CRAY RESEARCH, INC.

CRAY X-MP AND CRAY-1® COMPUTER SYSTEMS

CRAY-OS
MESSAGE MANUAL

SR-0039

RECORD OF REVISION

CRAY RESEARCH, INC.

PUBLICATION NUMBER SR-0039

Each time this manual is revised and reprinted, all changes issued against the previous version in the form of change packets are incorporated into the new version and the new version is assigned an alphabetic level. Between reprints, changes may be issued against the current version in the form of change packets. Each change packet is assigned a numeric designator, starting with 01 for the first change packet of each revision level.

Every page changed by a reprint or by a change packet has the revision level and change packet number in the lower righthand corner. Changes to part of a page are noted by a change bar along the margin of the page. A change bar in the margin opposite the page number indicates that the entire page is new; a dot in the same place indicates that information has been moved from one page to another, but has not otherwise changed.

Requests for copies of Cray Research, Inc. publications and comments about these publications should be directed to:

CRAY RESEARCH, INC.,
1440 Northland Drive,
Mendota Heights, Minnesota 55120

Revision	Description
	October, 1980 - Original printing.
01	July, 1981 - This change packet brings the manual into agreement with version 1.10 of the released software. Major changes include changing the SKOL messages from uncoded to coded messages and adding Exchange Processor (EXP) messages, Control Statement Processor (CSP) messages, $SYSLIB messages, MODSEQ messages, MODSET messages, SYSREF messages, Station Call Processor (SCP) messages, and EXEC Message Processor (MEP) messages. Section 1 has been rewritten and technical changes are noted by change bars. Miscellaneous technical and editorial changes are also included.
02	May, 1982 - This change packet brings the manual into agreement with version 1.11 of the released software. Major additions include Exchange Processor (EXP) messages, EXTRACT messages, ITEMIZE messages, LDR messages, MODSET messages, Station Call Processor (SCP) messages, Symbolic Interactive Debugger (SID) messages, $SYSLIB messages, Interactive Line Text Editor (TEDI) messages, UPDATE messages, and CFT messages. Miscellaneous technical and editorial changes are also included.
A	July, 1983 - This rewrite brings the manual into agreement with version 1.12 of the released software. Message descriptions are in a new 2-column format, and message classes are explained. New messages cover the Tape Management System, PDM privacy, CAL, APML, Startup error recovery, Extract, Fast Secondary Storage, Pascal, SEGLDR, and memory management. Most FTLIB messages are now issued by ARLIB, UTLIB, and IOLIB, with appropriate prefixes.
B	February, 1984 - This rewrite brings the manual into agreement with version 1.13 of the released software and obsoletes all previous editions of the manual.

C December, 1984 - This rewrite brings the manual into agreement with version 1.14 of the released software and obsoletes previous editions of the manual. Software Tools messages have been added. Startup messages are consolidated in a new subsection. Miscellaneous technical and editorial changes are also included.

PREFACE

The CRAY-OS Message Manual describes all user logfile, system logfile, and program output listing messages issued by CRAY-OS software. The manual is divided into two sections: coded messages and uncoded messages. The coded messages are alphabetized by the 2-letter code prefix. The uncoded messages are further divided to facilitate locating particular messages.

Each message is accompanied by the cause of the message, the action to be taken (if any), the source of the message, and the class (reprieve information) of the message.

CLASS

Message classes are designated according to severity and reprieve class. These are described at the beginning of section 1. Some groups of messages use different systems of classes. Explanations of these systems appear at the beginning of the appropriate subsections.

CAUSE AND ACTION

The cause of the problem and the appropriate action to take are described in the paragraph below a message. The division between cause and action is not always distinct, but the action is normally expressed as an imperative sentence.

Generally the cause is an amplification of what the message says. Some messages explain the cause adequately and have no further explanation.

The appropriate action usually consists of correcting the stated problem and rerunning. If the means of correcting the problem needs no explanation, the action is omitted. When no action is needed, such as on many informative messages, none is given.

SOURCE

Message sources can be COS routines, language processors, or programs such as CSIM. Products that issue messages are documented in the following Cray Research Inc., publications:

SR-0000	CAL Assembler Version 1 Reference Manual
SR-0009	FORTRAN (CFT) Reference Manual
SR-0011	CRAY-OS Version 1 Reference Manual
SR-0014	Library Reference Manual
SR-0066	Segment Loader (SEGLDR) Reference Manual
SR-0033	SKOL Reference Manual
SG-0055	Text Editor (TEDI) User's Guide
SG-0056	Symbolic Interactive Debugger (SID) User's Guide
SR-0073	COS Simulator (CSIM) Reference Manual

CONTENTS

CODED MESSAGES 1

Message classes indicate how serious a problem is, and whether it can be reprieved. They are shown in two separate systems, severity class and reprieve class. (Some sources of uncoded messages use their own set of classes, as explained in their respective subsections.)

Severity is indicated by the following three categories:

- *Fatal* indicates that execution of the program is stopped. The job step is aborted unless the error is reprievable and reprieve processing is requested. A fatal error can be reprieved only once per job step.
- *Caution* indicates a potential problem. An error in the caution class can be reprieved as many times as necessary.
- *Informative* indicates that program execution did not stop. Some informative messages indicate normal operation; others report situations which, though abnormal, do not require aborting.

Reprieve classes are indicated by numbers followed by short explanations. Normally, when a job step abort error occurs, control passes to the EXIT control statement and exit processing begins. Reprieve processing allows the user to attempt recovery from many job step abort errors or to perform clean-up functions before continuing with the abort.

Reprieve processing can also be used during normal termination of a job step. In this case, control transfers to the user's reprieve code instead of to the next normal job step.

When requesting reprieve processing, the user selects the error conditions to be reprieved by setting a mask in the SETRPV subroutine or macro call. Each number used in this mask is an octal code which represents the setting of 1 bit. If more than 1 bit is set, the numbers are added. To disable reprieve processing no bits are set. The numbers for reprieve processing are interpreted as follows:

1	Normal job step termination
2	User abort. This can be called by the user's program or any library used by the user's program.
4	System abort
10	Operator DROP
20	Operator RERUN
40	Memory error
100	Floating-point error
200	Time limit
400	Mass storage limit exceeded
1000	Memory limit exceeded
2000	Link transfer error
4000	Security violation
10000	Interactive console 'attention' interrupt

If a selected error condition occurs during job processing, the user's current job step maintains control. The user's Exchange Package, Vector Mask register, error code, and error class are saved and control passes to the user's reprieve code.

AB000 - JOB STEP ABORTED. P=*xxxxxx*.
The message preceding AB000 gives the cause of abort. AB000 supplies the user field address where the job step abort occurred. Use the P address supplied to debug the user program. From EXP. Class, informative.

AB001 - END-OF-FILE ON READ
End of file was reached on the control statement file and CSP tried to read another record from $CS. Forward a system dump of the error to a Cray Research analyst. From EXP. Class, 4; not reprievable.

AB002 - INVALID LOCK OR UNLOCK INDICATION
An F$IOA request (lock or unlock) was issued and the value specified by S1 was not a valid lock or unlock request code. Probable user error. Review the description of the F$IOA call in the CRAY-OS Version 1 Reference Manual, publication SR-0011. If the program specified the call correctly, obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB003 - DEVICE ALLOCATION TABLE SPACE EXHAUSTED
DAT space is not available in the system DAT area. Determine why the DATs are full. If a front-end system is down, DAT space may be freed as soon as the front

end successfully logs on. If too many system resources, such as SDR datasets, are using DAT space, delete the unused datasets. If the error occurs frequently, reassemble the system with a larger DAT area specified in the STP constant. From EXP. Class, fatal.

AB004 - DATASET NOT OPEN
A buffered I/O request was issued for a dataset that was not opened. Probable user error. Ensure that an F$OPN request is being issued before the first I/O request. If the F$OPN is being issued, obtain a dump of the job and forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB005 - INVALID OPEN
One of the following conditions was detected:

- The type of open request specified was invalid.
- The dataset is already open. See the Macros and Opdefs Reference Manual, CRI publication SR-0012.

Check the F$OPN request type for validity. If the dataset is opened more than once, a CLOSE request might have to be issued after each OPEN. If the request is being issued correctly, obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB006 - NO READ PERMISSION
An attempt was made to read a dataset and the dataset was not accessed with read permission. Probable user error. Specify the read permission control word when the dataset is accessed. If the read permission control word was specified correctly, obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB007 - NO WRITE PERMISSION
An attempt was made to write a dataset that had not been accessed with write permission and/or did not have unique access. Specify the write permission control word and/or unique access when the dataset is accessed. If the dataset belongs to another user, it may be necessary for that user to use a PERMIT statement or to change the dataset Public Access Mode setting to allow you to obtain write permission. From EXP. Class, 4; reprievable.

AB008 - ILLEGAL BITS SET IN MEMORY REQUEST WORD
The system received a user request for a system-privileged memory operation. If there is an error in the memory request being issued by the program, correct the error and resubmit the request. If there is no apparent error, obtain a dump of the job at the time of the request and forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB009 - ATTEMPT TO DELETE MEMORY OUTSIDE PROGRAM AREA
The system received a user request to delete memory. The specified first word address plus the specified length define an area beyond the user field. If the memory delete request is incorrect, modify the request and resubmit the job. Otherwise, obtain a dump of the job at the time of the request and forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB010 - NO AVAILABLE DISK SPACE
The system has no more disk space to allocate to the dataset being written. If there is a shortage of disk space, delete any datasets that are no longer required; then rerun the job. If there is not a shortage of disk space, obtain a system dump and forward the dump to a Cray Research analyst. From EXP. Class, fatal.

AB011 - SYSTEM DIRECTORY IS FULL
An ACCESS ENTER request was being processed for a dataset and the resident System Directory Table area is full. If there are entries in the system directory that are no longer needed, clear the system directory by performing a restart of COS and by specifying *SDR in the parameter file. Then run a JSYSDIR job, specifying only the datasets to be entered into the SDR. If all entries in the SDR are to be maintained, the value of NE@SDR in the STP system tables must be increased. COS must be reassembled and restarted to increase the number of resident SDR entries. If these two conditions do not exist, obtain a system dump and forward it to a Cray Research analyst. From EXP. Class, fatal.

AB012 - JTA OVERFLOW
The job tried to expand its JTA and was already at its maximum memory size. From EXP. Class, 4.

AB013 - MORE MEMORY REQUESTED THAN AVAILABLE
A memory allocation request was issued for more memory than is available in the machine. Determine whether the request is valid. If not, correct the program and resubmit the job. If it is valid, submit a dump to a Cray Research analyst. From EXP. Class, 4.

AB014 - MORE MEMORY REQUESTED THAN ALLOWED
A memory allocation request was issued for more memory than allowed by the maximum field length value supplied by the job statement MFL parameter or the

installation parameter I@MAXFL. Correct the memory request and resubmit the job. From EXP. Class, 4.

AB015 - ACCESS BEFORE ACQUIRE FAILED
During ACQUIRE processing, an ACCESS request failed. The previous message (a PDM message) should explain the failure. If the request appears correct, obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, 2000; reprievable.

AB016 - SUBDATASET, $IN, CANNOT BE DISPOSED
The system detected a request to DISPOSE a file named $IN. This request is invalid. If a job control statement or user program DISPOSE (F$DIS) request is in error, correct the request statement and resubmit the job. If there is no error in the request statement, forward a dump of the job to a Cray Research analyst. From EXP. Class, 2000; reprievable.

AB017 - INVALID CLOSE REQUEST
The ODN specified by the CLOSE request pointed to a DSP that should be in the user field, but the address is not below the value in JCHLM. Determine whether the address in the ODN is valid. If not a user error, obtain a dump of the job (JTA, ODN, DSPs) and forward these to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB018 - DATASET ALREADY OPENED
An F$DNT request was issued for a dataset that was already opened. Probable user error. Issue a CLOSE request before a second F$DNT request for a dataset is issued. If CLOSE is being issued, obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB019 - JOB COMMUNICATION BLOCK DESTROYED
One of the following conditions was detected:

- One of three words in the JCB has been destroyed - the word containing HLM, the word containing the base address of I/O buffers, or the word containing the base address of DSP or base address of LFTs.
- HLM is less than the length of the JCB.
- HLM is greater than the base of the LFTs.
- LFT area overlays DSP area.
- Calculated LFT area + DSP area overlays buffer area.
- The base address of I/O buffers exceeds the field length.

If an error in the program is causing the JCB to be overwritten, correct the program and resubmit the job. If there is no error in the program, obtain a dump of the job and forward the dump to a Cray Research analyst. From EXP. Class, fatal.

AB020 - INVALID SYSTEM REQUEST PARAMETER
An EXP system call routine received an invalid parameter, generally an address outside the user field. Probable user error. Obtain a dump of the job and verify the addresses or values passed to the system. If there is no apparent user error, forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB021 - DATASET NOT FOUND
A system routine required an existing DNT for processing the specified dataset but was unable to locate the DNT or a dataset not local to the job or known by the system was specified to be loaded. Obtain a dump of the job and verify that the dataset name being passed to the system is valid. If the name is correct, forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB022 - WRONG BUFFER LENGTH ON OPEN REQUEST
The user has tried to open a user-area system-managed DSP, and the buffer size specified in the DSP does not match that in the DNT. From EXP. Class, 4.

AB023 - JOB TIME LIMIT EXCEEDED
The job has exceeded the specified or default time limit. First, determine that the program was not looping. If execution was proper, raise the time limit value on the job statement and resubmit the job. If an operating system problem is suspected, obtain a system dump and forward the dump to a Cray Research analyst. From EXP. Class, fatal.

AB024 - OPERATOR DROPPED THE JOB
The system or terminal operator issued a DROP command for the specified job. Investigate the cause of the operator action. If an operating system problem is suspected, obtain a system dump and forward the dump to a Cray Research analyst. From EXP. Class, fatal.

AB025 - USER PROGRAM REQUESTED ABORT
This message normally appears after a message explaining why an abort occurred. The user program (which can include a control statement in this context) issued a request to abort the job step. Correct the condition indicated by the preceding message. From EXP. Class, 2; reprievable.

AB026 - INVALID (UNDEFINED) USER REQUEST
An invalid system call request was issued. Probable user error. Obtain a

dump of the job and validate the registers used for the system call. If the request was valid, forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB027 - CALL NOT BETWEEN USER BA AND LA
The system received a request for which the required address parameters were not valid. Probable user errors. Obtain a dump of the job and examine the registers used for the system call. If the addresses are correct, forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB029 - LOGICAL DEVICE NOT FOUND OR UNAVAILABLE
The specified logical device name could not be found in an existing equipment table or the device is inoperative. See a Cray analyst or site operations personnel to determine valid device names. If the device name was specified incorrectly, correct the control statement and resubmit the job. From EXP. Class, 4; reprievable.

AB030 - BLOCK NUMBER ERROR
The system I/O routines detected that the block control word of the record that was just read is not the expected block number. This is probably caused by reading a dataset which was created as an unblocked dataset as if it were in blocked format. Check the creation format of the dataset. If the dataset was created in blocked format, obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB031 - UNRECOVERABLE DATA ERROR
The physical I/O device driver/controller detected a discrepancy in the checksum during an I/O operation. See the system operator. The operator should dump the system log and determine if other datasets on the same device are receiving similar error messages. If so, a device could be failing. If it is not a common error and if the problem persists for the dataset, the dataset must be recreated. From EXP. Class, 4; reprievable.

AB032 - UNRECOVERABLE HARDWARE ERROR
The physical I/O device driver/controller detected a permanent hardware error and was unable to complete the I/O operation. Report the error to the system operator. The operator should check the status of the device. If the device was not ready, ready it and resubmit the job. If the device appears to be operational, notify a Cray Research analyst. From EXP. Class, 4; reprievable.

AB033 - READ AFTER WRITE OR PAST END-OF-DATA
An invalid I/O operation was attempted; either a read operation was issued after a write operation or a read request was issued after end of data (EOD) was detected by the system. In the first case, determine if the correct sequence of I/O operations is being issued. A read operation can follow a write operation without an intervening REWIND or BACKSPACE only when the access mode is random. In the second case, determine if EOD is being tested according to the documentation for the request being issued. If the message is being produced without apparent cause, obtain a dump of the job at the time of failure and forward it to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB034 - UNKNOWN ERROR
An unexpected or unrecoqnizable error has occurred. Obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB035 - INVALID PROCESSING DIRECTION
An I/O request was issued but the dataset was not open for the direction requested. Obtain a dump of the job. If the requested processing direction was valid, forward the dump to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB036 - DATASET PREMATURELY TERMINATED
A blocked read request was issued for a dataset. There were no more blocks to be read, but an end-of-data control word was not detected. Obtain a dump of the job at the time of the failure and forward them to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB037 - DATASET PARAMETER TABLE INVALID
One of the following conditions was detected:

- An I/O request was received by the system and the DSP was busy.
- One of the buffer pointers in the DSP was invalid.
- The record control word address in the DSP was invalid.
- A dataset was being closed and the buffer address in the DSP was in the user field. This condition is legal for an unblocked dataset, and when the DSP is also in the user field.
- The dataset name in the DSP has been cleared, or the user has overwritten the DSP.
- The I/O operation to be performed on the dataset is invalid for the status of the dataset.

If the user program is checking active I/O operations correctly and the DSP pointers look valid, forward a dump of the JTA, DSP, and buffers to a Cray

Research analyst. Otherwise, correct the program and resubmit the job. From EXP. Class, 4.

AB038 - OPERATOR KILLED THE JOB
Execution of the job was terminated by the operator. From EXP. Class, fatal.

AB039 - OPERATOR RERAN THE JOB
The operator restarted the job from the beginning. From EXP. Class, 20; reprievable.

AB040 - INVALID DISPOSITION CODE
The disposition code (DC) of the user request or JCL statement is not specified as documented in the CRAY-OS Version 1 Reference Manual, publication SR-0011. If the disposition code is incorrect, correct it and resubmit the job. If the disposition code is correct, forward the job output to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB041 - SYSTEM DSPS (ABOVE JCHLM) DESTROYED
The system-area DSPs (between JCDSP and JCBFB) have been corrupted. From EXP. Class 4.

AB042 - USER DSPS (BELOW JCHLM) DESTROYED
The user-area DSPs (pointed to by user-area LFTs) have been corrupted, or the user-area LFTs have been corrupted such that they no longer point to the user-area DSPs. From EXP. Class 4.

AB043 - ALLOWABLE USER LOG SIZE EXCEEDED
The size of the logfile for the job has exceeded the limit (I@LGUSZ). The job is writing log records in a loop, or the job is issuing more requests than are allowed. In the first case, correct the error causing the loop, and resubmit the job. In the second case, a Cray Research analyst should increase the value of I@LGUSZ. If there is no apparent cause of the failure, forward the job output to a Cray Research analyst. From EXP. Class, fatal.

AB044 - INVALID DATASET NAME
The dataset name in the control statement or other system request contains an invalid character or is longer than seven characters. Correct the dataset name and resubmit the job. If the name is valid, forward the job output to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB045 - SPECIFIED LIMIT OR SIZE IS TOO BIG
An attempt was made to open a dataset by specifying a limit or size greater than the allowable size for the dataset. Adjust the dataset size limit to be within the system specification. If a larger system specification is required, a Cray Research analyst should change the system parameter I@MAXLM or I@$OMLM in COSTXT and regenerate the system. From EXP. Class, 400.

AB046 - DATASET SIZE LIMIT EXCEEDED
The last write operation caused the file to exceed the size specified in the Dataset Name Table or to exceed the maximum size allowed by the site. Verify that the program was not in a loop. If the dataset size needs to be larger, use an ASSIGN control statement with the specified LM parameter. From EXP. Class, 400.

AB047 - DATASET NOT AVAILABLE FROM STATION
An ACQUIRE or FETCH control statement was issued for a dataset that could not be staged, or the front end where the ACQUIRE was issued does not support that function. If ACQUIRE is supported by the front end, ensure that the dataset exists. Resubmit the job. From EXP. Class, 2000.

AB048 - DATASET CAN'T BE SAVED ON FRONT-END
A DISPOSE or SUBMIT request was issued but the dataset could not be saved by the front end. Determine why the dataset could not be saved (for example, lack of space). Request assistance from a front-end system programmer if necessary. Resubmit the job. From EXP. Class, 2000.

AB049 - SYSTEM LFTS (ABOVE JCHLM) DESTROYED
The system-area LFTs have been corrupted such that they no longer match the copies saved in the JTA. From EXP. Class 4.

AB050 - USER LFTS (BELOW JCHLM) DESTROYED
The user-area LFTs have been corrupted such that they no longer match the copies saved in the JTA. From EXP. Class 4.

AB051 - INVALID POINTER TO FIRST JTA LFT
The system detected a zero value for the JTA pointer to the first Logical File Table. Obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, fatal.

AB052 - NO USER LFT DN MATCH IN JTA LFT'S
The system was searching the JTA Logical File Table and could not locate the dataset name specified in the search argument. Obtain a dump of the job. Forward all documentation to a Cray Research analyst. From EXP. Class, 4; reprievable.

AB053 - FLOATING POINT ERROR
The CPU detected a floating-point underflow or overflow while executing an instruction. From EXP. Class, 100.

AB054 - OPERAND RANGE ERROR
An attempt was made to load or store a register outside the user field. From EXP. Class, 4; reprievable.

AB055 - PROGRAM RANGE ERROR
An attempt was made to load an instruction from outside the user field. From EXP. Class, 4; reprievable.

AB056 - UNCORRECTED MEMORY ERROR
An uncorrectable memory error occurred while the job was executing. Resubmit the job. If the error continues to occur, notify the operator. From EXP. Class, 40.

AB057 - CONSOLE INTERRUPT
An abort command was entered at an interactive terminal, causing the executing program to stop. From EXP. Class, informative; not reprievable.

AB058 - ERROR EXIT
A zero instruction was executed in the user area. From EXP. Class, fatal.

AB059 - DISPOSE FAILED
PDM returned an error status on the save of the output dataset. Check the PDM message immediately preceding this one to see why the save failed and correct the DISPOSE if necessary. From EXP. Class, 4.

AB060 - NOT ENOUGH MEMORY TO LOAD CSP
The maximum field length for the job minus the length of the JTA and the number of LFTs/DSPs in buffers has not left enough room in the job for CSP. From EXP. Class, 4.

AB061 - NO INVOKE REQUEST WAS PROVIDED
A call was made to F$INV without the required information. From EXP. Class, 4; reprievable.

AB062 - INVOKE REQUEST ALREADY PENDING
An invoke request has already been sent and another is waiting. The second or subsequent requests will not be processed. From EXP. Class, 4; reprievable.

AB063 - INVOKE LEN NOT A MULTIPLE OF O'1000
The job class structure length is not a multiple of 1000 octal. Recreate the job class structure. From EXP. Class, 4; reprievable.

AB064 - INVOKE LEN GREATER THAN MAX ALLOWED
The length of the job class structure is greater than the maximum length allowed. Recreate the job class structure or change the installation parameter. From EXP. Class, 4.

AB066 - DATASET HAS RELATED DISPOSES/SUBMITS ACTIVE
The job tried to write to a dataset that had outstanding disposes or submits against it. Restructure the job so that it does not reuse the same local dataset name; or, if the DISPOSE control statement was used, delay the job with DISPOSE, WAIT. From EXP. Class, 4.

AB067 - INVALID PROCEDURE DATASET
Procedure datasets cannot reside in memory or be unblocked. Correct the ASSIGN statement for the procedure dataset and resubmit the job. From EXP. Class, 4.

AB068 - PROCEDURE NEST LEVEL EXCEEDED
Procedures that invoke other procedures have exceeded the limit of I@PRLVL-1. From EXP. Class, 4; reprievable.

AB069 - PDS FULL: DELAY COUNT EXCEEDED
PDS-full status was returned too many times for a job-initiated PDM call. The job was aborted. Resubmit, and release datasets when they are no longer needed. From EXP. Class, 4.

AB070 - CONSOLE ATTENTION REQUEST
An ATTENTION command was entered at an interactive terminal. From EXP. Class, 10000; reprievable.

AB071 - BAD CLASS STRUCTURE
An attempt was made to invoke an invalid class structure. Redefine the class structure and rerun JCSDEF. From EXP. Class, 4; reprievable.

AB072 - DSP DESTROYED BY USER
One of the following conditions was detected:

- An I/O request was received by the system and the DSP was busy.
- One of the buffer pointers in the DSP was invalid.
- The record control word address in the DSP was invalid.
- A dataset was being closed and the buffer address in the DSP was in the user field.
- The dataset name in the DSP has been cleared; the user has overlayed the DSP.
- The I/O operation to be performed on the dataset is not valid for the status of the dataset.

If the user program is checking active I/O operations correctly and the DSP pointers look valid, forward a dump of the JTA, DSP, and buffers to a Cray Research analyst. Otherwise, correct the program and resubmit the job. From EXP. Class, 4.

AB073 - UNDEFINED FUNCTION CODE IN F$INS
The system function F$INS was called with a parameter (subfunction code) outside

the limits of the defined subfunction table. F$INS provides for installation-defined subfunctions. The table is initially empty. Thus, a call to F$INS made with an empty subfunction table results in this error regardless of the subfunction code. Check the call of the F$INS function (or the INSFUN macro) to ensure that the subfunction code is being passed properly. If it is being passed properly, consult site operations personnel to obtain the current valid subfunctions. From EXP. Class, 4; reprievable.

AB074 - DUMPJOB PROCESSING INHIBITED
A DUMPJOB or F$DJA was attempted while an execute-only dataset was loaded in the user field. Remove the call to DUMPJOB when an execute-only dataset might be loaded. From EXP. Class, 4000; reprievable.

AB075 - NO PERMISSIONS GRANTED FOR EXECUTE-ONLY DATASET
An attempt was made to directly open, dump, dispose, or otherwise modify an execute-only dataset. Change the job so all references and modifications to the dataset are made through the Control Statement Processor (CSP). From EXP. Class, 4000; reprievable.

AB076 - DATASET IS ALREADY ACCESSED BY THE JOB
An attempt was made to access a dataset in the System Directory Table (SDR) with a dataset name that is already local to the job. Change the job so the dataset is not local at the time of the F$ASD call. From EXP. Class, 4; reprievable.

AB077 - INTERNAL CONTROL STATEMENT PROCESSOR (CSP) ERROR
COS detected an error in a request and/or stored information maintained by CSP. Obtain a dump of the job and forward it to a Cray Research analyst. From EXP. Class, not reprievable.

AB078 - PRIVILEGED SYSTEM REQUEST
The user program made a system request (F$*xxx*) that is restricted to privileged processes. Ensure that the request is the one desired and is in the correct format. If there is a valid need to execute this function, contact the site operations manager. From EXP. Class, 4000; reprievable.

AB079 - REFERENCE TO UNASSIGNED JCL SYMBOL
An attempt has been made to use the value of a JCL symbol that has not been given a value. Ensure that the assigned symbol is given a value before being used within a JCL expression. From EXP. Class, 4; reprievable.

AB080 - RECEIVE BUFFER TOO SMALL
The user-supplied table for a system request is too small to receive the requested data. This can occur with F$LIB and F$SYM requests. Ensure that enough area is allocated for the requested data. From EXP. Class, 4; reprievable.

AB081 - UNDEFINED JCL SYMBOL REFERENCED
An attempt was made to get or set the value for an unknown JCL symbol. See the CRAY-OS Version 1 Reference Manual, publication SR-0011, to ensure that the symbol name is correct. From EXP. Class, 4; reprievable.

AB082 - JCL SYMBOL CANNOT BE MODIFIED BY THE JOB
An attempt was made to modify the value of a symbol that is either a system constant symbol or a symbol that can be set only by COS. See the CRAY-OS Version 1 Reference Manual, publication SR-0011, to check the use of the symbol in question. From EXP. Class, 4; reprievable.

AB083 - INVALID MESSAGE CLASS
A F$MSG call was made in which the message class was out of the valid range. See the MESSAGE macro description in the Macros and Opdefs Reference Manual, CRI publication SR-0012, for valid message classes. From EXP. Class, 4; reprievable.

AB085 - DISPOSE TO CRAY INPUT QUEUE NOT ALLOWED
The DISPOSE control statement was used to place a job into the Cray mainframe input queue. Use the SUBMIT control statement to place jobs into the Cray mainframe input queue. From EXP. Class, 4; reprievable.

AB086 - BUFFER SIZE INVALID WITH UNBLOCKED FORMAT
A dataset existed with unblocked format, and an F$DNT call was made with an attempt to assign a buffer size. Remove the buffer size parameter. From EXP. Class, 4; reprievable.

AB087 - FIELD CAN'T BE CHANGED FOR OPEN DATASET
An F$DNT call was made on an open dataset with a field other than DDDN, DDLM, or DDDC. Correct the invalid field. From EXP. Class, 4; reprievable.

AB088 - FIELD CAN'T BE CHANGED FOR CLOSED DATASET
An F$DNT call was made on a closed dataset with DDLDV or DDSZ set. Correct the invalid field. From EXP. Class, 4; reprievable.

AB089 - INTERACTIVE I/O REQUEST TOO LONG
An attempt was made to send to an interactive terminal a record longer than the maximum allowed by the front-end station. Check the record size of the dataset. From EXP. Class, 4; reprievable.

AB090 - TEXT ADDRESS AND/OR TEXT LENGTH NOT BETWEEN USER BA AND LA
During processing of ACQUIRE, FETCH, or DISPOSE, the TEXT address and/or TEXT length specified in the PDD was found not to be between the caller's BA and LA. From EXP. Class, 4; reprievable.

AB091 - TEXT LENGTH EXCEEDS THE MAXIMUM ALLOWED
During processing of ACQUIRE, FETCH, or DISPOSE, the TEXT length was found to be greater than allowed by the system. Reduce the text length to be equal to or less than the maximum allowed. From EXP. Class, 4; reprievable.

AB092 - CANNOT 'ENTER' EMPTY DATASET INTO SDR
The permanent dataset that was accessed or acquired has no disk allocations. This can result from partially deleting a permanent dataset via DELETE (DN=,PARTIAL). From EXP. Class 4; reprievable.

AB093 - CPU REQUESTED IS NOT AVIALABLE
The CPU number that was used is too big, or the CPU that was designated was not started. Use CPU select code only if necessary. Check code to make certain the CPU number is valid. From EXP, F$OPT request. Class 4; reprievable.

AB094 - HARDWARE ERROR OCCURRED WHILE WRITING $LOG
The Disk Queue Manager detected a hardware error and was unable to complete the I/O operation. Report the error to the system operator and resubmit the job. From EXP. Class, 4; reprievable.

AB098 - MAINFRAME DOES NOT SUPPORT BIDIRECTIONAL TRANSFER MODE
An F$MDE call was made to enable the bidirectional transfer (BT) option on a machine other than a CRAY X-MP computer. Correct the MODE control statement or macro. From EXP. Class, 4; reprievable.

AB099 - AMBIGUOUS MODE SPECIFICATION
An F$MDE call was made with bits set to both enable and disable a particular option. Correct the MODE macro call. Class, 4; reprievable.

AB100 - NONSEQUENTIAL WRITE ILLEGAL ON TAPE DATASET
A nonsequential write was attempted on tape. A write to a tape dataset must follow a rewind or an initial open. Rewind to the beginning of the dataset when doing a write after a read. From EXP. Class, 4; reprievable.

AB101 - U INVALID WITH INTERCHANGE FORMAT
A tape dataset was accessed in which the data format was set to interchange; then a F$DNT call was made to declare it unblocked. Change the interchange format, or do not declare the format unblocked. From EXP. Class, 4; reprievable.

AB102 - TAPE DATASET CANNOT BE DISPOSED
An attempt was made to dispose a dataset that is an online tape. Change the job so that a DISPOSE request is not attempted. From EXP. Class, 4; reprievable.

AB103 - VSN REQUIRED FOR EXISTING DATASET
An ACCESS request was made without the NEW or VOL parameters. Use the VOL parameter on an ACCESS request to specify the desired VSNs. From EXP. Class, 4; reprievable.

AB104 - GENERIC RESCOURCE LIMIT EXCEEDED
A tape ACCESS was attempted when the job had no remaining resources, or a disk dataset associated with a generic resource (such as SSD or BMR) was extended beyond the limit declared on the JOB statement. Increase the job statement generic resource declaration. From EXP. Class, 4; reprievable.

AB108 - WRITE ATTEMPT ON PROTECTED VOLUME
An attempt was made to write on a password-protected volume. From EXP. Class, 4; reprievable.

AB109 - WRITE FORMAT ERROR
A tape write request cannot be executed because there is an error in the RCWs or BCWs in the circular buffer. Examine the circular buffer to ensure that all BCWs and RCWs are correctly chained. From EXP. Class, 4; reprievable.

AB110 - WRITE PROTOCOL ERROR
COS and IOP detected a protocol error during an I/O write operation. The tape subsystem has an internal error. Save the logfile output, and ask a Cray Research analyst to examine the TQM trace buffer from the logfile. If the error is repeatable, try to save the job for use as a test case. From EXP. Class, 4; reprievable.

AB111 - TAPE OFF END OF REEL
The tape came off the end of a reel for one of the following reasons:

- An EOT marker is not on the tape .
- There is insufficient space after the EOT reflective marker. This is caused by an attempt to write large blocks on a standard reel or

attempting an excessive write recovery at the reel's end (write only).

- The tapemark denoting end of reel was not sensed (write only).

If there is no reflective marker at EOT, add one. If the block is very large, the tape should be prepared with extra space between the marker and the end of the tape, according to the tape density and the maximum block size. If an excessive write recovery was the cause of going off the end of the reel, the tape should be unwound until an undamaged area is found and a new reflective marker applied. From EXP. Class, 4; reprievable.

AB112 - TAPE VOLUME IS PROTECTED
An attempt was made to access a password-protected volume. Password-protected volumes cannot be accessed on the Cray mainframe. From EXP. Class, 4; reprievable.

AB113 - TAPE DATASET IS PROTECTED
An attempt was made to access a password-protected dataset. Password-protected datasets cannot be accessed on the Cray mainframe. From EXP. Class, 4; reprievable.

AB114 - NEW TAPE DATASETS MUST BE WRITTEN TO FIRST
A read was the first attempted operation on a dataset that was accessed with a NEW parameter. A new dataset must be written before a read can occur. If the intent was to read an existing dataset, remove the NEW parameter from the ACCESS statement. From EXP. Class, 4; reprievable.

AB115 - DATASET DOES NOT RESIDE IN VOLUME SET
The correct volume was accessed, but the file ID field or the file sequence number of the header label did not match what was specified on the ACCESS statement. The job is aborted. The user must specify the proper dataset name to access the dataset. From EXP. Class, 4; reprievable.

AB116 - FILE (VOLUME) SECTION DOES NOT EXIST
The correct volume was mounted, but the file section number of the header label did not match the value of the FSEC parameter on the ACCESS statement. From EXP. Class, 4; reprievable.

AB117 - LDT CONSTRUCTION ERROR
The Label Definition Table (LDT) was incorrectly formatted. Ask a Cray Research analyst to check for problems in the $SYSLIB version of ACCESS or in EXP. From EXP. Class, 4; reprievable.

AB118 - TAPE LABEL GROUP CORRUPTED SEE TQM MESSAGE
An error exists in the trailer label group on a volume of the input dataset. For example, a required label is missing from the trailer label group. The job is aborted. From EXP. Class, 4; reprievable.

AB119 - DEFERRED LABELED TAPE FEATURE
A labeled tape was accessed with label field values not supported by COS. Tapes with a generation number, generation version number, file sequence number, or BLP label type cannot be processed by this version of COS. From EXP. Class, 4; reprievable.

AB120 - LABEL GROUP DOES NOT CONTAIN HDR1 LABEL
A labeled tape (with the correct VOL1 label) was accessed, but there is no HDR1 label in the label group. The tape cannot be processed by COS because it is not an ANSI or IBM standard labeled tape. From EXP. Class, 4; reprievable.

AB121 - INVALID RECORD FORMAT SPECIFIER
The record format field of the HDR2 label is unrecognizable or does not match what was specified on the ACCESS control statement. The tape cannot be processed by COS because it is not an ANSI or IBM standard labeled tape. Ensure that the specified field matches the label group or omit the value from the ACCESS statement so that the value is defaulted to the value from the label. From EXP. Class, 4; reprievable.

AB122 - INVALID TAPE BLOCK ATTRIBUTES
The block attribute field of the HDR2 label is unrecognizable and/or does not match what was specified on the ACCESS control statement. The tape cannot be processed by COS because it is not an ANSI or IBM standard labeled tape. Ensure that the specified field matches the label group or omit the value from the ACCESS statement so the value is defaulted to the value from the label. From EXP. Class, 4; reprievable.

AB123 - INVALID RECORD LENGTH SPECIFIER
The record length field of the HDR2 label is unrecognizable or does not match what was specified on the ACCESS control statement. The tape cannot be processed by COS because it is not an ANSI or IBM standard labeled tape. Ensure that the specified field matches the label group, or omit the value from the ACCESS statement so that the value defaults to the value from the label. From EXP. Class, 4; reprievable.

AB124 - INVALID TAPE BLOCK LENGTH SPECIFIER
The block length field of the HDR2 label is unrecognizable or does not match what was specified on the ACCESS control statement. The tape cannot be processed by COS because it is not an ANSI or IBM standard labeled tape. Ensure that the specified field matches the label group, or omit the value from the ACCESS statement so that the value defaults to the value from the label. From EXP. Class, 4; reprievable.

AB125 - INVALID BUFFER OFFSET SPECIFIER
The buffer offset field of the HDR2 label is unrecognizable or does not match what was specified on the ACCESS control statement. The tape cannot be processed by COS because it is not an ANSI or IBM standard labeled tape. Ensure that the specified field matches the label group or omit the value from the ACCESS statement so that the value defaults to the value from the label. From EXP. Class, 4; reprievable.

AB126 - INVALID OWNER IDENTIFIER
The owner ID field of the VOL1 label does not match the owner ID specified in the LDT. The job is aborted. The correct owner ID is needed to access the volume. From EXP. Class, 4; reprievable.

AB127 - INCOMPLETE VOLUME SERIAL LIST
The volume serial number list for the dataset is empty before the end of the input dataset is reached. Resubmit the job with the entire list of VSNs specified on the VOL parameter of the ACCESS statement. From EXP. Class, 4; reprievable.

AB128 - ATTEMPT TO READ AN EXPIRED DATASET
This message does not occur in the released system, but can occur if the condition is selected by an installation's local modification to TQM. From EXP. Class, 4; reprievable.

AB129 - ATTEMPT TO WRITE UPON NON-EXPIRED DATASET
The expiration date in the HDR1 label has not passed. COS does not allow an unexpired dataset to be overwritten. From EXP. Class, 4; reprievable.

AB130 - TAPE DATASET HAS INVALID EXPIRATION DATE
The expiration date field of the HDR1 label is unrecognizable. The tape cannot be processed by COS because it is not an ANSI or IBM standard labeled tape. From EXP. Class, 4; reprievable.

AB131 - BLOCKS READ DIFFER FROM LABEL BLOCK COUNT
The block count for the current file section (volume) does not match the block count recorded in the trailer label. If this happens on an output tape, there is an internal problem in TQM. Notify a Cray Research analyst. If this is an input tape, the tape cannot be processed by COS. In any event, save the tape and job for a Cray Research analyst. From EXP. Class, 4; reprievable.

AB132 - TAPE LABEL TYPE CANNOT BE SCRATCHED
An attempt was made to write over a tape containing a label type different from the requested label type and not scratchable at this installation. COS does not allow the label type to be changed. From EXP. Class, 4; reprievable.

AB133 - ACTION ILLEGAL FOR NON-TAPE DATASET
An F$POS request was made for a mass storage dataset. Use GETPOS and SETPOS for mass storage datasets and POSITION for tape datasets. From EXP. Class, 4; reprievable.

AB134 - MAXIMUM BLOCK SIZE EXCEEDED ON READ
During a read, a tape block was encountered that is larger than the maximum specified in the ACCESS statement. Increase the value of MBS. From EXP. Class, 4; reprievable.

AB135 - DEDICATED RESOURCE NOT AVAILABLE
A dedicated resource, such as a magnetic tape drive, is down (unavailable); operator is requesting job abort. Resubmit the job if possible. From EXP. Class, 4; reprievable.

AB136 - ACCESS DENIED BY SERVICING FRONT-END
The job tried to gain access to either a tape dataset or a tape volume. Access was denied by the servicing front end of that dataset. The front end might have issued a message to the job logfile indicating why access was denied. Correct the condition that caused the front end to deny access. From EXP. Class, 4; reprievable.

AB137 - SERVICING FRONT-END IS NOT SECURE; ACCESS DENIED
The job requested access to a tape dataset or tape volume where the servicing front end for the dataset does not have security mechanisms for such accesses, and COS is configured so that servicing systems must perform security checks. Change the servicing front end of the dataset; if this is impossible, see a Cray Research analyst. From EXP. Class, 4; reprievable.

AB138 - DATASET IS ALREADY CATALOGED ON SERVICING FRONT-END
The job tried to save a tape dataset on its servicing front end when the dataset is already resident there. Delete the dataset's definition in the servicing front end's catalog. From EXP. Class, 4; reprievable.

AB139 - DATASET DOES NOT RESIDE IN SERVICING FRONT-END CATALOG
The job tried to access an old tape dataset from its servicing front end when the dataset does not reside there. Either change the disposition of the dataset to NEW, or create the dataset to reside in the front end's catalog. From EXP. Class, 4; reprievable.

AB140 - SERVICING FRONT-END CATALOG UPDATE FAILED
The servicing front end was requested to update its dataset or volume catalog, and the request failed. The front end can supply information about the failure, via the job logfile. Resubmit the job. From EXP. Class, 4; reprievable.

AB141 - DEVICE IS NOT OPEN FOR I/O
A fatal error was encountered for the dataset, so that no more input or output requests can be accepted. Determine and correct the problem that caused the device to be closed for I/O. From EXP. Class, 4; reprievable.

AB142 - VOLUME DOES NOT RESIDE IN FRONT-END SERVICING CATALOG
The job requested a tape volume that does not reside in the servicing front end's tape management catalog. Either specify the correct volume, or change the ACCESS of the dataset so that tape management by the servicing front end is bypassed. From EXP. Class, 4; reprievable.

AB143 - TAPE VOLUME MOUNT CANCELLED BY OPERATOR
The mount of a tape volume was rejected by the operator. Correct the condition that caused the rejection. From EXP. Class, 4; reprievable.

AB144 - MAXIMUM BLOCK SIZE EXCEEDED ON WRITE OF TAPE DATASET
From EXP. Class, 4; reprievable.

AB147 - FSEC PARAMETER LARGER THAN NUMBER OF VSN
The MOD parameter is specified on the ACCESS statement of a tape dataset, and the value specified by the FSEC parameter is larger than the number of volume serial numbers in the VOL parameter. From EXP. Class, 4; reprievable

AB148 - INVALID TVT ADDR IN DNT
When processing a tape dataset, the TVT (Tape Volume Table) address is incorrect. From EXP. Class, 4; reprievable.

AB149 - F$TBL, TABLE NOT FOUND
An F$TBL request specified a table name which was not recognized. From EXP. Class, 4; reprievable.

AB150 - DEVICE OVERFLOW WITH NOF DECLARED
The size of a dataset assigned with the no overflow option (NOF) exceeded the device to which the dataset was assigned. From EXP. Class, 4; reprievable.

AB150 - SYNCH INPUT REQUEST - DATASET NOT AT EOR
The user program must have read a complete record before issuing a SYNCH request for an input tape dataset. From EXP. Class, 4; reprievable.

AB151 - SYNCH DATASET NOT INTERCHANGE FORMAT
The tape dataset being synchronized is not interchange format. Synchronization can only be done on interchange format datasets. From EXP. Class, 4; reprievable.

AB153 - RANDOM AND SEQUENTIAL MAY NOT BE SPECIFIED TOGETHER
Bits DDRDM and DDSEQ were both set in the DDL. Clear the appropriate bit. Class, 4; reprievable.

AB154 - BLOCKED AND UNBLOCKED MAY NOT BE SPECIFIED TOGETHER
Bits DDUDS and DDBLK were both set in the DDL. Clear the appropriate bit. Class, 4; reprievable.

AB162 - EMA MAY NOT BE CHANGED WHILE MULTITASKING
Extended addressing mode may not be changed while more than one task is active in a job step. Change the program to complete all MODE,EMA changes before or after the multitasking phases of execution. From EXP. Class, 4; reprievable.

AB163 - EXTENDED ADDRESSING NOT AVAILABLE
Issued when a MODE,EMA statement or macro is issued on a computer system which does not support extended memory addressing. From EXP. Class, 4; reprievable.

AB164 - SECOND VECTOR LOGICAL UNIT NOT AVAILABLE
Issued when a MODE,AVL statement or macro is issued on a computer system which does not support an additional vector logical functional unit. From EXP. Class 4; reprievable.

AB165 - OPERAND RANGE INTERRUPTS CAN NOT BE DISABLED
Issued when a MODE,ORI statement or macro is issued on a computer system which does not allow operand range interrupts to be suppressed. From EXP. Class, 4; reprievable.

AB171 - CANNOT MOVE DRIVER REPLY TO PARAMETER BLOCK
The parameter block was destroyed. From EXP. Class, 4; reprievable.

AB172 - INSUFFICIENT CONTIGUOUS DISK SPACE
Contiguous disk space was not available for the current space request. Options include not specifying the dataset as contiguous; specifying a different device for contiguous space; or not specifying a device, in which case the system will find a device containing enough contiguous space. From EXP. Class, 4; reprievable.

AB173 - INTERJOB CONNECTIONS WERE OPEN
At the time of a job step advance one or more interjob connections were still open. The system has closed the path and removed the RIT and IPT table entries. From EXP. Class, 4; reprievable.

AB174 - NO AVAILABLE LFT SPACE FOUND IN USER AREA
There is no available LFT space in the user area. If running in STACK or multitasking mode, increment the stack size and resubmit. From EXP. Class, fatal.

AB175 - INVALID F$PERF SUBFUNCTION
System request F$PERF was issued with an unknown subfunction. From EXP. Class, 4; reprievable.

AB176 - PERFORMANCE MONITOR NOT AVAILABLE
Issued when an F$PERF request is received but the hardware performance monitor feature is not present on the machine (as indicated in the site's configuration deck). From EXP. Class, 4; reprievable.

AB177 - INVALID GROUP NUMBER FOR F$PERF/PM$ON
The hardware performance monitor group number specified in the subfunction PM$ON on a F$PERF request is either negative or larger than the highest group number. From EXP. Class, 4; reprievable.

AB178 - JTA LFT ALREADY EXISTS
The user attempted an F$LFT LFTCREA call for a dataset name for which an LFT in the JTA already existed. From EXP. Class, 4; reprievable.

AB179 - DSP IN/OUT POINTER NOT ON A BLOCK BOUNDARY FOR WRITE/READ
The user has initiated I/O on a dataset and the proper pointer (IN/OUT) is not on a block boundary. For instance, on a write, the user moves data into the buffer and modifies IN. The system moves data out of the buffer and modifies OUT. If the user attempts a write and OUT is not on a block boundary, this error will occur. Similarly for IN on a read. From EXP. Class, 4; reprievable.

AB180 - DSP BUFFER POINTERS OVERLAP LFT/DSP AREA
The user has initiated I/O on a dataset and the buffer pointers overlap the area at the high end of user memory which contains the LFTs and DSPs. Specifically, DPFRST is less than JCBFB and DPLMT is greater than JCHLM. From EXP. Class, 4; reprievable.

AB181 - DSP LIMIT POINTER LESS THAN FIRST
The DSP LIMIT pointer, which points to the high-address end of the buffer, is actually less than the DSP FIRST pointer, which points to the low-address end of the buffer. From EXP. Class, 4; reprievable.

AB182 - DSP IN POINTER NOT BETWEEN FIRST AND LIMIT
The DSP IN pointer, which controls where data will be placed in the buffer, does not fall between the DSP FIRST pointer and the DSP LIMIT pointer. From EXP. Class, 4; reprievable.

AB183 - DSP OUT POINTER NOT BETWEEN FIRST AND LIMIT
The DSP OUT pointer, which controls where data will be moved from the buffer, is not between the DSP FIRST pointer and the DSP LIMIT pointer. From EXP. Class 4, reprievable.

AB184 - DSP FIRST POINTER OUT OF BOUNDS
The DSP FIRST pointer, which denotes the low-address end of the buffer, does not fall between user BA and LA. From EXP. Class, 4; reprievable.

AB185 - DSP LIMIT POINTER OUT OF BOUNDS
The DSP LIMIT pointer, which denotes the high-address end of the buffer, does not fall between user BA and LA. From EXP. Class, 4; reprievable.

AB186 - DSP RCW POINTER NOT BETWEEN FIRST AND LIMIT
The DSP RCW pointer, which points to the current record control word being used by user libraries or TIO, does not fall between the DSP FIRST pointer and the DSP LIMIT pointer. From EXP. Class, 4; reprievable.

AB187 - BUFFERED I/O RECORD ADDRESS OUT OF BOUNDS
The DSP pointers to the area for a buffer I/O record point to an area outside the user field length. From EXP. Class, 4; reprievable.

AB188 - UNKNOWN BUFFER I/O FUNCTION
An F$BIO request was made with field DPBF not equal to a valid function code. From EXP. Class, 4; reprievable.

AB189 - BUFFER LENGTH NOT MULTIPLE OF D'512
The DSP FIRST and limit pointers delimit an area that is not a multiple of 512 in length. Data is transferred by the system into or out of the buffer only in 512-word sectors. From EXP. Class, 4; reprievable.

AB190 - UNCLEARED ERROR IN DSP
An I/O request was made with a bit set in the DPERR field. From EXP. Class, 4; reprievable.

AB191 - ATTEMPT TO START I/O ON BUSY DATASET
I/O is still outstanding for the dataset and you are attempting to start more I/O for the dataset. Ensure I/O is done for the dataset before trying to start more. From EXP. Class 4, reprievable.

AB192 - DSP SAVE WORDS DESTROYED BY USER
You have destroyed words in the DSP that the system uses for buffered I/O. Correct the program to avoid destroying the words. From EXP. Class 4, reprievable.

AB193 - ALL USER TASKS DEACTIVATED
All tasks within the user's job have been deactivated via F$TASK(TASK$DEA) calls. Change the program to correctly synchronize user tasks. From EXP. Class 4, reprievable.

AB194 - USER DEADLOCK DETECTED
All tasks within the user's job are either waiting for a semaphore or are deactivated. Change the program to correctly synchronize user tasks. From EXP. Class 4, reprievable.

AB195 - ATTEMPT TO DEACTIVATE AN INACTIVE TASK
An attempt was made to deactivate (via F$TASK(TASK$DEA)) a user task that had already been deactivated. Change the program to correctly synchronize user tasks. From EXP. Class, 4; reprievable.

AB196 - ATTEMPT TO ACTIVATE AN ACTIVE TASK
An attempt was made to activate (via F$TASK(TASK$ACT)) a user task that had not been deactivated (via an F$TASK(TASK$DEA)). Change the user program to correctly synchronize user tasks. Class 4, reprievable.

AB197 - USER TASK ATTEMPT TO ACTIVATE SELF
An attempt was made to activate (via F$TASK(TASK$ACT)) a user task that was the task making the call. Change the program to correctly synchronize user tasks. From EXP. Class, 4; reprievable.

AB198 - INVALID USER TASK ID
An F$TASK call was made with a user task ID that does not match any user tasks within the job. Change the program to correctly synchronize user tasks. Source EXP. Class 4, reprievable.

AB199 - MAXIMUM USER TASKS PER JOB EXCEEDED
An attempt was made to create more tasks within a user job than the installation-defined limit. If multitasking is enabled for the site, correct the program to use only the number of tasks the site allows. If multitasking is not enabled, do not try to use it. Source EXP. Class, 4; reprievable.

AB237 - ZERO SECTOR COUNT SPECIFIED IN QUEUED I/O REQUEST
An F$QIO request was made with a data transfer length of zero sectors. Correct the program so that it does not issue queued I/O requests with zero sector counts. From EXP. Class, 4; reprievable.

AC002 - 16 CHARACTER ACCOUNT NUMBER ILLEGAL
In the utility ACCTDEF, one of the directives in the input file contained an AC value exceeding the 15 characters allowed for an account number. Either remove the erroneous directive or correct the error and rerun the program (ACCTDEF). Class, caution.

AC003 - 16 CHARACTER PASSWORD ILLEGAL
In the utility ACCTDEF, one of the directives in the input file contained an APW value exceeding the 15 characters allowed for a password. Either remove the erroneous directive or correct the error and rerun the program (ACCTDEF). From ACCTDEF. Class, caution.

AC004 - ACCOUNT NUMBER NOT DEFINED
In the utility ACCTDEF, one of the directives (CHANGE, ACTIVATE, INACTIVE, or DELETE) in the input file contained values for the account number and password that did not match an AC or APW entry in the accounting dataset. Either remove the directive from the input file or use a CREATE directive to place the specified account number or password in the accounting dataset. From ACCTDEF. Class, Caution.

AC005 - ENTRY TO BE CHANGED IS INACTIVE
In the utility ACCTDEF, one of the CHANGE directives in the input file referenced an inactive AC or APW entry in the accounting dataset. Remove the directive from the input file or use an ACTIVATE directive to activate the AC or APW to be changed; then use the CHANGE directive. Or use a DELETE directive to delete the AC/APW referenced; then use the CREATE directive to create a new AC or APW entry. From ACCTDEF. Class, caution.

AC006 - ACCOUNT NUMBER ALREADY EXISTS
In the utility ACCTDEF, one of the CREATE directives in the input file referenced an account number and password that already existed in the accounting dataset. From ACCTDEF. Class, caution.

AC007 - ENTRY IS ALREADY ACTIVE
In the utility ACCTDEF, one of the ACTIVATE directives in the input file attempted to activate an account number and password that were already active in the accounting dataset. From ACCTDEF. Class, caution.

AC008 - ENTRY IS ALREADY INACTIVE
In the utility ACCTDEF, one of the INACTIVE directives in the input file attempted to deactivate an account number and password that were already inactive in the accounting dataset. From ACCTDEF. Class, caution.

AC009 - ILLEGAL FOR GENERATION RUN
In the utility ACCTDEF, only the CREATE, VIEW, END, EXIT and BYE directives are legal for a creation run. From ACCTDEF. Class, caution.

AC010 - UNKNOWN DIRECTIVE-IGNORED
In the utility ACCTDEF, a directive was specified other than CREATE, CHANGE, ACTIVATE, INACTIVE, DELETE, VIEW, EXIT, END and BYE. From ACCTDEF. Class, caution.

AC020 - KEYWORD ERROR IN PPL
A directive contains an illegal or invalid keyword. Check keywords and rerun. From ACCTDEF. Class, fatal; not reprievable.

AC021 - DIRECTIVE CRACKER ENCOUNTERED AN ERROR
When the utility ACCTDEF was run, a fatal error was detected in at least one of the directives in the input file. Correct the directives that contain errors. From ACCTDEF. Class, fatal; not reprievable.

AC022 - I/O ERROR READING ACCOUNTING DATASET
The accounting dataset could not be read or written correctly. Retry; if problem remains, create a new accounting dataset. From ACCTDEF. Class, fatal; not reprievable.

AC023 - UNABLE TO OPEN ACCOUNTING DATASET
The accounting dataset contains a fatal error on open. Create a new accounting dataset. From ACCTDEF. Class, fatal; not reprievable.

AC100 - 16 CHARACTER ACCOUNT NUMBER ILLEGAL
A 16-character account number was specified on the ACCOUNT statement. Only 15 characters are allowed. Correct the account number. From ACCOUNT. Class, fatal.

AC101 - 16 CHARACTER PASSWORD ILLEGAL
A 16-character password was specified on the ACCOUNT statement. Only 15 characters are allowed. Correct the password. From ACCOUNT. Class, fatal.

AC102 - ACCOUNTING DATASET DOES NOT EXIST
The accounting dataset does not exist. If accounting is not mandatory, remove the ACCOUNT control statement from the job deck. If accounting is mandatory, see a Cray Research analyst. From ACCOUNT. Class, fatal.

AC103 - ERROR ON ACCESS OF ACCOUNTING DATASET
The accounting dataset was not accessed. See a Cray Research analyst. From ACCOUNT. Class, fatal.

AC104 - INVALID ACCOUNT NUMBER
The account number specified on the ACCOUNT statement is not in the accounting dataset. Check account number. See the analyst in charge for a proper account number . From ACCOUNT. Class, fatal.

AC105 - ACCOUNT NUMBER/PASSWORD IS INACTIVE
The account number and password specified on the account control statement is inactive and therefore invalid. Use an active account number and password. From ACCOUNT. Class, fatal.

AC106 - ACCOUNT PASSWORD IS REQUIRED.
A password on the ACCOUNT control statement has been defined by the site to be mandatory. No password was specified on the ACCOUNT statement. Obtain the appropriate password to be used with the account number. From ACCOUNT. Class, fatal.

AC107 - SPECIFY KEYWORD TO CHANGE NULL PASSWORD
The keyword must be specified without a value to change from or to a null account password. From ACCOUNT. Class, fatal.

AC108 - ACCOUNTING IS OPTIONAL, JOB CONTINUES
Accounting is not mandatory. Although an error on the account statement might have occurred, the job continues to run. From ACCOUNT. Class, informative.

AC109 - INVALID USER NUMBER
The user is not validated for system access with the user number and user password as indicated on the account statement. Resubmit the account statement with the correct user number and password. If the statement is rejected again, see the installation manager. From ACCOUNT. Class, fatal.

ACl10 - USER NUMBER IS REQUIRED
A user number has been defined by the site to be necessary for system access. Check the user number and user password; otherwise, see a Cray Research analyst. From ACCOUNT. Class, fatal.

ACl12 - VALIDATION DATASET DOES NOT EXIST
The system could not find the validation dataset. See a Cray Research analyst. From ACCOUNT. Class, fatal.

ACl13 - ERROR ON ACCESS OF VALIDATION DATASET
The validation dataset could not be read or written correctly. Retry; if the problem remains, see a Cray Research analyst. From ACCOUNT. Class, fatal.

ACl14 - PASSWORD EXPIRED, NEW PASSWORD REQUIRED
The existing password expired. Specify the existing password and a new password with the NUPW parameter and resubmit the job. From ACCOUNT. Class, fatal.

ACl16 - USER ENTRY IS DISABLED
The user number and user password has been disabled. See the installation manager to determine why the entry has been disabled and have the entry enabled. From ACCOUNT. Class, fatal.

ACl17 - USER NUMBER/PASSWORD MIS-MATCH
An incorrect user password was specified on the account statement. Resubmit the account statement with the correct combination of user number and user password. From ACCOUNT. Class, fatal.

ACl18 - ERROR ON OPEN OF VALIDATION DATASET
The validation dataset could not be read or written correctly. Retry; if the problem remains, see the installation manager. From ACCOUNT. Class, fatal.

ACl19 - 16 CHARACTER USER NUMBER ILLEGAL
A 16-character user number was specified on the account statement. Only 15 characters are allowed. Correct the user number. From ACCOUNT. Class, fatal.

ACl20 - 16 CHARACTER PASSWORD ILLEGAL
A 16-character user password was specified on the account statement. Only 15 characters are allowed. Correct the user password. From ACCOUNT. Class, fatal.

ACl21 - ACCOUNT NUMBER/PASSWORD MIS-MATCH
An incorrect account password was specified on the account statement. Resubmit the account statement with the correct account number and account password combination. From ACCOUNT. Class, fatal.

ACl22 - INVALID ACCOUNT NUMBER/PASSWORD; REENTER
This message is issued by ACCOUNT if an incorrect account number/password is entered during interactive account statement processing. The user is given four chances to enter the correct account number/password at which time, if the account number/password is still incorrect, the job will be deleted. From ACCOUNT. Class, warning.

ACl23 - ERROR OCURRED DURING ACCOUNT PROCESSING
The account dataset could not be read or written correctly. Retry; if the problem remains, see the installation manager. From ACCOUNT. Class, fatal.

ACl24 - ACCOUNT PASSWORD IS OPTIONAL, JOB CONTINUES
Account password is not mandatory. Although an incorrect account password may have been specified, the job continues to run. From ACCOUNT. Class, informative.

AD001 - BINARY MUST BE ABSOLUTE
A relocatable binary or bad data is present on the $BLD input dataset. Generate absolute binary load modules by adding the ABS pseudo instruction to the relocatable APML modules. From ADSTAPE. Class, fatal.

AD002 - LENGTH OR ORIGIN TOO LARGE
The binary load module on the input dataset (default is $BLD) contains a binary text length or origin that is too large and that results in binary text being loaded beyond an I/O Processor field length of 65K parcels (16K words). Reduce the binary size or origin; reassemble. From ADSTAPE. Class, fatal.

AD003 - PREMATURE END OF RECORD
End of record was encountered on the input dataset (default is $BLD) before all of the expected binary text or Program Descriptor Table was read. The input file probably does not contain binary load modules. Examine the job control language for errors. From ADSTAPE. Class, fatal.

AD004 - COMPILATION ERRORS
The binary load modules on the input dataset (default is $BLD) contain assembly or compilation errors. Correct the errors; if this is not feasible, add a DEBUG parameter to the associated APML statements to allow use of the binary modules with errors; reassemble. From ADSTAPE. Class, fatal.

AL001 - OAL DATASET MUST BE SPECIFIED
The dataset specified by the user as the OAL dataset does not exist. (The OAL dataset default is $OAL.) Correct the control statement and resubmit the job. From BIND. Class, fatal.

AL002 - BUFFERIN UNEXPECTED STATUS (*stat*)
The status returned by BUFFERIN is not one of the status values expected by the program. Make sure that the OAL dataset is not corrupt. (If it can be processed by ITEMIZE and BUILD, it is probably acceptable.) Save the relevant dataset, job stream, and listing; see a Cray Research analyst. From BIND pass 1 processing, subroutine READPDT. Class, fatal.

AL003 - WRONG TABLE TYPE (*type*)
The first loader table in an object record was not a PDT table. The octal digits in the message indicate the table type that was encountered. Make sure that the OAL dataset is not corrupt. Save the relevant dataset, job stream, and listing; see a Cray Research analyst. From BIND pass 1 processing, subroutine READPDT. Class, fatal.

AL004 - TOO MUCH TRAILER INFORMATION (*tinfo*)
The trailer information in the PDT is too long for a BIND internal buffer. (See figure PDT-7 in the COS Table Descriptions Internal Reference Manual, publication SM-0045. Trailer information is labeled PDT header data.) Make sure that the OAL dataset is not corrupt. If the trailer information is longer than 1024 words, additional space can be created by a change in the definition of parameter WORKSZ in the UPDATE comdeck ALWORK. (The actual size of the trailer information is given as a decimal number enclosed in parentheses in the message.) From BIND pass 1 processing, subroutine READPDT. Class, fatal.

AL005 - TOO MUCH EXTERNAL INFORMATION (*extinfo*)
The external information in the PDT is too long for a BIND internal buffer. Make sure that the OAL dataset is not corrupt. If the external information is longer than 1024 words, additional space can be created by a change in the definition of parameter WORKSZ in the UPDATE comdeck ALWORK. The actual size of the external information is given as a decimal number enclosed in parentheses in the message. From BIND pass 1 processing, subroutine READPDT. Class, fatal.

AL006 - UNEXPECTED STATUS READING HEADER (*iostat*)
BIND was unable to read a loader table during pass 2 processing. This condition is abnormal because the dataset was read successfully during pass 1. See a Cray Research analyst. From BIND pass 2 processing, subroutine PASS2. Class, fatal.

AL007 - UNDEFINED TABLE TYPE *type*
BIND encountered an undefined table type or a misplaced PDT-type table during pass 2 processing. (The table type is given in the two octal digits in the message.) This condition is abnormal because the dataset was successfully read during pass 1. See a Cray Research analyst. From BIND pass 2 processing, subroutine PASS2. Class, fatal.

AL008 - UNEXPECTED STATUS READING TABLE BODY *iostat*
BIND was unable to read the body of a table during pass 2 processing. This condition is abnormal because the dataset was successfully read during pass 1. See a Cray Research analyst. From BIND pass 2 processing, subroutine PASS2. Class, fatal.

AL009 - BUFFERIN UNEXPECTED STATUS (*stat*)
The status returned by BUFFERIN is not one of the status values expected by the program. This condition is abnormal because the dataset was read successfully during pass 1. See a Cray Research analyst. From BIND pass 2 processing, subroutine READPDT2. Class, fatal.

AL010 - WRONG TABLE TYPE (*type*)
The first loader table in an object record was not a PDT table. The octal digits in the message indicate the table type encountered. This condition is abnormal because the dataset was read successfully during pass 1. See a Cray Research analyst. From BIND pass 2 processing, subroutine READPDT2. Class, fatal.

AL011 - TOO MUCH ENTRY INFORMATION (*entinfo*)
The entry information in the PDT is too long for a BIND internal buffer. Make sure that the OAL dataset is not corrupt. If the entry information is longer than 1024 words, additional space can be created by a change in the definition of parameter WORKSZ in the UPDATE comdeck ALWORK. (The actual size of the entry information is given as a decimal number enclosed in parentheses in the message.) (Unlike messages AL009, AL010, and AL012, it is reasonable for this message to occur during pass 2 processing because pass 1 does not check this information.) From BIND pass 2 processing, subroutine READPDT2. Class, fatal.

AL012 - TOO MUCH EXTERNAL INFORMATION (*extinfo*)
The external information in the PDT is too long for a BIND internal buffer. This condition is abnormal because the dataset was read successfully during pass 1. See a Cray Research analyst. From BIND pass 2 processing, subroutine READPDT2. Class, fatal.

AL013 - UNIMPLEMENTED BLOCK INDEX REFERENCE
BIND can handle only block indices 0 and 2. Any other block index value, such as a common block reference, produces this message. Your APML source must not use common blocks. Revise the APML program. From BIND pass 2 processing, subroutine TXT2. Class, fatal.

AL014 - TXT TABLE INVALID
The Q bit was set in a TXT table. See the COS Table Descriptions Internal Reference Manual, publication SM-0045. See a Cray Research analyst. From BIND pass 2 processing, subroutine TXT2. Class, fatal.

AL015 - TABLE *tbl* IS NOT IMPLEMENTED
BIND encountered a loader table type that it cannot process. These table types are DMT, DIR, SMT, and USR. Make sure that the OAL dataset is not corrupt. Revise the APML source or the APML control statement to avoid unimplemented table types. From BIND. Class, fatal.

AL016 - BIND VERSION *vsn*
Initiation of BIND. From BIND. Class, informative.

AL017 - TOO MUCH BLOCK INFORMATION (*blinfo*)
The block information in the PDT is too long for a BIND internal buffer. Make sure that the OAL dataset is not corrupt. If the block information is longer than 1024 words, additional space can be created by a change in the definition of parameter WORKSZ in the UPDATE comdeck ALWORK. (The actual size of the block information is given as a decimal number enclosed in parentheses in the message.) From BIND pass 1 processing, subroutine READPDT. Class, fatal.

AL018 - *n* FIELD OVERFLOWS
When external references were being resolved, at least one external value would not fit in the receiving field. The number in the message is the count of occurrences of this problem. See a Cray Research analyst. From BIND. Class, caution.

AL019 - *a* DBL DEFS *b* UNDEFINED *c* USES
One or more undefined or doubly-defined symbols were encountered during processing.

- *a* Number of doubly-defined symbols
- *b* Number of undefined symbols referenced
- *c* Number of references to undefined symbols

Correct the APML source and resubmit. From BIND. Class, caution.

AL020 - ATTEMPT TO FREE UNALLOCATED BLOCK
A call to routine DSFREE atempted to free an unallocated block of dynamic memory. This should never occur. Save the relevant files and listings; see a Cray Research analyst. From BIND, subroutine DSFREE. Class, fatal.

AL021 - OUT OF STORAGE SPACE
DSGET was unable to satisfy a memory allocation request. Your program probably contains too many external symbols. BIND must be recompiled to correct this. The size of dynamic memory is defined in common deck CDSMEM with the parameter MEMSIZE. Increasing the value of the MEMSIZE parameter increases the dynamic storage available. From BIND, routine DSGET. Class, fatal.

AL022 - DSINIT PARAMETER ERROR
DSINIT was given an excessive parameter value for dynamic storage space. This should never occur. Save relevant files and listing; see a Cray Research analyst. From BIND, routine DSINIT. Class, fatal.

NOTE

Message classes for AP and CA messages, issued by APML and CAL, are different from those for other messages. The classes and meanings are as follows.

Abort: CAL or APML aborts

Fatal: if the ABORT option is off, the PDT fatal error flag is set. If the ABORT option is on, CAL or APML aborts when assembly of all modules in the current input is complete.

Warning: a possible error was detected, but no action is taken.

Informative: information only; does not imply that any problem exists.

AP000 - [APML] INTERNAL 'APML' ERROR DETECTED AT P = *paddress*
APML detects an internal error at parcel address *paddress* and is unable to proceed. See a Cray Research analyst. From APML. Class, abort (immediate).

AP001 - [APML] APML VERSION *x.xx* (*mm/dd/yy*) - *cpu*
At the beginning of each assembly, APML issues an informative message indicating

the version number *x.xx*, the date *mm/dd/yy* on which APML was assembled, and the *cpu* that APML is assembling for. From APML. Class, informative.

AP002 - [APML] ASSEMBLY TIME: *n* CPU SECONDS
All programs in the current file of the source dataset have been assembled. *n* is the assembly time in floating-point CPU seconds. From APML. Class, informative.

AP003 - [APML] MEMORY WORDS: *mw* + I/O BUFFERS: *bw*
All programs in the current file of the source dataset have been assembled. *mw* is the decimal number of memory words required in the user portion of the job field. *bw* is the decimal number of words needed for the I/O table and buffer area of this job step. From APML. Class, informative.

AP004 - [APML] ASSEMBLY ERRORS
If the user set the ABORT flag on the APML control statement and fatal errors were encountered during assembly, then APML will issue this message followed by an abort. Either remove the ABORT flag from the APML control statement or correct all fatal errors found by APML. From APML. Class, abort.

alternative forms:
AP010 - [APML] 1 WARNING ERROR, PROGRAM MODULE *pn*
AP010 - [APML] *n* WARNING ERRORS, PROGRAM MODULE *pn*
APML issues this message for all source lines, from the previous program module (if any) through program module *pn*, in which warning errors have been detected. *pn* will be equivalent to the name used on a particular IDENT pseudo statement. Examine all warning errors and correct if necessary. See Appendix D of the APML Assembler Version 1 Reference Manual, CRI publication SM-0036, for a list of warning errors. From APML. Class, warning.

alternative forms:
AP011 - [APML] 1 FATAL ERROR, PROGRAM MODULE *pn*
AP011 - [APML] *n* FATAL ERRORS, PROGRAM MODULE *pn*
APML issues this message for all source lines, from the previous program module (if any) through program module *pn*, in which fatal errors have been detected. *pn* will be equivalent to the name used on a particular IDENT pseudo statement. Correct all fatal errors. See the APML Assembler Version 1 Reference Manual, CRI publication SM-0036, for a list of fatal errors. From APML. Class, fatal.

