Prime®

Advanced Programmer's Guide, Volume 0: Introduction and Error Codes

Revision 22.0

DOC10066-3LA

Advanced Programmer's Guide, Volume 0: Introduction and Error Codes

Third Edition

Glenn S. Morrow

This guide documents the software operation of the Prime Computer and its supporting systems and utilities as implemented at Master Disk Revision 22.0 (Rev. 22.0).

Prime Computer, Inc., Prime Park, Natick, MA 01760

Printing History

Preliminary Edition (DOC9229-1LA) January 1985 for Revision 19.4.0
First Edition (DOC10066-1LA) November 1985 for Revision 19.4.2
Second Edition (DOC10066-2LA) September 1987 for Revision 21.0
Third Edition (DOC10066-3LA) October 1988 for Revision 22.0

Credits

Editorial: Barbara Fowlkes
Project Support: Joan Karp, Nick Fichter
Graphics Support: Mingling Chang, Robert Alba
Document Preparation: Mary Mixon
Composition: Julie Cyphers, Sharon Temple
Production: Judy Gordon
Design: Carol Smith

How To Order Technical Documents

Follow the instructions below to obtain a catalog, a price list, and information on placing orders.

United States Only: Call Prime Telemarketing, toll free, at 1-800-343-2533, Monday through Friday, 8:30 a.m. to 5:00 p.m. (EST).

International: Contact your local Prime subsidiary or distributor.

Customer Support Center

Prime provides the following toll-free numbers for customers in the United States needing service:

1-800-322-2838 (Massachusetts)
1-800-541-8888 (Alaska and Hawaii)
1-800-343-2320 (within other states)

For other locations, contact your Prime representative.

Surveys and Correspondence

Please comment on this manual using the Reader Response Form provided in the back of this book. Address any additional comments on this or other Prime documents to:

Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, MA 01701

Contents

About This Book

The *Advanced Programmer's Guide* is a four-volume series intended for programmers who are experienced with both 50 Series™ computer systems and at least one high-level language (preferably PL/I or FORTRAN). This series consists of four volumes:

- *Advanced Programmer's Guide, Volume 0: Introduction and Error Codes* (DOC10066-3LA) (this volume)
- *Advanced Programmer's Guide, Volume I: BIND and EPFs* (DOC10055-1LA)
- *Advanced Programmer's Guide, Volume II: File System* (DOC10056-2LA)
- *Advanced Programmer's Guide, Volume III: Command Environment* (DOC10057-1LA)

Users of this series should be familiar with the following Prime publications:

- *PRIMOS User's Guide* (DOC4130-5LA)
- *Programmer's Guide to BIND and EPFs* (DOC8691-1LA) and its update (UPD8691-11A)
- *Subroutines Reference I: Using Subroutines* (DOC10080-2LA)
- *Subroutines Reference II: File System* (DOC10081-1LA) and its update (UPD10081-12A)
- *Subroutines Reference III: Operating System* (DOC10082-1LA) and its update (UPD10082-12A)
- *Subroutines Reference IV: Libraries and I/O* (DOC10083-1LA) and its update (UPD10083-12A)
- *Subroutines Reference V: Event Synchronization* (DOC10213-1LA)

Users of this series should also be familiar with Prime system architecture, as described in the *50 Series Technical Summary* (DOC6904-2LA) and in the *System Architecture Reference Guide* (DOC9473-2LA).

Specifics of This Volume

This volume contains reference information applicable to the subjects described in the other volumes:

- An explanation of the presentation of subroutine calls and general coding guidelines (Chapter 1)
- Standard error codes used by PRIMOS, along with their messages and meanings (Appendices A and B)
- New features of recent PRIMOS revisions that may be of interest to advanced programmers (Appendix C)
- A master index encompassing the entire series

Specifics of the Series

The *Advanced Programmer's Guide* series is designed for system-level programmers. It describes the lowest-level interfaces supported by PRIMOS and its utilities. Higher-level interfaces not described in this series include

- Language-directed I/O
- The applications library (APPLIB)
- The sort packages (VSRTLI, SyncSort/PRIME, and MSORTS)
- Data management packages (such as MPLUSLB and PRISAMLIB)
- Other subroutine packages

All of the above higher-level interfaces are described in other books, such as language reference guides and the five volumes of the *Subroutines Reference* series.

This series documents low-level interfaces for use by programmers and engineers who are designing new products, such as language compilers, data management software, electronic mail subsystems, utility packages, and so on. Such products are themselves higher-level interfaces, typically used by other products rather than by end users, and therefore, must use some or all of the low-level interfaces described in this series for best results.

Because of the technical content of the subjects presented in this series, it is expected that these guides will be regularly used only by project leaders, design engineers, and technical supervisors, rather than by all programmers on a project. Most of the information in this series deals with interfaces to PRIMOS that are typically used only in small portions of a structured program, and with overall project design issues that should be considered before coding begins. Once the project is designed and the PRIMOS interfaces are designed and coded, most of the modules of a typical project can then be written by programmers whose knowledge of these issues is minimal.

Prime Documentation Conventions

The following conventions are used throughout this document. Examples illustrate the uses of these conventions in typical applications.

Convention	*Explanation*	*Example*
UPPERCASE	In calling sequence diagrams, words in uppercase boldface represent the subroutine name or keyword to be entered as shown.	**SLIST**
UPPERCASE WORDS (not boldface)	Represent the data type of subroutine arguments.	HALF INT
lowercase	In calling sequence diagrams, words in lowercase represent the subroutine arguments for which the user must substitute a suitable variable.	**(key, unit)**
Parentheses ( )	In calling sequence diagrams, parentheses must be entered exactly as shown.	**(key, unit, addr)**

Calling Sequences and Coding Guidelines

Calling Sequence Conventions

The *Advanced Programmer's Guide* series contains diagrams of the calling sequences of system subroutines. These diagrams are intended to complement the discussion of the subroutines in the *Subroutines Reference* series. Similar calling sequence diagrams are also found in an appendix to *Subroutines Reference V: Event Synchronization.*

Figure 1-1 is a sample diagram of a calling sequence. Each calling sequence diagram occupies one full page. The subroutine (or procedure) name is listed in the middle of the page, followed on the same line by dummy parameter names listed in parentheses and separated by commas. This is the basic calling sequence for the procedure.

Above this basic calling sequence are the input arguments; below the calling sequence are the output arguments. An arrow connects each argument to a dummy parameter name. The direction of these arrows indicates the flow of information. These arrows also visually connect parameter names to information about the parameters. This information includes the argument's data type and a brief description of the argument.

Some diagrams may contain other elements, such as

- A required value or a list of permitted values for keys or other parameters.
- An illustration of the format of an input or output argument.
- A dot and arrow indicating that a pointer to a data area must be supplied. Execution of the subroutine writes information into this data area.

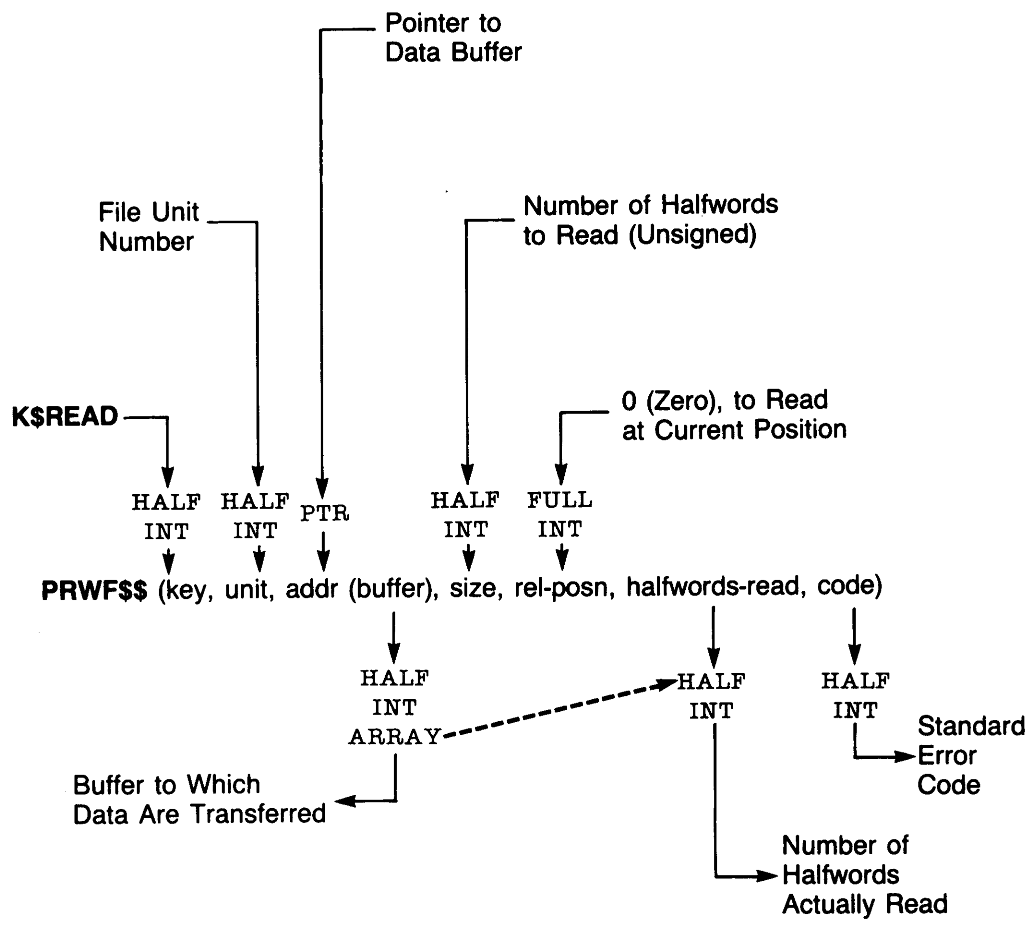
Data types are specified in a data type description language. This language is further described in this chapter. You must convert the data type used here to the appropriate data type for your programming language. In addition to the data type description language, this series often includes PL/I or FORTRAN versions of structures.

Procedures that are functions return a function value. This return value and its data type are illustrated below the name of the procedure itself.

In addition to showing the arguments and their data types, each calling sequence diagram

- Shows the calling sequence for a single type of operation performed by the procedure
- Illustrates the relationships between interdependent parameters in the calling sequence

Read a File


Pointer to Data Buffer
File Unit Number
Number of Halfwords to Read (Unsigned)
K$READ
0 (Zero), to Read at Current Position
HALF INT
HALF INT
PTR
HALF INT
FULL INT
PRWF$$ (key, unit, addr (buffer), size, rel-posn, halfwords-read, code)
HALF INT ARRAY
HALF INT
HALF INT
Standard Error Code
Buffer to Which Data Are Transferred
Number of Halfwords Actually Read

Side Effects: Contents of buffer elements halfwords-read +1 through size are undefined after the operation if fewer halfwords than requested were read.

Figure 1-1
Sample Subroutine Calling Sequence

Therefore, a multipurpose subroutine such as PRWF$$ is described using several different calling sequence diagrams: one for reading a file, another for writing a file, and a third for positioning within a file.

Some calling sequence diagrams contain dotted arrows between related arguments. These relationships often involve a parameter (such as a character string) whose length is specified by another parameter in the calling sequence.

Data Types

Table 1-1 lists the generic data types and their PL/I and FORTRAN equivalents that are used throughout the *Advanced Programmer's Guide* series. (The diagrams in *Subroutines Reference V* use PL/I data types.)

Table 1-1
Data Types and Their PL/I and FORTRAN Equivalents

Data Type	*PL/I*	*FORTRAN*
HALF INT	FIXED BIN(15)	INTEGER*2
FULL INT	FIXED BIN(31)	INTEGER*4
n STRING	CHARACTER(n)	INTEGER*2 ((n+1)/2)
<=n STRING	CHARACTER(n) VARYING	INTEGER*2 ((n+3)/2)
n BIT	BIT(n)	INTEGER*2 ((n+15)/16) w/masking
PTR	POINTER and ADDR( )	INTEGER*2 (3) and LOC( )
STRUC	[1]	[1]
ARRAY(n)	[2]	[2]

[1]Structures are usually illustrated in the same calling sequence diagram or in another related diagram, or their declarations are provided on a page near the diagram. Structures are also known as record data types in other languages.

[2]Arrays are either a constant length which is indicated in parentheses, or a varying length controlled by a parameter or a subfield in a parameter. Varying length arrays have dotted arrows from the word ARRAY to the parameter (or its subfield) that controls the length of the array.

The last three data types in Table 1-1 are discussed more fully in the subsection entitled Pointers, Arrays, and Structures, later in this chapter.

In cases where the length of an item is specified in the data type, such as <=128 STRING, and a dotted arrow is also drawn to a parameter that defines the operative length, then the length in the data type is the maximum length for that item.

If you are unsure as to the meaning of a keyword, arrow, or other illustrative mark, consult the *Subroutines Reference* series for more information on the subroutine or data structure.

Keys

Some subroutines take an input key argument. A **key** is a literal value that you use to specify the operation to be performed by the routine. In most calling sequence diagrams that involve a key argument, a list of valid (or appropriate) key values is provided. Each keyword corresponds to a specific operation. For example, the **k$read** key specifies a read operation.

When the construction of a key is complex, two or more lists of keywords are often shown, enclosed in braces { }, with + signs to indicate addition. As with command formats, choose one keyword from each list in braces. Specify the + signs in your program to indicate the addition of these multiple keywords. For example, your program might specify a key value of **k$rdwr+k$ndam+k$getu.**

To define keywords that have names beginning with K$, use a %INCLUDE or $INSERT statement to insert the appropriate SYSCOM>KEYS.INS.*language* file into your program. See the *Subroutines Reference* series for more information on this topic.

Standard Error Code

Many subroutines include a **standard error code** as a parameter. This is a HALF INT value returned by the subroutine to indicate the degree of success encountered by the subroutine. Each error code can be represented by an integer value or a mnemonic. All standard error code mnemonics begin with E$. Always use these mnemonic values in your programs.

