

RSX-11D

USER ENVIRONMENT TEST
MD-11-DBZBB-A
PACKAGE (UETP)

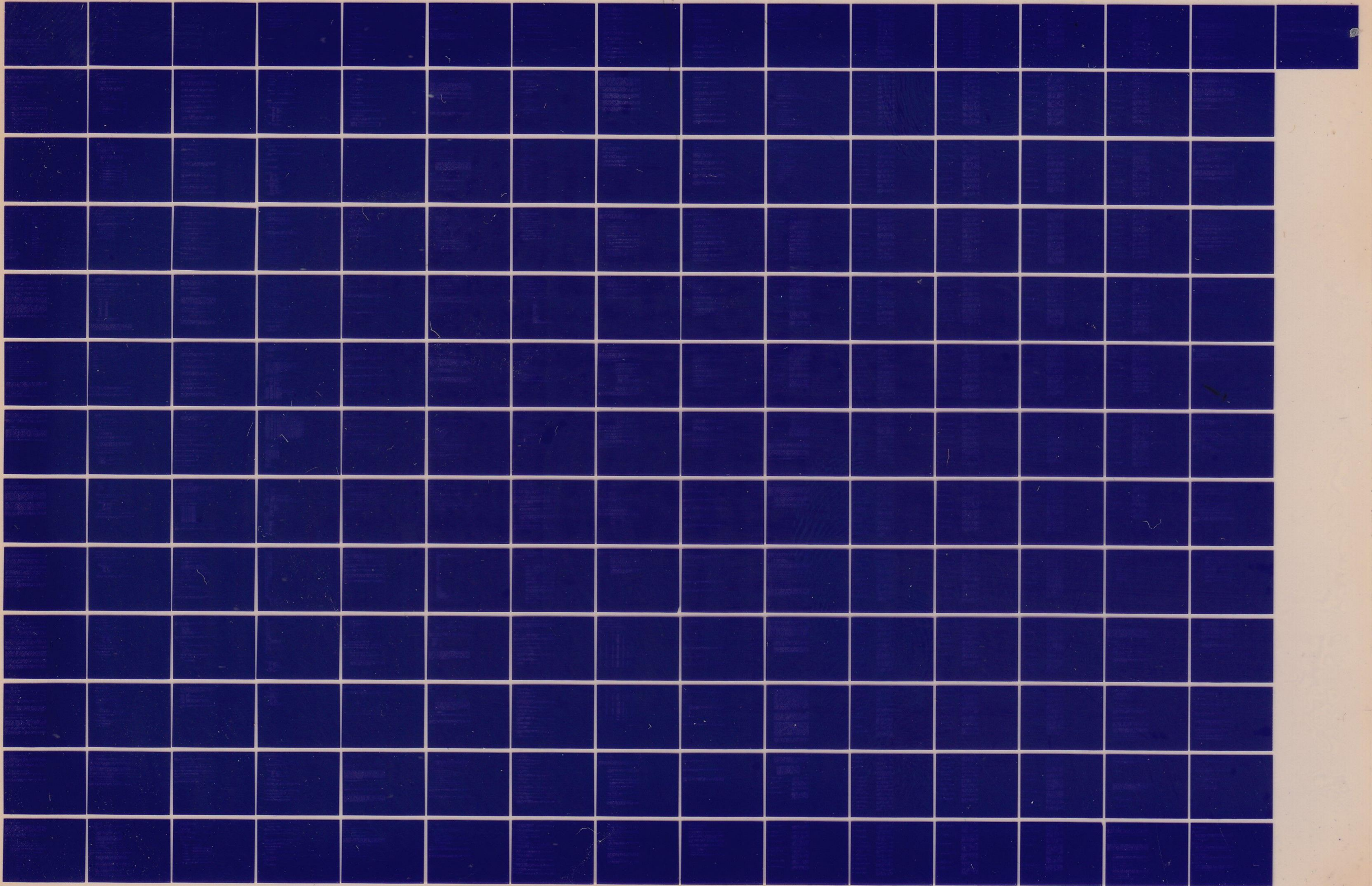
EP-DBZBB-A-DL

MAY 1978

COPYRIGHT 75-76
FICHE 1 OF 1

digital

MADE IN USA



IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DH2HR-A-D

PRODUCT NAME: PSX-110 USER ENVIRONMENT TEST PACKAGE (UETP)

REFERENCE: MAST APPENDIX D
 RELEASE #V03-01
 (PSX-110 MONITOR V&P)

DATE RELEASED: MARCH, 1976

MAINTAINER: SOFTWARE QUALITY MANAGEMENT

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSIDERED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT APPEAR IN THIS DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE OR EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1975, 1976 BY DIGITAL EQUIPMENT CORPORATION

DPA10 OVERVIEW

This test procedure is a system software exerciser routine based on the RSX-11D operating system. This procedure is applicable against those systems capable of operating RSX-11D (ref. DWP20) and having completed the prerequisite actions defined in Section PPA10.

The user is led through the system bootstrap system generation and simulated user environment system operation. Valid and erroneous system responses are defined at each stage of the procedure.

DPA15 RSX-11D UETP RESTRICTIONS

1. If you are to build an RF system, a minimum of two RF's is required.
2. RPA7 is not supported per this release.

DPA10 PRELIMINARY CHECKS AND TASKS

The following must have been accomplished or available before starting this test sequence:

1. Appendix B (DEC-X11) has run without error.
2. Latest RSX-11D distribution media for monitor VAK is available.
3. Insure that all devices ordered by the customer (Ref. construction req or key sheet) and not designated as field installed (manufacturing only) are physically connected to the system.
4. Latest RSX-11D UETP distribution media (VP3-00) is available.
5. Insure that the customer distribution PPA5 or PPA3/RPA4 has been backed up (copied) using PPFSPV or another appropriate program.
6. Insure that all test volumes (test disks, dectapes, etc.) are formatted volumes.
7. For manufacturing only:
 - a. Remove the ACT daughter station and install the terminator.
 - b. Insure that all hardware communications options are

cabled and ready to run on-line with all turn-arounds removed.

- c. Insure that a general PM of the system is performed.

DP220 RSK-110 UETP HARDWARE USAGE TABLE

DEVICE	HARDWARE TESTED	COMMENTS
MEMORY	YES	4PK minimum
PK25/PK23	YES	All units including W are exercised.
RP22/RP21	YES	Same as PK25 (RP22 not supported per this release)
RF24	YES	Same as RF25
RS23/RS24	YES	Same as RF25
RF/RS11	YES	Same as RF25
RC11/RS64	NO	Not supported by PSX
TM22/TU16/TU45	YES	Same as PK25
TM11/TU12	YES	Same as RF25
DECTAPE	YES	Same as RF25
LINE PRINTER	NO	Used only for batch log output.
CARD READER	NO	Not tested per this release.
DJ11	NO	Not supported by UETP
DH11	NO	Not supported by UETP
PC11	NO	Not supported by UETP
TA11	NO	Not supported by UETP
K111,LC11,DL11A,DL11-B, DL11-C,DL11-D,DC11,DL11-E DM11-BB,DG11,DP11,DU11	NO	Not supported by UETP
DM11-A,DM11-A,DP11-A,C, PA611,DT23-FP,DX11, GT44,LPS11,KW11	NO	Not supported

DIAP CONCEPTS AND TERMINOLOGY

PSX-11D is a partitioned multiprogramming system. Partitions are named, contiguous blocks of memory, the size and number of which are fixed during system generation. All tasks in all partitions can execute in parallel. Partitions can be either user-controlled or system-controlled.

A user-controlled partition can accommodate only one task at a time. A system-controlled partition can accommodate as many tasks as can fit in the defined physical space. All tasks in a system-controlled partition can run in parallel.

An active task in one in main memory that is competing for system resources. A task can be checkpointed to make room for a higher priority task to execute in that partition if the first task is designated as checkpointable.

Before a task can execute, it must be installed. More than one task can be installed to run in a partition. The main purpose of the installation procedure is to record disk retrieval pointers in main memory so that the task can be made ready to execute with minimum delay when a request is issued for it. The task can be either explicitly installed using the INS command to MCR or implicitly installed as a result of a RUN command issued by a nonprivileged user.

The system task directory (STD) establishes the maximum number of tasks that can be installed at one time. Normally the number of installed tasks is greater than the number of executing tasks. The number of simultaneously installed tasks is limited by the number of system task directory entries specified during system generation. The partitions and the number of tasks that can fit into system-controlled partitions. The number of STD entries for tasks should be greater than the number of available partitions so that a maximum number of tasks can execute simultaneously. Installed tasks can be removed as needed to free additional STD entries.

In PSX-11D dynamic memory requirements are satisfied from a pool of nodes. Nodes are variable-size memory blocks that are a multiple of 8 words. The size of the node pool is established during system generation.

The modular construction of PSX-11D allows the user to configure available hardware and software resources to fit a particular processing requirement. The use of memory partitions and priority scheduling facilitates user control over the execution of many parallel real-time functions.

PSX-11D features include:

- Fast interrupt response and servicing
- Simultaneous monitoring of multiple activities
- 256 priority levels for task execution
- Priority servicing of I/O requests
- Convenient storage and recall of disk-resident programs
- Efficient, convenient task scheduling facilities
- Multiple memory partitions to contain tasks of varying sizes
- Event Flags for task synchronization and notification
- Checkpointing, a form of memory sharing
- On-line program development, concurrent with task execution
- FORTRAN and MACRO-11 programming languages and utilities
- Asynchronous execution of I/O-dependent code

D1100 TASKS

The basic program unit under PSX-11D is called a task. A task consists of one or more programs that have been written in FORTRAN and/or MACRO-11 Assembly Language or COROL. Relocatable object modules are generated and installed into the system on-line, making them available in absolute memory-image format on the disk. A task can initiate another task's execution in various ways, such as:

1. Request immediate execution;
2. Request execution contingent upon available memory;
3. Schedule at a future time, with optional rescheduling at periodic intervals.

All of these task initiation functions can be accomplished from the MCP console, as well as from a currently executing task.

RSX-11D is event-driven, in contrast to systems which use a time slice mechanism for determining a task's eligibility to execute. Under RSX-11D, the highest priority task can run continuously until some event or condition in the system causes it to be suspended. Another event or change in system status can reactivate the task.

Tasks can be activated either by the operator or by another task. Activation can be conditional, based on currently available partition space (EXECUTE) or it can occur as soon as possible (REQUEST), or as soon as possible after some future time (SYNC, SCHEDULE, and RUN).

D1114 PARTITIONS

Partitions are areas of contiguous real memory that are used for task execution. There are two modes of partition usage: user controlled where only one task at a time can occupy the partition and system controlled where the system controls allocation of memory within the partition for execution of one or more tasks. The name, base address, size, and mode of each partition are specified at system generation and cannot be changed on-line. Tasks are installed to run in particular partition, but, upon specific request, can run in any partition that is large enough.

Normally, an active task remains resident in its memory space until its execution is completed. An exception to this is a checkpointable task.

D1124 MULTIPROGRAMMING

Effective multiprogramming is achieved when many tasks reside in memory simultaneously, spending some of their residency waiting for I/O completion, waiting for synchronization with other tasks, or in some way being unable to continue execution. While one or more tasks are waiting, another task can utilize the central processor's resources.

Under RSX-11D, tasks are run at a software priority level ranging from a low of 1 through a high of 256. The Executive grants central processor resources to the highest priority task that is capable of execution. When a task becomes ready to execute, and it has a higher priority than the currently executing task, the Executive interrupts the lower priority task and allows the higher priority task to run. Execution of the interrupted task continues when it once again becomes the highest priority task capable of execution. The environment of an interrupted task is preserved: except for elapsed time, interruption is transparent to an interrupted task.

This multiprogramming scheme normally applies only to memory-resident tasks. Once a task is in memory, the Executive allows it to run to completion in a multiprogramming fashion even if its memory becomes required for the execution of a higher priority, non-resident task. However, when it is desirable to free a partition for execution of a higher priority task, a task can be declared checkpointable when it is installed. A checkpointable task is swapped-out when its partition is required for a higher priority task, and swapped-in when it once again becomes the highest priority task requiring its partition.

Normally, a task is brought into memory only upon a request for its execution, and several tasks can use the same memory. However, when desirable, a task can be fixed in memory, permitting faster response to requests for execution, by dedicating a partition, or part of one, to a single task.

D1130 SIGNIFICANT EVENTS AND SYSTEM TRAPS

RSX-11D is an event-driven system in which task execution is governed by the occurrence of significant events. A significant event is any change in system status that affects the execution of a task. For example, completion of an I/O operation is a significant event.

One of the ways that significant events are signalled is through event flags. There are 64 event flags. Flags 1 through 32 are local to the task, while the 33 through 64 are common to all tasks. A task can set, clear, test, and wait for any event flag or combination of event flags, to achieve efficient synchronization between itself and other tasks in the system.

When a significant event occurs, the Executive scans an active task list seeking the highest priority task that can be executed. When an eligible task is found, it is run until it exits, suspends execution, waits for a significant event, or a significant event occurs.

System traps are another means of governing task execution. While significant events have a system-wide scope, traps are local to a task. Traps interrupt the sequence of instruction execution in the task, and cause control to be transferred to a prespecified point in the program. Traps can be either synchronous or asynchronous.

Synchronous traps allow servicing of fault conditions that can occur internally in a task, such as memory protection violation.

Asynchronous traps are the result of significant events in that the interrupts they generate inform a task that a significant event has occurred e.g., I/O complete.

Trap service routines may or may not be provided by the user to handle the synchronous and asynchronous traps. If no synchronous trap service routine is provided, the faulting task is aborted. If no asynchronous trap service routine is provided, the task continues to execute with no interruption.

D1142 I/O OPERATIONS -----

The Executive's main function in I/O operations is to handle I/O requests from tasks and pass the requests to the appropriate device handler task. The general method follows.

1. A QIO directive is issued by a task. The task specifies a number of parameters that are required in processing the I/O request. One of these parameters is the logical unit number (LUN), assigned to a device by the task.
2. The Executive fields the QIO directive, and examines the LUN parameter to determine which device handler is to process the request. The particular device handler is chosen by mapping the LUN of a particular task into an entry in the physical unit directory using the logical unit table.
3. The I/O request is put in the request queue of one of a set of special tasks (device handlers).

The requesting task can either suspend operation until the I/O request is completed or continue to operate until interrupted by an asynchronous system trap. RSX-11D permits parallel I/O requests to be issued by the same task. That is, the task continues executing after issuing a QIO and subsequently can issue further QIO requests without waiting for the previous request to be completed.

Some device handlers operate in conjunction with the file control primitives (FCP) to manipulate files. When an FCP routine is required, the device handler issues a SEND/REQUEST which initiates operation of the specified FCP routine.

I/O requests are queued for each unit by priority (usually requester task priority), and handler tasks pick requests from the top of request queues. Thus, preferential service is given to high priority requesters. However, when appropriate, devices can be attached to a task, in which case only requests from the attached task are dequeued. This continues until a "detach unit from task" request is dequeued, again.

The right to attach and detach devices is controlled by access privileges, which are defined for each device. Requests to attach a device are rejected if the requester does not have the proper access rights. Note that because device handler tasks can service many units, they are not themselves attached.

The interface between a device handler task and the RSX-11D system is accomplished by directives and by re-entrant system subroutines (via, to attach, detach, and dequeue). The major effort in developing an RSX-11D handler task is in driving the device, and not in completing an interface to a host system.

D1150 DEVICE HANDLERS

Device Handlers are tasks that support I/O devices. These tasks are similar to normal tasks within the system with the following additional features:

They usually contain an interrupt service routine to respond to hardware interrupts;

They are allowed to gain access to any memory areas including privileged ones;

A naming convention exists for device handlers. Their task names consist of two alphabetic characters, followed by four dots. For example, the line printer handler is named as follows:

LP....

Device handler tasks are loaded into memory on command of the operator as needed. Requests from user tasks are queued by the Executive to the device handler according to the priority of the I/O request. If no priority is specified, that of the requesting task is used by default. When necessary, however, the requesting task can reserve a device for its exclusive use for a period of time by attaching it.

D1160 THE MONITOR CONSOLE ROUTINE

Operator interface to the system is provided by a facility called the monitor console routine (MCR).

MCR dialogue is established by typing CTRL/C on a terminal. This causes an MCR dispatch task to run. It prints an MCR> prompting string and reads a line of command input. The command input line indicates what function is to be performed and contains parameters when necessary. The dispatch task causes an MCR function task to run to perform the requested function.

A typical system might have MCR functions to provide system status, perform task scheduling, change logical unit assignments, and to perform other necessary functions.

Since normal MSX-11D tasks are used to implement MCR functions, special purpose functions to provide added flexibility of convenience for a particular application or installation can be developed easily and added to the system.

D2200 GETTING PSX-110 UFTP ON THE AIR

This section describes the procedures to generate a PSX-110 VHA system from the distribution media. The user performs the system generation procedure using the Minimumly configured PSX-110 monitor as the system generation monitor. Patch command files are used as the basis for the target system generation process.

Following system generation the operator is led thru the PSX-110 initialization procedures which consist of creating the target disk. When this is completed the UFTP patch streams are executed and the testing begins.

D2210 CONVENTIONS USED IN THIS DOCUMENT

Throughout this document all responses which are to be typed by the user are indicated by being underlined as in the following example:

TIME: 12:45

All responses are terminated by a carriage return (<CR>) unless otherwise indicated by having the line terminator enclosed in carots (i.e., <ALTMODE> for Altmode).

D2100 BOOTSTRAP PROCEDURES

In order to transfer the distribution medium onto the system disk, the distribution medium must be bootstrapped. Five models of hardware bootstraps are available on systems used for MSX 110: MP110H, RM7921A, BM73YA, BM73YR and the M9301-YC. The type of bootstrap for a particular PDP-11 can be determined by consulting the equipment order. A section describing the procedure for bootstrapping TUIV magtape when one of the five above mentioned bootstraps is not available is also included.

Whenever a request to bootstrap the system is encountered in the following text, refer to one of the six sections that follow to perform the appropriate bootstrap.

D2110 MP110H BOOTSTRAP

Perform the following steps to use an MP110H Bootstrap.

1. On the console switches, set HALT/ENABLE switch to its HALT position and back to its ENABLE position.
2. Enter the address of the device from which the bootstrap is to occur into the console switches. Table D2110 provides the device addresses.
3. Press the LOAD ADDR switch followed by the START switch.

Table - D2110

Device Addresses for the MP110H Bootstrap

DEVICE	ADDRESS
-----	-----
RM03 Disk	173540
RK05 Disk	173110
RF11 Disk	173100
TUIV MAGTAPE	173135
DFCtape	173120

D2124 HM792YH BOOTSTRAP

Perform the following steps to use a HM792YH Bootstrap.

1. On the console switches, set HALT/ENABLE switch to its HALT position and back to its ENABLE position.
2. Enter 173144 into the display switches.
3. Press the LOAD ADDR switch.
4. Enter the address of the device from which the bootstrap is to occur into the console switches. Table D2124 provides the device addresses.
5. Press the START switch.

Table - D2124

Device Addresses for the HM792YH Bootstrap

DEVICE	ADDRESS
RP04 Disk	176716
RP05 Disk	177400
RF11 Disk	177462
TECtape	177344

NOTE: Magnetic tape cannot be booted with the HM792YH Bootstrap.

D2130 BMR73YA BOOTSTRAP

Perform the following steps to use the BMR73YA Bootstrap.

1. On the console switches, set HALT/FWAKE switch to its HALT position and back to its ENABLE position.
2. Enter the address of the device from which the bootstrap is to occur into the console switches. Table D2130 provides the device addresses.
3. Press the LOAD ADDR switch.

NOTE: If a unit other than 0 contains the device to be booted, set the switch register to the unit number of the device to be booted before pressing START.

4. Press the START switch.

Table - D2130

Device Addresses for BMR73YA Bootstrap

DEVICE -----	ADDRESS -----
RF11 (RS11 disk)	773000
RP11 (RP03 disk)	773100
RK11 (RK05 disk) -- Unit 0	773010
Unit specified in switch register	773020
TC11 (DECTape)	773030
TM11 (TU10 magnetic tape)	773050

D2140 RMA73YB BOOTSTRAP

Perform the following steps to use the RMA73YB bootstrap.

1. On the console switches, set HALT/ENARIF switch to its HALT position and back to its ENARIF position.
2. Enter the address of the device from which the bootstrap is to occur into the console switches. Table D2140 provides the device addresses.
3. Press the LOAD ADDR switch.

NOTE: If a unit other than 0 contains the device to be booted, set the switch register to the unit number of the device to be booted before pressing START.

4. Press the START switch.

Table - D2140

Device Addresses for RMA73YB Bootstrap

RM11 (RS03/4 disk) -- Unit 0	773100
Unit specified in switch register	773102
PK11 (PK05 disk) -- Unit 0	773130
Unit specified in switch register	773132
PM11 (PP04 disk) -- Unit 0	773320
Unit specified in switch register	773322
RP11 (RP03 disk) -- Unit 0	773350
Unit specified in switch register	773352
RF11 (RS11 disk)	773136
TC11 (DECTape)	773170
TM11 (TU11 magnetic tape)	773110
RH11 (TU16/TM02 magnetic tape)	773150

D2150 M9301-YC BOOTSTRAP

If the M9301-YC Bootstrap/Diagnostic loader is on the system perform the following steps.

1. Move the CPU console ENABLE/HAIT switch to its HAIT position and back to its ENABLE position.
2. Set the CPU switch register to 17773000.
3. Depress the CPU LOAD ADRS switch.
4. Set the CPU switch register to one of the following values depending on the system option from which bootstrapping is to be accomplished (unit 0 only):

00000010 for TM11/TU10 magtape

00000020 for TC11/TU56 DECTape

00000030 for RK11 disk cartridge

00000040 for RP03 disk pack

00000060 for RP04 disk pack

00000070 for TM02/TU16 magtape

5. Depress the CPU START switch.

D2160 BOOTSTRAPPING TMI1/TUI1 MAGTAPE WITHOUT MP11-DH

 OR HMA73 LOADERS

To bootstrap a TMI1/TUI1 magtape when the system has neither the HMA73 nor the MP11-DH loader, the user must manually enter a load routine into memory using the CPU console Switch Register and the DEP switch.

To load the routine, perform the following steps.

1. Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.
2. Set the CPU Switch Register to 010000.
3. Depress the CPU LOAD ADPS switch.
4. Load the following contents into memory using the Switch Register and DEP switch.

Address -----	Contents -----
010000	012700
010002	172520
010004	005310
010006	012740
010010	060011
010012	105710
010014	100376
010016	005710
010020	100767
010022	012710
010024	060003
010026	105710
010030	100376
010032	005710
010034	100777
010036	005007

5. Set the Console Switch Register to 010000.
6. Depress the CPU LOAD ADPS switch.
7. Depress the CPU START switch.

If the system reads the tape but halts at address 010034, the device generated a magtape error. The user can try another drive. If the system appears to take no action and halts, verify the accuracy of the routine by using the CPU Console EXAM switch. Use the Switch Register and the DEP switch to correct any erroneous contents. Rewind the tape to its load point before executing the routine again. If no recovery is successful, it will be necessary to have the hardware checked.

D2170 SUMMARY OF HARDWARE BOOTSTRAP ADDRESSES

Device to Bootstrap -----	Bootstrap Type				
	HMR73-YA	HMR73-YH	MP11-DH	M792-YH(1)	M9301-YC(3)
PF11 disk	773000	773136	773100	777462	-
PK11 disk cartridge	773010	773030	773110	777406	00000130
RP13 disk pack	773100	773350	773154	776716	00000140
RP14 disk pack	-	773320	-	-	00000070
TM11/TU10 magtape	773050	773110	773136	(2)	00000010
TM02/TU16 magtape	-	773150	-	-	00000060
TC11/TU56 DECtape	773030	773070	773120	777344	00000020

 (1) For the M792-YH loader, set the address 773100 in the Switch Register, depress the LOAD ADRS switch, set the value from the table in the Switch Register, and press the START switch.

(2) To bootstrap a magtape, use the loading routine described in Section E2160.

(3) For the M9301-YC Loader, set the address 17773000 in the switch register, depress the load adrs switch, set the value from the table in the switch register, and press the start switch.

D2120 BOOTSTRAPPING THE DISTRIBUTION MEDIUM--MAGTAPE (RSX-11D V6M

 SYSTEM TAPE)

1. Place the distribution magtape in the appropriate drive, e.g., 7- or 9-track magnetic tape.
2. Bootstrap the distribution magtape following the procedures in section D2100. The system prints the following on the console.

RSX-11D SYSTEM DISTRIBUTION TAPE

SYSTEM DISK?

3. Respond with one of the following to indicate which device is the system disk.

DK for RK05 system disk
 DP for RPA3 system disk
 DB for RPA4 system disk

4. Once the name of the system device has been typed, the system prints the following message to find out if the bad block utility has been run on the system disk.

HAS "BADBLOCKS" BEEN RUN?

5. If the bad block utility has been run type YES; if not, type NO. If bad block has been run, a system is created on the disk. If the utility has not been run, the system executes it and then creates a system on disk. Information is printed on the console while the system is created.

When the system prints this message, it is ready for use.

END OF SYSTEM GENERATION PHASE 2

6. Type "CNTRL/C" and the following MCR commands to save the RSX image of the newly created disk.

```

^C
--
MCR>HEL (1,1) - say hello
-----
MCR>DMU SY: - Dismount the disk messages will
----- be printed.
MCR<FIX FIIACP
-----
MCR><ALTMODE> - Altmode forces silent command
----- mode.
```

7. Type the following silent commands. They will not echo on the console.

```

SAVE <CR>      - Save the image
-----
MOUNT SY:<CR>  - Mount the disk
-----
TIME<SPACE>   - Ask for the time (space is needed)
-----
^C            - CTRL/C
--

```

The system image will now be saved and the disk will be bootstrapped. The following will be printed on the console.

124K (4096) RSX-11D V006H

```

SAV == PARTITION GEN   EXPANDED BY   2432*32 (DEC)
4096S
MCP>MOUNT OK:
MOUNT==VOLUME INFORMATION==
      DEVICE  =DK0
      CLASS   =FILE 11
      LABEL   =
      UIC     =1,1
      ACCFSS  =(RWED,RWFD,RWFFD,RWF1)
      CHARAC  =()
MCP>; SYSTEM CARTRIDGE
MCP>TIME

```

8. Remove the RSX-11D distribution cartridge (DEC-11-0XV6A-W-MC9 or MC7) from unit A.
9. The system distribution disk is now created. Proceed to Section D2335 if the UETP distribution media is an MKW5 or to section D2340 if cartridge distribution.

D2334 BOOTSTRAPPING THE DISTRIBUTION MEDIA--RK05

1. Place the distribution disk on Unit 0. Make it ready and write enabled.
2. Bootstrap the system disk following the procedures in Section D2104. The system prints the following on the console.

124K (WORD) PSX-11D V006H

SAV == PARTITION GEN EXPANDED BY 2432*32 (JFC)
 WORDS

MCP>MOU DK:

MOUNT==VOLUME INFORMATION==

DEVICE =DK0

CLASS =FILE 11

LABFL =

UIC =(1,1)

ACCESS =(RWED,RWED,RWED,RWED)

CHARAC =()

MCP>; SYSTEM CARTRIDGE

MCP>T14

3. The system distribution disk is now created. Proceed to section D2335 if the UFTP distribution media is an RK05 disk, or to section D2342 for magtape distribution.

D2335 MOUNTING THE UETP DISK
.....

1. Respond to the MCR>TIM prompt by entering today's date and the current time as follows:

```
MCR>TIM 7/15/75 8:25:00
.....
```

2. Place the UETP distribution disk on unit 1. Make it ready and write enabled. Type the following MCR command to mount the UETP disk.

```
MCR>MOU DK1:UETPSY
.....
```

The system will print the following on the console.

```

MOUNT-==VOLUME INFORMATION==
      DEVICE  =DF1
      CLASS   =FILE 11
      LABEL   =UETPSY
      UIC     =([1,1])
      ACCESS  =([RWED,RWED,RWED,WAFF])
      CHARAC  =([
MCR>
```

3. Respond to the MCR> prompt by entering the following command to install the UETP SYSGEN task.

```
MCR>INS DK1:[200,201]SYSGEN
.....
```

4. Respond to the MCR> prompt by entering the following command to run the SYSGEN task.

```
MCR>RIN SYSGEN<ALTMODE>
.....
```

5. Proceed to Section D2400.

D2340 MOUNTING THE UETP MAGTAPE

1. Respond to the MCR>TIM prompt by entering current date and time as follows:

```
MCR>TIM 12/30/75 8:25:00
-----
```

2. Place the UETP distribution magtape on unit 2 and take it ready.
3. Type the following MCR commands to mount the UETP magtape and transfer the SYSGEN task to the system disk:

(Replace MT in the commands below with UU if the magtape drive is a TU16.)

(If you are using the TU16, INS (11,1)TU16)

```
-----
MCR>LOA MT
-----
MCR>MOU MT:/CHAS(FOR)
-----
MCR>FLX SY:/CO=MT;(200,201)SYSGEN,ISP/HL:04.
-----
```

4. Respond to the MCR prompt by entering the following command to install the UETP SYSGEN tasks:

```
MCR>INS SYSGEN
-----
```

5. Respond to the MCR> prompt by entering the following command to run the SYSGEN task.

```
MCR>RUN SYSGEN<ALTMODE>
-----
```

6. Proceed to Section D2400.

D2400 SYSGEN PROCEDURES

.....

When the SYSGEN task starts it will enter into a dialogue with the user. The dialogue is a series of questions dealing with the hardware/software configuration of the target system.

If the operator desires to restart or abort the SYSGEN task he need only type RE or AH respectively. If an incorrect response is given before the operator types the carriage return he may type "ERASE" once for each character to be deleted.

If SYSGEN detects an incorrect response it will print a question mark or a warning message and repeat the question using the long form of the query. The operator should be aware, however, that some responses will not be checked. In this case the operator must be sure of his response to avoid errors during system generation PHASE 1.

After the operator answers the configuration questions batch jobs will be created. SYSGEN will then instruct the operator to perform the steps necessary to create the target disk.

When all of the SYSGEN procedures have been completed the following message will be printed on the console:

END OF SYSTEM GENERATION PHASE 2

At this point the user should proceed to section D2450.

If the user is uncertain of any reply or procedure of the SYSGEN process he should consult sections D2410 thru D2460. Some sample SYSGEN printouts are shown in section D2460.

D2410 SYSGEN PHASE 0

SYSGEN PHASE 0 consists of the question and answer session controlled by the SYSGEN task. When SYSGEN starts the following will be printed on the console.

MSX-110/IAS SYSGEN V.3-00 PHASE 0

THE VALIDITY OF SOME ANSWERS ARE NOT CHECKED.
 AN INCORRECT CHARACTER OR LINE RESPONSE MAY
 BE DELETED BY TYPING "HUPDT" OR "CONTROL H"
 RESPECTIVELY.