AP012 - [APML] MISSING IDENT STATEMENT
An END pseudo on the source dataset occurred before an IDENT pseudo instruction was found. Check the source dataset for matching IDENT and END pseudo instructions. From APML. Class, warning.

AP013 - [APML] MISSING END STATEMENT, PROGRAM MODULE *pname*
On the source dataset, an end of file occurred before an END pseudo instruction corresponding to the IDENT pseudo in program module *pname*. *pname* will be equivalent to the name used on that IDENT pseudo statement. Check the source dataset for matching IDENT and END pseudo instructions. From APML. Class, warning.

AP014 - [APML] EMPTY SOURCE FILE, DN = *dname*
An end of file or end of data was encountered on the source dataset before any source statements. Check the job control statements and the source dataset for a problem that causes a null file. From APML. Class, warning.

alternative forms:
AP015 - [APML] 1 LINE EXCEEDS 90 CHARACTERS, DN = *dn*
AP015 - [APML] *n* LINES EXCEED 90 CHARACTERS, DN = *dn*
The given number of records in the named dataset contain more than 90 characters. The most typical cause is UPDATE sequence numbers extending past column 90. (APML truncates the long records to 90 characters.) This message can also be issued when a binary dataset is erroneously read. If the records exceed 90 characters, it might be possible to break up the long records with continuation lines. From APML. Class, warning.

AP016 - [APML] OPEN ERROR, DN = *dn*
The dataset *dn* was not found in the user's local environment or in the system directory. Access or create the dataset *dn*. From APML. Class, abort.

AP017 - [APML] INVALID CPU TYPE SPECIFIED: *cpu*
The invalid CPU parameter *cpu* was passed on the APML control statement. Correct the CPU type on the APML job control statement. From APML. Class, warning.

AP030 - [APML] BAD BINARY TEXT, DN = *dn*, (ERROR CODE = *cc*)
An error was discovered in the binary system text *dn*. Following are the error codes and their meanings:

Code	Meaning
P1	Prologue field BSTTT ≠ 1
P2	Prologue field BSTWC < LE@BSTPR
P3	EOR was encountered while prologue was being read

Code	Meaning
P4	EOF, EOD, or null record was encountered while the prologue was being read
H1	EOF, EOD, or null record was encountered while a subtable header was being read
H2	Header field BSTTT ≠ 1
H3	Header field BSTWC < 1
H4	Header field BSTID is not recognized
M1	EOR was encountered while TMDF was being read
M2	EOF, EOD, or null record was encountered while TMDF was being read
M3	Length of TMDF entry < 0
M4	Length of TMDF entry = 0
M5	Global word count exceeded during TMDF processing
S1	EOR was encountered while TSYM entry was being read
S2	EOR, EOD, or null record was encountered while TSYM entry was being read
S3	Global word count was exceeded during TSYM processing
E1	Epilogue field BSTWC ≠ 1
E2	Global word count ≠ sum of subtable word counts

Do one of the following:

- Generate a new binary system text from the original source system text and rerun the job with the new binary system text.
- Rerun the job, replacing the binary system text with the source system text.
- Show the listing and DSDUMP output of the invalid binary system text to a Cray Research analyst.

From APML. Class, fatal.

AP031 - [APML] *symbol* DOUBLY-DEFINED IN BINARY TEXT *dn*
The named *symbol* is defined in the named binary system text but was defined differently in a previous system text. Remove one of the definitions from the source system texts, generate a new binary system text, and resubmit job. From APML. Class, fatal.

AP032 - [APML] MACRO *opsyn* NOT FOUND, BINARY TEXT *dn*
The named binary system text contains an OPSYN directive of the form name OPSYN *opsyn*, but no macro or pseudo-op with the name *opsyn* is known to the assembler. Do one of the following:

- Correct the spelling of *opsyn*.
- Remove the OPSYN from the named system text.
- Define the *opsyn* macro in a previous system text or before the OPSYN directive in the named system text. From APML. Class, fatal.

AP033 - [APML] MACRO *mn* REDEFINED IN BINARY TEXT *dname*
A definition for the named macro appears in the named dataset, but the macro has been previously defined. If the redefinition is intentional, the new definition will be used; otherwise, remove the unwanted macro definition. From APML. Class, warning.

AR000 - BAD CALL TO ARLIB ERROR PROCESSOR
The $ARLIB error message processor was entered with an illegal error message code. See a Cray Research analyst. From $ARLIB. Class, 2; user abort; reprievable.

AR001 - EXPONENT OVERFLOW
A floating-point number being converted by a format on input has an exponent larger than 2466. Check the format and the data line. Count the columns. Trailing and embedded blanks are interpreted as 0. From $ARLIB. Class, 2; user abort; reprievable.

AR002 - EXPONENT UNDERFLOW
A floating-point number being converted by a format on input has an exponent less than -2466. Check format and data line. Trailing and embedded blanks are interpreted as 0. From $ARLIB. Class, 2; user abort; reprievable.

AR003 - DIVIDE BY ZERO
An attempt was made to perform an integer divide by 0 or an attempt was made to use the integer MOD function with 0 for the second argument. Check for program error. From $ARLIB. Class, 2; user abort; reprievable.

AR004 - BAD SCALAR ARGUMENT TO ARLIB MATH ROUTINE
A scalar ARLIB math routine was called with one or more arguments that were out of range for the particular function (for example, SQRT(-5.), ATAN(0,0.)). Consult traceback listing to determine function. Check for proper use of function. Note that some pseudo vector library routines call a vector routine to process a vector one element at a time. From $ARLIB. Class, 2; user abort; reprievable.

AR005 - BAD VECTOR ARGUMENT TO ARLIB MATH ROUTINE
A vector ARLIB math routine was called with one or more elements of its first argument out of range for the particular function (for example, SQRTV(-5.), ATANV(0,0.)). This message does not imply that the second argument is not in error. No further check is made. Consult traceback listing to determine function. Check for proper use of function. Note that some pseudo vector library routines call a vector routine to process a vector one element at a time. From $ARLIB. Class, 2; user abort; reprievable.

AR006 - BAD SECOND VECTOR ARGUMENT TO ARLIB MATH ROUTINE
A vector ARLIB math routine was called with one or more elements of its second argument out of range for the particular function. This message does not imply that the first argument is not in error. No further check is made. Consult traceback listing to determine function. Check for proper use of function. Note that some pseudo vector library routines call a vector routine to process a vector one element at a time. From $ARLIB. Class, 2; user abort; reprievable.

AU002 - DSC READ ERROR
AUDIT encountered an error while reading the Dataset Catalog. See a Cray Research analyst. From AUDIT. Class, 2; user abort; reprievable.

alternative forms:
AU003 - *d* DATASETS, *b* BLOCKS, *w* WORDS
AU003 - NO DATASETS SELECTED
AUDIT reports to the user the number of datasets, blocks, and words it processed from the control statement selection requirements. From AUDIT. Class, informative.

AU004 - INVALID CONTROL WORD SPECIFIED
AUDIT detected a CW parameter that did not match the control word required by AUDIT. From AUDIT. Class, user abort.

AU006 - *n* IS AN UNKNOWN LO OPTION
AUDIT detected an LO parameter that did not match any of the possible list option selections. From AUDIT. Class, user abort.

AU007 - *n* IS AN UNKNOWN BO OPTION
AUDIT detected a BO parameter that did not match any of the possible binary output selections. From AUDIT. Class, user abort.

AU008 - LO=S EXCLUDES OTHER LO OPTIONS
AUDIT encountered an LO=S parmeter along with one or more other LO options. If 'S' is selected, no other LO options are allowed. From AUDIT. Class, User abort.

AU009 - DXT READ ERROR
AUDIT encountered an error while reading the Dataset Catalog Extension. See Cray Research analyst. From AUDIT. Class, 2; user abort; reprievable for a very short duration.

AU010 - US CANNOT BE SPECIFIED WITHOUT CW
AUDIT encountered a US parameter specified and no CW parameter existed on the control statement. This error is for non-private systems only. From AUDIT. Class, user abort.

AU011 - *n* IS AN UNKNOWN ACC OPTION
AUDIT detected an ACC parmeter that did not match any of the possible access mode selections. From AUDIT. Class, user abort.

BD001 - TABLE SIZE IS LARGER THAN 4000
The number of modules built into a new library exceeds 4000. See a Cray Research analyst. From BUILD. Class, fatal.

BD003 - BAD MODULE SKIPPED: *modname*
A bad module, in which the PDT had a fatal error flag set, was encountered. Exclude the module. From BUILD. Class, warning.

BD004 - DATASET IS NOT LOCAL: *dn*
The dataset for parameter OBL or B is not local. Make the dataset local to the job. From BUILD. Class, fatal.

NOTE

Message classes for AP and CA messages, issued by APML and CAL, are different from those for other messages. See the note preceding the AP messages.

CA000 - [CAL] INTERNAL 'CAL' ERROR DETECTED AT P = *paddress*
CAL detected an internal error at parcel address *paddress* and is unable to proceed. See a Cray Research analyst. From CAL. Class, abort (immediate).

CA001 - [CAL] CAL VERSION *x* (*m/d/y*) - *cpu*
At the beginning of each assembly CAL issues an informative message indicating the version number *x*, the date *m/d/y* on which CAL was assembled, and the *cpu* that CAL is assembling for. From CAL. Class, informative.

CA002 - [CAL] ASSEMBLY TIME: *n* CPU SECONDS
All programs in the current file of the source dataset have been assembled. *nnnnn.nnnn* is the assembly time in floating-point CPU seconds. From CAL. Class, informative.

CA003 - [CAL] MEMORY WORDS: *mw* + I/O BUFFERS: *bw*
All programs in the current file of the source dataset have been assembled. *mw* is the decimal number of memory words required in the user portion of the job field. *bw* is the decimal number of words needed for the I/O table and buffer area of this job step. From CAL. Class, informative.

CA004 - [CAL] ASSEMBLY ERRORS
If the user set the ABORT flag on the CAL control statement and fatal errors were encountered during assembly, then CAL issues this message followed by an abort. Either remove the ABORT flag from the CAL control statement or correct all fatal errors found by CAL. From CAL. Class, abort.

alternative forms:
CA010 - [CAL] 1 WARNING ERROR, PROGRAM MODULE *pn*
CA010 - [CAL] *n* WARNING ERRORS, PROGRAM MODULE *pn*
CAL issues this message from the previous program module (if any) through program module *pn*, in which warning errors have been detected. *pn* will be equivalent to the name used on the most recent IDENT pseudo statement. Examine all warning errors and correct if necessary. See the CAL Assembler Version 1 Reference Manual, publication SR-0000, for a list of warning errors. From CAL. Class, warning.

alternative forms:
CA011 - [CAL] 1 FATAL ERROR, PROGRAM MODULE *pn*
CA011 - [CAL] *n* FATAL ERRORS, PROGRAM MODULE *pn*
CAL issues this message from the previous program module (if any) through program module *pn*, where fatal errors were detected. *pn* is equivalent to the name used on the most recent IDENT pseudo statement. Correct all fatal errors. See the CAL Assembler Version 1 Reference Manual, publication SR-0000, for a list of fatal errors. From CAL. Class, fatal.

CA012 - [CAL] MISSING IDENT STATEMENT
An END pseudo on the source dataset occurred before an IDENT pseudo instruction was found. Check the source dataset for matching IDENT and END pseudo instructions. From CAL. Class, warning.

CA013 - [CAL] MISSING END STATEMENT, PROGRAM MODULE *pn*
On the source dataset, an end of file occurred before an END pseudo instruction corresponding to the IDENT pseudo in program module *pn*. *pn* is equivalent to the name used on that IDENT pseudo statement. Check the source dataset for matching IDENT and END pseudo instructions. From CAL. Class, warning.

CA014 - [CAL] EMPTY SOURCE FILE, DN = *dn*
An end of file or end of data was encountered on the source dataset before any source statements. Check the job control statements and the source dataset for a problem that causes a null file. From CAL. Class, warning.

alternative forms:
CA015 - [CAL] 1 LINE EXCEEDS 90 CHARACTERS, DN = *dn*
CA015 - [CAL] *n* LINES EXCEED 90 CHARACTERS, DN = *dn*
The given number of records in the named dataset contain more than 90 characters. The most typical cause is UPDATE sequence numbers that extend past column 90. (CAL truncates the long records to 90 characters.) This message can also be issued when a binary dataset is erroneously read. From CAL. Class, warning.

CA016 - [CAL] OPEN ERROR, DN = *dn*
The dataset *dn* was not found in the user's local environment or in the system directory. Access or create the dataset *dn*. From CAL. Class, abort.

CA017 - [CAL] INVALID CPU TYPE SPECIFIED: *cpu*
The invalid CPU parameter *cpu* was passed on the CAL control statement. CAL defaults to the machine currently executing. Correct the CPU type on the CAL job control statement. From CAL. Class, warning.

CA030 - [CAL] BAD BINARY TEXT, DN = *dn*, (ERROR CODE = *ec*)
An error was discovered in the binary system text *dn*. Following are the error codes and their meanings.

Code	Meaning
P1	Prologue field BSTTT ≠ 1
P2	Prologue field BSTWC < LE@BSTPR
P3	EOR was encountered while prologue was being read
P4	EOF, EOD, or null record was encountered while the prologue was being read
H1	EOF, EOD, or null record was encountered while a subtable header was being read
H2	Header field BSTTT ≠ 1
H3	Header field BSTWC < 1
H4	Header field BSTID is not recognized
M1	EOR was encountered while TMDF was being read
M2	EOF, EOD, or null record was encountered while TMDF was being read
M3	Length of TMDF entry < 0
M4	Length of TMDF entry = 0
M5	Global word count exceeded during TMDF processing
S1	EOR was encountered while TSYM entry was being read
S2	EOR, EOD, or null record was encountered while TSYM entry was being read
S3	Global word count was exceeded during TSYM processing
E1	Epilogue field BSTWC ≠ 1
E2	Global word count ≠ sum of subtable word counts

Do one of the following:

- Generate a new binary system text from the original source system text and rerun the job with the new binary system text.
- Rerun the job, replacing the binary system text with the source system text.
- Show the listing and DSDUMP output of the invalid binary system text to a Cray Research analyst.

From CAL. Class, fatal.

CA031 - [CAL] *symbol* DOUBLY-DEFINED IN BINARY TEXT *dname*
The named *symbol* is defined in the named binary system text but is defined differently in a previous system text. Remove one of the definitions from the source system texts, generate a new binary system text, and resubmit the job. From CAL. Class, fatal.

CA032 - [CAL] MACRO *opsyn* NOT FOUND, BINARY TEXT *dname*
The named binary system text contains an OPSYN directive of the form name OPSYN *opsyn*, but no macro or pseudo-op with the name *opsyn* is known to the assembler. Do one of the following:

- Correct the spelling of *opsyn*.
- Remove the OPSYN from the named system text.
- Define the *opsyn* macro in a previous system text or before the OPSYN directive in the named system text.

From CAL. Class, fatal.

CA033 - [CAL] MACRO *mn* REDEFINED IN BINARY TEXT *dn*
A definition for the named macro appears in the named dataset, but the macro was previously defined. If the redefinition is not intended, remove the unwanted macro definition. From CAL. Class, warning.

CA034 - [CAL] OPDEF *oname* REDEFINED IN BINARY TEXT *dn*
A definition for the named opdef appears in the named dataset, but the opdef's syntax has been previously defined. If the redefinition is not intended, remove the unwanted opdef definition. From CAL. Class, warning.

CB001 - FTREF VERSION *version*
The indicated version of FTREF was built on the indicated date. From FTREF. Class, informative.

CB002 - INPUT DATASET MUST BE SPECIFIED
The parameter I is not specified or equal to 0. I is a required parameter. From FTREF. Class, fatal.

CB003 - DATASET IS NOT LOCAL: *dn*
The dataset for I or DIR is not local to the job. From FTREF. Class, fatal.

CB004 - SPECIFIED MAIN ROUTINE IS NOT IN THE PROGRAM
The module specified by parameter ROOT is not in the input source. From FTREF. Class, informative.

CB005 - STACK OVERFLOW, MORE THAN 100 LEVELS
The static tree is more than 100 levels deep. From FTREF. Class, informative.

CB006 - NO CROSS REFERENCE TABLE, CHECK IF CFT,ON=XS
There is no cross reference table for at least one of the modules. This error occurs when ON=XS is not specified on the CFT control card. From FTREF. Class, informative.

CB007 - ERROR IN THE INPUT FILE
Read error in the input file or directive file. See site analyst. From FTREF. Class, fatal.

CB008 - PARAMETER MORE THAN 8 CHRS - parameter
The parameter in the directive is more than 8 characters long. From FTREF. Class, fatal.

CB009 - KEYWORD FOR DIRECTIVE IS NOT FOUND
The keyword was missing in the directive. From FTREF. Class, fatal.

CB010 - UNKNOWN DIRECTIVE : *directive*
An unknown keyword was used in the directive. From FTREF. Class, fatal.

CB011 - UNEXPECTED END OF FILE, CHECK IF CFT,ON=XS
The external table is not the last listing in the input file. This error occurs when ON=XS is not specified on the CFT control card, or there is a compiler directive CDIR$ NOLIST in the source. From FTREF. Class, informative.

CB012 - DIRECTIVE TERMINATES AT 80-TH CHARACTER
The directive has more than 80 characters. FTREF truncates the characters after the 80th column. From FTREF. Class, informative.

CC001 - CONTROL STATEMENT TERMINATOR MISSING
A terminator character - either a period or a right parenthesis - was expected and not found in the control statement. Correct the control statement and resubmit the request. From $SYSLIB. Class, not reprievable.

CC003 - CONTROL STATEMENT TABLE OVERFLOW - USE CONTINUATION CARDS
In processing the control statement, CCS exceeded the size of the expanded control statement area in the JCB. Separate the control statement parameters into

additional statements and reissue the request. From $SYSLIB. Class, not reprievable.

CC004 - CONTROL STATEMENT CONTAINS ILLEGAL CHARACTER
A character in the control statement is not in the range of allowable characters. Correct the control statement and reissue the request. From $SYSLIB. Class, not reprievable.

CC005 - NO CONTROL STATEMENT IN JCB
The $CCS routine was called but the control statement field in the JCB was empty. Correct the program that called $CCS and resubmit the job. From $SYSLIB. Class, not reprievable.

CC006 - MISSING OR EXTRA OPEN PARENTHESES
The control statement does not contain balanced parentheses. Ensure that every open parenthesis is paired with a closed parenthesis. From $SYSLIB. Class, not reprievable.

CF000 - CFT VERSION - *m/d/y x*
Initiation of CFT. The indicated version of CFT (for example, 1.09) was assembled on the indicated date (for example, 02/29/80). From CFT compile time. Class, informative.

CF001 - COMPILE TIME = *s* SECONDS
Termination of compilation From CFT. Class, informative.

CF002 - *l* LINES, *s* STATEMENTS
Termination of compilation. *l* lines were compiled, comprising *s* FORTRAN statements. From CFT. Class, informative.

CF003 - *w* WORDS, *b* I/O BUFFERS USED
Completion of compilation. At termination, *w* words were occupied by the user portion of the job field. *w* words were required in the I/O table and buffer area of the job field. From CFT. Class, informative.

alternative forms:
CF004 - 1 ERROR
CF004 - *n* ERRORS
One or more errors were encountered during compilation. Fix the errors and recompile. From CFT. Class, informative.

CF005 - JOB ABORTED
Fatal errors were encountered during compilation, and ON=A is in effect. Correct errors and recompile. From CFT. Class, informative.

CF006 - LINES-PER-PAGE < *n*; *n* USED
An OPTION control statement set LPP to less than the minimum accepted by CFT. Use the OPTION control statement to set the number of lines needed per page. From CFT. Class, informative.

CF007 - BAD PARAMETER TO KEYWORD
keyword = param
The parameter is out of range or undefined for the keyword. Correct the parameter. From CFT. Class, fatal.

CF008 - NULL INPUT FILE ILLEGAL
I=0 was specified on CFT control statement. Correct the input parameter. From CFT. Class, fatal.

CF009 - B=0 and ON=Z INCOMPATIBLE OPTIONS
B=0 and ON=Z were specified on CFT control statement. Change one or both parameters. From CFT. Class, fatal.

alternative forms:
CF010 - ON = *char* PARAMETER NOT ALPHA
CF010 - OFF = *char* PARAMETER NOT ALPHA
Nonalphabetic character in ON or OFF character string Delete the bad character from the string. From CFT. Class, fatal.

CF011 - *letter* OPTION NOT IMPLEMENTED
No existing ON/OFF option is associated with the letter. Delete the letter from the string. From CFT. Class, fatal.

alternative forms:
CF012 - 01 CFT CONTROL CARD ERROR
CF012 - *nn* CFT CONTROL CARD ERRORS
One or more errors were encountered in the CFT control statement. Fix any errors. From CFT. Class, fatal.

CF013 - WARNING: *string* WILL BE SET TO OFF
Options listed in *string* appear in both the ON= and OFF= keyword parameter lists. From CFT. Class, warning.

CF014 - DOUBLY DEFINED OPTION FOR OPT= KEYWORD
An option set by an OPT= keyword parameter was defined two times or redefined in the parameter list. From CFT. Class, fatal.

CF015 - HEAP BASED ALLOCATION NOT YET IMPLEMENTED
ALLOC = HEAP was specified on the CFT control statement. HEAP memory management is not yet implemented by CFT. From CFT. Class, fatal.

CF016 - CPU TYPE UNKNOWN - MAY AFFECT GENERATED CODE
The CPU type obtained from the Job Communication Block is unknown to CFT. Certain optimizations may be affected. From CFT. Class, warning.

alternative forms:
CF017 - 1 WARNING
CF017 - *n* WARNINGS
Warning errors were encountered during compilation. Correct warnings and recompile. From CFT. Class, informative.

CF018 - WARNING: MAXBLOCK WILL BE SET TO 1
When compiling with DEBUG on the CFT control statement, MAXBLOCK will be set to 1. From CFT. Class, warning.

CF019 - WARNING: Z WILL BE SET TO ON
When compiling with DEBUG on the CFT control statement, Z will be set to on. From CFT. Class, warning.

CF020 - WARNING: I WILL BE SET TO ON
When compiling with DEBUG on the CFT control statement, I will be set to on. From CFT. Class, warning.

CF023 - 1 NON-ANSI MESSAGE ISSUED
CF023 - *n* NON-ANSI MESSAGES ISSUED
Non-standard FORTRAN was detected when compiling with the ANSI option. From CFT. Class, informative.

CH001 - ILLEGAL SR PARAMETER OPTION
The SR parameter on the CHARGES control statement was assigned an illegal or unavailable option. Correct the SR parameter option in error. From CHARGES. Class, fatal.

CJ001 - JOB DOESN'T FIT INTO ANY CLASS
Invalid class assignment. Change the job card parameters so that the job can fit into a class. See site operations personnel for class definitions. From CSP. Class, fatal.

CJ002 - CLASS SPECIFIED BY CL PARAMETER NOT FOUND
The class specified by the CL parameter cannot be found. The job aborts. Change the CL parameter to a recognizable class. See site operations personnel for class definitions. From CSP. Class, fatal.

CJ003 - JOB DOESN'T FIT CLASS SPEC'D BY CL PARAMETER
Invalid class assignment by the CL parameter. Modify the CL parameter to fit the job into the class. See site operations personnel for class definitions. From CSP. Class, fatal.

CJ004 - MFL PARAMETER EXCEEDS MAXIMUM ALLOWED
The MFL parameter on the JOB card is too large. Modify the MFL parameter. From CSP. Class, fatal.

CJ005 - MFL PARAMETER INCREASED TO MINIMUM ALLOWED
The MFL parameter on the JOB card is too small. Modify the MFL parameter. From CSP. Class, informative.

CL001 - JOB CLASS STRUCTURE INVOKED: *class*
The job class structure named in the message was invoked. From JSH. Class, informative; system log only.

CL003 - JOB *jn* ASSIGNED TO CLASS *jcn*
P = *n*
A job is assigned to a class. From JSH. Class, informative; system log only.

CM001 - *f* FILES READ - *r* RECORDS READ - *d* DIFFERENCES
This message summarizes the results of COMPARE for one dataset. COMPARE compares complete datasets, giving the number of files, the number of records, and the number of differences in the datasets. If the two datasets have different record or file counts, or modified record synchronization (by content scanning), the results are approximate. From COMPARE. Class, informative.

CM002 - MORE THAN *d* DIFFERENCES. JOB TERMINATED
Number of differences exceeds value specified on ME parameter. Increase the ME parameter on the control statement if more differences are desired. If there are not nore than ME differences, obtain a dump of the job and both datasets and submit to a Cray Research analyst. From COMPARE. Class, 2; user abort; not reprievable.

CM003 - A FILE MAY NOT BE COMPARED TO ITSELF
The value of the A and B parameters on the control statement are the same. This is not legal for COMPARE. Change the value of either A or B. From COMPARE. Class, 2; user abort; not reprievable.

CM004 - OUTPUT MAY NOT BE WRITTEN TO AN INPUT FILE
The value of the L parameter is the same as the value of either the A or the B parameter. Change one of the values. From COMPARE. Class, 2; user abort; not reprievable.

CM005 - CONTEXT MODE NOT IMPLEMENTED FOR BINARY MODE. COMPARE RUN WITHOUT CONTEXT PRINTING
Context printing and binary mode options were both selected on the control statement. This combination is not handled by COMPARE. The comparison is made in binary mode without context printing. Do not select both options at once. From COMPARE. Class, caution.

CM006 - CP < 0 NOT ALLOWED. COMPARE RUN WITHOUT CONTEXT PRINTING
A negative value was illegally selected for the CP parameter. COMPARE does the comparison without context printing. Correct the value of the CP parameter. From COMPARE. Class, caution.

CM007 - COMPARE WIDTH NOT IMPLEMENTED FOR BINARY MODE
Both the CW and the DF=B parameters were selected in the COMPARE statement. Change the CW parameter or the DF parameter. From COMPARE. Class, fatal.

CM008 - PROGRAM ERROR
A program logic error was encountered. See a Cray Research analyst. From COMPARE. Class, fatal.

CS000 - LINE *l* ==> *cardimage*
An error has been detected during JCL block structure validation that concerns this line of the procedure dataset. The exact error message follows this message. Correct the problem as indicated by the error message that follows this message. From CSP. Class, informative.

CS001 - FIRST STATEMENT NOT JOB STATEMENT
First control statement is not a JOB statement; syntax error occurred in JOB statement; job input dataset is not blocked; or required JOB parameters are missing or badly equated. See the CRAY-OS Version 1 Reference Manual, publication SR-0011, for the format of the JOB statement; block the dataset if necessary. From CSP. Class, fatal; not reprievable.

CS003 - SYSTEM ERROR ON RELEASE
An internal COS error was detected by CSP. From CSP. Class, fatal; not reprievable.

CS005 - ILLEGAL PARAMETER VALUE OPTION
One of the following conditions was detected:

- The IOAREA parameter is not LOCK or UNLOCK.
- The NORERUN parameter is not ENABLE or DISABLE.
- The RERUN parameter is not ENABLE or DISABLE.
- The SWITCH parameter value is not ON or OFF.

From CSP. Class, fatal; not reprievable.

CS009 - UNKNOWN VERB, REST OF CONTROL STATEMENT SUPPRESSED
The verb was not found in the system verb list, local dataset list, library, or system directory. Only the verb is echoed because the statement might contain secure information. Check for misspelling in JCL, or access needed datasets. From CSP. Class, fatal; not reprievable.

CS010 - DATASET NAME EXCCEDS 7 CHARACTERS
The dataset name must be less than or equal to seven characters. Rename the dataset. From CSP. Class, fatal.

CS011 - *n* PARAMETER IN ERROR
The JOB statement contains an invalid parameter (for example, an illegal priority). See the CRAY-OS Version 1 Reference Manual, publication SR-0011, for the format of the JOB statement. From CSP. Class, fatal; not reprievable.

CS015 - BAD LIBRARY DIRECTORY: *n*
A user-defined library dataset (from the LIBRARY control statement) has a bad directory table. Either the named dataset is not a library or the directory for the named dataset was destroyed. From CSP. Class, fatal; not reprievable.

CS026 - PROC REDEFINED; FORMER DEFINITION LOST
The user redefined a procedure; the new definition replaces the old definition. From CSP. Class, informative.

CS030 - DUPLICATE * SPECIFICATION ILLEGAL
Library statement contains more than one asterisk. Remove all but one asterisk and resubmit. From CSP. Class, fatal.

CS031 - INVALID DATASET NAME: *dn*
The name indicated is not a valid COS dataset name. Correct the procedure body. From CSP. Class, fatal.

CS032 - DATASET NOT LOCAL: *dn*
The dataset is not local; access or create it to make it local. From CSP. Class, fatal.

CS033 - NO PROCEDURE ACTIVE - RETURN IGNORED
The user was not executing a procedure when a return was called. From CSP. Class, informative.

CS034 - NO ACCOUNT STATEMENT - ACCOUNTING MANDATORY
Accounting is mandatory; the job did not contain an ACCOUNT statement. Put an ACCOUNT statement in the job deck. From ACCOUNTING. Class, fatal.

CS035 - INVALID LOCATION OF ACCOUNT STATEMENT
The ACCOUNT statement did not appear after the JOB card. Place the ACCOUNT card immediately following the JOB card. From ACCOUNTING. Class, fatal.

CS036 - SYNTAX ERROR ON ACCOUNT STATEMENT
A syntax error was encountered on the ACCOUNT statement. Correct the ACCOUNT statement. From ACCOUNTING. Class, fatal.

CS038 - *verb* DOES NOT APPEAR WITHIN A *blocktype*
A statement that has either ELSE, ELSEIF, or EXITLOOP as the verb does not appear within the appropriate conditional or iterative block. If the statement is

intended to be a component of a block, then place it within the block. From CSP. Class, not reprievable; CSP abort.

CS039 - INVALID ELSE IN CONDITIONAL BLOCK
Either an ELSE has appeared before an ELSEIF or more than one ELSE is in the conditional block. Place all ELSEIF blocks before the ELSE block or remove the extra ELSE block. From CSP. Class, fatal; not reprievable.

CS040 - INCOMPLETE CONSTRUCTION OF *blocktype*
A block type of CONDITIONAL, ITERATIVE, or PROCEDURE DEFINITION did not have an ending block delimiter of ENDIF, ENDLOOP, or ENDPROC, respectively. Ensure the proper occurrence of the ending block delimiter. From CSP. Class, fatal; not reprievable.

CS041 - UNEXPECTED END OF CONTROL STATEMENT FILE
CSP has detected the end of the control statement file while it was skipping to the end of a JCL block. This condition should have been detected during block validation. Consult a Cray Research analyst. From CSP. Class, not reprievable.

CS042 - STRUCTURAL ERROR IN CONTROL STATEMENT FILE --- PROCEDURE NOT INVOKED
An error was detected within the block structure of a procedure that is to be invoked. From CSP. Class, fatal; not reprievable.

CS043 - WARNING ITERATIVE BLOCK DOES NOT CONTAIN AN EXITLOOP STATEMENT
A LOOP/ENDLOOP control statement sequence does not contain an EXITLOOP statement, causing the sequence to be iterated until a job step abort occurs. Add an EXITLOOP statement if desired. From CSP. Class, informative.

CS044 - INVALID LIBRARY DATASET NAME: *dsname*
The name is not a valid dataset name and therefore cannot be located. Remove the name from the library searchlist and correct its entry on the library searchlist. From CSP. Class, informative.

CS045 - NON-EXECUTABLE STATEMENT ENCOUNTERED
An interactive job tried to enter, via the terminal, a control statement that corresponds to a JCL block structure. Do not use block structure statements from an interactive terminal. From CSP. Class, fatal; not reprievable.

CS046 - JCL EXPRESSION IS TOO COMPLICATED
The expression has overflowed into the internal tables of the expression evaluator. Use parentheses to simplify the statement. From CSP. Class, fatal; not reprievable.

CS047 - CSP INTERNAL PROCESSING ERROR
The Control Statement Processor has detected its own error. Consult a Cray Research analyst. From CSP. Class, fatal; not reprievable.

CS048 - LIBRARY SEARCHLIST OVERFLOW
An attempt was made to expand the number of names in the library searchlist beyond 64. Consolidate the libraries. From CSP. Class, fatal; not reprievable.

CS049 - ILLEGAL CONTINUED CONTROL STATEMENT
A control statement was encountered in which the verb was followed by a continuation character. Place the continuation character after the statement's initial separator. From CSP. Class, fatal; not reprievable.

CS050 - EXPRESSION ERROR
- *reason*
- NEAR *data*

One of the following conditions occurred, as indicated by the *reason*: UNBALANCED PARENTHESES; MISSING OPERATOR; ILLEGAL OPERATOR; ILLEGAL NUMBER; ILLEGAL SYMBOL; ILLEGAL LITERAL STRING; SYNTAX ERROR; OPERATOR NOT UNARY; EXPRESSION TOO COMPLICATED; LITERAL TOO LONG; TABLE OVERFLOW: SIMPLIFY. Correct the error indicated by *reason*. From $SYSLIB. Class, not reprievable.

CS051 - COMMAND IGNORED. ENTER ACCOUNT STATEMENT
The first statement of an interactive session is not the ACCOUNT statement, and accounting is mandatory. Enter an ACCOUNT statement. From CSP. Class, informative.

CS052 - RE-ENTER ACCOUNT STATEMENT
There is a syntax error on the ACCOUNT statement of an interactive job. Reenter the ACCOUNT statement. From CSP. Class, informative.

CS053 - UNABLE TO EXPAND JTA WHILE ACCESSING SDR
There is insufficient memory for the SDR list in the job's JTA. From CSP. Class fatal.

CS056 - MODE STATEMENT ILLEGAL FOR NON-XMP MAINFRAMES
The user attempted to enable or disable the bi-directional memory mode feature, which applies only to X-MP mainframes. From CSP.

CS101 - BAD CHARACTER IN PROTOTYPE STATEMENT
A character in the prototype statement is not in the ASCII graphic set. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS103 - UNTERMINATED LINE IN PROTOTYPE STATEMENT
The prototype statement does not end with a terminator (period or right parenthesis). Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS104 - UNTERMINATED LITERAL IN PROTOTYPE STATEMENT
A closing apostrophe is missing from a literal string specification. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS105 - PROTOTYPE SEPARATOR OUT OF SEQUENCE AFTER *n*
The prototype statement contains a bad delimiter usage such as *key* = = ... or *key* = A/B/C. *n* is near the invalid sequence. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS106 - RESERVED NAME DATA USED IN PROTOTYPE
The name DATA is reserved. It cannot be used in the prototype as a positional parameter or as a keyword. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS107 - DUPLICATED PARAMETER NAME IN PROTOTYPE: *n*
A keyword or a positional parameter specified by the prototype was not unique. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS108 - PARAMETER TOO LONG IN PROTOTYPE: *n*
A keyword or a positional parameter specified by the prototype was longer than eight characters. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS109 - POSITIONAL PARAMETER AFTER KEYWORDS IN PROTOTYPE: *n*
A positional parameter followed a keyword parameter in the prototype statement. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS110 - NULL PARAMETER IN PROTOTYPE STATEMENT
A keyword or a positional parameter specified by the prototype had a null value. Correct the procedure prototype statement. From CSP. Class, fatal; not reprievable.

CS111 - BAD CHARACTER IN PROCEDURE INVOCATION
Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS113 - UNTERMINATED LINE IN PROCEDURE INVOCATION
Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS114 - UNBALANCED PARENTHESIS IN PROCEDURE INVOCATION
Correct the procedure call. From CSP. Class, fatal.

CS115 - SEPARATOR OUT OF SEQUENCE AFTER: *n*
Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS116 - UNRECOGNIZED KEYWORD: *n*
A keyword specified by the invocation has no matching keyword in the prototype. Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS117 - KEYWORD TOO LONG: *n*
A keyword is more than eight characters long. Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS118 - POSITIONAL PARAMETER AFTER KEYWORDS: *n*
A positional parameter followed the introduction of keyword parameters in the invocation. Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS119 - EXTRA POSITIONAL PARAMETER: *n*
Too many positional parameters were specified by the invocation statement. Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS120 - DUPLICATED KEYWORD: *n*
The indicated keyword appears two or more times in the invocation. Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS121 - KEYWORD USED WITHOUT ASSIGNING IT A VALUE: *n*
A non-null value was not provided for the indicated keyword. Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS122 - NO VALUE WAS ASSIGNED TO *n*
A non-null value was not provided for the indicated keyword. Correct the procedure invocation statement. From CSP. Class, fatal; not reprievable.

CS123 - &DATA NOT DELIMITED PROPERLY
The &DATA statement was used incorrectly in the procedure body. Correct the procedure body statement. From CSP. Class, fatal; not reprievable.

CS124 - &DATA NOT AT BEGINNING OF LINE
The keyword &DATA appears within another statement. Correct the procedure body. From CSP. Class, fatal.

CS125 - NO SUCH FORMAL PARAMETER: *n*
The indicated string was preceded by an ampersand, signaling a substitutable string. However, the string was not specified by the prototype statement. Correct the procedure body statement. From CSP. Class, informative.

CS126 - ZERO LENGTH PARAMETER NAME
A substitutable string has zero length. Correct the procedure body statement. From CSP. Class, fatal; not reprievable.

CS127 - PARAMETER NAME TOO LONG
A substitutable string is more than eight characters long. Correct the procedure body statement. From CSP. Class, fatal; not reprievable.

CS131 - ZERO-LENGTH &DATA FILENAME
The &DATA argument was improperly specified. Correct the procedure body statement. From CSP. Class, fatal; not reprievable.

CS132 - &DATA FILENAME TOO LONG: *n*
The &DATA argument was improperly specified. Correct the procedure body statement. From CSP. Class, fatal; not reprievable.

CS133 - UNEXPECTED END OF CONTROL STATEMENT FILE
The control statement ended prematurely while the procedure invocation was being interpreted. From CSP. Class, fatal; not reprievable.

CS134 - UNEXPECTED END OF PROCEDURE FILE
Premature end of procedure prototype and procedure body file occurred. Check the procedure file for completeness. From CSP. Class, fatal; not reprievable.

CS135 - ILLEGAL CHARACTER IN &DATA DSN: *n*
n is an illegal character encountered while the &DATA dataset name was being interpreted. Correct the dataset name. From CSP. Class, fatal.

CS136 - &DATA CREATED DSN *dn*
A &DATA JCL statement was executed, creating the dataset name. From CSP. Class, informative.

CT000 - JCB DESTROYED, JCSTN RESET
A user program overwrote at least one word in the Job Control Block that stores the step-failed information. Reorganize the job stream or remove the invalid step; resubmit the job. From STEP. Class, informative.

CT000 - PASSED
All job steps completed normally. From STEP. Class, informative.

CT000 - TEST FAILED
The LAST= parameter on the STEP control statement was specified as FAIL. The test failed. From STEP. Class, informative.

CT000 - TEST INDETERMINATE
The LAST= parameter on the STEP control statement was specified as INDET. Therefore, it is impossible to determine whether the test passed. From STEP. Class, informative.

DB001 - SYMBOL FILE IS NOT AVAILABLE
The dataset used with the S parameter is not local to the job. Check the spelling of the dataset name or access the dataset. From SID. Class, not reprievable; SID abort.

DB002 - INPUT FILE IS NOT AVAILABLE
The dataset used with the I parameter is not local to the job. Check the spelling of the dataset name, or access the dataset. From SID. Class, not reprievable; SID abort.

DB003 - CNT VALUE IS NOT A POSITIVE INTEGER
The value given for the CNT parameter was negative or contained non-numeric characters. Correct the value for CNT. From SID. Class, not reprievable; SID abort.

DB004 - OVERLAPPING BREAKPOINTS
A breakpoint was placed inside the debugger code or a library routine used by the breakpoint routines was stepped into. Avoid stepping into library routines and check breakpoint addresses carefully if symbolic addressing is not used. From SID. Class, not reprievable; SID abort.

DB005 - COUNT DECREMENTED TO ZERO
The variable initialized by the CNT parameter was decremented to 0; this variable is decremented once each time a breakpoint is reached. From SID. Class, not reprievable; SID abort.

DB006 - EMPTY CONTROL STATEMENT FILE
The control statement file was empty when SID, running in batch mode, tried to read the control statement for the user program. Add a control statement for the user program to the control statement file immediately following the statement that invokes SID. From SID. Class, not reprievable; SID abort.

DD001 - INITIAL SECTOR BEYOND END OF DATASET
The IS parameter value given on the control statement is greater than the size of the input dataset. Correct the IS value. From DSDUMP. Class, fatal; job step is terminated.

DD002 - BAD DF PARAMETER
DD003 - BAD DB PARAMETER
DD004 - BAD DSZ PARAMETER
The specified parameter value given on the control statement is invalid. Correct the value. From DSDUMP. Class, fatal; job step is terminated.

DD005 - I*=0 NOT VALID WITH NO Z PARAMETER
One or more of the IW, IS, IR, or IF parameters was set to 0, but the Z parameter, which allows zero-based addressing was not set. Either correct the I* value(s) or include the Z parameter. From DSDUMP. Class, fatal; job step terminated.

DD006 - BAD VALUE FOR I* PARAMETER
One or more of the IW, IS, IR, or IF parameters was set to a negative value or includes a non-numeric character other than a base indicator. Correct the I* value(s). From DSDUMP. Class, fatal; job step terminated.

DD007 - BAD VALUE FOR N* PARAMETER
One or more of the NW, NS, NR, or NF parameters was set to a negative value, or to 0, or includes a non-numeric character other than a base indicator. Correct the value(s). From DSDUMP. Class, fatal; job step terminated.

DI008 - DEFECTIVE $PDDERR FILE
The $PDDERR file (as either an SDR dataset or as a local dataset) cannot be read successfully to interpret a PDD error code. The file exists, but contains a record too long or has suffered other damage. From Software Tools.

DP101 - SYSTEM ERROR ON READING *dn*
DUMP asked the operating system to move part of $DUMP from disk to the I/O buffer and to suspend DUMP until the move was completed. When DUMP was resumed, the Dataset Parameter Table (DSP) indicated that no data was put into the buffer. Determine that the DUMPJOB statement precedes the DUMP statement or that the input dataset has been properly created and accessed. Otherwise, report this to the Cray Research analyst. From DUMP. Class, fatal; user-requested abort.

DP102 - JCB POINTERS IN *dn* ARE INVALID SO DUMP CANNOT LOCATE THE LFT AND DSP'S
When DSP is specified on the DUMP statement, DUMP uses the information in the Job Communication Block (JCB) to locate the LFT and DSPs in $DUMP. Before using this information, DUMP validates it to be sure that it points to the correct area. Use the FW and LW parameters to dump at least words 0 through 200 to see what information has overwritten the JCB. This should give a clue as to what part of the job is storing information in the wrong area. From DUMP. Class, informative; processing continues.

DP104 - END OF DUMP
DUMP has completed printing all requested information. From DUMP. Class, informative.

DP110 - FORMAT EXCHANGE PACKAGE CALLED WITH DSP ADDRESS=0
The Dataset Parameter Table address for the listing dataset is a required parameter for the subprogram that formats the Exchange Package. A value of 0 is not legal. If the user program calls subprogram FXP, check the code to determine if the DSP address is passed to FXP. If it is a standard CRI product, such as DUMP, that has called FXP, report this to a Cray Research analyst. From DUMP. Class, fatal; user-requested abort.

DP111 - FORMAT B REGISTERS CALLED WITH DSP ADDRESS=0
See message DP110; substitute FBR for FXP. From DUMP. Class, fatal; user-requested abort.

DP112 - FORMAT T REGISTERS CALLED WITH DSP ADDRESS=0
See message DP110; substitute FTR for FXP. From DUMP. Class, fatal; user-requested abort.

DP113 - FORMAT V REGISTERS CALLED WITH DSP ADDRESS=0
See message DP110; substitute FVR for FXP. From DUMP. Class, fatal; user-requested abort.

DP115 - END OF INPUT REACHED BEFORE LW
DUMP uses the LW parameter value from the DUMP control statement and the BA and LA values from the Exchange Package to determine when to stop reading $DUMP for more memory image. An error occurred when DUMP calculated that there was more to read and the system calculated that there was no more to read. Determine if $DUMP was properly created by the DUMPJOB statement. If it was, report this to a Cray Research analyst. From DUMP. Class, fatal; user-requested abort.

DP120 - ILLEGAL FORMAT TYPE :*f* * , ASSUMING O
DUMP gives the option of printing the memory in several different formats. The type specified ,*f*, was a type not recognized. DUMP used the O or octal format. Check the DUMP control statement

for the specified format type and compare with the CRAY-OS Version 1 Reference Manual, publication SR-0011, for the allowed types. From DUMP. Class, informative; processing continues.

DP203 - *dn* DOES NOT EXIST
When DUMP began reading the specified input dataset, it found the dataset to be empty. A DUMPJOB statement must precede the DUMP statement or the input dataset must have been properly created and accessed. From DUMP. Class, fatal; user-requested abort.

EP005 - PROBLEM IN TERMINATION, DN = *dn*, ABORT CODE = *ac*, FUNCTION = *fu*
EXP had difficulty releasing the dataset in termination. *ac* = the corresponding abort code. *fu* = the function that caused the abort condition. Try releasing the dataset before termination to get a better idea of the problem. From EXP. Class, informative.

EP006 - INVOKING REPRIEVE CODE
Reprieve processing is about to be invoked. From EXP. Class, informative.

EX001 - UNEXPECTED TOKEN: *token*
The program expected a verb such as SELECT, LINES, or JOBNAME but instead read the token printed in the message. Probably a user error. Correct the EXTRACT command and resubmit the job. From EXTRACT. Class, not reprievable.

EX002 - UNEXPECTED TOKEN: *token*
The token names a function that is not implemented. No user error can generate this message. Show the listing to a Cray Research analyst. From EXTRACT. Class, not reprievable.

EX003 - UNEXPECTED TOKEN: *token*
The operand of a TYPE verb is unrecognized. The program expected a token such as ASCII, HARDWARE, etc., but read the token printed in the message. This is probably a user error. Correct the EXTRACT command and resubmit the job. From EXTRACT. Class, not reprievable.

EX004 - UNEXPECTED TOKEN: *token*
Unrecognized SUBTYPE operand. Correct the directive and resubmit. From EXTRACT. Class, fatal; not reprievable.

EX006 - UNEXPECTED TOKEN: *token*
More than six message identifiers were specified. This is probably a user error. Correct the EXTRACT command and resubmit the job. From EXTRACT. Class, not reprievable.

EX007 - UNEXPECTED TOKEN: *token*
Incorrect format of a date or time. EXTRACT requires that a date be entered as *mm/dd/yy* and the time entered as *hh:mm:ss*. Leading zeros must be supplied. From EXTRACT. Class, not reprievable.

EX020 - UNEXPECTED TOKEN: *token*
The time-of-day parameter was omitted on a STARTIME or ENDTIME directive to EXTRACT. Correct the directive and resubmit the job. From EXTRACT. Class, fatal; not reprievable.

EX021 - UNEXPECTED TOKEN: *token*
The EXTRACT directive STARTIME = NOW is illegal. The NOW keyword is used only with ENDTIME. Correct the directive and resubmit the job. From EXTRACT. Class, fatal; not reprievable.

EX022 - UNEXPECTED TOKEN: *token*
The character string appeared where a delimiter (blank, comma, or semicolon) was expected. This is probably a user error. Correct the EXTRACT command and resubmit the job. From EXTRACT. Class, not reprievable.

EX023 - UNEXPECTED TOKEN: *token*
A SUBTYPE operand applies to more than one type and is not consistent with the operand of a previous type or subtype. Split the request into separate reports. For example, for requesting SUBTYPE = SCP SINGLE (SCP ASCII messages and single-bit memory errors), this error occurs because only one TYPE can be processed per report. Instead, submit two requests: SUBTYPE = SCP and SUBTYPE = SINGLE. From EXTRACT. Class, fatal; not reprievable.

EX024 - UNEXPECTED TOKEN: *token*
A SUBTYPE operand applies to more than one type, and no preceding TYPE or SUBTYPE operand resolves the ambiguity. Include a TYPE directive with the appropriate operand. From EXTRACT. Class, fatal; not reprievable.

EX025 - UNEXPECTED TOKEN: *token*
The SUBTYPE operand is inconsistent with a previously specified TYPE or SUBTYPE. Correct the directive and resubmit. From EXTRACT. Class, fatal; not reprievable.

EX026 - UNEXPECTED TOKEN: *token*
More than one SUBTYPE specification in a report. Correct the directive and resubmit. From EXTRACT. Class, fatal; not reprievable.

EX027 - UNEXPECTED TOKEN: *token*
The token is the operand of a duplicate TYPE specification; there can be only one TYPE specification in a report. This message also occurs if a SUBTYPE directive precedes the TYPE. Correct the directive and resubmit. From EXTRACT. Class, fatal; not reprievable.

EX077 - UNEXPECTED TOKEN: *token*
The internal delimiter ending a list of tokens was destroyed. The error cannot be caused by user input. Show the listing to a Cray Research analyst. From EXTRACT. Class, fatal; not reprievable.

EX081 - LINE LIMIT REACHED
The current report was terminated early because the line limit was exceeded. The default line limit is 1000. A line limit can be specified by the LINES directive and must be respecified for each report. From EXTRACT. Class, informative.

EX082 - *n* RECORDS READ FROM $SYSLOG
The specified number of records were read from $SYSLOG in processing the current report. From EXTRACT. Class, informative.

EX083 - *n* RECORDS DISPLAYED ON *dn*
The specified number of records were written on the local dataset *dn*. From EXTRACT. Class, informative.

EX084 - *n* MEMORY ERRORS REPORTED
The SUMMARY verb appeared in the directive and *n* memory errors are summarized. From EXTRACT. Class, informative.

EX085 - NO MEMORY ERRORS REPORTED
The SUMMARY verb appeared in the directive, but no memory appeared in the $SYSLOG file. From EXTRACT. Class, informative.

EX086 - MEMORY REPORT SORT FAILED
EXTRACT was unable to sort the memory errors in preparing a memory summary report. Preserve the $SYSTEMLOG file and check with a Cray Research analyst. From EXTRACT. Class, fatal; not reprievable.

EX088 - ERROR READING $SYSLOG
An error occurred reading $SYSLOG. The current report is terminated at the point of error. Messages EX082 and EX083 are generated. The program proceeds to the next report request, if there is one. See a Cray Research analyst. From EXTRACT. Class, caution.

EX089 - REPORT SUPPRESSED DUE TO ERRORS
There were errors in the report request. No report was generated. Examine previous error messages appearing in the log and possibly in the listing dataset. From EXTRACT. Class, informative.

EX603 - *nn* RECORDS WRITTEN ON *dataset*
nn is the actual number of records that comprise the EXTRACT report and are written on the output dataset. From EXTRACT. Class, informative.

EX604 - ERROR READING SYSLOG
An unidentified error occurred when a record was being read from the input dataset. If further attempts to run EXTRACT jobs result in this same message, consult the Cray site analyst. From EXTRACT. Class, user-requested abort.

FD001 - $DUMP NOT CREATED OR ACCESSED
$DUMP is not local to the job. Access a dump file to be analyzed, of insert a DUMPJOB control statement. From FLODUMP. Class, fatal.

FD002 - $DUMP DOES NOT CONTAIN FLOW TABLES
$DUMP does not contain tables set up by FLOWTRACE. Check that ON=F is included on CFT commands. From FLODUMP. Class, fatal.

FD003 - UNABLE TO OPEN $DUMP
$DUMP was local but could not be opened. See a Cray Research site analyst. From FLODUMP. Class, fatal.

FT000 - BAD CALL TO LIBRARY ERROR PROCESSOR
The $FTLIB error message processor was entered with an illegal error message code. See a Cray Research analyst. FROM $FTLIB. Class, 2; user abort; reprievable.

FT001 - CONCATENATION CALL OUT OF ORDER
The three concatenation calls normally issued by the FORTRAN compiler are out of order. This can happen only with a program written in CAL, a program that has altered itself, or a hardware problem. If the program is not at fault, the hardware should be inspected. From $FTLIB. Class, 2; user abort; reprievable.

FT002 - CONCATENATION RECURSION LIMIT EXCEEDED
The built-in stack for handling concatenation was overloaded. Correct the program. From $FTLIB. Class, 2; user abort; reprievable.

FT003 - CHARACTER ARGUMENT HAS INVALID LENGTH
A library was called with a null string or a string exceeding 504 characters. Correct the string. From $FTLIB. Class, 2; user abort; reprievable.

GP001 - BAD DELIMITER FOLLOWING KEYWORD
In processing a control statement, $GPARAM located a keyword and could not find a delimiter indicating the end of the operand. Correct the control statement and resubmit the request. From $SYSLIB. Class, 2; user abort; reprievable.

GP002 - KEYWORD *keyword* UNKNOWN OR DUPLICATED
A keyword in the control statement could not be identified or was identical to one that had already been processed. Correct the control statement and reissue the request. From $SYSLIB. Class, 2; user abort; reprievable.

GP003 - KEYWORD *keyword* MUST BE SPECIFIED
A required keyword is missing from the control statement. Correct the control statement and reissue the request. From $SYSLIB. Class, 2; user abort; reprievable.

GP004 - KEYWORD *keyword* MUST BE EQUATED
An equated value was expected for the keyword. Correct the control statement and reissue the request. From $SYSLIB. Class, 2; user abort; reprievable.

GP005 - KEYWORD *keyword* CANNOT BE EQUATED
An equated keyword was detected but the specified keyword cannot be equated. Correct the control statement and reissue the request. From $SYSLIB. Class, 2; user abort; reprievable.

GP006 - KEYWORD *keyword* VALUE OVERFLOWED WITH *data*
The value of the specified keyword overflowed the size of the parameter array. Correct the data and reissue the request. From $SYSLIB. Class, 2; user abort; reprievable.

GP007 - CONTINUATION CHARACTER NOT FOUND
A continuation character was detected but a subsequent continuation statement was not found. Correct the control statement and resubmit the request. From $SYSLIB. Class, 2; user abort; reprievable.

GP008 - SYNTAX ERROR ENCOUNTERED
GETPARAM has detected a violation of the control statement syntax. Ensure that the separator combinations are proper; for example an equal sign (=) does not follow a colon (:). From $SYSLIB. Class, 2; user abort; reprievable.

GP009 - POSITIONAL PARAMETER *n* OVERFLOWED ON '*data*'
The *n*th positional parameter has been given too large a value. Correct the positional value. From $SYSLIB. Class, 2; user abort; reprievable.

GP010 - INTERNAL ERROR WITH $CCS DETECTED
GETPARAM has detected an error with the cracked table. See a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

GP011 - INTERNAL ERROR WITH GETPARAM DETECTED
GETPARAM has detected an error that has occurred during its processing. See a Cray Research analyst. From $SYSLIB. Class, reprievable.

GP012 - INVALID CONTROL TABLE STRUCTURE
The call to GETPARAM specifies a parameter Control Table that has the wrong format. Ensure that the Control Table follows the rules set forth by GETPARAM. See the Library Reference Manual, CRI publication SR-0014. From $SYSLIB. Class, 2; user abort; reprievable.

GP013 - POSSIBLE KEYWORD *keyword* TREATED AS A POSITIONAL PARAMETER
The parameter *keyword* has been placed in a positional slot and *keyword* is a valid keyword. If *keyword* is intended as a positional value, ignore the message. If *keyword* is the keyword-only selection of *keyword*, use commas to skip over the positional parameters. From $SYSLIB. Class, informative.

GP014 - POSITIONAL PARAMETER *nnn* MUST BE SPECIFIED
The *n*th positional parameter has been omitted when its appearance is required. Specify the parameter on the statement. From $SYSLIB. Class, reprievable.

GP015 - ERROR IN SECURE CONTROL STATEMENT. VERB ONLY POSTED.
GPARM detected a violation of the control statement syntax while decoding a statement containing secure parameters. Correct the control statement and resubmit the request. From GPARM ($SYSLIB). Class, 2; user abort; reprievable.

IO000 - BAD CALL TO LIBRARY ERROR PROCESSOR
The $IOLIB error message processor was entered with an illegal error message code. See a Cray Research analyst. From $IOLIB. Class, 2; user abort; reprievable.

IO001 - BACKSPACE ON NULL DATASET
Attempt was made to backspace on a dataset that does not exist. Adjust program logic to create the dataset before backspacing, perhaps by rewinding. From $IOLIB. Class, 2; user abort; reprievable.

IO002 - REPETITION COUNT < 1
The size of an array named without subscripts in an I/O statement is 0 or negative. Adjustable dimensional arrays must be correctly dimensioned. From $IOLIB. Class, 2; user abort; reprievable.

IO003 - CALL OUT OF SEQUENCE
A routine was called out of sequence. The $IOLIB I/O routines are called in the following sequence. (1) Initialize; (2) Process list items; (3) Terminate. Check for the following errors:

- Compiler error
- Error in CAL routine coding
- Illegal arithmetic on variables assigned a label
- Attempting to output or encode a function referred to in an output or encode list
- Attempting to input or decode a function referred to in an input or decode list

From $IOLIB. Class, 2; user abort; reprievable.

IO004 - READ PAST EOD
Attempted to read a file after the end of data was read. Test for EOF and EOI WITH END= or UNIT or EOF functions. From $IOLIB. Class, 2; user abort; reprievable.

IO005 - DECODE RECORD LENGTH < 1 OR > 152
The record length in a DECODE statement is less than 1 or greater than 152. Correct the record length. From $IOLIB. Class, 2; user abort; reprievable.

IO006 - NUMERIC : ILLEGAL CHARACTER
An illegal character appears in an input field being converted by a numeric format (for example, 9 in an octal number or Q in a decimal number). Determine whether the correct data line is being processed by the correct format statement. Count the columns, and check for errors in data fields. From $IOLIB. Class, 2; user abort; reprievable.

IO007 - NUMERIC : OVERFLOW
A number converted by a format on input exceeds the maximum for that type of number. Check the format and the data record for a match. Count the columns. Embedded and trailing blanks are interpreted as 0. From $IOLIB. Class, 2; user abort; reprievable.

IO008 - EXPONENT OVERFLOW
A floating-point number being converted by a format on input has an exponent larger than 2466. Check the format and the data line. Count the columns. Trailing and embedded blanks are interpreted as 0. From $IOLIB. Class, 2; user abort; reprievable.

IO009 - EXPONENT UNDERFLOW
A floating-point number being converted by a format on input has an exponent less than -2466. Check format and data line. From $IOLIB. Class, 2; user abort; reprievable.

IO010 - NULL FIELD
No characters appear in a field being converted on input by a format. Check the format and data lines. From $IOLIB. Class, 2; user abort; reprievable.

IO011 - ILLEGAL LOGICAL VALUE
Input format conversion fields for logical values must begin with 'T', '.T', 'F', or '.F'. Check the format and data lines. Count the columns. Check for errors. From $IOLIB. Class, 2; user abort; reprievable.

IO012 - POSITION < 1 or > 152
An attempt was made to position before column 1 or after column 152 during formatted I/O. Check the format for fields that are too large. Also, any repeated parenthesis groups without an embedded slash should be examined for the correct repeat count. The tab or X count might be too large. From $IOLIB. Class, 2; user abort; reprievable.

IO013 - FORMAT MUST BEGIN WITH (
Formats must begin with a left parenthesis. Check variable formats. (Explicit formats are checked at compile time.) From $IOLIB. Class, 2; user abort; reprievable.

IO014 - FIELD SPECIFICATION MISSING
No width was specified after a format character (for example, I rather than I5). Check variable formats. (Explicit formats are checked at compile time.) From $IOLIB. Class, 2; user abort; reprievable.

IO015 - ILLEGAL SEQUENCE OF CHARACTERS
A field is missing in a format; for example, -7 must be followed by X or P). Check variable formats. (Explicit formats are checked at compile time.) From $IOLIB. Class, 2; user abort; reprievable.

IO016 - UNMATCHED OR TOO MANY PARENTHESES
Parentheses were nested more than 9 deep in a format or parentheses are out of order. Check variable formats. (Explicit formats are checked at compile time.) From $IOLIB. Class, 2; user abort; reprievable.

IO017 - ILLEGAL FORMAT CHARACTER
An unrecognized character appeared in a format. Check variable formats. (Explicit formats are checked at compile time.) From $IOLIB. Class, 2; user abort; reprievable.

IO018 - ILLEGAL FIELD WIDTH
Improper field width; for example, the d width is greater than the w width as in F10.20. Check variable formats. (Explicit formats are checked at compile time.) From $IOLIB. Class, 2; user abort; reprievable.

IO019 - VALUE AND SPECIFICATION TYPE DIFFER
A variable is being read or written under format control and the format conversion specification is inappropriate for the variable type; for example, an integer being printed with an F20.10 specification. Check the format and I/O list. Make sure that variables are properly spelled. If necessary, a typeless function can be used. For example, PRINT 10, OR(X,0) instead of PRINT 10, X. From $IOLIB. Class, 2; user abort; reprievable.

IO020 - RECORD LENGTH EXCEEDED
One of the following conditions was detected:

- An attempt was made to read more columns than were supplied on an input record. This is illegal for an unformatted read.
- An attempt was made to ENCODE or DECODE more columns than the specified length.
- An unformatted read called for more values than the record contained.

Check the format. Check the encode/decode record length. Check for proper array dimensions. Make sure the expected record is being read. From $IOLIB. Class, 2; user abort; reprievable.

IO022 - REQUESTED RECORD LENGTH < 1 OR > 152
One of the following conditions was detected:

- The record length in an ENCODE statement is less than 1 or greater than 152.
- On a dataset connected for direct-access formatted I/O, the requested or existing record length is greater than the maximum allowed (152).

Correct the record length. From $IOLIB. Class, 2; user abort; reprievable.

IO023 - READ OR WRITE AFTER UNCLEARED ERROR
An error occurred on a buffer in/out operation, and a subsequent operation was attempted on the same file without an intervening UNIT function. Use the UNIT function to detect I/O errors. From $IOLIB. Class, 2; user abort; reprievable.

IO024 - BUFFER WORD COUNT < 1
Input buffer word count must be 1 or more. Output buffer word count can be 0 to write a null record or to terminate a series of partial buffers. Check subscripts in arrays in CFT buffer in/out statements or CAL BUFIN/BUFOUT. Determine whether both list items are in either the same array or the same common block and if the second follows the first in storage order. From $IOLIB. Class, 2; user abort; reprievable.

IO025 - RECORD '*name*' ON '*unit*' SKIPPED
During a search for a NAMELIST input record, a record with name *name* was encountered and skipped. Determine whether NAMELIST input record and read statements are in the correct order. Disable the message by calling RNLSKIP(0), as described in the FORTRAN (CFT) Reference Manual, CRI publication SR-0009. From $IOLIB. Class, informative.

IO026 - ILLEGAL TYPE CONVERSION
A type conversion that is not defined (for example, . LOGICAL=1) was encountered while NAMELIST input was being processed. Check item types and spelling. Check for error in data. Determine whether RNLTYPE was called to disable all type conversion, as described in the FORTRAN (CFT) Reference Manual, CRI publication SR-0009. From $IOLIB. Class, 2; user abort; reprievable.

IO027 - UNRECOGNISED DATA TYPE
A system error was detected in NAMELIST input conversion. See a Cray Research analyst. From $IOLIB. Class, 2; user abort; reprievable.

IO028 - ERROR IN CONSTANT FIELD
An illegal or inappropriate character appeared in a constant being processed as NAMELIST input. Examine the data line for erroneous character. From $IOLIB. Class, 2; user abort; reprievable.

IO029 - BAD SUBSCRIPT
An illegal subscript appeared in a NAMELIST input record. Examine the data line for spelling errors. From $IOLIB. Class, 2; user abort; reprievable.

IO030 - TOO MANY SUBSCRIPTS
More subscripts were specified on a NAMELIST data line for an array than were declared in a DIMENSION statement. Examine the data line or the DIMENSION statement for spelling errors. From $IOLIB. Class, 2; user abort; reprievable.

IO031 - NOT FOLLOWED BY REPLACEMENT CHARACTER
The NAMELIST input item was not followed by a legal replacement separator. Examine the data line. Call RNLREP to add a legal separator; check calls to RNL routines for consistency. From $IOLIB. Class, 2; user abort; reprievable.

IO032 - NAME NOT IN GROUP LIST
A NAMELIST input line contains a variable name that is not in the group NAMELIST for this record. Examine the data line for spelling errors. Check the NAMELIST definition. From $IOLIB. Class, 2; user abort; reprievable.

IO033 - CONSTANT NOT PROPERLY TERMINATED
A NAMELIST input record contains an improper or missing terminator. Examine the data line. Call RNLSEP to add separation character. Check other calls to RNL routines for consistency. From $IOLIB. Class, 2; user abort; reprievable.

IO034 - ILLEGAL CHARACTER IN CONSTANT
A NAMELIST input record contains an illegal character (for example, 9B). Examine the data line for the erroneous character. From $IOLIB. Class, 2; user abort; reprievable.

IO035 - NAME TOO LONG
A NAMELIST input line contains a variable name with more than eight characters. Examine the data line for spelling errors. Examine calls to RNLREP for consistency. An alphanumeric character as a replacement separator is illegal without a space code separating the character from the parameter. From $IOLIB. Class, 2; user abort; reprievable.

IO036 - NAME MUST BEGIN WITH ALPHABETIC CHARACTER
The NAMELIST data record contains a variable name that does not begin with an alphabetic character. Examine the data line. Look for a missing comment separator or spelling error. Add a call to RNLCOMM. Check calls to other NAMELIST input routines for consistency. From $IOLIB. Class, 2; user abort; reprievable.

IO037 - MISSING OR ILLEGAL CONSTANT
A NAMELIST input record has a missing or unrecognizable constant. Examine the data line for spelling errors. Check for inconsistent calls to NAMELIST input routines. From $IOLIB. Class, 2; user abort; reprievable.

IO038 - CONSTANT LIST EXCEEDS VARIABLE LIST
More constants were specified for a NAMELIST input item than could be used to fill the item. Examine the data line. Look for extra delimiters. Check array dimensions and spelling. From $IOLIB. Class, 2; user abort; reprievable.

IO039 - NAMELIST RECORD OUT OF ORDER
A NAMELIST record with a name different from the one being searched for was encountered. RNLSKIP had been previously called to make this a fatal error. Check the order of reads and records. Call RNLSKIP to ignore this condition. From $IOLIB. Class, 2; user abort; reprievable.

IO040 - READ PAST EOF OR EOD
An end of file or end of data was encountered while a search was being made for a NAMELIST record. No END= branch was supplied. Check the spelling of data record names. Check the order of reads and records. Use END= specifier. From $IOLIB. Class, 2; user abort; reprievable.

IO041 - ILLEGAL CHARACTER ARGUMENT
The character arguments to the NAMELIST input routines must be in the range 1 to 127 inclusive. Check the call. From $IOLIB. Class, 2; user abort; reprievable.

IO042 - LENGTH OUT OF RANGE
The length parameter for WNLLONG must be either -1 or in the range 9 to 160 inclusive. Check the call. From $IOLIB. Class, 2; user abort; reprievable.

IO043 - LINE LENGTH TOO SHORT FOR NAMELIST OUTPUT
The length defined by a call to WNLLONG is too short to write the particular list item. Use a longer line, or use smaller names and values. From $IOLIB. Class, 2; user abort; reprievable.

IO044 - UNRECOGNIZED VARIABLE TYPE
The variable type defined in the NAMELIST description table is not recognized by the NAMELIST routines. Possible compiler error, or storage error by user program. Check the array subscripts. Try ON=0 (subscript bounds checking). See a Cray Research analyst. From $IOLIB. Class, 2; user abort; reprievable.

IO045 - READ AFTER WRITE OR EOD
An attempt was made to read a dataset when the previous operation was a write or the previous operation read the end of data. Check the program flow. Use END=. Check for a missing REWIND or BACKSPACE. From $IOLIB. Class, 2; user abort; reprievable.

IO047 - COPY OF *n* RECORDS COMPLETED
The copy routine processed *n* records. From $IOLIB. Class, informative.

IO048 - COPY OF *n* RECORDS *m* FILES COMPLETED
The copy routine copied *m* files that contained *n* records. From $IOLIB. Class, informative.

IO049 - *dp* ENCOUNTERED ON *dn*
Before termination, the indicated dataset position (EOD, EOF, or BOD) was encountered on dataset *dn*. From $IOLIB. Class, informative.

IO050 - *r* RECORDS *f* FILES SKIPPED ON *dn*
A number of records (*r*) and a number of files (*f*) were bypassed on dataset *dn*. This message appears at the end of every SKIP job step, whether the termination is normal or abort. From $IOLIB. Class, informative.

IO051 - DATASET POSITIONED AFTER EOD
An attempt was made to skip forward, but the datset was already positioned after the EOD. The dataset was then backspaced before the EOD. Reposition the dataset before executing a SKIP statement. From $IOLIB. Class, caution.

IO052 - ATTEMPT TO SKIP FORWARD AFTER WRITE
The DSP flag indicates that the previous operation on this dataset was a write, and a forward SKIP now is illegal. Make sure the previous operation on the dataset was not a write. From $IOLIB. Class, caution.

IO053 - DATASET POSITIONED AT EOF
An attempt was made to skip backward on a SKIPR call but the dataset is already positioned after an EOF. Reposition the dataset before executing a SKIPR statement. From $UTLIB. Class, caution.

IO054 - ATTEMPT TO BACKUP FROM BOD
The dataset is already positioned at the beginning of data and the skip direction is reverse. Make sure the dataset is not positioned at BOD if the skip direction is reverse. From $IOLIB. Class, caution.

IO055 - COPY OF *s* SECTORS COMPLETED
The unblocked dataset copy routine processed *s* sectors. From $IOLIB. Class, informative.

IO056 - REQUEST INVALID FOR BLOCKED DATASET
An I/O request that is not supported for blocked datasets was issued. The characteristics of the dataset may be changed through the ASSIGN statement. Correct the program and resubmit the job. Class 2; user abort; reprievable.

IO057 - INVALID LEN PARAMETER TO GETPOS OR SETPOS REQUEST
The value of length in the position array is too small to process the requested type of GETPOS or SETPOS. Specify a larger length value. From $IOLIB. Class, 2; user abort; reprievable.

IO058 - TAPE POSITIONING ERROR ENCOUNTERED
An attempt was made to position a tape dataset but could not be completed due to one of the following errors: position request not fully satisfied; error code in dataset parameter table is set; or internal library execution error. From $IOLIB. Class, 2; user abort; reprievable.

IO059 - FOREIGN DATASET INTERNAL TABLE OVERFLOW
The foreign dataset translation routines require internal tables to hold status and position information across I/O requests. These tables have fixed sizes. With simple changes, a version of $IOLIB can be built which increases the size of these tables. Your site analyst should be contacted to build this new version of the library. From $IOLIB. Class, 2; user abort; reprievable.

IO061 - INVALID BUFFER ADDRESS FOR UNBLOCKED DATASET
The starting or ending address of the buffer for an unblocked dataset operation is out of range. Check the subscripts. From $IOLIB. Class, 2; user abort; reprievable.

IO062 - INVALID WORD COUNT FOR UNBLOCKED DATASET
The array size on an unblocked I/O operation must be a multiple of 512 words. Correct the dimension or subscripts. From $IOLIB. Class, 2; user abort; reprievable.