For example, after each subroutine call your program should always check the standard error code to ensure that its value is E$OK (integer value 0). A value of E$OK means a successful call. Other values indicate specific errors or conditions worth noting.

Appendix A contains a list of PRIMOS standard error codes along with a description of the meaning of each code. This list is ordered numerically by error code number. Appendix B contains an alphabetical list of the error message displayed for each error code. The alphabetical list is cross-referenced with the numeric list.

To define standard error code mnemonics for your program, use a %INCLUDE or $INSERT statement to insert the appropriate SYSCOM>ERRD.INS.*language* file into your program. See the *Subroutines Reference* series for more information on this topic.

Side Effects

Where appropriate, the side effects of a subroutine are listed at the bottom of the calling sequence diagram. Side effects are those actions taken by the procedure that are not obviously a designed function of the procedure. For example, a side effect of a call to the TSRC$$ subroutine may change the cache attach point without notifying its caller.

General Coding Guidelines

When writing programs that use standard PRIMOS subroutines, observe the following guidelines to ensure that your programs continue to function normally on subsequent revisions of PRIMOS:

- Your program must ignore any reserved or undefined information returned to it by a subroutine. For example, if a 16-bit halfword contains one defined bit and fifteen reserved bits, your program must mask off the fifteen reserved bits before analyzing the halfword to determine the value of the one defined bit.
- Your program must zero-fill any reserved or undefined arguments that it passes to a subroutine, except where otherwise specified.
- The maximum number of defined character values in a returned character string is the operative length of the string; characters beyond that point have undefined values and must be ignored. For example, a character string with a data type of 32 STRING that has been returned to the caller along with an operative length of 13 (as indicated by the dotted arrow in Figure 1-1) has undefined values for characters 14-32 in the returned string.
- Arrays, structures, and similar items with operative lengths are considered undefined beyond those operative lengths.

Pointers, Arrays, and Structures

A number of PRIMOS subroutines deal with arrays and structures. The PRWF$$ subroutine, for example, uses an array as a buffer. Some of the ACL subroutines use structures to manipulate access control lists. A subroutine that deals with an array or a structure requires a pointer to the array or structure as part of its calling sequence.

Pointers, arrays, and structures are represented as a PL/I language construct in the following format:

addr(target-object)

Variable	*Meaning*
addr	The literal string *addr* with the data type PTR (pointer)
target-object	The name of the array or structure, enclosed in parentheses, as defined in the program by a data declaration statement

Figure 1-1 shows a calling sequence containing a pointer to a buffer having the data type HALF INT ARRAY.

In some cases, the array or structure serves as both an input and an output argument, although it is not necessarily used to the same extent in both. For example, a structure specified as an input argument might contain only a required version number that you set to a specific value, whereas

PRIMOS Error Codes

Error Code Presentation

This appendix contains an annotated list of the standard PRIMOS error codes. The error codes are listed in numerical order. Appendix B contains a cross-reference listing of these error codes, listed alphabetically by the text of the error message.

Each error code consists of a number, a mnemonic, and an error message. User programs should always check the mnemonic value of an error code, not the numeric value or error message. You can use the ER$PRINT subroutine to display an error message on your terminal or use the ER$TEXT subroutine to return an error message to a variable in your program. These subroutines are further described in *Subroutines Reference III: Operating System.*

The description of each error code is in the following format:

```
E$xxxx (nnn)                                text of error message
```
description of error

Variable	*Meaning*
`E$xxxx`	The mnemonic for the error code
`nnn`	The numeric value of the mnemonic
`text of error message`	The error message displayed by ER$PRINT or ER$TEXT for that error code
description of error	The description of the error code

Mnemonics for error codes are defined by files in SYSCOM for several languages:

Language	*Filename in SYSCOM*
C	ERRD.INS.CC
FORTRAN 77	ERRD.INS.FTN
FORTRAN IV	ERRD.INS.FTN
Pascal	ERRD.INS.PASCAL
PL/I	ERRD.INS.PL1

PMA	ERRD.INS.PMA
BASIC/VM	not available
COBOL	not available

Use the appropriate %INCLUDE (Pascal and PL/I), #include (C), or $INSERT (F77, FTN, and PMA) in your program to provide definitions of all the standard error codes for your program.

Subroutines Reference I: Using Subroutines contains more information on these files.

Notes

Severity code numbers, sometimes returned by CPL programs, have no correspondence in meaning with standard PRIMOS error codes with the same numeric values. Severity codes are chosen arbitrarily by the CPL programmer.

When running user programs that involve a subsystem such as DPTX, you may encounter messages that are not listed in this appendix. These messages are related to their respective subsystems, not to PRIMOS. Refer to the appendix subsystem documentation for further information on these error codes.

PRIMOS Standard Error Codes

`E$OK (0)` `Operation completed successfully.`

The operation completed successfully. No error was detected.

`E$EOF (1)` `End of file.`

The end-of-file point was reached during an operation on a file system object.

End-of-file errors may occur, for example, when

- Reading directory entries via DIRSE, DIRRD, DIR$LS, or RDEN$$
- Positioning a file system object via PRWF$$ or SGDR$$
- Reading data from a file via PRWF$$ or RDLIN$
- Attempting to open for reading a nonexistent member of a segment directory while positioned at the end of that segment directory

The interpretation of this error depends upon the operation performed. For example, when returned by PRWF$$ while trying to read data from a file, it indicates that end-of-file was reached but that some data may have been successfully read. However, when returned by DIRRD, EEOF indicates that the end of the directory was reached and no entry was returned to the calling program.

`E$BOF (2)` — `Beginning of file.`

An attempt was made to position a file system object to a point before the beginning of the file. This error results if PRWF$$ is called with a relative-position key and a negative relative position that would, when applied to the current position, produce an absolute position whose value is less than zero.

`E$UNOP (3)` — `Unit not open.`

The file-unit is closed or is not open for the type of operation being requested. For example, an attempt to read from a file that is open only for writing causes this error, as does an attempt to write to a file that is open only for reading.

This error code is also returned if an attempt is made to truncate a file that is not open for writing.

`E$UIUS (4)` — `Unit in use.`

The unit number supplied to a subroutine that is attempting to open a file system object is already in use. This error occurs only when static file-unit allocation is used (that is, when the k$getu subkey is not used).

`E$FIUS (5)` — `File in use.`

The file system object being accessed is already open on another file-unit or by another user. This error occurs if an attempt is made to

- Open an object that is already open by another user or by the same user on another file-unit, and the read/write lock of the object disallows the attempt
- Rename an object that is open by another user or by the same user on another file-unit
- Rename a file directory that is in use as an attach point by any user
- Set a quota on a nonquota directory that is in use or contains other files or directories that are in use
- Change the open mode of a file-unit, by calling CH$MOD or SRCH$$ (with the k$cacc key), when the object is open by another user or by the same user on another file-unit and the new open mode conflicts with the other open mode
- Truncate a file or segment directory that is open by another user or by the same user on another file-unit
- Access a file that is open for VMFA read

`E$BPAR (6)` — `Bad parameter.`

An invalid value or combination of values was supplied to a subroutine. Many system subroutines are capable of returning this error code. If this error occurs, check the parameter values used in your subroutine call against the description in the *Subroutines Reference* series.

`E$NATT (7)` — `No directory attached.`

Usually occurs when the directory to which the user is attached is removed from the system, as when a disk is shut down, or in the case of a network failure when attached to a directory on a remote disk. Use one of the AT$ subroutines, or the ATTACH or ORIGIN command, to reestablish an attach point.

`E$FDFL (8)` — `Directory entry list is full.`

An attempt was made to add an entry to a directory that does not have room for the entry. Such entries include entries for newly created file system objects, new entries for name changes of existing objects, ACL information placed on a file system object, and so on. FIX_DISK may compress such a directory sufficiently to allow new entries to be added (if the –UFD_COMPRESSION and –FIX options are used), but, because a directory must reside in a single segment, there is a limit of approximately 4000 entries per directory even in a fully compressed directory. (This limit varies according to the lengths of objectnames, ACL information present, and the current state of directory fragmentation.)

`E$DKFL (9)` — `Disk is full.`

The operation requires an additional record to be allocated on a disk partition, but all records on that partition are already allocated. Use the AVAIL command to display the number of total and available records on a disk partition.

Some operations are nonrecoverable after returning this error code. For example, the WTLIN$ subroutine does not restore the file location pointer to the original location when it encounters this error; the file location is undefined. On the other hand, the PRWF$$ subroutine does reset the file location pointer to the value it held before the disk full error was encountered.

`E$NRIT (10)` — `Insufficient access rights.`

The operation could not be performed because the user running the program has insufficient access to perform the operation. In most cases, access is determined by either the ACL placed on a file system object or the password protection. In some cases, only the System Administrator or the supervisor terminal user (User 1) may perform the operation. In a few cases, such as calling the LOGO$$ subroutine, access is determined by matching user names. Other cases exist, as indicated in the description of the subroutine that returned this error code.

`E$FDEL (11)` — `File open on delete.`

An attempt to delete a file, segment directory, or file directory failed because the object was either in use by another user, in use by the same user on another file-unit, or an EPF open for VMFA read.

`E$NTUD (12)` — `Not a directory.`

The attempted operation requires the target file system object to be a file directory, but it is not a file directory.

`E$NTSD (13)` — `Not a segment directory.`

The attempted operation requires the target file system object to be a segment directory, but it is not a segment directory.

`E$DIRE (14)` — `Operation illegal on directory.`

The object being referenced is a file directory or a segment directory. The requested operation, or the subroutine called to perform it, cannot act on a directory.

`E$FNTF (15)` — `Not found.`

The target of the operation does not exist. Typically, the target is a file system object, but it can be any entity whose existence or nonexistence can be determined.

`E$FNTS (16)` — `Not found in segment directory.`

The desired entry number was not found in the segment directory opened on the specified file-unit. Either no entry was found at the current position, or the specified entry could not be found by searching the segment directory.

`E$BNAM (17)` — `Illegal name.`

The name supplied as a parameter for the operation does not meet the syntactic requirements for the corresponding object. E$BNAM is also returned by the LOGO$$ subroutine.

`E$EXST (18)` — `Already exists.`

The object to be created already exists.

`E$DNTE (19)` — `Directory is not empty.`

An operation, such as the deletion of a directory, cannot be performed because the directory is not empty.

`E$SHUT (20)`

Not currently returned by PRIMOS.

`E$DISK (21)` — `Disk I/O error.`

The FORCEW subroutine returns this error code if a disk error occurred during the forced writing of locate buffers. Other file system and low-level disk subroutines may return this error code if a disk error occurs.

`E$BDAM (22)` — `Bad DAM file.`

EPF$MAP or EPF$RUN return this error code if the EPF DAM file structure has been corrupted.

`E$PTRM (23)` `Pointer mismatch found.`

Many PRIMOS subroutines (for example, RDLIN$ and WTLIN$) return this error code if a pointer mismatch is detected. This is usually caused by a corrupted disk. Run FIX_DISK to repair the disk.

`E$BPAS (24)` `Bad password.`

The password specified does not match the actual password.

`E$BCOD (25)`

Not currently returned by PRIMOS.

`E$BTRN (26)` `Bad truncate of segment directory.`

SGDR$$ returns this error code if an attempt was made to truncate a segment directory that has members beyond the desired truncation point. Such members must be removed before the truncation operation can succeed.

`E$OLDP (27)` `Old partition.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$BKEY (28)` `Bad key.`

Many PRIMOS subroutines use this error code to indicate that a *key* argument supplied by the caller is not a valid value. Check the description of the subroutine being called for valid values for the *key* argument.

`E$BUNT (29)` `Bad unit number.`

Either an invalid file-unit number was supplied to a system subroutine or an invalid device unit number was supplied.

Invalid File-unit Number: The file-unit number supplied is invalid (out of range). Note that file-units 1-128 are valid file-unit numbers (unless the System Administrator has reduced the number of valid file-units by using the FILUNT directive in the system configuration file). Larger file-units may become valid as a user uses more dynamically allocated units.

Invalid Device Unit Number: The device unit number is invalid. The range of valid unit numbers depends upon the type of device involved. (See the ASSIGN command in the *PRIMOS Commands Reference Guide.*)

`E$BSUN (30)` `Bad segment directory unit.`

The file-unit you specified was not a segment directory unit. This error code is not returned by currently used subroutines; it may be returned by old programs that use obsolete subroutine calls.

`E$SUNO (31)` — `Segment directory unit not open.`

An operation was attempted on a segment directory entry when the specified segment directory file-unit was not open, or was not open for the type of operation requested. The SRCH$$, SGDR$$, SGDOP, SGDEX, and SGD$DL subroutines may return this error code.

`E$NMLG (32)` — `Name is too long.`

A file system objectname is too long. For example, this error code is returned if a call to APSFX$ to append a suffix to the specified filename would result in a filename or a pathname longer than PRIMOS allows.

`E$SDER (33)` — `Segment directory error.`

SGDR$$ or SGD$OP return this error code when the segment directory member being opened is not a SAM or DAM file or a SAM or DAM segment directory. Contact your System Administrator or system operations staff to determine whether the situation can be corrected by file system maintenance.

`E$BUFD (34)` — `Directory is damaged.`

Integrity checking performed by many file system subroutines has detected an integrity error in the structure of a file directory. Contact your System Administrator or system operations staff to determine whether the situation can be corrected by file system maintenance.

`E$BFTS (35)` — `Buffer is too small.`

Either a caller-supplied buffer is too small to hold the data to be returned, or a buffer internal to the subroutine is too small to hold the data. In some cases, the error indicates that the requested operation could not be performed. In other cases, the operation may have been performed, but the data to be returned was truncated to fit into the caller-supplied buffer. Check the description of the subroutine you are calling to determine the appropriate error recovery.

`E$FITB (36)` — `File is too big.`

SGDR$$ returns this error code if the segment directory on which it is operating is discovered to be longer than 131,072 halfwords (65,536 entries).

