IBM / Technical Newsletter

This Newsletter No.	SN34-0762
Date	December 15, 1981
Base Publication No.	SC34-0313-2
File No.	S1-32
Previous Newsletters	SN34-0684

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

This Technical Newsletter provides replacement pages for the subject publication. Pages to be inserted and/or removed are:

1, 2	63 through 68	295 through 298
7 through 12	83 through 86	313, 314
15, 16	197, 198	319, 320
16.1, 16.2 (added)	216.1, 216.2 (added)	329, 330
17 through 22	217, 218	359, 360
47, 48	255 through 258	411 through 414
62.1 through 62.4 (added)	269, 270	435 through 466.30 (added)

A technical change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

Technical corrections and additions, plus editorial corrections, were made in this TNL.

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

IBM Corporation, Information Development, Department 27T, P.O. Box 1328, Boca Raton, Florida 33432

Printed in U.S.A.

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 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

The 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
$U	Invoke a user-written routine
$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

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 or 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 or when I/O is being directed to the screen. Also, while the hardcopy function is in progress, keys (such as the attention key, PF keys, or ENTER) should not be pressed. Simultaneous operation of I/O and the hardcopy function can cause unpredictable results.

In order to avoid erratic forms control during the hardcopy function, it is recommended that the top of forms be set on the hardcopy device after the IPL sequence. Also, if the hardcopy device is reconfigured using $TERMUT1, the number of history lines should be set to 0.

CHAPTER 2. OPERATOR COMMANDS

The 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
$U	Invoke a user-written routine
$VARYOFF	Set a device offline
$VARYON	Set a device online
$W	Display the date and time

Operator Commands

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.

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 *
```

Example- Program has $GETMAIN for an overlay area

```
>  $A
PROGRAMS AT 09:30:50
IN PARTITION #5
$EDXASM 1000 *
5D00 *
```

$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.
```

$D - DUMP STORAGE

Dumps the contents of storage in hexadecimal on the terminal.

Syntax:

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

Operand	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
```

| **Note:** No verification checking on input values.

$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
```

Operand	Description
n	The number of pages to eject.

Example - Eject page on printer

```
> $E 2

   Note: Printer ejects two pages.
```

$L - LOAD PROGRAM

The $L command 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; storage
            defaults to the amount specified on
            the PROGRAM statement of the program
            to be loaded.
```

Operand	Description
program	The name of the program being loaded.
volume	The name of the volume where the program being loaded is stored.
storage	The total additional storage (in bytes) to be added to the end of the loaded program (overrides the STORAGE= parameter specified in the PROGRAM statement)
data set(s)	Data set(s) to be passed to the program being loaded (if specified in PROGRAM statement); specify the data set(s) in the order the program expects.

Example - Load a Program and pass a single data set

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

This example shows the command and parameters entered in single line format. Enter as much of the required information as possible on the same input line as the $L command in order to minimize the time the loader is busy.

Operator Commands

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.

$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
```

Operand	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
```

Note: No verification checking of input values.

$T - SET DATE AND TIME

Enters a new date and time into the system and resets the real-time 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
```

Operand	Description
date	The current date.
time	The current time.

Example - Set date and time

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

$U - INVOKE YOUR OWN OPERATOR COMMAND

The $U command allows you to write your own operator commands. To do this you must link-edit a module with an entry '$USRCMD' into the supervisor before EDXINIT.

Example - Writing an Operator Command

```
         PROGRAM  MAIN=NO
         ENTRY    $USRCMD
$USRCMD  PRINTEXT '$U ENTERED'
         ENDATTN
         END
```

Assemble your program and link it to the supervisor. Next, IPL the system, press the ATTN key, and enter $U. The message $U ENTERED will appear on your terminal.

$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
```

Operand	Description
ioda	The hexadecimal device address of the device to be varied offline.
slot	The slot number of the diskette to be varied offline; This parameter applies to the 4966 only. The valid slot numbers are:

0	All diskettes (1,2,3,A,B)
1	Slot 1
2	Slot 2
3	Slot3
A	Magazine 1
B	Magazine 2

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
```

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.

$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.

The first $VARYON after powering up the 4966 may have to be done twice, since it may get an I/O error. Also, the 4966 door must be shut to $VARYON the device.

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 encounted, 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.

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 can be invoked via virtual terminal support. Refer to Chapter 14 of the System Guide for details of loading a utility from a virtual terminal.

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.

CV - Copy Volume

The CV command allows you to copy entire volumes, providing a volume backup capability. A disk volume may be copied to another disk volume, a diskette volume to another diskette volume, or a diskette volume to a preallocated disk data set of appropriate size in records, as follows:

Type	Number of Records at 128 Bytes/Sector
Diskette 1	949
Diskette 2	1924

Volume copy operations do not add the members in a source volume to the target volume; the original contents of the target volume are replaced, including the directory.

If you have two or more 4964 Diskette units, or a 4964 and a 4066 Diskette Magazine unit, diskette volume copies between diskette devices are possible. If you have a single diskette drive and disk, diskette volume copies can be performedusing the following procedure:

1. Allocate a target data set on disk of appropriate size.

2. Using the CV command, copy the diskette volume to the disk data set.

3. Mount the target diskette on the diskette device and vary online.

4. Using the CV command, copy the contents of the disk data set to the target diskette.

If you have a single 4966 Diskette Magazine unit and a disk, the above procedure is also recommended.

CV EXAMPLE

Copy A Diskette To A Backup Data Set On 4962 Disk:

```
COMMAND (?): CV

COPY VOLUME
ENTER SOURCE VOLUME: IBMEDX
ENTER TARGET VOLUME: EDX002
ENTER TARGET DATA SET NAME - DATA1
ARE ALL PARAMETERS CORRECT? Y
COPY COMPLETE
    949 RECORDS COPIED

COMMAND (?):
```

The CV command copies the entire diskette volume. Therefore, the target data set should be of equal or greater size than the diskette size in records. If the target data set is not large enough, you may choose to do a partial copy or allocate (using $DISKUT1) a target data set large enough to accomodate the source.