IO066 - DIRECT I/O PROHIBITED UNDER SEQUENTIAL ACCESS
An attempt was made to read or write using the REC= control list option on a dataset that was accessed as sequential. Remove REC= from the control list of the I/O statement or close the dataset and open it with direct access. From $IOLIB. Class, 2; user abort; reprievable.

IO067 - SEQUENTIAL I/O PROHIBITED UNDER DIRECT ACCESS
An attempt was made to perform I/O without the REC= option while the unit was connected for direct access. Use REC= option in read/write control list or close dataset under direct access and open it under sequential access. From $IOLIB. Class, 2; user abort; reprievable.

IO068 - RECORD NUMBER MUST BE > ZERO FOR DIRECT ACCESS
The REC= control list option on a read or write statement supplied a value less than or equal to 0. The parameter specifies a record number, which must be a positive integer. Refer to the FORTRAN (CFT) Reference Manual, CRI publication SR-0009, for the correct use of REC. Change the REC= specifier to a positive integer. From $IOLIB. Class, 2; user abort; reprievable.

IO069 - RECORD LENGTH MUST BE > ZERO FOR DIRECT ACCESS
A FORTRAN OPEN statement specifies a fixed record length for a direct access connection. This record length must be a

positive integer value. Refer to the FORTRAN (CFT) Reference Manual, CRI publication SR-0009, for the correct use of REC. Change the RECL= control list specifier to a positive integer. From $IOLIB. Class, 2; user abort; reprievable.

IO070 - RECORD ADDRESS TOO LARGE FOR DIRECT ACCESS
The address of the record number specified on the FORTRAN direct access read/write statement exceeds 33 bits. Verify the record number specified in the I/O statement. If the record number is valid, check the OPEN statement that connected the dataset to a unit. Verify the record length provided in the OPEN control list. If this is also correct, the maximum size under direct access was exceeded. From $IOLIB. Class, 2; user abort; reprievable.

IO071 - UNFORMATTED RECORD LENGTH MUST BE A WORD BOUNDARY
The OPEN statement specified direct access and unformatted data. Therefore, the record length must be a multiple of eight characters. (Unformatted I/O is done on a word basis with eight characters per word.) Change the record length defined within the FORTRAN OPEN statement to a positive multiple of eight or change the I/O statement from unformatted to formatted I/O. From $IOLIB. Class, 2; user abort; reprievable.

IO072 - DIRECT READ ATTEMPTED BEYOND LAST RECORD
An attempt was made to read an unwritten record. Verify the record number specified in the READ statement. If the record number is valid, the requested record must be written before it can be read. From $IOLIB. Class, 2; user abort.

IO073 - BLANK SPECIFIER ILLEGAL IF FORM = UNFORMATTED
The blank specifier in an OPEN statment is permitted on formatted I/O only. Remove the blank specifier or change the form specifier to formatted. From $IOLIB. Class, 1; reprievable.

IO074 - STATUS MUST BE DELETE FOR SCRATCH DATASET
The requested status in a CLOSE statement is not specified DELETE for a scratch dataset. Change the status specifier to DELETE. From $IOLIB. Class, 1; reprievable.

IO075 - ILLEGAL UNIT NUMBER
The unit number specified was not an integer value in the range [0,102]. Change the value of the unit number. From $IOLIB. Class, 2; user abort.

IO076 - OLD STATUS ILLEGAL WITHOUT FILE SPECIFIER
The status was specified OLD but no file name was specified on the OPEN statement. Provide a file specifier. From $IOLIB. Class, 2; user abort; reprievable.

IO077 - NEW STATUS ILLEGAL WITHOUT FILE SPECIFIER
The status was specified NEW but no file name was specified on the OPEN statement. Provide a file specifier or change the status specifier. From $IOLIB. Class, 1; reprievable.

IO078 - FILE ALREADY CONNECTED TO ANOTHER UNIT
Execution of an OPEN statement was attempted on a file that was already connected to another unit. Close the file. Open the file, using the desired unit specifier. From $IOLIB. Class, 1; reprievable.

IO079 - SCRATCH STATUS ILLEGAL WITH FILE SPECIFIER
The status was specified SCRATCH with a named file. Change the status specifier or omit the file specifier. From $IOLIB. Class, 1; reprievable.

IO080 - RECL SPECIFIER ILLEGAL IF SEQUENTIAL ACCESS
RECL was specified illegally when a dataset was being connected for sequential access. It can be specified only for direct access. Change the access method to direct or omit the RECL specifier. From $IOLIB. Class, 1; reprievable.

IO081 - ACCESS SPECIFIER NOT RECOGNIZED
The access method specified in an OPEN statement is not SEQUENTIAL or DIRECT. Change the access specifier to SEQUENTIAL or DIRECT. From $IOLIB. Class, 1; reprievable.

IO082 - FORM SPECIFIER NOT RECOGNIZED
The form specifier in an OPEN statement is neither FORMATTED nor UNFORMATTED. Change the form specifier to a legal value. From $IOLIB. Class, 1; reprievable.

IO083 - BLANK SPECIFIER NOT RECOGNIZED
The blank specifier in an OPEN statement is neither NULL nor ZERO. Change the blank specifier to a legal value. From $IOLIB. Class, 1; reprievable.

IO084 - STATUS SPECIFIER NOT RECOGNIZED
In an OPEN statement, the specified status is not OLD, NEW, SCRATCH, or UNKNOWN. In a CLOSE statement the status is not KEEP or DELETE. Change the status specifier to a legal value. From $IOLIB. Class, 1; reprievable.

IO085 - REQUESTED ACCESS CONFLICTS WITH EXISTING ACCESS
The current access method conflicts with the method requested in the OPEN statement. Dataset is already connected to a unit. Close the dataset and reconnect the unit and dataset, using the requested access specifier; or change the access specifier to match the access method of the existing connection. From $IOLIB. Class, 1; reprievable.

IO086 - REQUESTED STATUS CONFLICTS WITH EXISTING STATUS
Dataset is already connected to the unit. The current status conflicts with the status requested in the OPEN statement. Close the dataset and reconnect the unit and dataset, using the requested status specifier; or change the status specifier to match the status of the existing connection. From $IOLIB. Class, 1; reprievable.

IO087 - REQUESTED FORM CONFLICTS WITH EXISTING FORM
The current form conflicts with the form requested in the OPEN statement. Dataset is already connected to the unit. Close the dataset and reconnect the unit and dataset, using the requested form specifier; or change the form specifier to match the form of the existing connection. From $IOLIB. Class, 1; reprievable.

IO088 - REQUESTED RECL CONFLICTS WITH EXISTING RECL
The RECL requested differs from the RECL established with this connection. Unit and dataset are already connected for direct access. Change the RECL specifier to match that of the existing connection. From $IOLIB. Class, 1; reprievable.

IO089 - MISSING UNIT SPECIFIER
An OPEN or a CLOSE statement is missing a unit specifier; or an INQUIRE statement has neither a unit nor file specifier. Supply a unit specifier, or a file specifier for INQUIRE. From $IOLIB. Class, 1; reprievable.

IO090 - NEW STATUS ILLEGAL IF DATASET ALREADY EXISTS
The status in an OPEN statement is specified NEW for a dataset that already exists. Change the status specifier to OLD or UNKNOWN or omit the status specifier. From $IOLIB. Class, 1; reprievable.

IO091 - OLD STATUS ILLEGAL IF DATASET DOES NOT EXIST
Status in an OPEN statement is specified OLD for a dataset that does not exist. Change status specifier to NEW or UNKNOWN or omit the status specifier. From $IOLIB. Class, 1; reprievable.

IO094 - ILLEGAL FILE NAME
The library routine encountered an illegal file name specifier. Correct the file name. From $IOLIB. Class, 1; reprievable.

IO095 - RECL MUST BE SPECIFIED IF AND ONLY IF ACCESS IS DIRECT
A record length was requested in an OPEN statement for a dataset that was not opened to direct access. Remove the record length request from the OPEN statment, or request that the dataset be opened as direct access. From $IOLIB. Class, 1; reprievable.

IO096 - BAD CHARACTER ARGUMENT ADDRESS
A character argument passed to a library routine has an invalid address. From $IOLIB. Class, 1; reprievable.

IO097 - UNKNOWN CONVERSION TYPE
The foreign dataset translation type request is not supported. Make sure the RF parameter on the ACCESS control card is valid. From $IOLIB. Class, 1; reprievable.

IO098 - LIST DIRECTED I/O ILLEGAL UNDER DIRECT ACCESS
An attempt was made to perform list-directed I/O on a dataset that was accessed as direct. From $IOLIB. Class, 2; user abort; reprievable.

IO099 - UNRECOVERED DATA ERROR
A data discrepancy was detected during an I/O operation. A Cray Research analyst or system operator should dump the system log to determine if other datasets on the same device are receiving similar error messages. If so, there might be a failing device. If the problem persists for the dataset, the dataset must be recreated. From $IOLIB. Class, 2; user abort; reprievable.

IO101 - ACTUAL BLOCK SIZE > ACCESS MBS PARAMETER
The block length in the block descriptor word of a foreign dataset being read exceeds the maximum block size specified by the MBS parameter of the ACCESS statement. Specify a larger maximum block size. From $IOLIB. Class, 1; reprievable.

IO102 - ACTUAL RECORD SIZE > ACCESS RS PARAMETER
The record or segment length in a segment descriptor word of a foreign dataset being read exceeds the maximum record size specified by the RS parameter of the ACCESS statement. Specify a larger maximum record size. From $IOLIB. Class, 1; reprievable.

IO103 - INVALID SEGMENT OR RECORD LENGTH; > REMAINDER OF BLOCK
The segment length in a foreign segment, in variable-record format, is longer than the rest of the block. Therefore the wrong segment length was probably written when the file was created. From $IOLIB. Class, 1; reprievable.

IO104 - INVALID IBM SEGMENT CONTROL CODE
The segment control code in a foreign segment descriptor word cannot follow the previous segment control code; either the information in the segment descriptor word is invalid or there is a problem in the library translation routines. From $IOLIB. Class, 1; reprievable.

IO105 - BAD FORMAT ADDRESS PASSED TO IOLIB ROUTINE
The address of the FORMAT statement was rejected by an $IOLIB-formatted I/O routine. The address lies outside the user's code area. Check the format. From $IOLIB. Class, 2; user abort; reprievable.

IO106 - RECORD LENGTH EXCEEDED ON WRITE
The amount of data in a WRITE statement exceeded the maximum record size for the file. Specify a larger maximum record size for the file. From $IOLIB. Class, 1; reprievable.

IO107 - BUFFER IS TOO SMALL FOR VARIABLE RECORD FORMAT
The buffer space allocated to write a foreign file with variable records must be at least the following number of sectors: 2*MBS/8+D'1024. Allocate more buffer space on the ASSIGN statement. From $IOLIB. Class, 1; reprievable.

IO108 - CHARACTER TYPE ILLEGAL
The use of character-type constants, variables, arrays, array elements, or functions is not allowed in a subroutine that was invoked. Remove the illegal type characters. From $IOLIB. Class, 2; user abort; reprievable.

IO109 - BUFFER I/O ILLEGAL UNDER RANDOM ACCESS
An attempt was made to perform buffer I/O while the dataset was connected for random access. Do not attempt buffer I/O on a dataset connected for random access. From $IOLIB. Class, 2; user abort; reprievable.

IO998 - DUPLICATE CALL TO **ERP
IO999 - $OUT DSP POSSIBLY DESTROYED
These messages are issued as a pair. The $IOLIB error processor is unable to write a message on file $OUT. The $OUT DSP might have been destroyed; a catastrophic error might have occurred in the I/O library; or the user might have made a library call error after being reprieved from a library error. If $OUT file is not printable, examine the program for subscripts out of range. Try the ON=0 (bounds checking), CFT option or the IOAREA,LOCK control card to protect the DSPs. Note that IOAREA,LOCK does not protect the DSP's in a job step using the stack environment. From $IOLIB. Class, 2; user abort; reprievable.

IT001 - ITEMIZE VERS. *x.xx*
Initiation of ITEMIZE utility produces this message to identify version. From ITEMIZE. Class, informative.

IT002 - LISTING DEVICE CANNOT BE ZERO
The ITEMIZE control statement contained L=0. Correct the statement and resubmit the job. From ITEMIZE. Class, fatal; not reprievable.

IT003 - FILE TO BE ITEMIZED CANNOT BE ZERO
The ITEMIZE control statement specifies DN=0. Correct the statement and resubmit the job. From ITEMIZE. Class, fatal; not reprievable.

IT004 - NUMBER OF FILES MUST BE > 0
The ITEMIZE control statement specifies NF=0. Correct the statement and resubmit the job. From ITEMIZE. Class, fatal; not reprievable.

IT005 - THE FILE TO BE ITEMIZED CANNOT BE THE LISTING FILE
The ITEMIZE control statement (explicitly or by default) specifies the same dataset for the L and DN parameters. Correct the control statement and resubmit the job. From ITEMIZE. Class, fatal; not reprievable.

IT006 - LOCAL DATASET NAME *ldn* DOES NOT EXIST
The local dataset name to be itemized (which is $OBL by default or the operand of the DN parameter) does not exist. Correct the control statement or insert an ACCESS control statement and resubmit the job. From ITEMIZE. Class, fatal; not reprievable.

IT007 - REQUESTED PAGE SIZE TOO SMALL - LPP=8 USED
The lines per page requested by the LPP parameter of the OPTION control statement are less than the minimum required by ITEMIZE. Specify the correct number of lines per page. From ITEMIZE. Class, informative.

IT008 - EXPANSION PARAMETER *exparam* IGNORED
The user specified more than one of the keyword parameters E, B, and X. The parameter *exparam* is ignored. From ITEMIZE. Class, informative.

IT009 - THE 'T' REQUEST PROHIBITS 'E', 'B', or 'X' REQUESTS
The user specified the T parameter and E, B, or X on the ITEMIZE control statement. The T request forces ITEMIZE to ignore any E, B, or X request. From ITEMIZE. Class, informative.

IT010 - UNBLOCKED FILES MAY NOT BE ITEMIZED
The dataset to be itemized is unblocked. From ITEMIZE. Class, fatal.

JS001 - JOB *changing*. SIZE = *n*
A job's memory is changing as specified in the message (TERMINATING, CONTRACTING, EXPANDING, INITIATING). System log only. From JSH. Class, informative.

JS003 - JOB WAS ROLLED OUT TO WAIT FOR MEMORY
The job needs more memory than is currently available; therefore, the job is rolled out. System log only. From JSH. Class, informative.

JS004 - ROLLFILE WRITE FAILURE, RETRYING.
DQM returned an error status from a rollfile write attempt. If retries fail, consider killing the job. System log only. From JSH. Class, informative.

JS005 - ROLLFILE READ FAILURE, JOB SUSPENDED
The roll image is unreadable. The job is suspended for recovery. The job image is unrecoverable. The job should be resubmitted from the front end. System log only. From JSH. Class, informative.

JS006 - JOB RECOVERED
The job has been successfully resumed from a rolled image following a system interruption. From JSH. Class, informative.

JS007 - ROLLFILE ALLOCATION FAILURE, RETRYING
There is no room for a roll image of the job on the disk. From JSH. Class, informative.

KP004 - UNIMPLEMENTED ACCESS MODE *iii*
The user has attempted to open a dataset in other than READ, READWRITE, WRITE, or APPEND mode. The attempt to open the file results in an ERR return; the consequence of this is determined by the calling program. Correct the program and resubmit. From RATLIB, function XOPENX. Class, warning.

KP005 - BAD RESPONSE FROM QSTFFN *iii*
Something has happened to QSTFFN, a routine in $RATLIB. The attempt to open the file results in an ERR return; the consequence of this is determined by the calling program. From RATLIB, function XOPENX. Class, warning.

KP006 - NAME FORMAT ERROR
The user has attempted to open a dataset with a name not conforming to Software Tools standards. (An example of a violation would be a name with more than three slash ('/') characters.) The attempt to open the file results in an ERR return; the consequence of this is determined by the calling program. Correct the program and resubmit. From RATLIB, function XOPENX. Class, warning.

KP007 - ATTEMPT TO READ UNOPENED DEVICE *iii*
The user has attempted to read from file descriptor *iii* which has not been opened. Correct the program and resubmit. From RATLIB, function GETCH or GETLIN. Class, fatal.

KP008 - ATTEMPT TO READ PAST END-OF-DATA
The user has attempted to read past the end of a dataset. Correct the program and resubmit. From RATLIB, function GETCH or GETLIN. Class, fatal.

KP009 - BAD READCP STATUS (*iii*) ON DEVICE *device*
Please report the occurance of this message to your Cray Research site analyst. From RATLIB, function GETCH or GETLIN.

KP010 - BAD VALUE (*iii*) IN LASTC FOR *aaa*
Verify that the user program has not bashed a variable in the /io/ common block. If not, please report the occurence of this message to your Cray Research site analyst. From RATLIB, function GETCH or GETLIN.

KP011 - ATTEMPT TO WRITE UNOPENED DEVICE *iii*
The user has attempted to write on file descriptor *iii* which has not been opened. Correct the program and resubmit. From RATLIB, function PUTCH or PUTLIN. Class, fatal.

KP017 - SHELL SYNTAX ERROR (*error*)
A shell command line is incorrect, resulting in an ERR return. Correct the command and resubmit. From RATLIB, function STXERR. Class, fatal.

KP018 - DYNAMIC STORAGE ERROR CODE IS *iii*
The heap manager detected an error. Other messages written on $OUT further detail this message. For more information on the heap manager, see the Library Reference Manual, CRI publication SR-0014. From RATLIB, function QSTMEM. Class, fatal.

KP019 - CANNOT EVEN OPEN ERROUT
In attempting to initialize the tool, an error occured attempting to open the

ERROUT file. You have probably redirected ERROUT to something illegal (e.g., ?////). Correct the program and resubmit. From RATLIB, function INITST. Class, fatal.

KP021 - THE VERB. FORM OF THE COMMAND LINE REQUIRED
When using Software Tools without using the shell, you cannot code the operands surrounded by parentheses. That is, if you code "YECH.hello", "hello" will be echoed; if you code "YECH(hello)", this error message will appear. From RATLIB, function MAKARG. Class, fatal.

KP022 - ARGUMENT BUFFER OVERFLOW
You have supplied too many arguments or arguments too long for the internal buffers in MAKARG. Buffers are controlled by the ARGBUFSIZE in $RATDEF and is now set at 19*MAXLINE, which is 154 characters. From RATLIB, function MAKARG. Class, fatal.

KP023 - ARGUMENT COUNT OVERFLOW
You have supplied too many arguments. Arguments are controlled by the MAXARGS in $RATDEF,which is set at 32. Reduce the number of arguments and resubmit. From RATLIB, function makarg. Class, fatal.

KP025 - BLOCK EXPANSION OVERFLOW
An internal buffer was blown in processing a command line. The command line processed may be garbage or may have been truncated. Processing continues. From RATLIB, function GIOXPAND. Class, warning.

KP026 - CANNOT OPEN ENVIRONMENT FILE
$$SHENV does not exist when a tool begins execution. This message should no longer occur; if it does, notify a Cray site analyst. From RATLIB, function INITST. Class, informative.

KP027 - ATTEMPT TO READ WRITE-ONLY FILE *iii*
The user program has attempted to read from file descriptor *iii* which was opened as APPEND or WRITE. Correct the program and resubmit. From RATLIB, function GETLIN. Class, fatal.

KP028 - ATTEMPT TO READ WRITE-ONLY FILE *iii*
The user program has attempted to read from file descriptor *iii* which was opened as APPEND or WRITE. Correct the program and resubmit. From RATLIB, function GETCH. Class, fatal.

KP029 - LIBRARY VERSION MESSAGE
(Not normally compiled in). When LIBDEBUG in $RATDEF is defined, this message is produced at the initiation of each tool to define unambiguously the library version. From RATLIB, function INITST. Class, informative.

KP030 - TOO MANY OVERSTRIKES IN *file*
The LPR tool can overstrike a maximum of NUMBEROFLINES times in a given character position; this message occurs if a file specifies more overstrikes then NUMBEROFLINES. The maximum is a LPR source code definition. *file* identifies the file in which the error occurred. '-' and ' ' both can appear, both representing the standard input file. From RATLIB. Class, fatal.

NOTE

Message classes for LD messages, which originate in the loader, are interpreted differently than for other coded messages, as follows:

Informative: error does not stop program execution, and might or might not hinder execution.

Caution: job aborts when load process completes unless NA is selected; program might not execute properly.

Warning: job aborts when load process completes unless NX is selected; program execution is not possible.

Fatal: job aborts immediately.

LD000 - BEGIN EXECUTION
From LDR. Class, informative.

LD001 - DUPLICATE PROGRAM BLOCK *name* ENCOUNTERED AND SKIPPED (*dn*).
Check the program or associated libraries for a duplicated block. From LDR. Class, informative.

LD002 - DATASET (*dn*) NULL FILE OR BAD TABLE
The input to LDR is in error. See a Cray Research analyst. From LDR. Class, fatal.

LD003 - DATASET (*dn*) INITIAL TABLE NOT PDT
The input to LDR is in error. See a Cray Research analyst. From LDR. Class, fatal.

LD004 - BLANK COMMON REDEFINED *defn* DEFINITION USED
defn can indicate either PREVIOUS or LARGER for the definition used. From LDR. Class, informative.

LD005 - SHORTER COMMON (*name*)
REFERENCED IN (*pgm*)
1ST DEFINITION USED
From LDR. Class, informative.

LD006 - DUPLICATE ENTRY LOADED AND IGNORED (*name*)
A duplicate entry was encountered. Check the program for a duplicate entry. From LDR. Class, informative.

LD007 - RELOCATABLE LOAD MODULE IN ABSOLUTE LOAD OR ABSOLUTE LOAD MODULE IN RELOCATABALE LOAD
Check the load modules. From LDR. Class, caution; job aborts when load completes.

LD009 - UNSATISFIED EXTERNAL (*name*) CALLED BY (*rtn*)
The external called is not in one of the user-specified or default libraries. From LDR. Class, informative.

LD010 - UNSATISFIED EXTERNALS FOUND ABORT OPTION
Unsatisfied externals were found, and the USA parameter was specified on the LDR control statement, causing job step abort. Correct the problem in the user libraries, source datasets, user-supplied binary dataset, etc. From LDR. Class, fatal.

LD011 - LDR INITIATED:*statement*
The CNS parameter was used in the LDR control statement. From LDR. Class, informative.

LD012 - DATASET (*name*) INVALID BLOCK NAME
An invalid block name was encountered: possible problem in CAL or CFT. Correct the block name. From LDR. Class, fatal.

LD013 - START ENTRY NOT VALID
The start entry is missing. From LDR. Class, warning.

LD014 - LONGER COMMON (*name*) REFERENCED IN (*pgm*)
A previously defined, named common block is longer in a subsequently encountered routine. Except for blank common blocks, this error prevents program execution. From LDR. Class, caution.

LD015 - NO START ADDRESS FOUND, FIRST ENTRY USED
The start address is missing. Program execution begins at the first entry point from the first PDT. From LDR. Class, informative.

LD016 - MEMBER ERROR (*name*)
A member specified by an overlay directive was not found in the dataset specified by the currently active file directive. Correct the overlay directive or change the file name. From LDR. Class, caution.

LD017 - DIRECTIVE ERROR (*erroneous directive*)
There is a syntax error in the overlay directive. From LDR. Class, caution; job aborts when load completes.

LD018 - OVERLAY MEMBER NOT FOUND *name*
A member specified by the overlay directive was not found in the file. Correct the overlay directive or change the file name. From LDR. Class, informative.

LD019 - COMPILATION ERRORS IN (*pgm*)
From LDR. Class, caution; job aborts when load completes.

LD020 - DATASET NAME TOO LONG (*name*)
From LDR. Class, caution.

LD021 - MULTIPLE LOAD DATASETS IGNORED IN OVL MODE
The user has more than one dataset in the LDR statement in OVL mode. From LDR. Class, informative.

LD022 - ILLEGAL CHARACTER IN DIRECTIVE (*dir*)
From LDR. Class, caution.

LD023 - ILLEGAL MAP VALUE, CHANGING TO MAP=PART
From LDR. Class, informative.

LD024 - TYPE = (*tbltyp*) UNRECOGNIZABLE TABLE
The table is in error; there might be a problem in CAL or CFT. See a Cray Research analyst. From LDR. Class, fatal.

LD025 - INVALID TABLE STRUCTURE (BI) FOR MODULE (*tablnum*)
The table is in error; there might be a problem in CAL or CFT. See a Cray Research analyst. From LDR. Class, fatal.

LD026 - GENERATING BUILD DIRECTORY FOR: *dataset name*
The dataset does not have the directory. No action. From LDR. Class, informative.

LD027 - DATASET (*name*) BAD LIBRARY FORMAT
The library format is in error. Check the library for the proper format. Generate a new library if the format is incorrect; otherwise see a Cray Research analyst. From LDR. Class, fatal.

LD028 - SPEC BLANK COMMON ADDR NOT LARGE ENOUGH
The starting blank common address is not large enough. Modify the SBCA directive. From LDR. Class, caution.

LD029 - DATASET (*dn*) IS NOT LOCAL
The dataset specified in the DN or LIB list could not be opened or might not exist. Check the dataset name; see a Cray Research analyst. From LDR. Class, fatal.

LD030 - DISABLED PARAMETER SELECTED AND IGNORED: (*name*)
The parameter selected was disabled. The parameter was ignored by the system. From LDR. Class, informative.

LD031 - DATA FOR COMMON BLOCK (*pgm*) IGNORED
The last word address of the block into which the data will be loaded is out of program range. See a Cray Research analyst. From LDR. Class, caution.

LD032 - BAD XI (IN XRT TABLE)=(*index*)
The index to the externals list in PDT is out of range. See a Cray Research analyst. From LDR. Class, warning.

LD033 - DATASET (*name*) REPLACED BY FILE DN
The modules loaded from FILE DN are in overlay mode. From LDR. Class, informative.

LD034 - DATASET (*name*) ALL FILES SEARCH
All files were searched in selective load. From LDR. Class, informative.

LD035 - DATASET (*name*) INVALID READ TRY AGAIN
There is no dataset name to load selectively. Check the directives. From LDR. Class, informative.

LD036 - DATASET (*dn*) NO SELECTIVE MODULES
From LDR. Class, informative.

LD037 - NAME (*name*) INCLUDED BEFORE
The excluded module was previously included. Remove the duplicate. From LDR. Class, informative.

LD038 - NAME (*module name*) EXCLUDED BEFORE
The module name was previously excluded. Remove the duplicate. From LDR. Class, informative.

LD039 - SKIP (*module name*) INCLUDED BEFORE
The module was previously included. From LDR. Class, informative.

LD040 - DATASET (*name*) INVALID SELECTIVE FILE
Invalid selective file. Check the directives. From LDR. Class, informative.

LD041 - DUPLICATE PROGRAM BLK SKIPPED ----- TRYING TO SATISFY EXTERNAL
A duplicate block name was encountered during an attempt to satisfy externals from the library directory. From LDR. Class, informative.

LD042 - ILLEGAL REFERENCE TO RELOCOVL ENTRY POINT *entry*
A module was declared an external that is defined to be an entry point in a relocatable overlay. From LDR. Class, fatal.

LD043 - ILLEGAL REFERENCE TO RELOCOVL PROGRAM BLOCK *module* BLOCK *block*
A Block Relocation Table that defines an address not within the module itself (such as labeled COMMON) refers to an address within the overlay module. Correct the program. From LDR. Class, fatal.

LD044 - ILLEGAL REFERENCE TO BLANK COMMON-MODULE *name*
A blank common was declared in a module that is defined in a relocatable overlay. Take the blank common out of the module. From LDR. Class, fatal.

LD045 - TRANSFER IS TO SID; T PARAMETER IGNORED
The T parameter is invalid when a program is loaded with the Symbolic Interactive Debugger (SID). The T parameter is ignored and the load transfer goes to SID on completion of loading activities. If SID is specified, T should be absent. From LDR. Class, informative.

LD046 - SID LOADED; CNS PARAMETER IGNORED
The CNS parameter is invalid when a program is loaded with the Symbolic Interactive Debugger (SID). The CNS parameter is ignored and SID fetches the next control statement when it begins execution. If SID is specified, CNS should be absent. From LDR. Class, informative.

LD047 - ILLEGAL GRANT OPTION SPECIFIED *option name*
The option is not a legal grant option. Change the option name. Check the CRAY-OS Version 1 Reference Manual, publication SR-0011. From LDR. Class, fatal.

LD048 - ***CALLING SEQUENCE MISMATCH***
The relocatable binaries do not have a consistent calling sequence. Recompile the program. From LDR. Class, informative.

LD049 - MODULE *name* FROM DATASET *dataset name* HAD *xxx calling* SEQ. *xxx*: OLD OR NEW
The module does not have the same calling sequence as the others. Recompile the module. From LDR. Class, fatal.

LD050 - UNKNOWN C OPTION; DEFAULT USED
The LDR C control statement parameter is equated to an unknown value. Legal values are ALL, PART, and NONE. The default C=PART is used. Fix the LDR control statement. From LDR. Class, informative.

LD051 - DATASET IS NOT LOCAL OR IN SDR
The dataset is not local to the job or in the system directory. Check the dataset in the job or see a Cray Research anaylst. From LDR. Class, caution.

LD052 - NEGATIVE VALUE AFTER RELOCATION NEAR WORD ADDRESS *addr*
If the relocation bias is added to an address and the result is a negative value, the message is produced and the LDR aborts. Check the code at the invalid address. From LDR. Class, caution.

LD053 - FIELD WIDTH TRUNCATION NEAR WORD ADDRESS *addr*
The size (in bits) of the relocatable address is larger than the specified field width. For example, if the field width is 6 bits, an address larger than 63 is truncated, with only the least significant 6 bits retained. Check the code at the invalid address. From LDR. Class, warning.

LD054 - MMLOC VALUE MUST BE EITHER BEFORE OR AFTER
The MMLOC parameter is equated to an unknown value. Legal values are BEFORE and AFTER. Correct the parameter value. From LDR. Class, fatal.

LD055 - INC VALUE OF MM MUST BE 0 IF MMLOC=BEFORE
The increment value for managed memory is not zero when the MMLOC=BEFORE. Correct the parameter value for MMLOC or INC for MM. From LDR. Class, fatal.

LD056 - MMEPS TOO SMALL, CHANGED TO DEFAULT VALUE
The value for MMEPS is not large enough. Correct the parameter value for MMEPS. See a Cray Research analyst. From LDR. Class, informative.

LD057 - STK TOO SMALL, CHANGED TO DEFAULT VALUE
The value for STK is not large enough. Correct the parameter value for STK. See a Cray Research analyst. From LDR. Class, informative.

LD058 - MM TOO SMALL, CHANGED TO DEFAULT VALUE
The value for MM is not large enough. Correct the parameter value for MM. See a Cray Research analyst. From LDR. Class, informative.

LD059 - STACK IS SPECIFIED IN *module name*
LD061 - NOT SET IN ROOT OVERLAY, MM MAY BE TOO SMALL
These two messages are output together. A module not in the root overlay needs stack space, but the managed memory area already built into the module may be too small. Increase the parameter value for MM. From LDR. Class, caution.

LD060 - BECAUSE OF OVERLAY, MMLOC CHANGED TO BEFORE
The managed memory area must precede blank common for an overlay load. LDR had built the binary as if MMLOC were set to BEFORE. Correct the parameter value for MMLOC. From LDR. Class, caution.

LD062 - MIXED COMMON BLOCK USE: *name*
The named common block has been used as both a regular common block and as a task common block. Correct the code so that only one type is used. From LDR. Class, fatal.

LD063 - MULTITASKING LIBRARIES NOT USED
Task common blocks have been used in the application, but the libraries processed by LDR do not reference the control structure, #TASKCOM. Make sure that libraries of release level 1.13 or beyond are being used. From LDR. Class, fatal.

LD064 - RELOCAT. FIELD 22ND BIT SET, CAN'T RUN EMA: *name*
When building a binary on a machine with extended memory addressing, LDR has computed a 22-bit relocatable value for an 0'10, 0'11, 0'12, or 0'13 machine instruction such that the leftmost bit is set to one. Such an instruction will give bad results if run in extended mode. Options are:

- The binary can be run if EMA is turned OFF
- If the binary is reloaded with SEGLDR, a similar message may not result because of differences in storage allocations between the two loaders
- If the named module is CFT, recompile it with EMA turned ON
- If the named module is CAL, recode so that the relocatable expression is first loaded into an A-register, and then do the load or store based on the A-register

From LDR. Class, informative.

LD065 - CAN'T RUN EMA MODE & PROG. LENGTH > 4MW
One or more instructions have been found that will give bad results if run in EMA mode, but the program length is greater than 4 megawords, and therefore cannot be run with EMA turned OFF. See message LD064 for more information and options. From LDR. Class, fatal.

LD066 - CODE/LOCAL DATA IS GREATER THAN 4MW
A code and local data block is larger than 4 megawords in length. Change the source code so that large local data arrays are placed in common blocks. From LDR. Class, warning.

LD501 - BAD DELIMITER FOLLOWING KEYWORD
Syntax error in directive From LDR. Class, fatal.

LD502 - KEYWORD (*name*) UNKNOWN OR DUPLICATED
Correct the directive. From LDR. Class, fatal.

LD503 - KEYWORD (*name*) MUST BE SPECIFIED
Correct the directive. From LDR. Class, fatal.

LD504 - KEYWORD (*name*) MUST BE EQUATED
Correct the directive. From LDR. Class, fatal.

LD505 - KEYWORD (*name*) CANNOT BE EQUATED
Correct the directive. From LDR. Class, fatal.

LD506 - KEYWORD *keyword* VALUE OVERFLOWN WITH *value*
The value for the keyword is too big. Correct the parameter value. See a Cray Research analyst. From LDR. Class, fatal.

LD507 - CONTINUATION CARD NOT FOUND
Correct the directive. From LDR. Class, fatal.

LD601 - DIRECTIVE (*name*) TERMINATOR MISSING
Correct the loader directive. From LDR. Class, fatal.

LD602 - LITERAL SEPARATOR MISSING
The beginning of a literal string was found but no data followed the apostrophe. Correct the statement and reissue the request. From LDR. Class, not reprievable.

LD603 - DIRECTIVE (*name*) TABLE OVERFLOW - USE CONTINUATION CARDS
From LDR. Class, fatal.

LD604 - DIRECTIVE (*name*) CONTAINS ILLEGAL CHARACTER
From LDR. Class, fatal.

LD605 - NO DIRECTIVE STATEMENT
From LDR. Class, fatal.

MQ010 - MODSEQ, VERSION *x.xx* (*mm/dd/yy*) - TABLE CREATION RUN
The CREATE parameter was specified on the MODSEQ control statement. From MODSEQ. Class, informative.

MQ011 - *n* TRANSLATION-TABLE ENTRIES WRITTEN (CHECKSUM = *x*)
The *n* translation table entries are written with checksum = *x*. Save the checksum and compare it with the corresponding value in the MQ021 message written during a resequencing run. From MODSEQ. Class, informative.

MQ012 - *m* SOURCE LINES, *n* DECKS COPIED FROM SR TO NSR
A MODSEQ creation run ended. From MODSEQ. Class, informative.

MQ013 - *n* GHOST NAMES STRIPPED FROM NSR
Ghost deck names were found in columns 73-80 of some of the cards in the new source file (NSR) dataset. From MODSEQ. Class, informative.

MQ020 - MODSEQ, VERSION *x.xx* (*mm/dd/yy*) - RESEQUENCING RUN
Both the CREATE and REVERSE parameters are missing from the MODSEQ control statement. From MODSEQ. Class, informative.

MQ021 - *n* TRANSLATION-TABLE ENTRIES READ (CHECKSUM = *x*)
The input phase of a MODSEQ resequencing run ended. Compare the checksum to the corresponding value in the MQ011 message written during the creation run. From MODSEQ. Class, informative.

MQ022 - *m* LINES, *n* DIRECTIVES TRANSLATED
A MODSEQ resequencing run ended. From MODSEQ. Class, informative.

MQ030 - MODSEQ, VERSION *x.xx* (*mm/dd/yy*) - REVERSE RESEQUENCING RUN
The CREATE parameter is missing from the MODSEQ control statement and the REVERSE parameter is present. From MODSEQ. Class, informative.

MQ090 - *n* SOURCE LINES CONTAINED BAD SEQUENCE NUMBERS
One or more records in the old source file (SR) failed a test for a period in column 89. These records and possibly others contain missing or malformed UPDATE sequence numbers. If the number of source lines with bad sequence numbers equals the number of source lines read (see message MQ012), the SR dataset is bad. Make sure the SR dataset is generated by UPDATE with the selected options S=*sr*, DW=80, and SQ. If the numbers reported by MQ012 and MQ090 are different, the SR dataset was modified; some but not all sequence numbers were disturbed. From MODSEQ. Class, fatal.

MQ100 - *nnn* RESEQUENCING ERROR(S) DETECTED
One or more resequencing errors (MQ101, MQ102, MQ103, MQ104, or MQ105) were

detected during a MODSEQ resequencing run. See the entries for the individual error messages. From MODSEQ. Class, caution.

MQ101 - DIRECTIVE *nn* ON LINE *n*. ID NOT REPRESENTED IN PL: *x*
A reference is made by an old mod to an ID that is not present in the original PL. Correct the invalid directive, either before or after the MODSEQ resequencing run. From MODSEQ. Class, caution.

MQ102 - DIRECTIVE *m* ON LINE *n*. NUMBER MISSING FROM PL: *x*
A reference is made by an old mod to a line number absent in the original PL (although the ID is present). Correct the invalid directive, either before or after the MODSEQ resequencing run. From MODSEQ. Class, caution.

MQ103 - DIRECTIVE *d* ON LINE *l*. RANGE SPANS DECKS: *x*,*y*
A deletion range begins and ends in two different decks, or the range is contained within one deck but the beginning follows the end. The test that produces this message is suppressed if an MQ101 or an MQ102 error was detected in the same UPDATE directive. Correct the incorrect directive, either before or after the MODSEQ resequencing run. The correction might involve rewriting the mod. From MODSEQ. Class, fatal.

MQ109 - TRANSLATED LINE *n* TRUNCATED
The translated directive requires more than 80 characters. All characters to the right of column 80 have been discarded. Examine the translated line carefully, because the translation might be correct. If it is incorrect, modify the old mod by omitting comments from the old line image. From MODSEQ. Class, fatal; not reprievable.

MQ111 - THE SR DATASET CANNOT BE $IN OR $OUT
The old source file (SR) was equated to an improper dataset name. Omit the parameter or equate it to a proper dataset name. From MODSEQ. Class, fatal.

MQ112 - THE NSR DATASET CANNOT BE $IN OR $OUT
The new source file (NSR) was equated to an improper dataset name. Omit the parameter or equate it to a proper dataset name. From MODSEQ. Class, fatal.

MQ113 - THE TT DATASET CANNOT BE $IN OR $OUT
The Translation Table (TT) parameter was equated to an improper dataset name. Omit the parameter or equate it to a proper dataset name. From MODSEQ. Class, fatal.

MQ114 - THE OM DATASET CANNOT BE $OUT
The old mod (OM) parameter was equated to an improper dataset name. Omit the parameter or equate it to a proper dataset name. From MODSEQ. Class, fatal.

MQ115 - THE NM DATASET CANNOT BE $IN OR $OUT
The new mod (NM) parameter was equated to an improper dataset name. Omit the parameter or equate it to a proper dataset name. From MODSEQ. Class, fatal.

MQ116 - THE L DATASET CANNOT BE $IN
The L parameter was equated to an improper dataset name. Omit the parameter or equate it to a proper dataset name. From MODSEQ. Class, fatal.

MQ117 - THE E DATASET CANNOT BE $IN OR $OUT
The E parameter was equated to an improper dataset name. Omit the parameter or equate it to a proper dataset name. From MODSEQ. Class, fatal.

MQ121 - WARNING: NO SR FILE IS READ IN A RESEQUENCING RUN
The old source file (SR) parameter should have been omitted. Omit the parameter. From MODSEQ. Class, caution.

MQ122 - WARNING: NO NSR FILE IS WRITTEN IN A RESEQUENCING RUN
The new source file (NSR) parameter specified should have been omitted. Omit the parameter. From MODSEQ. Class, caution.

MQ123 - WARNING: NO OM FILE IS READ IN A CREATION RUN
The old mod (OM) parameter should have been omitted. Omit the parameter. From MODSEQ. Class, caution.

MQ124 - WARNING: NO NM FILE IS WRITTEN IN A CREATION RUN
The new mod (NM) parameter should have been omitted. Omit the parameter. From MODSEQ. Class, caution.

MQ125 - WARNING: THE L FILE IS NOT USED IN A CREATION RUN
The L parameter should have been omitted. Omit the parameter. From MODSEQ. Class, caution.

MQ126 - WARNING: THE E FILE IS NOT USED IN A CREATION RUN
The E parameter should have been omitted. Omit the parameter. From MODSEQ. Class, caution.

MQ127 - WARNING: THE REVERSE KEYWORD IS IGNORED IN A CREATION RUN
The REVERSE parameter should have been omitted. Omit the parameter. From MODSEQ. Class, caution.

MQ131 - THE SR DATASET IS NOT LOCAL TO THE JOB
No Dataset Name Table exists for the old source file (SR) dataset. This dataset is required input for a creation run. Examine the JCL for an error in misnaming or prematurely releasing the SR dataset. If necessary, add an ACCESS or ACQUIRE statement. From MODSEQ. Class, fatal.

MQ132 - THE OM DATASET IS NOT LOCAL TO THE JOB
No Dataset Name Table exists for the old mod (OM) dataset. This dataset is required input for a resequencing run. Examine the job control language for an error in misnaming or prematurely releasing the OM dataset. If necessary, add an ACCESS or ACQUIRE statement. From MODSEQ. Class, fatal.

MQ133 - THE TT DATASET IS NOT LOCAL TO THE JOB
No Dataset Name Table exists for the Translation Table (TT) dataset, which is required input for a resequencing run. Examine the job control language for an error in misnaming or prematurely releasing the TT dataset. If necessary, add an ACCESS or ACQUIRE statement. From MODSEQ. Class, fatal.

MQ141 - THE SR FILE DOES NOT BEGIN WITH THE MASTER CHARACTER, '*x*'
The master character specified by the master character (MC) parameter (an asterisk by default) is incorrect, or the old source file specified by the old source file (SR) parameter contains incorrect data. Verify that the master character displayed in the message is correct. Examine the SR file to determine if it contains the correct UPDATE source cards. From MODSEQ. Class, fatal.

MQ142 - THE OM FILE DOES NOT BEGIN WITH THE MASTER CHARACTER, '*x*'
The master character (MC) parameter (an asterisk by default) is incorrect; or the OM parameter specifies an old mods file that contains incorrect data. Verify that the master character displayed in the message is correct. Examine the OM file to determine if it contains the correct UPDATE mod decks. From MODSEQ. Class, fatal.

MS001 - DATASET *dsname* NOT FOUND
Named input dataset to MODSET was not found. Correct the specified dataset name or verify that the dataset exists, and rerun MODSET. From MODSET. Class, fatal.

MS002 - L=0 SPECIFICATION PREVENTS LISTING
A LIST directive could not be implemented because L=0 was specified on the control statement. Resubmit with the L control statement parameter set to some local dataset. Omitting the L parameter will produce the listing on $OUT. From MODSET. Class, Warning

MS003 - AT LEAST 1 INPUT DATASET REQUIRED FOR MODS
The V, U, and A keywords are set to 0. Equate V, U, or A to a valid dataset name. From MODSET. Class, fatal; not reprievable.

MS004 - MODSET DIRECTIVE ERROR IN THE FOLLOWING RECORD:
A modset directive contained a string of more than eight non-blank characters. MODSET ignored all but the first eight characters of the string. Recode the MODSET directive, insuring that all ident names and directive words (for example, LIST, OMIT) are blank-delimited. From MODSET. Class, warning.

PD000 - PDN = *pdn* ID = *id* ED = *ed* OWN = *owner*
This is a header message for PDM messages that follow. From PDM. Class, informative.

PD000 - DATASET SAVE ON SCRATCH DEVICE *dname*
The dataset resides on a device which, after a system interruption, was specified as *SCRATCH. The SAVE is done. Site operations should be notified and the dataset copied to a non-*SCRATCH device. From PDM. Class, informative.

PD000 - PREVIOUS EDITION EXISTS - ADN *xxxxxxx* IGNORED
On a SAVE or ACQUIRE, access control information was copied from the highest edition of the dataset, not from the specified Attributes Dataset Name *xxxxxxx*. From PDM. Class, informative.

PD000 - UNABLE TO PROPAGATE DXT ATTRIBUTES
The dataset being saved does not have DXT attributes propagated from the previous edition because of a DXT chain error. See a Cray Research analyst. From PDM. Class, informative.

PD000 - *n* UNUSED BLOCKS RELEASED FOR CONTRACTING DATASET
The indicated dataset requires less space than it did previously. The excess blocks (*n* in the message) have been returned to the system for reuse. From PDM as the result of an ADJUST or RELEASE. Class, informative.

PD000 - *n* ADDITIONAL BLOCKS ALLOCATED FOR EXPANDING DATASET
The indicated dataset requires more space than it did previously. In the message, *n* indicates the number of blocks by which it grew. From PDM as the result of an ADJUST. Class, informative.

PD000 - *xxx* TAKEN FROM PREVIOUS EDITION - SPECIFIED VALUE IGNORED
On a SAVE or ACQUIRE, parameter *xxx* was specified with a value different from that found in the highest edition of the dataset. The value of the parameter *xxx* was copied from the highest edition. From PDM. Class, informative.

PD001 - DATASET NOT LOCAL TO JOB
A SAVE, DELETE, ADJUST, MODIFY, or DUMPTM request was issued to PDM, which could not find the local dataset specified in the DN parameter. Create the dataset, or recheck the dataset name. From PDM. Class, informative.

PD002 - MAINTENANCE PERMISSION NOT GRANTED
- SAVE request: The user attempted to save a dataset with control words that did not match those of the previous edition.
- DELETE request: The requester did not supply the proper maintenance control word when the dataset was accessed in order to delete the dataset; or the requester did not supply the UQ parameter on the ACCESS statement.

Specify the correct maintenance control word on the request, or gain unique access. From PDM. Class, informative.

PD003 - EDITION ALREADY EXISTS
A request to SAVE a specific edition of a dataset was issued and a dataset with the specified edition already exists. Delete the saved edition and reissue the request or change the edition parameter. From PDM. Class, informative.

PD004 - DSC FULL
PDM has no room in the Dataset Catalog to save or update the new information. Consult a Cray Research analyst. From PDM. Class, informative.

PD005 - INVALID FUNCTION CODE
The function code given in the PDD is out of range, or a queue that was neither input nor output was given for an EQSDT request. Correct the function code. From PDM. Class, informative.

PD006 - LOCAL DATASET NAME IS ALREADY IN USE
A job tried to access a permanent dataset, but the specified local name (DN) is being used for another local dataset. Use a different local name. From PDM. Class, informative.

PD007 - NO PERMISSION GRANTED
- ACCESS statement: The requested permission did not match the SAVE permission.
- MODIFY statement: One of the three permission control words did not match. Specify the correct value for the requested permission.

From PDM. Class, informative.

PD009 - DATASET NOT FOUND
An attempt was made to access a dataset that does not exist in the permanant dataset base. If an ACQUIRE was issued, this is a normal informative message, indicating that the dataset specified to be transferred from the front end was first searched for on a Cray disk. Check PDN and ID to ensure that the correct dataset is being accessed. From PDM. Class, informative.

PD010 - EDITION NOT FOUND
An attempt was made to access a nonexistent edition of a permanent dataset. Check editions for this permanent dataset. From PDM. Class, informative.

PD012 - DATASET NOT PERMANENT
A request was made to either DELETE, ADJUST, or MODIFY a local dataset that is not permanent. Access the dataset before attempting to delete, adjust, or modify. From PDM. Class, informative.

PD013 - PDN 16 CHARACTERS LONG OR CONTAINS UNPRINTABLE CHARACTERS
The Permanent Dataset Name on a SAVE or MODIFY request either has more than 15 characters or contains unprintable characters (that is, octal codes 000-037 or 177-377). If this was encountered from a control statement, the station translation tables might be at fault; check with a Cray site analyst. For CFT internal or CAL macro calls, check the PDN your program is supplying for the PDD. From PDM. Class, informative.

PD014 - CONTINUATION ERROR
The Dataset Catalog contains erroneous data. See a Cray Research analyst. From PDM. Class, informative.

PD015 - DAT FULL
The system pool does not have room to move the dataset's disk allocation information. If the error recurs, consult a Cray Research analyst. From PDM. Class, informative.

PD018 - DATASET ALREADY ACCESSED BY JOB
An attempt was made to access a dataset that is already accessed. Either release the dataset before attempting to access it, or remove the second access. From PDM. Class, informative.

PD021 - SPECIFIED DATASET IS EMPTY
An attempt was made to change or create a permanent dataset having no disk allocation assigned to it. This message only occurs for datasets that do not meet the conditions for closing the dataset described in the CRAY-OS Version 1 Reference Manual, publication SR-0011. For such a dataset, the user must manually force a flushing of the buffer. From PDM. Class, informative.

PD025 - NO DATASET NAME IN PDD
A request was made in which neither the DN nor the PDN parameter was selected. Correct the parameters. From PDM. Class, informative.

PD026 - ACCESS CONTROL WORD VALIDATION ERROR
On an ACCESS or ACQUIRE, the read, write, or maintenance control word specified did not match the one saved in the Dataset Catalog entry for the requested dataset. From PDM. Class informative.

PD027 - NOTES LENGTH EXCEEDS MAXIMUM ALLOWABLE
On a SAVE, ACQUIRE, or MODIFY, the specified NOTE is longer than the maximum (480 characters). From PDM. Class, informative.

PD027 - MULTIPLE EDITIONS EXIST
An attempt was made to change one of the permission control words for a permanent dataset that has more than one edition. All corresponding permission control words must match for all editions. From PDM. Class, informative.

PD028 - NO UNIQUE ACCESS ON SYSDIR
A unique access was attempted on a dataset that is also entered into the system directory. From PDM. Class, informative.

PD029 - TEXT LENGTH IS ZERO
On a SAVE, ACQUIRE, or MODIFY, a TEXT address was specified, but no length was given. From PDM. Class, informative.

PD030 - MAX TEXT LENGTH EXCEEDED
An attempt was made to dispose a dataset with text associated in a version of COS that does not support that length of text. Decrease the text length to the maximum supported by this version of COS. From PDM. Class, informative.

PD031 - DEVICE DOWN
The dataset resides wholly or partially on a device currently not available. See a Cray Research analyst. From PDM. Class, informative.

PD034 - ACCESS DENIED DUE TO ALLOCATION CONFLICT
The dataset resides on a known flawed area of the disk or on an area that has been reserved for another dataset. See a Cray Research analyst. From PDM. Class, informative.

PD035 - DATASET IS ALREADY PERMANENT
A job tried to save a dataset that is already permanent. From PDM. Class, informative.

PD036 - DSC ENTRY INVALID - ACCESS IMPOSSIBLE
The Dataset Catalog entry for the dataset contains erroneous data. See a Cray Research analyst. From PDM. Class, informative.

PD040 - MULTI-TYPE DISK ALLOCATION INCONSISTENCY - ACCESS IMPOSSIBLE
A discrepancy exists in at least one DSC entry for a dataset that has multiple DSC entries. See a Cray Research analyst. From PDM. Class, informative.

PD041 - MULTI-TYPE DATASET REFERENCES INVALID QDT ENTRY ACCESS DENIED
The Queued Dataset Table became smaller. See a Cray Research analyst. From PDM. Class, informative.

PD042 - MAXIMUM EDITIONS REACHED
An attempt was made to save a dataset without specifying an edition number, after an edition of this dataset was saved with the highest possible edition number. Either save the dataset by specifying an edition number or delete the highest edition of the dataset. From PDM. Class, informative.

PD043 - DATASET IS ON AN ACTIVE SDT QUEUE
An attempt was made to access a spooled dataset (SDT) without first dequeueing it from its associated SDT queue. Dequeue the dataset from the appropriate queue. From PDM. Class, informative.

PD044 - BAD SDT ADDRESS ON ENQUEUE SDT REQUEST
The SDT address given is not within the System Dataset Table. Correct the SDT address. From PDM. Class, informative.

PD045 - DATASET IS ON A SCRATCH DEVICE
There was an attempt to save a dataset that was assigned to a *SCRATCH device. Make sure the dataset is not on a *SCRATCH device before a SAVE. See site operations personnel for valid device. From PDM. Class, informative.

PD046 - ACCESS DENIED DUE TO DSC EXTENSION ERROR
STARTUP found one or more catastrophic errors in the Dataset Catalog entry or the Dataset Catalog extension entries for the accessed dataset. See a Cray Research analyst. From PDM.

PD047 - NOTES LENGTH IS ZERO
On a SAVE, ACQUIRE, or MODIFY, a NOTES address was specified but no length was given. From PDM. Class, informative.

PD049 - DXT ENTRY LIMIT REACHED
No more DXT entries can be allocated to this dataset. The system allows $(2^{15} - 1)$ entries. Consult a Cray Research analyst to determine how the current number of DXT entries can be reduced. From PDM.

PD050 - ATTRIBUTE DATASET NOT LOCAL
A SAVE or PERMIT request was issued to PDM which could not find a local dataset with the dataset name specified in the ADN parameter. Create the dataset, or recheck the dataset name. From PDM.

PD051 - ATTRIBUTES DATASET NOT PERMANENT
The attributes dataset name specified in a SAVE or PERMIT request to PDM is not permanent. Create the dataset, or recheck the dataset name. From PDM.

PD052 - INVALID NOTES BUFFER SPECIFIED
The buffer containing notes information in a SAVE or MODIFY request to PDM is not completely contained within the job's field length. Correct the PMNOTE or PMNOTL value in the PDD. From PDM.

PD053 - INVALID TEXT BUFFER SPECIFIED
The buffer containing text information in a SAVE or MODIFY request to PDM is not completely contained within the job's field length. Correct the PMTXT or PMTXL value in the PDD. From PDM.

PD054 - SPECIFIED PERMIT ENTRY NOT FOUND
The PERMIT entry specified via a 'remove permit' request to PDM could not be located. Correct the USER value associated with the PERMIT request. From PDM.

PD801 - 16-CHARACTER PDS NAME ILLEGAL
The PDS parameter value given on the control statement is too long. Correct the PDS or PDN name to 15 or fewer characters and resubmit the job. From PDSLOAD. Class, fatal; job aborts.

PD802 - 16-CHARACTER USER NUMBER ILLEGAL
The US parameter value given on the control statement is too long. Correct the US length to 15 characters or less. From PDSLOAD. Class, fatal; job aborts.

PD804 - INVALID CONTROL WORD VALUE
The CW parameter value given on the control statement does not match the installation-defined control word. Correct the CW value. From PDSLOAD. Class, fatal; job aborts.

PD805 - ID PARAMETER EXCLUDES USE OF I/O PARAMETERS
An ID parameter was specified on the control statement as well as on the I or the O keyword. It is illegal to request spooled datasets when the ID is specified. Eliminate either the ID parameter or the I (or O). From PDSLOAD. Class, fatal; job aborts.

PD806 - ED PARAMETER REQUIRES PDS PARAMETER AND EXCLUDES USE OF I/O PARAMETERS
An ED parameter was specified on the control statement and either a PDS (or PDN) parameter was not specified or an I or O keyword was specified. Correct the combination of ED, PDS, PDN, I, and O parameters. From PDSLOAD. Class, fatal; job aborts.

PD807 - ED PARAMETER VALUE NON-NUMERIC
The ED parameter value given on the control statement contains a non-numeric character. Correct the ED value. From PDSLOAD. Class, fatal; job aborts.

PD808 - DATASET NOT CREATED BY PDSDUMP
The dataset specified by the PDS or PDN parameter does not have the correct header block to indicate a PDSDUMP creation. Verify the PDS parameter values on this PDSLOAD job and the previous PDSDUMP job. From PDSLOAD. Class, fatal; job aborts.

PD809 - DATASET BY-PASSED *dn* ED = *x*
An attempt was made to load edition *x* of dataset *dn*. The specified edition on the named permanent dataset already exists in the dataset catalog. From PDSLOAD. Class, informative; job continues processing.

PD810 - DATASET LOAD ERROR *dn* ED = *x*
An undefined error occurred while dataset *dn* was being loaded. See a Cray Research analyst. From PDSLOAD. Class, not reprievable; job aborts.

PD811 - I/O ERROR WHILE LOADING *dn*
An I/O error occurred during loading of dataset *dn*, or the number of blocks written does not match the number read. See a Cray Research analyst. From PDSLOAD. Class, fatal; job aborts.

PD812 - US PARAMETER DOES NOT MATCH JOB US
No CW parameter has been specified and the US parameter value given on the PDSLOAD control statement does not match the US parameter value from the JOB statement. Change the US parameter on either the JOB statement or the PDSLOAD statement. From PDSLOAD. Class, fatal; job aborts.

PD813 - IF CW NOT SPECIFIED, JOB MUST HAVE US
No CW parameter was specified on the PDSDUMP control statement, and no US parameter was specified on the JOB statement. Include a US parameter on the JOB statement if CW is not specified on the PDSLOAD statement. From PDSLOAD. Class, fatal; job aborts.

PD814 - UNABLE TO FIND D.S. MATCHING SPECIFICATIONS
In the $PDS dataset, the program did not find a dataset matching the specification on the PDSLOAD call. No reload was done. The failure may be caused by a mistyped control statement or an

incorrect dataset for $PDS. Select a different dataset for $PDS or correct the PDSLOAD control statement and resubmit. From PDSLOAD. Class, fatal, unless NA is specified on the control statement. If NA is specified, the message is a warning.

PD815 - ERROR ON DXT LOAD
An error was encountered while the DXT for a dataset was being loaded. Attempt the load again. From PDSLOAD. Class, fatal; job aborts.

PD816 - ILLEGAL SPECIFICATION OF OWN AND/OR NOWN PARAMETER
CW was not specified, and either NOWN was specified, or OWN was set to a value other than that of the owner. CW must be specified in either case. From PDSLOAD. Class, fatal; job aborts.

PD817 - $PDS MUST NOT BE IN INTERCHANGE FORMAT
An attempt was made to read a $PDS tape dataset in interchange format. This format cannot be successfully loaded. Use a transparent format for $PDS tape datasets. From PDSLOAD. Class, fatal; job aborts.

PD818 - DATASET SELECTED
Indicated dataset was selected by PDSLOAD for loading. From PDSLOAD. Class, informative.

PD819 - CW MUST BE SPECIFIED WITH US
The US parameter requires correct specification of the CW parameter. From PDSLOAD. Class, fatal; jpb aborts.

PD820 - DATASET SIZE LIMIT BYPASSED
The indicated dataset had a dataset size greater than I@MAXLM and could not be loaded. PDSLOAD continues processing. From PDSLOAD. Class, informative.

PD901 - 16-CHARACTER PDS NAME ILLEGAL
The PDS parameter value given on the control statement is too long. Correct the PDS name to 15 characters or less. From PDSDUMP. Class, fatal; job aborts.

PD902 - 16-CHARACTER USER NUMBER ILLEGAL
The US parameter value given on the control statement is too long. Correct the US number to 15 characters or less. From PDSDUMP. Class, fatal; job aborts.

PD903 - ED PARAMETER VALUE NON-NUMERIC
The ED parameter value given on the control statement contains a non-numeric character. Correct the ED value. From PDSDUMP. Class, fatal; job aborts.

PD904 - INVALID CONTROL WORD VALUE
The CW parameter value given on the control statement does not match the installation-defined control word. Correct the CW value. From PDSDUMP. Class, fatal; job aborts.

PD905 - US PARAMETER EXCLUDES USE OF I/O PARAMETERS
An ID parameter was specified on the control statement as well as on the I or the O keyword. It is illegal to request spooled datasets when the ID is specified. Eliminate either the ID parameter or the I (or O) parameter. From PDSDUMP. Class, fatal; job aborts.

PD906 - ED PARAMETER REQUIRES PDS PARAMETER AND EXCLUDES I/O PARAMETERS
Parameter ED was specified on the control statement and either PDS was not specified or an I or O keyword was specified. Correct the combination of ED, PDS, I, and O parameters. From PDSDUMP. Class, fatal; job aborts.

PD907 - ERROR ON SYSTEM DATASET CATALOG PAGE REQUEST
An undefined error was returned after PDM page request. The DSC might be bad. See a Cray Research analyst. From PDSDUMP. Class, fatal; job terminates.

PD908 - DELAY LIST FULL
The permanent dataset cannot be accessed now, but the PDSDUMP access delay list is already full. Rerun PDSDUMP job at a time when the requested datasets are less likely to be in use. From PDSDUMP. Class, fatal; job is terminated.

PD909 - UNABLE TO ACCESS ALL DATASETS. $OUT CONTAINS DATASETS NOT DUMPED
PDSDUMP was unable to access and dump all the datasets after three passes through the access delay list. Rerun PDSDUMP job at a less busy time. From PDSDUMP. Class, fatal; job terminates.

PD910 - SDT NOT FOUND *dn* ED = *xxxx*
An error was returned by PDM during an attempt to dequeue the SDT of spooled dataset *dn*. The SDT table might be bad. See a Cray Research analyst. From PDSDUMP. Class, fatal; job terminates.

PD911 - UNRECOVERABLE ACCESS ERROR *dn* ED = *xxxx*
An error was returned by PDM when trying to access dataset *dn*. See a Cray Research analyst. From PDSDUMP. Class, informative; reprievable.

PD912 - DATASET ALREADY LOCAL TO THE JOB
The dataset has already been accessed by this job. From PDSDUMP. Class, informative; job continues processing.

PD913 - DELETED WHILE WAITING *dn* ED = *xxxx*
The DSC does not contain the requested dataset *dn*. From PDSDUMP. Class, informative; job continues processing.

PD914 - DUMP TIME ERROR *dn* ED = *xxxx*
An error was returned by PDM when trying to put the current time into the DSC entry of the specified dataset *dn*. See a Cray Research analyst. From PDSDUMP. Class, fatal; job terminates.

PD915 - DELETE ERROR *dn* ED = *xxxx*
An error was returned by PDM when dumped dataset *dn* was being deleted. See a systems analyst. From PDSDUMP. Class, fatal; job terminates.

PD916 - QUEUE SDT ERROR *dn* ED = *xxxx*
An error was returned by PDM when trying to requeue the SDT of spooled dataset *dn*. See a Cray Research analyst. From PDSDUMP. Class, fatal; job terminates.

PD917 - DAT CONTINUATION ERROR *dn* ED = *xxxx*
An undefined error occurred during reading of a DSC continuation page or the entry read did not have its continuation flag bit set. See a Cray Research analyst. From PDSDUMP. Class, fatal; job terminates.

PD918 - I/O ERROR WHILE READING *dn*
An error occurred during reading of dataset *dn*. See a Cray Research analyst. From PDSDUMP. Class, fatal; job aborts.

PD921 - EXECUTE-ONLY DATASET CANNOT BE DUMPED
Execute-only datasets cannot be copied using PDSDUMP. From PDSDUMP. Class, caution; job continues with next dataset.

PD922 - DATASET RESIDES ON A DOWN DEVICE
The device on which the dataset resides is not currently active in the system. This dataset is skipped. From PDSDUMP. Class, caution; job continues with next dataset.

PD923 - DATASET HAS ALLOCATION CONFLICTS
The dataset resides on a known flawed area of the disk or on an area that has been reserved for another dataset. From PDSDUMP. Class, fatal; job aborts.

PD924 - DATASET FLAGGED AS HAVING CATASTROPHIC ERROR
The dataset cannot be dumped. Recreate the dataset if possible. From PDSDUMP. Class, fatal; job aborts.

PD925 - DATASET IS INCONSISTENTLY ALLOCATED
The dataset cannot be dumped. Recreate the dataset if possible. From PDSDUMP. Class, fatal; job aborts.

PD926 - DATASET TEMPORARILY INACTIVATED DUE TO SMALLER SDT SIZE
The dataset cannot be dumped because the QDT index for the dataset lies outside of the valid limits for the QDT. From PDSDUMP. Class, fatal; job aborts.

PD927 - DAT CONTINUATION PAGE READ ERROR
PDSDUMP called PDM to read a DSC CONTINUATION page and got a return status indicating a read error. Contact a Cray Research analyst to assess the state of the DSC. If it is correct, there is a problem within PDM. Class, fatal.

PD928 - CANNOT GAIN UNIQUE ACCESS TO SDR DATASET
The user requested PDSDUMP to delete datasets after dumping them. One dataset selected for dump/delete was entered into the SDR so it could not be accessed uniquely to allow the delete. That dataset was bypassed by PDSDUMP and not dumped or deleted. PDM issues a message immediately preceding this one, printing the name of the dataset that led to the problem. Re-invoke PDSDUMP without the D option to back up the datasets which are SDR resident. From PDSDUMP. Class, informative

PD929 - $PDS MUST NOT BE IN INTERCHANGE FORMAT
An attempt was made to build a tape-resident $PDS dataset in interchange format. Use a transparent format for these datasets. From PDSDUMP. Class, fatal; job aborts.

PS001 - PASCAL COMPILED *s*, *n* SOURCE LINES
The Pascal compiler has completed compilation of program module *s*, which consisted of *n* source lines. From Pascal. Class, informative.

PS002 - PASCAL CODE: *n* OCTAL, DATA: *m* OCTAL
The Pascal compiler generated *n* (octal) words of code and *m* (octal) words of static data for the current program module. From Pascal. Class, informative.

PS003 - PASCAL STACK: *n* OCTAL, HEAP: *m* OCTAL
At runtime, the program compiled by the Pascal compiler will initially request *n* (octal) words of stack space and *m* (octal) words of heap space. From Pascal. Class, informative.

PS004 - PASCAL *n* ERRORS IN *s*, NO CODE GENERATED
Errors were detected by the Pascal compiler during the compilation of program module *s*; no code was generated. The Pascal compiler will abort the job step if the A+ option was on the compiler call line. From Pascal. Class, fatal.

PS005 - PASCAL NORMAL TERMINATION
The Pascal compiler terminated without errors. From Pascal. Class, informative.

PS006 - PASCAL SOURCE LINE TOO LONG, NO CODE GENERATED
The Pascal compiler detected a source line greater than 140 characters in width. Compilation was aborted. From Pascal. Class, fatal.

PS007 - PASCAL PREMATURE EOF ON INPUT SOURCE FILE
The Pascal compiler detected an end of file on the input source file prior to the end of the current program module. Compilation was aborted. From Pascal. Class, fatal.

PS008 - PASCAL PREMATURE EOD ON INPUT SOURCE FILE
The Pascal compiler detected an end of data on the input source file prior to the end of the current program module. Compilation was aborted. From Pascal. Class, fatal.

PS009 - PASCAL HARDWARE I/O ERROR ON INPUT SOURCE FILE
The Pascal compiler detected an unrecoverable hardware I/O error on the input source file. Compilation was aborted. From Pascal. Class, fatal.

PS010 - PASCAL UNRECOGNIZED KEYWORD *s* ON COMMAND LINE IGNORED
The Pascal compiler does not recognize a keyword (*s*) on the Pascal control statement. From Pascal. Class, fatal.

PS011 - PASCAL INVALID INPUT FILE NAME *s*, COMPILE TERMINATED
The value specified for the I parameter on the Pascal control statement is invalid. From Pascal. Class, fatal.

PS012 - PASCAL INVALID LIST FILE NAME *s*, COMPILE TERMINATED
The value specified for the L parameter on the Pascal control statement is invalid. From Pascal. Class, fatal.

PS013 - PASCAL INVALID BLD FILE NAME *s*, COMPILE TERMINATED
The value specified for the B parameter on the Pascal control statement is invalid. From Pascal. Class, fatal.

PS014 - PASCAL INPUT FILE SPECIFIED TWICE, COMPILE TERMINATED
The I parameter occurred more than once on the Pascal control statement. From Pascal. Class, fatal.

PS015 - PASCAL LIST FILE SPECIFIED TWICE, COMPILE TERMINATED
The L parameter occurred more than once on the Pascal control statement. From Pascal. Class, fatal.

PS016 - PASCAL BLD FILE SPECIFIED TWICE, COMPILE TERMINATED
The B parameter occurred more than once on the Pascal control statement. From Pascal. Class, fatal.

PS017 - PASCAL BLD FILE AND LIST FILE SAME, COMPILE TERMINATED
The user specified the same dataset name for both the B and L parameters on the Pascal control statement. From Pascal. Class, fatal.

PS018 - PASCAL INPUT FILE AND LIST FILE SAME, COMPILE TERMINATED
The user specified the same dataset name for both the I and L parameters on the Pascal control statement. From Pascal. Class, fatal.

PS019 - PASCAL INPUT FILE AND BLD FILE SAME, COMPILE TERMINATED
The user specified the same dataset name for both the I and B parameters on the Pascal control statement. From Pascal. Class, fatal.

PS020 - PASCAL ABNORMAL TERMINATION
The Pascal compiler is unable to successfully complete compilation of the current program module. From Pascal. Class, fatal.

RI001 - JOB FAILED TO ROLL IN-RERUN
The job failed to roll in and was rerun. From CSP. Class, informative.

RI002 - JOB FAILED TO ROLL IN AND NOT RERUNNABALE
Resubmit the job. From CSP. Class, informative.

RJ001 - JOB RERUN BY SYSTEM RECOVERY
During restart a nonrecoverable job was rerun. From CSP. Class, informative.

RJ002 - SYSTEM RECOVERY - JOB NOT RERUNNABLE OR RECOVERABLE
During restart a nonrecoverable job could not be rerun. Resubmit the job. From CSP. Class, informative.

RT1001 - HEAP SPACE EXAUSTED
All of the memory space set aside for dynamic allocation has been used. To increase the heap space, set the MFL parameter on the JOB statement in the control statement file to a larger value. From Pascal routine P$NEW. Class, fatal.

RT1002 - DISPOSED AREA NOT IN HEAP
A DISPOSE statement attempted to deallocate a dynamic variable that was not currently allocated. Check the program logic to ensure that each DISPOSE statement is associated with a NEW statement to allocate the dynamic variable. From Pascal routine P$DISP. Class, fatal.

RT1003 - DISPOSED AREA HAS BAD LINKAGE WORD
The heap management linkage word has been altered. Either the pointer being

disposed has been stored indirect with negative offset or the allocated area before the disposed area has been stored indirectly past its bounds. Turn on range checking and recompile and rerun the program. From Pascal routine P$DISP. Class, fatal.

RT1004 - INTEGER OVERFLOW
An integer in the program has exceeded the highest possible value. If the variable was declared as type I24 (a 24-bit integer), a larger value can be accommodated by changing the declaration to type INTEGER (64 bits). MAXINT specifies the largest 64-bit integer value possible on a machine. (The value of MAXINT on a Cray computer is 2^{64} -1.) If MAXINT has been exceeded, the program must be modified. From any Pascal runtime library routine. Class, fatal.