`E$NULL (37)` — `(no message)`

This error code does not have any specific meaning attached to it. If specified in a call to ER$PRINT or ER$TEXT, this error code returns a null string. Many programs use E$NULL in calls to the obsolete subroutine ERRPR$, or to the ER$PRINT and ER$TEXT subroutines when the only error message desired is a user-specified error message.

`E$IREM (38)` — `Illegal remote reference.`

An operation was attempted that requires a reference to a remote node on the network. No PRIMOS support exists for such a reference. For example, this error code is returned when an attempt is made to spawn a phantom either while attached to a remote directory or while using a remote command file or CPL program.

`E$DVIU (39)` — `Device in use.`

An attempt was made to assign a peripheral device, such as a magnetic tape drive, that was already assigned to another user.

`E$RLDN (40)` — `Remote line is down.`

The system being referenced cannot be reached from the local system. No disks or other resources on that remote system can be accessed.

`E$FUIU (41)` — `File units all in use.`

The operation could not proceed because the system lacks either available file-units or available named semaphores.

No Available File-units: No more file-units are available for the calling process. This usually indicates that the program is not closing units it has finished using, since the number of available file-units is usually very large.

This error may also indicate that a remote system being used by the calling process has run out of file-units on which to handle this process's remote requests.

No Available Named Semaphores: No more semaphores are available on the system for access via the named-semaphore subroutines. Use the STATUS SEMAPHORES command to display information on both numbered and named semaphores. Typically, the SEM$OP subroutine returns this error code if it refers to the lack of availability of named semaphores.

`E$DNS (42)` — `Device not started.`

PRIMOS returns this error code if a low-level operation is requested on a device that is not started.

`E$TMUL (43)` — `Too many subdirectory levels.`

The Q$READ and Q$SET subroutines and programs that perform treewalks of subdirectories return this error code if the number of nested subdirectories exceeds the implementation-defined maximum.

`E$FBST (44)`

Not currently returned by PRIMOS.

`E$BSGN (45)` — `Bad segment number.`

An invalid (out-of-range) segment number was specified. For example, an attempt was made to set access on a segment (not a segment directory) with an invalid number via SEGAC$. This error code is also returned by the MM$MLPA and MM$MLPU subroutines.

`E$FIFC (46)` — `FAM - invalid function code.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$TMRU (47)`

Not currently returned by PRIMOS.

`E$NASS (48)` — `Device not assigned.`

An attempt was made to perform an operation on a peripheral device (such as a magnetic tape unit) that is not assigned to the user.

`E$BFSV (49)`

Not currently returned by PRIMOS.

`E$SEMO (50)` — `Semaphore overflow.`

SEM$NF returns this error code if the number of outstanding notifies on the semaphore is already 32,766.

`E$NTIM (51)` — `No timer.`

SEM$TN returns this error code if no timers are available to place on semaphores. Because of the potential lack of timers for numbered semaphores, you may wish to have your program use named semaphores and use the SEM$TW subroutine to wait for a specified amount of time.

`E$FABT (52)`

Not currently returned by PRIMOS.

`E$FONC (53)`

Not currently returned by PRIMOS.

`E$NPHA (54)` — `No phantoms available.`

An attempt to spawn a phantom (by calling PHNTM$ or PHANT$) failed because all phantoms are already in use.

`E$ROOM (55)` — `No room.`

More entries have been returned to a fixed-length table than the table has room for. Some subroutines return this error code after writing as many entries as possible into the table. This error code is also returned by storage allocation subroutines that do not signal conditions when they cannot find sufficient memory.

`E$WTPR (56)` — `Disk is write-protected.`

On a write-protected disk, you cannot open an object for writing, create an object, or change the attributes of an object.

`E$ITRE (57)` `Illegal treename.`

The pathname that was supplied to AT$, FIL$DL, SRSFX$, TSRC$$, or that is on a command line does not conform to the syntax rules for a pathname. See the *PRIMOS User's Guide* for a description of the syntax of a pathname.

`E$FAMU (58)`

Not currently returned by PRIMOS.

`E$TMUS (59)` `Too many users.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$NCOM (60)` `Null command line.`

The PRIMOS command environment listener uses this error code internally to distinguish a null command line from a successfully invoked command. It is not currently returned to the user.

`E$NFLT (61)` `No fault frame.`

CNSIG$ returns this error code to indicate that it could not find a condition frame in which to set the *continue_sw* bit to '1'b before it found the end of the stack. This error probably results from calling CNSIG$ outside of an on-unit.

`E$STKF (62)` `Bad stack format.`

PRIMOS subroutines, such as CNSIG$, use this error code to indicate that the stack seems to be circular. This may be due to a circular stack or a circular list of on-units. The stack is considered circular if approximately 20,000 stack frames have been examined without finding the desired frame or the end of the stack. The list of on-units for a particular stack frame is considered circular if approximately 1,000 on-units have been examined without finding the desired on-unit or the end of the list.

`E$STKS (63)` `Bad stack format signalling.`

The condition signaling mechanism generates this error code upon detection of a bad stack format when it calls the command environment reinitialization subroutine. The error code itself is not returned by any PRIMOS subroutine.

`E$NOON (64)` `No on-unit found.`

A spawned phantom encountered an error during startup that cannot be handled during startup. Or, a crawlout condition occurred while the process was in Ring 3, indicating a possible internal error or an error in a user program.

`E$CRWL (65)` `Fatal error in crawlout.`

An attempt was made to crawl out from one ring to another ring of equal or greater privilege, an invalid crawlout was attempted, or a new condition was signaled during a crawlout. In all cases, this error code is used only in the call to the subroutine that reinitializes the user's command environment, and is not returned by any PRIMOS subroutine to a calling program.

`E$CROV (66)` `Stack overflow in crawlout.`

Insufficient room exists on the Ring 3 stack to handle a crawlout from Ring 0 or Ring 1, or insufficient room exists due to a warm start following a system halt caused by a Ring 0 stack overflow by the user's process. This error code is used only in the call to the subroutine that reinitializes the user's command environment, and is not returned by any PRIMOS subroutine to a calling program.

`E$CRUN (67)` `Crawlout unwind failed.`

The stack could not be unwound during a crawlout. This error code is used only in the call to the subroutine that reinitializes the user's command environment, and is not returned by any PRIMOS subroutine to a calling program.

`E$CMND (68)` `Bad command format.`

The standard command processor (STD$CP or CP$) returns this error code if the command line is truncated because it is too long, if the command name does not conform to filename syntax rules, or if the command name is more than 32 characters long.

`E$RCHR (69)` `Reserved character.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$NEXP (70)` `Corruption detected during use of EXIT.`

PRIMOS has detected a stack frame that indicates the bottom of a static-mode program's stack when there is no known static-mode program suspended in the user's process. Such a situation is rarely encountered except in an errant program; it may be detected when a program calls the EXIT subroutine, in which case it causes the user's command environment to be reinitialized.

`E$BARG (71)` `Bad argument in command.`

An argument, such as a key or a pathname, is invalid, either because it is unrecognized or because it conflicts with other arguments. An unrecognized argument can occur if a required data area is not allocated. E$BARG is also used to indicate an invalid argument to a PRIMOS command.

`E$CSOV (72)` `Concealed stack overflow.`

PRIMOS has detected that the user's process has overflowed its Ring 0 concealed stack, which is an internal error. This error code is used only in the call to the subroutine that reinitializes the user's command environment, and is not returned by any PRIMOS subroutine to a calling program.

`E$NOSG (73)` `Segment does not exist.`

A reference was made to a nonexistent segment when calling a PRIMOS subroutine to manipulate a segment's access rights or when attempting to change the availability of the last page of a segment.

`E$TRCL (74)` `Command line truncated.`

Subroutines that read a command line or expand text using the abbreviation preprocessor return this error code to indicate that the command line or the expanded text was longer than could be held in the buffer, and was, therefore, truncated.

`E$NDMC (75)` `No SMLC DMC channels.`

No further DMC channels are available for synchronous communications lines.

`E$DNAV (76)` `Device not available.`

The requested peripheral device, such as a magnetic tape unit, is not available.

`E$DATT (77)` `Device already attached.`

The requested peripheral device is already attached to the user's process.

`E$BDAT (78)` `Bad output data.`

An incorrect data count or invalid data format exists. The SR$FR_LS subroutine returns this error code if it encounters an invalid pointer in a linked list. The MM$MLPA and MM$MLPU subroutines return this error code if you specify a page that cannot be operated on. E$BDAT is also returned by the LN$SET subroutine.

`E$BLEN (79)` `Bad length.`

The specified buffer length is invalid. The AS$LST and AS$SET subroutines return E$BLEN if the buffer length is not large enough.

`E$BDEV (80)` `Bad device number.`

An invalid number for a peripheral device, such as a communications device, was specified.

`E$QLEX (81)` `Queue length exceeded.`

An internal queue cannot hold another item.

`E$NBUF (82)` `No buffer space.`

An attempt to acquire internal buffer space failed.

`E$INWT (83)` `Input waiting.`

Pending input must be read before output can be sent to the peripheral device.

`E$NINP (84)` — `No input available.`

No input from the peripheral device is pending.

`E$DFD (85)` — `Device forcibly detached.`

The peripheral device was forcibly detached from the user's process; therefore, the desired operation cannot be performed.

`E$DNC (86)` — `DPTX not configured.`

An attempt was made to operate a peripheral device that requires DPTX to be configured on the system.

`E$SICM (87)` — `Illegal 3270 command.`

An attempt to use an invalid 3270-class command code was made.

`E$SBCF (88)` — `Bad device number copied.`

An invalid device number was copied during an output operation to a 3270-class device.

`E$VKBL (89)`

Not currently returned by PRIMOS.

`E$VIA (90)` — `Invalid AID byte.`

An invalid or nonexistent AID byte was supplied in the buffer for a 3270-class device.

`E$VICA (91)` — `Invalid cursor address.`

A cursor address in a cursor-addressing command is invalid or missing.

`E$VIF (92)` — `Invalid field address.`

A field address in a field-addressing command is invalid or missing.

`E$VFR (93)` — `Field required.`

An invalid field address was supplied for a formatted screen.

`E$VFP (94)` — `Field prohibited.`

A Set Buffer Address (SBA) command was performed in an unformatted buffer for a 3270-class device.

`E$VPFC (95)` — `Protected field check.`

An attempt was made to write into a protected field on the screen.

`E$VNFC (96)`

Not currently returned by PRIMOS.

`E$VPEF (97)` `Past end of field.`

An attempt was made to write past the end of a field on the screen.

`E$VIRC (98)`

Not currently returned by PRIMOS.

`E$IVCM (99)` `Magtape command invalid.`

PRIMOS returns this error code if an invalid magnetic tape operation is requested.

`E$DNCT (100)` `Device not connected.`

An operation was attempted on a peripheral device that was not connected to the system or to the user's process.

`E$BNWD (101)` `Bad number of words.`

An invalid number of halfwords was specified as the size of the buffer.

`E$SGIU (102)` `Segment in use.`

An attempt was made to copy a segment to another segment that already exists. (This refers to memory segments, not to segment directories or their members.)

`E$NESG (103)` `Not enough segments.`

Insufficient system segments are available for a program to be invoked or for additional storage to be acquired.

`E$SDUP (104)`

Not currently returned by PRIMOS.

`E$IVWN (105)` `Invalid VMFA window number.`

An EPF was corrupted, because it contains invalid VMFA window numbers. Rebuild the EPF by using BIND.

`E$WAIN (106)` `Window already in address space.`

PRIMOS uses this error code internally when mapping an EPF to memory to indicate that the EPF was already mapped to memory for this process. E$WAIN is not currently returned to the user.

`E$NMVS (107)` — `No more VMFA segments.`

Insufficient VMFA segments are available in the system to map in an EPF. Ask your System Administrator to increase, if possible, the number of segments available to your process. Meanwhile, removing inactive EPFs from memory may temporarily alleviate the problem.

`E$NMTS (108)` — `No more temporary segments.`

Insufficient temporary segments are available in the system to map in the impure procedure code of an EPF (or the pure procedure code of a remote EPF or an EPF being debugged with DBG). Ask your System Administrator to adjust (via NSEG) the number of temporary segments on your system. Meanwhile, removing inactive EPFs from memory may temporarily alleviate the problem.

`E$NDAM (109)` — `Not a DAM file.`

An attempt was made to open a file for VMFA-read (via the k$vmr key) when the file is not a DAM file.

`E$NOVA (110)` — `Not open for VMFA.`

The file-unit number supplied to EPF$RUN or EPF$MAP does not identify a unit open for VMFA-read (via the k$vmr key). See Volume III of this series for information on how to call EPF$RUN or EPF$MAP.

`E$NECS (111)`

Not currently returned by PRIMOS.

`E$NRCV (112)` — `Receive enabled required.`

SMSG$ is not allowing you to send a message because you are rejecting messages of the same type (immediate or deferred) that you are sending to another user.

`E$UNRV (113)` — `User not receiving now.`

The user to whom you are sending a message via SMSG$ is rejecting immediate (and possibly also deferred) messages.

`E$UBSY (114)` — `User busy, please wait.`

SMSG$ was unable to send a message to a user, either because the receiver already had a deferred message waiting to be displayed, or because the receiver's terminal output buffer was full and, therefore, an immediate message could not be sent.

`E$UDEF (115)` — `User unable to receive messages.`

The user number specified in a call to SMSG$ identifies a user who is not logged in to the system, but who is logged in either remotely to another system on the network or through the system from one node to another.

`E$UADR (116)` — `Unknown addressee.`

The user number specified in a call to SMSG$ does not correspond to a logged-in user or the user name specified could not be found in the list of logged-in users on the system.

`E$PRTL (117)` — `Message operation partially blocked.`

Not all of the users who were the target of a message sent by SMSG$ received the message (perhaps because they are deferring or rejecting messages).

`E$NSUC (118)` — `Operation unsuccessful.`

When returned by the inter-user message facility (the SMSG$ subroutine), this error code indicates that the message reached none of the potential recipients. When returned by the storage allocation subroutines (STR$FS, for example), this error code indicates a corrupted memory allocation structure. Also returned by IOCS$_GET_LOGICAL_UNIT. This error code is used as a generic positive severity code with a message slightly more meaningful than that displayed for E$EOF and E$NULL.