If the target data set is not large enough, you are prompted as follows:

```
SOURCE DATA SET HAS nn RECORDS
TARGET VOLUME HAS ONLY nn RECORDS
DO YOU WISH TO CONTINUE?  (Y/N):
```

If you respond Y, the source is copied to the target data set until the target is full. If you respond N, the CV command ends and you are promptedfor another command, COMMAND(?):.

> **Note:** Once you have copied a volume to a target disk volume, the original contents of the target volume are replaced, including the directory. As a result, the original contents of the target disk are no longer accessible.

> **Note:** See the System Guide for an explanation of disk and diskette organization.

RE - Copy from Basic Exchange

The RE command copies a basic exchange data set from a diskette to a disk data set. A basic exchange data set is contained on a diskette that was formatted for the Standard for Information Interchange. Only one-sided, 128-byte diskettes can be used because EDX recognizes only one volume on a diskette. The target disk data set must be alloc- ated using $DISKUT1 before using the RE command.

You are prompted for the source diskette data set name and volume, the target disk data set name and volume, and the basic exchange data set name.

RE EXAMPLE

Copy Entire Basic Exchange Diskette Data Set To Disk:

```
COMMAND (?): RE

SOURCE ($$EDXVOL,VOLUME): $$EDXVOL,IBMEDX
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 (?):
```

COPY BASIC EXCHANGE DATA SET TO DISK: In this example, the record number where the copy is to start on target disk is specified.

```
COMMAND (?):  RE

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

SPECIFY START/END? Y/N:  Y
FIRST RECORD:  10

ENTER BASIC EXCHANGE DATA SET NAME:  DATA
NUMBER OF RECORDS COPIED:  151
COPY COMPLETED

COMMAND (?):
```

WE - Copy to Basic Exchange

The WE command copies a disk data set to a basic exchange data set on diskette. The diskette data set must be allocated before using WE. Use $DASDI to format the diskette for Standard for Information Interchange. Under this format, $DASDI formats a volume called IBMEDX, initializes the basic exchange header on the diskette, and automatically allocates a data set named DATA. DATA consists of all the data tracks on the diskette.

You are prompted for the source disk data set name and volume, the starting or ending records, the target diskette data set name and volume, and the basic exchange data set name.

WE EXAMPLE

Copy A Disk Data Set To A Basic Exchange Diskette:

```
COMMAND (?): WE

SOURCE (NAME,VOLUME): DATAFIL1,IBMEDX

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

ENTER BASIC EXCHANGE DATA SET NAME: DATA
COPY COMPLETE

COMMAND (?):
```

Copy a Disk Data Set to a Basic Exchange Diskette: In this example, the beginning and ending records numbers on disk to be copied to the target diskette are specified.

```
COMMAND (?): WE

SOURCE (NAME,VOLUME): DATAFIL1,EDX002

SPECIFY START/END? (Y/N): Y
FIRST RECORD: 100
LAST RECORD: 150
TARGET ($$EDXVOL,VOLUME): $$EDXVOL,IBMEDX

ENTER BASIC EXCHANGE DATA SET NAME: DATA
COPY COMPLETE

COMMAND (?):
```

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.

$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 ($EDX-NUC). 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.

$COPYUT1

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
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 (?):
```

$DASDI - FORMAT DISK OR DISKETTE

$DASDI 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

$DASDI 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 $DASDI 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

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. To debug a program that uses a 4978 or 4979 terminal in static screen mode, load the program (through $DEBUG) from another, different terminal that the program will use in static screen mode.
2. $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.
3. Multiple breakpoints and trace ranges can be established.
4. Several users can each use separate copies of $DEBUG concurrently, if sufficient storage is available.
5. Series/1 assembler language as well as Event Driven Language instructions can be traced and tested.
6. Both supervisor and application programs can be debugged.
7. Task names are automatically obtained from the program to be tested.
8. Task registers #1 and #2 can be displayed and modified.
9. Hardware registers R0 through R7 and the IAR can be displayed and modified.
10. Task registers #1 and #2 can be displayed and modified.
11. Five different data formats are accepted by the list and patch functions.
12. No special preprocessing of a program is required to permit it to be debugged.
13. 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.

14. No processing overhead is incurred unless the hardware trace feature is enabled. Even then, the hardware trace feature is only enabled for specific tasks.

15. The debug facility can be activated for a program that is experiencing problems but was previously loaded without the debug facility.

16. A program can be debugged by loading $DEBUG from a terminal other than the one from which the program to be tested was loaded.

17. Breakpoints or trace ranges specified during a debug session can be listed.

18. $DEBUG can control the execution of programs containing up to 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 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
PRINT	- Direct output to another terminal
QUALIFY	- Modify base address
WHERE	- Display status of all tasks

How to Enter a Command

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.

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
```

Operand	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
```

Operand	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	Resets relative tab column positions to initial system defaults. Does not reset tab character.

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. You are also prompted for the line number of the current work data set after which the data is to be added.

```
MERGE DATA FROM (NAME,VOLUME):           LINES- 1ST LAST
                                ADD TO TARGET AFTER LINE #:
```

> **Note:** Always enter the leading zeros when specifying line numbers; for example, specify line 000010 rather than line 10.

If the entire data set is to be merged, an '*' can be entered instead of the line number specifications.

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 another section.) 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.

Notes:

1. 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.

2. When READ or MERGE functions are processed, line numbers are resequenced. This may cause sequence numbers to differ if they are not created by $FSEDIT.

This page intentionally left blank.

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.

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.

To save all non-protected (null) characters from all fields when additional non-protected fields have been created, depress the ENTER key to exit PF1 mode. Then depress the PF2 key and the ENTER key again. To display all the non-protected fields with the null characters, you may now reenter PF2 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 - 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 936-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 can initialize only on slot number 1.

$INITDSK

Disk Initialization

Each disk volume (primary and secondary) must be initialized by using $INITDSK.

> **Note:** 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.

```
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 supports after system IPL.