RT1005 - STACK OVERFLOW
All slots in the runtime stack are full as a result of too many recursive procedure or function calls. Check for the possibility of infinite recursion. To increase the stack size, reset the value of the S compiler directive (see the Pascal Reference Manual, CRI publication SR-0060). From any Pascal runtime library routine. Class, fatal.

RT1006 - DIVISION BY ZERO
The program attempted to divide a number by 0. Dividing by 0 is not a valid operation. Change the program to avoid division by 0. From any Pascal routine with the division by 0 checking option on. Class, fatal.

RT1007 - NO CASE PROVIDED FOR THIS VALUE
None of the labels in a CASE statement matched the value of the selector, and the optional OTHERWISE clause was not specified. Add an OTHERWISE clause to handle unmatched values. From any Pascal routine with the CASE statement checking option on. Class, fatal.

RT1008 - INDEX EXPRESSION OUT OF BOUNDS
The program attempted to access an element outside the bounds of the declared structure. Change the program to ensure that the index value does not exceed the declared bounds. From any Pascal routine with the index checking option on. Class, fatal.

RT1009 - ASSIGNMENT OUT OF BOUNDS
In an assignment statement, the value on the right side was outside of range of valid values for the variable on the left side. For example, a value of -3 on the right side is out of bounds for a variable on the left side with a declared range of -2..4. Change the program to prevent such an assignment. From any Pascal routine compiled with the assignment checking option on. Class, fatal.

RT1010 - INVALID POINTER REFERENCE
The program attempted to use a pointer variable that did not point to a valid item. Either the pointer was not initialized or the item to which it pointed had been deallocated (using the predefined procedure DISPOSE). Check these possible errors in the program and make appropriate changes. From any Pascal routine compiled with the pointer checking option on. Class, fatal.

RT1011 - SUCC FUNCTION OUT OF BOUNDS
The SUCC function attempted to access an element outside of the declared bounds of the type. If the last element in the type, for instance, is specified as a parameter in the SUCC function call, the function attempts to access an invalid element. Change the program to ensure a valid parameter. From any Pascal routine with the SUCC function checking option on. Class, fatal.

RT1012 - PRED FUNCTION OUT OF BOUNDS
The PRED function attempted to access an element outside of the declared bounds of the type. If the first element in the type, for instance, is specified as a parameter in the PRED function call, the function attempts to access an invalid element. Change the program to ensure a valid parameter. From any Pascal routine with the PRED function checking option on. Class, fatal.

RT1013 - SET ELEMENT OUT OF BOUNDS
The program referenced an element that is outside the bounds of the specified set. Change the program to ensure that only elements included in the set are referenced. From any Pascal routine with the SET checking option on. Class, fatal.

RT1015 - HEAP CHECKING CAUGHT DAMAGED HEAP
The heap data structure (used internally for dynamic allocation) was corrupted. A compiler error is probably responsible. Contact a Cray Research site analyst. From P$HEAP. Class, fatal.

RT1016 - DISPOSE OF UNALLOCATED AREA
A Pascal DISPOSE statement attempted to deallocate a dynamic variable that was not currently allocated. Check the program to ensure that the argument to the DISPOSE procedure is a valid pointer. From P$HEAP. Class, fatal.

RT1017 - CHR FUNCTION ARGUMENT OUT OF BOUNDS
The CHR function was called with an argument less than 0 or greater than 127. Change and rerun the program. From any Pascal routine with the CHR function checking option set on. Class, fatal.

RT1018 - MOD BY ZERO OR NEGATIVE
The program attempted to perform a MOD function with modulus less than or equal to 0. Change and rerun the program. From any Pascal routine with the MOD checking option turned on. Class, fatal.

RT3001 - NAMED FILE DOES NOT EXIST
The file that was to be opened could not be located. Check the spelling of the file name and ensure that a file of the same name is local to the job. From Pascal routine P$OPEN. Class, fatal.

RT3004 - ATTEMPTED READ ON UNOPENED FILE
The file could not be read because it was not previously opened. Open the file using the P$OPEN library procedure before attempting the read. From any Pascal I/O routine. Class, fatal.

RT3005 - ATTEMPTED WRITE TO AN UNOPENED FILE
The file could not be written because it was not previously opened. Open the file using the P$OPEN library procedure before attempting the write. From any Pascal I/O routine. Class, fatal.

RT3006 - FUNCTION ATTEMPTED ON UNOPENED FILE
The file could not be used because it was not previously opened. Open the file using the P$OPEN library procedure before attempting the function. From any Pascal I/O routine. Class, fatal.

RT3007 - ATTEMPTED READ PAST EOF/EOD
The program attempted to read beyond the end-of-file or the end-of-dataset indicator. For a text file, insert an EOLN (end-of-line) test and check for EOF only when EOLN is TRUE. From any Pascal input routine. Class, fatal.

RT3008 - ATTEMPTED WRITE PAST EOF/EOD
The program attempted to write beyond the end-of-file or the end-of-dataset indicator. The file was opened after being previously closed. Call the P$MODIFY procedure before writing the file. From any Pascal output routine. Class, fatal.

RT3009 - INPUT FORMAT ERROR ON INTEGER READ
The data to be read was not an integer. The type of the variable in the READ or READLN statement must match the type of the input data. From Pascal routine P$RDI*. Class, fatal.

RT3010 - INPUT FORMAT ERROR ON REAL READ
The data to be read was not of type REAL. The type of the variable in the READ or READLN statement must match the type of the input data. From Pascal routine P$RDR. Class, fatal.

RT3011 - INPUT RECORD EXCEEDS BUFFER SIZE
If the program is reading a text file, the line length is too long. The maximum line length is 140 characters. If the program is reading a nontext file, the data is probably of the wrong type. Ensure that the data is of the same type as the variable specified to contain that data. From Pascal routine P$RD. Class, fatal.

RT3012 - OUTPUT RECORD EXCEEDS BUFFER SIZE
The program tried to write more than 140 characters to a single line of a text file. Write an end-of-line (using the WRITELN statement) before reaching the 141st character. From Pascal routine P$WR. Class, fatal.

RT3013 - HARDWARE ERROR
A hardware error caused the program to fail. Attempt to run the program again. If this message continues to appear, contact a Cray Research customer engineer. From any Pascal I/O routine. Class, fatal.

RT3015 - ATTEMPTED EOLN ON NONTEXT FILE
The program invoked the EOLN function on a file other than a text file. A nontext file does not contain end-of-line indicators, and the EOLN function applies only to text files. Either remove the EOLN function or declare the file as type TEXT. From Pascal routine P$EOLN. Class, fatal.

RT3016 - ATTEMPTED EOF ON UNSTRUCTURED FILE
The program invoked the EOF function on an unblocked dataset. From Pascal routine P$EOF. Class, fatal.

RT3017 - CANNOT RESET OUTPUT OR $OUT
The RESET function was invoked on an output file. RESET is for input files only. Change RESET to REWRITE for an output file. From Pascal routine P$RESET. Class, fatal.

RT3018 - CANNOT REWRITE INPUT OR $IN
The REWRITE function was invoked on an input file. REWRITE is for output files only. Change REWRITE to RESET for an input file. From Pascal routine P$REWRIT. Class, fatal.

SC001 - CFFT2 CALLED WITH PARAMETER N NOT OF THE FORM N=2**I WHERE $2 \leq I \leq 16$
CFFT2 was called with an input vector that was not of length 2^I on the CRAY-1. See the Library Reference Manual, CRI publication SR-0014, for instructions on using the routine CFFT2. From $SCILIB. Class, 2; user abort; reprievable.

SC002 - CRFFT2 CALLED WITH PARAMETER N NOT OF THE FORM N=2**I WHERE 3 < I < 17
CRFFT2 was called with N out of range or not of length 2^i. See the Library Reference Manual, CRI publication SR-0014, for correct usage of CRFFT2. From $SCILIB. Class, 2; user abort; reprievable.

SC003 - FILTERG CALL WITH N LESS THAN M
In a call to FILTERG, two input vectors of length N and M, respectively, were encountered with M larger than N. See the Library Reference Manual, CRI publication SR-0014, for correct usage of FILTERG. From $SCILIB. Class, 2; user abort; reprievable.

SC004 - FILTERS CALLED WITH N LESS THAN M
In a call to FILTERS, two input vectors of length M and N, respectively, were encountered with M larger than N. See the Library Reference Manual, CRI publication SR-0014, for correct usage of FILTERS. From $SCILIB. Class, 2; user abort; reprievable.

SC005 - MXV CALLED WITH NAR NONPOSITIVE
NAR specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NAR is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC006 - MXV CALLED WITH NBC NONPOSITVE
NBC specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NBC is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC007 - MXVA CALLED WITH NAR NONPOSITIVE
The MXVA call exceeded the limit of 9 parameters. See the Library Reference Manual, CRI publication SR-0014. From $SCILIB. Class, 2; user abort; reprievable.

SC008 - MXVA CALLED WITH NAR NONPOSITIVE
NAR specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NAR is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC009 - MINV CALLED WITH MATRIX REDUCED TO A SINGULAR MATRIX
A system of linear equations cannot be meaningfully solved if the matrix describing the system is singular. When the algorithm encounters a singular matrix, it stops and prints an error message. Analyze the data or method closely. From $SCILIB. Class, 2; user abort; reprievable.

SC010 - MINV CALLED WITH A PARTIAL PRODUCT OF PIVOT ELEMENTS LESS THAN EPS
During the computation of MINV, the product of the pivot elements became less than the EPS specified in the call. Redefine EPS to be smaller or analyze data and method more carefully. From $SCILIB. Class, 2; user abort; reprievable.

SC011 - MXM CALLED WITH NAR NONPOSITIVE
NAR specifies one dimension of a matrix product. This dimension must be positive. Determine if the parameter NAR is nonpositive. From $SCILIB. Class, 2; user abort; reprievable.

SC012 - MXM CALLED WITH NBC NONPOSITIVE
NBC specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NBC is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC013 - MXM CALLED WITH NAC NONPOSITIVE
NAC specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NAC is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC014 - MXMA CALLED WITH NAR NONPOSITIVE
NAR specifies one dimension of a matrix product. This dimension must be positive. Determine if the parameter NAR is nonpositive. From $SCILIB. Class, 2; user abort; reprievable.

SC015 - MXMA CALLED WITH NBC NONPOSITIVE
NBC specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NBC is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC016 - MXMA CALLED WITH NAC NONPOSITIVE
NAC specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NAC is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC017 - MXVA CALLED WITH NAC NONPOSITIVE
NAC specifies one of the dimensions of a matrix multiply. This dimension must be positive. Determine if the parameter NAC is positive. From $SCILIB. Class, 2; user abort; reprievable.

SC018 - RCFFT2 CALLED WITH PARAMETER N NOT EQUAL TO FORM N=2**I WHERE 3 < I < 17
RCFFT2 is written to operate on an input vector of length N, where N is a power of 2 and is within a certain range. See the Library Reference Manual, CRI publication SR-0014, for correct use of the parameter N in a call to RCFFT2. From $SCILIB. Class, 2; user abort; reprievable.

SC019 - CMACH CALLED WITH PARAMETER JOB NOT EQUAL TO 1, 2 OR 3
The value of the JOB parameter does not equal 1, 2, or 3. Check the value of JOB and resubmit. From $SCILIB. Class, 2; user abort; reprievable.

SC020 - SROTM - CALLED WITH INCORRECT PARAMETER KEY
The last parameter of a call to SROTM is incorrect. See the Library Reference Manual, CRI publication SR-0014. From $SCILIB. Class, 2; user abort; reprievable.

SC021 - MXMA CALLED WITH TOO MANY PARAMETERS
A MXMA call exceeded the limit of 12 parameters. See the Library Reference Manual, CRI publication SR-0014. From $SCILIB. Class, 2; user abort; reprievable.

SC022 - SMACH CALLED WITH PARAMETER JJ NOT EQUAL TO 1, 2 or 3
The JJ parameter contains an illegal value. Check the parameter JJ in the call to SMACH. See the Library Reference Manual, CRI publication SR-0014. From $SCILIB. Class, 2; user abort; reprievable.

SC035 - CFFT2 CALLED WITH PARAMETER N NOT OF THE FORM N=2**I WHERE $3 \leq I \leq 16$
CFFT2 was called with an input vector that was not of length 2^I on the Cray X-MP. See the Library Reference manual, CRI publication SR-0014, for instructions on using the routine CFFT2. From $SCILIB. Class, 2; user abort; reprievable.

SF001 - *n* IDENTIFIERS, *m* REFERENCES FOR *program*
End of SKOL cross reference generation. *n* includes all programmer-invented names except those that occur only inside literal strings. *m* excludes those that occur inside a literal string and therefore do not appear in the cross reference listing. *program* is the name of the MAIN segment or the name of the first major segment if there is no MAIN segment. From SKOLREF. Class, informative.

SF002 - MORE THAN 5197 TABLE ENTRIES; REMAINDER IGNORED
The SKOL program contains too many programmer-invented names and therefore cannot be fully cross referenced. Break the program into separate compilations. From SKOLREF. Class, warning.

SF003 - PROGRAM LONGER THAN 9999 LINES; REMAINDER IGNORED
The SKOL program is too long to be fully cross referenced. Break the program into separate compilations. From SKOLREF. Class, warning.

SG000 - SEGLDR VERSION x.nn - *mm/dd/yy*
This logfile message identifies the copy of SEGLDR being used by version number and generation date. From SEGLDR.

SG010 - ILLEGAL INPUT DATASET NAME- *dn*
SG011 - ILLEGAL LIST DATASET NAME- *dn*
The above two messages flag illegal input and list dataset names. The SEGLDR control statement I or L keyword is equated to an inappropriate dataset name. From SEGLDR.

SG012 - INPUT DATASET NOT LOCAL- *dn*
This message indicates that the dataset containing SEGLDR directives is nonexistent. Probable cause is a misspelled value for the I control statement parameter or failure to access the input dataset. From SEGLDR.

SG013 - ILLEGAL DW PARAMETER VALUE - *value*
The width of input directives must be specified as a number greater than 0 and less than 81. From SEGLDR.

SG015 - GLOBAL DIRECTIVE ERRORS
The SEGLDR global directives contain one or more fatal errors. Examine the listing dataset for specific fatal error diagnostics. From SEGLDR.

SG017 - SEGMENT TREE STRUCTURE ERROR
The segment tree is faulty. Examine the listing dataset for specific fatal error diagnostics. From SEGLDR.

SG019 - SEGMENT DESCRIPTION ERRORS
One or more segments are improperly described by a SEGMENT/ENDSEG directive group. Examine the listing dataset for specific fatal error diagnostics. From SEGLDR.

SG020 - LOAD ERRORS, DN=*dn*
SEGLDR detects one or more loading errors while processing the indicated load dataset. Examine the listing dataset for specific fatal error diagnostics. From SEGLDR.

SG021 - SEGLDR RESIDENT ROUTINE MISSING
The routine that handles intersegment subroutine calls was not located. Normally, this subroutine appears in one of the default libraries. Possible causes of this error are:

- The NODEFLIB directive was used and a SEGLDR resident routine is not provided by the user.
- The resident routine is not provided in a system-supplied library (see the Cray Research site analyst).

From SEGLDR.

SG023 - 'FIRST' ENTRY NOT FOUND
The entry point named by the FIRST directive is not located in any binary input dataset. From SEGLDR.

SG024 - NO TRANSFER ENTRY POINT
The entry point named in the XFER directive is not located in any binary input dataset. From SEGLDR.

SG025 - MODULE ASSIGNMENT ERRORS
One or more subroutine linkages between user-fixed modules are improper. Subroutine calls must not call across segment tree branches. A subroutine only calls routines in the same segment, predecessor segments, or successor segments. Examine the listing dataset for specific fatal error diagnostics. From SEGLDR.

SG026 - TRANSFER ENTRY NOT IN ROOT SEGMENT
The transfer entry point in a segmented load must be included in the root segment. Either put it in the root segment or specify another transfer entry point. From SEGLDR.

SG027 - DYNAMIC COMMON BLOCK NOT LOADED
The common block named by the DYNAMIC directive was not found, or was discarded by the call tree trimming process. From SEGLDR.

SG028 - POSSIBLE CALLING SEQUENCE MISMATCH, DN = *dn*
Some included routines may have been compiled with different calling sequence versions. Make sure that all routines are compiled with the same calling sequence version. From SEGLDR.

SG163 - NULL INPUT FILE
The input file is empty. From SEGLDR.

SG200 - *nnn* UNSATISFIED EXTERNALS
The program that was loaded contains one or more unsatisfied externals. A complete report of unsatisfied external references is contained in the listing generated by SEGLDR for the load. From SEGLDR.

SG300 - AVAILABLE MEMORY EXHAUSTED
SEGLDR is unable to load the program in the available memory field. Increase the maximum memory field size (the JOB control statement MFL parameter). From SEGLDR.

SG800 - TOO MANY SUBROUTINE ARGUMENTS
Programmer error: the maximum number of arguments for a single subroutine that can be handled by $SEGRES is 512. From SEGLDR.

SG801 - SDT LOOKUP FAILURE
SG802 - SLT LOOKUP FAILURE
Both messages indicate an internal failure of SEGLDR routine $SEGRES. Forward the failing job to a Cray Research analyst. From SEGLDR.

SG803 - /$SEGRES/ DESTROYED
The user's program wrote over the $SEGRES common block. Use $TRBK or dump information to isolate the code that caused the problem. From SEGLDR.

SG804 - BUFFER I/O ERROR, DSN = *dn*
There was a disk malfunction as segments were read to or written from memory during program execution (*dn* identifies the dataset.) From SEGLDR.

SG900 - UNEXPECTED READ STATUS, DATASET *dn* POS=*n*
SG901 - FIRST TABLE NOT A PDT, DATASET *dn* POS=*n*
SG902 - BAD READ STATUS OR PDT WC, DATASET *dn* POS=*n*
SG903 - UNEXPECTED READ STATUS, DATASET *dn* MODULE *mod*
SG904 - UNEXPECTED TBL=*ii* DATASET *dn* MODULE *mod*
SG905 - UNKNOWN TBL=*ii* DATASET *dn* MODULE *mod*
The above logfile messages are issued in response to faulty binary input datasets (both LIB and BIN). *dn* is a dataset name, *nnn* is an octal number corresponding to the dataset word address where the error was detected, *ii* is a 2-digit octal number between 00 and 17, and *mod* is the name of the bad module when known. One possible cause of the above conditions is nonbinary input (a dataset containing textual data). From SEGLDR.

SG906 - BAD BI FOR MODULE *mod* IN DATASET *dn*
SG907 - PWT REFERENCES IRB, MODULE *mod* DATASET *dn*
SG908 - BAD BI IN DUP. TBL, MODULE *mod*
SG909 - DPT REFERENCES IRB, MODULE *mod*
SG910 - BAD BI IN PWT, MODULE *mod*
SG911 - PWT REFERENCES IRB, MODULE *mod*
SG912 - BAD XI, MODULE *mod*
SG913 - BAD XI IN XRT, MODULE *mod*
SG914 - XRT REFERENCES IRB, MODULE *mod*
SG915 - BAD BI0 IN BRT, MODULE *mod*
SG916 - BRT BI0 REFERENCES IRB, MODULE *mod*
SG917 - BAD BI IN BRT, MODULE *mod*
SG918 - BAD BI IN XBRT, MODULE *mod*
These messages result from a faulty binary module. The cause is an internal error in the assembler or compiler which created the module. SEGLDR does not try to recover from these errors; abort is immediate. Refer the binary input to a Cray Research analyst. From SEGLDR.

SG919 - ERROR RE-READING ROOT SEGMENT
An error was encountered while reading from the output dataset during creation of the executable dataset. From SEGLDR.

SG930 - CODE GENERATION IN */cbname/* BY MODULE *mod*
The user attempted to have code generation in the named common block. This is not allowed. From SEGLDR.

SI001 - NF or NR PARAMETER NON-NUMERIC
The NF or NR parameter on the control statement contains a non-numeric character. Correct the NF or NR value. From SKIP or COPY. Class, fatal; job aborts.

SK001 - SKOL PREPROCESSOR, REVISION LEVEL *x.n*
Start of macro translation From SKOL. Class, informative.

SK002 - TRANSLATION TIME = *min.sec*
End of macro translation From SKOL. Class, informative.

SK003 - *n* MACRO TRANSLATION ERRORS
End of a macro translation in which fatal errors occurred. Examine the error listing, correct the errors, and resubmit the job. If it is desired to proceed with the next job step following the macro translation without first correcting all the errors, place a %E*n* directive in the source text. The value of *n* must be greater than or equal to the number of fatal errors in order to change the class of this message from fatal to warning. From SKOL. Class, fatal.

SK003 - *n* MACRO TRANSLATION WARNINGS
End of a macro translation in which warnings were issued. Examine the error listing and correct the causes of the warning messages. From SKOL. Class, warning.

SK004 - AN %E DIRECTIVE WAS PROCESSED THAT ALLOWS UP TO *n* ERRORS
End of a macro translation in which the source text contained a %E*n* directive and in which more than *n* fatal errors were detected. Increase the value of *n* or reduce the number of fatal errors. From SKOL. Class, fatal.

SK006 - COULDN'T ASSIGN 'O' DATASET
The dataset that is to receive the generated FORTRAN code is already in use for some other purpose. Change the O parameter on the SKOL control statement, or omit the O parameter so that O=$SKF by default. From SKOL. Class, fatal.

SK007 - COULDN'T ACCESS *dataset*
The named dataset, which can be the source input (I) dataset, a supplementary input (M) dataset, or SKOLTXT, cannot be accessed. For SKOL's implicit dataset access method to work, an edition of the dataset must exist with no ID and no read permission word. If an ID or R parameter is required, add an ACCESS statement to the control statement file. From SKOL. Class, fatal.

SK008 - OUTPUT BUFFER OVERFLOW
A user-defined macro expansion is too large for the buffer. Redefine the macro in error, using inner macros where appropriate, to reduce the amount of expansion done at a single time. If necessary, turn on SKOL's macro trace feature. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*---------- ERROR - EXPANSION BUFFER OVERFLOW
Either a used-defined macro or constant caused a recursive match or a user-defined macro expansion was too large for the buffer. If the error is not obvious, turn on SKOL's macro trace feature. If no accidental recursion occurred, redefine the macro that is overflowing the buffer. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*---------- - ERROR - INPUT BUFFER OVERFLOW
A statement or a literal string contains more than 25 source lines. Check the column between the sequence number and the nesting level for an apostrophe that indicates a literal string continuation. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*---------- - ERROR - LABEL STACK OVERFLOW
An error occurred in a user-defined structural macro (one that manipulates the label stack) or the nesting level exceeded 46. Correct the user-defined macro or reduce the nesting level by breaking the major segment into separate routines, subroutines, or functions. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*---------- - ERROR - LABEL STACK UNDERFLOW
An error occurred in a user-defined structural macro (one that manipulates the label stack) or the standard macros that protect against stack underflow have been redefined. Correct the user-defined macro or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*---------- ERROR - MACRO BUFFER OVERFLOW
Too many user-defined macros for the available space. Make some of the macros local by placing their definitions inside a major segment, break up the program in separate compilations, or recompile the SKOL macro translator with a larger dimension for the array MM. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*---------- - ERROR - MACRO STACK OVERFLOW
An error occurred in a user-defined structural macro (one that manipulates

the macro symbol stack), or the nesting level exceeded 46. Correct the user-defined macro, or reduce the nesting level by breaking the major segment into separate routines, subroutines, or functions. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*------------ - ERROR - MACRO STACK UNDERFLOW
An error occurred in a user-defined structural macro (one that manipulates the macro symbol stack); or the standard macros for protection against stack underflow were redefined. Correct the user-defined macro or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*------------ - ERROR - OUTPUT BUFFER OVERFLOW
A user-defined macro expansion is too large for the buffer. Redefine the invalid macro, using inner macros where appropriate, to reduce the expansion done at a single time. If necessary, turn on SKOL's macro trace. From SKOL. Class, fatal.

SK009 - STORAGE OVERFLOW
*------------ - ERROR - PARAMETER BUFFER OVERFLOW
The total length of the parameters exceeds the space available for them. In the column between the sequence number and the nesting level, check for an apostrophe indicating a literal string continuation. Redefine the erroneous macro, using inner macros where appropriate, to reduce the expansion at one time. If necessary, use SKOL's macro trace feature. From SKOL. Class, fatal.

SL000 - BAD CALL TO ROUTINE INSERT ASCII
Improper control tables or an illegal message number. See a Cray Research analyst. From $SYSLIB. Class, user abort.

SL000 - BAD CALL TO SYSTEM ERROR PROCESSOR
Illegal error message number. See a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL001 - *action*, *dn* I/O SYSTEM ERROR
The system returned an undefined error after an I/O operation. *action* can take the value READ or WRITE. Send a dump to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL002 - *action*, *dn* UNRECOVERED HARDWARE ERROR
The system detected a permanent hardware error and was unable to complete the I/O operation. *action* can take the value READ or WRITE. Check the status of the device. If the device was not ready, make it ready and resubmit the job. If the device appears to be operational, inform a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL003 - *action*, *dn* UNRECOVERED DATA ERROR
During an I/O operation, the system detected a discrepancy in the data. Notify the system operator. If the problem persists, the dataset must be recreated. From $SYSLIB. Class, 2; user abort; reprievable.

SL004 - *action*, *dn* BLOCK NUMBER ERROR
An I/O request was completed, but the block number being read was not the expected block number. Either this is a system error or the dataset being read is unblocked or has been corrupted. Forward a dump to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL005 - *action*, *dn* READ ON WRITE ONLY DATASET
An input request was issued on a dataset that does not have read permission. If the dataset has write-only permission, correct the program and rerun the job. Otherwise, forward a dump to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL006 - *action*, *dn* DATASET PREMATURELY TERMINATED
No end-of-data control word is present. If the end of data is being tested incorrectly, correct the program and rerun the job. Otherwise, send a dump to a Cray Research analyst. Note that this error may occur if the AB046 - DATASET SIZE LIMIT EXCEEDED message occurred when the dataset was created. From $SYSLIB. Class, 2; user abort; reprievable.

SL007 - *action*, *dn* DATASET NOT OPEN
A I/O request was issued to a dataset that was not open. Verify that an OPEN request was issued and successfully completed before the I/O request was made. If not, correct the program and resubmit the job. Otherwise, show a dump of the JTA and the DSPs to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL008 - *action*, *dn* DATASET DOES NOT EXIST
An I/O request was issued but the dataset was not found. Determine if the dataset was created before an attempt to use it. It might be necessary to recreate the dataset. From $SYSLIB. Class, 2; user abort; reprievable.

SL009 - *action*, *dn* READ AFTER WRITE
A read operation was issued after a write operation without an intervening rewind

or backspace on a sequential access dataset. Determine if the correct sequence of I/O operations is being issued. If not, correct the program and resubmit the job. Otherwise, forward a dump of the job to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL010 - *action*, *dn* READ PAST END OF DATA
A read request was issued after end-of-data was detected. Determine if end-of-data is being tested correctly. If not, correct the program and submit the job. Otherwise, forward a dump of the job to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL011 - *action*, *dn* WRITE ON READ ONLY DATASET
An attempt was made to write to a dataset for which the user has only read access. Get write access by specifying the UQ parameter on the ACCESS statement along with the write permission password, if required. If the dataset belongs to another user, it may be necessary to obtain permission from that user via a PERMIT function executed by the dataset owner. From $SYSLIB. Class, 2; user abort; reprievable.

SL012 - *action*, *dn* READ/WRITE PAST ALLOCATED AREA
A sequential write request was issued to write a block number that is too large. Determine if the block number was outside the range of the expected block numbers. If so, correct the program and reissue the request. If not, forward a dump of the job to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL013 - *action*, *dn* RANDOM WRITE RECORD MUST END ON RECORD BOUNDARY
Only complete records are processed by random I/O. Determine if the program is correctly issuing write requests. If not, correct the program and reissue the request. Otherwise, send a dump of the job to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL014 - *action*, *dn* RANDOM FILE BUFFER MUST BE AT LEAST TWO BLOCKS LONG
The minimum buffer size for a random dataset is two blocks (1024 words). Correct the specified buffer size and resubmit the request. From $SYSLIB. Class, 2; user abort; reprievable.

SL015 - *action*, *dn* CHARACTER MODE ILLEGAL ON RANDOM FILE
A write characters request for a random access dataset was detected. A character mode I/O request is generated by a formatted WRITE or READ from FORTRAN. Correct the write request or the access mode. From $SYSLIB. Class, 2; user abort; reprievable.

SL016 - *action*, *dn* WRITE PAST END OF DATA
A write request was issued after the end of data was detected. If the end of data is not being tested correctly, correct the program and rerun the job. Otherwise, forward a dump of the job to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL017 - *action*, *dn* UNCLEARED END OF FILE
An input operation attempted to read beyond the end of file or a read was issued without an intervening test for the end of file. If the end of file was not tested correctly, correct the program and rerun the job. Otherwise, forward a dump of the JTA, DSP, and buffers to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL018 - *action*, *dn* INVALID PROCESSING DIRECTION
An I/O request was issued but the processing direction did not match the processing direction specified when the dataset was opened. Correct the I/O request and resubmit the job. From $SYSLIB. Class, 2; user abort; reprievable.

SL019 - *action*, *dn* UNDEFINED I/O ERROR
I/O system error. See a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL020 - INVALID DATASET NAME OR UNIT NUMBER
One of the following conditions exists:
- The dataset name has 0 or more than 7 characters.
- The unit number is less than 0 or greater than 102.

Correct the dataset name or unit number. From $SYSLIB. Class, 2; user abort; reprievable.

SL021 - *dn* ERROR ON OPEN
The dataset cannot be opened for input or output. Make sure the program uses the proper read or write permission, and rerun. Otherwise, send a dump of the job to a Cray Research analyst. From $SYSLIB. Class, 2; user abort; reprievable.

SL021 - INVALID KEYWORD INDEX
The keyword indexed by the user in the Encryption Parameter Table is invalid. Ask the installation manager for the correct index. From SYSREF. Class, fatal.

SL022 - *action*, *dn* REQUEST INVALID FOR UNBLOCKED DATASET
An I/O request that is not supported (for example, BACKSPACE or formatted I/O) was issued while an unblocked dataset was being processed. Consult the CRAY-OS Version 1 Reference Manual, publication SR-0011, for restrictions on unblocked datasets. Correct the program and resubmit the request. From $SYSLIB. Class, 2; user abort; reprievable.

SL023 - *action*, *dn* INVALID WORD ADDRESS
A position request was issued for an unblocked dataset or the word address for the request was not a multiple of 512. Negative word addresses, except -1, are also invalid. Correct the request and resubmit the job. From $SYSLIB. Class, 2; user abort; reprievable.

SL024 - *action*, *dn* INVALID BIO FUNCTION FOR UNBLOCKED DATASET
A buffered I/O request that is not supported was issued while an unblocked dataset was being processed. Consult the CRAY-OS Version 1 Reference Manual, publication SR-0011, for restrictions on unblocked datasets. Correct the program and resubmit the job. From $SYSLIB. Class, 2; user abort; reprievable.

SL027 - BUFFER SIZE INVALID FOR UNBLOCKED REQUEST
This message is issued by ASSIGN when the undefined dataset structure (U) parameter is specified in conjunction with the buffer size (BS) parameter. Undefined dataset structure (for example, unblocked) uses a user-defined buffer. This buffer is not maintained by COS, thus the BS parameter is illegal. From $SYSLIB. Class, not reprievable; CSP abort.

SL028 - *dn* MEMORY RESIDENT INVALID FOR UNBLOCKED DATASET
This message is issued by ASSIGN when the undefined dataset structure (U) parameter is specified in conjunction with the memory resident (MR) parameter. Undefined dataset structure (for example, unblocked) uses a user-defined buffer. This buffer is not maintained by COS, thus the MR parameter is illegal. From $SYSLIB. Class, not reprievable; CSP abort.

SL029 - INVALID DEVICE TYPE
Message issued by ACCESS if device type is not mass storage or tape. Correct the value of the DT parameter on the ACCESS call or make dataset tape resident. From $SYSLIB. Class, not reprievable; CSP abort.

SL031 - ILLEGAL BFI CHARACTER
The BFI character is not in the range 000 to 177_8, nonnumeric, and is not OFF. Correct the BFI character. From CSP. Class, not reprievable; CSP abort.

SL032 - *dn* INVALID LABEL TYPE
Message issued by ACCESS if the LB parameter is not: nonlabeled (NL), IBM standard-labeled (SL), by-passed (BLP), or ANSI (AL); or the field labels of the above (FNL, FSL, and FAL). Correct the value of the LB parameter on the ACCESS call. From $SYSLIB. Class, 2; user abort; reprievable.

SL033 - *dn* PDN TOO LARGE
Message issued by ACCESS if the PDN parameter is greater than 44 characters for a tape dataset or 15 for a disk dataset. Correct the length of the permanent dataset name parameter. From $SYSLIB. Class, 2; user abort; reprievable.

SL034 - *dn* ENTER INVALID FOR TAPE
The ENTER parameter is not allowed when accessing a tape dataset. Remove the ENTER parameter. From $SYSLIB. Class, 2; user abort; reprievable.

SL036 - *dn* INVALID FILE SECTION NUMBER
Message issued by ACCESS if the FSEC parameter has been given an illegal value. Correct the value of the FSEC parameter on the ACCESS call. The value must be less than or equal to 9999. From $SYSLIB. Class, 2; user abort; reprievable.

SL037 - *dn* INVALID CHARACTER SET
Message issued by ACCESS if the CS parameter does not specify ASCII data (AS), EBCDIC data (EB), or Display Code (DC). Correct the value of the CS parameter on the ACCESS call. From $SYSLIB. Class, 2; user abort; reprievable.

SL038 - *dn* MBS TOO LARGE
Message issued by ACCESS if the MBS parameter is greater than the installation parameter value I@TMBS. Correct the value of the MBS parameter on the ACCESS call. From $SYSLIB. Class, 2; user abort; reprievable.

SL039 - *dn* INVALID FILE SEQUENCE NUMBER
Message issued by ACCESS if the FSEQ parameter has been given an illegal value. Correct the value of the FSEQ parameter on the ACCESS call. The value must be less than or equal to 9999. From $SYSLIB. Class, 2; user abort; reprievable.

SL040 - *dn* INVALID EXPIRATION DATE
Message issued by ACCESS if the Julian date is invalid for the expiration date. Correct the value of the XDT parameter on the ACCESS call. From $SYSLIB. Class, 2; user abort; reprievable.

SL041 - *dn* ALREADY OPEN. CANNOT REDEFINE DATASET
This message is issued by ASSIGN when the buffer size (BS), dataset size (S),

logical device (DV), random (RDM), or undefined dataset structure (U) parameters are specified and the dataset is already open. Dataset characteristics listed above are not changed if the dataset is already open. The dataset must be closed before these characteristics can be modified. From $SYSLIB. Class, informative.

SL042 - *dn* IS A DSN ALREADY IN USE
This message is issued by ASSIGN when the alias name (A parameter) is the same as a previously assigned dataset name (DN parameter). An alias name must be unique among previously assigned datasets. For example,

```
ASSIGN, DN = ABC
ASSIGN, DN = DEF, A = ABC
```

is a misuse of the alias parameter. To reuse the name ABC, it must first be released to the system. From $SYSLIB. Class, not reprievable; CSP abort.

SL043 - *dn* INVALID DATASET DENSITY
The density parameter for tapes (DEN) does not equal either 1600 or 6250 bpi. Correct the DEN value on the ACCESS control statement. From $SYSLIB. Class, user abort; reprievable.

SL044 - *action*, *dn* RANDOM DATASET CANNOT BE EXTENDED
An attempt was made to extend the length of a random dataset. Determine if the program is correctly issuing write requests. If it is not, correct the program. Otherwise, use WRITEDS to initialize a larger random dataset. From $SYSLIB. Class, user abort; reprievable.

SL055 - BKSP *dn* INVALID FOR TAPE DATASET
This message is issued by BACKSPACE if the dataset device type is tape. Avoid use of backspace on a tape dataset. Backward positioning other than rewind is not supported on tape datasets. From $SYSLIB. Class, user abort; reprievable.

SL056 - *dn* ILLEGAL FOREIGN DATA CONVERSION FLAG REQUEST
This message is issued by ACCESS or ASSIGN if the CV (data conversion) parameter is not equal to ON or OFF. Correct the value of the CV parameter on the ACCESS or ASSIGN statement. From $SYSLIB. Class 2, user abort; reprievable.

SL057 - *dn* ILLEGAL FOREIGN DATASET RECORD FORMAT
This message is issued by ACCESS or ASSIGN if the RF (record format) parameter is not equal to an allowable value. Correct the value of the RF parameter on the ACCESS or ASSIGN statement. Refer to the COS reference manual for valid RF parameter values. From $SYSLIB. Class 2, user abort; reprievable.

SL058 - *dn* INVALID FOREIGN DATASET RECORD SIZE
This message is issued by ACCESS or ASSIGN if the RS (record size) parameter is equated to an illegal value. Correct the value of the RS parameter on the ACCESS or ASSIGN statement. Refer to the CRAY-OS Version 1 Reference Manual, publication SR-0011, for information on RS value restrictions. From $SYSLIB. Class, 2; user abort; reprievable.

SL059 - *rt* RT ILLEGAL IF SPECIFIED WITH XDT
The RT and XDT parameters are specified in the same control statement; the message is issued by ACCESS. Correct the call to ACCESS and specify either RT or XDT. From $SYSLIB. Class, not reprievable; CSP abort.

SL061 - *fcn dn* ED PARAMETER OUT OF RANGE
The ED parameter is less than 1 or greater than 4095; *fcn* is the function name (SAVE, ACCESS, etc.) and *dn* is the dataset name. Correct the ED parameter. From $SYSLIB. Class, informative; aborts CSP unless NA parameter is used.

SL062 - *fcn dn* RT PARAMETER OUT OF RANGE
The RT parameter is less than 1 or greater than 4095; *fcn* is the function name (SAVE, ACCESS, etc.) and *dn* is the dataset name. Correct the ED parameter. From $SYSLIB. Class, informative; aborts CSP unless NA parameter is used.

SL063 - *dn* DATASET IS NOT LOCAL
The specified dataset was determined to be non-local. Use an ACCESS or ASSIGN to make the dataset local. From $SYSLIB. Class, 2; user abort; reprievable.

SL064 - *modname* PREMATURE END OF PROGRAM MODULE
An end of file was encountered while the PDT of a program module was being read. Recreate the module. From $SYSLIB. Class, 2; user abort; reprievable.

SL065 - *pgm* INVALID PROGRAM LOAD TABLE
The program load table was not found or was in error. Recreate the program module. From $SYSLIB. Class, 2; user abort; reprievable.

SL066 - *pgm* FATAL PROGRAM COMPILATION ERRORS
Fatal program errors were encountered during compilation or assembly. Correct the program and reassemble or recompile. From $SYSLIB. Class, 2; user abort; reprievable.

SL067 - *pgm* PROGRAM IS NOT AN ABSOLUTE MODULE
The absolute program module flag was not set in the PDT. Recreate the absolute program module. From $SYSLIB. Class, 2; user abort; reprievable.

SL068 - *pgm* ILLEGAL PROGRAM LOAD ADDRESS
The load address in the PDT does not indicate that the program is to be loaded at the end of the JCB. Recreate the absolute program module. From $SYSLIB. Class, 2; user abort; reprievable.

SL069 - *dn* FOREIGN DATASET TRANSLATION REQUEST NOT SUPPORTED
This message is issued by ACCESS or ASSIGN if the FD (foreign dataset translation identifier) is not equal to a supported file type. Correct the value of the FD parameter on the ACCESS or ASSIGN statement. Refer to the CRAY-OS Version 1 Reference Manual, publication SR-0011, for supported FD parameter values. From $SYSLIB. Class 2; user abort; reprievable.

SL070 - *dn* INVALID FOREIGN DATASET BLOCK SIZE
This message is issued by ACCESS or ASSIGN if the MBS (maximum block size) parameter is equated to an illegal value. Correct the value of the MBS parameter on the ACCESS or ASSIGN statement. Refer to the CRAY-OS Version 1 Reference Manual, publication SR-0011, for information on MBS value restrictions. From $SYSLIB. Class 2; user abort; reprievable.

SL073 - NEITHER ADN NOR USER PARAMETER SPECIFIED
Either ADN or USER must appear on the PERMIT statement. Add the parameter and resubmit. From $SYSLIB. Class, fatal.

SL074 - PROCESSOR NUMBER OUT OF RANGE
The CPU number specified on an OPTION,PN call is not valid for the machine on which the job is being executed. From $SYSLIB. Class, 2; user abort; reprievable.

SL075 - UNKNOWN PERFORMANCE MONITOR FUNCTION
The ASCII string passed to subroutine PERF was not found in the list mapping such strings into the corresponding F$PERF subfunctions. From $SYSLIB. Class, 2; user abort; reprievable.

SL077 - INVALID RING PARAMETER - USE IN OR OUT
ACCESS issues this message if something other than 'IN' or 'OUT' is specified on the 'RING=' parameter of the ACCESS control statement. The only valid options are 'IN' and 'OUT'. Refer to the CRAY-OS Version I Reference Manual, SR-0011, for information on the RING parameter. From $SYSLIB. Class, fatal.

SL078 - RING=OUT ILLEGAL WITH NEW OR MOD
ACCESS issues this message if there is a conflict between the NEW or MOD parameter and the user's RING request. NEW and MOD assume RING=IN for writing on an online tape. From $SYSLIB. Class, 2; user abort; reprievable.

SL079 - 'BS' PARAMETER OUT OF RANGE
This message is issued by ASSIGN whenever the BS parameter is greater than or equal to 40000. Lower BS and rerun the job. From $SYSLIB. Class, 2; user abort; reprievable.

SP001 - INPUT DATASET NOT SPECIFIED
The I parameter on the SPAWN control statement was illegally equated to 0. Resubmit the job with I set to a valid dataset name or defaulted to $CPL. From SPAWN. Class, fatal.

SP002 - INPUT DATASET DOES NOT EXIST
The dataset specified by the I parameter on the SPAWN control statement could not be found. Change the specification of the I parameter on the SPAWN control statement or access the dataset and resubmit the job. From SPAWN. Class, fatal.

SP003 - BAD SL PARAMETER VALUE
The specification of the SL (sleep time) parameter on the SPAWN control statement contains non-numeric characters. Correct the SL parameter and resubmit the job. From SPAWN. Class, fatal.

SP004 - BAD DW PARAMETER VALUE
The data width (DW) parameter is calling for less than one character. The DW parameter might contain a non-numeric character. Correct the DW parameter and resubmit the job. From SPAWN. Class, fatal.

SP005 - BAD GL PARAMETER VALUE
The GL (group size) parameter on the SPAWN control statement contains a non-numeric character. Correct the GL parameter and resubmit the job. From SPAWN. Class, fatal.

SP006 - JOB DELIMITER NOT SPECIFIED
The SP parameter on the SPAWN control statement was illegally set to 0. Respecify the SP parameter to match the text on job separator statements on the input dataset. A minimum of eight characters of text are significant and required for this separator. From SPAWN. Class, fatal.

SP200 - INPUT DATASET EMPTY OR MISPOSITIONED
The first attempt to read from the input dataset resulted in EOF or EOD status. Rebuild or rewind the input dataset as appropriate and resubmit the job. From SPAWN. Class, fatal.

SP201 - INPUT DATASET MISPOSITIONED OR STRUCTURE ERROR
The first card image read from the input dataset was not a job separator card as was expected. Verify that the first card image in the dataset is a job separator card. Reposition or rewind the input dataset and resubmit the job. From SPAWN. Class, fatal.

SP202 - INPUT DATASET STRUCTURE ERROR
Read status of EOF or EOD was returned on the next read after encountering the job separator card in the input dataset. Rebuild the dataset, ensuring that a job separator card is not the last card image in it. Resubmit the job. From SPAWN. Class, fatal.

SP203 - INPUT DATASET STRUCTURE ERROR
Two successive job separator cards were encountered in the input dataset. Correct the input dataset, either by inserting job control text between the adjacent separator cards or by removing one of the cards. Resubmit the job. From SPAWN. Class, fatal.

SP800 - *n* JOBS DISPOSED TO THE *dn* QUEUE
This message indicates completion of the SPAWN execution. From SPAWN. Class, informative.

SP801 - SPAWN COMPLETE
This message indicates completion of the SPAWN execution. From SPAWN. Class, informative.

SP901 - DISPOSE TO SPECIFIED QUEUE UNSUCCESSFUL
Internal error occurred in processing the DISPOSE request. Dumps should be taken and analyzed. From SPAWN. Class, fatal.

SR001 - SYSREF, VERSION *x.xx*
Start of job step. From SYSREF. Class, informative.

SR002 - *nn* MODULES, *nnn* SYMBOLS, *nnnn* REFERENCES
End of job step. From SYSREF. Class, informative.

SR010 - BINARY SYMBOL FILE MISSING OR MISNAMED
The dataset specified by the X parameter ($XRF by default) does not exist. Check the SYSREF control statement for coding errors. If necessary, add an X parameter to one or more preceding CAL or APML control statements. From SYSREF. Class, fatal.

SR100 - READ_STATUS = *n*; RECP = *nn*; SYMBOL RECP.WORD_COUNT = *n*; TOTAL LENGTH = *nnn*; ACTUAL_LENGTH - *nnn*;
One of the four fatal messages SR101, SR102, SR103, or SR104 has just been issued. If any of these errors occur repeatedly, there is a program logic error in SYSREF, CAL, or APML. Use DSDUMP to dump the X dataset that was input to SYSREF. Give the listing and the job's logfile to a Cray Research analyst. From SYSREF. Class, fatal.

SR101 - PREMATURE END-OF-RECORD WHILE READING HEADER
SR102 - PREMATURE END-OF-RECORD WHILE READING SYMBOL ENTRIES
SR103 - RECORD LONGER THAN INDICATED BY HEADER
SR104 - SYMBOL COUNT WRONG OR SYMBOL ENTRIES MALFORMED
While SYSREF was reading the X dataset, either the dataset was found to be bad or a program logic error occurred. Use DSDUMP to dump the X dataset that was input to SYSREF. Give the listing and the job's logfile to a Cray Research analyst. From SYSREF. Class, fatal.

SR150 - *m* SYMBOL ENTRIES WERE READ BUT *n* WERE OUTPUT
The number of symbols output was different from the number read from the X dataset. If too few symbols were output, SYSREF's sort routine is not maintaining the linked list correctly. Give a copy of the X dataset and the listing of the job's logfile to a Cray Research analyst. From SYSREF. Class, warning.

SR200 - DUPLICATE PROGRAM NAME *pn* IN RECORDS *m* AND *n*
A binary symbol record from the X dataset has the same module name as an earlier binary symbol record on the same dataset. Record numbers such as *m* and *n* start at 1. Check to see if the routine in question is being assembled twice or if there is a name conflict between two different source files being assembled. From SYSREF. Class, warning.

SS001 - ACQUIRE FROM MF=*mfid* COMPLETE PDN=*pdn* ID=*id* ED=*ed*
Dataset successfully acquired from front-end mainframe. From SCP. Class, informative. System log only.

SS002 - ACQUIRE CANCELLED BY COS PDN=*pdn* ID=*id* ED=*ed*
Dataset acquire cancelled by COS. This can happen when DQM or PDM detects an error or when a hardware error occurs. From SCP. Class, informative. System log only.

SS003 - DATASET NOT AVAILABLE FROM FRONT END
Dataset acquire cancelled by the front-end mainframe. From SCP. Class, informative.

SS004 - DATASET RECEIVED FROM FRONT END
Dataset transferred successfully from front-end. From SCP. Class, informative.

SS005 - ACQUIRE CANCELLED BY MF=*mf* PDN=*pdn* ID=*id* ED=*ed*
Dataset acquire cancelled by a front-end mainframe. From SCP. Class, informative. System log only.

SS006 - DISPOSE CANCELLED BY COS MF=*mf* SDN=*sdn*
Disposed dataset cancelled by COS. This can occur if DQM or PDM detects an error or if a hardware error occurs. From SCP. Class, informative. System log only.

SS007 - DISPOSE CANCELLED BY MF=*mf* SDN=*sdn*
Dataset dispose cancelled by the front-end mainframe. From SCP. Class, informative. System log only.

SS008 - DATASET TRANSMITTED TO MF=*mf* JN=*jn* DN=*dn*
Dataset transmitted to a front-end mainframe successfully. From SCP. Class, informative System log only.

SS012 - DQM ERROR CODE=*nnn* DATASET=*xxx*
Message issued when DQM returns an error response to a request from SCP. Look up the meaning of the returned code from DQM in the COS EXEC/STP/CSP Internal Reference Manual, publication SM-0040, or contact a site analyst. If excessive unrecovered hardware errors occur, notify the site engineer. If DAT or disk space is exhausted, notify the system analyst. From SCP. Class, informative.

SS013 - PDM ERROR CODE=*nnn* DATASET=*xxx*
Message issued when PDM returns an error response to a request from SCP. This is caused by:

- An error in saving the dataset
- An error deleting a dataset

Look up the meaning of the returned code from PDM in the COS EXEC/STP/CSP Internal Reference Manual, publication SM-0040, or contact a site analyst. If DSC, DAT, or DNT is full, notify system analyst. From SCP. Class, informative.

SS200 - DQM ERROR CODE=*nnn* DATASET=*xxx*
Message issued when DQM returns an error response to a request from STG. Look up the meaning of the returned code from PDM in the COS Internal Reference Manual, CRI publication SM-0040, or contact a site analyst. If excessive unrecovered hardware errors occur, notify the site engineer. If DAT or disk space is exhausted, notify the system analyst. From STG. Class, informative.

SS201 - PDM ERROR CODE=*nnn* DATASET=*xxx*
Message issued when PDM returns an error response to a request from STG. This is caused by:

- An error in saving the dataset
- An error deleting a dataset

Look up the meaning of the returned code from PDM in the COS EXEC/STP/CSP Internal Reference Manual, publication SM-0040, or contact a site analyst. If the DSC, DAT, or DNT is full, notify system analyst. From STG. Class, informative.

ST200 - PRIVILEGE IS REQUIRED FOR F$ FUNCTION CODES
The user is not privileged for a requested system call. Ask the installation manager for the required privilege. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST201 - PRIVILEGE IS REQUIRED FOR PDM FUNCTION *f*
The user is not privileged for a requested PDM call. The *f* function codes and required privileges are as follows:

f	Priv.	PDM function
12	SCSPOL	Save input
14	SCSPOL	Save output
26	SCSPOL	Access spooled
36	SCSPOL	Delete spooled
40	SCRDSC	Page request
50	SCLUSR	Load user
52	SCSPOL	Load input
54	SCSPOL	Load output
60	SCTASK	PDS/release
100	SCDTIM	Dump time
110	SCQSDT	Dequeue SDT
120	SCQSDT	Queue SDT available
122	SCQSDT	Queue SDT input
124	SCQSDT	Queue SDT output
150	SCQSDT	Rewrite SDT
160	SCTASK	Pseudo access
170	SCUPDD	PDSDUMP user
176	SCSPOL	PDSDUMP spooled
210	SCQDXT	Link DXT
220	SCQDXT	Modify DXT
230	SCRDSC	Get DXT
240	SCACES	Access user DS

Ask the installation manager for the required privilege. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST202 - PRIVILEGE IS REQUIRED FOR ENTER SDR REQUEST
The user is not privileged for a requested ENTER SDR call. Ask the installation manager for the required privilege. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST203 - UNDEFINED F$PRV SUBFUNCTION CODE *n*
The user requested an illegal subfunction on a F$PRV call. Resubmit the job with a legal subfunction. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST204 - PRIVILEGE IS REQUIRED FOR F$PRV SUBFUNCTION, PRV$SDR
The user is not privileged for a requested ENTER SDR call. Ask the

installation manager for the required privilege. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST205 - PRIVILEGE IS REQUIRED FOR F$PRV SUBFUNCTION, PRV$SPF
The user is not privileged for a requested SET GLOBAL PRIVILEGE FLAG call. Ask the installation manager for the required privilege. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST206 - PRIVILEGE IS REQUIRED FOR F$PRV SUBFUNCTION, PRV$SWP
The user is not privileged for a requested SWAP SYSTEM VALUES call. Ask the installation manager for the required privilege. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST207 - SECURITY VIOLATION COUNT EXCEEDED. USER DISABLED.
The user exceeded allowed security violations. Ask the installation manager to renew the violation count. From EXP. Class, warning in Warn mode, fatal in Abort mode.

ST208 - PRIVILEGE IS REQUIRED FOR S ON JOB STATEMENT
The user is not privileged for use of S parameter on the JOB statement. Ask the installation manager for the required privilege. From EXP. Class, Warning in warn mode; fatal in Abort mode.

SU001 - NS PARAMETER NON-NUMERIC
The NS parameter on the control statement contains a non-numeric character. Correct the NS value. From SKIPU. Class, fatal; job aborts.

SX001 - ZERO INCREMENT AT LINE *m*
An attempt was made to execute a FOR statement whose increment value in the BY clause resolved to 0. *m* indicates the location of the FOR statement in the SKOL source listing. Correct the SKOL program's logic. From SKOL run time support routines. Class, fatal.

SX002 - FINAL VALUE NOT EXACT AT LINE *m*
An attempt was made to execute a FOR statement in which the difference between the final value in the TO clause and the initial value was not an exact multiple of the increment value in the BY clause. *m* indicates the location of the FOR statement in the SKOL source listing. Correct the SKOL program's logic. From SKOL run time support routines. Class, fatal.

SX003 - NO SITUATION WAS SIGNALLED AT LINE *m*
The end of an UNTIL block was reached before a situation statement was executed. *m* refers to either the THEN statement in a multiple situation case structure or the ENDUNTIL statement in a simple situation case structure. Add either a situation statement or a looping structure inside the UNTIL block. From SKOL run time support routines. Class, fatal.

SX004 - VALUE OUT OF RANGE AT LINE *m*
An attempt was made to execute a WHEN statement whose selector expression resolved to a value outside the range of the selector type. *m* indicates the location of the WHEN statement in the SKOL source listing. Change the selector expression or the selector type or define a new type. From SKOL run time support routines. Class, fatal.

SX005 - RECORD SPACE EXHAUSTED AT LINE *m*
The value of rtype-AVAILP was 0 when the NEW statement attempted to allocate a record. *m* indicates the location of the NEW statement in the SKOL source listing. Execute MAKEAVAIL for the appropriate record type before executing the first NEW statement. Provide enough FREE statements to ensure that all unneeded records are released promptly. Enlarge the maximum number of records by changing the RECORD declaration. As a last resort, output the error message and append (NOVALIDATE) to the NEW statement. See the SKOL Reference Manual, CRI publication SR-0033, for an example. From SKOL. Class, fatal.

SX006 - ILLEGAL COROUTINE FINISH AT LINE *m*
The end of a coroutine was reached before an ACTIVATE or SUSPEND statement was executed. *m* indicates the location of the ENDCOROUTINE statement in the SKOL source listing. See the SKOL Reference Manual, CRI publication SR-0033, for correct usage. From SKOL. Class, fatal.

SX007 - RECURSION UNDERFLOW AT LINE *m*
An attempt was made to access the recursion stack with a zero subscript. This indicates illegal nesting of recursive routines. *m* indicates the SKOL source line containing the declaration of the size of the recursion stack. Define all recursive routines at level 1. From SKOL. Class, fatal.

SX008 - RECURSION OVERFLOW AT LINE *m*
The recursion stack has insufficient room to hold the parameters, local variable, and two control variables needed for invoking the current recursive routine. If *m* refers to an EXECUTE statement, overflow occurred while trying to add parameters to the stack. If *m* refers to a ROUTINE statement, overflow occurred while trying to add LOCAL variables to the stack. Enlarge the recursion stack by modifying the first value specified in parentheses in the RECURSIVE declaration. For each potential level of

recursion, allow 2+P+L elements in the stack, where P is the number of parameters and L is the number of local variables. If two or more recursive routines are active simultaneously, their effect on the stack size is additive. From SKOL. Class, fatal.

SX009 - STRING OVERFLOW AT LINE *m*
The length of the result string exceeds the maximum length specified in the STRING declaration. *m* indicates the location of the invalid statement in the SKOL source listing. Increase the dimension specified in the STRING declaration. From SKOL. Class, fatal.

SX010 - COLLISION IN CHARACTER MAPPING AT *n*; DISCOVERED AT LINE *m*
A CHAR_SETUP statement detected a duplicated character in the program's TYPE CHAR statement. *n* is one greater than the ASCII value of the character. *m* indicates the location of the CHAR_SETUP statement in the SKOL source listing. If the TYPE CHAR statement specifies the same character twice, remove one of the instances. One of the duplicate specifications might be in a default subtype list; see the SKOL Reference Manual, CRI publication SR-0033, for a complete list of all the default members of the CHAR type. From SKOL. Class, fatal.

SX011 - HOLE IN OUTPUT CHARACTER TABLE AT *n*; DISCOVERED AT LINE *m*
A WRITESTRING statement's string argument contains a character whose internal (ordinal) value is defined with an identifier rather than with a single-character literal. *n* is the internal value of the character, corresponding to the sequential position of the character in the TYPE CHAR declaration. *m* indicates the location of the invalid statement in the SKOL source listing. Correct the SKOL program's logic so that any string to be output by WRITESTRING consists entirely of the ordinal values of characters with corresponding external values. From SKOL. Class, fatal.

SX012 - UNDEFINED CHARACTER VALUE *n* OUTPUT AT LINE *m*
A WRITESTRING statement's string argument contains a value less than 1 or greater than the number of characters declared in the TYPE CHAR statement. *n* is the invalid value from the string. *m* indicates the location of the WRITESTRING statements in the SKOL source listing. Any string to be output by WRITESTRING must consist entirely of ordinal values in the range [1...N], where N is the number of declared characters. From SKOL. Class, fatal.

SX013 - UNDEFINED CHARACTER '*c*' READ AT LINE *m*
A READSTRING statement encountered an undeclared character in the input. *m* indicates the location of the READSTRING statement in the SKOL source listing. Add all expected characters (including the character *c*) to the TYPE CHAR declaration or delete the TYPE CHAR, CHAR COMMON, and CHAR_SETUP statements; change all READSTRING and WRITESTRING statements to READ, WRITE, INPUT, and OUTPUT statements; and avoid all reference to the ORD, CHR, and VALUE functions and to the built-in CHAR subtypes. From SKOL. Class, fatal.

SY001 - RLS COULD NOT FIND A DNT FOR *dn*
An attempt was made to release a dataset which did not exist. From EXP. Class, informative.

SY002 - ATTEMPT TO DISPOSE AN EMPTY DATASET *dn*
Attempt to dispose an empty dataset. No dispose was done. If the dataset should not be empty, check the job to ensure that the dataset is created properly. From EXP. Class, informative.

SY003 - USER OPTION VALUE TRUNCATED
An OPTION value is too large for the corresponding field. Check the OPTION call to ensure that values are within bounds. From EXP. Class, informative.

SY005 - *dn ldv wd*W *req*RR *bl*SR *req*WR *bl*SW *sec*SC *dn ldv wd*W *req*RR *bl*SR *req*WR *bl*SW *sec*SC *etc.*
The user specified OPTION,STAT=ON or the site has I@STAT set as nonzero, to enable per-device accounting. This message prints when the dataset is released, defining the following information:

dn	Dataset name
ldv	Logical device on which the dataset resides
*wd*W	Dataset size in words
*req*RR	Number of read requests for the dataset
*bl*SR	Number of 512-word sectors transferred
*req*WR	Number of write requests
*bl*SW	Number of sectors written
*sec*SC	Time spent blocked for I/O in seconds

From EXP. Class, informative.

SY006 - *binary instructions system-hang macro*
The message format is in the STP table area and is normally put out at system recovery time. This message signals the beginning of the hang message which STP creates when it detects an internal error and intends to crash. The binary instructions are the last few parcels before the hang macro, and the hang macro is of the form ERR*xx* where *xx* is AN,

SN, AZ, SZ, etc. This pattern and the accompanying hang address can be used to find the cause of the crash. From common subroutine ERROR. Class, informative; system log only.

SY013 - SYSTEM *starttype* LEVEL *mm/dd/yy hh:mm:ss*
This message appears every time the system starts. The *starttype* can be INSTALL, DEADSTRT, or RESTART, depending on which Startup option was performed. From STARTUP. Class, informative; system log only.

SY014 - $SYSLOG RECOVERY SUCCESS CODE *x*
This recovery message appears after every restart, deadstart, and install. Either the system log is recovered at the proper point in the existing edition or a new edition is initialized. *x* can be one of the following:

OK The existing edition of the system log has been recovered successfully.
1 The system log does not exist, is on a down device, or has an AI conflict. A new edition is initialized.
2 The verify word is bad. The Dataset Parameter Table is rebuilt, and the system log is read to EOF.
3 The edition number does not match. A new edition is begun.
4 DSP failed validation. Rebuild and read to EOF.
5 DPEOR is not set. Rebuild the DSP and read to EOF.
6 Bad Record Control Word in log. Rebuild the DSP and read to EOF.
7 Bad Forward Word Index in RCW. Rebuild the DSP and read to EOF.
8 Bad block number in BCW. Rebuild the DSP and read to EOF.
9 The $SYSTEMLOG buffer was not recovered. The DSP is rebuilt, and the system log is read to EOF.
10*nnnn* The system log is read to EOF. The block with the EOF RCW is not the same block pointed to in the system log buffer. The recovered buffer is discarded and writing is resumed in the block which had the EOF. *nnnn* = Number of blocks lost.
11 Error on system log read to EOF. A new edition is created.

From MSG. Class, informative; system log only.

SY015 - $SYSTEMLOG SAVED AS EDITION *x*
This message appears whenever a new edition of $SYSTEMLOG is saved. From MSG. Class, informative; system log only.

SY031 - BLOCKED FOR MEMORY *log*
Space is not currently available in the memory pool for building a record for the system log (MS) or for a user's $LOG (MU). However, the MSG task will continue its processing of messages: first it transfers earlier records from the memory pool to the appropriate logs; then it attempts to rebuild the current blocked message and to build any subsequent message requests. From MSG. Class, informative; system log only.

SYDPS1 - WAIT AND NOWAIT ILLEGAL TOGETHER - DEFAULT USED
A DISPOSE request specified both WAIT and NOWAIT. The system issues the warning message and then uses the installation default. From $SYSLIB. Class, not reprievable.

SYDRV1 - UNKNOWN STATUS RETURNED
The system returned a bad status on a DRIVER call. See your site analyst. From $SYSLIB. Class, reprievable.

SYDRV2 - INCORRECT NUMBER OF PARAMETERS
User specified an incorrect number of parameters on a DRIVER call. Fix code to reflect the correct amount. From $SYSLIB. Class, reprievable.

SYDRV3 - INCORRECT NUMBER OF PARAMETERS
The user specified an incorrect number of parameters on a DRIVST call. Fix code to reflect the correct number of parameters. From $SYSLIB. Class, reprievable.

SYDRV4 - BAD DRIVER STATUS
User sent DRIVST a bad status. Check the program to see if status is wrong; otherwise see your site analyst. From $SYSLIB. Class, reprievable.

SYIJC1 - INCORRECT NUMBER OF ARGUMENTS PASSED
IJCOM detected that the user did not pass the correct number of arguments. Consult the Library Reference Manual, CRI publication SR-0014 for parameters. Correct the code and rerun. From $SYSLIB. Class, fatal.

SYIJC2 - PARAMETER BLOCK NOT CORRECT LENGTH
The parameter block array length was not equal to I@MPBS. Correct the code and rerun. From $SYSLIB. Class, fatal.

SYIJC3 - ARRAY SIZE NOT LARGE ENOUGH
The 'larray' parameter must be greater than or equal to 'lentry' * 'nentry'. See the Library Reference Manual, CRI publication SR-0014 for details. Correct the code and rerun. From $SYSLIB. Class, fatal.

SYIJC4 - UNKNOWN STATUS RETURNED
The system returned an illegal status. Consult the site analyst for recommended corrections. This error indicates a system problem, not a user problem. From $SYSLIB. Class, fatal.

SYEREC1 - UNKNOWN STATUS RETURNED
An unknown status was returned to $SYSLIB from an ERECALL request. See site analyst. From $SYSLIB. Class, reprievable.

SYEREC2 - *lentry* IS NEGATIVE
A negative L-entry was specified in an ERECALL. Correct L-entry. From $SYSLIB. Class, reprievable.

SYMEM1 - BAD CODE
An invalid code was specified in a CFT memory call. Correct the code. From $SYSLIB. Class, reprievable.

SYMEM2 - CODE MUST BE SPECIFIED
No code was specified in a CFT memory call. Specify a code. From $SYSLIB. Class, reprievable.

SYMEM3 - VALUE MUST BE SPECIFIED
No value was specified in a CFT memory call. Specify a value. From $SYSLIB. Class, reprievable.

SYMFY1 -INVALID EQUATE GIVEN FOR THE EXO PARAMETER
The EXO parameter in a MODIFY request contains a value that is neither ON nor OFF. The EXO parameter is ignored. Correct the value. From $SYSLIB. Class, not reprievable.

SYSAV1 - INVALID VALUE GIVEN FOR THE EXO PARAMETER
The EXO parameter in a SAVE request contains a value that is neither ON nor OFF. The EXO parameter is ignored. Correct the value. From $SYSLIB. Class, not reprievable.

TB001 - BEGINNING OF TRACEBACK
TRACEBACK processing has begun. Traceback is called during user aborts or can be called directly by the user. From $SYSLIB. Class, informative.

TB002 - END OF TRACEBACK
TRACEBACK has terminated normally. From $SYSLIB. Class, informative.

TB003 - TRACEBACK PREMATURELY TERMINATED
TRACEBACK has detected an error while trying to trace the user calls from the present location to the main level. This is usually caused by invalid B register contents. From $SYSLIB. Class, fatal.

TE000 - TEDI VERSION *x mm/dd/yy*
The TEDI program has begun processing, using the specified version and date. From TEDI. Class, informative.

TE001 - INVALID TEDI COMMAND ARGUMENT
A user-specified command argument was invalid for one of the following reasons:

- The line numbers are not within the legal range ($m > 0$ and $n \geq 0$).
- The column number references are not contained within the legal range (1-150).
- The argument list contained a special character unknown to TEDI.

Correct the invalid argument. From TEDI. Class, illegal.

TE002 - INCORRECT TEDI COMMAND MNEMONIC
The command's mnemonic is syntactically incorrect. Correct the bad mnemonic. From TEDI. Class, illegal.

TE003 - MISSING TEDI COMMAND ARGUMENTS
The TEDI command found a required argument or arguments missing. Complete the argument list. Refer to CRI publication SG-0055, Text Editor (TEDI) User's Guide. From TEDI. Class, illegal.

TE004 - DATA FIELD NOT FOUND
The replace data command could not find the specified data field. Correct the data field request. From TEDI. Class, illegal.

TE005 - U COMMAND NOT NESTABLE
A U command was executed in another U dataset or in an iterate command. Remove the U command from the iteration command or the U dataset. From TEDI. Class, illegal.

TE006 - UNKNOWN TEDI COMMAND
The user specified a command unknown to TEDI. Use correct commands. Refer to CRI publication SG-0055, Text Editor (TEDI) User's Guide. From TEDI. Class, illegal.

TE007 - INVALID TAB STOP REQUEST
The tab stop is out of the range (-150,+150). Request a tab stop in the legal range. From TEDI. Class, illegal.

TE008 - INVALID INITIAL INDENTATION LEVEL
Initial indentation level request is not in the range [1,150]. Request an indentation level in the range [1,150]. From TEDI. Class, illegal.

TE009 - MISSING INDENTATION LEVEL INCREMENT
Set indentation command was missing the indentation level increment. Specify the indentation level increment in the SI command. From TEDI. Class, illegal.

TE010 - REGISTER SYMBOL INVALID
Register symbol was not contained in the range [0,9]. Use valid register symbol. From TEDI. Class, illegal.

TE011 - WRITING OVER A PERMANENT DATASET
A permanent dataset will be overwritten if the command is continued. An option to continue is given. From TEDI. Class, warning.

TE012 - NO MAINTENANCE PERMISSION
The specified dataset does not have maintenance permission. If maintenance permission is desired, reaccess the dataset with required control words. From TEDI. Class, warning.

TE013 - PATTERN OR SYMBOL NOT FOUND
TEDI did not find specified pattern or symbol. From TEDI. Class, informative.

TE014 - DATASET NOT LOCAL
A TEDI command requested access to a dataset that is not local to the job. Create the dataset or recheck the dataset name. From TEDI. Class, informative.

TE015 - SWITCHES RECOVERED
TEDI's run-time switches are recovered from the previous editing session. From TEDI. Class, informative.

TE016 - ILLEGAL DATASET NAME
The dataset name contains an invalid character or is longer than seven characters. Correct the dataset name. From TEDI. Class, illegal.

TE017 - NEW DATASET
A local dataset has been created by TEDI. From TEDI. Class, informative.

TE018 - NO UPDATE WILL BE DONE
Since the dataset does not have maintenance permission, no update can be done. The current dataset can be written to a local dataset, which TEDI can create by using the write dataset commands. Otherwise the current dataset must be reaccessed with the required permission. From TEDI. Class, informative.

TE019 - READ ONLY DATASET
The dataset does not have maintenance permission and unique access. In order to write on the dataset, reaccess it with the correct maintenance control word and unique access. From TEDI. Class, informative.

TE020 - APPROACHING MAXIMUM DATASET SIZE
The user is approaching the maximum dataset size that TEDI can handle. Rewrite the dataset. From TEDI. Class, warning.

TE021 - DATASET TOO LARGE
The dataset is too large for TEDI to handle. Consult a Cray Research analyst. From TEDI. Class, fatal.

TE022 - BAD META-STRING CONSTRUCT
TEDI found an illegal meta-string construct. Correct meta-string request. From TEDI. Class, illegal.

TE023 - TOO MANY TEDI COMMAND ARGUMENTS
Too many command arguments were found. Correct argument list. From TEDI. Class, illegal.

TE024 - CANNOT EDIT AN INTERACTIVE DATASET
User was trying to edit an interactive dataset. Edit only non interactive datasets. From TEDI. Class, fatal.

TE025 - UNKNOWN SYSTEM COMMAND
The user specified an unknown system command. See system commands in CRI publication SG-0055, Text Editor (TEDI) User's Guide. From TEDI. Class, illegal.

TE026 - *n* NULL INPUT RECORDS SKIPPED
n null records were encountered on input and subsequently skipped. From TEDI. Class, informative.

TE027 - *n* LONG INPUT RECORDS TRUNCATED
n records containing over 150 characters were truncated. The maximum record length for TEDI is 150 characters. From TEDI. Class, informative.

TE028 - TRAILING BLANKS IN INPUT RECORD TRUNCATED
When TEDI is used as a batch job, the input record does not have the delimiter character in the end of the line. Put the delimiter at the end of the input line if trailing blanks are desired. From TEDI. Class, warning.

TE999 - *TEDI routine name*
A serious TEDI execution error occurred. See a Cray Research analyst. From TEDI. Class, fatal.