A RESPONSE OF "CARRIAGE RETURN" WILL DEFAULT A
 YES/NO QUERY TO "N" AND A NUMERIC QUERY TO "0".
 ALL OTHER QUERIES WILL BE DEFAULTED TO NULL.
 A RESPONSE OF "ALTMODE" OR "ESCAPE" WILL CAUSE
 THE LONG FORM OF THE QUERY TO BE PRINTED.

YOU MAY TYPE "RE" OR "AB" AS A RESPONSE TO A
 QUERY TO RESTART OR ABORT SYSGEN.

DATE: 23-DEC-1975 -type todays date

SYSTEM NAME: SYS #2027 -type the system name

A series of configuration questions will now be asked.

Please note that only those responses marked by an asterisk are required responses for NETP operation. All others are for example only.

1. LONG DIALOG RESULTS IN THE DISPLAY OF EXPLANATORY TEXT PRECEDING MOST QUERIES. SHORT DIALOG OMITTS THE TEXT. LONG DIALOG IS AVAILABLE ON A PER QUERY BASIS BY ENTERING ESCAPE OR ALTMODE FOLLOWING THE DISPLAY OF THE QUERY.

DO YOU WANT LONG FORM OF DIALOG (Y OR N)? N

Answer with "Y" if you desire long form of queries. Long form is also available by answering any query with ALTMODE or ESCAPE. A summary of the long form of dialog for each query can be found in section 7.4.

2. ARE YOU GENERATING AN IAS SYSTEM (Y OR N)? N

---Note (not included in this sysgen at this time)
 Answering this question with "Y" will create a SGW batch stream using IAS batch.

3. TARGET DISK# (DKN,DPN,DHN)? DPN

 This question requests the name and unit number of the disk

on which the MSX-110 or IAS system will be created. The unit number does not have to be 0 although the disk being created will run from unit 0. The PKMS (14) is not a valid IAS target disk.

4. CPU# (40,45,70)? 40

--
This question requests the particular type of processor on which the target system will run. The 11/40 processor cannot be used on an IAS system.

5. WHAT IS THE MEMORY SIZE (IN 1K BANKS)? 64

--
This question requests the maximum memory size of the target system. The minimum memory size for 11/40 and 11/45 RSX-11D systems is 48K while the minimum size for 11/70 and IAS systems is 64K. The maximum memory supported is 124K for 11/40 and 11/45 systems and 1024K for 11/70 systems.

6. IS THE FPP OPTION AVAILABLE (Y OR N)? N

-
This question is asked to determine if the target system has the floating point option. This question is asked if the target CPU specified in Question 4 was an 11/45 or 11/70.

7. IS THE FIS OPTION AVAILABLE (Y OR N)? Y

-
This question is asked to determine if the target system has the floating instruction set option. This question is only asked if the target CPU specified in question 4 was an 11/40.

8. IS A KW11-P CLOCK AVAILABLE (Y OR N)? Y

-
Answer this question with "Y" if a KW11-P programmable clock is configured on the target system. Answering "N" will cause the clock to be a KW11-L.

9. SHOULD IT RUN AT 60HZ (Y OR N)? N

-
Answer this question with "N" if you desire the KW11-P to run at a frequency other than 60. This question is not asked if the answer to question 8 was "N".

10. ENTER THE DESIRED KW11-P CLOCK SPECIFICATION IN THE FOLLOWING FORMAT: <HZ,TYPE,TICS/HZ> <CF>
NOTE: THE BRACKETS "<>" ARE NECESSARY.

KW11-P SPEC : <1200,1,10>

This question is asked if the answer to question 9 was "N". The user is requested to enter his clock specification in the format shown. Refer to the RSX-11D or IAS System Generation Manual for a detailed description of clock specifications.

11. IS THE POWER LINE FREQUENCY 50HZ (Y OR N)? N

-
If question 8 was answered "N" this question is asked to determine the line frequency for the KW11-L clock.

12. HOW MANY RK05 DISK DRIVES ARE AVAILABLE (N-R)? 2

This question requests the number of RK05/RK03 disk drives connected to the target system.

13. DO YOU WANT TO USE THE OVERLAPPED SEEK HANDLER (Y OR N)? Y

Answer this question with "Y" if you want to use the RK05 handler for overlapped seeks.

14. HOW MANY RPP4 DISK DRIVES ARE AVAILABLE (N-R)? 2

This question requests the number of RPP4 disk drives connected to the target system.

15. HOW MANY RPP2/RP03 DISK DRIVES ARE AVAILABLE (N-R)? 1

***WARNING--RPP2 DISK DRIVES ARE NOT SUPPORTED FOR UEFI OPERATION.

This question requests the number of RPP2 and RP03 disk drives connected to the target system. This question will not be asked if the answer to the RPP4 question was non-zero. If this question was answered non-zero then the following question will be asked for each drive specified.

16. IS DRIVE UNIT # N AN RP02 OR RP03? RP03

Answer this question with "RP02" or "RP03" depending on the drive type.

17. HOW MANY RSN3/RS04 DISK DRIVES ARE AVAILABLE (N-R)? 1

This question requests the number of RSN3 and RS04 disk drives connected to the target system. If this answer is non-zero the following question will be asked for each drive specified.

18. IS DRIVE UNIT # N AN RSN3 OR RS04? RSN3

Answer this question with "RSN3" or "RS04" depending on the drive type.

19. HOW MANY RF11 DISK PLATTERS ARE AVAILABLE (N-R)? 1

This question requests the number of RF/RS11 platters connected to the target system. This question will not be asked if the answer to question 17 was non-zero.

20. HOW MANY TU56 DECTAPE DRIVES ARE AVAILABLE (N-R)? 2

This question requests the number of DECTape drives connected to the target system. Note that each TU56 contains two drives.

21. HOW MANY TU14 MAGTAPE DRIVES ARE AVAILABLE (0-N)? 3
 This question requests the number of TU14 magtape drives (7 and 0 track) connected to the target system.
22. HOW MANY TU16 MAGTAPE DRIVES ARE AVAILABLE (0-N)? 4
 This question requests the number of TU16 magtape drives connected to the target system.
23. IS A LINE PRINTER AVAILABLE (Y OR N)? Y
 If a line printer is connected to the target system answer this question with "Y". If no line printer is specified, the "RP" or "BIP" device will not be entered in the SGN.CMD file.
24. DOES IT HAVE 132 COLUMNS (Y OR N)? Y
 Answer this question with a "Y" if the line printer connected to the target system has 132 columns.
25. IS THE PRINTER A CENTRONICS LPO5 (Y OR N)? N
 The answer to this question will determine which LP handler will be installed during Phase 2.
26. IS THE CONTROLLER AN LS11 (Y OR N)? N
 The answer to this question will determine which LP handler will be installed during Phase 2.
27. IS A CARD READER AVAILABLE (Y OR N)? N *
 If a card reader is connected to the target system answer this question with "Y".
28. IS THE CARD READER A CD11 (Y OR N)? N *
 If the card reader connected to the target system is a CD11 then answer this question with "Y". This question is not asked if the answer to question 27 was "Y".
29. IS A PAPER TAPE READER AND/OR PUNCH AVAILABLE (Y OR N)? N *
 Answer this question with a "Y" if a PC11 is connected to the target system.
30. IS A TA11/TU60 CASSETTE SYSTEM AVAILABLE (Y OR N)? N *
 If a TA11 and a dual TU60 cassette drive system is on the target system answer this question "Y".

31. CONSOLE (KSH33,KSP35,VT05,VT50,LA36,LA37S,LA37P)? LA37S/M

Answer this question according to the type of console terminal that is connected to the target system. Append /M to the response to indicate that there is more than one terminal connected to the target system. If only /M is the answer, the "TT" number will start at 1 for question 32. This permits non-standard console (TT) devices.

32. ENTER EACH TERMINAL SPECIFICATION IN THE FOLLOWING FORMAT: TYPE,VECTOR,PRIORITY,CSP <CR>
TYPE "/E" TO END SPECIFICATIONS.

TERMINAL SPEC : LA37S,320,4,176500

TERMINAL SPEC : <7,20100,56701,1207>,330,4,160000

TERMINAL SPEC : /E

--

This question is asked if /M was appended to the console terminal specification. Each terminal spec specified will be given a "TT" number starting with 1. No checks are made concerning the validity of the spec. SYSGEN will insert "DEV=TTn", where n is the unit number, before each specification. For further information concerning terminals connected to DM or DJ11 multiplexers consult the RSX-11D or IAS System Generation Manual and questions 33 and 34 below.

33. ARE ANY NON-STANDARD DEVICES PRESENT (Y OR N)? Y

-

Answer this question with "N" if the operator does not have any more devices to configure.

34. ENTER EACH NON-STANDARD DEVICE SPECIFICATION IN THE FOLLOWING FORMAT: NAME,TYPE,VECTOR,PRIORITY,CSP <CR>
TYPE "/E" TO END SPECIFICATIONS.

DEVICE SPEC : CT2,TA11,264,6,177520

DEVICE SPEC : TT2,<7,20100,56701,120>,332,4,160010

DEVICE SPEC : /E

--

This question is asked if the answer to question 33 was "Y". No checks are made concerning the validity of the device specs. SYSGEN will insert "DEV=" before each specification. For further information concerning device specifications consult the RSX-11D or IAS System Generation Manual.

35. DO YOU WANT TASK CHECKPOINTING (Y OR N)? Y *

Answer this question with "N" if no checkpointing is desired. If you answer "Y" the next two questions will be asked.

36. CHECKPOINT DISK: /D *

This question is asked if the answer to question 35 was "Y". Respond with the type and unit number of the disk that will contain the checkpoint area. A response of "/D" will cause the default parameters, system disk and 50K size, to be used.

37. CHECKPOINT AREA SIZE: 100K

Enter the desired size of the checkpoint area. This question is not asked if the response to question 36 was "/D". The validity of this response is not checked.

38. DO YOU WANT TO GENERATE A UFTP SYSTEM (Y OR N)? Y *

Answer this question with "Y" if you desire the target disk to be configured for UFTP use and have the UFTP batch jobs generated. The next question will then be asked.

39. DO YOU WANT TO CONCATENATE THE UFTP BATCH JOBS (Y OR N)? Y *

---Note: Require for manufacturing use only.
By answering this question with "Y" the UFTP batch job SCRIPT.BIS will be created which consists of the individual batch jobs concatenated into a continuous running stream. A answer of "N" will create individual batch streams.

40. DO YOU WANT A CRASH MODULE IN THE EXECUTIVE (Y OR N)? Y

Answer this question with "Y" if you want a crash dump module included in the PSX-11D executive. The next two questions will also be asked.

41. ONE OF THE FOLLOWING DEVICES MAY BE USED AS
THE CRASH DUMP MEDIUM: DTA MTM MWN DKN
CRASH DUMP MEDIUM: DT3

Enter the device that will contain the crash dump information after a system crash. This response will be checked against the target configuration.

42. EXECUTIVE OBJECT MODULES ARE ON WHICH DEVICE? DK1

Enter the device that has the executive object modules. The modules must be located under UIC (11,15). (Object Disk #1 for PK distribution of PSX-11D V6H). The Bootstrap modules are under UIC (11,17) (object disk #3 for PK distribution of PSV-11D V6B).

43. IS THE CONFIGURATION ABOVE CORRECT (Y OR N)? Y

This question is a confidence check. If in rechecking the configuration questions that were just answered an error is found, answer this question with "N" and SYSGEN will be restarted. If this question is answered "Y" then the following message is printed.

***END OF SYSTEM GENERATION PHASE ***

SGNM BATCH JOB AND SYSRLD.COMD WILL BE CREATED.

Proceed to section D2420 to continue with the SYSGEN batch job creation procedures.

D2411 SYSGEN PHASE N ERROR MESSAGES

A MINIMUM OF 2 RK05'S ARE NECESSARY FOR AN RK05 BASED SYSTEM.

This message will be printed if the target disk specified is DR and you only configured one RK05. SYSGEN will be restarted.

THE TARGET DISK MUST BE CONFIGURED.

This message will be printed if there were no units specified for the disk type chosen for the target disk.

NO CRASH DUMP DEVICE CONFIGURED.

This message will be printed if no decatape, magtape (MT and MM) or RK disk is configured in the target system and a crash dump module is wanted in the exec.

FATAL ERROR ATTEMPTING TO OPEN A FILE

This message will be printed if an error occurred while trying to open any of the files that are created by SYSGEN. SYSGEN will abort.

FATAL ERROR ATTEMPTING TO CLOSE A FILE

This message will be printed if an error occurred while trying to close any file created by SYSGEN. SYSGEN will abort.

FATAL ERROR ATTEMPTING TO "PUT" A RECORD

This message will be printed when an error occurs while attempting to write to a file. SYSGEN will abort.

FATAL SYSTEM DIRECTIVE ERROR

This message will be printed if a system directive returned an error status. SYSGEN will abort.

D2422 SYSGEN BATCH JOB CREATION

This portion of system generation uses the information obtained from the previous questions and answers and will create 17 batch streams for the UFTP jobs if question 38 was answered with "Y" or one batch job if question 39 was answered with "N", one batch stream (SGNR) which will create the actual target disk and one file (SYSMLD.COM) which is used as the Phase 2 input file. For RSX-11F systems two supplementary files (IOTFCF.MAC and IOTHLI.COM) are also created if a UFTP system was configured.

The following dialogue may occur during this portion of system generation if this is a UFTP system.

UFTP DISTRIBUTION MEDIA (DFN,DPN,DRN,MMN,MTN)? DK1

This question will be asked only if the target disk is not an RM5 type disk. If the target disk was an RM5 then the UFTP distribution media is assumed to be DK1.

The batch job(s) are now created. The following printouts will occur if the batch jobs are NOT concatenated.

```

CREATION OF SGNR COMPLETE
CREATION OF JOB2 COMPLETE
CREATION OF JOB6 COMPLETE
CREATION OF JOB8 COMPLETE
CREATION OF JOB20 COMPLETE
CREATION OF JOB22 COMPLETE
CREATION OF JOB24 COMPLETE
CREATION OF JOB30 COMPLETE
CREATION OF JOB32 COMPLETE
CREATION OF JOB40 COMPLETE
CREATION OF JOB42 COMPLETE
CREATION OF JOB1 COMPLETE
CREATION OF JOB10 COMPLETE
CREATION OF JOB39 COMPLETE
CREATION OF JOB49 COMPLETE
CREATION OF JOB59 COMPLETE
ALL UFTP BATCH JOBS ARE CREATED

```

The following printout will occur if the batch jobs are concatenated.

UFTP BATCH JOB IS CREATED

Proceed to section D2421 for RSX-11F systems or section D2422 for IAS systems and follow the messages printed on the console for the target disk setup.

D2421 PSX-110 TARGET DISK SETUP PROCEDURES
.....

Now SYSGEN will print instructions for setting up the system for PHASE 1 of system generation. Sample printout for PSX-110 follows.

TYPE "CNTRL/C" AND PROCEED TO DISMOUNT THE DISK (IF ANY) ON THE TARGET UNIT AS FOLLOWS:
MCH>DMD DK1:

PLACE A SCRATCH DISK ON DK1: FOR USE AS THE TARGET DISK. MAKE IT READY AND WRITE ENABLED.

IF YOU DESIRE TO MAKE ANY CHANGES TO THE SYSGEN CONFIGURATION FILE (1,1)SGN.CMD, OR THE SYSGEN BUILD FILE (1,1)SYSBLD.CMD, YOU MAY EDIT THESE FILES NOW. IF NO MODIFICATIONS ARE DESIRED THEN TYPE "PAT SGN<ALTMODE>" TO CREATE THE TARGET DISK.

Some of the above instructions may not be printed as they are dependent on the configuration questions previously answered.

Proceed to section D2434 to perform the above instructions and continue with system generation PHASE 1.

D2422 IAS TARGET DISK SETUP PROCEDURES
.....

To be defined.

D2423 UFTP BATCH JOB CREATION ERROR MESSAGES
.....

MAGTAPE DISTRIBUTION IS VALID ONLY FOR RP02/03/04 DISKS

UFTP magtape distribution is valid only if the target disk is an RP03 or RP04.

COROL NOT SUPPORTED ON THIS SYSTEM

This message is printed by JOB40 and JOB42. It indicates that an RP02/03/04 or at least 2 RP05 disks are not configured for an PSX-110 system or there is less than 64K of core configured.

D2410 SYSGEN PHASE 1

System generation PHASE 1 creates the target disk and optionally (configuration dependent) runs SGM1 to create the PSX.SAV or IAS.SAV file. This is accomplished by the SGM1 batch stream. SGM1 performs the following functions:

- INSTALLS INV
- PINS BADBLOCK
- LOADS, INITs AND MOUNTS TARGET DISK
- CREATES UFD'S
- PIP'S NECESSARY FILES
- OPTIONALLY BUILDS THE EXFC WITH A CHASH MODIFIER
- OPTIONALLY TRANSFERS UFTP FILES
- OPTIONALLY PINS SGM1

To initiate the SGM1 batch stream go to section D2431 for PSX-110 systems or section D2432 for IAS systems and perform the steps as stated by the printouts in section D2421 and D2422.

D2431 PSX-110 PHASE 1

1. Type the following MCP command to dismount the disk on the target unit. This step is configuration dependent.

```

^C
--
MCR>DMD DK1;
-----
F11ACP -- DK1:DISMOUNT COMPLETE
MCR>

```

2. Place a scratch disk on the unit designated previously as the target disk. Make it ready and write enabled.
3. Type the following MCP> command to start the SGM1 batch stream.

```

MCR>BAT SGM1<ALTMODE>
-----

```

4. The SGND batch stream will now be executed. Many messages will be printed on the console as each command is executed. When the batch stream is completed a series of SMESAGE lines will be printed instructing the operator to perform certain tasks. The printout will be similar to the following if the target disk is NOT the system disk.

```

SGND>END OF PHASE 1
07:59:14 $MCP RFW...SG1
07:59:24 $MESSAGE SGND BATCH COMPLETE
07:59:25 $MESSAGE THE PSX-110 TARGET DISK IS NOW CREATED.
07:59:36 $!
07:59:36 $MESSAGE FOLLOWING THIS STOP THE CPU AND
07:59:38 $MESSAGE PLACE THE TARGET DISK IN UNIT 4
07:59:40 $MESSAGE AND BOOTSTRAP IT. PHASE 2 OF
07:59:42 $MESSAGE SYSTEM GENERATION WILL AUTOMATICALLY START
07:59:45 $EOJ

```

5. Now perform the actions as described in the SGND batch for printout. Proceed to section D244P to execute system generation PHASE 2.

D2432 IAS PHASE 1

To be defined.

D244W SYSGEN PHASE 2
.....

Phase 2 of system generation is installed during Phase 1. On bootstrap from a Phase 1 target disk, Phase 2 is activated and proceeds as follows.

1. Loads the teletype handler.
2. Issues a MOUNT command for the system device (SY).
3. Opens the file SY:(11,17)SYSBLD.COM, and if the open is successful, begins to process the file and print it on the console.

The file SYSBLD.COM is created by SYSGEN during Phase 1. If the user wishes to modify it, he should do so before requesting Phase 1.

When SYSBLD.COM has been processed, Phase 2 writes a bootstrap on block 2 of the system disk and terminates.

When Phase 2 is complete, it prints the following message on the console.

END OF SYSTEM GENERATION PHASE 2

At this point, perform the steps in Section D245W. This process properly saves the system for continued use.

D2450 FINAL SYSTEM DISK CONFIGURATION

.....

The following convention will be used throughout this section.

xxx = system disk, i.e., DAP, DPP, DMV,
 vv = checkpoint disk,
 n = disk unit number.

Proceed to section D2451 for PSX-11D systems or section D2452 for IAS systems to save the configured system.

D2451 PSX-11D FINAL CONFIGURATION

.....

1. Type "CTRL/C" and the following MCR commands.

^C
 ^^

MCR>HEL (1,1) -load into system

MCR>UNL vv -unload checkpoint disk handler if
, -checkpoint disk is not system disk

MCR>LOA vv -reload checkpoint disk handler if
, -checkpoint disk is not system disk

MCR>LOA LP -load line printer handler if available

MCR>LOA MO -load message output handler

MCR>PED xxxzSP -redirect spooler to system disk

You may now load any device handlers that have been previously installed by phase 2.

2. Enter one of the following redirects to redirect the CI console log device to another device.

seeConsole Log Output to Terminal:

This is the default redirect, therefore no redirect command is required.

seeConsole Log Output to Line Printer:

MCR>RED LP=CL -redirect to line printer if available

3. Enter the following commands ONLY if this is a "FTP system". These installs will permit the indicated MCR functions to be executable in a batch stream.

MCR>INS (11,1)MFT/TASKS...IDA

MCR>INS (11,1)MFT/TASKS...AHO

4. Enter the following MCP commands to save the system image.

```

MCR>DMO yyn:      -dismount the checkpoint disk
      -----      -if it is not the system disk

MCR>DMO xxx:      -dismount the system disk
      -----

MCR>FIX F11ACP     -fix the file system in core
      -----

MCR><ALTMODE>     -Altmode forces silent command mode.
      -----

SAV               -save the system image
---

MOU xxx:/OVR      -mount the system disk
-----

MOU yyn:/OVR      -mount the checkpoint disk
-----
                  -if it is not the system disk

TIM
---

TIM ^C           -prompt the user for the time
-----

```

The system will now printout the following to the console.
Type the time and date where indicated.

```

124K (WORD) PSX-11D VPM6B

MCR>MOU xxx:/OVR
MOUNT-**VOLUME INFORMATION**          (SAMPLE ONLY)
      DEVICE =xxx
      CLASS  =FILE 11
      LABEL  =PSX11DSYS
      UIC    =[1,1]
      ACCESS =[RWED,RWED,PWED,RWED]
      CHARAC =[]

MCR>MOU yyn:
MOUNT-**VOLUME INFORMATION**
      DEVICE =yyn
      CLASS  =FILE 11
      LABEL  =
      UIC    =[1,1]
      ACCFSS =[RWED,RWED,PWED,RWED]
      CHARAC =[]

MCR>TIM
12/23/75 07:45:47
MCR>TIM 09:45:00 12/23/75
-----

```

5. The following commands are required for an RK UFTP system only.

Place the UFTP distribution disk in drive 1. Set it to run and write enabled. Type the following command to mount the UFTP disk:

```
MCR>MOUNT DF1:/OVR
-----
MCR>PTP @DF1:[200,201]PHAS13
```

6. The RSX-11D system program disk is now configured. Proceed to section D1000, step 6 for UFTP system operating procedures.

D2452 IAS FINAL CONFIGURATION

To be defined.

D2460 SYSGEN PRINTOUT EXAMPLES

72K (WORD) RSX-11D V006H

SAV -- PARTITION GEN EXPANDED BY 768*32 (DEC) WORDS

MCR>MOUNT DK:

MOUNT--VOLUME INFORMATION--

```
DEVICE =DK0
CLASS =FILE 11
LABEL =
UIC =1,1
ACCESS =[RWED,RWED,RWED,RWED]
CHARAC =[]
```

MCR>; SYSTEM CARTRIDGE

MCR>TIM 12/30/75 13:33:00

MCR>MOUNT DF1:/OVR

MOUNT--VOLUME INFORMATION--

```
DEVICE =DK1
CLASS =FILE 11
LABEL =UFTPSY
UIC =1,1
ACCESS =[RWED,RWED,RWED,RWED]
CHARAC =[]
```

MCR>INS DF1:[200,201]SYSGEN

MCR>RUN SYSGEN

RSX-11D/IAS SYSGEN V3-00 PHASE 1

THE VALIDITY OF SOME ANSWERS ARE NOT CHECKED.
AN INCORRECT CHARACTER OR LINE RESPONSE MAY
BE DELETED BY TYPING "PURROUT" OR "CONTROL U"
RESPECTIVELY.

A RESPONSE OF "CARRIAGE RETURN" WILL DEFAULT A
YES/NO QUERY TO "N" AND A NUMERIC QUERY TO "0".
ALL OTHER QUERIES WILL BE DEFAULTED TO NULL.
A RESPONSE OF "ALTMODE" OR "ESCAPE" WILL CAUSE
THE LONG FORM OF THE QUERY TO BE PRINTED.

YOU MAY TYPE "RE" OR "AB" AS A RESPONSE TO A
QUERY TO RESTART OR ABORT SYSGEN.

DATE: 30-DEC-75

SYSTEM NAME: 8123

LONG DIALOG RESULTS IN THE DISPLAY OF EXPLANATORY TEXT
PRECEDING MOST QUERIES. SHORT DIALOG OMITTS THE TEXT.
LONG DIALOG IS AVAILABLE ON A PER QUERY BASIS BY ENTERING
ESCAPE OR ALTMODE FOLLOWING THE DISPLAY OF THE QUERY.

DO YOU WANT LONG FORM OF DIALOG (Y OR N)? N

TARGET DISK (DKN,DPN,DRN)? DK1

CPUS (40,45,70)? 40

WHAT IS THE MEMORY SIZE (IN 1K BLOCKS)? 72

IS THE FIS OPTION AVAILABLE (Y OR N)? Y

IS A KW11-P CLOCK AVAILABLE (Y OR N)? Y

SHOULD IT RUN AT 6.0HZ (Y OR N)? Y

HOW MANY RK05 DISK DRIVES ARE AVAILABLE (0-R)? 2

DO YOU WANT TO USE THE OVERLAPPED SEEK HANDLER (Y OR N)? Y

HOW MANY RP04 DISK DRIVES ARE AVAILABLE (0-R)? 0

HOW MANY RP02/RP03 DISK DRIVES ARE AVAILABLE (0-R)? 0

HOW MANY RS02/RS03 DISK DRIVES ARE AVAILABLE (0-R)? 1

HOW MANY RF11 DISK PLATTERS ARE AVAILABLE (0-R)? 1

HOW MANY TU56 DECTAPE DRIVES ARE AVAILABLE (0-R)? 2

HOW MANY TU12 DECTAPE DRIVES ARE AVAILABLE (0-R)? 3

HOW MANY TUI6 DECTAPE DRIVES ARE AVAILABLE (Y OR N)? Y

IS A TAI1/TU6M CASSETTE SYSTEM AVAILABLE (Y OR N)? Y

IS A LINE PRINTER AVAILABLE (Y OR N)? Y

DOES IT HAVE 132 COLUMNS (Y OR N)? Y

IS THE PRINTER A CENTRONICS LP45 (Y OR N)? N

IS THE CONTROLLER AN LS11 (Y OR N)? N

IS A CARD READER AVAILABLE (Y OR N)? N

IS A PAPER TAPE READER AND/OR PUNCH AVAILABLE (Y OR N)? N

CONSOLE (KSR33, KSR35, VT05, VT50, LA36, LA305, LA30P)? LA30P

ARE ANY NON-STANDARD DEVICES PRESENT (Y OR N)? N

DO YOU WANT TASK CHECKPOINTING (Y OR N)? Y

CHECKPOINT DISK : DEF

CHECKPOINT AREA SIZE : 50K

DO YOU WANT TO GENERATE A UETP SYSTEM (Y OR N)? Y

DO YOU WANT TO CONCATENATE THE UETP PATCH JOBS (Y OR N)? Y

DO YOU WANT A CRASH MODULE IN THE EXECUTIVE (Y OR N)? N

IS THE CONFIGURATION ABOVE CORRECT (Y OR N)? Y

*** END OF SYSTEM GENERATION PHASE 0 ***

SYSBLD.CMD, SGW AND UETP PATCH JOBS WILL BE CREATED.

CREATION OF SGW AND SYSBLD.CMD COMPLETE
UETP PATCH JOB IS CREATED

TYPE "CNTRL/C" AND PROCEED TO DISMOUNT THE
UETP DISTRIBUTION DISK AS FOLLOWS:

MCR>DMO DK1:UETPSY

PLACE A SCRATCH DISK ON DK1: FOR
USE AS THE TARGET DISK. MAKE IT READY
AND WRITE ENABLED.

IF YOU DESIRE TO MAKE ANY CHANGES TO THE
 SYSGEN CONFIGURATION FILE (1,1)SGN.CMD, OR
 THE SYSGEN BUILD FILE (1,1)SYSBLD.CMD, YOU MAY
 EDIT THESE FILES NOW. IF NO MODIFICATIONS ARE
 DESIRED THEN TYPE "BAT SGN<ALTMODE>" TO CREATE
 THE TARGET DISK.