`E$NROB (119)`

Not currently returned by PRIMOS.

`E$NETE (120)` — `Network error detected.`

A problem occurred with a remote file access. Retry the operation. If this is not successful, close all file-units on the remote system and attach to a directory on a different system before retrying the remote access.

`E$SHDN (121)` — `Disk has been shut down.`

The disk on which the file system object resides was shut down. The disk is not available for use until the system operator has reenabled use of the disk.

`E$UNOD (122)` — `Unknown node name.`

A subroutine that takes a node name has found that the named node does not exist.

`E$NDAT (123)` — `No data found.`

No data was found. For example, a call to LON$R to read phantom logout information returns this error code if there is no additional record of any phantom logout. A call to LN$SET returns this error code if the EPF contains no library information.

`E$ENQD (124)` — `Enqueued only.`

A cross-process signaling message has been enqueued, but the user has not yet received the corresponding signal. This may be due to a low or idle user priority level or the message may have been deferred.

`E$PHNA (125)` `Protocol handler not available.`

The desired communications protocol handler is not available.

`E$IWST (126)` `E$INWT enabled by configuration.`

An attempt to set attributes for a device failed because input was waiting, and the configuration file specified inhibition of this operation when input is waiting.

`E$BKFP (127)` `Bad key for this protocol.`

An invalid key was supplied either in a call involving a communications device or when validating a system parameter.

`E$BPRH (128)` `Bad protocol handler specified.`

An internal error in DPTCFG occurred.

`E$ABTI (129)` `I/O abort in progress.`

An I/O abort was occurring during an attempt to output data or set attributes for a communications device.

`E$ILFF (130)` `Illegal DPTX file format.`

An invalid file format for the configuration file read during DPTX initialization exists.

`E$TMED (131)` `Too many emulate devices.`

DPTX did not initialize because there are too many devices to emulate.

`E$DANC (132)` `DPTX already configured.`

An attempt was made to configure DPTX after it was already configured.

`E$NENB (133)` `Remote node not enabled.`

A remote operation cannot be performed because the remote node is not allowing remote file access.

`E$NSLA (134)` `No NPX slaves available.`

The remote system on which the file system object resides has become overloaded with remote file access requests. The operation may be attempted later, with possible success.

`E$PNTF (135)` `Procedure not found.`

The LINKAGE_FAULT$ condition was raised in the slave process on the remote system while attempting to access a remote file system object.

`E$SVAL (136)` `Slave validation error.`

The user's remote ID for the system on which the file system object resides is incorrect. The user must use the ADD_REMOTE_ID command, described in the *PRIMOS Commands Reference Guide*, to establish the correct remote ID for the system. Until then, all attempts to access data on that remote system will fail with this error code.

`E$IEDI (137)` `I/O error or device interrupt.`

An error or interrupt occurred on a peripheral device on which low-level operations are being performed by the user program.

`E$WMST (138)` `Warm start occurred.`

A peripheral device should be reinitialized because a warm start was performed on that system.

`E$DNSK (139)` `PIO instruction did not skip.`

A Programmed I/O instruction to a peripheral device did not skip during a low-level operation being performed by a user program.

`E$RSNU (140)` `Remote system not up.`

The remote system on which the file system object resides is in the process of starting up, but is not yet honoring Remote File Access (RFA) requests because the operator has not yet set the date and time at the supervisor terminal for that system.

`E$S18E (141)`

Not currently returned by PRIMOS.

`E$NFQB (142)` `No free quota blocks.`

Internal storage used to keep track of quota information for directories was exhausted.

`E$MXQB (143)` `Maximum quota exceeded.`

The operation requires an additional record to be allocated in a directory, but the maximum quota on that directory or on one of its parent directories was already reached.

Some (but not all) operations are nonrecoverable after returning this error code. For example, the WTLIN$ subroutine does not restore the file location pointer to the original location when it encounters this error; the file location is undefined. Other operations, such as the PRWF$$ subroutine, reset the file location pointer to the value it held before the quota-exceeded error was encountered.

`E$NOQD (144)` `Not a quota disk.`

An attempt was made to perform a quota operation on a nonquota (pre-Rev. 19 format) disk. The DIRCR, QREAD, and Q$SET subroutines may all return this error code.

`E$QEXC (145)` `Quota set below current usage.`

A call to Q$SET set the maximum quota to a value that is below the number of records currently used in the directory. Although this is not an error, it does mean that no new records can be used in the directory until enough records are deleted so that the number of records used falls below the maximum quota.

`E$IMFD (146)` `Operation illegal on MFD.`

An operation was attempted that is invalid on the MFD for a disk partition.

`E$NACL (147)` `Not an ACL directory.`

An attempt to set or list ACL information was made for a file system object that resides in a password directory.

`E$PNAC (148)` `Parent not an ACL directory.`

An attempt to set or list ACL information was made for a file system object whose parent directory is a password directory rather than an ACL directory.

`E$NTFD (149)` `Not a file or directory.`

The target object of a call to AC$CAT, AC$DFT, or KLM$IF is not a file, a segment directory, or a file directory. You cannot protect an access category with another access category, nor can you set an access category to default protection.

`E$IACL (150)` `Operation illegal on access category.`

An attempt was made to open, close, delete, or set improper attributes on an access category. Use AC$LST to read an access category. Use CAT$DL to delete an access category. The only proper attributes to set on an access category are date/time attributes such as *date/time last modified.*

`E$NCAT (151)` `Not an access category.`

The file system object is not an access category. The AC$CAT, CAT$DL, and DIR$CR subroutines are all capable of returning this error code.

`E$LRNA (152)` `Like reference not accessible.`

AC$LIK cannot access the like reference object due to insufficient access.

`E$CPMF (153)` `Category protects MFD.`

An attempt was made to call CAT$DL to delete an access category that protects the MFD of a partition.

`E$ACBG (154)` `ACL too big.`

An attempt was made to specify more access control information than can fit in a directory entry. See Volume II of this series for a description of the limits on access control lists.

`E$ACNF (155)` `Access category not found.`

The access category referenced in a call to AC$CAT or DIR$CR could not be found. A common cause for this error is the lack of the .ACAT suffix in the call. None of the PRIMOS access control subroutines add this suffix to a filename. Therefore, your program should call APSFX$ to ensure addition of the suffix.

`E$LRNF (156)` `Like reference not found.`

The AC$LIK subroutine could not find the like reference. See Volume II of this series for details on setting access on one object to be like that of another object. A common cause for this error is the false assumption that supplying a simple pathname causes the like reference to be searched for in the target object's directory. In fact, it is searched for in the user's home directory.

`E$BACL (157)` `Bad access control list format.`

An invalid access control list was supplied to the AC$SET or AC$CHG subroutine. See Volume II of this series for detailed information on the syntax for access control lists.

`E$BVER (158)` `Bad version number.`

A version number supplied by the calling program in a structure or in the calling sequence is unrecognized or no longer supported. If this error occurs in an EPF file, it may be correctable by resubmitting the file to BIND.

`E$NINF (159)` `No information is accessible.`

An error occurred while you were attempting to access a file system object in a directory to which you have no List access. To prevent the determination of objectnames in the directory by inference or by the process of elimination, PRIMOS does not report the original error to the calling program or to the user. E$NINF is also returned by the LN$SET and DS$AVL subroutines.

`E$CATF (160)` `Access category found in directory.`

The AC$RVT subroutine cannot revert a directory (that is, change it from an ACL directory to a password directory), because the directory still contains access categories.

`E$ADRF (161)` `ACL subdirectory found in directory.`

AC$RVT returns this error code to indicate that the directory to be reverted (changed from an ACL directory to a password directory) still contains ACL subdirectories that must themselves be reverted before their parent directory can be reverted.

`E$NVAL (162)` `Validation error.`

CHG$PW returns this error code if the user's entry could not be found in the EDIT_PROFILE database (perhaps indicating that the user's entry was deleted since the user logged in).

`E$LOGO (163)` `(no message)`

PRIMOS uses this error code for internal communication when calling the subroutine that reinitializes a user's command environment to indicate that the user is logging out. Neither the error code nor the accompanying null message is ever returned to a user program or displayed on a user's terminal.

`E$NUTP (164)` `No unit table available for phantom.`

All unit tables are taken.

`E$UTAR (165)` `Unit table already returned.`

An internal PRIMOS error occurred when logging out a user.

`E$UNIU (166)` `Unit table not in use.`

A unit table that was not being used is being returned to the system.

`E$NFUT (167)` `No unit table available.`

No unit tables are available.

`E$UAHU (168)` `User already has unit table.`

An internal PRIMOS error occurred when logging in a user.

`E$PANF (169)` `Priority ACL not found.`

PA$LST returns this error code to indicate that no priority ACL was placed on the disk partition specified.

`E$MISA (170)` `Command line argument missing.`

A required argument was not specified on the command line. User-written programs may use this error code for similar purposes.

`E$SCCM (171)` `System console command only.`

The desired operation can be performed only by a program running at the supervisor terminal (User 1).

`E$BRPA (172)`

Not currently returned by PRIMOS.

`E$DTNS (173)` `Date and time not set.`

DIR$CR and Q$SET subroutines return this error code to indicate that proper disk-quota operations cannot be performed unless the system date and time are set.

`E$SPND (174)` `Remote procedure call still pending.`

A call to a remote system has not completed within a reasonable amount of time. This error code indicates a non-recoverable network error.

`E$BCFG (175)` `Network configuration mismatch.`

The remote system on which the file system object resides does not agree with the network configuration of the local system or the remote system requires a remote ID. Use the ARID command to establish a remote ID. If the problem persists, contact your Network Administrator for assistance.

`E$BMOD (176)` `Bad access mode.`

The AC$ subroutines return this error code if the access mode is not ALL, NONE, or one or more of the letters A, D, L, O, P, R, U, W, or X. See Volume II of this series and the *PRIMOS User's Guide* for detailed information on the syntax rules for access control lists.

`E$BID (177)` `Bad user identifier.`

AC$SET, VALID$, or CHG$SA return this error code to indicate an invalid identifier or user name. AC$SET may also return this error code if two specifications of $REST occur in the access control list. See Volume II of this series and the *PRIMOS User's Guide* for detailed information on the syntax of an access control list.

`E$ST19 (178)` `Operation illegal on pre-19 disk.`

An attempt was made to use file system features that are not available for files on the specified disk. This is usually because the disk was formatted using an earlier revision of PRIMOS that did not support these features.

`E$CTPR (179)` `Object is category-protected.`

AC$CHG returns this error code when an attempt is made to change the access of an object that is protected by an access category. See Volume II of this series for information on how your program can handle this situation. See the *PRIMOS User's Guide* for detailed information on the rules governing access control lists.

`E$DFPR (180)` `Object is default-protected.`

AC$CHG returns this error code when an attempt is made to change the access of an object that is default-protected.

If you wish your program to force the change anyway, have it call AC$LIK with the target object as both the target and reference objects; that is, set a specific ACL to match the existing ACL. Then, call AC$CHG to change the specific ACL on the target object.

`E$DLPR (181)` `File is delete-protected.`

FIL$DL or SRCH$$ return this error code when an attempt is made to delete a file that was delete-protected by SATR$$ (via the SET_DELETE command).

`E$BLUE (182)`

Not currently returned by PRIMOS.

`E$NDFD (183)`

Not currently returned by PRIMOS.

`E$WFT (184)` `Wrong file type.`

The file specified is of the wrong type. For example, this error code is returned by CF$EXT, CF$REM, or CF$SME if you specify a file that is not a CAM file.

`E$FDMM (185)` `Format/data mismatch.`

This error code is returned by the LIST$CMD subroutine if you specify an invalid wildcard string.

`E$FER (186)` `Bad format.`

This error code is returned by the ISNL, ISNRC, and ISN$UC subroutines if the file accessed is not formatted as a High Level Name File (HLNF).

`E$BDV (187)`

Not currently returned by PRIMOS.

`E$BFOV (188)`

Not currently returned by PRIMOS.

`E$NFAS (189)` `Top-level directory not found or inaccessible.`

The first directory name supplied in the pathname could not be located on any of the disks that are active and visible to the calling system. This error can also occur if the named directory does actually exist on one or more disks, but the user does not have List access to any of them.

`E$APND (190)` `Asynchronous procedure still pending.`

An attempt to initiate a new asynchronous remote procedure failed because there is a previous asynchronous procedure call to that remote node. Terminate the previous asynchronous procedure call and retry the operation.

`E$BVCC (191)` `Bad virtual circuit clearing.`

An error was made in clearing a virtual circuit when the user was terminating file access to a remote node. It does not indicate an error in the user program; it most likely indicates that a network problem occurred prior to the termination of the connection.

`E$RESF (192)` `Restricted access file.`

An attempt was made to access a file that is restricted to access by only a particular subsystem (such as ROAM).

`E$MNPX (193)` `Illegal multiple hops in NPX.`

A disk partition residing on a remote node is listed on that remote node as residing on yet another remote node, requiring a second remote access, which is not allowed. Ask your System Administrator to modify the system startup file appropriately.

`E$SYNT (194)`

Not currently returned by PRIMOS.

`E$USTR (195)` `Unterminated string.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$WNS (196)`

Not currently returned by PRIMOS.

`E$IREQ (197)`

Not currently returned by PRIMOS.

`E$VNG (198)`

Not currently returned by PRIMOS.

`E$SOR (199)`

Not currently returned by PRIMOS.

`E$TMVV (200)`

Not currently returned by PRIMOS.

`E$ESV (201)`

Not currently returned by PRIMOS.

`E$VABS (202)`

Not currently returned by PRIMOS.

`E$BCLC (203)` `Bad compiler library call.`

The compiler generated an invalid call to one of its runtime library routines. For example, the first argument to most of the I/O routines is a key that indicates which optional arguments have or have not been specified. If the compiler sets the key to indicate that a particular argument will be passed, but the compiler does not pass that argument, the error E$BCLC is raised. Contact your System Administrator for assistance.