TQ002 - DN=*dn*, *stat*, VSN=*vsn*(*lt*), VBC=*vbc*, LASTI/O=*op*
A significant point has been reached, as specified by the following variables:

dn	Local dataset name
stat	Dataset status as follows:
	BOV Beginning of volume
	EOV End of volume
	EOD End of data
	CLOSE Dataset is closed
	RLS Dataset is released
	REWIND Dataset is rewound
vsn	Volume serial number
lt	Label type
vbc	Volume block count
op	Last I/O operation (read or write)

From TQM. Class, informative.

TQ003 - *dn*(*vsn*) LABEL TYPES DIFFER - HEADER=*lt* TRAILER=*alt*
The requested label type differs from the actual label type. This message is associated with AB118 TAPE LABEL GROUP CORRUPTED SEE TQM MESSAGE, which indicates an abort condition. The following information is given:

dn	Local dataset name
vsn	Volume serial number
lt	Requested label type
alt	Actual label type

From TQM. Class, informative.

TQ005 - *dn* ASSIGNED TO *dvn*
TQ005 - *dn* REASSIGNED TO *dvn*
The dataset is assigned to the specified tape device. The following information is given:
dn Local dataset name
dvn Device name
From TQM. Class, informative

TQ007 - PERMANENT WRITE ERROR, EOV FORCED
Unrecoverable write error on tape. COS attempts to close the current volume and requests another volume so that writing can continue. From TQM. Class, informative.

TQ008 - *dn*(*vsn*) UNABLE TO WRITE THE *lt* LABELS (TDSTS=*stat*)
This message is associated with any of several abort conditions. The following information is given:
dn Local dataset name
vsn Volume serial number
lt Header or trailer
stat Tape device status indicating why label did not print
From TQM. Class, informative.

TQ017 - *dn*(*vol*) DATASET NOT ON VOLUME
The dataset *dn* is not on the tape volume *vol*. From TQM. Class, fatal.

TQ018 - *dn*(*vol*) EOF NOT ON VOLUME
When the parameter MOD is specified on the ACCESS statement tape, volume *vol* does not contain the end of file label of the dataset *dn*. From TQM. Class, fatal.

TQ019 - *dn*(*vol*) FILE ID ON HEADER AND TRAILER LABELS MISMATCH
When processing a MOD dataset *dn*, the file ID of the file on the header and trailer labels is not the same. From TQM. Class, fatal.

TQ020 - *dn*(*vol*) BLOCK COUNT ERROR
The block count field of the trailer label of the dataset *dn* does not agree with the block count of the file on the tape *vol*. From TQM. Class, fatal.

TQ021 - *dn*(*vol*) SEARCH TAPE MARK ERROR, TT, TMC=*c*(DEC),TDSTS=*st*
An error occurred while spacing tape marks.
dn Dataset name
vol Volume serial number
TMC Residual tape mark count in decimal
TDSTS Status returned from the IOP. See field TDSTS of the TDT table.
From TQM. Class, fatal.

TQ022 - *dn*(*vol*) ERROR IN TRAILER LABEL, ERROR CODE=*e*(OCT)
An error occurred while processing the trailer label of dataset *dn* on volume *vol*. The error code (octal) is *e*.
From TQM. Class, fatal.

TQ023 - *dn*(??????) NO VSN FOR MOD TAPE DATASET, WILL PROCESS AS NEW DATASET
No volume serial number is specified for a MOD dataset. The dataset will be treated as a new dataset. From TQM. Class, fatal.

TQ024 - *dn*(*vol*) BAD FILE SEQUENCE NUMBER
The file sequence number of dataset *dn* on volume *vol* is incorrect. From TQM. Class, fatal.

TQ025 - *dn*(*vol*) BAD GENERATION NUMBER
The generation number of dataset *dn* on volume *vol* is incorrect. From TQM. Class, fatal.

TQ026 - *dn*(*vol*) CANNOT POSITION FORWARD AFTER WRITE
A tape dataset cannot be positioned forward immediately after being written to. From TQM. Class, fatal.

TQ027 - *dn*(*vol*) CANNOT POSITION NEW TAPE FORWARD
A new tape dataset cannot be positioned forward. From TQM. Class, fatal.

TQ028 - *dn*(*vol*) TAPE DEVICE DOWN
The tape device that the dataset *dn* is on is down. From TQM. Class, fatal.

TQ029 - *dn*(*vol*) POSITION: TAPE VOLUME *vol* CANNOT BE MOUNTED
The volume *vol* requested by a POSITION request cannot be mounted because it is not in the volume serial number list. From TQM. Class, fatal.

TQ030 - *dn*(*vol*) POSITION: TAPE VOLUME *sign vol* CANNOT BE MOUNTED
The volume requested by a POSITION request cannot be mounted. If *sign* is positive or negative, then the request is a relative volume positioning request and *n* is the number of volumes to skip. If *sign* is blank, then *n* is the absolute volume position in the volume serial list. From TQM. Class, fatal.

TQ031 - *dn*(*vol*) POSITION: BLOCK NUMBER *bn* INVALID FOR VOLUME POSITIONING
The block number *bn* is invalid for volume positioning. Position by absolute block number must be used with volume positioning. From TQM. Class, fatal.

TQ032 - *dn*(*vol*) POSITION: UNABLE TO POSITION TAPE, TDSTS=*st*
An error occurred when positioning a tape. *st* is the status returned by the IOP. See TDT in COS Table Descriptions Internal Reference Manual, publication SM-0045. From TQM. Class, fatal.

TQ033 - *dn*(*vol*) IOP STATUS: *error* ERROR FUNCTION: *function*
An error *error* occurred while performing function *function*. From TQM. Class, fatal.

TQ034 - *dn(vol)* JOB DROPPED BY OPERATOR
Job was dropped by the operator. See the operator for details. From TQM. Class, fatal.

TQ035 - *dn(vol)* TAPEMARK WRITTEN, VBC=*vbc*
A tape mark was written to tape dataset. *vbc* is the block count of the tape mark. From TQM. Class, informative.

TQ036 - *dn(vol)* TAPEMARK READ, VBC=*vbc*
A tape mark was read. *vbc* is the block count of the tape mark. From TQM. Class, informative.

TQ037 - *dn(vol)* INVALID TAPE EOV/BOV SPECIAL PROCESSING REQUEST
The EOV processing request is invalid. From TQM. Class, fatal.

TQ038 - *dn(vol)* *function*: FUNCTION NOT ALLOWED
The requested function *function* is not allowed in user EOV processing routine. From TQM. Class, fatal.

TQ039 - *dn(vol)* POSITION: VOLUME SWITCH NOT ALLOWED AT USER EOV
The user has issued a tape position request in the user EOV processing routine. Volume switching in user EOV routine may only be done by issuing the CLOSEV macro. From TQM. Class, fatal.

TQ040 - *dn(vol)* *function*: USER EOV PROCESSING OPTION NOT ON
The user has issued an EOV processing request *function* before issuing the SETSP macro with the ON option. From TQM. Class, fatal.

TQ041 - *dn(vol)* INPUT: NO VALID SECTORS IN MOS
In user EOV processing routine, a read request is issued after all the output data in the IOS buffer has been read by the user. From TQM. Class, fatal.

TQ042 - *dn(vol)* CLOSE/POSITION/REWIND: EOV ENCOUNTERED
The user has requested user EOV processing by the SETSP macro with the ON option. During CLOSE, POSITION or REWIND processing, the EOV condition is encountered. The SYNCH macro should be issued and the user may process any EOV condition returned by the SYNCH macro before issuing the CLOSE, POSITION or REWIND request. From TQM. Class, fatal.

TQ043 - *dn(vol)* CLOSEV: NEXT VOLUME DOES NOT EXIST
The user issued the CLOSEV macro to switch volume but no more volumes exist. From TQM. Class, fatal.

TQ044 - *dn(vol)* CLOSEV: EOD ENCOUNTERED
The EOD condition is encountered when processing the CLOSEV request from the user. From TQM. Class, fatal.

TQ045 - *dn(vol)* ENDSP: STARTSP HAS NOT BEEN ISSUED
The user issued the ENDSP macro without issuing the STARTSP macro. From TQM. Class, fatal.

TQ046 - *dn(vol)* *function*: SYNCH MACRO NOT EXECUTED
The SYNCH macro has not been executed before the function *function* is requested. From TQM. Class, fatal.

TQ047 - *dn(vol)* NO DATA IN IOP FOR USER EOV ROUTINE TO READ
During EOV processing, the user read request encountered end-of-tape and there is no data in the IOS buffer. From TQM. Class, fatal.

TQ048 - *dn(vol)* UNABLE TO REPOSITION TAPE AFTER REMOUNT WITH RING
An error occurred when attempting to reposition the tape after a remount with ring. From TQM. Class, fatal.

TQ049 - *dn(vol)* ZERO LENGTH READ ATTEMPTED
The user program attempted to read a zero length record. From TQM. Class, fatal.

TQ050 - *dn(vol)* READ AT EOV BEFORE STARTSP
The user must begin EOV processing at EOV and may not read the dataset before EOV processing is started. From TQM. Class, fatal.

TQ051 - *dn(vol)* CLOSEV: TRAILER OPTION INVALID OUTSIDE USER EOV ROUTINE
The user program has issued a CLOSEV macro with the trailer option when the program is not in EOV processing routine. From TQM. Class, fatal.

TQ052 - *dn(vol)* CLOSEV TRAILER OPTION INVALID FOR INPUT DATASET
The user program has issued a CLOSEV macro in the EOV processing routine for an input dataset. The trailer option is ignored. From TQM. Class, informative.

TQ053 - *dn(vol)* NO ABORT CODE SPECIFIED BY CALLING ROUTINE
An abort request is issued with no abort code; the abort code ERABT is used. From TQM. Class, fatal.

TQ054 - *dn(vol)* CLOSE: UNABLE TO FLUSH USER BLOCKS TO TAPE AT EOV
The user requested user EOV processing. During CLOSE processing for an output dataset, the EOV condition is encountered. The data in the user I/O buffer cannot be flushed to tape because

there is no volume switching. A SYNCH request should be issued before the CLOSE request. The user may then process any EOV status returned by the SYNCH request before issuing the CLOSE request. From TQM. Class, fatal.

TQ056 - *dn(vol)* WRITE ILLEGAL WHEN RING=OUT SPECIFIED
This message is issued when a user attempts to write on a tape after requesting the tape be mounted without a write ring. This message is issued only when the site has selected ring processing via the I@RNGABT installation parameter. From TQM. Class, fatal.

TQ057 - RING OPTIONS NOT AVAILABLE AT THIS SITE
This message is issued during ACCESS processing for on-line tapes when the user has requested a tape without a ring and the site has selected not to do ring processing via the I@RNGABT installation parameter. From TQM. Class, informative.

TQ058 - *dn(vol)* USER EOV SPECIAL PROCESSING - ON
The user has set the EOV processing option to ON. From TQM. Class, informative.

TQ059 - *dn(vol)* USER EOV SPECIAL PROCESSING - OFF
The user has set the EOV processing option to OFF. From TQM. Class, informative.

TQ060 - *dn(vol)* START USER EOV PROCESSING
The user has started EOV processing. From TQM. Class, informative.

TQ061 - *dn(vol)* END USER EOV PROCESSING
The user has ended EOV processing. From TQM. Class, informative.

TQ062 - *dn(vol)* AT END OF VOLUME
End of volume has been encountered. From TQM. Class, informative.

TQ063 - *dn(vol)* STARTSP: VOLUME NOT AT EOV
The user program attempted to start EOV processing when the dataset is not at EOV. From TQM. Class, fatal.

TQ064 - *dn(vol)* SETSP OFF NOT ALLOWED IN USER EOV PROCESSING OR AT EOV
The user program attempted to set the EOV processing option OFF in the user EOV processing routine or when the EOV condition is encountered. The user may set the EOV processing option OFF when the dataset is not at EOV and when it is not in the EOV processing routine. From TQM. Class, fatal.

UB901 - UNB VERSION *vnum*
Message identifies version. From UNB. Class, informative.

UB902 - FATAL ASSEMBLY ERROR IN INPUT BINARY
The PDT table of the binary module indicates a fatal error at assembly time. The module must be reassembled, either with the fatal error corrected or with DEBUG specified on the CAL statement. From UNB. Class, fatal; job aborts.

UB903 - USE DEBUG PARAMETER ON CAL STATEMENT IF YOU STILL WANT THE BINARY
This is a continuation of message UT902, which was caused by a fatal assembly error. It always follows UT902 and precedes the job step abort. The module must be reassembled, either with the fatal error corrected or with DEBUG specified on the CAL statement. From UNB. Class, informative.

UB904 - SECTORS *m* THROUGH *n* WRITTEN
The indicated range of sectors was written in the unblocked file. *m* and *n* are octal numbers. Numbers are relative to the start of the file; the first sector is 0. From UNB. Class, informative.

NOTE

Classes for UD messages, issued by UPDATE, are different from those for other messages. The classes and meanings are as follows.

Informative: no action is taken.

Error: job aborts when UPDATE execution is finished, unless the UPDATE statement parameter NA is selected

Fatal error: aborts execution immediately

UD001 - PL: *dn* PL DATE: *m/d/y*
LAST ID: *id*
dn is the name of the dataset containing the program library, *m/d/y* is the creation date of this version of the PL, and *id* is the name of the last identifier added to the PL. Class, informative. From UPDATE.

UD002 - *n* UPDATE WARNINGS
n probable user errors were detected by UPDATE. Warning messages are written to the listing and error datasets if ML<4, E≠0, or L≠0. Class, informative. From UPDATE.

UD003 - EMPTY INPUT FILE, DN = *dn*
The next file in dataset *dn*, specified as an input dataset for UPDATE, is empty. Determine if the primary input file or READ datasets are non-null or need to be rewound. Class, informative. From UPDATE.

UD004 - DATASET NOT LOCAL, DN = *dn*
The dataset indicated was not accessed before UPDATE execution. It was the PL, an input dataset, or was named by a READ directive. Class, fatal error. From UPDATE.

UD005 - RECURSIVE READ OF DN = *dn*
An attempt was made to read dataset *dn* recursively with the READ directive. Class, error. From UPDATE.

UD006 - INVALID READ DATASET NAME, DN = *dn*
A READ directive encountered by UPDATE while reading input contains an invalid dataset name. Class, error. From UPDATE.

UD007 - ERROR IN UPDATE CONTROL STATEMENT
One or more of the following control statement errors exist:
- Both the new PL and the old PL datasets have the same name
- Both F and Q were specified
- P=0 and I=0 (PL creation mode and no input)
- Invalid comment and/or master character
- Invalid DW value

Class, fatal error. From UPDATE.

UD008 - MODS WITHOUT IDENTIFIER, NEW PL SUPPRESSED
A new PL cannot be generated with modifications that are not identified with an IDENT directive. Generation of a new program library has been suppressed. Occurs only when the N parameter specifies creation of a new PL, or in creation runs. Class, error. From UPDATE.

UD010 - INVALID PROGRAM LIBRARY, DN = *dn*
dn is not in a PL format recognized by UPDATE. Class, fatal error. From UPDATE.

UD011 - *n* FATAL UPDATE ERRORS
n fatal errors were detected by UPDATE. Error messages are written to the listing and error datasets. Class, informative. From UPDATE.

UD012 - PL FORMAT CONVERSION COMPLETE
A sequential format program library has been internally rewritten as a random format PL. Class, informative. From UPDATE.

UD013 - SEQUENCE NUMBER EXCEEDS 131071 FOR ID = *id*
An attempt was made to add more than 131,071 lines with one identifier. The insertion must be split over two or more identifiers or decks. Class, fatal error. From UPDATE.

UD014 - NUMBER OF IDENTIFIERS EXCEEDS 16383
Too many deck, common deck, and modification set identifiers have been defined for this program library. Before any new identifiers can be added, the PL must be resequenced by creating a new PL from the source dataset. Class, fatal error. From UPDATE.

UD015 - DECK SPECIFIED BY Q PARAMETER NOT FOUND, DECK=*deckname*
deckname was listed as a value for the Q control statement parameter but is not a deck or common deck in this program library and was not introduced by the input datasets. Class, error. From UPDATE.

UD016 - PL MASTER CHARACTER IS: *m*
m is the master character recorded when the program library was created. It is the default for the master character used in the input and source datasets. This is only written if the master character is not the default (*). Class, informative. From UPDATE.

UD017 - *n* MODIFICATION SETS SKIPPED
n modification sets were skipped due to unsatisfied IDENT directive dependency conditions. Use ML=1 on the UPDATE control statement to get a NOTE message written to the listing and error datasets for each skipped IDENT. Class, informative. From UPDATE.

UD018 - *n* UNPROCESSED MODIFICATION DIRECTIVES
n modification directives were left unprocessed when UPDATE finished execution, either because they modified decks and common decks that were not specified in a quick mode UPDATE run or because they referenced lines that were not found in the program library. Use the UM option on the UPDATE control statement to get a list of unprocessed modifications. Class, informative. From UPDATE.

UD019 - *n* INPUT LINES TRUNCATED TO *pldw* CHARACTERS
n input lines longer than *pldw* characters were truncated to *pldw* characters. *pldw* is the number of characters per line stored in the program library, and is defined by the DW control statement parameter. The minimum, and default, value for *pldw* is 80. Class, informative. From UPDATE.

UD020 - MORE THAN 100 FATAL INPUT ERRORS
More than 100 fatal input errors were detected and UPDATE aborted. A DECK or COMDECK directive may be missing, or the wrong master character may have been specified. Class, error. From UPDATE.

UD021 - *n* OVERLAPPING MODIFICATIONS
There were *n* directives that either referenced lines that were inserted earlier in the same UPDATE run or that deleted a range of text that included newly inserted lines. Use ML=1 on the UPDATE control statement to get NOTE and CAUTION messages about overlaps to determine if the overlaps were proper and expected. Class, informative. From UPDATE.

UD022 - INVALID DC PARAMETER VALUE
The UPDATE control statement parameter DC is equated to an invalid value. Valid values are ON and OFF. Class, fatal error. From UPDATE.

UD023 - INTERNAL UPDATE ERROR: ID NOT IN SEQUENCE TABLE
An UPDATE internal logic error caused the current identifier name not to be found in the Sequence Table. Class, fatal error. From UPDATE.

UD024 - INTERNAL UPDATE ERROR: INVALID DIRECTIVE KEY
An UPDATE internal logic error caused an invalid directive key to be assigned. Class, fatal error. From UPDATE.

UD025 - INTERNAL UPDATE ERROR: PL I/O STATUS ERROR
An UPDATE internal logic error caused an I/O status error to occur during a read of the PL. Class, fatal error. From UPDATE.

UD026 - INTERNAL UPDATE ERROR: $UDT1 I/O STATUS ERROR
An UPDATE internal logic error caused an I/O status error in a character read of temporary dataset $UDT1. Class, fatal error. From UPDATE.

UD027 - INTERNAL UPDATE ERROR: $UDT2 I/O STATUS ERROR
An UPDATE internal logic error caused a count exhaustion during a character read of temporary dataset $UDT2. Class, fatal error. From UPDATE.

UD028 - INTERNAL UPDATE ERROR: ID NAME NOT IN IDENTIFIER TABLE
An UPDATE internal logic error caused an identifier name to be omitted from the Identifier Table. Class, fatal error. From UPDATE.

UD029 - INTERNAL UPDATE ERROR: DECK NAME NOT IN IDENTIFIER TABLE
An UPDATE internal logic error caused an identifier name to be missing from the Identifier Table so it was not found when UPDATE tried to get the name of the next deck to process. Class, fatal error. From UPDATE.

UD030 - INTERNAL UPDATE ERROR: ID NUMBER NOT IN IDENTIFIER TABLE
An UPDATE internal logic error caused an identifier to be missing from the Identifier Table. Class, fatal error. From UPDATE.

UD031 - INTERNAL UPDATE ERROR: ERROR IN ID LIST IN OLD FORMAT PL
UPDATE was unable to read the identifier list in an old format program library. Class, fatal error. From UPDATE.

UD032 - INTERNAL UPDATE ERROR: OLD FORMAT PL IS UNREADABLE
UPDATE was unable to read an old format program library. Class, fatal error. From UPDATE.

UD033 - INTERNAL UPDATE ERROR: PL INFORMATION FILE READ ERROR
UPDATE had a read error on a partial record read of the PL information file. Class, fatal error. From UPDATE.

UD034 - INTERNAL UPDATE ERROR: UNKNOWN PROCESS TYPE
An UPDATE internal logic error caused an invalid process type (INSERT, BEFORE, DELETE, RESTORE) to be returned from a Modification Table entry. Class, fatal error. From UPDATE.

UD035 - INTERNAL UPDATE ERROR: DECK DIRECTIVE IN COMDECK
An UPDATE internal logic error caused a DECK directive to be placed in a common deck instead of starting a new deck. Class, fatal error. From UPDATE.

UD036 - INTERNAL UPDATE ERROR: COMDECK DIRECTIVE IN DECK
An UPDATE internal logic error caused a COMDECK directive to be placed in a deck instead of starting a new common deck. Class, fatal error. From UPDATE.

UD037 - LIST CONTROL STATEMENT PARAMETER IGNORED
The LIST control statement parameter is not supported and is ignored. Class, informative. From UPDATE.

UD038 - INTERNAL UPDATE ERROR: NEW PL TABLE IS UNSORTED
A call to library routine ORDERS failed; the new program library is ordered as specified by the K option, but future program libraries built from it will revert to the old ordering. Class, informative. From UPDATE.

UD039 - INTERNAL UPDATE ERROR: NO SECOND DELETE ENTRY
An UPDATE internal logic error caused the second entry in the Modification Table for a DELETE or RESTORE range to be missing. Class, fatal error. From UPDATE.

UD040 - INTERNAL UPDATE ERROR: BAD HDC IN PL LINE
When handling deleted lines with bad correction histories, an active line was found with a header descriptor count of 0. Class, fatal error. From UPDATE.

UD041 - *n* MULTIPLE INSERTIONS
There were *n* locations in which new text was inserted by directives in more than one modification set. Use ML=2 on the UPDATE control statement to have CAUTION messages written to the listing dataset for each multiple insertion. Class, informative. From UPDATE.

UD042 - INVALID ML PARAMETER VALUE; MUST BE 0-4
An invalid value was specified for the ML parameter on the UPDATE control statement. Class, fatal error. From UPDATE.

UD043 - INTERNAL UPDATE ERROR: DELETE TABLE ENTRY NOT FOUND
An UPDATE internal logic error caused an entry to be missing from the Delete Table. Class, fatal error. From UPDATE.

UD044 - DATASET *dsname* USED FOR MORE THAN ONE PURPOSE
The dataset was specified by more than one control statement parameter; for example, both C and S. From UPDATE. Class, fatal error.

NOTE

Classes for UD messages, issued by UPDATE, are different from those for other messages. See the beginning of the UD messages for an explanation.

UT000 - BAD CALL TO LIBRARY ERROR PROCESSOR
The $UTLIB error message processor was entered with an illegal error message code. See a Cray Research analyst. From $UTLIB. Class, 2; user abort; reprievable.

UT001 - ILLEGAL CHARACTER IN DECIMAL CONVERSION
The DTB routine encountered a non-numeric character. Check the character string input to DTB. From $UTLIB. Class, 2; user abort; reprievable.

UT002 - ILLEGAL CHARACTER IN OCTAL CONVERSION
The octal-to-binary converter (OTB) encountered a non-octal character. Check the character string input to OTB. From $UTLIB. Class, 2; user abort; reprievable.

UT003 - EXIT CALLED BY *routine*
The specified routine performed a CALL EXIT. From $UTLIB. Class, 1; normal termination; reprievable.

UT004 - PDUMP ABORT AFTER COMPLETION
The DUMP entry in PDUMP was called to request an abort after memory was dumped. From $UTLIB. Class, 2; user abort; reprievable.

UT006 - PDUMP ERROR: F LT 0 OR GT 7
The third parameter to PDUMP, F must be in the range 0 to 7, inclusive. The dump request is ignored. From $UTLIB. Class, informative

UT007 - PDUMP ERROR: INCORRECT NUMBER OF ARGUMENTS
PDUMP must be called with at least three arguments. From $UTLIB. Class, 2; user abort; reprievable.

UT008 - *user remark*
A user call was issued to subroutine REMARK to write a message to the logfile. From $UTLIB. Class, informative.

UT009 - *user remark*
A user call was issued to REMARKF to provide a message to the logfile under format control. REMARKF allows the UT009 to be replaced if the first eight characters of the message are *xxxxx* - . From $UTLIB. Class, informative.

UT010 - STOP *n* IN *m*
STOP *n* was executed in routine *m*. Class, 1; normal termination; reprievable.

UT011 - PAUSE NOT SUPPORTED, STOP SUBSTITUTED
A PAUSE was executed. This version UTLIB does not support PAUSE; substitute with STOP (see UT010). From $UTLIB. Class, informative.

UT012 - PAUSE *n* IN *m*
A PAUSE *n* was executed in routine *m*. Operator intervention is required for resuming execution or dropping the job. From $UTLIB. Class, informative.

UT013 - FATAL STACK OVERFLOW
A stack allocation request could not be met because there is not enough space on the stack and the stack could not be extended. Specify a stack increment on the LDR control statement or increase the initial stack size value. From $UTLIB. Class, fatal.

UT015 - EVREL CALLED WITH TASKS WAITING FOR EVENT
An event variable was released with EVREL, but was being waited for by some task. From $UTLIB. Class, fatal.

UT016 - LOCKREL CALLED WITH LOCK SET
A lock variable was released with LOCKREL but was currently in use by some task. From $UTLIB. Class, fatal.

UT017 - INVALID LOCK IDENTIFIER
LOCKREL was called but the specified lock variable appears to be invalid. Check that the lock variable was assigned and that it was not accidentally overwritten. From $UTLIB. Class, fatal.

UT018 - REQUESTED HEAP BLOCK LENGTH IS INVALID
The requested length for a block of memory was not an integer greater than zero. Check the value passed to the routine that aborted. From $UTLIB. Class, fatal.

UT019 - HEAP IS FULL, CAN'T SATISFY REQUEST
An allocation request could not be met because no block in the heap was large enough and the heap could not be extended. Specify a managed memory increment on the LDR control statement or increase the amount of memory allowed for the job. From $UTLIB. Class, fatal.

UT020 - ADDRESS IS OUTSIDE THE BOUNDS OF THE HEAP
The address of a heap block to be deallocated or changed was outside of the bounds of the heap. Check the address passed to the routine that aborted. From $UTLIB. Class, fatal.

UT021 - HEAP BLOCK IS FREE
The address for a heap block to be deallocated or changed was for a block already on the list of available space. Check for the deallocation routine being called more than once with the same address. From $UTLIB. Class, fatal.

UT022 - BAD CONTROL WORD FOR ALLOCATED BLOCK
The control word at the beginning of a heap block to be changed or deallocated (at the word preceding the address passed to the heap manager routine) has been overwritten. Check memory references to stack variables, managed tables, or other locations in the heap for incorrect subscripts, etc. From $UTLIB. Class, fatal.

UT023 - BAD CONTROL WORD IN FOLLOWING BLOCK
The control word at the beginning of the heap block following the block to be changed or deallocated has been overwritten. Check memory references to stack variables, managed tables, or other locations in the heap for incorrect subscripts, etc. From $UTLIB. Class, fatal.

UT024 - DEADLOCK - ALL USER TASKS WAITING FOR LOCKS, EVENTS, OR TASKS
The library has detected a situation in which all active tasks are suspended for locks, events, or other tasks. From $UTLIB. Class, fatal.

UT025 - UNRECOGNIZED SCHEDULER PARAMETER NAME
An ASCII string passed as a TSKTUNE parameter was not recognized. From $UTLIB. Class, fatal.

UT026 - INVALID INPUT TO CONVERSION ROUTINE
An invalid argument was passed to a data conversion routine in $UTLIB. Refer to the Library Reference Manual, CRI publication SR-0014, for the correct usage of the routine. From $UTLIB. Class, 2; user abort; reprievable.

UNCODED MESSAGES 2

This section shows messages that do not begin with a unique code. These messages are grouped for locating particular messages, as follows:

- Cray FORTRAN (CFT) messages (2.1). Messages beginning with variable names are alphabetized by the first word following the variable name; all others are alphabetized by the first word.
- Messages that do not begin with variable names (2.2)
- Messages that begin with variable names (2.3)
- SEGLDR messages beginning with *ERROR*, *WARNING*, *CAUTION*, *NOTE*, OR *COMMENT* (2.4)
- SID messages beginning with **** (2.5)
- SKOL messages beginning with *------ (2.6)
- CSIM messages (2.7)
- Startup messages (2.8).

Note that the first three of these categories are not immediately distinguishable. If you cannot find an uncoded message in one section, try the other two. CSIM messages have no identifying prefix, but are not intermixed with other kinds of messages.

2.1 CRAY FORTRAN (CFT) MESSAGES

Parameter E in the CFT control statement determines the highest severity level of CFT-produced messages to be suppressed. E=0 allows messages of all classes to be listed. The following levels are available.

Class	Description
Informative	Comments about programming inefficiencies (level 1), and notes about possible problems with other compilers (level 2).
Caution	Possible user error (level 3)
Warning	Probable user error (level 4)
Fatal	Fatal error (level 5)

When compiling with the ANSI Keyword specified on the CFT control statement, some informative, caution, and warning messages are issued as Class NON-ANSI. These messages refer to nonstandard coding practices. NON-ANSI class messages are not affected by the level of severity selected for the E= parameter on the control statement; they cannot be suppressed when the ANSI option is selected.

"*name*" ALREADY SAVED AT SEQUENCE NUMBER *n*
"*name*" appears in both a SAVE statement list and the preceding SAVE statement list at sequence number *n*. Ensure that "*name*" appears in at most one SAVE list. From CFT compile time. Class, fatal.

"*name*" APPEARS TWICE IN DUMMY ARGUMENT LIST
A symbolic name appears twice in the dummy argument list of a FUNCTION or SUBROUTINE statement. From CFT. Class, fatal.

"*name*" CANNOT BE DECLARED EXTERNAL
"*name*" has appeared in both an EXTERNAL statement and in an array declarator, a DATA statement, a PARAMETER statement, or a COMMON statement. From CFT. Class, fatal.

"*name*" CANNOT BE DECLARED INTRINSIC
The symbolic name has already been used in the program unit giving the name a type, for example, a dummy argument. From CFT. Class, fatal.

"*name*" DOUBLY ASSOCIATED IN EQUIVALENCE AT SEQUENCE "*number*"
The EQUIVALENCE statements involving "*name*" are in error, specifying an illegal storage sequence. The same storage unit must not occur more than once in a storage sequence. For example,

```
REAL A(2)
EQUIVALENCE (X,A(1)),(X,A(2))
```

is in error because it specifies the same storage unit for A(1) and A(2). Consecutive storage units must not be

specified as being nonconsecutive. For example,

```
REAL A(2)
DOUBLE D(2)
EQUIVALENCE (A(1),D(1)),(A(2),D(2))
```

is in error because it specifies that A(1) and A(2) be nonconsecutive. Fix the EQUIVALENCE statements in error. From CFT. Class, fatal.

"*name*" IS STATICALLY ALLOCATED
CFT will assign "*name*" to static storage. If this is what you want, fine; otherwise remove "*name*" from SAVE and DATA statements. From CFT. Class, warning.

"*name*" NO LONGER INTRINSIC
An intrinsic function name appears in a compiler directive which gives external subprograms special attributes (VFUNCTION, NO SIDE EFFECTS). The intrinsic function loses its intrinsic properties. Declare the intrinsic function in an EXTERNAL statement before the compiler directive declaration. From CFT. Class, warning.

"*name*" NOT AN INTRINSIC FUNCTION
A symbolic name in the list of an INTRINSIC statement is not an intrinsic function. From CFT. Class, fatal.

"*name*" NOT LOCAL OR COMMON BLOCK VARIABLE
List can contain only local or common block variables. From CFT. Class, fatal.

"*name*" STATEMENT IS A NONSTANDARD STATEMENT
The statement type indicated is a CFT extension to the FORTRAN language specified by the ANSI FORTRAN Standard. From CFT. Class, informative; NON-ANSI.

"*name*" USED AS SYMBOLIC CONSTANT AND AS COMMON BLOCK NAME
The same identifier was used for a common block name and a symbolic constant name. The ANSI 77 FORTRAN Standard prohibits this duplicate usage within the same program unit. From CFT. Class, NON-ANSI.

ABBREVIATION OF *name* IS NONSTANDARD
The logical operator or constant indicated by *name* has been abbreviated. The ANSI standard does not provide for abbreviation of logical constants or operators. From CFT. Class, informative; non-ANSI.

ADJUSTABLE DIMENSION ILLEGAL IN MAIN PROGRAM
Arrays in a main program must have constant subscripts. From CFT. Class, fatal.

ARGUMENTS IGNORED FOR ZERO ARGUMENT INTRINSIC
An intrinsic function defined to have zero arguments was referenced with one or more arguments. Reference zero argument intrinsics with a null argument list. See the FORTRAN (CFT) Reference Manual, CRI publication SR-0009, for information. From CFT. Class, warning.

ARITHMETIC EXPRESSION WITH DOUBLE PRECISION AND COMPLEX IS NONSTANDARD
An arithmetic expression has been detected with a COMPLEX and a DOUBLE PRECISION operand. CFT will convert the DOUBLE PRECISION operand to COMPLEX. The ANSI FORTRAN standard does not provide for such conversions. From CFT. Class, informative; non-ANSI.

ARRAY NAMED FORMAT IS POTENTIALLY AMBIGUOUS
An array with the symbolic name FORMAT was declared using the DIMENSION, COMMON, INTEGER, REAL, DOUBLE, or COMPLEX statement. Use an array name other than FORMAT to remove the warning message, or do not use statement labels on any assignment statements using the array FORMAT. See the FORTRAN (CFT) Reference Manual, CRI publication SR-0009, on array declarators. From CFT. Class, warning.

ASSUMED SIZE DIMENSION ILLEGAL IN MAIN PROGRAM
Arrays in a main program must have constant subscripts. From CFT. Class, fatal.

AT SEQUENCE NUMBER *n*: "*name*" CANNOT BE SAVED
name appears in a SAVE statement list at sequence number *n*; this is not permitted. From CFT compile time. Class, fatal.

BAD CONSTANT LIST IN DATA STATEMENT
A DATA statement is missing a constant list, or it contains an illegal separator character. From CFT. Class, fatal.

BAD HEXADECIMAL CONSTANT
Illegal seperator character between hexadecimal digits or the constant's length is greater than 16 digits. From CFT. Class, fatal.

BAD REPETITION FIELD
The repetition count in a DATA constant list must be an integer greater than 0. From CFT. Class, fatal.

BAD STATEMENT FUNCTION PARAMETER LIST
The formal parameter list in an arithmetic statement function definition statement contains an illegal element. From CFT. Class, fatal.

BAD SUBSCRIPT IN DATA STATEMENT
A subscript must be an integer constant or constant name in a DATA statement. From CFT. Class, fatal.

BAD TRIP COUNT IN IMPLIED DO
Incrementation parameter m_3 has been assigned a value of 0 or $(m_2-m_1+m_3)/m_3$ is negative or 0. From CFT. Class, fatal.

BLOCK IF STATEMENTS NESTED TOO DEEPLY
The nesting of block IF statements exceeds 1023. CFT cannot handle a nesting depth greater than this number. Reduce the number of nesting statements. From CFT. Class, fatal.

BRANCH INTO IF, ELSE, OR ELSE IF BLOCK; LABEL *l*
A branch into an IF, ELSE, or ELSEIF block to label *l* from outside the block has been detected. The ANSI FORTRAN standard prohibits transfer of control into an IF, ELSE, or ELSEIF block. From CFT. Class, fatal.

BUFFERED IO IS NONSTANDARD
A BUFFER IN or BUFFER OUT statement was used. Buffered input and output are CFT extensions to the ANSI 77 FORTRAN Standard. From CFT. Class, informative, NON-ANSI.

CALL OF NON EXTERNAL FUNCTION "*name*"
Called external procedure does not exist or a name has been used for both a variable and an external procedure. From CFT. Class, fatal.

CHARACTER COUNT TOO LARGE
An R-form Hollerith constant is specified with more than eight characters or an H- or L-form Hollerith constant is specified with more than eight characters in other than a DATA statement or an actual argument list. From CFT. Class, fatal.

CHARACTER LENGTH MUST BE >ZERO AND <16383
A character entity must be assigned a length greater than 0 and less than 16383. From CFT. Class, fatal.

COMMA EXPECTED
A required comma has been omitted. From CFT. Class, fatal.

COMMA EXPECTED IN EQUIVALENCE AT SEQUENCE "*number*"
A required comma has been omitted in an equivalence statement. From CFT. Class, fatal.

COMMA OR RIGHT PARENTHESIS EXPECTED
A required comma or right parenthesis has been omitted. From CFT. Class, fatal.

COMMA OR RIGHT PARENTHESIS EXPECTED IN EQUIVALENCE AT SEQUENCE "*number*"
A required comma or right parenthesis has been omitted in an equivalence statement. From CFT. Class, fatal.

COMMA REQUIRED BY STANDARD BETWEEN FORMAT FIELDS
A comma is required by the ANSI standard between most FORMAT edit descriptor fields, while CFT views most commas in FORMAT specifier lists as optional. From CFT. Class, Informative, NON-ANSI.

COMMON BLOCK "*name*" IS VERY LARGE; USE EXTENDED MEMORY ADDRESSING OPTION
A named or blank common block was declared with more than four million words of storage. The program must be compiled with the extended memory addressing (EMA) option specified on the CFT control card. From CFT. Class, fatal.

COMMON BLOCK NAME "*name*" MORE THAN 6 CHARACTERS
The common block name has seven or eight characters. ANSI 77 FORTRAN Standard common block names cannot have more than six characters. From CFT. Class, informative, NON-ANSI.

COMPILER ERROR
The CFT compiler detected an error in its internal tables. See a Cray Research analyst. From CFT. Class, fatal.

COMPILER ERROR --INTERNAL TABLE OVERFLOW, RECOMPILE WITH SMALLER BLOCK SIZE
One of CFT's internal tables has overflowed while compiling. The optimization block where the error occurs must be reduced in size. This can be done either with a CDIR$ BLOCK compiler directive inserted in the block or by reducing the maximum block size with the MAXBLOCK = control card parameter. From CFT. Class, fatal.

CONFLICTING TYPE FOR INTRINSIC FUNCTION "*name*" IGNORED
A type statement cannot change the type of an intrinsic function. The function must be declared EXTERNAL before its type can be changed. From CFT. Class, fatal.

CONFLICTING USE OF INTRINSIC FUNCTION "*name*"
An intrinsic function name has been used to reference the function in another way within the program unit. From CFT. Class, fatal.

CONSTANT DIMENSION TOO LARGE
All dimensions must be less than 2^{22}. From CFT. Class, fatal.

CONSTANT LIST LONGER THAN VARIABLE LIST
Constants and variables in a DATA statement must have a one-to-one correspondence. From CFT. Class, fatal.

CONSTANT SUBSCRIPT TOO LARGE
Statement contains a subscript that, when evaluated, yields a result greater than the size of the named array. From CFT. Class, fatal.

CONTROL LIST MUST INCLUDE ONE FILE OR UNIT OPTION
An INQUIRE statement does not specify a file or a unit. From CFT. Class, fatal.

CONTROL LIST MUST INCLUDE ONE UNIT OPTION
The I/O statement must specify a unit. From CFT. Class, fatal.

DATA ENTRY IN BLANK COMMON ILLEGAL
The DATA statement cannot be used to initialize blank common. From CFT. Class, fatal.

DATA INITIALIZATION OF COMMON VARIABLE *name* NOT IN BLOCK DATA SUBPROGRAM
The entity indicated by *name* has appeared in a COMMON block specification and has been initialized in a DATA statement in an executable subprogram. The ANSI FORTRAN standard allows DATA initialization of COMMON entities only in a BLOCK DATA subprogram. From CFT. Class, informative; non-ANSI.

DATA STATEMENT PRECEEDS SPECIFICATION STATEMENT
The ANSI standard specifies an order for nonexecutable statements. DATA statements are required to appear after all specification statements. CFT has detected a DATA statement preceding a specification statement. From CFT. Class, informative; non-ANSI.

DECLARATOR "*name*" MUST BE DUMMY ARGUMENT OR IN COMMON
The dimension declarator "*name*" in an adjustable array declarator was not a dummy argument or a variable in common. From CFT. Class, fatal.

DEPENDENCY INVOLVING ARRAY "*name*"
A dependency exists, involving two array references in the same statement. This dependency inhibits DO-loop vectorization. Change the program to eliminate the dependency. See the FORTRAN (CFT) Reference Manual, publication SR-0009, for a complete explanation of dependencies. From CFT. Class, informative.

DEPENDENCY INVOLVING ARRAY "*name*" IN SEQUENCE NUMBER *n*
A dependency exists involving an array reference in sequence number *n* and an array reference in the AT SEQUENCE NO *m* line that precedes this message. This dependency inhibits DO-loop vectorization. Change the program to eliminate dependency. See the FORTRAN (CFT) Reference Manual, publication SR-0009, for a complete explanation of dependencies. From CFT. Class, informative.

DIMENSION COUNT > SEVEN
More than seven dimensions appear in an array declarator. From CFT. Class, fatal.

DIMENSION EXCEEDED
A subscript in a DATA statement element exceeds the corresponding array declaration. From CFT. Class, fatal.

DIRECTIVE NO LONGER SUPPORTED
The SCHED and NOSCH directives are no longer supported. The directive will have no effect. From CFT. Class, caution.

DIVIDE BY ZERO
Dividing by the constant 0 is illegal. From CFT. Class, fatal.

DO ILLEGAL ON CONDITIONAL STATEMENT
This type of statement is not allowed as the conditional statement of a direct logical IF statement. From CFT. Class, fatal.

DO INDEX ACTIVE
The loop control variable is already active from a previous loop. From CFT. Class, fatal.

DO INDEX IN INPUT LIST
An attempt was made to read data into a DO variable. From CFT. Class, fatal.

DO LOOP MAY NOT CROSS BLOCK BOUNDARY
A DO-loop that begins within an IF-block, ELSE-block, or ELSE IF-block must be totally contained within that block. A block that begins within a DO-loop must be totally contained within the loop. From CFT. Class, fatal.

DO TERMINATOR ILLEGAL IN CONDITIONAL BLOCK STATEMENT
A DO-loop must not terminate on an IF(*e*)THEN, ELSE, ELSE IF, or ENDIF statement. From CFT. Class, fatal.

DO TERMINATOR PRECEDES DO STATEMENT
The statement label terminating a DO-loop precedes the corresponding DO statement. From CFT. Class, fatal.

DOUBLE PRECISION CONSTANT IN COMPLEX CONSTANT IS NONSTANDARD
A DOUBLE PRECISION constant is used to form a COMPLEX constant. This is allowed by CFT when DOUBLE PRECISION is disabled (OFF=P). It is, however, a violation of the ANSI FORTRAN standard. From CFT. Class, informative; non-ANSI.

DOUBLY DEFINED FUNCTION OR MISSING DIMENSION
An arithmetic statement function is defined more than once or an array was not dimensioned. From CFT. Class, fatal.

DOUBLY DEFINED STATEMENT NUMBER
Statement labels cannot be defined more than once in a program unit. From CFT. Class, fatal.

DUMMY ARGUMENT IN EXECUTABLE STATEMENT PREVIOUS TO ENTRY
A dummy argument name in an executable statement must also be specified in the FUNCTION, SUBROUTINE, or ENTRY statement referenced before the executable statement. From CFT. Class, fatal.

DUPLICATE COMMON DEFINITION "*name*"
Variable or array appears in common more than once. From CFT. Class, fatal.

DUPLICATE CONTROL OPTION IN LIST
An option is specified more than once in an I/O statement control list. From CFT. Class, fatal.

DUPLICATE DIMENSION "*name*"
Dimensions cannot be declared more than once. From CFT. Class, fatal.

DUPLICATE TYPE DEFINITION "*name*"
Variables cannot be given more than one type. From CFT. Class, fatal.

DYNAMIC BLOCK "*name*" NOT IN PREVIOUS COMMON
Dynamic name must be declared as a common block before its appearance in a DYNAMIC compiler directive. From CFT. Class, fatal.

EBCDIC NOT IMPLEMENTED
The current version of CFT allows only ASCII characters. From CFT. Class, fatal.

EMBEDDED COMMENTS ARE NONSTANDARD
A line of source code contains an embedded comment, which is text following an exclamation point. Embedded comments are a CFT extension to the ANSI 77 FORTRAN Standard. From CFT. Class, informative; NON-ANSI.

EMPTY PARENTHESES ILLEGAL IN FORMAT
CFT found an empty set of parentheses nested in a FORMAT specifier list. Only the outer-most set of parentheses is allowed to be empty. From CFT. Class, fatal.

ENCODE/DECODE MAY NOT BE LIST DIRECTED
The format identifier in an ENCODE or DECODE statement must not specify list-directed I/O; it cannot be an *. From CFT. Class, fatal.

ENTRY "*name*" USED AS DUMMY ARGUMENT
name is an entry point (that is, it has appeared in a FUNCTION, SUBROUTINE, ENTRY, or BLOCK DATA statement) and also appears as a dummy argument. An entry name cannot appear as a dummy argument. Give the entry or dummy argument a different name. From CFT. Class, fatal.

ENTRY NAME ILLEGAL
ENTRY name not a function or subroutine name. From CFT. Class, fatal.

ENTRY STATEMENT ILLEGAL IN DO LOOP OR BLOCK IF
The ENTRY statement must not be used in a DO-loop or in an IF-block. From CFT. Class, fatal.

ENTRY STATEMENT ILLEGAL IN MAIN PROGRAM
The ENTRY statement must not be used in a main program. It is used only in a subroutine or function. From CFT. Class, fatal.

EQUIVALENCE EXTENDS COMMON BLOCK BASE
Common block storage was illegally extended by adding storage units preceding the first storage unit specified in the COMMON statement. From CFT. Class, fatal.

EQUIVALENCE OF "*name*" IN DIFFERENT COMMON BLOCKS
An EQUIVALENCE statement must not associate the storage sequences of two different common blocks in the same program unit. From CFT. Class, fatal.

ERROR IN CONSTANT
Illegal characters in constant, or constant out of range. From CFT. Class, fatal.

EXECUTABLE CODE IN BLOCK DATA SUBPROGRAM
Executable statements appear in a BLOCK DATA subprogram. This is a violation of the ANSI FORTRAN standard. From CFT. Class, fatal.

EXPRESSION ILLEGAL IN INPUT LIST
Input list item is not a variable name, array element name, or array name. From CFT. Class, fatal.

EXPRESSION TYPE MUST BE INTEGER
Expressions in an alternate RETURN statement must be type integer. From CFT. Class, fatal.

EXTENDED RANGE DO LOOP IS NONSTANDARD
A branch into the range of a DO loop or a possible branch via an ASSIGN, END= or ERR= branch to a label defined in the range of a DO loop has been detected by CFT. Extended range DO loops are a CFT extension to ANSI standard FORTRAN. From CFT. Class, informative; non-ANSI.

EXTRA CHARACTERS AFTER END OF STATEMENT
Characters are specified after the syntactical end of a statement. From CFT. Class, fatal.

EXTRA CHARACTERS AFTER END OF STATEMENT IN EQUIVALENCE AT SEQUENCE "*number*"
Characters are specified after the syntactical end of an EQUIVALENCE statement. From CFT. Class, fatal.

EXTRA COMMA OR MISSING PARAMETER
The statement contains an extra comma or a parameter or list item was omitted. From CFT. Class, fatal.

FEWER SUBSCRIPTS USED THAN DECLARED
A reference to an actual array element has fewer subscript expressions in its subscript than dimension declarators in the corresponding array declarator. The missing subscript expressions are assumed rightmost in the subscript and are each assigned the value 1 by the compiler. From CFT. Class, warning; non-ANSI.

FIELD WIDTH MUST NOT BE ZERO
The field width following a FORMAT edit descriptor is zero, that is, F0.2 or G20.8E0. From CFT. Class, fatal.

FIELD WIDTH VALUE TOO SMALL
The value of the field width of a FORMAT edit descriptor is too small to print a value as specified, that is, F3.5. From CFT. Class, fatal.

FORMAT MUST BE CHARACTER EXPRESSION
A FORMAT specifier can be an expression only if the expression is a character expression. From CFT. Class, fatal.

FUNCTION "*name*" ALREADY DECLARED EXTERNAL
The symbolic name appearing in an INTRINSIC statement already appeared in an EXTERNAL statement. From CFT. Class, fatal.

FUNCTION "*name*" ALREADY DECLARED INTRINSIC
The symbolic name appearing in an INTRINSIC statement already appeared in an INTRINSIC statement. From CFT. Class, fatal.

FUNCTION "*name*" MORE THAN 6 CHARACTERS
An attempt has been made to declare a function whose name is greater than six characters as having a vector call-by-value version (CDIR$ VFUNCTION). Rename the function with a shorter name. From CFT. Class, fatal.

FUNCTION "*name*" MUST BE DECLARED IN INTRINSIC OR EXTERNAL STATEMENT
A function passed to another subprogram as an actual argument must be declared in an INTRINSIC statement (intrinsic functions) or an EXTERNAL statement (user-supplied functions). From CFT. Class, fatal.

FUNCTION "*name*" NOT DECLARED
IMPLICIT NONE or IMPLICIT SKOL was specified but function *name* did not appear in an EXTERNAL statement. This message is also issued if *name* was intended to be an array but did not appear in an array declarator. From CFT. Class, fatal.

FUNCTION OR CALL "*name*" REFERENCES ITSELF
A reference to the function or subroutine subprogram being compiled is encountered with that subprogram. From CFT. Class, fatal.

FUNCTION USED WITH INCORRECT NUMBER OF ARGUMENTS
The number of arguments in a function reference does not match the number of arguments in the function definition. From CFT. Class, fatal.

GROUP NAME DEFINED PREVIOUS TO NAMELIST
A group name can be defined only in NAMELIST. From CFT. Class, fatal.

H,L,R COUNT < OR = ZERO
In an *n*H, *n*L, or *n*R specification of a Hollerith value, *n* is less than or equal to 0. From CFT. Class, fatal.

H,L,R COUNT PAST END OF STATEMENT
In an *n*H, *n*L, or *n*R specification of a Hollerith value, *n* specifies more characters than are provided, or an apostrophe terminating a Hollerith string is missing. From CFT. Class, fatal.

HOLLERITH CONSTANTS ARE NONSTANDARD
A Hollerith constant was used in a statement other than a FORMAT statement. The ANSI 77 FORTRAN Standard allows Hollerith constants only in FORMAT statements. From CFT. Class, informative; NON-ANSI.

HEXADECIMAL IS NONSTANDARD
The ANSI FORTRAN standard does not provide for hexadecimal constants. From CFT. Class, informative.

HOLLERITH CONSTANT > EIGHT CHARACTERS
A Hollerith constant of more than eight characters is specified in other than H- or L-form and in other than an actual argument list or a DATA statement constant. From CFT. Class, fatal.

IDENTIFIER "*name*" MORE THAN 6 CHARACTERS
The identifier contains seven or eight characters. The ANSI 77 FORTRAN Standard provides for a maximum of six characters in identifier names. From CFT. Class, NON-ANSI.

IF BLOCK LEVEL NOT = ZERO AT END STATEMENT
An END IF statement is missing. From CFT. Class, fatal.

ILLEGAL ARITHMETIC EXPRESSION
One of the operands in an arithmetic expression is of type LOGICAL. This is a violation of the ANSI FORTRAN standard. From CFT. Class, fatal.

ILLEGAL BY VALUE CALL
A by-value function call requires more than seven S or V registers to pass all of its arguments. A by-value call cannot use more than seven registers. Either reduce the number to seven or less, or pass the arguments by address instead of by value. From CFT. Class, fatal.

ILLEGAL CHARACTER
A nonstandard FORTRAN character, misplaced character, or syntax error has been encountered. From CFT. Class, fatal.

ILLEGAL CHARACTER EXPRESSION
A character assignment statement, *v=e*, is illegal because *v* is used in expression *e*. Rewrite the assignment statement so that no character position being defined in *v* is referenced in *e*. From CFT.

ILLEGAL CHARACTER OR MISSING DIMENSION
Either the statement contains an illegal character or an array element has not been defined by a DIMENSION statement. From CFT. Class, fatal.

ILLEGAL CHARACTERS IN NAME FIELD
Illegal characters are in a field that must contain a symbolic name. From CFT. Class, fatal.

ILLEGAL CHARACTERS IN NAME FIELD IN EQUIVALENCE AT SEQUENCE "*number*"
Illegal characters are in an EQUIVALENCE field that must contain a symbolic name. From CFT. Class, fatal.

ILLEGAL CHARACTERS IN STATEMENT NUMBER FIELD
Non-numeric characters appear in what should be a numeric field. From CFT. Class, fatal.

ILLEGAL COMMON BLOCK NAME
The specification of a common block name does not conform to the rules for constructing symbolic names. From CFT. Class, fatal.

ILLEGAL COMPILER DIRECTIVE
CDIR$ omitted or misspelled in columns 1 through 5, 6 not blank or 0, compiler directive not in columns 7 through 72, or compiler directive is misspelled or does not exist for CFT. From CFT. Class, fatal.

ILLEGAL CONDITIONAL STATEMENT
The conditional statement of a logical IF must not be a logical IF statement or a block IF statement. From CFT. Class, fatal.

ILLEGAL CONTINUATION
More than 19 consecutive continuation cards encountered or the first card of a program unit is a continuation card. From CFT. Class, fatal.

ILLEGAL CONTROL OPTION
An option in the control list of an I/O statement is incorrect. From CFT. Class, fatal.

ILLEGAL CONVERSION IN DATA STATEMENT
The types of a variable and an associated constant in a DATA statement differ. The type conversion required is illegal or undefined. From CFT. Class, fatal.

ILLEGAL CPU TYPE FOR EMA OPTION
The EMA option was specified on the CFT control statement for a target machine that cannot extend memory addressing. When using extended memory, the target machine must be an XMP-4 or follow-on machine. From CFT. Class, fatal.

ILLEGAL DO INDEX
DO variable is not an integer, real, or double-precision variable. From CFT. Class, fatal.

ILLEGAL DO TERMINATOR
DO-loops must not terminate on unconditional transfer statements. From CFT. Class, fatal.

ILLEGAL DO VARIABLE OR PARAMETER TYPE
A DO variable or parameter is not of type integer, real, double-precision, or Boolean. From CFT. Class, fatal.

ILLEGAL FORMAT NAME
A format identifier cannot be recognized as a statement label or the name of an array. From CFT. Class, fatal.

ILLEGAL IMPLICIT STATEMENT ARGUMENTS
IMPLICIT statement argument is not an alphabetic character or the range of characters specified is illegal. From CFT. Class, fatal.

ILLEGAL LOGICAL EXPRESSION
One of the operands in a logical expression is not of type LOGICAL. This is a violation of the ANSI FORTRAN standard. From CFT. Class, fatal.

ILLEGAL MASKING OR BOOLEAN EXPRESSION
One or both of the operands in a masking or Boolean expression is of type DOUBLE PRECISION or COMPLEX. Masking and Boolean expression operands must be single word entities. From CFT. Class, fatal.

ILLEGAL MIX OF CHARACTER AND NONCHARACTER IN COMMON BLOCK "*name*"
It is illegal to mix character and noncharacter entities in the same common block. From CFT. Class, fatal.

ILLEGAL MIX OF CHARACTER AND NONCHARACTER IN EQUIVALENCE AT SEQUENCE "*number*"
It is illegal to mix character and noncharacter entities in the same EQUIVALENCE statement. From CFT. Class, fatal.

ILLEGAL MIXED MODE OR CONVERSION
The types of two operands in an expression are incompatible or the type of array element or variable being defined is incompatible with the type of expression being evaluated. From CFT. Class, fatal.

ILLEGAL NUMBER IN NAME FIELD
A symbolic name must not begin with a number. From CFT. Class, fatal.

ILLEGAL OR DUPLICATE PARAMETER DEFINITION
Symbolic name of type integer, real, double-precision, or complex not followed by an arithmetic expression. Symbolic name of type logical not followed by a logical expression. A symbolic name has been assigned more than once in the same program unit. From CFT. Class, fatal.

ILLEGAL PLACEMENT OF ALIGN, DIRECTIVE IGNORED
The ALIGN compiler directive was not placed immediately before a DO statement, a statement with a referenced statement label, a PROGRAM statement, a SUBROUTINE statement, a FUNCTION statement, or an ENTRY statement. The directive will be ignored. From CFT. Class, warning.

ILLEGAL POINTEE "*name*"
A pointee cannot be a dummy argument or a pointer. It cannot be equivalenced or be specified in a common block statement. From CFT. Class, fatal.

ILLEGAL POINTER VARIABLE "*name*"
A pointer must be a simple variable. It cannot appear in an EQUIVALENCE statement. If defined in a PARAMETER or DATA statement, the definition must not precede its definition as a pointer. From CFT. Class, fatal.

ILLEGAL RELATIONAL EXPRESSION
One or both of the operands in a relational expression is of an illegal type for a relational expression. The most common of these errors is comparing LOGICAL values with ".EQ." or ".NE.". The logical operators ".EQV." or ".NEQV." should be used in these cases. From CFT. Class, fatal.

ILLEGAL STATEMENT LABEL IN IO CONTROL LIST
A 1- to 5-digit statement number is missing after END= or ERR=. From CFT. Class, fatal.

ILLEGAL STATEMENT SEQUENCE
An improper sequence of statement types has been encountered (for example, a GO TO statement followed by a DIMENSION statement). From CFT. Class, fatal.

ILLEGAL STATEMENT TYPE
A statement keyword is misspelled (for example, DIMENSOIN) or is otherwise unidentifiable. From CFT. Class, fatal.

ILLEGAL STATEMENT TYPE IN BLOCK DATA SUBPROGRAM
A statement appears in a BLOCK DATA subprogram which is prohibited by the ANSI FORTRAN standard, that is, an INTRINSIC or EXTERNAL statement. From CFT. Class, fatal.

ILLEGAL SUBSCRIPT TYPE "*name*"
A subscript expression is not of type integer or contains a constant that exceeds 2^{24}-1. From CFT. Class, fatal.

ILLEGAL SUBSTRING
A substring for a character item is incorrectly formed; or an attempt was made to use a substring with an entity which cannot have a substring (such as a character constant). From CFT. Class, fatal.

ILLEGAL SYNTAX IN NAMELIST
Illegal element found in NAMELIST statement. From CFT. Class, fatal.

ILLEGAL TYPE FOR ASSIGNED VARIABLE
A variable referenced in an ASSIGN statement is not of type integer. From CFT. Class, fatal.

ILLEGAL TYPE LENGTH
Length specified is not allowed for this data type. From CFT. Class, fatal.

ILLEGAL UNIT SPECIFIER
The unit specifier for INQUIRE must be an integer expression. From CFT. Class, fatal.

ILLEGAL USE OF "*name*"
Group name referenced previous to its definition in a NAMELIST statement. From CFT. Class, fatal.

ILLEGAL USE OF "*name*" IN IO LIST
External, function, or program name not permitted in an I/O list. From CFT. Class, fatal.

ILLEGAL USE OF ** IN CONSTANT EXPRESSION
A constant expression specifies exponentiation to a noninteger power. From CFT. Class, fatal.

ILLEGAL USE OF ASSUMED CHARACTER LENGTH
Character entity with a length of * must be a dummy argument, the symbolic name of a constant or an external function, whose name appears in a FUNCTION or ENTRY statement within the same program unit. From CFT. Class, fatal.

ILLEGAL USE OF ASSUMED SIZE ARRAY "*name*"
An array with an asterisk for the last dimension cannot be used without subscripts in an I/O statement. From CFT. Class, fatal.

ILLEGAL USE OF COLON
A colon can only be used in a FORMAT statement or to separate the lower and upper dimensions in a declarative. From CFT. Class, fatal.

ILLEGAL USE OF DUMMY ARGUMENT "*name*"
A dummy argument in a procedure subprogram cannot be named the same as a local variable or another dummy argument. From CFT. Class, fatal.

ILLEGAL USE OF DUMMY ARGUMENT "*name*" IN EQUIVALENCE AT SEQUENCE "*number*"
Dummy arguments cannot appear in EQUIVALENCE. From CFT. Class, fatal.

ILLEGAL USE OF FUNCTION "*name*"
A function name cannot be used as an array name. From CFT. Class, fatal.

ILLEGAL USE OF FUNCTION "*name*" IN EQUIVALENCE AT SEQUENCE "*number*"
A function name cannot be used as an identifier in an EQUIVALENCE statement. From CFT. Class, fatal.

ILLEGAL USE OF NAMELIST GROUP "*name*"
Use of namelist group name outside of namelist read or write. From CFT. Class, fatal.

ILLEGAL USE OF TASK COMMON
The named common block was declared as both a task common block and a regular common block in the same subprogram. Declare the named common block to be regular or task but not both. From CFT. Class, fatal.

ILLEGAL USE OF TASK COMMON VARIABLE
A task common variable was used illegally in a DATA, NAMELIST I/O, or SAVE statement. Remove the use of task common variable from the illegal statement or do not declare the variable as a task common variable. From CFT. Class, fatal.

ILLEGAL VALUE IN CONSTANT EXPRESSION
The evaluation of a constant expression yields a result that is out of range. From CFT. Class, fatal.

IMPLICIT NONE MUST BE ONLY IMPLICIT STATEMENT
IMPLICIT NONE or IMPLICIT SKOL was specified and another IMPLICIT statement appeared as well. From CFT. Class, fatal.

IMPROPERLY NESTED DO LOOP
Inner DO-loop is not contained entirely within the outer DO-loop range. From CFT. Class, fatal.

INCORRECT ARGUMENT TYPE
Actual argument is the wrong type in a function reference. From CFT. Class, fatal.

INPUT FILE EMPTY
An end-of-file record was encountered as the first record of the source input dataset. From CFT. Class, fatal.

INTEGER *2 = 24 BIT INTEGER
Integer *2 is implemented as a 24-bit integer by CFT. From CFT. Class, warning.

INTEGER CONSTANT EXPECTED WHERE *char* OCCURS
When parsing a FORMAT edit descriptor field, the character *char* appears where an integer constant is expected. From CFT. Class, fatal.

INTEGER CONSTANT EXPRESSION REQUIRED
Subscript or substring expression is not an integer constant expression. Use an integer constant expression. From CFT. Class, fatal.

INTRINSIC FUNCTION *name* IS NONSTANDARD
The function specified by *name* is a CFT intrinsic function not specified by the ANSI standard. CFT will use the intrinsic version unless the function is declared EXTERNAL. This message is issued as NON-ANSI if ANSI is specified on the control statement, and the function in question has been affirmed as an intrinsic function in an INTRINSIC statement. From CFT. Class, caution; non-ANSI.

INTRINSIC FUNCTION "*name*" CANNOT BE ACTUAL AGRUMENT
Certain intrinsic functions cannot be passed to subprograms as actual arguments. From CFT. Class, fatal.

INTRINSIC FUNCTION TYPE USED WITH ILLEGAL ARGUMENT
The actual argument to the intrinsic function is incorrect. From CFT. Class, fatal.

IO CONTROL LIST SPECIFIER MUST BE CHARACTER EXPRESSION
The I/O control list specifier must be evaluated to a character value. From CFT. Class, fatal.

IO CONTROL LIST SPECIFIER MUST BE CHARACTER VARIABLE OR ARRAY ELEMENT
The I/O control list specifier can be a character variable or an array element. From CFT. Class, fatal.

IO CONTROL LIST SPECIFIER MUST BE INTEGER EXPRESSION
The I/O control list specifier must be evaluated to an integer value. From CFT. Class, fatal.

IO CONTROL LIST SPECIFIER MUST BE INTEGER VARIABLE OR ARRAY ELEMENT
The I/O control list specifier can be an integer variable or an array element. From CFT. Class, fatal.

IO CONTROL LIST SPECIFIER MUST BE LOGICAL VARIABLE OR ARRAY ELEMENT
The I/O control list specifier can be a logical variable or an array element. From CFT. Class, fatal.

LAST ARRAY ONLY PARTIALLY INITIALIZED
The last element in a DATA statement variable list is an unsubscripted array and not enough constants are specified to completely fill the array. Remaining elements of the array are not initialized. From CFT. Class, warning; non-ANSI.

LEFT PARENTHESIS EXPECTED
A required opening parenthesis has been omitted. From CFT. Class, fatal.

LEFT PARENTHESIS EXPECTED IN EQUIVALENCE AT SEQUENCE "*number*"
A required opening parenthesis has been omitted in an EQUIVALENCE statement. From CFT. Class, fatal.

LINE LENGTH > 133 CHARACTERS
One or more lines exceed 133 characters during FORMAT statement editing. From CFT. Class, caution.

LIST DIRECTED IO ILLEGAL FOR INTERNAL FILE
List-directed reads and writes are illegal operations on internal files. Internal file IO must be formatted. From CFT. Class, fatal.

LOGICAL OPERATOR MUST END IN PERIOD
A period does not follow an otherwise correct logical operator. From CFT. Class, fatal.

LOSS OF PRECISION IN TYPE CONVERSION
The type of a variable and the type of the associated constant in a DATA statement differ. The constant is converted to the type of the variable and precision is lost. From CFT. Class, caution.

LOWERCASE CHARACTERS IN KEYWORDS OR IDENTIFIERS ARE NONSTANDARD
At least one lowercase alphabetic character which is not part of a character constant or comment appears in a program unit. This lowercase alphabetic character may be in a keyword, identifier, or control character such as a Hollerith character constant descriptor, or may be a lower case 'c' beginning a comment. Lowercase characters are not provided for in the ANSI FORTRAN Standard. This message is issued only once in a program unit that contains lowercase characters. From CFT. Class, informative; NON-ANSI.

LOWERCASE CHARACTERS USED AS EDIT DESCRIPTORS ARE NONSTANDARD
When parsing a FORMAT specified list, CFT encountered at least one lower case character used as an edit descriptor. The ANSI standard character set does not include lower case characters. From CFT. Class, informative, NON-ANSI.

MAIN PROGRAM MUST BE NAMED FOR FLOW TRACE
Main program must be named if flowtrace is enabled. From CFT. Class, fatal.

MASKING OR BOOLEAN EXPRESSION IS NONSTANDARD
A masking or Boolean expression has been detected by CFT. These expressions are not provided for in the ANSI FORTRAN standard. From CFT. Class, informative; non-ANSI.

MAXIMUM LEGAL ITERATION COUNT EXCEEDED
A DO loop trip count has been computed to be greater than the allowable maximum of $2^{23} - 1$. From CFT. Class, fatal.

MINIMUM ONE PASS DO-LOOPS ARE NONSTANDARD
The control card option ON=J was selected, causing all DO-loops to execute at least once. This is a CFT extension not provided for in the ANSI 77 FORTRAN Standard. This message is issued for all DO-loops in a program compiled with ON=J. From CFT. Class, informative; NON-ANSI.

MISSING =
An equal sign is missing in a PARAMETER or statement function definition statement. From CFT. Class, fatal.

MISSING = IN CONTROL LIST
An equal sign is missing after an option in an I/O statement control list. From CFT. Class, fatal.

MISSING COLON
A required colon has been omitted (for example, in a character substring). From CFT. Class, fatal.

MISSING END STATEMENT
The last or only program unit being compiled lacks an END statement in its last line. From CFT. Class, warning; non-ANSI.

MISSING OR ILLEGAL CONSTANT LIST
PARAMETER or DATA statement has not specified a constant list or a list has a missing separator. From CFT. Class, fatal.

MISSING OR ILLEGAL STATEMENT NUMBER IN DO
Statement number is missing or it contains illegal characters in a DO statement. From CFT. Class, fatal.

MISSING OR ZERO COUNT FOR HOLLERITH STRING
The count field for a Hollerith edit descriptor is missing or zero in a FORMAT specifier list. From CFT. Class, fatal.

MISSING RIGHT PARENTHESIS OR UNEXPECTED END OF FORMAT
When parsing a FORMAT specifier list, CFT reached the end of the FORMAT statement before it expected to. This can occur when the parentheses are unbalanced or when a Hollerith string count is too large and consumes the closing parenthesis at the end of a FORMAT statement. From CFT. Class, fatal.

MISSING STATEMENT NUMBER IN ASSIGN
An ASSIGN statement lacks a statement label reference. From CFT. Class, fatal.

MISSING TO IN ASSIGN STATEMENT
An ASSIGN statement requires the keyword extension TO. From CFT. Class, fatal.

MODIFICATION OF DO CONTROL VARIABLE WITHIN LOOP IS NONSTANDARD
CFT detected the modification of the DO-loop control variable inside the DO-loop. This is allowed by CFT, but not allowed by the ANSI 77 FORTRAN Standard. From CFT. Class, informative; non-ANSI.

MORE THAN 511 DISTINCT CHARACTER LENGTHS DECLARED IN THIS PROGRAM UNIT
The user cannot have more than 511 distinct character lengths declared in a program unit. These lengths include character variables, character constants, and character temporaries.

MORE THAN 312 DUMMY ARGUMENTS IN PROGRAM UNIT
CFT does not accept more than 312 dummy arguments in a subroutine or function subprogram. Each argument to each entry point in a program unit counts as a separate argument for computing the number of arguments used as a program unit. From CFT.

MORE THAN ONE ELSE STATEMENT AT THIS IF-LEVEL
Only one ELSE statement is permitted per IF-level. From CFT. Class, fatal.

MORE THAN ONE UN-NAMED BLOCK DATA SUBPROGRAM IS NONSTANDARD
More than one unnamed BLOCK DATA subprogram appears in a compilation. CFT allows up to 26 unnamed BLOCK DATA subprograms per compilation. The ANSI FORTRAN standard allows only one. From CFT. Class, informative; non-ANSI.

NAME LONGER THAN EIGHT CHARACTERS
A symbolic name must not contain more than eight characters. From CFT. Class, fatal.

NO BLOCK IF ASSOCIATED WITH ELSE STATEMENT
An ELSE statement must follow a block IF statement and precede an END IF statement of the same level. From CFT. Class, fatal.

NO BLOCK IF ASSOCIATED WITH END IF STATEMENT.
An END IF statement must be uniquely associated with an IF(*e*)THEN statement of the same IF-level. From CFT. Class, fatal.

NO PATH TO THIS STATEMENT
The previous statement is an unconditional transfer, and this statement has no statement number. From CFT. Class, caution.

NONSTANDARD "*name*" SPECIFIER
A CFT extended form of a unit or format specifier appears in an I/O control list. This form is not allowed in the ANSI 77 FORTRAN Standard. From CFT. Class, informative; non-ANSI.

NONSTANDARD "*name*" STATEMENT SYNTAX
An extended form of an ANSI 77 FORTRAN Standard statement type indicated by "*name*" was used in a program. From CFT. Class, informative; non-ANSI.

NONSTANDARD ARITHMETIC EXPRESSION
An arithmetic expression is formed with operand types not provided for in the ANSI FORTRAN standard. An example of this is adding an INTEGER variable to a CHARACTER constant, which is a CFT extension. From CFT. Class, informative; non-ANSI.

NONSTANDARD BLOCK DATA STATEMENT SYNTAX
Parameters appear on a BLOCK DATA statement. These are a CFT extension to the ANSI FORTRAN standard. From CFT. Class, informative; non-ANSI.