MCR>DMD DK1:

 F11ACP -- DK1: ** DISMOUNT COMPLETE **
 MCR>BAT SGNAS

JOB SGN PSX-11D BATCH V026H 13:41:23 31-DEC-75 PAGE 1

13:41:27 \$JOB/NAME=SGN/LIMIT=200/MCP
 13:41:29 \$! SGN CREATES AN PSX-11D TARGET DISK WITH
 13:41:32 \$! ALL NECESSARY UFD'S. THE SYSBLD.CMD FILE
 13:41:34 \$! IS CREATED AND TRANSFERED TO (11,17) AND
 13:41:37 \$! A CRASH MODULE MAY BE BUILT INTO THE EXEC.
 13:41:40 \$! ALL NECESSARY FILES ARE TRANSFERED TO THE
 13:41:42 \$! TARGET DISK AND SGN1 MAY BE RUN TO CREATE
 13:41:45 \$! THE PSX.SAV FILE ON THE TARGET DISK.
 13:41:48 SMCR RM SYSGEN
 13:41:49 SMCR INS (11,1)MFT/TASK=...10A
 13:41:54 SMCR INS (11,1)INV
 13:41:58 SMCR LOA MO
 13:42:00 SMCR BAD DK1:

RAD -- TOTAL NO. OF RAD BLOCKS = 0

13:43:13 SMCR INI DK1:UFTPSYSDSK/RAD=[AUTO]
 INI -- CHECKING DK1:

13:43:19 SMCR MOU DK1:/OVR
 MOUNT--VOLUME INFORMATION--

DEVICE =DK1
 CLASS =FILE 11
 LABEL =UFTPSYSDSK
 UIC =(1,1)
 ACCESS =(RWED,RWED,RWED,RWED)
 CHARAC =[]

13:43:31 SMCR UFD DK1:(1,1)
 13:43:34 SMCR UFD DK1:(1,2)
 13:43:36 SMCR UFD DK1:(1,3)
 13:43:39 SMCR UFD DK1:(1,4)/PRO=(RWED,RWED,RWED,RWED)
 13:43:42 SMCR UFD DK1:(1,5)/PRO=(RWED,RWED,R,P)
 13:43:46 SMCR UFD DK1:(1,6)/PRO=(RWED,RWED,RWED,RWED)
 13:43:49 SMCR UFD DK1:(1,27)
 13:43:52 SMCR UFD DK1:(11,1)
 13:43:54 SMCR UFD DK1:(11,17)
 13:43:57 SMCR UFD DK1:11,22)
 13:43:59 SMCR UFD DK1:(11,24)
 13:44:02 SMCR UFD DK1:(11,27)
 13:14:05 SMCR UFD DK1:(11,42)

```

13:44:08 SMCR UFD DK1:(200,200)/PROG(RAFD,MWFD,PAFD,PAED)
13:44:12 SMCR UFD DK1:(200,201)
13:44:15 SMCR PIP DK1:(1,1)=(1,1)STR,TSK,HIS,SMI
13:44:47 SMCR PIP DK1:(1,1)=(1,1)SYSLIB,OLH,IOFCF,MAC,IOTMLD,CMT
13:45:00 SMCR PIP DK1:(1,2)=(1,2)FTNCOM,MSG,FTNCOM,MSG,FTNOTS,MSG
13:45:10 SMCR PIP DK1:(1,2)=(1,2)PIP,MSG,FMT,MSG,DIOSYM,MSG
13:45:20 SMCR PIP DK1:(11,17)=(11,17)TSP,CMD,STH
13:46:25 SMCR PIP DK1:(11,1)=(11,1)PIF,MOO,PURMAC,ACCPPT,MCH
13:46:57 SMCR PIP DK1:(11,1)=(11,1)PPT,PHX,UIF,SPP,SPP2
13:47:26 SMCR PIP DK1:(11,1)=(11,1)OPR,SPL,FAT,HFP,DMP
13:47:50 SMCR PIP DK1:(11,1)=(11,1)CRP,CRNP,DE,DK,DKOVL
13:48:27 SMCR PIP DK1:(11,1)=(11,1)DP,HD,DM,DS,TKR
13:48:59 SMCR PIP DK1:(11,1)=(11,1)F11MSG,INI,MCU,DMO,UFD
13:49:29 SMCR PIP DK1:(11,1)=(11,1)HIGHCP,MTAACP,INV,SAV
13:49:56 SMCR PIP DK1:(11,1)=(11,1)ACCOFF,ACCAHT,PHLOG,PROFF
13:50:21 SMCR PIP DK1:(11,1)=(11,1)PSF,FOR,PP,PR,CT,SXF
13:50:59 SMCR PIP DK1:(11,1)=(11,1)DT,LP,LPCNT,LS,TUIW
13:51:29 SMCR PIP DK1:(11,1)=(11,1)TT16,TT10,MO,RAD,ACCILOG
13:52:00 SMCR PIP DK1:(11,1)=(11,1)ACT,CHE,RVF,FDI,FLX
13:52:33 SMCR PIP DK1:(11,1)=(11,1)SYS,MFT,MCRFRP,HEL
13:52:59 SMCR PIP DK1:(11,1)=(11,1)MEM,LIN,DFE,DEMO,PWD
13:53:31 SMCR PIP DK1:(11,1)=(11,1)RFA,PF,SET,TIM,LAP
13:54:01 SMCR PIP DK1:(11,1)=(11,1)TKTN,UNL,INS,RFM
13:54:27 SMCR PIP DK1:(11,1)=(11,1)CON,WHO,POOL,MIN,CDA
13:55:00 SMCR PIP DK1:(11,17)SYSBLD,CMD/NV=(1,1)SYSBLD,CMD
13:55:06 SMCR INS (11,1)SGN1/TASK=...SG1

```

JOB SGN0 RSX-11D HATCH V006P 13:55:10 30-DEC-75 PAGE 2

```

13:55:10 SMCR SG1
13:55:15 R(1,1)SGN,CMD
TARGET=DK1:(11,17)PSX.SAV
/
PDP11=40,72K,,<60,2,1>
/
DEV=DK0,PK05,220,5,177400
DEV=DK1,PK05,220,5,177400
DEV=DF0,PF1,204,5,177460
DEV=DT0,DT11,214,6,177340
DEV=DT1,DT11,214,6,177340
DEV=MT0,TU10,224,5,172520
DEV=MT1,TU10,224,5,172520
DEV=MT2,TU10,224,5,172520
DEV=LPA,LP11R,200,4,177514
DEV=TPA,LA3AF,060,4,177560
DEV=MO,.,.,.
DEV=SP,.,.,.
DEV=BP,.,.,.
/
SCOM=,330,64
PAR=SYDISK,,50,11
PAR=MCP,,41,S
PAR=TTY,,130,U
PAR=GEN,,.,S

```

```

/
DPR=GEN
FOOL=17
SY=DKA
CKPNT=DEFU,50K
/
INS=SYDISK,(11,1)DPOVL
INS=GEN,(11,17)SGN2,(11,1)MQU,(11,17)INZ/UIC=(1,1),(11,1)MIGFCP
INS=TTY,(11,1)TT16
//
SGN>END OF PHASE 1
13:56:48  SMCR  DEF  ... SGI
13:56:50  SMCR  ROU  DK1:(11,17)RSX.SAV/WH
13:56:53  SMESAGE  SGN:  BATCH  JOB  COMPLETED
13:56:55  SMESAGE  THE  PSX-11D  TARGET  DISK  IS  60%  CREATED.
13:56:58  SMESAGE  !!!
13:57:00  SMESAGE  FOLLOWING  THIS  STOP  THE  CPU  AND
13:57:02  SMESAGE  PLACE  THE  TARGET  DISK  IN  UNIT  0
13:57:05  SMESAGE  AND  ROOTSTRAP  IT.  PHASE  2  OF
13:57:08  SMESAGE  SYSTEM  GENERATION  WILL  AUTOMATICALLY  START.
13:57:11  SEUJ
*** SYSTEM GENERATION PHASE 2 ***
YOU  DK0:/OVR
MOUNT==VOLUME INFORMATION==
        DEVICE  =DK0
        CLASS   =FILE 11
        LABEL   =UETPSYSDSK
        UIC     =(1,1)
        ACCESS  =(RWED,RWED,RWED,RWED)
        CHARAC  =()
INZ (P11,1)TKTN
INZ (11,1)MCR/UIC=(1,1)
INZ (11,1)MFT
INZ (11,1)MCRERR
INZ SYSRES/LI/ACC=RO/UIC=(1,1)
*DELAY
INZ (11,1)INS
*DELAY
*DELAY
INS (11,1)ACCLG
INS (11,1)ACCPPT
INS (11,1)ACT
INS (11,1)HAT/POOL=250
INS (11,1)BPR
INS (11,1)ROO
INS (11,1)BYE
INS (11,1)CDA
INS (11,1)CRF
INS (11,1)DEMO
INS (11,1)DMO
INS (11,1)DMP
INS (11,1)FDI/PRI=60
INS (11,1)ERRLOG
INS (11,1)FLX/PRI=55
INS (11,1)FOP

```

```

INS (11,1)F11MSG
INS (11,1)HML
INS (11,1)INI
INS (11,1)LRP
INS (11,1)LUN
INS (11,1)OPE
INS (11,1)OPP
INS (11,1)PIP/PP1=55
INS (11,1)POOL
INS (11,1)PUPMAC
INS (11,1)PWD
INS (11,1)QUF
INS (11,1)REA
INS (11,1)RED
INS (11,1)REM
INS (11,1)RUN
INS (11,1)SAV
INS (11,1)SET
INS (11,1)SPP
INS (11,1)SPR2/INC=46A
INS (11,1)SYS
INS (11,1)TIM
INS (11,1)TKA
INS (11,1)UFD
INS (11,1)UNL
INS (11,1)WHO
INS (11,1)DF
*DELAY
INS (11,1)LP
*DELAY
INS (11,1)DT
*DELAY
INS (11,1)TU10
*DELAY
INS (11,1)MO
*DELAY
*DELAY
PFM ... INZ
*DELAY
LOA LP
SFT /SP=LP:
UNL LP
LOA DF
*DELAY
INI DF?:
*DELAY
*DELAY
MOU DF?:
MO'INT=**VOLUME INFORMATION**
      DEVICE =DF0
      CLASS  =FILE 11
      LABEL  =
      UIC    = (1,1)
      ACCESS = (PWFD,PWFD,P6ED,P6FD)
      CHARAC = ( )

```

```

*DELAY
*DELAY
*** END OF SYSTEM GENERATION PHASE 2 ***
MCR>MFILE (1,1)
*****
MCR>INL DF
*****
MCR>LOA DF
*****
MCR>LOA LP
*****
MCR>LOA MO
*****
MCR>RED DM:SP:
*****
MCR>LOA DT
*****
MCR>LOA MT
*****
MCR>RFD LP=CL
*****
MCR>INS (11,1)MFT/TASK=...LOA
*****
MCR>INS (11,1)MFT/TASK=...ABO
*****
MCR>DMO DF:
*****
F11ACP -- DF: ** DISMOUNT COMPLETE **
MCR>DMO DK:
*****
F11ACP -- DK: ** DISMOUNT COMPLETE **
MCR>FIX F11ACP
*****
MCR>S
-
MCR>SAV

```

72K (WORD) PSX-110 VM6H

```

MCR>MOUNT DM:/OVP
MOUNT--VOLUME INFORMATION--
  DEVICE  #DF
  CLASS   #FILE 11
  LABEL   #UFTPSYSDSK
  UIC     #((1,1))
  ACCESS  #([RWF,RF,RF,RF])
  CHARAC  #[]
MCR>MOUNT DF:/OVP
MOUNT--VOLUME INFORMATION--
  DEVICE  #DF
  CLASS   #FILE 11

```

LABEL #
UIC # (1,1)
ACCESS # (RWFD, RWFD, RWFD, RWFD)
CHARAC # (1)

MCP>TIM
12/30/75 14:04:37
MCP>TIM 12/30/75 14:07:00

MCP>MOUNT DK1: /OVR

MOUNT--VOLUME INFORMATION--

DEVICE #DK1
CLASS #FILE 11
LABEL #HETPSY
UIC # (1,1)
ACCESS # (RWFD, RWFD, RWFD, RWFD)
CHARAC # (1)

MCP>PIP #DK1: (200,201)PHASE3

MCP>

D3.10 UFTP SYSTEM OPERATING PROCEDURES

1. Place the UFTP system disk in the appropriate drive.
2. Set it to run and write enabled.
3. Boot in the UFTP system disk as described in section D2.10.

The system will printout the following on the console. Enter time and date where indicated.

174K PSK-11D V006A

```
MCP>MOUNT DK0:/OVR
MOUNT--VOLUME INFORMATION--
  DEVICE  =DK0
  CLASS   =FILE 11
  LABEL   =UFTPSYSDSK
  UIC     =([,])
  ACCESS  =([RWED,RWED,RWED,RWED])
  CHARAC  =()
```

```
MCP>TIM
 07/23/75 07:45:47
MCP>TIM 09:30:00 07/23/75
-----
```

4. If the UFTP system is an RP, place the distribution disk in the appropriate drive. Set it to run and write enabled. Enter the following MCP command.

```
MCP>MOUNT DK1:/OVR
-----
```

5. To insure that the system disk is properly initialize, enter the following MCP command. This command will remove any old files generated from a previous run.

```
MCP>PIP @DK1:[200,201]CIPRAT---for a RK system
-----
```

```
MCP>PIP @([200,201]CLRRAT---for a RP system
-----
```

6. Physically mount scratch media to all devices to be tested. All devices should be ready and write enabled.

7. Enter the following MFP command to activate the error log routine.

```
MCP>MIN FRMLOG <ALTMODE>
-----
```

Input minimum number of errors capable of being logged in a 5 second period "carriage return." This value should not exceed 5.

If error logging not wanted input "Control Z." 5 second error rate = 3.

```
"C
---
```

You are now ready to run the UFTP batch jobs. If you specified the concatenated batch stream during SYSGEN, proceed to step 9.

8. PROCEDURE FOR RUNNING SEPARATE BATCH JOBS

```
-----
--Not for manufacturing use
Enter the following MCR commands:
```

```
MCR>BAT JOB1<ALTMODE>
-----
```

At the completion of JOB1, type in "CNTPL/C" and enter the next MCR command.

```
"C
---
```

```
MCP>BAT JOB2<ALTMODE>
-----
```

At the completion of JOB2, type in "CNTPL/C" and enter the next MCR command.

```
"C
---
```

```
MCP>BAT JOB6<ALTMODE>
-----
```

At the completion of JOB6, type in "CNTRL/C" and enter the next MCP command.

```
^C
==
MCP>BAT JOB6<ALTMODE>
```

At the completion of JOB8, type in "CNTRL/C" and enter the next MCP command.

```
^C
==
MCP>BAT JOB8<ALTMODE>
.....
```

At the completion of JOB10, type in "CNTRL/C" and enter the next MCP command.

```
^C
==
MCP>BAT JOB10<ALTMODE>
.....
```

At the completion of JOB20, type in "CNTRL/C" and enter the next MCP command.

```
^C
==
MCP>BAT JOB20<ALTMODE>
.....
```

JOB22 is terminated via JOB24. When you desire to terminate this job, type in "CNTRL/C" and enter the next MCP command.

```
^C
==
MCP>BAT JOB24<ALTMODE>
.....
```

At the completion of JOB24, type in "CNTRL/C" and enter the next MCP command.

```
^C
==
MCP>BAT JOB24<ALTMODE>
.....
```

At the completion of JOB30, type in "CNTRL/C" and enter the next MCP command.

```
^C
==
MCP>BAT JOB30<ALTMODE>
.....
```

At the completion of JOB32, type in "CNTPL/C" and enter the next MCR command.

"C
--
MCR>BAT JOB39<ALTMODE>

At the completion of JOB39, type in "CNTPL/C" and enter the next MCR command.

"C
--
MCR>BAT JOB42<ALTMODE>

At the completion of JOB42, type in "CNTPL/C" and enter the next MCR command.

"C
--
MCR>BAT JOB49<ALTMODE>

At the completion of JOB49, type in "CNTPL/C".

"C
--
MCR>BAT SCRIPT<ALTMODE>

9. PROCEDURE FOR RUNNING THE CONCATENATED BATCH STREAM

Enter the following MCR command:

MCR>BAT SCRIPT<ALTMODE>

The UETP will now run all batch jobs without operator intervention. When the batch stream has completed a message will be written to the console:

\$MESSAGE
\$MESSAGE ***
\$MESSAGE *** SCRIPT,BIS BATCH STREAM COMPLETED ***
\$MESSAGE ***
\$MESSAGE

14. This concludes the batch job operation of the IFTP system.

D3050 BATCH ABORT PROCEDURES

If for any reason any of the above batch jobs should halt or if the user desires to terminate any of the jobs, the following procedure must be followed.

HALT THE CPU

D3100 BATCH RE-START PROCEDURES

1. Halt the CPU.
2. Repeat steps 3 through 7 of section D3000.
3. If the line printer has been configured with this system, enter the following MCR command.
 MCR>OPP LP:/RE - Re-cycle the line printer

4. When the MCR prompt returns, continue normal batch operation as described in section D3000, step 8 or step 9.

D3154 UFTP BATCH JOB DESCRIPTIONS

D3155 JOP1

This job installs the error log preanalyzer and analyzer, runs them and prints and deletes the log.

D3160 JOB2

This job sets up the test system to be able to run the system general I/O tests. The configuration files will be built along with the I/O test task. The device handlers will be logically mounted. Scratch media is assumed to be physically mounted and write enabled.

D3170 JOB3

This job runs the general I/O tests. All volumes are assumed to be mounted.

D3175 JOB4

This job does the cleanup after the system I/O tests. All volumes are dismounted with the exception of the system disk.

D31A4 JOB14

This job installs the error log preanalyzer and analyzer, runs them and prints and deletes the log.

D31A5 JOB20

This job loads the diagnostic device handlers. All devices must have scratch media physically mounted with write enabled. They will be initialized and mounted. The diagnostic tasks will be built and installed.

D31A8 JOB22

This job runs the diagnostic programs for each device on the system. All device volumes should be scratch volumes and write enabled.

D32A4 JOB24

This job aborts and removes the diagnostic tasks for all devices on the system, dismounts the volumes, unloads the diagnostic handlers and loads the system handlers.

D3250 JOB34

This job runs the FORTRAN I/O tests. These tests are run as separated batch jobs linked together. Refer to section D6000 for description of each job.

D3255 JOB32

This job runs the FORTRAN user simulation. These are run as separate batch jobs linked together. Refer to section D6700 for description of each job.

D3264 JOB39

This job installs the error log preanalyzer and analyzer, runs them and prints and deletes the log.

D3265 JOB40

This job runs the COBOL I/O tests. These tests are run as separate batch jobs linked together. Refer to section D7000 for description of each job.

NOTE:

1. COBOL is not supported for a system less than 64K memory.
2. The following TRK diagnostic messages will be printed to the console while running JOB40.

TRK -- *DIAG=SEGMENT MSGR HAS NO P-SECTION

TRK -- *DIAG=SEGMENT RFCGR HAS NO P-SECTION

These are valid messages and have no effect on the COBOL operation.

D3274 JOB42

This job runs the COROL user simulation. Each program is executed as a separate batch job linked together. Refer to section D7100 for description of each job.

NOTE:

1. COROL is not supported for a system less than 64K memory.
2. The following TKB diagnostic messages will be printed on the console while running JOB42.

TKA -- @DIAG@-SEGMENT MSGP HAS PD P-SECTION

TKB -- @DIAG@-SEGMENT PFCSGH HAS PD P-SECTION

These are valid messages and have no effect on the COROL operation.

D3275 JOB49

This job installs the error log preanalyzer and analyzer, runs them and prints and deletes the log.

DSWAP USER MODE DIAGNOSTICS
.....

MSX-110 provides diagnostic tasks that can be run by the user to test the hardware reliability of disks, magnetic tape drives, and PCtape drives. If the user suspects a hardware malfunction on one of these devices, the appropriate diagnostic task can be installed and run. Results of the testing are printed on the terminal.

Because the diagnostics are tasks under MSX-110, they can execute concurrently with other system tasks. Normal operations need not be disturbed.

Two types of diagnostics are included in the system: diagnostic programs and data reliability tests. Each device for which online diagnostics are available has an associated diagnostic program and a data reliability test.

The data reliability tests perform a subset of the functions of the diagnostic programs. They do not provide the capability to select which of the subtests are to be performed as can be done in the diagnostic programs.

Unlike most system tasks which are distributed in task-image form, the diagnostic programs and data reliability tests must be task built before they can be installed and run. Task building is required to allow specification of variable information such as the specific unit to be tested.

Special device handlers are used in conjunction with the diagnostics. These device handlers pass error information to the diagnostics. The diagnostics interpret the information and print appropriate error messages.

D5100 DEVICES SUPPORTED BY DIAGNOSTICS

The table in section D5120 lists the devices for which diagnostic programs and data reliability tests are provided and supplies the following additional information.

1. System mnemonic, which is identical to the device handler name.
2. Diagnostic handler name.
3. Diagnostic mnemonic, which is used in ICA commands.
4. Diagnostic test name.
5. Data reliability test name.

This information is referred to throughout Section D5120.

D5110 NAMING CONVENTIONS FOR DIAGNOSTICS

The names of the diagnostic programs and data reliability tests are listed in the last two columns of the table in Section D5120.

The conventions described below are used in naming the diagnostics. Diagnostics have 6-character names in the following format.

xxxxyz

xxxx designates the type of unit that the diagnostic is to test, e.g., RP03, TU10, RS04, and TU16.

y indicates whether the test is a diagnostic program or a data reliability test by using one of the following letters.

S indicates a single unit diagnostic program.

D indicates a data reliability test.

z designates the number of a particular test for that class of diagnostics. If there is only one test in the class the number is omitted. For example, RP03S2 the second diagnostic program in the RP03S class.

DS124 DEVICES SUPPORTED BY DIAGNOSTIC HANDLERS

device	system mnemonic (handler name)	diagnostic handler name	diagnostic mnemonic	diagnostic test name	data reliability test name
RP23	DK	DPD	DA	RP23S	RP23D
RP25	DA	DPD	DA	RP25S	RP25D
RP23	DP	DPD	DP	RP23S	RP23D
RP24	RD	RPD	RD	RP24S	RP24D
RJS13/24	RD (if RJP24 is on same WH11			RS23S	RS23D
	DS	PSD	PS	RS24S	RS24D
TU56	DT	DTD	DT	TU56S	TU56D
TU12	MT	TU12D	MT	TU12S10	TU12D
TU16	MM	TU16D	MM	TU16S10	TU16D
PS11	DF	DFD	DF	PS11S	PS11D

• positioning test

D6100 FORTRAN I/O EXERCISES

D6101 CPU1

1. TO TEST

This program tests the speed of the CPU in performing scientific calculations.

2. GENERAL DESCRIPTION

The program consists of ten modules, each of which exercises a group of language features. Each module is placed in a loop and the number of times it is executed is adjusted to mimic as closely as possible the available statistical profiles of language feature usage. All the loops have been arranged so that it is not possible for an optimizing compiler to remove a significant amount of code from them. Features exercised include simple variable and array addressing, fixed and floating-point arithmetic, subroutine calls and parameter passing and standard mathematical functions.

Floating-point calculations are performed with the default precision of the implementation.

D6102 CPU2

1. TO TEST

Processors - can test large amount of core store - uses fixed point working only.

2. GENERAL DESCRIPTION

The program is designed to test binary systems using 2's complement for negative numbers. The program uses two dimensional array of any size and takes into account the word length. Firstly all 1's are written into each location then all zeros and 1 bit in each position of each word in turn. This is read back and checked. In the second pass of the program all 0's are written followed by all ones and one zero in each position of each word.

D6103 CPU3

1. TO TEST

Processors using a wide range of different floating point mixers

2. GENERAL DESCRIPTION

The program calculates binomial expansion of

$(q + p)^n = 1$ for values of n from 1 to 77 or more, for each value of p from $q = 0.1$ to 0.9 in variable steps.

Successive sums of expansions are multiplied together and printed out at the end. Result should be approximately 1. Also last sums of expansion for each value of n are added and the result should be approximately 77. Note that if a bit is dropped in one calculation of an expansion this will be carried forward to the final product.

$$(q + p)^n = q^n + nq^{n-1}p + \frac{n!}{(n-2)!2!} q^{n-2} p^2 + \frac{n!}{(n-3)!3!} q^{n-3} p^3 + \dots + p^n = 1$$

eg for $n = 1, = q + p$

$$n = 2, = q^2 + 2qp + p^2$$

$$n = 3, = q^3 + 3q^2p + 3qp^2 + p^3$$

$$n = 4, = q^4 + 4q^3p + 6q^2p^2 + 4qp^3 + p^4$$

D61.4 CP14

1. TO TEST

Processors - especially those which take assumptions on the result of a branch.

2. GENERAL DESCRIPTION

The program consists of all branching instructions. The number of different branching instructions generated may be small but for processors which take assumptions on the result of a branch some assumptions will be correct others incorrect.

D61.5 CP15

1. TO TEST

Processors double precision working, and check that various functions are provided.

2. GENERAL DESCRIPTION

Produces 1 from various formulae, e.g., $\sin(2)A + \cos(2)A = 1$ for many values of A, multiplying subsequent results together to produce printed answer approximately equal to 1. Note that if a bit is dropped in one calculation this will be carried forward to final answer. For checking purposes a sumcheck subroutine has been incorporated to check a complete word by shifting and adding. The sumcheck is printed along with the answer.

D6106 DISK1

1. TO TEST

Two sections of a disk or 2 separate disks.

2. DESCRIPTION

A number of words are written serially to a specified number of addresses on 2 disks files. The words are produced by random number generator. Firstly, the 2 files are read serially and compared. A number of random addresses within the appropriate range are produced and again the 2 disks are compared. The first word of each block written contains the address which is also checked with the address generated. A print is given of any discrepancies found and 4 re-attempts made to read any failing block. Also a print-out is given of random addresses generated.

The above applies to a system with sequential access methods. With random access methods the program will work except the write and first read will not be serial, and slight modifications may be required to the address pointing.

Certain changes may be required to the writing, reading and indexing on other systems and certain systems may not have indexing facilities available in Fortran.

D6107 DISK2

1. TO TEST

Disks - binary file.

2. DESCRIPTION

A variable number of fixed length blocks are written to a file with different binary numbers on each block. The file is then read a variable number of times and compared with the regenerated data. A print out is given of any errors.

06240 FORTRAN USER SIMULATION

06241 FUSMDC

PURPOSE

1. To represent a standard statistics package run.
2. To test the random number subroutine RAN(I,J).

PROGRAM DESCRIPTION

1. Initialize run time parameters.
2. Input run time limit via operator's console or batch stream.
3. Summations are made after random numbers are generated. Deviations are then computed for both distributions, and both are checked for values greater than 3.0 standard deviations away from the mean. Any such values are discarded, and unless more than 3 per cent of the values were thus discarded, the means and standard deviations are recomputed using only the good data points. The coefficient of correlation between the two distributions is then computed and the various values printed.

DATA INPUT

Distribution data is generated internally via random number generator.

PROGRAM RUN TIME INFORMATION

Run time checks are made throughout this program. When run time is exceeded, job terminates. Data output occurs on pass 1 only.

ERROR HANDLING

1. Correlation errors are reported to console for all program passes.
2. Gross random number generation errors will be reported to console and job will then be aborted.

DE202 FUSSSD

PURPOSE

1. To represent a user who is structuring collected data from disk for future data reduction operations.
2. To utilize direct disk access I/O commands for all to data transfer operations.
3. To equalize disk and CPU access time for maximum usage.

PROGRAM DESCRIPTION

1. Generate random data from the FORTRAN random number generator (Intrinsic Function, RAN).
2. Store each value in memory (ARRAY P) for disk data verification.
3. Store each value on disk (XPAN).
4. Verify data generated on disk (500 values).
5. Sort data from disk to array "ASA" in sequential ascending order.
6. Verify that the number of values in array "ASA" equal the number of random values in disk file "XPAN".
7. Verify that array "ASA" is in sequential ascending order.
8. Sort data from disk to array "DSA" in sequential descending order.
9. Verify that the number of values in array "DSA" equal the number of random values in disk file "XPAN".
10. Verify that array "DSA" is in sequential descending order.

11. Search disk file for maximum and minimum value and calculate cell size and bounds for a three bin sort.
12. Sort data from disk to the appropriate cell based on the calculated bounds.
13. Verify that the total contents of all three cells equal the contents of disk file "XFA4".
14. Verify that the contents of each cell fall into the proper bounds.
15. Output random data file and sort files to the batch stream log file on pass 1 only.

PROGRAM RUN TIME INFORMATION

Program run time value is entered via operator console or batch stream at the beginning of the program. This value is only entered once during the entire program execution time.

Run time checks are made throughout the program. When the run time exceeds the entered run time value, the program terminates. The program continues to loop until the above condition is satisfied.

PROGRAM ERROR HANDLING

All data and sort verification errors are reported to the log file for all passes.

Program will restart after reporting error, unless if run time has been exceeded.

D6273 FUSLS1

PURPOSE

1. To represent a typical engineering design application.
2. To exercise logical operations utilizing the FORTRAN library function subroutines INTR(M,N), IAND(M,N), NOT(M).

MODEL DESCRIPTION

The TTL/MSI 74142 look ahead carry generator accepts up to four pairs of active low carry propagate [P1(RAP), P1(BAP), P2(RAP), P2(BAP)] and carry generated [G0(BAR), G1(BAP), G3(BAP)] signals and an active high carry input (CN) and provides anticipated active high carries [CNX, CNY, CNZ] across four groups of binary adders. The 74142 also has active low carry propagate [P(RAP)] and carry generate [G(RAP)] outputs. The following boolean equations represent the simulation model:

$$\begin{aligned}
 CNX &= G0 + (P0)(CN) \\
 CNY &= G1 + (P1)G0 + (P1)(P0)(CN) \\
 CNZ &= G2 + (P2)(G1) + (P2)(P1)(G0) + (P2)(P1)(P0)(CN) \\
 G(RAP) &= ((G3 + (P2)(G2) + (P3)(P2)(G1) + (P3)(P2)(P1)(G0))(BAP) \\
 P(RAP) &= ((P3)(P2)(P1)(P0))(BAP)
 \end{aligned}$$

PROGRAM DESCRIPTION

All data is internal. 1 represents high (true), 0 represents low (false). Each output is computed for various sets of input combinations. The results are then checked against known expected values. Results of the simulation are printed out on the user's batch log device in the form of a truth table. The program requests the run time desired and stops when that time is reached and/or exceeded.

06244 FUSPOL

PURPOSE

1. To illustrate alphanumeric data processing by translating an algebraic expression into polish notation.
2. To simulate input data coming from a card reader.

PROGRAM DESCRIPTION

1. Initialize program run time parameters.
2. Input run time limit.
3. Fetch algebraic string records (source) and polish translation verification string record (pol) from batch and write records to disk.
4. The program will process each algebraic record in the following manner:
 - a. Read algebraic record in sequential order from disk.
 - b. Translate record to polish notation.
 - c. Verify translation.
 - d. Output algebraic and polish record on pass 1 only. Repeat this process until all records are processed.
5. If run time has not been exceeded, restart program.
6. When run time has been exceeded, end program.

PROGRAM RUN TIME INFORMATION

Program run time value is entered via operator's console or batch stream at the beginning of the program. This value is only entered once during the entire program.

Run time checks are made throughout the program.

PROGRAM ERROR HANDLING

All data and polish translation errors are reported to the output device for all passes.

Program will continue after reporting error, unless run time has been exceeded.

DISK DATA FORMAT

The disk file will consist of 43 records. Each record will contain 80 words. The order and format is the same as batch.

INPUT DATA INFORMATION

The data is structured as if it appeared on a data card in the following manner:

1. Each record consists of 80 characters or words.
2. Every odd record contains the algebraic source string and every even record contains the expected polish translation string.
3. Record #43 contains a blank character, this terminates the polish translation.

BATCH DATA LIST AND STRUCTURE

RECORD # RECORD COLUMN
 00000000000111111111122222222223333333334.....
 12345678901234567890123456789012345678901.....

-
- 1 A*(B+C)
 - 2 AHC**
 - 3 (A+B)*C
 - 4 AB+C*
 - 5 A+B*C*D
 - 6 ABC**D*
 - 7 (A+B)*(C+D)
 - 8 AB*CD**
 - 9 A=B/C
 - 10 ABC/=
 - 11 (A-B)/C
 - 12 AB=C/
 - 13 A/R+C
 - 14 AB/C*
 - 15 A/(H+C+D)
 - 16 ABC+D*/
 - 17 A/B/C
 - 18 AB/C/
 - 19 (A/R)/C
 - 20 AB/C/
 - 21 A*B-C*D
 - 22 AB*C-D*
 - 23 A*B-(C+D)
 - 24 AB*CD**
 - 25 A
 - 26 A
 - 27 ((((((A))))))
 - 28 A
 - 29 ((A)+((B)))
 - 30 AB*
 - 31 A+B+C*D
 - 32 AB+C*D*
 - 33 (A+B)*(C+D)
 - 34 AB*CD**
 - 35 (E+V)*(D*(A*B))
 - 36 FV+DAN***
 - 37 (C-(H-A/S))*(J-(U-D/F))
 - 38 CHAS/--JUPE/--*
 - 39 C*(I*(N*(D+Y)))+(G/(I/(N/(N/(I-E))))))
 - 40 CINDY****GINNIE-////*
 - 41 (P+(A+C*H))/(+(I+Z*A))/(T+O**)
 - 42 RACH**LIZA**/TOM**/
 - 43
-

D6205 FUSLS2

PURPOSE

1. To represent a typical engineering design application.
2. To exercise logical operations utilizing the FORTRAN logical operations, .AND., .OR., and .NOT., along with the logical constants .TRUE. and .FALSE..

