	EX
Write	A hex value to a DO address	\$IOTEST	DO
	Digital output using external sync	\$IOTEST	XO
	Tapemark	\$TAPEUT1	MT

\$COMPRES - COMPRESS LIBRARY

\$COMPRES compresses a library on disk or diskette. It is used so that new data sets can be allocated when a library is fragmented (due to deletion of data sets).

Caution:

- Do not compress a library while it is being accessed.
- You must IPL the system after using \$COMPRES if the volume that was compressed contains the supervisor (\$EDXNUC).

The following is an example of using the \$COMPRES utility. In response to the prompting message >, the \$A operator command determines if another program is active and the \$L operator command invokes the \$COMPRES utility.

\$COMPRES

Example

> \$A

PROGRAMS AT 08:14:19
IN PARTITION #5
NONE

> \$L

PGM(NAME,VOLUME): \$COMPRES
\$COMPRES 11P,14.00.35, LP=4C00

COMPRESS SYSTEM LIBRARY
WARNING! SHOULD BE RUN ONLY WHEN
NO OTHER PROGRAMS ARE ACTIVE

VOLUME LABEL = EDX001

COMPRESS LIBRARY ON EDX001? Y

\$EDXNUC COPIED
\$INITIAL COPIED
\$UPDATE COPIED
\$COMPRES COPIED
\$DISKUT1 COPIED
\$DISKUT2 COPIED
\$COPY COPIED
LIBRARY COMPRESSED

ANOTHER VOLUME? N
\$COMPRES ENDED AT 14.05.03

\$COPY - COPY DATA SET

\$COPY copies a disk or diskette data set, in part or in its entirety, to another disk or diskette data set. Source and target data sets must have the same organization; they need not be on the same disk or diskette drive or volume. You can specify the number of records of the source data set to copy and the relative starting record in each data set to begin copying. All target data sets must have been preallocated using \$DISKUT1. Partial copy is only permitted if the data set organization is of type DO (Direct Organization, see \$DISKUT1).

Note: For any copying related to tape, see "\$TAPEUT1 - Tape Management" on page 311

\$COPY Commands

The commands available under \$COPY are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

```

COMMAND (?): ?

CD - COPY DATA SET
CV - COPY VOLUME
RE - COPY FROM BASIC EXCHANGE
WE - COPY TO BASIC EXCHANGE
    (-CA- WILL CANCEL)
EN - END PROGRAM
COMMAND (?):
  
```

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, CD).

Absolute Record Copy

An absolute record copying function is also provided as part of the Copy Data (CD) command. See the data set naming conventions in the System Guide for a description of the special data set names, \$\$EDXVOL and \$\$EDXLIB, which are used when doing absolute record and basic exchange copying.

SCOPY

Volume Copy

SCOPY can be used to copy an entire disk or diskette volume for the purpose of creating new or backup volumes. See the System Guide for an explanation of disk and diskette organization.

Basic Exchange Diskette Data Set Copy

SCOPY can be used to read a basic exchange data set from a diskette and write it to a disk data set, or write a disk data set to a basic exchange data set on diskette. All data sets must exist before the copy operation. Only one sided, 128-byte diskettes can be used. The Event Driven Executive recognizes only one volume on a diskette.

SCOPY prompts you for the name of the basic exchange volume. The volume specified must already be varied online.

You are prompted for the diskette basic exchange data set name. If you use \$DASDI to format and initialize the basic exchange header on the diskette, a data set named DATA will be automatically allocated. If you use \$INITDSK to initialize the volume directory, the data set name becomes EDXLIB. DATA consists of all the data tracks on the diskette. A record size of 128 bytes must be specified.

Notes:

1. Errors may occur if the diskette contains uninitialized HDR1s. Data on the diskette is read and written two sectors per I/O operation.
2. The diskette data set accessed must start on an odd sector boundary.

You can control the location in the disk data set where data is read or written by entering the starting and ending record numbers when prompted for that information.

The following error message is issued if the output data set is too small to accommodate the amount of data to be copied from the input data set.

DATA SET TOO SMALL

When the output data set is on disk or diskette, the end of data pointers are updated.

ExamplesCD - Copy Entire Data Set

```
COMMAND (?): CD
SOURCE (NAME,VOLUME): DATAFIL1
COPY ENTIRE DATA SET? Y
TARGET (NAME,VOLUME): DATAFIL1,EDX002
ARE ALL PARAMETERS CORRECT? Y
COPY COMPLETE
    50 RECORDS COPIED

COMMAND (?):
```

CD - Copy Where Receiving Member Is Too Small

```
COMMAND (?): CD
SOURCE (NAME,VOLUME): DATAFIL1
COPY ENTIRE DATA SET? Y
TARGET (NAME,VOLUME): DATAFIL2

***TARGET DATA SET TOO SMALL***
***COPY REQUEST CANCELLED***
```

Note: No data is copied in this case

SCOPY

CD - Copy A Partial Data Set

```
COMMAND (?): CD
SOURCE (NAME,VOLUME): DATAFIL1
COPY ENTIRE DATA SET? N
FIRST RECORD: 1
LAST RECORD: 3
TARGET (NAME,VOLUME): DATAFIL2
FIRST RECORD: 1
ARE ALL PARAMETERS CORRECT? Y
COPY COMPLETE
    3 RECORDS COPIED

COMMAND (?):
```

CV - Copy A Diskette To A Backup Data Set On 4962 Disk

```
COMMAND (?): CV

COPY VOLUME
ENTER SOURCE VOLUME: EDX001
ENTER TARGET VOLUME: EDX002
ENTER TARGET DATA SET NAME - EDX001
ARE ALL PARAMETERS CORRECT? Y
COPY COMPLETE
    949 RECORDS COPIED

COMMAND (?):
```

The CV command copies the entire diskette volume. Therefore the target data set on the 4962 disk must equal the diskette size in records; 949 records for Diskette1, 1924 records for Diskette2. Use \$DISKUT1 to allocate the data set on the disk.

RE - Copy A Basic Exchange Diskette Data Set To Disk

```

COMMAND (?): RE

SOURCE ($$EDXVOL,VOLUME): $$EDXVOL,BASIC
TARGET (NAME,VOLUME): DATAFIL1,EDX002

SPECIFY START/END? Y/N: N

ENTER BASIC EXCHANGE DATA SET NAME: DATA
NUMBER OF RECORDS COPIED: 52
COPY COMPLETED

COMMAND (?):
    
```

WE - Copy A Disk Data Set To A Basic Exchange Diskette

```

COMMAND (?): WE

SOURCE (NAME,VOLUME): DATAFIL1,EDX002

SPECIFY START/END? Y/N: N
TARGET ($$EDXVOL,VOLUME): $$EDXVOL,BASIC

ENTER BASIC EXCHANGE DATA SET NAME: DATA

COPY COMPLETED

COMMAND (?):
    
```

\$COPYUT1

\$COPYUT1 - COPY DATA SET WITH ALLOCATION

\$COPYUT1 performs several related copy functions. These functions determine the size and organization of the source data set to be copied, allocates a member on the target volume, and then copies the source member to the target member.

Caution: If a member already exists on the target volume, it is deleted, reallocated, and the new source copied to the target volume. There are no prompting messages asking if you wish to replace the existing member.

Note: For any copying related to tape, see "\$TAPEUT1 - Tape Management" on page 311

\$COPYUT1 Commands

The commands available under \$COPYUT1 are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

```
COMMAND (?): ?
```

```
CM---COPY MEMBER FROM SOURCE TO TARGET  
----- MULTIPLE COPY COMMANDS-----  
CALL---COPY ALL MEMBERS FROM SOURCE TO TARGET  
CAD---COPY ALL DATA MEMBERS FROM SOURCE TO TARGET  
CAP---COPY ALL PROGRAMS FROM SOURCE TO TARGET  
CG----COPY ALL MEMBERS STARTING WITH TEXT FROM ...  
CNG---COPY ALL MEMBERS NOT STARTING WITH TEXT FROM ...  
-----END OF MULTIPLE COPY COMMANDS-----  
SQ----SET PROMPT MODE FOR ALL MULTIPLE COPY COMMANDS  
NQ----RESET PROMPT MODE FOR ALL MULTIPLE COPY COMMANDS  
--CA-- WILL CANCEL MULTIPLE COPY COMMANDS  
CV---CHANGE SOURCE AND TARGET VOLUMES  
EN---END PROGRAM  
? ---HELP
```

```
COMMAND (?):
```

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, CM).

The following commands modify the way the multiple copy commands work; if needed, they must be entered before you start a multiple copy function.

SQ You are asked if you want to copy the current member.

NQ No questions are asked. All matched members are copied. This is the default.

The following keyboard function is invoked with the ATTENTION key. (It is not a command.)

CA If entered, CA stops the multiple copy after the current member is copied.

When \$COPYUT1 is loaded, the source and target volumes are set to the IPL volume. You can then change the source and target volumes. Once the volumes are set, the copy commands copy members from the source volume to the target volume until you do a CV to change a volume.

The CG (copy generic) and CNG (copy not generic) commands prompt you for a text string. The source volume directory is then searched for names beginning with this text string. Use CG to copy only those members beginning with the text string. Use CNG to copy only those members that do not begin with the text string.

If the multiple copy commands stop because the target volume is full, you can mount a new volume and continue the copy. Thus, you can create a disk backup spanning several diskettes. The actual copy process may take longer than with the utility \$MOVEVOL, but may use fewer diskettes as only members are copied. In addition, single and double-sided diskettes can be intermixed.

Since \$COPYUT1 deletes a member if it already exists, the multiple copy functions run faster if the target volume does not contain the same member names. If you are creating a new volume, use \$INITDSK to start with an empty target volume.

The multiple copy commands will not copy the supervisor (\$EDXNUC). This prevents the inadvertent loss of a tailored supervisor. Furthermore, since the supervisor is allocated when the disk is initialized, the CM command will not allocate \$EDXNUC on the target volume. It will copy \$EDXNUC from source to target but only if the size of \$EDXNUC on the target is the same size as on the source.

SCOPYUT1

No absolute record copy from disk or diskette is provided. Therefore the special names \$\$, \$\$EDXLIB, \$\$EDXVOL are not allowed. The \$COPY utility provides an absolute copy by record number.

Example

```
> $L $COPYUT1
$COPYUT1      35P,11:16:57, LP= 6900

                ** WARNING **
MEMBERS ON TARGET VOLUME WILL BE DELETED.
REALLOCATION AND COPYING OF MEMBERS IS
DEPENDENT ON SUFFICIENT CONTIGUOUS SPACE.

THE DEFINED SOURCE VOLUME IS EDX003, OK ? Y
THE DEFINED TARGET VOLUME IS EDX003, OK ? N
ENTER NEW TARGET VOLUME: MIKES
MEMBER WILL BE COPIED FROM EDX003 TO MIKES OK?: Y
COMMAND (?): CM
ENTER FROM(SOURCE) MEMBER: COFFEE
ENTER TO (TARGET) MEMBER OR * FOR SAME NAME AS SOURCE:GO
GO      COPY COMPLETE           4 RECORDS COPIED
COMMAND (?): CM LEM *
LEM     COPY COMPLETE           10 RECORDS COPIED
COMMAND (?): CG

ENTER GENERIC TEXT: MIKE
MIKEEDIT COPY COMPLETE           54 RECORDS COPIED
MIKEANL  COPY COMPLETE           13 RECORDS COPIED
MIKEDATA COPY COMPLETE           50 RECORDS COPIED

MIKENAME TOO LARGE TO COPY, ONLY 92 RECORDS LEFT IN LIB
TARGET VOL IS FULL,DO YOU WISH TO CONT ON A NEW VOL?: Y
MOUNT NEW VOLUME AND DO A $VARYON
THEN ENTER ATTN RESTART TO CONTINUE COPY
> $VARYON 2
EDX001 ONLINE
> RESTART

THE DEFINED TARGET VOLUME IS MIKES, OK ? Y
VOLUME NOT MOUNTED
ENTER NEW TARGET VOLUME: EDX001
MIKE1    COPY COMPLETE           100 RECORDS COPIED
COMMAND(?): SQ
COMMAND(?): CALL
COPY TEMP      ? Y
TEMP        COPY COMPLETE           40 RECORDS COPIED
COPY EDITWORK ? N
COPY DATAFILE ? Y
DATAFILE    COPY COMPLETE           110 RECORDS COPIED
COMMAND (?):
```


SDASDI

SDASDI - FORMAT DISK OR DISKETTE

SDASDI initializes your 4962 or 4963 disk or formats diskettes on the 4964 or 4966 diskette units. The utility can be used online. When this utility is invoked, you are prompted for one of the following disk or diskette initialization options:

- Option 1 - 4964, 4966 diskette initialization.
- Option 2 - 4962 disk initialization
- Option 3 - 4963 disk initialization

SDASDI must be loaded into partition 1.

Caution: For disk initialization, a program that accesses the disk being initialized should not be run concurrently with this utility.

Diskette initialization can run concurrently with other programs.

Option 1 - 4964, 4966 Diskette Initialization

Diskette Formats

The SDASDI utility reformats single and double-sided diskettes. Three formats are available:

1. Format for use with the Series/1 Event Driven Executive
2. Format to the IBM Standard for Information Interchange
3. Format entire diskette to 128, 256, or 512 byte records.

If you select the Event Driven Executive format, all tracks are formatted for 128 byte sectors. Also, cylinder 0 is formatted according to the IBM Standard for Information Interchange. The assigned volume label is IBMEDX.

Note: Use this format if all cylinders are to be formatted to 128-byte sectors.

If you initialize according to the IBM Standard for Information Interchange, Cylinder 0 is formatted for 128-byte sectors, and the remaining cylinders are formatted for either 128-, 256-, or

512-byte records. You are asked to select the desired size. The assigned volume label is IBMEDX.

When you initialize according to sector size, all cylinders are formatted to the size you select (128-, 256-, or 512-bytes). No volume labels, headers, or deleted records are written on Cylinder 0.

Caution: In this format, the diskette is not usable on an Event Driven Executive system except for reformatting.

Operating Characteristics

You are asked to identify the diskette drive by device address. The selected diskette drive is varied offline and a warning message issued to you before proceeding. On a 4966, diskette initialization can be performed only on slot 1. You are then prompted to select the options desired. Completion of the reformatting is indicated by the displayed messages:

'IBMEDX VARIED ONLINE'
'FORMATTING COMPLETE'

In the process of reformatting, a new volume label (IBMEDX) is written.

Caution: Do not use \$C to cancel formatting operation. Enter > \$DASDI to force termination.

SDASDI

Examples

Format Diskette for Event Driven Executive

SDASDI 15P,00:28:55, LP= 7E00

DIRECT ACCESS DEVICE INITIALIZATION

DISK INITIALIZATION OPTIONS:

- 1 = 4964, 4966 DISKETTE INITIALIZATION
- 2 = 4962 DISK INITIALIZATION
- 3 = 4963 DISK INITIALIZATION
- 4 = EXIT DISK INITIALIZATION

ENTER DISK INITIALIZATION OPTION: 1

```
*****
*          DISKETTE FORMATTING PROGRAM          *
* IF FORMATTING IS IN PROGRESS, DO NOT         *
* CANCEL (C) THIS PROGRAM. INSTEAD,           *
* ENTER ATTN/SDASDI TO FORCE TERMINATION.      *
*****
```

ENTER DISKETTE ADDRESS IN HEX 12

INITIALIZE FOR USAGE WITH THE EVENT DRIVEN EXECUTIVE? Y

EDX VARIED OFFLINE

** WARNING **

FORMATTING WILL DESTROY ALL DATA. CONTINUE? Y

IBMEDX VARIED ONLINE

FORMATTING COMPLETE

LOAD \$INITDSK? Y

\$INITDSK 16P,00:30:16, LP= 8D00

COMMAND (?): I

LIBRARY INITIALIZATION

1=ENTER VOLUME LABEL

2=ENTER DEVICE ADDRESS

SELECT OPTION: 2

ENTER DEVICE ADDRESS IN HEX: 12

Format Diskette for Event Driven Executive (continued)

```
WRITE VOLUME LABEL? Y

ENTER DESIRED VOLUME LABEL (1-6 CHARACTERS) EDX
ENTER OWNER ID (1-14 CHARACTERS): FCS
LABEL WRITTEN
CREATE A DIRECTORY? Y

HOW MANY RECORDS IN DIRECTORY? (2 - 120): 13
  MAXIMUM NO. OF MEMBERS = 102, OK? Y
DO YOU WISH TO RESERVE SPACE FOR A NUCLEUS? Y
ENTER MAXIMUM SIZE IN K-BYTES (16-64): 64
DIRECTORY INITIALIZED

WRITE IPL TEXT? Y
IPL TEXT WRITTEN

COMMAND (?): EN

$INITDSK ENDED AT 00:31:10

ANOTHER DISKETTE? N

$DASDI ENDED AT 00:31:15
```

SDASDI

Format Diskette to IBM Standards for Information Interchange

```
ENTER DISKETTE ADDRESS IN HEX 02
INITIALIZE FOR USAGE WITH THE EVENT DRIVEN EXECUTIVE? N
INITIALIZE TO STANDARDS FOR INFORMATION INTERCHANGE? Y
SELECT SECTOR SIZE (1=128, 2=256, 3=512): 2
NND002 VARIED OFFLINE
      ** WARNING **
FORMATTING WILL DESTROY ALL DATA. CONTINUE? Y
IBMEDX VARIED ONLINE
FORMATTING COMPLETE
ANOTHER DISKETTE? N
SDASDI ENDED AT 00:44:30
```

Format Diskette to 128-, 256-, or 512-Byte Records

```
ENTER DISKETTE ADDRESS IN HEX 02
INITIALIZE FOR USAGE WITH THE EVENT DRIVEN EXECUTIVE? N
INITIALIZE TO STANDARDS FOR INFORMATION INTERCHANGE? N
SELECT SECTOR SIZE (1=128, 2=256, 3=512): 3
      ** WARNING **
FORMATTING WILL DESTROY ALL DATA. CONTINUE? Y
FORMATTING COMPLETE
ANOTHER DISKETTE? N
SDASDI ENDED AT 01:01:27
```

Option 2 - 4962 Disk Initialization

The disk initialization utility for the 4962 initializes your disk, writes sector addresses on the entire volume, analyzes and locates defective sectors, and assigns alternate sectors. After initialization, the disk is ready for use with the Event Driven Executive. For a new disk device, initialization should be performed before the Event Driven Executive system is installed on it.

When using this option, you are required to select one of two initialization types:

- PI (primary) - initialize a disk for the first time or to completely reinitialize the disk.

Caution: This option rewrites the complete disk surface and destroys all data that may have been on the disk.

- AS (alternate sector assignment) - assign alternate sectors without destroying the data currently on the disk.

Note: Use the AS option only when necessary. Cylinder 1 has a limited number of available alternate sectors. Once an alternate sector is assigned, it can only be recovered by writing all sector IDs during a primary initialization.

The PI option verifies and corrects sector IDs and analyzes the disk surface to find defective sectors. If the programmer's console is present, the data buffer displays the number of the cylinder currently being initialized. If a defective sector is found, either on a movable or a fixed head, an alternate sector is assigned from Cylinder 1 and a message is issued by the utility. When an alternate sector is assigned, the sector ID of the defective sector references the location of its alternate on Cylinder 1. Defective sectors are marked defective. Skewed IDs are written where normal IDs fail. For a defective sector on Cylinder 0, an alternate good sector under the same head on Cylinder 0 is reassigned to the defective sector.

The AS option forces the assignment of alternate sectors without destruction of data on the disk. \$DASDI tries to move data from the defective sector to its assigned alternate. If data recovery fails, \$DASDI issues a message and the alternate data field is flagged with one bits (hexadecimal FFFF). If an already assigned alternate is found defective, it is marked defective and a new alternate is assigned. Data recovery is attempted in this case also.

\$DASDI

The number of alternate sectors available on Cylinder 1 depends on the 4962 model:

Model	Capacity	Alternates
1,1F,2,2F	9.3 MB	120
3,4	13.9 MB	180

Examples

Primary Initialization of a 4962 Disk

```

$DASDI I5P,00:28:55,LP=7E00

DIRECT ACCESS DEVICE INITIALIZATION
DISK INITIALIZATION OPTIONS:
  1 = 4964, 4966 DISKETTE INITIALIZATION
  2 = 4962 DISK INITIALIZATION
  3 = 4963 DISK INITIALIZATION
  4 = EXIT DISK INITIALIZATION
ENTER DISK INITIALIZATION OPTION: 2

*****
*                               WARNING                               *
*   NO USER PROGRAM SHOULD BE RUNNING                               *
*   WHILE PERFORMING DISK INITIALIZATION                             *
*****
DISK INITIALIZATION STARTED
ENTER DEVICE ADDRESS - NNN: 003

ENTER INITIALIZATION TYPE - PI OR AS: PI

ENTER INITIALIZATION MODE
REMOVE PREVIOUS ... DEFECTIVE SECTOR FLAGS? Y|N: NO

```

In the above example, you are prompted for the following:

- Disk or diskette initialization option: 1, 2, or 3
- Initialization type: PI for primary or AS for alternate sector
- Initialization mode:
 - NO - Retain defective flag byte of each sector ID.
 - YES - Rewrite sector flag IDs and reinitialize the flag byte where possible. Allows a disk with invalid sector flags to be initialized.

Caution: Respond YES only if you wish to rewrite all sector IDs. This causes the loss of any IBM factory assigned defective sector flags. If you respond YES, the following verify operation occurs:

\$DASDI

FACTORY MARKED DEFECTIVES MAY BE LOST
IS CHANGE OF REPLY DESIRED? Y|N: NO

NO Operation is to continue with flags considered invalid.

YES A reprompt of the previous message results, allowing you
to change the status of the defective flags.

The following message is repeated for each alternate sector
assignment, if any occur:

ALTERNATE SECTOR ASSIGNED FOR cchss

Note: cchss=the address of the alternate sector assigned.

Alternate Sector Assignment on a 4962 Disk

```

$DASDI I5P,00:28:55,LP=7E00

DIRECT ACCESS DEVICE INITIALIZATION
DISK INITIALIZATION OPTIONS:
  1 = 4964, 4966 DISKETTE INITIALIZATION
  2 = 4962 DISK INITIALIZATION
  3 = 4963 DISK INITIALIZATION
  4 = EXIT DISK INITIALIZATION
ENTER DISK INITIALIZATION OPTION: 2

*****
*                               WARNING                               *
*   NO USER PROGRAM SHOULD BE RUNNING                               *
*   WHILE PERFORMING DISK INITIALIZATION                             *
*****
DISK INITIALIZATION STARTED
ENTER DEVICE ADDRESS - NNN: 003

ENTER INITIALIZATION TYPE - PI OR AS: AS

ALTERNATE SECTOR MODE
ENTER SECTOR ADDRESS - CCHSS

```

The following message is displayed at your terminal indicating completion of the disk initialization.

```

ALTERNATE SECTOR ASSIGNED FOR CCHSS
DISK INITIALIZATION COMPLETE

$DASDI ENDED AT 00:31:15

```

ccchss: The address of the sector presumed to be defective. The utility assigns an alternate sector on Cylinder 1, then tries to move the data from the defective sector to the alternate sector. Alternates on Cylinder 0 are located on the same track and head as the defects on Cylinder 0. This process may reveal that the sector IDs on Cylinder 0 are in an inconsistent condition. Processing continues if possible. You cannot assign an alternate to a defective sector on Cylinder 1.

\$DASDI

Note: The fixed head area is always referred to as Cylinder 303. You should consider that this cylinder contains eight heads (zero through seven). To refer to sector five under fixed head four, you would specify 303405 for ccchss.

Option 3 - 4963 Disk Initialization

The \$DASDI utility program identifies and restores defective sectors on a 4963 disk device. The 4963 comes from the factory already formatted with all logical sector addresses assigned and tested and with alternates assigned to any defective sectors; you do not have to initialize a newly installed 4963.

With this function, you are given the option of:

- Identifying a specific sector as being defective, causing the utility to assign an alternate to it.
- Restoring a previously identified defective sector, causing the utility to free its alternate.
- Printing a map of all defective sectors, indicating if the defective sector was factory or user identified.

Alternate sectors are assigned as follows:

- If the primary alternate (the extra sector on the same track) is available, it is used as the alternate for the defective sector.
- If the primary alternate is not available (either it is defective or already assigned), a secondary alternate is assigned from the nearest track under the movable heads having an available primary alternate.

Note: The primary alternate under a fixed head is assigned to a sector that is under the same fixed head.

When restoring sectors from defective status, \$DASDI physically moves the sectors within the track to minimize the processing time between consecutive logical sectors. You cannot restore:

- A factory assigned defective sector
- A primary defect (one that cause the primary alternate sector for the track to be assigned)

- A sector whose ID has been written extended

Examples

Invoking 4963 Disk Initialization

```
$DASDI ISP,00:28:55,LP=7E00

DIRECT ACCESS DEVICE INITIALIZATION
DISK INITIALIZATION OPTIONS:
  1 = 4964, 4966 DISKETTE INITIALIZATION
  2 = 4962 DISK INITIALIZATION
  3 = 4963 DISK INITIALIZATION
  4 = EXIT DISK INITIALIZATION
ENTER DISK INITIALIZATION OPTION: 3
*****
*                               WARNING                               *
*   NO USER PROGRAM SHOULD BE RUNNING                               *
*   WHILE PERFORMING DISK INITIALIZATION                             *
*****

DISK INITIALIZATION STARTED
ENTER DEVICE ADDRESS - NNN: 048

THE AVAILABLE OPTIONS ARE:
1 - IDENTIFY DEFECTIVE SECTOR(S)
2 - RESTORE DEFECTIVE SECTOR(S)
3 - MAP DEFECTIVE SECTOR(S)
4 - EXIT UTILITY

ENTER OPTION: n
```

The entry n must be one of the four options listed in the command menu. You can choose to identify, restore, or map defective sectors. The utility terminates when you enter Option 4.

SDASDI

Obtaining Map of Defective 4963 Sectors

ENTER OPTION: 3

DEFECT ALTERNATE EXTENDED
000101
020114

Assigning an Alternate Sector

ENTER OPTION: 1

ENTER CCCHSS OF SECTOR TO BE MARKED
DEFECTIVE OR END: 0010205

ENTER 'Y' TO ASSIGN ALTERNATE,
ANYTHING ELSE TO CANCEL: Y

ccchss ASSIGNED ALTERNATE OF 0010205

ENTER CCCHSS OF SECTOR TO BE MARKED
DEFECTIVE OR END: END

ENTER OPTION:

Note: In the preceding example, the disk address for a 4963 is entered as a seven-digit number (0010205): the cylinder is 1 (001), the head is 2 (02), and the sector is 5 (05).

Restoring a Previously Assigned Alternate Sector

ENTER OPTION: 2

ENTER CCCHSS OF SECTOR TO BE
RESTORED OR END: 0010207

0010207 HAS BEEN RESTORED FROM ccchss

ENTER CCCHSS OF SECTOR TO BE
RESTORED OR END: END

ENTER OPTION: 4
DISK INITIALIZATION ENDED
\$DASDI ENDED AT 01:01:27

\$DEBUG

\$DEBUG - DEBUGGING TOOL

\$DEBUG is a tool for locating errors in programs. By operating a program under control of **\$DEBUG**, you can:

- Stop the program each time execution reaches any of one or more instruction addresses that you have specified. These addresses are known as breakpoints.
- List the contents of specified storage locations or register contents each time the program execution reaches one or more of your breakpoints.
- Trace the flow of execution of instructions within the program by specifying one or more ranges of instruction addresses (known as trace ranges). Each time the program executes an instruction within any of the specified trace ranges, the terminal displays a message identifying the task name and the instruction address just executed. Optionally, program execution can be stopped after each instruction is executed within a trace range. Also, optionally, storage locations or register contents can be displayed on the terminal after the execution of each instruction within a trace range.
- Restart program execution at the breakpoint or trace range address where it is currently stopped. Or, in the case of Event Driven Language instructions, restart program execution at other than the next instruction.
- List additional registers and storage location contents while the program is stopped at a breakpoint or at an instruction within a trace range.
- Patch the contents of storage locations and registers.

Using these functions, you can determine the results of computations performed by the program and the sequence of instruction execution within the program. You can also modify data or instructions of the program during execution.

To use **\$DEBUG** effectively, you must have a printed listing of the program to be debugged which shows the storage addresses of each instruction and data area of interest. To obtain such a listing, specify **PRINT GEN** in the source program, after the **PROGRAM** statement, at assembly time. A **PRINT NOGEN** should precede the **PROGRAM** statement to prevent the unnecessary printing of many system **EQU** statements, etc. For **\$EDXASM** a satisfactory listing is produced by specifying **LIST**.

Debug Usage Considerations

The program debug facility aids in testing multitasked programs in a multiprogramming and multiuser environment. All of your interactions are via terminals and do not require the use of the machine console. A summary of the major features of the \$DEBUG program follows:

Notes:

1. \$DEBUG should be invoked from a terminal other than the one used by the program to be tested if the program uses 4978/4979 terminals in STATIC screen mode.
2. Multiple breakpoints and trace ranges can be established.
3. Several users can each use separate copies of \$DEBUG concurrently, if sufficient storage is available.
4. Series/1 assembler language as well as Event Driven Language instructions can be traced and tested.
5. Both supervisor and application programs can be debugged.
6. Task names are automatically obtained from the program to be tested.
7. Task registers #1 and #2 can be displayed and modified.
8. Hardware registers R0 through R7 and the IAR can be displayed and modified.
9. Five different data formats are accepted by the list and patch functions.
10. No special preprocessing of a program is required to permit it to be debugged.
11. All address specifications are made as shown in the program assembly listing without concern for the actual memory addresses where the program is loaded into storage for testing.
12. No processing overhead is incurred unless the hardware trace feature is enabled. Even then, the hardware trace feature is only enabled for specific tasks.
13. The debug facility can be activated for a program that is experiencing problems but was previously loaded without the debug facility.

\$DEBUG

14. A program can be debugged by loading \$DEBUG from a terminal other than the one from which the program to be tested was loaded.
15. Breakpoints or trace ranges specified during a debug session can be listed.
16. \$DEBUG can only control the execution of programs containing no more than 20 tasks.

The \$DEBUG program can be used to test different types of programs. The most common usage is to debug application programs written using the Event Driven Language instruction set. However, it can also be used to test portions of application programs that are written in assembler language and portions of the supervisor program that are written in either Event Driven Language or Series/1 assembler language. Testing of the supervisor should normally be required only if you are making your own modifications or additions to this program.

You can use \$DEBUG to debug overlay programs by loading the primary program that will subsequently load the overlay program to be debugged. Load \$DEBUG after the overlay program is in storage. (For more information on debugging overlay programs that are part of the Event Driven Language compiler, \$EDXASM, refer to the Internal Design). To suspend execution of the overlay program so that \$DEBUG can be loaded, enter a READTEXT or QUESTION as the first instruction of the overlay program. Multiple Terminal Manager users should code a CALL ACTION instruction to provide the required function. When the overlay program is entered, it pauses at the first instruction and waits for input. At this point, load \$DEBUG. This can be done from another terminal assigned to the same partition. Specify the overlay program name when prompted for the program name and indicate that no new copy of the overlay program is to be loaded.

The \$DEBUG utility can then be used to set breakpoints and perform other functions as required. If the overlay program causes a program check, it is cancelled by the system. If an overlay program terminates through a PROGSTOP or for any other reason and is reloaded by the primary program, any breakpoints or patches made prior to the termination are lost.

Use of certain capabilities of \$DEBUG requires a thorough knowledge of both the supervisor and debugging techniques. For example, altering the contents of storage locations occupied by the supervisor or contents of the Series/1 hardware registers could have undesirable effects on the operation of the supervisor or application programs in operation concurrently with \$DEBUG.

Note: Only those instructions that execute as part of a task can be debugged. Those portions of the supervisor program that service interrupts created by various hardware devices (disk, timers, terminals, etc) cannot be executed under control of \$DEBUG.

Start and Termination Procedure

The primary method for activating the debug facility is to load \$DEBUG and then specify the name of the program to be tested, when prompted (DBUGDEMO in the following example). \$DEBUG then loads your program, inserts a breakpoint at the first executable instruction, and notifies you that your program is stopped at this point. For example:

```
> $L $DEBUG
$DEBUG      26P,09:10:17, LP=5200
PROGRAM NAME: DBUGDEMO
DBUGDEMO    4P,09:10:28, LP=6700
DBUGDEMO    STOPPED AT 009E
```

\$DEBUG

\$DEBUG Commands

The following commands are available:

AT	- Set breakpoints and trace ranges
BP	- List breakpoints and trace ranges thus far specified
END	- Terminate debug facility
GO	- Activate stopped task
GOTO	- Change execution sequence
HELP	- List debug commands
LIST	- Display storage or registers
OFF	- Remove breakpoints and trace ranges
PATCH	- Modify storage or registers
POST	- Post an event or process interrupt
QUALIFY	- Modify base address
WHERE	- Display status of all tasks

Command Entry

A command is entered by pressing the ATTENTION key on your terminal and entering the command name, or the command name plus the required parameters for the command, in response to the prompting message '>'.

Syntax Summary

In the command syntax examples and descriptions in the following sections, keyword parameters are capitalized and variable parameters are shown in lower case. Whenever one of several keywords can be chosen, these keywords are separated by a slash(/). The examples show the various formats of each command which are available for different purposes. Detailed syntax descriptions are presented under \$DEBUG Command Descriptions.

Set breakpoints and trace ranges:

AT ADDR address NOLIST/LIST NOSTOP/STOP
AT TASK taskname start-add end-add EDX/ASM NOLIST/LIST
NOSTOP/STOP
AT ALL NOLIST/LIST NOSTOP/STOP
AT * NOLIST/LIST NOSTOP/STOP

Terminate \$DEBUG:

END

Activate breakpoints or trace ranges:

GO ADDR address
GO TASK taskname start-add end-add
GO ALL
GO *

List \$DEBUG commands:

HELP

Display storage or registers:

LIST ADDR address length mode
LIST R0...R7 taskname length mode
LIST #1/#2 taskname length mode
LIST IAR taskname length mode
LIST *

Remove breakpoints or trace ranges:

OFF ADDR address
OFF TASK taskname start-add end-add
OFF ALL
OFF *

\$DEBUG

Modify storage or registers:

PATCH ADDR address length mode
PATCH R0...R7 taskname length mode
PATCH #1/#2 taskname length mode
PATCH IAR taskname length mode
PATCH *

Post events or process interrupts:

POST ADDR address code
POST PIXx code
POST *

Modify base address:

QUALIFY base disp
Q base displ

Display status of all tasks:

WHERE

Display breakpoints and trace ranges:

BP

Change execution sequence:

GOTO current-address new-address

Example: The command syntax permits most keyword parameters to be abbreviated to a single character, except ALL which conflicts with ADDR; entry of AL for ALL and A for ADDR is permitted. You are prompted for command parameters individually, unless you are sufficiently familiar with the syntax to enter a complete command on a single line. For example, to set the task TIMET into a program trace between addresses 0 and 3000 and also print the contents of both task registers in task LOOP2 in decimal mode but continue processing, the following interactive keyboard sequence may occur. Each response is terminated by a RETURN key entry.

```

> AT
OPTION(* / ADDR / TASK / ALL): TASK
TASK NAME: TIMET
FIRST ADDRESS: 0
LAST ADDRESS: 3000
TRACE TYPE(EDX / ASM): EDX
LIST / NOLIST: LIST
OPTION(* / ADDR / R0...R7 / #1 / #2 / IAR): #1
TASK NAME: LOOP2
LENGTH: 2
MODE(X / F / D / A / C): F
STOP / NOSTOP: NOSTOP
      1 BREAKPOINT(S) SET

```

Identical results are obtained by entering the single response:

```

> AT T TIMET 0 3000 E L #1 LOOP2 2 F N
      1 BREAKPOINT(S) SET

```

\$DEBUG

\$DEBUG Command Descriptions

AT - Set Breakpoints and Trace Ranges

AT sets breakpoints and trace ranges. The LIST and STOP options established for a breakpoint or trace range are executed prior to executing the instruction that satisfied the breakpoint or trace range specification. When the specification for a breakpoint or trace range is satisfied, the task currently active is detoured and \$DEBUG performs the following actions for the subject task: prints its status, prints the LIST specification, and optionally puts the task into a wait state. If the NOSTOP option was requested, task status is printed as "taskname CHECKED AT XXXX". The STOP option generates a "taskname STOPPED AT XXXX" message.

The LIST and STOP options for all currently defined breakpoints and trace ranges can be modified by entering AT ALL. Similarly, the specifications for the most recently entered AT command can be altered by the AT * option.

Notes:

1. You cannot set breakpoints in ATTNLIST routines.
2. You can only set breakpoints on EDL instructions within an EDL program.

Syntax

```
AT ADDR address NOLIST/LIST NOSTOP/STOP
AT TASK taskname start-add end-add EDX/ASM NOLIST/LIST
  NOSTOP/STOP
AT ALL NOLIST/LIST NOSTOP/STOP
AT *   NOLIST/LIST NOSTOP/STOP
```

<u>Operands</u>	<u>Description</u>
ADDR	Keyword indicating this is an instruction program breakpoint specification.
address	Instruction address where a breakpoint is to be inserted.

Caution: Be sure that this is the first word of an executable instruction, since the low-order byte of this word will be modified by \$DEBUG. Unpredictable results can occur if you specify the address of data or the address of other than the first word generated by an instruction.

- NOLIST** No special print request is needed at this breakpoint or trace range.
- LIST** Complete specification for a storage or register display. See the LIST command for a description of all list options and parameters.
- NOSTOP** Processing is to continue after the breakpoint notification has occurred.
- STOP** The task is to stop whenever this breakpoint or trace range specification is satisfied.
- TASK** Trace range specification.
- taskname** Name of task to be traced. If the program contains only one task, omit this parameter.
- start-add** Trace range starting address. Since your program is not actually modified by a trace specification, no special care needs to be exercised in entering trace addresses.
- end-add** Trace range ending address. To establish a breakpoint at an individual assembler instruction, specify both the starting and ending address to coincide with the instruction address.
- EDX** Only Event Driven Language instructions are to be traced.
- ASM** All Series/1 assembler instructions within the specified range are to be traced.
- ALL** All currently defined breakpoints and trace ranges are redefined with new list and stop options.
- *** The most recently defined breakpoint or trace range specification is to be redefined. This specification is determined by the last usage of an AT, GO, or OFF command.

\$DEBUG

BP - List Breakpoints and Trace Ranges

BP displays all breakpoints and trace ranges that have been specified during the current debug session. The information supplied for each breakpoint includes the task address, instruction type, associated list options, and whether a stop was specified.

Syntax

BP

No operands are required.

END - Terminate \$DEBUG

END removes all currently active breakpoints and trace ranges, activates all currently stopped tasks, and terminates \$DEBUG.

Syntax

END

No operands are required.

GO - Activate a Stopped Task

GO reactivates any task that has been stopped by \$DEBUG. If a task is stopped at a breakpoint, specify the exact breakpoint address. If a task is stopped as a result of a trace specification, supply the name of the task and an address range which brackets the addresses in the original trace request. If only one task is being debugged, no operands need be specified.

Syntax

```
GO ADDR address
GO TASK taskname start-add end-add
GO ALL
GO *
GO
```

<u>Operands</u>	<u>Description</u>
ADDR	Keyword indicating this is a breakpoint specification.
address	Instruction address where the task is stopped.
TASK	Keyword indicating this is a trace range specification.
taskname	Name of task to be activated. For programs containing only a single task, omit this parameter.
start-add	Trace range starting address.
end-add	Trace range ending address.
ALL	All currently stopped tasks are to be activated.
*	The most recently referenced breakpoint or trace range specification is to be used. This specification is determined by the last usage of an AT, GO, or OFF command.

\$DEBUG

GOTO - Change Execution Sequence

GOTO reactivates, at a different instruction, any task that has been stopped at an Event Driven Language instruction. If stopped using a breakpoint or trace, supply the current address and the address at which execution should be resumed. Breakpoint or trace specifications are not changed.

Syntax

```
GOTO current-address new-address
```

<u>Operands</u>	<u>Description</u>
-----------------	--------------------

current-address	
-----------------	--

	Address where the task is stopped.
--	------------------------------------

new-address	Address where execution is to be restarted.
-------------	---

HELP - List \$DEBUG Commands

The HELP command produces a list of \$DEBUG commands and functions.

Syntax

```
HELP
```

No operands are required.

LIST - Display Storage or Registers

LIST displays the contents of memory locations, or task registers, or hardware registers, or the IAR. The LSB can be displayed by listing the IAR with a length of 11 words. Any register data is only guaranteed to be current if the corresponding task is inactive or stopped by a \$DEBUG breakpoint or trace range. To repeat the most recently specified LIST command or to verify (list) a patch you have just entered, use "LIST *".

Syntax

```
LIST ADDR address          length mode
LIST R0/.../R7 taskname length mode
LIST #1/#2      taskname length mode
LIST IAR        taskname length mode
LIST *
```

Operands

Description

ADDR	Keyword indicating this is a display of a storage location.
address	Address of the storage location to be displayed.
length	Length of display in words, doublewords, or characters . depending on mode.
mode	Mode of data display: X - hexadecimal word F - decimal number(word) D - decimal number(doubleword) A - relocatable address C - EBCDIC character
R0/.../R7	One of the Series/1 hardware registers R0 through R7. To define the start of the LIST.
taskname	Name of task from which register data is to be displayed. For programs containing only a single task, omit this parameter.
#1/#2	Task register #1 or #2 specification.

\$DEBUG

IAR Keyword indicating the IAR (Instruction Address Register) is to be displayed.

* The most recently specified LIST specification is to be used. This is determined by the last usage of a LIST or PATCH command.

OFF - Remove Breakpoints and Trace Ranges

OFF removes a breakpoint or trace range established with the AT command. To remove a breakpoint, specify the exact breakpoint address. To remove a trace request, specify the name of the task and an address range which brackets the addresses in the original trace request. If a task is currently stopped at the requested breakpoint or trace range, this task is automatically reactivated.

Syntax

```

OFF ADDR address (breakpoints)
OFF TASK taskname start-add end-add (trace ranges)
OFF ALL
OFF *
    
```

<u>Operands</u>	<u>Description</u>
ADDR	Keyword indicating this is the removal of the breakpoint specification.
address	Instruction address where a breakpoint has previously been established.
TASK	Keyword indicating a trace range is to be removed.
taskname	Name of task associated with a trace range. For programs containing only a single task, omit this parameter.
start-add	Trace range starting address.
end-add	Trace range ending address.
ALL	All breakpoints and trace ranges are to be removed.
*	The most recently referenced breakpoint or trace range specification is to be used. This specification is determined by the last usage of an AT, GO, or OFF command.

\$DEBUG

PATCH - Modify Storage or Registers

PATCH modifies the contents of memory locations, task registers, hardware registers, and the IAR (Instruction Address Register). The entire LSB (Level Status Block) can be modified by patching the IAR with a length specification of 11 words. The patch to any register data is only guaranteed if the corresponding task is inactive or stopped by a \$DEBUG breakpoint or trace range. To respecify the data for the most recent patch or to patch the data displayed by the most recent LIST command, enter 'PATCH *'.

After the patch command is entered, the current memory or register content is displayed, and you are prompted for the patch data (a string of data entries that satisfy the length and mode specifications). The entries are separated by spaces, for example, data...data. After the patch data is entered, you can apply the patch by responding YES, abort by responding NO, or indicate additional patches with a CONTINUE reply to the prompting message. By specifying CONTINUE, the patch is performed and prompting continues for entry of new length, mode, and data specifications to memory or register locations immediately behind your previous patch.

If less data than specified with the length operand is entered, the effective patch is padded with blanks for character data and zeros for all other data types.

Syntax

```
PATCH ADDR address          length mode
PATCH R0/.../R7 taskname length mode
PATCH #1/#2      taskname length mode
PATCH IAR       taskname length mode
PATCH *
```

Operands

Description

ADDR	Keyword indicating this is a storage patch.
address	Address of the storage location to be modified.
length	Length of patch in words, doublewords, or characters depending on mode.

mode Mode of data entry:

- X - hexadecimal word
- F - decimal number(word)
- D - decimal number(doubleword)
- A - relocatable address
- C - EBCDIC character

R0/.../R7 One of the Series/1 hardware registers R0 through R7, where the patch is to be started.

taskname Name of task for which register data is to be modified. For programs containing only a single task, omit this parameter.

#1/#2 Task register #1 or #2 specification.

IAR Keyword indicating the IAR (Instruction Address Register) is to be modified.

***** The most recently referenced LIST or PATCH specification is to be used. This is determined by the last usage of a LIST or PATCH command.

\$DEBUG

POST - Post an Event or Process Interrupt

POST activates a task waiting for an event or a process interrupt. To duplicate a previous posting, enter POST *. The address of the ECB (Event Control Block) to be posted is contained in the second word of a WAIT command as shown on a program assembly listing. Process interrupts can also be posted by name, using the PIxx option.

Syntax

```
POST ADDR address code
POST PIxx code
POST *
```

<u>Operands</u>	<u>Description</u>
ADDR	The address of an ECB (Event Control Block).
address	ECB address to be posted.
code	Decimal code to be posted to the specified ECB.
PIxx	Name of process interrupt PI1 to PI99.
*	The most recently referenced ECB address or PIxx name and code specification is to be used.

QUALIFY - Modify Base Address

QUALIFY modifies the base address used by \$DEBUG to reference physical storage locations. This command is useful for debugging supervisor program modifications.

Syntax

```
QUALIFY base displ  
Q base displ
```

<u>Operands</u>	<u>Description</u>
base	New hexadecimal base address. Enter 0 when working with the supervisor.
displ	Hexadecimal offset to be added to base to form the new base address for all subsequent address references. Enter the origin of the supervisor program module as shown on the link editor listing.

\$DEBUG

WHERE - Display Status of All Tasks

WHERE displays the current status of each task. If a task is currently processing its breakpoint routine, it is marked CHECKED. If a task has been stopped by a breakpoint or trace request or if the main task has not yet been attached by \$DEBUG, the task is marked STOPPED. In all other cases, each task is shown to be at the currently executing instruction, at the command it will start executing when dispatched by the task supervisor, or at the last command executed prior to entering a wait state.

Syntax

WHERE

WH

No operands are required.

Tips and Techniques

Additional breakpoints and trace ranges that are desired should usually be established before initiating program execution with a GO ALL command. At program load time all task names are obtained by \$DEBUG, and the current status of each task can be requested by entering WHERE. For example:

```
> WHERE
DEBUGDEMO  STOPPED AT  009E
USERT      AT   02EE
LOOP3      AT   0230
LOOP2      AT   01C0
TIMET      AT   0124
```

Actual task names are only available if the program is loaded by \$DEBUG. If the program to be tested is already loaded and executing, task names have been destroyed, and artificial names are generated by \$DEBUG by appending the relative address (in hexadecimal) of the TCB (task control block) to TASK. For example:

```
> WHERE
TASK0308 AT  00BE
TASK02A0 AT  0300
TASK01E2 AT  0298
TASK0172 AT  01C0
TASK00D6 AT  0164
```

If the program to be debugged has previously been loaded into storage multiple times, the load points of all currently active copies are listed, and you have the option to specify which copy you wish to debug. You also have the option to request a fresh copy of the same program to be loaded. For example:

\$DEBUG

```
> $L $DEBUG
$DEBUG      21P,09:37:17, LP=C200
PROGRAM NAME: DBUGDEMO
ALREADY ACTIVE AT 5200 5600 5A00
DO YOU WANT A NEW COPY TO BE LOADED? N
PROGRAM LOAD POINT: 5600
```

or

```
DO YOU WANT A NEW COPY TO BE LOADED? Y
DBUGDEMO    4P,09:38:02, LP=5E00
```

When more than one disk/diskette drive is available, programs from a volume other than the IPL volume can be loaded. For example:

```
PROGRAM NAME: DBUGDEMO,EDX002
```

Notes:

- \$DEBUG is terminated by the command END. It cannot be ended by the supervisor utility function command \$C.
- When \$DEBUG is ended the program which was being tested under the control of \$DEBUG remains in storage with all of its tasks active and operating.

SDICOMP - DISPLAY COMPOSER

The composer allows you to add display profiles and modify existing display profiles. Through the use of interactive dialogue, the program guides you through the generation or alteration of a display profile. Because this program does not change the basic structure of the online data base, you can use it at the same time other functions are being performed. This program can be used to generate a portion of a new display profile. Then you can use \$DIINTR to cause the partial display to be generated. If corrections or additions are necessary, you can then use \$DICOMP to alter the display profile. These steps can be repeated until the desired results are obtained.

Invoking SDICOMP

To start execution of \$DICOMP:

1. Load the program \$DICOMP specifying the appropriate data set name. \$DIFILE, the online data set, or any other data set can be used. However, you should make sure that another user or program is not changing or using the same data set.
2. The system responds with the program loaded message followed by:

```
$DICOMP - DISPLAY DATA BASE COMPOSER  
COMMAND (?):
```

SDICOMP Commands

The commands available under \$DICOMP are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

SDICOMP

COMMAND (?): ?

AD - ADD A NEW MEMBER
AL - ALTER EXISTING MEMBER
EN - EXIT PROGRAM
IN - INSERT OR DELETE DISPLAY ELEMENTS IN A MEMBER
PR - PRINT MEMBER FORMATTED
TD - TEST DISPLAY

COMMAND (?):

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, AD).

AD - Add a New Member to Data Base

Allows you to generate a new display profile. The display can be in either graphic or report form. You are prompted to enter a 1-8 character Display Profile name used by the Interpreter to retrieve the display. The next prompt message asks if a display heading is desired. If so, a report display is assumed. If not, a graphic display is assumed and you are prompted to proceed with generating the display. You are also permitted to select the device to which the output from the Interpreter is to be routed.

Report Display: If the response to the question IS THIS A GRAPHIC DISPLAY? is NO, you are prompted to enter the column headings desired. One line up to 132 characters is allowed. Following the entry of the column headings, you are prompted to enter the name of the print report data member. You are then prompted to enter the next command.

Graphic Display: If a graphic display is desired, you should respond to the message IS THIS A GRAPHIC DISPLAY? by entering the character Y. The composer then asks if the display is to be a 3D object. If you respond with a character Y, then all following references to X and Y values will also include the Z value. The composer asks you to enter the values X, Y, and Z, if 3D object. They are used to position the first character of the display heading. You are then prompted with the request COMMAND (?):'. Subcommands should then be entered.

AL - Alter an Existing Member

This function allows you to display each element of a display profile and make changes, using subcommands, provided the size of the element and the sequence of commands is not changed. This function is of great value during the trial and error period when you are generating a new display. You can generate a display using the AD function and display the results using the Interpreter. You are allowed to either start alteration at the beginning of the member and display each element in turn or to skip to a specific element within it. Use the PR command to display the elements and their sequence numbers. As each element is displayed, you are questioned whether or not this element is to be altered.

ALTER ENTRY?

When Y is selected, you are prompted to reenter the element as described in the AD/Add Member/ section. When the end of the display profile is reached, the Composer is terminated and you can redisplay the profile to see if the corrections are satisfactory.

EN - Exit Program

This function provides an immediate exit from the Composer.

IN - Insert or Delete Elements in an Existing Member

This function combines the facilities of the Alter and Add functions with the ability to delete individual display elements. Because the Insert function creates a new member in the data base, the size and sequence of display elements can be changed.

Note: Use of the utility program to verify that sufficient space remains in the data base is recommended. By using the LA and ST functions, you can determine the size of the member to be modified and the remaining space in the data base. As described in the Alter function, the Composer displays each element in turn with the following questions asked:

SDICOMP

KEEP ENTRY?
DELETE ENTRY?
ALTER ENTRY?

If you elect to keep the entry, the Composer proceeds to the next element. If N is selected, the DELETE ENTRY? question is displayed. If N is entered, the ALTER ENTRY? question is displayed. If Y is entered, the Composer proceeds with the alteration process as described in the AL function description.

Following the Alteration function, control is returned to the Insert function and the process is repeated for the next element. If the element was not altered, you are prompted to insert a new subcommand. At this point, all the functions of the AD facility are available. You can add one display element after which control is returned to the Insert function where the previous element is redisplayed and the sequence is repeated.

Again, as in the Alter function, you must step through each element in the Display Profile before completion. When the end is reached, the message END OF DATA - ISSUE SAVE OR ADD NEW COMMANDS is displayed. The Composer then reverts to the Add function and additional commands can be entered.

Note: You must issue a Save command to terminate the Insert function. When the Save command is issued, the Composer deletes the old member and renames the newly built member with the old name. This procedure makes the modified version available to the Interpreter. You are urged to use the utility to compress the data base following insert activity to prevent fragmentation of the data base and reclaim unused space.

PR - Print Member Formatted

Display on the terminal or printer the contents of a display profile member formatted in the same way as used by the Alter and Insert functions. This display is useful as an aid in maintaining display profiles. Routing to \$SYSPRTR is allowed to provide a high speed hard copy.

TD - Test Display as Currently Entered

You are prompted for the name of a plot control member and then \$DIINTR is invoked to generate the specified display after which control is returned to you to make alterations.

Composer Subcommands

When adding, altering, or inserting elements in a member, subcommands are used. These are listed below and described on the following pages. When entered, subcommands are placed in or modify the member. The member can later be used by the interpreter to generate the desired display. The following subcommands are available:

```

AD - ADVANCE X,Y
DI - DIRECT OUTPUT
DR - DRAW A SYMBOL
EN - EXIT PROGRAM
EP - END DISPLAY
HX - DISPLAY HEX WORDS
IM - INSERT MEMBER
JP - JUMP TO ADDRESS
JR - JUMP REFERENCE
LB - DISPLAY CHARACTERS
LI - DRAW A LINE TO X,Y
LR - DRAW LINE RELATIVE
MP - MOVE BEAM TO X,Y
PC - PLOT CURVE ONLY
PL - PLOT DATA
RT - ACTIVATE REALTIME DATA MEMBER
SA - SAVE ACCUMULATED DATA
TD - DISPLAY TIME AND DATE
VA - DISPLAY VARIABLE
  
```

Most subcommands perform actions based on the X and Y coordinates of the viewing screen. It is important that you keep track of the current X and Y values as the display is developed. The suggested method to produce a graphic display is to first draw the display on graph paper and assign X and Y coordinates to the key nodes in the display. Then use this drawing as a guide to the generation of the display, keeping in mind the screen limits for the terminal to be used. The view area of the graphic terminals supported is shown in Figure 17 on page 110. Figure 18 on page 111 shows the space supported in 3D mode.

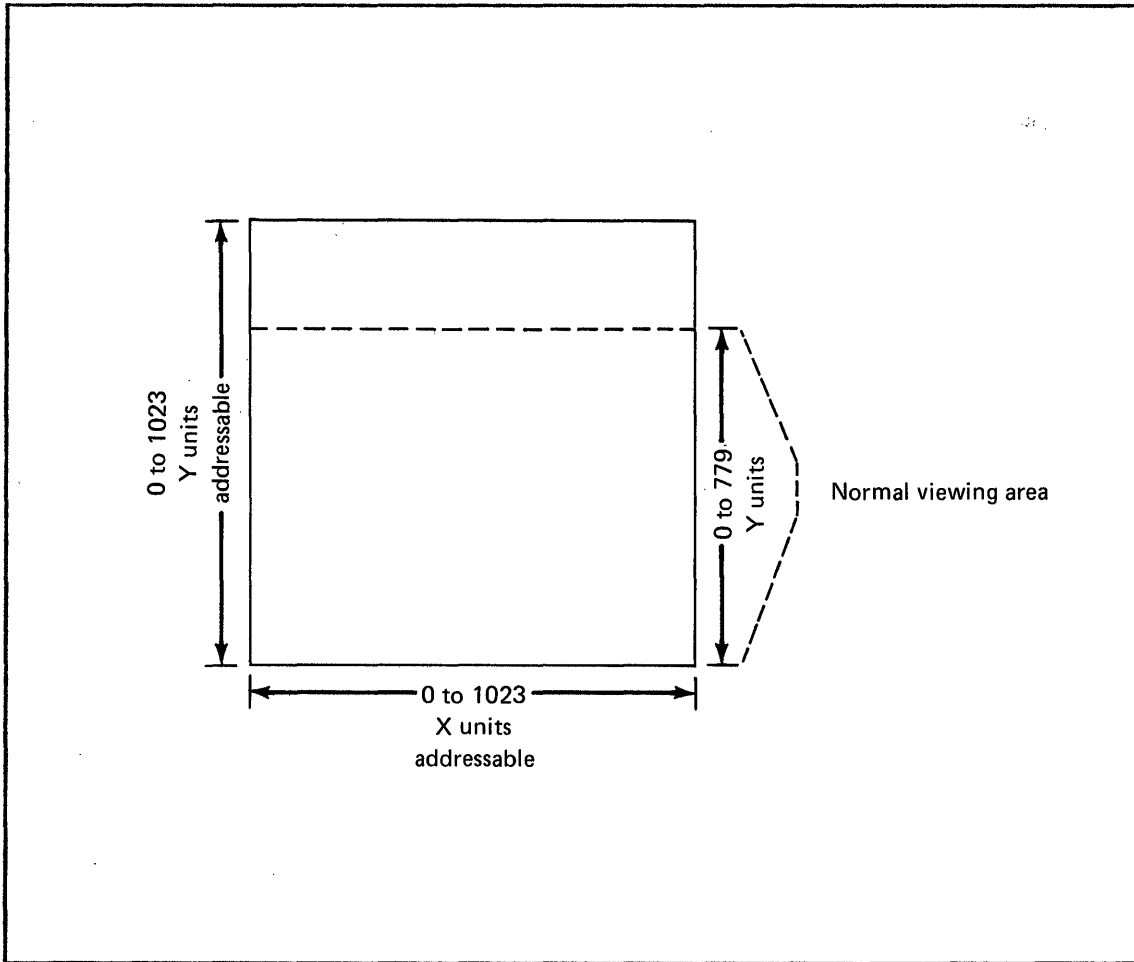


Figure 17. X,Y Coordinate Grid and Viewing Area

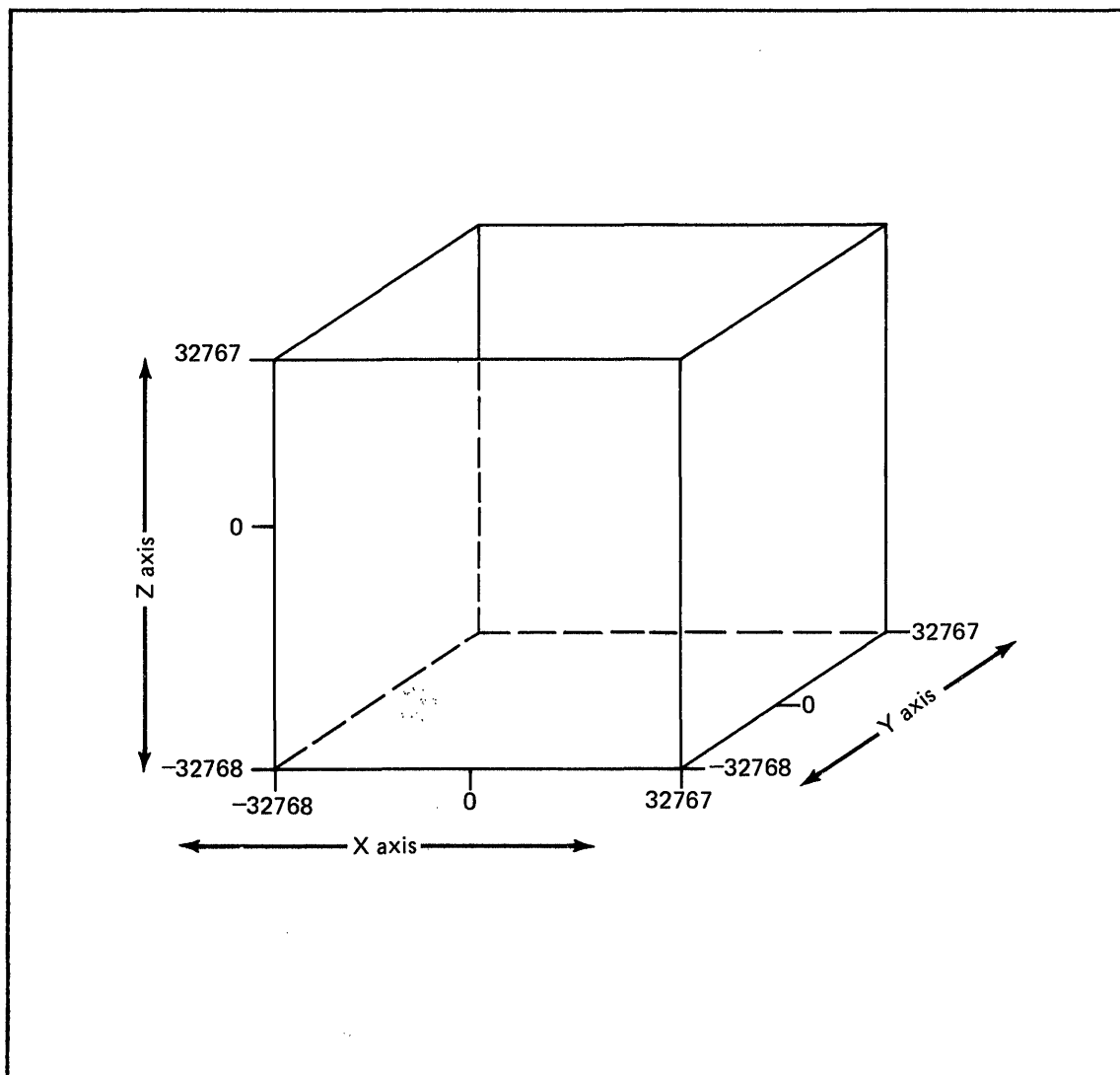


Figure 18. X,Y,Z Coordinate Grid and Viewing Area

AD - Advance X,Y

Moves the beam position by the specified value. This could be helpful in displaying data with even spacing on the screen. After issuing a 'DR' command using a symbol, AD could advance the X,Y position to the next position without regard to the actual screen X,Y location. The limit for the specified X or Y value is plus or minus 512 units. If a 3D object is being defined, then the Z axis value is also requested.

\$DICOMP

DI - Direct Output

Directs the resulting graphic output to appear at a terminal other than the one used to enter commands. The terminal name to be entered is the label of the TERMINAL statement used to describe the desired terminal.

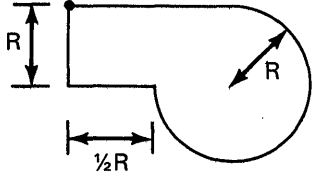
DR - Draw a Symbol

Draws a predefined symbol. Several commonly used symbols have been provided. In specifying a symbol, you are prompted to enter the symbol number and the symbol modifier. These values are used by the Interpreter to generate the requested symbol. Some of the symbols require additional information. If required, you are prompted for this additional information. Valid symbol numbers are 1 through 14.

Symbol # = 1 Draw Fan Symbol Left Hand Format

Modifier = Radius of fan body and opening on left side of fan. Must be a multiple of 4.

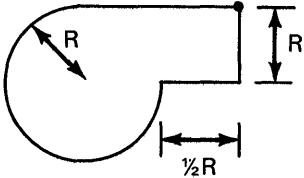
= start and end X,Y current position.



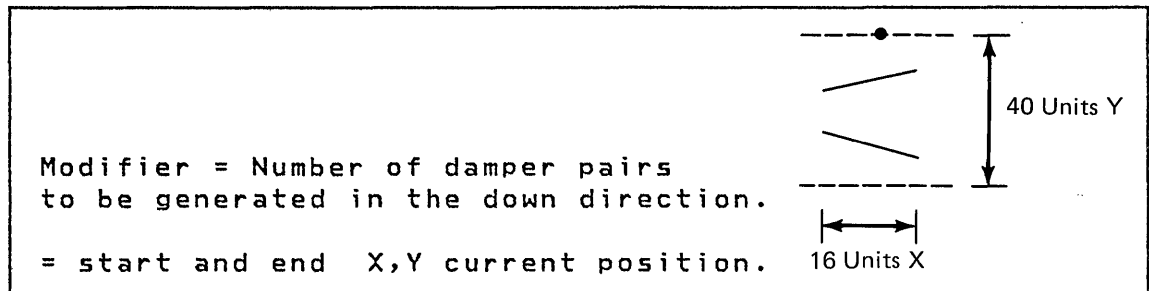
Symbol # = 2 Draw Fan Symbol Right Hand Format

Modifier = Radius of fan body and opening on right side of fan. Must be a multiple of 4.

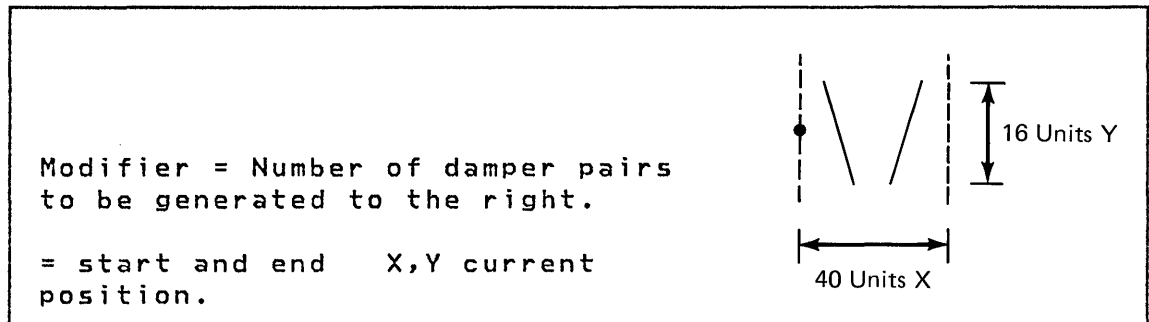
= start and end X,Y current position.



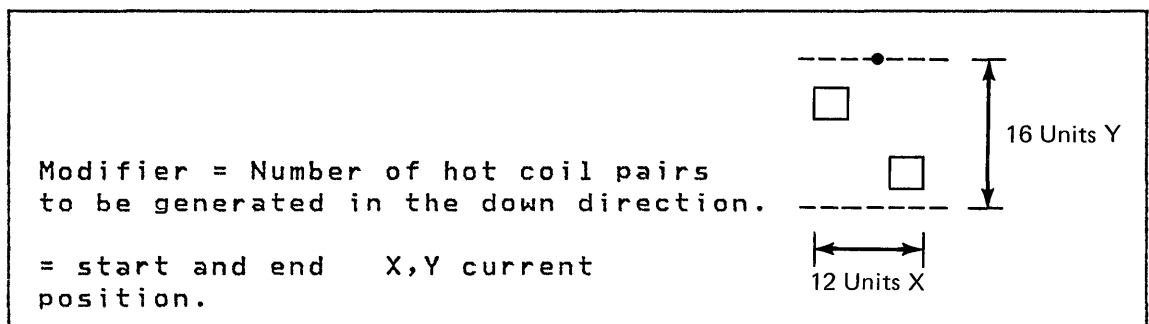
Symbol # = 3 Draw Damper Vertical



Symbol # = 4 Draw Damper Horizontal



Symbol # = 5 Draw a Hot Coil



\$DICOMP

Symbol # = 6 Draw a Cold Coil

Modifier = Number of double pairs to be generated in the down direction.

= start and end X,Y current position.

Symbol # = 7 Draw a Filter Element

Modifier = Number of elements to be generated in the down direction.

= start and end X,Y current position.

Symbol # = 8 Draw a Valve

For 2-way valve
Modifier = 2

For 3-way valve
Modifier = 3

= start and end X,Y current position.