NONSTANDARD DIMENSION DECLARATOR
A dimension declarator expression contains non-INTEGER constants or variables, or function references. The ANSI FORTRAN standard specifies only INTEGER variables and constants may be used in a dimension declarator expression. From CFT. Class, non-ANSI.

NONSTANDARD EDIT DESCRIPTOR FIELD
An edit descriptor that is not provided for in the ANSI standard or an extended form of a standard edit descriptor was used in a FORMAT specifier list. From CFT. Class, informative, non-ANSI.

NONSTANDARD OPERATOR *name*
An operator which is not provided for in the ANSI FORTRAN standard has been used, for example, .XOR.. These operators are CFT extensions to the FORTRAN language. From CFT. Class, informative; non-ANSI.

NONSTANDARD RELATIONAL EXPRESSION
A relational expression compares a pair of operands in a manner not provided for in the ANSI FORTRAN standard. An example of this is comparing a CHARACTER constant to an INTEGER variable, which is a CFT extension. From CFT. Class, informative; non-ANSI.

NONSTANDARD STRING DELIMITER
CFT allows string constants to be delimited by quotes in place of apostrophes. Asterisks may also be used in FORMAT specifier lists. Neither quotes nor asterisks are provided for in the ANSI standard. From CFT. Class, informative, non-ANSI.

NONSTANDARD TYPE DECLARATION
A TYPE * BYTE COUNT type declaration or nonstandard IMPLICIT statement, such as an IMPLICIT NONE statement appears in a program unit, or a DOUBLE declaration type statement is used in place of a DOUBLE PRECISION type statement. These are CFT extensions to the ANSI 77 FORTRAN Standard. From CFT. Class, informative; non-ANSI.

NOT ENOUGH DO PARAMETERS
Fewer than two arguments have been encountered after the equal sign in a DO statement. From CFT. Class, fatal.

NOT ENOUGH MEMORY TO COMPILE
The program unit is too long to compile in the available memory. From CFT. Class, fatal.

OCTAL CONSTANT IS NONSTANDARD
Octal constants are not provided for in the ANSI FORTRAN standard. They are a CFT extension to the FORTRAN language. From CFT. Class, informative; non-ANSI.

OPTIMIZATION BLOCK BROKEN AT THIS POINT
The size of the code forced CFT to terminate an optimization block at this point. A new optimization block begins with the next statement. No action is necessary. However, if the break occurs in an inner DO-loop, vectorization is prevented. Converting a large loop into two or more smaller loops might be beneficial. The MAXBLOCK directive can also be used. From CFT. Class, comment.

PARAMETER USED TWICE IN STATEMENT FUNCTION PARAMETER LIST
A given symbolic name can appear only once in a single dummy argument list. From CFT. Class, fatal.

PARENTHESES NESTED TOO DEEPLY
CFT places a limit of 9 on the number of nested parentheses allowed in a FORMAT specifier list. This limit has been exceeded. From CFT. Class, fatal.

PASS TWO SKIPPED BECAUSE OF FATAL PASS ONE ERRORS
Pass two of CFT compilation will be skipped for this program unit due to fatal pass one errors. Class, warning.

PERIOD EXPECTED WHERE *char* OCCURS
When parsing a FORMAT edit descriptor field the character *char* appears where a period is expected. From CFT. Class, fatal.

PLEASE RERUN WITH SMALLER VALUE FOR MAXBLOCK
The value for MAXBLOCK exceeds the system default. Follows an error message from the internal compiler, such as COMPILER ERROR. Rerun with a smaller MAXBLOCK value. Break large loops into smaller loops. See a Cray Research analyst if the error persists. From CFT. Class, informative.

POINTER MUST BE TYPE INTEGER
A pointer variable must not be assigned a type other than integer. From CFT. Class, fatal.

POSSIBLE BRANCH INTO BLOCK IF VIA ASSIGN OR END=/ERR= WITH LABEL *l*
Label *l* is a FORTRAN statement number which is defined in an IF, ELSEIF, or ELSE block and which has appeared in an ASSIGN statement, or in an END= or ERR= branch of an I/O statement. Branches into IF, ELSEIF or IF blocks are prohibited by the ANSI FORTRAN standard. From CFT. Class, caution; non-ANSI.

POSSIBLE BRANCH INTO INACTIVE DO LOOP, STATEMENT LABEL '*xx*'
CFT has detected a statement label "*xx*," which is defined in the range of a DO-loop and referenced in a branch statement outside the range of the DO-loop. From CFT. Class, warning.

POSSIBLE BRANCH INTO INACTIVE DO LOOP VIA ASSIGN OR END=/ERR= WITH LABEL "*xx*"
"*xx*" is a statement label defined within the range of a DO-loop. It has appeared in an ASSIGN statement or in the END= or ERR= branch of an I/O statement in the program unit. Verify that the branches to these labels occur only from within the range of the innermost DO-loop where the labels are defined. From CFT. Class, caution; non-ANSI.

PREVIOUS IMPLICIT REFERENCES THIS CHARACTER
Only one IMPLICIT reference is permitted per character. From CFT. Class, fatal.

PREVIOUS REFERENCES TO "*name*"
An ENTRY name was used before its declaration as an ENTRY. From CFT. Class, fatal.

PROGRAM UNIT TOO LARGE TO COMPILE
One of CFT's internal tables has overflowed because there was too much code in a program unit. Break the program unit into smaller subroutines. If the array bounds checking option (ON=O) has been selected, it can be turned off, or the BOUNDS compiler directive can be used to check arrays. (The O option generates a large amount of code.) From CFT. Class, fatal.

REAL*8 = SINGLE PRECISION
REAL*8 is implemented as single-precision by CFT. From CFT. Class, warning.

RECURSIVE SUBROUTINE OR FUNCTION REFERENCE
The function, subroutine, or entry name was referenced within the same program unit that defined it. From CFT. Class, non-ANSI.

RECURSIVE SUBROUTINE REFERENCE USED AS AN ARGUMENT
The main subroutine name was used as an argument to another function or subroutine call. From CFT. Class, non-ANSI.

REFERENCES TO ARRAY "*name*" WITH NO SUBSCRIPTS
The array named was referenced without subscripts in a statement that required them. From CFT. Class, fatal.

RELATIONAL EXPRESSION WITH DOUBLE PRECISION AND COMPLEX IS NONSTANDARD
A relational expression has been detected with a COMPLEX and a DOUBLE PRECISION operand. CFT will convert the DOUBLE PRECISION operand to COMPLEX. The ANSI FORTRAN standard does not provide for such conversions. From CFT. Class, informative; non-ANSI.

REPETITION COUNT ILLEGAL FOR *name*
A repetition count appears before the non-repeatable edit descriptor *name* in a FORMAT specifier list. From CFT. Class, fatal.

REPETITION COUNT MUST BE > ZERO
The repetition count before a repeatable edit descriptor in a FORMAT specifier list is zero. From CFT. Class, fatal.

REPETITION COUNT TOO LARGE
The value of *n* in the -*n*X edit descriptor field moves the next character position to the left of the first position. From CFT. Class, fatal.

RETURN ILLEGAL IN MAIN PROGRAM
A RETURN statement is encountered in a main program unit. From CFT. Class, fatal.

RIGHT PARENTHESIS EXPECTED
A required closing parenthesis has been omitted. From CFT. Class, fatal.

RIGHT PARENTHESIS EXPECTED IN EQUIVALENCE AT SEQUENCE "*number*"
A required closing parenthesis has been omitted in an EQUIVALENCE statement. From CFT. Class, fatal.

SCALAR DUMMY ARGUMENT "*name*" USED AS FORMAT IDENTIFIER
The named integer variable appears both as a format identifier and as an entry in a dummy argument list. Check for missing DIMENSION statement. From CFT. Class, fatal.

SCAN STOPPED, TOO MANY ERRORS IN FORMAT
CFT attempts recovery of up to three errors before abandoning the FORMAT statement. From CFT. Class, fatal.

SPECIFIER RECL LEGAL IF AND ONLY IF ACCESS IS DIRECT
Access was not specified as direct in an OPEN statement. From CFT. Class, fatal.

STATEMENT FUNCTION "*name*" IN COMMON OR ARGUMENT LIST
Statement function must not appear as a variable in a COMMON block or an argument list. From CFT. Class, fatal.

STATEMENT FUNCTION "*name*" REFERENCES ITSELF
A statement function definition statement cannot be recursive. From CFT. Class, fatal.

STATEMENT FUNCTION PARAMETER MUST NOT BE ARRAY
The names of variables appearing as dummy arguments of a statement function have a scope of that statement only. From CFT. Class, fatal.

STATEMENT LABEL IGNORED
Statement label is ignored because transfer to this statement is prohibited. From CFT. Class, fatal.

STATEMENT LENGTH EXCEEDED
The statement is too long to fit into CFT's internal format, either as it is written or after arithmetic functions have been expanded. Divide the statement into smaller segments. Example: Change
A=B*C*D/E to
```
    TEMP=B*C*D
    A=TEMP/E.
```
From CFT. Class, fatal.

STATEMENT NUMBER ILLEGAL ON DECLARATIVE
CFT does not allow a statement number on ENTRY statements. From CFT. Class, fatal.

STATEMENT NUMBER ON BLANK LINE IGNORED
Blank lines cannot contain statement labels. From CFT. Class, fatal.

SUBROUTINE "*name*" NOT DECLARED
IMPLICIT NONE or IMPLICIT SKOL was specified and the program unit contains a call of *name* but *name* did not appear in an EXTERNAL statement. From CFT. Class, fatal.

SUBSCRIPT OUT OF DIMENSION BOUNDS IN EQUIVALENCE AT SEQUENCE "*number*"
The subscript exceeds the value given in the dimensions. From CFT. Class, fatal.

SUBSTRING EXPRESSION OUT OF BOUNDS
In a substring specifier (C1:C2), the relations $1 \leq C1 \leq C2 \leq LEN$ are not all true (where LEN is the declared length of the character entity). Ensure that $1 \leq C1 \leq C2 \leq LEN$. From CFT. Class, fatal.

SYNTAX ERROR
Illegal element, name where number required, or extra or missing punctuation. From CFT. Class, fatal.

SYNTAX ERROR IN ENCODE OR DECODE STATEMENT
Illegal element in ENCODE or DECODE statement. From CFT. Class, fatal.

SYNTAX ERROR IN IMPLIED DO
An implied-DO list specified in a DATA statement is of improper syntactical form; a variable referenced is not an implied-DO variable; or an array element referenced does not specify the implied-DO variable for this implied-DO list in its subscript. From CFT. Class, fatal.

SYNTAX ERROR IN IO CONTROL LIST
Illegal element in I/O control list. From CFT. Class, fatal.

TAB COUNT MUST NOT BE ZERO
A tab edit descriptor (T, TL, or TR) appears in a FORMAT specifier list followed by a tab count of 0. From CFT. Class, fatal.

TASK COMMON BLOCK "*name*" IS STATICALLY ALLOCATED
A task common block was declared when the allocation specification was defined as STATIC. Use ALLOC=STACK on the CFT control card or do not declare any task common blocks. From CFT. Class, warning.

TASK COMMON MUST BE NAMED
A blank common block was declared with the CFT extension keyword 'TASK'. Make the blank common block a named common block. From CFT. Class, fatal.

TEST EXPRESSION MUST BE LOGICAL
Expression type in a logical IF must be logical or Boolean. From CFT. Class, fatal.

TEST EXPRESSION MUST NOT BE CHARACTER
The expression type in a logical or arithmetic IF is a character. From CFT. Class, fatal.

TEST EXPRESSION MUST NOT BE LOGICAL
Expression type in an arithmetic IF must not be logical. From CFT. Class, fatal.

TOO MANY COMMON BLOCKS DECLARED
More than 120 distinct common blocks were declared in a single program unit. Reduce the number of common blocks declared in the offending program unit to 120 or less. From CFT. Class, CFT compile-time fatal error.

TOO MANY DO PARAMETERS
More than three arguments were encountered after the equal sign in a DO statement. From CFT. Class, fatal.

TOO MANY DOS ON STATEMENT
More than 15 DO-loops ended on the same statement. From CFT. Class, fatal.

TOO MANY POINTERS DECLARED
More than 312 pointers were declared in a single program unit. From CFT. Class, warning.

TOO MANY SUBSCRIPTS
An array reference has more subscripts than were declared. From CFT. Class, fatal.

TOO MANY SUBSCRIPTS IN EQUIVALENCE AT SEQUENCE "*number*"
An array reference in an EQUIVALENCE statement has more subscripts than were declared. From CFT. Class, fatal.

TOO MANY UN-NAMED BLOCK DATA SUBPROGRAMS
The ANSI FORTRAN standard allows only one unnamed BLOCK DATA subprogram to be used in a program. CFT allows 26 unnamed BLOCK DATA subprograms. More than 26 such subprograms have appeared during the compilation. From CFT. Class, fatal.

TWO BRANCH IF STATEMENT IS NONSTANDARD
A 2-branch arithmetic or logical IF statement appears in a program. These statements are CFT extensions to the ANSI 77 FORTRAN standard. From CFT. Class, informative; non-ANSI.

TYPE CONVERSION IN DEFINITION
A constant in a PARAMETER statement was not converted to the type of the corresponding symbolic name. From CFT. Class, fatal.

TYPE OF "*name*" NOT DECLARED
name was declared in an EXTERNAL statement, but did not appear in an explicit type statement. From CFT. Class, caution.

TYPE STATEMENT IGNORED FOR INTRINSIC FUNCTION *name*
Type statements do not change the type of an intrinsic function and are ignored. From CFT. Class, caution.

UNBALANCED PARENTHESIS
Opening and closing parentheses do not match; required parenthesis is not present. From CFT. Class, fatal.

UNDEFINED ITEM IN CONSTANT EXPRESSION
A constant expression in a PARAMETER or DATA statement is not specified with constants, the symbolic names of constants, or, in a DATA statement, with the names of implied-DO variables. From CFT. Class, fatal.

UNDEFINED STATEMENT NUMBER *"number"*
A referenced statement label is not defined. From CFT. Class, fatal.

UNEXPECTED CHARACTER IN FORMAT
When a FORMAT specifier list was being parsed, an unknown edit descriptor or premature end of an edit descriptor was found. CFT will attempt to recover at the next character following a comma, or at the next parenthesis or string delimiter. From CFT. Class, fatal.

UNEXPECTED END OF STATEMENT
A statement encountered is syntactically incomplete. From CFT. Class, fatal.

UNIT=* ILLEGAL FOR DIRECT ACCESS
[UNIT=]* appeared in a direct access READ or WRITE statement. From CFT. Class, fatal.

UNIT=* ILLEGAL FOR UNFORMATTED IO
[UNIT=]* appeared without a format identifier in a READ or WRITE statement. From CFT. Class, fatal.

UNIT=* LEGAL ONLY IN READ OR WRITE
[UNIT=]* appeared in an auxiliary I/O statement. From CFT. Class, fatal.

UNKNOWN LOGICAL OPERATOR
The characters following a period do not represent a logical operator. From CFT. Class, fatal.

UPPER DIMENSION < LOWER DIMENSION
The lower dimension is not less than or equal to the upper dimension. From CFT. Class, fatal.

USE OF END ILLEGAL IN WRITE CONTROL LIST
END= was illegally specified in a WRITE statement. From CFT. Class, fatal.

VALUE NOT ASSIGNED TO FUNCTION NAME
A value statement for a function is missing in a function subprogram. From CFT. Class, fatal.

VARIABLE "*name*" USED AS ARRAY OR FUNCTION
A simple variable is referenced with either subscripts or an argument list. From CFT. Class, fatal.

VARIABLE DIMENSION ARRAY "*name*" MUST BE DUMMY ARGUMENT
A variably dimensioned array did not appear as a dummy argument at an entry point. From CFT. Class, fatal.

VARIABLE DIMENSION ILLEGAL FOR ARRAY IN COMMON
An array named in a COMMON statement has a subscript that is not a constant. Check for a missing PARAMETER statement. From CFT. Class, fatal.

VARIABLE LIST LONGER THAN CONSTANT LIST
Constants and variables must correspond one-to-one in a DATA statement. From CFT. Class, fatal.

VERY LARGE LOCAL DATA BLOCK; USE EXTENDED MEMORY ADDRESSING OPTION
Very large local arrays were declared, causing the generated code to end at more than two million words of memory. The large local arrays must be declared in a common block and the program must be compiled with the extended memory addressing(EMA) option specified on the CFT control statement. From CFT. Class, fatal.

VERY LARGE OFFSET ENCOUNTERED; USE EXTENDED MEMORY ADDRESSING OPTION
A calculated offset greater than four million words of memory was detected. The program must be compiled with the extended memory addressing(EMA) option specified on the CFT control statement. From CFT. Class, fatal.

ZERO SUBSCRIPT INCREMENT
A subscript CII must have a nonzero increment. From CFT. Class, fatal.

ZERO TO NEGATIVE POWER
Raising zero to a zero or negative power produces unpredictable results in an executable program. From CFT. Class, fatal.

2.2 MESSAGES NOT BEGINNING WITH VARIABLE NAME

ACCOUNT.
An ACCOUNT statement was processed. From CSP. Class, informative.

ACQUIRE CANCELLED BY COS.
The dataset is not available from the COS Dataset Catalog, and, due to a condition detected by COS, cannot be transferred from the specified station. This is typically a "disk full", "DSC full", "PDS

full", or "DAT full" condition encountered while attempting to transfer the dataset from the station, or attempting to SAVE the transferred dataset. From SCP. Class, informative.

ACQUIRE CANCELLED BY MF=*mf*. PDN=*pdn* ID=*id* ED=*ed*
The dataset is not available from the front end. From SCP. Class, informative.

ACQUIRE FROM MF=*mf* COMPLETE. PDN=*pdn* ID=*id* ED=*ed*
A dataset was acquired from the front end. From SCP. Class, informative.

ARGUMENT *n* IS *x*
A macro operation containing one or more # operators was matched to the source text while a %X+ or %T+ directive was in effect. (This message does not appear for any argument strings that are null.) This message appears on the master operator console only. From SKOL macro trace. Class, informative.

BLOCKS RECEIVED FROM FRONT END - *n*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

BLOCKS SENT TO THE FRONT END - *n*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

CALL ERROR *overlay*
$OVERLAY encountered an error while processing a call. Correct the request and resubmit the job. From $OVERLAY. Class, not reprievable.

CANNOT READ OVERLAY *overlay name*
The read issued to obtain an overlay failed. Retry the request. If the problem persists, recreate the overlay. From $OVERLAY. Class, not reprievable.

CANNOT READ SYSTEM DUMP AREA
ENTER GO TO TRY TO RE-ALLOCATE AREA
ENTER RETRY TO RE-READ
Startup received an error from DQM when attempting to read the reserved system dump area. Determine the cause of the error and correct the problem. If the error persists, the reserved area must be re-allocated. A reply of RETRY causes Startup to make another attempt to successfully read the dump area. A reply of GO causes Startup to attempt to allocate the system dump area to another portion of the master device. Sufficient space must exist on the master device for a second copy of the dump area, since the first is not deallocated. The master device label will be updated to indicate the new dump beginning allocation index. Any other reply causes the message to be re-issued. This message appears on the master operator console only. From Startup. Class, warning.

CLASS STRUCTURE ROLL DATASET NOT FOUND - NO RECOVERY POSSIBLE
Permanent Dataset Manager did not find the class structure roll dataset in the DSC. Run a job to establish the desired structure. Default structure is in effect. From job class recovery in Startup, system log only. Class, caution.

CATENATION STRING IS *x*
The macro pattern %'#'='#' was encountered in the source text (invoking the macro operator @MG) while a %D directive was in effect. From SKOL macro trace. Class, informative.

CLOSE CALLS - *n*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

*** CORRECTED DISK ERROR ***
A hardware disk checkword error was not recovered by retry but the data was corrected by the Disk Error Correction (DEC) system task. From DQM. Class, informative.

CPU PRIORITIES AFTER *x.x* CYCLES:

JOB	RHO	OLD CP	NEW CP	NAME
0	.0	.0	.0	*n*

CPU priority calculations are traced in the system log. From JSH. Class, informative.

DATASET LOST FOR JOB *jobname* NUMBER *jsq*
A disk error occurred on the Cray mainframe when dataset was sent to the front end. From SCP. Class, not reprievable.

DATASET NOT AVAILABLE FROM FRONT END
The dataset is not available from the front end. From SCP. Class, informative.

DISK BLOCKS MOVED - *n*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

DISPOSE CANCELLED BY COS. MF=*mf*. SDN=*sdn*
A disposed dataset was cancelled by COS. From SCP. Class, informative.

DISPOSE CANCELLED BY MF=*mf*. SDN=*sdn*
A disposed dataset was canceled by the front end. From SCP. Class, informative.

DN=SYSLOG FAILED ACCESS OF ED # *n*
An attempt was made by STATS to access an edition beyond the current one. Check the edition numbers available for retrieval and the correlation between the SDATE and ED parameters or the EDATE and LASTED parameters. From Stats. Class, informative.

EXPANSION IS *x*
A macro pattern match resulted in the insertion of new text into the expansion buffer while a %X, %X+, %T, or %T+ directive was in effect. (This message does not appear if the replacement string is null). From SKOL macro trace. Class, informative.

FL EXCEEDED *overlay name*
The specified overlay will not fit into the user's field length. Increase the memory for the job and resubmit. From OVERLAY. Class, not reprievable.

FOLLOWED BY *x*
A macro pattern match resulted in the insertion of new text into the expansion buffer while a %X, %X+, or %Y directive was in effect. The text displayed by this message represents the contents of the expansion buffer immediately following the inserted text. Only the first 20 characters are displayed; excess characters are represented by '...'. From SKOL macro trace. Class, informative.

FORTRAN -- *text*
A SKOL statement was completely processed and the resulting FORTRAN code was being output while a %X, %X+, or %O directive was in effect. This message represents a single FORTRAN statement including any necessary continuation cards. More than one FORTRAN statement, therefore more than one message can be generated by a single SKOL statement. From SKOL macro trace. Class, informative.

HARDWARE ERROR ON UNIT; *dn*
A disk error occurred during an attempt to read from or write to dataset *dn*. Class, fatal.

ILLEGAL DATASET *dataset name*
$OVERLAY detected that an illegal dataset was called. Correct the request and resubmit the job. From OVERLAY. Class, fatal.

INTERNAL COUNTER = *n*
An *n*@I= macro operation was executed while a %X, %X+, or %H directive was in effect. From SKOL macro trace. This message appears on the master operator console only. Class, informative.

IOP SUBSYSTEM IS DOWN
Central processing unit has detected that communication with the I/O Subsystem (IOS) has terminated. System log only. From MEP. Class, informative.

IOP SUBSYSTEM IS UP
The I/O Subsystem (IOS) has completed communication connection to central processing unit. System log only. From MEP. Class, informative.

IOP SUBSYSTEM LOGGING IS DISABLED
The I/O Subsystem (IOS) sent a message disabling hardware error detection. System log only. From MEP. Class, informative.

IOP SUBSYSTEM LOGGING IS ENABLED
The I/O Subsystem (IOS) sent a message enabling hardware error detection. System log only. From MEP. Class, informative.

LIST CANNOT HAVE AN ARGUMENT
A dataset or module name was encountered in a LIST directive. Keyword is the only parameter permitted. Correct the LIST directive. From BUILD. Class, fatal; job aborts.

LOGON MF=*mf*
Mainframe logon. From SCP. This message appears on the master operator console and system log only. Class, informative.

LOGOFF MF=*mf*
Mainframe logoff. This message appears on the master operator console and system log only. From SCP. Class, informative.

MACRO DEFINITION; PATTERN IS *x*
The macro pattern %'#'='#' (macro generation, invoking the operator @MG) or %'#'=' '#' (macro catenation, invoking the operator @MC) was encountered in the source text while a %D directive was in effect. From SKOL macro trace. Class, informative.

MAXIMUM EDITIONS REACHED
A dataset already exists with the same PDN and ID and the edition number is 4095. Delete or modify the 4095th edition. From PDM

MAXIMUM MEMORY USED - *n* WORDS
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

MEMORY INTEGRAL (EXECUTION TIME) *n*
MWDS-SEC
When the CHARGES program is run, this message issues if requested. From CHARGES. Class, informative.

MEMORY INTEGRAL (I/O WAIT TIME) - *n*
MWDS-SEC
When the CHARGES program is run, this message issues if requested. From CHARGES. Class, informative.

MEMORY PRIORITIES AFTER *x.x* CYCLES:
JOB DELTA TIT OLD MP NEW MP NAME STATUS
0 .0 .0 .0 *n* *s*
Memory priority calulations are traced in the system log. From JSH. Class, informative.

MEMORY RESIDENT DATASETS - *n*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

MINIMUM MEMORY USED - *n* WORDS
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

OPEN CALLS - *n*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

OPERATOR MESSAGE (*station operator text*)
Operator message issued to Station Call Processor from the front-end station. System log only. From SCP. Class, informative.

OVERLAY NOT FOUND *overlay name*
$OVERLAY was unable to find the specified overlay. Correct the request and resubmit the job. From OVERLAY. Class, fatal.

PATTERN MATCHED IS *x*
A macro operation was matched to the source text while a %X, %X+, or %T %T+ directive was in effect. From SKOL macro trace. Class, informative.

PERMANENT FILE SPACE ACCESSED - *n* BLOCKS
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

PERMANENT FILE SPACE SAVED - *n* BLOCKS
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

PHYSICAL I/O REQUESTS - *n*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

RECEIVED *jsqpdn* SZ = *words* SID = *fe* DID = *fe* TID = *tid* USERID = *acctnum*
Dataset was received from the front end. System log only. From SCP. Class, informative.

REPLACEMENT STRING IS *x*
The macro pattern %'#'='#' was encountered in the source text (invoking the macro operator @MG) while a %D directive was in effect. From SKOL macro trace. Class, informative.

SAVE CANCELLED BY COS. PDN=*pdn* ID=*id* ED=*ed*
An input dataset to be saved on the Cray system was cancelled by COS. System log only. From SCP. Class, informative.

SAVE CANCELLED BY MF=*mf*. PDN=*pdn* ID=*id* ED=*ed*
An input dataset to be saved on the Cray system was cancelled by the front end. From SCP. Class, informative.

SAVE FROM MF=*mf* COMPLETE. PDN=*pdn* ID=*id* ED=*ed*
A dataset was saved on the Cray system from the front end. From SCP. Class, informative.

STATION MESSAGE CANCEL MESSAGE TID = *tid*, ID = *id*, TASK = *tsknum*, MSG NUMBER = *msgnum* *text*...
A station message was cancelled by a system task. From SCP. System log only. Class, informative.

STATION MESSAGE INFORMATION ONLY TID = *tid*, ID = *id*, TASK = *tsknum*, MSG NUMBER = *msgnum* *text*...
A station message was issued from a system task. System log only. From SCP. Class, informative.

STATION MESSAGE REPLY TO MESSAGE TID = *tid*, ID = *id*, TASK = *tsknum*, MSG NUMBER = *msgnum* *text*...
A station message reply was issued from the front-end station. System log only. From SCP. Class, informative.

STATION MESSAGE RESPONSE REQUIRED TID = *tid*, ID = *id*, TASK = *tsknum*, MSG NUMBER = *msgnum* *text*...
A station message was received by a system task. System log only. From SCP. Class, informative.

STATION OPERATOR COMMAND CHANNEL, *c*, OFF
STATION OPERATOR COMMAND CHANNEL, *c*, ON
STATION OPERATOR COMMAND CLASS, ALL, OFF
STATION OPERATOR COMMAND CLASS, ALL, ON
STATION OPERATOR COMMAND CLASS, *jcn*, OFF
STATION OPERATOR COMMAND CLASS, *jcn*, ON
STATION OPERATOR COMMAND DEVICE, *d*, OFF
STATION OPERATOR COMMAND DEVICE, *d*, ON
STATION OPERATOR COMMAND DISCONNECT, *id*
STATION OPERATOR COMMAND DROP, *jsq*, JOBNAME = *jn*
STATION OPERATOR COMMAND ENTER, *jsq*, CLASS, *jcn*, JOBNAME = *jn*
STATION OPERATOR COMMAND ENTER, *jsq*, ID, *id*, *tid*, JOBNAME = *jn*
STATION OPERATOR COMMAND ENTER, *jsq*, PRIORITY, *pri*, JOBNAME = *jn*
STATION OPERATOR COMMAND ENTER, *jsq*, TIME, *time*, JOBNAME = *jn*
STATION OPERATOR COMMAND KILL, *jsq*, JOBNAME = *jn*
STATION OPERATOR COMMAND LIMIT, *lim*
STATION OPERATOR COMMAND OPERATOR, *nid*, *tid*, *pw*, *npw*
STATION OPERATOR COMMAND RECOVER
STATION OPERATOR COMMAND RERUN, *jsq*, JOBNAME = *jn*
STATION OPERATOR COMMAND RESUME, ALL
STATION OPERATOR COMMAND RESUME, *jsq*, JOBNAME = *jn*

STATION OPERATOR COMMAND ROUTE, *oid*, *nid*
STATION OPERATOR COMMAND SHUTDOWN
STATION OPERATOR COMMAND STREAM, *id*, *ni*, *no*, *na*
STATION OPERATOR COMMAND SUSPEND, ALL
STATION OPERATOR COMMAND SUSPEND, *jsq*, JOBNAME = *jn*
STATION OPERATOR COMMAND SWITCH, *jsq*, *ssw*, OFF
STATION OPERATOR COMMAND SWITCH, *jsq*, *ssw*, ON
The indicated command was issued from the front-end station. System log only. From SCP. Class, informative.

TEMPORARY FILE SPACE USED - *n* BLOCKS
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

TIME EXECUTING IN CPU - *h:m:s*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

TIME WAITING FOR I/O - *h:m:s*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

TIME WAITING FOR JXT - *h:m:s*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

TIME WAITING TO EXECUTE - *h:m:s*
When the CHARGES program is run, this message is issued if requested. From CHARGES. Class, informative.

TOTAL SBUS USED - *n*
When the CHARGES program is run, this message is issued if an installation parameter is set. From CHARGES. Class, informative.

TRANSMIT *jsq pdn* SZ = *w* SID = *fe* DID = *fe* TID = *tid* USERID = *id*
A dataset was sent to the front end. SZ is the size in words. USERID is the account number. From SCP. Class, informative.

2.3 MESSAGES BEGINNING WITH VARIABLE NAMES

var ACCESS ERROR: *xx*
Error number *xx* occurred during an attempt to access dataset *var*. From BUILD. Class, fatal.

var ADDED TO TOP OF LABEL STACK AT LEVEL *n-d*
An @LS*d* macro operation was executed while a %X, %X+, or %H directive was in effect. From SKOL macro trace. Class, informative.

var ADDED TO TOP OF LABEL STACK (LEVEL *n*)
An @*n*@LS0 macro operation was executed while a %X, %X+, or %H directive was in effect. From SKOL macro trace. Class, informative.

var CALCULATED POSITION DISAGREES WITH W@DPRCW
A request was issued to get the current position in the dataset. The calculated position did not agree with the position recorded in the DSP. Probable system error. Forward a dump to the Cray Research analyst. From POS.

var CHARACTERS IN LONGEST MACRO TRACE STRING
End of a macro translation in which the macro trace feature was activated. From SKOL. Class, informative.

var DATASET CONTAINS COMPRESSED DUMP - MUST USE XCOMP
Use XCOMP if decompression is desired. From FDUMP. Class, fatal.

var ERROR WHILE COMPRESSING SYMBOL TABLES
Error occurred when undesirable duplicate symbols (field definitions, etc.), were being deleted. From FDUMP. Class, fatal.

var FDUMP - AUTO FILE/ARRAY CONTAINS UNRECOGNIZABLE DIRECTIVE From FDUMP. Class, Caution.

var FDUMP - BAD CHARACTER WHILE CONVERTING OCTAL PARAMETER
Input must consist of only octal digits. From FDUMP. Class, fatal.

var FDUMP - CONTROL STATEMENT ERROR
Correct control statement. From FDUMP. Class, fatal.

var FDUMP - DATASET TO BE DECOMPRESSED UNRECOGNIZABLE
Bad input dataset. The input dataset must contain a dump with a recognizable header. From FDUMP. Class, fatal.

var FDUMP - DMEM LIMITS TOO SMALL FOR XP DUMP
LWA-FWA+1 must be at least 16 words. From FDUMP. Class, Caution.

var FDUMP - DUMP DATASET SPECIFIED DOES NOT EXIST OR IS NULL
A FILES directive must precede the first DMEM directive or specified dataset is empty. From FDUMP. Class, fatal.

var FDUMP - ERROR ON AUTO DIRECTIVE
The format of the directive is bad. From FDUMP. Class, caution.

var FDUMP - ERROR ON COMP DIRECTIVE
Default used on COMP directive; or if FWA is present, LWA must also be specified; or if LWA is present, FWA must also be specified. From FDUMP. Class, fatal.

var FDUMP - ERROR ON DMEM DIRECTIVE
FWA is greater than LWA or no FWA and LWA were found when attempt was being made to dump the memory. From FDUMP. Class, fatal.

var FDUMP - ERROR ON FILES DIRECTIVE
No default is used on the FILES directive. From FDUMP. Class, fatal.

var FDUMP - ERROR ON READING/WRITING UNBLOCKED DATASET
Dump dataset must be an unblocked dataset and must contain a memory dump written by SYSTEM STARTUP. From FDUMP. Class, fatal.

var FDUMP - ERROR ON DSDT DIRECTIVE
var FDUMP - ERROR ON DSYM DIRECTIVE
var FDUMP - ERROR ON DXTR DIRECTIVE
var FDUMP - ERROR ON SETBIAS DIRECTIVE
var FDUMP - ERROR ON SPACE DIRECTIVE
var FDUMP - ERROR ON XCOMP DIRECTIVE
Correct the directive. From FDUMP. Class, caution.

var FDUMP - EXEC TRACE CURRENT POINTER BAD - TREATED AS ZERO
Correct the symbol table datasets. From FDUMP. Class, fatal.

var FDUMP - HARDWARE ERROR WHILE READING SYMBOL TABLE DATASET
There is a hardware I/O error. Retry; if the error remains, see a Cray Research analyst. From FDUMP. Class, fatal.

var FDUMP - INPUT DATASET NULL
From FDUMP. Class, caution.

var FDUMP - NO MDW SPACE EXISTS IN OUTPUT HEADER
(MDW=memory descriptor word pair). The output dataset must contain a dump header according to dump format description. See the COS Operational Aids Reference Manual, CRI publication SM-0044, for the system dump format. From FDUMP. Class, fatal.

var FDUMP - OUTPUT DATASET EXISTS AND IS UNRECOGNIZABLE
A COMP or XCOMP directive specified an existing dataset to receive the compressed or decompressed output, and the existing dataset did not contain a valid system-dump header. From FDUMP. Class, fatal.

var FDUMP - SPECIFIED MEMORY NOT IN DUMP
var FDUMP - SPECIFIED SYMBOL TABLE DATASET NOT FOUND
var FDUMP - SYMBOL REQUIRED FOR EXEC TRACE DUMP NOT FOUND
var FDUMP - SYMBOL(S) REQUIRED FOR SDT DUMP NOT FOUND
var FDUMP - SYMBOLIC NAME NOT FOUND

var FDUMP - UNRECOGNIZABLE DIRECTIVE VERB IGNORED
Correct the directive. From FDUMP. Class, caution.

var IBN SHOULD EQUAL OBN AFTER RANDOM WRITE
A position request was issued for a dataset in write mode; a write has finished but the data was not flushed to disk, and is still in the buffer. After the I/O operation completed, IBN and OBN were not equal. Probable system error. Forward a dump to the Cray Research analyst. From POS. Class, not reprievable.

var IS A MULTIPLE NAME; IT CANNOT BE RENAMED
A group name cannot be specified for the new name option. Change *var* to a single module name, or eliminate its use in a COPY OLDNAME=NEWNAME directive. From BUILD. Class, fatal.

var IS A MULTIPLE NAME; IT CANNOT BE USED IN A RANGE
A group name cannot be specified in a range of modules. Change *var* to a single module name or eliminate its use in a module range. From BUILD. Class, fatal.

var IS NOT A VALID DIRECTIVE
var is not one of the four valid directives: FROM, OMIT, COPY, and LIST. Check the directives set for invalid names. From BUILD. Class, fatal.

var IS NOT FOLLOWED BY AN APPLICABLE DELIMITER
Bad syntax in a FROM, OMIT, or COPY directive. Check punctuation following directive name, dataset name, or module name. From BUILD. Class, fatal.

var IS WRONG LIBRARY TYPE; PROCESSING PROCEDURES
The BUILD job is processing procedure modules, but dataset *var* is a binary library. If a binary dataset is desired, delete the PROC keyword from the control statement. If a procedure library dataset is desired, determine if the input datasets specified by the OBL and B parameters are composed of procedures. From BUILD. Class, 2.

var MACRO TRANSLATION ERRORS
End of a macro translation in which fatal errors occurred. Examine the error listing, correct the errors, and resubmit the job. To proceed with the next job step following the macro translation without first correcting all the errors, place a %E directive in the source text. The value of *var* must be greater than or equal to the number of fatal errors to change the class of this message from fatal to warning. From SKOL. Class, fatal.

var MACRO TRANSLATION WARNINGS
End of a macro translation in which warnings were issued. Examine the error listing and correct the causes of the warning messages. From SKOL. Class, caution.

var MATCHES NO FILES IN *dn*
The group name *var*, specified in a COPY or OMIT directive, has no representatives in the current input dataset *dn*. Check the accuracy of the module and dataset names and the contents of the dataset. From BUILD. Class, fatal.

var MUST BE FOLLOWED BY *y*
An OMIT or COPY directive has bad syntax in a module range form. The parentheses are not balanced or the first and last module names are not separated by a comma. Check the syntax of all module ranges in the directives set. From BUILD. Class, fatal.

var NOT AT BEGINNING OF DATA AFTER POSITION REQUEST
A request to position to beginning of data was issued, but upon completion the dataset was not correctly positioned. Probable system error. Forward a dump to the Cray Research analyst. From POS. Class, not reprievable.

var NOT AT BEGINNING OF RECORD AFTER POSITION REQUEST
A position to the beginning of a record request was issued, but upon completion the dataset was not correctly positioned. Probable system error. Forward a dump to the Cray Research analyst. From POS. Class, not reprievable.

var NOT AT END OF DATA AFTER POSITION REQUEST
A position to end of data request was issued, but upon completion the dataset was not correctly positioned. Probable system error. Forward a dump to the Cray Research analyst. From POS. Class, not reprievable.

var NOT FOUND IN *dn*
The single module *var*, specified explicitly in a COPY or OMIT directive, does not exist on the current input dataset *dn*. Check the accuracy of the module and dataset names and the contents of the dataset. From BUILD. Class, fatal.

var POPPED OFF OF MACRO STACK (LEVEL *n*)
An @MU macro operation was executed while a %X, %X+, or %H directive was in effect. From SKOL macro trace. Class, informative.

var PUSHED ONTO MACRO STACK (LEVEL *n*)
An @MS*x* macro operation was executed while a %X, %X+, or %H was in effect From SKOL macro trace. Class, informative.

var REMOVED TO LABEL STACK AT LEVEL *n-d*
An @LU*d* macro operation was executed while a %X, %X+, or %H directive was in effect. From SKOL macro trace. Class, informative.

var REMOVED TO TOP OF LABEL STACK AT LEVEL *n*
An @LU0 macro operation was executed while a %X, %X+, or %H directive was in effect. From SKOL macro trace. Class, informative.

var REPLACED ON MACRO STACK (LEVEL *n*) BY '*y*'
An @MR*y* macro operation was executed while a %X, %X+, or %H directive was in effect. From SKOL macro trace. Class, informative.

var UNABLE TO READ SYMBOL TABLE DATASET
The symbol table dataset is bad. From FDUMP. Class, fatal.

var WAS CALLED BY *rtn* AT LINE NUMBER *n*
This message is generated by TRACEBACK during error processing. From $SYSLIB. Class, informative.

var WAS CALLED BY *rtn* AT LOCATION *n*
This message is generated by TRACEBACK during error processing. From $SYSLIB. Class, informative.

var WORDS (*n.n*%) USED FOR GLOBAL MACROS
End of a macro translation in which at least one global macro was defined. One word is used for each character in a global macro. From SKOL. Class, informative.

var WORDS (*n.n*%) USED FOR LOCAL MACROS
End of macro translation. One word is used for each character in a local macro. From SKOL. Class, informative.

var WORDS (*n.n*%) USED FOR SKOLTXT MACROS
End of macro translation. One word is used for each character in the standard macros. The number of words used is constant as long as SKOLTXT is unchanged; the percentage is constant as long as both SKOL and SKOLTXT are unchanged. From SKOL. Class, informative.

2.4 SEGLDR MESSAGES BEGINNING WITH *ERROR*, *WARNING*, *CAUTION*, *NOTE*, or *COMMENT*

NOTE MODULE - *modname* AT ADDRESS - *addr* CONTAINS A RELOCATABLE FIELD 22 BITS LONG -- CANNOT RUN IN EMA MODE
The reference at address *addr* is exactly 22 bits long. If the program were to be run with EMA mode enabled, the hardware treats this large address as a negative number. If you wish to run with EMA mode enabled, you must change these references. Class, informative.

CAUTION PROGRAM CANNOT RUN IN EMA MODE BECAUSE MODULE - *modname* REFERENCES *num* ADDRESSES BETWEEN 2MWD and 4MWD
On a machine with greater than 4 million words, the module *modname* has references that would not be made correctly if run with the EMA mode enabled. Changing the MLEVEL print level to NOTE will cause all occurences of these reference types to be listed. Class, caution.

CAUTION CODE OR LOCAL DATA LOADED AT ADDRESS GREATER THAN 4MWD
The program has a module loaded at an address greater than 4 million words. Code cannot execute correctly if loaded above this address. Data can only be accessed correctly if special coding instructions have been used. You should move data to common blocks or make the size of local data areas smaller. Class, caution.

ERROR (EXPECTED AFTER- *symbol*
Missing or misplaced open parenthesis. Lists must be enclosed in parentheses. Put an open parenthesis in the correct place. Class, fatal.

ERROR) OR , EXPECTED AFTER- *symbol*
The list has improper delimiters. Lists must be enclosed in parentheses. Items must be separated by commas. Class, fatal.

ERROR , EXPECTED AFTER- *symbol*
Too long a name (more than eight characters), or an unexpected delimiter such as: (= : ; was found where a comma was expected. Class, fatal.

ERROR , OR : EXPECTED AFTER- *symbol*
Too long a name (more than eight characters), or an unexpected delimiter such as) = ; was found where a comma or colon was expected.

ERROR = EXPECTED AFTER- *symbol*
Keyword *symbol* must be followed by =. Class, fatal.

ERROR ABORT VALUE MUST BE 'ON' OR 'OFF'
ABORT was set to a value other than ON or OFF. Class, fatal.

ERROR ALIGN VALUE MUST BE 'IGNORE','NORMAL',OR 'MODULES'
ALIGN was set to a value other than those shown. Class, fatal.

ERROR AMBIGUOUS LINK FROM CALLER- *module* TO DUPLICATED CALLEE- *epname*
Two copies of a module assigned to two segments on a common branch, or copies of a module specified in two segments on different branches called from a common predecessor. Change the tree description directives or segment description directives. Class, fatal.

ERROR AMBIGUOUS REFERENCE TO DUPLICATED COMMON BLOCK- *cbname* FROM MODULE- *modname*
Two or more links exist between *cbname* and *modname*. Rewrite the module assignments to resolve ambiguity or remove a copy of *cbname*. Class, fatal.

ERROR BINARY DATASET CONTAINS GARBAGE- *dn*
Invalid data in dataset *dn*. Class, fatal.

ERROR BIN FILE- *dn* IS FORCE-LOADED INTO SEGMENT- *segname*
Dataset *dn* is already owned by segment *segname*. A binary input file cannot belong to more than one segment. Make *dn* belong to only one segment. Class, fatal.

ERROR BIN FILE- *dn* IS A GLOBAL BINARY INPUT FILE
A binary input file cannot be both global and local. Make dataset *dn* either local or global. Class, fatal.

ERROR CALLER- *epname* IS AN UNFIXED PREDECESSOR OF DUPLICATE CALLEE- *epname*
The caller is not fixed (assigned to a module). Use the MODULES directive to fix caller. Class, fatal.

ERROR COMMON BLOCK - *cbname* REDEFINED WITH DIFFERENT LENGTH BY MODULE - *modname*
The user code in *modname* defines *cbname* as using more or less memory than was originally allocated to *cbname*. Make all references to *cbname* consistent (see the REDEF directive). Class, fatal.

ERROR COMPILATION ERRORS IN MODULE- *modname* IN LOAD DATASET- *dn*
User code contains compilation errors. Class, fatal.

ERROR DATA-LOAD TO BLANK COMMON FROM MODULE- *modname*
The user code attempts to preload blank common, which is illegal. Class, fatal.

ERROR DATASET DOES NOT EXIST- *dn*
SEGLDR cannot find dataset *dn*. Access or create dataset *dn* and rerun. Class, fatal.

ERROR DOUBLY DEFINED SEGMENT- *segname*
The segment tree description directives involving *segname* are inconsistent with each other and therefore cause the segment to be doubly defined. Correct or remove those directives causing multiple definitions. Class, fatal.

ERROR DYNAMIC COMMON BLOCK- *cbname*
DATA-LOADED BY MODULE- *modname*
The user code attempts to preload dynamic common, which is illegal. Class, fatal.

ERROR DYNAMIC COMMON BLOCK- *cbname*
NOT LOADED
The common block named by the DYNAMIC directive is not found in any user code. Include the common block in the code or remove the directive. Class, fatal.

ERROR ENDTREE DIRECTIVE EXPECTED
The end of the directive file was encountered before the ENDTREE directive was found. Include the ENDTREE directive in the segment tree definition directives. Check for end-of-file record embedded within the directive file. Class, fatal.

ERROR ENTRY- *epname* CANNOT BE REDEFINED AS A SYNONYM
epname is already used as a synonym in an EQUIV directive. Remove all but one of the occurrences of *epname* from the synonym list. Class, fatal.

ERROR ENTRY- *epname* DUPLICATED BUT NOT NAMED BY A 'DUP' DIRECTIVE FOR SEGMENT- *segname*
More than one entry of *epname* exists, probably because a MODULES directive was used instead of DUP directive. Use the DUP directive to duplicate *epname*. Class, fatal.

ERROR ENTRY- *epname* HAS BEEN PREVIOUSLY DEFINED
epname is already defined as a primary entry point. Change the EQUIV directive or change *epname* to another name. Class, fatal.

ERROR ENTRY- *epname1* IS ALREADY A SYNONYM OF ENTRY- *epname2*
An attempt to define *epname1* as a synonym was made using the EQUIV directive when *epname1* is already defined as a synonym. Rewrite EQUIV directives so that *epname1* is a synonym for only one other symbol. Class, fatal.

ERROR ENTRY- *epname* NOT DECLARED DUPLICATE IN ALL CASES
A reference to *epname* is made from a segment that does not have access to a copy of it. Using the DUP directive, provide a copy of *epname* to the segment needing it. Class, fatal.

ERROR ENTRY POINT- *epname* WAS DECLARED BY A DUP DIRECTIVE BUT NOT INCLUDED IN SEGMENT- *segname*
epname was not included in segment *segname*. Include *epname* in the segment description or remove it from DUP directive. Class, fatal.

ERROR 'FIRST' ENTRY- *epname* NOT FOUND
epname was not found. Include *epname* in code. Check the spelling. Class, fatal.

ERROR FORCE VALUE MUST BE ON OR OFF
FORCE is set to a value other than ON or OFF. Class, fatal.

ERROR GARBAGE AFTER NODEFLIB- *symbol*
An unexpected symbol was encountered following keyword NODEFLIB. The symbol following NODEFLIB must be either a semicolon (;) or end-of-line. Class, fatal.

ERROR ILLEGAL ABS DATASET NAME- *dn*
dn does not follow COS naming conventions or is reserved by SEGLDR. Change *dn* to a valid name. Class, fatal.

ERROR ILLEGAL BCINC VALUE - *value*
BCINC was given a value containing a non-numeric character, which is invalid. Class, fatal.

ERROR ILLEGAL BIN DATASET NAME- *dn*
dn does not follow COS naming conventions or is reserved by SEGLDR. Change *dn* to a valid name. Class, fatal.

ERROR ILLEGAL DATASET NAME- *dn*
dn does not follow COS naming conventions or is reserved by SEGLDR. Change *dn* to a valid name. Class, fatal.

ERROR ILLEGAL GRANT FUNCTION - *value*
GRANT is set to an undefined value. Use a valid privilege name. Class, fatal.

ERROR ILLEGAL LIB DATASET NAME- *dn*
dn does not follow COS naming conventions or is reserved by SEGLDR. Change *dn* to a valid name. Class, fatal.

ERROR ILLEGAL LOAD ORIGIN VALUE- *value*
The load origin value is set to a reserved word or a number that includes a non-octal digit. Change the value to an octal number or a symbol that is not a reserved word. Class, fatal.

ERROR ILLEGAL ORDER STRING- *symbol*
The order string has characters other than one each of B, C, and L. Class, fatal.

ERROR ILLEGAL PADINC VALUE - *value*
PADINC was given a value containing a non-numeric character, which is illegal. Class, fatal.

ERROR ILLEGAL PRESET VALUE- *value*
The PRESET value is set to a reserved word or a number that includes a non-octal digit. Change the value to an octal number or a symbol that is not a reserved word. Class, fatal.

ERROR ILLEGAL SLT VALUE- *value*
SLT was given a value containing a non-numeric character, which is invalid. Class, fatal.

ERROR LINK FROM MODULE- *modname* IN SEGMENT- *segname* TO MODULE- *modname* IN SEGMENT- *segname* IS ILLEGAL
No valid path exists between copies of the two modnames. Rewrite module assignments and/or use the DUP directive to provide a valid link. Class, fatal.

ERROR MISPLACED BIN DIRECTIVE
The BIN directive appears outside SEGMENT/ENDSEG directive group. Place the BIN directive inside the SEGMENT/ENDSEG directive group. Class, fatal.

ERROR MISPLACED COMMONS DIRECTIVE
The COMMONS directive appears outside the SEGMENT/ENDSEG directive group. Class, fatal.

ERROR MISPLACED DEVICE DIRECTIVE
The DEVICE directive appears outside the SEGMENT/ENDSEG directive group. Class, fatal.

ERROR MISPLACED DUP DIRECTIVE
The DUP directive appears inside or after the SEGMENT/ENDSEG directive group. Place the DUP directive immediately before the SEGMENT/ENDSEG directive group. Class, fatal.

ERROR MISPLACED MODULES DIRECTIVE
The MODULES directive appears outside the SEGMENT/ENDSEG directive group. Class, fatal.

ERROR MISPLACED SAVE DIRECTIVE
The SAVE directive appears outside the SEGMENT/ENDSEG directive group. Class, fatal.

ERROR MODULE- *modname* IS ALREADY FIXED AND IS A SYNONYM
modname appears in an EQUIV directive, but is already fixed. Float *modname* or remove from EQUIV. Class, fatal.

ERROR MULTIPLE SEGMENT TREE ROOT- *nodename*
The object defined by the segment tree description directives has more than one root and therefore is not a tree. There is more than one node with no predecessor. *nodename* is one of these nodes. Correct the segment tree description directives to define a tree with exactly one root. There must be exactly one node with no predecessors. Class, fatal.

ERROR NAME- *modname* IS A SYNONYM OF- *modname*
A synonym for a primary entry point cannot have the same name (be defined) as a primary entry point (named in the MODULES directive). Class, fatal.

ERROR NO LEGAL LINK FROM CALLER- *epname1* TO DUPLICATED CALLEE- *epname2*
There is no branch common to caller and any copy of callee. Rewrite the module assignments to provide a valid link. Class, fatal.

ERROR NO LEGAL LINKAGE FROM MODULE- *modname* TO DUPLICATED COMMON BLOCK- *cbname*
No copy of common block *cbname* is accessible by *modname*. Rewrite the module assignments or provide another copy of *cbname* to provide a valid link. Class, fatal.

ERROR NO MODULES LOADED INTO SEGMENT- *segname*
segname has no entry points named by MODULES or BIN directives. Every segment must have at least one entry point. Provide *segname* with at least one entry point. Class, fatal.

ERROR NORED VALUE MUST BE ON OR OFF
NORED is set to a value other than ON or OFF, which is invalid. Class, fatal.

ERROR NO SEGMENT TREE ROOT DEFINED
Other errors in the segment tree description directives leave the segment tree incompletely specified; therefore, the root is not defined. A null tree also causes this message to be printed. A null tree results if the only directives are: TREE and ENDTREE. Correct the segment tree definition directives. Class, fatal.

ERROR NO TRANSFER ENTRY POINT
No transfer entry point was specified with the XFER directive, or no entry point exists to which control can be given to begin execution of the root segment. Use the XFER directive to specify the desired initial entry point of the root segment. Class, fatal.

ERROR NULL VALUE ASSIGNED TO- *symbol*
Nothing follows = following keyword *symbol*. Insert a parameter after =. Class, fatal.

ERROR PREMATURE END OF *dir* DIRECTIVE
A comma was encountered at the end of a line but an end of file was then encountered instead of the next record. Use the correct form of the *dir* directive; complete the list to be processed or remove the trailing comma. Check for an embedded end-of-file record. Class, fatal.

ERROR PREMATURE END OF SUCCESSOR LIST- *segname*
Syntax error: end of line was encountered before expected, after (or *segname* . Complete the successor list on one line or end the line with a comma to continue the list on the next line. Class, fatal.

ERROR PRESET PATTERN EXCEEDS 16 BITS- *value*
The PRESET directive is set to a value too large to be represented by 16 binary digits. Change the value so that it is in range. Class, fatal.

ERROR READING BINARY FILE- *dn* RECORD NUMBER- *value*
A bad binary file or a read error was encountered. Rerun the job. Recreate the dataset and rerun the job. If the job still fails, see a Cray Research site analyst. Class, fatal.

ERROR REDEF VALUE MUST BE 'IGNORE', 'WARNING', or 'ERROR'
REDEF is set to a value other than IGNORE, WARNING, or ERROR.
Class, fatal.

ERROR REFERENCE FROM MODULE- *modname* IN SEGMENT- *segname* TO COMMON BLOCK- *cbname* IS AMBIGUOUS
Two copies of a common block were assigned to two segments on a common branch, or copies of a common block specified in two segments on different branches were called from a common predecessor. Change the tree description directives or segment description directives. Class, fatal.

ERROR REFERENCE TO BLOCK- *cbname* IN SEGMENT- *segname* from MODULE- *modname* IS ILLEGAL
No copies of *cbname* and *modname* are contained in segments having a common branch. Rewrite the module assignments to provide a valid link. Class, fatal.

ERROR RELOCATION OVERFLOW IN MODULE- *modname* AT ADDRESS *address*
The field in VWD is too small for the relocated value or the value is too large to fit in a 22-bit field. Class, fatal.

ERROR SAVE VALUE MUST BE ON OR OFF
SAVE is set to a value other than ON or OFF. Class, fatal.

ERROR SECONDARY ENTRY- *epname* IN MODULE- *modname* NOT DECLARED BY A DUP DIRECTIVE
A secondary entry point in a duplicated module is not declared as a duplicate. Include *epname* in a DUP directive. Class, fatal.

ERROR SECURE VALUE MUST BE ON OR OFF
SECURE is set to a value other than ON or OFF, which is invalid. Class, fatal.

ERROR SEGMENT DESCRIPTION DIRECTIVES MISSING
An end of file was encountered before the segment description directives were found. Include the segment descriptors in the directive file. Check for an embedded end-of-file mark. Class, fatal.

ERROR SEGMENT- *segname* IS UNDEFINED
A segment in a DUP list is not defined in the segment tree. Include *segname* in the segment tree description directives. Class, fatal.

ERROR SEG. LOADER RESIDENT ROUTINE NOT LOADED- *modname*
$SEGRES cannot be found. See a Cray Research site analyst to find the library containing $SEGRES. Class, fatal.

ERROR SEGMENT NOT INCLUDED IN TREE- *segname*
segname is not defined in the segment tree. Include *segname* in the segment tree description directives. Class, fatal.

ERROR SEGMENT- *segname* WAS NOT DESCRIBED BY A 'SEGMENT' DIRECTIVE
An end of file was reached before a SEGMENT/ENDSEG group describing *segname* was found. Include a SEGMENT/ENDSEG group describing *segname* or remove all references to *segname*. Class, fatal.

ERROR SEPARATOR EXPECTED AFTER- *symbol*
No separator such as , or) or ; was included where one is required. Class, fatal.

ERROR SUCCESSOR=PREDECESSOR
Directives have caused a node in the segment tree to be defined as its own successor. Correct the segment tree description directives. Class, fatal.

ERROR SYMBOLS VALUE MUST BE ON OR OFF
SYMBOLS is set to a value other than ON or OFF. Class, fatal.

ERROR TOO MANY SEGMENT CALLS-- INCREASE SLT ACTUAL SLT REQUIREMENT- *value*
Too little space exists in the Segment Linkage Entry Table for callers and callees. Use the SLT directive to

increase the Segment Linkage Entry Table to the required size as stated. Class, fatal.

ERROR TOO MANY SEGMENTS
The 1000-segment limit was exceeded. Reduce the number of segments. Class, fatal.

ERROR TRANSFER ENTRY POINT - *epname* IS NOT THE ROOT SEGMENT
The entry point *epname* is not in the root segment and is specified as the entry point for beginning execution. Either put the transfer entry point in the root segment or specify another entry point within the root segment as the transfer entry point, using the XFER directive. Class, fatal.

ERROR UNEXPECTED EOF
An end of file was encountered after a delimiter such as a comma in descriptor list. Complete the description directives or remove the trailing delimiter. Class, fatal.

ERROR UNFIXED MODULE- *modname* REFERENCES DUPLICATED COMMON BLOCK- *cbname*
Duplicate common blocks were specified but not all modules which reference them are fixed. Fix the modules which reference duplicate common blocks. Class, fatal.

ERROR UNKNOWN DIRECTIVE- *symbol*
symbol is not a keyword. Check spelling and form required for directives. Class, fatal.

ERROR UNKNOWN OR MISPLACED DIRECTIVE- *symbol*
A keyword was not recognized (syntax error). Use the proper form and spelling of the keyword. Class, fatal.

ERROR USX OPTION VALUE MUST BE 'ERROR', 'WARNING' OR 'IGNORE'
The USX directive is set to a value other than ERROR or WARNING or IGNORE. Class, fatal.

ERROR XFER ENTRY- *epname* NOT FOUND
epname was not found. Include *epname* in the code. Check the spelling. Class, fatal.

NOTE MODULE - *modname* AT ADDRESS - *addr* CONTAINS A RELOCATABLE FIELD 22 BITS LONG -- CANNOT RUN IN EMA MODE
The reference at address *addr* is exactly 22 bits long. If the program were to be run with EMA mode enabled, the hardware treats this large address as a negative number. If you wish to run with EMA mode enabled, you must change these references. Class, informative.

WARNING ABSOLUTE MODULE SKIPPED- *modname* IN LOAD DATASET- *dn*
An absolute binary module was encountered when a relocatable module was expected. Remove the absolute binary module or do not specify *dn*.

WARNING COMMON BLOCK- *cbname* REDEFINED WITH DIFFERENT LENGTH BY MODULE- *modname*
The user code in *modname* defines *cbname* as using less memory than was originally allocated to *cbname*. Make all references to *cbname* consistent. Class, caution.

WARNING DUPLICATE ENTRY- *epname* DISCOVERED IN FILE- *dn*; IGNORED
Two or more copies of *epname* exist. Any subsequent occurrences found in *dn* are ignored. Class, caution.

WARNING DUPLICATE MODULE- *modname* ENCOUNTERED AND IGNORED; BIN=*dn*
This message means that *modname* appears more than once, probably in a force-loaded file. The extra copies are ignored. Class, caution.

WARNING ECHO VALUE MUST BE ON OR OFF; ON ASSUMED
The ECHO directive is set to a value other than ON or OFF (default is ON). Class, caution.

WARNING ENDSEG DIRECTIVE MISSING; ASSUMED
Two SEGMENT directives were found without an intervening ENDSEG directive. To eliminate the warning, insert an ENDSEG directive before beginning a new set of directives with a SEGMENT directive. Class, caution.

WARNING ENTRY POINT- *epname* IS IN FORCE-LOAD FILE- *dn* BUT WILL BE LOADED FROM FILE- *dn*
epname appears in the load file *dn* and in a file specified in a MODULES directive. *epname* is loaded from the second *dn*. Class, caution.

WARNING ENTRY POINT- *epname* UNUSED; DISCARDED
No references are made to *epname* and it is discarded. If a reference to *epname* is intended, check the spelling. Class, caution.

WARNING GARBAGE AFTER ENDSEG IGNORED
Extraneous characters were found following the ENDSEG directive. To eliminate the warning, take out the extraneous characters. Class, caution.

WARNING ILLEGAL MANAGED MEMORY INCREMENT-DEFAULT USED
Attempted to assign a value which contained a non-numeric character to the heap increment. Class, caution.

WARNING ILLEGAL MANAGED MEMORY SIZE-DEFAULT USED
Attempted to assign a value which contained a non-numeric character to the heap size. Class,caution.

WARNING ILLEGAL MEMORY EPSILON VALUE-DEFAULT USED
Attempted to assign a value which contained a non-numeric character to the heap *min* value. Class,caution.

WARNING ILLEGAL STACK INCREMENT-DEFAULT USED
Attempted to assign a value which contained a non-numeric character to the stack increment. Class,caution.

WARNING ILLEGAL STACK SIZE-DEFAULT USED
Attempted to assign a value which contained a non-numeric character to the stack size. Class,caution.

WARNING INITIAL MANAGED MEMORY TOO SMALL-MINIMUM USED
Managed memory could not expand and its specified initial size was not large enough to include the stack and necessary overhead. Specify a larger initial heap size or do not fix the heap size. Class, caution.

WARNING INTER-SEGMENT DATA-LOAD TO COMMON BLOCK- *cbname* FROM MODULE- *modname* DATA LOADING SKIPPED
An attempt was made to preset common variables with a DATA statement in a module that is in a different segment than the segment where the common block is assigned. If data loading is needed, put the DATA statement in a module in the same segment as *cbname*. Class, caution.

WARNING LINK AT ADDRESS- *addr* IN MODULE- *modname* HAS PARCEL ATTRIBUTE ENTRY- *epname* HAS WORD ATTRIBUTE EXTERNAL REFERENCE LINKED TO PARCEL ZERO
Link and entry-point address types do not match. The needed conversion is made automatically. Class, caution.

WARNING LINK AT ADDRESS- *addr* IN MODULE- *name* HAS WORD ATTRIBUTE ENTRY- *epname* HAS PARCEL ATTRIBUTE EXTERNAL REFERENCE LINKED AS A WORD ADDRESS
Link and entry-point address types do not match. The needed conversion is made automatically. Class, caution.

WARNING MEMORY EPSILON VALUE TOO SMALL-MINIMUM USED
The value specified as the heap *min* value was less than two. Class, caution.

WARNING MISPLACED 'ENDSEG' DIRECTIVE IGNORED
Two or more ENDSEG directives appear in a row. To eliminate the warning, take out the extra ENDSEG directives. Class, caution.

WARNING MODULE- *modname* HAS NO ENTRIES; MODULE SKIPPED
modname has no entry points. If *modname* must be included, try placing an entry point in the code to force loading. Class, caution.

WARNING MODULE- *modname* IS A RELOCATABLE OVERLAY; MODULE SKIPPED
modname is an operating system relocatable overlay. *modname* cannot be loaded and is skipped. Class, caution.

WARNING REFERENCE TO COMMON BLOCK- *cbname* IN SEGMENT- *segname* FROM MODULE- *modname* IN SEGMENT- *segname* IS UNSAFE
cbname might not be in memory. The user should verify that *cbname* is in memory. If it is not, rewrite the module assignments. Class, caution.

WARNING SEGMENT- *segname* IS NULL
No calls are made to any modules in *segname*. All modules assigned to *segname* have been discarded during the segment tree trimming process. Class, caution.

WARNING STACK TOO SMALL-DEFAULT SIZE USED
The value specified as the initial stack size was less than 128. Class, caution.

WARNING UNKNOWN MAP OPTION- *symbol*; MAP=PART ASSUMED
The MAP directive is set to a value other than PART or ALL. PART is assumed. Class, caution.

2.5 SID MESSAGES BEGINNING WITH ****

**** ERROR: a file name is required
The directive ALTERNATEINPUT was used without a file name. Enter the directive again with a legal file name following the keyword. From SID. Class, informative.

**** ERROR: a parcel address is required
An INSTRUCTION directive did not contain a parcel address or the special symbol PC. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: a period must be followed by a symbol
A symbol was followed by a period, causing SID to treat the first symbol as a module name and expect another symbol following the period. Enter the directive again, using the correct syntax. From SID. Class, informative.

**** ERROR: a relation is required in an IF clause
The word address in the conditional clause of a BREAKPOINT or PACKAGE directive was not followed by a legal test relation (one of =, <>, <, >, or >). Enter the directive again using a valid test relation. From SID. Class, informative.

**** ERROR: a symbol or address is required
The directive name WHERE was followed by something other than a symbol, a word address, or a parcel address. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: address is out of range
A word address is outside of the user address space (larger than JCFL). Check the address to see if the offset is too large. From SID. Class, informative.

**** ERROR: address is too large
An address took more than 24 bits. Correct the address and reenter the directive. From SID. Class, informative.

**** ERROR: alternate input file is not available
The dataset used in an ALTERNATEINPUT directive is not local to the job. Quit the debugging session, access the datasets desired for use with ALTERNATEINPUT, and restart SID. From SID. Class, informative.

**** ERROR: bad syntax in complex number
A complex number used the wrong format. Check the syntax for the directive and for complex numbers and try again. From SID. Class, informative.

**** ERROR: bad syntax in number
An illegal character was encountered while parsing a number. Look at the arrow above the error message to find the illegal character and reenter the directive. From SID. Class, informative.

**** ERROR: base must be OCTAL, DECIMAL, or MIXED
Something other than OCTAL, DECIMAL, or MIXED or their abbreviations followed the keyword BASE. Try again. Base modes can be abbreviated to O, D, and M. From SID. Class, informative.

**** ERROR: bit count is too large
The bit count in a masked PATCH directive does not fit into the remainder of the word; the starting bit number plus the bit count is greater than 64. Recompute the starting bit and bit count and enter the directive again. In mixed-base mode (the default), bit numbers and counts are decimal values. From SID. Class, informative.

**** ERROR: bit number is too large
The starting bit for a masked PATCH or masked DISPLAY directive was greater than 63 decimal. Check the value for the starting bit and enter the directive again. In mixed-base mode (default), bit numbers and counts are decimal values. From SID. Class, informative.

**** ERROR: breakpoint address refused
A SID logic problem caused the breakpoint address to be rejected. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: breakpoint number needed
A BREAKPOINT or PACKAGE directive was missing a breakpoint number. This number, between 1 and 17 (octal) or 1 and 15 (decimal), must follow the directive keyword. Enter the directive again with a number between 1 and 17 octal (in octal base mode) or 1 and 15 decimal (in decimal or mixed base mode) following the word BREAKPOINT or PACKAGE. From SID. Class, informative.

**** ERROR: can't be used in package and one is open
A directive that cannot be used in a breakpoint package was entered while a package was still open. Close the package with the ENDPACKAGE directive and reenter the directive. From SID. Class, informative.

**** ERROR: common block table is missing
The symbol table record for a common block is missing, probably because the subroutine was not compiled or assembled with the necessary option. Check to see if the subroutine where the common block appears was compiled with the ON=IZ option or assembled with the SYM option. If the 'subroutine.commonblock.symbol' form was used, look at the same common block in an earlier subroutine. From SID. Class, informative.

**** ERROR: condition refused
A SID logic failure caused the breakpoint condition to be refused. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: count must be numeric
One of the directives FORWARD or BACKWARD was followed by a non-numeric input token when a count was expected. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: count must be unsigned integer
The bit count for a masked PATCH or masked DISPLAY directive was not an unsigned constant as expected. Reenter the directive this time with an unsigned octal integer for the bit count. From SID. Class, informative.

**** ERROR: count not allowed for masked display
A masked display directive included a display count, but only one word at a time may be shown with a masked display. Check the syntax for the directive. From SID. Class, informative.

**** ERROR: directive must begin with keyword
The first token of an input record or the first token following a semicolon was not a SID keyword; the directive name may have been misspelled. Check the spelling of the desired directive and try again. From SID. Class, informative.

**** ERROR: directive must end with ; or end of line
An extra token was found at the end of a directive. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: dual format not allowed for masked display
The keyword AND followed the display format in a masked DISPLAY directive, but masked displays can use only one display format. Check the syntax for the directive and try again. To show the field in two formats, two separate display commands are necessary. From SID. Class, informative.

**** ERROR: empty parentheses
Empty parentheses were used in a word address expression. From SID. Class, informative.

**** ERROR: exponent overflow
The exponent for a floating-point value was too large. Check the FORTRAN (CFT) Reference Manual, CRI publication SR-0009, for the allowable range of floating-point constants. From SID. Class, informative.

**** ERROR: exponent underflow
The exponent for a floating-point value was too small. Check the FORTRAN (CFT) Reference Manual, CRI publication SR-0009, for the allowable range of floating-point constants. From SID. Class, informative.

**** ERROR: fewer subscripts than declared
An array reference contained fewer subscripts than there were declared dimensions in the array. From SID. Class, informative.

**** ERROR: first RUN requires a parcel address
A starting address must be given the first time the user code is entered with a RUN or STEP directive. Reenter the directive, this time giving the program's starting address. From SID. Class, informative.

**** ERROR: first STEP requires a parcel address
A starting address must be given the first time the user code is entered with a RUN or STEP directive. Reenter the directive, this time giving the program's starting address. From SID. Class, informative.

**** ERROR: HELP file is not available
The file $DBHELP, which contains the HELP text, is not local to the job and is not available on the System Directory. Ask to have the file added to the System Directory or access it. From SID. Class, informative.

**** ERROR: HELP topic tables are too small
The tables holding the directory for the help files are too small to hold information about all of the HELP topics. Report the problem to a Cray Research analyst, or recompile SID with a larger value for MAXFILES in subroutine DBGHELP. From SID. Class, informative.

**** ERROR: indirect address is too large
An indirect address is outside the user's address area. Check the variable used for indirect addressing and try again. From SID. Class, informative.

**** ERROR: input token is not a symbol
A SID logic error caused a routine for looking up symbols in the symbol file to be called with a nonsymbol argument. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: invalid hexadecimal digit
A value preceded by the prefix X' contained an invalid hexadecimal digit and a hexadecimal value was expected. Check for a non-hexadecimal digit in a value with the prefix X'. From SID. Class, informative.

**** ERROR: invalid octal digit
An invalid character was detected in an octal number. From SID. Class, informative.

**** ERROR: invalid parcel specifier
A character other than A, B, C, or D followed an octal address when a parcel specifier was expected. Check the parcel address for invalid characters. From SID. Class, informative.