MODEL DESCRIPTION

The model represented is a simple 4-bit input, 4-bit output binary adder. C1, C2, C4 represent the output bits and A1, A2, B1, B2 represent the input bits. The following boolean equations represent the simulation model:

```

K2=(A1)(B1)
C1=(K2(BAR))((A1+B1))
X=(A2)(B2)(K2)
W=A2+B2+K2
T=(B2)(K2)
S=(A2)(K2)
H=(A2)(B2)
C4=R+S+T
C2=X+((W)(C4(BAR)))

```

PROGRAM DESCRIPTION

All data is internal where .TRUE. represents a high state and .FALSE. represents a low state. The outputs are computed for all combinations (16) of the four inputs. The results are compared against expected states. If any error occurs, it is reported to the user's output device. The simulation results are outputted to the user's output device for the first pass only. The only external data required for this program is the desired run time. The program checks the elapsed time throughout its execution. When the time limit has been exceeded, the program will stop execution.

D62.16 FUSPBH

PURPOSE

To demonstrate a particular statistical analysis function.

PROGRAM DESCRIPTION

1. Initialize run time parameters.
2. Input run time limit via user's console or batch stream.
3. The total point count of a hand is computed with ace = 4, king = 3, queen = 2, and jack = 1 (points are not counted here for distributional or other features). The probability of each point count from 0 through 17 is computed, taking into account all the possible honor distributions leading to each particular point count. The probabilities then represent the frequency function, from which the cumulative frequency function and the mean and variance of the point count distribution are also obtained.

PROGRAM RUN TIME INFORMATION

Run time checks are made throughout this program. When run time is exceeded, job will terminate. Data output occurs on first pass only.

ERROR HANDLING

The variance calculation is checked against an expected value. If the calculation is in error, it will be reported to the user's output device. The program will then restart and continue until run time has been exceeded.

D62:7 FUSSNO

PURPOSE

1. To represent a typical operation research model used in industry and commerce and in the public sector to forecast the overall financial and operational effects of changes in basic policies.
2. To demonstrate the use of multiple subroutines for this task.

PROGRAM DESCRIPTION

1. Initialize run time parameters.
2. Input run time limit via user's console or batch stream.
3. Initialize conditions for each new run.
4. Compute estimated enrollment.
5. Compute overhead
6. Compute estimated revenue and budget.
7. Calculate decision based on student/staff policy.
8. Output results to the user's output device.

PROGRAM RUN TIME INFORMATION

Run time checks are made throughout this program. When run time is exceeded, the job is terminated. Simulation output will occur on first pass only.

ERROR HANDLING

If a calculation error should occur, the error is reported to the user's output device. The program will then restart and continue until run time is exceeded.

D620R FUSFER

PURPOSE

1. To show a typical control system analysis application.
2. To exercise the FORTRAN complex number routines.

PROGRAM DESCRIPTION

1. Initialize run time parameters.
2. Input run time limit via user's console or batch stream.
3. Calculate the transfer function $T(j\omega)$ from the following equation:

$$T(j\omega) = \frac{K(1 + j.4\omega)(1 + j.2\omega)}{(j\omega(1 + j2.5\omega)(1 + j1.43\omega)(1 + j.02\omega)^2)}$$

PROGRAM RUN TIME INFORMATION

Run time checks are made throughout this program. When run time is exceeded, the job is terminated. Response output occurs on first pass only.

ERROR HANDLING

The transfer function $T(j\omega)$ is verified for each iteration. If an error occurs, it is reported to the user's output device. The program will then continue until run time has been exceeded.

D62.19 FUSDT5

PURPOSE

1. To utilize the double precision capability of FORTRAN in a general application.
2. To verify the accuracy of the calculation by utilizing the FORTRAN library function for comparison (DSIN(X)).

PROGRAM DESCRIPTION

1. Initialize run time parameters.
2. Input run time limit via user's console or batch stream.
3. Using Taylor's series expansion, calculate SIN X for X=1 through 360 degrees in increments of 0.5 degrees.
4. Calculate SIN X using the FORTRAN library function DSIN(X).

PROGRAM RUN TIME INFORMATION

Run time checks are made throughout this program. When run time is exceeded, the job will terminate. Calculation output occurs on first pass only.

ERROR HANDLING

If the calculated value and functional value differ by $1E-9$, an error will be reported. The program will then continue until the run time has been exceeded.

D6600

 FORTRAN USER SIMULATION PROGRAM ERROR MESSAGES

A.0 PROGRAM = FUSSSD

1. *DISK DATA TRANSFER ERROR*
 DATA VALUE FROM DISK = MEMORY VALUE =

 Data written or read from disk did not match memory
 value.

2. *ASCENDING SORT COUNT ERROR*
 SORT COUNT = SHOULD BE =

 All random numbers have not been sorted to
 ascending sort array.

3. *ASCENDING SORT ERROR*
 - RANDOM DATA FILE -

 R(1)
 .
 .
 .
 R(500)

 - ASCENDING SEQUENTIAL SORT

 ASA(1)
 .
 .
 . ASA(500)

 Incorrect sort has occurred. Value out of sequence.

4. *DESCENDING SORT COUNT ERROR*
 SORT COUNT = SHOULD BE =

 All random numbers have not been sorted to descending
 sort array.

5. *DESCENDING SORT ERROR*
- RANDOM DATA FILE -

R(1)
.
R(500)

- DESCENDING SEQUENTIAL SORT

DSA(1)
.
DSA(500)

Incorrect sort has occurred. Value out of sequence.

6. *BIN SORT COUNT ERROR*
BIN COUNT = SHOULD BE =

All random numbers have not been sorted.

7. *BIN SORT ERROR*
- RANDOM DATA FILE -

R(1)
.
R(500)

- THREE BIN SORT -

CELLX(1)
CELLX(1)
.
CELLX(I)

CELLY
CELLY(1)
.
CELLY(J)

CELLZ
CELLZ(1)
.
CELLZ(K)

Incorrect sort to appropriate cell.

A.3 PROGRAM = FUSMDC

1. *RANDOM NUMBER GENERATION FAILURE*
XXX ERRORS FOUND --JOB FUSMDC ABORTED

The random numbers generated did not meet the following criteria:

$$ABS(R(I)-R(BAR)) < 3 * PHOR$$

where $R(I)$ = random number generated

$R(BAR)$ = mean of all random numbers generated

PHOR = standard deviation of all random numbers generated

2. *POSSIBLE RANDOM NUMBER GENERATOR FAILURE*
XXX ERROR FOUND

3% of the total random numbers generated did not meet the following criteria:

$$ABS(R(I)-R(BAR)) < 3 * PHOR \text{ (symbols same as above)}$$

3. *CORRELATION ERROR DETECTED*
CALCULATED VALUE = SHOULD BE = 2.223E-2

Inaccurate arithmetic operation has occurred in calculating coefficient of correlation of the two created distributions.

C.J. PROGRAM = FUSPOL

1. *BATCH DATA ERROR--APOST JON FUSPOL*

The last record from the batch input file was not a blank record. The data input file is incorrect.

2. SOURCE RECORD IN ERROR = NO BLANKS

One of the "SOURCE" input records from batch did not contain a blank terminator.

3. *TERMINATOR ERROR, BLANK REC NOT SENSED*

Input data on disk file is incorrect. Possible disk I/O data transfer error.

4. *POLISH TRANSLATION ERROR = REC = 11*

Invalid alphanumeric data manipulation has occurred.

REC #	SOURCE
-----	-----
1	A*(B+C)
2	(A+H)*C
3	A+B*C+D
4	(A+B)*(C+I)
5	A-B/C
6	(A-B)/C
7	A/B+C
8	A/(B+C+D)
9	A/B/C
10	(A/B)/C
11	A+H-C+D
12	A*B-(C+D)
13	A
14	(((((A))))))
15	((A)+((B)))
16	A+H+C+D
17	(A+B)*(C+I)
18	(E+V)*(D+(A*N))
19	(C-(H-A/S))*(J-(I-D/E))
20	C*(I*(N*(D+Y)))+(G/(I/(N/(N/(I-E))))
21	(R+(A+C*H))/(L*(I+Z*A))/(T+D*M)

D.0 PROGRAM = FUSDT5

1. *CALCULATION OF FUNCTION DSIN(X) ERROR FOR DEGREE *

Inaccurate SIN function calculation has occurred. Double precision error.

2. *GROSS CALCULATION OF DSIN(X) ERROR*

possible arithmetic operation error library function DSIN(X) inoperative.

E.0 PROGRAM = FUSPHH

1. *VARIANCE ERROR*

MEAN = VARIANCE =

*VALUE SHOULD BE 17.4588303

Possible arithmetic operation error.

F.0 PROGRAM = FUSSD0

1. *CALCULATION ERRORS - PEV *
- *VALUE SHOULD BE IN THE 200-300 RANGE*
2. *CALCULATION ERROR - CTOT *
- *VALUE SHOULD BE IN THE 200-300 RANGE*

The above error messages refer to a possible arithmetic operation error or subroutine data transfer error.

G.P PROGRAM - FUSSEP

1. *COMPLEX TRANSFER FUNCTION ERROR*

Complex arithmetic operator error.

H.A PROGRAM - FUSLS1

1. *CNX RESPONSE ERROR*
2. *CNY RESPONSE ERROR*
3. *CNZ RESPONSE ERROR*
4. *GHAP RESPONSE ERROR*
5. *PRAR RESPONSE ERROR*

All of the above error messages indicate system library logical operator errors.

I.2 PROGRAM - FUSIS2

1. *LOGICAL ERROR DETECTED*
- *SHOULD BE :

Logical operator error in .AND., .OR., .NOT., operation.

D65.00 FORTRAN ERROR DIAGNOSTICS

D65.01 COMPILER ERROR DIAGNOSTICS

The PDX-11M FORTRAN Compiler, while reading and processing the FORTRAN source program, can detect syntax errors (or errors in general form) such as unmatched parentheses, illegal characters, unrecognizable key words, missing or illegal statement parameters.

The error diagnostics are generally clear in specifying the exact nature of the error. In most cases, a check of the general form of the statement in question as described in the PDP-11 FORTRAN Language Reference Manual will help determine the location of the error.

Some of the most common causes of syntax errors, however, are typing mistakes. A typing mistake can sometimes cause the compiler to give very misleading error diagnostics. The user should be careful of the following common typing mistakes:

1. Missing commas or parentheses in a complicated expression or FORMAT statement.
2. Misspelling of particular instances of variable names. If the compiler does not detect this error (it usually cannot), execution may also be affected.
3. An inadvertent line continuation signal on the line following the statement in error.
4. If the user terminal does not clearly differentiate between 0 (zero) and the letter O, what appear to be identical spellings of variable names may not appear so to the compiler, and what appears to be a constant expression may not appear so to the compiler.

If any errors were detected in a compilation, the message:

ERRORS DETECTED: n

will be printed on the initiating terminal; n is the number of errors, not including warnings, detected by the compiler.

The next three sections describe the initial phase and secondary phase error diagnostics and the fatal FORTRAN Compiler error diagnostics.

D6502 ERRORS REPORTED BY THE INITIAL PHASE OF THE COMPILER

The error diagnostics are printed after the source statement to which they apply (the I error diagnostic is an exception). The general form of the diagnostic is as follows:

..... c

where c is a code letter whose meaning is described below:

Code Letter	Description
R	Columns 1-5 of continuation line are not blank. Columns 1-5 of a continuation line must be blank except for a possible 'D' in column 1.
C	Illegal continuation. Comments cannot be continued and the first line of any program unit cannot be a continuation line.
E	Missing END statement. An END statement is supplied by the Compiler if end-of-file is encountered.
H	Hollerith string or quoted literal string longer than 255 characters or longer than the remainder of the statement.
I	Non-FORTRAN character used. The line contains a character that is not in the FORTRAN character set and is not used in a Hollerith string or comment line.
R	Illegal statement label definition. Illegal (non-numeric) character in statement label.
L	Line too long to print. There are more than 80 characters (including spaces and tabs) in a line. Note: this diagnostic is issued preceding the line containing too many characters.
M	Multiply defined label.
P	Statement contains unbalanced parentheses.
S	Syntax error. Multiple equal signs, etc. Statement not of the general FORTRAN statement form.
U	Statement could not be identified as a legal FORTRAN statement.

06503 ERRORS REPORTED BY SECONDARY PHASES OF THE COMPILER

Those compiler error diagnostics not reported by the initial phase of the compiler will appear immediately after the source listing and immediately before the storage map. Since the diagnostics appear after the entire source program has been listed, they must reference the statement to which they apply by using the internal sequence numbers assigned by the compiler.

The general form of the diagnostic is:

IN LINE nnnn MSG#m text

where nnnn is the internal sequence number of the statement in question, m is an integer constant specifying the error number, and text is a short description of the error.

Below, listed alphabetically, are the error diagnostics. Included with each diagnostic is a brief explanation. Refer to the PDP-11 FORTHAN Language Reference Manual for information to help correct the error.

The notation *** signifies that a particular variable name or statement label will appear at that place in the text.

ADJUSTABLE DIMENSIONS ILLEGAL FOR ARRAY ***

All arrays must be dimensioned with integer constants except as specified in the Language Reference Manual.

ARRAY *** HAS TOO MANY DIMENSIONS

An array can have up to seven dimensions.

ATTEMPT TO EXTEND COMMON BACKWARDS

While attempting to equivalence arrays in COMMON, an attempt was made to extend COMMON past the recognized beginning of COMMON storage.

COMMON BLOCK EXCEEDS MAXIMUM SIZE

An attempt was made to allocate more space to COMMON than is physically addressable (>32k words).

DANGLING OPERATOR

An operator (+, -, *, /, etc.) is missing an operand.

Example: I=J+

DEFECTIVE DOTTED KEYWORD

A dotted relational operator was not recognized. Also, possible misuse of decimal point.

DO TERMINATOR ** PRECEDES DO STATEMENT**

The statement specified as the terminator of a DO loop must come after the DO statement.

EXPECTING LEFT PARENTHESES AFTER ****

An array name or function name reference is not followed by a left parenthesis.

EXTRA CHARACTERS AT END OF STATEMENT

All the necessary information for a syntactically correct FORTRAN statement has been found on this line, but more information exists. Possibly due to inadvertent continuation signal on next line, or a missing comma.

FLOATING CONSTANT TOO SMALL

A floating constant in an expression is too close to zero to be represented in the internal format. Use zero if possible.

ILLEGAL ADJACENT OPERATOR

Two operators (+, /, logical operators, etc.) are illegally placed next to each other. Example: I/*J.

ILLEGAL ELEMENT IN I/O LIST

An item, expression, or implied DO specifier in an I/O list is of illegal syntax.

ILLEGAL DO TERMINATOR STATEMENT ****

A DO statement terminator must not be a GO TO, arithmetic IF, RETURN, or DO statement or logical IF containing one of these statements.

ILLEGAL STATEMENT ON LOGICAL IF

The statement contained in a logical IF must not be another logical IF or DO statement.

ILLEGAL TYPE FOR OPERATOR

An illegal variable type has been used with an exponentiation or logical operator.

ILLEGAL USAGE OF OR MISSING LEFT PARENTHESIS

A left parenthesis was required but not found, or a variable reference or constant is illegally followed by a left parenthesis.

INTEGER OVERFLOW

An integer constant or expression value must not fall outside the range -32767 to +32767.

INVALID COMPLEX CONSTANT

A complex constant has been improperly formed.

INVALID DIMENSIONS FOR ARRAY

An attempt was made while dimensioning an array to explicitly specify zero as one of the dimensions.

INVALID DO TERMINATOR ORDERING AT IAPFI ****

Do loops are incorrectly nested.

INVALID EQUIVALENCE

Illegal equivalence, or equivalence that is contradictory to a previous equivalence.

INVALID FORMAT SPECIFIER

A format specifier is not the label of a FORMAT statement or an array name.

INVALID IMPLICIT RANGE SPECIFIER

Illegal implicit range specifier, i.e., non-alphabetic specifier, or specifier range is in reverse alphabetic order.

INVALID LOGICAL UNIT

A logical unit reference must be an integer variable or constant in the range 1 to 99.

INVALID OCTAL CONSTANT

An octal constant is too large or contains a digit other than 0-7.

INVALID OPTIONAL LENGTH SPECIFIER

A data type declaration optional length specifier is illegal. For example, REAL*4 and REAL*8 are legal, but REAL*6 is not.

INVALID RADIX50 CONSTANT

Illegal character detected in a RADIX50 constant.

INVALID RECORD FORMAT

The third parenthetical argument in a DEFINE FILE statement must be the single character U.

INVALID STATEMENT IN BLOCK DATA

It is illegal to have any executable or FORMAT statements in a BLOCK DATA Subprogram.

INVALID STATEMENT LABEL REFERENCE

Reference has been made to a statement number that is of illegal construction. GO TO 999999 is illegal since the statement number is too long.

INVALID SUBROUTINE OR FUNCTION NAME

A name used in a CALL statement or function reference is not valid. Example: use of an array name in a CALL statement routine name reference.

INVALID TARGET FOR ASSIGNMENT

The left side of an arithmetic assignment statement is not a variable name or array element reference.

INVALID TYPE SPECIFIER

An unrecognizable data type was used.

INVALID USAGE OF FUNCTION OR SUBROUTINE NAME

A function name cannot appear in a DIMENSION, COMMON, DATA, OR EQUIVALENCE statement.

INVALID VARIABLE NAME

A variable name contains an illegal character.

LABEL ON DECLARATIVE STATEMENT

It is illegal to place a label on a declarative statement.

MISSING ASSIGNMENT OPERATOR

The first operator seen in an arithmetic assignment statement was not an equal sign (=). Example: I+J=K.

MISSING COMMA

The comma delimiter was expected but was not found. See the section of the FORTRAN Reference Manual that describes the general form of the statement in question.

MISSING DELIMITER IN EXPRESSION

Two operands have been placed next to each other in an expression with no operator between them.

MISSING LABEL

Expecting a statement label but one was not found. Example: ASSIGN J TO I. A valid statement label reference should precede "TO" but does not.

MISSING RIGHT PARENTHESIS

Expecting a right parenthesis but one was not found. Example: READ(5,100). The first non-blank character after the format reference should be a right parenthesis but is not.

MISSING QUOTATION MARK

In a FIND statement, the logical unit number and record number must be separated by a single quotation mark.

MISSING VARIABLE

Expecting a variable, but one was not found. Example: ASSIGN 100 TO I. A variable name should follow the "TO" but one does not.

MISSING VARIABLE OR CONSTANT

Looking for an operand (variable or constant) but found a delimiter (comma, parenthesis, etc.). Example: WRITE(). A unit number should follow the open parenthesis, but a delimiter (close parenthesis) is encountered instead.

MODES OF VARIABLE ** AND DATA ITEM DIFFFF**

The data type of each variable and its associated data list item must agree in a DATA statement.

MULTIPLE DECLARATION FOR VARIABLE ****

A variable cannot appear in more than one data type declaration statement or dimensioning statement. Subsequent declarations are ignored.

NUMBER IN FORMAT STATEMENT NOT IN RANGE

An integer constant in a FORMAT statement is greater than 255 or is zero.

PARENTHESES NESTED TOO DEEPLY

Group repeats in a FORMAT statement have been nested too deeply.

P-SCALE FACTOR NOT IN RANGE -127 TO +127

P-scale factors must fall in the range -127 to +127.

REFERENCE TO INCORRECT TYPE OF LABEL ****

A statement label reference that should be a label on a FORMAT statement is not such a label, or a statement label reference that should be a label on an executable statement is not such a label.

REFERENCE TO UNDEFINED STATEMENT LABEL

A reference has been made to a statement number that has not been defined anywhere in the program unit.

STATEMENT MUST BE UNLABELED

A DATA, SUBROUTINE, FUNCTION, BLOCK DATA, arithmetic statement function definition, or declarative statement must not be labeled.

STATEMENT TOO COMPLEX

An arithmetic statement function has more than 10 dummy arguments. Or the statement is too long to compile. Break it up into 2 or more smaller statements.

SUBROUTINE OR FUNCTION STATEMENT MUST BE FIRST

A SUBROUTINE, FUNCTION or BLOCK DATA statement, if present, must be the first statement in a program unit.

SYNTAX ERROR

Check the general form of the statement with the general form outlined in the Language Reference Manual section that describes that type of statement.

D650A WARNING DIAGNOSTICS

.....

Warning diagnostics report conditions which are not true error conditions, but which may be potentially dangerous at execution time, or which may present compatibility problems with FORTRAN Compilers running on other PDP-11 operating systems. The warning diagnostics are normally enabled, but may be suppressed by use of the /-WP Compiler switch. The form and placement of the warning diagnostics are the same as those for the secondary phase error diagnostics (see section C.1.2) except that the line number reference is replaced with "WARNING". A listing of the warning diagnostics follows:

ADJUSTABLE DIMENSIONS ILLEGAL FOR ARRAY ***

Adjustable arrays must be a dummy argument in a subprogram, and the adjustable dimensions must be integer dummy arguments in the subprogram. Any variation from this rule will cause a dimension of 1 to be used and this warning message to be issued.

NON-STANDARD STATEMENT ORDERING

Although the RSX-11M FORTRAN IV Compiler has less-restrictive statement ordering requirements than those outlined in chapter 7 of the PDP-11 FORTRAN Language Reference Manual, non-adherence to the stricter requirements may cause error conditions on other FORTRAN Compilers. See section 3.5 of this document.

VARIABLE *** IS NOT WORD ALIGNED

Placing a non-LOGICAL*1 variable or array after a LOGICAL*1 variable or array in COMMON or equivalencing non-LOGICAL*1 variables or arrays to LOGICAL*1 variables or arrays may cause this condition. An attempt to reference the variable at runtime will cause an error condition.

VARIABLE *** NAME EXCEEDS SIX CHARACTERS

A variable name of more than six characters was specified. The first six characters were used as the true variable name. Other FORTRAN Compilers may treat this as an error condition. See section 3.1 of this document.

TARGET MUST BE ARRAY

The third argument in an ENCODE or DECODE statement must be an array name.

SYNTAX ERROR IN INTEGER OR FLOATING CONSTANT

An integer or floating constant has been incorrectly formed. For example, 1.23.4 is an illegal floating constant because it contains two decimal points.

UNLABELED FORMAT STATEMENT

All FORMAT statements must be labeled.

USAGE OF VARIABLE ** INVALID**

An attempt was made to EXTEND a common variable, an array variable, or a dummy argument. Or an attempt was made to place in COMMON a dummy argument or external name.

VARIABLE ** INVALID IN ADJUSTABLE DIMENSION**

A variable used as an adjustable dimension must be an integer dummy argument in the subprogram unit.

WRONG NUMBER OF SUBSCRIPTS FOR ARRAY ****

An array reference does not have the same number of subscripts as specified when the array was dimensioned.

D6505 FATAL COMPILER ERROR DIAGNOSTICS

Listed below are the fatal Compiler error diagnostics. These diagnostics, which are sent directly to the initiating terminal, report hardware error conditions, conditions which may require rewriting of the source program, and conditions which may require attention from DEC Software Support. The form of the diagnostic is:

FATAL ERROR n

where n is an error code having one of the following values:

Code	Meaning
C	Constant subscript overflow. Too many constant subscripts have been employed in a statement. SOLUTION - simplify the statement
L	More than 80 characters in input record. SOLUTION - simplify statement or use continuation lines.
O	Unrecoverable error occurred while the Compiler was writing the object file (.OBJ). Possibly, insufficient output file space. SOLUTION - rectify hardware problem, or make more space available for output by deleting unnecessary files.
P	Optimizer push down overflow - statement too complex, or too many common subexpressions occurred in one basic block of a program. SOLUTION - simplify complex statements; report the error to your local software support representative.
R	Unrecoverable hardware error occurred while the Compiler was reading source file. SOLUTION - rectify hardware problem.
S	Subexpression stack overflow - statement too complex. SOLUTION - simplify complex statements.

T Memory Overflow

SOLUTION - break up program into subprograms or compile in a larger partition.

W Unrecoverable error occurred while the Compiler was writing listing file. Possibly, listing file space is not large enough.

SOLUTION - rectify hardware problem, or make more space available for listing file by deleting unnecessary files.

Y Code generation stack overflow - statement too complex.

SOLUTION - simplify complex statements.

Z Compiler error

SOLUTION - report this error to your local software support representative. Please include program listing.

D66:00 OBJECT TIME SYSTEM ERROR DIAGNOSTICS

D66:01 Error Processing Algorithm

The Object Time System detects many Input/Output, arithmetic, invalid argument and other kinds of errors and reports them on the user's terminal via logical device TI:. The action taken for each error is determined by an error control table within the OTS. (This table may be modified during program execution by means of the ERRSET subroutine, see section P.A.)

Error processing for each error is controlled by a control byte. Significant bits are as follows:

- | | |
|-----------------------|--|
| Continuation Bit | If not set, this bit directs the task to exit as a result of this error. If set, the task will continue provided certain other conditions are met. |
| Count Bit | If set, this error is counted against the task error count limit. If that limit is exceeded, the task will exit. |
| Continuation Type Bit | Two types of continuation action are possible: <ol style="list-style-type: none"> 1. Return to routine that reported the error to take appropriate recovery action and proceed, or 2. Take an ERR= transfer in an I/O statement. If an ERR= transfer is specified for the error and none was included in the Input/Output statement, the task will exit. |

The above three conditions must all be satisfied for the task to continue.

- | | |
|---------|---|
| log bit | If the task continues, then the log bit is tested. If the bit is set, an error message is produced before continuing; otherwise the task continues. |
|---------|---|

If any of the above conditions is not satisfied, the task will exit and an error message will always be produced. In this case, the additional text "EXITING DUE TO" is included in the error message so that it is clear why a task is abnormally terminating.

Two additional bits are of interest here since they control the acceptability of EPPSET arguments.

- | | |
|----------------------|---|
| Return permitted bit | If set, then the continuation type bit may be set by EPPSET to specify return. |
| EPPs permitted bit | If set, then the continuation type bit may be set by EPPSET to specify that an EPPs transfer is to occur. |

These two bits are used by EPPSET to check the validity of EPPSET arguments. At least one of these must be set in order to set the continuation bit. Also the continuation type argument is checked against these bits for acceptability.

All four combinations of these two bits occur in the OIS, although most errors are in one of two groups.

1. I/O errors generally permit EPPs continuation type but not return continuation.
2. Most other errors permit return continuation but not EPPs transfer continuation (even if they occur during I/O statement processing).

Notable exceptions are the synchronous system trap errors (numbers 3 through 10) and recursive Input/Output error (number 40) which will always result in task termination, and the Input and Output Formatted Conversion Errors (numbers 63 and 64) which allow both types of continuation.

The initial setting of the error control bits is shown together with error messages in section C.2.3.

D6677 OBJECT TIME SYSTEM FRROR MESSAGE FORMAT

An OTS error message consists of several lines of information formatted as follows:

```

tsknam -- (EXISTING DUE TO) FRROR number
text
(AT PC = address)
(FCS: f.err f.err1 filename unit)
IN xxxxxx AT yyy
FROM xxxxxx AT yyy
...
FROM .MAIN, AT yyy

```

(In the above message prototype, fixed parts of the message are shown in capital letters and variable parts in lower case letters).

The variable parts of the message are:

```

tsknam    -the name of the task in which the error occurred.
number    -the error number
text      -a one-line description of the error.

```

If the OTS error resulted from one of the synchronous system traps, then the program counter will be shown in the line "AT PC =". This line is only produced for errors numbered 5 through 12.

If the OTS error resulted from an error reported to it by File Control Services, the line beginning "FCS:" will be included. Consult the I/O Operations Reference Manual for a description of the FCS error codes.

```

f.err      the value of the F.FPR field of the File
           Descriptor Block (FDH).
f.err1     the value of the F.ERR+1 field of the FDR.
filename   the name of the file (not including type or
           version)
unit       the logical unit on which this error occurred.

```

Next follows a traceback of the subprogram calling nest at the time of the error. Each line represents one level of subprogram call and shows

XXXXXX

the name of the subprogram.

The name of the main program is shown as .MAIN. The name of a subprogram is the same as the name used in the SUBROUTINE or FUNCTION statement. Arithmetic statement functions, CTS system routines and routines written in assembly language will not be shown in the traceback.

VVV

The internal sequence number of the subprogram at which the error, call statement, or function reference occurred.

A question mark, "?", instead of a number indicates that the subprogram was compiled with the /-SN compiler switch (suppress sequence number accounting) in effect and hence the line number is not known for that program unit.

D6700 OBJECT TIME SYSTEM ERROR CODES

D6701 INITIAL CONTROL BIT SETTINGS

The following table shows the initial settings of the significant bits in the error control byte as described in section D6601.

ERROR CONTROL BIT SETTINGS

ERROR NUMBER	CONTINUE?	COUNT?	CONTINUE TYPE	LOG?	PERMITTED	
					ERR?	RETURN?
1	NO	NO	FATAL	YES	NO	NO
2	NO	NO	FATAL	YES	NO	NO
3	NO	NO	FATAL	YES	NO	NO
4	NO	NO	FATAL	YES	NO	NO
5	NO	NO	FATAL	YES	NO	NO
6	NO	NO	FATAL	YES	NO	NO
7	NO	NO	FATAL	YES	NO	NO
8	NO	NO	FATAL	YES	NO	NO
9	NO	NO	FATAL	YES	NO	NO
10	NO	NO	FATAL	YES	NO	NO
20	YES	YES	ERR#	YES	YES	NO
21	YES	YES	ERR#	YES	YES	NO
22	YES	YES	ERR#	YES	YES	NO
23	YES	YES	ERR#	YES	YES	NO
24	YES	YES	ERR#	YES	YES	NO
25	YES	YES	ERR#	YES	YES	NO
26	YES	YES	ERR#	YES	YES	NO
27	YES	YES	ERR#	YES	YES	NO
28	YES	YES	ERR#	YES	YES	NO
29	YES	YES	ERR#	YES	YES	NO
30	YES	YES	ERR#	YES	YES	NO
31	YES	YES	ERR#	YES	YES	NO
32	YES	YES	ERR#	YES	YES	NO
33	YES	NO	RETURN	YES	NO	YES
34	YES	YES	ERR#	YES	YES	NO
37	YES	YES	ERR#	YES	YES	NO
38	YES	YES	ERR#	YES	YES	NO
39	YES	YES	ERR#	YES	YES	NO
40	NO	NO	FATAL	YES	NO	NO
41	YES	YES	ERR#	YES	YES	NO
42	YES	YES	ERR#	YES	YES	NO
43	YES	YES	RETURN	YES	NO	YES
44	YES	YES	ERR#	YES	YES	NO

ERROR CONTROL HIT SETTINGS (Cont)

ERROR NUMBER	CONTINUE?	COUNT?	CONTINUE TYPE	LOG?	PERMITTED	
					ERRS?	RETURN?
60	YES	YES	ERRS	YES	YES	NO
61	YES	YES	ERRS	YES	YES	YES
62	YES	YES	ERRS	YES	YES	NO
63	YES	NO	RETURN	NO	YES	YES
64	YES	YES	ERRS	YES	YES	YES
65	YES	YES	ERRS	YES	YES	NO
66	YES	YES	ERRS	YES	YES	NO
67	YES	YES	ERRS	YES	YES	NO
70	YES	YES	RETURN	YES	NO	YES
71	YES	YES	RETURN	YES	NO	YES
72	YES	YES	RETURN	YES	NO	YES
73	YES	YES	RETURN	YES	NO	YES
74	YES	NO	RETURN	NO	NO	YES
80	YES	YES	RETURN	YES	NO	YES
81	YES	YES	RETURN	YES	NO	YES
82	YES	YES	RETURN	YES	NO	YES
83	YES	YES	RETURN	YES	NO	YES
84	YES	YES	RETURN	YES	NO	YES
85	YES	YES	RETURN	YES	NO	YES
86	YES	YES	RETURN	YES	NO	YES
90	NO	NO	FATAL	YES	NO	NO
91	YES	NO	RETURN	NO	NO	YES
100	NO	NO	FATAL	YES	NO	NO
101	NO	NO	FATAL	YES	NO	NO

D6707 ERROR MESSAGES

GROUP 0 - SEVERE ERRORS

These messages result from severe error conditions for which no error recovery is possible. Consult the PSX-11M Executive Reference Manual for details of what error conditions will cause traps to the System Synchronous Trap Table entries cited below.

