The Hacker's Guide to the Prime

Prime Computer

42

The Hacker's Guide to the Prime

Don't Panic!

Prime

The Hacker's Guide to the Prime

PE-TI 1300

Donald M. Koch

PRIME ENGINEERING HANDBOOK
This revision corresponds to PRIMOS REV 21.0.

This is the third release of the Prime Engineering Handbook, a document produced and maintained by Prime Computer Research and Development. Comments should be addressed to:

Donald Koch
Prime Computer Research and Development
MS 10C-13
500 Old Connecticut Path
Framingham, MA 01701
Internet mail: aardvark@marvin.prime.com

Published by Prime Computer, Inc.

March 1988

Table of Contents

1. Introduction

This handbook provides a summary of information needed for the development and maintenance of Prime 50 Series hardware and software systems. While this book contains information useful to a general user community, the information is presented in very condensed form. It is assumed that the reader has had prior contact with this material and, therefore, that detailed descriptions are unnecessary.

Some of the information contained herein pertains only to the latest revision of PRIMOS. This information will be updated on a regular basis as new revisions are released. (Refer to the inside of the cover for the revision currently reflected in this version of the handbook.)

NOTICE

Some of the information contained within this document is **not** released. Unless stated otherwise, or verified by reviewing a published Tech. Pubs. document, none of the information in this manual should be disclosed. Prime confidential information will be shown in shaded print or will be marked with the NR notation; non-shaded print does not necessarily indicate release of a function.

1.0.1. Acknowledgments

I would like to thank the many reviewers, some of whom, due to my wonderful memory for names, I have undoubtedly forgotten. Those among the remembered are: Dick Snyder (who funded this clam bake), Ewan Milne, Marilyn Hammond (who kept fueling the architecture chapter), Don Slutz, Dave Hornbaker, Dave Peterson, John P. Jones, Kent Fielden, Chris Allen, Peter Borner, Martin Phillips, Martin Doughty, Peter Hassall, Patrick O'Kane, Doug Rand, Cathy Phipps and Jerry Kazin. The cover design was done by John Gustin.

1.0.2. Corrections and updates

Heaven forbid that there should be errors in a document this size, but in the event that one (or even two) be found, or some new feature should be added, please send mail indicating the update to me via PDNmail (aardvark@marvin.prime.com) or xmail (aardvark -on enb).

2. COMMANDS

2.1. Command Syntax

The command descriptions in this manual use the following syntax:

2.1.1. ABBREVIATIONS

Uppercase letters represent abbreviations for commands and options. (When actually typing the command or option, either uppercase or lowercase can be used.) For example:

```
COMOutput
```

specifies the COMOUTPUT command.

2.1.2. COMMAND LINE VARIABLES

Italics indicate a variable for which specific information is to be substituted; for example:

filename

should be replaced with a valid filename.

2.1.3. OPTIONAL PARAMETERS

Brackets enclose optional parameters for a command; for example:

```
SHutdn [ALL]
```

2.1.4. ALTERNATIVE OPERAND SPECIFICATION, DEFAULTS

When an operand has more than one possible specification, choices are enclosed in braces ({}), brackets ([]), if optional, and separated by vertical bars (|). A default option, if any, is underscored; for example:

```
OPRpri { 1 | 0 }
```

The OPRPRI command accepts a single parameter of 1 or 0. If none is specified, the default parameter is 0.

2.1.5. Repeated Operands

Ellipsis indicate an operand that may be repeated one or more times; for example:

```
Close funit ...
```

The CLOSE command accepts one or more file unit specifications (separated by blanks or commas).

2.2. Wildcards and Name Generation

Some commands accept wildcard names. Names are divided into components by periods. A wildcard name is a filename that contains one or more of the following characters:

@ matches zero or more characters in the corresponding component.

@@
matches zero or more characters including periods.

+ matches any single character in the corresponding component except periods.

^ selects the subset of objects whose names do NOT match the wildcard name. If used, the "^" must be the first character in the wildcard name.

Name generation from wildcarded and non-wildcarded names may be done by utilizing one or more of the following:

= Copy the corresponding component.

== Copies one or more components.

^= Excludes a single component.

^== Excludes one or more components.

literal-string
Replace component with *literal-string*.

+*literal-string*
Adds the component given by *literal-string*.

In addition to the options for the command, the following may be added to control wildcard action:

-FILE
SAM, DAM or CAM files.

-DIRectory
Directories.

-SEGment_DIRectory
Segment directories (SAM or DAM).

-Access_CATegory
Access categories.

-RBF
Recovery based files.

-MoDified_After,-AFter *date.time*
Objects last modified after *date.time*.

-MoDified_Before,-BeFore *date.time*
Objects last modified before *date.time*.

-ACcessed_After *date.time*
Accessed after *date.time*.

-ACcessed_Before *date.time*
Accessed before *date.time*.