Symbol # = 9 Draw An Arrow

Modifier =

- 1 for left
- 2 for right
- 3 for up
- 4 for down

= start and end X,Y current position.

Symbol # = 10 Draw a Logic Block Right

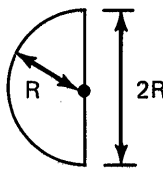
Modifier = Radius of Half Circle.
Must be a multiple of 4.

= start and end X,Y current position.

\$DICOMP

Symbol # = 11 Draw a Logic Block Left

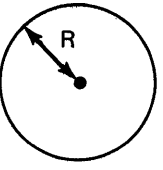
Modifier = Radius of half circle.
Must be a multiple of 4.
= start and end X,Y current position.



The diagram shows a half-circle on the left side of a vertical line. A horizontal line segment from the center of the vertical line to the edge of the half-circle is labeled 'R'. A vertical double-headed arrow to the right of the vertical line is labeled '2R', indicating the diameter of the full circle.

Symbol # = 12 Draw a Circle

Modifier = Radius of Circle.
Must be a multiple of 4.
= start and end X,Y current position.



The diagram shows a circle with a center point. A line segment from the center point to the edge of the circle is labeled 'R', representing the radius.

Symbol # = 13 Draw An Arc Right

Modifier = Radius of circle.
Must be a multiple of 4.

= start and end X,Y current position.

Note: This symbol requires additional values.

1. Draw arc up or down.
Enter zero for down or one for up.
2. Number of Y units to draw arc.
Must be a multiple of 4.



Note: This symbol always starts at X=0 and proceeds until the Y units have been exhausted.

Symbol # = 14 Draw An Arc Left

Modifier = Radius of circle.
Must be a multiple of 4.

= start and end. X,Y current position.

Note: See note under Symbol 13 for additional information.

**EN - Exit Program**

Causes the Composer to be terminated without updating the Display Profile Data Base Directory. All data collected up to this point for this member is lost.

\$DICOMP

The preceding list of available subcommands are those that are available when using the AD function. These descriptions are also valid when using the AL Alter function or the IN Insert function.

EP - End Display

Specifies that the end of this section of the display has been reached. Normally, this command is followed by a SA (Save) command. However, this command can be useful if a jump zero/not zero causes the Interpreter to take alternate paths. You can use the EP command at the end of each of these paths instead of an unconditional jump to a common ending point.

HX - Send Data

Sends to the terminal up to 16 words of data without conversion. All bit patterns are valid; therefore, control or special data can be sent to the terminal.

IM - Insert Member

Combines display profile members to form one display. This allows you to conserve disk space, decrease time required to enter display profiles, and standardize display formats. For example, you can build a display profile member to represent a common background of a physical system or floor plan. Then, by defining another display profile member, superimpose on the background the variables that will make the display unique. Only one level of nesting is permitted. That is, a member inserted using the 'IM' command cannot contain any 'IM' commands. However, a primary member can include multiple 'IM' commands.

JP - Jump to Address

Causes a change in the sequence of execution of subcommands. There are three types of Jump to Address subcommands that can be used. They are:

- Jump Unconditional
- Jump if Zero
- Jump if Not Zero

As described in the Display Variable command, the conditional jump commands are dependent on the use of the Realtime Data Member. If conditional jump is selected, then the jump is based on the current condition (zero/not zero) of the specified word and record. Jump Unconditional prompts you to enter a Jump to Reference. This reference is two characters and is*

resolved when a 'JR' Jump Reference is defined. See the 'JR' command definition. If you select a conditional jump, prompt messages requesting word number and record number are issued. Following the definition of these two codes, you are requested to enter the Jump to Reference. The Jump to Reference for a Conditional Jump is the same as that of an unconditional Jump. The following example shows the use of the Jump command.

Command sequence using Jump:

```

MP      X=200, Y=200
JP      Zero, WORD#=0, RECORD#=4, JR=AA
DR      SYM=1, MOD=40
JP      JR=BB
JR      AA
DR      SYM=2, MOD=40
JR      BB
EP
SA
```

The preceding example draws a fan symbol at 200,200 either right or left depending on the zero/not zero condition of the Realtime Data Member word 0, record 4.

In the preceding sequence, the first JP causes a jump to JR AA if word 0 of record 4 is zero. The second JP causes a jump to JR BB unconditionally.

JR - Jump Reference

Indicates to the Composer that this location in the command sequence is referred to in a JP command. As defined in the JP command, the location is defined by 2 characters. If these characters have already been used, an error message is displayed. If the capacity of the JR table is exceeded, an error message is displayed. The capacity of the jump reference table is 40 unique jump reference points for each display.

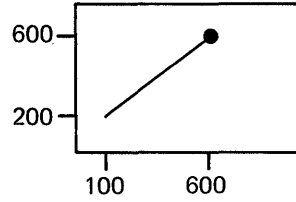
LB - Display Characters

Places a character string on the screen. It is not necessary to use an MP command to position the beam because this command allows specification of the location of first character. If a 3D object is being defined, then X, Y, and Z values are requested. Up to 72 characters can be displayed. The ending X,Y position are 1 character position beyond the last character in the string.

SDICOMP

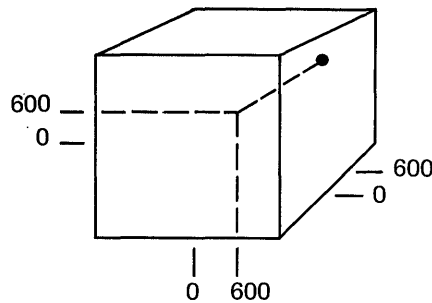
LI - Draw a Line to X,Y

Draws a vector to the specified X and Y coordinates from wherever the beam was left with the previous command.



Draw line to X=600 Y=600 Line shown on screen
END X,Y Current position

When generating a 3D display, 3 values are required. These values are X, Y, and Z.



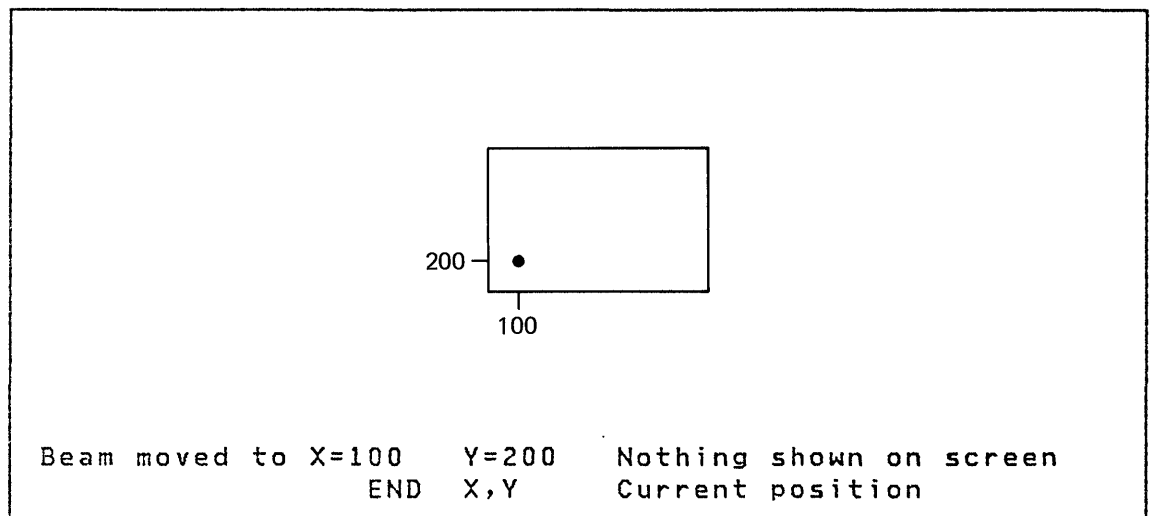
Draw line to X=600 Y=600 Z=600
END X,Y,Z

LR - Draw Line Relative

Draws a line relative to the current position. For example, you can, through the use of the 'MP', 'JP', and 'JR' commands, position the beam at various current positions based on Realtime Data Member conditions. Then a series of lines can be drawn to form a symbol using the 'LR' command. This would have the effect of placing the symbol at various screen locations based on external conditions. The limits allowed for the X,Y values are plus or minus 512 units. If a 3D object is being defined, then the Z axis value is also requested.

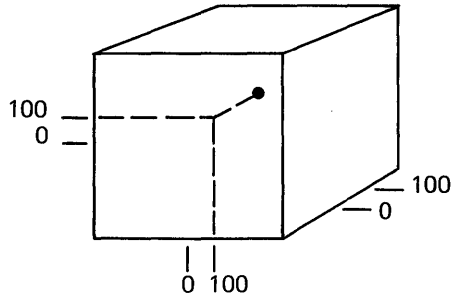
MP - Move Beam to X,Y

Draws a dark vector to the specified X and Y coordinates. A dark vector is not visible and, therefore, results in moving the beam to the specified location.



When generating a 3D display, 3 values are required. These values are X, Y and Z.

\$DICOMP



```
Beam moved to X=100  Y=100  Z=100
                END X,Y,Z  Current position
```

PC - Plot Curve Only

Provides multiple curves on an existing background as defined by a preceding PL command. Refer to the following section (PL) for descriptions of entry procedure. Steps 9 and 10 are the only required action. As many PC commands as are necessary to obtain the desired results can be included.

PL - Plot Data

Formats the viewing area into a basic plotter. Options are provided for X and Y labels as well as X and Y grids. You are prompted to include the name of a plot curve data member. Refer to the "\$DIUTIL Utility Program" in "\$DIUTIL - Display Data Base Utility" on page 150 for information regarding the allocation and formatting of the plot curve data member. The sequence of questions to be answered for the PL command:

1. Enter # of Y axis divisions

To present a readable display, it is suggested that this value be under 20. However, if Y axis division values are bypassed (Step 7), then larger values are appropriate. Y axis divisions become unreadable when this value exceeds 125.

2. Enter # of X axis divisions

To present a readable display, it is suggested that this value be under 40. However, if X axis division values are bypassed (Step 8), then larger values are appropriate. X axis divisions become unreadable when this value exceeds 200.

3. Vertical Grid?

A Y answer causes the Composer to include commands to connect the X axis divisions (specified in 2 preceding) to the top of the viewing area. An N bypasses this feature.

4. Horizontal Grid?

A Y answer causes the Composer to include commands to connect the Y axis divisions specified in 1 preceding to the right side of the viewing area. An N bypasses this feature.

5. Enter Y axis label - 24 characters

You must enter the Y axis label. If no axis label is desired, press the enter key. This label is general in nature and is placed at the left side of the viewing area. This label is vertical, that is, one character under the next.

6. Enter X axis label - 24 characters

You must enter the X axis label. If no X axis label is desired, press the enter key. This label is general in nature and is placed near the lower portion of the plot viewing area.

7. Y axis division values?

If Y axis division values are desired, respond with a Y. The Composer asks for as many values as you have specified divisions plus 1 (see Step 1 preceding). You must enter 6 characters for each division. The first value requested is the value for the Y base line and each succeeding value is for the next division in the plus Y direction.

8. X axis division values?

If X axis division values are to be displayed, respond with a Y. The composer asks for as many values as you have specified divisions plus 1 (see Step 2 preceding). You must enter 6 characters for each division. The first value requested is the value for the X base line and each succeeding value is for the next division in the plus X direction.

9. Enter Name of Member for Plot Data

Enter the name of a plot curve data member. This member must have been allocated and initialized by the use of the utility program \$DIUTIL. Refer to "\$DIUTIL Utility Program" in "\$DIUTIL - Display Data Base Utility" on page 150 for procedures on allocation and initializing of this member.

10. Is This Plot a Point Plot?

The composer allows the use of any valid printable character to be used for the plot. If Yes is selected, then a plot character is requested. If N is selected, then a normal line for the curve is used.

The preceding 10 steps generate the necessary commands to cause a basic plot background to be displayed and one curve to be superimposed on that background. If additional curves are desired, then 'PC' commands should be issued next.

RT - Activate New Realtime Data Member

You can define multiple Realtime Data Members. This command allows you to switch from one to another during the generation of a Display. The default name for the realtime data member is 'REALTIME'.

SA - Save Accumulated Data

Specifies that completion of a display profile has been reached. The Composer enters the member name into the directory of the display profile data base and make it available for the Interpreter.

TD - Display Time and Date

This command allows you to display the current time of day and date from the realtime clocks used by the Event Driven Executive. You are reminded that prior to issuing a 'TD' command, a 'MP' may be required to position the beam to the desired display location. The 'TD' command displays the time and date in the following format:

```
HH:MM:SS  MM/DD/YY
```

```
where:    HH is Hours  
          MM is Minutes  
          SS is Seconds  
          MM is Month  
          DD is Day  
          YY is Year
```

VA - Display Variable

Places a data variable from the Realtime Data Member on the screen. A prompt message is issued asking if you wish to locate the data at a location other than the current X,Y position. If a 3D object is being defined, then X, Y, and Z are requested. The use of this command requires that the Realtime Data Member be allocated. The Composer continues by asking for the record number and word number. The record number is the record number within the Realtime Data Member. The word number is the word number within the record specified. This value is in the range of 0-8.

The function code is requested next and is used to indicate the type of variable to be displayed. Valid function codes are as follows:

```
0      Single precision integer  
1      Double precision integer  
2      Standard precision floating point  
3      Extended precision floating point  
15     Character data
```

Type code is requested next and is an indicator of the format of the value to be displayed. Valid type codes are:

\$DICOMP

- 0 Integer
- 1 Floating point F format
- 2 Floating point E format

Field width and number of decimal places are requested next.
If the variable is an integer, the number of decimals should be zero.

\$DIINTR - DISPLAY INTERPRETER

The Interpreter program searches the data base and generates the requested display. Both graphic and report displays are generated in this manner. Each display profile is made up of many display profile elements. Each element, when retrieved from the data base by the Interpreter, is decoded and converted to the appropriate command to cause the requested action to be performed. Each display profile element contains various parts, such as Display Code, X and Y coordinates, Symbol ID, and Symbol Modifier. Realtime Data Member record number and Additional Member names are included in the display profile element.

To begin operation of the Interpreter, you must first load \$DIINTR. Output is directed to the terminal that requests the display or as directed by the Display Profile. The following steps are required to initiate the Processor Monitor:

1. Load the program \$DIINTR.
2. The system responds with the prompt message:

```
ENTER DISPLAY ID--XXXXXXXX OR EXIT TO TERMINATE
```

3. To terminate the Interpreter, enter 'EXIT'. To cause the Interpreter to prepare the display, enter the Display ID.

Using \$DIINTR from an Application Program

\$DIINTR can be loaded from an application program to allow displays without operator assistance. The following example is the method used to cause this action to occur.

\$DIINTR

```
.
.
.
* Your program
.
LOAD  $DIINTR, MBRNME, DS=( $DIFILE ), EVENT=#WAIT, C
      LOGMSG=NO
WAIT  #WAIT
.
.
MBRNME DATA CL8'DISPLAY'
S1     DATA F'0'      THESE 8 VALUES ARE FOR 3D OBJECTS
S2     DATA F'0'      *
S3     DATA F'0'      *
D      DATA F'0'      *
T      DATA F'0'      *
R      DATA F'0'      *
D1     DATA F'0'      *
T1     DATA F'0'      *
```

When using \$DIINTR to display a 3D object there are 8 values that are needed to describe the manner in which you want the object displayed. The preceding example shows the eight values passed to \$DIINTR as follows:

```
S1 Platform Location X=
S2 Platform Location Y=
S3 Platform Location Z=
D Platform Direction in Degrees
T Platform Tilt in Degrees
R Platform Rotate in Degrees
D1 View Direction in Degrees
T1 View Tilt in Degrees
```

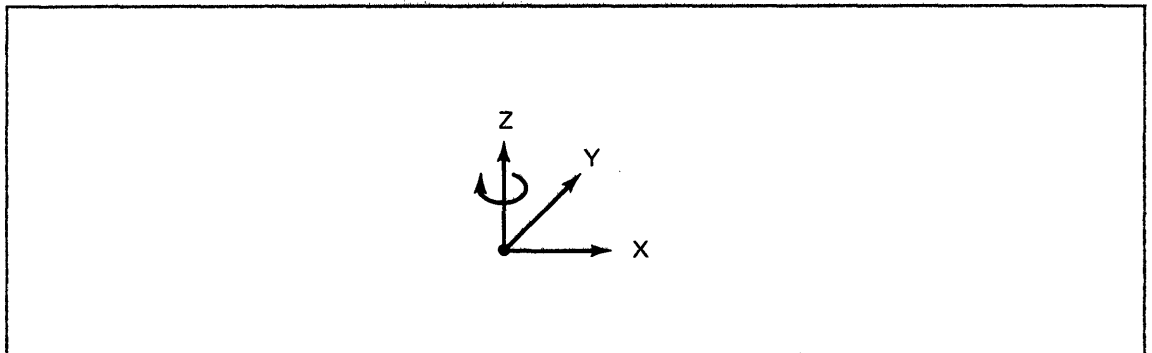
These values are single precision integers and may contain a numeric value from -32768 to +32767.

Displaying 3D images requires a 4955 processor with floating point hardware installed.

3D Concepts as used by \$DIINTR

3D objects can be defined by \$DICOMP and placed on disk or diskette much the same as with a 2D object. The only difference is that each point in space has three values associated with it instead of 2. These three values represent the X, Y, and Z coordinates of the point in space. Figure 5-2 shows the limits of the defined area in space. The maximum limits of the defined areas in space are -32,768 to +32,767. You can define one or more objects within this cube. Once the object is defined, you can view this object from any location within the same space. To specify the location from where you wish to view the object, either pass these eight values through the use of the PARM= parameter in the LOAD instruction or \$DIINTR requests this input if it is invoked by the \$L command. The concept used to compute the 2D representation of a 3D object is as follows. The user is assumed to be suspended on a platform at a specific location in space. The first three values are the X, Y, and Z values that define the location in space of the viewing platform. The next five values represent the physical orientation of the platform and the viewers orientation on that platform.

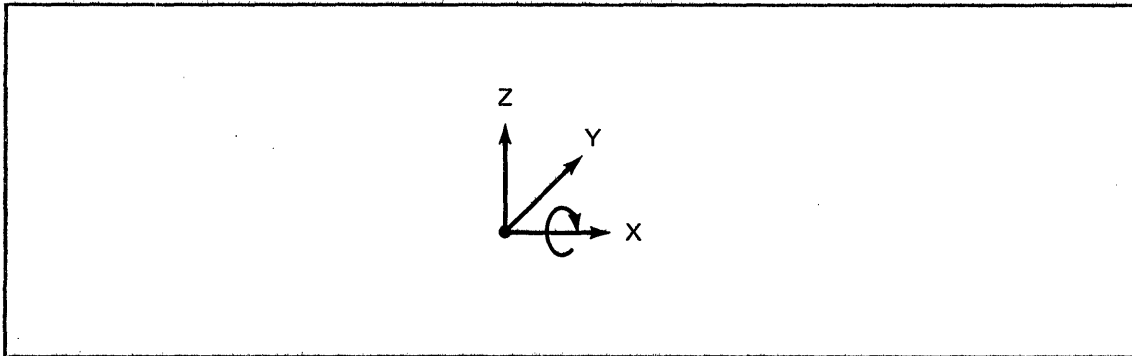
Platform Direction in Degrees: Assume the following unit vector:



If we rotate this unit vector in the direction Y to X around the Z axis, we can turn the view in any direction. A plus value causes the unit vector to rotate clockwise as viewed from the +Z axis to zero.

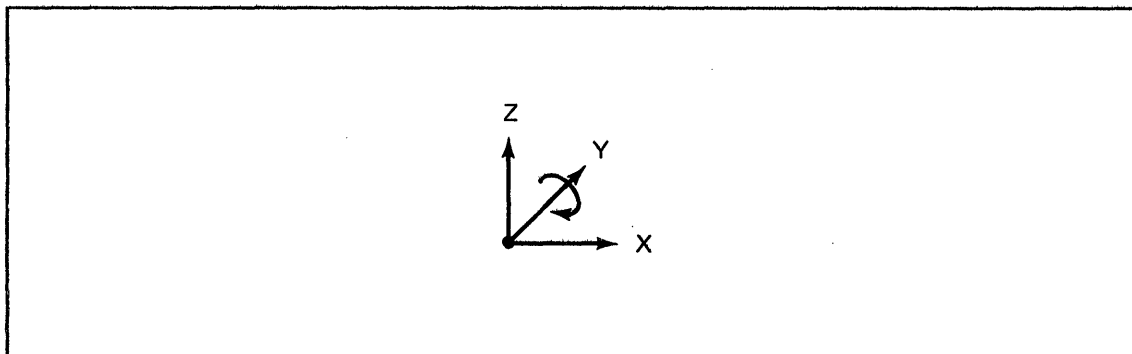
\$DIINTR

Platform Tilt in Degrees: Assume the following unit vector:



If we rotate this unit vector in the direction Z to Y around the X axis, we can tilt the view to any angle. A plus value causes the unit vector to rotate clockwise as viewed from the +X axis to zero.

Platform Rotate in Degrees: Assume the following unit vectors:



If we rotate this unit vector in the direction Z to X around the Y axis, we can rotate the view to any angle. A plus value causes the unit vector to rotate clockwise as viewed from the -Y axis to zero.

View Direction In Degrees: This value is used the same as the Platform Direction but is calculated after the above 3 are computed. This calculation rotates the unit vector in a Y to X direction around the Z axis with a plus value causing the unit vector to rotate clockwise as viewed from the +Z axis to zero.

View Tilt In Degrees: This value is used the same as the Platform Tilt but is calculated after the above 4 are computed. This calculation rotates the unit vector in a Z to X direction around the Y axis with a plus value causing the unit to rotate clockwise as viewed from the -Y axis to zero.

Once the 8 values provided are computed, the object in space is converted to its 2D representation and sent to the terminal. It is possible to view an object with all or a portion outside the viewing area. Points and lines that do not fall within the viewing area are not shown on the 2D screen. The viewing area is shown in Figure 19 on page 132.

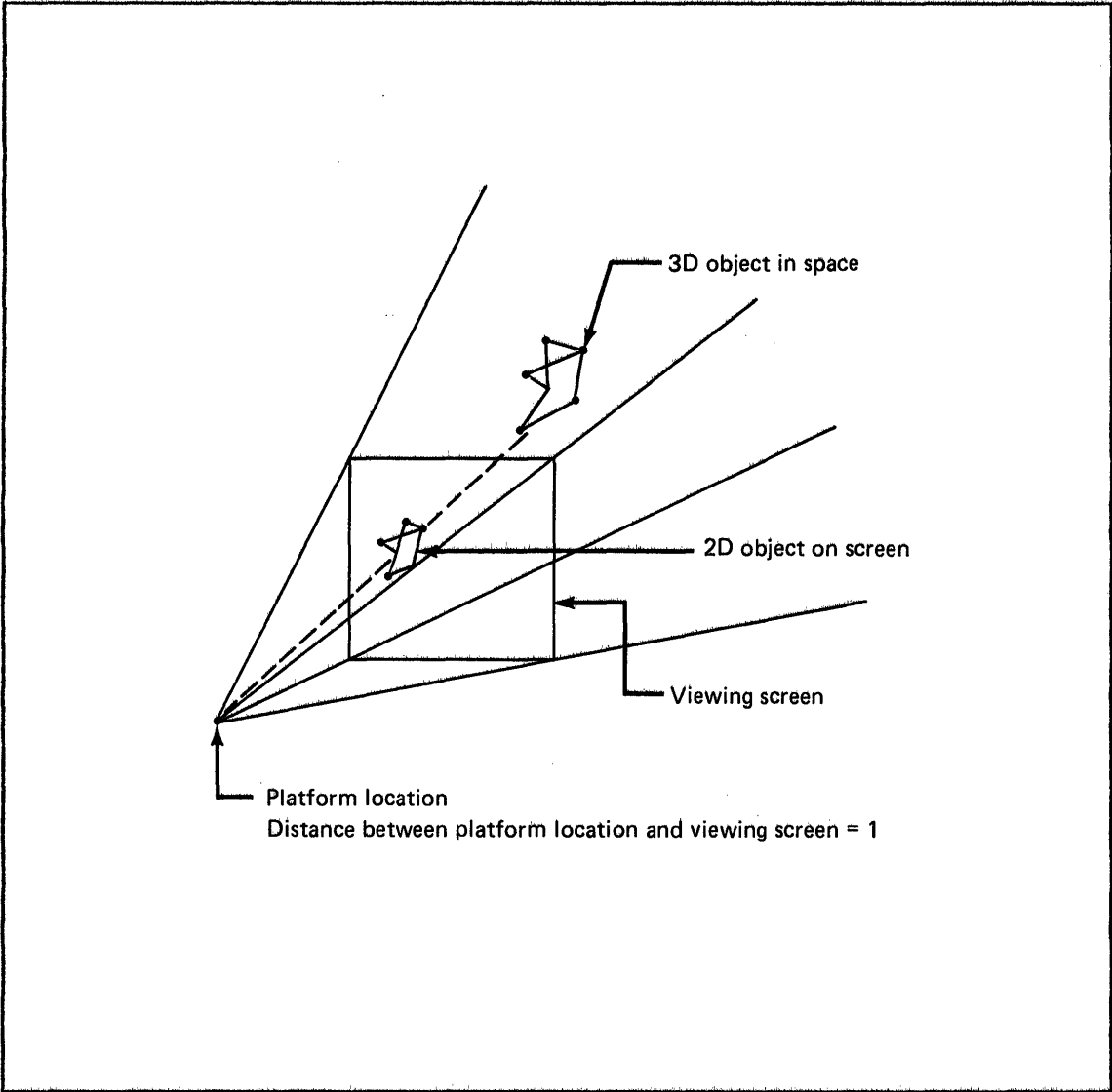


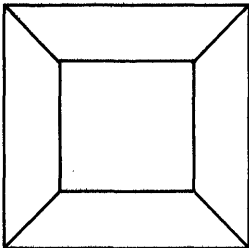
Figure 19. Viewing Area in 3D Mode.

The following example defines a 3D object in space

```
A Cube
CMD      X      Y      Z
MP      -100   -100   -100
LI       +100   -100   -100
LI       +100   -100   +100
LI       -100   -100   +100
LI       -100   -100   -100
LI       -100   +100   -100
LI       +100   +100   -100
LI       +100   +100   +100
LI       -100   +100   +100
LI       -100   +100   -100
MP       +100   -100   -100
LI       +100   +100   -100
MP       +100   -100   +100
LI       +100   +100   +100
MP       -100   -100   +100
LI       -100   +100   +100
EP
SA
```

Object as viewed from:

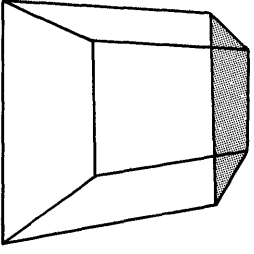
S1	0
S2	-400
S3	0
D	0
T	0
R	0
D1	0
T1	0



\$DIINTR

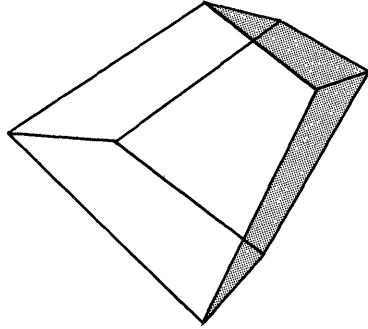
Object as viewed from:

S1	-150
S2	-400
S3	0
D	0
T	0
R	0
D1	0
T1	0



Object as viewed from:

S1	-150
S2	-400
S3	100
D	0
T	0
R	45
D1	0
T1	0



\$DISKUT1 - ALLOCATE/DELETE; LIST DIRECTORY DATA

\$DISKUT1 performs several commonly used disk or diskette storage management functions.

Note: For tape management functions, see "\$TAPEUT1 - Tape Management" on page 311

\$DISKUT1 Commands

The commands available under \$DISKUT1 are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

```
COMMAND (?): ?
AL ---- ALLOCATE SPACE
CV ---- CHANGE VOLUME
DE ---- DELETE MEMBER
EN ---- END THE PROGRAM
LA *--- LIST ALL(DS/PGM)
LACTS-* LIST ALL (CTS MODE)
LD *--- LIST DATA SETS
LDCTS-* LIST DATA SETS (CTS MODE)
LM ---- LIST 1 MEMBER
LP *--- LIST PROGRAMS
LPCTS-* LIST PROGRAMS (CTS MODE)
LS ---- LIST SPACE
LV *--- LIST THROUGH VOLUMES (DS/PGM)
LISTP-- DIRECT LISTING TO $SYSPTR
LISTT-- DIRECT LISTING TO TERMINAL
RE ---- RENAME A MEMBER
      *--- PREFIX (OPTIONAL)
COMMAND (?):
```

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, AL).

Note: In CTS mode, disk locations are shown in cylinder, track, and sector format instead of by record number.

§DISKUT1

The program prompts you for any parameters required by the requested function. Examples of some of the §DISKUT1 command execution prompts and replies are given on the following pages.

§DISKUT1 Parameters

The following table identifies the parameters that can be used for the various §DISKUT1 commands.

Function	Command	Parameters
Help	?	None
Allocate Space	AL	Name Size Type
Change Volume	CV	New-volume-label
Delete Member	DE	Member-name
End Utility	EN	None
List All Members	LA	Prefix (optional)
List All Members in CTS mode	LACTS	Prefix (optional)
List Data Sets	LD	Prefix (optional)
List Data Sets in CTS mode	LDCTS	Prefix (optional)
List One Member	LM	Member-name
List Programs	LP	Prefix (optional)
List Programs in CTS mode	LPCTS	Prefix (optional)
List Space	LS	None
List Through Volumes	LV	Prefix (optional)
Direct Listing to §SYSPRTR	LISTP	Terminal-name (optional)
Direct Listing to Terminal	LISTT	None
Rename Member	RE	Old-name New-name

Note: When using the AL command, select either of the following organization types for the data set to be allocated:

D - Data Organization

P - Program: Use this only for executable object programs
(the output of §UPDATE/§UPDATEH)

All the functions listed (except for ?, CV, LV, and EN) initially act upon the IPL volume and is indicated by the message USING VOLUME XXXXXX when §DISKUT1 is loaded. To point to another volume to perform one or more of the previously listed functions, enter the CV command and the name of the volume. All functions act upon the specified volume until changed by another CV command or until §DISKUT1 is terminated and reloaded. If a prefix of up to 8 characters is specified in the List commands (except LM), only those data sets/programs beginning with

these characters are listed. The LV command scans through all existing volumes, listing members in each volume. When the scan is completed, the utility points to the last volume accessed. The LV command is useful in finding a data set when the volume is not known or when the same data set appears in multiple volumes.

Examples

AL - Allocate a 100 Record Data Set Named DATAFILE

```
COMMAND (?): AL
MEMBER NAME: DATAFILE
HOW MANY RECORDS? 100
DEFAULT TYPE = DATA - OK? Y
DATAFILE CREATED

COMMAND (?):
```

CV - Change Volume to be Accessed by Subsequent Commands

```
COMMAND (?): CV
NEW VOLUME LABEL = EDX001

COMMAND (?):
```

DE - Delete a Member Named DATAFILE

```
COMMAND (?): DE
MEMBER NAME: DATAFILE
DATAFILE DELETE? Y
DATAFILE DELETED

COMMAND (?):
```

\$DISKUT1

LD - List Description of the Data Sets in a Volume

COMMAND (?): LD

USING VOLUME EDX001

NAME		FREC	SIZE
TEXTWORD	DATA	105	22
\$SAMDATA	DATA	127	36
\$NAME3	DATA	450	10

506 FREE RECORDS IN LIBRARY

COMMAND (?):

Note: FREC is the number of the record containing the first record of the data set.

LM - List Description of an Individual Member

COMMAND (?): LM

MEMBER NAME: TEST12

USING VOLUME EDX001

NAME		FREC	SIZE
TEST12	DATA	305	7

IODA,CTS=003,022116,022122

COMMAND (?):

Note: IODA,CTS= I/O Device Address, Cylinder, Track, and Sector. In this example, the extent of the member is on the device at device address 003 at cylinder 22, track 1, from sector 16 through sector 22.

LP - List Description of the Program Members in Volume

COMMAND (?): LP \$DISK

USING VOLUME EDX001

NAME	FREC	SIZE
\$DISKUT1 PGM	256	32
\$DISKUT2 PGM	288	30

2550 FREE RECORDS IN LIBRARY

COMMAND (?):

Note: Only members with a prefix of \$DISK are listed in this example.

\$DISKUT1

LS - List Free Space Available in Volume

COMMAND (?): LS

USING VOLUME EDX001

LIBRARY

AT REC 1
SIZE 3600 RECORDS
UNUSED 665 RECORDS

DIRECTORY

SIZE 24 RECORDS
UNUSED 2450 BYTES

NO. MEMBERS - 35

NO. FREE SPACE ENTRIES - 2

LIST FREE SPACE CHAIN? Y

FREC	SIZE
3000	600
247	65

COMMAND (?):

LV - List Members with a Prefix of 'S' in All Volumes

COMMAND (?): LV S

NAME		FREQ	SIZE	VOLUME
SEW	DATA	1646	100	EDX002
SSRC	DATA	7748	5	EDX002
SMODUL	DATA	1386	10	ASMLIB
SWORK	DATA	15522	300	EDX003
SDATA	DATA	6989	5	EDX005

USING VOLUME EDX005

COMMAND (?):

SDISKUT2

SDISKUT2 - PATCH, DUMP OR CLEAR MEMBER

SDISKUT2 dumps or patches data or program members of a volume. It also can clear (set to zero) all or portions of a data set and reset its end-of-data pointer, list any data set created using \$EDIT1N or \$FSEDIT, and list the I/O error log data set.

Note: For tape management functions, see "\$TAPEUT1 - Tape Management" on page 311

SDISKUT2 can also be used to modify the default load time storage allocation associated with a program. The SS (set storage) command allows you to change the allocation without reassembling the source code or providing an override on the LOAD instruction.

Program dumps and patches are made by relative address (hexadecimal) within the program. The relative address corresponds exactly to the address specified in the LOC field of an assembly listing. Data can be entered in hexadecimal or EBCDIC as shown in the examples that follow. To convert an Event Driven Language instruction to a no operation (NOP), patch all of the generated DCs to hexadecimal zero.

Data set dumps and patches are made by specifying a record number and a first word. The numbering for both record and word number begins with 1. Data can be entered in either decimal or hexadecimal. Each field of patch data should be separated with a non-numeric character other than a carriage return.

Dumps of programs or data sets are formatted when hexadecimal is selected as an option.

SDISKUT2 can list data sets created by \$EDIT1N or \$FSEDIT. You can list all or part of a source data set on a terminal or printer.

A special feature of SDISKUT2 allows dumping and/or patching of any area on a disk volume by referencing absolute record numbers. This mode is selected by entering the characters \$\$EDXVOL as a member data set name. When using this mode, record numbering begins with one.

The I/O error log list commands allows the specification of any data set as the log data set. The option is available to print every log record or only those log records containing data for a specific device address. The output is a formatted dump.

You are prompted, as necessary, for information required by any of the functions of SDISKUT2.

SDISKUT2 Commands

The commands available under SDISKUT2 are listed below. To display this list at your terminal, enter a question mark in response to the prompting command COMMAND (?).

```
COMMAND (?): ?  
  
CD - CLEAR DATA SET  
CV - CHANGE VOLUME  
DP - DUMP DS OR PGM ON PRINTER  
DU - DUMP DS OR PGM ON CONSOLE  
    (-CA- WILL CANCEL)  
PA - PATCH DS OR PGM  
SS - SET PROGRAM STORAGE PARM  
LP - LIST DS ON PRINTER  
LU - LIST DS ON CONSOLE  
PL - LIST LOG ON PRINTER  
LL - LIST LOG ON CONSOLE  
EN - END PROGRAM  
  
COMMAND (?):
```

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, CD).

\$DISKUT2

Examples

CD - Clear a Data Set (to Zeros)

```
COMMAND (?): CD
DATA SET NAME? DATA
CLEAR ENTIRE DATA SET? N
FIRST   RECORD: 1
LAST    RECORD: 100

RESET THE E.O.D. POINTER? Y
HOW MANY RECORDS TO EOD? 100

ARE ALL PARAMETERS CORRECT? Y
CLEAR COMPLETED

COMMAND (?):
```

DU/DP - Dump a Data Set on Terminal/Printer

```
COMMAND(?): DU
PGM OR DS NAME:  EDITWORK
EDITWORK IS A DATA SET
FIRST   RECORD: 1
LAST    RECORD: 1
WORDS / RECORD: 52
(D)EC OR HE(X): X

RECORD 1
 73  7A2E 78D0 0088 7A30 000A 101A D240 4040  | :.....:..
 81  0000 0000 34D6 0000 0000 0000 0000 FFFF  | .....0...
 89  0000 0000 14D0 5600 0000 7A02 0000 0000  | .....
 97  0000 0000 0000 0000 0000 0000 0000 0000  | .....
105  0000 0000 0000 0000 5040 6F03 023C 0254  | .....&
113  7B96 402F 7BA0 0000 182C 6808 00C4 680D  | #.  .#....
121  7BA4 6808 00F6 680D  | #.....6..

DUMP COMPLETE
ANOTHER AREA? N
COMMAND(?)
```

DU/DP - Dump a Program on Terminal/Printer

```
COMMAND (?): DU
PGM OR DS NAME: MYPROG
MYPROG IS A PROGRAM
ADDRESS: 22
DUMP TO PROGRAM END? N
HOW MANY WORDS? 22

0022    0000 0000 1C66 FFFF 0000 0000 |.....$$|
0032    4040 4040 4040 0606 4040 4040 |.. ..|

0042    0000 0001 0001 0000 0000 |.....|

DUMP COMPLETE
ANOTHER AREA? N

COMMAND (?):
```

LL - List Log Data Set

```
COMMAND (?): LL
LOG DS NAME: $LOGDS
DEVICE ADDRESS(NULL FOR ALL): 003

SOFT ERR
DEV ADDR: 0003          DEV ID: 0304
DATE: 10/26/79         LVL: 0002   AKR: 0001
TIME: 11:59:59         RETRY: 13   IDCB: 7000 2100
INTCC: 07              ISB: 80
DCB1: XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
DCB2: XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
CSSW: XXXX XXXX XXXX
```

\$DISKUT2

LU - List a Source Data Set on Terminal

```
COMMAND(?): LU
DATA SET NAME? CALSRC
LIST ALL OF THE DATA SET? N
FIRST RECORD: 4
LAST RECORD: 8

ATTNLIST ATTNLIST (STOP,POST1,CALC,POST2)
          SPACE 1
POST1    POST      KBEVENT,1
          ENDATTN
          SPACE 1

LIST COMPLETE

COMMAND(?):
```

PA - Patch a Program in Hexadecimal

```
COMMAND (?): PA
PGM OR DS NAME: MYPROG
MYPROG IS A PROGRAM
ADDRESS: 312
HOW MANY WORDS? 2
(D)EC, (E)BCDIC, OR (H)EX?: H

NOW IS:
 0312      D3C9  E2E3          |LIST
ENTER DATA: C3C1 D3D3

NEW DATA:
 0312      C3C1  D3D3          |CALL

OK? Y
PATCH COMPLETE
ANOTHER PATCH? N

COMMAND (?):
```


\$DISKUT2

PA - Patch a Program in EBCDIC

COMMAND (?): PA
PGM OR DS NAME: MYPROG
MYPROG IS A PROGRAM
ADDRESS: 2D8
HOW MANY WORDS? 7
(D)EC, (E)BCDIC, OR (H)EX?: E

NOW IS:
02D8 D4C5 D4C2 C5D9 40C4 C5D3 C5.. .. 1 MEMBER DELETED

ENTER DATA: DELETE MEMBER

NEW DATA:
02D8 C4C5 D3C5 E3C5 40D4 C5D4 C2.. .. 1 DELETE MEMBER

OK? Y
PATCH COMPLETE
ANOTHER PATCH? N

COMMAND (?):

SS - Set Program Storage Parm: The following example shows reducing the dynamic storage to be allocated for the COBOL compiler at program load. The SS command requires the size in bytes to be expressed in decimal. The value requested, if not an even multiple of 256, will be rounded up.

```
> $DISKUT2
```

```
USING VOLUME EDX002  
COMMAND (?): CV ASMLIB
```

```
COMMAND (?): SS $COBOL  
ENTER NEW STORAGE SIZE IN BYTES: 2816  
OLD STORAGE SIZE WAS 8448  
OK TO CONTINUE? Y
```

```
COMMAND (?): EN  
$DISKUT2 ENDED AT 08:36:02
```

SDIUTIL

SDIUTIL - DISPLAY DATA BASE UTILITY

SDIUTIL maintains the disk resident data base used with graphics applications. This utility provides comprehensive facilities to keep the data base current by means of the following functions:

- Initialize the Disk Resident data base
- Delete a member
- Reclaim space in data base due to deleted members
- Display contents of data base
- Copy data base
- Copy individual members of data base
- Allocate and build a data member

This utility is normally used only when no other programs of the Display Processor are in use. The online data base can be changed or you may select another data base to be referenced. This allows you to create displays in a data base other than the online data base and then copy the members into the online data base after testing.

Invoking SDIUTIL

To start execution of SDIUTIL:

1. Load the program SDIUTIL specifying the appropriate data set. SDIFILE, the online data set, or any other data set can be used. However, you should make sure that another user or program is not changing or using the same data set.
2. The system responds with the Program Loaded message followed by:

```
DISPLAY DATA BASE UTILITY  
COMMAND (?):
```

\$DIUTIL Commands

The commands available under \$DIUTIL are listed below. To display this list at your terminal, enter a question mark in reply to the prompting message COMMAND (?):.

```
COMMAND (?): ?  
  
AL - ALLOCATE DATA MEMBER  
BU - BUILD DATA MEMBER  
CP - COMPRESS DATA BASE  
CM - COPY MEMBER  
DE - DELETE A MEMBER  
EN - EXIT PROGRAM  
IN - INITIALIZE DATA BASE  
LA - DISPLAY MEMBER DIRECTORY  
LH - DISPLAY MEMBER HEADER  
MD - MOVE DATA BASE  
RE - RENAME MEMBER  
ST - DISPLAY DATA SET STATUS  
  
COMMAND (?):
```

After the commands are displayed, you are again prompted with COMMAND (?): to which you respond with the command symbol for the function of your choice (for example, AL).

AL - Allocate Data Member

Reserves space in a data base for one of several types of data members. Information such as size in sectors and member code is requested. Member codes are specified as follows:

4 - Print Report Data Member: Information such as number of lines and line length are requested. Each line is then entered, limited to 132 characters.

5 - Plot Curve Data Member: Information such as X and Y ranges, X and Y base values and number of points to plot are requested. Automatic entry of the X points can be selected to reduce the data entry requirements. A sawtooth pattern option is provided to shade under the curve for more vivid presentation of plotted data. Using less than 200 points on the X axis gives an inadequate shading effect.

\$DIUTIL

6 - Realtime Data Member: The number of records is requested. You can enter hexadecimal data for testing.

7 thru 9 - User Data Member: These codes are used by the build function to guide you through the correct data entry procedure.

```
COMMAND (?): AL
MEMBER NAME: TDATA
ENTER # OF RECORDS TO ALLOCATE? 10
ENTER MEMBER CODE #: 4
MEMBER TDATA ALLOCATED

COMMAND (?):
```

BU - Build Data Member

Inserts fixed data into a data member. This allows you to enter data records to describe a fixed display or enter records, which normally will be dynamic, with a fixed value, to allow testing of the display.

The member may have been allocated using AL; if not, you are prompted for the requisite allocation information before proceeding with the "build" process. You are guided one step at a time through the initialization of the data member.

```

COMMAND (?): BU
ENTER MEMBER NAME: RDATA
INITIALIZE REPORT DATA MEMBER
ENTER # OF LINES IN REPORT: 2
LINE LENGTH=32
ENTER LINE ITEMS
LINE ONE OF REPORT
LINE TWO OF REPORT
MEMBER LOADED

COMMAND (?):
    
```

In this case, the member had already been allocated.

CP - Compress Data Base

Compress reclaims unused space in the data base. Deleted members are not actually removed; the space is merely flagged unusable. The Insert function of \$DICOMP also flags space as unusable. CM is used to reclaim this space for future use. As each member is moved, a message is displayed. At the completion of the compress function, the message COMPRESS COMPLETED is displayed.

Caution should be exercised in using this function as it actually rearranges the members in the data base. It is advisable, in order to prevent unpredictable results, to restrict the use of the Interpreter (\$DIINTR) during this process.

Caution: If an unrecoverable I/O error occurs, the data set is destroyed.

```
COMMAND (?): CP
WARNING--COMPRESS IN PLACE.  IF AN ERROR
SHOULD OCCUR DATASET WILL BE DESTROYED
  DO YOU WISH TO PROCEED? Y
DATA      COPIED
RDATA     COPIED
REPORT    COPIED
SQUARE    COPIED
CIRCLE    COPIED
RPT       COPIED
ARC       COPIED
PLOT      COPIED
COMPRESS  COMPLETED

COMMAND (?):
```

CM - Copy Member

Copies a member from the source data base to the target data base. The options available in MD are also included in CM.

```
COMMAND (?): CM
SOURCE DATASET NAME: $DIFILE
LOCATED ON VOLUME: EDX002
CHANGE SOURCE DATASET? N
TARGET (NAME, VOLUME): $DIFILE,EDX003
SAVE EXISTING MEMBERS IN TARGET DATA BASE? Y
ENTER MEMBER NAME TO BE COPIED
PLOT
PLOT      COPIED
COPY COMPLETED

COMMAND (?):
```


\$DIUTIL

DE - Delete a Member

Removes display or data members from the data base. You are prompted for the name of the member to be deleted and asked to verify the accuracy of your entry prior to actual deletion.

```
COMMAND (?): DE
MEMBER NAME: PLTT
DELETE MEMBER PLTT? Y
PLTT DELETED

COMMAND (?):
```

EN - Exit Program

Causes the Display Processor utility to be terminated.

IN - Initialize Data Base

Formats the entire data base to zeros and formats the directory to reflect the starting and ending record numbers. After entry, you are prompted to proceed.

Caution: This function destroys any data in the data base.

Make sure the data set name entered is correct. The command is terminated when the message DATA SET FORMATTED is displayed. \$DIFILE was allocated by \$DISKUT1. Each directory record allocated by IN contains sixteen directory entries, except the first, which contains fifteen.

```
COMMAND (?): IN
*-*-WARNING THIS FUNCTION WILL DESTROY ANY DATA
                      CURRENTLY IN DATA SET-*-*

DO YOU WISH TO PROCEED? Y
ENTER DIRECTORY SIZE IN RECORDS: 2
DATA SET FORMATTED
DATASET NAME: $DIFILE
LOCATED ON VOLUME: EDX002
- DATA SET -- DIRECTORY-
  NEXT TOTAL  NEXT TOTAL
    3      100      1      31

END OF STATUS

COMMAND (?):
```

\$DIUTIL

LA - Display Directory

Displays all active members. Each line of display shows the member name followed by four values:

1. Starting sector relative to the start of the data base.
2. Length of member in records.
3. Member usage code.
4. User defined member code.

COMMAND (?) :	LA			
PLOT	11	4	2	0
DATA	15	10	5	0
RDATA	25	10	4	0
REPORT	35	1	1	0
SQUARE	36	1	2	0
CIRCLE	38	1	2	0
RPT	39	1	1	0
ARC	40	1	2	0

COMMAND (?) :

LH - Display Member Header

Displays the header of a data member (types 4-9). The header describes the characteristics and use of the member. For a description of header contents, see Data Set Format in IBM Series/1 Event Driven Executive System Guide, SC34-0312

```

Example:

COMMAND (?): LH
MEMBER NAME: RDATA
MEMBER RDATA HEADER
          0      0      0      0      2      32      80      0
END OF HEADER

COMMAND (?):
    
```

\$DIUTIL

MD - Move Data Base

Moves the data base on the same or another volume when the data base becomes too small to add a member. You can temporarily move the online data base to another location, delete the old version, reallocate and initialize the new expanded version, and move back the previous contents. During this procedure, care should be observed in the use of the Interpreter.

Caution: If the data base is being moved and the Interpreter uses a member, unpredictable results will occur.

During the execution of MD, you are prompted for a new source data base if desired and a target data base. You have the option of saving the members in the target data base. MD is helpful if you wish to use \$DICOMP to develop display members in a different data base than the online version and then, at a later time, combine the new members with those in the online data base.

```
COMMAND (?): MD
SOURCE DATASET NAME: $DIFILE
LOCATED ON VOLUME: EDX002
CHANGE SOURCE DATASET? N
TARGET (NAME,VOLUME): $DIFILE,EDX003
SAVE EXISTING MEMBERS IN TARGET DATA BASE? Y
PLOT          COPIED
DATA          COPIED
RDATA        COPIED
REPORT       COPIED
SQUARE       COPIED
CIRCLE       COPIED
RPT          COPIED
ARC          COPIED
COPY COMPLETED

COMMAND (?):
```

RE - Rename Member

Changes the Display Profile ID name. You are prompted for each step and no action is taken unless your response is first obtained. RE is useful when an online member needs to be modified. The member that needs changing can be copied to another data base, modified and tested, then renamed and copied back to the online data base. By using the rename and delete functions, you can exchange the new for the old without interfering with any online functions.

```

COMMAND (?): RE
MEMBER NAME: PLOT
ENTER NEW NAME: PLTT
RENAME COMPLETED

```

```

COMMAND (?):

```

\$DIUTIL

ST - Display Data Set Status

Displays the current data base status. The first line shows the data base location and name. The data that follows is the current status of the data base. There are 4 values presented. The first is the next available record. The second is the total number of records in the data base. You can then see how much space is available for new members. If space is running short, you can compress the data base or allocate a larger area. The next value displayed is the next available directory entry. The last value displayed is the total number of directory entries available. Refer to these two values to determine if more or less space is needed for directory entries. Following the completion of the status display the message END OF STATUS is displayed.

```
COMMAND (?): ST
DATASET NAME: $DIFILE
LOCATED ON VOLUME: EDX002
- DATA SET -- DIRECTORY-
  NEXT TOTAL  NEXT TOTAL
    41      100     10     159

END OF STATUS

COMMAND (?):
```

\$DUMP - FORMAT AND DISPLAY SAVED ENVIRONMENT

\$DUMP displays on a terminal or printer the contents of the data set generated by the \$TRAP utility. After the successful execution of \$TRAP and the subsequent occurrence of a trap condition, the data set assigned to \$TRAP will contain a storage image. Use \$DUMP to retrieve, format, and print the data on a terminal or printer.

Invoking \$DUMP

\$DUMP can be invoked by the session manager using the Diagnostic Utilities option menu or by the \$L command.

\$DUMP

Example

Dump Part of Supervisor Partition to Printer

```
< $L $DUMP                                     Notes
DUMPDS(NAME,VOLUME): DUMP,EDX003                (1)
$DUMP      22P,12:20:17,   LP=8F00

ENTER DEVICE NAME FOR OUTPUT                    (2)
$SYSPRTR

PARTIAL DISPLAY? (Y/N): Y                       (3)

ENTER PARTITION # OR S FOR SUPERVISOR S        (4)
ENTER START END ADDR IN HEX  0 100            (5)

EVENT DRIVEN EXECUTIVE FORMATTED STORAGE DUMP

AT TIME OF TRAP PSW WAS 8002 ON HARDWARE LEVEL 2

      LEVEL 0      LEVEL 1      LEVEL 2      LEVEL 3

IAR      0892      0892      0892      0892
AKR      0000      0000      0000      0000
LSR      00D0      0090      0090      0090
R0       0000      0000      0000      0000
R1       0000      0000      0000      0000
R2       0000      0000      0000      0000
R3       0000      0000      0000      0000
R4       0000      0000      0000      0000
R5       0000      0001      0002      0003
R6       8000      8000      8000      8000
R7       0000      0000      0000      0000
```

Dump Part of Supervisor Partition to Printer (cont.)

Notes:

1. The data set specified here must be the same as that defined when \$TRAP was executed.
2. You can specify a terminal to receive the output from \$DUMP. If the operator presses the 'ENTER' key or enters '\$DUMP', the dump program assumes that the output is to be directed to the terminal that loaded \$DUMP. Using the attention key followed by 'CA' cancels the \$DUMP program.
3. If a display of all storage is desired then respond to this question with a 'N'. If 'N' is used, the output display begins immediately and continue until all of storage is dumped or an attention 'CA' is entered. If 'Y' is used, \$DUMP allows sections of storage to be displayed.
4. Enter an 'S' for the supervisor partition or the number 1 through 8 for the partition number to dump.
5. Enter the starting and ending addresses that should be included in this section of the output.

\$DUMP

Dump Part of Supervisor Partition to Printer (cont.)

SEGMENTATION REGISTERS:

BLOCK	ADS0	ADS1	ADS2	ADS3	ADS4	ADS5	ADS6	ADS7
0000	0004	0104	0204	0304				
0800	000C	010C	020C	030C				
1000	0014	0114	0214	0314				
1800	001C	011C	021C	031C				
2000	0024	0124	0224	0324				
2800	002C	012C	022C	032C				
3000	0034	0134	0234	0334				
3800	003C	013C	023C	033C				
4000	0044	0144	0244	0344				
4800	004C	014C	024C	034C				
5000	0054	0154	0254	0354				
5800	005C	015C	025C	035C				
6000	0064	0164	0264	0364				
6800	006C	016C	026C	036C				
7000	0074	0174	0274	0374				
7800	007C	017C	027C	037C				
8000	0084	0184	0284	0384				
8800	008C	018C	028C	038C				
9000	0094	0194	0294	0394				
9800	009C	019C	029C	039C				
A000	00A4	01A4	02A4	03A4				
A800	00AC	01AC	02AC	03AC				
B000	00B4	01B4	02B4	03B4				
B800	00BC	01BC	02BC	03BC				
C000	00C4	01C4	02C4	03C4				
C800	00CC	01CC	02CC	03CC				
D000	00D4	01D4	02D4	03D4				
D800	00DC	01DC	02DC	03DC				
E000	00E4	01E4	02E4	03E4				
E800	00EC	01EC	02EC	03EC				
F000	00F4	01F4	02F4	03F4				
F800	00FC	01FC	02FC	03FC				

Dump Part of Supervisor Partition to Printer (cont.)

STORAGE MAP: \$SYSCOM AT ADDRESS 3420

PART#	NAME	ADDR	PAGES	TCB
P1	**PART**	9F00	97	
P1	\$TRAP	9F00	21	B2B4
P1	**FREE**	B400	76	
P2	**PART**	0000	256	
P2	**FREE**	0000	256	
P3	**PART**	0000	256	
P3	**FREE**	0000	256	
P4	**PART**	0000	256	
P4	\$SMMAIN	0000	4	02F8
P4	\$SMLOG	0400	34	1978
P4	**FREE**	2600	218	

\$DUMP

Dump Part of Supervisor Partition to Printer (cont.)

TERMINAL LIST:

NAME	CCB	ID	ADDR
\$SYSLOG	1876	0406	0004
\$TERM1	1A82	040E	0005
\$TERM2	0C32	040E	0006
\$SYSLOGA	1DCC	0010	0000
\$SYSPRTR	1F86	0306	0001

DSK(ETTE) LIST:

NAME	DDB	TYPE	ID	ORG	SIZE	LIB	ADDR
EDX001	165E	PRI	0106	0	75	27	0002
EDX002	16F0	PRI	00AA	0	92	2461	0003 IP
ASMLIB	1782	SEC		92	16	1	
SUPLIB	17A2	SEC		108	16	1	
MACLIB	17C2	SEC		124	78	1	
EDX002	17E2	SEC		202	102	1	

SUPV BEGINNING AT ADDRESS 0000 FOR 139 PAGES

0000	6802	882A	0000	0000	8968	8826	8969	8826
0010	0000	0000	8968	8826	0A76	0A12	8968	8826
	SAME AS ABOVE								
0100	12DC	8BC2	0004	0006	1010	6A08	03F8	5B22	...B..

ANOTHER AREA? (Y/N): N

Note

Note: \$DUMP allows you to request several partial dumps. If a 'Y' response is entered, then \$DUMP prompts you for the starting and ending addresses that are to be dumped. See note 4.

`$EDIT1` AND `$EDIT1N` - LINE EDITORS

`$EDIT1` and `$EDIT1N` provide a text editing facility (primarily used for source program entry and editing) that can be invoked simultaneously with the execution of other programs. The Host Communication Facility related version (`$EDIT1`) provides a few commands for data communication using the Host Communications Facility IUP on the System/370 so that almost the entire process of program preparation can be controlled from a Series/1 terminal. The native program preparation version (`$EDIT1N`) produces members that can be processed by the Series/1 assembler.

Both versions work with 80-character lines that are line numbered in positions 73-80 and are invoked by the `$L` command.

Data Set Requirements

One work data set is required by the editing facility and must be allocated on disk or diskette using `$DISKUT1`. You are prompted for its name when either version is loaded. This data set contains both your data and some index information during the editing session, and the size (number of records) of the data set determines the maximum number of data records that it can contain. It is divided into three parts:

1. One header record
2. A series of index records (32 entries per record)
3. A series of data records (3 entries per record)

The required data set size can be calculated as follows: number of text lines (n) divided by 30, times 11, plus 1 ($(n/30 \times 11) + 1$).

Sequence of Operations

When the edit program is loaded, it prompts you for the name of the work data set to be used. If an existing data set is to be edited, the READ command should be used to copy the data set to the work data set. For a new data set, edit mode should be invoked. The contents of the work data set can be printed using the LIST command.

The EDIT command is used to enter edit mode. Edit subcommands are then recognized until terminated by the END command.

Note: You should use the VERIFY ON subcommand until you become familiar with the editing process.

The TABSET subcommand is used, if desired, to specify the tab character and tab column. This eliminates the entry of blanks when a substantial amount of the text to be entered is in tabular format or begins in a particular column.

Data can be entered a line at a time under the INPUT subcommand (recommended for new data sets and bulk sequential updates because of the automatic prompting feature) or by using the line editing function (for single line corrections). Portions of the edited data can be listed at the terminal using the LIST command.

The position of the current line pointer is controlled by the FIND, TOP, BOTTOM, UP, and DOWN subcommands.

Edit mode is terminated with the END command. When the text has been edited, copy the work data set to a permanent data set using either the WRITE or SAVE subcommand. The work data set is in a blocked format that is incompatible with most Event Driven Executive functions. Automatic translation from text editor format to source statement format is performed.

The following figure shows the primary commands and subcommands available under \$EDIT1/\$EDIT1N.

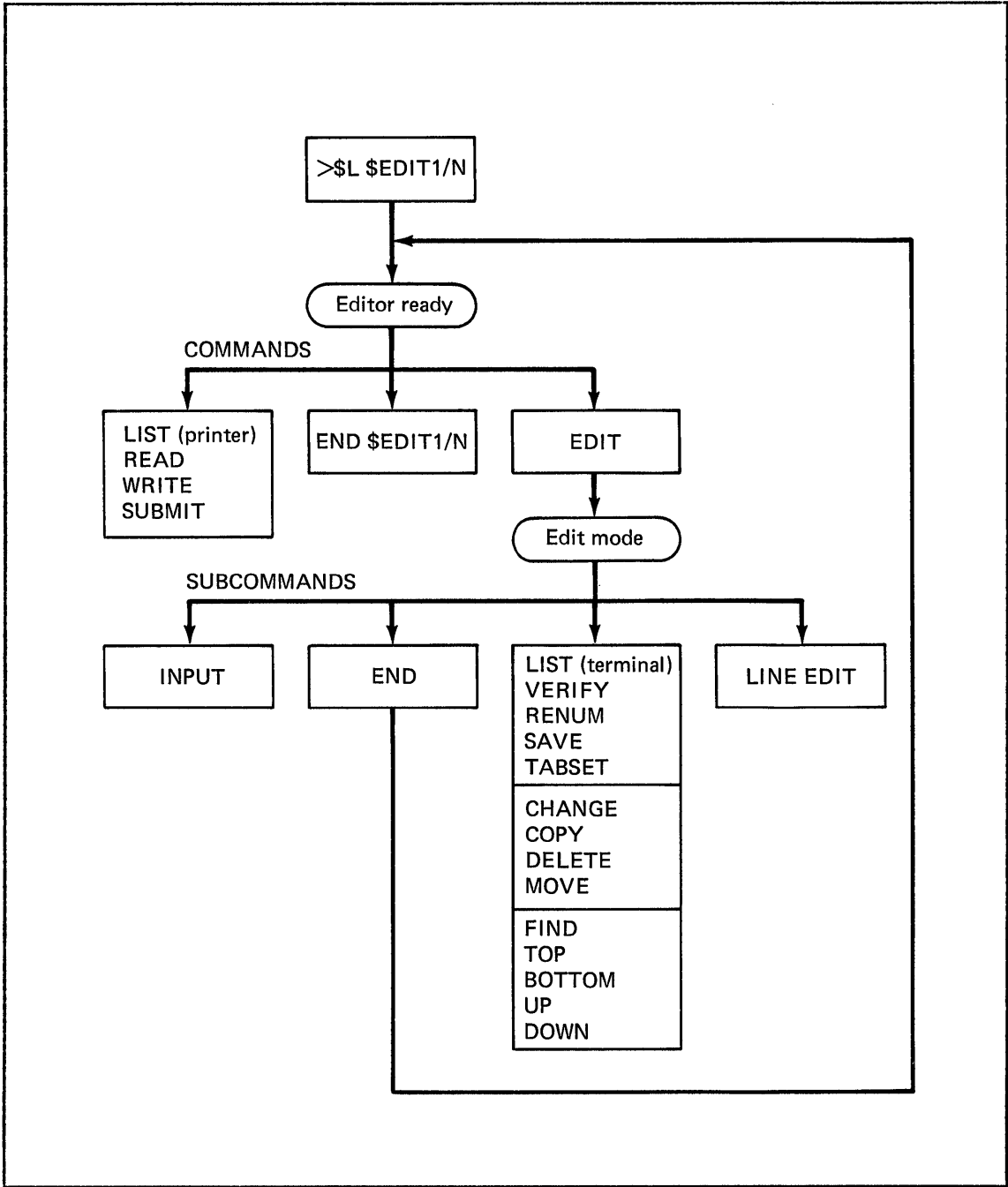


Figure 20. \$EDIT1/\$EDIT1N Commands and Subcommands

Special Control Keys

1. End of Line Character (see note below). The Carriage Return Key (CR)/ENTER is used to end an input line.
2. Line Delete Character (see note below). The Delete Key (DEL) of certain teletypewriter terminals is used to delete an input line.

Note: CR and DEL keys can be defined in the TERMINAL statement. See the System Guide.

3. Character Delete Character. The backspace (BS) key on terminals is used for the character delete function. On teletypewriter terminals, use the CTRL and H keys simultaneously.
4. Tabulation Character. You can set the TAB character to the character of your choice. '%' is the default TAB character. Columns 10, 20, 40, and 72 are the default TAB columns.
5. ATTN Key (4978/4979) or ESC or ALT MODE Key (teletypewriter terminals). The subcommands CHANGE, FIND, and LIST, described below, can be cancelled by pressing the ATTN/ESC key and entering, as a special system utility function, the two character code CA. This feature is useful, for example, to terminate a long listing.

Editor Commands

The editor commands are described in the following pages. Unless specifically indicated, the commands apply to both the host and native versions of this utility. The editor commands are:

COMMAND	DESCRIPTION
EDIT	Enters edit mode; allows edit subcommands
END	Terminates \$EDIT1/\$EDITN
LIST	Lists the work data set on the system printer
READ	Reads a source data set into the work data set
SUBMIT	Submits a job to the host batch job stream
WRITE	Writes the work data set into a source data set

\$EDIT1 and \$EDIT1N

EDIT - Enter Edit Mode

EDIT is used to begin editing source data.

Syntax

EDIT	OLD/NEW
Required:	None
Defaults:	NEW when using a newly allocated work data set. OLD when using an old work data set.
Alias:	E,ED

Operands Description

OLD Indicates that data exists in the data set you want to modify.

NEW Indicates that you are creating new data.

Notes:

1. The EDIT command must be entered before the editor subcommands can be used.
2. When in edit mode, the subcommand END or SAVE must be entered before the editor commands listed on the preceding page can be used.

END - End \$EDIT1/\$EDIT1N

END terminates execution of \$EDIT1 or \$EDIT1N.

The contents of the edit work data set are unchanged. You can reinvoke \$EDIT1/N at a later time and continue.

Syntax

END

Required: None

Defaults: None

Alias: EN

No operands are required.

\$EDIT1 and \$EDITN

LIST - List Work Data Set

LIST prints all or part of the work data set on the system printer (\$SYSPRTR). A single line number can be specified or a pair of line numbers can be entered to specify a line range. If no line numbers are specified, the entire data set is listed. Listing can be terminated by entering ATTN and CA. Note a similarity to the EDIT subcommand. As a command following READY, the data set is printed on \$SYSPRTR. As a subcommand following EDIT, the data set is displayed on your terminal.

Syntax

```
LIST      line-spec
```

```
Required: None
```

```
Defaults: None
```

```
Alias:    L,LI
```

Operands Description

line-spec '*' (for the current line) or 'line-number' to indicate a single line to be listed. '* COUNT' or 'linenum1 linenum2' to display a range of lines. The entire data set is printed if this operand is omitted.

Examples

```
LIST 10 100  
L * 5  
L *  
LI
```

READ - Retrieve Host Data Set (\$EDIT1)

READ retrieves a data set from the host system and stores it in your Series/1 work data set.

The Host Communications Facility on the System/370 is required.

Syntax

READ dsname

Required: None

Defaults: If dsname is omitted, the system prompts you

Alias: None

Operands Description

dsname The fully qualified name of the host data set to be retrieved. It must contain fixed length records, 80 bytes in length, with line numbers in columns 73-80.

You can enter the command and name together on the same line or enter the command READ and the system prompts for the data set name.

\$EDIT1 and \$EDITN

READ - Retrieve Series/1 Data Set (\$EDITN)

READ retrieves a named data set from a volume on the Series/1 disk or diskette and stores it in a Series/1 work data set

Syntax

READ dsname volname

Required: None - System prompts for operands

Default:

Alias: R, RE

Operands Description

dsname Name of data set to be retrieved.

volname Name of the volume containing the data set to be retrieved.

Note: These operands are entered as responses to system prompts.

SUBMIT - Submit Job to Host (\$EDIT1)

SUBMIT injects a job (JCL and optional data) into the host batch job stream.

The Host Communications Facility on the System/370 is required.

Note: This option is only to be used in systems with a HASP or JES/Host Communication Facility interface.

Syntax

SUBMIT	dsname
SUBMIT	DIRECT

Operands

Description

dsname The fully qualified name of the host data set, the contents of which are to be entered into the job stream. This data set must contain fixed length, 80 byte records.

DIRECT If specified, the contents of your edit work data set are transferred directly to the host job stream.

\$EDIT1 and \$EDIT1N

WRITE - Write Work Data Set to Host (\$EDIT1)

WRITE transfers your Series/1 work data set to a host data set. It is assumed that your data set has been created or edited with the \$EDIT1 utility program.

The Host Communications Facility on the System/370 is required.

If a host data set has been previously specified, you are asked if you wish to reuse it. If not, or if one was not previously specified, you are prompted for a new host data set name.

Syntax

WRITE dsname

Operands

Description

dsname The fully qualified name of the target host data set. This data set should contain fixed length 80-byte records.

You can enter the command and name together on the same line or enter only the command WRITE and the system prompts you for the data set name.