1 INVALID ERROR CALL

A TRAP instruction has been executed where low byte is within the range used by the OTS for error reporting (see Section C.2.4) but for which no error condition is defined.

2 TASK INITIALIZATION FAILURE

Task start up has failed for one of the following reasons:

1. The directive to initialize synchronous system trap handling (SVTKSS) has returned an error indication.
2. The executive directive to enable the FPP asynchronous trap (SFPASS) has returned an error indication.
3. The File Control Services initialization call (FINITS) has returned an error indication.

3 ODD ADDRESS TRAP (SST 0)

4 SEGMENT FAULT (SST 1)

This is most likely due to a subscript value out of range on an array reference.

5 T-BIT OR EPT TRAP (SST 2)

6 IOT TRAP (SST 3)

7 RESERVED INSTRUCTION (SST 4)

The program has attempted to execute an illegal instruction. This may be caused by task building with the wrong FORTPAN library for the given hardware configuration. Hardware may have been linked.

8 NON-RSX EMT (SST 5)

The program has executed an EMT instruction whose low byte is not in the range used by the RSX-11M executive.

9 TRAP INSTRUCTION TRAP (SST 6)

A trap instruction has been executed whose low byte is outside the range used for ODS error messages (see C.2.4 below).

10 PDF11/40 FIS TRAP (SST 7)

A module using FIS was linked with a non-FIS FORTRAN library.

11 FPP HARDWARE FAULT

The FPP Floating Exception Code (FFC) register contained the value 0 following an FPP interrupt. This is probably a hardware malfunction.

12 FPP ILLEGAL OPCODE TRAP

The FPP has detected an illegal floating point instruction.

13 FPP UNDEFINED VARIABLE TRAP

The FPP loaded an illegal value (-0.0). This trap should not occur since the ODS initialization routine does not enable this trap condition. A negative zero value should never be produced by any FORTRAN operation.

14 FPP MAINTENANCE MODE TRAP

The FPP has interrupted with a Floating Point Exception Code register value of 14 (octal). This is probably a hardware malfunction.

GROUP 1 - GENERAL INPUT/OUTPUT ERRORS

These messages result from errors related to the file system.

20 REWIND ERROR

An error condition was detected by FCS during the .POINT operation used to position to the beginning of a file.

21 DEFINEFILE ALREADY DONE

A DEFINEFILE statement was attempted on a unit for which one has already been done. The second DEFINEFILE is ignored. To change a DEFINEFILE specification a CLOSE operation may be performed.

22 RECORD TOO LONG

A record has been read which is too large to fit into the buffer specified by the MAXBUF TKM option. Rebuild the task using a larger MAXBUF specification.

23 BACKSPACE ERROR

One of the following errors has occurred:

- a. BACKSPACE was attempted on a file opened for appending
- b. FCS has detected an error condition during the .POINT operation used to rewind the file
- c. FCS has detected an error condition while reading forward to the desired record.

24 END-OF-FILE DURING READ

Either an end-file record produced by the ENDFILE statement or the FCS end-of-file condition has been encountered during a READ statement and no FNDs transfer specification was provided.

25 INVALID RECORD NUMBER

A direct-access READ, WRITE or FIND statement has specified a record number outside the range from one to the value specified in a DEFINEFILE statement.

26 DEFINEFILE NOT DONE

A direct access READ, WRITE, or FIND operation was attempted before a DEFINE FILE was performed.

- 27 MORE THAN ONE RECORD
An attempt was made to read or write more than a single record in an FNCODE or DFCODE statement.
- 24 CLOSE ERROR
An error condition has been detected by FCS during a CLOSE operation when attempting to close a file.
- 29 NO SUCH FILE
A file with the specified name could not be found during an open operation.
- 30 OPEN FAILURE
FCS has detected an error condition during an open operation. (This message is used when the error condition is not one of the more common conditions for which specific error messages are provided).
- 31 MIXED ACCESS MODES
An attempt was made to use both formatted and unformatted operations, or both sequential and direct access operations, on the same unit.
- 32 INVALID LOGICAL UNIT NUMBER
A logical unit number was used which is outside the range specified by the TKB UNITS= option.
- 33 ENDFILE TO DIRECT ACCESS FILE
An end-file record may not be written to a direct access file.
- 34 UNIT ALREADY OPEN
A DEFINEFILE statement, CALL ASSIGN, or CALL FDRSFT was attempted which specified a logical unit already opened for input/output.
- 37 INCONSISTENT RECORD LENGTH
An existing direct access file has been opened whose record length attribute is not the same as specified in the DEFINEFILE or OPEN statement. The record length is not changed.

38 ERROR DURING WRITE

FCS has detected an error condition while writing.

39 ERROR DURING READ

FCS has detected an error condition while reading.

40 RECURSIVE I/O ATTEMPT

An expression in the I/O list of a READ or WRITE statement has caused initiation of another READ or WRITE operation. This can happen if a FUNCTION that performs I/O is referenced in an expression in a READ or WRITE statement I/O list.

41 NO FCS BUFFER ROOM

There is not enough free core left in the File Control Services buffer area to set up required I/O buffers. Rebuild the task with a larger ACTFIL declaration or reduce the level of multibuffering.

42 DEVICE HANDLER NOT RESIDENT

During open operation, the filename specification included a device for which no handler task is resident.

43 FILE NAME SPECIFICATION ERROR

The file name string used in a CALL ASSIGN is syntactically invalid, contains a switch specification, references an undefined device mnemonic, or is otherwise not acceptable to the RSX-11M operating system.

44 RECORDSIZE TOO BIG FOR "MAXBUF"

A DEFINEFILE statement has specified a record size which exceeds the size available in the record buffer. Rebuild the task using a larger THAN MAXBUF specification.

GROUP 2 - ELEMENT TRANSMISSION ERRORS

These messages result from errors related to transmitting data between a FORTRAN program and an internal record.

- 60 INFINITE FORMAT LOOP
- The format associated with an I/O statement that includes an I/O list has no field descriptors to use in transferring those variables.
- 61 FORMAT/VARIABLE - TYPE MISMATCH
- An attempt was made to output a real variable with an integer field descriptor or an integer variable with a real field descriptor.
- 62 SYNTAX ERROR IN FORMAT
- A syntax error was encountered while the OTS was scanning format specification stored in an array.
- 63 OUTPUT CONVERSION ERROR
- During a formatted output operation, the value of a particular number could not be output in the specified field length without loss of significant digits.
- 64 INPUT CONVERSION ERROR
- During a formatted input operation an illegal character was detected in an input field or the input value overflowed the range representable in the input variable. The value of the variable is set to zero.
- 65 FORMAT TOO BIG FOR "FMTRBUF"
- The OTS has run out of memory while scanning an array format that was generated at run time. The default internal format buffer length is 64 bytes.
- 66 RECORD TOO BIG FOR "MAXBUF"
- During an output operation a record was specified that was longer than the maximum record length. The default maximum record length is 132 (decimal) bytes.
- 67 RECORD TOO SMALL FOR I/O LIST
- A READ statement has attempted to input more data than existed in the record being read.

GROUP 3 - ARITHMETIC ERRORS

These messages result from arithmetic overflow and underflow conditions.

- 70 INTEGER OVERFLOW
- During an arithmetic operation an integer's magnitude has exceeded 32767.
- 71 INTEGER ZERO DIVIDE
- During an integer mode arithmetic operation an attempt was made to divide by zero.
- 72 FLOATING OVERFLOW
- During an arithmetic operation a real value has exceeded the largest representable real number. The result of the operation is set to zero.
- 73 FLOATING ZERO DIVIDE
- During a real mode arithmetic operation an attempt was made to divide by zero. The result of the operation is set to zero.
- 74 FLOATING UNDERFLOW
- During an arithmetic operation a real value has become less than the smallest representable real number, and has been replaced with a value of zero.
- 75 FPP FLOATING TO INTEGER CONVERSION OVERFLOW
- During a type conversion, an FPP overflow trap occurred.

GROUP 4 - ARGUMENT ERRORS

These messages result from incorrect calls to FORTRAN-IV supplied functions or subprograms.

- R0 WRONG NUMBER OF ARGUMENTS
- An improper number of arguments were used in a call to a FORTRAN library function or system subroutine.
- R1 INVALID ARGUMENT
- One of the FORTRAN Library Functions or System Subroutines has detected an invalid argument value. See Appendix B.
- R2 UNDEFINED EXPONENTIATION
- An exponentiation has been attempted which is mathematically undefined; e.g., 4.000 .
- R3 LOGARITHM OF NEGATIVE VALUE
- An attempt was made to take the logarithm of a negative number. The result returned is zero.
- R4 SQUARE ROOT OF NEGATIVE VALUE
- An attempt was made to evaluate the square root of a negative value. Zero is returned as the result.
- R5 INVALID ARGUMENT TO LIBRARY FUNCTION.
- An invalid argument was used in a call to a FORTRAN library function.
- R6 INVALID ERROR NUMBER
- The error number argument to one of the subroutines FPPSET or EHRTST is not a valid error number.

GROUP 7 - MISCELLANEOUS ERRORS

90 COMPILER DETECTED ERROR

If an attempt is made to link and run an object file, with errors reported during compilation, generated by the FORTRAN Compiler, this error will result when the illegal source statement is executed.

91 COMPUTED GO TO OUT OF RANGE

The integer variable or expression in a computed GO TO statement was less than 1 or greater than the number of statement label references in the list. Control is passed to the next executable statement (see the PDP-11 FORTRAN Language Reference Manual).

GROUP 8 - SYSTEM DIRECTIVE SUBROUTINES ERRORS

These messages result from incorrect calls to RSX-11M system directive subroutines.

100 DIRECTIVE: MISSING ARGUMENTS

A call to a system directive subroutine was made in which one or more of the arguments required for directive execution was not given.

101 DIRECTIVE: INVALID EVENT FLAG NUMBER.

A call to a system directive subroutine was made in which the argument used for event flag specification was not in the valid range (1 to 64).

D7A00 CONOL I/O EXERCISES

D7A01 I01

Program Description: This program creates a sequential file PD1101 on the system disk of 100 fixed length records.

D7A02 I01

Program Description: This program creates a relative file PD1102 on the system disk which will contain 100 records. Only odd number areas will be used.

D7A03 I03

Program Description: This program reads from PD1101 sequentially then writes the output to a listing device. The line# clause is also tested in this program.

D7A04 I04

Program Description: This program extends file PD1101 with another ten identical format records.

D7A05 I05

Program Description: This program tests rewrite statement on sequential file PD1101. The records which will be rewritten are #30 and #35. Both of them will be read back to verify the results.

D7006 106

Program Description: This program tests delete statement on relative file PD1102. The records which will be deleted are 05 and 07. This test also attempts to delete a non-existent record and a output record. The file status is also checked.

D7007 108

Program Description: This program tests "use" statement together with "close" with lock.

D7008 1010

Program Description: This program tests "same record area" clause of I-O-CONTROL PARAGRAPH.

D7009 1012

Program Description: This program creates relative file - "WFL01" sequentially. writes 100 records and tests "start" in sequential access mode. It also test-before advancing phrase.

D7010 1013

Program Description: This program opens relative file "WFL01", deletes odd number records, rewrites even number records, then reads in sequential access mode.

D7100 COROL USER SIMULATION

The following set of programs are Navy Audit Routines used for COROL compiler checkout.

D7101 CUS1

Program Description: The features tested by this program are 'multiply' and 'divide'.

D7102 CUS2

Program Description: The features tested by this program are 'note', 'go to', 'alter', 'exit' and 'perform'.

D7103 CUS3

Program Description: The features tested by this program are 'if' statements, level numbers, switch-status conditions, relation conditions, class conditions and initialization of items.

D7104 CUS4

Program Description: The feature tested by this program is 'move'. For further tests see D7105.

D7105 CUS5

Program Description: The feature tested by this program is 'move' (continued from CUS4).

D7106 CUS6

Program Description: The features tested by this program are 'add', 'subtract', 'truncation', 'rounded', 'on size error' and 'examine'.

D7107 CUS7

Program Description: The features tested by this program are the following:

Abbreviations:

1. ON is
2. OFF is
3. CURRENCY is (incorporating a test to see if the less than sign is acceptable. The standard explicitly excludes certain characters, primarily those which may be ordinarily found in picture clauses.)
4. PIC
5. COMP
6. JUST
7. SYNC
8. BLANK ZERO

COBOL Character Set
Complete Data Format.

D7108 CUS8

Program Description: The features tested by this program are the options of the 'set' statement used in conjunction with internal tables.

D7109 CUS9

Program Description: The features tested by this program are subscripts used to reference internal tables. The subscripts are numeric literals.

D7110 CUS10

Program Description: The features tested by this program are internal tables referenced using subscripts. The subscripts are composed of numeric integer data names.

D7111 CUS11

Program Description: The features tested by this program are all functions of "set" statements used in conjunction with redefined tables. Set to numeric integer, data name, usage index data item, and indexes assigned to other tables are used to exercise "set" statements.

D7500 RUNTIME FILE I/O ERROR PROCEDURES
.....

When it meets an error condition during I/O operations, the OTS follows the procedure shown below:

1. If the file status key for the file is present, the OTS sets it to the appropriate code for the error condition. Refer to sections D7501 and D7502.
2. If an AT END or INVALID KEY imperative condition is specified for the I/O operation, the OTS takes the path indicated by the imperative statement. The file system performs no other processing in the file for the current statement.
3. If a USE procedure is declared for the file, the OTS performs the USE procedure section, then returns control to the program. The file system performs no further processing for this file.

If no USE procedure is declared for the file, a fatal error condition exists; the OTS aborts the program and displays the following I/O error message:

```
"CAL -- W00037 FILE: NN... NO USE PROCEDURE FOR
      I/O ERROR"
"CAL -- IO ERROR NUMBER - XX"
```

NN represents the name of the file:

XX represents the file control service error code. (See section D7503 for these error codes.)

The following tables show various error numbers and error codes that identify error conditions and messages. The error codes in sections 07501 and 07502 are accessible to the user's program through declaration and use of the FILE STATUS key in the program. The error codes in Table 07503 are returned to the ODS (not the user's program) and represent error conditions detected by the File Control System.

The error message numbers in Section 07510 are merely identifying numbers for the messages and appear at the user terminal in the following form:

"OBL --X001nn -- message ..."

X may be any one of the following:

I - Information message
 W - warning error
 F - Fatal error

nn is the message number.

A table of status key codes follows. The left-hand digit of the status key code is status key 1, and the right-hand digit is status key 2.

D7501 SEQUENTIAL I/O FILE STATUS VALUES

Status Key Code	Meaning
00	No further information (successful)
10	End-Of-File indicator detected
32	Permanent error
34	Permanent error (boundary error)
93	REWRITE attempted without prior READ
95	Allocation failure (no file space on device)
96	No buffer space (program tried to open a file that is sharing buffer space (SAME AREA) with another file)
97	No such file (the file named in an OPEN statement was not found)

D7502 RELATIVE I/O FILE STATUS VALUES

STATUS KEY CODE	MEANING
00	NO FURTHER INFORMATION (SUCCESSFUL)
10	END-OF-FILE INDICATOR DETECTED
22	DUPLICATE KEY ERROR
23	NO SUCH RECORD ERROR
24	PERMANENT ERROR (BOUNDARY ERROR)
93	REWRITE or DELETE attempted without prior READ
95	Allocation failure (no file space on device)
96	No buffer space (program tried to open a file that is sharing buffer space (SAME AREA) with another file)
97	No such file (the file named in an OPEN statement was not found)

D7503 FILE CONTROL SERVICE ERROR CODES

Any of the following I/O error conditions could occur during COMOL program execution. The codes appear in a COMOL message in the form shown below:

"CBL -- IO ERROR NUMBER = nn"

(nn represents the 2-digit file control service error code).

Code ----	Meaning -----
11	End of volume detected
12	write attempted to a locked unit
24	Device full (allocation failure)
26	No such file
27	File locked from write access
38	File not properly closed
39	No buffer space available for file
40	Record too long on READ
46	Record number too large
50	Bad directory file
53	File already open
54	Bad filename
55	Bad device name

D751P RUN-TIME ERROR MESSAGES

NUMBER -----	MESSAGE -----	MEANING -----
1		(Not used)
6	FILE: NN... ATTEMPT TO OPEN 2 "SAME AREA" FILES SIMULTANEOUSLY	The program tried to open a file that uses the same buffer area of another file that is still open. (NN... represents the filename.)
7	FILE: NN... NOT OPEN	The program attempted to perform an I/O operation on a file that was not open. (NN... represents the filename.)
10	FILE: NN... ALREADY OPEN	The program attempted to open a file that was already open. (NN... represents the filename.)
11	SUBSCRIPT TOO BIG	A subscript value used in a subscripted data item reference has exceeded the upper bounds of the number of items in the table.
12	TOO MANY ACTIVE PERFORMS	The number of nested active PERFORM statements being executed by the program has exceeded 30 levels.
13	SUBJECT TO ALTER NOT "GO TO"	An ALTER statement attempted to alter the path of a statement that is not a "GO TO" statement.
14	STOP, CR TO CONTINUE	The program executed a STOP statement. The OTS waits indefinitely. To continue, type carriage return.
15	STOP RUN	The program executed a STOP RUN statement. The program stops all activity and closes all open files.
16	SUBSCRIPT TOO SMALL	The subscript value of data item is less than or equal to zero.
17	UNDEFINED PROCEDURE REFERENCE	Some malfunction in the COHOL compiler or OTS has caused a fatal error.

- 20 FILE: NN... OPTIONAL
FILE MOUNTED? Y OR N? The OTS is asking the operator to specify whether the file NN... is available to the running program. (NN... represents the filename.) Type a Y for yes, or N (or some other character) for no.
- 21 FILE: NN... NOT
ALLOCATED The program requested that the OTS open file NN... with some number of contiguous blocks. The operating system cannot provide the number of blocks requested. This is a warning message; however, the file is not opened. (NN... represents the filename.)
- 22 INDEX VALUE TOO SMALL
OR TOO LARGE AT
SOURCE LINE NNNNN A value for an index name is being used in a SFT statement that is outside the bounds of the table. (NNNNN represents the source program's page-line number.)
- 23 (Not used)
- 24 WRITE ERROR IN DISPLAY A DISPLAY statement encountered a bad device or a record length of more than 132 characters.
- 25 (Not used)
- 26 (Not used)
- 27 OPEN/CLOSE ERROR
IN ACCEPT DISPLAY The program's attempt to open a logical unit and communicate with the specified device has failed. The device is not in the system, or the device handler is not installed in the operating system. The OTS opens an input-only device for DISPLAY or an output-only device for ACCTOP.
- 30 ACCEPT--INPUT TOO LONG A single ACCEPT statement has attempted to read more than 80 characters. The OTS currently imposes a limit of 80 characters on the ACCEPT statement.

- 31 FILE: NN...OPEN ERROR
IO ERROR NUMBER = XX
The program attempted to open file NN... but the open failed. The file control services error code specifies the kind of error. (See Table 6-8 for the FCS error codes.) (NN... represents the filename. XX represents the error code.)
- 32 FILE: NN...CLOSE ERROR
IO ERROR NUMBER = XX
The program attempted to close file NN... but the close operation failed. The FCS error code specifies the kind of error. (See Table 6-8 for the FCS error codes.) (NN... represents the filename. XX represents the error code.)
- 33 FILE: NN... NOT OPEN
The program attempted to close file NN... but file NN... is not open. (NN... represents the filename.)
- 34 FILE: NN... INVALID
LINAGE
The LINAGE clause specified a page body size that has been calculated to be zero. (NN... represents the filename.)
- 35
(Not used)
- 36 FILE: NN... REWRITE/
DELETE NOT LEGAL
WITHOUT PRIOR READ
The program requested a REWRITE or a DELETE operation on a sequential file and the last I/O operation in the file was not a READ.
- 37 FILE: NN... NO USE
PROCEDURE FOR I/O
ERROR.
IO ERROR NUMBER = XX
The ODS detected an I/O error for file NN... and no USE procedure is specified for the file (explicitly or implicitly). The FCS error code XX, specifies the kind of error. (See Table 6-8 for the FCS error codes.) This message results from a fatal error; the ODS executes a STOP RUN and closes all open files.

- 41 FILE: NN... INVALID OPERATION
The program attempted to issue one of the following I/O statements on file open in an incompatible mode:
- A READ on a file open for output;
 - A WRITE on a file open for input or I=0;
 - A REWRITE or DEIFTE on a file open for input or output.
- 42 FATAL ERROR ON SOURCE LINE NNNNN
The RTS is executing an object program that has fatal compilation errors on the indicated source line. This message appears only during debugging of a PDP-11 COROL program. (Fatal compilation errors usually suppress object program generation.) (NNNNN represents the source program's page-line number.)

D76AA COMPILER SYSTEM ERRORS -----

The PDP-11 COROL compiler is a complex system program consisting of many program overlays that manipulate numerous data structures. Throughout the compiler, consistency checks are performed on program flow and the contents of data fields. If the compiler detects an inconsistency, it types a message on the console and terminates the compilation.

Since these messages are very infrequent and require the attention of DEC software support personnel with additional compiler documentation, the message contains only a number. For example, if the compiler detects system error 1, it displays "SYSTEM ERROR 1" on the console before it terminates the compilation. Some consistency checks can occur only when certain language elements are used in the source program.

In the event of a PDP-11 COROL compiler system error, contact your DEC Software Support Specialist immediately.

D7614 DIAGNOSTIC ERROR MESSAGES

.....

This chapter contains a numerical listing of the diagnostic messages generated by the PDP-11 COBOL compiler. The compiler generates these messages whenever it detects an error in the source program. In general, a source error detected by the compiler results in the associated diagnostic message being embedded within the source program listing. That is, when an error is detected in the source program the compiler prints the diagnostic message either before or after the erroneous source program line. There are two exceptions to the general concept of "embedded diagnostics":

1. There may be diagnostic messages listed after the last entry in the Data Division and before the PROCEDURE DIVISION header. These diagnostic messages indicate the detection of duplicate data-name declarations and erroneous data-names referenced in the RELATIVE KEY, FILE STATUS, IMAGE, and VALUE OF ID clauses.
2. There may be diagnostic messages listed after the last line of the Procedure Division. These diagnostic messages indicate the detection of duplicate procedure names and references to undefined procedure names.

In addition to the error-message number and message text, the display contains a source line number, which identifies the error line, and an alphabetic code (discussed below) which informs the user of the seriousness of the error. The information within a diagnostic message line is displayed (from left to right) in the following order:

1. the alphabetic code,
2. the source line number,
3. the numerical error number,
4. the text of the diagnostic message.

For convenience, the alphabetic code is left-justified in the listing so the user merely scans the listing to identify any diagnostic message issued during compilation. Again, for the user's convenience, a summary of the number of errors detected during the compilation is given at the end of the source listings. If no errors are detected during the compilation, the compiler prints "NO ERRORS" at the end of the source listing.

The following illustration shows a typical diagnostic message and the manner in which it appears on the source listing:

```
COBOL 21.00 SPC:X4V903.CHL;1000 14-AUG-74 18:49:10 PAGE 023
```

```
00096      MOVE 72.5 TO N2
00097      IF N2 NOT = T2 DISPLAY "? #10".
00098      *
00099      MOVE 3250 TO N3.
```

```
I 00099 372 POSSIBLE LOW ORDER RECEIVING FIELD TRUNCATION.
```

```
00100      IF N3 NOT = T3 DISPLAY "? #11".
00101      *
00102      MOVE -432 TO N4.
00103      IF N4 NOT = T4 DISPLAY "? #12".
00104      *
```

In the example, the diagnostic message is immediately identified by the appearance of the left-justified alphabetic code "I". The alphabetic code indicates that the message is an I-type (informational) diagnostic; the diagnostic is issued for source line number 99; the error number is 372; and the text of the message is "POSSIBLE LOW ORDER RECEIVING FIELD TRUNCATION." Note that the diagnostic message line, in this example, appears after the source line for which it was issued.

The error messages, used in conjunction with this chapter, provide the user with an important debugging tool. This chapter contains information necessary for interpreting the messages. It explains what caused the error and how the compiler handled the error.

Since different errors cause varying degrees of problems for the compiler (some do not affect the compilation at all, while others may be so critical that they cause an abort of the compilation), the PDP-11 COROL compiler provides four general types (or severity levels) of diagnostic messages. Alphabetic codes (I, W, F, and A) identify these error levels. When it detects an error in the source program, the compiler attempts to recover from the error and continue to compile the program. This recovery action may force the compiler to make an assumption about the source program. The four levels of diagnostic messages are categorized according to the likelihood that the result of the compiler's assumption will be an object program that runs as originally intended by the programmer.

The following list explains the purpose of and the compiler's action for each of the four message levels:

- I (Informational) Informative diagnostic. The purpose of such a diagnostic is to convey information to the user in an observational advisory capacity. The compiler's error recovery (if any is required) is almost certain to be that desired by the user.
- W (Warning) Warning diagnostic. The purpose of this type of message is to warn the user that something is wrong with the associated source statement, but that the compiler can take corrective action on the source element in error. The compiler's recovery action may not be that desired by the user, but the statement, as corrected by the compiler, will be executable.

F (Fatal) Fatal diagnostic. The purpose of such a diagnostic is to indicate to the user that something is fatally wrong with the indicated source statement. By fatal, the compiler means it cannot generate the object code required for the functionality the programmer coded in the erroneous source statement. The compiler's error recovery action will probably leave out a portion of the source program. In general, the compiler will not produce an object program for COBOL source program which have F-type errors in them. However, the user can force the compiler to generate an object program by specifying the /ACC:2 switch in the command string input to the compiler prior to compilation (See Chapter 2 for detailed explanation of the /ACC:n switch.) The /ACC:2 switch instructs the compiler to generate an object program, even if the source program contains F-type errors. In this case, when an F-type error is detected in the Procedure Division, the compiler generates special error trap object code in place of the incorrect source statement. When the object program is executed and special error trap code is encountered, the software displays the following message on the console and aborts the program execution:

FATAL ERROR ON SOURCE LINE XXXXX

where XXXXX is the source line number for which an F-type diagnostic was issued during compilation. For F-type diagnostics issued in the Identification, Environment, and Data Divisions, no special error trap coding is generated since, in general, executable code is not generated for these divisions. However, the fact the F-type diagnostics are issued for these divisions can have a definite effect on the behavior of the execution of the object program.

WARNING: when the user specifies the /ACC:2 switch, the user formally acknowledges to the software a willingness to let the program go into execution even though it may have fatal errors in it. Because the source program has very severe errors in it, the behavior of the associated object program is, in general, unpredictable. In certain cases, such as a COBOL program with files OPENed in I-O mode, letting the program with F-type errors go into execution could be disastrous. Thus, the /ACC:2 switch should be used with caution. The facility is provided as an extra debugging option. It can be useful in shortening the compile-debug cycle, particularly if applied to large COBOL programs which take considerable compilation time. The point is that the user should use the /ACC:2 facility wisely and discretely.

A (Abortive) Abortive diagnostic. The purpose of this type of diagnostic is to inform the user that the compiler must abort compilation. The compiler's error recovery is not possible: it can make no valid assumptions and has no choice but to abort the compilation.

The following pages contain the PDP-11 COBOL compiler diagnostic error messages arranged in numerical presentation to give the error message number and the rest of the diagnostic message to the left. On the right, a detailed explanation of the diagnostic is given indicating the reason(s) for which the diagnostic message is issued and the recovery action taken by the compiler.

NOTE: In many explanations, the word "Fatal." appears as the very last sentence of the explanation. This means that this is fatal diagnostic issued in the Procedure Division. If the /ACC:2 switch is specified in the command string input to the compiler, the associated diagnostic message will cause the generation of the special error trap coding discussed previously.

001 CONTINUE PUNCH WITH BLANK STATEMENT. IGNORED.	A blank line has a continue punch. The continue punch is ignored.
002 QUOTE OR CONTINUE PUNCH MISSING. QUOTE ASSUMES.	A non-numeric literal has no quote and the following line has no continue punch. A terminal quote is assumed at the end of the line.
003 VIOLATION OF AREA A. ASSUMED CORRECT.	The first non-blank character on a continues line occurs in Area A. The error is ignored.
004 LINE LENGTH EXCEEDS INPUT BUFFER. TRUNCATED.	Continuation lines cause a COBOL word to exceed the capacity of the input buffer. The word is truncated on the right; the number of characters retruned depends on the type of word being processed.

- 005 .IO CONTROL. WITHOUT .FILE CONTROL. IGNORED. An I-O-CONTROL paragraph appears when no FILE-CONTROL paragraph was present. The I-O-CONTROL paragraph is ignored.
- 006 .STRING. DATA ITEM MUST HAVE DISPLAY USAGE. A data item in a STRING statement has been given a COMP or INDEX usage. Fatal.
- 007 NAME EXCEEDS 30 CHARACTERS. TRUNCATED TO 30. A character string which appears to be a name exceeds 30 characters in length. The string is truncated on the right to 30 characters.
- 010 NUMERIC LITERAL OVER 18 DIGITS. TRUNCATED TO 18. A numeric literal exceeds 18 digits in length. The literal is truncated on the right, with any necessary adjustment to scaling. The sign is retained.
- 011 NUMERIC LITERAL HAS MULTIPLE DECIMAL POINTS. A numeric literal has more than one decimal point.
- 012 PICTURE CLAUSE ILLEGAL ON GROUP LEVEL. IGNORED. A group level item has a PICTURE clause. The clause is ignored.
- 013 .SELECT. NOT FOUND. SENTENCE IGNORED. A FILE-CONTROL statement should begin with the word SELECT, but does not. All words up to the next period are ignored.
- 014 JUST, SYNC, BLANK CLAUSES WRONG AT GROUP. IGNORED. A group level item may not contain JUSTIFIED, SYNCHRONIZED, or BLANK WHEN ZERO clauses. The clause is ignored.
- 015 FILENAME MISSING OR INVALID. SELECT IGNORED. A SELECT statement either contains no user name or the user name is invalid. The SELECT statement is ignored.