-BacKedup_After *date.time*
Backed up after *date.time*.

-BacKedup_Before *date.time*
Backed up before *date.time*.

-CReated_After *date.time*
Created after *date.time*.

-CReated_Before *date.time*
Created before *date.time*. -VeriFY
Force verification of generated names before execution.

-No_VeriFY
Suppress verification.

2.3. Filename Suffix Convention

BASIC	BASICV source file (BASICV).
BIN	binary file.
C,CC	C source file (CC, CI).
CBL	CBL source file (CBL).
COBOL	COBOL source file (COBOL).
COMI	command input file.
COMO	command output file.
CPL	CPL file (CPL, RESUME, JOB, PHANTOM).
FTN	FORTRAN source file or insert file (FTN).
F77	FORTRAN 77 source or insert file (F77).
LIST	listing file.
MAP	load map file.
MOD	Modula-2 source file (MODULA).
PASCAL	PASCAL source or insert file (PASCAL).
PLP	PLP source or insert file (PLP).
PL1	PL1 source or insert file (PL1).
PL1G	PL1G source or insert file (PL1G).
PMA	PMA source or insert file (PMA).
RPx	Replacements of EPFs (COPY, BIND).
RPG	RPG source file (RPG, VRPG).
RUN	EPF runfile (BIND, RESUME).
RUNI	Runoff input file(RUNOFF).

RUNO Runoff output file(RUNOFF).

SAVE R-mode runfile (RESUME).

SEG segmented runfile (SEG).

SPSS SPSS input file (SPSS).

SPL SPL source or insert file (SPL).

SYM Modula-2 symbol table file (MODULA).

VRPG VRPG source file (VRPG).

2.4. Command Resume Order

Resumable commands end with one of the suffixes: .RUN, .CPL, and .SAVE; or with no suffix at all. If more than one of these is found, the order of preference by which they are executed are: .RUN, .SAVE, .CPL, *none*.

2.5. Command Procedure Language (CPL)

To invoke CPL, type:

```
Resume pathname[.CPL]
  or
CPL pathname
```

CPL allows one statement per line. A statement is either a CPL directive or a PRIMOS command. A CPL directive has the form:

```
&directive_name arguments
```

where *arguments* are expressions, CPL directives, or PRIMOS commands. Commands may be continued onto a second line by appending a tilde (~) to the end of the line.

2.5.1. CPL Directives

&ARGS [*name*[:*type*][=*default*] | *name*: *-ctl_list*;...]
argument specification and validation. Arguments can be positional or control arguments. They can also be assigned types and default values.

&CALL *routine_name*
invoke a routine defined by a &ROUTINE. Routine returns when &RETURN executes.

&CHECK *expr* &ROUTINE *handler*
invoke *handler* if *expr* is true.

&DATA *stmt*
compute input for a subsystem call at runtime. Format:

```
&DATA stmt
 data1
    ...
 datan
&END
```

&DEBUG *opt_list*
enable and disable debugging facilities. Options are:

&OFF
turn off all debugging options.
&NO_EXECUTE, &NEX
suppress execution of PRIMOS commands but interpret CPL directives.
&EXECUTE, &EX
enable execution of PRIMOS commands (default).
&ECHO [ALL | COM | DIR]
ALL echo PRIMOS commands and CPL directives (dflt); COM echo PRIMOS commands; DIR echo CPL directives.
&NO_ECHO [ALL | COM | DIR]
ALL cancel all echoing (dflt); COM cancel echoing of PRIMOS commands; DIR cancel echoing of CPL directives.
&WATCH [$var_1 ... var_{16}$]
add var_i to the list of watched variables. If no list is specified, all variables are watched.
&NO_WATCH [$var_1 ... var_{16}$]
remove var_i from the list of watched variables.

&DO [*iteration*]
consider all statements between the &DO and &END as a single statement. Format:

```
&DO [iteration]
        stmt1
          . . .
        stmtn
&END
```

iteration is: [*var* {:= *start* [&TO *expr*] [&BY *expr*] | &LIST *list* | &ITEMS *items*}] [&WHILE *cond*] [&UNTIL *cond*]

&EXPAND *switch* [&USING *processor_name*]
enable and disable statement expansion. *switch* can be ON or OFF.

&GOTO *label*
transfer control to *label*.

&IF *cond* &THEN *true_stmt* [&ELSE *false_stmt*]
conditional test.

&LABEL *label_name*
define label that identifies the next statement.

&ON *condition* &ROUTINE *handler_label*
define handler for *condition*.

&RESULT *expr*
return the value of a user defined function.

&RETURN [*severity*] [&MESSAGE *text*]
return severity code to the invoker.

&REVERT *condition*
cancel the handler for *condition*.

&ROUTINE *routine_name*
identify following code as an internal routine.