WRITE - Write Work Data Set to Series/1 Data Set (\$EDIT1N)

WRITE copies the Series/1 work data set to a named data set in a Series/1 disk or diskette volume.

Syntax

WRITE
Required: None - System prompts you for operands
Default: Copy to the originating data set, if any
Alias: W, WR

No operands are required.

The following prompt is issued by EDIT1N:

WRITE TO 'READVOL' OR 'READVS'?YES/NO

where 'READVOL' is the originating volume and 'READVS' is the originating data set. This prompt is issued only if the work data set was initialized via the READ command. If the response is 'NO' or the data set is new, the following prompt is issued:

ENTER VOLUME LABEL: volname

\$EDIT1 and \$EDIT1N

Edit Mode Subcommands

The subcommands used to edit your work data set while in EDIT mode are described as follows:

Subcommand	Operands
BOTTOM	
CHANGE	line-spec /text1/text2/ALL
COPY	line-spec
DELETE	line-spec
DOWN	count
END	
FIND	/character-string/
INPUT	line-number increment
Line Editing	line-number character-string
LIST	line-spec
MOVE	line-spec
RENUM	new-line-number increment
SAVE	
TABSET	ON(integer list), OFF, CH(character)
TOP	
UP	count
VERIFY	ON/OFF

BOTTOM - Set Line Pointer to Bottom

BOTTOM repositions the current line pointer (*) to the last line of the data set being edited.

Syntax

BOTTOM

Required: None

Defaults: None

Alias: B,BO

No operands are required.

\$EDIT1 and \$EDITIN

CHANGE - Change Character String

CHANGE modifies a character string in a line or range of lines.

Syntax

```
CHANGE    line-spec /text1/text2/ALL
```

```
Required: /text1/text2
```

```
Defaults: line-spec defaults to *.
```

```
Alias:    C,CH
```

Operands Description

line-spec '*' or blank for the current line.

'* count' or 'linenum1 linenum2' for a range of lines.

'line-number' for a particular line.

/text1/text2/ALL

'/' can be any non-numeric character except BLANK, TAB, and ASTERISK. It is not a part of, and cannot appear within the character strings 'text1' and 'text2'. The line or range of lines is searched for 'text1', which, if found, is replaced by 'text2'. Note that the same character must be used for both delimiters in any one change command.

The keyword 'ALL' is optional and causes every occurrence of 'text1' to be replaced in the line(s).

Two adjoining delimiters denote a null operand. If text1 is a null operand, then text2 is inserted at the start of the line. The line is shifted right. If text2 is a null operand and text1 is specified, text2 is removed from the line and the rest of the line shifted left.

Example

```
C 20 /ABC/ADC/  
C 100 250 =/*=//=ALL  
C * //XYZ  
C /PROG/PGM/
```

\$EDIT1 and \$EDIT1N

COPY - Copy Text

COPY duplicates text, from one location in a data set, at another location within that data set. The 'from' and 'to' text both remain in the data set.

Syntax

```
COPY      linenum1 linenum2 linenum3  
Required: linenum1 linenum3  
Defaults: linenum1 linenum3 defaults to  
          a single line copy of '1' to '3'.  
Alias:    CO
```

Operands Description

linenum1 The first line of text to be copied.

linenum2 The last line of text to be copied.

linenum3 The line of text after which the copied text is to be placed.

All specified line numbers must exist. 'linenum2' must be equal to or greater than 'linenum1'. 'linenum3' must be less than 'linenum1' or equal to or greater than 'linenum2' when three line numbers are specified. The data set is renumbered with standard specifications. The original 'linenum2' is listed with its new line number on exit.

Example

```
CO 100 300 60
CO 120 250 820
CO 150 150 310
COPY 150 310
```

Note: The last two examples are equivalent.

\$EDIT1 and \$EDITN

DELETE - Delete Text

DELETE removes records from the data set. The current line pointer (*) is repositioned prior to the deleted lines.

Syntax

```
DELETE    line-spec
```

```
Required: None
```

```
Defaults: *
```

```
Alias:    DE
```

Operands Description

line-spec * for current line.

'* count' or 'linenum1 linenum2' for a range of lines.

'line-number' for a particular line.

Example

```
DELETE *  
DE * 4  
DE 100 150  
DE 125
```

DOWN - Move Line Pointer Down

DOWN moves the current line pointer (*) toward the end of the data set.

Syntax

```
DOWN      count

Required: None
Defaults: 'COUNT' defaults to 1.
Alias:    DO
```

Operands Description

count Specifies the number of lines the current line pointer is to be moved.

Example

```
DOWN 5
DO 10
```

\$EDIT1 and \$EDITIN

END - Exit Edit Mode

END requests that the EDIT mode be terminated. The editor commands can now be used relative to your finished source data. To save or list your data set or to write or submit your data set to the host, see "Editor Commands" on page 173. The contents of the work data set remain unchanged. You can re-enter the edit mode using the EDIT command and continue editing the work data set.

Syntax

END

Required: None

Defaults: None

Alias: EN

No operands are required.

FIND - Find Character String

FIND searches for a specified character string beginning with the current line, if operands are specified (see Syntax.) The current line pointer (*) is moved to the first line found to contain the string. The search is made at every position within each line.

Note: VERIFY should be set to ON when using the FIND command.

Syntax

FIND =char-string=

Required: None

Defaults: If no operands are specified, those specified on the last previous issue of the FIND subcommand are assumed. The search begins at the line following the current line.

Alias: F,FI

Operands Description

=char-string=

The string delimiter can be chosen to be any non-numeric character, except BLANK, TAB, or ASTERISK and which does not appear within the specified character string. The second occurrence can be replaced by a carriage return. Note that both delimiters must be the same character.

Example

```
FIND /START/  
F  
FI =DATA X'00F1' =
```

\$EDIT1 and \$EDIT1N

INPUT - Input Text

INPUT allows lines to be added or replaced. INPUT can be used any time in edit mode by pressing the ENTER key. Lines are then added to the end of the data set.

To terminate INPUT mode, press the ENTER key immediately after you receive the prompt for the next line number to be entered.

Syntax

```
INPUT      line-number  increment
           or
           * increment
```

Required: None

Defaults: Increment defaults to previous or 10

Alias: I,IN

Operands Description

line-number

The first line inserted will have this number, or this number plus the increment if the specified line number already exists.

increment The increment for numbering inserted lines. The default is the previously specified increment or 10 if not specified.

* Lines are to be inserted at the current line position plus the default increment. If no operands are specified, lines are to be inserted at the end of the data set plus the default increment.

Example

```
INPUT * 1  
IN 100 5  
I 20  
I
```

\$EDIT1 and \$EDIT1N

LIST - List Work Data Set

LIST displays, at the terminal, lines of the data set being edited.

Syntax

```
LIST      line-spec
```

Required: None

Defaults: line-spec defaults to entire data set

Alias: L,LI

Operands Description

line-spec (*) or line-number to indicate a single line to be listed. '* count' or 'linenum1 linenum2' to display a range of lines.

Example

```
LIST 10 100  
L * 5  
L *  
LI
```

MOVE - Move Text

MOVE moves text from one location in a data set to another location within that data set. The 'from' text is deleted and only the 'to' text remains in the data set.

Syntax

```
MOVE      linenum1 linenum2 linenum3

Required: linenum1 linenum3
Defaults: linenum1 linenum3 defaults to move one line.
Alias:    MO
```

<u>Operands</u>	<u>Description</u>
-----------------	--------------------

linenum1	The first line of text to be moved.
----------	-------------------------------------

linenum2	The last line of text to be moved.
----------	------------------------------------

linenum3	The line of text after which the moved text is to be placed.
----------	--

All specified line numbers must exist. 'linenum2' must be equal to or greater than 'linenum1'. 'linenum3' must be less than 'linenum1' or greater than 'linenum2' when three line members are specified. The data set is renumbered with standard specifications. The original 'linenum2' is listed with its new line number on exit.

Example

```
MO 100 300 60
MO 120 250 820
MO 87 87 310
MOVE 87 310
```

Note: The last two examples are equivalent.

\$EDIT1 and \$EDIT1N

RENUM - Renumber Work Data Set

RENUM renumbers each line of a line numbered data set or assigns line numbers to each line of an unnumbered data set.

Syntax

RENUM new-line-number increment

Required: None

Defaults: Both new-line-number and increment
 default to 10.

Alias: R, RE

Note: 'new-line-number' is required if 'increment' is
specified.

Operands Description

new-line-number

The sequence number to be assigned to the first line
processed.

increment The increment to be used in renumbering.

Example

```
RENUM 10 10
RE 100 5
RENUM
R
```

SAVE - Save Work Data Set

SAVE writes the current contents of the work data set to a host data set with the host related version (\$EDIT1) or to a Series/1 data set with the native related version (\$EDIT1N).

If a data set has been previously specified (e.g., in a READ command), you are asked if you wish to write onto that data set; otherwise, you are prompted for a new data set name.

Syntax

```
SAVE      dsname

Required: None
Defaults: None
Alias:    S, SA
```

Operands Description

dsname When using \$EDIT1, you are prompted for the target host data set name. It must be a fully qualified data set name.

When using \$EDIT1N, the target data set must have been previously allocated in a volume on a Series/1 disk or diskette. The data set should contain fixed length records, 80 bytes in length. You are prompted for the target volume name.

Example

```
SA
S
SAVE
```

TABSET - Set Tabs

TABSET reestablishes tab values or nullifies existing tab values. The tabulation character and tab stop values are maintained as part of your work data set. (They can be changed later).

The tab character can be entered anywhere in the data line under the INPUT subcommand or line editing function. It causes a skip to the next tab position when the data line is entered into the work data set. The resulting line is not visible, but can be displayed if desired.

Syntax

```
TABSET    ON(integer-list)
TABSET    OFF
TABSET    CH(tab-character)
```

Required: ON, OFF, or CH

Defaults: None

Alias: TA

Operands Description

integer-list

The relative column positions in each line to which tab values are to be set. Initial system defaults are 10, 20, 40, and 72.

tab-character

A new tab character. The standard is a percent sign.

OFF Terminates the tab function.

Examples

```
TABSET ON(10 20 40 72)
TA ON(10 16 31)
TA CH(#)
TA OFF
```

```
TABSET ON(10 20)
36 %TAB POSITION 1
INPUT 37 1
INPUT
00037 %%TAB POSITION 2
00038

EDIT
LIST 36 37
00036          TAB POSITION 1
00037          TAB POSITION 2
```

\$EDIT1 and \$EDITIN

TOP - Set Line Pointer to Top

TOP positions the current line pointer (*) before the first line of the data set.

TOP

Required: None

Defaults: None

Alias: TO

No operands are required.

Note: If VERIFY is ON, no line is printed because the current line number precedes the first line.

UP - Move Line Pointer Up

UP moves the current line pointer (*) toward the start of the data set.

Syntax

```
UP          count

Required:  None
Defaults:  Count defaults to 1
Alias:     U
```

Operands Description

count The number of lines that the current line pointer (*) is to be moved.

Example

```
UP 10
```

\$EDIT1 and \$EDIT1N

VERIFY - Display Changes on Terminal

VERIFY causes the changes you made to be shown on the terminal (ON), or not shown (OFF). Verification is off until it is invoked the first time during an edit.

Syntax

VERIFY ON/OFF

Required: None

Defaults: ON

Alias: V,VE

Operands Description

- | | |
|-----|--|
| ON | Each time the position of the current line pointer (*) changes, the line to which it moves should be printed. In addition, modifications made in fields of records using the 'character-string' or 'text' forms of the CHANGE subcommand are verified. |
| OFF | Changes of the position of the current line pointer (*), and of fields of records by means of the CHANGE subcommand, are not to be verified. |

Example

```
V ON
V
V OFF
VERIFY
```

Line Editing Commands

The line editing commands allow a single line to be added, replaced, or deleted from the data set being edited.

Note: Line editing functions are not subcommands.

Syntax

```
line-number character-string
```

Required: line-number

Defaults: None

Note: If specified, 'character-string' must be separated from 'line-number' by a single blank or tab.

<u>Operands</u>	<u>Description</u>
-----------------	--------------------

line-number

'line-number' with an immediate carriage return deletes the line having the specified number (it does nothing if the line does not exist).

'Line-number', followed by a character string, adds the string to the data set. If a line having the specified number already exists, it is replaced.

character-string

The text of the line to be added.

\$EDIT1 and \$EDITIN

Examples

Add Line (Line #12345 does not exist)

12345 This line is being added

Delete Line 12345

12345

Replace line 12345

12345 This line replaces 12345

\$FONT - PROCESS 4978 CHARACTER IMAGE TABLES

\$FONT, a special purpose utility, creates or modifies character image tables for the 4978 display station. Each character image is defined by a dot matrix that is coded into eight bytes of data. The entire table of codes requires 2048 bytes of storage. For details on the associated 4978 hardware, see the Bibliography for the 4978 Display Station manuals.

\$FONT requires one preallocated data set of 2048 bytes (8 records). The data set can contain a character image table, or it can represent storage for a new table to be constructed.

\$FONT Commands

The commands available under **\$FONT** are listed below. To display this list at your terminal, enter a question mark in response to the prompting message **COMMAND (?)**:

```
COMMAND (?): ?  
  
DISP -- DISPLAY TABLE  
EDIT -- ENTER EDIT MODE  
SAVE -- SAVE TABLE  
PUT  -- LOAD TABLE INTO DEVICE  
GET  -- READ TABLE FROM DEVICE  
END  -- END PROGRAM  
  
COMMAND (?):
```

After the commands are displayed, you are again prompted with **COMMAND (?)**:. You respond with the command of your choice (for example, **DISP**).

DISP - Display Table

The character images defined by the table are displayed along with their associated EBCDIC codes.

EDIT - Enter Edit Mode

Edit mode enlarges the display for modification of the dot matrix patterns (a complete description of the edit mode func-

\$FONT

tions is given following this section).

END - End Program

When you enter the END subcommand, the system displays the message SAVE TABLE? if edit mode has been entered at least once since the last SAVE operation. This allows you to save the current table, if desired, before ending the program.

GET - Read Table from Device

The image store is read from a (4978) terminal and becomes the current table. The GET command followed by SAVE provides a means for initializing a data set with a character image table.

PUT - Load Table into Device

The current table is written to the image store of a (4978) terminal. A terminal name can follow the command on the same line, or you are prompted for it. The image store is a table which contains codes for generating characters or images for the 4978 terminal.

SAVE - Save Table

The current table, reflecting any changes made during edit mode, is written to the data set designated at load time.

Edit Mode

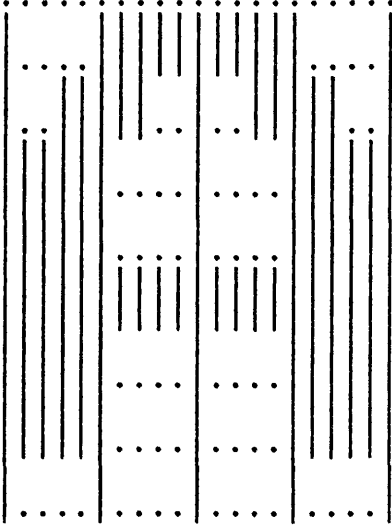
When you enter edit mode, a 4X8 grid is displayed in the center of the screen. For each grid row, the centers of seven overlapping dot areas occur at the centers of each of the four squares and at each of the three interior grid lines. (See 4978-1 Display Station and Attachment General Information for details on the character matrix.) Dots are represented with the following pattern:



Screen Format

The following complete screen format appears after you select a character for display:

PF1	--	TAB FORWARD
PF2	--	TAB BACK
PF3	--	NEXT LINE
PF4	--	INVERT DOT
ENTER	--	SET PATTERN
PF5	--	COMMAND MODE
CODE	A	(C1)



A

Character Image Display and Modification

You can display and modify the character images as follows:

1. Specify the character to be viewed either by entering a character at the cursor position (the alphanumeric field), or by entering the hexadecimal representation of the EBCDIC code between the parentheses (the hex field).
2. Press the ENTER key. The character image, as determined by the table designated at load time, is displayed on the grid. For 4978 terminals, the image is also displayed beneath the grid in normal dimensions. If the character was designated with an alphanumeric field entry, then the EBCDIC code as determined by the keyboard in use is displayed in the hex field.
3. The program function keys can then be used according to the displayed instructions to modify the dot pattern. Note that adjacent or overlapping dots appear as a continuous solid line on the grid. This reflects a feature of the character generation hardware on the 4978 display.
4. When the desired pattern has been constructed, press ENTER. The word CODE on the code entry line changes to SET, displayed at high intensity. At this point, press ENTER again and the new pattern replaces the existing pattern for the specified code. To associate the pattern with a new code, alter either the alphabetic or hex field before pressing ENTER.

§FSEDIT - FULL SCREEN EDITOR

§FSEDIT is a full-screen text editing utility that helps you develop and modify programs. It operates the terminal as a static screen device and therefore must be run from a terminal with static screen capability (4978/4979).

With §FSEDIT you can:

- Edit programs using a full screen
- Scroll information forward and backward
- Use PF (program function) keys for frequently used functions
- Insert a mask for prefilling inserted lines
- Merge data from other data sets
- Communicate with a System/370 in conjunction with the Host Communication Facility IUP installed on the host System/370

Note: To use §FSEDIT, the modules §FSUTIL, §FSUTLN, §FSHELP and §FSIM1 to §FSIM8 must reside on the system IPL volume.

Data Set Requirements

§FSEDIT requires a preallocated work data set for use as a text edit work area. This work data set is automatically allocated if §FSEDIT is invoked by the session manager. If you use the \$L command, you can provide the name of the work data set when you enter the load request, or when §FSEDIT issues a prompt for the name of the work data set.

Text data (source statements) within this work data set are in a special text editor format, identical to that used by the §EDIT1N text editor. See "Data Set Requirements" on page 169. Data within a work data set can be edited by either §EDIT1N or §FSEDIT.

Note: §FSEDIT uses source data sets of 80-character lines that are line numbered in positions 73-80 for host or Series/1 data sets or in positions 1-6 (COBOL convention) for host data sets.

\$FSEEDIT

When you end a text editing utility session, save the contents of the work data set in a source data set on disk, diskette or tape. If a new data set is used, a prompt is issued asking if the work data set is to be written to the same disk, diskette, or tape it was read from. If your response is N, a prompt is issued asking for a new data set to be used. If the data set does not exist on the volume specified, \$FSEEDIT creates it automatically. Automatic translation from text editor format to source statement format is performed.

Caution: If you write the work data set out to a multifile tape, all files following this data set will no longer be accessible.

Scrolling

During editing and browsing, the information to be displayed usually exceeds the size of the display screen. Scrolling allows you to page up or down through the information. Two PF keys are used for this purpose -- one for each direction. Whenever scrolling is allowed, a scroll amount, displayed at the end of the second line of the display, shows the number of lines scrolled with each use of a scroll key.

PAGE or P Specifies scrolling one page (22 lines).

HALF or H Specified scrolling a half page (11 lines).

MAX or M Specifies scrolling to the top or bottom of the data set.

n Specifies number of scrolling lines

You can change the scroll amount by moving the cursor to the scroll field and overtyping the amount currently displayed. To change the scroll amount, type over the first character with a P, H or M to change the scroll amount to a page, half page, or maximum, respectively. To specify a number of lines to scroll, overtype the field with the number of lines desired.

In browse mode, the scroll amount is initialized to PAGE; in edit mode, it is initialized to HALF. When you make a change, the new value remains in effect for the remainder of the session unless you change it. The value for MAX is an exception; following a MAX scroll, the scroll amount defaults back to the initial value.

Program Function Keys

The ATTN and six program function keys are used to request commonly used or special \$FSEDIT operations as follows:

- PF1** Redisplays the screen image. All changes are ignored and the original data is displayed.
- PF2** SCROLL UP scrolls up the amount shown in scroll amount field.
- PF3** SCROLL DOWN scrolls down the amount shown in the scroll amount field.
- PF4** REPEAT FIND repeats the action of the previous FIND primary command.
- PF5** REPEAT CHANGE repeats the action of the previous CHANGE primary command (Applies only to edit mode).
- PF6** PRINT SCREEN prints the screen image on the system printer (\$SYSPRTR). The use of PF6 for hardcopy may have been changed by use of the TERMINAL HDCOPY statement (see the System Guide).
- ATTN CA** Cancel the list option. Pressing the ATTN key and typing CA stops the list option of \$FSEDIT and returns to the primary option selection list.

Note: The PF2 - PF5 keys are active only during browse and edit modes. PF1 use is only meaningful during edit mode.

\$FSEDIT

\$FSEDIT Options and Command Summary

This summary shows the options and commands available under \$FSEDIT. A description of each follows.

Primary Options:

BROWSE - Display Data Set
EDIT - Edit Data Set
END - Terminate \$FSEDIT
HELP - Display Commands
LIST - Print Data Set
MERGE - Merge Data from a Source Data Set
READ - Retrieve Data Set From Native/Host
SUBMIT - Submit Job to Host
WRITE - Transfer Data Set to Native/Host

BROWSE Primary Commands:

END - Same as MENU
FIND - Find a Character String
LOCATE - Find Data by Line Number
MENU - Return to Primary Option Menu

EDIT Primary Commands:

CHANGE - Modify Character Strings
CLEAR - Clear Work Data Set
END - Return to Primary Option Menu
FIND - Find Character String
LOCATE - Find Data by Line Number
MENU - Return to Primary Option Menu
RENUM - Renumber Data Set
RESET - Clear Line Commands

EDIT Line Commands:

A,B - Define Copy/Move Destination
C,CC - Copy Lines of Text
COLS - Display Columns
D,DD - Delete Text
I,II - Insert Text
M,MM - Move Line of Text
MASK - Display Insert Mask

Primary Option Menu

When SFSEEDIT is loaded, the following primary option menu (selection list) is displayed. To select the desired function, enter the number of the option in the 'Select Option' input field.

```
----- SFSEEDIT PRIMARY OPTION MENU -----  
SELECT OPTION ==> 1  
  
1  BROWSE - DISPLAY DATASET  
2  EDIT   - CREATE OR CHANGE DATASET  
3  READ   - READ DATASET FROM HOST/NATIVE (H/N)  
4  WRITE  - WRITE DATASET TO HOST/NATIVE (H/N)  
5  SUBMIT - SUBMIT BATCH JOB TO HOST SYSTEM  
6  LIST   - PRINT DATASET ON SYSTEM PRINTER  
7  MERGE  - MERGE DATA FROM A SOURCE DATASET  
8  END    - TERMINATE SFSEEDIT  
9  HELP   - DISPLAY TUTORIAL
```

Option 1 - BROWSE

The BROWSE mode allows you to display and examine a source file in the work data set, but prevents the possibility of changing it.

In this mode, you can view all parts of the work data set using the scrolling function (invoked by pressing PF keys).

In addition, two primary commands are used to locate specific information within the data set.

FIND Searches for a designated text string

LOCATE Searches for a designated line number.

These primary option commands are discussed under "Primary Commands" on page 218.

During browsing, the current number of lines in your data set and the maximum number of lines the work data set can hold is displayed on the top line of the display, following the data set name and the volume identification. (The maximum number of lines is displayed in parentheses.)

\$FSEDIT

Browsing is terminated by entering the primary command MENU in the command input field to return to the Primary Option Menu.

Option 2 - EDIT

In the edit mode, you can modify an existing source data set or create a new one. To do this, you use:

- Program function keys for two-way scrolling and repeat change and find
- Primary commands (CHANGE, CLEAR, END, FIND, LOCATE, MENU, RENUM, and RESET)
- Line edit commands to manipulate whole lines or blocks of lines.

Creating a Source Data Set

To create a new source data set, you enter EDIT mode (option 2 of the primary option menu) with an empty data set (the work data set specified when \$FSEDIT was invoked). Because the work data set is empty, the editor assumes insertion (creation) of lines is desired and the INSERT function is active. Following is an example of the initial display when editing an empty data set.

```

EDIT --- EDITWORK, EDX002      0(24)--- COLUMNS 001 072
COMMAND INPUT ==>                SCROLL ==>HALF
***** ***** TOP OF DATA *****
..... -
***** ***** BOTTOM OF DATA *****

```

The top line of the screen, from left to right, displays utility mode (EDIT), the name and volume of the work data set (EDITWORK,EDX002), the number of source statements in the work data set, and in parentheses, the total number of statements the data set will hold.

The cursor is positioned at character position 1 of the insert line. After you enter information on this line, press the ENTER key to write the data on the screen to your work data set.

The utility then numbers the entered line and sets up for the next insert line.

As you continue in this manner, a new insert line is readied each time the preceding line is entered (ENTER key). The insert (creation) operation is terminated by pressing the ENTER key without entering anything on the new insert line.

Note: \$FSEDIT does not distinguish between input mode and edit mode during editing operations and data on the screen can be changed at any time.

To save the source statements just created, return to the primary option menu (using the MENU command). Use the WRITE option to save the newly created data set. You are prompted for the name of the target data set and volume.

Modifying an Existing Data Set

To modify (edit) an existing data set, it must first be read into the work data set using option 3 (READ) of the primary option menu.

Once your data set has been read into the work data set, you can locate and change information by scrolling the data set by pressing the PF keys. The PF keys and definitions are described under "Program Function Keys" on page 211.

To modify data on the screen, move the cursor to the desired location and enter the new information. Several lines can be changed before pressing the ENTER key. A single line or blocks of lines can be deleted, inserted, duplicated, or rearranged using the edit line commands. These are discussed under "Edit Line Commands" on page 226.

For general editing purposes, primary edit commands are used to find and change designated character strings and to change the line numbering sequence. These commands are discussed under "Primary Commands" on page 218.

After you finish modifying the data set, use WRITE to save the data in the same data set it was read from or to a new data set on disk or diskette.

Editing is terminated by entering the primary command END or MENU in the command input field, which returns you to the primary option menu.

Option 3 - READ

The READ option retrieves a data set from either a host system or a data set on disk, diskette, or tape on the native Series/1 system and stores it in your work data set. The primary option menu remains on the display and the area below it is used to prompt for the data set name and to provide information. The data set name entered must be fully qualified and must contain fixed length 80-byte records.

Line numbers for native data sets must be in columns 73-80. For host data sets, line numbers can be in either columns 1-6 or 73-80. If the line numbers in the data set exceed the maximum allowed by SFSEEDIT (32767), the data is automatically renumbered with a smaller line number increment.

When the READ is completed or terminated because of an error, the number of lines transferred, or the appropriate error message, is displayed and the cursor is moved to the 'Select Option' input field. This indicates the completion of the READ function and another option can be selected. The READ to host requires the Host Communications Facility on the System/370.

Option 4 - WRITE

The WRITE option transfers the contents of the work data set to a host/native data set. A prompt is issued asking if the work data set is to be written to the same disk, diskette, or tape data set it was read from. If the response is NO (N), a prompt is issued asking for a new data set to be used. If the data set does not exist on the volume specified, it is created automatically. The WRITE to host requires the Host Communications Facility on the System/370.

Note: The WRITE option does not destroy the contents of the work data set. Therefore, the WRITE option can be directed to another data set to obtain a backup copy. The WRITE option can also be used following further editing of the work data set.

Caution.

1. If you increase the contents of the work data set so that it is too big to be written back to the source data set, SFSEEDIT deletes the source data set and attempts to reallocate it with enough space to save the work data set contents. If the allocation fails, the original source data set is lost. However, the work data set remains intact and can be saved in a suitable source data set.
2. If you write the work data set out to a multifile tape, all files following this data set will no longer be accessible.

Option 5 - SUBMIT

The SUBMIT option injects a job (JCL and optional data) into the host job stream. The display and operation are similar to the READ and WRITE commands. The data set name entered must be the fully qualified name of the host data set containing the JCL to be submitted. If the keyword DIRECT is entered instead of a data set name, the contents of the work data set are transferred directly into the host job stream. The SUBMIT to host requires the Host Communications Facility on the System/370.

Note: The DIRECT option is only to be used in systems with a HASP or JES2 interface.

Option 6 - LIST

The LIST option prints the entire contents of the work data set on the hardcopy device assigned to the terminal. (The listing can be terminated at any time by pressing the ATTN key and typing CA.)

Option 7 - MERGE

The MERGE option merges all, or part, of a source data set into the current edit work data set. You are prompted for the names of the Series/1 source data set and volume. If the specified data set is found, you are then prompted for the first and last line numbers of the data to be copied. If the entire data set is to be merged, an '*' can be entered instead of the line number specifications. You are also prompted for the line number of the current work data set after which the data is to be added. The specification of an asterisk is only to be used for the source data set. (If the format of the line number specifications is incorrect, an error message is displayed and you are prompted for the data again.) If all parameters are correct, the data is then read from the source data set, added to the current work data set and the current work data set is renumbered.

To cancel the MERGE function, press the ENTER key when prompted for MERGE FROM data set name.

Caution: Once the merge has started, it must be allowed to complete normally or unpredictable results may occur. Series/1 source data sets are defined to consist of 80 character lines which are numbered in columns 73-80.

\$FSEDIT

Option 8 - END

The END option terminates \$FSEDIT.

Option 9 - HELP

The HELP option displays tutorial text on the use of \$FSEDIT.

Primary Commands

Primary commands are entered on line 2 of the display in the Command Input Field. All primary commands can be entered while in edit mode. In browse mode, three primary commands are recognized by \$FSEDIT: LOCATE, FIND, MENU.

Most of the primary commands can be entered in abbreviated format. Only the first character is required. Minimum free form format is indicated with each command enclosed in ().

The function of each of the primary commands is described on the following pages.

C (CHANGE) - Change Text (Edit Mode Only)

Changes text strings. The search for the 'text1' string proceeds until the string is found or the bottom of the data set is reached. If found, 'text1' is replaced with 'text2'. If the two text strings are not the same length, automatic shifting is performed by expanding or collapsing blank characters at the end of the line. If insufficient blanks exist for shifting right without shifting a non-blank character into column 72, the change is not made and the line is displayed with an error message in the line number field. (If the ALL option was selected, the change is terminated at this point.) If the 'text1' string is not found, you are notified with an error message displayed on the top line of the screen. PF5 would repeat the previous CHANGE specification.

Syntax

```
CHANGE /text1/text2/option
C      /text1/text2/option

Required: /text1/text2/
Defaults: option defaults to 'NEXT'
```

Operands Description

/text1/text2/

The delimiter (/) can be any alphanumeric character except blank. It is not part of, and cannot appear in, the character strings 'text1' and 'text2'. All three delimiters are required and all must be the same character. 'text1' and 'text2' can be any character string not containing the delimiter used.

option Defines the beginning and the extent of the search. The following are valid options:

NEXT

Locate and change the next occurrence of 'text1' to 'text2'. The search starts with the first line displayed.

FIRST

\$FSEEDIT

Locate and change the first occurrence of 'text1'
beginning the search at the first line of data set.

ALL

Locate and change all occurrences of 'text1' begin-
ning at the first line of the data set.

CL (CLEAR) - Clear Work Data Set (Edit Mode Only)

Clear the work data set.

Syntax

```
CLEAR  
CL
```

```
Required: none  
Defaults: none
```

No operands are required.

E (END) - End Primary Command Input

Return to Primary Option Menu. The END command terminates edit or browse mode and returns to the Primary Option Menu screen.

Syntax

```
END  
E
```

No operands are required.

F (FIND) - Find Text String

Find and display text strings. The search proceeds until the text string is found or until the end of the data set is reached. If the string is found, automatic scrolling takes place to display the line containing the text string to be displayed at the top of the data area of the display. If the string is not found, you are notified. PF4 repeats the previous FIND specification.

Syntax

```
FIND /text/ option
F      /text/ option

Required: /text/
Defaults: option defaults to 'NEXT'
```

Operands Description

/text/ The delimiter (/) can be any alphanumeric character except blank which does not appear within the text string. Both delimiters are required and must be the same character.

option Defines the beginning of the search. The valid options are:

NEXT

The search starts with the first line of the current display.

FIRST

The search starts at the first line of the data set.

L (LOCATE) - Locate Line Number

Locate and display the requested line number. The data set is searched for the requested line. If found, automatic scrolling takes place and the requested line is displayed at the top of the display. If the requested line number does not exist, an error message is displayed.

Syntax

```
LOCATE line-number
L      line-number

Required:  line-number
Defaults:  none
```

Operands Description

line-number

The number of the line to be located and displayed.

M (MENU) - Return to Primary Option Menu

Return to Primary Option Menu. The MENU command terminates edit or browse mode and returns to the Primary Option Menu screen.

```
MENU
M
```

No operands are required.

\$FSEDIT

R (RENUM) - Renumber Data Set (Edit Mode Only)

Assign new line numbers to each line of the data set.

Syntax

RENUM first increment
R first increment

Required: none

Defaults: first and increment default to 10 or to
the values last used.

Operands Description

first The number to be assigned to the first line of the
data set.

increment The increment to be used in generating line
numbers.

Note: If the number of lines in the data set is so large that
the maximum line number of 32767 is exceeded, the 'first' and
'increment' values are automatically reduced until the data
set can be properly renumbered.

RESET - Reset Line Commands (Edit Mode Only)

The RESET command can be used to reset erroneous or unwanted line commands, to reset line numbers to normal after they were replaced with ERR messages and to terminate the display of the MASK line.

Syntax

RESET

Required: none

No operands are required.

Edit Line Commands

The edit line commands are used to delete, insert, duplicate, or rearrange a single line or a group of lines. They are valid only in edit mode.

Single character line commands operate on single lines of data and are specified by entering the line command in the first position of the line preceding the line number.

Double character line commands operate on blocks of data and are specified by entering the block command in the first two positions on the first and last line of the block of data.

A (After) and B (Before)

Defines the destination for a copy or move operation. The 'A' and 'B' line commands are entered to define the destination of a copy or move operation. The 'A' defines the destination as being after the line and the 'B' defines the destination as being before the line. Thus it is possible to move or copy anywhere in the data set, including before the first line or after the last line.

C (Copy Line) and CC (Copy Block)

Duplicate lines of data within the data set. The procedure for duplicating lines is the same as for moving except that the 'C' or 'CC' replaces the 'M' or 'MM'. The copy operation leaves the original data intact and inserts a duplicate copy of the data at the destination specified. The entire data set is renumbered at the completion of the copy operation.

Example

Copy Block

```
EDIT --- EDITWORK, EDX002  2( 24)--- COLUMNS 001 072
COMMAND INPUT ==>          SCROLL ==>HALF
***** ***** TOP OF DATA *****
CC00010 LINE 1
CC00020 LINE 2
B**** ***** BOTTOM OF DATA *****
```

Screen image after Block Copy

```
EDIT --- EDITWORK, EDX002  4( 24)-BLOCK- DATA RENUMBERED
COMMAND INPUT ==>          SCROLL ==>HALF
***** ***** TOP OF DATA *****
    00010 LINE 1
    00020 LINE 2
    00030 LINE 1
    00040 LINE 2
***** ***** BOTTOM OF DATA *****
```


\$FSEEDIT

COLS - Display Columns

The COLS command are used to display a line showing column numbers. To display the column numbers, type COLS starting in the left margin of the line where the display is desired.

D (Delete Line) and DD (Delete Block)

Delete a line, or a block, of data. A 'D' on a line causes the line to be deleted when the ENTER key is pressed. More than one line can be deleted by entering a 'D' on each line. The 'DD' line commands are used to delete a block of data. The 'DD' is entered on the first and last line of the block of code to be deleted. The first line of the block does not have to be on the same display page as the end of the block (scrolling can take place between defining the two 'DD' lines). The block of data is deleted when the ENTER key is pressed the first time after both 'DD' lines have been specified.

Example

Delete Block of Lines

```
EDIT --- EDITWORK, EDX002      7(  24)--- COLUMNS 001 072
COMMAND INPUT ==>                SCROLL ==>HALF
***** ***** TOP OF DATA *****
  00010 LINE 1
DD00020 LINE 2
  00030 LINE 3
DD00040 LINE 4
  00050 LINE 5
  00060 LINE 6
  00070 LINE 7
***** ***** BOTTOM OF DATA *****
```

Screen Image after Block Delete

```

EDIT --- EDITWORK, EDX002 7( 24)-BLOCK-DATA RENUMBERED
COMMAND INPUT ==>                                SCROLL ==>HALF
***** ***** TOP OF DATA *****
00050 LINE 5
00060 LINE 6
00070 LINE 7
***** ***** BOTTOM OF DATA *****

```

I (Insert) - Insert New Line

Causes a new line to be inserted after this line. Any information typed on the inserted line is assigned a line number and becomes part of your data when the ENTER key is pressed. If the line number assigned to the newly inserted line is equal to, or greater than, the line number of the next sequential line, all data to the end of the data set is automatically renumbered. If no information is entered, the inserted line is automatically deleted the next time the ENTER key is pressed. If information is entered on the inserted line and the cursor is still on the inserted line when the ENTER key is pressed, another new line is automatically inserted. This allows line after line to be generated in a continuous insert mode. The cursor is set to the first position where data appears in the following line.

The inserted line duplicates the current value of the edit mask line. The initial value of the mask line is 72 blanks. It can be changed at any time as noted in the description of the MASK command.

Note: The I line command can be entered on the TOP OF DATA message line to insert a line ahead of what is currently the first line. It is typed in the first position of the TOP of DATA line.

\$FSEDIT

Example

I Line Command

```
EDIT --- EDITWORK, EDX002    2(  24)--- COLUMNS 001 072
COMMAND INPUT ==>          SCROLL ==>HALF
I**** **** TOP OF DATA ****
  00010 LINE 1
  00020 LINE 2
**** **** BOTTOM OF DATA ****
```

Screen Image After I Line Command

```
EDIT --- EDITWORK, EDX002    2(  24)--- COLUMNS 001 072
COMMAND INPUT ==>          SCROLL ==>HALF
**** **** TOP OF DATA ****
.....
  00010 LINE 1
  00020 LINE 2
**** **** BOTTOM OF DATA ****
```

II (Insert Block) - Insert Block of Lines

Insert a block of new data. The line with the II command is displayed at the top of the display with twenty-one inserted lines following it. Data can be entered on all twenty-one lines before entering it with the ENTER key. The new data is then saved as with the 'I' command. If all inserted lines have data entered on them and the cursor is left on the last line of the display when the ENTER key is pressed, another twenty-one lines are generated. If data are not entered on one or more of the lines, the unchanged lines are deleted and the insert mode is terminated.

Notes:

- The II command can be entered on the TOP OF DATA message line to insert data in front of what is now the first line. It is typed over the first two asterisks of the TOP OF DATA line.
- The II command is different from the rest of the double character line commands in that it is entered on only one line and generates a block of new data instead of operating on a block of data.
- The inserted lines duplicate the current value of the edit line mask.

Example

Block Insert Line Command

```

EDIT --- EDITWORK, EDX002      2(  24)--- COLUMNS 001 072
COMMAND INPUT ==>                SCROLL ==>HALF
***** ***** TOP OF DATA *****
      00010 LINE 1
      II00020 LINE 2
***** ***** BOTTOM OF DATA *****

```

\$FSEDIT

Screen Image After Block Insert Command

```
EDIT --- EDITWORK, EDX002      2(  24)--- COLUMNS 001 072
COMMAND INPUT ==>                SCROLL ==>HALF
00020 LINE 2
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

MASK - Display Insert Mask

The MASK line command is used to display the insert mask which is used to prefill inserted lines. The insert mask is 72 bytes long and is initialized to all blanks at the beginning of the session. Any data filled into it remains in effect for the remainder of the current session unless changed by you. The insert mask can be changed any time it is displayed by overtyping it with the desired information.

Note: To display the insert mask requires that all four characters of the MASK line command be entered by overtyping the first four characters of the line number.

M (Move Line) and MM (Move Block)

Move a line, or block of lines, from one location to another. When a 'M' line command is entered, a single line is moved to the location specified by an 'A' or 'B' line command. The 'MM' line commands cause the block of data defined by the two 'MM' line commands to be moved to the location specified by an 'A' or 'B' line command. The moved lines are removed from their original location and the entire data set renumbered after the move. The move occurs when the ENTER key is pressed the first time after both the lines to be moved and the destination are defined. The destination does not have to be on the same page of the display as the move line command(s) and the two 'MM' line commands can also be separated.

Example

Move Block of Lines

```

EDIT --- EDITWORK, EDX002      7(  24)--- COLUMNS 001 072
COMMAND INPUT ==>                SCROLL ==>HALF
A**** ***** TOP OF DATA *****
    00010 LINE 1
    00020 LINE 2
    00030 LINE 3
MM00040 LINE 4
    00050 LINE 5
MM00060 LINE 6
    00070 LINE 7
**** ***** BOTTOM OF DATA *****

```

\$FSEDIT

Screen Image After Block Move

```
EDIT --- EDITWORK, EDX002      7(24)-BLOCK-DATA RENUMBERED
COMMAND INPUT ==>                SCROLL ==>HALF
***** ***** TOP OF DATA *****
 00010 LINE 4
 00020 LINE 5
 00030 LINE 6
 00040 LINE 1
 00050 LINE 2
 00060 LINE 3
 00070 LINE 7
***** ***** BOTTOM OF DATA *****
```

\$IAMUT1 - BUILD AND MAINTAIN INDEXED DATA SET

\$IAMUT1 helps you manage your indexed data sets. The \$IAMUT1 utility is shipped with an input/output buffer of 512 decimal bytes. This allows you to define an indexed data set with a maximum block size of 512 bytes, and to load, unload, and reorganize indexed data sets with a maximum record size of 512 bytes. If you want to change the maximum record size, refer to the Program Directory for use with Version 1, Modification level 1 of program 5719-AM3, section J, Programming Considerations

\$IAMUT1 can be invoked using the \$L command, \$JOBUTIL, or the Session Manager.

\$IAMUT1 Commands

The commands available under \$IAMUT1 are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

ENTER COMMAND (?): ?

CR - INVOKE \$DISKUT1
EC - SET/RESET ECHO MODE
EN - END THE PROGRAM

SE - SET DEFINE PARAMETERS
DF _ DEFINE AN INDEXED FILE
DI _ DISPLAY CURRENT SE PARAMETERS
RE _ RESET CURRENT VALUES FOR DEFINE

LO - LOAD INDEXED FILE FROM SEQUENTIAL FILE
RO - REORGANIZE INDEXED FILE
UN - UNLOAD INDEXED FILE TO SEQUENTIAL FILE

ENTER COMMAND (?):

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command for of your choice (for example, EC).

\$IAMUT1

CR - Create Data Set

CR allocates space for your data set in a volume by internally invoking the \$DISKUT1 utility. The SE command should have been used to determine the number of records to allocate. When CR is entered on a terminal, the \$DISKUT1 utility is loaded. You can then use the AL command of \$DISKUT1 to allocate a data set; any other \$DISKUT1 function can also be performed. Communication to the \$DISKUT1 utility continues until the END command (EN) is entered, at which time communication to \$IAMUT1 is restored. For information on the \$DISKUT1 utility, refer to "\$DISKUT1 - Allocate/Delete; List Directory Data" on page 135.

Note: Echo mode is not active during use of \$DISKUT1.

The following example shows a use of the CR command:

```
ENTER COMMAND (?): CR
$DISKUT1 ACTIVE
USING VOLUME EDX002

COMMAND (?): AL SAMPLE1
HOW MANY RECORDS? 72
DEFAULT TYPE = DATA - OK? Y
SAMPLE1 CREATED

ENTER COMMAND (?):
```

DF - Define Indexed Data Set

DF command defines an indexed data set using an existing data set and the information you specify. When DF is entered, you are prompted for the immediate write back option and the names of the data set and volume to be formatted. Size calculations are made and the data set is formatted. The size calculation information is returned to you at your terminal on completion of the DF command. Before entering DF, you must use the SE command to set up parameters that determine the size and format of the indexed data set. The data set must have been allocated previously (the CR command can be used to allocate it).

DF allows you to select the immediate write back option. With this option, each modification to the indexed data set records that is the result of your request to update a data record is written to the data set immediately, thus contributing to the integrity of the data set. If you enter N to the immediate write back prompt, modifications are held in the main storage buffers for a period of time before being written back to the indexed data set. In most cases, not using the immediate write back option often results in fewer I/O operations and in better performance.

The following example shows a use of the DF command:

```
ENTER COMMAND (?): DF
DO YOU WANT IMMEDIATE WRITE-BACK? N
ENTER DATASET (NAME,VOLUME): SAMPLE1,EDX002
```

After entering the above information, the following is displayed:

\$IAMUT1

TOTAL LOGICAL RECORDS/DATA BLOCK:	3
FULL RECORDS/DATA BLOCK:	2
INITIAL ALLOCATED DATA BLOCKS:	1
INDEX ENTRY SIZE:	32
TOTAL ENTRIES/INDEX BLOCK:	7
FREE ENTRIES/PIXB:	1
RESERVE ENTRIES/PIXB(BLOCKS):	0
FULL ENTRIES/PIXB:	6
RESERVE ENTRIES/SIXB:	0
FULL ENTRIES/SIXB:	7
DELETE THRESHOLD ENTRIES:	7
FREE POOL SIZE IN BLOCKS:	0
# OF INDEX BLOCKS AT LEVEL 1:	9
# OF INDEX BLOCKS AT LEVEL 2:	2
# OF INDEX BLOCKS AT LEVEL 3:	1
DATA SET SIZE IN EDX RECORDS:	72
INDEXED ACCESS METHOD RETURN CODE:	-1
SYSTEM RETURN CODE:	-1
ENTER COMMAND (?):	

DI - Display Parameter Values

DI displays the current parameter values entered via the SE command. The parameter values can be used to format a data set via the DF command or they can be modified by reusing the SE command.

The following example shows a use of the DI command.

```
ENTER COMMAND (?): DI
CURRENT VALUES FOR SE COMMAND ARE:
BASEREC          100
BLKSIZE          256
RECSIZE          80
KEYSIZE          28
KEYPOS           1
FREEREC          1
FREEBLK          10
RSVBLK           NULL
RSVIX            0
FPOOL            NULL
DELTHR           NULL
ENTER COMMAND (?):
```

\$IAMUT1

EC - Control Echo Mode

EC allows you to enter or leave echo mode. When in echo mode, all \$IAMUT1 input and output is logged on the \$SYSPRTR device. This allows you to save information about the data sets you maintain using \$IAMUT1. When in echo mode, all input and output is logged until either the current utility session is ended or echo mode is reset by use of the EC command. Echo mode is off when \$IAMUT1 is loaded.

Note: Input and output from \$DISKUT1 (CR command) is not logged.

The following examples show the commands to set and reset echo mode:

```
ENTER COMMAND (?): EC
DO YOU WANT ECHO MODE? (Y/N) Y   (Set echo mode)
ENTER COMMAND (?): EC
DO YOU WANT ECHO MODE? (Y/N) N   (Reset echo mode)
ENTER COMMAND (?):
```

LO - Load Indexed Data Set

LO loads an indexed data set from a sequential data set. The sequential data set must contain unblocked records. They can span two or more 256-byte records. The records in the sequential data set must be in ascending order by the data that represents the key field. If a record with a duplicate or out of sequence key is found, you are given the option to either omit the record and continue loading, or to end loading. The indexed data set must have been allocated and defined by use of the CR, SE and DF commands before using the LO command.

The record lengths of the two data sets do not have to be the same. When the indexed data set is opened, the record length is displayed on the terminal. At this point, you can specify the record length of the sequential data set if it is different than that of the indexed data set. If the indexed data set records are longer than the sequential data set records, the loaded records are left justified and filled with binary zeros. If the indexed data set records are shorter than the sequential data set records, the following message appears on the terminal:

```
INPUT REC GT OUTPUT REC. TRUNCATION WILL OCCUR.  
OK TO PROCEED?
```

Reply 'Y' to proceed (records will be truncated). Reply 'N' to terminate the load function. If the end of the input sequential data set is reached, you can continue loading from another sequential data set. You are asked if there is more data to load. If you reply yes (Y), you are prompted for the data set and volume name of the new input sequential data set to use. The load operation continues, putting the first record of the new input sequential data set in the next available record slot of the indexed data set.

Note: The record lengths of subsequent input data sets are assumed to be the same as the initial input data set.

If the end of input data set is reached and you choose not to name another input data set, the load operation is complete.

The following example shows use of the LO command:

SIAMUT1

```
ENTER COMMAND (?): LO
LOAD ACTIVE
ENTER INPUT DATASET (NAME,VOLUME): SEQDATA,EDX002
ENTER OUTPUT DATASET (NAME,VOLUME): SAMPLE1,EDX002
INPUT RECORD ASSUMED TO BE 80 BYTES. OK?: Y
LOAD IN PROCESS
END OF INPUT D/S
ANY MORE DATA TO BE LOADED?: N
    100 RECORDS LOADED
LOAD SUCCESSFUL
ENTER COMMAND (?):
```

RO - Reorganize Indexed Data Set

RO unloads an indexed data set filled by insert activity into an empty indexed data set and reorganizes the records to allow additional inserts. The empty indexed data set must have been allocated and defined by use of the SE, CR, and DF commands before using the RO command.

The record lengths of the two data sets need not be the same. Unloaded records are truncated or filled with binary zeros if record lengths differ (see LO command). The key fields and key positions of the two data sets must be the same; however, the other data set specifications (SE parameters) may differ.

The following example shows use of the RO command:

```
ENTER COMMAND (?): RO
REORG ACTIVE
ENTER INPUT DATASET (NAME,VOLUME): SAMPLE,EDX002
ENTER OUTPUT DATASET (NAME,VOLUME): SAMPLE1,EDX002
REORG IN PROCESS
END OF INPUT D/S
    100 RECORDS LOADED
REORG SUCCESSFUL
ENTER COMMAND (?):
```

RE - Reset Parameters

RE resets the parameters set up by the SE command to their default values.

The following example shows a use of the RE command:

```
ENTER COMMAND (?): RE
ENTER COMMAND (?):
```


\$IAMUT1

SE - Set Parameters

SE prompts you for parameters that determine the structure and size of the indexed data set. The parameter values entered are saved by \$IAMUT1. This allows you to reuse the SE command to change one or more parameters without having to reenter all of them. The current values can be displayed by the DI command.

Note: The values are retained only as long as \$IAMUT1 remains loaded.

Size calculations are performed using the parameter values you have specified. The results are returned to you at your terminal. The SE command can be followed by the CR command to create a data set of the size specified on the SE command. After the data set has been allocated, or if a data set of the correct size already exists, the DF command can be used to format it.

The following list shows the default values for parameters on the SE command (all values are decimal):

BASEREC	0
BLKSIZE	0
RECSIZE	0
KEYSIZE	0
KEYPOS	1
FREEREC	0
FREEBLK	0
RSVBLK	NULL
RSVIX	0
FPOOL	NULL
DELTHR	NULL

If the default value is acceptable, enter a null line for the parameter when prompted for it. If you wish to change the value for any parameter, enter the new value in response to the prompting message. The new value becomes the new default value for the current \$IAMUT1 session. The only parameters for which a null can be specified are RSVBLK, FPOOL, DELTHR, and RSVIX. To specify a null parameter after the original default has been modified, enter an ampersand (&) in response to the prompting message. For an explanation of the SE command parameters, refer to "Determining Data Set Size and Format" on page 247.

The following example shows a use of the SE command in establishing the size and structure of an indexed data set.

```

ENTER COMMAND (?): SE
ENTER BASEREC 100
ENTER BLKSIZE 256
ENTER RECSIZE 80
ENTER KEYSIZE 28
ENTER KEYPOS 1
ENTER FREEREC 1
ENTER FREEBLK 10
ENTER RSVBLK
ENTER RSVIX
ENTER FPOOL
ENTER DELTHR

```

After the above information has been entered, the following is displayed showing the size and structure of the defined indexed data set.

```

TOTAL LOGICAL RECORDS/DATA BLOCK:          3
FULL RECORDS/DATA BLOCK:                   2
INITIAL ALLOCATED DATA BLOCKS:            50
INDEX ENTRY SIZE:                          32
TOTAL ENTRIES/INDEX BLOCK:                 7
FREE ENTRIES/PIXB:                          1
RESERVE ENTRIES/PIXB(BLOCKS):              0
FULL ENTRIES/PIXB:                          6
RESERVE ENTRIES/SIXB:                       0
FULL ENTRIES/SIXB:                          7
DELETE THRESHOLD ENTRIES:                  7
FREE POOL SIZE IN BLOCKS:                   0
# OF INDEX BLOCKS AT LEVEL 1:               9
# OF INDEX BLOCKS AT LEVEL 2:               2
# OF INDEX BLOCKS AT LEVEL 3:               1

DATA SET SIZE IN EDX RECORDS:              72
INDEXED ACCESS METHOD RETURN CODE:          -1
SYSTEM RETURN CODE:                        -1
ENTER COMMAND (?):

```

UN - Unload Indexed Data Set

UN unloads an indexed data set to a sequential data set. The record lengths of the two data sets need not be the same. Unloaded records are truncated or padded with zeros if the records lengths of the two data sets differ. Refer to the LO command.

Records are placed in the sequential data set in ascending key sequence as indicated by the indexed data set. Unloaded records are not blocked. They can span two or more 256 byte records. If the indexed data set contains more records than are allocated in the sequential data set, you are given the option to continue unloading to another sequential data set. If you choose to continue unloading, you are prompted for the name of the data set and volume to use to continue the unload operation. The unload operation continues, putting the records read from the indexed data set into the new sequential data set.

Note: The record length of subsequent output sequential data sets is assumed to be the same as the initial output sequential data set.

If the end of the output data set is reached and you choose not to continue, the unload operation ends.

Caution: Do not specify the same data set for input and output.

The following is an example of the commands and responses of an UN command:

```
ENTER COMMAND (?): UN
UNLOAD ACTIVE
ENTER INPUT DATASET (NAME,VOLUME): SAMPLE1,EDX002
ENTER OUTPUT DATASET (NAME,VOLUME): SAMPLE2,EDX002
OUTPUT RECORD ASSUMED TO BE 80 BYTES. OK?: N
ENTER RECORD SIZE: 256
UNLOAD IN PROCESS
END OF INPUT D/S
    100 RECORDS UNLOADED
UNLOAD SUCCESSFUL
ENTER COMMAND (?):
```

Building an Indexed Data Set

The SE and DF commands allow you to specify the size and format of your indexed data set and to do the actual data set formatting. Use the SE command to enter those values that determine the size of the indexed data set. Use the DF command to actually format the data set using the values previously specified on the SE command.

If the data set is too small to support the specified format, the amount of space required is returned to you. Knowing the available space and using the SE command, you can vary the SE parameters to design a data set of proper size.

Determining Data Set Size and Format

The data set design is determined by these SE command parameters:

BASEREC	The estimated number of records to be loaded into the data set in ascending key sequence. These records can be loaded by \$IAMUT1 or by a PUT request after either a LOAD or PROCESS request.
----------------	---

BASEREC should never be zero.

BLKSIZE	The length, in bytes, of blocks in the data set. It must be a multiple of 256.
----------------	--

RECSIZE	The length, in bytes, of records in the data set. Record length must not exceed block length minus 16.
----------------	--

KEYSIZE	The length (1 to 254 bytes) of the key to be used for this data set.
----------------	--

KEYPOS	The position, in bytes, of the key within the record. The first byte of the record is position 1.
---------------	---

FREEREC	The number of free records to be reserved in each block. It should be less than the number of records per block (block size minus 16, divided by record size). It can be zero.
----------------	--

FREEBLK	The percentage (0-99) of each cluster to reserve for free blocks. The percentage calculation result is rounded up so that at least one free block results. The calculation is adjusted to ensure that there is at least one allocated block in the cluster; that is,
----------------	--

there cannot be 100% free blocks.

RSVBLK The percentage of the entries in each lowest level index block to reserve for cluster expansion. These reserved entries are used to point to new data blocks as they are taken from the free pool to expand the cluster. The result of the calculation is rounded up so that any non-zero specification indicates at least one reserved index entry. The calculation is adjusted to ensure that there is at least one allocated block in the cluster. Enter a null character (*) for this prompt if you do not want initial reserved blocks and do not want the indexed access method to create reserved blocks as records are deleted and blocks become empty. Specify a value of zero for this prompt if you do not want initial reserved blocks but you do want the indexed access method to create reserved blocks as records are deleted and blocks become empty (See the DELTHR prompt). Note that the sum of the FREEBLK and RSVBLK prompts should be less than 100.

RSVIX The percentage (0-99) of the entries in each second level index block to reserve for use in case of cluster splits. A cluster split is required when a cluster expands to its maximum potential size (as defined by the RSVBLK prompt) and another data set is inserted into the cluster. Each cluster split uses one reserved entry of the second level index block. The result of this calculation is rounded up so that any non-zero specification indicates at least one reserved index entry. The calculation is adjusted so that there is at least one unreserved entry in each second level index block. This value defaults to zero.

FPOOL The percentage (0-100) of the maximum possible free pool to allocate. The RSVBLK and RSVIX prompts result in a data set structure capable of drawing on the free pool for expansion. If insertion activity is evenly distributed throughout the data set, every reserve entry of every index block can be used. The number of blocks drawn from the free pool to support this highly unlikely condition is the maximum free pool size needed for the data set. In more realistic cases, insertion activity is not evenly distributed throughout the data set, so fewer free blocks are needed. The percentage specified here represents the evenness of the distribution of inserted records. Specify a large number (90, for example) if you expect insertions to be evenly distributed. Specify a small number (20, for example) if insertions are

anticipated to be concentrated in specific key ranges. If null is specified for this prompt, a free pool is not created for this indexed data set. If zero is specified, an empty free pool is created. Blocks can then be added to the free pool as records are deleted and blocks become empty (see the DELTHR prompt explanation). If you do not specify a null for this prompt, the RSVBLK must not be null and/or the RSVIX must be non-zero or an error is returned. Conversely, if the RSVBLK and/or RSVIX is non-zero, FPOOL must not be null or an error is returned.

DELTHR The percentage (0-99) of blocks to retain in the cluster as records are deleted and blocks made available. This is known as the delete threshold. When a block becomes empty, it is first determined if the block should be given up to the free pool by checking the response to this prompt. If the block is not given up to the free pool, it is retained in the cluster, either as a free block or as an active empty block. The result of this calculation is rounded up so that any non-zero specification indicates at least one block. The calculation is adjusted to ensure that the cluster always contains at least one block. If specified as null, this value defaults to the number of allocated blocks in the cluster plus one half of the value calculated by the FREEBLK prompt. Specify null unless data set usage indicates that tuning is required.

The define (DF) command sets the size of the data set. Therefore, the BASEREC, FREEREC, FREEBLK, RSVBLK, RSVIX, and FPOOL parameters should be large enough to accommodate the maximum number of records planned for the data set.

To calculate the size of the data set for a given combination of parameters, use the SE command.

\$IMAGE

\$IMAGE - DEFINE 4978/4979 FORMATTED SCREEN IMAGE

\$IMAGE defines formatted screen images for the 4978/4979 display terminals, or for any terminal whose support includes the static screen functions. The images (formatted screens), which are defined in terms of protected and non-protected alphanumeric fields, can be saved in disk or diskette data sets, to be retrieved later by application programs for display, or by this utility for modification.

You must allocate the data set where the image will be stored before using \$IMAGE. Most logical screens require a data set of two records. There are two modes of interaction with the \$IMAGE program: command mode and edit mode. For retrieval information, refer to screen formatting in the System Guide.

You can find an example using \$IMAGE to format a static screen in the System Guide. For an aid in laying out the format of the screen to be defined, refer to the IBM 3270 Information Display System Layout Sheet, GX27-2951.

\$IMAGE Commands

\$IMAGE is in command mode when loaded. When \$IMAGE is in command mode, you specify the function to be performed by entering an alphabetic code followed by a parameter list. You are prompted for each item that is not specified in advance and does not have a default value.

The commands available under \$IMAGE are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

```
COMMAND(?): ?  
  
DIMS -- DEFINE IMAGE DIMENSIONS  
HTAB -- DEFINE HORIZONTAL TAB SETTINGS  
VTAB -- DEFINE VERTICAL TAB SETTINGS  
NULL -- DEFINE NULL REPRESENTATION  
EDIT -- ENTER EDIT MODE  
KEYS -- LIST PROGRAM FUNCTION KEYS  
SAVE -- SAVE IMAGE ON DISK  
END -- END PROGRAM  
  
COMMAND(?):
```

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, DIMS).

DIMS - Define Image Dimensions

The DIMS command is followed by two values, giving the number of lines and the line size, for example:

```
DIMS 10 20 (10 20-character lines)  
DIMS 24 80 (full screen image)
```

EDIT - Enter Edit Mode

The EDIT command can be followed by a data set name in the form:

```
dataset,volume
```

Here the volume, if unspecified, defaults to the IPL volume. If a data set name is not given, then the program displays the currently defined image. Some examples are:

\$IMAGE

```
EDIT
EDIT  IMAGE1
EDIT  IMAGEB,EDX002
```

END - End the Program

END (or EN) ends the utility program. If the screen image has been modified since the last SAVE operation, then the question:

```
SAVE FINAL IMAGE?
```

appears. The image is saved or not, depending on your response (YES/NO), and the command prompting message is issued again.

HTAB - Define Horizontal Tab Settings

HTAB defines tab settings for the duration of edit mode, for example:

```
HTAB 10 15 20 40 45 50 73
```

The default settings are 10, 20, 30, 40, 50, 60 and 70. If a tab value exceeds the line size or is not in ascending order, then the cursor moves to the next line when that setting is encountered. When the horizontal tab key (PF1 when in EDIT mode) is pressed, the cursor moves to the next horizontal tab position.

KEYS - List Program Function Keys

KEYS lists on the terminal the meaning of the PF keys while in the edit mode.

```
COMMAND(?): KEYS

PF1 -- DEFINE PROTECTED FIELDS
PF2 -- DEFINE DATA FIELDS
PF3 -- ENTER COMMAND MODE

COMMAND(?):
```

PF1 and PF2 function as the horizontal and vertical tab keys (respectively) when you are defining protected or data fields in EDIT mode.

NULL - Define Null Representation

NULL defines a character that is interpreted as the null character during editing of the image. Null characters define the non-protected positions on the screen image to be used for data entry. Some examples are:

```
NULL .
NULL /
NULL
CHARACTER:
```

In the last example, no character was indicated causing the prompt message to be issued. If again no character is entered, then the null character itself is used.

Note: If the NULL command is not defined prior to an editing session, the null character defaults to a period (.). Therefore, all periods defined as protected in subsequently edited screens assume an unprotected status.

§IMAGE

SAVE - Save Image on Disk

The screen image, as currently defined, is saved in the last data set edited, or in a new data set if one is specified following SAVE. The formatting information and text that define the image are stored in the specified data set in a special packed format to conserve space. For packed format information, refer to the System Guide. Some examples are:

```
SAVE
SAVE IMAGE2
SAVE IMAGE2,EDX003
```

When the image has been saved, the following message appears, with the number of 256 byte records indicated in parentheses.

```
SAVED n(RECORDS)
```

VTAB - Define Vertical Tab Settings

VTAB defines vertical tab settings to conveniently edit the screen image by columns rather than rows. The default vertical tabs range from 1 to 24 in 1-line increments. These can be redefined as in the following example:

```
VTAB 5 10 20 24
```

When the vertical tab key (PF2 when in edit mode) is pressed, the cursor moves to the next indicated line at the last encountered horizontal tab setting. When the last vertical tab setting is passed, the cursor moves to line 0 at the next horizontal tab setting.

Edit Mode

When you enter edit mode, the current image is displayed within a rectangular frame whose upper lefthand corner is at line 0, indent 0. The frame and all screen positions outside it are protected in the display buffer; this limits the cursor to positions within the frame when the field-advance key is pressed. If the image dimensions do not allow display of the entire frame, then its sides are omitted according to the following priority: top before bottom, left before right.

Null characters, dimension fields, and tab settings should be defined before entering edit mode. If you are modifying an existing screen image, the null character must be redefined each time \$IMAGE is invoked.

Once the image is displayed, you can invoke the edit phases by means of the PF keys.

PF1 This key causes the protected fields of the image to be displayed as non-protected, so you can redefine them directly on the screen. The non-protected (data) fields of the image are indicated with the null representation character, and you use that character to redefine those fields if desired. Once this edit phase has been entered, PF1 acts as the horizontal tab key and PF2 as the vertical. When the ENTER key is pressed, the newly defined image is displayed, with protected fields in their proper mode.

PF2 This key prepares the \$IMAGE program for modification of the data (non-protected) fields of the image. The cursor is displayed at position (0,0) of the image, and you can use the field-advance keys to move to each data field in turn, or the tab keys (PF1 and PF2) can be used when appropriate. When the ENTER key is pressed, the new data values are saved, but not yet written to disk.

PF3 This key is used to return from edit mode to command mode.

Note: The ENTER key must have been used for PF3 to function correctly.

\$INITDSK

\$INITDSK - INITIALIZE OR VERIFY VOLUME

\$INITDSK initializes and/or read verifies a Series/1 direct access storage device volume for use with the Event Driven Executive.

\$INITDSK performs the following functions:

- Initialization (I)
 - Initializes a library directory for the Event Driven Executive
 - Writes IPL text on a disk or diskette, if desired. The IPL device address for 4962 disk, 4963 disk, 4964 diskette, and 4966 diskette magazine units are hexadecimal 03, 48, 02, and 22, respectively. An initialized diskette can be used to IPL from either a 4964 or 4966 diskette device.
 - Writes a volume label on a diskette.

Note: A label is not required on a disk since it is not a removable device.

- Verification (V) verifies the readability of:
 - A group of records within a disk or diskette volume
 - A disk or diskette volume
 - All disk volumes at a specified address
- Writes (W) only IPL text on a primary volume for which \$EDXNUC has been allocated.

\$INITDSK Commands

The commands available under \$INITDSK are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):

```
COMMAND (?): ?  
  