- 016 USAGE CONFLICTS WITH GROUP
USAGE. USES GROUP. The usage specified for this item differs from the usage stated at a higher group level. The group level usage is used.
- 017 ILLEGAL NUMERIC DATANAME
IN .STRING. A numeric data item in a STRING statement has an illegal description. Fatal.
- 020 .ALL. ILLEGAL IN CONTEXT OF
.STRING STATEMENT. All ALL literal has been used in a STRING statement. Fatal.
- 021 SYNTAX ERROR OR NO
TERMINATOR. CLAUSES SKIPPED. A SELECT statement is missing its terminating period; or an error causes the statement to be processed before all clauses were found. The SELECT statement is ignored.
- 022 NUMERIC LITERAL ILLEGAL
IN THIS STATEMENT. A STRING, UNSTRING, or INSPECT statement contains a numeric literal. Fatal.
- 023 SENDING LIST OMITTED IN
.STRING. STATEMENT. A STRING statement contains no sending fields before a DELIMITED BY phrase. Fatal.
- 024 MORE THAN ONE FILENAME
IN .ASSIGN. The non-numeric literal of an ASSIGN clause contains more than one file specification. Only the first specification is used.
- 025 ILLEGAL DATANAME FOLLOWS
.INTO. IN .STRING. The receiving field of a STRING statement is invalid. Fatal.
- 026 SUBSCRIPTING DEPTH EXCEEDS
3. OVER 3 IGNORED. This OCCURS clause is nested more than 3 deep. The OCCURS clause is ignored.
- 027 VALUE ILLEGAL IN OCCURS
ITEM. IGNORED. A VALUE clause appears in an item with an OCCURS clause or in an item subordinate to an OCCURS clause. The VALUE clause is ignored.
- 030 VALUE ILLEGAL IN
REDEFINES ITEM. IGNORED. A VALUE clause appears in an item which either contains a REDEFINES clause, or is subordinate to an item with a REDEFINES clause.

- #31 NO TERMINATOR FOR .IO CONTROL. PARAGRAPH. The I-O-CONTROL paragraph is not terminated by a period. The terminator is assumed present.
- #32 .MAP. NOT APPLICABLE TO SEQ. FILE. IGNORED. An APPLY clause with the MAP option was given for a file that has SEQUENTIAL organization. The APPLY clause is ignored.
- #33 AN IO CONTROL CLAUSE WITHOUT FILES. A file-name is missing in a clause of the I-O-CONTROL paragraph. The clause is ignored.
- #34 SYNTAX ERROR IN .APPLY.. An APPLY clause has illegal syntax. The clause is ignored.
- #35 INVALID ACCESS MODE. TREAT AS SEQUENTIAL. The SELECT statement contains an invalid ACCESS mode. SEQUENTIAL access mode is assumed.
- #36 INVALID FILE ORGANIZATION. TREAT AS SEQUENTIAL. The SELECT statement contains an invalid ORGANIZATION specification. SEQUENTIAL organization is assumed.
- #37 NO SELECT STATEMENTS. A FILE-CONTROL paragraph either contains no SELECT statements or none of those present are valid. The FILE-CONTROL paragraph is ignored.
- #40 .ASSIGN. OMITTED FROM SELECT. SELECT IGNORED. A SELECT statement contains no ASSIGN clause. The SELECT statement is ignored.
- #41 DECIMAL PLACES TRUNCATED. Decimal places have been truncated from a numeric literal during conversion for use as an integer. The integer positions are used.
- #42 INTEGER EXPECTED, ZERO ASSUMES. An integer literal was expected but fractional positions were found. The literal is ignored and a value of zero is assumed.

- 043 INTEGER VALUE TOO BIG,
LARGEST VALUE USED. A numeric literal is too big for conversion as an integer in the give context. A value of 32,767 is used.
- 044 ERROR IN DATA RECORDS
CLAUSE. CLAUSE SKIPPED. The word DATA is not followed by RECORD or RECORDS in the DATA RECORDS clause. The DATA RECORDS clause is ignored.
- 045 ERROR IN LABEL RECORDS
CLAUSE. CLAUSE SKIPPED. The word LABEL is not followed by RECORD or RECORDS in the LABEL RECORDS clause. The LABEL RECORDS clause is ignored.
- 046 NO INTEGER IN BLOCK
CLAUSE. CLAUSE SKIPPED. The BLOCK clause does not contain a numeric literal. The BLOCK clause is ignored.
- 047 BAD VALUE IN BLOCK
CLAUSE. CLAUSE SKIPPED. The numeric literal in the BLOCK clause is not greater than zero. The BLOCK clause is ignored.
- 050 NO INTEGER IN RECORD
CLAUSE. CLAUSE SKIPPED. The RECORD CONTAINS clause does not contain a numeric literal. The RECORD CONTAINS clause is ignored.
- 051 INVALID VALUE IN RECORD
CLAUSE. CLAUSE SKIPPED. The numeric literal in the RECORD CONTAINS clause is not greater than zero. The RECORD CONTAINS clause is ignored.
- 052 INVALID FILENAME.
FD SKIPPED. The word following FD is not valid as a filename. The FD entry is ignored.
- 053 FD TERMINATOR MISSING.
ASSUMED PRESENT. The file description entry contains no period terminator. The error is ignored.
- 054 KEY WORD EXPECTED.
REMAINING CLAUSES SKIPPED. A keyword, which begins a clause, such as BLOCK, LABEL, DATA, etc. is missing. The remainder of the FD entry is ignored.

- 055 NO LABEL CLAUSE IN FD.
.STANDARD. ASSUMED.
- The FD entry contains no LABEL RECORD clause. LABEL RECORD IS STANDARD is assumed.
- 056 NO SFLCT. FILE DELETED.
- The FI entry's filename has no corresponding SFLCT statement. The FD entry is ignored. All references to the filename will be diagnosed as undefined.
- 057 ALLOCATED SPACE EXCEEDS LARGEST RECORD.
- The maximum record size specified by the RECORD CONTAINS clause exceeds the space required for any 01 entry under the same file. The value specified by the RECORD CONTAINS clause is used.
- 060 RECORD AREA EXTENDED TO CONTAIN LARGEST RECORD.
- The space required by the largest 01 record under a file description exceeds the space required by the RECORD CONTAINS clause in the FD entry. The value derived from the 01 record description is used.
- 061 NO RECORD AREA. FILE DELETED.
- No record area is allocated for a file description. The file description is ignored. All references to the file will be diagnosed as undefined.
- 062 ILLEGAL DATANAME FOLLOWS .WITH POINTER. PHRASE.
- The data item used as a pointer in a STRING or UNSTRING statement is illegal. Fatal.
- 063 ILLEGAL SYNTAX IN .STRING. STATEMENT
- A STRING statement contains illegal syntax. Fatal.
- 064 77 ILLEGAL IN FILESECTION. CHANGED TO 01.
- A 77 level item description has been found in the FILE SECTION. The 77 level is treated as an 01 level.

- 065 ILLEGAL WORD FOLLOWS
.DELIMITED BY. PHRASE.
A data-name or literal is expected following a DELIMITED BY phrase in a STRING or UNSTRING statement. Fatal.
- 066 ILLEGAL USE OF .ALL..
IGNORED.
In the VALUE clause, an ALL numeric literal is detected. This is illegal. ALL is ignored by the compiler.
- 067 CONDITION NAME MISSING OR
INVALID. OR IGNORED.
The condition-name in an OR level entry is either missing or invalid. The entire entry is ignored.
- 070 .AREAS. NOT PRESENT IN
.RESERVE.. ASSUMED.
The RESERVE ... AREAS clause has incorrect syntax. The error is ignored.
- 071 .REDEFINES. ON 01 LEVEL
IN FILE SECTION INVALID.
The REDEFINES clause is present on the 01 level in the FILE SECTION, where redefinition is implicit. REDEFINES clause is ignored.
- 072 PICTURE IGNORED
FOR INDEX ITEM.
An item defined as USAGE INDEX has a PICTURE clause. The PICTURE clause is ignored.
- 073 NONNUMERIC PIC ON COMP
ITEM. TREATED AS DISPLAY.
An item defined as USAGE COMP has a picture-string with non-numeric characters. The stated usage is ignored. The item is treated as DISPLAY usage.
- 074 SUBSCRIPT OUT OF RANGE.
ASSUME 1.
A literal subscript is either less than 1 or greater than the maximum allowable value. A value of 1 is used.
- 075 .STATUS. OMITTED FROM
.FILE STATUS.. ASSUMED.
The FILE STATUS clause has incorrect syntax. The error is ignored.

- 276 SOME FILES WITHOUT POSIT.
NO. IN MUL. FILE TAPE. A MULTIPLE FILE TAPE clause contains file-names with POSITION CLAUSES. Not all the file-names contain POSITION clauses. The error is ignored. File searching during OPEN will find the file.
- 277 .MULTIPLE FILE TAPE. SYNTAX ERROR. A MULTIPLE FILE TAPE clause contains a syntax error. The clause is ignored.
- 120 OPERAND CLASSES IN CONFLICT. One or more operands in a statement have invalid class. Fatal.
- 121 POSSIBLY RECEIVING FIELD TRUNCATION. A MOVE statement results in right hand truncation of the receiving field value. This is not an error and is ignored.
- 122 TOO FEW SOURCE FIELDS FOR ADD .GIVING.. At least two valid source operands must appear in an ADD.GIVING statement. Fatal.
- 123 .EXIT. WAS NOT THE ONLY VEPB IN PARAGRAPH. An EXIT statement is not the only statement in a paragraph. The EXIT statement is ignored.
- 124 SENDING ITEM INVALID OR OMITTED. A MOVE statement contains an invalid or missing sending operand. Fatal.
- 125 SENDING ITEM NOT FOLLOWED BY .TO.. A MOVE statement does not have a TO following the sending operand. Fatal.
- 126 RECEIVING ITEM INVALID OR OMITTED. A MOVE statement has no valid receiving operand. Fatal.
- 127 INVALID CLASS FOR DESTINATION FIELD. The receiving operand of an ADD or SUBTRACT statement is not numeric or numeric edited. Fatal.
- 110 RELATIVE KEY OR STATUS NAME INVALID. IGNORED. The name referenced in a RELATIVE KEY or file status clause is invalid. The clause is ignored.

- | | |
|--|--|
| 111 .STOP. SYNTAX ERROR. | The STOP statement is not followed by a literal or the word MIN. Fatal. |
| 112 SIZE ERROR. STATEMENT INCORRECT. | The word ERROR is not found in ON SIZE clause. Fatal. |
| 113 .PROCEDURE DIVISION. OMITTED. | The source program does not contain a Procedure Division. Fatal. |
| 114 INTERMEDIATE RESULT TOO LARGE. HIGH ORDER TRUNC. | An arithmetic statement calls for an intermediate result in excess of 18 digits. The intermediate result is truncated on the left to 18 digits with a possible loss of high order non-zero digits at execution time. |
| 116 .DIVISION. OMITTED AFTER PROCEDURE.. | The word DIVISION is missing in the Procedure Division header. The error is ignored. |
| 117 TERMINATOR MISSING AFTER DIVISION HEADER. | The period terminator is missing from a Division header. The error is ignored. |
| 120 LITERAL INCOMPATIBLE WITH ATTEMPTED USAGE. | Conversion of a literal from one form to another has failed. Fatal. |
| 121 DATA NAME MUST FOLLOW .INTO. IN THIS STATEMENT. | A valid data-name is not present following INTO in a STRING or UNSTRING statement. Fatal. |
| 122 NUMERIC OPERAND MUST BE INTEGER. | A non-numeric operand is illegal in the context of this IF statement. Fatal. |
| 123 OPERANDS CONFLICT IN .SET... TO. STATEMENT. | A SET...TO statement references invalid operands. Fatal. |
| 124 OPERANDS CONFLICT IN .SET... BY. STATEMENT. | A SET...BY statement references invalid operands. Fatal. |

- 125 ILLEGAL FILENAME LITERAL
OR FILENAME DATANAME.
An ASSIGN statement or a
VALUE OF IF statement
contains an invalid file
specification or data-name.
The statement is ignored.
- 126 INVALID SUBJECT OF
SIGN CONDITION.
The subject of a sign
condition is not a numeric
data-name. Fatal.
- 127 ITEM IN TABLE MAY NOT
BE USED AS A SUBSCRIPT.
A data item used as a
subscript is itself a table
element. Fatal.
- 130 .POINTER. MUST FOLLOW .WITH.
IN THIS STATEMENT.
A STRING or UNSTRING
statement has an invalid WITH
POINTER phrase. Fatal.
- 131 RELATIVE KEY INVALID FOR
SEQ. FILE. IGNORED.
A RELATIVE KEY clause has
been applied to a file with
SEQUENTIAL organization. The
RELATIVE KEY clause is
ignored.
- 132 INVALID KEY WORD OF
CONDITION CLAUSE.
An IF statement contains
an invalid condition. Fatal.
- 133 UNIDENTIFIABLE WORD FOUND IN
SUBSCRIPT.
A subscript list contains
a word which is neither a
data-name or numeric literal.
The remainder of the list or
sentence is ignored. Fatal.
- 134 INVALID OBJECT OF
CONDITION.
The object of a relation
condition is an invalid
operand. Fatal.
- 135 SUBSCRIPTS OMITTED. ASSUME
VALUE OF 1.
A reference to a table item
contains no subscript list.
Literal subscripts of 1 are
supplied as defaults.
- 136 RELATIVE INDEX LITERAL OUT
OF RANGE. INDEX USED.
The literal value of a
relative index causes an out
of range reference to the
table. The literal value is
ignored, and the index-name
only is used.
- 137 SUBSCRIPTS GIVEN WHERE NOT
REQUIRED. IGNORED.
A reference is made to
a non-table item, and a
subscript list follows the
reference. The subscript
list is ignored.

- 140 TOO FEW SUBSCRIPTS GIVEN.
ASSUME 1 FOR REST.
- A reference to a table item contains a subscript list with too few subscripts. Default literal subscripts of 1 are supplied for missing subscripts.
- 141 TOO MANY SUBSCRIPTS GIVEN. IGNORE EXCESS.
- A reference to a table item contains too many subscripts in subscript list. Extra subscripts are ignored.
- 142 SUBJECT AND OBJECT USAGE MUST MATCH.
- A relation condition between non-numeric operands requires the same usage for both operands. Fatal.
- 147 ABSOLUTE VALUE STORED.
- A negative value has been supplied for an unsigned numeric item. The absolute value of the numeric literal is stored in the item.
- 151 VERB FOUND IN AREA A.
ALLOWED.
- A statement begins in Area A. The error is ignored.
- 152 EXPECTED. RELATIVE KEY.
DATANAME NOT DEFINED.
- The data-name given in a RELATIVE KEY clause has not been defined in the Data Division. Fatal.
- 153 .LINAGE. CLAUSE DATAITEM
IS TOO LONG.
- A data item named in a LINAGE clause is declared in the DATA DIVISION with more than four decimal integer positions of precision. Fatal.
- 154 PROCEDURE NAME DUPLICATES
DATA NAME. ALLOWED.
- A procedure name is identical to a data-name. The error is ignored, since there can be no ambiguity in legal references.
- 155 STATEMENTS FOLLOWING .GO.
CAN NEVER BE EXECUTED.
- A statement follows an unconditional GO statement. The statements following the GO are compiled, but can not be executed.

- 156 NONSEQUENTIAL FILE MAY NOT BE OPTIONAL. The SELECT statement may specify OPTIONAL only on files with sequential organization. The word OPTIONAL is ignored.
- 157 FILE HAS IO CONTROL CLAUSE CONFLICTS. A file is given conflicting clause specifications in the I-O CONTROL paragraph of the INPUT-OUTPUT SECTION.
- 160 FILE REQUIRES REL. KEY. TREATED AS SEQ. ACCESS. A file with relative organization and random or dynamic access has no RELATIVE KEY clause. The access mode is changed to sequential.
- 161 INVALID SUBJECT OF CONDITION. The word following IF is invalid as the subject of a condition. Fatal.
- 162 UNKNOWN WORD. SCAN TO NEXT CLAUSE. An unknown word is encountered when a clause keyword is expected. All words are ignored up to the next valid clause.
- 163 CLAUSE DUPLICATED. SECOND OCCURRENCE USED. A SELECT statement contains two occurrences of the same clause. The second occurrence is used.
- 164 NO FD FOR THIS SELECT. The filename supplied in a SELECT statement is not further described in an FD in the Data Division. The SELECT statement is ignored, causing the filename to become undefined.
- 165 DIFFERENT SAME REC. AREAS FOR SAME AREA. The compiler has detected a conflict between the SAME RECORD AREA clause and the SAME AREA clause. The compiler rectifies the error in the best possible manner.
- 166 .READ. WITHOUT .INVALID KEY. .AT END. OR .USF. A READ statement contains no conditional clauses and the file being read has no USF procedure applied to it. Fatal.

- 167 IO CONTROL CLAUSE HAS FILE WITH NO .SFILECT.
An I-O-CONTROL clause references a file-name which was not named in a SFILECT statement. The filename is ignored in the I-O-CONTROL statement.
- 170 INTEGER OMITTED IN .RESERVE.. ONE ASSUMED.
A RESERVE clause fails to specify the number of buffer areas to reserve. The clause is ignored, and a default of 1 area is supplied.
- 171 INVALID SUBJECT OF CLASS CONDITION.
The subject of a class condition is not a data item with acceptable class. Fatal.
- 172 VALUE EXCEEDS FIELD CAPACITY. TRUNCATED.
A numeric literal supplied by a VALUE clause exceeds the length of the field. The value is right truncated and stored in the field.
- 173 NO DATA DIVISION STATEMENTS PROCESSED.
The Data Division contains no valid entries. This is an observation only.
- 174 INVALID GRP LEV NUM. REST OF RECORD IGNORED.
A level number is encountered which terminates a previous group item, but does not match any previous group item's level number. All data entries are skipped until the next #1 level, level indicator or header.
- 175 RESERVED WORD AS PARAGRAPH NAME. IGNORED.
A CONTROL reserved word is used as a paragraph or section name. The name is ignored. Fatal.
- 176 MISSING QUOTE ON CONTINUE LINE. QUOTE ASSUMED.
A non-numeric literal is continued, but the first non-space character is not a quote. The error is ignored by assuming a quote in front of the first non-space character.

- 177 COMPARISON OF LITERALS IS NOT PERMITTED. A relation condition has a literal as both subject and object. Fatal.
- 200 COPY IGNORED WITHIN LIBRARY TEXT. A COPY statement is encountered within library text. The COPY statement is ignored.
- 201 INVALID FILENAME ON COPY. COPY IGNORED. A COPY statement supplies a file specification which is invalid. The COPY statement is ignored.
- 202 COPY FILENAME NOT FOUND. A COPY statement supplies a valid file specification, but the file cannot be found on the specified device. The COPY statement is ignored.
- 203 PERIOD OMITTED AFTER .DECLARATIVES.. The word DECLARATIVES is not followed by a period. The error is ignored.
- 204 .DECLARATIVES. OMITTED FROM .END. STATEMENT. The word END is not followed by DECLARATIVES. END DECLARATIVES is assumed.
- 205 PERIOD OMITTED AFTER .END DECLARATIVES.. The words END DECLARATIVES are not followed by a period. The error is ignored.
- 206 SOURCE PROGRAM ENDS IN DECLARATIVES. The end of the source program occurs in the Declaratives area. Fatal.
- 207 DATANAME MUST FOLLOW .WITH POINTER. PHRASE. A STRING or UNSTRING statement contains an invalid WITH POINTER phrase. Fatal.
- 210 .OVERFLOW. MUST FOLLOW .ON. IN THIS STATEMENT. A STRING or UNSTRING statement contains an invalid ON OVERFLOW phrase. Fatal.
- 211 ILLEGAL SENDING FIELD DATANAME IN .UNSTRING. The sending field of an UNSTRING statement has invalid class. Fatal.
- 212 ILLEGAL SYNTAX IN .UNSTRING. STATEMENT. An UNSTRING statement has invalid syntax. Fatal.
- 213 MULTIPLE SIGN CLAUSES ON THIS ITEM. More than one SIGN clause appears in a data description.

- 215 SIGN CLAUSE ON
NONNUMERIC ITEM. A SIGN clause appears in a non-numeric data description. The SIGN clause is ignored.
- 216 SIGN CLAUSE APPLIED
TO NONDISPLAY ITEM. A SIGN clause appears in a numeric data description with usage other than DISPLAY. The SIGN clause is ignored.
- 217 SIGN CLAUSE APPLIED
TO UNSIGNED DATAITEM. A SIGN clause appears in a numeric data description which has no "S" in its PICTURE string. The SIGN clause is ignored.
- 220 ILLEGAL DELIMITING DATA
ITEM IN .UNSTRING. An UNSTRING statement references an invalid delimiter. Fatal.
- 221 .ALL. FIGURATIVE CONSTANT
ILLEGAL IN .UNSTRING. An INSTRING statement contains an ALL literal reference. Fatal.
- 222 ILLEGAL RECEIVING DATANAME
IN .UNSTRING. An UNSTRING statement references a receiving data item which is invalid. Fatal.
- 223 .DELIMITED. CLAUSE REQUIRED
IN THIS .UNSTRING. An UNSTRING statement contains no DELIMITED BY clause. Fatal.
- 224 DATANAME MUST FOLLOW
.DELIMITER IN. PHRASE. An UNSTRING statement contains a DELIMITER IN phrase with an illegal reference. Fatal.
- 225 ILLEGAL DATANAME FOLLOWS
.DELIMITER IN. PHRASE. An UNSTRING statement contains a DELIMITER IN phrase referencing a data item which is invalid.
- 226 DATANAME MUST FOLLOW
.COUNT IN. PHRASE. An UNSTRING statement contains a COUNT IN phrase with an illegal reference.
- 227 ILLEGAL DATANAME FOLLOWS
.COUNT IN. PHRASE. An UNSTRING statement contains a COUNT IN phrase which references a data item which is invalid. Fatal.

- 230 DATANAME MUST FOLLOW
.TALLYING IN. PHRASE
An INSPECT statement
contains a TALLYING phrase
referencing a data item which
is invalid. Fatal.
- 231 ILLEGAL DATANAME FOLLOWS
.TALLYING IN. PHRASE.
An INSPECT statement
contains a TALLYING phrase
referencing a data item which
is invalid. Fatal.
- 232 DATANAME MUST FOLLOW
.INSPECT. VERB.
An INSPECT statement
references a data item which
is invalid. Fatal.
- 233 ILLEGAL DATANAME FOLLOWS
.INSPECT. VERB.
An INSPECT statement
references a data item which
is invalid. Fatal.
- 234 ILLEGAL DATANAME PRECEEDS
.FOR. IN .INSPECT.
An INSPECT...TALLYING
statement references a tally
data item which is invalid.
- 235 .FOR. OMITTED IN
.INSPECT. STATEMENT
An INSPECT...TALLYING
statement has invalid syntax.
- 236 DATANAME MUST FOLLOW
.TALLYING. PHRASE.
An INSPECT...TALLYING
statement does not reference
a tally data-name. Fatal.
- 237 ILLEGAL WORD FOLLOWS
.FOR. IN .INSPECT.
An INSPECT...TALLYING
statement does not state a
valid search condition.
- 240 DATAITEM OMITTED AFTER
.ALL. .LEADING. OR .FIRST.
An INSPECT statement
does not reference a valid
search condition. Fatal.
- 241 .ALL. FIGURATIVE CONSTANT
ILLEGAL IN .INSPECT.
An ALL literal appears in an
INSPECT statement. Fatal.
- 242 ILLEGAL DATANAME FOLLOWS
.ALL. OF .LEADING.
An INSPECT statement
does not reference a valid
search argument. Fatal.
- 243 ILLEGAL DATANAME FOLLOWS
.BEFORE. OR .AFTER.
An INSPECT statement does
not reference a valid
delimiter in the BEFORE/AFTER
phrase. Fatal.
- 244 ILLEGAL DATANAME FOLLOWS
.BY.
An INSPECT statement
does not reference a valid
replacement argument. Fatal.

- 245 ILLLEGAL DATANAME PRECEDES
.BY. An INSPECT statement does not reference a legal dataname or literal preceding the BY phrase. Fatal.
- 246 DATAITEM OMITTED IN
.BEFORE. OR .AFTER. PHRASE. An INSPECT statement does not reference a legal data name or literal after the BEFORE or AFTER phrase. Fatal.
- 247 ILLLEGAL SYNTAX IN
.INSPECT. STATEMENT. Both the TALLYING and REPLACING keywords are missing in the INSPECT statement. Fatal.
- 250 .BY. MUST FOLLOW .CHARACTERS.
IN REPLACING LIST. The INSPECT...REPLACING statement must have CHARACTERS BY phrase completely specified. Fatal.
- 251 DATA ITEM OMITTED AFTER
.BY. IN .INSPECT. The INSPECT...REPLACING statement does not reference a legal data-name or literal after BY. Fatal.
- 252 DATAITEM FOLLOWING .BY.
EXCEEDS 1 CHARACTER. In an INSPECT...REPLACING statement, either when the CHARACTERS BY phrase is specified or when a figurative constant preceding the BY keyword of the ALL, LEADING, or FIRST phrase is specified, the data-name or literal after the BY keyword must be defined as one character in length. Fatal.
- 253 DATAITEMS BEFORE AND AFTER
.BY. UNEQUAL IN SIZE. In an INSPECT...REPLACING statement, the data items before and after the BY keyword of the ALL, LEADING, or FIRST phrase must be equal in length. Fatal.
- 254 .BEFORE. OR .AFTER. OPFRAND
EXCEEDS 1 CHARACTER. In an INSPECT...REPLACING CHARACTERS BY statement, the data-name or literal following the BEFORE or AFTER keyword must be one character in length. Fatal.

- 255 ILLEGAL WORD FOLLOWS
.REPLACING. IN INSPECT..
- A legal keyword was not recognized following REPLACING in the INSPECT statement. Fatal.
- 256 .BY. OMITTED AFTER REPLACING
COMPARISON OPERAND.
- The keyword BY is omitted in the ALL, LEADING, or FIRST phrase where it separates operands to be compared. Fatal.
- 275 INDEX DATA ITEM ILLEGAL
AS INDEX ON TABLE.
- An index data item is used as an index on a table. The index data item reference is ignored. A literal subscript of 1 replaces the index data item reference.
- 276 INDEX NAME NOT DEFINED
FOR THIS TABLE.
- An index-name used in a subscript list either is not defined for this table or appears in the wrong logical position of the subscript list for this table. The index-name is ignored and a default value of 1 is assumed as the subscript.
- 277 RELATIVE INDEX IS INVALID.
- The literal component of a relative index is zero or less in value or is an invalid word. Relative indexing is ignored and the index-name only is used.
- 300 THIS ELEMENTARY ITEM
CANNOT BE A 01 RECORD.
- In the FILE SECTION, an elementary item cannot be a 01 record if it is an edited item, or computational or numeric with sign processing.
- 301 LINAGE # OR LESS
THAN FOOTING.
- The LINAGE clause must specify a page body of at least one line and that page body size must be equal to or greater than the footing size specified in the FOOTING phrase.
- 303 PRINT CONTROL ON RELATIVE
FILE IGNORED.
- An APPLY PRINT-CONTROL clause references a file with RELATIVE organization. The filename is ignored in the APPLY clause.

- 304 SECTION NAME TOO LONG
FOR .USE. HANDLE.
There are too many characters
in the section name in the
USE statement procedure.
Fatal.
- 305 SECTION OR PARAGRAPH
NAME MISSING.
The Procedure Division does
not start with a section or
paragraph name or a section
header is not followed by a
paragraph name. Fatal.
- 306 .PROCEDURE. MISSING IN .USE.
STATEMENT. ASSUMED.
The keyword PROCEDURE is
missing in the USE statement.
It is assumed and processing
is continued.
- 307 .START. WITHOUT .INVALID
KEY. OF .USE.
The INVALID KEY option is
missing from the START
statement and no USE
procedure is declared for the
referenced file. Fatal.
- 310 .WRITE. WITHOUT .INVALID
KEY. OF .USE.
The INVALID KEY option
is missing from the WRITE
statement and no USE
procedure is declared for the
referenced file. Fatal.
- 311 DATA DIVISION MUCH TOO
LARGE.
Too much buffer space is
being used for the files in
this program. Too many files
are declared to be OPEN
simultaneously.
- 320 FILENAME MUST FOLLOW .CLOSE.
VERB.
The data item following the
CLOSE verb was not a
filename. Fatal.
- 321 .NO. MUST FOLLOW .WITH.
IT IS ASSUMED.
The keyword NO is missing in
the WITH NO REWIND phrase of
the CLOSE statement. NO is
assumed present.
- 322 .REWIND. MUST FOLLOW .NO.
IT IS ASSUMED.
The WITH NO REWIND phrase of
the CLOSE statement must be
completely specified. It is
assumed present.
- 323 .REMOVAL. MUST FOLLOW .FOR.
IT IS ASSUMED.
The FOR REMOVAL phrase of the
CLOSE statement must be
completely specified. It is
assumed present.

- 324 .LOCK. OMITTED AFTER .WITH.
IT IS ASSUMED. The keyword WITH in a CLOSE statement is recognized but is not followed by one of the keywords NO or LOCK. The WITH LOCK phrase is assumed present.
- 325 DATANAME SPECIFIED WHERE
FILENAME EXPECTED. The name used in an I/O verb to reference a file was not a filename but was one other data-name. Fatal.
- 326 FILENAME MUST FOLLOW
MODE SPEC. IN .OPEN.. The OPEN statement does not reference a valid filename where a filename reference is expected. Fatal.
- 327 ILLEGAL MODE SPECIFIED
AFTER .OPEN. VERB. One of the OPEN mode keywords INPUT, OUTPUT, I-O, or EXTEND is required immediately after the OPEN verb. None of these four keywords were recognized. Fatal.
- 328 .END. MUST FOLLOW .AT..
IT IS ASSUMED. The keyword END was omitted in the AT END phrase of the READ statement. The AT END phrase is assumed present.
- 331 FILENAME MUST FOLLOW
.READ. VERB. Either the filename was omitted following the READ verb or the data item following the READ verb is not a valid filename reference. Fatal.
- 332 DATANAME OMITTED AFTER .INTO.
IN .READ. The data-name reference following the INTO keyword of the READ statement was omitted. Fatal.
- 333 RECORDNAME MUST FOLLOW
.WRITE. OR .REWRITE. The @1 record-name reference immediately following the WRITE or REWRITE verb was omitted. Fatal.