`E$NSB (204)` `BRMS-labeled tape was detected.`

A non-BRMS product has tried to read a BRMS-labeled tape.

`E$WSLV (205)` `Slave ID mismatch.`

One of the nodes involved in your network connection has had the network restarted since you last used this remote file access connection. Attach to a directory on a different system, then reestablish attach points and retry the RFA operation.

`E$VCGC (206)` `Virtual circuit was cleared.`

The virtual circuit used for RFA access to a particular node was cleared by PRIMENET. Close all units open to that node and issue the ORIGIN command to reset the condition, then reestablish attach points and open files on the remote node as desired.

`E$MSLV (207)` `Maximum slaves per user exceeded.`

The maximum number of remote file accesses to remote systems per user has been reached and no new RFAs to other remote systems are allowed.

`E$IDNF (208)` `Slave ID number not found.`

Internal RFA error.

`E$NACC (209)` `Not accessible.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$UDMA (210)` `Not enough DMA channels.`

There are too few DMA channels during a low-level operation on a peripheral device.

`E$UDMC (211)` `Not enough DMC channels.`

There are too few DMC channels during a low-level operation on a peripheral device.

`E$BLEF (212)`

Not currently returned by PRIMOS.

`E$BLET (213)` `Bad tape record length and EOT.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$ALSZ (214)` `Allocation request too small.`

A call to STR$AL to allocate memory specified too few halfwords to allocate. You must allocate a minimum of four halfwords.

`E$FRER (215)` `Free request with invalid pointer.`

A call to STR$FR or STR$FS was made with an invalid pointer. An invalid pointer is a pointer to an area of memory already freed, or to a location other than the beginning of an allocated or freed area.

`E$HPER (216)` `User storage heap is corrupted.`

The heap storage for program-class storage was corrupted. Issue the ICE command to reset the condition.

Alternatively, if you believe the program you were running caused the problem, issue the DUMP_STACK command to trace the program's history; then issue the ICE command to reinitialize your command environment. (Errant user programs can corrupt program-class and process-class storage.)

`E$EPFT (217)` `EPF type invalid.`

The EPF type is not valid for this revision of PRIMOS. The EPF$MAP subroutine is typically the subroutine that returns this error code, although other EPF-related subroutines also may return this error code. Resubmit the file to BIND. See Volume III of this series for more information.

`E$EPFS (218)`

Not currently returned by PRIMOS.

`E$ILTD (219)` `EPF LTD linkage descriptor invalid.`

An invalid LTD linkage descriptor type was found in an EPF file. The EPF file is corrupted or an internal error occurred in BIND. Resubmit the file to BIND.

`E$ILTE (220)` `EPF LTE linkage descriptor invalid.`

An invalid LTE linkage descriptor type was found in an EPF file. The EPF file is corrupted or an internal error occurred in BIND. Resubmit the file to BIND.

`E$ECEB (221)` `Command environment breadth exceeded.`

An attempt was made to invoke CP$, EPF$RUN, or EPF$INVK when the maximum command environment breadth (as displayed by LIST_LIMITS) was already reached by the running program. Use the RD$CE_DP subroutine to determine the current command environment breadth and use the CE$BRD subroutine to determine the maximum command environment breadth within your program.

`E$EPFL (222)` `EPF file exceeds file size limit.`

The EPF is too large for the EPF$MAP or EPF$RUN subroutine to handle. Consider breaking up the program or library into separate program and library EPFs, if possible.

`E$NTA (223)` `EPF file not active for this user.`

REMEPF$ and internal PRIMOS subroutines use this error code to indicate that an attempt was made to remove from memory an EPF that was not mapped to memory for this user.

`E$SWPS (224)`

Not currently returned by PRIMOS.

`E$SWPR (225)` `EPF file suspended within this process.`

The EPF being removed (by EPFDEL, EPFRUN, or REMEPF$) is suspended (active) within the user's process. Removal of the EPF from memory is not allowed in this case.

`E$ADCM (226)` `System Administrator command only.`

A user other than the System Administrator attempted to set system defaults for command environment limits.

`E$UAFU (227)` `Unable to allocate file-unit.`

PRIMOS was unable to allocate a file-unit entry for a user because insufficient system-class storage was available.

`E$FIDC (228)` `File inconsistent data count.`

Either a corrupted disk or a problem with the file system exists. This error is returned during the truncation of a SAM file if the data count for the last record of the file implies that the current position of the unit in the file is beyond the end-of-file mark. Contact your System Administrator or system operations staff to determine whether the situation can be corrected.

`E$INDL (229)` `Insufficient DAM file index levels.`

A DAM file has an insufficient number of index record levels for its size. This error may be returned during the truncation of a DAM file by PRWF$$ or during the deletion of a DAM file. Contact your System Administrator or system operations staff to determine whether the situation can be corrected.

`E$PEOF (230)` `Past end of file.`

Either a corrupted disk or a problem with the file system exists. This error is returned during the truncation of a DAM file if the data count for the last record of the file implies that the current position of the unit in the file is beyond the end-of-file mark. Contact your System Administrator or system operations staff to determine whether the situation can be corrected.

`E$EXMF (231)` Extent map full.

The extent map of a Contiguous Access Method (CAM) file is full. The file cannot be extended because no additional extents can be added to the extent map.

`E$BKIO (232)` Unit open for block mode I/O.

The file-unit is open for block mode I/O. Operations requiring locate mode cannot be performed.

`E$AWER (233)` Asynchronous write error.

An error occurred during an asynchronous writing action.

`E$RAMC (234)` ROAM access mode conflict.

A ROAM error, not a file system error, exists.

`E$RIER (235)` ROAM internal error.

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$NSLV (236)` Process not a slave.

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$RSIN (237)`

Not currently returned by PRIMOS.

`E$ATNS (238)` Attribute not supported in directory.

The target object does not have the date/time created (DTC) and date/time last accessed (DTA) attribute fields. These attribute fields are not present because the object is not an entry in a hashed directory. Attempts to set these attribute fields return this error code.

`E$RSHD (239)` Remote disk has been shut down.

A file system operation cannot be performed because it would take place on a remote disk that was shut down from the supervisor terminal on the local system. No further accesses to the disk are permitted from the local system. Accesses to the disk from other nodes on the network, including the system on which the disk resides, may still be permitted.

`E$NOPD (240)` No paging device defined.

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$NRFC (241)`

Not currently returned by PRIMOS.

`E$CPOV (242)` `Overflow of CPU seconds.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$IOOV (243)` `Overflow of I/O seconds.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$BHOV (244)` `Overflow of CPU and I/O seconds.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$AELE (245)` `Library is non-executable.`

You tried to invoke a library EPF as a program EPF. If you want an EPF to function as both a program and a library EPF, you must use the MAIN subcommand of BIND to tell BIND what to use as a starting address.

`E$LIST (246)` `Search list not found or invalid.`

A search rule subroutine specified the name of a search list that is not currently set for the user's process. This error code is also returned if you attempt to create a search list with an illegal search list name. Use the LIST_SEARCH_RULES command to determine which search lists are set for your process. Search list names are not case sensitive.

`E$RULE (247)` `Search rule not found or invalid.`

A search rule subroutine specified a search rule that PRIMOS cannot find in the specified search list. Sometimes this error code is issued because the search rule in the list and the one specified in your subroutine differ in case. Use the LIST_SEARCH_RULES command to list the rules in your search lists.

`E$NTOP (248)` `Search rule not an optional rule.`

You attempted to enable or disable a search rule that is not an optional search rule. You can use the SR$READ subroutine to determine if a search rule is optional.

`E$NEST (249)` `Search lists nested too deeply.`

You attempted to set a search list using a search rules file (template file) that contains **-insert** keywords that result in either of the following conditions. Either these **-insert** keywords would result in the nested insertion of template files in excess of 100 levels, or the **-insert** keywords would result in a circular reference, such as two files that attempt to include each other.

`E$ADMN (250)` `Administrator rules not modifiable.`

You attempted to delete or modify an administrator rule in a search list, or you attempted to insert a search rule before an administrator rule.

`E$EOL (251)` `End of search list.`

You attempted to read past the end of a search list.

`E$ADRL (252)` `Administrator rules contain error.`

You attempted to create an illegal administrator rule.

`E$IFCB (253)` `Insufficient free contiguous blocks.`

Not enough contiguous disk blocks are available to extend the CAM file. (When CAM files are extended, they are extended more than one record at a time.)

`E$IMEM (254)` `Insufficient memory for extent map.`

The user does not have enough dynamic memory to read in the CAM file's extent map. The extent map, which contains the physical location of the extents on the disk, is read into memory when it is opened.

`E$NRES (255)` `No resources available for request.`

A system process was not available for use or not enough memory was available to carry out the request.

`E$ILUS (256)` `Illegal use of PRIMIX gate.`

The user called a gate reserved for PRIMIX. This error may be returned when the user is not currently in PRIMIX or when that user's PRIMIX state data was corrupted.

`E$NCHD (257)` `No child found for this process.`

A process attempted to wait for the termination of a child when the process has no children. The PX$WAITP subroutine returns this error code.

`E$INT (258)` `PRIMIX wait terminated by interrupt.`

A process was taken off a PRIMIX wait by an interrupt. This error code is returned by PX$WAITP and PX$PAUSP.

`E$XSHD (259)` `PRIMIX can not be initialized when running.`

The user attempted to start PRIMIX when PRIMIX is already active. This error code is returned by PX$INIT, called through the START_PRIMIX command.

`E$NOPX (260)` `PRIMIX can not be shut down when not running.`

The user attempted to stop PRIMIX when PRIMIX is not currently active. This error code is returned by PX$SHDN, called through the STOP_PRIMIX command.

`E$NOUS (261)` `PRIMIX process table has no users.`

The PRIMIX process table is empty when at least one entry for the caller should have been found. This error code indicates a serious problem with the PRIMIX process data structure.

`E$INCO (262)` `PRIMIX process table returned is incomplete.`

PX$DUMP (the subroutine that returns the PRIMIX process table to the caller) ran out of dynamic memory so that only a partial listing of the table was returned.

`E$IREQ (263)` `Illegal EPF registration.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$INAI (264)` `Invalid number of initialization arguments.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$ILLN (265)` `Illegal link at EPF registration.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$BUID (266)` `Bad user ID.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$INRE (267)` `Invalid request.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$NPSG (268)` `Not enough per-user DTAR1 segments.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$UINF (269)` `User ID not found.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$IVPT (270)` `Invalid block pointer.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$SNAL (271)` `Segment not allocated.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$NATF (272)` `Not able to free storage.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$ND3S (273)` `No DTAR3 segments available.`

Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$BSMT (274)` — `Null smt_ptr or bad field within SMT.`
Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$IALN (275)` — `Illegal alias name.`
Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$BPTR (276)` — `Bad pointer within SMT.`
Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$IDBT (277)` — `Illegal database.`
Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$BDTR (278)` — `Bad DTAR.`
Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$LUNR (279)` — `Library unregistered.`
Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$ENRG (280)` — `EPF has not been registered.`
Returned only for PRIMIX users at Rev. 21.0 and subsequent revisions.

`E$NDRB (281)` — `No directory block for unit.`
PRIMOS uses this error code internally. It is not currently returned to the user.

`E$CQPT (282)` — `Circular quota parent thread.`
PRIMOS uses this error code internally. It is not currently returned to the user.

`E$AREA (283)` — `Corrupted area encountered.`
A space allocation routine found an error in internal consistency.

`E$NOWN (284)` — `Not owner of resource.`
You attempted to return space that you do not own.

`E$BLOK (285)` — `Bad block encountered.`
A space allocation routine found an error in internal consistency.

`E$ISMR (286)` — `Invalid static mode resume.`

The command processor uses E$ISMR to indicate that the module INVKSM was told to restore a file that is not a valid static-mode program image. The CP$ subroutine returns E$ISMR if a program called CP$ exists to resume or restore an invalid image. E$ISMR is displayed as an error message when the user has attempted to restore or resume an invalid image from command level.

`E$BLIN (287)` — `Bad line number.`

A line number out of the legal range is passed to the gate.

`E$BBUF (288)` — `Bad buffer number.`

A buffer number out of the legal range is passed to the gate.

`E$BPRO (289)` — `Bad protocol.`

A protocol index out of the legal range is passed to the gate.

`E$LNUS (290)` — `Line in use.`

A line type is being changed on a line that is already assigned to a user.

`E$BFUS (291)` — `Buffer in use.`

The specified buffer number is already being used by another user.

`E$IRBF (292)` — `Invalid use of remote buffer.`

A buffer number in the remote buffer range is specified for a local asynchronous line.

`E$IABF (293)` — `Invalid use of assign line buffer.`

A buffer number in the assignable line buffer range is specified for a terminal user line.

`E$IASD (294)` — `Invalid ASD use.`

An attempt was made to enable ASD on an assignable line or an NTS line.

`E$IASP (295)` — `Invalid sample speed for ASD.`

The passed line speed used as the ASD sample speed is invalid.

`E$ILOD (296)` — `Invalid use of DISLOG.`

You tried to enable DISLOG on an assignable, remote, or NTS line.

`E$NSNI (297)` — `NSS database not initialized.`

An attempt was made to access the Node Status database that was created.

`E$NSNC (298)` — `Node/LAN naming conflict.`

An attempt to add a node to the Node Status database failed because of a naming conflict. The node name conficts with an existing LAN name, host name, or LTS name in the database.

`E$NSAC (299)` — `Node/MAC address conflict.`

An attempt to add a node to the Node Status database failed because of a MAC address conflict. The node has a MAC address that conflicts with an existing MAC address in the database.

`E$NTHN (300)` — `NTS host not configured.`

NTS was started for a host that was not configured for that NTS in the NTS configuration file.

`E$NTNS (301)` — `NTS not started.`

NTS was not started and an operation requiring NTS was attempted.

`E$NTST (302)` — `NTS already started.`

NTS was started and an operation requiring that NTS not be started was attempted.