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
```

Note: Communication device cards not installed at IPL time will not show up as supervisor-supported.

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

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 members, of the volume to be dumped. To determine the size in records, use the following formula:

         members/8 = records

         Add 1 if you have a remainder.

      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 directory of the volume being dumped.

   c. Use $COPYUT1 to:

      1) Copy other data sets onto the control data set. For example, you may required $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 $$EDX-LIB,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.

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
```

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).

CD EXAMPLES

Copying Data from a Disk to a Tape (Standard Label 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 (?):
```

Copying Data from a Disk to a Tape (Non-Labeled Tape)

```
COMMAND (?): CD

SOURCE (NAME,VOLUME): X,TAPE 01
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 (?):
```

Note: TAPE 01 is the ID assigned to the tape drive at system generation.

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.

If an error is encountered the return code and the contents of the buffer are printed. The buffer contains all FFFF's except for the last word, which is the record count of the failing record.

Note: If tape exerciser is not successful, do the following:

- turn on error logging
- Run exerciser again
- Provide information to your CE

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.

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 (?):
```

Note: If a tape error is encountered during a device restore, the utility informs you of the error and tells you which disk track corresponds to the bad tape record. The utility then attempts to resume processing. Once the restore has ended, the disk contents must be verified. Some tape errors can cause an indeterminate read head position in relation to the actual tape records, thereby displacing data on the disk.

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 (?):
```

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. Enter an asterisk (*) 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, an 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 Programs" on page 370 for more information.

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.

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.

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 ON EDX002

COMMAND (?):
```

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 ON EDX002

COMMAND (?):
```

End $UPDATE

```
COMMAND (?):  EN

$UPDATE ENDED AT 11:39:34
```

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 Chapter 6, Messages and Codes.

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:

***** EVENT DRIVEN EXECUTIVE *****

Issued By: IPL Operation

Explanation: After $SYSLOG terminal initialization this message appears on the $SYSLOG terminal.

System Action: None.

Response: None.

I/O ERROR INITIALIZATION FIXED HEAD DEV, DISK RETURN CODE= xxx IN THE TWO RECORDS STARTING WITH RECORD xxx

Issued By: Disk Initialization

Explanation: An error was encountered during fixed head initialization starting with record xxx.

System Action: The initialization is terminated.

Response: Check the Disk return code to find the cause of the problem and take the appropriate action.

INITIALIZATION ERROR

Issued By: Multiple Terminal Manager

Explanation: Initialization was unsuccessful. This message is written to the terminal that loaded the Multiple Terminal Manager. Additional messages are printed on the Multiple Terminal Manager log device.

System Action: Multiple Terminal Manager terminates.

Response: Determine the cause of the error and take corrective action.

Messages

INVALID COMMAND

Issued By: $IAMUT1

Explanation: An invalid command was entered by the user.

System Action: Reprompts for command.

Response: Enter a question mark (?) to obtain a list of valid commands and try again.

INVALID PROGRAM NAME

Issued By: Multiple Terminal Manager

Explanation: The name of the program requested from the primary menu was not found in the Multiple Terminal Manager program table or invalid parameters were supplied on a DISCONNECT command.

System Action: The requested function is not performed.

Response: Correct the program name or parameters and retry the request.

INVALID SIGNON CHARACTER

Issued By: Multiple Terminal Manager

Explanation: The SIGNON specification for the TERMINAL file record listed immediately before this message is not "Y" or "N".

System Action: The terminal is not connected.

Response: Correct the TERMINAL record. Stop and restart the manager.

INVALID TERMINAL

Issued By: Multiple Terminal Manager

Explanation: The terminal name entered with a DISCONNECT command is not a Multiple Terminal Manager terminal.

System Action: The terminal is not disconnected.

Response: Retry specifying a valid terminal name.

KEY OF INPUT REC xxxxxx IS DUPLICATE OR OUT OF SEQUENCE. OMIT THE RECORD AND CONTINUE?

Issued By: $IAMUT1

Explanation: A duplicate key exists in the input sequential data set and could not be written to indexed data set.

System Action: If reply = Y, the next record will be read from input data set and processing continues. If reply = N, command terminates.

Response: Make sure the input data set contains the proper data. Check the KEYSIZE and KEYPOS parameters used to define the indexed data set against the input data set records. Make sure the data in the input data set is in the proper sequence. Redefine and reload data set if necessary.

LOAD ERROR RC=xxx

Issued By: Multiple Terminal Manager

Explanation: A load failure occurred.

System Action: The terminal is not available to the Multiple Terminal Manager.

Response: Check the LOAD return code to find the cause of the problem and take the appropriate action.

Messages

LOAD FOR SERVER xxxxxxxx FAILED, RC=xxx

Issued By: Multiple Terminal Manager

Explanation: A load failure occurred during initialization for the server for terminal xxxxxxxx.

System Action: The terminal is not available to the &m..

Response: Check the LOAD return code to find the cause of the problem and take the appropriate action.

MENUNAME INVALID

Issued By: Multiple Terminal Manager

Explanation: The primary menu name specified for the TERMINAL file record listed immediately before this message is invalid.

System Action: The terminal is not connected.

Response: Correct the TERMINAL record. Stop and restart the manager.

MULTIPLE TERMINAL MANAGER SYSTEM FAILURE

Issued By: Multiple Terminal Manager

Explanation: The Multiple Terminal Manager task error exit routine has been entered due to a machine or program error.

The PSW and LSB at the time of failure has been saved at a displacement of X'172' into the program storage. Register 1 in the LSB contains the address of the failing instruction in the case of a program check.

The following example shows a specification check which occurred at location X'053C'.