- 334 STATEMENT IGNORED DUE
TO ILLEGAL RECORDNAME.
The data-name immediately
following the WRITE or
REWRITE verb is not a valid
record-name reference.
Fatal.
- 335 .ADVANCING. OPTION OMITTED
IN .WRITE. 1 ASSUMED.
A data-name reference, numeri-
c integer literal reference,
or the keyword PAGE was not
recognized in the
BEFORE/AFTER ADVANCING phrase
of the WRITE statement. A
numeric integer literal value
of 1 is assumed.
- 336 .FOP. MUST FOLLOW .AT..
IT IS ASSUMED.
The keyword FOP was omitted
in the AT FOP phrase of the
WRITE statement. The AT FOP
phrase is assumed present.
- 337 DATANAME OMITTED AFTER
.FROM.
The data-name reference
following the FROM keyword of
the WRITE or REWRITE
statement was omitted.
Fatal.
- 340 .ADVANCING. INTEGER TO HIGH.
TRUNCATED TO 63.
The numeric integer in the
BEFORE/AFTER ADVANCING phrase
of the WRITE statement is
greater than 63 in value. 63
is assumed present.
- 341 .NO REWIND. ILLEGAL WITH
.IO. OR .EXTEND. MODE.
An OPEN statement with the
I-O or EXTEND mode specified
cannot have the NO REWIND
phrase also specified.
Fatal.
- 342 ILLEGAL .ADVANCING. DATANAME.
1 IS ASSUMED.
The data-name in the BEFORE/
AFTER ADVANCING phrase of the
WRITE statement is not an
elementary numeric integer
data-name reference. A
numeric integer literal value
of 1 is assumed.
- 343 FILENAME MUST FOLLOW
.DELETE. VERB.
Either the file-name was
omitted following the DELETE
verb or the data item
following the DELETE verb is
not a valid file-name
reference. Fatal.

- 344 FILENAME MUST FOLLOW
.STMT. VERB.
Filter the filename was omitted following the STMT verb or the data iter following the STMT verb is not a valid filename reference. Fatal.
- 345 .IFSS. OMITTED AFTER .NOT.
IN .STMT. ASSUMED.
The keyword IFSS is omitted after NOT in the relational condition of the STMT statement. IFSS is assumed present.
- 346 DATANAME OMITTED IN .KEY
IS. PHRASE. ASSUMED.
The RELATIVE KEY data-name for the referenced file was omitted in the KEY IS phrase of the STMT statement. The RELATIVE KEY data-name is assumed present.
- 347 RELATIONAL WORD OMITTED
AFTER .KEY IS. PHRASE.
None of the relational keywords EQUAL, GREATER, or NOT was recognized following the KEY IS phrase of the STMT statement. Fatal.
- 350 TERMINATOR IGNORED IN
.IO CONTROL. PARAGRAPH.
A clause is terminated by a period, but a header does not follow in Area A. The period is ignored; the compiler assumes it is still in the I-O-CONTROL paragraph.
- 351 TERMINATOR IGNORED IN
.SPECIAL NAMES. PARAGRAPH.
A clause is terminated by a period, but is not followed by a header in Area A. The period is ignored, and the compiler continues processing the SPECIAL-NAMES paragraph.
- 352 .NATIVE. MISSING IN
SPECIALNAMES CLAUSE.
The alphabet-name clause does not contain NATIVE or STANDARD-1. The alphabet-name clause is ignored.
- 353 SYNTAX ERROR IN .OBJECT
COMPUTER. PARAGRAPH.
The OBJECT-COMPUTER paragraph contains an unrecognizable word. Recovery is made by scanning over all words until a word is found in area A.

- 354 TERMINATOR OMITTED IN
OBJECT COMPUTER. PARA.
- 356 INVALID USAGE ON
CONDITINAL VARIABLE.
- 357 ILLEGAL SEPARATOR IN
COBOL STATEMENT. IGNORED.
- 360 ILLEGAL CHARACTER FOUND
WITHIN A COBOL WORD.
- 361 UNRECOGNIZABLE TEXT FOUND
IN COBOL STATEMENT.
- 362 COBOL WORD BEGINS WITH
OR ENDS IN HYPHEN.
- 363 NONNUMERIC LITERAL TOO LONG.
TRUNCATED TO MAX.
- The OBJECT-COMPUTER paragraph is not terminated by a period. Recovery is made by scanning over all words until a word is found in area A.
- The level NR condition variable does not have DISPLAY or COMPUTATIONAL USAGE.
- An illegal character was detected between two consecutive words of a COBOL statement. The illegal character is ignored.
- Illegal characters were found in an alphanumeric COBOL word, not within an alphanumeric literal. The illegal characters are replaced by dollar signs in the internal representation of the COBOL word.
- In scanning the source text, the compiler was unable to recognize an alphanumeric COBOL word (i.e., a keyword or user-defined word), an alphanumeric literal, or a numeric literal. The error is not internally corrected and usually will propagate further error messages.
- In attempting to recognize a keyword or user-defined word, the compiler has detected that the COBOL word begins or ends with a hyphen character.
- An alphanumeric literal greater than 132 characters in length is detected. The literal is truncated on right, retaining the first 132 characters as the literal.

- 364 COBOL SOURCE LINE TOO LONG.
TRUNCATED TO MAX. The indicated COBOL source line contains more than 65 characters in terminal format. The excess characters are ignored and only those characters in the printed COBOL source line are retained.
- 365 .HY. OMITTED IN REPLACING
OPTION. COPY IGNORED. The keyword HY was not found in this COPY...REPLACING statement. The statement will be ignored.
- 366 TERMINATOR OMITTED IN
.COPY. IT IS ASSUMED. The required period terminating the COPY statement is omitted. It is assumed present.
- 367 .LINAGE. CLAUSE DATANAME
MUST BE AN INTEGER. A data-name referenced in the LINAGE clause of the FILE SECTION is defined in the WORKING-STORAGE SECTION with decimal places.
- 370 LINAGE. CLAUSE DATANAME
MUST BE UNSIGNED. A numeric data-name referenced in the LINAGE clause of the FILE SECTION is defined in the WORKING-STORAGE SECTION as a signed data item.
- 371 POSSIBLE HIGH ORDER
RECEIVING FIELD TRUNCATION. Truncation of high order information during a MOVE or an arithmetic operation upon a receiving field is possible. This is an observation only.
- 372 POSSIBLE LOW ORDER
RECEIVING FIELD TRUNCATION. Truncation of low order information during a MOVE or an arithmetic operation upon a receiving field is possible. This is an observation only.
- 373 PD HEADER NOT FOLLOWED
BY AN AREA A WORD. The word following the PROCEDURE DIVISION header does not begin in Area A. A scan is made over all words until a word is found in Area A.

- 374 OPEN OPTIONAL FILES ONLY
IN .INPUT. MODE.
An OPTIONAL file can be opened in INPUT mode only. The compiler assumes that the OPTIONAL file is opened in INPUT mode.
- 375 EXPECTED .FILE STATUS.
DATANAME NOT DEFINED.
A data-name referenced in a FILE STATUS phrase of a SELECT clause in the FILE-CONTROL paragraph is not defined in the WORKING-STORAGE SECTION of the DATA DIVISION.
- 376 EXPECTED .VALUE OF ID.
DATANAME NOT DEFINED.
The data-name referenced in a VALUE OF ID clause of an FD is not defined in the WORKING-STORAGE SECTION of the DATA DIVISION.
- 377 EXPECTED .LINAGE. CLAUSE
DATANAME NOT DEFINED.
A data-name referenced in the LINAGE clause of the FILE SECTION is not defined in the WORKING-STORAGE SECTION of the DATA DIVISION.
- 400 .RELATIVE KEY. DATANAME
HAS INVALID CLASS.
A data-name referenced in a RELATIVE KEY phrase of a SELECT clause in the FILE-CONTROL paragraph is defined in the WORKING-STORAGE SECTION with non-numeric class.
- 401 .RELATIVE KEY. DATANAME
HAS INVALID USAGE.
A data-name referenced in a RELATIVE KEY phrase of a SELECT clause must be defined with COMPUTATIONAL or DISPLAY usage in the WORKING-STORAGE section.
- 402 .RELATIVE KEY. DATAITEM
IS TOO LONG.
A numeric integer data-name referenced in a RELATIVE KEY phrase is defined with more than eight digits of precision in the WORKING-STORAGE SECTION.

- 403 .RELATIVE KEY. DATANAME
MUST BE AN INTEGER.
A numeric dataname referenced
in a RELATIVE KEY phrase is
defined in the
WORKING-STORAGE SECTION with
decimal places.
- 404 FILE STATUS. DATANAME
HAS INVALID CLASS.
A data-name referenced in
the FILE STATUS phrase of a
SELECT clause is defined in
the WORKING-STORAGE SECTION
with non-alphanumeric class.
- 405 .FILE STATUS. DATANAME
HAS INVALID USAGE.
A data-name referenced in a
FILE STATUS phrase of a
SELECT clause must be defined
with DISPLAY usage in the
WORKING-STORAGE SECTION.
- 406 LENGTH OF .FILE STATUS.
DATAITEM IS ILLEGAL.
An alphanumeric data-name
referenced in a FILE STATUS
phrase of a SELECT clause
must be defined as an
alphanumeric variable
consisting of two characters
in the WORKING-STORAGE
SECTION.
- 407 .VALUE OF ID. DATANAME
HAS INVALID CLASS.
A data-name referenced in the
VALUE OF ID clause of an FD
is defined in the WORKING
STORAGE SECTION with
non-alphanumeric class.
- 410 .VALUE OF ID. DATANAME
HAS INVALID USAGE.
A data-name referenced in a
VALUE OF ID clause of an FD
must be defined with DISPLAY
USAGE in the WORKING-STORAGE
SECTION.
- 411 LENGTH OF .VALUE OF ID.
DATAITEM IS ILLEGAL.
An alphabetic data-name
referenced in a VALUE OF ID
clause of an FD must be
defined in the
WORKING-SECTION as an
alphabetic variable whose
eloth L falls in the range
9<=L<=40 characters.

- 412 .LINAGE. CLAUSE DATANAME
HAS INVALID USAGE. A data-name referenced in the
LINAGE clause of the FILE
SECTION must be defined with
COMPUTATIONAL usage in the
WORKING-STORAGE SECTION.
- 414 INVALID RECEIVING OPERAND
IN .SET.. IGNORED. A receiving operand of a
SET statement is invalid.
Fatal.
- 415 NO RECEIVING OPERAND
SPECIFIED IN .SET.. No receiving operands are
specified in a SET statement.
Fatal.
- 416 OMITTED OR ILLEGAL OPERAND
AFTER .TO. IN .SET.. A SET statement has no valid
sending operand. Fatal.
- 417 ILLEGAL SYNTAX IN
.SET. STATEMENT. The words TO, UP or DOWN do
not follow the receiving
operands of a SET statement.
Fatal.
- 420 .BY. MUST FOLLOW .UP,
OR .DOWN.. ASSUMED. The keyword BY does not
follow the word UP or DOWN in
a SET statement. BY is
assumed present.
- 421 OMITTED OR ILLEGAL OPERAND
AFTER .BY. IN .SET.. The operand following the UP
BY or DOWN BY phrase in a SET
statement is invalid or
omitted. Fatal.
- 422 NO OPERANDS SPECIFIED
IN .DISPLAY. No operands to be displayed
were recognized by the
compiler in this DISPLAY
statement. Fatal.
- 423 SETTING INDEX NAME OUT
OF RANGE. .SET. IGNORED. A SET statement is attempting
to set an index name using a
literal that is too large.
Fatal.
- 424 .IF. TRUE PATH OMITTED.
ASSUME .NEXT SENTENCE. The true path code is omitted
from the IF statement. NEXT
SENTENCE is assumed as the
true path of the IF
statement.

- 425 CONFLICTING SIGN SYMBOLS
IN PICTURE STRING.
- The compiler has recognized both the + and - sign symbols in this PICTURE string. The compiler ignores the usersupplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 426 ZERO SUPPRESSION CONFLICTS
IN PICTURE STRING.
- The compiler has recognized both the Z and 0 zero suppression symbols in this PICTURE string. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 427 ILLEGAL CHARACTER IN
THE PICTURE STRING.
- A character which is not in the PICTURE string character set is recognized in this PICTURE by the compiler. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 430 .BLANK WHEN ZERO, CONFLICTS
WITH ZERO SUPPRESS.
- A BLANK WHEN ZERO clause has been recognized with a zero suppression field specified in the PICTURE string. The compiler ignores the BLANK WHEN ZERO clause and continues with its processing.
- 431 PARENTHESIZED SPECIFIER
EXCEEDS FOUR DIGITS
- The specification contained inside parentheses of a PICTURE exceeds four digits in length. The compiler ignores the extra digits.
- 432 SPECIFIER MISSING INSIDE
PARENTHESES.
- The specification contained inside parentheses of a PICTURE string is missing. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.

- 431 ILLEGAL SYMBOL PRECEDES
LEFT PAREN. IN PICTURE.
The compiler has recognized an S, V, C, P, D, or "." character preceding a left parenthesis in a PICTURE string. The error is ignored and processing continues.
- 502 INTEGER 1 BEYOND AREA A
TREATED AS LEVEL NUMBER.
An #1 level item was detected beyond Area A and accepted as if in Area A.
- 503 MULTIPLE PICTURES FOR
SAME ITEM. LAST USED.
A data item has more than 1 PICTURE clause. The compiler used the last PICTURE clause specified.
- 504 CLOSING PARENTHESIS MISSING
IN PICTURE.
The right parenthesis is missing in the PICTURE string. The compiler uses the last four digits of the PICTURE string.
- 506 PICTURE EXCEEDS 30 CHARACTERS.
PIC X ASSUMED.
The PICTURE string exceeds 30 characters after expansion. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 507 SPECIFIER OMITTED BEFORE
LEFT PAREN. IN PIC.
The first character of a PICTURE string is a left parenthesis. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 510 SECTION NO. GREATER THAN
49 TREATED AS 49.
A segment number greater than 49 follows the word SECTION. The segment is treated as if it were 49.

- 511 INVALID ITEM LENGTH IN PARENTHESES OF PICTURE. The parenthesized length specifier in a PICTURE contains a non-numeric character. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 514 MULTIPLE FLOATING FIELDS IN NUMERIC EDIT ITEM. The PICTURE string contains multiple floating fields. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 515 MULTIPLE ZERO SUPPRESSION FIELDS IN PICTURE STRING. Multiple zero suppression fields are detected in PICTURE string. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 516 ZERO SUPPRESSION ILLEGAL WITH FLOATING FIELD. The PICTURE string contains both floating and zero suppression fields. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 517 ILLEGAL SYNTAX IN PICTURE STRING. The PICTURE string is not specified correctly according to the rules of PICTURE string syntax. The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.
- 520 MULTIPLE DECIMAL POINTS IN PICTURE. The PICTURE string contains multiple decimal point specifications (V's, P's, or periods). The compiler ignores the user-supplied PICTURE and declares the data-name alphanumeric with a "PICTURE X" declaration.

- 522 INVALID USAGE. IGNORED. The USAGE clause contains an invalid word. The compiler ignores the entire USAGE clause.
- 523 MULTIPLE USAGE CLAUSES. LAST USED. The defined dataname has multiple USAGE clauses specified. The last USAGE clause specified is used by the compiler.
- 524 MULTIPLE OCCURS CLAUSES. LAST USED. The defined dataname has multiple OCCURS clauses specified. The compiler uses the last OCCURS clause specified.
- 525 OCCURS SPECIFICATION ERROR. 1 ASSUMED. The integer entry of the OCCURS clause is either non-numeric or non-integer or does not lie in the range 1 to 4095. The compiler assumes an integer value of 1.
- 526 ILLEGAL WORD AS DATANAME. ASSUME FILLER. A reserved word other than FILLER was seen after a level number in a data description. The compiler assumes the word to be FILLER.
- 527 INVALID INDEX NAME. IGNORED. The compiler did not recognize a valid index name in the INDEXED BY phrase. The compiler ignores the INDEXED BY phrase.
- 530 USAGE OPTION NOT YET IMPLEMENTED. IGNORED. The compiler detected COMP-1 in the USAGE clause. This option is not implemented and is ignored. The default USAGE of DISPLAY is used by the compiler.
- 531 TERMINATOR OMITTED AFTER DATAITEM DESCRIPTION. A data item description entry in the DATA DIVISION is not terminated by a period. The compiler assumes the period is present and continues processing.

- 532 INVALID SIGN IN NUMERIC PICTURE.
The sign character S is detected in a position other than the leading character position of a numeric PICTURE string. The compiler ignores the user-supplied PICTURE and declares the dataname alphanumeric with a "PICTURE X" declaration.
- 533 PICTURE CLAUSE OMITTED ON ELEMENTARY ITEM.
An elementary item is recognized with its PICTURE clause omitted in the description. The compiler declares the dataname either alphanumeric or numeric.
- 534 NUMERIC ITEM EXCEEDS 18 DIGIT MAX. TRUNCATED.
A numeric field is defined in this PICTURE with more than 18 digits of precision. The numeric field is truncated to 18 digits.
- 535 COMP ITEM EXCEEDS 18 DIGITS. ASSIGN 4 WORDS.
A COMPUTATIONAL data item exceeds 18 digits in its specification. The compiler truncates it and allocates four words for its runtime storage.
- 536 INDEX ITEM HAS ILLEGAL CLAUSE.
The compiler recognized a JUSTIFIED, SYNCHRONIZED, VALUE, PICTURE, or SIGN clause on a data item description which has INDEX USAGE. This is illegal. The compiler ignores the offensive clause.
- 537 NUMERIC VALUE FOR DISPLAY ITEM. IGNORED.
The VALUE clause specifies numeric value initialization for a non-numeric data item which is defined with DISPLAY USAGE. This is illegal. The VALUE clause is ignored.
- 540 VALUE TOO LONG. TRUNCATED.
The length of the non-numeric literal in the VALUE clause is longer than the associated data-item. The literal is truncated on the right to fit in the storage allocated to the dataitem.

- 541 CLAUSE DUPLICATION, IGNORED. This clause has been previously recognized for this item. The duplicate clause is ignored.
- 542 INVALID WORD IN .BLANK WHEN ZERO., IGNORED. The keyword ZERO was not recognized in the .BLANK WHEN ZERO clause. The entire clause is ignored.
- 545 LEVEL ILLEGAL AFTER 77, TREATED AS 71. An invalid level number (02-49) follows a 77 level item. The 77 level item is treated as an 01 level item. This action may propagate further flags if it is not a valid group item.
- 550 REDEFINING LENGTH SHOULD MATCH ORIGINAL LENGTH. The length of a non-01 level redefines item is not the same as the length of the item it REDEFINES. The new length is used.
- 551 REDEFINITION OF .OCCURS. ITEM, IGNORED. Data items with the OCCURS clause cannot be REDEFINED. The REDEFINES clause is ignored.
- 552 PROCESSING RESUMES AFTER BAD FD. Prior to issuing this message, the compiler had discovered bad syntax in the FD of the FILE SECTION. The compiler at that time issued an error message identifying the syntax error. Then the compiler went into recovery mode attempting to recognize another FD. the WORKING-STORAGE SECTION header of the PROCEDURE DIVISION. Upon recognizing one of these three language elements, the compiler issues this diagnostic indicating that normal processing resumes.

- 553 INVALID CLAUSE KEYWORD,
OTHER CLAUSES SKIPPED.
- A reserved clause keyword was expected at this point in a data item description entry of the DATA DIVISION, but was not recognized by the compiler. The compiler skips to the next level number data item description.
- 554 INVALID WORD FOLLOWING
.VALUE.. IGNORED.
- The VALUE clause contains an invalid word for this data description. The entire VALUE clause is ignored.
- 555 VALUE CONFLICT,
GROUP VALUE USED.
- This VALUE clause assigns a value to an item subordinate to a group item that also has a VALUE clause. The subordinate VALUE clause is ignored.
- 556 LEVEL NUMBER OMITTED,
ITEM IGNORED.
- The level number has been omitted in a data item description. All the source text is ignored up to and including the next period.
- 557 NO VALUE AFTER CONDITION
NAME. OR IGNORED.
- An 88 level condition-name has no VALUE clause specified. The entire 88 level data item is ignored.
- 561 .NO. MISSING IN
ADVANCING PHRASE. ASSUMED.
- The keyword NO is missing in the ADVANCING phrase of the DISPLAY statement. NO is assumed present.
- 562 ADVANCING. MISSING AFTER
.NO..ASSUMED.
- The keyword ADVANCING is missing in the ADVANCING phrase of the DISPLAY statement. ADVANCING is assumed present.
- 563 DUPLICATE DATANAME,
FIRST USED.
- In the DATA DIVISION, the same dataname is defined more than once. Qualification is not yet implemented. The first definition of the dataname is used.

- 564 ILLEGAL PARAGRAPH HEADER
IN DIV. PAR IGNORED. An illegal paragraph header appears in the IDENTIFICATION DIVISION. The paragraph is ignored.
- 565 ILLEGAL PARAGRAPH HEADER
ENV DIV. PAR IGNORED. An illegal paragraph header appears in the ENVIRONMENT DIVISION. The paragraph is ignored.
- 566 NUMERIC LITERAL ILLEGAL
ON GROUP ITEM. IGNORED. A numeric literal is illegal in the VALUE clause of a group item. The VALUE clause is ignored.
- 567 .ENVIRONMENT. NOT FOLLOWED
BY .DIVISION.. The word ENVIRONMENT is not followed by the word DIVISION. DIVISION is assumed present.
- 570 TERMINATOR MISSING AFTER
.DATA DIVISION. HEADER. The DATA DIVISION header is not followed by a period. The period is assumed present and processing continues.
- 571 TERMINATOR MISSING AFTER
PARAGRAPH HEADER. A paragraph header in the IDENTIFICATION or ENVIRONMENT DIVISION is not terminated by a period. The period is assumed present and processing continues.
- 572 LEVEL 66 NOT IMPLEMENTED.
IGNORED. A level 66 (RENAMES) data item has been recognized by the compiler. Level 66 items are not implemented by the compiler. The entire data description entry is ignored.
- 573 .SECTION. OMITTED FROM
SECTION HEADER. An ENVIRONMENT DIVISION section name is not followed by the word SECTION. The error is ignored.
- 574 TERMINATOR MISSING AFTER
SECTION HEADER. An ENVIRONMENT DIVISION section header is not terminated by a period. The error is ignored.

- 575 IDENTIFICATION DIVISION
WAS OMITTED. The program contains no
IDENTIFICATION DIVISION. The
compiler assigns a default
PROGRAM-ID of C0RLPW and
continues processing at the
DATA DIVISION header.
- 576 NO IDENTIFICATION OR
ENVIRONMENT DIVISION FOUND. The program contains no
IDENTIFICATION OR ENVIRONMENT
DIVISIONS. The compiler
assigns a default PROGRAM-ID
of C0RLPW and continues
processing at the DATA
DIVISION header.
- 577 IDENTIFICATION DIVISION HEADER
OMITTED. The program contains no
IDENTIFICATION DIVISION
header. The compiler resumes
processing at the next
paragraph header.
- 600 ILLEGAL LEVEL NUMBER.
TREAT AS 01. This level number is not an
01-49, 66, 77, or 88 level
number. The level number is
assumed to be 01.
- 601 TERMINATOR MISSING AFTER
ENV DIV HEADER. The ENVIRONMENT DIVISION
header is not terminated by a
period. The period is
assumed present and
processing continues.
- 602 .DATA. NOT FOLLOWED BY
.DIVISION. The word DATA is not
followed by the word
DIVISION. DIVISION is
assumed present.
- 603 ENVIRONMENT DIVISION HEADER
OMITTED. The program contains no
ENVIRONMENT DIVISION header.
The compiler resumes
processing at the next
paragraph header.
- 604 NO VALID HEADERS FOUND. Over 50 words have been
scanned without finding the
word IDENTIFICATION. The
compiler assumes that its
input is not a COBOL source
program. Compilation is
aborted.

- 625 .IDENTIFICATION. NOT FOLLOWED
BY .DIVISION.. The word IDENTIFICATION is
not followed by the word
DIVISION. DIVISION is
assumed present.
- 626 TERMINATOR OMITTED AFTER
.ID DIVISION. HEADER. The IDENTIFICATION DIVISION
header is not terminated by a
period. The period is
assumed present and
processing continues.
- 627 .PROGRAMID. EXPECTED AFTER
DIVISION HEADER. The IDENTIFICATION DIVISION
header is not followed by the
PROGRAM-ID paragraph. The
error is ignored and
processing continues.
- 610 TERMINATOR OMITTED AFTER
.PROGID. PARA HEADER. The PROGRAM-ID paragraph-name
is not terminated by a
period. The period is
assumed present and
processing continues.
- 611 INVALID PROGRAM NAME IN
.PROGRAM ID. PARAGRAPH. The program name of the
PROGRAM-ID paragraph contains
an invalid character or
exceeds nine characters in
length. The error is ignored
and processing continues.
- 612 TOO MANY FILES FOR LUNS
OR TEMPORARY SPACE. The compiler has discovered
either that more than 30
files are declared in the
program or that more than 30
SAME RECORD AREA clauses are
specified in the program.
The compiler imposes a limit
of 30 in both cases, because
the associated compiler
and/or object time table
space is exhausted.
- 613 INVALID WORD SUSPENDS
PROCESSING. SCAN FORWARD. An unidentifiable word is
found where a verb is
expected. A scan is made to
a verb, or period, or word in
Area A.

- 614 PROCESSING RESTARTS ON
VERR.
- Due to a previous syntax error, the compiler went into recovery mode looking for the next verb, period, or Area A word upon which to resume compilation. The compiler has recognized a verb and resumes normal compilation at this point. This message is an observation only.
- 615 PROCESSING RESTARTS ON
PROCEDURE NAME.
- Due to a previous syntax error, the compiler went into recovery mode looking for the next verb, period, or Area A word upon which to resume compilation. The compiler has recognized an Area A word and resumes compilation at this point. This message is an observation only.
- 616 PROCESSING RESTARTS AFTER
TERMINATOR
- Due to a previous syntax error, the compiler went into recovery mode looking for the next verb, period, or Area A word upon which to resume compilation. The compiler has recognized a period and resumes normal compilation on the word following the period. This is an observation only.
- 620 PARAGRAPH TERMINATOR
ASSUMED OMITTED.
- A paragraph was terminated without a period. The period is assumed and processing continues.
- 621 .LINAGE. FOR RELATIVE
FILE. CLAUSE IGNORED.
- The LINAGE clause must not be specified for a file which has a RELATIVE organization. The LINAGE clause is ignored.
- 622 TERMINATOR MISSING AFTER
PROCEDURE NAME.
- A section or paragraph name is not terminated by a period. The period is assumed present and processing continues.
- 623 .ELSE DOES NOT HAVE
ASSOCIATED .IF.. IGNORED.
- The word ELSE has no associated IF statement. The ELSE is ignored.

- 624 WHEN EXPECTED TO FOLLOW
ELSE.. ELSE. IGNORED.
- 625 JUSTIFY. WITH NUMERIC OR
EDITED ITEM. IGNORED.
- 626 BLANK WHEN ZERO. ILLEGALLY
SPECIFIED. IGNORED.
- 627 REDEFINED ITEM NOT FOUND.
REDEFINES. IGNORED.
- 630 REDEFINES. MUST FOLLOW
DATA NAME. IGNORED.
- 631 DEPTH OF NESTED .IF.
EXCEEDS LIMIT.
- 632 PROCEDURE NAME DUPLICATED.
- 633 UNDEFINED PROCEDURE NAME.
- A sentence ends with the word ELSE. The ELSE is ignored.
- The JUSTIFIES clause must not be specified for a numeric or numeric-edited dataitem. The JUSTIFIED CLAUSE IS IGNORED.
- The BLANK WHEN ZERO clause must be specified only for a numeric or numeric-edited data item. The clause is ignored.
- The second data-name in a REDEFINES clause is not a data-name previously defined. The REDEFINES clause is ignored.
- The REDEFINES keyword appears in the wrong position of a data description entry. The REDEFINES clause is ignored.
- A nested IF statement has exceeded the maximum depth of 31 levels. The compiler ignores nesting beyond this depth of nesting.
- In the PROCEDURE DIVISION, the same procedure-name is defined more than once. The compiler uses the first occurrence of the name and ignores the duplicate entry. All references to the procedure-name will refer to the first definition of the procedure-name.
- In the PROCEDURE DIVISION, a reference is made to an undefined paragraph or section name. Fatal.

- 634 FILENAME LITERAL TOO LONG,
TRUNCATED. A file specification in the ASSIGN clause exceeds 41 characters in length. It is truncated to 40 characters.
- 635 BAD PROCEDURE NAME AFTER
.GO TO. The word after GO TO is an invalid procedure-name. Fatal.
- 636 INVALID INTEGER OR
DATANAME. In the LINKAGE clause, the compiler failed to recognize a non-negative integer literal or a numeric integer data-name. This phrase of the LINKAGE clause is ignored.
- 637 .GO TO. HAS MULTIPLE
PROCEDURE NAMES. A simple GO TO statement (i.e., without the DEPENDING ON phrase) has more than one procedure-name. The GO TO statement is ignored.
- 640 INVALID WORD FOLLOWS
.DATA DIVISION. The word following the DATA division header either does not start in Area A or is not one of the reserved words FILE, WORKING-STORAGE, or PROCEDURE. The compiler goes into recovery mode skipping all source text until one of the keywords FILE, WORKING-STORAGE or PROCEDURE is recognized.
- 641 INVALID WORD IN FILE
SECTION. SCAN FORWARD. An invalid word was detected in the FILE SECTION where the keyword FD is expected. The compiler goes into recovery mode skipping all source text until one of the keywords FD, WORKING-STORAGE, or PROCEDURE is recognized.
- 642 .VALUE OF ID. WITH
.OMITTED LABELS. IGNORED. The VALUE OF ID clause is specified with the LABEL RECORDS ARE OMITTED clause. This is illegal. The VALUE OF ID clause is ignored.
- 643 .SECTION. EXPECTED AFTER
HEADER WORD. The keyword SECTION is omitted after the word FILE or WORKING-STORAGE. SECTION is assumed present and processing continues.

- 644 TERMINATOR EXPECTED AFTER SECTION HEADER.
The FILE SECTION or WORKING-STORAGE SECTION header is not terminated by a period. The period is assumed and processing continues.
- 645 OTS BUFFER SPACE OVERFLOW.
The total buffer space required by the files in the program has exceeded 327 characters. This cannot be accommodated by the object-code system.
- 646 .OF, OR .ID, MISSING IN .VALUE OF ID..
One or both of the keywords OF or ID is omitted in the VALUE OF ID clause. Their presence is assumed and processing continues.
- 647 ILLEGAL WORD IN AREA A, SCAN FORWARD.
In the WORKING-STORAGE SECTION, an 21 or 77 level number or the PROCEDURE keyword was expected in Area A, but was not recognized. The compiler goes into recovery mode skipping source text until one of the three language elements aforementioned is recognized in Area A.
- 651 GROUP LEVEL .VALUE, DISALLOWED.
The VALUE clause on this group item is not permitted because a subordinate elementary item has a non-DISPLAY usage specified or has a SYNCHRONIZED clause specified. The group VALUE CLAUSE IS IGNORED.
- 652 RELATIVE FILE IN .MULTIPLE, FILE TAPE, CLAUSE.
In the I-O-CONTROL paragraph, the MULTIPLE FILE TAPE clause is specified for a file whose organization is RELATIVE. This is illegal. The MULTIPLE FILE TAPE clause is ignored for this file.