**** ERROR: invalid relation
The IF clause in a BREAKPOINT or PACKAGE directive used a relation other than =, <>, <, $\leq$, >, or $\geq$. Use one of the relations listed above. From SID. Class, informative.

**** ERROR: keyword AND requires a format
The keyword AND in a DISPLAY directive was followed by something other than a

valid format specifier. Check the list of available display formats and try again. From SID. Class, informative.

**** ERROR: keyword AT requires a parcel address
The keyword AT in a RUN or STEP directive was not followed by a valid parcel address expression. Check the syntax for the directive and the form of a valid parcel address expression and try again. From SID. Class, informative.

**** ERROR: keyword FOR requires a count
The keyword FOR in a DISPLAY, FORWARD, BACKWARD, or INSTRUCTION directive was followed by something other than an unsigned integer. Reenter the directive with an unsigned integer constant following the keyword FOR. From SID. Class, informative.

**** ERROR: keyword FOR requires a step count
The keyword FOR in a STEP directive was followed by something other than an unsigned integer. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: keyword IN requires a format
The keyword IN in a DISPLAY directive was followed by something besides a recognized format name. Check the list of available display formats and try again. From SID. Class, informative.

**** ERROR: label not saved; see HELP LABELS
The address for a statement label was not saved. The help topic LABELS lists several possible reasons for labels not being saved. Use a different label. From SID. Class, informative.

**** ERROR: LOAD requires an X-variable
The keyword LOAD was followed by something other than an X-variable name. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: maximum string length is 8 characters
A literal string used as a value had more than eight characters. If the string is a patch value, break it up into groups of eight characters. From SID. Class, informative.

**** ERROR: memory address is missing
No address was given when a word expression was expected. Check the syntax for the directive being used and try again. From SID. Class, informative.

**** ERROR: module name must be a symbol
The keyword MODULEBASE was followed by something other than a symbol. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: module not found
The module name preceding a variable name or label was not found in the debug map table. Check the spelling of the module name and try again. From SID. Class, informative.

**** ERROR: module with this address not found
An address used in a WHERE directive was outside the range of subroutines and common blocks listed in the debug map table. From SID. Class, informative.

**** ERROR: must be used in package and none is open
A directive that can only be used in a breakpoint package was entered when no package was open. Set up a package with the PACKAGE directive. From SID. Class, informative.

**** ERROR: need constant to specify a register
A register was specified with a symbol other than a constant; for example, 'B.ZA' where ZA was not defined with the '=' pseudo-op. Use the regular register name; for example, B3, T77. From SID. Class, informative.

**** ERROR: nesting of ALTERNATEINPUT is not implemented
An ALTERNATEINPUT directive was used in an alternate input file. From SID. Class, informative.

**** ERROR: no module has been specified
There is no current default module; absence of the default module should happen only following a MODULEBASE directive containing an error or if no MODULEBASE directive has been used. Use the MODULEBASE directive to specify which module (subprogram or common block) should be searched for symbols, or precede each variable name or label with a module name followed by a period. From SID. Class, informative.

**** ERROR: not a normal symbol
Only symbols classified as constant or normal in the symbol table can currently be processed by the debugger. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: null field
A value decoded by $NICV had a null field; probably from a logic problem in SID. Check the value pointed to by the arrow above the error message; if it looks correct, report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: number is too large
A number in a SID directive takes more than 64 bits. Correct the number and try again. From SID. Class, informative.

**** ERROR: number must be between 00 and 15 decimal
The number for a local X or C variable was not in the correct range. For mixed-base mode (the default), this number is decimal and must be between 00 and 15. Try again, this time using a variable number in the correct range. From SID. Class, informative.

**** ERROR: number must be between 00 and 17 octal
The number for a local X or C variable was not in the correct range. In octal-base mode, this number is octal and must be between 00 and 17. Try again, this time using a variable number in the correct range. From SID. Class, informative.

**** ERROR: number must be between 1 and 15 decimal
A breakpoint or package number was not in the correct range. For mixed-base mode (the default) or decimal mode, this number is decimal and must be between 1 and 15. Try again with a number in the correct range. From SID. Class, informative.

**** ERROR: number must be between 1 and 17 octal
A breakpoint or package number was not in the correct range. For octal-base mode, this number is octal and must be between 1 and 17. Try again with a number in the correct range. From SID. Class, informative.

**** ERROR: package contains undecodeable command
A SID logic failure caused an undecodeable command to be placed in a breakpoint package. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: package overflow
The last directive added to a package caused the package to overflow. The package is removed. Restart the package using fewer directives. Array references with variable subscripts should be avoided because they use too much space in the package. From SID. Class, informative.

**** ERROR: parcel address is missing
No parcel address was given when a parcel address expression was expected. Check the syntax for the directive and for parcel address expressions and try again. From SID. Class, informative.

**** ERROR: parcel address is out of range
A parcel address expression is out of the user's address area. Check the address and try again. From SID. Class, informative.

**** ERROR: parcel value must fit in 16 bits
The value given in a PATCH PARCEL directive was too large to fit into 16 bits. Check the value, or use a masked patch if more than 16 bits are to be patched. From SID. Class, informative.

**** ERROR: PATCH list is too long
The list of values used in a PATCH directive is longer than the current limit. Break up the list and use more than one PATCH command. From SID. Class, informative.

**** ERROR: problem with format of $DEBUG
The symbol table in $DEBUG does not have the expected format. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: requested format is not implemented
The format requested for a DISPLAY directive is not implemented. Check the list of available formats and try again. From SID. Class, informative.

**** ERROR: REWIND requires a file name
The keyword REWIND was followed by something other than a valid file name. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: routine GETNONBL failed
Routine GETNONBL in subroutine DBGTOKEN failed due to a SID logic error. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: search for dimension descriptor is lost
A symbol table search was lost due to a SID logic failure. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: search for dimension symbol entry is lost
A symbol table search was lost due to a SID logic failure. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: STORE requires an X-variable
The keyword STORE was followed by something other than an X-variable name. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: subroutine table is missing
The symbol table record for a subroutine is missing, probably because the subroutine was not compiled or assembled with the necessary option. Check to see if the subroutine was compiled with the ON=IZ option or assembled with the SYM option. From SID. Class, informative.

**** ERROR: subroutine table is missing for *modulename*
The subroutine table for the module named in the message is missing. The message is issued in response to the WHERE directive using a word or parcel address, and usually gives the name of a library routine containing the address. From SID. Class, informative.

**** ERROR: subscript must be an integer
An array subscript was something other than an integer constant or an integer variable. Check the subscripts. Class, informative.

**** ERROR: subscript must be followed by , or)
An array subscript was followed by something other than a comma or closing parenthesis. Check the subscripts and their separators and try again. From SID. Class, informative.

**** ERROR: symbol * not permitted in an IF clause
The symbol * was used for the word address in the IF clause of a BREAKPOINT or PACKAGE directive. Use a word-address expression instead of *. From SID. Class, informative.

**** ERROR: symbol file is not available
A symbolic address was used in a directive but the file containing the symbol table is not available. The file may be missing for either of two reasons: the program was not compiled or assembled with the necessary options; or the absolute binary file was saved to be used later, but the symbol file was not saved or not accessed. Recompile a CFT program with the ON=IZ option; reassemble a CAL program with the SYM option. If the absolute binary load module is saved, save $DEBUG also. Absolute addressing can be used when the symbol file is not available. From SID. Class, informative.

**** ERROR: symbol is external
A symbol has been looked up in a module where it is not defined. Try again, this time specifying the module where the symbol is defined. From SID. Class, informative.

**** ERROR: symbol is not a parcel
A symbol used in a parcel-address expression was not a label or entry point. Try again, using an absolute parcel address or a label or entry-point symbol. From SID. Class, informative.

**** ERROR: symbol is not an array
A symbol followed by subscripts is not defined as an array. Use a word-address offset with the symbol. From SID. Class, informative.

**** ERROR: symbol is not word aligned
The symbol used in a word-address expression has a bit offset and cannot be used. If the symbol is needed for a display, get its address using the WHERE directive and use that. From SID. Class, informative.

**** ERROR: symbol is too long
A symbol in a directive was longer than the longest valid symbol that could be expected in that position. Check to see if blanks were left out between input tokens. From SID. Class, informative.

**** ERROR: symbol search is lost
A SID logic error caused the wrong symbol table record to be searched. Report the problem to a Cray Research analyst. From SID. Class, informative.

**** ERROR: symbol was not found
A variable or label was not found in the symbol table record for the subprogram or common block last set up as the default with the MODULEBASE directive, or for the module whose name preceded the variable or label. Use the MODULEBASE directive to specify the subprogram in which the symbol is defined and try again. From SID. Class, informative.

**** ERROR: the keyword AT is required
The X variable in a STORE directive was not followed by the required keyword AT. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: the keyword COUNT is required
A masked PATCH or masked DISPLAY directive did not include the keyword COUNT between the number of the first bit and the bit count. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: the keyword FOR is required
The display count in an INSTRUCTIONS directive was not preceded by the keyword FOR. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: the keyword FROM is required
The X variable in a LOAD directive was not followed by the required keyword FROM. Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: the keyword IF is required
The parcel address in a BREAKPOINT or PACKAGE directive was followed by something other than the keyword IF. Check the syntax for the directive. From SID. Class, informative.

**** ERROR: the keyword WITH is required
A PATCH directive was entered without the keyword WITH between the first address to be patched and the list of patch values.

Check the syntax for the directive and try again. From SID. Class, informative.

**** ERROR: there is no address to scroll from
A FORWARD or BACKWARD directive was used when the last display did not use an address that could be used again, or before any DISPLAY directive was used. Use a DISPLAY directive to show the desired memory locations. From SID. Class, informative.

**** ERROR: this address overlaps another breakpoint
When a breakpoint is installed, two parcels of the user code are replaced by a jump to the debugger. Installing more than one breakpoint on a parcel can cause the wrong code to be replaced when breakpoints are removed. Remove the earlier breakpoint that the new one overlaps or find a different location for the new breakpoint. From SID. Class, informative.

**** ERROR: this format not allowed for masked display
An invalid display format was used with a masked DISPLAY directive. Check the list of formats allowed for masked displays and try again. From SID. Class, informative.

**** ERROR: this value cannot be tested
The value to be tested against in the IF clause of a BREAKPOINT or PACKAGE directive is of an unrecognized type or a type that cannot be tested. See HELP BREAKPOINT for a list of value types that can be tested. From SID. Class, informative.

**** ERROR: too many subscripts in array reference
An array reference contained more subscripts than there were dimensions declared in the array. Correct the problem and try again. From SID. Class, informative.

**** ERROR: two-word formats not allowed in dual display
One of the formats DOUBLE or COMPLEX was used in a DISPLAY directive along with another format. Only 1-word formats are allowed in dual-format displays. To see a value in double precision or complex and in another format (for example, octal), use two DISPLAY directives. From SID. Class, informative.

**** ERROR: unbalanced parentheses
Unbalanced parentheses were used in a word-address expression. Correct the problem and try again. From SID. Class, informative.

**** ERROR: unrecognized base, octal used
A prefix other than O', D', and X' was used with a number, so octal was used as the base for converting the number. From SID. Class, informative.

**** ERROR: unrecognized directive
The first token in an input record or following a semicolon was not recognized as a directive name or a valid abbreviation of a directive name. Check the spelling of the directive or its abbreviation and try again. From SID. Class, informative.

**** ERROR: value is too large for bit field
The value given in a masked PATCH directive does not fit into the specified bit field. Check the values given for the field size and the patch value and try again. In mixed-base mode (the default), bit numbers and counts are decimal and bit field values are octal. From SID. Class, informative.

**** ERROR: value must be string or number
The value used in a PATCH directive was something other than a string or a recognized type of number. Check the list of the types of values that can be patched and try again. From SID. Class, informative.

**** ERROR: value must be unsigned integer
The value for a masked or parcel patch must be an unsigned integer. Negative values must be entered using their octal representation. Convert the value to its Boolean representation and try again. From SID. Class, informative.

**** ERROR: width must be COLUMN, SCREEN, or PAGE
Something other than COLUMN, SCREEN, or PAGE followed the keyword WIDTH. Try the directive again. Width specifiers can be abbreviated to C, S, and P. From SID. Class, informative.

**** ERROR: X or C variable required
One of the keywords INCREMENTVARIABLE or DECREMENTVARIABLE is followed by something other than an X variable; or SETCONDITION, RESETCONDITION, or TESTCONDITION is followed by something other than a condition variable. Check the correct syntax for the directive and enter the command again. From SID. Class, informative.

**** WARNING: existing breakpoint replaced
When the last breakpoint was installed, an earlier breakpoint with the same number was replaced. Before installing new breakpoints, use the ACTIVE directive to see numbers currently in use. From SID. Class, informative.

**** WARNING: subscript out of bounds
The subscript for an array is not in the range declared for the array. The subscript is still used and could cause the wrong location to be used. From SID. Class, informative.

**** WARNING: the first parcel has been moved at label *label*
When a breakpoint is installed, two parcels of the user code are replaced by a jump to the debugger. If the second parcel is referenced in a jump instruction, the second half of the jump instruction is used instead of the instruction that was replaced. Remove the breakpoint or make sure that no jump to the label is executed. From SID. Class, informative.

2.6 SKOL MESSAGES BEGINNING WITH *

Refer to SKOL Reference Manual, CRI publication SR-0033, for the correct usage of SKOL.

*---------- - ERROR - BAD PARAMETER NUMBER *c*
The character *c* following a # operator in the replacement portion of a macro definition is not a digit in the range from 1 to *n* where *n* is the number of formal parameters in the pattern portion of the macro. Correct the replacement portion of the macro definition. If necessary, turn on SKOL's macro trace feature. From SKOL. Class, fatal.

*---------- - ERROR - MISSING RIGHT BRACKET
An error occurred in a user-defined structural macro. Correct the user-defined macro or change the patterns of any user-defined macros that may conflict with the standard macros. From SKOL. Class, fatal.

*---------- - ERROR - UNCLOSED LOOP OR BLOCK
An error occurred in a user-defined structural macro. Correct the user-defined macro or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

*---------- - ERROR - UNCLOSED STRING
The program ended without the closing apostrophe for a literal string. Check the column between the sequence number and the nesting level for an apostrophe that indicates a literal string continuation. From SKOL. Class, fatal.

*---------- - ERROR - UNCLOSED COMMENT
The program ended without the closing quotation mark for a comment. Check the column between the sequence number and the nesting level for a quotation mark that indicates a comment continuation. From SKOL. Class, fatal.

*---------- - INVALID PARAMETER ON % DIRECTIVE; *n* ASSUMED
A translator directive that required a number in columns 2 and 3, or a plus or minus sign in column 2, contained invalid digits or a number that was outside the range allowed for the directive. In all directives, a plus sign is treated as a 2, a blank as a 1, and a minus sign as a 0. From SKOL. Class, caution.

*---------- COULDN'T ACCESS *dataset*
The named dataset, which may be either the source input (I) dataset, a supplementary input (M) dataset, or SKOLTXT itself, could not be accessed. For SKOL's implicit dataset access method to work, an edition of the dataset must exist with no ID and no read permission word. If an ID or R parameter is required, add an ACCESS statement to the control statement file. From SKOL. Class, fatal.

*---------- ERROR *n*: ARCHAIC FORM '.*x*.'
The program contains a FORTRAN-style relational operator .*x*., where *x* is EQ, LT, GT, LE, GE, or NE. Substitute the applicable SKOL relational operator from the set =, <, >, <=, >=, and <>. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD 'DO' PHRASE
The text found between the reserved word DO and the following colon does not conform to the rules for DO situations. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD 'FOR' PHRASE
The text found between the reserved word FOR and the following colon does not conform to the rules for FOR situations. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD BYTE DESCRIPTION '*x*'
A partial-word declaration in a WORD statement contains a badly formed byte specification. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD CONSTANT DEFINITION '*x*'
An item in a list of CONSTANT definitions is badly formed. Check for unbalanced parentheses or brackets. Only two apostrophes can be present: one at the beginning and one at the end. A single-letter suffix can follow the second apostrophe. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD FIELD '*x*'
In a record or table reference, the pointer and the field with which it is used do not belong to the same record or table. Declare the field if it is missing from the record or table. Check for a misspelling or a declaration whose scope is local to another major segment. Match the number of subscripts with the number of declared dimensions for both the pointer and the field. If the record or table reference is abbreviated for use in a WITH structure, do not nest WITH structures. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD KEYWORD PARAMETER '*x*'
In the invocation of a macro defined with a MACRO statement, a parameter contains no equal sign. Assign to all macro parameters values that use the notation keyword=value. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD SPECIFICATION OF SCALAR LIST FOR '*x*'
The declaration of a scalar type contains a badly formed constant or subtype specification. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD SUBTYPE SYNTAX '*x*'
In a type declaration, a subtype does not have the required form, "*subtype*=(*list*)". From SKOL. Class, fatal.

*---------- ERROR *n*: BAD SYNTAX AT '...*x*'
A SKOL keyword is followed by text that is badly formed or inapplicable for that keyword. "*x*" represents the remainder of the source statement that begins with the keyword. Terminate the statement with either a colon or a semicolon. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD SYNTAX IN BYTE PARAMETERS '*x*'
The BYTE pseudofunction has the wrong number of arguments; three are required. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD SYNTAX IN SCALAR RANGE '*x*'
A WHEN statement or an IN, FIRST, or LAST expression omitted the periods that separate the two integer expressions within the square brackets. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD SYNTAX
A SKOL keyword is followed by text that is badly formed or inapplicable for that keyword. Check for unbalanced parentheses or unbalanced square brackets. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD TABLE SYNTAX '*x*'
A table declaration contains a statement that does not begin with the required form, "WORD *n*:". From SKOL. Class, fatal.

*---------- ERROR *n*: BAD TYPE '*x*' IN FIELD LIST
A RECORD structure is badly formed or contains a type declaration that is not permitted within records. Check for a missing or misplaced colon or semicolon. Check for a misspelled type name. Within a record, the valid types are: CHAR; COMPLEX; DOUBLE; LOGICAL; INTEGER; POINTER (but not POINTER TO...); REAL; SMALL INTEGER (space required); WORD. From SKOL. Class, fatal.

*---------- ERROR *n*: BAD WORD DESCRIPTION '*x*'
The square brackets are missing or misplaced in a partial-word declaration in a WORD statement. From SKOL. Class, fatal.

*---------- ERROR *n*: CHAR '*x*' CANNOT BE OUTPUT
The identifier, *x*, which is declared to be a member of the CHAR scalar type, was used with the C format in an OUTPUT or REMARK statement, or was operated on by the built-in CHR function. The external or CHR form of a character is undefined if the character is declared as an identifier rather than as a single-character literal. Avoid reference to the external value of a character that is known only by its internal name. From SKOL. Class, fatal.

*---------- ERROR *n*: END OF 'THEN' CLAUSE EXPECTED
The symbol representing a THEN structure was popped off the macro symbol stack by an END-type statement at the wrong time. Either the ENDUNTIL statement is missing or another END-type statement has intervened. From SKOL. Class, fatal.

*---------- ERROR *n*: ERROR IN '*c*'
The character string "$%@" does not match any macro. Because this character string occurs commonly in the standard macros, a failure in the translation process is indicated, perhaps triggered by a user-defined macro. Correct the user-defined macro, or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

*---------- ERROR *n*: ILLEGAL BACKWARD ROUTINE REFERENCE '*x*'
An EXECUTE statement refers to a routine *x* that is defined earlier in the program. Organize the major segment to define each ROUTINE below the last reference to it. Place the MAIN segment at the top and define each SUBROUTINE or FUNCTION segment below the last reference to it, wherever possible. From SKOL. Class, fatal.

*---------- ERROR *n*: INCORRECT PLACE FOR STRING LENGTH
A character string containing a vertical bar does not match any macro pattern. Check for a badly formed substring specification. Since this character is normally associated only with substring specifications, it is not necessary to turn on the macro trace. From SKOL. Class, fatal.

*---------- ERROR *n*: LABEL LEFT ON STACK
Structures with no ending statement have caused the label stack to be unbalanced at the end of the program. Remove all structural errors from the program. If the message persists, check for a user-defined macro that fails to balance each @*n*@LS*d* operation with a corresponding @LU*d* operation. Turn on the macro trace. From SKOL. Class, fatal.

*---------- ERROR *n*: LABELS BEGINNING WITH 'I00' ARE RESERVED
The character string "I00" does not match any macro. If no such character string exists in the source input, a failure in the translation process is indicated, perhaps triggered by a user-defined macro. Use a different identifier. If this message appears to be false, caused by an error in a macro, turn on the macro trace feature. Then correct the user-defined macro, or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

*---------- ERROR *n*: LEFTOVER '<'
The character "<" does not match any macro. Check for a missing IF or ELSEIF. Since this character is not generated by any of the standard macros, it is not necessary to turn on the macro trace. From SKOL. Class, fatal.

*---------- ERROR *n*: LEFTOVER '>'
The character ">" does not match any macro. Check for a missing IF or ELSEIF. Since this character is not generated by any of the standard macros, it is not necessary to turn on the macro trace. From SKOL. Class, fatal.

*---------- ERROR *n*: LEFTOVER '!'
The character "!" does not match any macro. Because this character occurs commonly in the standard macros, a failure in the translation process is indicated, perhaps triggered by a user-defined macro. Correct the user-defined macro or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

*---------- ERROR *n*: LEFTOVER '$%*c*'
The character string "$%" does not match any macro. Because this character string occurs commonly in the standard macros, a failure in the translation process is indicated, perhaps triggered by a user-defined macro. Correct the user-defined macro, or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

*---------- ERROR *n*: LEFTOVER '$%%*c*'
The character string "$%%" does not match any macro. Because this character string occurs commonly in the standard macros, a failure in the translation process is indicated, perhaps triggered by a user-defined macro. Correct the user-defined macro, or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

*---------- ERROR *n*: LEFTOVER '?'
The character "?" does not match any macro. Because this character occurs commonly in the standard macros, a failure in the translation process is indicated, perhaps triggered by a user-defined macro. Correct the user-defined macro or change the patterns of any user-defined macros that conflict with the standard macros. From SKOL. Class, fatal.

*---------- ERROR *n*: MATCHING 'WHEN' STATEMENT EITHER BAD OR SUPPRESSED
An ENDWHEN statement matches an earlier WHEN statement because they are at the same nesting level. The WHEN statement is enclosed in conditional compilation brackets and is suppressed or refers to an undeclared scalar type. Fix the WHEN statement or ensure that the WHEN and ENDWHEN statements are both inside or both outside any conditional code brackets. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'COROUTINE'
An ENDCOROUTINE statement is not preceded by a matching COROUTINE statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'DATA'
The character string "/;" is not preceded by a matching DATA keyword. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'DO'
An ENDDO statement is not preceded by a matching DO statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'FOR'
An ENDFOR statement is not preceded by a matching FOR statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'FUNCTION'
An ENDFUNCTION statement is not preceded by a matching FUNCTION statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'IF'
An ELSE, ELSEIF, or ENDIF statement is not preceded by a matching IF statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'LINK'
An ENDLINK statement is not preceded by a matching LINK statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'LOOP'
A WHILE or an ENDLOOP statement is not preceded by a matching LOOP statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'MAIN'
An ENDMAIN statement is not preceded by a matching MAIN statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'REPEAT'
An ENDREPEAT statement is not preceded by a matching REPEAT statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'ROUTINE'
An ENDROUTINE statement is not preceded by a matching ROUTINE statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'SUBROUTINE'
An ENDSUBROUTINE statement is not preceded by a matching SUBROUTINE statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'UNTIL'
A THEN or an ENDUNTIL statement is not preceded by a matching UNTIL statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'WHEN'
An ENDWHEN statement is not preceded by a matching WHEN statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'WHILE'
An ENDLOOP statement is not preceded by a matching WHILE statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING 'WITH'
An ENDWITH statement is not preceded by a matching WITH statement. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING '['
A statement beginning with [is not part of a scalar case structure (WHEN block) or situation case structure (THEN block). From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING '['
A WHEN statement is not immediately followed by a scalar case label. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING PARENTHESES AROUND 'FOR' CLAUSE
A FOR clause, which corresponds to an implied DO-loop in FORTRAN, was identified by the reserved word FOR followed by a colon. Such a combination, which is permitted in INPUT, OUTPUT, REMARK, READ, WRITE, LOG, ENCODE, DECODE, and DATA statements, must be enclosed in parentheses. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING RIGHT PAREN IN INPUT/OUTPUT LIST
An INPUT, OUTPUT, or REMARK statement is badly formed. From SKOL. Class, fatal.

*---------- ERROR *n*: MISSING SITUATION LABEL
The THEN block in a situation case structure does not contain labels for all of the situations that are named in the UNTIL statement that begins the structure. From SKOL. Class, fatal.

*---------- ERROR *n*: NO PROCESS DECLARED FOR COROUTINE '*x*'
No PROCESS statement was encountered with *x* in its list of coroutines. Add a PROCESS declaration or change one that already exists. Check for a misspelling or a declaration whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: OBLIGATORY PARAMETER '*x*' MISSING
In the invocation of a macro defined with a MACRO statement, a parameter is omitted and has no default value assigned by the MACRO statement. Each invocation of a macro must explicitly assign a value to any parameter that has no default value. From SKOL. Class, fatal.

*---------- ERROR *n*: ONLY ONE ITEM PERMITTED IN A SIMPLE 'FOR' LIST
A simple FOR clause contains one or more commands, suggesting that a list of variables is present. The standard macros do not provide for the automatic addition of the implied subscript to more than one variable name. Remove the comma(s) or change the simple FOR clause to an extended FOR clause. From SKOL. Class, fatal.

*---------- ERROR *n*: PARAMETER OR LOCAL OUTSIDE SCOPE
The formal name of a parameter or local variable belonging to a recursive routine appears outside the scope of the routine. Do not apply formal names of parameters and local variables to ordinary variables. From SKOL. Class, fatal.

*---------- ERROR *n*: POINTER '*x*' UNDECLARED
A reference to a field in a record or table, or a NEW or FREE statement, refers to an unknown pointer variable. Check for a misspelling or a declaration whose scope is local to another major segment. If the pointer is declared as an array, use the subscript notation in all references to it. From SKOL. Class, fatal.

*---------- ERROR *n*: RECORD '*x*' UNDECLARED
A "POINTER TO..." declaration or a MAKEAVAIL statement references a non-record. Declare the record. Check for a misspelling or a declaration whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: ROUTINE '*x*' DEFINED BEFORE AN EXECUTE
A ROUTINE statement is not preceded by an EXECUTE statement that references it. Organize the major segment so that each ROUTINE is defined below the last reference to it. Organize the program so that, wherever possible, the MAIN segment is at the top and each SUBROUTINE or FUNCTION segment is defined below the last reference to it. From SKOL. Class, fatal.

*---------- ERROR *n*: ROUTINE OR COROUTINE '*x*' NOT DEFINED
A routine or coroutine reference was made but never defined with a ROUTINE...ENDROUTINE structure. Define the routine following the last reference to it. Check for a misspelling or a definition whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: SCALAR VALUE '*x*' NOT DECLARED
A label of the form "[*x*]:" in a scalar case structure does not match any of the scalar values or subtypes declared to belong to scalar type identified in the WHEN statement that begins the structure. From SKOL. Class, fatal.

*---------- ERROR *n*: SCALAR VALUE AND ']' MISSING
A label of the form "[*x*]:" was expected but not found in a scalar case structure. From SKOL. Class, fatal.

*---------- ERROR *n*: SCALAR VALUE REQUIRED IN BRACKETS
A label of the form "[*x*]:" was expected but not found in a scalar case structure. From SKOL. Class, fatal.

*---------- ERROR *n*: SITUATION LABEL '*x*' NOT DECLARED
A label of the form "[*x*]:" in a situation case structure does not match any situations named in the UNTIL statement that heads the structure. Check for a misspelling or a missing situation in the UNTIL statement. From SKOL. Class, fatal.

*---------- ERROR *n*: TABLE '*x*' UNDECLARED
A "POINTER TO TABLE..." declaration references a non-table. Declare the table. Check for a misspelling or a declaration whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: TOO MANY ENDS
Structures with no beginning statement have caused the macro symbol stack to underflow. This error indicates a failure in the translation process, perhaps triggered by a user-defined macro. Remove all structural errors from the program. If the message persists, check for a user-defined macro that conflicts with the standard macros. Turn on the macro trace. From SKOL. Class, fatal.

*---------- ERROR *n*: UNBALANCED STRUCTURE SIGNALLED BY '?%X*c*'
The ending statement required by a user-defined structure is missing and no macro exists with the pattern "?%X*c*", where *c* is the character that represents the structure on the macro symbol stack. Refer to the SKOL Reference Manual, CRI publication SR-0033, for an example of a simple REPEAT: ENDREPEAT structure. From SKOL. Class, fatal.

*---------- ERROR *n*: UNC *b* AT *e*
This is an incomplete form of the message, "UNCLOSED *beginning* FOUND AT *ending*". It is caused by a user-defined structural macro that is either misused or badly formed. If the error is not obvious, turn on SKOL's macro trace feature. To obtain the complete form of the message, provide two additional macros:

'$E%(UNC *b* '= '$E%(UNCLOSED *beginning* '
'$E%(UNC #@1 AT *e*)'= '$E%(UNC #1 FOUND AT *ending*)'

The first of these two macros must be placed in SKOLTXT preceding all the macros that are of the second type. From SKOL. Class, fatal.

*---------- ERROR *n*: UNC *b* FOUND AT *ending*
This is an incomplete form of the message, "UNCLOSED *beginning* FOUND AT *ending*". It is caused by a user-defined structural macro that is either misused or badly formed. If the cause is not obvious, turn on SKOL's macro trace feature. To obtain the complete form of the message, provide two

additional macros:

'$E%(UNC *b* '= '$E%(UNCLOSED *beginning* '

This macro must be placed in SKOLTXT with the other macros of the same type. From SKOL. Class, fatal.

*---------- ERROR *n*: UNCLOSED *beginning* AT *e*

This is an incomplete form of the message, "UNCLOSED *beginning* FOUND AT *ending* ". It is caused by a user-defined structural macro that is either misused or badly formed. If the cause is not obvious, turn on SKOL's macro trace feature. To obtain the complete form of the message, provide one additional macro:

'$E%(UNC #@1 AT *e*)'= '$E%(UNC #1 FOUND AT *ending*)'

From SKOL. Class, fatal.

*---------- ERROR *n*: UNCLOSED *beginning* FOUND AT *ending*

This message is caused by a user-defined structural macro that is either misused or badly formed. If the error is not obvious, turn on SKOL's macro trace feature. See the SKOL Reference Manual, CRI publication SR-0033, for the correct beginning and ending macro keywords. From SKOL. Class, fatal.

*---------- ERROR *n*: UNDECLARED CHARACTER '*c*'

A single-character literal without an R, L, or H suffix does not match any of the characters declared either explicitly or implicitly by the TYPE CHAR statement. Change or declare the character, delete the TYPE CHAR statement, or add an R, L, or H suffix to the literal. Check for a TYPE declaration whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: UNDECLARED COROUTINE OR PROCESS '*x*'

A reference was made to a routine or process that was not declared in a PROCESS statement. Add a PROCESS declaration or change one that already exists. Check for a misspelling or a declaration whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: UNDECLARED POINTER ARRAY '*x*'

A reference to a field in a record or table, or a NEW or FREE statement, refers to an unknown pointer array. Check for a misspelling or a declaration whose scope is local to another major segment. If the pointer is declared as a simple variable, avoid subscript notation in all references to it. From SKOL. Class, fatal.

*---------- ERROR *n*: UNDECLARED SCALAR TYPE OR SUBTYPE '*x*'

A WHEN statement or an IN, FIRST, or LAST expression references an undeclared scalar type. This message might appear more than once because WHEN statements and IN expressions generate references to both FIRST(*x*) and LAST(*x*). Check for a misspelling or a declaration whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: UNDECLARED STRING '*x*'

A string-manipulation statement has a non-string as its operand where a string is required. This message might appear more than once for a single operand because the macro expansion generates multiple references to LENGTH(*x*) and SIZE(*x*). Check for a misspelling or a declaration whose scope is local to another major segment. From SKOL. Class, fatal.

*---------- ERROR *n*: UNDEFINED FORMAT '*x*'

A READ, WRITE, or LOG statement references a format whose name is not known. Check for a FORMAT statement illegally following the reference to it. The FORMAT statement must be local to the major segment in which it is used. (If a format is used in more than one major segment, it must be redefined locally in each segment because the format's text is restricted to a local scope by FORTRAN.) From SKOL. Class, fatal.

*---------- ERROR *n*: UNDIMENSIONED STRING '*x*'

A string declaration does not have the required form, "*string*(*size*)". From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED '_'

A character string containing an underscore sign does not match any macro pattern. Check for a misspelling, a missing definition, or a declaration whose scope is local to another major segment. Since many standard macro patterns contain underscore signs, it might be necessary to turn on the macro trace feature to determine if this is a spurious message caused by an earlier error in the translation process. From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED ' '

A character string containing an up arrow does not match any macro pattern. Check for a missing period. If the up arrow is followed by a period, check for other syntax errors. Since this character is normally associated only with pointer references to records and tables, it is not necessary to turn on the macro trace. From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED '$'
A character string containing a dollar sign does not match any macro pattern. Check for a misspelling, a missing definition, or a declaration whose scope is local to another major segment. Since many standard macro patterns contain dollar signs, it might be necessary to turn on the macro trace feature to determine if this is a spurious message caused by an earlier error in the translation process. From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED '[' DISCARDED
A left bracket was not matched by any macro. This message may be spurious if it follows other error messages. Check for a badly formed IN expression, scalar case label, situation case label, or WORD declaration. From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED ']' DISCARDED
A right bracket was not matched by any macro. This message may be spurious if it follows other error messages. Check for a badly formed IN expression, scalar case label, situation case label, or WORD declaration. From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED COLON DISCARDED
A character string containing a colon does not match any macro pattern. Check for a misspelling, a missing definition, or a declaration whose scope is local to another major segment. Determine if a semicolon or colon is missing from the previous line. From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED END OF RECORD
The symbol representing a RECORD structure was popped off the macro symbol stack by an END-type statement at the wrong time. Either the ENDRECORD statement is missing or another END-type statement has intervened. From SKOL. Class, fatal.

*---------- ERROR *n*: UNEXPECTED END OF SCALAR CASE
The symbol representing a WHEN structure was popped off the macro symbol stack by an END-type statement at the wrong time. Either the ENDWHEN statement is missing or another END-type statement has intervened. From SKOL. Class, fatal.

*---------- ERROR *n*: UNTREATED SCALAR VALUE '*x*'
The scalar constant *x* is a member of the type named in a WHEN statement, but is not present as a label in the body of the scalar case structure; or an ELSE label is missing. Add either *x* or ELSE as a label in the WHEN block. From SKOL. Class, fatal.

*---------- ERROR *n*: WRONG NUMBER OF PARAMETERS FOR '*x*'
The recursive routine *x* is defined in a ROUTINE statement with parameters that do not match the number of asterisks used in the routine's RECURSIVE statement declaration. Change the declaration to match the definition, or vice versa. From SKOL. Class, fatal.

*---------- ERROR *n*: WRONG POINTER USAGE
The abbreviated form of a record or table reference is used outside a WITH structure. Explicitly name the pointer or provide a WITH structure. Avoid the nesting of WITH structures. From SKOL. Class, fatal.

*---------- WARNING *n*: 'AND' USED AS FUNCTION
The reserved word AND was used as a function rather than as a logical connective. Change the expression to avoid the function reference. Determine if AND was used out of context. From SKOL. Class, caution.

*---------- WARNING *n*: 'EQV' USED AS FUNCTION
The reserved word EQV was used as a function rather than as a logical connective. Change the expression to avoid the function reference. Determine if EQV was used out of context. From SKOL. Class, caution.

*---------- WARNING *n*: 'OR' USED AS FUNCTION
The reserved word OR was used as a function rather than as a logical connective. Change the expression to avoid the function reference. Determine if OR was used out of context. From SKOL. Class, caution.

*---------- WARNING *n*: 'XOR' USED AS FUNCTION
The reserved word XOR was used as a function rather than as a logical connective. Change the expression to avoid the function reference. Determine if XOR was used out of context. From SKOL. Class, caution.

*---------- WARNING *n*: BAD VALIDATE PARAMETER '*x*' IGNORED
A VALIDATE list contains an item *x* that does not have the form *v* or *-v*, where *v* must be one of the following: ALL; CALL; FOR; LOOP; RECUR_OVER; RECUR_UNDER; UNTIL; WHEN. Check for a misspelling or an illegal character such as +. From SKOL. Class, caution.

*---------- WARNING *n*: COLON SUPPLIED AFTER 'IS'
A WHEN statement was identified by its two keywords, WHEN and IS (with a colon between them), but the final colon is missing. Add the colon to ensure the

automatic indentation logic. From SKOL. Class, caution.

*---------- WARNING *n*: MISSING COLON AFTER ']'
A situation case label or a scalar case label was identified by its square brackets, but the colon is missing. Place a colon after the square brackets to ensure the automatic indentation logic. From SKOL. Class, caution.

*---------- WARNING *n*: SUPERFLUOUS PERIODS AROUND '*x*'
The pattern .AND., .OR., .XOR., or .EQV. was encountered. Remove the periods. From SKOL. Class, caution.

*---------- WARNING *n*: UNDECLARED CONDITIONAL CODE PREFIX '*x*' ASSUMED ENABLED
A conditional code segment was identified by the pattern "<<*x*:" but *x* had not yet appeared in a CONDITIONAL declaration. Check for a misspelling, a missing definition, or a declaration whose scope is local to another major segment. Add or relocate the required CONDITIONAL statement. From SKOL. Class, caution.

*---------- WARNING *n*: UNUSED ROUTINE, COROUTINE OR PROCESS '*x*'
A routine, coroutine, or process was declared or defined but never referenced. Remove or suppress the declaration or definition, or add a referencing statement. From SKOL. Class, caution.

2.7 CSIM MESSAGES

CSIM messages are divided into five classes: fatal, warning, informative, syntax, and internal. The first three classes generally indicate problems that show up when instructions are interpreted. Syntax and internal messages concern errors that prevent an instruction from being interpreted. In this section, descriptive paragraphs are indented to separate them from messages.

hdw is a nonexistent hardware type.
The hardware type specified on a START directive was not 1 for CRAY-1 machines or X for CRAY X-MP machines. The only permitted types are 1 and X. The command is ignored. Rerun, specifying the correct type. From CSIMINPT. Class, syntax.

hdw event queue is empty and *hdw* has no active channels.
No channels are active on the hardware denoted by *hdw*, which can be the IOP or the CPU, and CSIM has nothing to do. Operating system being simulated probably is hung and a station has not been activated. Simulation stops. From TIMERCHK, TIMERCLR, or IOPTCHK. Class, syntax.

istck internal error.
IOP emulator input time-event queue has probably been corrupted. *istck* can be PUTISTCK or GETISTCK. Simulation stops. From PUTISTCK and GETISTCK. Class, syntax.

Activity on unassigned channel *chan*
The simulated operating system issued a CA instruction for a channel that has no device assigned to it. The channel error flag is set and an interrupt is sent. From CSIMCPU. Class, syntax.

Activity on unassigned controller unit *u*, channel *c*
The simulated operating system issued a request for a unit that has no device attached. For this unit, set the Device Reservation Error status bit. From CSIMCPU. Class, syntax.

All CPUs in monitor mode and waiting on semaphores.
The system is deadlocked while in monitor mode. Simulation stops. From INTRPRTX. Class, fatal.

Bad ISCB in station input *inp*
The simulated station detected a bad internal input stream control byte. The simulation stops. See a Cray analyst. From STATIO. Class, internal.

Bad LCP received with message code: *msg*
Some LCP fields did not contain the correct data. If possible, the station sends an error LCP to the Cray mainframe; otherwise the LCP is ignored after attempting to process the stream control bytes. From STATIO. Class, syntax.

Bad location specified.
The memory/register modification request is not recognizable. The directive is ignored. Rerun with a valid address or register. From CSIMINPT. Class, syntax.

Bad OSCB in station output *outp*
The simulated station detected a bad internal-output SCB (stream control byte). The simulation stops. See a Cray analyst. From STATIO. Class, internal.

Breakpoint not found at P=*addr*
A breakpoint was encountered but was not found in the tables. The tables are probably corrupted. The breakpoint is ignored. From CMDOUT. Class, internal.

C and P2 are mutually exclusive, P2 used.
Both C and P2 keywords were specified on a TRACE directive. The value specified for P2 on a TRACE is used and the C parameter is ignored. From CSIMINPT. Class, syntax.

Call level is above maximum of *m*.
DCALL ignored.
Too many alternative input datasets were specified without returning. From CSIMINPT. Class, syntax.

Channel *c* is not a station channel.
Routine STATDE:A was called with a channel that is not assigned to a station. The tables are probably corrupt. The simulation stops. See a Cray analyst. From STATDELA. Class, internal.

Channel must be specified for this DCU.
On the initial directive for the DCU, the channel pair number was not specified. The directive is ignored. From CSIMINPT. Class, syntax.

Channel request on already active channel *c*
The simulated system issued a second I/O request before the first request was complete. The channel error flag is set; the request is ignored. From CSIMCPU. Class, syntax.

Checkpoint dataset name conflict.
CHECKPT SUPPRESSED.
The dataset name for a checkpoint dataset is the same as a dataset listed on the CSIM control statement. The directive is ignored. Correct the name, and rerun. From CHECKPT. Class, syntax.

Checkpoint dataset name conflict. RELOAD SUPPRESSED.
The dataset name for checkpoint dataset is the same as a dataset listed on the CSIM control statement. The directive is ignored. Correct it and rerun. From RELOAD. Class, syntax.

Checkpoint file has too many DSU's.
A checkpoint dataset contains more DSUs than the maximum allowed. Reload aborts. From RELOAD. Class, syntax.

Clusters nonexistant on 1S hardware.
A display of the cluster register was requested during a 1S simulation. Display suppressed From SCREEN. Class, syntax.

COS and OTHER are mutually exclusive.
Both keywords were specified on the same directive. Only specify one keyword. From CSIMINPT. Class, syntax.

CPU event queue is empty..., see *hdw* event queue is empty... (start of section).

CPU-*n* interprocessor interrupt sent to nonexistent CPU-*m* at P=*addr*1 IBA=*addr*2.
CPU-*n* tried to send an interrupt to CPU-*m* Instruction ignored. From INTRPRIX. Class, warning.

CPU-*n* changing to unknown cluster number at P=*addr*1 IBA=*addr*2
CPU-*n* tried to go to cluster *c*. Instruction ignored. From INTRPRTX. Class, warning.

CPU-*n* is not deadstarted and CPU-*m* is in wait state at P=*addr1* Base address = *addr2*
One CPU of the simulated CRAY X-MP issued a test-and-set instruction for a semaphore that was already set, before the other CPU had been started. If CPU-*m* is in user mode, a deadlock is forced and an exchange is made. If CPU-*m* is in monitor mode, the simulation stops. From INTRPRTX. Class, warning.

Dataset not found - *dn*
The dataset specified on SUBMIT or SAVE is not local to the job. The command is ignored. Rerun with a dataset accessed before invoking CSIM. From STATCMD. Class, syntax.

Dataset size greater than maximum of *n*
The maximum dataset size specified by the DSZ parameter on the DEFSTAT command is too large. The parameter is ignored. From CSIMIUPT. Class, syntax.

DCU number greater than maximum DCU's allowed (*n*).
An attempt was made to configure a DCU with a number greater than the maximum. The command is ignored. From CSIMINPT. Class, syntax.

DCU-*n* does not exist.
The DCU number specified on a DSU directive does not exist. The command is ignored. From CSIMINPT. Class, syntax.

DCU-4 controller detected error. Code = *c*
DCU-4 controller detected an error. Simulation stops. From IOSFUNC. Class, syntax.

DD19 and DD29 may not be specified together. DD19 assumed.
DD19 and DD29 were both specified on a DSU directive. DD19 is assumed. From CSIMINPT. Class, syntax.

DD29 device assigned to DCU-2 controller.
A DD29 keyword was specified on a DSU directive, but the DSU is assigned to a type 2 controller. From CSIMINPT. Class, syntax.

DD49TQ internal error.
The DD49 command queue is corrupt. Simulation stops. From DD49TQ. Class, internal.

DEADLOCK in CPU-*n* cluster - *c*, semaphore - *s*
All CPUs in a cluster are waiting on a test-and-set instruction. Force a deadlock exchange in all CPUs. From INTRPRTX. Class, informative.

Deadstart error: Error in reading system dataset.
The OPSYS dataset has not been successfully read. A possible system error. The directive is ignored. Make sure the dataset that is being loaded is correct, and rerun. If the problem persists, see a Cray analyst. From DEADSTRT. Class, syntax.

Deadstart error: Error in reading parameter dataset.
The OSPAR dataset was not read successfully. There might be a system error. The directive is ignored. Make sure the dataset that is being loaded is correct, and rerun. If the problem persists, see a Cray analyst. From DEADSTRT. Class, syntax.

Directive *verb* not found.
The directive name specified on an ENABLE, DISABLE, or REMOVE could not be located. The directive is ignored. Check the spelling, correct it, and rerun. From CSIMINPT, CMDOUT. Class, syntax.

Directive - *verb*. Keyword - *kw* must be specified.
No value was specified for the required parameter. The directive is ignored. Supply the required keyword or positional parameter and rerun. From PARSE. Class, syntax.

DSU - *ldn* not found.
The DSU labelled with ldn is not known to CSIM. From CSIMINPT. Class, syntax.

DSU assigned to undefined DCU.
A DSU directive assigned the DSU to an undefined DCU. From CSIMINPT. Class, syntax.

DSU-*ldn1* already on unit *u*, DCU-*d*. DSU-*ldn1* assigned to unit *u*, DCU-*d*, and DSU-*ldn2* assigned to unit *v*, DCU-*e*.
A DSU was assigned to DCU and unit that already had a DSU assigned to it. The second assignment is changed to DCU=*e*, unit=*v*. From CSIMINPT. Class, syntax.

Error in *verb* tape dataset. Error is *error*
The expander tape file cannot be read or opened. *verb* can be opening or reading. *error* contains the DSP error flags. Error status is returned to the IOP. From CTAPE. Class, syntax.

Error in CMEM. Illegal request.
CMEM was called with bad parameters. No virtual memory is supplied. From CMEM. Class, internal.

Error in CMEM. Illegal type.
CMEMINIT was called with an invalid memory type. No virtual memory is supplied. From CMEMINIT. Class, internal.

Error in CMEM. I/O error.
There is probably a system error or insufficient disk space. CSIM continues if possible. Rerun the job. If the problem persists, see a Cray analyst. From CMEM, CMEMINIT. Class, syntax.

Error in CMEM. Too much VM requested for type *type*
A request was made to CMEMINIT for more virtual memory than is allowed for the specified memory type. No virtual memory is supplied. From CMEMINIT. Class, internal.

Error in cracking directive, STATUS = *st*
Routine CRACK detected an error. For information on status, see the Library Reference Manual, CRI publication SR-0014. Correct it and rerun. The directive is ignored. From PARSE. Class, syntax.

Error in directive *verb*. Module *mod* and symbol *sym* not found.
The module and symbol cannot be found in the specified symbol tables. The directive is ignored. Check the spelling, and make sure the symbol table matches what is specified on the CSIM control statement. From CSIMINPT,CPADDR. Class, syntax.

Error in directive *verb*. Illegal character in *base* conversion of *value*.
An illegal character was included in a number, either octal or decimal (*base*). The directive is ignored. Check for illegal characters in the parameter. Correct and rerun. From CSIMINPT,CPADDR. Class, syntax.

Error in FROMCRAY. CHANNADD < 0 or NWORDS > SUBSEGL
    The simulated system has issued an I/O request with the channel limit less than zero and it was not previously detected or the number of words requested is too large. The simulation stops. From FROMCRAY. Class, syntax.

Error in MDW for SSD - *ssdnum*, SSD not reloaded.
    RELOAD detected a non-SSD memory descriptor word (MDW) when it expected the SSD. SSD is not reloaded. From RELOAD. Class, syntax.

Error in parcel conversion. Parcel address greater than 24 bits. See a Cray analyst.
    Routine INSTR detected a probable internal error. ILLEGAL INSTRUCTION message is returned. See a Cray analyst. From INSTR. Class, internal.

Error in PARSE.
    A separator or terminator character cannot be found. The directive is ignored. See an analyst. From PARSE. Class, internal.

Error in processing memory/register modification value.
    An internal error was detected. The directive is ignored. See a Cray analyst. From PARSE. Class, internal.

Error in reading *type* dataset. Error is *n*.
    An error occurred while the tapeload routine was being read into the MIOP. *type* can be IOP or tape. *n* is the flags from the DSP. Simulation stops. From DEADIOP, CTAPE. Class, syntax.

Error in routine INSTR. CIP=*cip* is in error. See a Cray analyst.
    The CIP tables in routine INSTR have been corrupted. ILLEGAL INSTRUCTION is returned. From INSTR. Class, internal.

Error in STATIO staging near 300 CONTINUE.
    The simulated station detected a corrupted staging buffer. The simulation stops. See a Cray analyst. From STATIO. Class, internal.

Error in station command argument number *n*.
    An illegal station command argument was specified. The command is ignored. From STATCM. Class, syntax.

Error in stream assignments.
    An open stream cannot be found, but there should be an open stream. The simulation stops. From STATIO. Class, internal.

Error in TOCRAY. CHANNADD < 0 or NWORDS > SUBSEGL.
    The simulated system has issued an I/O request with the channel limit less than zero and it was not previously detected or the number of words requested is too large. The simulation stops. From TOCRAY. Class, syntax.

Error in trap processing. Opcode = *op* is typed as memory reference.
    The SIMOP table has a bad entry. The trap output is suppressed. From CMDOUT. Class, internal.

Error while reading virtual memory. CHECKPOINT SUPPRESSED. CMEMACNT = *a* CMEMRCNT = *r*
    Possible system error. The directive is ignored. See a Cray analyst. From CHECKPT. Class, internal.

Error while reloading checkpoint dataset. RELOAD SUPPRESSED, ERROR = *n*
    Possible system error. The reload aborts. See BUFFER IN status in the Library Reference Manual, CRI publication SR-0014. Correct and rerun. From RELOAD, INSERT. Class, internal.

Error while writing checkpoint dataset. CHECKPOINT SUPPRESSED. ERROR = *n*
    Possible system error. The directive is ignored. Correct and rerun. From CHECKPT. Class, syntax.

File *fn* not found.
    File *fn* as specified on a DCALL directive is not a local dataset. From CSIMINPT. Class, syntax.

FLOATING POINT ERROR at P=*addr1* Base address = *addr2*
    A floating-point error was detected in the simulated system. In monitor mode, the simulation continues with the next instruction. In user mode when floating-point interrupts are enabled, an exchange is made; otherwise, the simulation continues with the next instruction. From INTRPRTX, INTERPRT. Class, warning.

GETISTCK internal error, see *istck* internal error (start of section).

I/O error in simulated system.
    The interpreter detected an I/O error. The simulation stops. From CSIMCPU and CSIMIOS. Class, syntax.

I/O error on simulated DSU dataset.
The simulated DCU cannot process the I/O request due to a possible system error. The I/O request is ignored and the channel is deactivated. Rerun; if the problem persists, see a Cray analyst. From CSIMCPU. Class, syntax.

I/O error while reading *dn*
An I/O error was detected when the station script dataset was read. A system error is possible. The simulation stops. From STATIO. Class, internal.

IF processing for TRAP in error.
An internal error was detected. The directive is ignored. From CSIMINPT. Class, syntax.

Illegal channel pair.
An illegal channel number is specified on a DCU directive. The command is ignored. From CSIMINPT. Class, syntax.

Illegal channel number-*c* at P=*addr1*
Base address = *addr2*
A channel instruction has been issued with a bad channel number. The request for I/O on channel *c* is ignored. From INTERPRT, INTRPRTX. Class, warning.

Illegal character in decimal conversion of TIMLIMIT. Set to 100 seconds.
A non-numeric character was specified for the T parameter on the CSIM control statement. The time limit is set to 100 seconds. From CRAYBOOT. Class, syntax.

Illegal character in *base* conversion for command *verb* parameter *param*
An illegal character was specified. *base* indicates either octal or decimal. The directive is ignored. From STATCMD. Class, syntax.

Illegal Cray channel request from IOP-n
Configuration problem or CSIM's internal tables have been corrupted. The IOP configuration tables do not agree with the CPU configuration tables on the Cray-MIOP or Cray-HSP channels. Simulation stops. From CFE and MEMCHAN. Class, syntax.

Illegal display type - *t*
The type specified on a DIS or ALT directive is unknown. The directive is ignored. From CSIMINPT. Class, syntax.

ILLEGAL EXCHANGE at P=*addr1* Base address = *addr2*.
A user-user exchange was attempted. The simulation stops. This also occurs if a RUN directive is issued before an operating system has been loaded into memory via the START or RELOAD commands. From INTERPRT, INTRPRTX. Class, Fatal.

Illegal EXITREQ from interpreter: *nn*
SIMULATOR ERROR.
Interpeter has returned a bad request. The simulation stops. See a Cray analyst. From CSIMCPU and CSIMIOS. Class, internal.

Illegal FORMAT specifier.
A bad format has been passed to the display routine. The directive is ignored. Check the format; if it is valid, see an analyst. From SCREEN. Class, syntax.

Illegal hardware combination in SNAP.
An unknown hardware type was specified. Directive was ignored. From CSIMINTP. Class, syntax.

Illegal information type requested.
A bad type has been passed to the display routine. The directive is ignored. Check the type; if it is valid, see an analyst. From SCREEN. Class, syntax.

ILLEGAL INSTRUCTION at P=*addr1* Base address = *addr2* Instruction is *c n*.
A nonstandard opcode is being simulated. From INTERPRT, INTRPRTX. Class, warning. *c* is the current instruction parcel; *n* is the next instruction parcel.

Illegal IOP request - *a*
The packet received from the CPU has a bad DID field. Simulation stops. From IOSFUNC. Class, syntax.

Illegal message code received from Cray: MC = *mc*
The message code received is invalid. The station sends an error LCP to the Cray mainframe. From STATIO. Class, syntax.

Illegal mode specified for type *t*
The mode and type for a display are inconsistent. The directive is ignored. From CSIMINPT. Class, syntax.

Illegal number for console.
A console number greater than the maximum was specified on a DEFCONS directive. From CSIMINPT. Class, syntax.

Illegal register number in *name*
A register display was requested with a bad register number The display is suppressed. From SCREEN. Class, syntax.

Illegal request on channel *c* channel address = *chaddr* channel limit = *chlim*
The simulated system has issued an I/O request with the channel limit less than the channel address. Channel error and interrupt flags set and request ignored. From CSIMCPU. Class, syntax.

Illegal request on SSD channel *ssdch*
During the simulation of a CRAY X-MP system, an improper sequence of CA instructions was detected. The instruction is ignored. From CSIMCPU. Class, syntax.

Illegal response on Cray-1 input stream: *nn*
The input stream control byte for stream *nn* is invalid. The station ignores the LCP. From STATIO. Class, syntax.

Illegal response on Cray-1 output stream: *nn*
The output stream control byte for stream *nn* is invalid. The station ignores the LCP. From STATIO. Class, syntax.

Illegal station channel *c* SIMULATOR ERROR
STATIO has been called with a bad channel number or tables corrupted. The simulation stops. See a Cray Research analyst. From STATIO. Class, internal.

Illegal status *status* for station *statnum*
A status other than SENDLCP, SENDSEG, RECLCP, or RECSEG has been set for station. Tables are probably corrupted. Simulation stops. See analyst. From STATIO. Class, internal.

Illegal task pointer: *nn* SIMULATOR ERROR
Simulator task tables have been corrupted. The simulation stops. See a Cray analyst. From CSIMCPU. Class, internal.

Illegal task pointer, IOP-*n*, channel-*c*
Simulator task tables have been corrupted or configuration is in error. Simulation stops. From CSIMIOS. Class, syntax.

Illegal type *type* specified on checkpoint/reload.
Valid types are DISK and ALL. The directive is ignored. Correct and rerun. From CSIMINPT. Class, syntax.

Illegal type specified for DCU.
An illegal DCU type was specified on a DCU command. Type 2 assumed. From CSIMINPT. Class, syntax.

Illegal type specified for TRACE directive.
Output processing detected an illegal type. The directive table is probably corrupt. TRACE is suppressed. See a Cray analyst. From CMDOUT. Class, internal.

Illegal type *type* specified on *dir* directive.
dir can be TRACE or TRAP. Valid types are R, W, and RW. The directive is ignored. Rerun with correct type specified. From CSIMINPT. Class, syntax.

Illegal unit number.
A unit number greater than 3 was given on a DSU directive. The command is ignored From CSIMINPT. Class, syntax.

Increment for vector memory reference is zero at P=*addr1* Base address = *addr2*.
Increment for vector memory reference is 0. From INTERPRT, INTERPRTX. Class, informative.

Invalid checkpoint dataset name. CHECKPT SUPPRESSED.
A probable CSIM error: the dataset name is binary zero. The directive is ignored. See a Cray analyst. From CHECKPT. Class, syntax.

Invalid checkpoint dataset name. RELOAD SUPPRESSED.
A probable CSIM error: the dataset name is binary zero. The directive is ignored. See a Cray analyst. From RELOAD. Class, syntax.

Invalid dataset name - *name*
An invalid or incorrect local dataset name was specified. Directive ignored. From CSIMINPT. Class, syntax.

Invalid disk request.
The simulated DCU cannot process the I/O request due to a request error. The I/O request is ignored and the channel is deactivated. From CSIMCPU. Class, syntax.

Invalid IOPA call on drive control call, IOPA=*accum*
The accumulator contained an invalid value. Simulation stops. From SDSKS. Class, syntax.

Invalid IOPA value on status select call, IOPA=*accum*
The accumulator contained an invalid value. Simulation stops. From SDSKS. Class, syntax.

Invalid parameter for keyword - *keyword*
A probable CSIM error: no separator or terminator follows the parameter. See a Cray analyst. The directive is ignored. From PARSE. Class, syntax.

Invalid register specification in *param*
An illegal register number was specified on a SNAP directive. Directive ignored. From CSIMINPT. Class, syntax.

IOP event queue is empty..., see *hdw* event queue is empty... (start of section).

IOPA invalid. DSUNUM=*dsu* IOPA=*accum*
On a read or write request to a DCU-5 controller, bits are set in the accumulator which must be zero. Simulation stops. From SDSKS. Class, syntax.

IOP-*n*. Block mux not supported by CSIM.
CSIM does not currently support the Block Multiplexer. The Channel Busy and Done flags are set. From CSIMIOS. Class, syntax.

IOP-*n*. Error *error* on channel *c*, function *f*
The DCU-4 controller detected an error. Simulation stops. From CSIMIOS. Class, syntax.

IOP-*n*. No device assigned to channel *c*
The configuration is bad. The Channel Busy and Done flags are set. From CSIMIOS. Class, syntax.

IOP-*n*. Unknown function *f* on channel *c*
An unknown I/O function was specified for the channel. Simulation stops. From CSIMIOS. Class, syntax.

IOS emulator input stack overflow.
More than 73 input packets were queued to be processed by IOSFUNC. CPU is probably in a loop. Simulation stops. From PUTISTCK. Class, internal.

IOS emulator output stack overflow.
More than 85 ouput packets were queued to be sent to the CPU. The CPU is probably not activating its input channel. Simulation stops. From PUTOSTCK. Class, internal.

KERNEL, STATION, and UNDEF are mutually exclusive.
At least two of the keywords were specified on the same directive. Specify only one keyword. From CSIMINPT. Class, syntax.

Keyword not found - *kw*
Keyword *kw* is not valid for the directive. The directive is ignored. From PARSE. Class, syntax.

Maximum dataset size cannot be redefined after LOGON.
The DSZ keyword was specified on a DEFSTAT directive after the station logged on. The command is ignored. From STATCMD. Class, syntax.

MONITOR ERROR EXIT at P=*addr1* Base address = *addr2*.
An error exchange was attempted while CSIM was in monitor mode, probably trying to execute data. The simulation stops. From INTERPRT, INTRPRTX. Class, Fatal.

MONITOR ONLY INSTRUCTION at P=*addr1* Base address = *addr2* Instruction is *contents*
The user code included a privileged instruction. The instruction is ignored. From INTERPRT, INTRPRTX. Class, warning.

New DSU specified. Current DCU and unit are occupied and all DCU's are full. DSU not assigned.
A DSU directive was issued, but there is no place to put the DSU. The DSU is defined, but not assigned to a DCU. From CSIMINPT. Class, syntax.

No console for IOP-*n* channel *c*
There might be a configuration problem. The IOP software expects a console on channel *c*, but CSIM was not informed of this. The request is ignored. From CONSOLE. Class, syntax.

No DNT exists for the *dn* dataset.
The OPSYS or OSPAR datasets have not been accessed. The directive is ignored. Access the datasets and rerun. From DEADSTRT and DEADIOP. Class, syntax.

Number of operator messages cannot be redefined after LOGON.
The MRE keyword was specified on a DEFSTAT directive after the station logged on. The command is ignored. From STATCMD. Class, syntax.

Number of operator messages greater than maximum of *maxos*
The value of the MRE parameter on a DEFSTAT directive is too large. The parameter is ignored. From CSIMINPT. Class, syntax.

Number of subsegments/segment cannot be redefined after LOGON.
The NSSG keyword was specified on a DEFSTAT directive after the station logged on. The command is ignored. From STATCMD. Class, syntax.

ON and OFF are mutually exclusive.
Default value used for LOGMSG.
Both ON and OFF were specified on a LOGMSG directive. From CSIMINPT. Class, syntax.

OPERAND RANGE ERROR at P=*addr1* Base address = *addr2*
An operand range error was detected in the simulated system. In user mode, an exchange is made. In monitor mode, the simulation stops. From INTERPRT, INTRPRTX, IOPINTRP. Class, warning.

Operating system not loaded, checkpoint suppressed.
A CHECKPT directive was issued before a START or RELOAD directive. The directive is ignored. From CSIMINPT. Class, syntax.

Parameter longer than 8 characters. Bad card is: *cardimage*
The display increments on + or - are too large. The directive is ignored. From PARSE. Class, syntax.

Parameter too long, truncated to 16 characters *value*
The station command includes a parameter that is too long. Informative only. From PARSE. Class, syntax.

PROGRAM RANGE ERROR at P=*addr1* Base address = *addr2*.
A program range error was detected in the simulated system. In user mode, an exchange is made. In monitor mode, the simulation is stopped. From INTERPRT, INTRPRTX, IOPINTRP. Class, warning.

PUTISTCK internal error, see *istck* internal error (start of section).

Referenced vector element *vnum* is greater than vector length *vl* near P= *addr1* Base address = *addr2*
The referenced vector element *vnum* is greater than vector length *vl*. From INTERPRT, INTRPRTX. Class, informative.

REG may not be specified with MOS or SSD
For SSD or MOS hardware, a register snap which has no meaning was specified. The directive is ignored. From CSIMINPT. Class, syntax.

RELOAD ABORTED. (HEADER)
The specified checkpoint dataset lacks a proper header or does not exist. The directive is ignored. From RELOAD. Class, syntax.

RELOAD ERROR.
Compression and/or type code in MDW do not match expected codes.
Expected comp. = *ccode*
Actual comp. = *ccode*
Expected type = *tcode*
Actual type = *tcode*
The specified checkpoint dataset contains unexpected data, or does not contain expected data. The reload aborts. From RELOAD. Class, syntax.

Reprieve abort - SEE SIMULATOR ANALYST
A bad reprieve status was received. CSIM aborts. From SIMREPV. Class, syntax.

Requested dataset - *dn* is not a local dataset.
An ACQUIRE was issued by COS. The requested dataset must be local to the CSIM run. An LCP is sent to COS stating that the dataset does not exist. Access the dataset before invoking CSIM and rerun. From STATIO. Class, syntax.

Segment received for LCP code - *nn*. No segment should have been sent.
A possible CSIM error. The simulation stops. See a Cray Research analyst. From STATIO. Class, syntax.

Segment size greater than maximum of *maxseg1*
The segment size is too large, as calculated from the subsegment size and the number of subsegments per segment. From CSIMINPT. Class, syntax.

Semaphore number is too large at P=*addr1* Base address = *addr2*
A semaphore number greater than 31 was encountered in the simulated system. The instruction is ignored. From INTRPRTX. Class, warning.

SIM and FUNC are mutually exclusive.
Both keywords were specified on the same directive. Specify only one keyword. From CSIMINPT. Class, syntax.

SSD not found on channel *ch*
SSD channel table is corrupted. From SSD. Class, syntax.

SSD size too large; set to maximum of *maxssdsz*
The size parameter specified on a DEFSSD directive is larger than the maximum allowed. The size is set to the maximum allowed. From CSIMINPT. Class, syntax.

Station *id/tid* already logged on.
SET ignored.
A SET command was issued after the station logged on. The command is

ignored. From STATCMD. Class, syntax.

Station and request status do not match for *id*
The emulator station is not synchronized with the CPU. Example: The CPU expects input LCP, but the emulator station is ready to send an input segment. Simulation stops. From IOSFUNC and CFE. Class, syntax.

Station packet received for ID *id* but ID not initialized.
The CPU requested communication with a station before the station requested to LOGON. An error status is returned to the CPU. From IOSFUNC. Class, syntax.

Station slot may not be redefined after logon.
The SLOT parameter on the DEFSTAT command was specified after the station logged on. The parameter is ignored. From CSIMINPT. Class, syntax.

Station type may not be redefined after LOGON.
The STYP keyword was specified on a DEFSTAT directive after the station was logged on. The command is ignored. From STATCMD. Class, syntax.

Station type must be between 0 and 3.
An illegal value is shown for the STYP parameter on the DEFSTAT command. The parameter is ignored. From CSIMINPT. Class, syntax.

Status channel *sch* not active, but SSD transfer complete.
During the simulation of a 1S system, the status channel was not activated before the transfer was complete. The channel error flag is set. From SSD. Class, syntax.

STP base address not found in initial S5 register. EXEC, STP, JOB modes disabled.
If the system is running COS level 1.11 or earlier, the modes named are disabled. The CSIM flag is placed in location 176_8 of the simulated memory. All mode processing specific to COS is disabled. From DEADSTRT and CSIMIOS. Class, syntax.

STP length not found, JOB mode disabled.
CSIM could not find the string 'STP ENDS AT'. JOB mode is disabled. From LOADREG and CSIMIOS. Class, syntax.

Stream parameters may not be reset by DEFSTAT command after logon. Use STREAM.
A DEFSTAT directive contains parameters to reset the stream definitions after the station logs on. The parameter is ignored. From CSIMINPT. Class, syntax.

Subsegment size greater than maximum segment size of *maxsegl*.
The subsegment length on a DEFSTAT directive is too large. The command is ignored. From STATCMD. Class, syntax.

Subsegment size may not be redefined after LOGON.
The SSGZ keyword was specified on a DEFSTAT directive after the station logged on. The command is ignored. From STATCMD. Class, syntax.

Too many *stype* streams specified. Maximum is *max*
An attempt was made to set too many streams of type *stype*, which can be input, output, or active. The parameter is ignored. From STATCMD, CSIMINPT. Class, syntax.

Too many disk sectors specified. Maximum of *dsmax* assumed.
The NS keyword of the DSU directive specifies too many sectors. The command is processed with the maximum number of sectors. From CSIMINPT. Class, syntax.

Too many DSU's specified. Maximum number of DSU's allowed is *numdsu*
Too many DSUs are defined. The directive is ignored. From CSIMINPT. Class, syntax.

Too many SSD's defined. Maximum is *numssd*
An attempt was made to define more SSDs than are allowed. The directive is ignored. From CSIMINPT. Class, syntax.

Too many stations defined. Maximum is *maxstat*.
The most recent DEFSTAT directive creates a new station that puts the number of stations above the maximum. The command is ignored. From STATCMD. Class, syntax.

Unknown device address - *addr* on expander channel.
A device address other than the address for the expander tape was detected. The Channel Busy and Done flags are set. From CSIMIOS. Class, Syntax.

Unknown device type *type* on DSU directive.
A device type other than DD-19, DD-29, or DD-49 was specified on a DSU directive. Directive is ignored. From CSIMINPT. Class, syntax.

Unknown disk I/O function *f*
The CPU has sent an unknown function request for a particular DSU. Simulation stops. From IOSFUNC. Class, syntax.

Unknown ID - *id* encountered by *rtn*
The station ID specified is not known to CSIM. *rtn* can be IOSFUNC or CFE. Simulation stops. From IOSFUNC and CFE. Class, syntax.

Unknown message error code = *code*
An error LCP was received with an unknown error code. The simulation stops. From STATIO. Class, syntax.

Unknown packet request - *a*
The packet received from the CPU has an unknown DID field. From IOSFUNC. Class, syntax.

Unknown register type - *type* found in CONSOLE.
The channel mnemonic is not TI or TO. The configuration is probably bad. The request is ignored. From CONSOLE. Class, syntax.

Unknown type - *type* found in CONSOLE.
The channel mnemonic is not TI or TO. The configuration is probably bad. The request is ignored. From CONSOLE. Class, syntax.

Unrecognized station command or directive - *verb*
The specified verb is invalid. The command is ignored. From STATCMD. Class, syntax.

UP and DOWN may be specified together. The command is ignored.
UP and DOWN were both specified on either a DCU or DSU directive. The directive is ignored. From CSIMINPT. Class, syntax.

VECTOR FLOATING POINT ERROR at P=*addr1*
Base address = *addr2*
A floating-point error has been detected in the simulated system. In monitor mode, the simulation continues with the next instruction. In user mode when floating-point interrupts are enabled, an exchange is made; otherwise, the simulation continues with the next instruction. From INTRPRT, INTRPRTX. Class, warning.

2.8 STARTUP MESSAGES

Startup messages are divided into three subsections:

- An alphabetic list of Startup messages, beginning with variable name (*var*) messages
- Startup messages going to System Log only
- Two-part Startup messages that begin with ***** FATAL STARTUP ERROR *****

2.8.1 STARTUP MESSAGES (ALPHABETIC)

var DATASETS WERE MARKED IN ERROR DUE TO *errortext*
OF THESE *count* WERE DELETED
The specified number of permanent datasets had an error flag set in the DSC entry due to the reason given in error-text. The value of count can be a decimal number or the word ALL or NONE, and specifies the number of datasets which were deleted due to operator replies to earlier messages. This message is always preceded by one or more other messages identifying error conditions detected for specific datasets. The value of *error-text* may be one of the following:
RESIDING ON A RELEASED OR MISSING DEVICE
RESIDING ON A DOWN DEVICE
AI CONFLICTS
CATASTROPHIC ERRORS
MULTITYPE DATASETS WITH AI CONFLICTS
MULTITYPE DATASETS WITH QDT INDEX OUT OF RANGE
This message appears on the master operator console only. From Startup.

device DEVICE IS EITHER UNAVAILABLE OR IS OFF.
REPLY WITH THE (DXT,DV=,SZ=,OVF=,CAI=) COMMAND TO CONTINUE.
The device specified to contain the DXT is either a read-only device or is marked unavailable in the EQT. Specify new options via the DXT command, COS Operational Procedures Reference Manual, publication number SM-0043. From Startup. This message appears on the master operator console only. Class, warning.

var DEVICE LABEL NOT FOUND
STARTUP was unable to locate a device label on the specified device (*var*), and the UP flag was not set in the EQT entry for the device. Determine whether the device is functioning properly. If not, correct the hardware problem and attempt to start the system again. If the device is functioning properly, the device label has been destroyed or the track containing it has become unreadable. A new label must be written on the device using WDL=Y, a CONFIG parameter file command. If the device was not properly released before the label was destroyed, one or more permanent datasets might be lost. If the device is the system master device, the WDL=Y command cannot be used, and an install type of startup is necessary. ZLOG buffer only. From Startup. Class, fatal.

device DEVICE LABEL UPDATE FLAG SET *master*.
REPLY 'GO' TO RE-WRITE LABEL.
REPLY 'SKIP' TO IGNORE UPDATE FLAG.
The STARTUP task is preparing to rewrite the device label for the named device.