`E$NTCF (303)` — `Not an NTS configuration file.`

NTS was started with other than an NTS configuration file (for example, a PRIMENET or an SNA configuration file). This error code is also returned when the NTS config subfile 0 cannot be opened, the NTS config file version number is not current, or the NTS config file checksum is not accurate.

`E$NTLC (304)` — `LHC not configured.`

An LHC is either not present or was not configured with an LHC directive, but that LHC was specified in the NTS configuration file or the PRIMENET configuration file.

`E$NTIN (305)` — `NTS database not initialized.`

An operation that requires access to the NTS database was attempted, but the NTS database is not initialized.

`E$NTDL (306)` — `LHC not downline loaded.`

An attempt was made to start PRIMENET/LAN300 or NTS on an LHC that is either broken or was not downline loaded.

`E$PLAA (307)` — `NTS line already associated.`

An attempt was made to associate an NTS line that was already associated.

`E$LLAA (308)` — `LTS line already associated.`

An attempt was made to associate an LTS line that was already associated.

`E$NASO (309)` — `Line not associated.`

An attempt was made to unassociate an NTS line in PRIMOS or an LTS line that is not currently associated.

`E$NCFG (310)` — `Line not configured.`

An attempt was made to start NTS, but no NTS lines are configured.

`E$NXCB (311)` — `XCB unavailable for request.`

An operation requires a buffer to be sent to an LHC, but no control blocks are available.

`E$DOQF (312)` — `Device output queue full.`

An operation requires a buffer to be sent to an LHC, but the output queue is full.

`E$LNOC (313)` — `Line not connected.`

A request for a connection between an LTS line and PRIMOS has been rejected. This can occur when another connection is pending, when a disconnection is pending, or when the line is not connectable.

`E$RQF (314)` — `Request queue full.`

The internal request queue to the NTS_SERVER is full.

`E$CREJ (315)` — `Connection request rejected.`

An LTS line rejected a connect request from PRIMOS. This occurs when the line is already connected.

`E$CTMO (316)` `Connection request timed out.`

An LTS line did not respond to a connect request (assignment) from PRIMOS. This occurs when the LTS is not present or is not currently operational.

`E$LHDN (317)` — `LHC down.`

An operation that requires an LHC to not be in the "down" state was requested when the LHC is down.

`E$LTDN (318)` — `LTS down.`

An operation that requires an LTS to not be in the "down" state was requested when the LTS is down.

`E$NTSH (319)` — `NTS is shut down.`

An operation was attempted during an NTS shutdown.

`E$QFUL (320)` — `Queue is full.`

The controller queue is full.

`E$QEMP (321)` — `Queue is empty.`

The controller queue is empty.

`E$NOQ (322)` — `Queue not found.`

An operation was requested on a queue that does not exist. This queue does not exist because it was not created by IG$FIND.

`E$VAL (323)` — `Validation error.`

This error code is returned when a process request is rejected. A request is rejected if your process does not have the proper access rights, if your process does not own the connection, or if the logical connection ID has been corrupted.

`E$COMM (324)` — `Command illegal.`

This error code is returned if you specify a command for a routine that cannot accept commands. A command is a fifteen-bit standalone quantity. Some data transfer routines, such as IG$ENQ and IG$DEQ, accept either buffers or commands. Other routines, such as IG$ABUF and IG$RBUF, have arguments formatted to accept commands, but cannot take commands. Specifying a command for these routines returns E$COMM.

`E$AWIR (325)` — `Page is already wired.`

You tried to wire a page that is already wired.

`E$IWIR (326)` — `Page is not wired.`

You tried to unwire a page that is not wired.

`E$NPDA (327)` — `No password directories allowed.`

Password directories are disabled on the system and you tried to either create a password directory or revert an ACL directory to a password directory.

`E$NINT (328)` — `Spooler subsystem not initialized.`

You tried to spool a file before the Spooler subsystem was initialized. The system operator must initialize the Spooler, using the PROP –COLDSTART command, before users can access the Spooler.

`E$REIU (329)` — `Registered EPF is in use.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$NBA (330)` — `No buffers available.`

No buffers are available to make a line a terminal line.

`E$LNOW (331)` — `Line not owned by you.`

You tried to change line characteristics for a line other than your terminal line or a line assigned to you.

`E$LNP (332)` — `Line not present on system.`

AS$LIN returns this error code if you specify a line number that does not correspond to an asynchronous line on the system.

`E$LNA (333)` — `Lock not allocated.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$LDES (334)` — `Lock has been destroyed.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$LNY (335)` — `Lock is not yours.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$RMLN (336)` — `Illegal operation on remote line.`

AS$LIN returns this error code if you try to get a line number for a remote line. AS$LIN returns line numbers of local NTS lines only.

`E$ITLB (337)` — `Invalid use of terminal line buffer.`

AS$SET returns this error code if you try to set the user number for an assignable asynchronous line.

`E$IPS (338)` — `Invalid parameter setting.`

AS$SET returns this error code if you specify an invalid value for one of the asynchronous line characteristics.

`E$DPAR (339)` — `Duplicate parameter.`

AS$SET returns this error code if you specify duplicate values for an asynchronous line characteristic in the list of line characteristics.

`E$PNS (340)` `Parameter not settable.`

AS$SET returns this error code if you specify a nonexistent asynchronous line characteristic, or a characteristic that you are not permitted to modify.

`E$BCHK (341)`

Not currently returned by PRIMOS.

`E$EXPD (342)`

Not currently returned by PRIMOS.

`E$DNTS (343)` `Density not selected.`

Your attempt to specify a tape density when assigning a magnetic tape drive was rejected. This can occur if the specified tape is not fully rewound.

`E$SNTS (344)` `Speed not selected.`

Your attempt to specify a tape speed when assigning a magnetic tape drive was rejected. This can occur if the specified tape is not fully rewound.

`E$BMPC (345)` `Magtape controller hung.`

A magnetic tape drive did not respond to an I/O request within a reasonable amount of time. This can occur if the power switch on the interface box between a 6250 tape drive and its tape controller is off, or if there is a problem with the tape drive hardware.

`E$GPON (346)` `Password generation on.`

CHG$PW returns this error code if you attempt to manually set a user password when automatic generation of all login validation passwords is enabled. Either disable automatic password generation or use the GEN$PW subroutine to create a computer-generated password. The System Administrator can use the EDIT_PROFILE command to disable automatic password generation.

`E$NGPW (347)` `Password generation off.`

GEN$PW returns this error code if you attempt to create a computer-generated password when automatic password generation is not enabled. Either enable automatic password generation, or use the CHG$PW subroutine to manually change the password. The System Administrator can use the EDIT_PROFILE command to enable automatic password generation.

`E$ISTA (348)` `Invalid state.`

PRIMOS uses this error code internally. It is not currently returned to the user.

`E$ZERO (349)` `Uninitialized block on robust partition.`

An unintialized block was detected in a file on a robust partition. This usually happens following a system halt that was recovered from by running FIX_DISK with the –FAST option. The operation that returned this error code also reinitialized (zeroed out) the bad block. Perform recovery procedures (if necessary) and rerun the program. To prevent multiple E$ZERO errors, you can run FIX_DISK with the –FULL option. This reinitializes all bad blocks on the robust partition.

Alphabetical List of Error Messages

In the course of debugging and running application programs, you will undoubtedly encounter errors. These are reported on your terminal in the form of one or more lines of text, the first of which is a standard PRIMOS error message. The error message may be followed by additional program-specific information.

The purpose of this appendix is to make it easier, when all you have is the text of the error message, to find the corresponding error message description in Appendix A. The message descriptions in Appendix A are ordered numerically.

In the list that follows, messages are listed alphabetically by the text of the displayed error message. Following each error message is the numeric value for each error code in the form *nnn*, and the mnemonic for each error code in the form E$*xxxx*.

Error Message	*Numeric Value*	*Mnemonic*
(Null message)	37	E$NULL
(Null message for logout)	163	E$LOGO
Access category found in directory.	160	E$CATF
Access category not found.	155	E$ACNF
ACL subdirectory found in directory.	161	E$ADRF
ACL too big.	154	E$ACBG
Administrator rules contain error.	252	E$ADRL
Administrator rules not modifiable.	250	E$ADMN
Allocation request too small.	214	E$ALSZ
Already exists.	18	E$EXST
Asynchronous procedure still pending.	190	E$APND
Asynchronous write error.	233	E$AWER
Attribute not supported in directory.	238	E$ATNS
Bad access control list format.	157	E$BACL
Bad access mode.	176	E$BMOD
Bad argument in command.	71	E$BARG
Bad block encountered.	285	E$BLOK
Bad buffer number.	288	E$BBUF
Bad command format.	68	E$CMND
Bad compiler library call.	203	E$BCLC
Bad DAM file.	22	E$BDAM

Error Message	*Numeric Value*	*Mnemonic*
Bad device number.	80	E$BDEV
Bad device number copied.	88	E$SBCF
Bad DTAR.	278	E$BDTR
Bad format.	186	E$FER
Bad key.	28	E$BKEY
Bad key for this protocol.	127	E$BKFP
Bad length.	79	E$BLEN
Bad line number.	287	E$BLIN
Bad number of words.	101	E$BNWD
Bad output data.	78	E$BDAT
Bad parameter.	6	E$BPAR
Bad password.	24	E$BPAS
Bad pointer within SMT.	276	E$BPTR
Bad protocol.	289	E$BPRO
Bad protocol handler specified.	128	E$BPRH
Bad segment directory unit.	30	E$BSUN
Bad segment number.	45	E$BSGN
Bad stack format.	62	E$STKF
Bad stack format signalling.	63	E$STKS
Bad tape record length and EOT.	213	E$BLET
Bad truncate of segment directory.	26	E$BTRN
Bad unit number.	29	E$BUNT
Bad user ID.	266	E$BUID
Bad user identifier.	177	E$BID
Bad version number.	158	E$BVER
Bad virtual circuit clearing.	191	E$BVCC
Beginning of file.	2	E$BOF
BRMS-labeled tape was detected.	204	E$NSB
Buffer in use.	291	E$BFUS
Buffer is too small.	35	E$BFTS
Category protects MFD.	153	E$CPMF
Circular quota parent thread.	282	E$CQPT
Command environment breadth exceeded.	221	E$ECEB
Command illegal.	324	E$COMM
Command line argument missing.	170	E$MISA
Command line truncated.	74	E$TRCL
Concealed stack overflow.	72	E$CSOV
Connection request rejected.	315	E$CREJ
Connection request timed out.	316	E$CTMO
Corrupted area encountered.	283	E$AREA
Corruption detected during use of EXIT.	70	E$NEXP
Crawlout unwind failed.	67	E$CRUN
Date and time not set.	173	E$DTNS
Density not selected.	343	E$DNTS

Error Message	*Numeric Value*	*Mnemonic*
Device already attached.	77	E$DATT
Device forcibly detached.	85	E$DFD
Device in use.	39	E$DVIU
Device not assigned.	48	E$NASS
Device not available.	76	E$DNAV
Device not connected.	100	E$DNCT
Device not started.	42	E$DNS
Device output queue full.	312	E$DOQF
Directory is damaged.	34	E$BUFD
Directory entry list is full.	8	E$FDFL
Directory is not empty.	19	E$DNTE
Disk has been shut down.	121	E$SHDN
Disk I/O error.	21	E$DISK
Disk is full.	9	E$DKFL
Disk is write-protected.	56	E$WTPR
DPTX already configured.	132	E$DANC
DPTX not configured.	86	E$DNC
Duplicate parameter.	339	E$DPAR
E$INWT enabled by configuration.	126	E$IWST
End of file.	1	E$EOF
End of search list.	251	E$EOL
Enqueued only.	124	E$ENQD
EPF file exceeds file size limit.	222	E$EPFL
EPF file not active for this user.	223	E$NTA
EPF file suspended within this process.	225	E$SWPR
EPF has not been registered.	280	E$ENRG
EPF LTD linkage descriptor invalid.	219	E$ILTD
EPF LTE linkage descriptor invalid.	220	E$ILTE
EPF type invalid.	217	E$EPFT
Extent map full.	231	E$EXMF
FAM — invalid function code.	46	E$FIFC
FAM — operation not complete.	53	E$FONC
Fatal error in crawlout.	65	E$CRWL
Field prohibited.	94	E$VFP
Field required.	93	E$VFR
File in use.	5	E$FIUS
File inconsistent data count.	228	E$FIDC
File is delete-protected.	181	E$DLPR
File is too big.	36	E$FITB
File open on delete.	11	E$FDEL
File units all in use.	41	E$FUIU
Format/data mismatch.	185	E$FDMM
Free request with invalid pointer.	215	E$FRER
Illegal 3270 command.	87	E$SICM

Error Message	*Numeric Value*	*Mnemonic*
Illegal alias name.	275	E$IALN
Illegal database.	277	E$IDBT
Illegal DPTX file format.	130	E$ILFF
Illegal EPF registration.	263	E$IREG
Illegal link at EPF registration.	265	E$ILLN
Illegal multiple hops in NPX.	193	E$MNPX
Illegal name.	17	E$BNAM
Illegal operation on remote line.	336	E$RMLN
Illegal remote reference.	38	E$IREM
Illegal treename.	57	E$ITRE
Illegal use of PRIMIX gate.	256	E$ILUS
Input waiting.	83	E$INWT
Insufficient access rights.	10	E$NRIT
Insufficient DAM file index levels.	229	E$INDL
Insufficient free contiguous blocks.	253	E$IFCB
Insufficient memory for extent map.	254	E$IMEM
Invalid AID byte.	90	E$VIA
Invalid ASD use.	294	E$IASD
Invalid block pointer.	270	E$IVPT
Invalid cursor address.	91	E$VICA
Invalid field address.	92	E$VIF
Invalid number of initialization arguments.	264	E$INAI
Invalid parameter setting.	338	E$IPS
Invalid request.	267	E$INRE
Invalid sample speed for ASD.	295	E$IASP
Invalid state.	348	E$ISTA
Invalid static mode resume.	286	E$ISMR
Invalid use of assign line buffer.	293	E$IABF
Invalid use of DISLOG.	296	E$ILOD
Invalid use of remote buffer.	292	E$IRBF
Invalid use of terminal line buffer.	337	E$ITLB
Invalid VMFA window number.	105	E$IVWN
I/O abort in progress.	129	E$ABTI
I/O error or device interrupt.	137	E$IEDI
LHC down.	317	E$LHDN
LHC not configured.	304	E$NTLC
LHC not downline loaded.	306	E$NTDL
Library is non-executable.	245	E$AELE
Library unregistered.	279	E$LUNR
Like reference not accessible.	152	E$LRNA
Like reference not found.	156	E$LRNF
Line in use.	290	E$LNUS
Line not associated.	309	E$NASO
Line not configured.	310	E$NCFG