```
MULTIPLE TERMINAL MANAGER SYSTEM FAILURE > $A

PROGRAMS  AT  00:06:24  IN  PARTITION #2 $MTM     0000 *
CDMSVR33 6C00 > $D 0 172 30 X  0172: 8002 28E6 0110 10D0
0DDC 053C 0DAC 7361
 0182: 0540 815C 00B8 0DDA 0000 00FA 0004 0028
 0192: 0052 007C 00A6 0017 0E72 A0A2 0E72 FFFF
 01A2: 0102 8026 1616 40C9 D5C9 E3C9
 ANOTHER DISPLAY?
```

The PSW is 8002 at 0172 and R1 is 053C on same line.

System Action: The Multiple Terminal Manager program remains active waiting for an event which will not be posted.

Response: Use Event Driven Executive operator facilities to display storage.

MULTIPLE TERMINAL MANAGER TERMINAL FILE RECORDS

Issued By: Multiple Terminal Manager

Explanation: The TERMINAL file records processed by the Multiple Terminal Manager are listed after this message. Any messages pertaining to a specific TERMINAL file record will be displayed immediately after the file record.

System Action: TERMINAL file records processed are listed.

Response: Review the listing and take action as needed.

MTMSTORE DATA SET LIMITS EXCEEDED

Issued By: Multiple Terminal Manager

Explanation: The specified MTMSTORE file is too small. This can occur after adding a new program with a storage requirement greater than any previous program's requirement or after adding a new terminal or screen.

System Action: The manager terminates.

Response: Delete the MTMSTORE file and recreate it with more space.

NO BUFFER SUPPLIED. $IAMUT1 TERMINATING

Issued By: $IAMUT1

Explanation: The $STORAGE field in program header was set to 0 and no buffer exists for $IAMUT1.

System Action: The program terminates.

Response: Use the SS command of $DISKUT2 to set the $STORAGE field to the desired buffer size (must be > 0). For LO,UN,and RO commands, this buffer must be large enough to contain the entire input and/or output record (whichever is larger).

NO PROGRAM LOAD FACILITY

Issued By: Load Utility Program

Explanation: The load utility program ($LOADER) cannot be found on the IPL volume.

System Action: The Load utility is terminated.

Response: If $LOADER is not on the IPL volume, you must copy $LOADER from XS-3001 to the IPL volume and restart the Load utility.

NO TERMINALS ARE AVAILABLE

Issued By: Multiple Terminal Manager

Explanation: No valid terminal specification records were found in the TERMINAL file, or, no terminal servers can be loaded, or, all terminals are busy. Other messages generated indicate the problem area.

System Action: The manager terminates.

Response: Determine the cause of the problem and take corrective action.

NO TRAP CONDITIONS SPECIFIED. $TRAP TERMINATED

Issued By: $TRAP Utility

Explanation: No trap conditions were specified. Some are required.

System Action: The $TRAP utility is terminated.

Response: Specify the necessary trap conditions and restart the $TRAP utility.

NULL INVALID FOR PARAMETER

Issued By: $IAMUT1

Explanation: An attempt was made to specify a null response (&) to a parameter on which this is invalid.

System Action: Reprompts for parameter.

Response: Enter the proper response to the parameter prompt. See determining data set size and format section in the $IAMUT1 chapter of the utilities manual for a description of each parameter. A null response is only valid for RSUBCK, RSUIX, FPOOL AND DELTHR parms.

OPEN FOR LOAD RETURN CODE = xxx. RETRY ?:

Issued By: $IAMUT1

Explanation: $IAMUT1 attempted to open the specified IAM file in LOAD mode and a bad return code was received from the IAM request.

System Action: If retry = Y, reprompts for DSNAME, VOLUME and retries the IAM open request. If retry = N, command terminates.

Response: Check the Indexed Access Method return code to find the cause of the problem and take the appropriate action.

OPEN FOR PROCESS RETURN CODE = xxx. RETRY?:

Issued By: $IAMUT1

Explanation: $IAMUT1 attempted to open the specified IAM file in process mode and a bad return code was received from the IAM request.

System Action: If retry = Y, reprompts for DSNAME, VOLUME and retries the IAM open request. If retry = N, the command terminates.

Response: Check the Indexed Access Method return code to find the cause of the problem and take the appropriate action.

PARTITION NUMBER IS INVALID

Issued By: $DUMP Utility

Explanation: An invalid partition number was entered during a partial storage dump.

System Action: The $DUMP utility is terminated.

Response: Enter a valid partition number and restart the $DUMP utility.

PRIMARY MENU mmmmmmmm FAILED FOR TERMINAL tttttttt

Issued By: Multiple Terminal Manager

Explanation: A SETPAN function failed for the terminal tttttttt using the primary menu mmmmmmmm.

System Action: The primary menu is not displayed.

Response: Ensure that a valid menu name is specified in the TERMINAL file for the specified terminal.

PROGRAM AREA TOO SMALL TO HOLD PGM pppppppp

Issued By: Multiple Terminal Manager

Explanation: The manager's program area is too small to hold the named program.

System Action: The program is not used.

Response: Increase the program area size by reallocating CDMDUMMY or split the program into smaller link-edited programs.

PROGRAM CAPACITY EXCEEDED

Explanation:

The amount of working storage allocated to $VERIFY is insufficient to process the indexed data set specified.

System Action:

$VERIFY terminates.

Response:

Increase the amount of working storage available to $VERIFY. Refer to Modifying Working Storage Requirements for a description of how to calculate the amount of working storage required and how to modify the amount supplied.

PROGRAM FILE LARGER THAN PROGRAM MANAGER BUFFER

Issued By: Multiple Terminal Manager

Explanation: The program table built during initialization exceeds the size of the buffer used by the program manager.

System Action: Multiple Terminal Manager terminates.

Response: Increase the program manager buffer size in module CDMCOMMN.

PROGRAM LOAD ERROR

Issued By: Multiple Terminal Manager

Explanation: An Event Driven Executive LOAD error occurred for the requested program.

System Action: The program is not loaded.

Response: Determine the cause of the problem. Rebuild the program if the problem persists.

PROGRAM xxP,00.00.00,LP=zzzz

Issued By: Program Load

Explanation: Any program invoked using $L (Load a Program) results in this message being displayed, indicating that the program you requested has been loaded. Here, xxP indicates that the program is xx pages long (256 bytes equals one page). 00.00.00 is the time in hours, minutes and seconds. LP=xxxx indicates that the load point of the program is at location X'zzzz'. If the timer support is not included in the supervisor, the time is not printed.

System Action: None.

Response: None.

READ INPUT DATASET RETURN CODE = xxx. RECORD NUMBER = xxxxxx

Issued By: $IAMUT1

Explanation: An attempt to read the indexed input data set failed for a sequential data set.

System Action: Command terminates.

Response: LO command - check the READ/WRITE return codes to find the cause of the problem and take the appropriate action. UN, RO commands -check the Indexed Access Method return codes find the cause of the problem and take the appropriate action.

RECONNECT SYNTAX INVALID

Issued By: Multiple Terminal Manager

Explanation: The correct syntax was not used on the RECONNECT operator command.

System Action: The command is ignored.

Response: Retry the RECONNECT command with correct syntax.

RECONNECT TERMINAL DEFINITION ERROR

Issued By: Multiple Terminal Manager

Explanation: The RECONNECT operator interface facility has encountered a failure while attempting to reconnect a terminal to the Multiple Terminal Manager. Since initialization would have already performed all functions necessary to include the terminal in the terminal table, the TERMINAL file, SCRNS volume or source table in RECONNEC has probably been altered since the Multiple Terminal Manager was started.