- 653 .VALUE. CLAUSE ILLEGAL IN
FILE SECTION.
- A .VALUE. clause is specified
for a data description entry
given in the FILE SECTION.
This is illegal. The .VALUE.
clause is ignored.
- 654 SYNTAX ERROR IN CURRENCY
CLAUSE.
- The alphanumeric literal
expected in the CURRENCY SIGN
clause of the SPECIAL-NAMES
paragraph is omitted. The
clause is ignored and the
currency sign defaults to the
dollar sign.
- 655 ILLEGAL CURRENCY SIGN.
- The alphanumeric literal in
the CURRENCY SIGN clause is
not allowed as the currency
sign either because the
literal is longer than one
character or because it is an
invalid COMOL currency sign.
The CURRENCY SIGN clause is
ignored and the currency sign
defaults to the dollar sign.
- 656 SPECIALNAMES CLAUSE INVALID.
- An unrecognizable word
appears in a position where a
SPECIAL-NAMES paragraph
clause keyword is expected.
All source text is skipped up
to the next recognizable
keyword.
- 657 SYNTAX ERROR IN
DECIMALPOINT CLAUSE.
- The keyword COMMA is omitted
in the DECIMAL-POINT IS COMMA
clause of the SPECIAL-NAMES
paragraph. The clause is
ignored.
- 660 .AFTER. MISSING IN
.USE. STATEMENT. ASSUMED.
- The keyword AFTER is omitted
in the USE statement. AFTER
is assumed present and
processing continues.
- 661 NO .ERROR. OR .EXCEPTION.
IN .USE. ASSUMED.
- One of the keywords ERROR or
EXCEPTION is omitted in the
USE statement. The missing
keyword is assumed present
and processing continues.
- 662 NO KNOWN CLAUSES IN
SPECIALNAMES.
- The SPECIAL-NAMES PARAGRAPH
contains no valid clauses.
This is an observation only.

- 663 REDUNDANT .USE. COVERAGE.
PREV. .USE. IGNORED.
Multiple USE statements have referenced the same file. The last USE statement specified is then applied to the referenced file.
- 664 UNKNOWN OPEN MODE IN
.USE. STATEMENT.
An unrecognizable OPEN mode option was specified in the USE statement. Fatal.
- 665 GROUP ITEM HAS BEEN CALLED
FILLER.
A FILLER item cannot have any elementary items subordinate to it.
- 666 MISSING ENVIRONMENT
DIVISION.
The program does not contain an ENVIRONMENT DIVISION. The compiler skips to the DATA DIVISION and continues processing.
- 667 DIVISION BY ZERO.
The divisor of a DIVIDE statement is a literal of zero value. The error is ignored.
- 670 VALUE NOT PERMITTED WITH
THIS ITEM.
A VALUE clause is recognized in a data description entry which contains a PDEFINIS or an OCCURS clause. This is illegal. The VALUE clause is ignored.
- 671 INVALID CONSTANT OR
LITERAL FOLLOWING .ALL..
The reserved word ALL is not followed by a non-numeric literal or a figurative constant. Thus, this is not a valid ALL literal. ALL is ignored and processing continues.
- 672 BAD FILENAME IN .USE.
STATEMENT.
An unrecognizable word appears where a filename is expected in the USE statement. Fatal.
- 673 FILE NOT CLOSED.
The referenced file was OPENed but there was no CLOSE statement detected for this file in the program.

- 675 FILE COVERED BY CONFLICTING USE PROCEDURE.
There was more than one conflicting USE procedure specified for the referenced file. Fatal.
- 677 SUPPLIED VALUE INVALID FOR NUM ITEM. IGNORED.
The VALUE clause specifies invalid value initialization for a numeric data item. The compiler ignores the VALUE clause.
- 720 FILE ACCESSED BY VERB REQUIRING RELATIVE ORG.
A file whose organization is SEQUENTIAL is referenced by the START or DELETE verbs or by an I/O verb which has the INVALID KEY clause specified. This is illegal. In all these cases, the referenced file must have RELATIVE organization. Fatal.
- 701 FILE ACCESSED BY VERB REQ. SEQUENTIAL ORG.
A file whose organization is RELATIVE is referenced by an I/O verb which has the AT EOP or ADVANCING clauses specified. This is illegal. The referenced file must have SEQUENTIAL organization. Fatal.
- 702 VERB NOT IMPLEMENTED.
An ANS 1974 COBOL verb appears that is not implemented in this release of the compiler. The compiler scans to another verb, period, or word in Area A.
- 704 OCCURS ILLEGAL FOR 01 OR 77 ITEM. IGNORE.
An OCCURS clause is specified for an 01 or 77 level data-name. The compiler ignores the OCCURS clause.
- 725 ACCEPT FROM. OBJECT NOT IN SPECIALNAMES.
The mnemonic name used in the ACCEPT statement was not defined in the SPECIAL-NAMES paragraph. Fatal.
- 706 ACCEPT IDENTIFIER INVALID.
The word following the ACCEPT verb is not a data-name or is a data-name which has non-DISPLAY usage or invalid class. Fatal.

- 707 VERB OR COND. CLAUSE
CONFLICTS WITH FILE ACCESS. There is a conflict between the ACCESS MODE of the referenced file and the I/O verbs and/or condition clauses which reference this file. Fatal.
- 710 DATANAME AFTER .GO
DEPENDING. INVALID. The word following the DEPENDING ON phrase of the GO TO statement is not a dataname or is a data-name which has INDEX usage. This is illegal. Fatal.
- 711 INVALID CLASS OF DATANAME
AFTER .GO DEPENDING. The data-name following the DEPENDING ON phrase of the GO TO statement is not a numeric data-name or is a numeric, non-integer data-name. This is illegal. Fatal.
- 712 .DISPLAY UPON. OBJECT
NOT IN SPECIALNAMES. The mnemonic name used in the DISPLAY statement was not defined in the SPECIAL-NAMES paragraph. Fatal.
- 713 .DISPLAY. OPERAND IS INVALID. A data item in the DISPLAY statement has an invalid class or USAGE.
- 714 MISSING OR INVALID OPERAND.
OF .MULTIPLY.. One of the operands of the MULTIPLY statement either is missing or is invalid. Fatal.
- 714 ILLEGAL .MULTIPLY. DUE
TO MISSING .BY.. The keyword BY is omitted in the MULTIPLY statement. Fatal.
- 716 MISSING OR INVALID OPERAND
OF .DIVIDE.. One of the operands of the DIVIDE statement either is missing or is invalid. Fatal.
- 717 ILLEGAL .DIVIDE. DUE TO
MISSING .BY. OR .INTO.. One of the keywords BY or INTO is omitted in the DIVIDE statement. Fatal.

- 720 .GIVING. OPTION OF .DIVIDE.
MISSING.
The GIVING option must be specified in a DIVIDE statement when one of the following syntactic elements is present in the DIVIDE statement: (1) a numeric literal follows the keyword INTO or (2) the keyword BY is specified. In this DIVIDE statement, the GIVING option was omitted while one of the two aforementioned syntactic elements were present. Fatal.
- 721 MISSING OR INVALID OPERAND
OF .ADD..
One of the operands of the ADD statement either is missing or is invalid. Fatal.
- 722 .TO. OR .GIVING. MISSING
FROM .ADD..
One of the keywords TO or GIVING is omitted in the ADD statement. Fatal.
- 723 MISSING OR INVALID
OPERAND OF SUBTRACT.
One of the operands in the SUBTRACT statement either is missing or is invalid. Fatal.
- 724 FILE NEEDS DYNAMIC ACCESS
FOR .READ NEXT..
In a READ NEXT statement, the referenced file must have ACCESS MODE IS DYNAMIC specified in the FILE-CONTROL paragraph. Fatal.
- 725 HAD PROCEDURE NAME IN
.PERFORM..
A missing or invalid procedure-name is recognized in the PERFORM statement. Fatal.
- 726 ILLEGAL OPERAND OF .TIMES.
OPTION OF .PERFORM..
The TIMES operand of the PERFORM statement is not a numeric integer data-name or numeric integer literal. The compiler assumes a value of 1 for the TIMES operand.
- 727 .TIMES. MISSING FROM
.PERFORM.. ASSUMED.
The PERFORM statement does not contain the keyword TIMES but does contain the iteration value required to execute the PERFORM correctly. The keyword TIMES is assumed present.

- 730 PROCEDURE NAME OMITTED
IN .ALTER..
A valid procedure name was
not recognized in the ALTER
statement. Fatal.
- 731 ILLEGAL .ALTER. DUE
TO MISSING .TO..
The keyword TO was not
recognized in the ALTER
statement. Fatal.
- 732 FILE HAS VAR. SIZE RECS.
.READ INTO. ILLEGAL.
It is illegal for the READ
INTO statement to reference a
file which has multiple
record descriptions of
different lengths. Fatal.
- 733 FILE ACCESSED BY VERB
REQUIRING .LINAGE.
A file is accessed by an I/O
verb which did not have a
LINAGE clause in its
specification. Fatal.
- 734 .DELETE. OR .REWRITE.
WITHOUT INV. KEY OR USE.
A DELETE or REWRITE statement
references a file for which
there was no USE procedure
specified and for which the
INVALID KEY option was not
specified in that DELETE or
REWRITE statement. Fatal.
- 735 OPEN MODE OR NO READ
PROHIBITS REWRITE OR DELETE.
A DELETE or REWRITE statement
references a file which was
not OPENed in the proper mode
or which has no READ
statement referencing it in
the program. Fatal.
- 736 .START. CONFLICTS WITH OPEN
MODE.
A START statement references
a file which was not opened
in the proper mode. Fatal.
- 737 .WRITE. CONFLICTS WITH OPEN
MODE.
A WRITE statement references
a file which was not opened
in the proper mode. Fatal.
- 740 .READ. CONFLICTS WITH OPEN
MODE.
A READ statement references
a file which is only opened
in OUTPUT or EXTEND mode.
Fatal.

- 741 USE NOT IN DECLAR. OR NOT FOLLOWING SECTION NAME.
The USE statement is not in the DECLARATIVES section of the PROCEDURE DIVISION or is not immediately following a section name inside the DECLARATIVES. Fatal.
- 743 INTEGER IN SWITCH CLAUSE INVALID OR OMITTED.
A SWITCH clause of the SPECIAL-NAMES paragraph either contains an invalid numeric integer or has omitted the integer in its specification. A SWITCH clause integer, say n, must fall in the decimal range 1<n<=16. The SWITCH clause is ignored.
- 744 .IS. OMITTED IN SPECIALNAMES. ASSUMED PRESENT.
The required keyword is omitted in a clause of the SPECIAL-NAMES paragraph. It is assumed present and processing continues.
- 745 DEVICE MNEMONIC OMITTED IN SPECIALNAMES.
A valid device mnemonic-name is not recognized in one of the CONSOLE, LINE-PRINTER, CARD-READER, PAPER-TAPE-HEADER, or PAPER-TAPE-PUNCH clauses of the SPECIAL-NAMES paragraph. All source text is skipped up to the next recognizable keyword.
- 746 TERMINATOR OMITTED IN SPECIALNAMES.
The SPECIAL-NAMES paragraph is not terminated by a period. The period is assumed present and processing continues.
- 750 KEYWORD OMITTED IN .SWITCH. CLAUSE.
One of the keywords OFF or ON is omitted in the SWITCH clause of the SPECIAL-NAMES paragraph. The SWITCH clause is ignored.
- 751 CONDITION NAME MISSING IN .SWITCH. CLAUSE.
A valid condition-name is not recognized in the SWITCH clause of the SPECIAL-NAMES paragraph. The SWITCH clause is ignored.

- 752 .CP. OR .DP. NOT AT RIGHT
END OF PICTURE. The PICTURE symbol CP or DP
does not appear at the right
end of the PICTURE string.
The compiler ignores the
user-supplied PICTURE and
declares the data-name
alphanumeric with a "PICTURE
X" declaration.
- 753 .CP. OR .DP. USED WITH
SIGNED ITEM. Both the PICTURE symbols, CP
or DP, and a sign, + or -,
appear in the same PICTURE.
The compiler ignores the
user-supplied PICTURE and
declares the data-name
alphanumeric with a "PICTURE
X" declaration.
- 754 MULTIPLE DEFINITION OF
SWITCH. FIRST USED. Multiple definitions of a
COBOL switch are detected in
the SPECIAL-NAMES paragraph.
All but the first definition
of the SWITCH are ignored.
- 755 .SENTENCE. ASSUMED AFTER
.NEXT.. The keyword NEXT is not
followed by the keyword
SENTENCE. SENTENCE is
assumed present and
processing continues.
- 756 SUBSCRIPT NOT NUMERIC
INTEGER. A data-name used as a
subscript is not numeric in
class. A default value of 1
is assumed as the subscript.

D7700 LOGICAL UNIT NUMBER (LUN) ASSIGNMENTS

LUN	ASSIGNMENT
---	-----
1	Console, input
2	Console, output
3	work file
4	Listing File
5	Source file
6	Unused
7	Intermediate object file and load map listing
10	Object file
11	COPY library file
12	ACCEPT device I/O
13	DISPLAY device LUN
14-32	Punch-tape COBOL files

D7R02 PDP-11 COBOL FIELD RELEASE VERSION 1.02 RESTRICTIONS
-----D7R01 INTRODUCTION

The following is a list of restrictions for the Field Release Version 1.02 of the PDP-11 COBOL compiler. These restrictions represent areas of the COBOL language which we know that the compiler supports incorrectly or does not support at all at the time of the release of the Field Version 1.02 of the PDP-11 COBOL compiler. Therefore, the user is advised not to use these "restricted" language constructs as their results will be unpredictable. In all cases, the user can "program around" the restriction. As these restrictions are removed, the user will be notified via PATCH updates in the Software Bulletin.

D7R02 RESTRICTIONS

1. The user may not use the DAY option of the ACCEPT statement.

e.g., ACCEPT DAY=STRING FROM DAY.

There is not Julian date facility available in the operating system.

2. The user may not specify the ROUNDED phrase for the "rounding" of a numeric-edited GIVING operand in all arithmetic statements.

e.g., 1) MULTIPLY A BY B GIVING NUM-ED ROUNDED.

2) DIVIDE A BY B GIVING NUM-ED ROUNDED.

3. All procedure-names must contain an alphabetic character. Procedure names containing only numeric characters or numeric characters and hyphens are rejected. These conditions are diagnosed by the compiler.

4. #1 entries in the File Section of the Data Division may be described as elementary only if they are not

```
.alphanumeric edited class
.numeric edited class
.COMPUTATIONAL usage
.contain the SIGN clause.
```

These items may be described in the File Section with level numbers other than #1. Thus

```
#1 NUM=EDIT PIC Z(10).
```

may be described as

```
#1 RECORD=LEVEL.
#2 NUM=EDIT PIC Z(10).
```

This restriction is diagnosed by the compiler.

5. MOVE of a subscripted sending item to multiple receiving items where one of the receiving items other than the last is also one of the subscripts of the sending item. This delivers the wrong sending item to some receiving fields. Example:

```
MOVE A(I) TO B, I, C.
```

C will receive a different A(I) than B. The above statement will execute correctly if rewritten as

```
MOVE A(I) TO B, C, I.
```

6. No more than one file with LINAGE clause may appear in the same program, if reference is to be made to the LINAGE-COUNTER, since duplicate LINAGE-COUNTERS receive warning diagnostics and qualification of the LINAGE-COUNTER reference is not operational.

7. The SET statement delivers the wrong sending field value to multiple receiving fields under the following conditions: The SET of multiple receiving items TO an indexed sending field where one of the receiving items (other than the last) is also one of the indexes of the sending field. Example:

```
SET A, INDEX-NAME, B TO C(INDEX-NAME).
```

B will receive a different C(INDEX-NAME) than A.

The above statement will execute correctly if rewritten as

```
SET A,B,INDEX-NAME TO C(INDEX-NAME)
```

8. The user may not specify RA level alphanumeric literals as this may produce unpredictable results at object-time.
9. The compiler fails to diagnose the illegal MOVE from a numeric field to an alphabetic field whose PICTURE character string contains the editing character "H".

Example:

```

.
.
.
01 A PIC 9(4).
01 B PIC AABBA.

```

```

.
.
.
      MOV A TO B.
.
.
.

```

10. The compiler fails to issue a warning diagnostic in the case where the user mixes index-names and data-names in a subscript list used to reference a table item. Although the compiler does not warn the user of the violation, there is no difficulty at run time in correctly referencing the desired table element. The compiler simply fails to enforce a somewhat arbitrary rule of the ANS-COBOL standard.

Example:

```

.
.
.
01 I PIC 9(4).
01 TABLE.
   02 TBL-1 OCCURS 5 TIMES.
      03 TABLE-ITEM PIC 9(8) OCCURS 5 TIMES INDEXED BY J.

```

```

.
.
.
      MOVE I TO TABLE-ITEM (1,J).

```

DB200 ERROR LOGGING

Many device handler tasks developed for PSX-110 have the capability to pass error information to a set of error logging and analysis tasks. The operator can use the output from the error logging tasks to determine the reliability of devices attached to a system that runs PSX-110.

Error statistics are accumulated by handlers for the following devices:

1. Disks.
2. DECTape.
3. Magnetic tape.

The report produced by error logging can contain device-specific error statistics with a summary following it, or it can contain only the summary information. Additionally, the system manager can select the time frame that the report is to encompass and can indicate that the report is to include only those errors that happen on a specified device type, unit, or volume. The report for device-specific errors contains the following entries for each error logged.

1. Name of the device on which the error occurred.
2. Date and time of the error.
3. Error number since the system was loaded last or since the last power failure. Errors are numbered sequentially as they occur regardless of the device on which they occur.
4. Device mnemonic (handler name) and the unit number, e.g., DFW.
5. Volume label if present.
6. The UIC of the owner of the volume.
7. Device type and physical unit number; e.g., TU56 UNIT-1 for DECTape.
8. Contents of the device registers at the first occurrence of the error. The contents of the device registers on retries are not recorded. The device register names are the same as those used in the PDP11 Peripherals Handbook.
9. Number of retries performed.
10. Name of the task that issued the I/O request.
11. UIC under which the task was running.

12. Physical starting address of the task in memory.
13. Function issued. Both the name of the command to be performed and the actual value placed in the command register are provided.
14. Physical location in memory where the transfer occurred. The address is expressed in octal.
15. Actual transfer size in an octal number of bytes.
16. A count of I/O currently in progress for the task.
17. A count of other I/O requests that are queued for the task.
18. Error diagnosis. If the number of retries listed under RETRIES PERFORMED as described in item 9, above, is less than the number that the handler normally attempts during error recovery, the device error was not persistent. The comment under ERROR DIAGNOSIS is RECOVERED. If the number was greater, the error was not recoverable.
19. The total number of functions issued to this unit since a system load or power failure.
20. Vectors with active I/O. The number printed on the report is the vector that the device traps to upon completion.

Summary error logging information can be produced as a separate report or as the last portion of a device-specific report. The summary provides the following information.

1. The command used to request the report.
2. File specifications for the input and output file.
3. Date and time of the first and last entry in the file.
4. Number of errors missed.
5. Number of system power failures.
6. Number of reproducible and nonreproducible device parity errors.
7. Number of undefined system interrupts.
8. Number of system reloads.

Following the information described above are a series of entries providing the number of hard (nonrecoverable) and soft (recoverable) errors that occurred for each unit. When all entries are completed, the number of pages in the report is printed.

D8210 FRROR LOGGING FUNCTIONAL DESCRIPTION

Error logging consists of two distinct functions. The first function is the gathering of information pertinent to the errors that occur and the second is error analysis and the creating of a list file. These functions are performed by three tasks: ERRLOG, PSE, and SYE.

ERRLOG gathers volatile information when a device error occurs. It places this information in a temporary file named ERR.TMP under UFD (1,6) on the system device.

When a report of errors is desired, the preanalyzer and the analyzer tasks are run. When PSE starts, it sets an event flag to notify ERRLOG that it is ready to process the raw data file. ERRLOG renames the file ERR.TMP to ERROR.TMP and passes it to PSE. ERRLOG then creates a new ERR.TMP and continues logging errors. The preanalyzer (PSE) uses the information in ERROR.TMP to produce a formatted file name FRROR.SYS. FRROR.SYS is under UFD (1,6). When the analyzer (SYE) is run, it uses FRROR.SYS to produce a list file capable of being printed.

FRROR.SYS remains on the system disk until the system manager deletes it. Because it contains information that can be processed by user-written tasks to provide a report with different content, FRROR.SYS is not deleted automatically by the system.

D8220 OPERATIONAL INFORMATION

This section provides operating procedures for the three error logging and analysis tasks: ERRLOG, PSE, and SYE.

DP221 RUNNING FRPLOG

FRPLOG must be running in order for error statistics to be accumulated and for the raw statistics file to be passed to the reanalyzer (PSR). Normally, FRPLOG is installed during system generation. To run FRPLOG, type the following command to MCF and press ALTMODE.

MCF>RUN FRPLOG <ALT>

The task responds with the following message.

INPUT MINIMUM NUMBER OF ERRORS CAPABLE OF BEING
LOGGED IN A 5 SECOND PERIOD "CARRIAGE RETURN." THIS
VALUE SHOULD NOT EXCEED 5.
IF ERROR LOGGING NOT WANTED INPUT "CONTROL 7."
5 SECOND ERROR RATE = 3

-

At this point, type a value in the range from 1 through 5 and press RETURN. The value indicates the number of 72-word nodes to be assigned permanently to the error log task. The number of nodes allocated determines the number of errors that can be logged by FRPLOG.

If insufficient node space is allocated, the summary report contains an entry under the heading NUMBER OF ERRORS MISSED. This entry indicates the number of errors that were not logged due to insufficient node space. Because the device-specific report provides sequential numbers for errors, the user can determine at which point errors occurred, but were not logged.

If a large amount of node space is allocated, it may affect the ability of the task to acquire enough memory to run. The number of nodes required varies from installation to installation.

DR227 TERMINATION OF ERRLOG

The ERRLOG task terminates automatically in three cases:

1. When the desired number of nodes cannot be obtained.
2. If the error logging device used by ERRLOG becomes full.
3. If an error occurs when writing to the logging device.

If the system manager wishes to terminate, the ARP command, described in the PSX-11D User's Guide should be used.

The ERRLOG task terminates at task startup when the task cannot obtain the number of nodes specified by the user. When this situation occurs, the following message is printed on the console.

```
"ERRLOG" TASK FAILED TO PICK LARGE ENOUGH ERROR LOG
NODE BUFFER. "ERRLOG" TASK TERMINATED. IF YOU WANT TO TRY
AGAIN EXECUTE THE FOLLOWING SEQUENCE.
```

RUN ERRLOG

To attempt to run ERRLOG again, type the request to run ERRLOG again. When the request for the number of nodes is printed respond with a smaller number to 5 SECOND ERROR RATE =.

The second case that causes ERRLOG to terminate is when the error logging device becomes full. The following message is printed on the console.

```
ERROR LOGGING DEVICE device and unit number FULL. "ERRLOG"
TASK TERMINATED. IF YOU WISH TO CONTINUE LOGGING ERRORS
EXECUTE THE FOLLOWING SEQUENCE.
```

PFA ERRLOG 4 device and unit number
RUN ERRLOG

The PFA command is detailed in the PSX-11D User's Guide.

Prior to reassigning the loading device, the new device to which it is to be reassigned must be given a UFD of (1,6) if it does not have one already. The UFD must have the following access rights [RwED,RwED,RwED,RwED].

Type the following command to place the UFD on the disk.

UFD dev and unit number:UICs(1,6)/PROs(RwED,RwED,RwED,RwED)

The UFD command is detailed in the FSX-III User's Guide.

After creating the UFD, type the sequence provided in the console printout.

The third case in which FRRLOG terminates happens if an error occurs while trying to write to the logging device. The following information is printed on the console.

```
FRROR - XX ON ERROR LOGGING DEVICE device name  
"FRRLOG" TASK EXITING  
TASK "FRRLOG" TERMINATED  
VIA "EXIT" WITH PENDING I/O
```

XX is the standard system code as defined in the PSX-110 Device Handlers Reference Manual. If it is desirable to continue error logging, reassign the logging device as described previously in this section.

DB224 RUNNING PSF

The function of the preanalyzer, PSF, is to format the raw data collected by ERPLLOG into a file to be processed by SYE. In order to run PSE, ERPLLOG must be running and the user must be operating under UIC (1,1). Either log on under UIC (1,1) using the RPLLO function or use the SFT function with the /UIC switch.

To initiate PSF, type the following command to MCP.

```
MCP>PSE
---
```

The preanalyzer responds with the prompt PSF> and waits for the user to type a command line. The format of the PSF command line follows.

```
outdev:(uf1)file.ext=indev:
```

The output file specification is a standard PSX-110 specification except that the version number is omitted.

The input file specification consists only of the input device specification. The file name always is ERROP.TMP; the name is assigned by ERPLLOG.

The following defaults are used for omitted portions of the file specifications.

```
outdev  defaults to SY:
uf1     defaults to (1,6)
file.ext defaults to ERROP.SYS
indev   defaults to SY:
```

If the default values are to be used, press RETURN in response to the PSE prompt.

DB334 RUNNING SYF

The analyzer produces an error report in the form of a listing file. The system manager can either queue the file for printing or use PLS to list it.

Before SYF can run, the user must be operating under 11,11 and SYF must be installed. Type the following command to install SYF.

```
MCP>INS (11,1)SYF
-----
```

Once SYF is installed, type the following command to run it.

```
MCP>SYF
---
```

The analyzer responds with the following prompt SYF> and waits for the user to type a command line. The format of the SYF command line follows.

```
outdev:[ufd]xxxyyy.LST=indev:[ufd]file.ext/switch1.../switchn
```

The output file specification is identical with the standard RSX-11i file specification with the following two exceptions.

1. The filename (xxxyyy) must correspond to the values specified for xxx and yyy for the /RR: switch described below.
2. The version number of the file is not specified.

The input file specification is a standard RSX-11D file specification except that the version number is not included. The input device, UFD, filename, and extension must be identical with the output file specification used with running PSF. It is the output of PSF that SYF analyzes.

The following switches can be used as part of the input file specification.

/BP:xxxxyy is the breakout switch that determines what information is to be included in the report.

xxx can have one of the following values:

ALL indicates that error statistics for all disk, magnetic tape, and DfCtape units are to be included.

DSK indicates that error statistics for all disk units are to be included.

MAG indicates that error statistics for all tape devices, both magnetic tape and DfCtape, are to be included.

ALL is the default value for xxx.

vvv can have one of the following values:

ALL indicates that both the device-specific and the summary information is to be included in the report.

SUM indicates that only the summary information is to be included.

/ID:name indicates that the report of errors is to contain only those errors that occurred while a specified volume is mounted. The value name provides the volume identification. Where can user find out what id for a particular volume is?

/DV:devn indicates that the report is to contain only those errors that occur on a specified device type or on a specified unit. For example if dev is specified as DA, error statistics for all RKA3 or RKA5 units are provided. If devn is specified as DA1, error statistics for RKA3 or RKA5 unit 1 are provided.

/RG:time:date indicates that only those errors that occur after the specified time and date are to be included in the report. The format of the time and date specification follows.

hh:mm:ss:tt:dd:yy

All numbers are decimal.

/FD:time:date indicates that only those errors that occurred on or before the specified time and date are to be included in the report. Time and date have the same format as in the /RG switch.

The following are the default values for the SYF command string.

SYF:(user UIC)ALLSUM,LST-SYN:(1,6)ERROR,SYS/MP:ALLSUM

DB230 ERROR MESSAGES

The preanalyzer (PSE) and the analyzer (SYF) both issue error messages to inform the user of operational difficulties.

DB231 PSE ERROR MESSAGES

COMMAND STRING PARSE ERROR

A syntax or semantic error was encountered while examining the input command string to PSE. PSE prompts again for a new command line.

Type a corrected version of the command line.

DELETE ERROR

When the preanalyzer was through processing the input file ERROR.TMP, it was unable to delete it.

Use PIP to delete the file.

INPUT FILE ERROR

An error was encountered while trying to open or obtain data from the input file ERROR.TMP. ERR.TMP is closed, processing is terminated, and the input file is not deleted.

Try running PSE again. If this fails, delete the file.

NO ERROR FILES FROM SYSTEM

The preanalyzer is unable to locate a file named ERROR.TMP. This message can be caused by either of the following situations.

1. No errors have occurred. Therefore ERPILOG has no raw data file to pass to PSE.
2. ERPILOG is not running and, therefore, cannot rename the ERR.TMP file to ERROR.TMP and pass it to PSE.

If the cause of the message is that ERPILOG is not running, follow the procedures in Section DA221 to run the task.

OUTPUT FILE ERROR

An error was encountered while working with the PSF output file. Both the input and output files are closed. PPROR.TMP is not deleted.

Try to rerun PSE.

PSE-ANALYZER OUTPUT DEVICE FULL

The output device became full while PSE was writing data to the output file. Both the input and output files are closed. PPROR.TMP is not deleted.

Rerun PSE using a different output volume.

UNABLE TO CLOSE INPUT FILE

PSF is unable to close the file PPROR.TMP. The file is not deleted. File processing is terminated.

Use PIP to delete the file.

UNABLE TO CLOSE OUTPUT FILE

PSF is unable to close the output file.

Rerun PSE or use PIP to delete the file.

DB232 SYE ERROR MESSAGES
.....

INPUT DEVICE ERROR
FATAL ERROR = n

SYE was attempting to obtain further information from the input file but could not get the next record. n is an FCS error code. Refer to the RSX-11 I/O Operations Reference Manual to determine the meaning of n.

Both the input and output files are closed. SYE issues a prompt for the next command.

OUTPUT DEVICE ERROR
FATAL ERROR = n

SYE was unable to write information in the output file. No files remain open. n is an FCS error code. Refer to the RSX-11 I/O Operations Reference Manual to determine the meaning of n.

Both the input and output files are closed. SYE issues a prompt for the next command.

SYE COMMAND STRING ERROR
portion of the string in error

The format conventions within a particular portion of the command string is violated. No files remain open. SYE issues a prompt.

Correct the error and type the command.

SYE COMMAND STRING ERROR
ERROR NUMBER n

The command string interpreter detects an error while attempting to get a command line. n is a CSI error code. Refer to the RSX-11 I/O Operations Reference Manual to determine the meaning of n. No files remain open. SYE issues a prompt.

Correct the error and type the command.

SYE COMMAND STRING SYNTAX ERROR
COMMAND STRING TYPED

The proper format was not used in the command string. No files are open. SYE issues a prompt.

Type the corrected command.

SYE ILLEGAL BREAKOUT SWITCH
/RXXXXXXXXXX

SYE issues this message when the operator attempts to request a breakout of the input file that is not legal. No files remain open. SYE issues a prompt for another command.

Retype the command with a correct use of /R:.

SYE - OVERLAY FAULT - ERROR - n

SYE failed in an attempt to load the message overlay. The input and output files remain open. n is an FCS error code. Refer to the RSX-11 I/O Operations Reference Manual to determine the meaning of n.

UNABLE TO OPEN DESIRED FILES
FATAL ERROR - n

SYE was unable to open either the input or the output file. No files remain open. n is an FCS error code. Refer to the RSX-11 I/O Operations Reference Manual to determine the meaning of n.