Error Message	*Numeric Value*	*Mnemonic*
Line not connected.	313	E$LNOC
Line not owned by you.	331	E$LNOW
Line not present on system.	332	E$LNP
Lock has been destroyed.	334	E$LDES
Lock is not yours.	335	E$LNY
Lock not allocated.	333	E$LNA
LTS down.	318	E$LTDN
LTS line already associated.	308	E$LLAA
Magtape command invalid.	99	E$IVCM
Magtape controller hung.	345	E$BMPC
Maximum quota exceeded.	143	E$MXQB
Maximum slaves per user exceeded.	207	E$MSLV
Message operation partially blocked.	117	E$PRTL
Name is too long.	32	E$NMLG
Network configuration mismatch.	175	E$BCFG
Network error detected.	120	E$NETE
No buffer space.	82	E$NBUF
No buffers available.	330	E$NBA
No child found for this process.	257	E$NCHD
No data found.	123	E$NDAT
No directory attached.	7	E$NATT
No directory block for unit.	281	E$NDRB
No DTAR3 segments available.	273	E$ND3S
No fault frame.	61	E$NFLT
No free quota blocks.	142	E$NFQB
No unit table available.	167	E$NFUT
No information is accessible.	159	E$NINF
No input available.	84	E$NINP
No more temporary segments.	108	E$NMTS
No more VMFA segments.	107	E$NMVS
No NPX slaves available.	134	E$NSLA
No on-unit found.	64	E$NOON
No paging device defined.	240	E$NOPD
No password directories allowed.	327	E$NPDA
No phantoms available.	54	E$NPHA
No resources available for request.	255	E$NRES
No room.	55	E$ROOM
No SMLC DMC channels.	75	E$NDMC
No timer.	51	E$NTIM
No unit table available for phantom.	164	E$NUTP
Node/LAN naming conflict.	298	E$NSNC
Node/MAC address conflict.	299	E$NSAC
Not a DAM file.	109	E$NDAM
Not a directory.	12	E$NTUD

Error Message	*Numeric Value*	*Mnemonic*
Not a file or directory.	149	E$NTFD
Not a quota disk.	144	E$NOQD
Not a segment directory.	13	E$NTSD
Not able to free storage.	272	E$NATF
Not accessible.	209	E$NACC
Not an access category.	151	E$NCAT
Not an ACL directory.	147	E$NACL
Not an NTS configuration file.	303	E$NTCF
Not enough DMA channels.	210	E$UDMA
Not enough DMC channels.	211	E$UDMC
Not enough per-user DTAR1 segments.	268	E$NPSG
Not enough segments.	103	E$NESG
Not found.	15	E$FNTF
Not found in segment directory.	16	E$FNTS
Not open for VMFA.	110	E$NOVA
Not owner of resource.	284	E$NOWN
NSS database not initialized.	297	E$NSNI
NTS already started.	302	E$NTST
NTS database not initialized.	305	E$NTIN
NTS host not configured.	300	E$NTHN
NTS line already associated.	307	E$PLAA
NTS not started.	301	E$NTNS
NTS is shut down.	319	E$NTSH
Null smt_ptr or bad field within SMT.	274	E$BSMT
Null command line.	60	E$NCOM
Object is category-protected.	179	E$CTPR
Object is default-protected.	180	E$DFPR
Old partition.	27	E$OLDP
Operation completed successfully.	0	E$OK
Operation illegal on access category.	150	E$IACL
Operation illegal on directory.	14	E$DIRE
Operation illegal on MFD.	146	E$IMFD
Operation illegal on pre-19 disk.	178	E$ST19
Operation unsuccessful.	118	E$NSUC
Overflow of CPU and I/O seconds.	244	E$BHOV
Overflow of CPU seconds.	242	E$CPOV
Overflow of I/O seconds.	243	E$IOOV
Page is already wired.	325	E$AWIR
Page is not wired.	326	E$IWIR
Parameter not settable.	340	E$PNS
Parent not an ACL directory.	148	E$PNAC
Password generation off.	347	E$NGPW
Password generation on.	346	E$GPON
Past end of field.	97	E$VPEF

Error Message	*Numeric Value*	*Mnemonic*
Past end of file.	230	E$PEOF
PIO instruction did not skip.	139	E$DNSK
PRIMIX can not be initialized when running.	259	E$XSHD
PRIMIX can not be shut down when not running.	260	E$NOPX
PRIMIX process table has no users.	261	E$NOUS
PRIMIX process table returned is incomplete.	262	E$INCO
PRIMIX wait terminated by interrupt.	258	E$INT
Priority ACL not found.	169	E$PANF
Procedure not found.	135	E$PNTF
Process not a slave.	236	E$NSLV
Protected field check.	95	E$VPFC
Protocol handler not available.	125	E$PHNA
Pointer mismatch found (FAM only).	23	E$PTRM
Queue is empty.	321	E$QEMP
Queue is full.	320	E$QFUL
Queue length exceeded.	81	E$QLEX
Queue not found.	322	E$NOQ
Quota set below current usage.	145	E$QEXC
Receive enabled required.	112	E$NRCV
Registered EPF is in use.	329	E$REIU
Remote disk has been shut down.	239	E$RSHD
Remote line is down.	40	E$RLDN
Remote node not enabled.	133	E$NENB
Remote procedure call still pending.	174	E$SPND
Remote system has initialized.	237	E$RSIN
Remote system not up.	140	E$RSNU
Request queue full.	314	E$RQF
Reserved character.	69	E$RCHR
Restricted access file.	192	E$RESF
ROAM access mode conflict.	234	E$RAMC
ROAM internal error.	235	E$RIER
Search list not found or invalid.	246	E$LIST
Search lists nested too deeply.	249	E$NEST
Search rule not an optional rule.	248	E$NTOP
Search rule not found or invalid.	247	E$RULE
Segment directory error.	33	E$SDER
Segment directory unit not open.	31	E$SUNO
Segment does not exist.	73	E$NOSG
Segment in use.	102	E$SGIU
Segment not allocated.	271	E$SNAL
Semaphore overflow.	50	E$SEMO
Slave ID mismatch.	205	E$WSLV
Slave ID number not found.	208	E$IDNF
Slave validation error.	136	E$SVAL

Error Message	*Numeric Value*	*Mnemonic*
Speed not selected.	344	E$SNTS
Spooler subsystem not initialized.	328	E$NINT
Stack overflow in crawlout.	66	E$CROV
System administrator command only.	226	E$ADCM
System console command only.	171	E$SCCM
Too many emulate devices.	131	E$TMED
Too many subdirectory levels.	43	E$TMUL
Too many users.	59	E$TMUS
Top-level directory not found or inaccessible.	189	E$NFAS
Unable to allocate file-unit.	227	E$UAFU
Uninitialized block on robust partition.	349	E$ZERO
Unit in use.	4	E$UIUS
Unit not open.	3	E$UNOP
Unit open for block mode I/O.	232	E$BKIO
Unit table already returned.	165	E$UTAR
Unit table not in use.	166	E$UNIU
Unknown addressee.	116	E$UADR
Unknown node name.	122	E$UNOD
Unterminated string.	195	E$USTR
User already has unit table.	168	E$UAHU
User busy, please wait.	114	E$UBSY
User ID not found.	269	E$UINF
User not receiving now.	113	E$UNRV
User storage heap is corrupted.	216	E$HPER
User unable to receive messages.	115	E$UDEF
Validation error.	323	E$VAL
Validation error.	162	E$NVAL
Virtual circuit was cleared.	206	E$VCGC
Warm start occurred.	138	E$WMST
Window already in address space.	106	E$WAIN
Wrong file type.	184	E$WFT
XCB unavailable for request.	311	E$NXCB
Unused code.	20	E$SHUT
Unused code.	25	E$BCOD
Unused code.	44	E$FBST
Unused code.	47	E$TMRU
Unused code.	49	E$BFSV
Unused code.	52	E$FABT
Unused code.	58	E$FAMU
Unused code.	89	E$VKBL
Unused code.	96	E$VNFC
Unused code.	98	E$VIRC
Unused code.	104	E$SDUP
Unused code.	111	E$NECS

Error Message	*Numeric Value*	*Mnemonic*
Unused code.	119	E$NROB
Unused code.	141	E$S18E
Unused code.	172	E$BRPA
Unused code.	182	E$BLUE
Unused code.	183	E$NDFD
Unused code.	187	E$BDV
Unused code.	188	E$BFOV
Unused code.	194	E$SYNT
Unused code.	196	E$WNS
Unused code.	197	E$IREQ
Unused code.	198	E$VNG
Unused code.	199	E$SOR
Unused code.	200	E$TMVV
Unused code.	201	E$ESV
Unused code.	202	E$VABS
Unused code.	212	E$BLEF
Unused code.	218	E$EPFS
Unused code.	224	E$SWPS
Unused code.	241	E$NRFC
Unused code.	341	E$BCHK
Unused code.	342	E$EXPD

New Features of Recent PRIMOS Revisions

This appendix lists new features significant to the system-level programmer in recent revisions of PRIMOS. Summaries of new functionality appear in the *Software Release Document* for the appropriate PRIMOS revision. For details on enhanced compiler functionality, consult the individual language guides. For further information on new or modified subroutines, consult the *Subroutines Reference* series.

This appendix lists enhancements made in several recent PRIMOS revisions. The most recent revision is listed first.

New Features at Revision 22.0

Subroutines

The following subroutines have been added at Revision 22.0:

- The SYN$ subroutines permit you to create and destroy event synchronizers, post notices on event synchronizers, wait for the posting of a notice on an event synchronizer, and retrieve a notice from an event synchronizer. Other SYN$ subroutines enable you to group several event synchronizers into an event group and wait for a notice or retrieve a notice from that event group. Additional SYN$ subroutines enable you to check the status of event synchronizers and event groups. These subroutines are described in *Subroutines Reference V: Event Synchronization.*
- The TMR$ timer subroutines permit you to create timers that post a notice on a specified event synchronizer after a specified interval. There are subroutines to establish timers for a specified elapsed period of time, a specified time of day, or a specified recurrent interval of time. These subroutines are described in *Subroutines Reference V: Event Synchronization.*
- The TMR$GTIM and TMR$GINF subroutines return current system time or permanent time information. These subroutines are described in *Subroutines Reference III: Operating System.*
- The TMR$UNIVCONVERT and TMR$LOCALCONVERT subroutines convert Universal Time to local time and local time to Universal Time. These subroutines are described in *Subroutines Reference III: Operating System.*

- The SRS$ subroutines permit you to determine the server name associated with a process, the processes that share the same server name, and the list of all server names on your system. These subroutines are described in *Subroutines Reference V: Event Synchronization.*
- The ISN$ subroutines permit you to catalog the server name of a process in a High Level Name File (HLNF), thus making that server name available to other users, and to look up the server name of a process by specifying the pathname of an HLNF. These subroutines are described in *Subroutines Reference V: Event Synchronization.*
- The IS$ subroutines permit you to use the InterServer Communications (ISC) facility to exchange messages between processes. The processes can be on the same system or on different systems connected using PRIMENET. Subroutines are provided for requesting a message exchange session between two processes, specifying event synchronizers and other features used during the session, sending and receiving messages, and terminating the session. There are also subroutines for retrieving information about a session. These subroutines are described in *Subroutines Reference V: Event Synchronization.*
- The AS$SET, AS$LST, and AS$LIN subroutines permit you to set the characteristics of an asynchronous line, retrieve the characteristics of an asynchronous line, and retrieve the line number of an asynchronous line. These subroutines are described in *Subroutines Reference IV: Libraries and I/O.*
- The ER$PRINT and ER$TEXT subroutines permit you to display an error message on your terminal or return an error message to a variable. These subroutines replace ERRPR$ and ERTXT$, which are now considered obsolete. They are described in *Subroutines Reference III: Operating System.*
- CF$EXT extends or truncates a CAM file. This subroutine is described in *Subroutines Reference II: File System.*
- CF$REM gets a CAM file's extent map. This subroutine is described in *Subroutines Reference II: File System.*
- CF$SME sets a CAM file's extent length value. This subroutine is described in *Subroutines Reference II: File System.*
- LN$SET modifies a user's search rules to permit dynamic linking to an EPF library. This subroutine is described in *Subroutines Reference II: File System.*
- GEN$PW generates a login validation password. This subroutine is described in *Subroutines Reference III: Operating System.*
- GTROB$ determines whether a specified file is on a robust partition. This subroutine is described in *Subroutines Reference II: File System.*
- ECL$CC and ECL$CL supervise editing of input from a terminal or a command file. ECL$CC is callable from C. ECL$CL is an interface to ECL$CC for non-C programs. These subroutines are described in *Subroutines Reference III: Operating System.*
- NT$LTS returns the characteristics of a PRIMOS network terminal service line. This subroutine is described in *Subroutines Reference IV: Libraries and I/O.*

- ICE$ has been enhanced to support synchronizers, timers, ISC sessions, and other features of PRIMOS. This subroutine is described in *Subroutines Reference III: Operating System.*

PRIMOS Commands

Revision 22.0 has the following new PRIMOS commands:

- The LIST_SESSIONS and LIST_SERVER_NAMES commands and the –SERVER option for the INITIALIZE_COMMAND_ENVIRONMENT (ICE) command support servers and ISC sessions.
- The LIST_CONTIGUOUS_BLOCKS and LIST_EXTENT_MAP (LEM) commands support CAM files.
- The UX_TAPE command saves files to tape in a format that the UNIX CPIO and TAR utilities can read. It restores files from a tape created by either CPIO or TAR.