System Action: Terminal is not connected.

Response: Determine the cause of the error (check TERMINAL file for correct data).

SCREEN TABLE LARGER THAN INPUT BUFFER

Issued By: Multiple Terminal Manager

Explanation: The screen table built during initialization exceeds the Input Buffer size.

System Action: Initialization is aborted.

Response: Increase the Input Buffer size in module CDMCOMMN.

SENSOR I/O DEVICE AT ADDRESS xxxx IS OFFLINE
BSCA NOT THE DEVICE AT ADDR: zzzz

Issued By: Sensor I/O Status Check

Explanation: The system checks the status of any defined sensor I/O or Binary Synchronous Communications Adapter devices and prints appropriate status messages.

System Action: None.

Response: None.

SET DATE AND TIME USING COMMAND $T

Issued By: System

Explanation: 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.

System Action: None.

Response: None.

SIGNON PROGRAM NOT AVAILABLE FOR TERMINAL tttttttt

Issued By: Multiple Terminal Manager

Explanation: The specified terminal is required to sign on and off but no program named SIGNON was found in the PRGRMS volume.

System Action: The terminal is not signed on.

Response: Place a program named SIGNON in the PRGRMS file or designate that no signon is needed for the specified terminal. Reconnect terminal.

TAPE xxxx IS NOT A TAPE * * *
TAPE xxxx MARKED UNUSABLE

Issued By: Tape Initialization

Explanation: 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.

System Action: The initialization is terminated.

Response: Correct the problem and restart the initialization.

TAPE xxxx OFFLINE FOR BLP yyyy BPI

Issued By: Tape Initialization

Explanation: If the address (yyyy) is valid and the tape is not mounted this message is printed.

System Action: The initialization is terminated.

Response: Mount the tape and restart the initialization.

Messages

TAPE xxxx TAPE01 ONLINE FOR BLP yyyy BPI

Issued By: Tape Initialization

Explanation: If the address (yyyy) is valid and the tape is mounted this message is printed.

System Action: None.

Response: None.

TERMINAL tttttttt BUSY

Issued By: Multiple Terminal Manager

Explanation: Terminal tttttttt specified in the TERMINAL file is connected to another program.

System Action: The terminal is not used.

Response: Try to RECONNECT at a later time.

TERMINAL tttttttt NOT DEFINED IN EVENT DRIVEN EXECUTIVE SYSTEM

Issued By: Multiple Terminal Manager

Explanation: The specified terminal was not included in the definition of terminals when the Event Driven Executive system was generated.

System Action: The terminal is not connected.

Response: Include a terminal definition for the specified terminal when the Event Driven Executive system is generated.

TERMINAL ttttttttt RECONNECTED

Issued By: Multiple Terminal Manager

Explanation: The named terminal has been reconnected to the Multiple Terminal Manager.

System Action: The terminal is reconnected to the Multiple Terminal Manager.

Response: Use the terminal as needed.

TERMINAL NAME INVALID

Issued By: Multiple Terminal Manager

Explanation: The terminal name specified for the TERMINAL file record listed immediately before this message is invalid.

System Action: The terminal is not connected.

Response: Correct the TERMINAL record. Stop and restart the manager.

TERMINAL TABLE OR STORAGE SIZE EXCEEDED

Issued By: Multiple Terminal Manager

Explanation: While building the terminal table and loading servers, the storage size or the the maximum number of terminals (10) allowed has been exceeded. The work space, defined in CDMINIT, is defined to allow a maximum of 50 terminals.

System Action: The extra terminals are not connected.

Response: Increase the terminal table size by changing module CDMCOMMN. If there is not enough room, make the partition larger, decrease the number of terminals, or make CDMDUMMY smaller.

Messages

VALUE OUT OF RANGE

Issued By: $IAMUT1

Explanation: An invalid value was entered for the parameter prompt.

System Action: Reprompt for parameter.

Response: Enter the proper response to the parameter prompt. See Determining Data Set Size and Format section under $IAMUT1 chapter of Utilities manual for a description of each parameter.

VERIFICATION COMPLETE, ____ ERROR(S) ENCOUNTERED

Explanation:

$VERIFY has completed normally.

System Action:

$VERIFY terminates.

Response:

Examine any reports printed as needed.

WARNING: NO FSE IN SOURCE DIRECTORY POINTS TO END OF VOLUME. EXTRA DISK SPACE IN TARGET WILL NOT BE ADDED TO SOURCE VOLUME DEFINITION.

Issued by:$TAPEUT1 (RT)

Explanation: You are restoring a volume where the disk target volume is larger than the tape source volume definition. The utility attempted to update the disk volume directory to reflect the additional available space. However, to do so one free space entry must point to the end of volume. The additional space will be added to that FSE. In this case, no FSE in the directory pointed to the end of volume.

System Action:The utility terminates.

Response: The disk volume is useable, except that the additional space cannot be accessed. To recover the additional soace, the original volume must be saved again. Before saving it, you must compress it. Then run this utility again.

WARNING: I/O ERROR ACCESSING DIRECTORY. RC= XXX EXTRA DISK SPACE IN TARGET WILL NOT BE ADDED TO SOURCE VOLUME DEFINITION

Issued by: $TAPEUT1 (RT)

Explanation: You are restoring a volume where the disk target volume is larger than the tape source definition. An I/O error was encountered while the utility was attempting to update the disk volume directory to reflect the added space available.

System Action:The utility terminates.

Response: The disk volume directory may be only partially updated and therefore unuseable. Delete the volume and re-allocate it in a different spot on the disk and run the utility again.

Messages

WRITE OUTPUT DATASET RETURN CODE = xxx. RECORD NUMBER = xxxxxx.

Issued By: $IAMUT1

Explanation: An attempt to write to a sequential (output) data set failed.

System Action: Command terminates.

Response: LO, RO commands - Check the Indexed Access Method return code to find the cause of the problem and take the appropriate action (NOTE1). UN command - Check the READ/WRITE return code to find the cause of the problem and take the appropriate action (NOTE2).

NOTE1: It may be necessary to redefine data set (SE,DF) and retry LO command.

NOTE2: It may be necessary to reallocate the data set and retry the UN command.

xxxxxxxx DISCONNECT

Issued By: Multiple Terminal Manager

Explanation: Terminal xxxxxxxx has been issued a successful DISCONNECT command.

System Action: The terminal is disconnected.

Response: Reconnect terminal as needed.

xxxxxxxx PROGRAM TYPE INVALID

Issued By: Multiple Terminal Manager

Explanation: Program xxxxxxxx in the PRGRMS volume is not a program type data set.

System Action: The program named is not used.

Response: Specify program type members only for use as programs.

xxxxxxxx SCREEN SIZE TOO LARGE

Issued By: Multiple Terminal Manager

Explanation: Screen xxxxxxxx in the SCRNS volume will not fit in the screen manager buffer.

System Action: The screen is not available during this Multiple Terminal Manager session.

Response: Increase the screen manager buffer size in CDMCOMMN.

xxxxxxxx SETPAN FAILED, RC=xxx

Issued By: Multiple Terminal Manager

Explanation: A SETPAN failed for the screen named xxxxxxxx.

System Action: Processing continues.

Response: Check the Multiple Terminal Manager return code to find the cause of the problem and take the appropriate action.

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 $SYSLOG terminal:

```
PGM CHK:  PLP     TCB     PSW     LSB
          6B00    0138    8002    IE6A    0000    88D0
```

Where:

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 under Procesor Status Word).

LSB Level status block, consisting of the following:

```
WORD 1       - instruction address register
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/1, 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:

```
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. Program check in EDL command interpreter may cause R1 to be invalid.

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
```

```
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)
```

PROCESSOR STATUS WORD

The processor status word (PSW) is used to record error or exception conditions in the system that bey prevent further processing. It also contains certain status flags related to error recovery. Error or exception conditions recorded in the PSW cause four of the possible seven class interrupts to occur. These are machine check, program check, soft exception trap, and power/thermal warning.

Messages

The Copy Processor Status and Reset (CPPSR) instruction can be used to examine the PSW. This instruction stores the contents of the PSW into a specified location in main storage.

The PSW is contained in a 16-bit register with the following bit representation:

	Processor Type 495x					
Bit	2	3	5	Condition	Class Interrupt	Note
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

Following is an explanation of the bit representations:

Bit 00 Specification Check: Set to one if (1) the storage address violates the boundary requirements of the specified data type, or (2) the effective address is odd when attempting to execute a floating-point instruction and the floating-point feature is not installed.

Bit 01 Invalid Storage Address: Set to one when an attempt is made to access a storage address outside the storage size of the system. This can occur on an instruction fetch, an operand fetch, or an operand store.

Bit 02 Privilege Violate: Set to one when a privileged instruction is attempted in the problem state (supervisor state bit in the level status register is not on).

Bit 03 Protect Check: In the problem state, this bit is set to one when (1) an instruction is fetched from a storage area not assigned to the current operation, (2) the instruction attempts to access a main storage operand in a storage area not assigned to the current operation, or (3) the instruction attempts to change a main storage operand in violation of the read-only control.

Bit 04 Invalid Function: Set to one by the following conditions:

1. Attempted execution of an illegal operation code or function combination. These are:

Op code	**Function**
00101	All (when register 7 is specified in the R1 or R2 field of the instruction)
00111	All
01000	0001, 0010, 0011, 0101, 0110, 0111
01011	0001, 1001 (when in supervisor state and the relocation translator feature is not installed)
01011	0101, 0111
01100	111
01110	11000, 11010, 11011, 11100, 11110, 11111
11011	All
10110	All
11101	1100, 1101, 1110, 1111

Note: The preceding illegal conditions cause a program check class interrupt to occur.

2. The processor attempts to execute an instruction associated with a feature that is not installed. These are:

Op code	**Function**
00100	All (floating-point feature not installed)
01011	0011, 1011 (if the floating-point feature is not installed and the processor is in supervisor state)

Note: The preceding condition causes a soft-exception-trap class interrupt to occur.

Bit 05 Floating-Point Exception: Set to one when an exception condition is detected by the option floating-point processor. The arithmetic indicators (carry, even, and overflow) define the specific condition.

Bit 06 Stack Exception: Set to one when an attempt has been made to pop an operand from an empty main storage stack or push an operand into a full main storage stack. A stack exception also occurs when the stack cannot contain the number of words to be stored by a Store Multiple (STM) instruction.

Bit 08 Storage Parity: Set to one when a parity error has been detected on data being read out of storage by the processor. This error may occur when accessing a storage location that has not been validated since power on.

Bit 10 CPU Control Check: A control check will occur if no levels are active but execution is continuing. This is a machine-wide error. (See I/O check note.)

Bit 11 I/O Check: Set to one when a hardware error has occurred on the I/O interface that may prevent further communication with any I/O device. PSW bit 12 (sequence indicator) is a zero if the error occurred during an Operate I/O instruction and is set to one if the error occurred during a non-DPC transfer. The sequence indicator bit is not an error in itself but reflects the last interface sequence at any time. An I/O check cannot be caused by a software error. (See note.)