E - END PROGRAM  
I - INITIALIZE DISK(ETTE)  
V - VERIFY DISK(ETTE) AREA  
W - WRITE IPL TEXT ONLY  
  
COMMAND (?):
```

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, I).

Initialization

Directory Creation

A directory can be created on each volume with \$INITDSK. The minimum directory size is 2 records. The maximum sizes are 120 records on a 4962, 4963, 4964, or 4966 and 60 records on the fixed head volume of a 4962 or 4963. The maximum volume size, including directory, is 32,767 records. The directory size determines the maximum number of programs and data sets that can be stored. A directory of n records can catalog a maximum of $8n-2$ data sets.

Diskette Initialization

The volume label on a diskette conforms to the standard for an EBCDIC Basic Exchange format. One EBCDIC Header Label (HDR1) is written which describes the entire diskette as an allocated data set. An entire diskette is considered as an Event Driven Executive volume. A single-sided diskette is initialized to contain up to a 13-record directory and a 949-record data area. A double-sided diskette is initialized to contain up to a 26 record directory and an 1898 record data area. A diskette must have been previously initialized to 128 bytes per sector by using the \$DASDI utility. On a 4966 diskette magazine unit, you

\$INITDSK

can initialize only on slot number 1.

Disk Initialization

Each disk volume (primary and secondary) must be initialized by using \$INITDSK.

Caution: If you initialize and create a directory on disk or diskette, any data previously stored on the disk or diskette will no longer be accessible.

Example

Initializing and Writing IPL Text on Diskette

```

INITDSK      13P,13.44.14, LP= 5000

COMMAND (?): I

LIBRARY INITIALIZATION

    1 = ENTER VOLUME LABEL
    2 = ENTER DEVICE ADDRESS
SELECT OPTION: 2

ENTER DEVICE ADDRESS IN HEX: 2

WRITE VOLUME LABEL? Y
ENTER DESIRED VOLUME LABEL (1-6 CHARACTERS) EDX001
ENTER OWNER ID (1-14) CHARACTERS: DEPT.L78 09/77
LABEL WRITTEN

CREATE A DIRECTORY? Y
HOW MANY RECORDS IN DIRECTORY? (2 - 120): 13
    MAXIMUM NO. OF MEMBERS = 102, OK? Y
DO YOU WISH TO RESERVE SPACE FOR A NUCLEUS? Y
ENTER MAXIMUM SIZE IN K-BYTES (16-64): 32
DIRECTORY INITIALIZED
WRITE IPL TEXT? Y
IPL TEXT WRITTEN

COMMAND (?): EN

INITDSK ENDED AT 13:54:00

```


\$INITDSK

Verification

\$INITDSK can be used to perform a readability check of an entire disk device, or an entire disk or diskette volume, or any portion thereof, to determine if any defective records exist for which alternates should be assigned. Only record read operations are performed. Therefore, a verify operation can be performed with no risk of destroying existing data.

Examples

Verifying Portion of a Volume

```
COMMAND(?): V
DISK OR DISKETTE VERIFICATION

1 = ENTER VOLUME LABEL
2 = ENTER DEVICE ADDRESS
SELECT OPTION: 1

ENTER VOLUME LABEL: EDX002

EDX002 AT 0003 IS A PRIMARY DISK
CHECK THE ENTIRE VOLUME? NO
FIRST RECORD = 300
NO. OF RECORDS = 1200
1200 RECORDS CHECKED

COMMAND (?):
```

Verifying All Volumes on a Disk

```

COMMAND(?): V
DISK OR DISKETTE VERIFICATION

1 = ENTER VOLUME LABEL
2 = ENTER DEVICE ADDRESS
SELECT OPTION: 2

ENTER DEVICE ADDRESS IN HEX: 3

CHECK THE ENTIRE VOLUME? Y
ASSIGNED ALTERNATE SECTOR AT RECORD    121
ASSIGNED ALTERNATE SECTOR AT RECORD    122
ASSIGNED ALTERNATE SECTOR AT RECORD    151
EDX002 32000 RECORDS CHECKED
EDX003 30000 RECORDS CHECKED
EDX004 12000 RECORDS CHECKED

COMMAND (?):
    
```

Verifying Diskette Volume (No Errors Found)

```

COMMAND (?): V
DISK OR DISKETTE VERIFICATION

1 = ENTER VOLUME LABEL
2 = ENTER DEVICE ADDRESS
SELECT OPTION: 2

ENTER DEVICE ADDRESS IN HEX: 2

CHECK THE ENTIRE VOLUME? YES
962 RECORDS CHECKED

COMMAND (?):
    
```

\$INITDSK

Write IPL Text Where \$EDXNUC Has Been Preallocated

COMMAND (?): W

WRITE IPL TEXT

1=ENTER VOLUME LABEL

2=ENTER DEVICE ADDRESS

SELECT OPTION: 1

ENTER VOLUME LABEL: EDX002

EDX002 AT 0002 IS A PRIMARY DISKETTE

WRITE IPL TEXT? Y

IPL TEXT WRITTEN

COMMAND (?):

Write IPL Text on a Volume That Does Not Contain \$EDXNUC

COMMAND (?): W

WRITE IPL TEXT

1=ENTER VOLUME LABEL

2=ENTER DEVICE ADDRESS

SELECT OPTION: 1

ENTER VOLUME LABEL: EDX002

EDX002 AT 0002 IS A PRIMARY DISKETTE

IPL WRITE ABORTED. NO NUCLEUS FOUND

COMMAND (?):

\$IOTEST - TEST SENSOR I/O; LIST CONFIGURATION

\$IOTEST determines the complete I/O configuration of a Series/1 and tests the operation of sensor based I/O features. **\$IOTEST** performs the following functions:

- Reads and writes digital (group or subgroup)
- Writes digital with selected time intervals
- External sync DI and DO
- Processes interrupts (normal, special bit and group)
- Reads and writes analog
- Lists the hardware configuration of the Series/1
- Lists the devices supported by the system
- Lists volume information

Invoking \$IOTEST

\$IOTEST can be invoked by the session manager using the Diagnostic Utilities option menu or by the **\$L** command.

\$IOTEST

\$IOTEST Commands

The commands available under \$IOTEST are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

```
$IOTEST      24P,13:26:42  LP=5000

ATTNLIST (ALTER) TO STOP LOOPING FUNCTIONS

COMMAND (?): ?

EN = END PROGRAM
DO = DIGITAL OUTPUT
PD = UP AND DOWN DO WITH TIME
XO = EXTERNAL SYNC DO
DI = DIGITAL INPUT
XI = EXTERNAL SYNC DI
PI = PROCESS INTERRUPT
SG = SPECIAL PROCESS INTERRUPT GROUP
SB = SPECIAL PROCESS INTERRUPT BIT
AI = ANALOG INPUT
AO = ANALOG OUTPUT
LD = LIST ALL HARDWARE DEVICES
LS = LIST SUPERVISOR CONFIGURATION
VI = DISPLAY VOLUME INFORMATION
WS = PUT PGM IN WAIT STATE

COMMAND (?): EN

$IOTEST ENDED AT 13:27:29
```

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, DO).

AI and DI issue a read every 10 milliseconds and only prints the value if it is different from the last reading. PI issues a wait and prints on each occurrence. External sync DI and DO perform their functions and do not return to command status until the number of words indicated are read or written. ALTER is used to terminate repetitive functions and to reactivate the program if it was put in the wait state.

Examples

DO - Write X'A5A5' to DO Address 52

```

COMMAND (?) : DO
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 52
ENTER START BIT (0-15) : 0
ENTER # OF BITS : 16
ENTER DATA (HEX) : A5A5
COMMAND (?):
    
```

PD - Pulse DO Address 53 Bit 8 on for 10 Milliseconds, Off for 50 Milliseconds 100 times

```

COMMAND (?): PD
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 53
ENTER START BIT (0-15) : 8
ENTER # OF BITS : 1
ENTER DATA1 (0-FFFF) : 1
ENTER TIME1 IN MS : 10
ENTER DATA2 (0-FFFF) : 0
ENTER TIME2 IN MS : 50
ENTER NUMBER OF TIMES TO LOOP : 100
COMMAND (?):
    
```

Note: If number of times to loop is set less than or equal to zero, the looping continues until the next ALTER command.

SIOTEST

XO - Write Digital Output using External Sync

```
COMMAND (?): XO
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 53
ENTER WORD COUNT 1-256 : 100
COMMAND (?):
```

Note: Data is written from a buffer within this program that is used for external sync DI and DO. Therefore, data can be input via DI and written via DO.

DI - Read Digital Input

```
COMMAND (?): DI
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 50
ENTER START BIT (0-15) : 0
ENTER # OF BITS : 16
VALUE = A5A5
VALUE = COFE
> ALTER
COMMAND (?):
```

XI - Read Digital Input using External Sync

```
COMMAND (?): XI
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 51
ENTER WORD COUNT : 100
COMMAND (?):
```

PI - Test Process Interrupt for the Occurrence of this Event

```
COMMAND (?): PI
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 50
ENTER BIT (0-15) : 3
PI OCCURRED
PI OCCURRED
PI OCCURRED
> ALTER
COMMAND (?):
```

SG/SB - Special Process Interrupt Group/Bit

Commands SG and SB functionally operate differently within the supervisor but they print basically the same information as normal PI with this utility program.

An easy way to test the system is to use the Customer Engineer's wrap back connectors. The wrap cable for the IDIO unit connects the first DI address on the card to the first DO address and the same for the second DI and DO. These connections include the external sync functions also. Therefore, two copies of \$IOTEST can be executed simultaneously. There are similar connectors available for the 4982 Sensor I/O Unit.

\$IOTEST

AI - Read Analog Input

```
COMMAND (?): AI
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 61
ENTER RANGE: 1=5V, 2=500MV, 3=200MV, 4=100MV,
             5=50MV, 6=20MV, 7=10MV : 7
ZERO CORRECTION ? N
ENTER MPXR POINT (0-15) : 1
AI VOLTAGE =      -629 MV, E-3
AI VOLTAGE =      -688 MV, E-3
> ALTER
COMMAND (?):
```

Analog input has a testing facility to convert diagnostic zero or voltage. This utility allows these functions if the ADC address is given instead of the multiplexor address.

```
COMMAND (?): AI
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 60
CONVERT DIAGNOSTIC ZERO ? Y
ENTER RANGE: 1=5V, 2=500MV, 3=200MV, 4=100MV,
             5=50MV, 6=20MV, 7=10MV : 1
AI VOLTAGE =          0 MV, E-0
> ALTER

COMMAND (?): AI
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 60
CONVERT DIAGNOSTIC ZERO ? N

CONVERTING DIAGNOSTIC VOLTAGE, SHOULD BE 4.5 +- 0.5
AI VOLTAGE =      4604 MV, E-0
AI VOLTAGE =      4602 MV, E-0
> ALTER
COMMAND (?):
```

```

COMMAND (?): AI
ENTER DEVICE ADDRESS ,(HEX 1-FF) : 60
CONVERT DIAGNOSTIC ZERO ? N

CONVERTING DIAGNOSTIC VOLTAGE, SHOULD BE 4.5 +- 0.5
AI VOLTAGE =      4604 MV, E-0
AI VOLTAGE =      4602 MV, E-0
> ALTER
COMMAND (?):

```

LD - List Devices

LD reads the actual hardware addresses, their IDs, and displays a list of the descriptions. If a device exists but is not powered on, the description for that device is displayed.

```

COMMAND (?): LD

ACTUAL SERIES/1 HARDWARE CONFIGURATION

00 = TELETYPEWRITER ADAPTER
01 = 4974 PRINTER
02 = 4964 DISKETTE UNIT
04 = 4979 DISPLAY STATION
09 = SINGLE LINE BSC
40 = TIMER FEATURE
41 = TIMER FEATURE
50 = IDIO DI/PI NON-ISOLATED
51 = IDIO DI/PI NON-ISOLATED
52 = IDIO DO WITH EXTERNAL SYNC
53 = IDIO DO WITH EXTERNAL SYNC

COMMAND (?):

```

\$IOTEST

LS - List Supervisor Configuration

LS provides a display similar to LD except that it lists the devices your supervisor is generated to support (whether or not they are in the hardware of the system currently being used).

VI - Display Volume Information

VI displays information about volumes as follows:

COMMAND (?) : VI							
VOLSER	TYPE	IODA		STATUS	VOLORG	VOLSIZE	LIBORG
NRQ021	PRI.	0002		ONLINE	0	75	27
NRP001	PRI.	0022	1	ONLINE	0	75	27
SMVOL	SEC.	0022	2	OFFLINE	0	75	27
NRP002	SEC.	0022	3	ONLINE	13	74	14
NRP003	SEC.	0022	1A	ONLINE	13	75	27
NEWPRG	SEC.	0022	3A	ONLINE	0	74	14
FORT	SEC.	0022	4A	ONLINE	13	75	27
DEV	SEC.	0022	10A	ONLINE	0	74	14
EDX002	PRI.	0003		ONLINE (IPL)	0	130	241
ASMLIB	SEC.	0003			130	16	1
SUPLIB	SEC.	0003			146	16	1
EDX003	SEC.	0003			162	141	1
EDX005	PRI.	0048		ONLINE	0	50	705
EDX006	SEC.	0048			51	50	1
EDX007	SEC.	0048			150	50	1
EDX008	SEC.	0048			200	50	1

\$JOBUTIL - JOB STREAM PROCESSOR

\$JOBUTIL is a batch job processing capability that can be invoked simultaneously with the execution of other programs. This allows you to sequentially execute a series of programs without intervention. You include the names of the programs, and other information in a collection of \$JOBUTIL commands created via \$EDIT1N or \$FSEEDIT.

A procedure can invoke another procedure, however, the called procedure cannot invoke another. A typical use of \$JOBUTIL would be to execute a procedure that causes the assembly, link editing, and formatting of your program. Refer to "Batch Job Example" on page 290.

\$JOBUTIL is the main program that calls two overlay programs: \$JP1 which opens required data sets and \$JP2 which processes all commands.

Programs that are capable of receiving parameters in the format used by \$JOBUTIL are:

\$COBOL	\$PL/I
\$EDXASM	\$PREFIND
\$FORT	\$SIASM
\$LINK	\$UPDATE

Setup Procedure

The steps required to set up and start a procedure are:

1. Using \$DISKUT1, allocate a data set on either disk or diskette which is to contain the procedure. The size of the data set depends on the number of commands. Allocate one record for each two commands.
2. Using \$EDIT1N or \$FSEEDIT, enter the \$JOBUTIL commands necessary for your procedure into an editor work data set, and 'SAVE' them in the data set allocated in Step 1.
3. To run your job, load \$JOBUTIL and specify the name of your procedure data set. Loading can be done by use of the session manager or the \$L command.

\$JOBUTIL

\$JOBUTIL Commands

The \$JOBUTIL commands are listed below. Commands must be entered in the following format:

Command - Position 1 to 8

Operands - Position 10 to 17

Comment - Position 18 to 71

For internal comments - '*' in position 1

Note: All commands without operands can have comments starting in position 10.

Command	Function
AL	Allocate data set
DE	Delete data set
DS	Identify data set
EOJ	Identify end of job
EOP	Identify end of nested procedure
EXEC	Load and execute program
JOB	Identify job
JUMP	Jump to label
LABEL	End jump
LOG	Log Job Stream Processor Commands
NOMSG	Suppress load messages
PARM	Identify program parameter
PAUSE	Await manual intervention
PROC	Identify nested procedure
PROGRAM	Identify program
REMARK	Remark to operator
*	Internal comment

AL - Allocate Data Set

AL identifies a data set to be allocated. It returns a \$DISKUT3 return code that can be used by the JUMP command.

Syntax

```
AL          name,volume size type
Required:   name,volume
            size
            type
Default:    None
```

Operands

Description

name	Defines the data set to be allocated
volume	Defines the volume on which the data set is to be allocated
size	Defines the size in blocks of the data set
type	Defines the type of data set as follows: D indicates data (default) P indicates program U indicates undefined

Example

```
AL          TEMPDS,EDX003 25 D
```

\$JOBUTIL

DE - Delete Data Set

DE identifies a data set to be deleted.

Syntax

DE name, volume

Required: name, volume

Default: None

Operands

Description

name Defines the data set to be deleted

volume Defines the volume from which the data set is to be deleted

Example

DE TEMPDS, EDX003

DS - Identify Data Set

DS identifies a data set to be opened and accessed by a program. DS commands are allowed only between PROGRAM and EXEC commands. Up to nine (9) DS commands can be specified for a program.

Syntax

```
DS          name,volume  
Required:  name  
Default:   volume is IPL volume
```

<u>Operands</u>	<u>Description</u>
-----------------	--------------------

name	Defines the data set to be accessed by a program
volume	Defines the volume that contains the data set to be accessed by a program

Example

```
DS          WORK1,EDX001
```


\$JOBUTIL

EOJ - End of Job

EOJ indicates the end of the primary procedure and must be the last command in the procedure data set. data set.

Syntax

EOJ

No operands are required.

EOP - End of Procedure

EOP indicates the end of a nested procedure and must be the last command in the called procedure data set.

Syntax

EOP

No operands are required.

EXEC - Execute Program

EXEC loads and executes the program identified in the preceding PROGRAM command. EXEC must have been preceded by a PROGRAM command.

Syntax

EXEC	STORAGE=size
Required:	None
Default:	STORAGE=0

Operands Description

STORAGE	Specifies the amount of dynamic storage to be allocated to the program to be executed. This value overrides the value specified when the program was compiled. STORAGE=0 (the default) indicates that the dynamic storage specified (if any) during compilation should be allocated.
---------	--

\$JOBUTIL

JOB - Identify Job

JOB, an optional command, identifies the procedure or a collection of commands within a procedure data set. When a JOB command is read by \$JOBUTIL, a message is printed on the terminal with the Job name, time started and date (if timer support is included).

Syntax

JOB	job-name
Required:	job-name

<u>Operands</u>	<u>Description</u>
-----------------	--------------------

job-name	Names the current group of commands as a job
----------	--

Example

JOB	TEST1
-----	-------

JUMP - Jump to Label

JUMP bypasses \$JOBUTIL commands by testing the completion code of the previously executed program. The test compares the program completion code to 'cc'. The commands between the JUMP and the LABEL command with the same label name are ignored if the condition tested for is 'true'. An unconditional JUMP to a LABEL is also permitted. Jumps are forward only and are limited to the range of the current procedure, for example, a jump cannot occur from one procedure to a label in another procedure. A JUMP with no label name jumps to the next unlabeled LABEL command or to an EOJ or an EOP, whichever occurs first. If an EOJ or an EOP is encountered while searching for a LABEL command, the search for the LABEL command terminates and the EOJ or EOP is executed.

Note: The JUMP and LABEL commands are not permitted in the PAUSE mode.

Syntax

```
JUMP      label-name,op,cc
```

```
Required:  None
```

Operands Description

label-name The name associated with a LABEL command.

op LT, GT, EQ, GE, LE, or NE.

cc Program completion code.

Examples

```
JUMP      LBL1,EQ,20  
JUMP      LBL2  
JUMP
```

\$JOBUTIL

LABEL - Identify Continuation Point

Job processing continues at the LABEL command after a related JUMP command is encountered. The LABEL command is not valid in the PAUSE mode.

Syntax

```
LABEL      label-name  
Required:  None
```

Operands Description

label-name Defines the command where processing can continue.

Examples

```
LABEL      LBL1  
LABEL
```

LOG - Log Control

LOG indicates whether the \$JOBUTIL commands are to be printed as they are read and which terminal device is to print them. LOG commands can be placed anywhere in the procedure data set. If a terminal device is not specified, the commands are logged on the terminal from which the Job Processor was invoked.

Syntax

LOG	Operand
Required:	None
Default:	ON

Operands Description

ON	Indicates that \$JOBUTIL commands are to be printed
OFF	Indicates that \$JOBUTIL commands are not to be printed

terminal-name

Identifies the terminal device which is to print the job processor commands. If omitted indicates that the job processor commands are to be logged on the terminal from which the job processor was called.

Examples

LOG	ON
LOG	OFF
LOG	\$SYSPRTR
LOG	

\$JOBUTIL

NOMSG - No Message Logging

NOMSG sets LOGMSG=NO for the program identified in the preceding PROGRAM command. See the LOAD instruction in the Language Reference for the definition of LOGMSG. NOMSG is invalid if not preceded by a PROGRAM command.

The NOMSG command must be between the PROGRAM and EXEC commands.

Syntax

NOMSG

Required: None

No operands are required.

Example

```
PROGRAM  CALCDEMO
NOMSG
```

PARM - Pass Parameter

PARM identifies the parameters to be passed to the program in the preceding PROGRAM command. PARM must be between the PROGRAM and EXEC commands. Maximum length of the operand on the PARM command is 62 bytes. The parameters specified on the PARM command are passed to the specified program as an EBCDIC character string. In the specified program they can be referenced as beginning at the label \$PARM1 and are packed two characters per word. The length of the string is determined by the 'PARM=n' operand of the PROGRAM statement in the specified program. For example, 'PARM=31' would cause the maximum 62 characters from positions 10-71 of the PARM command to be transferred to the specified program starting at the label \$PARM1.

Syntax

PARM	parameters
Required:	None

Operands Description

parameters Defines parameters to be passed to the program

Example

PARM	NOLIST,XREF
------	-------------

\$JOBUTIL

PAUSE - Await Manual Intervention

PAUSE allows \$JOBUTIL commands to be entered from a terminal keyboard. JUMP and LABEL commands are not permitted.

When a PAUSE command is read, you are prompted for input. You communicate with the processor using the attention key and three PAUSE subcommands.

You can initiate the PAUSE mode by pressing the attention key and typing PAUSE.

PAUSE subcommands are:

- ABORT - To end \$JOBUTIL processing
- ENTER - To enter \$JOBUTIL commands
- GO - To end the PAUSE mode and read the next command in the procedure data set

Syntax

PAUSE

Required: None

No operands are required.

Example

Data Set with PAUSE

```
JOB          PAYROLL
LOG          ON
PROGRAM     WAGES
EXEC
REMARK      LAST DAY OF MONTH?
REMARK      RUN  MONTHEND
PAUSE
EOJ
```

In PAUSE Mode (Terminal Output)

```
PAUSE - * - ATTN: GO/ENTER/ABORT

PAUSE
>ENTER

ENTER COMMAND PROGRAM
ENTER OPERAND MONTHEND
ENTER COMMAND EXEC
ENTER OPERAND
```

PROC - Execute Procedure

PROC allows you to pass control to another procedure data set. The operand identifies the data set. A PROC command is not allowed in the called procedure data set. PROC cannot be placed between a PROGRAM command and EXEC command. The completion code of the last program executed in the sub-procedure will be available for testing by the main procedure.

Syntax

PROC procedure-name, volume

Required: procedure-name

Default: volume is IPL volume

Operands Description

procedure-name

Defines the procedure data set to which control is to be passed.

volume

Defines the volume containing the procedure data set to which control is to be passed.

Examples

PROC PAYROLL, EDX002
PROC PAYROLL

PROGRAM - Identify Program

PROGRAM identifies a program to be executed.

Syntax

```
PROGRAM    program-name,volume
```

```
Required:  program-name
```

```
Default:   volume is IPL volume
```

Operands Description

program-name

Defines the name of the program member to be executed

volume

Defines the name of the volume containing the program member to be executed

Examples

```
PROGRAM    $DEBUG,EDX002
PROGRAM    CHECKS
```

\$JOBUTIL

REMARK - Display Remark

REMARK allows you to output a message to the operator with log ON or OFF. The operand contains the message. The maximum message length is 62 bytes.

Syntax

REMARK comment

Required: None

Operands Description

comment Defines the comment to be displayed

Example

REMARK INSERT DISKETTE EDX005

*** - Comment**

An asterisk (*) in position 1 allows internal comments in the procedure data set. Comments can start in position 2. The internal comment is not printed.

Syntax

* internal comment

Required: None

No operands are required.

Example

* THIS PROCEDURE IS A TEST CASE

\$JOBUTIL

Batch Job Example

The following examples list three procedure data sets. The last two procedures shown are invoked by the first.

List Data Set BATCH on EDX003

```
JOB          BATCH
LOG          $SYSPRTR
REMARK      THIS JOB SHOULD RUN UNINTERRUPTED
PROGRAM     JUTEST1,EDX003
NOMSG
DS          LOAD1,EDX003
PARM        FOF4F1F7
EXEC
PROGRAM     JUTEST2,EDX003
EXEC
JUMP        JMP1,EQ,-10
PROGRAM     CALCDEMO,EDX002
EXEC
LABEL       JMP1
PROGRAM     JUTEST3,EDX003
PARM        2
EXEC
JUMP        JMP2,LT,5
PROGRAM     $EDIT1N,EDX002
DS          EDITWORK,EDX002
EXEC
LABEL       JMP2
PROC        PROC1,EDX003
PROC        PROC2,EDX003
EOJ
```

List Data Set PROC1 on EDX003

```
LOG      ON
REMARK   PROC1
PROGRAM  JUTEST2,EDX003
EXEC
EOP
```

List Data Set PROC2 on EDX003

```
REMARK   PROC2
PROGRAM  JUTEST1,EDX003
DS       LOAD1,EDX003
PARM     ENDT
EXEC
EOP
```


\$LOG

\$LOG - LOG I/O ERRORS INTO DATA SET

\$LOG records information concerning I/O errors onto a disk or diskette data set. The information recorded on the log data set can then be displayed using \$DISKUT2.

Note: You must include the supervisor I/O error logging module (SYSLOG) in your generated system to have I/O error logging capability.

Log Data Set

Before I/O errors can be recorded, you must allocate a log data set to contain the device and system information available at the time of the I/O error. The log data set should be at least eight records long and should be named \$LOGDS. \$LOGDS may reside in a disk or diskette volume. The log data set contains one 256-byte log entry per record. The first two records are used for control information.

If the log data set was previously used and contains valid entries, new entries are added after the old ones. If the data set is not initialized, a new log control record is written, indicating that no entries are in the log data set. When the log data set becomes full, the following message is sent to the terminal:

```
$LOG - *** INSUFFICIENT BUFFERS FOR LOG RATE ***
```

Following this message, subsequent log entries received are written over the oldest entries.

Invoking \$LOG

To activate error logging, load \$LOG into any partition. Use the session manager diagnostic utilities option menu or the \$L operator command. You are prompted for the name of the log data set. After \$LOG is loaded and error logging is activated, you can use attention commands provided by \$LOG to deactivate, reactivate or terminate error logging. You can also reinitialize the log data set using one of these attention commands. For

information about I/O error logging, system generation considerations, and execution time interfaces, refer to the System Guide.

Example: The following example shows how to activate error logging and the attention commands that are available.

```
> $L $LOG $LOGDS,EDX002
$LOG          21P, LP= 7400

*****
*           $ L O G       U T I L I T Y           *
*                                                                 *
* THE FOLLOWING ATTENTION COMMANDS ARE AVAILABLE: *
*   ATTN/$LOGOFF  - TEMPORARILY DEACTIVATE LOGGING *
*   ATTN/$LOGON   - REACTIVATE LOGGING             *
*   ATTN/$LOGINIT - INITIALIZE LOG DATA SET      *
*   ATTN/$LOGTERM - TERMINATE LOGGING             *
*   ATTN/$LOG     - REISSUE COMMAND LIST          *
*                                                                 *
* WARNING: DO NOT CANCEL($C) THIS PROGRAM.        *
*****
LOGGING ACTIVATED
```

\$MOVEVOL

\$MOVEVOL - DISK VOLUME DUMP/RESTORE

\$MOVEVOL dumps the contents of an Event Driven Executive direct access volume to diskette when such a volume spans several diskettes.

\$MOVEVOL also restores a volume from diskette to disk. Thus, \$MOVEVOL provides the facility for transferring large amounts of data from one system to another or for creating backup copies of an online data base.

Diskette Usage

Diskette Contents

The first of the set of diskettes used for the dump function, called the control diskette, records control information and the volume directory. The rest of the diskettes store the data portion of the volume being dumped. Control information is recorded on each data diskette to identify the diskette contents and to ensure that it contains data related to the dump operation described on the control diskette.

Diskette Format

All diskettes must be formatted using \$DASDI to contain 128-byte sectors. Either single-sided or double-sided diskettes can be used; however, all diskettes in a set must be the same type. Each diskette must contain a volume label in the standard format. The volume label must be a six-character field in which the last two characters are used for sequencing; for example, SAVE00, SAVE01, ..., SAVEnn, where nn is the last diskette used. All diskettes used must have the same four-character prefix.

4966 Diskette Usage Considerations

If you are using the 4966 diskette magazine unit for your dump/restore operation, you can use diskette magazines or an individual diskette slot. If you use an individual diskette slot, then all of the subsequent diskettes mounted must be placed in the same slot. If you use diskette magazines, you must have all of your diskettes in the correct sequence with no empty slots in the magazine. The first volume with the suffix 00 must be in slot number 1 of the first magazine. You can use either or both of the diskette magazines, A and B.

Data Set Specification

If \$MOVEVOL is invoked with the \$L command, you are prompted to enter the names of the data sets and volumes to be used.

Figure 21 shows the parameter menu displayed when \$MOVEVOL is invoked using the session manager. Enter the requested information and press ENTER.

```

|
| $SMM0308: SESSION MANAGER $MOVEVOL PARAMETER INPUT MENU-
| ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN
|
|     DISK      ($$EDXLIB,VOLUME) ==>
|
|     DISKETTE  (NAME,VOLUME)  ==>
|
|
```

| Figure 21. \$MOVEVOL parameter input menu

\$MOVEVOL

Dump Procedure

The following steps are required to dump the contents of a direct access volume onto diskette.

1. Set up a control diskette.
 - a. Use \$INITDSK to:
 - 1) Initialize the control diskette with a volume label that is suffixed with 00 (for example, SAVE00).
 - 2) Create a directory of at least 2 records.
 - 3) If the diskette is to be used to IPL another system, reserve space for a nucleus of the appropriate size and write the IPL text.
 - b. Use \$DISKUT1 to:
 - 1) Determine the directory size, in records, of the volume to be dumped.
 - 2) Change volume to the control diskette (for example, SAVE00) and allocate a control data set with the same name as the name of the volume to be dumped. The member size of the control data set must be one record larger than the size of the volume being dumped.
 - c. Use \$COPYUT1 to:
 - 1) Copy other data sets onto the control data set. For example, you may require \$EDXNUC, the transient loader, or a copy of \$MOVEVOL.

Note: The first record in the control data set contains control information and up to 50 characters of text describing the data being dumped. The remaining space stores a copy of the directory of the volume being dumped.
2. Set up a series of data diskettes. For each data diskette:
 - a. Use \$INITDSK to:
 - 1) Create a volume label. The volume label of each diskette must have the same four-character prefix as the control diskette and a two-character suffix indicating the sequence number, for example,

SAVE01, SAVE02,, SAVEnn.

- 2) Create an owner ID field.
- 3) Allocate a two-record directory.

b. Use \$DISKUT1 to:

- 1) Allocate a one-record data set named \$CONTROL.
- 2) Allocate a second data set with the name of the volume to be dumped (for example, EDX002). The second data set must use the remaining available space on the diskette (946 records for a single-sided diskette and 1921 records for a double-sided diskette).

3. Mount the control diskette, vary it online and load \$MOVEVOL for execution.

You must specify two data sets at load time:

DISK The volume on disk to be dumped. Specify \$EDXLIB,volser.

DISKETTE The control data set on diskette. Specify dsname,volser.

\$MOVEVOL asks if you wish to dump from disk to diskette.

\$MOVEVOL then determines the number of additional diskettes required to dump the referenced volume (DISK), informs you of this requirement, and asks whether the procedure should be continued. A negative response terminates the operation. If the response is positive, the control information and disk directory are recorded on the control diskette and you are asked to mount a new diskette for transfer of the data portion of the volume being dumped.

4. Each time a diskette is filled, \$MOVEVOL requests another diskette. Mount as many data diskettes as requested.

\$MOVEVOL

Example

Dump Operation Using a 4966 Diskette Magazine Unit

```
> $L $MOVEVOL
DISK(NAME,VOLUME): $$EDXLIB,EDX002
DISKETTE(NAME,VOLUME): $EDX002,SAVE00

$MOVEVOL      20P,10:07:52, LP=5200

DUMP LIBRARY FROM VOLUME EDX002 TO DISKETTE? Y

PROCESSING DISKETTE VOLUME SAVE00
ENTER LIBRARY IDENTIFICATION (1-50 CHAR.):
DUMP OF EDX002 - DATE IS 09/14/77
    11 MORE DISKETTES ARE REQUIRED, CONTINUE? Y

PROCESSING DISKETTE VOLUME SAVE01
PROCESSING DISKETTE VOLUME SAVE02
PROCESSING DISKETTE VOLUME SAVE03
PROCESSING DISKETTE VOLUME SAVE04
PROCESSING DISKETTE VOLUME SAVE05
PROCESSING DISKETTE VOLUME SAVE06
PROCESSING DISKETTE VOLUME SAVE07
PROCESSING DISKETTE VOLUME SAVE08
PROCESSING DISKETTE VOLUME SAVE09

MOUNT NEXT DISKETTE OR MAGAZINE
REPLY -Y- WHEN DONE: Y

PROCESSING DISKETTE VOLUME SAVE10
PROCESSING DISKETTE VOLUME SAVE11

VOLUME DUMP OPERATION COMPLETE
9200 RECORDS TRANSFERRED

$MOVEVOL ENDED 10:10:13
```

Restoration Procedure

The following steps are required for a restore operation.

1. Mount the control diskette, vary it online and load \$MOVEVOL for execution.
 - a. Respond as described previously in Dump Procedure to requests for data sets.
 - b. Select the restoration mode by responding N to the query for disk to diskette dump and Y to the query for diskette to disk restoration.
2. Mount data diskettes as requested.

Examples

Restore Operation Using a 4964 Diskette Unit: The source is smaller than the receiving volume in size.

```
> $L $MOVEVOL,GREYV2
DISK      (NAME,VOLUME): $$EDXLIB,MACLIB
DISKETTE(NAME,VOLUME): $MACLIB,BACK00
$MOVEVOL      20P,00:26:08, LP= 7000
DUMP LIBRARY FROM VOLUME MACLIB TO DISKETTE? N

RESTORE LIBRARY FROM DISKETTE TO DISK VOLUME MACLIB ? Y
RESTORING 5719-XX2 VO1M01 JAN. 12, 1979 TO VOLUME MACLIB
CONTINUE? Y
SOURCE VOLUME IS SMALLER THAN THE TARGET. CONTINUE? Y
COMPRESS THIS LIBRARY AFTER INSTALLATION.
PROCESSING DISKETTE VOLUME BACK00
MOUNT NEXT DISKETTE OR MAGAZINE
REPLY -Y- WHEN DONE: Y
PROCESSING DISKETTE VOLUME BACK01
.
.
.
```


\$MOVEVOL

Restore Operation Using an Individual 4966 Diskette Magazine Slot: The source and target volumes are equal in size.

```
> $L $MOVEVOL $$EDXLIB,EDX003 $EDX003,SAVE00
$MOVEVOL      20P,11:05:05, LP=6300
DUMP LIBRARY FROM VOLUME EDX002 TO DISKETTE? N
RESTORE LIBRARY FROM DISKETTE TO DISK VOLUME EDX002? Y
RESTORING DUMP OF EDX002 - DATE IS 09/14/77
CONTINUE? Y

PROCESSING DISKETTE VOLUME SAVE00
MOUNT NEXT DISKETTE OR MAGAZINE
REPLY -Y- WHEN DONE: Y

PROCESSING DISKETTE VOLUME SAVE01
MOUNT NEXT DISKETTE OR MAGAZINE
REPLY -Y- WHEN DONE: Y

PROCESSING DISKETTE VOLUME SAVE02
MOUNT NEXT DISKETTE OR MAGAZINE
REPLY -Y- WHEN DONE: Y

PROCESSING DISKETTE VOLUME SAVE03
MOUNT NEXT DISKETTE OR MAGAZINE
REPLY -Y- WHEN DONE: Y

PROCESSING DISKETTE VOLUME SAVE04

VOLUME INSTALLED
3600 RECORDS TRANSFERRED

$MOVEVOL ENDED 11.10.56
```

\$PFMAP - IDENTIFY 4978 PROGRAM FUNCTION KEYS

The \$PFMAP utility identifies the program function keys on the 4978 display station. As each key is pressed, its associated system code is displayed in decimal and hexadecimal. A key's associated system code is the identification returned at completion of a WAIT KEY instruction or an ATTNLIST interrupt. The hardcopy key is active during execution of this program, and its code is not displayed. The program is terminated by pressing the ENTER key.

\$PREFIND

\$PREFIND - PREFIND DATA SETS AND OVERLAYS

\$PREFIND locates the disk, diskette, or tape data sets and overlay programs to be referenced by your program and stores their addresses in the header of your program. After \$PREFIND has executed, the tasks performed by \$LOADER are reduced and future program load times for the your program are shortened.

Under certain carefully controlled conditions, \$PREFIND reduces the time required to load other programs from disk or diskette for execution. \$PREFIND is normally used as the last step in the program preparation process, following the execution of \$UPDATE; however, it can be used to process a program at any time after the program has been processed by \$UPDATE or after a previous processing by \$PREFIND as is described as follows.

\$PREFIND is most effective when it is used to process programs that reference a large number of disk, diskette, or tape data sets and overlay programs and when these programs must be loaded frequently from disk or diskette.

Program Load Process Overview

If a program uses data sets or overlays programs (DS= and PGMS= keyword parameters in PROGRAM statement), the assembly process creates control blocks in the program header for each data set and overlay program specified. Space is reserved in these control blocks for the physical disk/diskette/tape addresses of the data sets and overlay programs defined.

After completion of the program preparation process (\$LINK if required, and then \$UPDATE), the executable load module can be loaded to storage. The system program that performs the load operation is a transient loader program (\$LOADER), and part of that operation includes filling in the actual physical addresses of data sets and overlay programs in the control blocks of the program header of the program that was loaded. When a large number of data sets and/or overlay programs are defined, this can be a time-consuming process, as \$LOADER must search a volume directory for each data set/program used.

In this manner, all data sets and overlay programs required by a program are located and their sizes determined each time the using program is loaded for execution. Thus, the loaded program will execute correctly even if the size or location of one or more of the data sets or overlay programs it uses has changed.

The program loading process just described provides maximum safety and flexibility in loading programs for execution. However, the time required to locate each data set and overlay program each time a program is loaded may be undesirable in some circumstances.

\$PREFIND allows data sets and overlay programs to be located prior to program load time, and written directly into the program header on disk or diskette. When a program is loaded, the information required is already present, and load time is therefore reduced.

Note: See Internal Design for a description of the Relocating Program Loader (\$LOADER), the program header, and Data Set Control Blocks (DSCBs).

\$PREFIND Usage Cautions

1. Use \$PREFIND only to locate data sets and/or overlay programs in well tested production systems where the size and location of the referenced data sets and overlay programs are unlikely to change and where programs are loaded so frequently that the saving of a few seconds in program load time results in a useful performance increase for the system.

2. Use \$PREFIND only in situations where the disk, diskette, or tape locations of referenced data sets and overlay programs will remain relatively static. If a referenced data set or overlay program is moved to a new location on disk or diskette, or to another tape volume, then programs that have been processed by \$PREFIND will not be aware of the new location or size. This will produce undesirable results the next time the programs are executed.

\$PREFIND must be used again to insert the new information into the program headers of all referencing programs.

3. Before loading a program that references data sets on removeable media, such as diskette or tape, make sure that the proper diskette or tape volume containing the prefound data sets has been mounted and varied online. If the correct diskette or tape volume is not mounted, the program will use whatever volume is mounted at the time of execution. If you should access an incorrect volume, unpredictable results may occur and possibly destroy the data on the diskette or tape.

\$PREFIND

4. The system does not keep track of referenced data sets or overlay program locations that are no longer valid. It is your responsibility to prevent disk, diskette, or tape data sets and program overlays from being incorrectly modified.

System functions that can change the size or location of data sets or overlay programs are:

- DE and AL commands of \$DISKUT1
 - CD, ST, and RT commands of \$TAPEUT1
 - \$COPYUT1
 - \$UPDATE, following reassembly of a program
 - \$COMPRES
5. If you generate a new system and IPL from that system, make sure that the disk, diskette, or tape control blocks have not been relocated. A change in control block location within the supervisor requires that \$PREFIND be used again to locate any referenced data sets and overlay programs.

Changes made during system generation that can affect the location of control blocks are:

- Adding, deleting, or changing \$EDXDEF statements.
- Modifying the \$LNKCNTL data set to add additional INCLUDE statements.
- Relinking the supervisor with new or modified object modules, for example, \$SYSCOM or PTF module replacements.

\$PREFIND Commands

The \$PREFIND commands are listed below. When invoked using \$L, \$PREFIND prompts you for the information required for each command. When invoked through \$JOBUTIL as a batch processing job, \$PREFIND must be supplied with the necessary information with the PARM command of \$JOBUTIL, as described on the following pages.

Command	Description
PF	PRELOCATE DATA SETS AND OVERLAYS
DE	DELETE PREFOUND STATUS
EN	END \$PREFIND
?	LIST AVAILABLE COMMANDS

Invoking \$PREFIND

\$PREFIND can be invoked by the \$L command, the job stream processor (\$JOBUTIL), or the program preparation option of the session manager.

Invoking \$PREFIND Using \$L

When invoked with \$L, \$PREFIND prompts you for the command name and, if either PF or DE is entered, prompts you for the name and volume of the program, as well as the numbers (1 through 9) of the data sets and overlay programs that are to be located or whose prefound status is to be deleted. All of the required information can be entered without waiting for the prompting messages. For example:

```
PF MYPROG,EDX003 D=(1,2,4,7) P=(1,2,3)
```

here the numbers in parenthesis correspond to the numbers used in the DSn and PGMn parameters on the READ/WRITE and LOAD Event Driven Language instructions.

The data set and program numbers must always be entered in the formats D=(_,_,_) and P=(_,_). Data set numbers, if present, must always precede overlay program numbers. The word ALL can be used in place of the number string within the parenthesis if desired. A null response to the prompt for either (but not both) the data set or program numbers can be made, if appropri-

\$PREFIND

ate, and no change in the status of the data sets or programs will occur.

Only those data set or program numbers whose status is to be changed should be entered. For example, if a program references six data sets and it is desired to prelocate the first three and the sixth, then D=(1,2,3,6) would be entered when using the PF command. If at a later time it is desired to no longer have the third data set in the prefound state, a DE command specifying D=(3) would be used. After performing the two commands described above, data sets 1, 2, and 6 would be prefound and data sets 3, 4, 5 would not. In the example given here, the execution of the DE command only affects DS3 and does not update the information in the program header concerning DS1, DS2, or DS6 should they have been moved to different locations on disk between the execution of the PF and DE commands described.

Note: During execution of the DE command, a data set can be deleted which was originally defined in the format (dsname,??). In this case, \$PREFIND prompts you to enter the prompting dsname to be placed back into the program header since the original name was overridden during the previous PF command. If the DE command is invoked via \$JOBUTIL, an error message occurs if the above condition is encountered.

Any data set or overlay program not marked as being in the prefound state is located by \$LOADER whenever the using program is loaded into storage for execution.

Examples

Processing One Program Without Prompting Messages

```
> $L $PREFIND
$PREFIND      22P,00:06:15, LP= 9800

COMMAND (?): PF TESTPREF,EDX001 D=(1,2,3,5,7,9) P=(ALL)
COMMAND COMPLETED
COMMAND (?): EN

$PREFIND ENDED AT 00:07:30
```

Processing Multiple Programs with/without Prompting Messages

```
COMMAND (?): PF
PGM(NAME,VOLUME): TESTPREF,EDX001
ENTER DATA SET NUMBERS: D=(1,2,3,7,9)
ENTER OVERLAY PGM. NUMBERS: P=(ALL)
```

```
COMMAND COMPLETED
COMMAND (?):
```

```
COMMAND (?): PF
PGM(NAME,VOLUME): TESTPRE2,EDX001 D=(ALL) P=(ALL)
```

```
COMMAND COMPLETED
COMMAND (?): EN
```

```
$PREFIND ENDED AT 00:10:59
```


\$PREFIND

Invoking \$PREFIND Using \$JOBUTIL

When \$PREFIND is invoked through \$JOBUTIL, it requires the same information as was described under 'Invoking \$PREFIND Using \$L'. This information is provided with a PARM command having the format shown below. The number of spaces between the operands in the PARM command may be one or more, as long as the total number of characters, including spaces, does not exceed 62.

Syntax

Character Position 1 PARM	10 programe,volume c D=(,_,_) P=(,_,_)
------------------------------------	---

<u>Operands</u>	<u>Description</u>
programe	The name of the program whose data set and overlay program status is to be modified.
volume	The volume of residence of the program whose data set and overlay program status is to be modified.
c	Either character P for PREFIND or character D for DELETE
,,_	The string of data set or overlay program numbers, or the word ALL

When invoked with \$JOBUTIL, \$PREFIND performs either a single prefind or delete function and then terminates its execution. \$PREFIND directs its error and/or termination messages to the device defined as \$SYSPRTR.

Example

Messages and responses on your terminal:

```
> $L $JOBUTIL
$JOBUTIL      3P,02:23:15, LP= 6500
ENTER PROCEDURE (NAME,VOLUME): PREPROC,EDX001

$JOBUTIL ENDED AT 02:23:38
```

Output on \$SYSPRTR:

```
LOG          $SYSPRTR
*** JOB - PREFIND - STARTED AT 02:23:28 00/00/00 ***

JOB          PREFIND
PROGRAM      $PREFIND,EDX001
NOMSG
PARM         D TESTPREF,EDX001 D=(ALL) P=(ALL)
EXEC

COMMAND COMPLETED

$PREFIND ENDED AT 02:23:37
```

The procedure used:

```
00010 LOG          $SYSPRTR
00020 JOB          PREFIND
00030 PROGRAM      $PREFIND,EDX001
00040 NOMSG
00050 PARM         D TESTPREF,EDX001 D=(ALL) P=(ALL)
00060 EXEC
00070 EOJ
```

\$PREFIND

Invoking \$PREFIND Using the Session Manager

The session manager option menu for program preparation can be used to invoke \$PREFIND. Selection of \$PREFIND causes the following parameter option menu to be displayed:

```
$SMM0208: SESSION MANAGER $PREFIND PARAMETER INPUT MENU--  
ENTER/SELECT PARAMETERS                DEPRESS PF3 TO RETURN
```

```
COMMAND (P/D)           ==> D  
PROGRAM (NAME,VOLUME) ==> TESTPREF,EDX001  
DATASET #'S (OR ALL)  ==> ALL  
PROGRAM #'S (OR ALL)  ==> ALL
```

After the parameters are entered, \$PREFIND executes as if it were invoked using \$JOBUTIL.

\$TAPEUT1 - TAPE MANAGEMENT

\$TAPEUT1 performs several commonly used tape management functions. You can initialize tapes, allocate tape data sets, copy data sets or volumes to or from tape, copy tape to tape, print tape records, dump/restore disk devices, and test the tape transport hardware.

\$TAPEUT1 is invoked with the \$L command or primary option 3 of the session manager. Once invoked and prior to accepting commands, \$TAPEUT1 displays the following information about the tapes defined to the system:

- TAPEID
- density selection (800, 1600, DUAL)
- label type (SL, NL, BLP)
- current density setting
- online or offline
- volume information (if an online SL tape)
- device address

Example: Loading \$TAPEUT1 and its automatic display of the system tapes.

```
> $L $TAPEUT1
$TAPEUT1      21P,11:11:33, LP= 0000
TAPE01 DUAL SL 1600 OFFLINE
      DEVICE ADDRESS = 004C
TAPE02 DUAL NL 1600 OFFLINE
      DEVICE ADDRESS = 004D

COMMAND (?):
```

Note: Error logging of tape errors is available. Refer to the System Guide.

\$TAPEUT1

\$TAPEUT1 Commands

The commands available under \$TAPEUT1 are listed below. To display this list at your terminal, enter a question mark in reply to the prompting message COMMAND (?):.

COMMAND (?): ?

- CD - COPY DATASET
- CT - CHANGE TAPE ATTRIBUTES
- DP - DUMP TAPE
- EN - END \$TAPEUT1
- EX - EXERCISE TAPE
- IT - INITIALIZE TAPE
- MT - MOVE TAPE
- RT - RESTORE DISK/VOL FROM TAPE
- ST - SAVE A DISK/VOL ON TAPE
- TA - ALLOCATE TAPE DATASET

COMMAND (?):

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, CD).

CD - Copy Data Set

CD copies a disk or diskette data set onto a tape, copies a tape data set into a disk or diskette data set or onto another tape. The command writes a trailer label at the end of the data set on the target tape if it is a standard label tape. Header labels are not written on standard or non-labeled tapes, therefore, the target tape data set must be preallocated.

If a disk or diskette data set is being copied to tape, the tape records will be 256-bytes. If a tape data set from another system (for example, a S/370) is copied to a disk or diskette and the source records are not 256-bytes, the source records are split into multiple 256-byte records with any unused bytes padded with zeros. Prior to copying, you are prompted for the maximum input record size.

Consider the following when you are copying data sets:

- When you reach a tapemark (end of input data), you are prompted to continue. If you have more records to copy, you can continue; however, make sure that there is sufficient room on the target tape. You are prompted at every tapemark encountered on the source tape. If you do not wish to continue, the trailer label is written on the target tape.
- To copy the contents of one tape to another tape, thereby creating an exact duplicate of the entire tape (header label and data records or only data records), you can use either of two methods:
 - To copy only data records, initialize the target tape (using the IT command) so that it has the same label type as the source tape. Copy (using the CD command) the source tape to the target tape. This allows you to create a new header label on the target tape and to duplicate only the data records from the source tape.
 - To create an exact duplicate of the source tape, mount the source and target tapes on drives specified for bypass label-processing. Then copy (using the CD command) the entire source tape. The target tape becomes an exact duplicate of the source all label records, all data records, and all trailer labels).

\$TAPEUT1

Example

Copying Data from a Disk to a Tape

COMMAND (?): CD

SOURCE (NAME,VOLUME): \$TAPEUT1,EDX002
TARGET (NAME,VOLUME): DATA1111,123456
ENTER SOURCE BLOCKSIZE (NULL=DISK(ETTE)):
USE ATTN/CA TO CANCEL COPY

ARE ALL PARMS CORRECT? (Y,N): Y
EOD ON SOURCE DATASET
25 RECORDS COPIED

COMMAND (?):

CT - Change Tape Drive Attributes

CT resets the label type and density for any tape drive. The label type and density are set at system generation. CT allows you to dynamically reconfigure the tape drives. You must vary off the tape drive before you can change its attributes.

```
CT nn
```

where nn = hexadecimal device address

The current settings for label and density are displayed. You are prompted to enter any changes.

The CT command fails and issues an error message on the terminal for the following conditions:

- invalid device address
- tape drive is not varied offline
- invalid label type
- invalid density
- tape drive not defined as dual density

\$TAPEUT1

Examples

Specify Changes to the Label Processing and Density Selection
for Tape Drive at Address 4C.

```
COMMAND (?): CT TAPE01

ENTER TAPEID (1-6 CHARS): TAPE01
TAPE TAPE01 AT ADDR 4C IS BLP 1600 BPI
DO YOU WISH TO MODIFY (Y OR N)?: Y
LABEL (NULL,SL,NL,BLP)?: SL
DENSITY (NULL,800,1600): 800
TAPE TAPE01 AT ADDR 4C IS SL 800 BPI

COMMAND (?):
```

No Changes are Made for Label Processing and Density Selection
for Tape Drive at Address 4D.

```
COMMAND (?): CT TAPE02

ENTER TAPEID (1-6 CHARS): TAPE02
TAPE TAPE02 AT ADDR 4D IS SL 800 BPI
DO YOU WISH TO MODIFY (Y OR N)?: N

COMMAND (?):
```

DP - Dump Records

DP dumps tape record(s) on the system printer (\$SYSPRTR), on the terminal from which you invoked \$TAPEUT1, or on the terminal you specify. The output format is similar to the \$DISKUT2 utility; hexadecimal data plus the EBCDIC conversion.

The record number for each record is shown and the words 'tapemark' are printed when a tapemark is encountered. You are prompted for the number of records to be printed. When a tapemark is detected, you are prompted to continue or to terminate the dump.

You are prompted for the maximum record size. This allows you to print record sizes up to the maximum amount of storage available. If the record being read is smaller or larger than the maximum size specified, a message will be issued indicating:

- a wrong length record was encountered
- the actual record size

Smaller records are printed and padded with zeros. Larger records are truncated and printed to the maximum size specified. You are informed of the actual record size.

Notes:

1. The tape must be varied offline before you can dump records.
2. The TAPEID specified must match the one specified at system generation.
3. This is an offline utility; therefore, the tape will not be automatically rewound when the dump is finished. Use the move tape command (MT) to rewind the tape.
4. The dump command (DP) and the move tape command (MT) can be used together to selectively search and dump portions of a tape.

\$TAPEUT1

Example

Dump Five Records on Display Terminal

```
COMMAND (?): DP

ENTER MAXIMUM BLOCKSIZE: 256
USE ATTN/CA TO CANCEL DUMP
ENTER TAPEID (1-6 CHARS): TAPE01
ENTER NUMBER OF RECORDS TO DUMP: 5
PRINT DUMP ON $SYSPRTR? (Y,N): N
ENTER TARGET TERMINAL NAME (*,NAME): *
```

(The dump is directed to the terminal
you are currently assigned to)

```
COMMAND (?):
```

To direct the dump to the \$SYSPRTR or to a terminal of your choice, you are prompted and can respond as follows:

- Y Print dump on \$SYSPRTR
- N Display dump on the terminal of your choice. You are prompted for the terminal name and can respond as follows:
- * = terminal on which you are currently assigned
 - name = device name of the target terminal

If the tape you want to dump is currently in use, you are prompted as follows:

```
DEVICE NOT OFFLINE
DO YOU WISH TO CONTINUE? (Y,N):
```

If you specify Y, the dump will continue but the device will be unuseable by the original user.

| **EX - Exercise Tape**

| EX, a software exerciser, performs two operations:

- | • It exercises any of the three label type tapes to ensure that the I/O commands to that tape are executing correctly.
- | • It analyzes the surface of the tape to verify that the surface is free of defects. Any errors and their approximate location on the tape are printed on the system printer (\$SYSPRTR).

| **Caution:** Surface analysis writes records over the information currently on the tape. Any existing data records or labels are destroyed.

| Each operation is optional and you are prompted before continuing.

| The EX command performs the following functions:

- | • Writes 600 unique records to a data set on the tape
- | • Closes and reopen the data set
- | • Finds a particular record within the data using NOTE/POINT
- | • Verifies that the correct record is accessed
- | • Performs a surface analysis of the tape by writing over the tape and then verifies each record.

\$TAPEUT1

Example

```
COMMAND (?): EX
TARGET (NAME,VOLUME: MYDATA,123456
USE ATTN/CA TO CANCEL THE EXERCISER
DO YOU WANT TO EXERCISE THE SOFTWARE (Y/N): Y
...
  Exerciser runs and prints status on printer
...
WRITE/READ ENTIRE SURFACE OF TAPE? (Y/N): Y
...
  Exerciser writes on entire surface of tape, then
  reads and verifies each record
...
TAPE EXERCISER ENDED
```

A sample of the data printed by the EX command follows.

```

TAPE EXERCISER STARTED
WRITE 600 UNIQUE RECORDS TO TAPE

TEST DATA WRITTEN
STEP          1 SUCCESSFUL
CLOSE DATASET FOR REUSE
STEP          2 SUCCESSFUL
READ 250 OF THE RECORDS
NOTE PRESENT POSITION
STEP          3 SUCCESSFUL
POINT TO RECORD 150 AND READ THAT RECORD
STEP          4 SUCCESSFUL
NOTE PRESENT POSITION
STEP          5 SUCCESSFUL
.
.
.
CLOSE DATASET FOR REUSE
STEP          18 SUCCESSFUL
REOPEN THE DATASET
READ 598 OF THE RECORDS
ATTEMPT TO READ MORE RECDS WITH 1 STMT THAN ARE AVAILABLE
VERIFY END-OF-DATA RETURN CODE
STEP          19 SUCCESSFUL
VERIFY 2ND WORD OF TCB CONTAINS ACTUAL # REALLY READ
STEP          20 SUCCESSFUL
WRITE/READ ENTIRE SURFACE OF TAPE? Y
THIS SECTION WILL DESTROY ALL LABEL FIELDS
DO YOU WISH TO CONTINUE? (Y,N): Y
REWIND TAPE TO LOAD POINT
STEP          21 SUCCESSFUL
WRITE ENTIRE TAPE
WRITE TAPEMARKS ON END OF TAPE
STEP          22 SUCCESSFUL
REWIND TAPE TO LOAD POINT
STEP          23 SUCCESSFUL
READ AND VERIFY ALL THE RECORDS
REWIND TAPE TO LOAD POINT
STEP          24 SUCCESSFUL
CLOSE TAPE OFFLINE
STEP          25 SUCCESSFUL

TAPE EXERCISER SUCCESSFUL

```

\$TAPEUT1

IT - Initialize Tape

IT completely initializes a new tape, or changes the label information on a used tape. The IT command initializes tapes for non-labeled and standard label use.

When you initialize a tape as non-labeled, the IT command writes three tapemarks deleting any previous labels on the tape.

When you initialize a tape as standard label, the IT command writes on the tape:

- a volume label (VOL1)
- a header label (HDR1)
- 2 tapemarks to delimit the label information and to indicate an empty data set
- a trailer label (EOF1) and 2 tapemarks signifying the end of data on the tape

You are prompted for the the contents of all required label fields.

Example

```
COMMAND (?): IT TAPE01

ENTER TAPEID (1-6 CHARS): TAPE01
STANDARD LABEL 1600 BPI? Y
TAPEDS (NAME,VOLUME): DATA1111,123456
OWNERID (1-10 CHARS): OWNER-ID
EXPIRATION DATA (YYDDD): 79001
TAPE INITIALIZED

COMMAND (?):
```

Notes:

1. Your tape must be varied offline before you can initialize it. If the tape is not offline, a warning message is issued and you are prompted to continue.

2. If you are changing the label information on a used tape, you must use the EX parameter of \$VARYON to override an unexpired expiration date.
3. Your tape is initialized with the same attributes (label and density) as those defined for the tape drive on which the tape is mounted. Refer to the TAPE statement in the System Guide.
4. When specifying the volume and data set names, do not use the same names as were specified for tape ID at system generation.

\$TAPEUT1

MT - Move Tape

MT provides functions to the terminal user that control tape motion on the specified tape. The available control functions are:

BSF - Back space file

BSR - Back space records

FSF - Forward space file

FSR - Forward space records

OFF - Set device offline

REW - Rewind

ROFF - Rewind offline

WTM - Write tapemark

A count is available for FSR, BSR, FSF, BSF, and WTM so that you can specify the number of records to be spaced over or the number of tapemarks to be written.

An EOT (end-of-tape) terminates only the write tapemark (WTM) function and issues a return code. If you wish to proceed past the EOT, you must request another WTM.

Notes:

1. You can proceed past the EOT, however, make sure that there is sufficient tape to perform the operation.
2. The tapemark record is smaller than the end-of-tape (EOT) indicator so you could possibly receive two or more end-of-tape indications for the same EOT.

A tapemark terminates FSR or BSR operations and the tape is positioned following that tapemark.

Example

Move TAPE01 Forward 3 Records

```

COMMAND (?): MT

ENTER TAPEID (1-6 CHARS): TAPE01
TYPE? FSR/BSR/FSF/BSF/WTM/REW/ROFF/OFF: FSR 3
* the action occurs *
FSR SUCCESSFUL
TYPE? FSR/BSR/FSF/BSF/WTM/REW/ROFF/OFF: END
$TAPEMT ENDED AT 00:20:39

COMMAND (?):
    
```

If the tape was positioned at the first record and the utility forward spaces 3 records, the tape is positioned at the 4th record.

Notes:

1. Your tape must be varied offline before you can issue the motion commands. If the tape is not offline, a warning message is issued and you are prompted to continue.
2. The response to 'ENTER TAPEID' must be the same TAPEID that was specified at system generation.
3. This is an offline utility; therefore, the tape will not be repositioned or rewound when it end. Use the MT command to rewind the tape.

\$TAPEUT1

RT - Restore Disk or Disk Volume From Tape

RT restores a disk or disk volume from a tape. The tape must have been previously created using the ST command. You can restore a disk volume from a tape to the same device type or a different device type, except for the IPL volume. (The IPL volume is treated similarly to the disk device; it must be restored to the same device type). To restore an entire disk device from tape, the device type and model number of the source and target disks must match.

Certain conditions (for example, disaster recovery) can make it necessary to restore an entire disk from tape. To perform a restore when the disk has been destroyed, the following procedure is suggested since tape support is not included in the distributed starter system:

1. Create on a diskette a nucleus that includes tape support and include \$TAPEUT1 on the diskette. Keep this diskette with the backup tapes you create with the ST command.
2. When a restore is necessary, IPL with this diskette, restore the disk from the backup tapes, and IPL from the restored disk.

To create the diskette that you will IPL:

1. Use your current system with tape support included
2. Use \$INITDSK to initialize and write IPL text on the diskette with the following attributes:
 - DIRECTORY SIZE=3
 - CREATE NUCLEUS?: Y
 - SIZE=64
 - IPL TEXT?: Y
3. Copy the current system using \$COPY (CD) onto the diskette as follows:
 - SOURCE: \$EDXNUC,EDX002
 - TARGET: \$EDXNUC,diskette ID
4. Use \$COPYUT1 (CM) to copy the following modules to the diskette:
 - \$LOADER

- \$TAPEUT1
- \$TAPERT
- Your terminal support (for 4978/4979 support, copy \$4978ISO and \$4978CS0)

With this procedure, you create a diskette copy of the IPL text and support modules necessary to IPL your system and restore the disk from tape without generating your system again.

Examples

Restore Volume from a Tape

```

> $VARYON 4C
TAPE01 ONLINE
> $L $TAPEUT1
$TAPEUT1      19P,00:08:48, LP=  7A00
TAPE01 DUAL NL 1600 ONLINE
      DEVICE ADDRESS =  004C
TAPE02 SL 1600 OFFLINE
      DEVICE ADDRESS =  004D

COMMAND (?): RT

*****
*   WARNING:  TO ENSURE PROPER   *
*   DISK CONTENTS, THE SYSTEM   *
*   SHOULD BE INACTIVE WHILE    *
*   RUNNING THIS UTILITY        *
*****

SOURCE  (NAME,VOLUME): X,TAPE01
TARGET  (NAME,VOLUME): $$EDXVOL,ASMLIB
DEVICE RESTORE? N
ARE ALL PARMS CORRECT? (Y,N): Y

USE ATTN/CA TO CANCEL THE RESTORE

VOLUME RESTORED
COMMAND (?):

```

\$TAPEUT1

Restore Disk Device from a Tape

```
> $VARYON 4C
TAPE01 ONLINE
> $L $TAPEUT1
$TAPEUT1 19P,00:12:06, LP= 7A00
TAPE01 DUAL NL 1600 ONLINE
    DEVICE ADDRESS = 004C
TAPE02 SL 1600 OFFLINE
    DEVICE ADDRESS = 004D

COMMAND (?): RT

*****
*   WARNING:  TO ENSURE PROPER   *
*   DISK CONTENTS, THE SYSTEM   *
*   SHOULD BE INACTIVE WHILE   *
*   RUNNING THIS UTILITY       *
*****

SOURCE  (NAME,VOLUME): X,TAPE01
TARGET  (NAME,VOLUME): $$EDXVOL,EDX002
DEVICE RESTORE? Y
ARE ALL PARMS CORRECT? (Y,N): Y

USE ATTN/CA TO CANCEL THE RESTORE

DISK RESTORED
COMMAND (?):
```

Restore Disk Device from More than One Tape

COMMAND (?): RT

```
*****  
*   WARNING:  TO ENSURE PROPER   *  
*   DISK CONTENTS, THE SYSTEM   *  
*   SHOULD BE INACTIVE WHILE   *  
*   RUNNING THIS UTILITY       *  
*****
```

```
SOURCE  (NAME,VOLUME): SAVE1,TAPE02  
TARGET  (NAME,VOLUME): $$EDXVOL,EDX002  
DEVICE RESTORE? Y  
ARE ALL PARMS CORRECT? (Y,N): Y
```

USE ATTN/CA TO CANCEL THE RESTORE

```
MOUNT SAVE2,TAPE02  
REPLY Y WHEN TAPE MOUNTED AND VARIED ONLINE?  
> $VARYON 4D  
TAPE02 ONLINE  
? Y  
DISK RESTORED
```

COMMAND (?):

\$TAPEUT1

ST - Save a Disk Device or Disk Volume on Tape

ST saves an entire disk device or a single disk volume on a tape. ST prompts you to specify whether you are saving a device or volume. The ST command can be used in conjunction with the restore command (RT) to backup data you wish to protect.

Examples

Save Disk Volume on Tape

```
> $VARYON 4C
TAPE01 ONLINE
> $L $TAPEUT1
$TAPEUT1 19P,00:06:26, LP= 7A00
TAPE01 DUAL NL 1600 ONLINE
    DEVICE ADDRESS = 004C
TAPE02 SL 1600 OFFLINE
    DEVICE ADDRESS = 004D

COMMAND (?): ST

*****
* WARNING: TO ENSURE PROPER *
* TAPE CONTENTS, THE SYSTEM *
* SHOULD BE INACTIVE WHILE *
* RUNNING THIS UTILITY.    *
*****

SOURCE (NAME,VOLUME): $$EDXVOL,ASMLIB
TARGET (NAME,VOLUME): X,TAPE01
DEVICE SAVE? N
VOLUME SAVE OF ASMLIB ONTO TAPE X,TAPE01
OK? (Y,N): Y

USE ATTN/CA TO CANCEL THE SAVE

VOLUME SAVED
COMMAND (?):
```

Save Disk Device on Tape

```
> $VARYON 4C
TAPE01 ONLINE
> $L $TAPEUT1
$TAPEUT1 19P,00:02:16, LP= 7A00
TAPE01 DUAL NL 1600 ONLINE
    DEVICE ADDRESS = 004C
TAPE02 SL 1600 OFFLINE
    DEVICE ADDRESS = 004D
```

COMMAND (?): ST

```
*****
* WARNING: TO ENSURE PROPER *
* TAPE CONTENTS, THE SYSTEM *
* SHOULD BE INACTIVE WHILE *
* RUNNING THIS UTILITY.    *
*****
```

```
SOURCE (NAME,VOLUME): $$EDXVOL,EDX002
TARGET (NAME,VOLUME): X,TAPE01
DEVICE SAVE? Y
DEVICE SAVE OF DISK CONTAINING EDX002
ONTO TAPE X,TAPE01
OK? (Y,N): Y
```

USE ATTN/CA TO CANCEL THE SAVE

```
DISK SAVED
COMMAND (?):
```


\$TAPEUT1

Save Disk Device on More than One Tape

COMMAND (?): ST

```
*****  
* WARNING: TO ENSURE PROPER *  
* TAPE CONTENTS, THE SYSTEM *  
* SHOULD BE INACTIVE WHILE *  
* RUNNING THIS UTILITY. *  
*****
```

SOURCE (NAME,VOLUME): \$\$EDXVOL,EDX002
TARGET (NAME,VOLUME): SAVE1,TAPE01
DEVICE SAVE? Y
DEVICE SAVE OF DISK CONTAINING EDX002
ONTO TAPE SAVE1,TAPE01
OK? (Y,N): Y

USE ATTN/CA TO CANCEL THE SAVE

END OF TAPE ENCOUNTERED. CONTINUE? Y
ENTER NEXT TAPE (NAME,VOLUME): SAVE1,TAPE01
REPLY Y WHEN TAPE SAVE2,TAPE01 IS MOUNTED
AND VARIED ONLINE?
> \$VARYON 4C
TAPE01 ONLINE
? Y

DISK SAVED
COMMAND (?):

TA - Allocate a Tape Data Set

TA deletes an existing data set and reallocates a null data set, or adds a null data set after the last data set on the volume.

Notes:

1. This command is used to place data set labels on previously initialized standard labeled (SL) tapes; the tape unit must be in the standard label processing mode.
2. All the data on the tape following the newly allocated data set is destroyed.
3. The tape must be varied online to the file number of the data set being allocated.
4. To be accessed by a program, the tape must be varied online.

Example

Allocate Data Set (DATA2222) on Volume 123456

```
COMMAND (?): TA
TAPEDS (NAME,VOLUME): DATA2222,123456
EXPIRATION DATA (YYDDD): 79001
DATA SET ALLOCATED
COMMAND (?):
```

\$TERMUT1

\$TERMUT1 - CHANGE TERMINAL PARAMETERS

\$TERMUT1, a general purpose terminal utility program, alters logical device names, address assignments or terminal configuration parameters. Changes remain in effect until the next IPL.

\$TERMUT1 Commands

The commands available under \$TERMUT1 are listed below. To display this list on your terminal, enter a question mark in response to the prompting message COMMAND(?):.

```
COMMAND(?): ?  
  
LA -- LIST TERMINAL ASSIGNMENTS  
RE -- RENAME  
RA -- REASSIGN ADDRESS  
RH -- REASSIGN HARDCOPY  
CT -- CONFIGURE TERMINAL  
EN -- END PROGRAM  
  
COMMAND(?):
```

After the commands are displayed, you are again prompted with COMMAND(?):. You respond with the command of your choice (for example, LA).

CT - Configure Terminal

CT modifies the page formatting parameters associated with a terminal. In the following example, the conditional prompt message associated with each parameter is shown. Default values are indicated in parentheses.