The EDIT_CMD_LINE (ECL) facility has been enhanced to include the user's ability to define terminal key functions for editing a command line. ECL is described in *PRIMOS Commands Reference Guide* and *PRIMOS User's Guide.*

Subsystem Enhancements

The following subsystems have been enhanced with additional features and options. These are further described in the *Software Release Document* and the documentation for the individual subsystems.

- The Spooler subsystem has been enhanced with additional embedded control code options and several new command features. The SPOOL command has four new options: –XLATE for character set mapping, –FROM and –TO for printing a part of a document, and –SPOOL_W for printing a file while it is open for writing. The new AUXILIARY command passes environment parameters to print handlers. The PROP command –BACK option has been extended.
- MAGNET has been extended to handle the Prime Extended Character Set (Prime ECS) and to support large tape buffers.
- Tape utilities (such as MAGSAV and MAGRST) at Rev. 22.0 permit a larger maximum record size. This enhancement is due to a change in the T$MT subroutine.

New Features at Revision 21.0

Subroutines

The following subroutines are either added or enhanced at Revision 21.0:

- DS$AVL returns data about a disk partition in a structure. Data returned includes the version number of the structure to be returned, the name of the partition, its maximum capacity, the number of free records, and the date and time the partition was last backed up.
- DS$ENV returns data about the user's process. Data returned includes the filename of the currently active abbreviation file; the unit number of the current command input file; the user's current command level, erase character, and kill characters; the default and current user timeslice; the CPU and login time remaining; the QUIT inhibit count; the number and name of the ACL groups to which the user belongs; and the number, name, node, user ID, and project ID for the user's remote IDs.
- DS$UNI returns data about file-units. Data returned includes information about attach points, the user number, access bits if the file is open on a local system, open mode, the command output file-unit, and the system name if the file is open on a remote system.
- GSNAM$ is used by any program to determine the name of the system the program is running on.
- G$METR returns system metering information, such as that provided by the USAGE command. This information can be for general system meters, file system meters, interrupt process meters, system meters for an individual user, meters for memory usage, meters for disk usage, and meters for ROAM usage. Returned information includes the CPU, I/O, and real time used, the number of I/O operations since system boot, the number of users configured, information about locate buffers, and read and write operations performed.
- KLM$IF enables a program to obtain serialization data from a specified file. KLM$IF uses a simple filename, supplied by a program, and system search rules to obtain serialization data from an installed product of that name. Data obtained about the product can include its version number, its name, its revision number, its serial number, the name of the licensed user, the software expiration date, PRIMOS support, the name of the organization distributing the software, the name of the individual responsible for software revision, the software distribution date, the order number of the distributed software, and the customer service number for the product license.
- LOV$SW indicates if the login-over-login function is currently permitted.
- LUDEV$ returns a list of devices that a user can access. The devices listed are those that are specified by the user with the ASSIGN command. Information returned includes the version number, the maximum number of devices that may be accessed, and a list of devices that the user may access.
- MM$MLPA makes the last page of a segment available.
- MM$MLPU makes the last page of a segment unavailable. Subsequent attempts to access the page result in the OUT_OF_BOUNDS$ condition.
- SGD$EX determines if there is a valid entry at the current position within the segment directory on a specified unit.

- SNCHK$ checks the validity of the system name passed to it. SNCHK$ enables subsystems that deal with system names at a command interface to check the names for validity without knowing the syntax rules for system names.
- SP$REQ inserts a file into the spool queue.
- SR$ABSDS (or SR$ABS for FTN) disables optional search rules enabled by SR$ENABL. SR$ABSDS absolutely disables an enabled rule, regardless of how many times the rule has been enabled.
- SR$ADDB (or SR$ADB for FTN) adds a rule to the start of a search list or before a specified rule within the list.
- SR$ADDE (or SR$ADE for FTN) adds a rule to the end of a search list or after a specified rule within the list.
- SR$CREAT (or SR$CRE for FTN) creates a blank search list. The created search list does not contain any user-specified or system default search rules. This search list does, however, contain administrator rules if the System Administrator has established administrator rules for the search list.
- SR$DEL deletes a specified search list. Both the user's search list and its contents (including administrator rules) are deleted. The search rules file that was used to set the search list is unaffected.
- SR$DSABL (or SR$DSA for FTN) disables an optional search rule enabled by SR$ENABL. This subroutine reverses a single SR$ENABL operation. Compare this with SR$ABSDS.
- SR$ENABL (or SR$ENA for FTN) enables an optional search rule. You can disable enabled rules using SR$DSABL or SR$ABSDS.
- SR$EXSTR (or SR$EXS for FTN) determines if a search rule exists in a specified search list. The search rule can be a pathname, an optional search rule, or a search rule keyword. SR$EXSTR determines the existence of both disabled and enabled optional search rules.
- SR$FR_LS (or SR$FRL for FTN) frees list structure space allocated by SR$LIST or SR$READ. Invoke SR$FR_LS after every successful invocation of SR$LIST or SR$READ. SR$FR_LS deletes a structure by following the structure's internal pointers.
- SR$INIT (or SR$INI for FTN) initializes all search lists to system defaults. System default rules include all rules found in the directory SEARCH_RULES*, including system rules and administrator rules. If no system defaults exist for a search list, SR$INIT deletes the search list.
- SR$LIST (or SR$LIS for FTN) returns the names of the user's search lists. SR$LIST copies information about all of the user's search lists into a user-specified structure. SR$LIST creates a separate structure entry for each of the user's search lists.
- SR$NEXTR (or SR$NEX for FTN) reads the rules from a search list, sequentially and one at a time. Each invocation of SR$NEXTR reads one rule. To read all of the rules in a search list, use SR$READ. SR$NEXTR reads locator pointer values. SR$NEXTR does not read disabled optional search rules.

- SR$READ (or SR$REA for FTN) reads all of the rules in a search list into a structure established by the user. SR$READ reads all rules, including disabled rules. SR$READ creates a separate structure entry for each search rule.
- SR$REM removes a search rule from a specified search list. SR$REM can delete user-specified and system default search rules and keywords. SR$REM cannot delete administrator search rules.
- SR$SETL (or SR$SET for FTN) sets or modifies the locator pointer for a search rule. SR$SETL can set locator pointers of search rules in user-defined search lists and search rules in the ENTRY$ search list.
- SR$SSR sets a search list via a user-defined search rules file. SR$SSR can create a new search list, overwrite an existing search list, or append rules to an existing search list.
- K$BKUP was added to SRSFX$ to allow a file to be read by the backup facility.

Other New Features

Revision 21.0 has the following new features and changes:

- Extension to the use of search lists and ability for the user to define search lists. See the *Advanced Programmer's Guide, Volume II: File System* for a complete discussion.
- Prime ECS support (expanded character set).
- CBL support of INCLUDE$ search rules, enhanced magnetic tape support, relative file enhancements for MIDASPLUS™ and PRISAM,™ and new compiler options.
- CC support of INCLUDE$ search rules, the UNIX/ANSI restriction on files opened with FOPEN, and a new meaning of the returned value of OPEN().
- F77 support of INCLUDE$ search rules, SHORTCALL functionality in I mode, longer string constants, and optimization enhancements.
- FTN generation of V-mode code as the compiler default.
- PMA support of the MIP pseudo-op, mode determination of variables and expressions, assembler listing, general register relative format, and IX-mode instructions.
- Pascal support of INCLUDE$ search rules and some changes concerning the ANSI/IEEE standard.
- VRPG support of INCLUDE$ search rule.
- BIND support of COMPRESS and INITIALIZE_DATA.
- EMACS interface with Prime Common LISP.

New Features at Revision 20.2

New Features

Revision 20.2 has the following new features and changes:

- CBL_LIBRARY supports sequential file access and variable length tables and records.
- CC_LIBRARY resolves potential library routine and runfile conflicts.
- System Library supports F77 octal and decimal formatting and an improved random number generator.
- VRSTLI becomes an Executable Program Format (EPF).
- MATRIX_LIBRARY (MATHLB) becomes an Executable Program Format (EPF).
- VRPG supports new options.
- PL/I supports new options.
- F77 supports new options, statements, constants, static mapping to tape unit, and enhanced cross-reference functionality.
- CC supports 32IX mode, new options, new switches, a FORTRAN interface, and has changes in the ctype.h header file.
- Pascal supports new options, conforms to the ANSI/IEEE standards, and provides new options for ANSI/IEEE standards conformance.
- The Source Level Debugger supports variable length records, octal and hexadecimal constants, and has enhancements to MACRO.
- BIND supports two new subcommands.
- EMACS provides UNIX pathname support, two new PEEL functions, and a new PEEL atom.
- K$DTA and K$DTC keys added to SATR$$ to allow setting of date/time accessed and date/time created.
- The subroutine SRSFX$, which supports pathnames, can now be used to search for a file. T$SRC, which was previously used, is obsolete at this revision.

System Library

The System Library supports the following changes at Rev. 20.2:

- F77 octal and hexadecimal formatting
- Random number generation

New Features at Revision 20.0

Subroutines

The following subroutines are either added or enhanced at Revision 20.0:

- DIR$CR creates a new directory. This subroutine accepts pathnames and replaces CREA$$ and CREPW$, which are obsolete at this revision.
- DIR$RD reads the contents of a directory sequentially, entry by entry.
- DIR$SE searches the directory with caller-specified selection criteria.
- DKGEO$ counts the sectors of a disk that has been formatted in a nonstandard manner.
- IOCS$_FREE_LOGICAL_UNIT frees a logical file-unit number and makes it available in the Logical Unit Table (LUTBL).
- IOCS$GET_LOGICAL_UNIT provides an available logical file-unit number to the calling program.
- SIZE$ returns the size of a file system entry without updating Date Time Accessed (DTA).
- UNIT$ reads the current minimum and maximum unit number for this user.

Other New Features

Revision 20.0 has the following new features and changes:

- Directories are now organized as hashed ACL directories.
- The new file attributes, date and time created (DTC) and date and time last accessed (DTA) may appear in Rev. 20.0 or later directories (hashed directories).
- The structure returned after calls to DIR$RD or ENT$RD includes the new file attributes DTC and DTA.

Master Index

Key to Master Index:

Abbreviation	Document Title	Document Number
0	Advanced Programmer's Guide, Volume 0: Introduction and Error Codes	DOC10066-3LA
I	Advanced Programmer's Guide, Volume I: Bind and EPFs	DOC10055-1LA
II	Advanced Programmer's Guide, Volume II: File System	DOC10056-2LA
III	Advanced Programmer's Guide, Volume III: The Command Environment	DOC10057-1LA

Master Index

Surveys

Reader Response Form

Advanced Programmer's Guide, Volume 0: Introduction and Error Codes
DOC10066-3LA

Your feedback will help us continue to improve the quality, accuracy, and organization of our user publications.

1. How do you rate this document for overall usefulness?

   ☐ *excellent* ☐ *very good* ☐ *good* ☐ *fair* ☐ *poor*

2. What features of this manual did you find most useful?

   ____

3. What faults or errors in this manual gave you problems?

   ____

4. How does this manual compare to equivalent manuals produced by other computer companies?

☐ *Much better*	☐ *Slightly better*	☐ *About the same*
☐ *Much worse*	☐ *Slightly worse*	☐ *Can't judge*

5. Which other companies' manuals have you read?

   ____

Name:____________________
Position:____________________
Company:____________________
Address:____________________
____
____________________Postal Code:____________________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

First Class Permit #531 Natick, Massachusetts 01760

BUSINESS REPLY MAIL

Postage will be paid by:

Prime™

Attention: Technical Publications
Bldg 10
Prime Park, Natick, Ma. 01760

Reader Response Form

Advanced Programmer's Guide, Volume 0: Introduction and Error Codes
DOC10066-3LA

Your feedback will help us continue to improve the quality, accuracy, and organization of our user publications.

1. How do you rate this document for overall usefulness?

   ☐ *excellent* ☐ *very good* ☐ *good* ☐ *fair* ☐ *poor*

2. What features of this manual did you find most useful?

   ____
   ____
   ____
   ____
   ____
   ____

3. What faults or errors in this manual gave you problems?

   ____
   ____
   ____
   ____
   ____
   ____

4. How does this manual compare to equivalent manuals produced by other computer companies?

   ☐ *Much better* ☐ *Slightly better* ☐ *About the same*
   ☐ *Much worse* ☐ *Slightly worse* ☐ *Can't judge*

5. Which other companies' manuals have you read?

   ____
   ____

Name:____
Position:____
Company:____
Address:____
____
______________________________Postal Code:______________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

First Class Permit #531 Natick, Massachusetts 01760

BUSINESS REPLY MAIL

Postage will be paid by:

Prime™

Attention: Technical Publications
Bldg 10
Prime Park, Natick, Ma. 01760

Reader Response Form

Advanced Programmer's Guide, Volume 0: Introduction and Error Codes
DOC10066-3LA

Your feedback will help us continue to improve the quality, accuracy, and organization of our user publications.

1. How do you rate this document for overall usefulness?

   ☐ *excellent* ☐ *very good* ☐ *good* ☐ *fair* ☐ *poor*

2. What features of this manual did you find most useful?

   ____
   ____
   ____
   ____
   ____
   ____

3. What faults or errors in this manual gave you problems?

   ____
   ____
   ____
   ____
   ____
   ____

4. How does this manual compare to equivalent manuals produced by other computer companies?

☐ *Much better*	☐ *Slightly better*	☐ *About the same*
☐ *Much worse*	☐ *Slightly worse*	☐ *Can't judge*

5. Which other companies' manuals have you read?

   ____
   ____

Name:____
Position:______________________________________
Company:______________________________________
Address:______________________________________
____
__________________________Postal Code:_______________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

First Class Permit #531 Natick, Massachusetts 01760

BUSINESS REPLY MAIL

Postage will be paid by:

Prime™

Attention: Technical Publications
Bldg 10
Prime Park, Natick, Ma. 01760

DOC10066-3LA