Note: The machine check class interrupt initiated by a CPU control check or I/O check causes a reset. The I/O channel and all devices in the system are reset as if a Halt I/O (channel directed command) had been executed. The processor, sensor-based output points, and timer values are not reset.

Bit 12 Sequence Indicator: This bit reflects the last I/O interface sequence to occur. See I/O check described above.

Bit 13 Auto IPL: Set to one by hardware when an automatic IPL occurs.

Set to zero by:

- A power on reset when Auto IPL mode is not selected
- Pressing the Load key
- An IPL initiated by a host system

Refer to the appropriate hardware manual for a description of initial program load.

Bit 14 Translator Enabled: When the Storage Address Relocation Translator Feature is installed, this bit is set to one or zero as follows:

1. Set to one (enabled)
   - An Enable (EN) instruction is executed with bit 12 of the instruction word set to zero and bit 14 set to one
2. Set to zero (disabled)
   - A Disable (DS) instruction is executed with bit 14 of the instruction word set to one
   - An Enable (EN) instruction is executed with bit 12 of the instruction word set to one
   - A processor reset (power-on reset, check restart, IPL, or system reset key)

Bit 15 Power Warning and Thermal Warning: Set to one when these condition occur (refer to the appropriate hardware manual for a description of a Power/Thermal Warning class interrupt). The power/thermal class interrupt is controlled by the summary mask.

For a description of class interrupts, I/O interrupts and the basic instruction set (including indicator settings and possible exceptions conditions) for your specific processor, refer to the appropriate hardware manual.

Completion Codes

CODES

This section presents three types of codes issued by the Event Driven Executive:

Completion	Issued by utility programs upon completion to indicate if execution was successful or not
Return	Issued as the result of executing an Event Driven language instruction or subroutine to indicate success or failure of the operation
Post	Issued by the system to signal the occurrence of an event

The codes and their meanings are presented by type and alphabetically by functional grouping.

Utility Completion Codes

The completion codes and their meanings are presented in alphabetic order according to function as follows:

- $EDXASM
- $IAMUT1
- $JOBUTIL
- $LINK
- $UPDATE

The utility completion codes are printed on the specified 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 and appear at the end of the $EDXASM listing. 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 Code	Condition
-1	Successful completion - no errors in assembly
8	Successful completion - one or more statements had assembly errors
12	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	Location counter error - program size exceeds 64K
100	Operator cancelled assembly with ATTN CA command

Completion Codes

$IAMUT1 Completion Codes (Part 1 of 2)

Completion Code	Condition
-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
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

$IAMUT1 Completion Codes (Part 2 of 2)

Completion Code	Condition
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

Completion Codes

$JOBUTIL Completion Codes

The $JOBUTIL completion codes are displayed on the terminal used to access $JOBUTIL. The codes are as follows.

Completion Code	Condition
-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

$LINK Completion Codes (Part 1 of 3)

Comp. Code	Condition	Cause code	Action code	Retur 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: - misspelled name or volume - data set not allocated	1	5	12
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

$LINK Completion Codes (Part 2 of 3)

Comp. Code	Condition	Cause code	Action code	Return code
23	End of file reached on DS9	1	2	12
24	Disk error reading DS9	2	2	12
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

$LINK Completion Codes (Part 3 of 3)

<u>Cause Codes</u>

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

<u>Action Codes</u>

1	Log warning message and continue at next 'INCLUDE'
2	Terminate $LINK with error message
3	Continue as if expected occurance 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'

<u>Return Code Definitions</u>

-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

Completion Codes

$UPDATE Completion Codes

The $UPDATE completion codes are displayed on the terminal used to access $UPDATE. The codes are as follows:

Completion 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

EVENT DRIVEN LANGUAGE AND FUNCTION RETURN CODES

The return codes and their meanings are presented in alphabetic order according to function as follows:

- $DISKUT3
- $PDS
- BSC
- Data Formatting
- Disk and Tape (READ/WRITE)
- EXIO
- Floating-point
- Formatted Screen Image as follows:
  - $IMDATA subroutine
  - $IMOPEN subroutine
  - $IMPROT subroutine
- Indexed Access Method
- Multiple Terminal Manager
- SBIO (Sensor Based I/O)
- Terminal I/O as follows:
  - General
  - ACCA
  - Interprocessor Communications
  - Virtual Terminal
- TP (Host Communication Facility)

The return codes are issued by EDL instructions and EDL-invokable functions. They are returned in the first word of the task control block of the calling program unless otherwise noted.

Return Codes

$DISKUT3 Return Codes

The $DISKUT3 utility places a return code in the first word of the DSCB specified. The return codes for $DISKUT3 are listed below.

Return Code	Condition
1	Invalid request code parameter (not 1-6)
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 (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 $DISKUT3 failed ($RMU only)
21	Tape data sets are not supported

$PDS Return Codes

The $PDS utility returns the status of an event in the event control block (ECB) specified by the EVENT= parameter on the LOAD instruction. The return codes for $PDS are listed below.

Return 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

Return Code	Condition	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	Time-out 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:

1. 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.

Return Codes

Data Formatting Return Codes

Return Code	Description
-1	Successful completion
1	No data in field
2	Field omitted
3	Conversion error

These return codes are issued by the CONVTB, CONVTD, GETEDIT, and PUTEDIT instructions.

Disk and Tape (READ/WRITE) Return Codes

Disk and tape return codes resulting from READ/WRITE instructions are returned in two places:

1. the Event Control Block (ECB) named DSn, where n is the number of the data set being referenced.

2. the task code word referred to by taskname.

The disk and 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.

Return Codes

Disk Return Codes

Return Code	Condition
-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	Read device status I/O instruction error
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, for example, 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	Data set not open or device marked unusable when I/O was requested
12	DSCB was not OPEN; DDB address = 0

Note: The actual number of records transferred is in the second word of the TCB.

TPAE RETURN CODES

Return Code	Condition
-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 or DSCB not open
31	Incorrect request type or close request
32	Block counter error during close
33	EOV1 label encoutnered during close
76	DSN not found

Note: The actual number of records transferred is in the second word of the TCB.