```
ENTER TERMINAL NAME: $SYSLOG      (loading terminal)
PAGE SIZE: 24                    (from TERMINAL statement)
TOP MARGIN: 12                   (0)
BOTTOM MARGIN: 23                (page size -1)
HISTORY LINES: 6                 (0)
LEFT MARGIN: 0                   (0)
RIGHT MARGIN: 79                 (line size -1)
OVERFLOW LINES? N                (N)
OUTPUT PAUSE? N                  (N)
```

The option OUTPUT PAUSE allows the 'screen full' pause for screen devices to be disabled so that unattended systems do not enter an indefinite wait state.

Note: For more information on terminal parameters, see the TERMINAL statement in the System Guide.

EN - End Program

EN terminates the \$TERMUT1 utility.

\$TERMUT1

LA - List Terminal Assignments

LA lists the current terminal names, addresses and types.

COMMAND(?): LA

NAME	ADDRESS	TYPE
§SYSLOG	04	4979
§SYSPRTR	01	4974
§SYSLOGA	00	TTY
	06	4979
DISPLAY2	07	4978

COMMAND(?):

No name appears for the device at address 06 because there was none on the original TERMINAL statement.

RA - Reassign Address

RA reassigns the physical address of a terminal specified in the address= parameter of the TERMINAL statement during system generation. The form for this function is:

RA name address

where the address in question must be currently unassigned. Some examples are:

RA DISPLAY2 7
RA §SYSPRTR 12
RA (05) 06

RE - Rename Logical Device

RE renames the logical terminal name (the label on the **TERMINAL** statement) that you specified during system generation. The form for this function is:

```
RE oldname newname
```

The new name replaces the old name. As shown in the following examples, the old name can be a logical name or a hexadecimal device address. If a device address is indicated, it must consist of 1 or 2 digits enclosed in parentheses.

```
RE DISPLAY2 DISPLAY3  
RE (06) TERM1
```

The changes are verified by entering the **LA** command.

\$TERMUT1

RH - Reassign Hardcopy

RH changes the hardcopy device associated with a 4979/4978 display, and indicates which program function key will produce the hardcopy. The form is:

RH name keycode

Here 'name' is the logical name (not device address) of the hardcopy device, and 'keycode' is the code for the desired hardcopy key (e.g., 1 to 6 for the 4979 display).

The hardcopy device is changed for the terminal from which \$TERMUT1 was loaded. Some examples are:

**RH \$SYSLOGA 6
RH \$SYSLOGA 4
RH PRTR2 6**

| \$TERMUT2 - PROCESS 4974/4978 IMAGE/CONTROL STORE

\$TERMUT2 is used to:

- Assign a DEFINE key in a 4978 control store.
- Change the definition of one or more keys in a 4978 control store.
- Load a 4978 control store from a direct access data set or saving a newly redefined 4978 control store into a direct access data set. The control store is a data set containing the 4978 control store and microcode to access the image store.
- Load a 4978 image store from a direct access data set or save a 4978 image store into a direct access data set. (Refer to the description of the \$FONT utility program for a description of image store definition.)
- Restore the image buffer of a 4974 printer to the standard 64 character set.

You may wish to invoke these functions from a terminal other than the one you are using; therefore, you are requested to specify a terminal. If the selected terminal is not a 4978, you are notified and the command is rejected.

4978 Support

Use \$TERMUT2 to to make special character string assignments on the 4978 keyboard. Key definitions can be changed, perhaps to be appropriate to a special key data application, and the redefined keyboard definitions saved on disk. The keyboard definition can be reloaded later using \$TERMUT2 or by using the TERMCTRL instruction within your application. 4096 bytes are transferred during an image control save so a 16 record data set must be defined for each control store image to be stored. Use \$FONT to change the display image of redefined keys. For detailed information on the 4978 display station functions and the 4978 keyboards, refer to the Bibliography for 4978-1 Display Station manuals.

\$TERMUT2

4974 Support

Use \$TERMUT2 to restore the image buffer of a 4974 printer to the standard 64 character set. The 4974 printer uses the Extended Binary Coded Decimal Interchange Code (EBCDIC), which includes 64 standard characters and five characters for international use. The standard key definition can be changed by using the TERMCTRL instruction within your application program and the redefined character set is stored in the image buffer of the 4974. For detailed information on the 4974 printer, refer to the Bibliography for the 4974 Printer manual.

Data Set Names

Naming conventions for image store and control store data sets follow the conventions of the Event Driven Executive: the label can be up to eight characters. Two special names are reserved by the system and used during initial program load time:

\$4978ISO	-	image store label
\$4978CS0	-	control store label

These data sets are automatically searched for and loaded to defined 4978 display stations during the initialization phase at IPL time.

\$TERMUT2 Commands

The commands available under \$TERMUT2 are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

COMMAND (?): ?

AD - ASSIGN DEFINE KEY
C - CHANGE KEY DEFINITION
EN - END PROGRAM
LC - LOAD CONTROL STORE
LI - LOAD IMAGE STORE
RE - RESTORE 4974 TO STD. 64 CHAR. SET
SC - SAVE CONTROL STORE
SI - SAVE IMAGE STORE

COMMAND (?):

After the commands are displayed, you are again prompted with COMMAND (?):. You respond with the command of your choice (for example, AD).

\$TERMUT2

Examples

AD - Assign a Define Key

```
COMMAND (?): AD
  ENTER SCAN CODE OF KEY TO BE ASSIGNED
    AS THE DEFINE KEY (HEX): 1
  ENTER TERMINAL NAME (CR OR * = THIS ONE): $SYSLOG
```

C - Change a Key Definition

```
COMMAND (?): C
  ENTER TERMINAL NAME (CR OR * = THIS ONE): $SYSLOG
  ENTER SCAN CODE OF THE KEY TO BE REDEFINED (HEX): 1
  ENTER NEW FUNCTION ID (HEX): 20
  ENTER NEW CHARACTER/FUNCTION CODE (HEX): 0
  ENTER NEW INTERRUPT CODE (HEX): 1
  ANOTHER KEY? N
```

LC - Load a Control Store

```
COMMAND (?): LC
  FROM DATA SET (NAME,VOLUME): $4978CS0
  ENTER TERMINAL NAME (CR OR * = THIS ONE): *
```

RE - Restore 4974 to Standard 64 Character Set

```
COMMAND (?): RE
ENTER TERMINAL NAME (CR OR * = THIS ONE): MPRINTER
```

SC - Save a Control Store

```
COMMAND (?): SC
SAVE DATA SET (NAME,VOLUME): $4978CS0
ENTER TERMINAL NAME (CR OR * = THIS ONE): $SYSLOG
```

\$TERMUT3

\$TERMUT3 - SEND MESSAGE TO A TERMINAL

\$TERMUT3 sends a single line message from your terminal to any other terminal defined in the system. One message line is sent at a time and you are prompted for the message, the terminal the message is to be sent to, and if additional messages are to be sent. There are no commands, only prompting messages as follows:

ENTER TERMINAL NAME:

Label of the terminal to which message is to go.

ENTER MESSAGE TO SEND

Type in message and press the ENTER (CR) key.
The message is transmitted to the destination terminal.

ANOTHER LINE?

Yes or No.

If yes, you are prompted to enter another message line.

ANOTHER TERMINAL?

Yes or No. If yes, select another terminal and repeat the above process.

SEND MESSAGE LATER ('ATTN SEND')?

Yes or No. If yes, the program remains loaded and inactive. Pressing the ATTN key and entering 'SEND' starts the above process again.

Examples

Send a Message to Terminal \$SYSLOGA

```
$TERMUT3      3P,00:41:10, LP= 8800  
  
TERMINAL BROADCAST PROGRAM  
  
ENTER TERMINAL NAME: $SYSLOGA  
ENTER MESSAGE TO SEND  
THIS IS A TEST MESSAGE.  
  
ANOTHER MESSAGE ? Y  
ENTER MESSAGE TO SEND  
THIS IS ANOTHER TEST MESSAGE.  
  
ANOTHER MESSAGE ? N  
  
ANOTHER TERMINAL ? N  
  
DETACH PGM (WAIT FOR 'ATTN SEND') ? N  
  
$TERMUT3 ENDED AT 00:43:05
```

\$TERMUT3

Select Another Terminal After Sending a Message

.
.
.
ANOTHER LINE ? N

ANOTHER TERMINAL ? Y
ENTER TERMINAL NAME: \$SYSPRTR
ENTER MESSAGE TO SEND
HELLO \$SYSPRTR

ANOTHER LINE ? N

ANOTHER TERMINAL ? N

SEND MESSAGE LATER ('ATTN SEND') ? N

\$TERMUT3 ENDED AT 00:54:00

Keep \$TERMUT3 Active, Send a Message Later

```
.  
. .  
ANOTHER LINE ? N  
ANOTHER TERMINAL ? N  
SEND MESSAGE LATER ('ATTN SEND') ? Y  
  
> SEND  
ENTER TERMINAL NAME: TTY30  
ENTER MESSAGE TO SEND  
TTY30 - ARE YOU THERE  
  
ANOTHER LINE ? N  
ANOTHER TERMINAL ? N  
SEND MESSAGE LATER ('ATTN SEND') ? N  
  
$TERMUT3 ENDED AT 01:42:15
```


\$TRAP

\$TRAP - SAVE STORAGE ON ERROR CONDITION

\$TRAP intercepts certain class interrupts and saves the contents of hardware registers and processor storage on a disk or diskette data set. This utility helps you to diagnose system or application program problems. The companion utility, \$DUMP, prepares a formatted display of the data saved by \$TRAP.

\$TRAP must be executed before an expected failure so that the requested class interrupts can be intercepted when they occur. It can also be activated if a class interrupt such as a program check or machine check do not occur.

Two methods are provided to accomplished forced traps:

- ATTN: TRAPDUMP
- The console interrupt button .

\$TRAP does not affect the execution of any program or operator command unless it is activated by a class interrupt such as a program check or a machine check, or by the operator.

Example

```

> $L $TRAP
DUMPDS(NAME,VOLUME): DUMP,EDX003
$TRAP 15P,12:16:42, LP8F00
Notes
(1)

*-* WARNING *-*-*-* WARNING *-*-*-* WARNING *-*

TO ACTIVATE TRAP USE ATTN: TRAPON (2)
TO DEACTIVATE TRAP USE ATTN: TRAPOFF (2)
TO FORCE TRAP USE ATTN: TRAPDUMP (2)
TO END $TRAP USE ATTN: TRAPEND (3)
DO NOT USE $C TO END $TRAP (3)
IF TRAP OCCURS SYSTEM MUST BE IPLED (4)
TRAP ON MACHINE CHECK? (Y/N) Y (5)
TRAP ON PROGRAM CHECK? (Y/N) Y (6)
SPECIFICATION CHECK? (Y/N) Y
INVALID STORAGE ADDRESS? (Y/N) Y
PRIVILEGE VIOLATE? (Y/N) Y
PROTECT CHECK? (Y/N) Y
INVALID FUNCTION? (Y/N) Y
TRAP ON SOFT EXCEPTION? (Y/N) Y (7)
INVALID FUNCTION? (Y/N) Y
FLOATING POINT EXCEPTION? (Y/N) Y
STACK EXCEPTION? (Y/N) Y
TRAP ON CONSOLE INTERRUPT? (Y/N) Y (8)
SAVE FLOATING POINT REGS? (Y/N) Y (9)

TRAP SET OFF
READY FOR ATTN: X---X (10)

> TRAPON (11)
TRAP SET ON
> TRAPOFF (11)
TRAP SET OFF

```

Notes:

1. The \$TRAP data set must be as large as the storage used: for a 64K-byte system, it must be at least 256 records; for a 128K-byte system at least 512 records.
2. Trap initiation is via the attention handler. TRAPON activates the trap facility but does not produce a storage dump. You must activate the trap facility (TRAPON) first; then specify TRAPDUMP to force a storage dump. If TRAPDUMP

\$TRAP

is used, IPL to resume operations.

3. \$TRAP should not be cancelled using \$C. Use TRAPEND.
4. During trap processing, all I/O activities are halted. All configured devices are reset. IPL to resume operations.
5. A 'Y' response saves storage and hardware registers when any machine check (storage parity, CPU control or I/O error check) occurs.
6. A 'Y' response prompts you for the type of program check to trap. Any combination of the five types can be selected.
7. A 'Y' response prompts you for the type of soft exception to trap. Any combination can be selected.
8. It may be desirable to save storage even though no hardware detectable error has occurred. This option can be used to cause a trap when the console interrupt button is pressed. (Programmer console must be installed on Series/1)
9. Respond with 'Y' if the Series/1 has the floating point hardware installed.
10. \$TRAP initializes to a TRAPOFF state. Any time after this message you can activate trap using the \$TRAP attention commands. See notes 2-4.
11. These are the TRAPON and TRAPOFF attention commands.

CHAPTER 5. APPLICATION PROGRAM PREPARATION

Application program preparation consists of four major operations:

1. Entering the program source statements onto disk or diskette.
2. Compiling or assembling the source program into an object module.
3. Link editing two or more object modules together to form the final object module.
4. Converting the object module into an executable relocatable load module which can then be loaded and executed.

Entering Source Statements

Entering program source statements for assembly or compilation on the Series/1 is normally performed using either of the editor utilities, \$EDIT1N or \$FSEDIT. (For descriptions of the text editors, refer to "Chapter 4. The Utilities" on page 47.

When entering source programs which will be assembled by the System/370 host assembler, \$EDIT1 or \$FSEDIT can be used to enter the program and store it in a host data set. This operation requires that IUP 5796-PGH, Event Driven Executive Host Communication Facility, be installed on the host computer. Alternatively, a source program entered using \$EDIT1N or \$FSEDIT can be transmitted to the host for assembly using utility programs \$RJE2780 or \$RJE3780 when the host computer supports the IBM 2780 or 3780 RJE stations.

Programs to be assembled on the host System/370 can also be entered into the System/370 as either punched card decks or by any type of terminal input supported by the host system.

Application Program Preparation

Program Assembly/Compilation

Program assembly or compilation can be performed by several assemblers and compilers each with some restrictions on their capabilities. They include:

- The Event Driven Language compiler, \$EDXASM, from the Event Driven Executive Program Preparation Facility
- The Series/1 Macro Assembler (\$S1ASM)
- The Macro Assembler supplied by the System/370 Program Preparation Facility
- The COBOL compiler
- The FORTRAN compiler
- The PL/I compiler

Some restrictions that apply to the use of the two assembler programs (\$S1ASM and Host Preparation) and the \$EDXASM compiler are discussed in more detail in the chapters on each of them. The \$EDXASM compiler and the \$S1ASM assembler can be executed concurrently with other programs in an Event Driven Executive based system. Both the Series/1 and Host assemblers are macro assemblers which permit the assembly of both Event Driven Language instructions and Series/1 assembler language instructions. \$EDXASM provides for the compilation of Event Driven Language instructions only. Use of the host assembler requires installation of the Event Driven Executive Macro Library/Host. Assembly of Series/1 assembler instructions and Event Driven Language instructions using \$S1ASM requires the Event Driven Executive Macro Library.

Object modules produced by the host assembler must be transmitted to Series/1 disk or diskette for link editing and conversion into executable programs. Possible means of transmission include the use of utility programs \$HCFUT1, \$RJE2780, or \$RJE3780. If the host assembly is a complete main program, it can be transmitted and converted in one step by using the utility program \$UPDATEH. Use of \$HCFUT1 or \$UPDATEH requires that the Host Communications Facility (IUP 5796-PGH) be installed on the host computer.

Minor differences exist between the format of source programs to be compiled by \$EDXASM and either of the macro assemblers. When using \$EDXASM, you must explicitly code any ECB and QCB statements that are required. Also, \$EDXASM provides the COPY statement function in a different manner than the host assembler, or the limited manner available with the Event Driven

Executive Macro Library as a macro instruction. \$EDXASM permits labels in programs to be composed of any printable characters with a maximum label length of eight characters.

Linkage Editor

Link editing object modules together is an optional program preparation step except when the Series/1 Macro Assembler (\$SIASM) is used. In that case, the output object module of \$SIASM must always be link edited (\$LINK). If your program is compiled or assembled as a single module it is not necessary to use the linkage editor, \$LINK, in the program preparation process (except when \$SIASM is used). However, when creating large programs it is frequently desirable to segment the program into functional modules. The modules can then be compiled or assembled separately, as they are created, and then linked together to form the final program. In this manner, a change in one module requires only recompiling or reassembling the modified section followed by a link editing with the unchanged modules. Object modules that are input to \$LINK can be created by \$EDXASM, \$SIASM, COBOL, FORTRAN, PL/I, or the Host Assembler as previously described.

Object Module Conversion

The program modules which are the output of the various compiler or assembler programs or \$LINK are not in the format that is required by the program loading function of the system supervisor. The utility programs \$UPDATE or \$UPDATEH must be used to convert an object module into loadable form and to update the directory of the volume on which the loadable program is stored.

Prefind of Data Sets and Overlays

Each time your program is loaded into storage for execution, the location of the data sets and overlays it uses must be determined by the loader. \$PREFIND can be used to perform the data set and overlay location function before program load time. The result is a faster load operation. Use of \$PREFIND is optional.

Application Program Preparation

Caution must be exercised in using \$PREFIND. See "\$PREFIND Usage Cautions" on page 303.

Summary

The following sections describe in more detail the use of the various assemblers and compilers and the operation of \$LINK, \$UPDATE, \$UPDATEH, and \$PREFIND. See "Chapter 4. The Utilities" on page 47 for descriptions of the editing utilities \$EDIT1, \$EDIT1N, and \$FSEDIT (they are not used exclusively for source program entry).

The programs \$EDXASM, \$LINK, \$PREFIND, \$UPDATE, and \$UPDATEH can be invoked individually from a terminal using either the session manager or the \$L function. They can also be invoked as part of a batch job stream operation under the control of \$JOBUTIL utility. For a description of \$JOBUTIL operations, see "\$JOBUTIL - Job Stream Processor" on page 271.

Figure 22 on page 355 shows the program preparation process of assembling or compiling, link editing and updating. The programs can be invoked individually by the session manager or through the batch job stream processing utility. An example of preparing an application program can be found in the System Guide.

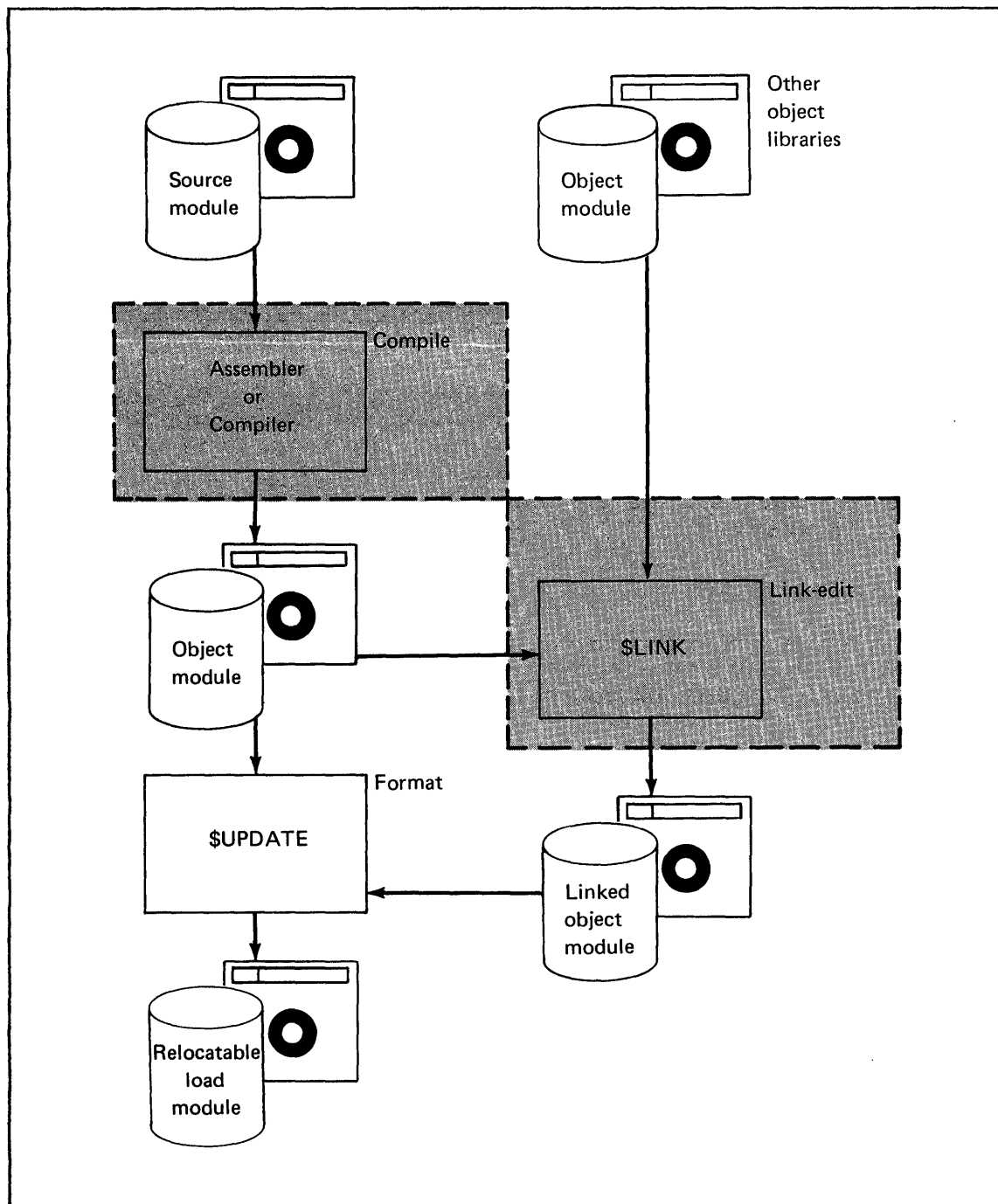


Figure 22. Event Driven Executive program preparation

\$EDXASM

\$EDXASM - EVENT DRIVEN LANGUAGE COMPILER

\$EDXASM provides an online program compilation capability that can be executed concurrently with other programs in an Event Driven Executive based system. Multiple copies of \$EDXASM can operate concurrently.

\$EDXASM is composed of a main program plus a large number of overlay programs and, therefore, makes extensive use of the program loading facilities of the supervisor. If during execution of \$EDXASM, you invoke other programs by means of the \$L function, the execution of \$EDXASM may be delayed until all information needed by the \$L command (e.g. data set names) has been entered. This delay is because the program load facility is a serially usable function and must complete one program load request before starting another. It is good practice to enter as much of the required information as possible on the same line of input as the \$L command to minimize the time that the loader is busy.

Example

```
> $L PROGRAM,VOLUME DATASET1,VOLUME DATASET2,VOLUME
```

Note: Volume parameters are optional if the IPL volume is used.

The description of \$EDXASM assumes that you have already entered the source program to be compiled into a disk or diskette data set by means of one of the editor utilities, \$EDIT1N or \$FSEEDIT.

\$EDXASM can be invoked either by the \$L command, by the \$JOBUTIL utility, or by the program preparation utilities option selection menu operating under the Session Manager. In each case, the same information must be provided when \$EDXASM is loaded for execution. Invoking \$EDXASM using \$JOBUTIL is shown under "Invoking \$EDXASM Using \$JOBUTIL" on page 368. Invoking \$EDXASM using the Session Manager is shown under "Invoking \$EDXASM Using the Session Manager" on page 369.

Language Control Data Set

\$EDXASM uses a language control data set, \$EDXL. The data set is divided into two logical parts: the error messages and the operation codes to process module specifications. This data set is produced by \$EDIT1N. As distributed, \$EDXL in the volume ASMLIB contains the standard compiler error messages and Event Driven Language instruction set specifications. You may wish to add COPY code definitions or additional processing modules and error messages and may even desire to have differently modified versions assigned to different users. The contents of \$EDXL are described in the Internal Design. The use of the *COPYCOD function within \$EDXL is described under "COPY Statements" on page 362.

Required Data Sets

\$EDXASM requires three data sets. These data sets, in order of specification when \$EDXASM is invoked using \$L are:

1. The input, or source, data set. The format of this data set is the same as produced by \$EDIT1N/\$FSEEDIT.
2. The work data set. This data set contains object code, relocation pointers and the symbol table. A minimum size of 100 records is recommended; 250 records might be considered average and 500 records would be large. This data set is automatically allocated by the Session Manager to be 400 records.
3. The object data set. This data set contains the output object module from the compilation and serves as input to \$UPDATE (or to \$LINK, if a link edit is required). The appropriate size for the object module data set is also dependent on the program size. For estimating purposes, divide the program length in bytes by 100 to get the number of records required.

\$EDXASM

Sample printouts showing the prompt messages and replies follow:

```
> $L $EDXASM,ASMLIB
SOURCE (NAME,VOLUME): ASMSRC
WORKFILE(NAME,VOLUME): ASMWORK
OBJECT (NAME,VOLUME): ASMOBJ
$EDXASM 64P,02:48:50, LP= 6300

SELECT OPTIONS (?): ?
```

The program name and data sets required can be entered in the same line. For example:

```
> $L $EDXASM,ASMLIB ASMSRC ASMWORK ASMOBJ
```

This single line entry is equivalent to the multiline entries above.

Compiler Options

When the compiler is loaded into storage, enter the options to be used. The option list follows:

```
SELECT OPTIONS (?): ?

LIST      - SPECIFY LIST DEVICE
NOLIST    - DO NOT PRINT LISTING
ERRORS    - LIST ERRORS ONLY
CONTROL   - SPECIFY CONTROL LANGUAGE
END       - END OPTION SELECTION
('ATTN - CA' TO CANCEL ASSEMBLY)
```

If no options are selected, because you entered only a carriage return/ENTER in response to the select option message, the defaults are LIST on \$SYSPRTR using the language control data set \$EDXL on the volume ASMLIB. If a listing is required on another device, specify LIST or L. You can enter the name of the device in response to the prompt for device name or on the same line. A null entry or an * is used to specify your terminal.

If no listing is required, specify NOLIST or N. In this case, the compile statistics are printed on the same terminal as \$EDXASM was loaded from. If only statements containing errors are to be printed, specify ERRORS or ER. In this case, a device name is required as in LIST. Similarly, a null entry or asterisk (*) indicates that the error messages are to be listed on the loading terminal. This option is very useful for the first few compilations to remove typographical or simple syntactical errors from the source program.

If a control data set other than \$EDXL on the volume ASMLIB is required, enter CONTROL followed by the name and volume of the data set to be used. All option entries can be entered on a single line or in response to prompt messages. The last entered listing option takes precedence. The compilation or the subsequent listing can be cancelled at any time by pressing ATTN and entering CA.

\$EDXASM Output

When the compilation process is complete, the compiler prints out statistics. They indicate the source, work, and object data sets used, the date and time the compilation started, the elapsed time for the compilation, the number of statements processed, the number of statements flagged with error messages, and the compilation completion code. A completion code of -1 is normal. At this time, the object module is stored and is ready for input to \$UPDATE or \$LINK. If a listing has been requested, it is then printed on the appropriate output device. The printing of duplicate lines of object code is automatically suppressed by the listing routine of \$EDXASM. Before the data sets specified in the compilation are reused, it is possible to request a listing of the compilation using the program \$EDXLIST. Refer to "\$EDXLIST - Compiler Listing Program" on page 370 for more information.

\$EDXASM

Examples

```
LIST on $SYSPRTR:
```

```
SELECT OPTIONS (?): null entry
```

```
ERRORS on PRINTER1:
```

```
SELECT OPTIONS (?): ERRORS
```

```
  DEVICE NAME: PRINTER1
```

```
SELECT OPTIONS: END
```

```
  or
```

```
SELECT OPTIONS: ER PRINTER1 END
```

```
NOLIST and use $EDXL on EDX002:
```

```
SELECT OPTIONS (?): N CONTROL
```

```
CONTROL(NAME,VOLUME): $EDXL,EDX002
```

```
SELECT OPTIONS (?): END
```

Processing Compiler Output with \$UPDATE or \$LINK

The output object module has been completed before the listing is started; therefore, the object module can be processed by \$LINK and/or \$UPDATE while the listing is being produced. The operation of \$UPDATE is described under "\$UPDATE - Object Program Converter" on page 408. The operation of \$LINK is described under "\$LINK - Linkage Editor" on page 390.

Compiler Features Supported

Print Control Supported:

EJECT
SPACE
SPACE n 1 < n < 25
PRINT OFF/ON
PRINT DATA/NODATA

PRINT NODATA suppresses the printing of all lines of object code which are not on the same line as a source statement.

PRINT DATA is equivalent to PRINT ON.

Print Control Unsupported:

TITLE
PRINT GEN/NOGEN

Establishing Symbolic Representation:

EQU
CSECT
ENTRY
EXTRN
WXTRN

Programming Considerations

Source Line Continuation

Continuation of source lines is indicated by placing any character in position 72 of the line to be continued. If the line to be continued contains a blank prior to position 71, then any further information on that line is ignored. Continuation lines must begin in position 16. The data in positions 16 up to 71 (or the first blank) is concatenated to the data from the preceding line.

\$EDXASM

Recommended practice is to either code the operand fields through position 71 of a line to be continued, or to terminate the line by placing a blank after the comma which terminates one operand in the string of operands.

The maximum number of continuation lines is limited only by the maximum of 254 characters in the operand field.

COPY Statements

COPY statements can only be used in the first level of source code. They cannot be used within COPY code.

\$EDXL contains two *COPYCOD statements, *COPYCOD ASMLIB and *COPYCOD EDX002. This indicates to \$EDXASM that volumes ASMLIB and EDX002 are to be searched for the special source module each time a COPY statement appears in a source program. If your COPY modules are not in either of the above volumes, you must add an *COPYCOD statement to \$EDXL for each additional volume which contains COPY modules.

To define your COPY code to the compiler:

1. Create your COPY code source statements by using \$EDITIN or \$FSEDIT
2. Add *COPYCOD statements (if required) to \$EDXL (or your \$EDXL special copy), with \$EDITIN or \$FSEDIT.

The format of the *COPYCOD statements is:

	*COPYCOD	volume
Column	1	10

The libraries are searched in the order in which the *COPYCOD statements appear in \$EDXL.

ECB and QCB

Certain conditions apply when any of the following are coded:

```
LOAD    ...,EVENT=ecbname
WAIT/POST      ecbname
ENQ/DEQ       qcbname
```

The ECB or QCB is not automatically created but must be explicitly specified by you. If EVENT=ecbname is specified in a PROGRAM or TASK statement, do not code an ECB statement; otherwise, duplicate labels will result.

Multiple Declarations on DC and DATA statements

When implementing DC (or DATA) statements, up to 10 declarations can be made on a single DC statement. A sample statement follows:

```
DC  F'5',A(TEST),CL5'ABC'
```

The following DC types are supported:

SEDXASM

F'x',mF'x' where 'm' is the number of times the data is to be duplicated
A(y),mA(y)
X'z',mX'z' where 'z' can be from 1-8 hexadecimal digits.
H's',mH's'
C't',mC't',CLn't',mCLn't'
where 'n' is the length of the data in bytes
D'w',mD'w'
E'u',mE'u'
L'v',mL'v'

IODEF Statement Placement

IODEF statements must appear after the PROGRAM statement and before the ENDPROG statement. Also, all IODEF statements of the same type (e.g., PI, DI, DO, AI, AO) must appear together in a group. For example, if a program requires two DI and three DO definitions, the statements in the program would be coded as follows:

```
IODEF  DI1,...  
IODEF  DI2,...  
IODEF  DO1,...  
IODEF  DO2,...  
IODEF  DO3,...
```

In other words, the DI and DO definitions must be grouped together.

GETEDIT and PUTEDIT

The instructions GETEDIT and PUTEDIT cannot have the format statement included in the instruction. A separately coded FORMAT statement must be referenced by the GETEDIT or PUTEDIT instruction.

ATTNLIST Statement

The ATTNLIST statement can have only one list coded. The list can be up to 254 characters in length and can contain a total of 50 suboperands or 25 ATTNLIST entries.

EQU Statement

Labels referenced in EQU statements must have been previously defined.

Arithmetic Expression Operators

Only one operator is allowed in an arithmetic expression. Valid operators are +, -, *, and /. For example, the following expressions are valid: A+B, C-1, D*4, E/2. If an expression containing more than one term is required, multiple equate statements can be coded. For example, to compute the address A+B-2, the following statements can be coded:

APB	EQU	A+B
APBM2	EQU	APB-2

Instructions Requiring Support Modules

Certain instructions require support modules link edited into the executable load module. These modules are referenced in

\$EDXASM

the linkage editor autocall data set \$AUTO. Therefore, the modules are automatically included in your program by running the link editor on the object module output of \$EDXASM. Use of the following instructions requires a link edit.

Formatting instructions

```
GETEDIT
PUTEDIT
FORMAT
```

Graphics instructions

```
CONCAT
GIN
PLOTGIN
SCREEN
XYPLOT
YTPLOT
```

Square root instruction

```
SQRT
```

Series/1 Macro Assembler interface instruction

```
USER
```

The supervisor interface module \$\$RETURN must be link-edited with any module that includes the USER statement.

A link edit is also required if any of the following screen formatting subroutines are called in an application program:

```
$IMOPEN  
$IMGEN  
$IMDEFN  
$IMPROT  
$IMDATA  
$PACK  
$UNPACK
```

The supervisor interface module \$\$RETURN must be link-edited with \$IMOPEN and \$IMGEN if they are called in an application program.

See the description of "\$LINK - Linkage Editor" on page 390 for information on using the linkage editor.

\$EDXASM

Invoking \$EDXASM

Invoking \$EDXASM Using \$JOBUTIL

\$EDXASM can be invoked using \$JOBUTIL. The same options are available through the PARM facility of the job stream processor (\$JOBUTIL) as were described previously under the \$L command. The listing option is specified in column 10, the device name in column 20, and the control data set name and volume in column 40. If the default options are required, the PARM statement can be blank but must be included in the procedure. A sample procedure is shown below:

```
LOG          PRINTER1
PROGRAM     $EDXASM,ASMLIB
DS          ASMSRC
DS          ASMWORK,EDX003
DS          ASM OBJ
PARM        LIST          PRINTER1
NOMSG
EXEC
```

Invoking \$EDXASM Using the Session Manager

\$EDXASM can be invoked using the session manager by selecting option 1 from the program preparation secondary option menu.

The following parameter selection menu is displayed for entry of the required data sets and optional parameters. The control data set is assumed to be \$EDXL on ASMLIB. The example shows the data set ASMSRC is to be compiled with the object output being directed to the data set ASMOBJ.

```
$SMM0201: SESSION MANAGER $EDXASM PARAMETER INPUT MENU -
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN
```

```
SOURCE INPUT (NAME,VOLUME) ==> ASMSRC,EDX002
```

```
OBJECT OUTPUT (NAME,VOLUME) ==> ASMOBJ,EDX002
```

```
ENTER OPTIONAL PARAMETERS BY POSITION ==> NOLIST
```

```
1-----
LIST
NOLIST
ERRORS
```

```
LPRINTER
2-----
PRINTER
NAME
```

```
DEFAULTS ARE:
LIST
$SYSPRTR
```

`SEDXASM`

`SEDXLIST` - Compiler Listing Program

`SEDXLIST` is used to obtain a listing of the last `SEDXASM` compilation performed.

You can specify `LIST` or `NOLIST` in response to the `SELECT OPTIONS (?)`: prompt when `SEDXASM` is loaded. If you respond with the `ENTER` key or `LIST`, `SEDXASM` automatically loads `SEDXLIST` as part of the compilation step and produces a listing of the compilation. If you respond with `NOLIST`, the statistics from the compilation are displayed on the loading terminal and a listing is not produced.

`SEDXLIST` can be loaded as a separate program. For example, if `NOLIST` were selected, and the statistics displayed at the end of the compilation indicate that there are compilation errors, `SEDXLIST` can then be loaded to print a listing.

`SEDXLIST` will prompt for the source data set and the work data set and will get the name of the language control data set from the work data set, in which it is stored, at the end of the compilation. As long as an intervening compilation has not altered the contents of the work data set, and you have not modified the source or language control data sets, `SEDXLIST` will produce the same listing when loaded by you after a compilation as it would were it loaded by `SEDXASM` as part of the compilation step.

Invoking `SEDXLIST`

`SEDXLIST` is invoked using `$L`. Two data set names are required: the source (`SOURCE`) and work (`WORKFILE`) data set used for compilation. The name of the control data set used for the compilation is kept in the work data set. If `SEDXASM` was invoked by the session manager, the work data set will be named `$SM1user` (where `user` is the sign on ID used).

Caution: If you want a listing of the latest compilation, use `SEDXLIST` before invoking `SEDXASM` again. Any subsequent compilation modifies the contents of the source and work data sets.

An example using `SEDXLIST` follows:

```
> $L $EDXLIST,ASMLIB
SOURCE (NAME,VOLUME): ASMSRC,EDX002
WORKFILE(NAME,VOLUME): ASMWORK,EDX002
$EDXLIST      21P,00:07:49, LP= 6500

SELECT OPTIONS (?): ?

LIST      - SPECIFY LIST DEVICE
ERRORS   - LIST ERRORS ONLY
END       - END OPTION SELECTION
          ('ATTN - CA' TO CANCEL LISTING)

SELECT OPTIONS (?): LIST
          DEVICE NAME: $SYSPRTR

SELECT OPTIONS (?): END

$EDXLIST ENDED AT 00:08:46
```


\$\$SIASM - SERIES/1 ASSEMBLER

\$\$SIASM provides an online program assembly capability that can be executed concurrently with other programs in an Event Driven Executive system. \$\$SIASM produces object modules from source programs written in Series/1 assembler language or Series/1 assembler language macros such as the commands provided by the Event Driven Executive Macro Library. Multiple copies of \$\$SIASM can operate concurrently if each copy has its own work files and printer. \$\$SIASM is composed of a main program and a number of overlays.

The following description of \$\$SIASM assumes that you have entered the source program to be assembled into a disk or diskette data set. This is accomplished by using one of the editor utility programs, \$EDITIN or \$FSEDIT. \$\$SIASM can be invoked by:

- The Program Preparation utilities option selection menu operating under the session manager
- The \$L operator command
- The batch job stream processing utility, \$JOBUTIL.

Required Data Sets

\$L and \$JOBUTIL require that the following information be provided when \$\$SIASM is loaded for execution.

1. The source input data set.
2. Three work data sets, used as work files, containing object code, relocation pointers, the symbol table, and other information. For most programs, sizes of 2000, 2000, and 800 for ASMWRK1, ASMWRK2, and ASMWRK3, respectively, are sufficient. This requirement can increase when assembling a large program containing many macro calls.
3. The object data set. This data set will contain the output object module. It serves as input to the linkage editor (\$LINK) which must always be run when using \$\$SIASM. A size of 50 to 100 sectors is sufficient for most programs.

The Session Manager provides the work data sets and requires only that the input and output data sets be specified.

Invoking \$\$1ASMInvoking \$\$1ASM using \$L

For illustrative purposes, let us assume that:

- EDX002 is the IPL volume
- Data sets ASMSRC and ASMOBJ are on that volume
- EDX003 contains the ASMWORK1, ASMWORK2, and ASMWORK3 data sets
- PRNTR1 is the name of a print device.

Specify Data Sets (Prompt/Reply Mode)

A sample printout showing the prompt/reply format of \$L to invoke \$\$1ASM and specifying these data sets follows:

```
> $L $$1ASM,ASMLIB
SOURCE (NAME,VOLUME): ASMSRC
WORK1 (NAME,VOLUME): ASMWORK1,EDX003
WORK2 (NAME,VOLUME): ASMWORK2,EDX003
WORK3 (NAME,VOLUME): ASMWORK3,EDX003
OBJECT (NAME,VOLUME): ASMOBJ

MACLIB1 (?):
```

Specify Data Sets (Single Line Mode)

A sample printout showing the single line format of \$L to invoke \$\$1ASM and specifying these data sets follows:

\$\$1ASM

```
> $L $$1ASM,ASMLIB ASMSRC ASMWORK1,EDX003 ASMWORK2,EDX003  
ASMWORK3,EDX003 ASMOBJ
```

```
MACLIB1 (?):
```

Specify Macro Libraries

In both of the previously shown modes, you are prompted as follows:

```
MACLIB1 (?):
```

If you require macros for this assembly, you can specify one or two macro library volumes. A null response (ENTER) takes you to the next prompt (SELECT OPTIONS). If a MACLIB1 is specified (as shown in the following example), another prompt (MACLIB2) appears:

```
MACLIB1 (?): MACLIB1
```

```
MACLIB2 (?):
```

You can supply the name of another macro library or enter a null response.

Note: If your source program contains Event Driven Language instructions, you must specify, as either MACLIB1 or MACLIB2, the name of the volume(s) that contains the Event Driven Executive Macro Library (5719-LM5) and copy code.

Specify Assembly Options

After entering the macro library volume name(s) and/or a null response, you are prompted as follows:

ENTER OPTIONS (?):

Following is a list of possible replies:

- Enter a null response and take the default options (LIST, OBJECT, MACRO)
- Enter the options you desire (for example, LIST, NOXREF, NORLD). See "List of Options" on page 378.
- Enter a question mark to display a list of the available options. After the options are displayed, the ENTER OPTIONS (?): prompt is displayed.
- Enter the options of your choice, followed by the name of your output device. This prevents the next prompt from appearing.

If you do not specify the name of the output device with your options, as follows:

ENTER OPTIONS (?): LIST,NOXREF,NORLD

\$\$1ASM

You are prompted with:

ENTER OUTPUT DEVICE NAME:

You can enter the name of the device on which your listings and diagnostic messages are to be written. If you do not specify an output device (a null response), it defaults to \$SYSPRTR.

The next message displayed is:

CPA000I ASSEMBLER STARTED

Complete Example:

A complete example follows using:

- ASMSRC for the source library
- ASMWORK1, ASMWORK2, and ASMWORK3 for work files
- ASM OBJ for the output file
- MACLIB1 for the macro library
- PRNTR1 for the output listing.

Invoke \$SIASM and Specify Data Sets to Use

```
> $L $$SIASM,ASMLIB  
  
SOURCE (NAME,VOLUME): ASMSRC  
WORK1 (NAME,VOLUME): ASMWORK1,EDX003  
WORK2 (NAME,VOLUME): ASMWORK2,EDX003  
WORK3 (NAME,VOLUME): ASMWORK3,EDX003  
OBJECT (NAME,VOLUME): ASM OBJ  
  
MACLIB1 (?): MACLIB1  
  
MACLIB2 (?):  
  
ENTER OPTIONS (?): ?
```

\$\$IASM

List of Options

If you enter a question mark in response to the prompt ENTER OPTIONS, the following list of options is displayed:

```
VALID OPTIONS ARE:
LIST/NOLIST - COMPLETE ASM LISTING/
                ERRORS ONLY
TEXT/NOTEXT - SOURCE AND OBJECT LISTING/
ESD/NOESD   - LIST EXTERNAL SYMBOL DICTIONARY/
                SUPPRESS THIS OPTION
RLD/NORLD   - LIST RELOCATION DICTIONARY/
                SUPPRESS THIS OPTION
XREF/NOXREF/FULLXREF - LIST REFERENCED SYMBOLS/
                NO XREF LISTING/
                LIST ALL SYMBOLS
OBJECT/NOOBJECT - WRITE OBJECT TO OBJECT FILE/
                SUPPRESS THIS OPTION
MACRO/NOMACRO --- PROCESS MACROS IN SOURCE/
                DO NOT PROCESS MACROS IN SOURCE
SYSPARM('...') -- SUBSTITUTION STRING FOR MACRO PROCESSIN
LINECOUNT(N) ---- LINE/PAGE FOR ASM LIST
                DEFAULT N = 55 LINES/PAGE
SETC(N) ----- NUMBER OF CHARACTERS ASSIGNED TO
                SETC SYMBOLS
END ----- TERMINATE OPTIONS PROCESSING
CA ----- TO CANCEL ASSEMBLY

ENTER OPTION(S) SEPARATED BY COMMAS
DEFAULT OPTIONS ARE: "LIST,OBJECT,MACRO"
```

Select processing options

You now enter the options of your choice and are prompted as follows. A message is displayed indicating that the assembler has started.

```

ENTER OPTIONS (?): LIST,NOXREF,NORLD
ENTER OUTPUT DEVICE NAME: PRNTR1
CPA000I ASSEMBLER STARTED
    
```

Single line format

Combining your inputs, using the single line format, the previous example looks like this:

```

> $L $$1ASM,ASMLIB ASMSRC ASMWORK1,EDX003 ASMWORK2,EDX003
ASMWORK3,EDX003 ASMOBJ
MACLIB1 (?): MACLB1
MACLIB2 (?):
ENTER OPTIONS (?): LIST,NOXREF,NORLD PRNTR1
CPA000I ASSEMBLER STARTED
    
```


\$SIASM

Invoking \$SIASM Using \$JOBUTIL

\$SIASM can be invoked using \$JOBUTIL. The same options are available through the PARM facility of the job stream processor (\$JOBUTIL) as were described previously under the \$L command. If the default options are required, the PARM card can be blank but must be included in the procedure. A sample procedure that parallels the previous example using \$L follows:

```
PROGRAM $SIASM,ASMLIB
DS      ASMSRC
DS      ASMWORK1,EDX003
DS      ASMWORK2,EDX003
DS      ASMWORK3,EDX003
DS      ASMOBJ
PARM    MACLB1          PRNTR1    LIST,NOXREF,NORLD
EXEC
```

Note: Parameters of the PARM statement in the preceding sample are column dependent:

MACLB1 must start in column 10
MACLB2, if coded, must start in column 20
PRNTR1 must start in column 30
The option list must start in column 40

Invoking \$SIASM Using the Session Manager

To invoke \$SIASM using the session manager, select option 2 from the program preparation secondary option menu.

The following parameter selection menu is displayed for entry of the required data sets and optional parameters. The example shows the data set ASMSRC is to be assembled with the object output being directed to the data set ASM OBJ.

```

$SMM0202: SESSION MANAGER $SIASM PARAMETER INPUT MENU ---
ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN

SOURCE INPUT (NAME,VOLUME) ==> ASMSRC,EDX002

OBJECT OUTPUT (NAME,VOLUME) ==> ASM OBJ,EDX002

ENTER OPTIONAL PARAMETERS BY POSITION :

MACLB1                PRNTR1    LIST,NORLD,NOXREF
1-----2-----3-----4-----
MACLIB1  MACLIB2  PRINTER  ASSEMBLER OPTIONS
(VOLUME) (VOLUME)  NAME      SEPERATED BY COMMAS
                                LIST/NOLIST  XREF/NOXREF/
                                FULLXREF
                                TEXT/NOTEXT  MACRO/NOMACRO
                                ESD/NOESD    OBJECT/
                                NOOBJECT
                                RLD/NORLD    LINECOUNT(N)
                                                N=LINES/PAGE

                                DEFAULT OPTIONS ARE:
                                    LIST
                                    OBJECT
                                    MACRO

```

Host Assembler

HOST ASSEMBLER

Assembly of Event Driven Executive programs on an IBM System/370 requires installation of the following programs on the System/370.

- FDP 5798-NNQ, System/370 Program Preparation Facilities for Series/1
- 5740-LM2, Event Driven Executive Macro Library/Host, as is described in the System Guide.

Invoking the Host Assembler

It is assumed in the following discussion that the procedure EDXPREP (described in the System Guide) is installed as a cataloged procedure. If the procedure is not cataloged, it can be used inline. This procedure makes use of the host assembler program of 5798-NNQ and the macro library and listing program EDXLIST provided by 5740-LM2. The function and operation of EDXLIST is described under "EDXLIST - Host Listing Formatter" on page 383.

Program assembly is accomplished by modifying the source and object program names specified in the SYSIN and SYSOUT DD statements of EDXPREP to reflect your program names, followed by execution of the procedure.

Source programs can contain both Event Driven Language instructions and Series/1 assembler language instructions. Refer to the USER instruction in the Language Reference for information concerning inclusion of Series/1 assembler instructions.

The object module produced by the assembly process can be either a main program or a subprogram, as described in the MAIN= operand of the PROGRAM instruction in the Language Reference.

Transferring the Object Module to Series/1

There are various methods which can be used to transfer object modules created on the System/370 to the Series/1.

If IUP 5796-PGH, Event Driven Executive Host Communication Facility, is installed on the System/370 then utility programs \$HCFUT1 or \$UPDATEH can be used. If the object module is a main program and is not to be linked together with any other object modules, \$UPDATEH can be used to read the object module from the System/370, convert it to an executable program, and store it in a Series/1 disk or diskette. If the object module is to be link edited with other object modules, it can be transferred to the Series/1 by using the READOBJ command of utility program \$HCFUT1.

If IUP 5796-PGH is not installed on the System/370, but a binary synchronous communication link exists between the System/370 and the Series/1, then the utility program \$RJE2780 or \$RJE3780 can be used to transfer object modules from System/370 to the Series/1 if suitable 2780 or 3780 support is installed on the System/370. Modules transferred in this manner must then be processed by \$LINK and/or \$UPDATE to convert them into executable programs. \$RJE2780 or \$RJE3780 can also be used to submit the source program for assembly on the System/370.

If the object module is a subprogram, it must be transmitted to the Series/1 for link editing with a main program before it can be converted into executable format by \$UPDATE.

A complete main program object module only requires processing by \$UPDATE on the Series/1 to convert it to executable form. Refer to "\$UPDATE - Object Program Converter" on page 408.

Program preparation of Event Driven Language programs on a System/370 does not use the Series/1 Native Application Load Facility, 5798-NNR.

EDXLIST - Host Listing Formatter

The purpose of the EDXLIST program is to reduce the size of, and improve the readability of, the program listing. When the PRINT GEN option is used for an assembly containing macros, the macro generated code appears on the assembler produced listing following each source statement. Event Driven Language instructions are implemented as macros, each macro generating an average of six bytes of object code. The object code is gen-

Host Assembler

erated using the assembler DC statement.

Example:: The following example shows the format of the assembler listing for a small section of a program:

LOC	OBJECT CODE	STMT	SOURCE STATEMENT
		1	MOVE A,B
0000	005C	2+	DC A(\$MOV2)
0002	000C	3+	DC A(A)
0004	000E	4+	DC A(B)
		5	ADD B,C
0006	0033	6+	DC A(\$AD222)
0008	000E	7+	DC A(B)
000A	0010	8+	DC A(C)
		9 A	DATA F'O'
000C	0000	10+A	DC F'O'
		11 B	DATA F'O'
000E	0000	12+B	DC F'O'
		13 C	DATA F'O'
0010	0000	14+C	DC F'O'

This listing is difficult to read because of the mixture of source statements and macro generated object code. The purpose of the list post processing program, EDXLIST, is to condense as much of the listing as possible and to print only essential information in a more readable format.

EDXLIST takes each source statement and prints on the same line with it the object code generated by the macro expansion. Each line has room for 10 bytes of data so that nearly all instructions can be printed on one line. This gives the assembly the appearance of being an actual assembler type language. The results of processing the above example with EDXLIST is shown below:

LOC	OBJECT CODE	STMT	SOURCE STATEMENT
0000	005C 000C 000E	1	MOVE A,B
0006	0033 000E 0010	6	ADD B,C
000C	0000	9	DATA F'0'
000E	0000	11	DATA F'0'
0010	0000	13	DATA F'0'

Program Options

A number of options are provided to further condense or eliminate cross reference and relocation dictionary, as well as your own and macro generated comments.

To use EDXLIST, the assembly output must be directed to a data-set rather than the customary SYSOUT class. For example:

```
//SYSPRINT DD DSN=##PRINT,DISP=(,PASS),
//      SPACE=...,UNIT=...,DCB=(BLKSIZE=...)
```

This data set then becomes input to the EDXLIST program. Example job control statements are shown below:

```
//LIST EXEC PGM=EDXLIST,PARM='ISA(16)/... PARMS'
//STEPLIB DD DSN=...
//SYSPRINT DD SYSOUT=A
//WORK DD DSN=##WORK,UNIT=SYSDA,SPACE=(TRK,5)
//IN DD DSN=##PRINT,DISP=(OLD,PASS)
//LIST DD SYSOUT=A
```

The IN DD card refers to the data set containing original assembler listing and the LIST DD card refers to the new listing produced by EDXLIST.

Host Assembler

If in addition to the above, the original assembler listing is also required, the host utility program IEBTPCH can be used in the following manner to produce it:

```
//BIGLIST EXEC PGM =IEBTPCH
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=$$$PRINT,DISP=(OLD,PASS)
//SYSUT2 DD SYSOUT=A,DCB=BLKSIZE=133
//SYSIN DD *
PRINT PREFORM=A
```

EXEC Statement Parameters

A number of parameters for the EDXLIST program can be specified on the EXEC statement. A list of the options follows. Default values can be found under "Defaults" on page 388.

- PRINTER** Output is listed on a printer.
- TERMINAL** Output is listed on a terminal.
- TERMINALE** Same as TERMINAL, except only error messages and associated source statements are printed.
- SEQ** Source code sequence numbers are printed.
- STMT** Assembler statement numbers are printed.
- CONTROL=***** Defines a special control card that is recognized as an execution time control statement. Any two characters can be specified for the second and third characters; the first, however, must be an '*'. See control options below.
- COMMENTS=NONE** Print no comments.
- COMMENTS=ALL** Print all comments.
- COMMENTS=PROGRAM** Print only comments not generated by macros.

COMMENTS=CONTROL	Print comments only if columns 1, 2, and 3 match the value specified in the CONTROL= parameter.
EQUATES=* 	Print only macro generated equates of the form +LABEL EQU *.
EQUATES=PROGRAM	Print only equates not generated by macros.
LABELS=PROGRAM	Print labels not generated by macros.
LABELS=PEND	Same as PROGRAM except all labels after the ENDPROG statement are printed.
START=PROGRAM	All listings before the PROGRAM statement are not printed.
START=START	Printing is to begin at first line of listing.
XREF=NONE	Cross-reference listing is not printed.
XREF=\$	Print abbreviated cross-reference listing with the exception of labels beginning with the character \$.
XREF=ALL	Same as \$, except \$ labels are included.
XREF=AREF	Same as ALL, except statement references are included.
XREF=\$REF	Same as AREF, except \$ LABELS are included.
RLD=YES	Print condensed relocation dictionary.
RLD=NO	Do not print relocation dictionary.

Notes:

1. If both STMT and SEQ are specified, then statement numbers appear to the left of the source code and sequence numbers in the right margin. If only one is specified, the numbers appear to the left of the source code.
2. XREF=ALL or XREF=\$ print only the label name and its address and these are placed in seven columns across the page.
3. XREF=AREF or \$REF prints one label per line and all statement numbers referencing the label are printed. If this option is selected, it is assumed STMT was also specified.

Host Assembler

4. If START=PROGRAM is specified and there is no PROGRAM statement, then the listing is printed as if START=START was coded.
5. Whenever one of the following control statements appears in the assembler listing, then the special listing functions occur.

Control Options

*XXEJECT	Skip to a new page
*XXSPACE 1	Skip one line
*XXSPACE 2	Skip two lines
*XXTITLE X	-----maximum 50 characters-----X Will print this character string at the top of each successive page.
*XXLIST ON	List input exactly as it appears on original listing
*XXLIST OFF	Resume listing compression.

Note: '*XX' represents the three characters coded in the CONTROL= parameter discussed previously.

These control statements are treated as comments by the assembler. The macros EJECT, SPACE, and TITLE distributed in the library S1.EDX.LISTMAC.ASM will create control statements with the characters *** in columns 1, 2, and 3. Thus, when the corresponding assembler listing control statements appear in the source program, the appropriate control statements are created for EDXLIST. They require CONTROL=*** in the PARM field.

Defaults

The two main parameter options are PRINTER and TERMINAL. TERMINAL assumes output on a slow typewriter device. It, therefore, attempts to print as little nonessential information as possible. PRINTER, on the other hand, assumes a high speed printer is being used, and although there is still around a 5 to 1 output compression ratio, the listing has a little extra information to make it more readable. The following defaults apply:

PRINTER	TERMINAL
STMT,SEQ	STMT
COMMENTS=PROGRAM	COMMENTS=NONE
EQUATES=*	EQUATES=*
LABELS=PEND	LABELS=PEND
START=PROGRAM	START=PROGRAM
XREF=§	XREF=NONE
RLD =YES	RLD=NO

\$LINK

\$LINK - LINKAGE EDITOR

The \$LINK program provides a linkage editor capability to aid in preparing programs to execute in an Event Driven Executive system. With \$LINK, you can:

- Combine two or more separately compiled or assembled object modules into a composite module
- Combine modules with multiple control sections, for example, COBOL, FORTRAN, and PL/I programs, into a composite module
- Format the output of \$S1ASM for input to \$UPDATE
- Eliminate duplicate control sections from the input modules
- Automatically include standard routines referenced by the program (AUTOCALL)
- Provide a storage map of the combined, edited modules
- Build an EDX supervisor nucleus

Each of these is explained in this section.

The object modules that are input to \$LINK can be created by any of the following program preparation facilities:

- \$EDXASM
- \$S1ASM
- FORTRAN
- COBOL
- PL/I
- Host Assembler (FDP 5798-NNQ)

\$LINK accepts object modules that are in the compressed record format created by the Realtime Programming System assembler.

A \$LINK output module must be formatted by the utility program \$UPDATE to convert it to executable (loadable) form. Figure 23 on page 391 illustrates the steps necessary to prepare a program for execution.

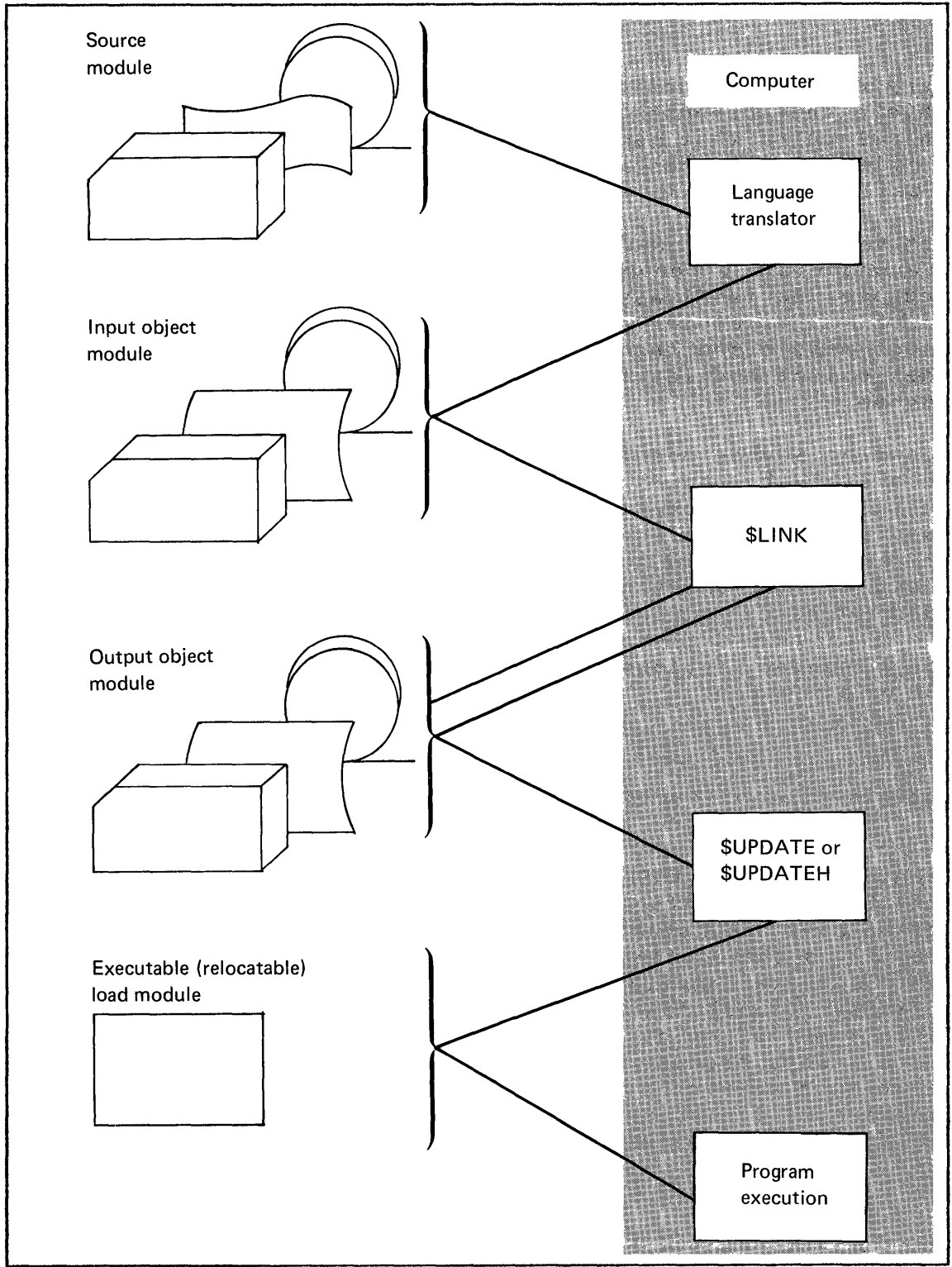


Figure 23. Programming with a linkage editor

\$LINK

Combining Modules

Every program is designed to fulfill a particular purpose. To achieve that purpose, the program can often be divided into logical units that perform specific functions. A logical unit of code that performs a function, or several related functions, is called a source module.

Each source module can be programmed as a separate entity using an appropriate programming language (for example, Event Driven Language, Series/1 Assembler Language, COBOL, FORTRAN, PL/I, or the host assembler) and assembled or compiled. The resulting object module can then be processed by \$LINK, along with the other object modules that are necessary to make up the entire program.

Preparing source modules as separate logical units allows you the ability to modify one part of a program and requires the reassembly or recompilation of only the modified unit (followed by another execution of \$LINK), as opposed to requiring the reassembly or recompilation of the entire program. It also allows a function that is common to several programs to be coded and assembled or compiled once and then included wherever needed, saving effort and ensuring consistency. Linkage editing, if needed, follows the source program assembly or compilation of a program.

Multiple Control Sections

A control section (CSECT) is a unit of code (instructions and data) that is, in itself, an entity. All elements of a control section have a constant relationship to each other and, therefore, it is the smallest separately relocatable unit of a program. Separate compilations always produce individual control sections, but multiple control sections can be produced from a single compilation as well. The COBOL, PL/I, and FORTRAN compilers always produce multiple CSECTs and, therefore, their output must always be processed by \$LINK. \$SIASM and the host assembler will also produce multiple control sections if the CSECT instruction is used.

Formatting Modules for \$UPDATE

The output object modules of \$SIASM and the Realtime Programming System Program Preparation Subsystem assembler must be

reformatted by \$LINK prior to \$UPDATE processing. The standard link edit process does the reformatting and no special parameters are required.

Notes:

1. \$UPDATEH can handle host assembler modules without link-editing.
2. The output of the Realtime Programming System language processors must be link-edited before processing by \$UPDATE.

Elimination of Duplicate Control Sections

If the input to \$LINK contains more than one control section (CSECT) with the same name, the first section is kept, all subsequent sections with the same name are deleted and all references are adjusted to point to the first CSECT. This makes the resulting program smaller and assures that there is only one copy of any shared data areas.

Automatic Inclusion (Autocall)

Frequently used routines can be assembled or compiled as separate modules and automatically included in any program that references them by using the AUTOCALL facility of \$LINK. This technique makes it convenient to define and use standard routines almost as if they were instructions. The Event Driven Language FORMAT, GETEDIT, and PUTEDIT instructions use this technique. See "Autocall Option" on page 401 for detailed information on using the autocall option.

Storage Map

\$LINK produces a table showing the name, location, and length of all control sections in the program plus the location of symbols (entries) defined as referenceable by the entire program. This is useful in debugging and in determining the total size of the program.

\$LINK

Building an Event Driven Executive Supervisor

\$LINK and the link edit process are used to generate a customized supervisor by selectively including the modules needed for the desired functions. See the System Guide for further information.

The Link Edit Process

The logical structure of an object module consists of three entities:

ESD An external symbol dictionary that includes the name of each external symbol, the type of reference it is, and its location in the program. ESD entries represent symbols such as the label on a PROGRAM, CSECT, or ENTRY statement, or the label of a DATA or DC statement that defines an A-type constant.

RLD A relocation dictionary that contains an entry for each address that must be relocated (adjusted) before the module is executed. Each entry specifies an address constant by indicating its location in the program.

TEXT An area that contains the instructions and data of the program in the binary codes used by the Event Driven Executive and/or the Series/1 processor.

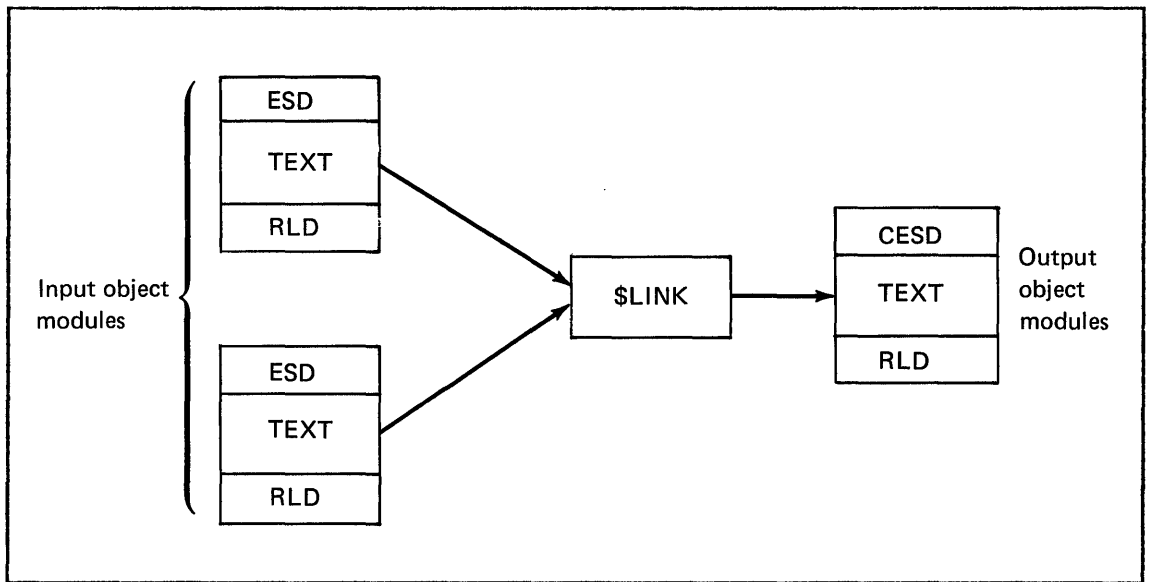
To produce a program from a collection of object modules compiled or assembled separately by different compilers or assemblers (for example, \$EDXASM, \$S1ASM, COBOL, FORTRAN, PL/I, or the host assembler FDP 5798-NNQ), the object modules must be link-edited (combined) to resolve the program origins (load points) of each object module and the external references between the different modules.

Each object module is assigned a program origin (usually zero) during assembly or compilation. When these object modules are link-edited, \$LINK assigns an origin (load point) to the text for the primary program and adjusts the RLDs describing address constants accordingly. The text for the other object modules, as well as the address constants, are assigned addresses relative to the origin (load point) assigned to the primary program. This causes the link-edited (combined) text for all object modules to occupy consecutive addresses in the output object module produced by \$LINK. See Figure 24 on page 395.

Any object module that refers to an external symbol defined in another object module must identify it as an externally defined symbol. External symbols (EXTRNS or WXTRNS) between these modules are resolved by matching the referenced symbols to defined symbols.

The ESDs and RLDs of the object modules are combined to form composite ESDs and RLDs for the complete program. The output RLDs include relocatable information for the complete text of the output object module of \$LINK.

An output object module produced by \$LINK looks as follows.



\$LINK

Input to \$LINK

Input to \$LINK consists of LINKCNTL, the data set containing control records, the various data sets containing the object modules to be linked together, and the optional autocall data set.

You must provide the names of the input and output object modules to \$LINK with a set of control records stored in a disk or diskette data set called LINKCNTL. How to format and create these control records is described under "Control Records."

In addition to the data set containing the control records, You must also allocate two work data sets on disk or diskette. The message data set \$LEMSG is supplied with your installation package and is required by \$LINK. A description of all of the data sets is contained under "Data Sets Used by \$LINK" on page 400, along with suggestions for selecting the sizes of the data sets.

A sample of input to \$LINK follows:

```
OUTPUT  LEOUT,DAN01 AUTO=LEAUTO,DAN01
INCLUDE LETEST,DAN01
INCLUDE LETEST1,DAN01
INCLUDE LETEST2
INCLUDE LETEST3,DAN01
END
```

Control Records

You must allocate the control record data set (LINKCNTL) using \$DISKUT1. Then enter the control records with \$EDIT1N or \$FSEDIT and store the records with either SAVE or WRITE.

LINKCNTL should contain the names of the input object modules, the names of the output object modules, and other control information. Both data set name and volume name are required on all OUTPUT and INCLUDE records, unless the data set resides on the IPL volume. In the latter case, only the data set name is required.

Any number of leading blanks are permitted in control records, but the record must not be longer than 71 characters. Any record with an '*' in position 1 is treated as a comment and will list out on the printed output.

All the parameters used must be coded in the order shown. The parameters must be separated by one or more spaces.

END Record

The END record terminates the control record string. The format is:

Syntax

END
Required: None
Default: None

No operands are required.

\$LINK

INCLUDE Record

The INCLUDE records specifies the input object module to be link-edited. You can include several input object modules but you must have an INCLUDE record for each input object module. The format is:

Syntax

```
INCLUDE name,volume

Required: name
Default: Volume defaults to IPL volume.
```

Operands Description

- name** The name of the input object module to be link-edited. There must be one or more spaces between 'INCLUDE' and 'name'.
- volume** The disk or diskette volume that contains 'name'. Not required if 'name' resides on the IPL volume. The first INCLUDE record cannot specify the name of a subprogram (i.e., one with MAIN=NO in the PROGRAM statement or one that does not contain a PROGRAM statement). All other INCLUDE records should specify the names of subprograms.

All of the 'name,volume' specifications on INCLUDE records must be unique in the input to \$LINK. In a like manner, all ENTRY labels and program section (CSECT) definitions must be unique (i.e., the same name cannot be used for both a CSECT and an entry, or for two ENTRIES, etc.). Duplicate symbol type errors will result if these rules are not observed. If \$LINK encounters two or more CSECTs with the same name, the first one will be accepted and all others ignored.

CSECTs input to \$LINK must not have a length of zero.

OUTPUT Record

There can only be one OUTPUT record in LINKCNTL and it must be the first record (excluding comment records) in the data set. The format is:

Syntax

OUTPUT name,volume NOMAP AUTO=name2,volume ENTRY=label

Required: name

Default: Volume defaults to IPL volume.

Operands Description

name The name of the data set (previously allocated on disk or diskette) to contain the output object module from \$LINK. There must be one or more spaces between 'OUTPUT' and 'name'.

volume The name of the disk or diskette volume containing the output data set.

NOMAP This parameter suppresses printing of the storage map portion of \$LINK output starting with the line which begins 'OUTPUT NAME=' and continuing through the final CSECT or ENTRY listing line. See the sample printout for details.

AUTO=name2,volume

This parameter specifies the name (and volume if the data set is not on the IPL volume) of the data set containing the list of names of object modules which may be included via the autocall facility. A further discussion of the autocall option is provided under "Data Sets Used by \$LINK" on page 400 and "Autocall Option" on page 401.

ENTRY=label

This parameter is used only if the output of \$LINK is to be an Event Driven Executive supervisor program (\$EDXNUC). When generating a supervisor, 'ENTRY=\$START' must be coded.

\$LINK

Data Sets Used by \$LINK

\$LINK requires you to specify three data set names at the start of program execution (one if invoked by the Session Manager).

LINKCNTL (DS1) The data set that will contain the control records.

LEWORK1 (DS2) A work data set that will contain various intermediate work areas for \$LINK. This data set must be at least 265 records and will require approximately 1 additional record for each 60 RLD items or for each 42 program section definition (CSECT) or ENTRY specifications. 400 records should be sufficient for almost any program.

\$LINK dynamically divides DS2 into three sections and refers to them as DS2, DS7, and DS8. Therefore, error messages which reference DS7 or DS8 actually refer to portions of DS2.

The session manager automatically allocates 400 records to DS2. Its name is \$SM1user (where user is the sign on ID used). the sign on ID used).

LEWORK2 (DS3) A second work data set. This data set contains a work area for \$LINK, plus the storage area where all \$LINK printed output is stored until it is printed at the end of \$LINK execution. A size of 20 records plus 1 record for each 3 lines of printed output expected should be sufficient. The amount of printed output varies with the number of INCLUDE records, the number of unresolved EXTRNs and WXTRNs, etc. A data set size of 150 records should be sufficient for most programs.

If the space available for messages in DS3 becomes filled, then the message file will be printed before the end of \$LINK execution. \$LINK then refills the message file as needed. This condition can result in the printed If you are executing other programs from the same terminal doing \$LINK, the printed output will be interspersed with the output from the other programs.

The session manager automatically allocated 400 records to DS2. Its name is \$SM2user (where user is the sign on ID used).

The optional data set specified by the AUTO= parameter of the OUTPUT record is dynamically opened as DS9 by \$LINK. At least one record in DS9 is required for every object module named in the autocall list.

In addition, the data set \$LEMSG must be installed on volume ASMLIB. \$LEMSG is supplied as part of the distributed system and contains messages used by \$LINK and is referenced as 'DS4' by \$LINK. \$LINK dynamically opens the data sets specified by the OUTPUT and INCLUDE records as 'DS5' and 'DS6', respectively.

The data set specified by the OUTPUT control record may need to be as large as 350-450 records when link editing \$EDXNUC.

Autocall Option

\$LINK provides a limited autocall option whereby modules that are not explicitly included (via the INCLUDE control record) in the output object module can be automatically incorporated. When you code the AUTO= parameter of the OUTPUT record, specify the name (and volume unless the data set is on the IPL volume) of the data set that contains the list of object module names/volumes, along with their entry point names, which are available for automatic inclusion in the output object module.

Standard Autocall List

As part of the distribution of the Event Driven Executive system, a standard autocall list is included in the data set \$AUTO in volume ASMLIB. This standard list contains the names of the modules and entry points that can be autocalled as a result of including the following functions in your program:

- graphics formatting instructions
- data formatting instructions
- screen formatting subroutines
- square root function.

\$LINK

These functions generate EXTRN statements that are resolved by specifying AUTO=\$AUTO,ASMLIB.

Using a Special Autocall List

To add your own autocall list names to those provided in \$AUTO, use \$EDIT1N or \$FSEDIT to make the additions and then save the resulting modified list as a saved data set with a name of your choice. This data set name (and volume) are specified in the AUTO= parameter of the OUTPUT record. If you use the functions listed previously in your program, you will need to merge the contents of \$AUTO into your autocall list.

Using the Autocall Feature

Perhaps the most common use of the autocall feature occurs when you have assembled some number of commonly used subroutines that may or may not be needed by a given primary program. When you assemble a primary program, you can link edit it with the required subroutines by providing an INCLUDE record for the primary program and for each required subroutine. An alternate method is to put the names of all potentially needed subroutines and their entry point names into the autocall list, then provide only an INCLUDE record for the primary program and add the AUTO= parameter to the OUTPUT record. The method that is most efficient depends upon how your programs are organized and the frequency with which common subroutines are used.

You must provide EXTRN statements in the primary program for the entry point names in the modules to be autocalled as well as linkage (branches) to the desired entry points in the autocalled modules. Modules that have been included with the autocall can contain EXTRNs that are to be resolved by further autocall processing. \$LINK searches the autocall list repeatedly until either all EXTRNs have been resolved or until no ENTRY in the autocall list matches an unresolved EXTRN.

WXTRN Processing

The autocall list is not searched for ENTRY points to match unresolved WXTRNs. They are resolved only from the ENTRY definitions contained in the modules included (via the INCLUDE control record) or autocalled. An unresolved WXTRN has no affect

upon \$LINK completion codes and the storage location containing the WXTRN is set to relocatable zero. Therefore, at program execution time, the addresses of unresolved WXTRN's are the program load point.

Autocall List Record Format

The format of the autocall list records is shown below. They can most conveniently be created as the saved output of a \$EDIT1N or \$FSEDIT editing session using tab settings of 20, 30, 40, 50 and 60. Each input line must contain the 'name,volume' entry of the object module to be autocalled, starting in position 1. Up to 5 entry point names can be specified per line, starting in positions 20, 30, 40, 50, and 60. If a data set has more than five entry points, multiple lines are required. Each line must contain the module 'name,volume' entry beginning in position 1 and at least one entry point name starting in position 20. The autocall list must be terminated by an '**END ' entry in an entry point name field or in a module name field. 'Volume' is required for each 'name' that is not on the IPL volume. For example:

line position:					
1	20	30	40	50	60
NAME1,VOL1	ENTRY1	ENTRY2	ENTRY3		
NAME2,VOL1	ENTRYA	ENTRYB	ENTRYC	**END	

Output from \$LINK

The result of executing \$LINK consists of a printed listing and an output object module. A sample of the printed listing follows. It consists of a start message, a list of the input control records, a list of any unresolved EXTRN or WXTRN labels (none in example), an optional map of program section (CSECT) and ENTRY point locations in the output module, a length message, and an ending message. All numeric values are given in hexadecimal, except the RLD COUNT field.

\$LINK

```
$LINK EXECUTION STARTED
$LINK EXECUTION CONTROL RECORDS
  FROM LINKCNTL,EDX002
  OUTPUT LEOUT,DAN01 AUTO=LEAUTO,DAN01
  INCLUDE LETEST,DAN01
  INCLUDE LETEST1,DAN01
  INCLUDE LETEST2
  INCLUDE LETEST3,DAN01
  INCLUDE SUB4,DAN01          VIA AUTOCALL
  INCLUDE SUB5,DAN01          VIA AUTOCALL
  END
OUTPUT NAME=  LEOUT
  ESD TYPE    LABEL      ADDR      LENGTH
CSECT        $PROGRAM  0000      00F0
CSECT
  ENTRY      SUB1       00F2
  ENTRY      DEF        010F
CSECT        0116      0044
  ENTRY      I          0116
  ENTRY      MSG        011A
  ENTRY      SUB2       013E
CSECT        015A      0034
  ENTRY      X          015A
  ENTRY      SUB3       0160
  ENTRY      ABC        0180
CSECT        018E      0022
  ENTRY      SUB4       0190
CSECT        01B0      001E
  ENTRY      SUB5       01B2
MODULE TEXT LENGTH= 01CE, RLD COUNT= 40
LEOUT      ADDED TO DAN01
$LINK COMPLETION CODE= -1
```

The printed output can be directed to any terminal on the system by passing the terminal name as a parameter (\$PARM1) to \$LINK. If the parameter is not passed to \$LINK at program load time, you will be prompted to enter the desired name from the invoking terminal. A null response to the prompt for the terminal name routes the output to the system printer, \$SYSPRTR.

The output object module is stored in the disk or diskette data set specified by the OUTPUT control record if no errors occur that are severe enough to cause \$LINK to terminate (e.g., disk hardware errors). However, the output module is probably not usable if any errors occurred during \$LINK execution. Refer to

"\$LINK Completion Codes" on page 440 for more information.

Invoking \$LINK

Invoking \$LINK using \$L and \$JOBUTIL

If you invoke \$LINK with the \$L from a terminal, you are prompted for the names (and volumes) of data sets LINKCNTL, LEWORK1, and LEWORK2. You are also prompted to enter the name of the terminal to which the printed output is to be directed. An invalid name reply to the device name prompt causes the output to be directed to the terminal you are currently assigned to. A null reply (pressing the ENTER key or carriage return) causes the output to be directed to \$SYSPRTR.

\$LINK can also be invoked as part of a batch procedure under the control of the job stream processor utility, \$JOBUTIL. In this instance, you must supply the same information as above, with the DS and PARM commands of \$JOBUTIL.

An example of invoking \$LINK via \$JOBUTIL commands follows:

```
LOG          $SYSPRTR
PROGRAM     $LINK
DS          LNKCONTR,MYLIB
DS          LNKWORK1,EDX003
DS          LNKWORK2,EDX003
PARM        $SYSPRTR
NOMSG
EXEC
```

\$LINK sets a completion code that can be tested by means of the JUMP command of \$JOBUTIL. Successful completion results in a code of -1. Errors encountered during \$LINK execution cause completion codes as shown in "\$LINK Completion Codes" on page 440. Any disk I/O error which causes abnormal termination of \$LINK results in a completion code of 12. The most severe (numerically greatest) completion code encountered is the one posted when \$LINK terminates execution.

\$LINK

Invoking \$LINK Using the Session Manager

To invoke \$LINK using the session manager, select option 5 from the program preparation secondary option menu.

The following parameter selection menu is displayed for entry of the control data set and volume, and the output device. The example shows the control data set as being LINKCNTL on volume EDX002. The required linkage editor work data sets were allocated at session manager log on time.

```
$SMM0205: SESSION MANAGER $LINK PARAMETER INPUT MENU ---  
ENTER/SELECT PARAMETERS:          DEPRESS PF3 TO RETURN
```

```
LINK CONTROL (NAME,VOLUME) ==> LINKCNTL,EDX002
```

```
OUTPUT DEVICE (DEFAULTS TO TERMINAL) ==> $SYSPRTR
```

Object Module Record Format

Records in object modules are 80 bytes in length, packed 3 per disk or diskette record or variable-length records blocked 256. The following general description of their format is given as an aid in diagnosing error printouts. If error message 13 is received while link editing, error records will be printed. Object module records containing errors are printed in hexadecimal form if byte one equals '02'; otherwise they are printed in character string form.

Byte	Contents
1	Hexadecimal '02'
2-4	One of the following EBCDIC strings: ESD = External Symbol dictionary record TXT = Text data record RLD = Relocation dictionary record END = End of object module
5-6	Unused
7-8	Blank for ESD, RLD, END Address of 1st data word for TXT
9-10	Blanks
11-12	Number of bytes of ESD, RLD, or TXT data in record
13-14	Blanks
15-16	ESD= External Symbol Dictionary Identification (ESDID) of first ESD item in record; blank if all ESD items in record are label definitions (LDs) TXT= ESD ID of control section containing the text. RLD,END= blanks
17-72	Data items for this record
73-80	Unused by \$LINK

\$UPDATE

\$UPDATE - OBJECT PROGRAM CONVERTER

\$UPDATE converts an object module on a specified volume into an executable relocatable load module and stores it into a specified volume.

The object module used as input to \$UPDATE may have been compiled by \$EDXASM or the host assembler provided by FDP 5798-NNQ. An object module that is the output of the linkage editor, \$LINK, can be the input to \$UPDATE. Object modules created by the host assembler must be transmitted to a Series/1 disk or diskette volume by a facility such as the IBM 2780/3780 RJE emulation program \$RJE2780/\$RJE3780, or by utility program \$HCFUT1, before they can be used as the input to \$UPDATE. The object output of language translators other than \$EDXASM or the host assembler must be processed by the linkage editor, \$LINK, before it can be used as input to \$UPDATE.

\$UPDATE can be invoked either by the \$L command, by the batch job stream processor (\$JOBUTIL), or by the session manager. When invoked, \$UPDATE prompts you for the information it requires. Examples of this interactive usage follow. An example of batch job stream invocation is given at the end of this section.

\$UPDATE Commands

\$UPDATE commands are:

```
COMMAND (?): ?  
  
CV - CHANGE VOLUME  
RP - READ PROGRAM  
EN - END  
COMMAND (?):
```

Examples

Load \$UPDATE

```
> $L $UPDATE
$update      22P,, LP= 4800

THE DEFINED INPUT VOLUME IS EDX002, OK? N
ENTER NEW INPUT VOLUME LABEL:  EDX003
THE DEFINED OUTPUT VOLUME IS EDX003, OK? Y
```

Change Volume

```
COMMAND (?):  CV

THE DEFINED INPUT VOLUME IS EDX002, OK? N
ENTER NEW INPUT VOLUME LABEL:  EDX003
THE DEFINED OUTPUT VOLUME IS EDX003, OK? Y

COMMAND (?):

If volume not mounted:

COMMAND (?):  CV
ENTER NEW OUTPUT VOLUME LABEL:  EDXDFG
VOLUME NOT MOUNTED

COMMAND (?):
```

\$UPDATE

Convert and Store a New Program: This example allocates a new data set TESTPROG (type PGM) of the required size if a data set with that name has not already been allocated.

```
COMMAND (?):  RP
OBJECT MODULE NAME:  OBJSET
OUTPUT PGM NAME:  TESTPROG
TESTPROG STORED
COMMAND (?):
```

Convert and Store a Nonexistent Program

```
COMMAND (?):  RP
OBJECT MODULE NAME:  DUMMY
OUTPUT PGM NAME:  PROGRAM
INPUT DATA SET 'DUMMY' NOT FOUND
COMMAND (?):
```

Note: No action is taken when error occurs.

Convert Existing 'Program' That is Not a Program Type Member

```

COMMAND (?):  RP
OBJECT MODULE NAME:  PROG1
OUTPUT PGM NAME:  PROG2
ILLEGAL HEADER FORMAT
COMMAND (?):
    
```

Note: No action is taken when error occurs.

Convert Program Where Existing Output Data Set is Not Program Type

```

COMMAND (?):  RP
OBJECT MODULE NAME:  OBJSET
OUTPUT PGM NAME:  TSTPROG
TSTPROG IS NOT A PROGRAM
COMMAND (?):
    
```

Note. No action is taken when error occurs.

SUPDATE

Convert and Replace Existing Output Program with Same Output Name: In this example, if the existing output program is to be replaced with the new output program and the new and old sizes are the same, then the new program data replaces the old with no other changes. If the new space required is different from the existing space, the existing data set is deleted and a new one of the proper size is allocated wherever enough free space is available.

```
COMMAND (?):  RP
OBJECT MODULE NAME:  OBJSET
OUTPUT PGM NAME:  TSTPRG1
OUTPUT PGM NAME:  TSTPRG1
TSTPRG1 REPLACE? Y
TSTPRG1 STORED
COMMAND (?):
```

\$UPDATE

Convert and Rename New Output Program if an Output Program Already exists: The existing output data set is undisturbed and a new data set (type PGM) of the proper size and with the new name is allocated.

```
COMMAND (?):  RP
OBJECT MODULE NAME:  OBJSET
OUTPUT PGM NAME:  TESTPROG
TESTPROG REPLACE?  N
RENAME?  Y
NEW PGM NAME:  TSTPRG
TSTPRG STORED
COMMAND (?):
```

End \$UPDATE

```
COMMAND (?):  EN
$UPDATE ENDED AT 11:39:34
```

\$UPDATE

Invoking \$UPDATE

Invoking \$UPDATE Using \$JOBUTIL

When \$UPDATE is invoked as part of a batch job under the control of \$JOBUTIL, certain restrictions apply to its operation. In this mode, the command is assumed to be RP. The Rename function is not supported; however, the Replace function is. Refer to the preceding examples for a description of Rename and Replace.

In batch mode, \$UPDATE terminates its execution after performing one RP command. A completion code is set by \$UPDATE depending upon the success or failure of the requested operation. This code can be tested by the JUMP command of \$JOBUTIL. The \$UPDATE completion codes are described in **Figure reference 'ccupd' unresolved**.

When \$JOBUTIL is used to invoke \$UPDATE, the information required by \$UPDATE must be passed to it by means of the PARM command of \$JOBUTIL. The required information consists of:

1. The name of the device to receive the printed output resulting from \$UPDATE execution
2. The name, volume of the data set containing the input object module
3. The name, volume of the data set to contain the output loadable program
4. An optional parameter YES if the output module is to replace an existing module of the same name, volume

The volume names of the data sets must be given unless they reside on the IPL volume.

The first three items of information are required and must be given in the order described. At least one blank must occur between each of these four items in the PARM command.

An example of invoking \$UPDATE via \$JOBUTIL commands follows:

```
.  
. .  
PROGRAM $UPDATE  
PARM $SYSPRTR OBJMOD,VOL1 MYPROG YES  
NOMSG  
EXEC  
. .  
.
```

In this example, \$SYSPRTR receives the printed messages, the input object module is OBJMOD on VOL1, the output program is MYPROG on the IPL volume. If MYPROG already exists on the IPL volume it is replaced by the new version. If MYPROG does not already exist then space is allocated for it by \$UPDATE.

\$UPDATE

Invoking \$UPDATE Using the Session Manager

To invoke \$UPDATE using the session manager, select option 6 from the program preparation secondary option menu.

The following parameter selection menu is displayed for entry of the required data sets and other parameters. The example shows the object program in the data set ASMOBJ is to be formatted and placed in the data set TSTPGM. The REPLACE parameter is left blank when a member does not already exist. If the member did exist, that parameter should be entered as YES.

```

|
| $SMM0206: SESSION MANAGER $UPDATE PARAMETER INPUT MENU -
| ENTER/SELECT PARAMETERS:                DEPRESS PF3 TO RETURN
|
| OBJECT INPUT (NAME,VOLUME) =====> ASMOBJ,EDX002
| PROGRAM OUTPUT (NAME,VOLUME) =====> TSTPGM,EDX003
| REPLACE (ENTER YES IF PROGRAM EXISTS) ===>
| LISTING (TERMINAL NAME/*) =====> $$SYSPRTR
|
|
```

Note: The object input, program output, and listing terminal name must be entered. An '*' indicates the listing will be displayed on the terminal you are currently assigned to.

Creating a Supervisor

The name \$EDXNUC for the output program receives special treatment by \$UPDATE since the creation of a supervisor results in an absolute, rather than a relocatable program. The following rules apply:

1. If the first seven characters of the output program name are \$EDXNUC then an absolute supervisor program will be formatted.
2. If the eighth character of the name is either a blank or is not present, then the output program will automatically replace the existing supervisor program on the specified volume.
3. If the eighth character of the program name is any character except a blank, then the output supervisor program will be stored in the library on the specified volume using the eight character name.

In this manner, you can create multiple supervisor programs for different machine configurations on one Series/1. You can then copy them to the diskettes which can be used on the Series/1 having the proper configuration.

Supervisors created and stored under 8-character names (e.g., \$EDXNUC2, \$EDXNUCX, etc.) can be tested by:

1. Copying the member into \$EDXNUC on the IPL volume, and IPL the system again, or by
2. Providing the CTS address of the stored supervisor in response to the Stand Alone Utilities IPL message:

'EXEC='.

\$UPDATEH

\$UPDATEH - OBJECT PROGRAM CONVERTER (HOST)

\$UPDATEH transfers, over a communications link, Series/1 object programs that are members of a host partitioned data set (PDS) and stores them in a Series/1 disk or diskette volume in the proper format to be loaded for execution. These programs were previously assembled on the host. To change the name of the default host library, locate the label, HOSTNAME, in the \$UPDATEH listing and change the name from the supplied default library name to the host library name desired by you. \$UPDATEH must then be assembled and installed in the program library.

\$UPDATEH requires that the Event Driven Executive Host Communication Facility (IUP 5796-PGH) be installed on the host computer.

\$UPDATEH can be invoked either by the \$L command, by the batch job stream processor (\$JOBUTIL), or by the session manager. When invoked, \$UPDATE prompts you for the information it requires. Examples of this interactive usage follow.

\$UPDATEH Commands

The commands available under \$UPDATEH are listed below. To display this list at your terminal, enter a question mark in response to the prompting message COMMAND (?):.

```
COMMAND (?): ?
```

```
CH      Change Host Library
CV      Change Series/1 Library
RP      Read a Program
EN      End $UPDATEH
```

```
COMMAND (?):
```

After the commands are displayed, you are again prompted with COMMAND (?): to which you respond with the command symbol for the function of your choice (for example, CH).

Examples

Transfer of a New Program

```
> $L
PGM(NAME,VOLUME): $UPDATEH
$UPDATEH      23P,15.29.59, LP=4400
THE DEFINED HOST LIBRARY IS S1.EDX.LOADLIB, OK? Y
THE DEFINED VOLUME IS EDX001, OK? Y

COMMAND (?): RP

PGM NAME: TPTEST
TPTEST STORED

COMMAND (?):
```

Request Nonexistent Program

```
COMMAND (?): RP

PGM NAME : DUMMY
DUMMY IS UNKNOWN
COMMAND (?):
```

Transfer and Replace Existing Program

```
COMMAND (?): RP

PGM NAME: PROG1
PROG1 REPLACE? Y
PROG1 STORED

COMMAND (?):
```


\$UPDATEH

Transfer and Rename an Existing Program: This example transfers an existing program, PROG1, and renames it PROG2. The existing program is not replaced.

```
COMMAND(?): RP  
  
PGM NAME: PROG1  
PROG1 REPL? N  
RENAME? Y  
  
PROG NAME: PROG2  
PROG2 STORED  
  
COMMAND (?):
```

Change Host Library

```
COMMAND (?): CH  
ENTER HOST LIBRARY NAME: S1.EDX.LOADLIB2  
  
COMMAND (?): EN  
  
$UPDATE ENDED 15.30.54
```

CHAPTER 6. MESSAGES AND CODES

This chapter contains messages and codes issued by the Event Driven Executive system.

SYSTEM OPERATION MESSAGES

IPL Operation

The supervisor can be IPLed from the 4962 or 4963 disk or the 4964 or 4966 diskette. The IPL switches on the Series/1 operator console must be in the appropriate position for the type of IPL and the \$SYSLOG terminal should be turned on. For example, if DISK is used to IPL, the IPL switch must be set to PRIMARY if the DISK is wired as the primary IPL device or to SECONDARY if the DISK is wired as a secondary IPL device.

After IPL, the following message appears on the \$SYSLOG terminal:

*** EVENT DRIVEN EXECUTIVE ***

Messages

Volume Initialization

The supervisor then attempts to read each of the defined disk/diskette volumes and prints a status message similar to the following:

VOLSER	TYPE	IODA	STATUS
EDX001	PRI.	0002	ONLINE
EDX002	PRI.	0003	ONLINE (IPL)
SUPLIB	SEC.	0003	
MACLIB	SEC.	0003	
	PRI.	0012	UNUSABLE
EDX013	PRI.	0013	ONLINE

Note: This information can be displayed again by using the \$IOTEST utility.

VOLSER The volume identification for the disk/diskette or the name of a volume on a disk or diskette.

TYPE If primary, VOLSER is the volume identification of the disk/diskette.

If secondary, VOLSER is the name of defined volume on the device.

IODA The hardware device address.

STATUS The status of the hardware device.

If a diskette was not inserted in a defined 4964 drive, the STATUS is UNUSABLE and the volume identification for the IODA is blank. If a diskette is subsequently put into the drive, it will be considered OFFLINE until a \$VARYON command is performed. If the diskette contains a nonstandard label, the STATUS is marked OFFLINE.

During IPL, the system initializes the 4966 Diskette Magazine Unit by reading the diskette label on each diskette. The diskettes are read in the following order:

1. The diskettes in individual diskette slots 1 through 3 are read until all three are read or an empty slot is found

2. The diskettes in diskette magazine A are read until all ten are read or an empty slot is found
3. The diskettes in diskette magazine B are read until all ten are read or an empty slot is found

Each diskette read that contains a valid label is marked ONLINE. All others are marked OFFLINE.

If the device is not turned on or has a hardware failure, the STATUS is marked UNUSABLE.

| Tape Initialization

| Tape initialization is performed after volume initialization. The supervisor attempts to read each tape address, as defined at system generation. If the address is valid, the tape is marked OFFLINE and is therefore usable. A message is issued for each valid tape address similar to the following:

```
TAPE 004C OFFLINE
TAPE 004D OFFLINE
```

| If an address is incorrectly defined (for example, the device is not a tape), if the tape drive is not turned on, or if the tape drive has a hardware failure, messages describing the problem are issued. Examples of messages are:

```
TAPE 004C IS NOT A TAPE * * *
TAPE 004C MARKED UNUSABLE
```

Storage Map Generation

At this point, the storage map for the supervisor is printed. If 64K or less storage is available, one partition is available for your programs. If more than 64K is available and the

Messages

address translator feature is installed, multiple partitions can be defined as described in the system configuration statement SYSTEM. The starting address and size of each partition is printed as shown in the following example:

STORAGE MAP			
PART #	START	SIZE	
1	30720	34816	
2	65536	34816	
3	100352	30720	

The supervisor occupies the first 30720 bytes of storage. Three partitions are available for your programs. Partitions 1 and 2 are 34816 bytes long, and partition 3 is 30720 bytes long.

Load Utility Location

At this point, the system locates the load utility program, \$LOADER. If it cannot be found on the IPL volume, the following message is printed:

NO PROGRAM LOAD FACILITY

Sensor I/O Status Check

The system also checks the status of any defined sensor I/O or Binary Synchronous Communications Adapter devices and prints appropriate status messages, for example:

```
SENSOR I/O DEVICE AT ADDRESS 0050 IS OFFLINE  
BSCA NOT THE DEVICE AT ADDR: 0019
```

Date and Time Printing

If timer support was included during system generation, the system prints a message indicating that the date and time can be optionally entered (or reset) using the \$T supervisor utility:

```
SET DATE AND TIME USING COMMAND $T
```

If a system has been defined to include fixed head support for the 4962, a period of about 10 seconds will elapse before the 'Set Date....' message appears. During this time, the fixed head portion of the disk is being initialized.

Normally you enter date and time at IPL time; however, if the system started as a result of an AUTO IPL, the date and time can be entered later.

The supervisor is now ready for operation.

Program Load Message

Any program invoked using \$L (Load a Program) results in the following message being displayed, indicating that the program you requested has been loaded.

```
PROGRAM 15P,14.00.35,LP=4C00
```

Messages

Here, 15P indicates that the program is 15 pages long, where 256 bytes equals one page. 14.00.35 is the time in hours, minutes, and seconds. LP=4C00 indicates that the load point of the program is at location X'4C00'. If timer support is not included in the supervisor, the time is not printed.

ERROR MESSAGES

Program Check Error Message

If a program check occurs during execution of a program, a message with the following format is printed on the loading terminal:

PGM	CHK:	PLP	TCB	PSW	LSB			
		6B00	0138	8002	1E6A	0000	88D0	...

PLP The program load point of the failing program.

TCB The location of the task control block for the failing program (the address appearing on the assembly listing).

PSW The processor status word when the check occurred (described later in this chapter).

LSB Level status block, consisting of the following:

WORD 1	- instruction address register (IAR)
WORD 2	- address key register (AKR)
WORD 3	- level status register (LSR)
WORD 4 - 11	- general registers (R0-R7)

If the program is written in assembler language, COBOL, FORTRAN, or PL/I, the contents of the registers depend upon the conventions unique to that language. If the program is written in Event Driven Language, registers 0 through 7 (words 4-11) contain:

Messages

WORD 4	-	(R0)	work register
WORD 5	-	(R1)	address of Event Driven Executive instruction
WORD 6	-	(R2)	address of EDL TCB
WORD 7	-	(R3)	address of EDL operand 1
WORD 8	-	(R4)	address of EDL operand 2
WORD 9	-	(R5)	EDL command
WORD 10	-	(R6)	work register
WORD 11	-	(R7)	work register

The program in which the error occurred is either aborted or, if it has a task error exit, the exit is entered. In either case, normal system execution is resumed after the program check message has been printed.

System Program Check Error Message

If a program check occurs in the supervisor, the following message prints on the \$SYSLOG terminal:

```
SYSTEM PGM CHK:  PSW AND LSB  
8000 0000 1014 80DP 6F00 6F22 1015 54F5 6F26 805C ...
```

where:

```
WORD 1          - processor status word (PSW)  
WORD 2          - instruction address register (IAR)  
WORD 3          - address key register (AKR)  
WORD 4 - 11     - level status register (LSR)
```

Messages

Processor Status Word (PSW)

Bit	Processor Type 495x			Condition	Class Interrupt	Note
	2	3	5			
00	X	X	X	Specification Check	Program Check	
01	X	X	X	Invalid Storage Addr	Program Check	
02	X	X	X	Privilege Violate	Program Check	
03	X		X	Protect Check	Program Check	
		X		Not Used		1
04	X	X	X	Invalid Function		
					Soft Exception Trap	
05			X	Floating Point Exception	Soft Exception Trap	
	X	X		Not Used		1
06	X	X	X	Stack Exception	Soft Exception Trap	
07	-	-	-	Not Used		1
08	X	X	X	Storage Parity Check	Machine Check	
09	-	-	-	Not Used		1
10	X	X	X	CPU Control Check	Machine Check	
11	X	X	X	I/O Check	Machine Check	
12	X	X	X	Sequence Indicator	None	2
13	X	X	X	Auto IPL	None	2
14	X		X	Translator Enabled	None	
		X		Not Used	-	1
15	X	X	X	Power/Thermal Warning	Power/Thermal	3

Notes:

1. Always Zero
2. Status Flag
3. Controlled by summary mask

\$DUMP Error Messages

Error messages may appear when the \$DUMP utility is active.

The following error message occurs if the data on the data set to be dumped is not the output of \$TRAP:

```
dsname ON dsvol DOES NOT CONTAIN $TRAP OUTPUT
$DUMP TERMINATED
```

where dsname and dsvol are the names of the data set and the volume in which it resides used as input to \$DUMP.

The following error message occurs if an invalid partition number is entered during a partial storage dump:

```
PARTITION NUMBER IS INVALID
```

The following error message occurs if an invalid starting or ending storage address is entered during a partial storage dump:

```
DUMP RANGE INVALID
VALID RANGE IS xxxx TO yyyy
```

where xxxx and yyyy are the valid range of storage addresses.

Messages

\$LOG Error Message

The following error message occurs if the error log becomes full:

\$LOG - *** INSUFFICIENT BUFFERS FOR LOG RATE ***

SRMU Error Messages

The following error messages are issued when the Remote Management Utility encounters an error and are written to the terminal that loaded the utility.

SRMU Error 1

The size of the buffer defined for use by the utility is less than the 512-byte minimum. The default 1024-byte buffer size has been modified incorrectly.

```
SRMU ERROR 1 - INSUFFICIENT BUFFER.  SIZE: nnnn
```

SRMU Error 2

The OPEN of the BSC communications line failed. The return code is defined in the description of the BSC Access Method for the Event Driven Executive.

```
SRMU ERROR 2 - COMMUNICATIONS OPEN FAILED,  
                RETURN CODE: nnnn
```

SRMU Error 3

The CLOSE of a BSC communications line failed. The return code is defined in the description of the BSC Access Method for the Event Driven Executive.

```
SRMU ERROR 3 - COMMUNICATIONS CLOSE FAILED,  
                RETURN CODE: nnnn
```

Messages

\$RMU Error 4

A communications error has been detected by the utility. The I/O function (aaaaaa) indicates the type of request as follows:

```
READ INITIAL
READ CONTINUE
WRITE EOT
WRITE INITIAL
WRITE EOT (ABORT)
WRITE CONTINUE
```

The return code is defined in the description of the BSC Access Method for the Event Driven Executive.

```
$RMU ERROR 4 - COMMUNICATIONS I/O ERROR.
                I/O FUNCTION: aaaaaa
                RETURN CODE: nnnn
```

\$RMU Error 5

The utility attempted to load an overlay program via a LOAD instruction, and the load failed. The return code is issued for the LOAD instruction.

```
$RMU ERROR 5 - LOAD OVERLAY FAILED, RETURN CODE: nnnn
                OVERLAY NUMBER: mmmm
```

\$RMU Error 6

The utility's function table defined a function as being contained within an overlay, but it was not. This error can occur if a user-written function is not added properly to the function table.

```
$RMU ERROR 6 - OVERLAY FUNCTION MISSING. FUNCTION: nnnn
                OVERLAY NUMBER: mmmm
```

\$TRAP Error Messages

Error messages may appear when the \$TRAP utility is active.

The following error message occurs if \$TRAP is loaded into a partition other than partition 1:

```
$TRAP MUST BE IN PARTITION #1
$TRAP TERMINATED
```

The following error message occurs if the data set assigned to \$TRAP is not large enough to contain the amount of storage being saved:

```
dsname ON dsvol IS ONLY xxx RECORDS. MINIMUM SIZE IS yyy
RECORDS
$TRAP TERMINATED
```

The following error message occurs if no trap conditions have been specified:

```
NO TRAP CONDITIONS SPECIFIED
$TRAP TERMINATED
```


Completion Codes

UTILITY COMPLETION CODES

The utility completion codes are printed on the list device by the utility programs upon their completion unless otherwise noted.

\$EDXASM Completion Codes

\$EDXASM completion codes are accompanied by an appropriate error message. The completion codes can be tested by the job stream processor, allowing steps subsequent to the assembly to be skipped, if appropriate. The completion codes are:

Completion Conditions	Completion Code
Successful completion - no errors in assembly.	-1
Successful completion - one or more statements had assembly errors.	8
Out of space in work or object data set.	12
I/O error in source, work, or object data set.	12
Overlay-instruction table full.	12
Unable to locate overlay program or copy code module.	12
Operator cancelled assembly with ATTN CA command.	100

SIAMUT1 Completion Codes

CODE	DESCRIPTION
- 1	Successful completion
01	Data set not found (OPEN failed)
02	Invalid IODA exit (OPEN failed)
03	Volume not mounted (OPEN failed)
04	Library not found (OPEN failed)
05	Disk I/O error (OPEN failed)
06	No VTOC exit address (OPEN failed)
07	Link module in use
08	Load error for \$IAM
12	Data set shut down
13	Module not included in load module
23	Get storage error - IACB
31	FCB WRITE error during IDEF processing
32	Blocksize not multiple of 256
33	Data set is greater than 32,767 Event Driven Executive records
34	Data set is too small
36	Invalid block size during file definition processing
37	Invalid record size
38	Invalid index size
39	Record size greater than block size
40	Invalid number of free records
41	Invalid number of clusters
42	Invalid key size
43	Invalid reserve index value
44	Invalid reserve block value
45	Invalid free pool value
46	Invalid delete threshold value
47	Invalid free block value
48	Invalid number of base records
49	Invalid key position
50	Data set is opened for exclusive use
51	Data set opened in load mode
52	Data set is opened, cannot be opened exclusively
54	Invalid block size during PROCESS or LOAD
55	Get storage for FCB error

Completion Codes

CODE	DESCRIPTION
56	FCB READ error
60	LOAD mode key is equal to or less than previous high key in data set
61	End of file
62	Duplicate key found
100	READ error
101	WRITE error
110	WRITE error - data set closed

\$JOBUTIL Completion Codes

The \$JOBUTIL completion codes are displayed on the terminal used to access \$JOBUTIL. The codes are as follows.

Code	Description
-1	Successful completion
61	The transient loader (\$LOADER) is not included in the system
64	No space available for the transient loader
67	A disk or diskette I/O error occurred during the load process
70	Not enough main storage available for the program
71	Program not found on the specified volume
72	Disk or diskette I/O error while reading directory
73	Disk or diskette I/O error while reading program header
74	Referenced module is not a program
75	Referenced module is not a data set
76	Data set not found on referenced volume
77	Invalid data set name
78	LOAD instruction did not specify required data set(s)
79	LOAD instruction did not specify required parameter(s)
80	Invalid volume label specified; for example, greater than eight characters

Completion Codes

\$LINK Completion Codes

Message number	Description	Cause code	Action code	Return code
-	Successful completion	-	-	-1
01	DS2 less than 265 records	1	2	12
02	Disk error reading DS1	2	2	12
03	End of file reached on DS1	1	3	4
04	Disk error reading object module	2	1	8
05	Invalid 'OUTPUT' record	1	2	12
06	Invalid 'INCLUDE' record	1	6	8
07	Error opening object output module:	1	5	12
	- misspelled name or volume			
	- data set not allocated			
08	Error opening input object module (see Error 07)	1	6	8
09	Error opening output module (hardware error)	2	5	12
10	Error opening an input module (hardware error)	2	6	8
11	Error opening autocall list (DS9). See Error 07 for causes	1	5	12
12	Error opening autocall list (DS9) (hardware error)	2	5	12
13	Invalid input object module record type	4	4	8
14	Entry point label not found	1	3	4
15	No valid ESDID for TXT or RLD	4	4	8
16	Invalid ESD item type	4	4	8
17	Duplicate ESDID number	4	4	8
18	Invalid Symbol	4	4	8
19	Duplicate Entry point symbol	3	4	8
20	Invalid ESDID number	4	4	8
22	Invalid ESD symbol	4	1	8
23	End of file reached on DS9	1	2	12
24	Disk error reading DS9	2	2	12

Completion Codes

Message number	Description	Cause code	Action code	Return code
25	Disk error reading DS4	2	2	12
26	End of File reached on DS3	6	2	12
27	Disk error Read/Write on DS8	2	2	12
28	End of file reached on DS8	5	2	12
29	End of file reached on DS7	6	2	12
30	End of file reached on DS4	6	2	12
31	Disk error writing on DS5	2	2	12
32	End of file reached on DS5	5	2	12
33	End of file reached on DS2	6	2	12
34	Duplicate section definition (CSECT)	3	1	4
36	End of file reached on DS6	4	1	8
37	Disk error, read/write on DS7	2	2	12
38	Disk error, read/write on DS3	2	2	12
39	Invalid RLD record data length	4	4	8
40	Disk error, read/write on DS2	2	2	12
42	DS2 not large enough (program size over 64K)	5	2	12
45	No 'INCLUDE' records	1	2	12
46	No CSECT length field	4	3	4
None	Unresolved EXTRN			4

Completion Codes

Cause Codes

- 1 - Your error
- 2 - System error
- 3 - Possible duplicate 'name,volume' or duplicate CSECT or ENTRY names
- 4 - Input object record(s) in error. Probable cause is that 'name,volume' is not a valid object module
- 5 - Data set is of insufficient size
- 6 - Probable \$LINK error, this condition should not occur

Action Codes

- 1 - Log warning message and continue at next 'INCLUDE'
- 2 - Terminate \$LINK with error message
- 3 - Continue as if expected occurrence had happened
- 4 - Log error message plus invalid object module record and continue at next 'INCLUDE'
- 5 - Log error message plus OUTPUT record and terminate \$LINK
- 6 - Log error message plus INCLUDE record, continue at next 'INCLUDE'

Return Code Definitions

- 1 Successful completion
- 4 Warning: A module has been written - execution will probably work
- 8 Warning: A module has been written - execution will probably fail
- 12 Severe error: Module is not written

\$UPDATE Completion Codes

The \$UPDATE completion codes are displayed on the terminal used to access \$UPDATE. The codes are as follows.

Code	Condition
-1	Successful completion
8	No supervisor space in this library
8	Output name specified is not a program
8	Disk volume already in use by another program
8	No space in directory
8	No space in data set (output library)
8	Invalid header format
8	Invalid program name
8	Disk volume not mounted
8	Disk volume off line
8	Library not found
8	Input data set not found
8	No parameter supplied via \$JOBUTIL
8	No data set names provided via \$JOBUTIL
8	Replacement of output data set not allowed
12	Any disk or diskette I/O errors

Return Codes

EVENT DRIVEN LANGUAGE AND FUNCTION RETURN CODES

The Event Driven Language return codes are returned in the first word of the Task Control Block of the program which invoked the respective language instruction or function unless otherwise noted.

SDISKUT3 Return Codes

The SDISKUT3 utility places a return code in the first word of the DSCB specified. The return codes for SDISKUT3 are listed below.

Code	Condition
1	Invalid request code parameter (not 1-5)
2	Volume does not exist (All functions)
4	Insufficient space in library (ALLOCATE)
5	Insufficient space in directory (ALLOCATE)
6	Data set already exists - smaller than the requested allocation
7	Insufficient contiguous space (ALLOCATE)
8	Disallowed data set name, eg. \$EDXVOL or \$EDXLIB (All functions)
9	Data set not found (DELETE, OPEN, RELEASE, RENAME)
10	New name pointer is zero (RENAME)
11	Disk is busy (ALLOCATE, DELETE, RELEASE, RENAME)
12	I/O error writing to disk (ALLOCATE, DELETE, RELEASE, RENAME)
13	I/O error reading from disk (All functions)
14	Data set name is all blanks (ALLOCATE, RENAME)
15	Invalid size specification (ALLOCATE)
16	Invalid size specification (RELEASE)
17	Mismatched data set type (DELETE, OPEN, RELEASE, RENAME)
18	Data set already exists - larger than the requested allocation
19	SETEOD only valid for data set of type 'data'
20	Load of SDISKUT3 failed (\$RMU only)
21	Tape data sets are not supported

\$PDS Return Codes

Use the EVENT=parameter on the LOAD command when using the \$PDS utility. The \$PDS utility returns the status of the request in the Event Control block specified in the LOAD command. The return codes for \$PDS are listed below.

Code	Condition
-1	Successful operation
1	Member not found
2	Member already allocated
3	No space
4	Directory is full
5	Member was not used
7	Record not in member
8	Member control block invalid
9	Space not released
10	Not a data member

Return Codes

BSC Return Codes

Code	Description	Notes
-2	Text received in conversational mode	
-1	Successful completion	
END=		
1	EOT received	
2	DLE EOT received	
3	Reverse interrupt received	
4	Forward abort received	
5	Remote station not ready (NAK received)	4
6	Remote station busy (WACK received)	4
ERROR=		
10	Timeout occurred	1
11	Unrecovered transmission error (BCC error)	1
12	Invalid sequence received	3
13	Invalid multi-point tributary write attempt	2
14	Disregard this block sequence received	1
15	Remote station busy (WACK received)	1
20	Wrong length record - long (No COD)	6
21	Wrong length record - short (write only)	2
22	Invalid buffer address	2
23	Buffer length zero	2
24	Undefined line address	2
25	Line not opened by calling task	2
30	Modem interface error	2
31	Hardware overrun	2
32	Hardware error	5
33	Unexpected ring interrupt	2
34	Invalid interrupt during auto-answer attempt	2
35	Enable or disable DTR error	2
99	Access method error	2

Notes:

- Retried up to the limit specified in the RETRIES= operand of the BSCLINE definition.

2. Not retried.
3. Retried during write operation only when a wrong ACK is received following an ENQ request after timeout (indicating that no text had been received at the remote station).
4. Returned only during an initial sequence with no retry attempted.
5. Retried only after an unsuccessful start I/O attempt.
6. Retried only during read operations.

Data Formatting Return Codes

Code	Description
-1	Successful completion
1	No data in field
2	Field omitted
3	Conversion error

Return Codes

EXIO Return Codes

I/O Instruction Return Codes (word 0 of TCB)
(Word 1 of TCB contains supervisor instruction
address)

Code	Description
------	-------------

-1	Command accepted
1	Device not attached
2	Busy
3	Busy after reset
4	Command reject
5	Intervention required
6	Interface data check
7	Controller busy
8	Channel command not allowed
9	No DDB found
10	Too many DCBs chained
11	No address specified for residual status
12	EXIODEV specified zero bytes for residual status
13	Broken DCB chain (program error)
16	Device already opened

Interrupt Condition Codes (Bits 4-7 of word 0 of ECB)
(If bit 0 is on, bits 8-15=device ID)

Code	Description
0	Controller end
1	Program Controlled Interrupt (PCI)
2	Exception
3	Device end
4	Attention
5	Attention and PCI
6	Attention and exception
7	Attention and device end
8	Not used
9	Not used
10	SE on and too many DCBs chained
11	SE on and no address specified for residual status
12	SE on and EXIODEV specified no bytes for residual status
13	Broken DCB chain
14	ECB to be posted not reset
15	Error in Start Cycle Steal Status (after exception)

Return Codes

Floating Point Return Codes

Code	Description
-1	Successful completion
1	Floating point overflow
3	Floating point divide check (divide by '0')
5	Floating point under flow

Formatted Screen Image Return Codes

These return codes are issued by the \$IMOPEN subroutine. They are returned in the second word of the Task Control Block of the calling program.

Code	Description
-1	Successful completion
1	Disk I/O error
2	Invalid data set name
3	Data set not found
4	Incorrect header or data set length
5	Input buffer too small

Indexed Access Method Return Codes

CODE	DESCRIPTION
- 1	Successful completion
- 57	Data set has been loaded
- 58	Record not found
- 80	End of data
- 85	Record to be deleted not found
01	Function code not recognized
07	Link module in use
08	Load error for \$IAM
10	Invalid request
12	Data set shut down due to error
13	Module not included in load module
22	Invalid IACB address
23	Get storage error - IACB
50	Data set is opened for exclusive use, cannot be opened exclusively
51	Data set opened in load mode
52	Data set is opened, cannot be opened exclusively
54	Invalid block size during PROCESS or LOAD processing
55	Get storage error - FCB
56	READ error - FCB
60	Out of sequence or duplicate key
61	End of file
62	Duplicate key found in PROCESS mode
70	No space for insert
80	FCB WRITE error during DELETE processing
85	Key field modified by user
90	Key save area in use
100	READ error
101	WRITE error
110	WRITE error - data set closed

Return Codes

LOAD Return Codes

Code	Description
-1	Successful completion
61	The transient loader (\$LOADER) is not included in the system
62	In an overlay request, no overlay area exists
63	In an overlay request, the overlay area is in use
64	No space available for the transient loader
65	In an overlay load operation, the number of data sets passed by the LOAD instruction does not equal the number required by the overlay program
66	In an overlay load operation, no parameters were passed to the loaded program
67	A disk(ette) I/O error occurred during the load process
68	Reserved
69	Reserved
70	Not enough main storage available for the program
71	Program not found on the specified volume
72	Disk(ette) I/O error while reading directory
73	Disk(ette) I/O error while reading program header
74	Referenced module is not a program
75	Referenced module is not a data set
76	Data set not found on referenced volume
77	Invalid data set name
78	LOAD instruction did not specify required data set(s)
79	LOAD instruction did not specify required parameters(s)
80	Invalid volume label specified
81	Cross partition LOAD requested, support not included at system generation
82	Requested partition number greater than number of partitions in the system

Note: If the program being loaded is a sensor I/O program and a sensor I/O error is detected, the return code will be a sensor I/O return code, not a load return code.

Multiple Terminal Manager Return Codes

These return codes are returned in a caller-specified variable on the SETPAN or FILEIO function.

CODE	DESCRIPTION
-1	Successful completion
1	Warning: uninitialized panel. Input buffer has been set to unprotected blanks (x'00') and cursor position set to zero.
2	Data table truncated.
201	Data set not found
202	Volume not found
203	No file table entries are available; all have updates outstanding
204	I/O error reading volume directory
205	I/O error writing volume directory
206	Invalid function request
207	Invalid key operator
208	SEOD record number greater than data set length
-500	Terminal is not an IBM 4978/4979. No action has been taken
-501	Screen data set not found.
other	Return code from READ/WRITE or the Indexed Access Method

Return Codes

READ/WRITE Return Codes

Disk/tape return codes resulting from READ/WRITE instructions are returned in two places:

1. The Event Control Block (ECB) named DS n , where n is the number of the data set being referenced.
2. The task code word referred to by taskname.

The disk/tape return codes and their meanings are shown below.

If further information concerning an error is required, it may be obtained by printing all or part of the contents of the Disk Data Blocks (DDBs) located in the Supervisor. The starting address of the DDBs can be obtained from the linkage editor map of the supervisor. The contents of the DDBs are described in the Internal Design. Of particular value are the Cycle Steal Status Words and the Interrupt Status Word save areas, along with the contents of the word that contains the address of the next DDB in storage.

Disk Return Codes

Code	Description
-1	Successful completion
1	I/O error and no device status present (this code may be caused by the I/O area starting at an odd byte address)
2	I/O error trying to read device status
3	I/O error retry count exhausted
4	Error on issuing I/O instruction to read device status
5	Unrecoverable I/O error
6	Error on issuing I/O instruction for normal I/O
7	A 'no record found' condition occurred, a seek for an alternate sector was performed, and another 'no record found' occurred i.e., no alternate is assigned
9	Device was 'offline' when I/O was requested
10	Record number out of range of data set--may be an end-of-file (data set) condition
11	Device marked 'unusable' when I/O was requested

Return Codes

Tape Return Codes

Code	Description
-1	Successful completion
1	Exception but no status
2	Error reading STATUS
4	Error issuing STATUS READ
5	Unrecoverable I/O error
6	Error issuing I/O command
10	Tape mark (EOD)
20	Device in use or offline
21	Wrong length record
22	Not ready
23	File protect
24	EOT
25	Load point
26	Uncorrected I/O error
27	Attempt WRITE to unexpired data set
28	Invalid blksize
29	Data set not open
30	Incorrect device type
31	Incorrect request type on close request
32	Block count error during close
33	EOV1 label encountered during close
76	DSN not found

SPIO (Sensor-based I/O) Return Codes

Code	Description
-1	Successful completion
90	Device not attached
91	Device busy or in exclusive use
92	Busy after reset
93	Command reject
94	Invalid request
95	Interface data check
96	Controller busy
97	Analog Input over voltage
98	Analog Input invalid range
100	Analog Input invalid channel
101	Invalid count field
102	Buffer previously full or empty
104	Delayed command reject

Return Codes

Terminal I/O Return Codes

Code	Description
-1	Successful completion
1	Device not attached
2	System error (busy condition)
3	System error (busy after reset)
4	System error (command reject)
5	Device not ready
6	Interface data check
7	Overrun received
11	Codes greater than 10 represent possible multiple errors. To determine the errors, subtract 10 from the code and express the result as an 8-bit binary value. Each bit (numbering from the left) represents an error as follows:
Bit	Description
0	Unused
1	System error (command reject)
2	Not used
3	System error (DCB specification check)
4	Storage data check
5	Invalid storage address
6	Storage protection check
7	Interface data check

Note: If an error message code greater than 128 is returned for devices supported by IOS2741 (2741,PROC), subtract 128; the result then contains status word 1 of the ACCA. (Refer to Communication Features Description to determine the special error condition.)

Terminal I/O - ACCA Return Codes

-1	Successful completion.
Bit	Description
0	Unused
1-8	ISB of last operation (I/O complete)
9-10	Unused
11	1 if a write or control operation (I/O complete)
12	Read operation (I/O complete)
13	Unused
14-15	Condition code +1 after I/O start (or) Condition code after I/O complete

Return Codes

Terminal I/O - Interprocessor Communications Return Codes

	CODTYPE=		Return Codes
	EBCD/CRSP	EBCDIC	
End of Transmission (EOT).	1F	FDFE	-2
End of Record (NL).	5B	FEFF	-1
End of Subrecord (EOSR).	Not used.	FCFF	Handled by device support.

Terminal I/O - Virtual Terminal Communications Return Codes

Value	Transmit	Receive
x'8Fnn'	NA	LINE=nn received
x'8Enn'	NA	SKIP=nn received
-2	NA	Line received (no CR)
-1	Successful completion	New line received
1	Not attached	Not attached
5	Disconnect	Disconnect
8	Break	Break

LINE=nn (x'8Fnn'): This code is posted for READTEXT or GETVALUE instructions if the other side sent the LINE forms control operation; it is transmitted so that the receiving program may reproduce on a real terminal (for printer spooling applications for example) the output format intended by the sending program.

SKIP=nn (x'8Enn'): The sending program transmitted SKIP=nn.

Line Received (-2): This code indicated that the sending program did not send a new line indication, but that the line was transmitted because of execution of a control operation or a transition to the read state. This is how, for example, a prompt message is usually transmitted with READTEXT or GETVALUE.

New Line Received (-1): This code indicates transmission of the carriage return at the end of the data. The distinction between a new line transmission and a simple line transmission is, again, made only to allow preservation of the original output format.

Not attached (1): If the virtual terminal accessed for the operation does not reference another virtual terminal, then this code is returned.

Disconnect (5): This code value corresponds to the 'not ready' indication for real terminals; its specific meaning for virtual terminals is that the program at the other end of the channel terminated either through PROGSTOP or operator intervention.

Return Codes

Break (8): The break code indicates that the other side of the channel is in a state (transmit or receive) which is incompatible with the attempted operation. If only one end of the channel is defined with SYNC=YES (on the TERMINAL statement), then the task on that end will always receive the break code, whether or not it attempted the operation first. If both ends are defined with SYNC=YES, then the code will be posted to the task which last attempted the operation. The break code may thus be understood as follows: when reading (READTEXT or GETVALUE), the other program has stopped sending and is waiting for input; when writing (PRINTTEXT or PRINTNUM), the other program is also attempting to write. Note that current Event Driven Executive programs, or future programs which do not interpret the break code, must always communicate through a virtual terminal which is defined with SYNC=NO (the default).

TP Return Codes

Code	Description	Module
-1	Successful completion	Supervisor
1	Illegal command sequence	Supervisor
2	TP I/O error	Supervisor
3	TP I/O error on host	HCFCOMM
4	Looping bidding for the line	Supervisor
5	Host acknowledgement to request code was neither ACK0,ACK1,WACK, or a NACK	Supervisor
6	Retry count exhausted - last error was a timeout: the host must be down	Supervisor
7	Looping while reading data from the host	Supervisor
8	The host responded with other than an 'EOT' or an 'ENQ' when an 'EOT' was expected	Supervisor
9	Retry count exhausted - last error was a 'modem interface check'	Supervisor
10	Retry count exhausted - last error was not a timeout,modem check, block check or overrun	Supervisor
11	Retry count exhausted - last error was a transmit overrun	Supervisor
50	I/O error from last I/O in DSWRITE.	DSCLOSE
51	I/O error when writing the last buffer	DSCLOSE
100	Length of DSNAME is zero	HCFCOMM
101	Length of DSNAME exceeds 52	HCFCOMM
102	Invalid length specified for I/O	HCFINIT

Return Codes

Code	Description	Module
200	Data set not on volume specified for controller	HCFINIT
201	Invalid member name specification	DSOPEN
202	Data set in use by another job	DSOPEN
203	Data set already allocated to this task	DSOPEN
204	Data set is not cataloged	DSOPEN
205	Data set resides on multiple volumes	DSOPEN
206	Data set is not on a direct access device	DSOPEN
207	Volume not mounted (archived)	DSOPEN
208	Device not online	DSOPEN
209	Data set does not exist	DSOPEN
211	Record format is not supported	DSOPEN
212	Invalid logical record length	DSOPEN
213	Invalid block size	DSOPEN
214	Data set has no extents	DSOPEN
216	Data set organization is partitioned and no member name was specified	DSOPEN
217	Data set organization is sequential and a member name was specified	DSOPEN
218	Error during OS/ OPEN	DSOPEN
219	The specified member was not found	DSOPEN
220	An I/O error occurred during a directory search	DSOPEN
221	Invalid data set organization	DSOPEN
222	Insufficient I/O buffer space available	DSOPEN
300	End of an input data set	DSREAD
301	I/O error during an OS/ READ	DSREAD
302	Input data set is not open	DSREAD
303	A previous error has occurred	DSREAD

Return Codes

Code	Description	Module
400	End of an output data set	DSWRITE
401	I/O error during an OS/ WRITE	DSWRITE
402	Output data set is not open	DSWRITE
403	A previous error has occurred	DSWRITE
404	Partitioned data set is full	DSCLOSE
700	Index, key, and status record added	SET
701	Index exists, key and status added	SET
702	Index and key exist, status replaced	SET
703	Error - Index full	SET
704	Error - Data set full	SET
710	I/O Error	SET
800	Index and key exist	FETCH
801	Index does not exist	FETCH
802	Key does not exist	FETCH
810	I/O error	FETCH
900	Index and/or key released	RELEASE
901	Index does not exist	RELEASE
902	Key does not exist	RELEASE
910	I/O error	RELEASE
1xxx	An error occurred in a subordinate module during SUBMIT. 'xxx' is the code returned by that module.	S7SUBMIT



EVENT DRIVEN EXECUTIVE LIBRARY SUMMARY

The library summary is a guide to the Event Driven Executive library. By briefly listing the content of each book and providing a suggested reading sequence for the library, it should assist you in using the library as a whole as well as direct you to the individual books you require.

Event Driven Executive Library

The IBM Series/1 Event Driven Executive library materials consist of five full-sized books, a quick reference pocket book, and a set of tabs:

- IBM Series/1 Event Driven Executive System Guide (or System Guide), SC34-0312
- IBM Series/1 Event Driven Executive Utilities, Operator Commands, Program Preparation, Messages and Codes (or Utilities), SC34-0313
- IBM Series/1 Event Driven Executive Language Reference (or Language Reference), SC34-0314
- IBM Series/1 Event Driven Executive Communications and Terminal Application Guide (or Communications Guide), SC34-0316
- IBM Series/1 Event Driven Executive Internal Design (or Internal Design), LY34-0168
- IBM Series/1 Event Driven Executive Multiple Terminal Manager Internal Design (or Multiple Terminal Manager Internal Design), LY34-0190
- IBM Series/1 Event Driven Executive Indexed Access Method Internal Design (or Indexed Access Method Internal Design), LY34-0189
- IBM Series/1 Event Driven Executive Reference Summary (or Reference Summary), SX34-0101
- IBM Series/1 Event Driven Executive Tabs (or Tabs), SX34-0030

Summary of Library

System Guide

The System Guide introduces the concepts and capabilities of the Event Driven Executive system. It discusses multi-tasking, program and task structure, program overlays, storage management, and data management.

Planning aids include hardware and software requirements, along with guidelines for storage estimating.

The System Guide also presents step-by-step procedures for generating a supervisor tailored to your Series/1 hardware configuration and software needs.

The description of the Indexed Access Method contains the information on how to write applications that use indexed data sets.

The description of the session manager includes a procedure for modifying the session manager to include application programs in the primary option menu so that you can execute them under the session manager. You can also add a procedure to compile, link, and update programs.

Information is also provided concerning partitioned data sets, tape data organization, diagnostic aids, inter-program communication, logical screens, and dynamic data set allocation.

Utilities

Utilities describes:

- Event Driven Executive utility programs
- Operator commands
- Procedures to prepare and execute system and application programs
- The session manager -- a menu-driven interface program that will invoke the programs required for program development
- Messages and codes issued by the Event Driven Executive system

The operator commands, program preparation facilities, and session manager are grouped by function and discussions include detailed syntax and explanations. The utilities are presented in alphabetical order.

Language Reference

The Language Reference familiarizes you with the Event Driven Language by first grouping the instructions into functional categories. Then the instructions are listed alphabetically, with complete syntax and an explanation of each operand.

The final section of the Language Reference contains examples of using the Event Driven language for applications such as:

- Program loading
- User exit routine
- Graphics
- I/O level control program
- Indexing and hardware register usage

Communications Guide

The Communications Guide introduces the Event Driven Executive communications support -- binary synchronous communications, asynchronous communications, and the Host Communications Facility.

The Communications Guide contains coding details for all utilities and Event Driven language instructions needed for communications support and advanced terminal applications.

Internal Design

Internal Design describes the internal logic flow and specifications of the Event Driven Executive system so that you can understand how the system interfaces with application programs. It familiarizes you with the design and implementation by describing the purpose, function, and operation of the various Event Driven Executive system programs.

Multiple Terminal Manager Internal Design and Indexed Access Method Internal Design describe the internal logic flow and specifications of these programs.

Unlike the other manuals in the library, the Internal Design books contain material that is the licensed property of IBM and they are available only to licensed users of the Event Driven Executive system.

Reference Summary

The Reference Summary is a pocket-sized booklet to be used for quick reference. It lists the Event Driven language instructions with their syntax, the utility and program preparation commands, and the completion codes.

Tabs

The tabs package must be ordered separately. The package contains 33 index tabs by subject, with additional blank tabs. These extended tabular pages can be inserted at the front of various sections of the library. The tabs are color coded according to the major library topics.

Reading Sequence

All readers of the Event Driven Executive library should begin with the first three chapters of the System Guide ("Introduction," "The Supervisor and Emulator," and "Data Management") for an overview of the Event Driven Executive concepts and facilities.

Readers responsible for installing and preparing the system should then continue in the System Guide with "System Configuration" and "System Generation."

All readers should review the Utilities "Introduction" to become familiar with the utility functions available for the Event Driven Executive system. Then you can read more specific sections for particular utilities, operator commands, and program preparation facilities.

After you have a basic understanding of the Event Driven Executive system and how you can best use the system for your applications, you should read the Language Reference "Introduction." This will familiarize you with the potential

of the Event Driven Language and prepare you to start coding application programs.

If you have communications support for your Event Driven Executive system, you should read the Communications Guide, which is an extension of the System Guide, Utilities, and the Language Reference.

After you know the functions of the various Event Driven Language instructions, utilities, and program preparation facilities, you may wish to refer only to the Reference Summary for correct syntax while coding your applications.

Only readers responsible for the support or modification of the Event Driven Executive system need to read Internal Design.

OTHER EVENT DRIVEN EXECUTIVE PROGRAMMING PUBLICATIONS

- IBM Series/1 Event Driven Executive FORTRAN IV User's Guide, SC34-0315.
- IBM Series/1 Event Driven Executive PL/I Language Reference, GC34-0147.
- IBM Series/1 Event Driven Executive PL/I User's Guide, GC34-0148.
- IBM Series/1 Event Driven Executive COBOL Programmer's Guide, SL23-0014.
- IBM Series/1 Event Driven Executive Sort/Merge Programmer's Guide, SL23-0016
- IBM Series/1 Event Driven Executive Macro Assembler Reference, GC34-0317.
- IBM Series/1 Event Driven Executive Study Guide, SR30-0436.

OTHER SERIES/1 PROGRAMMING PUBLICATIONS

- IBM Series/1 Programming System Summary, GC34-0285.
- IBM Series/1 COBOL Language Reference, GC34-0234.
- IBM Series/1 FORTRAN IV Language Reference, GC34-0133.

- IBM Series/1 Host Communications Facility Program Description Manual, SH20-1819.
- IBM Series/1 Mathematical and Functional Subroutine Library User's Guide, SC34-0139.
- IBM Series/1 Macro Assembler Reference Summary, SX34-0128.
- IBM Series/1 Data Collection Interactive Programming RPQ P82600 User's Guide, SC34-1654.

OTHER PROGRAMMING PUBLICATIONS

- IBM Data Processing Glossary, GC20-1699.
- IBM Series/1 Graphic Bibliography, GA34-0055.
- IBM OS/VS Basic Telecommunications Access Method (BTAM), GC27-6980.
- General Information - Binary Synchronous Communications, GA27-3004.
- IBM System/370 Program Preparation Facility, SB30-1072.

SERIES/1 SYSTEM LIBRARY PUBLICATIONS

- IBM Series/1 4952 Processor and Processor Features Description, GA34-0084.
- IBM Series/1 4953 Processor and Processor Features Description, GA34-0022.
- IBM Series/1 4955 Processor and Processor Features Description, GA34-0021.
- IBM Series/1 Communications Features Description, GA34-0028.
- IBM Series/1 3101 Display Terminal Description, GA34-2034.
- IBM Series/1 4962 Disk Storage Unit and 4964 Diskette Unit Description, GA34-0024.
- IBM Series/1 4963 Disk Subsystem Description, GA34-0051.
- IBM Series/1 4966 Diskette Magazine Unit Description, GA34-0052.

- IBM Series/1 4969 Magnetic Tape Subsystem Description, GA34-0087.
- IBM Series/1 4973 Line Printer Description, GA34-0044.
- IBM Series/1 4974 Printer Description, GA34-0025.
- IBM Series/1 4978-1 Display Station (RPQ D02055) and Attachment (RPQ D02038) General Information, GA34-1550
- IBM Series/1 4978-1 Display Station, Keyboard (RPQ D02056) General Information, GA34-1551
- IBM Series/1 4978-1 Display Station, Keyboard (RPQ D02057) General Information, GA34-1552
- IBM Series/1 4978-1 Display Station Keyboards (RPQ D02064 and D02065) General Information, GA34-1553
- IBM Series/1 4979 Display Station Description, GA34-0026
- IBM Series/1 4982 Sensor Input/Output Unit Description, GA34-0027
- IBM Series/1 Data Collection Interactive RPQs D02312, D02313, and D02314 Custom Feature, GA34-1567



This glossary contains terms that are used in the Series/1 Event Driven Executive software publications. All software and hardware terms are Series/1 oriented. This glossary defines terms used in this library and serves as a supplement to the IBM Data Processing Glossary (GC20-1699).

\$\$SYSLOGA. The name of the alternate system logging device. This device is optional but, if defined, should be a terminal with keyboard capability, not just a printer.

\$\$SYSLOG. The name of the system logging device or operator station; must be defined for every system. It should be a terminal with keyboard capability, not just a printer.

\$\$SYSPRTR. The name of the system printer.

ACCA. See asynchronous communications control adapter.

address key. Identifies a set of Series/1 segmentation registers and represents an address space. It is one less than the partition number.

address space. The logical storage identified by an address key. An address space is the storage for a partition.

application program manager. The component of the Multiple Terminal Manager that provides the program management facilities required to process user requests. It controls the contents of a program area and the execution of programs within the area.

application program stub. A collection of subroutines that are appended to a program by the link-age editor to provide the link from the application program to

the Multiple Terminal Manager facilities.

asynchronous communications control adapter. An ASCII terminal attached via #1610, #2091 with #2092, or #2095 with #2096 adapters.

attention list. A series of pairs of 1 to 8 byte EBCDIC strings and addresses pointing to EDL instructions. When the attention key is pressed on the terminal, the operator can enter one of the strings to cause the associated EDL instructions to be executed.

backup. A copy of data to be used in the event the original data is lost or damaged.

base records. Records that have been placed into an indexed data set while in load mode.

basic exchange format. A standard format for exchanging data on diskettes between systems or devices.

binary synchronous device data block (BSCDDB). A control block that provides the information to control one Series/1 Binary Synchronous Adapter. It determines the line characteristics and provides dedicated storage for that line.

block. (1) See data block or index block. (2) In the Indexed Method, the unit of space used by the access method to contain indexes and data.

BSCDDB. See binary synchronous device data block.

buffer. An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. See input buffer and output buffer.

bypass label processing. Access of a tape without any label processing support.

CCB. See terminal control block.

character image. An alphabetic, numeric, or special character defined for an IBM 4978 Display Station. Each character image is defined by a dot matrix that is coded into eight bytes.

character image table. An area containing the 256 character images that can be defined for an IBM 4978 Display Station. Each character image is coded into eight bytes, the entire table of codes requiring 2048 bytes of storage.

cluster. In an indexed file, a group of data blocks that is pointed to from the same primary-level index block, and includes the primary-level index block. The data records and blocks contained in a cluster are logically contiguous, but are not necessarily physically contiguous.

COD (change of direction). A character used with ACCA terminal to indicate a reverse in the direction of data movement.

command. A character string from a source external to the system that represents a request for action by the system.

common area. A user-defined data area that is mapped into every partition at the same address. It

can be used to contain control blocks or data that will be accessed by more than one program.

completion code. An indicator that reflects the status of the execution of a program. The completion code is displayed or printed on the program's output device.

conversion. See update.

cross partition service. A function that accesses data in two partitions.

data block. In an indexed file, an area that contains control information and data records. These blocks are a multiple of 256 bytes.

data set. A group of contiguous records within a volume pointed to by a directory member entry in the directory for the volume.

data set control block (DSCB). A control block that provides the information required to access a data set, volume or directory using READ and WRITE.

data set shut down. An indexed data set that has been marked (in main storage only) as unusable due to an error.

DCE. See directory control entry.

DDB. See disk data block.

direct access. (1) The access method used to READ or WRITE records on a disk or diskette device by specifying their location relative the beginning of the data set or volume. (2) In the Indexed Access Method, locating any record via its key without respect to the previous operation.

directory. A series of contiguous records in a volume that describe the contents in terms of allocated data sets and free spaces.

directory control entry (DCE). The first 32 bytes of the first record of a directory in which a description of the directory is stored.

directory member entry (DME). A 32-byte directory entry describing an allocated data set.

disk data block (DDB). A control block that describes a direct access volume.

display station. An IBM 4978 or 4979 display terminal or similar terminal with a keyboard and a video display.

DME. See directory member entry.

DSCB. See data set control block.

dynamic storage. An increment of storage that is appended to a program when it is loaded.

end-of-data indicator. A code that signals that the last record of a data set has been read or written. End-of-data is determined by an end-of-data pointer in the DME or by the physical end of the data set.

ECB. See event control block.

EDL. See Event Driven Language.

emulator. The portion of the Event Driven Executive supervisor that interprets EDL instructions and performs the function specified by each EDL statement.

end-of-tape (EOT). A reflective marker placed near the end of a tape and sensed during output. The marker signals that the tape is nearly full.

event control block (ECB). A control block used to record the status (occurred or not occurred) of an event; often used to synchronize the execution of tasks. ECBs are used in conjunction with the WAIT and POST instructions.

event driven language (EDL). The language for input to the Event Driven Executive compiler (\$EDXASM), or the Macro and Host assemblers in conjunction with the Event Driven Executive macro libraries. The output is interpreted by the Event Driven Executive emulator.

EXIO (execute input or output). An EDL facility that provides user controlled access to Series/1 input/output devices.

external label. A label attached to the outside of a tape that identifies the tape visually. It usually contains items of identification such as file name and number, creation data, number of volumes, department number, and so on.

external name (EXTRN). The 1- to 8-character symbolic EBCDIC name for an entry point or data field that is not defined within the module that references the name.

FCA. See file control area.

FCB. See file control block.

file control area (FCA). A Multiple Terminal Manager data area that describes a file access request.

file control block (FCB). In an indexed data set, the first block of the data set. It contains descriptive information about the data contained in the data set.

file manager. A collection of subroutines contained within the program manager of the Multiple Terminal Manager that provides common support for all disk data transfer operations as needed for transaction-oriented application programs. It supports indexed and direct files under the control of a single callable function.

formatted screen image. A collection of display elements or display groups (such as operator prompts and field input names and areas) that are presented together at one time on a display device.

free pool. In an indexed data set, a group of blocks that can be used as either a data block or an index block. These differ from other free blocks in that these are not initially assigned to specific logical positions in the data set.

free space. In the Indexed Access Method, record spaces or blocks that do not currently contain data, and are available for use.

free space entry (FSE). A 4-byte directory entry defining an area of free space within a volume.

FSE. See free space entry.

hardware timer. The timer features available with the Series/1 processors. Specifically, the 7840 Timer Feature card or the native timer (4952 only). Only one or the other is supported by the Event Driven Executive.

host assembler. The assembler licensed program that executes in a 370 (host) system and produces object output for the Series/1. The source input to the host assembler is coded in Event Driven Language or Series/1 assembler language. The host assembler

refers to the System/370 Program Preparation Facility (5798-NNQ).

host system. Any system whose resources are used to perform services such as program preparation for a Series/1. It can be connected to a Series/1 by a communications link.

IACB. See indexed access control block.

IAR. See instruction address register.

ICB. See indexed access control block.

IIB. See interrupt information byte.

image store. The area in a 4978 that contains the character image table.

index. In the Indexed Access Method, an ordered collection of pairs, each consisting of a key and a pointer, used to sequence and locate the records in an Indexed Access Method data set.

index block. In an indexed file, an area that contains control information and index entries. These blocks are a multiple of 256 bytes.

indexed access control block (IACB/ICB). The control block that relates an application program to an indexed data set.

indexed access method. An access method for direct or sequential processing of fixed-length records by use of a record's key.

indexed data set. A data set specifically created, formatted and used by the Indexed Access Method. An indexed data set may also be called an indexed file.

indexed file. Synonym for indexed data set.

index entry. In an indexed file, a key-pointer pair, where the pointer is be used to locate a lower-level index block or a data block.

index register (#1, #2). Two words defined in EDL and contained in the task control block for each task. They are used to contain data or for address computation.

input buffer. (1) See buffer. (2) In the Multiple Terminal Manager, an area for terminal input and output.

input output control block (IOCB). A control block containing information about a terminal such as the symbolic name, size and shape of screen, the size of the forms in a printer.

instruction address register (IAR). The pointer that identifies the instruction currently being executed. The Series/1 maintains a hardware IAR to determine the Series/1 assembler instruction being executed. It is located in the level status block (LSB).

interactive. The mode in which a program conducts a continuous dialogue between the user and the system.

internal label. An area on tape used to record identifying information (similar to the identifying information placed on an external label). Internal labels are checked by the system to ensure that the correct volume is mounted.

interrupt information byte (IIB). In the Multiple Terminal Manager, a word containing the status of a previous input/output

request to or from a terminal.

job. A collection of related program execution requests presented in the form of job control statements, identified to the jobstream processor by a JOB statement.

job control statement. A statement in a job that specifies requests for program execution, program parameters, data set definitions, sequence of execution, and, in general, describes the environment required to execute the program.

job stream processor. The job processing facility that reads job control statements and processes the requests made by these statements. The Event Driven Executive job stream processor is \$JOBUTIL.

key. In the Indexed Access Method, one or more consecutive characters in a data record, used to identify the record and establish its order with respect to other records. See also key field.

key field. A field, located in the same position in each record of an Indexed Access Method data set, whose content is used for the key of a record.

level status block (LSB). A Series/1 hardware data area that contains processor status.

library. A set of contiguous records within a volume. It contains a directory, data sets and/or available space.

line. A string of characters accepted by the system as a single input from a terminal; for example, all characters entered before the carriage return on the teletypewriter or the ENTER key on the display station is pressed.

link edit. The process of resolving symbols in one or more object modules to produce another single module that is the input to the update process.

load mode. In the Indexed Access Method, the mode in which records are initially placed in an indexed file.

load module. A single module having cross references resolved and prepared for loading into storage for execution. The module is the output of the \$UPDATE or \$UPDATEH utility.

load point. A reflective marker placed near the beginning of a tape to indicate where the first record is written.

lock. In the Indexed Access Method, a method of indicating that a record or block is in use and is not available for another request.

LSB. See level status block.

member. A term used to identify a named portion of a partitioned data set (PDS). Sometimes member is also used as a synonym for a data set. See data set.

menu. A formatted screen image containing a list of options. The user selects an option to invoke a program.

menu-driven. The mode of processing in which input consists of the responses to prompting from an option menu.

multifile volume. A unit of recording media, such as tape reel or disk pack, that contains more than one data file.

multiple terminal manager. An Event Driven Executive licensed program that provides support for

transaction-oriented applications on a Series/1. It provides the capability to define transactions and manage the programs that support those transactions. It also manages multiple terminals as needed to support these transactions.

multivolume file. A data file that, due to its size, requires more than one unit of recording media (such as tape reel or disk pack) to contain the entire file.

non-labeled tapes. Tapes that do not contain identifying labels (as in standard labeled tapes) and contain only files separated by tapemarks.

null character. A user-defined character used to define the unprotected fields of a formatted screen.

option selection menu. A full screen display used by the Session Manager to point to other menus or system functions, one of which is to be selected by the operator. (See primary option menu and secondary option menu.)

output buffer. (1) See buffer. (2) In the Multiple Terminal Manager, an area used for screen output and to pass data to subsequent transaction programs.

overlay. The technique of reusing a single storage area allocated to a program during execution. The storage area can be reused by loading it with overlay programs that have been specified in the PROGRAM statement of the program.

overlay area. A storage area within a program reserved for overlay programs specified in the PROGRAM statement.

parameter selection menu. A full screen display used by the Session Manager to indicate the parameters to be passed to a program.

partition. A contiguous fixed-sized area of storage. Each partition is a separate address space.

physical timer. Synonym for hardware timer.

prefind. To locate the data sets or overlay programs to be used by a program and to store the necessary information so that the time required to load the prefound items is reduced.

primary-level index block. In an indexed data set, the lowest level index block. It contains the relative block numbers (RBNs) and high keys of several data blocks. See cluster.

primary menu. The program selection screen displayed by the Multiple Terminal Manager.

primary option menu. The first full screen display provided by the Session Manager.

primary task. The first task executed by the supervisor when a program is loaded into storage. It is identified by the PROGRAM statement.

priority. A combination of hardware interrupt level priority and a software ranking within a level. Both primary and secondary tasks will execute asynchronously within the system according to the priority assigned to them.

process mode. In the Indexed Access Method, the mode in which records may be retrieved, updated, inserted or deleted.

processor status word (PSW). A 16-bit register used to (1) record error or exception conditions that may prevent further processing and (2) hold certain flags that aid in error recovery.

program. A disk- or diskette-resident collection of one or more tasks defined by a PROGRAM statement; the unit that is loaded into storage. (See primary task and secondary task.)

program header. The control block found at the beginning of a program that identifies the primary task, data sets, storage requirements and other resources required by a program.

program/storage manager. A component of the Multiple Terminal Manager that controls the execution and flow of application programs within a single program area and contains the support needed to allow multiple operations and sharing of the program area.

protected field. On a display device, a field in which the operator cannot enter, modify, or erase data from the keyboard. It can contain text that the user can read.

PSW. See processor status word.

QCB. See queue control block.

QD. See queue descriptor.

QE. See queue element.

queue control block (QCB). A data area used to serialize access to resources that cannot be shared. See serially reusable resource.

queue descriptor (QD). A control block describing a queue built by the DEFINEQ instruction.

queue element (QE). An entry in the queue defined by the queue descriptor.

record. (1) The smallest unit of direct access storage that can be accessed by an application program on a disk or diskette using READ and WRITE. Records are 256 bytes in length. (2) In the Indexed Access Method, the logical unit that is transferred between \$IAM and the user's buffer. The length of the buffer is defined by the user.

recovery. The use of backup data to recreate data that has been lost or damaged.

reflective marker. A small adhesive marker attached to the reverse (nonrecording) surface of a reel of magnetic tape. Normally, two reflective markers are used on each reel of tape. One indicates the beginning of the recording area on the tape (load point), and the other indicates the proximity to the end of the recording area (EOT) on the reel.

relative record number. An integer value identifying the position of a record in a data set relative to the beginning of the data set. The first record of a data set is record one, the second is record two, the third is record three.

reorganize. For an indexed data set, the copying of the data to a new indexed data set in a manner that rearranges the data for more optimum processing and free space distribution.

return code. An indicator that reflects the results of the execution of an instruction or subroutine. The return code is placed in the task code word (at the beginning of the task control block).

roll screen. A display screen on which data is displayed 24 lines at a time or data is entered line by line, beginning with line 0 at the top of the screen and continuing through line 23 at the bottom of the screen. When a roll screen device's screen is full (all 24 lines used), an attempt to display the next line results in removal of the old screen (screen is erased) and the new line on line 0 is displayed at the top of the screen.

SBIOCB. See sensor based I/O control block.

second-level index block. In an indexed data set, the second-lowest level index block. It contains the addresses and high keys of several primary-level index blocks.

secondary option menu. In the Session Manager, the second in a series of predefined procedures grouped together in a hierarchical structure of menus. Secondary option menus provide a breakdown of the functions available under the session manager as specified on the primary option menu.

secondary task. Any task other than the primary task. A secondary task must be attached by a primary task or another secondary task.

sector. The smallest addressable unit of storage on a disk or diskette. A sector on a 4962 or 4963 disk is equivalent to an Event Driven Executive record. On a 4964 or 4966 diskette, two sectors are equivalent to an Event Driven Executive record.

sensor based I/O control block (SBIOCB). A control block containing information related to sensor I/O operations.

sequential access. The processing of a data set in order of occurrence of the records in the data set. (1) In the Indexed Access Method, the processing of records in ascending collating sequence order of the keys. (2) When using READ/WRITE, the processing of records in ascending relative record number sequence.

serially reusable resource (SRR). A resource that can only be accessed by one task at a time. Serially reusable resources are usually managed via (1) a QCB and ENQ/DEQ statements or (2) an ECB and WAIT/POST statements.

session manager. A series of predefined procedures grouped together as a hierarchical structure of menus from which you select the utility functions, program preparation facilities, and language processors needed to prepare and execute application programs. The menus consist of a primary option menu that displays functional groupings and secondary option menus that display a breakdown of these functional groupings.

shared resource. A resource that can be used by more than one task at the same time.

shut down. See data set shut down.

source module/program. A collection of instructions and statements that constitute the input to a compiler or assembler. Statements may be created or modified using one of the text editing facilities.

standard labels. Fixed length 80-character records on tape containing specific fields of information (a volume label identifying the tape volume, a header label preceding the data records, and a

trailer label following the data records).

static screen. A display screen formatted with predetermined protected and unprotected areas. Areas defined as operator prompts or input field names are protected to prevent accidental overlay by input data. Areas defined as input areas are not protected and are usually filled in by an operator. The entire screen is treated as a page of information.

subroutine. A sequence of instructions that may be accessed from one or more points in a program.

supervisor. The component of the Event Driven Executive capable of controlling execution of both system and application programs.

system configuration. The process of defining devices and features attached to the Series/1.

SYSGEN. See system generation.

system generation. The processing of user selected options to create a supervisor tailored to the needs of a specific Series/1 configuration.

system partition. The partition that contains the supervisor (partition number 1, address space 0).

tapemark. A control character recorded on tape used to separate files.

task. The basic executable unit of work for the supervisor. Each task is assigned its own priority and processor time is allocated according to this priority. Tasks run independently of each other and compete for the system resources. The first task of a program is the primary task. All tasks attached by the primary task

are secondary tasks.

task code word. The first two words (32 bits) of a task's TCB; used by the emulator to pass information from system to task regarding the outcome of various operations, such as event completion or arithmetic operations.

task control block (TCB). A control block that contains information for a task. The information consists of pointers, save areas, work areas, and indicators required by the supervisor for controlling execution of a task.

task supervisor. The portion of the Event Driven Executive that manages the dispatching and switching of tasks.

TCB. See task control block.

terminal. A display station, teletypewriter or printer.

terminal control block (CCB). A control block that defines the device characteristics, provides temporary storage, and contains links to other system control blocks for a particular terminal.

terminal environment block (TEB). A control block that contains information on a terminal's attributes and the program manager operating under the Multiple Terminal Manager. It is used for processing requests between the terminal servers and the program manager.

terminal screen manager. The component of the Multiple Terminal Manager that controls the presentation of screens and communications between terminals and transaction programs.

terminal server. A group of programs that perform all the input/output and interrupt hand-

ing functions for terminal devices under control of the Multiple Terminal Manager.

trace range. A specified number of instruction addresses within which the flow of execution can be traced.

transaction oriented applications. Program execution driven by operator actions, such as responses to prompts from the system. Specifically, applications executed under control of the Multiple Terminal Manager.

transaction program. See transaction-oriented applications.

transaction selection menu. A Multiple Terminal Manager display screen (menu) offering the user a choice of functions, such as reading from a data file, displaying data on a terminal, or waiting for a response. Based upon the choice of option, the application program performs the requested processing operation.

unprotected field. On a display device, a field in which the user can enter, modify, or erase data using the keyboard. Unprotected fields on a static screen are defined by the null character.

update. (1) To alter the contents of storage or a data set. (2) To convert object modules, produced as the output of an assembly or compilation, or the output of the linkage editor, into a form that can be loaded into storage for program execution and to update the directory of the volume on which the loadable program is stored.

user exit. (1) Assembly language instructions included as part of an EDL program and invoked via the USER instruction. (2) A point in an IBM-supplied program where a

user written routine can be given control.

vary offline. (1) To change the status of a device from online to offline. When a device is offline, no data set can be accessed on that device. (2) To place a disk or diskette in a state where it is not available for use by the system; however, it will still be available for executing I/O at the basic access level (EXIO).

vary online. To restore a device to a state where it is available for use by the system.

volume. A disk or diskette subdivision defined during system configuration. A volume may contain up to 32,767 records. As many volumes may be defined for a disk as will physically fit. A diskette is limited to one volume.

volume label. A label that uniquely identifies a single unit of storage media.



This index is common to the Event Driven Executive library. The index includes entries from the seven publications listed below. (The Glossary is not indexed.) Each publication has a copy of the index, which provides a cross-reference between the publications.

Each page number entry contains a single letter prefix which identifies the publication where the listed subject can be found. The letter prefixes have the following meanings:

- C = Communications and Terminal Application Guide
- I = Internal Design
- L = Language Reference
- S = System Guide
- U = Utilities, Operator Commands, Program Preparation, Messages and Codes
- M = Multiple Terminal Manager Internal Design
- A = Indexed Access Method Internal Design

Special Characters

\$\$EDXLIB system name L-228, S-57
 \$\$EDXVOL system name L-228, S-57
 \$A display active programs,
 operator command S-63, U-11
 \$ATTASK special task control
 block L-61
 \$AUTO link edit auto call data
 set S-403, U-401
 \$B blank (clear) screen, operator
 command S-63, U-12
 \$BSCTRCE trace utility for BSC
 lines C-61
 \$BSCUT1 trace printing utility for
 BSC C-62
 \$BSCUT2 test utility for BSC
 lines C-64
 \$C cancel a program, operator
 command S-63, U-13
 \$COMPRES library compress S-64,
 U-57
 \$COPY copy data sets S-64, U-59
 \$COPYUT1 copy data sets with
 allocation S-64, U-64
 \$CP change terminal's partition
 assignment command
 overview I-73, S-63
 syntax U-14
 \$D dump storage, operator command
 S-63, U-15
 \$DASDI format disk or diskette
 S-64, U-68
 \$DEBUG debug module description
 I-77
 \$DEBUG debugging tool U-82
 \$DICOMP display composer
 command description U-106
 create partitioned data set
 member S-247
 invoking U-105
 overview S-67
 \$DIINTR display interpreter U-150
 \$DISKUT1 allocate/delete, list
 directory data
 \$JOBUTIL procedure S-229
 allocate partitioned data set
 S-248
 command descriptions U-135
 overview S-64
 \$DISKUT2 patch, dump, or clear
 member
 description U-142
 overview S-64
 printing I/O error logs S-275
 syntax U-143
 \$DISKUT3 data management utility
 description S-315
 input to S-316
 request block contents S-317
 return codes S-319, U-444
 \$DIUTIL display data base utility
 S-248, U-150
 \$DUMP dump saved storage and
 registers utility U-163
 \$E eject printer page, operator
 command S-63, U-16
 \$EDIT1/\$EDIT1N text editors
 command syntax
 EDIT U-174
 EDIT mode subcommands
 U-182
 END U-175
 LIST U-176
 READ U-177
 SUBMIT U-179
 WRITE U-180
 control keys U-172
 data set requirements U-169
 line editing commands U-203
 overview S-66, U-169
 summary of commands and
 subcommands U-171
 \$EDXASM Event Driven Language
 compiler
 features supported U-361
 internal overview I-5, I-211
 invoking
 with \$JOBUTIL U-368

- with \$L U-370
- with session manager U-369
- listing program (\$EDXLIST) U-370
- options U-358
- output U-359
- overlay program example I-244
- overview S-71, U-356
- programming considerations U-361
 - arithmetic expression operators U-365
 - ATTNLIST U-365
 - COPY statements U-362
 - ECB and QCB U-362
 - EQU U-365
 - GETEDIT and PUTEDIT U-365
 - instructions requiring support modules U-365
 - IODEF statement placement U-364
 - multiple declarations on DATA/DC U-363
 - source line continuation U-361
 - required data sets U-357
 - usage example S-397
 - using the compiler U-356
- \$EDXATSR supervisor interface routine I-48
- \$EDXDEF hardware configuration
 - editing to match hardware configuration S-117
 - overview I-5, I-6
 - storage map I-7
- \$EDXL language control data set of \$EDXASM I-221, U-357
- \$EDXLIST compiler listing program U-370
- \$EDXNUC supervisor data set
 - in system generation S-126
 - overview I-5
 - with \$LINK utility U-399
- \$EDXNUC supervisor data sets U-399
- \$EXEC language emulator linkage I-279, I-313
- \$EXEC session manager option S-216, U-41
- \$FONT 4978 character image tables utility S-68, U-205
- \$FSEDIT full-screen editor, host and native
 - data set requirements U-209
 - options
 - BROWSE U-213
 - EDIT U-214
 - END U-218
 - READ U-216
 - SUBMIT U-217
 - WRITE U-216
 - overview S-66, U-209
 - primary commands U-218
 - program function (PF) keys U-211
 - scrolling U-210
 - summary of options and commands U-212
- \$HCFUT1 Host Communications Facility utility C-107
- \$IAM Indexed Access Method load module S-155
- \$IAM task error exit S-178
- \$IAMUT1 Indexed Access Method utility S-148, U-235
- \$IDEF \$EDXASM instruction definition
 - description I-241
 - instruction format I-226
- \$IMAGE define screen image utility S-68, U-250
 - usage example S-387
- \$IMDATA subroutine S-303
 - usage example S-375
- \$IMDEFN subroutine S-301
 - usage example S-375
- \$IMOPEN subroutine S-300
 - usage example S-374
- \$IMPROT subroutine S-302
 - usage example S-375
- \$INDEX subroutine, \$EDXASM I-233
- \$INITDSK initialize or verify volume S-64, U-256
- \$INITIAL automatic initialization and restart
 - description S-129
 - with session manager S-209, U-28
- \$IOTEST test sensor I/O, list configuration S-67, U-263
- \$JOBUTIL job stream processor S-69, U-271
 - commands U-272
 - set up procedure U-271
 - usage example S-408, U-290
- \$L load program, operator command
 - internals I-23
 - overview S-63
 - syntax U-17
- \$LEMSG \$LINK message data set U-401
- \$LINK linkage editor
 - data set requirements U-400
 - description U-390
 - in system generation I-5
 - invoking
 - with \$JOBUTIL U-405
 - with \$L U-405
 - with session manager U-406
 - overview S-71
 - usage example S-402
- \$LNKCNTRL data set S-118
- \$LOADER I-19, I-22
 - module description I-78
- \$LOG I/O error logging utility
 - description S-270, U-292
 - overview S-67
- \$LPARSE subroutine I-240
- \$MOVEVOL disk volume dump/restore S-65, U-294
- \$P patch storage, operator command S-63, U-18
- \$PACK/\$UNPACK subroutines S-309
- \$PDS partitioned data set utility
 - in a program S-259
 - overview S-65
- \$PFMAP identify 4978 program
 - function keys S-68, U-301
- \$PREFIND prefind data sets and overlays S-69, U-302
- \$PRT2780 spooled print utility C-72
- \$PRT3780 spooled print utility C-72
- \$RJE2780 remote job entry utility C-73, S-66

\$RJE3780 remote job entry utility
 C-73, S-66
 \$RMU (see Remote Management Utility)
 \$SMCTL session manager program
 S-209, S-212
 \$SMEND session manager program
 S-212
 \$SMJOB session manager program
 S-212
 \$SMLOG session manager program
 S-212
 \$SMMAIN session manager program
 S-210, S-212, U-28
 \$SMMLOG, logon menu for session
 manager S-212
 \$SMMPRIM, primary option menu for
 session manager S-212, U-27,
 U-35
 \$SMM02, program preparation sec-
 ondary option menu S-214, U-37
 \$SMM03, data management secondary
 option menu S-215, U-39
 \$SMM04, terminal utilities
 secondary option menu S-215,
 U-41
 \$SMM05, graphics utilities second-
 ary option menu S-216, U-41
 \$SMM06, execute program utilities
 secondary option S-216
 \$SMM07, job stream processor
 utilities secondary option S-216
 \$SMM08, communications utilities
 option S-217, U-43
 \$SMM09, diagnostic utilities
 S-217, U-44
 \$START supervisor entry point
 I-279, I-313
 \$STOREMAP example I-27
 \$SYSCOM data area I-12, I-279,
 I-313, S-113
 \$SYSLOG system logging device
 overview S-110
 \$SYSLOGA alternate system logging
 device
 overview S-111
 \$SYSPRTR system printer
 overview S-111
 \$SIASM Series/1 macro assembler
 description U-372
 internals I-5, I-253
 overview S-9
 storage map, general I-256
 \$T set date/time, operator
 command S-63, U-19
 \$TAPEUT1 tape management utility
 U-311
 \$TCBCCB (ATTACH) L-59
 \$TERMUT1 change terminal
 parameters S-68, U-334
 \$TERMUT2
 process 4978 image or control
 store S-68, U-339
 restore 4974 image U-339
 \$TERMUT3 send message to a
 terminal S-68, U-344
 \$TRAP class interrupt trap
 utility S-67, U-348
 \$UNPACK/\$PACK subroutines S-309
 \$UPDATE object program converter
 description U-408
 in system generation I-5
 overview S-69
 usage example S-407

\$UPDATEH object program converter
 (host) S-69, U-418
 \$VARYOFF set disk, diskette, or
 tape offline S-63, U-20
 \$VARYON set disk, diskette, or
 tape online S-63, U-22
 with standard labeled tape
 S-237
 \$W display date/time, operator
 command S-63, U-25
 #1 index register 1 L-6
 #2 index register 2 L-6

A

A after, \$FSEDIT line command
 U-226
 A-conversion L-153
 A/I (see analog input)
 A/O (see analog output)
 abort task level (SVC abend) I-49
 ACCA terminal C-7, L-295
 Access Method, Indexed
 (see Indexed Access Method)
 ACTION, Multiple Terminal Manager
 CALL
 coding description C-130,
 L-360
 internals M-9
 overview C-117, L-29
 activate
 error logging, \$LOG utility
 U-293
 realtime data member, RT
 \$DICOMP subcommand U-124
 stopped task, GO \$DEBUG
 command U-93
 task supervisor execution
 state I-43
 TRAP function of storage dump,
 \$TRAP utility U-348
 AD
 add member, \$DICOMP command
 U-106
 advance, \$DICOMP subcommand
 U-111
 advance X,Y (PDS) S-255
 assign define key, \$TERMUT2
 command U-342
 add
 add member, AD \$DICOMP com-
 mand U-106
 null data set on tape volume,
 TA \$TAPEUT1 command U-330
 options to the session
 manager S-224
 support for new I/O terminals
 I-117
 calling conventions I-118
 code translation tables
 I-118
 linkage conventions I-119
 terminal instruction
 modification I-119
 ADD data manipulation instruction
 coding description L-52
 overview L-19
 precision table L-53
 address relocation translator
 I-71, S-42
 addressing indexing feature L-6

ADDV data manipulation instruction
 coding description L-54
 index register use L-55
 overview L-19
 precision table L-55
 advance, AD \$DICOMP subcommand U-111
 advance and prompting input, terminal I/O L-46
 AI (see analog input)
 AL
 allocate data member, \$DIUTIL command U-151
 allocate data set, \$DISKUT1 command U-137
 allocate data set, \$JOBUTIL command U-273
 allocate member, \$DICOMP command U-107
 allocate
 data set
 \$JOBUTIL command U-273
 AL \$DISKUT1 command U-137
 ALLOCATE function C-214
 tape, TA \$TAPEUT1 command U-333
 member
 \$DICOMP command U-107
 \$DIUTIL command U-151
 \$PDS S-261
 ALLOCATE function C-216, I-166, I-174
 allowable precision table L-20
 alter member AL \$DICOMP command U-107
 alter terminal configuration, \$TERMUT1 U-334
 alternate system logging device (\$SYSLOGA) S-47
 alternate tracks S-58, U-73, U-78
 ALTIAM Indexed Access Method subroutine S-167
 analog input S-49
 AI \$IOTEST command U-268
 control block I-129
 IODEF statement L-187
 overview S-49
 SBIO instruction L-263
 SENSORIO configuration statement L-39
 analog output
 AO \$IOTEST command U-264
 control block I-129
 description S-49
 IODEF statement L-186
 SBIO instruction L-264
 SENSORIO configuration statement L-39, S-84
 AND data manipulation instruction
 coding description L-57
 overview L-19
 AO (see analog output)
 application program
 automatic initialization and restart S-129
 indexed access S-149
 introduction L-1
 manager C-119
 preparation U-351
 size estimating S-344
 structure L-8
 support S-20
 ASCII terminals
 codes S-110
 configuring S-96
 devices supported C-6, S-14
 graphics L-26, S-46
 TERMINAL statement examples S-106
 ASMERROR, \$EDXASM instruction I-230
 assembler
 (see \$EDXASM)
 (see \$SIASM)
 (see host assembler)
 assign
 alternate for defective 4963 sector, \$DASDI utility U-78
 DEFINE key in 4978 control store, AD \$TERMUT2 command U-341
 asynchronous communications control adapter (see ACCA)
 AT set breakpoints and trace ranges, \$DEBUG command U-90
 ATTACH task control instruction
 coding description L-59
 internals I-44
 overview L-42, S-34
 attention handling, terminal I/O I-108, L-47, S-63
 attention keys, terminal I/O L-47
 attention list (see ATTNLIST)
 ATTN key (see attention handling)
 ATTNLIST task control statement
 \$ATTASK L-61
 coding description L-61
 overview L-42, S-30
 attribute character, 3101 C-122
 autocall
 option, \$LINK U-401
 AUTOCALL statement requirement (WXTRN) L-323
 automatic
 application initialization S-13, S-129
 application restart S-13, S-129

B

B before, \$FSEDIT line command U-226
 backup disk or disk volume on tape, ST \$TAPEUT1 command U-330
 backup dump restore utility, \$MOVEVOL U-294
 base records, indexed data set
 definition S-149
 loading S-160
 basic exchange
 diskette data set copy utility, \$COPY U-59
 basic supervisor and emulator (see supervisor/emulator)
 batch job processing (see \$JOBUTIL)
 BEEP, Multiple Terminal Manager CALL
 coding description C-137, L-361
 internals M-9
 overview C-117, L-29
 binary synchronous communications
 automatic retry S-17
 BSCAM/BSCAMU module

descriptions I-80
 BSCLINE configuration state-
 ment C-42, S-76
 control flow (BSCAM) I-147
 device data block (BSCDDB)
 I-133
 features C-35, S-16
 Host Communications Facility
 protocol I-156
 instruction formats C-38,
 I-144
 multipoint operation C-36,
 S-16
 overview S-16
 point-to-point lines S-16
 Remote Management Utility
 requirements C-208
 sample programs C-59
 special labels for,
 description I-149
 system internal design I-133
 test utility, \$BSCUT2 C-64
 trace printing routine,
 \$BSCUT1 C-62
 trace routine, \$BSCTRCE C-61
 blank screen, \$B operator command
 S-63, U-12
 BLANK TERMCTRL function L-288
 BLDTXT subroutine, \$EDXASM I-237
 BLINK TERMCTRL function L-288
 BLP (see bypass label processing)
 BOT (beginning-of-tape) L-40
 BOTTOM reposition line pointer,
 \$EDIT1/N editor subcommand U-183
 boundary requirement, full-word
 DO L-34
 IF L-34
 PROGRAM L-225
 BP list breakpoints and trace
 ranges, \$DEBUG command U-92
 breakpoints and trace setting, AT
 \$DEBUG command U-90
 BROWSE display data set, \$FSEDIT
 option U-213
 BSC (see binary synchronous
 communications)
 BSCAM (see binary synchronous com-
 munications)
 BSCCLOSE BSC statement I-144,
 I-148
 coding description C-38
 BSCDDB binary synchronous device
 data block
 description of I-133
 equates I-291
 BSCEQU L-11
 BSCIA immediate action routine
 (BSC) I-148
 BSCIOCB BSC statement C-39, I-144
 BSCLINE configuration statement
 C-42, S-76
 BSCOPEN BSC statement C-44,
 I-145, I-148
 BSCREAD BSC statement C-45,
 I-145, I-148
 BSCWRITE BSC statement C-49,
 I-146, I-148
 BSF (backward space file) L-75
 BSR (backward space record) L-75
 BTE, buffer table entry A-20
 BU build data member, \$DIUTIL
 command U-153
 buffer
 table entry
 definition A-20

description A-31
 terminal I/O buffer
 management I-109
 BUFFER data definition statement
 coding description L-65
 overview L-17
 build data member, BU \$DIUTIL
 command U-153
 building an indexed data set
 U-247
 burst output with electronic dis-
 play screens L-46
 bypass label processing U-311
 description S-244

C

C
 change a key definition,
 \$TERMUT2 command U-342
 copy line, \$FSEDIT line
 command U-226
 CA cancel
 assembly, \$EDXASM attention
 request U-358
 copy, \$COPYUT1 attention
 request U-64
 list option, \$FSEDIT attention
 request U-217
 listing, \$EDXLIST attention
 request U-358
 CAD copy all data members,
 \$COPYUT1 command U-64
 CALL
 copy all members, \$COPYUT1
 command U-64
 program control instruction
 coding description L-68
 Indexed Access Method
 syntax S-146
 Multiple Terminal Manager
 syntax L-359
 overview L-32, S-31
 program L-68
 subroutine L-68
 callable routines L-30
 CALLFORT program control
 instruction
 coding description L-70
 overview L-32
 cancel
 \$C operator command U-13
 assembly, CA \$EDXASM attention
 request U-358
 copy, CA \$COPYUT1 attention
 request U-64
 dump, CA \$DUMP command U-165
 list option, CA \$FSEDIT
 attention request U-217
 listing, CA \$EDIT/N attention
 request U-172
 CAP copy all programs, \$COPYUT1
 command U-64
 CC copy block, \$FSEDIT line
 command U-226
 CCB
 equate table I-292
 internals I-105, I-119
 interprocessor communications
 C-30
 use in terminal I/O support
 I-113

CCBEQU L-11

CD
 clear data set, \$DISKUT2 command U-144
 copy data set, \$COPY command U-61
 copy data set, \$TAPEUT1 command U-313
 CDATA, Multiple Terminal Manager
 CALL
 coding description C-139, L-362
 internals M-9
 overview L-29
 CDRRM equates C-292
 CG copy all members (generic)
 \$COPYUT1 command U-64
 CH
 change hardcopy device, \$BSCUT2 command C-70
 change host library, \$UPDATEH command U-420
 chain, ECB/QCB/TCB I-55
 CHAIN supervisor service routine I-54
 CHAIND supervisor service routine I-54
 CHAINE supervisor service routine I-54
 chaining L-27
 CHAINP supervisor service routine I-54
 change
 address assignment of terminal, RA \$TERMUT1 command U-336
 base address, QUALIFY \$DEBUG command U-101
 character string, CHANGE \$EDIT1/N editor subcommand U-184
 character string, change \$FSEDIT primary command U-219
 execution sequence, GOTO \$DEBUG command U-94
 graphics or report display profile, \$DICOMP utility U-105
 hardcopy device, CH \$BSCUT2 command C-70
 hardcopy device, RH \$TERMUT1 command U-338
 host library, CH \$UPDATEH command U-420
 key definition in 4978 control store, C \$TERMUT2 U-342
 name of logical device, RE \$TERMUT1 command U-337
 output volume, CV \$UPDATE command U-409
 page formatting parameters of a terminal, CT \$TERMUT1 U-335
 partition assignment, \$CP operator command U-14
 realtime data member name RT (\$PDS) S-258
 tape label support U-322
 volume
 CV \$BSCUT1 command C-62
 CV \$COPYUT1 command U-64
 CV \$DISKUT1 command U-137
 CV \$DISKUT2 command U-143
 CV \$UPDATEH command U-418

character constants L-89
 character image table U-205
 CHGPAN, Multiple Terminal Manager
 CALL
 coding description C-135, L-364
 internals M-9
 overview C-124, L-29
 CL clear work data set, \$FSEDIT primary command U-221
 class interrupt vector table I-10, I-277
 class interrupts, intercepting, \$STRAP utility U-348
 clear
 data set, CD \$DISKUT2 command U-144
 screen, \$B operator command U-12
 CLOSE Host Communications Facility, TP operand C-90
 CLSRU (close tape data set) L-75
 cluster, indexed data set S-200
 CM copy member
 \$COPYUT1 command U-64
 \$DIUTIL command U-155
 CMDEQU L-12
 CMDSETUP I-13, I-67
 CNG copy all members (non-generic), \$COPYUT1 command U-64
 CO command, \$RJE2780/\$RJE3780 C-76
 COBOL
 execution requirements S-23
 link editing S-71
 overview S-7
 program preparation requirements S-23
 use with Multiple Terminal Manager C-193
 code translation
 new support tables I-111
 terminal I/O layer 2 I-109
 code words, task L-8
 COLS display columns, \$FSEDIT line command U-228
 command area, \$EDXASM I-214
 command descriptions U-235
 COMMAND send to host, \$RJE2780/\$RJE3780 C-75
 command table I-68, I-282, I-301
 common data area (see \$SYSCOM)
 common emulator setup routine
 command table I-13, I-282, I-301
 operating conventions I-67
 communication error function I-166
 communications utilities
 \$BSCTRCE C-61
 \$BSCUT1 C-62
 \$BSCUT2 C-64
 \$HFCUT1 C-107
 \$PRT2780 C-72
 \$PRT3780 C-72
 \$RJE2780 C-73
 \$RJE3780 C-73
 \$RMU C-282
 communications utilities (session manager) S-217, U-42
 communications vector table I-11, I-278, I-313
 compiler (see \$EDXASM)

completion codes (see return codes)
 \$EDXASM U-436
 \$IAMUT1 U-437
 \$JOBUTIL U-439
 \$LINK U-440
 \$UPDATE U-443
 compress
 data base, CP \$DIUTIL command U-154
 library, \$COMPRES utility U-57
 compressed byte string S-309
 CONCAT graphics instruction
 coding description L-72
 overview L-26
 concatenating indexed data sets S-167
 concurrent access L-27
 concurrent execution L-42
 configuration statements S-75
 configure terminal CT \$TERMUT1 command U-335
 connecting an indexed data set S-159
 continuation, source program line, \$EDXASM U-361
 control, device instruction level L-24
 control block (see DSCB)
 control block and parameter tables
 BSCEQU I-133, I-291, L-11
 CCBEQU (see also CCB) L-11
 CMDEQU (see also emulator command table) L-12
 DDBEQU I-92, I-308, L-12
 DSCBEQU (see also DSCB) L-12
 ERRORDEF L-12
 FCBEQU A-20, L-12
 IAMEQU L-12
 PROGEQU I-312, L-13
 referencing I-289
 TCBEQU (see also TCB) L-13
 control block module (ASMOBJ) description I-76
 CONTROL IDCBC command L-175
 control keys for text editors U-172
 control records, \$LINK U-396
 control statements, program listing L-28
 task L-42
 terminal I/O forms control L-45
 CONTROL tape instruction L-74
 conversion
 algorithm for graphics I-201
 alphanumeric data L-152
 definition
 EBFLCVT module description I-80
 floating point/binary I-205
 numeric data L-148
 program modules by \$UPDATE/H U-418
 terminal I/O binary/EBCDIC I-110
 CONVTB data formatting instruction
 coding description L-79
 internals I-207
 overview L-18
 CONVTD data formatting instruction
 coding description L-82
 internals I-207
 overview L-18
 copy
 block of text, CC \$FSEEDIT line command U-226
 data members, all, CAD \$COPYUT1 command U-64
 data set, CD \$COPY command U-61
 data sets with allocation, \$COPYUT1 utility U-64
 line of text, C \$FSEEDIT line command U-226
 member
 CM \$COPYUT1 command U-64
 CM \$DIUTIL command U-155
 members
 all, CALL \$COPYUT1 command U-64
 generic, CG \$COPYUT1 command U-64
 non-generic, CNG \$COPYUT1 command U-64
 programs, all, CAP \$COPYUT1 command U-64
 text, \$EDIT1/N editor subcommand U-186
 volume, CV \$COPY command U-62
 copy code library, instruction parsing (\$EDXASM) I-222
 COPY instruction
 coding description L-86
 overview L-33
 Count record C-256
 CP compress data base, \$DIUTIL command U-154
 CR invoke \$DISKUT1, \$IAMUT1 command U-236
 create
 character image tables, \$FONT U-205
 source data set, \$FSEEDIT U-214
 supervisor for another Series/1 S-132
 unique labels, \$SYSNDX (\$EDXASM) I-242
 create indexed data set S-156
 cross partition instructions I-71
 cross partition services S-286
 CSECT list, supervisor
 Version 1.1 S-347
 Version 2 S-357
 CSECT program module sectioning statement
 coding description L-87
 overview L-33
 CT
 change tape drive attributes, \$TAPEUT1 command U-315
 configure terminal, \$TERMUT1 command U-335
 CV