This is usually caused by a change in the flaws specified via parameter file *FLAW or *DELFLAW directives. In some cases it may be caused by allocation of datasets for use by the I/O Subsystem, or by reallocation of existing system datasets whose allocation information is contained in the device label (system dump area, Dataset Catalog). If the device is the master device, the 'master' indicator above will be '(MASTER)'. If it is not the master device, the indicator will contain blanks. If the reply is not GO or SKIP the message will be displayed again with the additional line
text INVALID. CORRECT AND RE-ENTER.
This message appears on the master operator console only. From Startup. Class, warning.

device DEVICE NAME NOT FOUND IN ANY EQT.
REPLY WITH THE (DXT,DV=,SZ=,OVF=,CAI=) COMMAND TO CONTINUE STARTUP.
No device has sufficient space to contain the DXT without overflow. Specify new options via the DXT command, COS Operational Procedures Reference Manual, publication number SM-0043. This message appears on the master operator console only. From Startup. Class, warning.

var FILE ACCESS FAILED *status*
Permanent Dataset Manager returned the specified error status when STARTUP attempted to access a dataset named in the $SDR dataset. Determine the cause of the error and correct it. The dataset must then be entered into the System Directory by a separate job. System log only. From Startup. Class, caution.

device HAS INSUFFICIENT CONTIGUOUS SPACE FOR THE $DSC-EXTENSION DATASET.
REPLY WITH THE (DXT,DV=,SZ=,OVF=,CAI=) COMMAND TO CONTINUE STARTUP.
Startup cannot allocate the $DSC-EXTENSION dataset contiguously on the named device. Specify new options for allocation of the dataset via the DXT command, described in the Operational Procedures Reference Manual, CRI publication number SM-0043. This message appears on the master operator console only. From Startup. Class, warning.

device HAS INSUFFICIENT SPACE FOR THE $DSC-EXTENSION.
REPLY WITH THE (DXT,DV=,SZ=,OVF=,CAI=) COMMAND TO CONTINUE STARTUP.
Startup is unable to allocate sufficient space on the named device to contain the $DSC-EXTENSION without overflow. Specify new options for the DXT allocation via the DXT command, Operational Procedures Reference Manual, CRI publication number SM-0043. This message appears on the master operator console only. From Startup. Class, warning.

device HAS NO SPACE. CANNOT START THE $DSC-EXTENSION ON IT.
REPLY WITH THE (DXT,DV=,SZ=,OVF=,CAI=) COMMAND TO CONTINUE STARTUP.
The named device is full, and the current options for allocation of the DXT specify that device to begin the DXT. Specify new options for the DXT allocation via the DXT command, Operational Procedures Reference Manual, CRI publication number SM-0043. This message appears on the master operator console only. From Startup. Class, warning.

device IS THE MASTER DEVICE AND CANNOT BE DEFINED AS A MEMBER OF DISK GROUP *sname* REPLY 'GO' TO CONTINUE WITH DEVICE REMOVED FROM GROUP
During construction of device group tables, Startup detected that the master device has been specified to be a member of a stripe group. This is illegal. Reply GO to continue Startup with the master device removed from the named group. This message appears on the master operator console only. From Startup. Class, caution.

device name - NO ENGINEERING FLAW TABLE FOUND ENTER GO TO CONTINUE STARTUP SKIP TO CONTINUE W/O FURTHER WARNINGS
Startup was unable to locate the Engineering Flaw Table on the named device. Notify the engineers to have EFT recreated; or reply GO to continue Startup with no EFT data for the named device; or reply SKIP to continue Startup with no EFT data for the named device and no further warnings of missing EFTs. From Startup. Class, informative.

var UNABLE TO READ DEVICE LABEL ON *num* TRACKS.
I/O ERROR STATUS WAS RETURNED ON *errcnt* OF THE TRACKS.
REPLY 'GO' TO CONTINUE WITH DEVICE DOWN.
REPLY 'RETRY' TO TRY READING LABEL AGAIN.
REPLY 'LABEL' TO LABEL DEVICE AND CONTINUE.
No valid device label was found on the device named. The operator should check to see that the device is online and ready. If there is no equipment fault and the label cannot be found, either the device must be DOWN or a label must be written. A reply other than GO, RETRY, or LABEL causes the message to be displayed again with the additional line
text INVALID. CORRECT AND RE-ENTER.
This message appears on the master operator console only. From Startup. Class, warning.

var UNABLE TO WRITE DEVICE LABEL ON *num* TRACKS.
I/O ERROR STATUS WAS RETURNED ON errcnt OF THE TRACKS.
REPLY 'GO' TO CONTINUE WITH DEVICE DOWN.
REPLY 'RETRY' TO TRY WRITING LABEL AGAIN
(MASTER DEVICE MAY NOT BE DOWNED)
The STARTUP task was unable to write a

device label on the named device. The operator should ensure that the device is online and ready. If the condition persists and there is no equipment fault, the device cannot be used, and the site engineers should be notified. A reply other than GO or RETRY causes the message to be displayed again with the additional line
text INVALID. CORRECT AND RE-ENTER.
This message appears on the master operator console only.

***** ALLOCATION CONFLICT ON PERMANENT DATASET *****
PDN = *pdn* ID = *id* OWN = *owner* ED = *ed*
DEVICE devname ALLOCATION UNIT alloc
ENTER 'GO' TO CONTINUE WITH THIS DATASET
ENTER 'SKIP' TO CONTINUE THIS DATASET WITHOUT FURTHER WARNINGS
ENTER 'DELETE' TO DELETE THIS DATASET
EITHER 'SKIP,ALL' OR 'DELETE,ALL' CAUSES FURTHER WARNINGS
TO BE BYPASSED FOR ALL SUBSEQUENT ERRORS OF THIS TYPE.
Startup detected that an allocation index, specified within the DAT for a permanent dataset, references disk space that is already allocated. This is normally the addition of a flaw for a device. Other causes could be software or hardware failure at some previous time. A reply of GO causes Startup to continue scanning the DAT for this device and reporting each occurrence of a conflict. A reply of SKIP causes Startup to stop reporting occurrences of conflicts for the current dataset, although they continue to be reported to the system log. A reply of DELETE causes Startup to delete the dataset. The ALL parameter may be included with either SKIP or DELETE, and causes Startup to assume the operator response to all subsequent messages (for this error condition) for all datasets is the same. Any other reply causes the message to be re-issued with the additional line
text INVALID. CORRECT AND RE-ENTER
This message appears on the master operator console only. From Startup. Class, warning.

APPARENT DEVICE LABEL FOUND. EXPECTED DEVICE NAME=*name*
ACTUAL DEVICE NAME READ FROM LABEL=*name*
REPLY 'USE' TO CHANGE CONFIGURED NAME TO NAME READ
REPLY 'LABEL' TO RE-WRITE DEVICE LABEL WITH CONFIGURED NAME
REPLY 'IGNORE' TO CONTINUE SEARCH FOR MATCHING LABEL
FLAWS FROM LABEL WILL BE USED FOR 'USE' OR 'LABEL' REPLY
Startup has found a valid device label but the name in the label does not match the name in the EQT. Operator options are:

USE - Startup changes the name in the EQT to the one from the label and accepts the label as valid. Flaws listed in the label will be assumed to be valid;
LABEL - Startup rewrites the device label with the name from the EQT. Flaws in the label are assumed to be valid;
IGNORE - Startup discards the label and continues searching for a label with a matching name.

Any other reply causes Startup to re-issue the message with the additional line
text INVALID. CORRECT AND RE-ENTER.
From Startup. Class, warning.

BAD ADDRESS
A parameter file command contains an out-of-range address, or a required address is missing. The *EBP command requires the second address if the first address is followed by a comma. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

BAD BREAKPOINT NUMBER
The breakpoint number on a *EBP command is missing or not in the range 0-7. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

BAD DATA SPECIFICATION
The value specified on an enter-memory command is invalid or missing. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

BAD END OF FILE
No *END command was found in the parameter file. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

BAD END OF LINE
An unexpected character was found on a parameter file entry when only a carriage return or a blank was legal. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

BAD FORMAT ON FLAW CARD
A card within a *FLAW/*ENDFLW sequence contains a non-octal digit or the C or T indicator was not present where expected. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

BAD MEMORY SIZE
A *MEMSIZ command has a missing or bad memory size value. *MEMSIZ cannot be used to increase the configured memory size. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

BAD PARAMETER VALUE
An invalid option was specified or a keyword parameter was followed by an out-of-range value. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

CANNOT READ SYSTEM DUMP AREA
ENTER GO TO TRY TO RE-ALLOCATE AREA
ENTER RETRY TO RE-READ
Startup received an error from DQM when attempting to read the reserved system dump area. Determine the cause of the error and correct the problem. If the error persists, the reserved area must be re-allocated. A reply of RETRY causes Startup to make another attempt to successfully read the dump area. A reply of GO causes Startup to attempt to allocate the system dump area to another portion of the master device. Sufficient space must exist on the master device for a second copy of the dump area, since the first is not deallocated. The master device label will be updated to indicate the new dump beginning allocation index. Any other reply causes the message to be re-issued. This message appears on the master operator console only. From Startup. Class, warning.

CANNOT RESTORE INFORMATION ON VOLATILE DEVICE *device* BECAUSE *errortext*.
TYPE - GO TO CONTINUE STARTUP WITHOUT RESTORING THIS DEVICE.
TYPE - DOWN TO CONTINUE STARTUP WITH THE DEVICE CONFIGURED DOWN.
Startup was unable to successfully restore data to the named volatile device. The reason for the failure is given by *error-text*, and may have the following values:
NO BACKUP PERMANENT DATASET EXISTS
NO DATA EXISTS ON DATASET
UNABLE TO READ BACKED UP DATASET
OF SIZE DIFFERENCE BETWEEN BACKUP DATASET AND DEVICE
This message appears on the master operator console only. From Startup. Class, warning.

CLASS STRUCTURE ROLL DATASET NOT FOUND - NO RECOVERY POSSIBLE
Permanent Dataset Manager did not find the class structure roll dataset in the DSC. Run a job to establish the desired structure. Default structure is in effect. From job class recovery in Startup, system log only. Class, caution

CREATING NEW EDITION OF THE SYSTEM DIRECTORY
Startup is creating a new edition of the system directory dataset. Occurs when Startup is unable to access an existing edition, or when the *SDR parameter file directive specifies that the directory is to be re-initialized or during an INSTALL. This message appears on the master operator console only. From Startup. Class, informative.

DATASET CATALOG ALLOCATES RESERVED TRACK *trnum* ON *ldv* (C*n* H*n* S*n*)
The DSC as defined in the master device label is attempting to reserve a track that is already reserved. A flaw might have been added since the previous install. The C, H, and S values indicate the disk address of the beginning sector of the overlap. All numbers are octal. From Startup. Class, informative.

DATASET CATALOG RECOVERY, PASS TWO
Startup is beginning a second pass over the Dataset Catalog during a restart or deadstart. The second pass is to complete error processing for conditions detected during the first pass such as AI conflicts or multi-type datasets with error conditions. Additional messages will have been sent to the operator identifying the errors which were detected. This message appears on the master operator console only. From Startup. Class, informative.

DATASET RECOVERED ON SCRATCH DEVICE *dname*
A *SCRATCH STARTUP directive was specified for a device that contains permanent datasets. The appropriate action depends on the site. From Startup, recovery of permanent datasets. Class, informative.

DEVICE *device* - APPARENT DEVICE LABEL CONTAINS INVALID FLAW
BAD VALUE IS *n* DEVICE IS CONFIGURED AS *devtype*
REPLY 'GO' TO IGNORE ERROR BUT CONTINUE REPORTING
REPLY 'SKIP' TO IGNORE THIS AND FURTHER ERRORS ON DEVICE
REPLY 'QUIT' TO STOP PROCESSING THIS LABEL
While searching for the device label on the named device, a track was found which appeared to contain a valid label, except that one or more entries in the flaw map were out-of-range for the device type indicated. The message gives the device name, the configured device type, and the value of the erroneous flaw entry. This may be caused by using a previously labeled DD29 in place of a DD19.
Operator options are:
GO - Ignore the error reported and continue reporting additional errors;
SKIP - Ignore the error and stop reporting errors found in the label for this device. Additional valid flaws will be processed;
QUIT - Stop processing the flaw list. The label is considered to be invalid, and Startup will ask if a label is to be written or the device downed. If a label is to be written, flaws already set from this label will be included in the new label.
If the apparent label is not to be used as the valid label, it must be overwritten by site engineers and another Startup must be performed. This message

appears on the master operator console only. From Startup. Class, warning.

DEVICE *device* - CHECKSUM ERROR IN DEVICE LABEL
REPLY 'LABEL' TO USE LABEL FOUND AND CORRECT CHECKSUM
FLAWS FROM LABEL WILL BE USED IF REPLY IS LABEL
REPLY 'IGNORE' TO CONTINUE SEARCH FOR LABEL
Startup located an apparent device label on the named device, but the checksum read does not match the calculated checksum. Operator options are:

LABEL - The label is used, and Startup rewrites the label with a corrected checksum;
IGNORE - Startup continues searching for a valid label.

A checksum error means that the data on disk is probably not completely valid, and the flaw list might contain invalid information. Some datasets could be falsely determined to have allocation conflicts if the flaw list is bad. The device should be dumped, reformatted, and reloaded as soon as possible. Any other reply causes the message to be reissued with the additional line *text* INVALID. CORRECT AND RE-ENTER.
This message appears on the master operator console only. From Startup. Class, warning.

***** CATASTROPHIC ERROR ON PERMANENT DATASET *****
PDN = *pdn* ID = *id* OWN = *owner* ED = *ed*
ENTER 'GO' TO FLAG DSC ENTRY AND RETAIN DATASET
ENTER 'DELETE' TO DELETE THIS DATASET
EITHER 'GO,ALL' OR 'DELETE,ALL' CAUSES FURTHER WARNINGS
TO BE BYPASSED FOR SUBSEQUENT ERRORS OF THIS TYPE
Startup detected that the DSC entry for the dataset contains an error, and the dataset cannot be successfully recovered. Each occurrence of the condition will be reported unless the ALL parameter is included in the reply. To retain the dataset in the DSC, reply GO. To delete the dataset, reply DELETE. In any case, all errors will be reported to the system log. Any other reply causes the message to be re-issued with the additional line *text* INVALID. CORRECT AND RE-ENTER
This message appears on the master operator console only. From Startup.

CLASS STRUCTURE LOADED FROM PDN=*pdn*, ID=*id*, ED=*ed*
The job class structure was successfully loaded from the specified permanent dataset. System log only. From job class recovery in Startup. Class, informative.

CNT ENTRY INVALID - DEVICE = *devname*
ENTER ANY REPLY TO CONTINUE
Inconsistent information in a Configuration Table entry. Issue any reply to cause Startup to request the configuration change needed to correct the entry. This message appears on the master operator console only. From Startup. Class, caution.

DEVICE *device* CONTAINS APPARENT EFT WITH INVALID AI. BAD AI NUMBER IS *n*.
DEVICE IS CONFIGURED AS *devtype*
REPLY 'USE' TO ACCEPT EFT, NO FURTHER WARNINGS THIS DEVICE
REPLY 'GO' TO IGNORE THIS ERROR - WARN IF MORE
A sector was found that appears to be a valid Engineering Flaw Table (EFT), except that one or more of the flaws indicated in the table is out-of-range for the device as it is configured. The message indicates the device name, the configured device type, and the value of the erroneous flaw entry. The most likely cause of this condition is the use of a previously labeled DD29 in place of a DD19. Operator options are:

GO - Accept the EFT as is, and continue to report errors. The bad flaw information is ignored;
USE - Accept the EFT as is. The bad flaw information is ignored, and no more warnings will be issued for the same device.

If the apparent EFT is not to be treated as a valid EFT, it must be overwritten by site engineers and another Startup must be performed. This message appears on the master operator console only. From Startup.

DEVICE *device* IS BEING RESTORED
The contents of the named volatile device are being restored from the image written to non-volatile storage via the FLUSH command. Occurs when a *RESTORE parameter file directive is present. This message appears on the master operator console only. From Startup. Class, informative.

DEVICE *device* - NO ENGINEERING FLAW TABLE FOUND
ENTER GO TO CONTINUE STARTUP
SKIP TO CONTINUE W/O FURTHER WARNINGS
No Engineering Flaw Table (EFT) was found on the named device. If an EFT needs to be on the device, the site engineers must create it. A reply of GO causes Startup to report each device on which no EFT is found. A reply of SKIP causes Startup to stop reporting devices which do not contain an EFT. This message appears on the master operator console only. From Startup. Class, caution.

DISK SPACE NEGATIVE WHILE RECOVERING ROLLED JOBS
The amount of available space on a disk has gone negative while space was being allocated for rolled jobs. The problem might be circumvented by not recovering rolled jobs, but a Cray Research analyst should investigate. From Startup, system log only. Class, fatal.

DQM ERROR ON REWRITE OF JTA DURING RRJ
DQM returned an error status after unsuccessfully attempting to rewrite the JTA of a recovered rolled job. Check the DQM error status; it might be possible to circumvent the problem by not recovering rolled jobs. From Startup. Class, fatal.

DQM ERROR WHILE ALLOCATING ROLL INDEX
While attempting to write the index buffer to $ROLL, DQM returned an error after being unable to allocate space on the new $ROLL file. Check DQM error status and ensure that adequate mass storage space exists on the system. From Startup. Class, fatal.

DSC DAT BAD OR FLAW EXISTS IN DSC
While space was being reserved for the DSC, either cross allocation occurred or a bad DAT was encountered. Check whether there are flaws in the DSC area or the DSC DAT to ensure that space is valid. From Startup. Class, fatal.

DSC EXTENSION ALLOCATES RESERVED TRACK
trnum ON *ldv* (C*n* H*n* S*n*)
The Dataset Catalog Extension dataset is reserving a track that is already reserved. A flaw might have been added since the dataset was created or extended. The dataset cannot be accessed. The C, H, and S values indicate the disk address of the first sector of the overlap. All numbers are octal. From Startup. Class, informative.

ENTER COMMENT DESCRIBING REASON FOR TAKING SYSTEM DUMP
Startup is processing a system dump taken prior to the current Startup. Any reply up to 80 ASCII characters is accepted, and is stored in the header of the dump when it is saved, as well as being sent to the system log. The reply must be at least one character in length. This message appears on the master operator console only. From Startup. Class, informative.

ENTER CONFIGURATION CHANGES OR CONTINUE
Startup is beginning and is requesting the operator to make any necessary changes in disk or tape device statuses and definitions. The operator should reply with the CONFIG reply or with either CONTINUE or GO to signal that no more changes are desired. This message appears on the master operator console only. From Startup.

ERROR WHILE RE-WRITING DUMP HEADER
ENTER GO TO SKIP WRITE
ENTER RETRY TO TRY AGAIN
Startup received an error from DQM while attempting to rewrite the header of the reserved dump area. If the drive has faulted, correct the condition and reply RETRY. Otherwise, reply GO to continue Startup without rewriting the header. After replying GO, the next Startup may try to process the dump again, depending on the extent to which the header rewrite was processed by the device. It may be advisable to flaw out the existing dump area and reallocate it on a subsequent Startup. Any reply other than GO or RETRY causes the message to be re-issued. This message appears on the master operator console only. From Startup. Class, warning.

ERROR OCCURRED DURING PARAMETER FILE PROCESSING ON THIS PARAMETER FILE COMMAND
parmtext
TYPE - GO TO CONTINUE STARTUP
Startup detected an error during parameter file processing. The bad parameter file directive is displayed. A reply of GO causes Startup to continue. The erroneous directive was not processed, although it may have been partially processed if the error was detected in a second or subsequent subparameter. The directive should be corrected, and another Startup performed. This message appears on the master operator console only. From Startup. Class, caution.

ERROR WRITING SYSTEM DUMP COPY
ENTER GO TO RE-ALLOCATE AND RE-COPY
ENTER RETRY TO TRY WRITE AGAIN
ENTER QUIT TO ABANDON COPY OF DUMP
Startup received an error from DQM while writing to the copy of the system dump which would normally be saved as a permanent dataset. The most likely cause is insufficient disk space. Determine the cause of the problem and correct it if possible. A reply of GO will attempt to write the dump again. A reply of RETRY will cause Startup to re-issue the request which failed. A reply of QUIT will cause Startup to stop attempting to copy the dump and clear the flag in the reserved area which indicates the presence of an uncopied dump. Any other reply causes the message to be re-issued. This message appears on the master operator console only. From Startup. From Startup. Class, warning.

ILLEGAL SINCE MASTER DEVICE
A parameter attempted to make the system master device unavailable. Correct the parameter file. From Startup, ZLOG buffer only. Class, caution.

***** INCONSISTENT ALLOCATION ON MULTI-TYPE DATASET *****
PDN = *pdn* ID = *id* OWN = *owner* ED = *ed*
ENTER 'GO' TO FLAG DSC ENTRY AND RETAIN DATASET
ENTER 'DELETE' TO DELETE THIS DATASET
EITHER 'GO,ALL' OR 'DELETE,ALL' CAUSES FURTHER WARNINGS
TO BE BYPASSED FOR SUBSEQUENT ERRORS OF THIS TYPE
Startup detected that two or more datasets having the same QDT ordinal have non-identical DATs. This message occurs once for each subsequent DSC entry having the same QDT ordinal. In pass two over the DSC, it occurs for each DSC entry with the QDT ordinal that has not already been flagged. To prevent the operator from having to reply to each message, the ALL parameter may be used. In this case, the operator is not warned of any subsequent inconsistent allocations, even if they belong to a different QDT ordinal set. The errors will continue to be reported to the system log. Any reply other than GO or DELETE will cause the message to be re-issued with the additional line
text INVALID. CORRECT AND RE-ENTER.
This message appears on the master operator console only. From Startup. From Startup. Class, warning.

INITIATING THE $DSC-EXTENSION RECOVERY AND VALIDATION.
From Startup, system log only. Class, informative.

INSUFFICIENT DISK SPACE FOR THE $DSC-EXTENSION.
REPLY WITH THE (DXT,DV=,SZ=,OVF=,CAI=) COMMAND TO CONTINUE.
Insufficient disk space exists in the system to allocate the $DSC-EXTENSION dataset using the current options. specify new options via the DXT command, described in the Operational Procedures Reference Manual, CRI publication number SM-0043. From Startup, system log only. Class, warning.

INSUFFICIENT MEMORY FOR STARTUP TO EXECUTE
Startup and its associated tables have exceeded the field length available to system tasks. Determine the reason for the change in the size of STARTUP or system tables and buffers. From Startup, system log only. Class, fatal.

INTERNAL ERROR - BAD DUMP DAT
While moving the system dump DAT to the first sector of the dataset, errors were detected in the DAT. Check the system dump DAT for errors. From Startup. Class, fatal.

INTERNAL STARTUP/CONFIGURATION ERROR
STARTUP encountered an error condition that prevents continuation. The error could be caused by errors in the configuration or by serious software/hardware problems. See a Cray Research analyst. This message appears on the master operator console only. From Startup. Class, fatal.

INVALID ENTRY. CORRECT AND RE-ENTER
Invalid reply to a configuration request message. Re-enter the correct reply. From Startup. Class, caution.

I/O ERROR RETURN FROM DQM WHILE EXTENDING THE DXT.
Startup received an error from DQM while extending the DXT dataset. Determine the cause of the error and correct if possible, then perform another Startup. This message appears on the master operator console only. From Startup. Class, fatal.

I/O ERROR WHILE READING THE SYSTEM DIRECTORY
DQM returned an error status after unsuccessfully attempting to read $SDR. Check the DQM error status. *SDR in the parameter file might circumvent the problem. This message appears on the master operator console only. From Startup. Class, fatal.

I/O ERROR WHILE WRITING ROLL INDEX DATASET
DQM returned an error status after unsuccessfully attempting to write the index buffer to the $ROLL dataset. Check the DQM error status. This message appears on the master operator console only. From Startup. Class, fatal.

MEMORY POOL SPACE INSUFFICIENT FOR TEXT
While I/O datasets were being recovered, the space in the text memory pool could not contain the whole text. Check whether the memory pool, I@MPSZ, or MPSZ has changed since the system was last running. A deadstart might be required. This message appears on the master operator console only. From Startup. Class, fatal.

MISSING COMMA
An unexpected character was found when the syntax of a parameter file entry required a comma. Correct the parameter file. ZLOG buffer only. From Startup. Class, caution.

MISSING STARTUP OPTION
If a parameter file exists, it must contain at least an *INSTALL, *DEADSTART, or *RESTART directive. None were found. Correct the parameter file. ZLOG buffer only. From Startup. Class, caution.

***** MULTI-TYPE DATASET QDT ORDINAL INVALID *****
ORDINAL IN DSC ENTRY = *dsc-ord* MAXIMUM VALID ORDINAL = *max-ord*
PDN = *pdn* ID = *id* OWN = *owner* ED = *ed*

ENTER 'GO' TO FLAG DSC ENTRY AND RETAIN DATASET
ENTER 'DELETE' TO DELETE THIS DATASET
EITHER 'GO,ALL' OR 'DELETE,ALL' CAUSES FURTHER WARNINGS
TO BE BYPASSED FOR ALL SUBSEQUENT ERRORS OF THIS TYPE
Startup has detected that a multi-type dataset has a QDT ordinal greater than the maximum allowed in the version of COS being started. This may occur if the installation parameter NE@SDT is changed. To retain the dataset in the DSC and allocate its disk space, reply GO. To delete the dataset, reply DELETE. Each DSC entry containing this error condition will cause an error message to be issued unless the ALL parameter is included in the reply, in which case Startup will act as if the operator has replied to each subsequent message with the same reply (for the same error condition). Any other reply causes the message to be re-issued with the additional line *text* INVALID. CORRECT AND RE-ENTER. This message appears on the master operator console only.

MULTIPLE MASTER DEVICES
More than one device was defined as the master device. This message occurs during an INSTALL if and only if an existing label was found specifying a device to be a 'master' device, and the master device specified in the EQT is not the same device. Site engineers must over-write the label on the device. From Startup. Class, fatal.

MULTIPLE MASTER DEVICES DEFINED. ENTER DEVICE NAME TO BE USED AS A MASTER (IF THIS IS DEADSTART OR RESTART THE SPECIFIED DEVICE MUST HAVE A MASTER LABEL).
During an INSTALL more than one EQT entry specified a master device, or during a RESTART or DEADSTART, more than one device label was found containing a master device label. The operator must specify which device Startup should use as the master device. This message appears on the master operator console only. From Startup. Class, warning.

NO LABEL WAS FOUND ON DEVICE *device*.
INFORMATION CONTAINED ON THE DEVICE MAY BE INVALID. TO PROCEED WITH STARTUP AND REWRITE LABEL TYPE - CONTINUE
TO PROCEED WITH STARTUP WITH THIS DEVICE DOWN TYPE - DOWN
TO PROCEED WITH STARTUP AND REWRITE THE DEVICE LABEL AND RESTORE THE INFORMATION TO THE DEVICE TYPE - RESTORE
Startup could not find a valid device label for a volatile device. The device may have been powered down since the last use. To rewrite the device label and retain any other data on the device, the operator should reply CONTINUE. If the label is to be written and the data restored from the latest backup dataset, the operator should reply RESTORE. If the device is to be made unavailable, the operator should reply DOWN. This message appears on the master operator console only. From Startup. Class, warning.

NO MASTER DEVICE FOUND DURING DEADSTART/RESTART
No device had the master device flag set in the label. A master device has to exist because a DSC has to be predefined. This message appears on the master operator console only. From Startup. Class, fatal.

NO ROOM IN STP DAT FOR DSC DATASET
The STP table's DAT area is too small to contain the DAT for the DSC. Check SZ@DAT and the size of the DSC DAT to ensure that there is sufficient space. This message appears on the master operator console only. From Startup. Class, fatal.

NO ROOM IN STP DAT TO RECOVER PERMANENT DATASET
During permanent dataset recovery, there was insufficient space in the STP DAT area to move the DAT for a dataset to memory from the DSC. Determine whether the size of the STP DAT has decreased since the system was previously running. This message appears on the master operator console only. From Startup. Class, fatal.

ONLY FLAW CARD ALLOWED
A non-flaw card was found within a *FLAW/*ENDFLW sequence. Correct the parameter file. ZLOG buffer only. From Startup. Class, caution.

PARAMETER ERROR COUNT *n*
Parameter checking is complete and ZY gives the total number of errors encountered. ZLOG buffer only. From Startup. Class, informative.

PARAMETER FILE CONTAINS AN INVALID DXT COMMAND.
TO CORRECT THE COMMAND RE-ENTER THE DXT COMMAND - DXT,SZ= ,DV= ,OVF= ,CAI=
Startup detected an invalid *DXT parameter file directive. Determine the cause of the error and correct the parameter file. Enter the corrected DXT directive via the DXT command, Operational Procedures Reference Manual, CRI publication number SM-0043. ZLOG buffer only. From Startup. Class, warning.

PDN = *pdn* ID = *id* ED = *ed*
OWN = *owner*
***** DATASET HAS CATASTROPHIC DXT ERROR *****
REPLY 'GO' TO CLEAR DXT FROM DSC ENTRY - (ALSO SET PAM TO NONE)
REPLY 'SKIP' TO MARK DSC ENTRY IN ERROR
REPLY 'DELETE' TO DELETE DATASET
ANY REPLY FOLLOWED BY 'ALL' WILL CAUSE

ALL FURTHER ERRORS TO BE PROCESSED IDENTICALLY WITHOUT FURTHER WARNINGS
Startup detected an invalid DXT entry associated with the named dataset. A reply of GO clears all DXT information associated with the dataset and sets the Public Access Mode (PAM) to NONE for the dataset. A reply of SKIP causes the DSC entry to be marked as invalid but leaves the dataset in the catalog. Any reply followed by ALL causes that response to be assumed for all subsequent DXT error conditions. This message appears on the master operator console only. From Startup. Class, warning.

PDS IS FULL
The PDS has insufficient space to build an entry for an accessed permanent dataset. This dataset could be a system dataset, a dataset in the SDR, or a permanent dataset accessed by a recovered rolled job. Check the size of the PDS and determine whether it has changed since the previous system. From Startup. Class, fatal.

***** PERMANENT DATASET RELEASE REQUESTED ON DEVICE *****
DEVICE = *device*
PDN = *pdn* ID = *id* OWN = *owner*
ED = *ed*
ENTER 'DELETE' TO DELETE THIS DATASET
ENTER 'DELETE,ALL' CAUSES FURTHER WARNINGS TO BE BYPASSED FOR ALL SUBSEQUENT DELETIONS OF DATASETS ON DEVICES BEING RELEASED
When a device is flagged to be released, either by the RLS parameter in the EQT or parameter file, or by some reconfiguration constraint (such as deleting a striped device), the above message will be posted when a dataset has been found on the RELEASED device and is to be deleted. If the response to the above is neither DELETE or DELETE,ALL the following is appended to the above text:
text INVALID. CORRECT REPLY AND RE-ENTER
This message appears on the master operator console only. From Startup. Class, warning.

***** PERMANENT DATASET RESIDES ON DOWN DEVICE *****
DEVICE = *device*
PDN = *pdn* ID = *id* OWN = *owner* ED = *ed*
ENTER 'GO' TO FLAG DSC ENTRY AND RETAIN DATASET
ENTER 'DELETE' TO DELETE THIS DATASET
EITHER 'GO,ALL' OR 'DELETE,ALL' CAUSES FURTHER WARNINGS
TO BE BYPASSED FOR SUBSEQUENT ERRORS OF THIS TYPE
Startup detected that the DAT for the named permanent dataset references one or more devices whose EQT entry is either not present or marked unavailable. Each occurrence of the condition will be reported unless the ALL parameter is included in the reply. To retain the dataset in the DSC and to allocate any space which may be on available devices, reply GO. To delete the dataset, reply DELETE. Any other reply causes the message to be re-issued, with the additional line *text* INVALID. CORRECT AND RE-ENTER. From Startup. Class, warning.

RE-ALLOCATION FAILED. TYPE GO TO CONTINUE WITHOUT DUMP.
Startup was unable to re-allocate the system dump area. The most likely cause is that the master device is too full to allocate sufficient space for the dump. The current dump is lost, and no future dumps should be taken until the system dump area can be successfully allocated. This may require dumping permanent datasets from the device. Any response other than GO causes the message to be re-issued. This message appears on the master operator console only. From Startup. Class, warning.

ROLLED JOBS ARE BEING RECOVERED
Startup is beginning the recovery of rolled jobs during a restart. This message appears on the master operator console only. From Startup. Class, informative.

SPECIFIED DEVICE IS NOT A MASTER DEVICE. ENTER A CORRECT DEVICE NAME.
Operator has replied to a request to identify the correct one of multiple master devices during a RESTART/DEADSTART, and the named device did not contain a master device label. The operator should respond with the name of the correct master device. This message appears on the master operator console only. From Startup. Class, warning.

SPECIFIED DEVICE IS NOT IN THE EQT. ENTER A CORRECT DEVICE NAME.
Operator has supplied the name of a non-existent device in response to a request to identify the correct master device. The operator should respond with the name of the correct master device. This message appears on the master operator console only. From Startup. Class, warning.

STARTUP CANNOT CONTINUE.
THE $DSC=EXTENSION CAN'T BE ACCESSED.
PDM ERROR = *errstat*
Startup received an unexpected error from PDM while attempting to ACCESS the DXT. Determine the cause of the error and correct the problem. Perform another Startup. This message appears on the master operator console only. From Startup. Class, fatal.

STARTUP CANNOT CONTINUE.
THE $DSC-EXTENSION CAN'T BE ADJUSTED.
PDM ERROR = errstat
Startup received an error from PDM while attempting to ADJUST the DXT dataset. Determine the cause of the error and correct the problem. Perform another Startup. This message appears on the master operator console only. From Startup. Class, fatal.

STARTUP CANNOT CONTINUE.
THE DXT COULD NOT BE SAVED. PDM ERROR = *errstat*
Startup received the indicated error status from PDM when attempting to SAVE the $DSC-EXTENSION dataset. Correct the problem and perform another Startup. This message appears on the master operator console only. From Startup. Class, fatal.

STARTUP CANNOT CONTINUE
CANNOT SETUP UNIQUE ACCESS FOR THE DXT.
PDS IS FULL.
Startup received an error from PDM when attempting to obtain a PDS entry for the $DSC-EXTENSION dataset. A DEADSTART may succeed. This message appears on the master operator console only. From Startup. Class, fatal.

STARTUP CANNOT CONTINUE.
FATAL I/O ERROR ENCOUNTERED READING THE $DSC-EXTENSION.
-or-
FATAL I/O ERROR ENCOUNTERED WRITING THE $DSC-EXTENSION.
Startup received an error from DQM while attempting to read or write the DXT. Determine the cause of the problem and correct it if possible. Perform another Startup. This message appears on the master operator console only. From Startup. Class, fatal.

STARTUP CANNOT CONTINUE.
THE $DSC-EXTENSION IS FULL. ANOTHER STARTUP MUST BE PERFORMED WITH A REQUEST TO INCREASE THE SIZE OF THE DXT.
The DXT is full. Increase the size of the DXT via the *DXT parameter file directive and perform another Startup. This message appears on the master operator console only. From Startup. Class, fatal.

STARTUP SELECTED CREATION OF THE $DSC-EXTENSION.
Startup was unable to locate the $DSC-EXTENSION dataset in the DSC and is creating it. This message appears on the master operator console only. From Startup. Class, informative.

STARTUP IS PERFORMING *type*
Startup is beginning label processing. *Type* will be one of the values 'AN INSTALL', 'A DEADSTART', or 'A RESTART'. This message appears on the master operator console only. From Startup. Class, informative.

STRIPE DEVICE *sdvn* IS TYPE *stype*
DEVICE *ldvn* IS TYPE *ltype*
INCOMPATIBLE DEVICE TYPES. ENTER 'GO' TO CONTINUE WITH DEVICE DELETED FROM GROUP
A disk stripe has been defined but the named device (*ldvn*) had different hardware characteristics than other devices previously encountered which are defined as being in the same stripe group (*sdvn*). Reply 'GO' to continue startup with the named device not included in the group. This message appears on the master operator console only. From Startup. Class, caution.

SYSTEM DIRECTORY DATASET IS BAD
STARTUP determined that the $SDR dataset was invalid. Check for errors; *SDR might be required in the parameter file, followed by execution of the JSVDIR job. From Startup. Class, fatal.

SYSTEM DUMP AREA ALLOCATES RESERVED TRACK *trnum* ON *ldv* (C*n* H*n* S*n*)
On the master device, the area reserved for the system dump contains a track that is already reserved. This is usually caused by the addition of a flaw. The C, H, and S values indicate the disk address of the first sector of the overlap. All numbers are in octal. System log only. From Startup. Class, informative.

SYSTEM DUMP AREA INVALID OR CONTAINS AI CONFLICTS.
ENTER GO TO RE-ALLOCATE
ENTER RETRY TO RE-READ EXISTING HEADER
Startup was unable to reserve the system dump using the information in the master device label and the first sector of the allocation unit indicated by the label. Either the dump area has been destroyed, or a flaw has been added to the master device such that an allocation conflict occurs. If the master device is the correct device and the flaws are correct, the dump area must be re-allocated. Any reply other than GO or RETRY causes the message to be re-issued. This message appears on the master operator console only. From Startup. Class, warning.

SYSTEM DUMP SAVED - STATUS = *status*
PDN = *pdn* UID = *id* ED = *ed*
The system dump was successfully saved with the permanent dataset characteristics given. From Startup. Class, informative.

TABLE SIZE CHANGED - CAN'T RECOVER ROLLED JOBS
Since the last running system, there has been a system change, including a change in the length of JXT, DNT, or the rolled job index. This message appears on the master operator console only. From Startup. Class, fatal.

THE BACKED UP INFORMATION FOR DEVICE *device* MAY NOT BE VALID.
IF THE RESTORE PROCESS IS TO CONTINUE WITH THIS INFORMATION TYPE - GO.
IF THE DEVICE IS TO BE RECOVERED WITHOUT ANY RESTORE PROCESSING TYPE - SKIP.
The header in the backup dataset for a volatile device indicates that the data has been invalidated (previously restored). The data may not be current. A reply of GO causes Startup to restore the device from the dataset. A reply of SKIP causes Startup to stop restoring this device. This message appears on the master operator console only. From Startup. Class, warning.

THE BACKUP DATASET DN = *name errortext* TYPE - GO TO CONTINUE STARTUP WITH THIS DEVICE NOT MARKED AS VOLATILE. TYPE - NEW TO CREATE A NEW EDITION OF THIS DATASET.
The backup dataset for the named device cannot be used for the reason given by error-text. A subsequent operator FLUSH command will not be successful. To clear the volatile designation and disallow the FLUSH command for the device, reply GO. To allocate and initialize a new backup dataset, reply NEW. This message appears on the master operator console only. From Startup. Class, warning.

THE DATASET CATALOG IS BEING RECOVERED
Startup has completed label processing and is beginning the recovery of the dataset catalog. This message appears on the master operator console only. From Startup. Class, informative.

THE $DSC-EXTENSION DATASET CAN ONLY BE CREATED DURING AN INSTALL OR A DEADSTART. RESTART ABANDONED.
A RESTART is being performed, and there is no DXT. The DXT must exist for a RESTART to be performed. Perform either an INSTALL or a DEADSTART to create the DXT. This message appears on the master operator console only. From Startup. Class, fatal.

THE $DSC-EXTENSION IS nn% FULL.
This message appears on the master operator console only. From Startup. Class, informative.

THE $DSC-EXTENSION WAS CREATED AND SAVED SUCCESSFULLY.
Startup successfully created the DXT. This message appears on the master operator console only. From Startup. Class, informative.

THE $DSC-EXTENSION WAS RECOVERED AND VALIDATED SUCCESSFULLY.
This message appears on the master operator console only. From Startup. Class, informative.

THE DXT SIZE CAN'T BE ZERO.
REPLY WITH THE (DXT, SZ=,DV=,OVF=,CAI=) COMMAND TO CONTINUE.
The size of the DXT is specified as zero. Specify new options via the DXT command, Operational Procedures Reference Manual, CRI publication SM-0043. This message appears on the master operator console only. From Startup. Class, warning.

THE SYSTEM DUMP IS BEING SAVED
Startup is saving the system dump as a permanent dataset. Occurs only when a SYSDUMP command was executed at the MCU console prior to the startup. This message appears on the master operator console only. From Startup. Class, informative.

THE SYSTEM DIRECTORY IS BEING RECOVERED
Startup is beginning the recovery of the system directory ($SDR). This message appears on the master operator console only. From Startup. Class, informative.

TOO MANY BAD TRACKS WERE ENCOUNTERED WHILE TRYING TO INITIALIZE THE DXT ON DISK. THE BAD TRACKS HAVE BEEN FLAWED OUT.
ENTER (GO) TO RETRY.
Startup was unable to successfully write the $DSC-EXTENSION dataset to disk due to errors. Determine the reason for the errors, and if needed, add *FLAW directives to the parameter file. Perform another Startup. This message appears on the master operator console only. From Startup. Class, warning.

TOO MANY DUMP ITEMS - SOME LOST
The dump contained in the reserved dump area specifies more dump items than the dump header can contain. The items which could not be contained in the header are not copied. Any reply causes Startup to continue and SAVE the part of the dump which was copied. From Startup. Class, warning.

UNABLE TO ALLOCATE THE SYSTEM DUMP AREA
Space was unavailable for the preallocated area for the system dump. Either no space remained or allocated space could not be read. Space must be allocated on the master device. Check that there is sufficient space; determine if there was an I/O error. This message appears on the master operator console only. From Startup. Class, fatal.

UNABLE TO COMMUNICATE WITH I/O SUBSYSTEM
ENTER GO TO SKIP CONFIGURATION CHANGES
ANY OTHER REPLY CAUSES RETRY.
Startup was unable to successfully initialize its link with the I/O Subsystem station. Configuration changes cannot be made unless the station is logged on. Occurs only on systems with an I/O Subsystem. This message appears on the master operator console only.

UNABLE TO CREATE DSC DATASET
During the creation of the DSC as a blocked dataset, STARTUP was unable to write one of the pages. Check the DQM error status. This message appears on the master operator console only. From Startup. Class, fatal.

UNABLE TO RESTORE INFORMATION ON DEVICE *device*
ERROR OCCURRED DURING WRITE ON CYLINDER *cyl* TRACK *trk*
TO CONTINUE TYPE - GO.
Startup received an error from DQM while attempting to restore data to a volatile device. This message appears on the master operator console only. From Startup. Class, warning.

UNABLE TO SAVE COPY OF DUMP. TYPE GO TO IGNORE.
Startup received an error from PDM when issuing the SAVE request after copying the dump, and either the error was unexpected, or Startup was unable to generate a permanent dataset name which did not receive the error. Normally, Startup will attempt to recover from the DSC FULL error status by altering the name of the dump in the PDD and re-issuing the request. The name under which the dump is successfully saved is sent to the system log. Startup will also attempt to recover from the TOO MANY EDITIONS and the MAINTENANCE PERMISSION NOT GRANTED by changing the permanent dataset name in the PDD. Any other error is unexpected. A reply of GO causes Startup to continue without saving the dump. This message appears on the master operator console only. From Startup. Class, warning.

UNABLE TO WRITE CSP TO DISK
DQM returned an error status on all attempts to write CSP to disk. For all I@NCSP copies of CSP, DQM returned errors. Check the DQM error status. This message appears on the master operator console only. From Startup. Class, fatal.

UNABLE TO WRITE DAT TO SYSTEM DUMP
The DAT for the system dump area could not be written to the first sector of this area. Check the DAT for an error; determine whether an I/O error occurred. This message appears on the master operator console only. From Startup. Class, fatal.

UNEXPECTED PDM ERROR ON RELEASE OF ROLL INDEX DATASET
An I/O error occurred on the read of $ROLL; STARTUP then attempted to release the bad dataset and PDM returned an error status. Check PDM and DQM error status. This message appears on the master operator console only. From Startup. Class, fatal.

UNEXPECTED PDM ERROR ON SAVE OF ROLL INDEX DATASET
PDM issued an unexpected response to the request sent by STARTUP to save the roll index. Check the PDM error status to determine the error type. This message appears on the master operator console only. From Startup. Class, fatal.

UNEXPECTED STATUS *status* ON ACCESS OF CLASS STRUCTURE ROLL DATASET
Permanent Dataset Manager returned the specified error status when STARTUP attempted to access the class structure roll dataset. Run a job to re-establish the desired structure. Default structure is in effect. From job class recovery in Startup. System log only. Class, caution

UNEXPECTED STATUS ON ACCESS OF $ROLL WAS *status* - NEW EDITION CREATED - RRJ NOT POSSIBLE
PDM returned an unexpected status when RRJ attempted to access the roll index dataset. Recovery of rolled jobs does not occur. From recovery of rolled jobs in Startup. System log only. Class, informative

UNKNOWN LOGICAL DEVICE *ldv*
A parameter file entry specified a device name that has no matching EQT entry. Correct the parameter file. ZLOG buffer only. From Startup. Class, caution.

UNKNOWN PARAMETER TYPE
ZY was unable to recognize a parameter file entry. Correct the parameter file. ZLOG buffer only. From Startup. Class, caution.

*** W A R N I N G ***
THE $DSC-EXTENSION IS nnn% FULL. THE SIZE OF THIS DATASET SHOULD BE INCREASED IN THE NEAR FUTURE VIA THE (DXT,SZ=+XXX) COMMAND.
ENTER (GO) TO CONTINUE STARTUP.
Startup has determined that the DXT is nearly full. The size should be increased in a subsequent Startup via the *DXT parameter file directive. This message appears on the master operator console only. From Startup. Class, caution.

Z HALTED AT BREAKPOINT DUE TO *DEBUG CARD
A *DEBUG card was encountered during parameter file processing, specifying that STARTUP was to wait for operator action before continuing. Enter "RUN TASK 0" at operator station console to cause STARTUP to begin processing a restart, deadstart, or install. This message appears on the master operator console when the Task exchange package is displayed (the default for the station "Y" display. From Startup. Class, informative.

Z HALTED AT BREAKPOINT DUE TO ERRORS IN ZY
ZY detected at least one error in the parameter file. Examine ZLOG from the operator station console. If the errors flagged are acceptable, enter "RUN TASK 0" to cause STARTUP to continue. If the errors are not acceptable, correct the parameter file and begin the startup again. This message appears on the master operator console when the Task exchange package is displayed (the default for the station "Y" display. From Startup. Class, caution.

2.8.2 STARTUP MESSAGES GOING TO SYSTEM LOG ONLY

var DATASET DAT CONTAINS AI CONFLICT AT TRACK *trnum* ON *ldv* (C*n* H*n* S*n*)
The specified dataset is attempting to reserve a previously reserved track on the specified device. A new flaw has been added or a system error has occurred. The dataset cannot be accessed. The C, H, and S values (as cylinder, head group and sector) indicate the disk address of the first sector of the overlap. All numbers are in octal. From permanent dataset recovery in Startup. System log only. Class, informative.

var DATASET RESIDES ON DOWN DEVICE *ldv*
The specified dataset resides wholly or partially on a device whose EQT entry shows the device is not available and for which the release flag is not set, or the DAT contains a reference to a device for which no matching EQT entry could be found. Determine why the device is unavailable, correct the EQT, and start COS again. From permanent dataset recovery in Startup. System log only. Class, informative.

var DATASET RESIDES ON MISSING/RELEASED DEVICE *ldv*
The specified dataset resides wholly or partially on a device whose EQT entry specifies that the device is to be released or on a device for which no matching EQT entry was found. The dataset has been deleted from the DSC. From permanent dataset recovery in Startup. System log only. Class, informative.

var DSC ENTRY CONTAINS CATASTROPHIC ERROR - ENTRY *status*
The Dataset Catalog entry for the specified device contains at least one error condition from which STARTUP is unable to recover. *status* is DELETED if the DSC entry was cleared; *status* is RETAINED if the DSC entry was flagged as an error but left intact. If the status is RETAINED, the dataset cannot be accessed. The following error conditions are considered catastrophic:

- DAT contains an AI that is out of range for the device.
- A continuation DSC entry does not exist when one is required (current AI count not exhausted, first DAT body is not on first DSC page, etc.).
- The DSC entry is a second or subsequent entry for a multitype dataset, and a previously encountered entry for that dataset was flagged as having a catastrophic error.
- The DSC entry contains no DAT bodies at all.
- The continuation entry pointed to by the DSC entry is not a continuation entry, or points back to a different DSC entry.
- The DSC entry contains a larger text block than can be handled by the system being started.
- DAT contains an invalid allocation style. Action: If *status* is DELETED, none possible. If *status* is RETAINED, examine the DSC entry to determine the reason for the error.

From permanent dataset recovery in Startup. System log only. Class, informative.

var MULTI-TYPE DATASET HAS INCONSISTENT ALLOCATION - QDT INDEX = *index*, ENTRY *status*
One or more allocation units in a second or subsequent DSC entry for the specified multitype dataset does not match the corresponding allocation unit from the first DSC entry for the dataset. The *status* is as described under CATASTROPHIC error. If *status* is deleted, no action is possible. If *status* is retained, examine the entry to determine the cause of the error. From permanent dataset recovery in Startup. System log only. Class, informative.

var MULTI-TYPE DATASET INACTIVATED DUE TO QDT INDEX OUT OF RANGE - INDEX = *index*, QDT SIZE = *size*
The DSC entry for the specified multitype dataset contains a QDT index that exceeds the size of QDT in the system being started. Restart, using a correct system file. From permanent dataset recovery. System log only. Class, informative.

DATE/TIME MISMATCH ON CLASS STRUCTURE ROLL DATASET
The class structure was being written to disk but had not yet been completely written when the system was interrupted. From job class recovery in Startup, system log only. Class, informative.

DATE/TIME MISMATCH PDN=*pdn* ID=*id* ED=*ed*. DEFAULT STRUCTURE USED
The dataset specified on the *JCLASS directive does not appear to be a valid class structure. Correct the dataset or change the *JCLASS directive. From job

class recovery in Startup, system log only. Class, caution.

DEADSTART SELECTED - RECOVERY OF ROLLED JOBS DISABLED
Recovery of rolled jobs was requested, but the startup option is not a restart. From recovery of rolled jobs in Startup, system log only. Class, informative.

DEFAULT JOB CLASS STRUCTURE USED
The default job class structure was put into effect. From job class recovery in Startup, system log only. Class, informative.

INDEX AI *ai* MISMATCH ENTRY ZERO AI *ai*
The allocation unit in the DAT for $ROLL does not match the allocation unit in entry 0. This could result from a user save of a new edition during system operation. The dataset is cleared; recovery is not possible. From recovery of rolled jobs in Startup. System log only. Class, informative.

INDEX DEVICE *ldv* MISMATCH ENTRY ZERO DEVICE *ldv*
The device named in the DAT for $ROLL does not match the device named in entry 0. This could result from a user save of a new edition during system operation. The dataset is cleared; recovery is not possible. From recovery of rolled jobs in Startup. System log only. Class, informative.

INVALID FLAWS IN DEVICE LABEL
The table describing flaws in the label had an AI exceeding the maximum. Check the table and ensure that the correct device type was specified. ZLOG buffer only, from Startup. Class, fatal.

I/O ERROR ON $ROLL - DQM REPLY WAS *status*
DQM returned an error status while reading the roll index. A new edition is created; recovery is impossible. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jobname* - DAT SPACE FULL (JSQ=*jsq*)
The STP table area does not have sufficient room to read in the roll image DAT. Too many datasets are in the queues or the system had too little DAT space at startup. Verify the size of the DAT area in STP. A deadstart might be necessary. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jobname* - DATASET *dsname* DAT POINTS TO STP TABLES (JSQ=*jsq*)
A dataset other than $CS or $IN has a DAT pointer that points to the STP table area. The job is not recoverable. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - DATASET *dn* DAT VALIDATION ERROR *n* (JSQ=*jsq*)
One of the following errors was found while the DAT was being verified for the named dataset.

- DAT page numbers are not consecutive from page 1.
- Continuation page and current page are not both in the STP tables or not both in the JTA.
- JXT ordinal is nonzero for an STP DAT, or it does not match the ordinal in the DNT.
- A DAT page pointer is out of range.
- A device named in a DAT is nonexistent or down.
- An AI listed in a DAT is invalid for the device or is already reserved.
- DANPA does not point correctly to the next partition header.

The following values of *n* indicate which error condition was encountered:

n	Definition
1	JTA offset of the first page is out of range.
2	The first page number is not 1.
3	The first page JXT ordinal is not correct for the origin of the DAT.
4	DAT mentions a device not in the configuration.
5	DAT mentions a DOWN device.
6	An invalid allocation index was found.
7	The dataset attempts to reserve an allocation index already reserved.
8	A multitype dataset has an inconsistent allocation.
9	The DAT continuation page has an invalid offset.
10	The DAT continuation page has an invalid page number.
11	The DAT continuation page has an incorrect JXT ordinal.
12	The DAT page source (JTA or STP) is inconsistent.
13	The continuation page offset at end of partition is bad.
14	The next-partition pointer is invalid.
15	The DAT ended prematurely.
16	DAT names device whose stripe group characteristics have changed.

From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - DATASET *dn* - MAGNETIC TAPE DATASET NOT RECOVERABLE
The job being recovered had a magnetic tape assigned to it and the correct tape position is not guaranteed; therefore, the job is considered irrecoverable. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - DATASET *dn* PDM STATUS *st* ON PSEUDO-ACCESS (JSQ=*jsq*)
Permanent Dataset Manager returned an error status when recovery attempted to access a dataset for the job being recovered. The dataset resides on a down

or released device, or the Dataset Allocation Table in the Job Table Area fails to match the DAT in the DSC. Determine the cause of the error and correct it if possible. Rerun the job. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - DATASET *dn* - QDT INDEX EXCEEDS VALID RANGE *n* (JSQ=*jsq*)
A Dataset Name Table was found in the Job Table Area for the specified job that had a Queued Dataset Table index greater than the current size of the QDT. This can occur if the number of SDT entries is reduced in the new system. The job is not recoverable. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - DNT CHAIN BAD (JSQ=*jsq*)
Recovery detected an error in the DNT chain. The reason is one or more of the following:

- A DNT entry points to an inactive memory pool address.
- The DNT entry points outside the JTA.
- The dataset name in an active DNT is 0.
- A permanent dataset DAT pointer points to STP tables.

From recovery of rolled jobs in Startup. System log only. Class, informative.

JOBNAME *jn*, DQM STATUS *dqmst*, ROLLFILE FAILURE WHILE *operation*
An error status was returned from DQM during a job roll-in or roll-out. The *operation* field can show READING, WRITING, or ALLOCATING. If the error occurred during writing or allocating and if retries fail, consider killing the job. From JSH. Class, informative.

JOB *jn* - INDEX DECLARES NOT RECOVERABLE - DN = *dn* - STATUS = *nn* (JSQ=*jsq*)
The system roll index indicates that the specified job was in an irrecoverable state when the system was interrupted. The job might never have been rolled out, or it might have performed one of the following irrecoverable functions since it was last rolled:

- Delete, adjust, or modify a permanent dataset
- Random write to a dataset
- Write following any positioning operation on a dataset
- Release of a local dataset

In any case, the job is put in the input queue. The value of the STATUS parameter indicates the reason the job being declared is not recoverable in the index. If a dataset operation is the cause, the DN value indicates the first dataset to cause the job to be not recoverable. Values for STATUS follow:

ST	Definition
1	Job has not been rolled out.
2	A write to a random dataset was performed.
3	A write followed a read or rewind or was the first write to a sequential dataset.
4	The named dataset was saved.
5	The named dataset was deleted.
6	The named dataset was adjusted.
7	The named dataset was modified.

From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - I/O ERROR *errnum* ON ROLL DATASET (JSQ=*jsq*)
The Disk Queue Manager returned an error when recovery of rolled jobs attempted to read the roll image for the specified job. Determine the cause of the error and add a flaw card if necessary. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - JOB IS NOT RERUNNABLE (JSQ=*jsq*)
A job that is not recoverable is also not rerunnable. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative

JOB *jn* - JOB SUCCESSFULLY RECOVERED (JSQ=*jsq*)
Recovery of a rolled out job has been successfully completed. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - JOB WILL BE RERUN (JSQ=*jsq*)
A job that cannot be recovered can be rerun. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - JTA IMAGE ROLL DAT BAD ORDINAL (JSQ=*jsq*)
The Job Execution Table ordinal from one or more Dataset Allocation Table pages does not match the job's JXT ordinal. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - JTA IMAGE ROLL DAT NEXT PAGE POINTER BAD (JSQ=*jsq*)
The pointer to the next page of the Dataset Allocation Table for the rollout dataset does not point to the System Task Processor table area. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - JTA IMAGE ROLL DAT PAGE NUMBER ERROR (JSQ=*jsq*)
The image of the rollout dataset Dataset Allocation Table from the Job Table Area does not have consecutive page numbers beginning with page 1. The job is not recovered. From recovery of rolled jobs

in Startup. System log only. Class, informative.

JOB *jn* - JTA LENGTH (JTL) *jtl* JXT LENGTH (JXJTL) *jxjtl* MISMATCH (JSQ=*jsq*)
The Job Table Area length from field JTL of the roll image JTA does not match field JXJTL in the roll image copy of the Job Execution Table. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - JXT NAME *jobname 1* MISMATCH JTA NAME *jn 2* (JSQ=*jsq*)
The job name from the Job Execution Table does not match the job name from the Job Table Area. *jobname 1* is from the roll index; *jn 2* is from the roll image. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - NO INPUT SDT (JSQ=*jsq*)
No input queue entry corresponding to the specified job could be found. Possible cause is an error such as a dataset residing on a down device. Determine the cause of the error and correct it if possible. Rerun the job. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - REQ SIZE *size* GREATER THAN AVAIL. LWA=*addr* FWA=*addr* (JSQ=*jsq*)
This job requires more than the available memory, which is either the space available at startup time for the Job Table Area or the requested job size compared with the space available for a user job. The required amount of space and the first and last available addresses are shown. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - ROLL DAT POINTS TO MISSING DEVICE *ldv* (JSQ=*jsq*)
The roll image for the dataset is on a nonexistent device. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - ROLL DAT POINTS TO DOWN DEVICE *ldv* (JSQ=*jsq*)
The roll image for the dataset is on an unavailable device. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB *jn* - ROLLOUT APPARENTLY INCOMPLETE (JSQ=*jsq*)
The system was interrupted when a rollout was initiated but not completed. The date/time of rollout in the first sector of the roll image does not match the date/time in the last word of the field length. The job is not recovered. From recovery of rolled jobs in Startup. System log only. Class, informative.

JOB CLASS STRUCTURE RECOVERY COMPLETE
The rolled class structure has been successfully recovered. From job class recovery. Class, informative.

MISSING/BAD RRJ CODE
An *RRJ command was present but end-of-line was encountered too soon, or the parameter value was out of range. Correct the parameter file. ZLOG buffer only. Class, caution.

NUMBER OF RESIDENT SDR ENTRIES DECREASED
The system being started has room for fewer SDR entries in the STP table area than the entries on the roll dataset. Regenerate the system with more SDR entries or restart the system with a parameter file specifying *SDR to allocate a new roll dataset. Then run JSYSDIR to initialize the file entries. This message appears on the master operator console only. From SDR. Class, fatal.

OLD SYSTEM JXT COUNT *count* GREATER THAN CURRENT SYSTEM COUNT *count*
Recovery is not possible because the previously running system was assembled with more JXT entries than the current system. The roll index is cleared. From recovery of rolled jobs. System log only. Class, informative.

RECOVERY OF ROLLED JOBS ABORTED
Recovery is not possible due to a previously described error. From recovery of rolled jobs in Startup. System log only. Class, informative.

$SDR, FILE ACCESS FAILED *status*
Permanent Dataset Manager returned the specified error status when STARTUP attempted to access the System Directory rollout dataset. Determine the cause of the error and correct it. The JSYSDIR job can be run to initialize the System Directory. System log only. From Startup. Class, fatal.

$SDR, FILE ADJUST FAILED *status*
Permanent Dataset Manager returned the specified error status when STARTUP attempted to increase the size of the System Directory roll dataset to allow for an increase in the number of SDR entries in the system being started. If the $SDR roll dataset is unusable, restart the system, specifying *SDR in the parameter file to force creation of a new edition. Then run a JSYSDIR job to initialize the $SDR entries. System log only. From Startup. Class, fatal.

STATUS *status* ON ACCESS OF *JCLASS
DATASET PDN=*pdn*, ID=*id*, ED=*ed*
DEFAULT STRUCTURE USED
STARTUP was unable to access the specified dataset; the default structure was put into effect. Determine the cause of the error and correct it. Run a job to establish the desired structure. System log only. From job class recovery. Class, caution.

SYSTEM DUMP NOT SAVED - PDM RETURNED STATUS *status*
STARTUP was unable to save the system dump as a permanent dataset. The *status* value is the error code returned by PDM. The dump remains available in the reserved area as long as no new dump is taken. Determine the reason for failure to save the dump and correct it. Restart the system. System log only. From System Dump. Class, informative.

2.8.3 STARTUP MESSAGES BEGINNING WITH ***** FATAL STARTUP ERROR *****

These Startup messages have a two-part error format, as indicated below:

***** FATAL STARTUP ERROR *****
error-text-1
AT THE TIME OF ERROR BO = *n* P = *n*
error-text-2
Startup has detected an error for which no recovery is provided. The error condition must be corrected and another Startup performed.

The contents of *error-text-1* describe which fatal Startup error was encountered. *error-text-1* messages are listed below.

The contents of *error-text-2* describe the location of the error. This portion of the message is not always present. *error-text-2* messages are listed immediately following *error-text-1*.

Error-text-1 messages

INTERNAL STARTUP/CONFIGURATION ERROR
Startup has detected an impossible condition, such as not being able to locate a DRT entry for a device when the entry previously had been found.

MULTIPLE MASTER DEVICES DEFINED
Startup found an existing device label during an INSTALL which has the master device flag set, but it is not the device that the EQT defines as the master. The label must be overwritten by site engineers and another Startup performed. From Startup.

UNABLE TO CREATE DSC DATASET
Startup received an unexpected error from DQM when allocating the Dataset Catalog. Determine the error status and correct the problem, then perform another Startup. Occurs only for INSTALL. From Startup.

UNABLE TO ALLOCATE SYSTEM DUMP AREA
Startup received an unexpected error from DQM when allocating the system dump reserved area. Determine the error status and correct the problem, then perform another Startup. Occurs only for INSTALL. From Startup.

UNABLE TO WRITE DAT TO SYSTEM DUMP
Startup received an unexpected error from DQM when attempting to write the dump DAT to the first sector. Determine the error status and correct the problem, then perform another Startup. Occurs only during INSTALL. From Startup.

INVALID FLAWS IN DEVICE LABEL
A device label was found during a RESTART/DEADSTART which has invalid flaws in the flaw map. The operator will have been given the option of accepting the bad flaw map earlier. The device must be overwritten by site engineers to eliminate the bad flaw information, and another Startup must be performed. From Startup.

NO MASTER DEVICE FOUND DURING DEADSTART/RESTART
Startup did not find a device with a master device label, and the current Startup is not an INSTALL. Correct the configuration and perform another Startup, or perform an INSTALL. From Startup.

INTERNAL ERROR - BAD DUMP DAT
Startup could not process the system dump reserved area DAT. The system dump must be reallocated via INSTALL. From Startup.

NO ROOM IN STP DAT FOR DSC DATASET
Startup unable to copy the DAT for the Dataset Catalog into the STP DAT space. If the correct version of COS is being started, an INSTALL is required. If a version of COS with an unusually small DAT space defined is being started, a different version must be started with sufficient STP DAT space. From Startup.

DSC DAT BAD OR FLAW EXISTS IN DSC
The Dataset Catalog DAT read from the master device label has one or more allocation conflicts with disk space reserved by flaws or named in the EFT. Remove the offending flaws, or perform an INSTALL. From Startup.

NO ROOM IN STP DAT TO RECOVER PERMANENT DATASET
Startup was unable to read the DAT for a permanent dataset into the STP DAT space from the Dataset Catalog. If the dataset is a spooled dataset, a DEADSTART might succeed. Otherwise, a version of COS with more STP DAT space defined must be started, or an INSTALL must be performed. From Startup.

MEMORY POOL SPACE INSUFFICIENT FOR TEXT
Startup was unable to allocate space in the memory pool for the text information associated with a spooled dataset. A DEADSTART must be performed. From Startup.

UNEXPECTED PDM ERROR ON SAVE OF ROLL INDEX DATASET
Startup received an unexpected error while attempting to SAVE the roll index dataset. Determine the cause of the problem and perform another Startup. From Startup.

UNEXPECTED PDM ERROR ON RELEASE OF ROLL INDEX DATASET
Startup received an unexpected error while attempting to release the roll index dataset. This occurs only if Startup is unable to read the dataset, and must create a new edition. From Startup.

TABLE SIZE CHANGED - CAN'T RECOVER ROLLED JOBS
The size of a system table has been changed, and rolled jobs cannot be recovered. A RESTART specifying the *RRJ,0 parameter file directive, or a DEADSTART, must be performed. From Startup.

DISK SPACE NEGATIVE WHILE RECOVERING ROLLED JOBS
Startup detected that the available space on a disk became negative while reserving disk space for datasets during recovery of rolled jobs. A DEADSTART or a RESTART with the *RRJ,0 parameter file directive might succeed. From Startup.

DQM ERROR ON REWRITE OF JTA DURING RRJ
Startup received an error status from DQM while rewriting the JTA of a job that has been recovered. Determine the cause of the error and correct the problem. If the problem persists, a DEADSTART or a RESTART with the *RRJ,0 parameter file directive may succeed. From Startup.

DQM ERROR WHILE ALLOCATING ROLL INDEX
Startup was unable to allocate disk space to contain the roll index dataset. A DEADSTART might succeed. From Startup.

I/O ERROR WHILE WRITING ROLL INDEX DATASET
Startup received an error from DQM while attempting to rewrite the roll index dataset following recovery of rolled jobs. A RESTART with the *RRJ,0 parameter file directive or a DEADSTART may succeed. If a flaw has developed on the disk addition of a flaw parameter file directive will cause Startup to allocate a new edition of the index dataset, without recovering existing rolled jobs. From Startup.

I/O ERROR WHILE WRITING SYSTEM DIRECTORY
Startup has received an error from DQM while initializing the system directory dataset. Determine the cause of the problem and correct it and then perform another Startup. If the problem persists, a RESTART or DEADSTART with the *SDR parameter file directive might succeed, in which case the JSYSDIR job must be run following the Startup. From Startup.

UNEXPECTED PDM ERROR ON SYSTEM DIRECTORY DATASET
Startup received an error from PDM while attempting to ACCESS, ADJUST, or SAVE the system directory dataset. Determine the cause of the error and correct it. A RESTART or DEADSTART with the *SDR parameter file directive might succeed, in which case the JSYSDIR job must be run following the Startup. From Startup.

I/O ERROR WHILE READING SYSTEM DIRECTORY
Startup received an error from DQM while attempting to read the system directory dataset. Determine the cause of the error and correct it. If the problem persists, a RESTART or DEADSTART with the *SDR parameter file directive might succeed. From Startup.

INSUFFICIENT MEMORY FOR STARTUP TO EXECUTE
Startup is unable to allocate sufficient memory to hold all of its internal tables. A version of COS with sufficient memory configured must be started. From Startup.

PDS IS FULL
Startup was unable to obtain a PDS entry for a permanent dataset such as the roll index or system directory datasets. A DEADSTART or RESTART with the *RRJ,0 parameter file directive might succeed. A version of COS with additional space defined for the PDS table might be started. From Startup.

SYSTEM DIRECTORY DATASET IS BAD
Startup found that the expected format of the system directory did not match the data in the dataset. A RESTART or DEADSTART with the *SDR parameter file directive might succeed. From Startup.

NO SPACE REMAINS IN AUT FOR RECOVERED INTERACTIVE INPUT DATASET
Startup was unable to construct an AUT entry for an interactive job being recovered. Either a version of COS with more AUT space defined must be started, or a DEADSTART must be performed. From Startup.

UNEXPECTED PDM ERROR ON ACCESS OF $SDR
Startup received an unexpected error from PDM while attempting to ACCESS the system directory. Determine the cause of the problem and correct it. A RESTART or DEADSTART with the *SDR parameter file directive might succeed. From Startup.

ENTRIES CONTAINED IN RECOVERED $SDR EXCEED AVAILABLE SPACE
There is insufficient space defined in the STP SDR table to contain all of the entries in the system directory dataset. A version of COS with more SDR space might be started. A RESTART or DEADSTART with the *SDR parameter file directive might succeed. From Startup.

INTERNAL ERROR - BAD ERROR CODE GIVEN
The fatal Startup error reporting routine was called with an undefined error code. Determine from the value of BO displayed and a source listing what the error condition should have been and take action accordingly. From Startup.

Error-text-2 messages

The value of *error-text-2* will be one of the following, if it is present. For some errors, no *error-text-2* will exist:

DQM STATUS = *n*
This is the error status which Startup received from DQM. From Startup.

PDM STATUS = *n*
n is the error status which Startup received from PDM. From Startup.

DEVICE NAME = *device*
n is the device which the *error-text-1* refers to. From Startup.

ZEND = *n* MEMMAX = *n*
The value shown for ZEND is the highest memory address Startup requires for its internal tables. The value shown for MEMMAX is the highest usable memory address, after log buffers, etc. are allocated. Startup cannot execute successfully if the ZEND value is greater than MEMMAX. From Startup.

PDN=*pdn* ID=*uid* ED=*nnn*
US=*usernum*
Identifies the permanent dataset referred to in preceding text. From Startup.

READERS COMMENT FORM

CRAY-OS Message Manual

SR-0039 C

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME _____

JOB TITLE _____

FIRM _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention:
PUBLICATIONS

1440 Northland Drive
Mendota Heights, MN 55120
U.S.A.

READERS COMMENT FORM

CRAY-OS Message Manual SR-0039 C

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME _____

JOB TITLE _____

FIRM _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention:
PUBLICATIONS

1440 Northland Drive
Mendota Heights, MN 55120
U.S.A.