Return Codes

EXIO Return Codes (Part 1 of 2)

I/O Instruction Return Codes (word 0 of TCB; word 1 of TCB contains supervisor instruction address)

Return Code	Condition
-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

EXIO Return Codes (Part 2 of 2)

Interrupt Condition Codes (bits 4-7 of word 0 of ECB)
(If bit 0 is on, bits 8-15=device ID)

Return Code	Condition
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

Return Code	Description
-1	Successful completion
1	Floating point overflow
3	Floating point divide check (divide by '0')
5	Floating point underflow

Formatted Screen Image Return Codes

These return codes are issued by the $IMDATA, $IMOPEN, and $IMPROT subroutines. They are returned in the second word of the task control block (TCB) of the calling program.

$IMDATA—Screen Image Unprotected Fields

Return Code	Condition
-1	Successful completion
9	Invalid format in buffer

$IMOPEN - Formatted Screen Image

Return Code	Condition
-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
6	Invalid volume name
7	No 3101 image available
8	Data set name longer than eight-bytes

Return Codes

$IMPROT - Screen Image Protected Fields

Return Code	Condition
-1	Successful completion
9	Invalid format in buffer
10	FTAB truncated due to insufficient buffer size
11	Error in building FTAB from 3101 format; partial FTAB created

INDEXED ACCESS METHOD RETURN CODES

The following codes indicate that the indexed data set contains errors:

Return Code	Condition
-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 to load module
14	Invalid index block found
22	Invalid IACB address
23	Get storage error - IACB
50	Data set is open for exclusive use, cannot be opened exclusively
51	Data set opened in loadmode
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
120	Invalid extract type
122	Invalid file for extract type FCBEXT

Return Codes

LOAD RETURN CODES

Return Code	Condition
-1	Successful completion
61	The transient loader ($LOADER) is not include in the system
62	In an overlay request, no overlay area exists
63	In an overlay request, overlay area is in use
64	No space available for the transient loader
65	In an overlay load operation, number of data sets passed by the LOAD instruction does not equal number required by the overlay program
66	In an overlay load operation, no parameters were passed to the loaded program
67	A disk or diskette I/O error occurred during the load process
68	Reserved
69	Reserved
70	Not enough main storage available for program
71	Program not found on the specified volume
72	Disk or diskette error while reading directory
73	Disk or diskette error while reading program header
74	Referenced module is not a program
75	Referenced module is a data set
76	One of the data sets not found on referenced volume
77	Invalid data set name
78	LOAD instruction did not specify required dat sets(s)
79	LOAD instruction did not specify required parameter(s)
80	Invalid volume label specified (see note)
81	Cross partition LOAD requested, support not included at system generation
82	Requested partition number greater than numbe of partition in the system

Notes:

1. 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.

2. Return code 80 will occur if two or more data sets reference the same tape volume.

Multiple Terminal Manager Return Codes

These return codes are returned in a caller-specified variable on the SETPAN, FILEIO, FTAB, or SETFMT function.

CODE	DESCRIPTION
-501	Screen data set not found
-500	Terminal is not an IBM 4978/4979 or 3101; no action has been taken
-2	FTAB code not link edited with application
-1	Successful completion
1	Warning: For SETPAN, this is an uninitilized panel. Input buffer has been set to unprotected blanks (x'00') and cursor position set to zero. For FTAB, no fields were found.
2	For SETPAN, unprotected data is truncated. For FTAB, the FTAB table is truncated. For SETFMT, data stream is truncated.
3	No data stream found
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
Other	Return code from READ/WRITE or the Indexed Access Method

Return Codes

SBIO (Sensor-based I/O) Return Codes

Return Code	Condition
-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

Terminal I/O Return Codes

These codes are returned by the PRINTEXT, READTEXT, and TERMCTRL instructions. The codes differ depending on the type of terminal being accessed. Separate tables show general codes, ACCA, General Purpose Interface Bus, Interprocessor Communications, Series/1 to Series/1, and Virtual Terminal return codes.

Terminal I/O - General

Return Code	Condition
-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
>10	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 0 - Unused
-	Bit 1 - System error (command reject)
-	Bit 2 - Not used
-	Bit 3 - System error (DCB specification check)
-	Bit 4 - Storage data check
-	Bit 5 - Invalid storage address
-	Bit 6 - Storage protection check
-	Bit 7 - Interface data check

Note: For 2741 or PROC devices, subtract 128, not 10; the result then contains status word 1 of the ACCA. (Refer to Communication Features Description to determine the special error condition.)

Return Codes

Terminal I/O - ACCA Return Codes

Return Code	Condition
-1	Successful completion
Bit	**Condition**
0	Unused
1-08	ISB of last operation (I/O complete)
9-10	Unused
11	1 if a write of 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

Terminal I/O - Interprocessor Communications Return Codes

	CODTYPE=		
Return Code	EBCD/CRSP	EBCDIC	Condition
-2	1F	FDFF	End of transmission (EOT)
-1	5B	FEFF	End of record (NL)
Handled by device support	Not used	FCFF	End of subrecord (EOSR)

Return Codes

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.

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 (PRINTEXT 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).

Return Codes

TP (Host Communication Facility) Return Codes (Part 1 of 3)

Return Code	Condition	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 Supervisor request code was neither ACK0, ACK1, WACK, or a NACK	
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 thanan 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

TP (Host Communication Facility) Return Codes (Part 2 of 3)

Return Code	Condition	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

TP (Host Communication Facility) Return Codes (Part 3 of 3)

Return Code	Condition	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 LANGUAGE AND FUNCTION POST CODES

The Event Driven language and function post codes are returned to the first word of the event control block (ECB) (unless stated otherwise) to signal the occurrence of an event.

Post Codes

Tape Post Codes

If you initialize a tape by loading $TAPEIT from a program or by invoking $JOBUTIL and passing the above parameters, the following post codes are returned to the event control block (ECB) of the calling program.

Post Code	Condition
-1	Function successful
RC	Any tape I/O return code
101	TAPEID not found
102	Device no offline
103	Unexpired data set on tape
104	Cannot initialize BLP tapes