 change output volume U-409
 \$UPDATE command U-409
 \$UPDATEH command U-418
 change volume
 \$BSCUT1 command C-62
 \$COPYUT1 command U-64
 \$DISKUT1 command U-137
 \$DISKUT2 command U-143
 copy volume, \$COPY command U-59

CYCLE
 coding description C-132,
 L-365
 internals M-9
 overview C-116, L-29
 cylinder S-60
 cylinder track sector (CTS) U-135

D

D delete line, \$FSEDIT line command U-228
 D/I (see digital input)
 D/O (see digital output)
 data
 conversion (see conversion)
 conversion specifications (see also conversion) L-146
 definition statements L-17
 files for \$S1ASM I-254
 floating-point arithmetic instructions L-20
 formatting functions L-18
 formatting instructions L-18
 integer and logical instructions L-19
 length of transmitted, host communications I-159
 management S-45
 management system, Indexed Access Method L-27
 manipulation instructions L-19
 record contents, text editor I-325
 representation L-20
 floating-point L-20
 integer L-19
 terminal input L-45
 terminal output L-45
 transfer initialization, terminal I/O support I-112
 transfer rates, Host Communications Facility C-84
 transfer ready, (DTR) BSCOPEN I-148
 Data Collection Interactive S-11
 DATA data definition statement coding description L-88
 overview L-17
 data management utilities
 \$COMPRES S-64, U-57
 \$COPY S-64, U-59
 \$COPYUT1 S-64, U-64
 \$DASDI S-64, U-68
 \$DISKUT1 S-64, U-135
 \$DISKUT2 S-64, U-142
 \$DISKUT3 S-315
 \$IAMUT1 S-148, U-235
 \$INITDSK S-64, U-256
 \$MOVEVOL S-65, U-294
 \$PDS S-247
 \$TAPEUT1 U-311
 session manager S-215, U-38
 data manipulation, vector L-19
 data manipulation instructions L-19
 Data record C-257
 data representation, terminal I/O L-45
 data set
 allocation/deletion

\$DISKUT1 U-137
 \$DISKUT3 S-315
 \$JOBUTIL U-273
 \$PDS S-248
 session manager U-29
 characteristics, HCF C-83
 format
 \$FSEDIT U-210
 \$PDS S-249
 \$PRT2780 C-72
 \$PRT3780 C-72
 naming conventions C-82, S-56
 transfer
 RECEIVE function C-243
 SEND function C-247
 utilities (see data management utilities)
 data set naming conventions, Host Communications Facility C-82
 data-set-shut-down condition S-179
 date/time
 display, \$W operator command U-25
 set, \$T operator command U-19
 DC data definition statement coding description L-88
 overview L-17
 DCB EXIO control statement coding description L-91
 overview L-24
 DCE directory control entry format I-88
 DCI (Data Collection Interactive) S-11
 DD block delete, \$FSEDIT line command U-228
 DDB disk data block description I-92
 equate table I-308
 DDBEQU L-12
 DE delete member
 \$DISKUT1 command U-137
 \$DIUTIL command U-156
 delete data set, \$JOBUTIL command U-274
 deadlocks C-238, S-180
 debug
 \$EDXASM overlay programs I-248
 aids (see also diagnostic aids) S-18
 facility, \$DEBUG utility U-82
 define
 horizontal tabs, HTAB \$IMAGE command U-252
 image dimensions, DIMS \$IMAGE command U-251
 indexed data set, DF \$IAMUT1 command U-237
 null representation, NULL \$IMAGE command U-253
 vertical tabs, VTAB \$IMAGE command U-254
 DEFINEQ queue processing statement coding description L-94
 overview L-37
 definition statements, data L-17
 delete
 data set
 \$JOBUTIL command U-274
 DELETE function C-216
 tape data set, TA \$TAPEUT1 command U-333

elements, IN \$DICOMP command U-107
 member
 \$PDS S-261
 DE \$DISKUT1 command U-137
 DE \$DIUTIL command U-156
 text
 \$EDIT1(N) editor subcommand U-188
 line, D \$FSEdit line command U-228
 with \$PREFIND U-305
 DELETE function C-216, I-166, I-174
 DELETE instruction
 coding description L-329
 overview L-27, S-147
 return codes L-330
 DEQ task control instruction
 coding description L-95
 internals I-59
 overview L-42, S-33
 supervisor function I-46
 DEQBSC dequeue BSC DDB routine I-149
 DEQT terminal I/O instruction
 coding description L-97
 overview L-44, S-47
 DETACH task control instruction
 coding description L-98
 internals I-45
 overview L-42, S-30
 detached, task supervisor execution state I-43
 device
 busy (EXOPEN) L-129
 data block description, EXIO I-123
 instruction level control L-24
 interrupt handling, EXIO I-125
 test utility, \$IOTEST U-263
 vector table I-11, I-278
 DF define indexed file, \$IAMUT1 command U-237
 DI (see digital input)
 diagnostic
 aids S-265
 summarized S-18
 utilities
 \$DEBUG U-82
 \$DUMP U-163
 \$IOTEST U-263
 \$LOG U-292
 \$TRAP U-348
 with session manager S-217, U-38
 digital input
 \$IOTEST command U-266
 digital I/O control block I-129
 direct output, \$DICOMP subcommand U-112
 direct output to another device (\$PDS) S-255
 display parameters, \$IAMUT1 command U-239
 external sync, XI \$IOTEST command U-266
 IODEF statement L-186
 overview S-48
 SBIO instruction L-265
 SENSORIO configuration statement S-84
 digital output
 digital I/O control block I-129
 DO \$IOTEST command U-265
 external sync, XO \$IOTEST command U-266
 IODEF statement L-186
 overview S-48
 SBIO instruction L-267
 SENSORIO configuration statement L-84
 DIMS define image dimensions, \$IMAGE command U-251
 direct access common I/O module, DISKIO, description I-77
 direct access storage device organization S-52
 direct output, DI \$DICOMP subcommand U-112
 directory
 control entry (DCE) I-88
 entries S-249
 member entry (DME) I-89
 disaster recovery from tape, RT \$TAPEUT1 command U-326
 DISCONN Indexed Access Method CALL
 coding description L-332
 overview L-27, S-148
 return codes L-333
 DISCONNECT Multiple Terminal Manager utility C-119, C-159
 disconnecting an indexed data set S-159
 DISK configuration statement S-78
 disk/diskette
 capacity S-58
 data block (DDB) I-92
 fixed-head S-15, S-61
 I/O task I-95
 IPL S-16, S-61
 primary volume S-60
 resident loading code I-19
 secondary volume S-60
 symbolic addressing L-10
 utilities
 \$COMPRES S-64, U-57
 \$COPY S-64, U-59
 \$COPYUT1 S-64, U-64
 \$DASDI S-64, U-68
 \$DISKUT1 S-64, U-135
 \$DISKUT2 S-64, U-142
 \$DISKUT3 S-315
 \$IAMUT1 S-148, U-235
 \$INITDSK S-64, U-256
 \$MOVEVOL S-65, U-294
 \$PDS S-247
 utility function table U-49
 volume S-16, S-52
 disk I/O instructions L-22
 DISKIO direct access common I/O module description I-77
 display (see also list)
 character image tables, DISP \$FONT command U-205
 contents of storage or registers, LIST \$DEBUG command U-95
 control member (\$PDS) S-250
 control member format (\$PDS) S-252
 initial data values for image S-303
 processor composer, \$DICOMP U-105

processor interpreter,
 \$DIINTR U-150
 processor utility, \$DIUTIL
 U-150
 processor utility, general
 description U-105
 profile elements (\$PDS) S-252
 protected and null fields of
 an image S-302
 report line items (\$PDS)
 S-255
 status of all tasks, WHERE
 \$DEBUG command U-102
 storage, \$D operator command
 S-63, U-15
 time and data, TD (\$PDS)
 S-258
 time and date, \$W operator
 command S-63, U-25
 utility program set (\$PDS)
 S-248
 variable, VA(\$PDS) S-254
 4978 program function keys,
 \$PFMAP utility U-301
 DISPLAY TERMCTRL function L-288
 DIVIDE data manipulation
 instruction
 coding description L-99
 overview L-19
 precision table L-100
 DME directory member entry
 format I-89
 updated by SETEOD S-324
 DO
 digital output (see digital
 output)
 program sequencing
 instruction
 coding description L-101
 overview L-34
 double-precision L-19
 floating-point arithmetic
 L-21
 integer and logical L-19
 DOWN move line pointer, \$EDIT1/N
 editor subcommand U-189
 DP
 dump to printer
 \$DISKUT2 command U-144
 \$TAPEUT1 command U-317
 print trace file, \$BSCUT1
 command C-62
 DR draw symbol, \$DICOMP
 subcommand U-112
 DR draw symbol (\$PDS) S-254
 draw
 line, LI \$DICOMP subcommand
 U-120
 line relative LR (\$PDS) S-257
 symbol, DR \$DICOMP subcommand
 U-112
 DS data set identifier, \$JOBUTIL
 command U-275
 DSCB data set control block
 statement
 coding description L-105
 equate table, DSCBEQU I-311
 for tape, internals I-99
 internals I-92
 overview L-22
 DSCBEQU L-12
 DSECT (see control block and
 parameter equate tables) L-11
 DSOPEX subroutine
 description S-322

DSR data set ready in BSCOPEN
 I-148
 DTR data transfer ready in
 BSCOPEN I-148
 DU
 dump on terminal, \$DISKUT2
 command U-144
 dump trace file on terminal,
 \$BSCUT1 command C-62
 dump
 restore volume utility
 \$MOVEVOL U-294
 storage partition, DUMP
 function C-218
 to printer
 \$DUMP utility U-163
 DP \$DISKUT2 command U-143
 DP \$TAPEUT1 command U-317
 PR \$DICOMP command U-108
 to terminal
 \$DUMP utility U-163
 DP \$TAPEUT1 command U-317
 DU \$DISKUT2 command U-143
 PR \$DICOMP command U-108
 trace file on printer, DP
 \$BSCUT1 command C-62
 trace file on terminal, DU
 \$BSCUT1 command C-62
 DUMP function C-218, I-166, I-175
 D4969, tape device handler module
 description I-82

E

E-conversion (Ew.d) L-150
 EBFLCVT, EBDIC to floating-point
 conversion I-205
 module description I-80
 EC control echo mode, \$IAMUT1
 command U-240
 ECB task control statement
 coding description L-107
 internals I-55
 overview L-42, S-30
 with SBIOCB I-128
 EDIT
 begin editing source data,
 \$EDIT1/N command U-174
 create or change data set,
 \$FEDIT option U-214
 enter edit mode, \$FONT
 command U-205
 enter edit mode, \$IMAGE
 command U-251
 edit data set subroutine examples,
 text editor I-326
 editor subcommands, \$EDIT1/N
 U-182
 EDL (see Event Driven Language)
 compiler (\$EDXASM) U-356
 instruction format I-67
 interpreter, EDXALU, module
 description I-77
 operation codes I-67
 EDXALU Event Driven Language
 interpreter description I-5,
 I-77
 EDXFLOAT floating-point operations
 module description I-79
 EDXINIT supervisor initialization
 control module I-15
 description I-81

EDXLIST host listing formatter
 U-383
 EDXSTART supervisor initialization
 task module description I-81
 EDXSVCX/EDXSVCXU task supervisor
 addr. trans. support desc I-5,
 I-76
 EDXSYS system data tables,
 description I-75
 EDXTIMER 7840 timer feature card
 module description I-80
 EDXTIMR2 4952 timer module
 description I-80
 EDXTIO terminal I/O
 EDXTIO/EDXTIOU module
 description I-78
 internals I-105
 EJECT listing control statement
 coding description L-109
 overview L-28
 eject printer page
 \$E operator command U-16
 ELSE program sequencing
 instruction
 coding description L-110,
 L-178
 overview L-34
 emulator (see
 supervisor/emulator)
 emulator command table I-13,
 I-282, I-301
 emulator functional flow I-69
 emulator setup routine I-67
 command table I-13, I-282,
 I-301
 EN end program, \$IAMUT1 command
 U-235
 END
 \$LINK control record U-396
 option selection, \$EDXASM
 command U-358
 option selection, \$EDXLIST
 command U-371
 option selection, \$S1ASM
 U-378
 primary command input, \$FSEDIT
 primary command U-221
 task control statement
 coding description L-111
 overview L-42
 end display, EP \$DICOMP
 subcommand U-118
 end-of-file, indicating with
 SETEOD S-324
 ENDATTN task control instruction
 coding description L-112
 overview L-42, S-30
 ENDDO program sequencing
 instruction
 coding description L-103,
 L-113
 overview L-34
 ENDIF program sequencing
 instruction
 coding description L-114,
 L-178
 overview L-34
 ENDPROG task control statement
 coding description L-115
 overview L-42, S-30
 ENDSEQ Indexed Access Method CALL
 coding description L-334
 overview L-27, S-147
 return codes L-335
 ENDSPOOL switch spool to print,
 \$RJE2780/\$RJE3780 C-75
 ENDTASK task control instruction
 coding description L-116
 overview L-42, S-30
 ENQ task control instruction
 coding description L-117
 internals I-60
 overview L-42, S-33
 supervisor function I-45
 ENQT terminal I/O instruction
 S-293
 coding description L-119
 overview L-44, S-47
 enqueue, task supervisor function
 (see ENQ)
 entering and editing source state-
 ments S-66, U-192
 entry points, supervisor
 Version 1.1 S-347
 Version 2 S-357
 ENTRY program module sectioning
 statement
 coding description L-121
 overview L-33
 EOF (end-of-file) L-74
 EOJ end of job, \$JOBUTIL command
 U-276
 EOP end of nested procedure,
 \$JOBUTIL command U-276
 EOR data manipulation instruction
 coding description L-122
 overview L-19
 EOT (end-of-tape) L-41
 EP end display, \$DICOMP
 subcommand U-118
 EQ (equal) L-34
 EQU data definition instruction
 coding description L-124
 overview L-17
 equate tables
 \$EDXASM compiler common area
 I-214
 BSCDDB, BSC line control
 block I-291
 CCB, terminal control block
 I-292
 DDB, disk/diskette control
 block I-308
 DDB for sensor I/O I-309
 DSCB, data set control block
 I-311
 emulator command table I-282,
 I-301
 Indexed Access Method A-19
 parameter and control block
 L-11
 program header I-312
 referencing I-30
 supervisor I-279, I-313
 TCB, task control block I-314
 ERASE terminal I/O instruction
 coding description L-126
 overview L-44, S-47
 error codes (see return codes)
 error handling
 I/O error logging S-270
 Indexed Access Method error
 exit S-178
 Remote Management Utility
 C-277
 software trace S-265
 task error exit S-33, S-268
 terminal I/O L-44
 ERRORDEF L-12

ERRORS list error option
 \$EDXASM command U-358
 \$EDXLIST command U-370
 estimating storage (see storage
 estimating)
 event control block (see ECB)
 Event Driven Language (see EDL)
 EX exercise tape, \$TAPEUT1 com-
 mand U-319
 EXEC function C-220, I-166, I-178
 EXEC load and execute program,
 \$JOBUTIL command U-277
 execute program
 EXEC function C-220
 PASSTHRU function C-225
 SHUTDOWN function C-251
 utilities (session manager)
 S-216
 executing, task supervisor exe-
 cution state I-43
 exercise tape, EX \$TAPEUT1
 command U-319
 EXFLIH command start I-125
 EXIO control instruction
 coding description L-128
 EXIODDB device data block
 description I-123
 internals I-125
 overview L-24, S-51
 EXIOCLEN, EXIO termination module
 I-126
 EXIODEV configuration statement
 S-82
 EXIOINIT, system initialization
 I-125
 EXOPEN EXIO control instruction
 coding description L-129
 internals I-125
 interrupt codes L-132
 overview L-24
 return codes L-131
 external sync DI/DO, XI/XO \$IOTEST
 command U-266
 EXTRACT, Indexed Access Method
 CALL
 coding description L-336
 overview L-26, S-148
 return codes L-337
 EXTRN program module sectioning
 statement
 coding description L-134
 overview L-33

F

F-conversion (Fw.d) L-149
 FADD data manipulation
 instruction
 coding description L-135
 overview L-19
 return codes L-136
 FAN, Multiple Terminal Manager
 CALL
 coding description C-139,
 L-366
 overview L-31
 FCA file control area, Multiple
 Terminal Manager C-143
 FCB file control block for Indexed
 Access Method
 definition A-9, A-20
 description A-11, A-21, S-194

location A-20
 map provided by FCBEQU S-155
 FCBEQU Indexed Access Method copy
 code module L-12, S-155
 FDIVD data manipulation
 instruction
 coding description L-137
 overview L-19
 return codes L-138
 FETCH Host Communications
 Facility, TP operand C-92
 fetch record (\$PDS) S-261
 fetch status, FE \$HCFUT1 command
 C-110
 file L-75
 backward space file (BSF)
 L-75
 control area (see FCA)
 control block (see FCB)
 definition L-40
 forward space file (FSF) L-75
 manager, Multiple Terminal
 Manager M-8
 tape control commands L-75
 FILEIO, Multiple Terminal Manager
 CALL
 coding description C-141,
 L-367
 internals M-9
 overview C-118, L-29
 FIND
 editor commands
 character string, \$EDIT1/N
 subcommand U-191
 character string, \$FSEDIT
 primary command U-222
 program sequencing
 instruction
 coding description L-139
 overview L-34
 FINDNOT program sequencing
 instruction
 coding description L-141
 overview L-34
 FIRSTQ queue processing
 instruction
 coding description L-143
 overview L-37, S-32
 fixed-head devices S-61
 fixed storage area, contents I-9
 floating-point
 arithmetic instruction
 equates I-283, I-303
 arithmetic instructions L-20
 binary conversions I-205
 command entries module,
 NOFLOAT, description I-79
 operations module, EDXFLOAT,
 description I-79
 return codes L-21
 FMULT data manipulation
 instruction
 coding description L-144
 overview L-19
 return codes L-145
 format
 illustrated L-5
 instruction (general) L-3
 FORMAT data formatting statement
 'A' conversion L-153
 'E' conversion L-150
 'F' conversion L-149
 'H' conversion L-152
 'I' conversion L-148
 coding description L-146

- conversion of alphanumeric data L-153
- conversion of numeric data L-148
- data conversion specifications L-146
- module names L-18
- multiple field format L-155
- overview L-18
- repetitive specification L-155
- using multipliers L-155
- X-type format L-154
- formatted screen images S-300, U-250
- formatting instructions, data L-18
- forms control
 - burst output with electronic display screens L-46
 - forms interpretation L-46
 - output line buffering L-46
 - parameters, terminal I/O L-44
 - terminal I/O L-45
- FORTRAN IV
 - execution requirements S-24
 - link editing S-71
 - overview S-6
 - program preparation requirements S-24
 - use with Multiple Terminal Manager C-197
- FPCONV data manipulation instruction
 - coding description L-157
 - overview L-19
- free pool in Indexed Access Method L-27
- free space
 - definition S-148
 - estimating S-168
 - in Indexed Access Method L-27
- free space entry I-90
- FREEMAIN storage allocation function I-25
- FSE free space entry I-90
- FSR (forward space record) L-75
- FSUB data manipulation instruction
 - coding description L-159
 - index registers L-160
 - overview L-19
 - return codes L-160
- FTAB, Multiple Terminal Manager CALL
 - coding description C-138, L-372
 - overview C-124, L-31
 - return codes L-373
- full-screen static configuration S-293
- full-screen text editor host and native, \$FSEDIT U-209
- full-word boundary requirement
 - DO L-34
 - IF L-34
 - PROGRAM L-225
- function process overlays I-162
- function process subroutines I-162, I-170
 - new subroutines I-187

function table I-164, I-167

G

- GE (greater than or equal) L-34
- general instruction format L-3
- generating the supervisor S-115
- GENxxxx macro I-120
- GET Indexed Access Method CALL
 - coding description L-338
 - overview L-27, S-147
 - return codes L-340
- GETEDIT data formatting instruction
 - coding description L-162
 - overview L-18
- GETMAIN storage allocation instruction I-25
- GETPAR3 I-69
- GETSEQ Indexed Access Method CALL
 - coding description L-342
 - overview L-27, S-147
 - return codes L-343
- GETSTORE TERMCTRL function L-288
- GETTIME timing instruction
 - coding description L-167
 - overview L-50, S-32
- GETVAL subroutine, \$EDXASM I-234
- GETVALUE terminal I/O instruction
 - coding description L-169
 - overview L-44, S-47
- GIN graphics instruction
 - coding description L-172
 - overview L-26
- global area, \$EDXASM I-224
- GLOBAL ATTNLIST L-61
- GO activate stopped task, \$DEBUG command U-93
- GOTO
 - change execution sequence, \$DEBUG command U-94
 - coding sequencing instruction coding description L-173
 - overview L-34
- graphics
 - conversion algorithm I-201
 - functions overview L-26
 - hardware considerations C-6, C-300
 - instructions L-26
 - CONCAT L-72
 - GIN L-172
 - PLOTGIN L-210
 - SCREEN L-270
 - XYPLOT L-324
 - YTPLOT L-325
 - requirements L-26
 - terminals S-46
 - utilities
 - \$DICOMP U-105
 - \$DIINTR U-127
 - \$DIUTIL U-150
 - session manager S-216, U-40
 - summarized S-64, U-5
- GT (greater than) L-34

H

H-conversion L-152
 hardcopy function for terminals
 PF6 I-114, U-7
 hardware levels S-30
 HCF (see Host Communications Facility)
 HDR1 tape label S-239
 header labels, tape S-235
 header record
 Remote Management Utility
 C-209
 header record format, text editor
 I-323
 HELP list debug commands, \$DEBUG
 command U-94
 higher-level index block S-197
 horizontal tabs, defining with
 \$IMAGE U-252
 host assembler U-382
 Host Communications Facility
 C-81, I-153
 data set naming conventions
 C-82
 Program Preparation
 System/370 I-153, U-382
 TPCOM module description I-81
 utility program, \$HCUT1 C-107
 host program, Remote Management
 Utility C-205
 host system considerations C-83
 HOSTCOMM configuration statement
 S-83
 HX display hex words, \$DICOMP
 subcommand U-118

I

I
 initialization, \$INITDSK com-
 mand U-257
 insert line, \$FSEDIT line
 command U-229
 I-conversion (Iw) L-148
 I/O device instruction level L-24
 I/O error logging
 data set list utility,
 \$DISKUT2 U-142
 device table S-276
 invoking S-273, U-292
 log control record S-276
 log data set U-292
 LOG macro
 equates S-278
 syntax S-272
 printing the errors S-275
 recording the errors S-270
 tape log entries S-245
 utility, \$LOG U-292
 I/O functions
 disk/diskette I-95, L-22
 summarized S-46
 EXIO control I-123, L-24
 summarized S-51
 HOSTCOMM configuration
 statement L-39, S-83
 overview S-45
 sensor I-127
 summarized S-51

tape L-40, L-75
 terminal S-46
 timers L-50, S-32
 I/O instructions
 disk L-22
 diskette L-22
 tape L-40
 IACB indexed access control block
 built by connecting data set
 S-159
 definition A-20
 description A-35
 location A-14
 IAM Indexed Access Method link
 module S-155
 IAMEQU Indexed Access Method copy
 code module L-12, S-155
 IDCBC EXIO control statement
 coding description L-175
 overview L-24
 IDCHECK function C-223, I-166,
 I-177
 identification, verify
 host system C-223
 IDCHECK function C-223
 remote system C-223
 IF program sequence instruction
 coding description L-177
 overview L-34
 II insert block, \$FSEDIT line
 command U-231
 IIB interrupt information byte,
 Multiple Terminal Manager C-128
 IM insert member
 \$DICOMP subcommand U-118
 \$PDS S-257
 image dimensions, define, DIMS
 \$IMAGE command U-251
 image store U-205
 immediate action routines I-46
 binary synchronous access
 method I-149
 specifying maximum number
 S-88
 task supervisor I-48
 immediate data L-4
 IN
 initialize data base, \$DIUTIL
 command U-157
 insert or delete elements,
 \$DICOMP command U-107
 INCLUDE \$LINK control record
 U-398
 INCLUDE statement requirement
 (EXTRN) L-134
 index block A-20, A-33
 overview S-151
 index entry A-12
 index record contents, text
 editor I-323
 index registers
 floating-point operations
 using L-21
 integer operations using L-19
 software introduction L-6
 indexed access control block (see
 IACB/ICB)
 Indexed Access Method L-26, L-327
 \$IAM load module S-155
 \$IAMUT1 utility U-235
 overview S-148
 parameters S-187
 used in data set
 reorganization S-166
 application program

preparation
 \$JOBUTIL procedure S-158
 link edit control S-158
 CALL instruction syntax L-68, S-146
 CALL processing A-4
 coding instructions L-327
 control block linkages A-15
 control flow A-3
 data block location
 calculation A-9
 devices supported by S-146
 diagnostic aids A-10
 I/O requests
 DELETE L-329, S-147
 DISCONN L-332, S-148
 ENDSEQ L-334, S-147
 EXTRACT L-336, S-148
 GET L-338, S-147
 GETSEQ L-341, S-147
 LOAD L-344, S-147
 PROCESS L-347, S-147
 PUT L-350, S-147
 PUTDE L-352, S-147
 PUTUP L-354, S-147
 RELEASE L-356, S-147
 IAM link module S-155
 operation S-148
 overview L-27, S-145
 performance S-205
 program preparation procedure S-155
 record processing A-6
 request processing A-5
 request verification A-10
 storage requirements S-204
indexed applications, planning and designing
 connecting and disconnecting data sets S-159
 handling errors
 data-set-shut-down condition S-179
 deadlocks S-180
 error exit facilities S-178
 long-lock-time condition S-180
 system function return codes S-179
 loading base records S-160
 processing indexed data sets
 delete S-165
 direct read S-161
 direct update S-162
 extract S-165
 insert S-146
 sequential read S-162
 sequential update S-146
 resource contention S-181
indexed data set
 base records S-149
 building U-247
 concatenating with ALTIAM subroutine S-167
 control block arrangement A-8
 creation with \$IAMUT1 utility U-236
 formatting S-187
 procedure S-156
 design A-7
 determining size and format U-247
 format
 blocks S-192
 cluster S-200
 data block S-194
 file control block (FCB) S-151, S-194
 free blocks S-200
 free pool S-203
 free records S-200
 free space S-184
 higher-level index block S-197
 index S-195
 index block S-194
 introduction S-151
 last cluster S-203
 primary-level index block (PIXB) S-152, S-195
 relative block number (RBN) S-152
 reserve blocks S-201
 reserve index entries S-202
 second-level index block (SIXB) S-152, S-197
 sequential chaining S-203
 loading and inserting records S-150
 maintenance
 backup and recovery S-165
 deleting S-167
 dumping S-167
 recovery without backup S-166
 reorganization S-166
 overview S-148
 physical arrangement A-8
 preparing the data
 defining the key S-166
 estimating free space S-168
 selecting the block size S-167
 putting records into S-149
 RBN, relative block number A-7, A-12
 record locking S-146, S-160
 verification A-11
 indexed data set, defining U-237
 indexed file (see Indexed Access Method)
 indexing, address feature L-6
 initial program load (see also IPL) I-15
 initialization
 automatic application S-129
 disk (4962) U-68, U-73
 disk (4963) U-68, U-78
 diskette (4964,4966) U-68
 libraries, \$INITDSK utility U-256
 modules I-16
 nucleus I-15
 Remote Management Utility, internals I-166, I-171
 tape, \$TAPEUT1 utility U-322
 task I-15
 initialize data base, IN \$DIUTIL command U-157
 initializing secondary volumes S-132
 INITMODS, initialization modules I-16
 INITTASK, initialization task I-15
 input, terminal I/O L-46

Input Buffer, Multiple Terminal Manager C-116
 contents during 4978/4979/3101
 buffer operation C-129
 description C-116
 input data parsing, description of I-218
 Input Error function I-166, I-182
 input/output (see I/O)
 input output control block (see IOCB)
 INPUT switch to input mode, \$EDIT1/N editor subcommand U-192
 insert
 block, II \$FSEDIT line command U-231
 elements, IN \$DICOMP command U-107
 line, I \$FSEDIT line command U-229
 member, IM \$DICOMP subcommand U-118
 instruction address register (see IAR)
 instruction and statements - overview L-15
 instruction definition and checking (\$EDXASM) I-241
 instruction format, Event Driven Language I-67, L-3
 instruction format, general L-3
 instruction operands L-3
 integer and logical instructions L-19
 interactive program debugging S-67, U-82
 interface routines, supervisor I-61
 interprocessor communications C-29
 interprogram dialogue S-282
 interrupt, from EXIO device I-125
 interrupt information byte (see IIB)
 interrupt line S-313
 interrupt servicing I-46, I-113
 INTIME timing instruction
 coding description L-181
 overview L-50, S-32
 introduction to EDL L-1
 invoking the loader I-23
 invoking the session manager U-27
 invoking the utilities U-47
 IOCB terminal I/O instruction
 coding description L-183
 constructing, for formatted screen (\$IMDEFN) S-301
 overview L-44, S-47
 structure S-296
 terminal I/O instruction L-183
 TERMINAL statement converted to S-96
 IODEF sensor based I/O statement U-364
 coding description L-185
 overview L-39, S-51
 SPECPI - process interrupt user routine L-189
 IOLOADER, function of I-127
 IOLOADER/IOLOADRU sensor based I/O
 init. module desc. I-78
 IOR data manipulation instruction
 coding description L-191
 overview L-19

IPL
 automatic application initialization and restart S-129
 messages U-421
 date and time U-425
 IPL operation U-421
 load utility location U-424
 sensor I/O status check U-424
 storage map generation U-423
 tape initialization U-423
 volume initialization U-422
 procedure U-421
 IPLSCRN, Multiple Terminal Manager C-125

J

job U-278
 job control statement U-278
 JOB job identifier, \$JOBUTIL command U-278
 job stream processor, \$JOBUTIL S-69, U-271
 job stream processor utilities (session manager) S-216
 JP
 jump (\$PDS) S-255
 to address, \$DICOMP subcommand U-118
 JR jump reference, \$DICOMP subcommand U-118
 JUMP, \$JOBUTIL command U-279
 jump reference, JR \$DICOMP subcommand U-118
 jump to address, JP \$DICOMP subcommand U-118

K

key (see program function (PF)
 keys
 keyboard and ATTNLIST tasks, terminal I/O L-47
 keyboard define utility for 4978, \$TERMUT2 U-339
 KEYS list program function keys \$IMAGEDEFN command U-253
 keyword operand L-5

L

LA
 display directory, \$DIUTIL command U-158
 list all members, \$DISKUT1 command U-135, U-136
 list terminal assignment, \$TERMUT1 command U-336
 label L-3
 field L-3
 syntax description L-4

LABEL end jump, \$JOBUTIL command U-280
 labels, tape (see tape)
 LABELS subroutine, \$EDXASM I-238
 LACTS list all members CTS mode, \$DISKUT1 command U-135
 language control data set, \$EDXASM I-221, U-357
 LASTQ queue processing instruction
 coding description L-191
 overview L-37, S-32
 layers, terminal I/O I-108
 LB display characters
 \$DICOMP display character subcommand U-119
 \$PDS S-252
 LC load control store, \$TERMUT2 command U-342
 LD
 list all hardware devices, \$IOTEST command U-269
 list data members, \$DISKUT1 command U-138
 LDCTS list data members CTS mode, \$DISKUT1 command U-135
 LE (less than or equal) L-34
 level status block (see LSB)
 LEWORK1 work data set for \$LINK U-400
 LEWORK2 work data set for \$LINK U-400
 LH display member header, \$DIUTIL command U-159
 LI
 draw line \$DICOMP subcommand U-120
 draw line \$PDS S-253
 load image store, \$TERMUT2 command U-342
 library
 definition S-52
 directory, disk or diskette I-87
 origin S-60
 line
 commands, \$FSEDIT U-229
 continuation, source statement L-4
 editing, \$EDIT1/N U-203
 pointer reposition (see move line pointer)
 source line continuation U-361
 LINK, Multiple Terminal Manager CALL
 coding description C-131, L-374
 internals M-9
 overview C-115, L-29
 link edit process, \$LINK U-394
 autocall option U-393
 building an EDX supervisor U-394
 combining program modules U-392
 control records U-396
 elimination of duplication control sections U-393
 formatting modules for \$UPDATE U-392
 input to \$LINK U-396
 multiple control sections U-392
 object module record format U-407
 output from \$LINK U-403
 storage map U-393
 link edit program object modules U-390
 link module, Indexed Access Method S-155
 linkage editor S-71, U-353
 LINKON, Multiple Terminal Manager CALL
 coding description C-132, L-376
 internals M-9
 overview C-115, L-29
 list
 active programs, \$A operator command U-11
 breakpoints and trace ranges, BP \$DEBUG command U-92
 characters, LB \$DICOMP subcommand U-119
 data members, LD \$DISKUT1 command U-138
 data members, LDCTS \$DISKUT1 command U-135
 data set
 BROWSE \$FSEDIT option U-213
 LP \$DISKUT2 command U-143
 LU \$DISKUT2 command U-146
 status, ST \$DIUTIL command U-162
 date/time, \$W operator command U-25
 date/time, TD \$DICOMP subcommand U-124
 devices, LD \$IOTEST command U-269
 end, EP \$DICOMP subcommand U-117
 error specification, ERRORS \$EDXASM command U-358
 hardware configuration, LD \$IOTEST command U-264
 insert mask, MASK \$FSEDIT line command U-232
 member, LM \$DISKUT1 command U-138
 member, PR \$DICOMP command U-108
 member header, LH \$DIUTIL command U-159
 members, all
 LA \$DISKUT1 command U-135
 LA \$DIUTIL command U-158
 LACTS \$DISKUT1 command U-135
 processor program, \$EDXLIST U-370
 program function key codes, \$PFMAP utility U-301
 program function keys, KEYS \$IMAGE command U-253
 program members, LP \$DISKUT1 command U-139
 program members, LPCTS \$DISKUT1 command U-135
 status of all tasks, WHERE \$DEBUG command U-102
 storage, \$D operator command U-15
 terminal
 names/types/addresses, LA \$TERMUT1 command U-335
 variables, VA \$DICOMP

subcommand U-125
 volume information, VI \$IOTEST
 command U-270
LIST commands
 data set
 LIST \$EDIT1/N command
 U-193
 LIST \$FSEDIT option U-217
 display lines of text,
 \$EDIT1/N editor subcommand
 U-193
 display storage or registers,
 \$DEBUG command U-95
 lines of text, LIST \$EDIT1/N
 editor command U-176
 list device option, \$EDXASM
 command U-358
 list device option, \$EDXLIST
 command U-370
 obtain full listing, LIST
 \$EDXASM command U-358
 print data set, \$EDIT1/N
 command U-176
 print data set, \$FSEDIT
 option U-217
 registers, LIST \$DEBUG
 command U-95
 storage, LIST \$DEBUG command
 U-95
 listing control functions U-29
 listing control instructions
 EJECT L-109
 overview L-28
 PRINT L-216
 SPACE L-275
 TITLE L-308
 LISTP list to \$SYSPRTR, \$DISKUT1
 command U-135
 LISTT list to terminal, \$DISKUT1
 command U-135
 LL list log data set, \$DISKUT2
 command U-145
 LM list member, \$DISKUT1 command
 U-138
 LO load indexed file, \$IAMUT1
 command U-241
LOAD
 Indexed Access Method CALL
 coding description L-344
 connect file S-159
 overview L-27, S-146
 return codes L-346
 task control instruction
 coding description L-194
 internals I-24
 overview L-42
 return codes L-199
 used with automatic
 initialization S-129
 used with overlays S-40
 load mode S-149
 load point defined L-40
 load program
 \$L operator command I-23,
 U-17
 automatic initialization
 S-129
 EXEC \$JOBUTIL command U-277
 loading overlays I-22
 loading programs I-19
 locate data sets and overlay
 programs, \$PREFIND U-302
 LOCATE locate requested line
 number \$FSEDIT primary comman
 U-223

location dictionary I-250
 lock
 locks, block and record A-16
 locks, file A-17
 record S-146
 LOCK TERMCTRL function L-288
LOG
 I/O error logging macro S-271
 job processor commands,
 \$JOBUTIL command U-281
 log data set for I/O errors U-292
 logical end-of-file on disk S-324
 logical screens S-293
 logon menu for session manager
 S-212, U-27
 long-lock-time condition S-180
 low storage
 during IPL I-16
 during program load I-20
LP
 list data set on printer,
 \$DISKUT2 command U-144
 list program members, \$DISKUT1
 command U-139
 LPCTS list program members CTS
 mode, \$DISKUT1 command U-135
 LR draw line relative
 \$DICOMP subcommand U-121
 \$PDS S-257
LS
 list space, \$DISKUT1 command
 U-140
 list supervisor configuration,
 \$IOTEST command U-270
 LSB level status block I-52,
 U-427
 LT (less than) L-34
 LU list data set on console,
 \$DISKUT2 command U-146
 LV list through volumes, \$DISKUT1
 U-141

M

M move line, \$FSEDIT line command
 U-233
 macro assembler
 internal overview \$S1ASM
 I-253
 overview S-9
 macro library S-6
 macro library/host S-5
 magazine diskette (see 4966
 diskette magazine unit)
 magnetic tape (see tape)
 MASK display insert mask, \$FSEDIT
 line command U-232
 master control block (see MCB)
 Mathematical and Functional Sub-
 routine Library S-6
 MCB master control block
 \$PDS S-260
 definition A-20
 description A-28
 MD move data base, \$DIUTIL
 command U-160
 member area S-250
 member control block (MCB) S-260
MENU
 Multiple Terminal Manager
 CALL
 coding description C-137,

L-377
 internals M-9
 overview C-116, L-29
 return to primary option,
 \$FSEDIT U-223
 menu-driven U-2
 menus
 (see option selection menu)
 (see parameter selection
 menu)
 (see primary menu)
 (see primary option menu)
 (see secondary option menu)
 (see session manager, menus)
 (see transaction selection
 menu)
 MENUSCRN, Multiple Terminal Manag-
 er C-126
 MERGE merge data, \$FSEDIT option
 U-217
 message, PRINTTEXT instruction
 L-217
 message sending utility, \$TERMUT3
 U-344
 messages U-421
 error U-427
 \$DUMP U-431
 \$LOG U-432
 \$RMU U-433
 \$TRAP U-435
 program check U-427
 system program check
 U-429
 IPL (see IPL messages)
 Multiple Terminal Manager
 C-178
 Remote Management Utility
 C-279
 minimum execution system config-
 uration S-22
 minimum program preparation
 requirements S-22
 mirror image
 description C-7, S-109
 in TERMINAL configuration
 statement S-101
 mixed precision combinations L-20
 MM move block, \$FSEDIT line
 command U-233
 modified data S-307
 modify character image tables
 U-339
 modify character string, CHANGE
 \$EDIT1/N editor subcommand
 U-184
 \$FSEDIT primary command U-219
 modify default storage allocation,
 \$DISKUT2 U-149
 modifying an existing data set,
 \$FSEDIT U-215
 modifying TERMINAL statement for
 new I/O terminal I-119
 module descriptions
 \$S1ASM I-269
 supervisor I-75
 module names and entry points,
 supervisor
 Version 1.1 S-347
 Version 2 S-357
 move
 block, MM \$FSEDIT line com-
 mand U-233
 line pointer
 BOTTOM \$EDIT1/N editor
 subcommand U-183
 DOWN \$EDIT1/N editor
 subcommand U-189
 TOP \$EDIT1/N editor
 subcommand U-200
 UP \$EDIT1/N editor
 subcommand U-201
 tape U-324
 text
 \$EDIT1/N editor subcom-
 mand U-195
 \$FSEDIT line command
 U-233
 volumes on disk or diskette,
 \$MOVEVOL utility U-294
 MOVE data manipulation
 instruction
 coding description L-201
 overview L-19
 MOVEA data manipulation
 instruction
 coding description L-204
 overview L-19
 MOVEBYTE subroutine, \$EDXASM
 I-236
 MP
 move beam, \$DICOMP subcommand
 U-121
 move position (\$PDS) S-253
 MT move tape, \$TAPEUT1 command
 U-324
 MTMSTORE file, Multiple Terminal
 Manager C-120, C-171, M-12
 MTMSTR, Multiple Terminal Manager
 C-169, C-170, M-12
 multiple field format L-155
 multiple program execution I-36
 multiple program structure S-26
 multiple-task programs I-33
 Multiple Terminal Manager
 accessing the terminal envi-
 ronment block C-139, M-22
 application program C-116
 application program languages
 L-30
 application program manager
 C-119, M-4
 automatic OPEN/CLOSE C-140,
 M-8
 CALL
 ACTION C-130, L-360
 BEEP C-137, L-361
 CDATA C-139, L-362
 CHGPAN C-135, L-364
 CYCLE C-132, L-365
 FAN C-139, L-366
 FILEIO C-141, L-367
 FTAB C-138, L-372
 LINK C-131, L-374
 LINKON C-132, L-376
 MENU C-137, L-377
 SETCUR C-137, L-378
 SETPAN C-134, L-379
 WRITE C-133, L-381
 coding instructions L-359
 components C-123, M-4
 considerations for 3101
 terminal C-122
 data files C-120
 MTMSTORE file C-120,
 C-171, M-12
 PRGRMS volume C-120,
 C-173
 SCRNS volume C-120, C-173
 TERMINAL volume C-120,
 C-171

direct file request types
 C-144, L-370
 disk file support C-140
 distribution and installation
 C-161
 dynamic screen modification
 and creation C-136
 file control area C-142
 file I/O considerations (Event
 Driven Executive) C-146
 file management C-118, M-8
 FILEIO, disk file support
 C-140
 FILEIO Indexed Access Method
 considerations C-148
 fixed screen formats C-125
 functions (callable routines)
 C-117, C-124
 indexed file request types
 C-144, L-369
 indexed file support C-140,
 L-367
 initialization programs
 C-119, C-158, M-4, M-6
 Input Buffer C-116, C-127
 Input Buffer Address C-116
 Input Buffer during
 4978/4979/3101 buffer oper-
 ation C-127
 interrupt information byte
 C-128
 messages C-178
 module list M-4
 operation C-115
 Output Buffer C-116
 Output Buffer Address C-127
 Output Buffer during
 4978/4979/3101 buffer oper-
 ation C-128
 overview L-29, S-10
 program management C-115, M-4
 program preparation
 COBOL C-166
 Event Driven Language
 C-164
 FORTRAN C-165
 PL/I C-167
 programming considerations
 COBOL C-153
 Event Driven Language
 C-151
 FORTRAN C-152
 PL/I C-155
 return codes (FILEIO) C-145,
 L-371
 screen definition C-121
 screen formats C-125
 IPLSCRN C-125
 MENUSCRN C-126
 SCRNSREP C-126
 SIGNONSC C-126
 screen panel manager M-7
 SIGNON/SIGNOFF C-156
 SIGNONFL C-174
 storage requirements C-168
 swap out data set C-116
 system generation
 considerations C-169
 data set requirements
 C-171, C-175
 volume requirements C-169
 terminal environment block
 (TEB) C-128, M-13
 TERMINAL file C-124, C-172
 terminal manager C-121

terminal/screen management
 C-117
 terminal server C-119, M-7
 terminal support C-114, C-126
 transaction oriented
 applications C-121
 user application programs
 C-124
 utilities C-159
 DISCONNECT turn off
 specified terminals
 C-159
 programs report C-159
 RECONNECT turn on
 specified terminals
 C-159
 screens report C-160
 terminal activity report
 C-159
 work areas, control blocks and
 tables M-11
 buffer areas M-15, M-29
 common area M-11, M-25
 file table M-15, M-27
 MTMSTORE data set M-12
 program table M-14, M-21
 screen table M-14, M-21
 terminal environment block
 (TEB) M-13, M-22
 terminal table M-13, M-21
 MULTIPLY data manipulation
 instruction
 coding description L-205
 overview L-19
 precision table L-206
 multiprogramming
 automatic application initial-
 ization S-129
 design feature S-13
 multitasking I-42

N

NE (not equal) L-34
 newline subroutine, terminal I/O
 I-112
 NEXTQ queue processing
 instruction
 coding description L-207
 overview L-37, S-32
 NOFLOAT floating-point command
 entries module description I-79
 NOLIST no list option, \$EDXASM
 command U-358
 NOMSG message suppression,
 \$JOBUTIL command U-282
 non-compressed byte string S-309
 non-labeled tapes
 description S-241
 layout S-242
 processing S-243
 NOTE disk/tape I/O instruction
 coding description L-209
 overview L-22
 notify of an event (see POST)
 NQ reset prompt mode, \$COPYUT1
 command U-64
 nucleus initialization I-15
 null character U-253
 NULL define null representation
 \$IMAGE command U-253

null representation, defining U-253
number representation conversion (see conversion)

O

object data set for \$EDXASM U-357
object module record format, \$LINK U-407
object text elements, format of, \$EDXASM I-215
OFF (set tape offline) L-75
OFF remove breakpoints and trace ranges, \$DEBUG command U-97
OLE operand list element, \$EDXASM format of I-216
 in instruction parsing (\$EDXASM) I-220
 used in \$IDEF I-241
online debug aids S-67
op (operation field) L-3
OPCHECK subroutine, \$EDXASM I-232
opcode table, instruction parsing (\$EDXASM) I-220, I-223
open a data set
 disk or diskette I-90
 tape I-99
open EXIO device, EXOPEN I-125
open member (\$PDS) S-261
OPENIN Host Communications Facility, TP operand C-93
OPENOUT Host Communications Facility, TP operand C-94
operands
 defined L-3
 keyword L-5
 parameter naming (Px) L-8
operating conventions, supervisor program I-67
operating environment S-22
operation code, instruction parsing (\$EDXASM) I-220
operation codes, Event Driven Language I-68
operations using index registers L-20
operator commands S-63, U-9
operator signals, terminal I/O L-49
option selection menu U-33
optional features support L-15
OTE define object text element \$EDXASM instruction I-227
OUTPUT \$LINK control record U-399
Output Buffer, Multiple Terminal Manager C-116, C-128
 contents during 4978/4979/3101 buffer operation C-129
 definition M-29
overflow L-20
overlay function processor table I-167, I-220
overlay program S-40
 instructions, \$EDXASM I-259
 loading I-22
 locating, \$PREFIND U-302
 subroutines, \$EDXASM I-231
 user I-38
overlay program execution I-38
overlay selection, instruction parsing (\$EDXASM) I-223

overlay table I-167, I-220
overview
 data definition statements L-17
 data formatting instructions L-18
 data format module names L-18
 data manipulation
 instructions L-19
 data representation L-19
 mixed-precision operations L-20
 operations using index registers L-20
 overflow L-20
 vector L-19
 disk I/O instructions L-22
 EXIO control instructions L-24
 floating-point arithmetic L-20
 floating-point arithmetic instructions L-20
 data representation L-21
 operations using index registers L-21
 return codes L-21
 graphics instructions L-26
 Indexed Access Method instructions L-27
 instructions and statements L-15
 integer and logical instructions L-19
 listing control statements L-28
 Multiple Terminal Manager instructions L-29
 program control statements L-32
 program module sectioning statements L-33
 program sequencing instructions L-34
 queue processing L-37
 sensor-based I/O statements L-39
 single-precision L-19
 system configuration statements L-39
 tape I/O instructions L-40
 task control instructions L-42
 terminal I/O instructions L-44
 timing instructions L-50

P

P/I (see process interrupt)
PA patch, \$DISKUT2 command U-147
page eject S-63, U-16
parameter equate tables L-11
parameter naming operands in the instruction format L-8
parameter passing, Remote Management Utility C-212
parameter selection menu U-33
parameter tables, control block and L-11

PARM program parameter passing,
 \$JOBUTIL command U-283
 parsing, input data (\$EDXASM)
 I-218
 partition assignment changing, \$CP
 operator command U-14
 partitioned data sets S-247
 partitions S-42
 PASSTHRU function
 conducting a session C-227
 establishing a session C-225
 internals I-166, I-179
 overview C-225
 programming considerations
 C-237
 sample program C-265
 types of records C-232
 virtual terminals C-239
 Passthru record C-209
 patch
 disk/diskette, PA \$DISKUT2
 command U-147
 Remote Management Utility
 defaults C-283
 storage, \$P operator command
 S-63, U-18
 storage or registers, PATCH
 \$DEBUG command U-98
 PATCH modify storage or registers,
 \$DEBUG, command U-98
 PAUSE operator intervention,
 \$JOBUTIL command U-284
 PC plot curve
 \$DICOMP subcommand U-119
 from plot curve data member
 (\$PDS) S-255
 PD pulse DO, \$IOTEST command
 U-265
 PF, code TERMCTRL function L-288
 PF keys (see program function
 keys)
 phase execution and loading,
 \$S1ASM I-255
 PI process interrupt (see process
 interrupt) U-267
 PID program directory S-27
 PIXB (see primary-level index
 block)
 PL/I
 execution requirements S-24
 link editing S-71
 overview S-8
 program preparation
 requirements S-23
 supported by Multiple Terminal
 Manager C-200
 PL plot data, \$DICOMP subcommand
 U-122
 plot control block (see PLOTGB)
 plot curve data member (\$PDS)
 S-251
 PLOTGB graphics plot control
 block L-210
 PLOTGIN graphics instruction
 coding description L-210
 overview L-26
 POINT
 disk/tape instruction
 coding description L-212
 overview L-22, S-54
 point-to-point (BSC) S-65
 point-to-point vector drawing
 S-46
 POST
 post an event, \$DEBUG command
 U-100
 task control instruction
 coding description L-213
 internals I-58
 overview L-42, S-34
 supervisor function I-46
 power outage, restoring after
 S-129
 PR print member, \$DICOMP command
 U-108
 precision L-19
 floating-point arithmetic
 L-21
 integer and logical L-19
 precision combinations,
 allowed L-20
 precision table
 ADD L-53
 ADDV L-54
 DIVIDE L-101
 MULTIPLY L-206
 overview L-20
 SUBTRACT L-284
 prefind U-302
 PREPARE IDCB command L-175
 PRGRMS volume, Multiple Terminal
 Manager C-120, C-173
 primary
 commands, \$FSEDIT U-218
 option menu, \$FSEDIT U-213
 option menu, session manager
 S-218, U-35
 task
 internals I-29
 overview S-29
 volume S-60
 primary-level index block
 description S-195
 overview S-151
 PRINDATE terminal I/O instruction
 coding description L-215
 overview L-44, S-47
 timer-related instruction
 S-33
 PRINT listing control statement
 coding description L-216
 overview L-28
 print member, PR \$DICOMP command
 U-108
 PRINTTEXT terminal I/O instruction
 coding description L-217
 overview L-44, S-47
 return codes L-219
 PRINTIME terminal I/O instruction
 coding description L-221
 overview L-44, L-50, S-47
 timer-related instruction
 S-33
 PRINTNUM terminal I/O instruction
 coding description L-222
 overview L-44, S-47
 PRINTON define terminal name,
 \$RJE2780/\$RJE3780 C-75
 priority
 assigned to tasks S-29
 design feature S-13
 illustrated S-38
 internals I-31
 task L-226, L-286
 PROC identify nested procedure,
 \$JOBUTIL command U-286
 procedures, session manager (see
 session manager)
 PROCESS Indexed Access Method
 CALL

coding description L-347
 overview L-27, S-147
 return codes L-349
 process interrupt
 control block (SBIOCB) I-128
 description S-48
 IODEF statement L-189
 IOTEST command U-267
 supported by sensor I/O S-15
 user routine (SPECPI) L-189
 process mode
 definition S-150
 processing compiler output with
 \$LINK or \$UPDATE U-360
 processor status word (see PSW)
 PROGEQU L-13
 program
 equates I-312
 assembly/compilation U-352
 control L-32
 disabling S-102
 entry (see \$FSEEDIT, \$EDIT1/N)
 function (PF) keys L-47
 internals I-108
 listing, KEYS \$IMAGE
 command U-253
 listing 4978, \$PFMAP
 utility U-301
 when using \$FONT edit
 mode U-206
 when using \$FSEEDIT U-211
 when using \$IMAGE edit
 mode U-255
 when using session
 manager U-28
 header I-30
 identifier, \$JOBUTIL command
 U-287
 internal processing I-30
 library update (see \$UPDATE)
 load process, \$PREFIND U-302
 loading (see also LOAD) I-19
 module sectioning functions
 L-33
 organization S-29
 sequencing functions L-34
 structure S-29
 termination, EXIO I-126
 types I-32
 program check error messages
 U-427
 program execution via Remote Man-
 agement Utility
 EXEC function C-220
 PASSTHRU function C-225
 SHUTDOWN function C-251
 PROGRAM identifier, \$JOBUTIL
 command U-287
 program preparation
 \$EDXASM I-211, U-356
 \$\$IASM I-253, U-372
 host assembler U-382
 of Remote Management Utility
 I-184
 summary S-18
 usage example S-367
 Program Preparation Facility
 description S-71
 overview S-5
 program preparation utilities
 U-351
 program preparation utilities
 (session manager) OU-36, S-214
 program/storage manager, Multiple
 Terminal Manager M-4
 program structure S-36
 internals I-33
 program/task concepts I-29, S-29
 PROGRAM task control instruction
 coding description L-225
 internals I-30
 overview L-42, S-31
 PROGSTOP task control statement
 coding description L-234
 overview L-42, S-31
 prompting and advance input,
 terminal I/O L-46
 protected field S-307, U-253
 protocol, BSC transmission I-156
 PSW processor status word U-430
 PU PUNCHO/PUNCHS function,
 \$RJE2780/\$RJE3780 reset type
 C-76
 pulse a digital output address, PD
 \$IOTEST command U-264
 PUNCHO/PUNCHS define output file,
 \$RJE2780/\$RJE3780 C-75
 purpose of EDL L-1
 PUT Indexed Access Method CALL
 coding description L-350
 overview L-27
 return codes L-351
 PUTDE Indexed Access Method CALL
 coding description L-352
 overview L-27
 return codes L-353
 PUTEDIT data formatting
 instruction
 coding description L-236
 overview L-18
 return codes L-238
 PUTSTORE TERMCTRL function L-288
 PUTUP Indexed Access Method CALL
 coding description L-354
 overview L-27
 return codes L-355
 Px L-8

Q

QCB task control statement S-33
 coding description L-240
 overview L-42
 queue control block I-45,
 I-54
 QD queue descriptor I-64, L-37
 QE queue entry
 functions I-64
 overview L-37
 processing S-32
 QUALIFY modify base address,
 \$DEBUG command U-101
 QUESTION terminal I/O instruction
 coding description L-242
 overview L-44, S-47
 queueable resource S-33
 queue control block (see QCB)
 queue descriptor (see QD)
 queue entry (see QE)
 queue processing I-64
 queue processing instructions
 L-37
 queue processing support module,
 QUEUEIO, description I-81
 QUEUEIO queue processing support
 module description I-81

R

RA reassign address, \$TERMUT1 command U-336
 random access S-53
 random work file operation, \$S1ASM I-260
 RCB (see Remote Management Utility, control block)
 RDCURSOR terminal I/O instruction coding description L-244 overview L-44, S-47
 RE
 copy from basic exchange data set, \$COPY command U-59
 rename, \$TERMUT1 command U-337
 rename member, \$DISKUT1 command U-135, U-136
 rename member, \$DIUTIL command U-161
 reset parameters, \$IAMUT1 command U-243
 restore 4974 to standard character set, \$TERMUT2 U-339
 read
 analog input, AI \$IOTEST U-268
 character image table from 4978, GET \$FONT U-206
 data set into work file \$EDIT1 U-177 \$EDIT1N U-176 \$FSEdit U-216
 digital input, DI \$IOTEST command U-266
 digital input using external sync U-266
 Host Communications Facility, TP operand C-95
 IDCB command L-175
 operations (BSC) I-157
 program, RP command \$UPDATE U-410 \$UPDATEH U-419
 READ instruction
 disk/diskette return codes L-249, U-455
 disk/diskette/tape I/O instruction coding description L-245 overview L-22
 tape return codes L-249, U-456
 READDATA read data from host, \$HCFUT1 command C-108
 READID IDCB command L-175
 READOBJ read object module, \$HCFUT1 command C-109
 READTEXT terminal I/O instruction coding description L-251 overview L-44, S-48
 return codes L-255
 return codes, virtual terminal communications L-256
 ready a task supervisor execution state I-43
 READ1 IDCB command L-175
 READ80 read 80 byte records, \$HCFUT1 command C-109
 real image ACCA terminals C-7

realtime data member \$PDS S-251
 RT \$DICOMP subcommand U-124
 RECEIVE function overview C-243 sample program C-262
 RECONNECT Multiple Terminal Manager utility C-120, C-159
 record
 blocking, Remote Management Utility C-211
 definition S-53
 exchange, Remote Management Utility C-208
 format for object module, \$LINK U-407
 header, Remote Management Utility C-209
 sizes, Host Communications Facility C-83
 reformat diskettes U-68
 register, index L-6
 register, software L-6
 register conventions \$S1ASM I-257
 BSCAM processing I-147
 common emulator setup routine I-68
 EBCDIC to floating-point conversion I-205
 summary chart \$S1ASM I-258
 terminal I/O support I-106
 REL release a status record, \$HCFUT1 command C-110
 relational statements L-180
 RELEASE
 Host Communications Facility, TP operand C-96
 Indexed Access Method CALL S-147
 coding description L-356
 overview L-27, S-147
 return codes L-357
 release a status record, REL \$HCFUT1 command C-110
 release space (\$PDS) S-261
 relocating program loader I-19
 relocation dictionary, \$EDXASM I-250
 REMARK operator comment, \$JOBUTIL command U-288
 remote job entry to host, \$RJE2780/\$RJE3780 C-73
 Remote Management Utility
 CDRRM equates C-292
 control block (RCB) description I-164, I-169
 equate tables C-292, I-295
 use in problem determination I-190
 defaults C-283
 error handling C-277
 function table I-164, I-167
 functions C-206, I-166
 installation C-281
 interface C-207
 internals I-216
 logic flow I-170
 messages C-279
 modifying defaults C-283
 module descriptions I-191
 module list I-186
 operation C-213
 overlay function processor

table I-167, I-220
 overlay table I-167, I-220
 overview C-205
 program preparation I-184
 requirements C-207
 sample host programs C-259
 system generation
 considerations C-281
 TERMINAL statement example
 S-107
 terminating C-251
 remote system (see Remote
 Management Utility) C-205
 remove breakpoints and trace
 ranges, OFF \$DEBUG command U-97
 rename member
 RE \$DISKUT1 command U-135,
 U-136
 RE \$DIUTIL command U-161
 RENUM renumber lines
 \$EDIT1/N subcommand U-196
 \$FSEDIT primary command U-224
 reorganize an indexed data set
 U-242
 procedure S-166
 report data member (\$PDS) S-251
 reposition line pointer (see move
 line pointer)
 Request record C-209
 reserved labels L-4
 reset
 function, \$RJE2780/\$RJE3780
 attention request C-76
 IDCB command L-176
 Indexed Access Method
 ECHO mode, EC \$IAMUT1 com-
 mand U-240
 SE command parameters, RE
 \$IAMUT1 command U-243
 line command, \$FSEDIT primary
 command U-225
 RESET task control instruction
 coding description L-258
 overview L-42, S-31
 resident assembler routines I-256
 resolution, enhanced I-201
 resolution, standard graphics
 I-201
 resource control, supervisor I-54
 restart, automatic S-129
 restore
 disk or disk volume from tape,
 RT \$TAPEUT1 command U-326
 dump volume utility, \$MOVEVOL
 U-294
 4974 to standard character
 set, RE \$TERMUT2 command
 U-343
 resulting field (EOR) L-122
 return codes (see also completion
 codes)
 \$DISKUT3 S-319, U-444
 \$PDS U-445
 BSC C-57, U-446
 CONVTB L-80
 CONVTD L-83
 data formatting instructions
 U-447
 DELETE L-330
 DISCONN L-333
 ENDSEQ L-335
 EXIO U-448
 EXIO instruction L-131
 EXIO interrupt L-132
 EXTRACT L-337
 FADD L-136
 FDIVD L-138
 FILEIO C-145
 floating point instruction
 U-450
 FMULT L-145
 formatted screen image U-450
 FSUB L-160
 FTAB C-138, L-373
 GET L-340
 GETSEQ L-343
 in Remote Management Utility
 control block I-190
 Indexed Access Method U-451
 LOAD L-199, U-452
 LOAD (Indexed Access Method)
 L-346
 Multiple Terminal Manager
 U-453
 PRINTTEXT L-219
 PROCESS L-349
 PUT L-351
 PUTDE L-353
 PUTEDIT L-238
 PUTUP L-355
 READ disk/diskette L-249,
 U-455
 READ tape L-250, U-456
 READTEXT L-255
 RELEASE L-357
 SBIO U-457
 SBIO instruction L-262
 SETPAN C-135
 tape L-77
 TERMCTRL L-288
 terminal I/O L-255, U-458
 ACCA U-459
 interprocessor
 communications C-31,
 U-460
 virtual terminal L-256,
 U-461
 TP (Host Communications Facil-
 ity) C-102, U-463
 WHEREAS L-316
 WRITE disk/diskette L-320,
 U-455
 WRITE tape L-320, U-456
 return from immediate action
 routine (SUPEXIT) I-49
 return from task level (SUPRTURN)
 I-49
 RETURN program control
 instruction
 coding description L-259
 overview L-32, S-31
 supervisor entry point I-279,
 I-313
 supervisor interface I-62
 REW (rewind tape) L-75
 rewind tape, MT \$TAPEUT1 command
 U-324
 RH reassign hardcopy, \$TERMUT1
 command U-338
 RI read
 transparent/non-transparent,
 \$BSCUT2 command C-68
 RJE (see Remote Job Entry)
 RLOADER I-19, I-22
 RLOADER/RLOADRU module
 description I-78
 RO reorganize indexed file,
 \$IAMUT1 command U-242
 ROFF (rewind offline) L-75

roll screen, terminal I/O L-48,
S-293
RP read program
 \$UPDATE command U-410
 \$UPDATEH command U-419
RPQ D02038, 4978 display station
 attachment C-6, S-97
 different device
 configurations C-8
RSTATUS IDCB command L-175
RT
 activate realtime data member,
 \$DICOMP subcommand U-124
 change realtime data member
 name (\$PDS) S-258
 disk or disk volume from tape,
 \$TAPEUT1 utility U-326
RWI read/write non-transparent,
 \$BSCUT2 command C-58
RWIV read/write non-transparent
 conversational, \$BSCUT2 C-71
RWIVX read/write transparent
 conversational, \$BSCUT2 C-70
RWIX read/write transparent,
 \$BSCUT2 command C-67
RWIXMP read/write multidrop
 transparent, \$BSCUT2 command
 C-60

S

SA save data, \$DICOMP subcommand
U-124
SAVE
 data set on disk, \$IMAGE com-
 mand U-254
 work data set, \$EDIT1/N
 subcommand U-197
save current task status
(TASKSAVE) I-54
save data, SA \$DICOMP subcommand
U-124
save disk or disk volume on tape,
 \$TAPEUT1 utility U-330
save storage and registers, \$STRAP
 utility U-348
SB special PI bit, \$IOTEST
 command U-267
SBAI sensor based I/O support
 module description I-80
SBA0 sensor based I/O support
 module description I-80
SBCOM sensor based I/O support
 module description I-80
SBDIDO sensor based I/O support
 module description I-80
SBIO sensor based I/O instruction
 coding description L-260
 control block (SBIOCB) I-127
 overview L-39, S-51
 return codes L-262
SBIOCB sensor based I/O control
 block I-127
SBPI sensor based I/O support
 module description I-80
SC save control store, \$TERMUT2
 command U-343
screen format builder utility,
 \$IMAGE S-68, U-250
SCREEN graphics instruction
 coding description L-270
 overview L-26

screen image format building
 U-250
screen images, retrieving and dis-
 playing S-300
screen management, terminal I/O
 L-48
SCRNS volume, Multiple Terminal
 Manager C-120, C-173
SCRNSREP, Multiple Terminal
 Manager C-125
scrolling, \$FSEDIT U-210
SCSS IDCB command L-176
SE set parameters, \$IAMUT1
 command U-244
SE set status, \$HCFUT1 command
 C-110
second-level index block
 description S-197
 overview S-153
secondary
 disk volumes S-132
 volumes S-60
secondary option menus S-218,
 U-36
 (see session manager)
sectioning of program modules
 L-33
sector S-52
self-defining terms L-4
send
 data, HX \$DICOMP subcommand
 U-118
 data set, SEND function C-247
 message to another terminal,
 \$TERMUT3 utility U-344
SEND function
 internals I-166, I-172
 overview C-247
 sample program C-274
sensor based I/O
 assignment L-188
 I/O control block (SBIOCB)
 I-127
 modules (IOLOADER/IOLOADRU)
 I-78
 statement overview L-39
 support module descriptions
 I-81
 symbolic L-9
SENSORIO configuration statement
 S-51, S-84
sequence chaining L-27
sequencing instructions, program
 L-34
sequential access
 in Indexed Access Method
 S-145
 overview S-53
sequential work file operations
 (\$SIASM) I-259
serially reusable resource (SRR)
 I-59, S-33
session, PASSTHRU
 conducting C-227
 establishing C-225
 logic flow diagram C-230
 using \$DEBUG utility C-272
session manager U-27
 \$SMALLOC data set allocation
 control data set S-222, U-30
 \$SMDELETE data set deletion
 control data set S-222, U-32
 adding an option S-209, S-224
 communications utilities U-42
 communications utilities

S-217
 data management S-215
 diagnostic utilities
 S-217
 disk utilities (see data
 management)
 execute program utilities
 S-216
 graphics utilities S-216
 job stream processor
 utilities S-216
 logon menu U-27
 primary S-218, U-35
 program preparation
 utilities S-214
 secondary S-218, U-36
 summary of S-213
 terminal utilities S-215
 updating primary option
 S-224
 creating a new menu S-224
 data management U-38
 data set deletion U-32
 data sets creation U-29
 diagnostic utilities U-43
 execute program utilities
 U-41
 graphics utilities U-40
 invoking U-27
 invoking a \$JOBUTIL procedure
 S-229
 job stream processor
 utilities U-42
 menus U-33
 minimum partition size
 required U-27
 operational overview S-209
 primary option menu, \$SMMPRIM
 S-218, U-35
 procedures
 communications utilities
 S-217
 data management utilities
 S-215
 diagnostic utilities
 S-217
 execute program utilities
 S-216
 graphics utilities S-216
 job stream processor
 utilities S-216
 overview S-220
 program preparation
 utilities S-214
 terminal utilities S-215
 updating S-225
 program function keys U-28
 program preparation utilities
 U-36
 secondary option menus S-218,
 U-36
 storage usage S-211
 terminal utilities U-40
 text editing utilities U-36
 utilities not supported U-46
 SET, ATTN TERMCTRL function L-288
 set breakpoints and trace ranges,
 AT \$DEBUG command U-90
 set date and time, \$T operator
 command S-63, U-19
 SET Host Communications Facility
 TP operand C-97
 SET, LPI TERMCTRL function L-288
 set status, SE \$HCFUT1 command
 C-110
 set tape offline, MT \$TAPEUT1 com-
 mand U-324
 set time, \$T operator command
 U-19
 SETBUSY supervisor busy routine
 I-48, I-63
 SETCUR, Multiple Terminal Manager
 CALL
 coding description C-137,
 L-378
 internals M-9
 overview C-117, L-29
 SETEOD subroutine S-324
 SETPAN, Multiple Terminal Manager
 CALL
 coding description C-134,
 L-379
 internals M-9
 overview C-117, L-29
 return codes L-380
 setup procedure for \$JOBUTIL
 U-271
 SG special PI group, \$IOTEST com-
 mand U-267
 SHIFTL data manipulation
 instruction
 coding description L-271
 overview L-19
 SHIFTR data manipulation
 instruction
 coding description L-273
 overview L-19
 SHUTDOWN function C-251, I-166,
 I-181
 SI save image store, \$TERMUT2 com-
 mand U-341
 SIGNON/SIGNOFF, Multiple Terminal
 Manager C-156
 SIGNONFL C-174
 single program execution I-35
 single-task program I-33
 single task program S-34
 SIXB (see second-level index
 block)
 SLE sublist element, \$EDXASM
 format of I-217
 in instruction parsing
 (\$EDXASM) I-220
 instruction description and
 format I-229
 used in \$IDEF I-241
 software register L-6
 software trace table S-265
 sort/merge S-9
 source program compiling S-71
 source program entry and editing
 S-66, U-351
 source program line continuation
 using \$EDXASM L-4, U-361
 source statements, \$EDXASM overlay
 generated I-243
 SP spool function,
 \$RJE2780/\$RJE3780 reset type
 C-76
 SPACE listing control statement
 coding description L-275
 overview L-28
 special control characters S-46
 special PI
 bit, SB \$IOTEST command U-267
 group, SG \$IOTEST command
 U-267
 specifications, data conversion
 L-146

SPECPI define special process
 interrupt L-189
 SPECPIRT instruction
 coding description L-276
 overview L-39
 split screen configuration S-293
 SPOOL define spool file,
 \$RJE2780/\$RJE3780 C-76
 SQ set prompt made, \$COPYUT1
 command U-64
 SQRT data manipulation
 instruction
 coding description L-277
 overview L-19
 SS set program storage parameter,
 \$DISKUT2 command U-149
 ST
 display data set status,
 \$DIUTIL command U-162
 save disk or disk volume on
 tape, \$TAPEUT1 command U-330
 standard labels, tape
 EOF1 S-240
 EOV1 S-239
 fields S-238
 HDR1 S-239
 header label S-235
 layouts S-236
 processing S-236
 trailer label S-235
 volume label S-235
 VOL1 S-238
 START
 IDCB command L-176
 PROGRAM statement operand
 L-225
 start and termination procedure,
 \$DEBUG U-85
 STARTPGM I-16
 statement label L-4
 static screen, terminal I/O
 accessing example S-297
 overview L-48
 status, set, SE \$HCFUT1 command
 C-110
 STATUS data definition statement
 coding description L-278
 overview L-17
 status data set, system Host
 Communications Facility C-85
 Status record C-258
 STIMER timing instruction
 coding description L-280
 overview L-50, S-32
 with PASSTHRU function C-238
 storage estimating
 application program size
 S-344
 supervisor size S-333
 utility program size S-342
 storage management
 address relocation translator
 I-71, S-42
 allocating I-25
 description S-42
 design feature S-13
 storage map, resident loader I-26
 storage map (\$SIASM) phase to
 phase I-262
 storage resident loader, RLOADER
 I-19
 storage usage during program load
 I-20
 store next record (\$PDS) S-261
 store record (\$PDS) S-261

strings, relational statement
 L-180
 SU
 submit (X) function,
 \$RJE2780/\$RJE3780 reset type
 C-77
 submit job to host, \$HCFUT1
 command C-111
 SUBMIT
 Host Communications Facility,
 TP operand C-98
 send data stream to host,
 \$RJE2780/\$RJE3780 C-77
 submit job to host, \$EDIT1
 command U-179
 submit job to host, \$FSEDIT
 option U-217
 SUBMITX send transparent,
 \$RJE2780/\$RJE3780 C-77
 SUBROUT program control statement
 coding description L-281
 overview L-32, S-31
 subroutines
 \$IMDATA S-303
 \$IMDEFN S-301
 \$IMOPEN S-300
 \$IMPROT S-302
 ALTIAM concatenation S-167
 DSOPEN S-322
 overview S-31
 SETEOD S-324
 SUBTRACT data manipulation
 instruction
 coding description L-283
 overview L-19
 precision table L-284
 suggested utility usage U-48
 supervisor/emulator
 class interrupt vector table
 I-10, I-277
 communications vector table
 I-11, I-278, I-313
 control block pointers I-11
 design features S-13
 device vector table I-11,
 I-278
 emulator command table I-13,
 I-282, I-301
 entry routines I-47
 equate table I-279, I-313
 exit routines I-49
 features S-13
 fixed storage area I-9
 functions I-44
 calling I-60
 generation I-5, S-115
 initialization control module,
 EDXINIT, description I-81
 initialization task module,
 EDXSTART, description I-81
 interface routines I-61
 introduction I-5
 module names and entry points
 S-309
 module summary I-8
 overview S-29
 PASSTHRU session with C-225
 referencing storage locations
 in I-12
 service routines I-53
 size, estimating S-333
 task supervisor work area
 I-13, I-280
 utility functions (see
 operator commands)

with the address translator support I-72

SUPEXIT supervisor exit routine I-49, I-63

support for optional features L-15

SUPRTURN supervisor exit routine I-49

surface analysis of tape, \$TAPEUT1 utility U-319

SVC supervisor entry routine I-47, I-62

SVCABEND supervisor exit routine I-49

SVCBUF supervisor request buffer I-48

SVCI supervisor entry routine I-48

symbol dictionary, \$EDXASM I-250

symbol table types, \$EDXASM I-216

symbolic L-10

- address (disk,tape) L-10
- disk/tape I/O assignments L-10
- diskette L-10
- reference to terminals S-110
- sensor I/O addresses L-9
- terminal I/O L-10

symbols (EXTRN) L-134

symbols (WXTRN) L-323

syntactical coding rules L-4

syntax checking in instruction parsing (\$EDXASM) I-221

syntax rules L-4

SYSGEN (see system generation)

system

- alternate logging device S-46, S-111
- class interrupt vector table I-10, I-277
- commands (see operator commands)
- common area I-12
- communications vector table I-11, I-278, I-313
- control blocks, referencing I-289
- data tables, EDXSYS, module description I-75
- device vector table I-11, I-278
- emulator command table I-13, I-282, I-301
- generation
 - procedure S-115
- host/remote C-205
- logging device S-46, S-110
- operational and error messages U-421
- printer S-46, S-110
- program check and error messages U-427
- task supervisor work area I-13, I-280

SYSTEM configuration statement L-39, S-86

system configuration statements S-75

system control blocks S-42

system reserved labels L-4

T

TA allocate tape data set, \$TAPEUT1 command U-333

tables, parameter equate L-11

tabs

- HTAB \$IMAGE command U-252
- TABSET \$EDIT1/N subcommand U-198
- VTAB \$IMAGE command U-254

TABSET establish tab values

\$EDIT1/N editor subcommand U-198

tape

- bypass label processing S-244
- control L-74
- data set L-40
- defining volumes S-62
- definitions for data sets L-40
- end-of-tape (EOT) L-41
- I/O instructions L-40
- internals I-97
- labels
 - external S-233
 - internal S-233
- load point (BOT) L-40
- non-label
 - layout S-242
 - processing S-243
 - support S-241
- record L-40
- return codes L-77, U-455
- standard label
 - fields S-238
 - layout S-236
 - processing S-236
 - support S-235
- storage capacity S-59
- symbolic addressing L-10
- utility, \$TAPEUT1 S-233, U-311
- volume L-40

TAPE configuration statement S-94

tape data set control block I-99

tape device data block (see TDB)

TAPEINIT, tape initialization module description I-82

tapemark L-74

task

- active/ready level table I-50
- concepts I-29
- control I-42
- control block (see TCB)
- definition and control functions
- dispatching I-52
- error exit facility
 - check and trap handling S-268
 - linkage conventions S-269
- execution states I-43, S-39
- internals I-42
- multiple-task program I-33, S-34
- overview L-42, S-29
- priority (see priority, task execution)
- single-task program I-33, S-34
- states S-39
- status display, WHERE \$DEBUG command U-102
- structure S-29

supervisor I-42
 supervisor address translator
 support module I-76
 supervisor functions I-44
 supervisor work area I-13,
 I-280
 switching I-51, S-30
 synchronization and control
 I-54, S-30
 task code words L-8
 TASK task control statement
 coding description L-285
 overview L-42, S-31
 TASKSAVE supervisor service
 routine I-54
 TCB task control block I-32,
 I-43, I-49, I-56, I-314
 TCBEQU L-13
 TD
 display line and data (\$PDS)
 S-258
 display time and date, \$DICOMP
 subcommand U-124
 test display, \$DICOMP command
 U-108
 TDB, tape device data block
 description I-97
 equate listing I-316
 TEB terminal environment block
 C-128, M-13
 Tektronix C-6
 devices supported S-14, S-45
 support for digital I/O S-312
 teleprocessing (see TP)
 teletypewriter adapter C-7, C-21
 TERMCTRL terminal I/O instruction
 coding description L-288
 overview L-44
 return codes L-301
 TERMERR L-44
 terminal
 #7850 teletypewriter adapter
 C-21
 ACCA support C-7, L-295
 ASCII C-7
 assignment list, LA \$TERMUT1
 command U-336
 attention handling L-47
 attention keys L-47
 code types C-303
 configuration utility,
 \$TERMUT1 U-334
 connected via digital I/O
 S-312
 control block (see CCB)
 data representation L-46
 definition and control
 functions S-47
 device configurations C-8
 EDXTIO/EDXTIOU module
 description I-78
 environment block (see TEB)
 error handling L-44
 forms control L-46
 forms interpretation for
 display screens L-46
 functions
 data formatting C-16
 definition C-16
 interrupt processing C-17
 hardware jumpers C-18
 I/O L-46
 attention handling L-47
 data representation L-45
 error handling L-44
 forms control L-45
 prompting and advance
 input L-46
 screen management L-48
 I/O internal design I-105
 I/O support layer 3 I-112
 input L-46
 keyboard and ATTNLIST tasks
 L-47
 message sending utility,
 \$TERMUT3 U-344
 new I/O terminal support
 I-117
 operations C-14
 operator signals L-49
 output L-46
 output line buffering L-46
 program function keys L-47
 prompting and advance input
 L-46
 return codes C-20, L-219,
 L-255, U-458
 roll screens L-48
 sample terminal support
 program C-26
 screen management L-48
 server, Multiple Terminal
 Manager C-119, M-7
 session manager (see session
 manager)
 special considerations for
 attachments of devices
 via #1610 or #2091 with
 #2092 adapters C-17
 via #2095 with #2096
 adapters C-21
 special control characters
 S-46
 static screens L-48
 supported devices and
 features C-6
 terminal I/O L-47
 terminology for supported
 terminals C-7
 transmission protocol C-31
 utilities (session manager)
 S-215, U-40
 virtual I/O I-115
 TERMINAL configuration statement
 defaults S-105
 definition S-96
 overview S-48
 TERMINAL volume, Multiple Terminal
 Manager C-120, C-171
 terminate
 logging, \$LOG utility U-292
 Remote Management Utility
 C-251
 test
 BSC lines, \$BSCUT2 utility
 C-64
 generated report or graphics
 profile member U-108
 label types, \$TAPEUT1 utility
 U-319
 process interrupt for
 occurrence of event, \$IOTEST
 U-267
 TEXT data definition statement
 coding description L-305
 overview L-17
 text editing utilities
 edit dataset subroutine exam-
 ples I-326
 full screen-editor \$FSEEDIT

U-209
 line editors, \$EDIT1/N U-169
 overview S-66
 work data set, format of
 I-321
 text wrapping, WRAP function
 C-254
 time/date
 display, \$W operator command
 U-25
 set, \$T operator command U-19
 set, automatic initialization
 facility S-130
 time of day
 GETTIME instruction L-167
 TIMEDATE Host Communications
 Facility, TP operand C-100
 TIMER configuration statement
 S-33, S-112
 timer control L-50
 timer module descriptions
 (EDXTIMER, EDXTIMR2) I-80
 timing instructions L-50, S-32
 TITLE listing control statement
 coding description L-308
 overview L-28
 TONE TERMCTRL function L-288
 TOP repostiton line pointer,
 \$EDIT1/N editor subcommand U-200
 TP host communication instruction
 description
 coding description C-90
 internals I-153
 subcommand operations I-157
 TPCOM host communications support
 module description I-81
 trace printing routine for BSC,
 \$BSCUT1 C-62, S-65
 trace ranges and breakpoints
 setting, AT \$DEBUG command U-90
 trace routine for BSC, \$BSCTRCE
 C-61
 trace table, software S-265
 transaction program, Multiple
 Terminal Manager
 functions L-28
 Multiple Terminal Manager
 C-121
 transfer data set to host
 SEND function C-247
 WR \$HCFUT1 command C-112
 WRITE \$EDIT1 command U-180
 WRITE \$FSEDIT option U-216
 transfer rates for data, Host
 Communications Facility C-84
 transient program loader I-19
 transmission codes S-98
 transmission protocol, host
 communications I-156
 transmitted data, length of, host
 communications I-159
 TRAPDUMP force trap dump, \$TRAP
 attention command U-349
 TRAPEND end \$TRAP use, \$TRAP
 attention command U-349
 TRAPOFF deactivate error trap,
 \$TRAP attention command U-349
 TRAPON activate error trap, \$TRAP
 attention command U-349

U

UN unload indexed file, \$IAMUT1
 command U-246
 UNBLINK TERMCTRL function L-288
 undefined length records, tape
 S-245
 UNLOCK TERMCTRL function L-288
 unprotected field S-307, U-253
 UP move line pointer, \$EDIT1/N
 editor subcommand U-201
 update utility
 \$UPDATE convert object program
 to disk U-408
 \$UPDATEH convert host object
 program to disk U-418
 updating a menu for the session
 manager S-224
 user defined data member (\$PDS)
 S-252
 user exit routine L-310
 requires Macro Assembler S-71
 user initialization modules I-17
 USER program control instruction
 coding description L-310
 overview L-32
 utilities U-47
 BSC communications C-61
 invoking U-2
 listed by type S-64, U-3
 overview S-5
 utilities not supported by session
 manager menu U-46
 utility program size S-342
 utility usage U-48

V

V verify, \$INITDSK command U-260
 VA
 display, variable, \$DICOMP
 subcommand U-125
 display variable (\$PDS) S-254
 variable length record, Host
 Communications Facility C-84
 variable length records, tape
 S-244
 variable names L-4
 vary disk, diskette, or tape
 offline, \$VARYOFF U-20
 vary disk, diskette, or tape
 online, \$VARYON U-22
 vector
 addition L-19, L-54
 data manipulation L-19
 vector addition (ADDV)
 coding description L-54
 overview L-19
 verify
 disk or diskette data set, V
 \$INITDSK U-260
 tape executing correctly, EX
 \$TAPEUT1 command U-319
 tape surface free of defects,
 EX \$TAPEUT1 command U-319
 verify and initialize disk or
 diskette library, \$INITDSK U-256
 verify identification
 host system C-223
 remote system C-223

VERIFY verify changes, \$EDIT1/N
 editor subcommand U-202
 vertical tabs, defining U-254
 VI list volume information,
 \$IOTEST command U-270
 virtual terminal communications
 accessing the virtual terminal S-281
 creating a virtual channel S-280
 establishing the connection S-280
 inter-program dialogue S-282
 internals I-115
 loading from a virtual terminal S-281
 Remote Management Utility requirements C-281
 volume
 definitions (disk/diskette) L-22, S-52
 dump restore utility, \$MOVEVOL U-294
 labels S-60
 VTAB define vertical tab setting, \$IMAGE command U-254

W

WAIT program sequencing statement
 coding description L-313
 overview L-42, S-31
 supervisor function I-45, I-58
 wait state, put program in, WS \$IOTEST command U-264
 waiting, task execution state I-43
 WE copy to basic exchange diskette data set, \$COPY command U-63
 WHERE display status of all tasks, \$DEBUG command U-102
 WHERE task control function
 coding description L-315
 overview L-42, S-287
 return codes L-316
 WI write non-transparent, \$BSCUT2 command C-69
 WIX write transparent, \$BSCUT2 command C-69
 word boundary requirement
 DO L-34
 IF L-34
 PROGRAM L-225
 work data set
 \$EDXASM I-249
 \$LINK U-400
 \$SIASM I-258
 work files, \$SIASM, how used I-258
 WR write a data set to host, \$HCFUT1 command C-112
 WRAP function C-254, I-166, I-176
 WRITE
 disk/diskette I/O instruction
 coding description L-317
 overview L-22
 return codes L-320, U-455
 Host Communications Facility,
 TP operand C-101
 IDCB command L-175
 Multiple Terminal Manager

CALL
 coding description C-133, L-381
 internals M-9
 overview C-118, L-29
 save work data set
 \$EDIT1 command U-180
 \$EDIT1N command U-181
 \$FSEDIT primary option U-216
 tape I/O instruction
 coding description L-317
 overview L-22
 return codes L-320, U-456
 write data set to host, WR \$HCFUT1 command C-112
 write operations, HCF I-156
 WRITE1 IDCB command L-175
 WS put program in wait state, \$IOTEST command U-264
 WTM (write tape mark) L-75
 WXTRN program module sectioning statement
 coding description L-323
 overview L-33

XYZ

X-type format L-154
 XI external sync DI, \$IOTEST command U-266
 XO external sync DO, \$IOTEST command U-266
 XYPLOT graphics instruction
 coding description L-324
 overview L-26
 YTPLOT graphics instruction
 coding description L-325
 overview L-26
 ZCOR, sensor I/O L-189

Numeric Subjects

1560 integrated digital input/output non-isolated feature C-6
 different device configurations C-8
 use with different terminals C-7
 1610 asynchronous communications single line controller C-6
 considerations for attachment of devices C-17
 different device configurations C-8
 for interprocessor communications C-29
 to a single line controller S-99
 use with different terminals C-7
 2091 asynchronous communications eight line controller C-6, S-99
 considerations for attachment of devices C-17
 different device configurations C-8
 use with different terminals

- C-7
- 2092 asynchronous communications
 - four line adapter C-6
 - considerations for attachment of devices C-17
 - different device configurations C-8
 - to attach ACCA terminal S-99
 - use with different terminals C-7
- 2095 feature programmable eight line controller C-6
 - considerations for attachment of devices C-21
 - different device configurations C-8
 - use with different terminals C-7
- 2096 feature programmable four line adapter C-6
 - considerations for attachment of devices C-21
 - different device configurations C-8
 - use with different terminals C-7
- 2741 Communications Terminal supported S-45
 - TERMINAL statement example S-106
- 3101 Display Terminal
 - attribute character C-122
 - block mode considerations C-25
 - character mode considerations C-22
 - interface with Multiple Terminal Manager C-121, L-29
 - TERMINAL configuration statement examples S-108
- 3585 4979 display station attachment C-6, S-97
- 4952 Processor
 - partitions on S-42
 - timer feature installed on S-32
- 4953 Processor
 - partitions on S-42
 - timer feature installed on S-32
- 4955 Processor
 - partitions on S-42
 - timer feature installed on S-32
- 4962 Disk Storage Unit
 - storage capacity S-58
 - supported by Indexed Access Method S-146
- 4963 Disk Subsystem
 - storage capacity S-58
 - supported by Indexed Access Method S-146
- 4964 Diskette Storage Unit
 - part of minimum system configuration S-22
 - required for program preparation S-22
 - supported by Indexed Access Method S-146
- 4966 Diskette Magazine Unit
 - part of minimum system configuration S-22
 - required for program preparation S-22
- supported by Indexed Access Method S-146
- 4969 Magnetic Tape Subsystem S-233
- 4973 Line Printer
 - defined in TERMINAL configuration statement S-96
 - end of forms S-307
 - TERMINAL statement example S-105
- 4974 Matrix Printer
 - defined in TERMINAL configuration statement S-96
 - end of forms S-307
 - restore to standard character set, RE \$TERMUT2 U-339
 - TERMINAL statement example S-105
- 4978 Display Station
 - defined in TERMINAL configuration statement S-96
 - part of minimum system configuration S-22
 - reading modified data S-307
 - required for program preparation S-22
 - TERMINAL statement example S-105
- 4979 Display Station
 - defined in TERMINAL configuration statement S-96
 - part of minimum system configuration S-22
 - required for program preparation S-23
 - TERMINAL statement example S-105
- 4982 sensor I/O unit S-84
- 5230 Data Collection Interactive S-11
- 5620 4974 matrix printer attachment C-6
 - defined in TERMINAL statement S-97
 - different device configurations C-8
- 5630 4973 line printer attachment C-6
 - defined in TERMINAL statement S-97
- 5719-AM3 (see Indexed Access Method)
- 5719-ASA (see Macro Assembler)
- 5719-CB3 (see COBOL)
- 5719-CB4 (see COBOL)
- 5719-F02 (see FORTRAN IV)
- 5719-LM3 (see Mathematical/Functional Subroutine Library)
- 5719-LM5 (see Macro Library)
- 5719-MS1 (see Multiple Terminal Manager)
- 5719-SM2 (see Sort/Merge)
- 5719-UT3 (see Utilities)
- 5719-UT4 (see Utilities)
- 5719-XS1 (see Basic Supervisor and Emulator)
- 5719-XX2 (see Program Preparation Facility)
- 5740-LM2 (see Macro Library/Host)
- 5799-TDE (see Data Collection Interactive)
- 7850 teletypewriter adapter C-6, C-21



READER'S COMMENT FORM

SC34-0313-2

IBM Series/1 Event Driven Executive Utilities, Operator Commands, Program Preparation, Messages and Codes

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. 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.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or the IBM branch office serving your locality.

Corrections or clarifications needed:

Page	Comment
------	---------

Cut or Fold Along Line

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A.
(Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut Along Line

Fold and tape

Please Do Not Staple

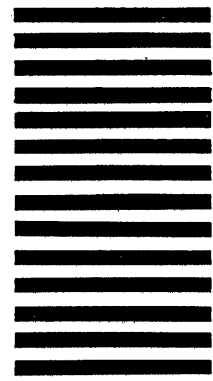
Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE



IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape

Please Do Not Staple

Fold and tape



International Business Machines Corporation
General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150, Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
(International)

SC34-0313-2
Printed in U.S.A.

READER'S COMMENT FORM

SC34-0313-2

IBM Series/1 Event Driven Executive Utilities, Operator Commands, Program Preparation, Messages and Codes

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. 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.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or the IBM branch office serving your locality.

Corrections or clarifications needed:

Page	Comment
------	---------

Cut or Fold Along Line

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A.
(Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut Along Line

Fold and tape

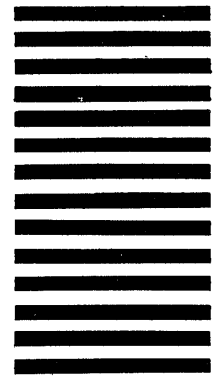
Please Do Not Staple

Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK
POSTAGE WILL BE PAID BY ADDRESSEE



IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape

Please Do Not Staple

Fold and tape



International Business Machines Corporation
General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150, Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
(International)

SC34-0313-2
Printed in U.S.A.



International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P. O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
(International)