&SELECT *expr*
evaluate &SELECT *expr*, compare to &WHEN *expr*, and execute appropriate *stmt*.

```
&SELECT expr
   &WHEN expr1 [,...,exprn ]
   stmt
   [&WHEN expr1 [,...,exprn ]
   stmt
    ...
   [&OTHERWISE
   stmt ]]
&END
```

&S[ET_VAR] var_1 [,...,var_n] := *value*
set one or more local or global variables.

&SEVERITY {&ERROR | &WARNING} {&FAIL | &IGNORE | &ROUTINE *label*}
specify the action to be taken when certain severity codes are produced.

&SIGNAL *condition* [&NO_RETURN]
raise the *condition* and search its handler.

&STOP [*severity*] [&MESSAGE *text*]
abort current CPL procedure and any routines it has invoked.

&TTY
take input from terminal. Used within an &DATA block.

&TTY_CONTINUE
take input from previous command stream. Used within an &DATA block.

2.6. Command Functions

In the following examples beginning and ending brackets are entered literally.

2.6.1. Logical, Arithmetic, and Relational Functions

[CALC *infix_expr*]
evaluate expressions containing the following logical operators in the order indicated:

```
(highest):   ^   unary +   unary -
             /   *
             +   -
             =   ^=   <   >   <=   >=
             &
(lowest):    |
```

[HEX *hex_string*]
return a string representation of the decimal equivalent of *hex_string*.

[MOD dec_str_1 dec_str_2]
return the string representation of the decimal equivalent of dec_str_1 modulo dec_str_2.

[OCTAL *oct_str*]
return the string representation of the decimal equivalent of *oct_str*.

[TO_HEX *dec_str*]
return a string representation of the hexadecimal equivalent of *dec_str*.

[TO_OCTAL *dec_str*]
return a string representation of the octal equivalent of *dec_str*.

2.6.2. String Functions

[AFTER *str find_str*]
return in quotes the substring of *str* that occurs to the right of the leftmost occurrence of *find_string* in *str*.

[BEFORE *str find_str*]
return in quotes the substring of *str* that occurs to the left of the leftmost occurrence of *find_str* in *str*.

[INDEX *str find_str*]
return the position of the leftmost occurrence of *find_str* in *str*, else 0.

[LENGTH *str*]
return the number of characters in *str*.

[NULL *str*]
return TRUE if *str* is the true null string, else '' and FALSE.

[QUOTE str_1...]
add outer pair of quotes and double quotes already in str_i.

[SEARCH str_1 str_2]
returns position in str_1 of first character contained in str_2, otherwise 0.

[SUBST str_1 str_2 str_3]
replace all occurrences of str_2 in str_1 with str_3.

[SUBSTR *str strt_pos* [*num_chars*]]
return in quotes the *num_chars* characters in *str* to the right of and including the character in position *strt_pos*.

[TRANSLATE *str* [*out_chars in_chars*]]
return the string that is the result of replacing each character in *str* that appears in the ith position in *in_chars* with the ith character in *out_chars*.

[TRIM *str* [*which_side*] [*trim_char*]]
return in quotes the result of trimming a leading or trailing sequence from *str*. *which_str* can be -Right, -Left, or -Both.

[UNQUOTE *str*]
remove one outer pair of quotes and change every pair of adjacent quotes remaining to a single quote.

[VERIFY str_1 str_2]
returns the first position in str_1 where a character has been found that is not in str_2, otherwise 0.

2.6.3. File System Functions

[ATTRIB *path option* [-BRief]]
return information about *path*. *option* can be -TYPE, -DTM, -DTB, or -LENgth (-L). -BRief suppresses some error messages.

[DIR *path* [-BRief]]
return in quotes the directory portion of *path*. -BRief suppresses some error messages.

[ENTRYNAME *path*]
return the entryname portion of *path*.

[EXISTS *path* [*type*] [-BRief]]
return TRUE if pathname *path* of type *type* exists, else FALSE. *type* can be -ANY, -FILE, -DIRectory, -SEGment_DIRectory, or -Access_CATegory. -BRief suppresses some error messages.

[GVPATH]
return the pathname of the active global variable file, if any, otherwise returns -OFF.

[OPEN_FILE *path status_var* -mode *m*]
open *path* on an available unit and return the unit number. *m* can be R, W, or WR.

[PATHNAME *rel_path* [-BRief]]
return in quotes the full pathname of *rel_path*. -BRief suppresses some error messages.

[READ_FILE *unit status_var* [-BRief]]
read a record from the file open on *unit* and return the quoted value of that record. -BRief suppresses some error messages.

[WILD $wild_1$ $wild_2$...$wild_n$ [*ctl_arg*...] [-BRief]]
list the entrynames that match $wild_i$ and *ctl_arg*. *ctl_arg* can be -BF *date*, -AF *date*, -FL, -DIRS, -SEGDIRS, and -SINGLE *unit_var*. -BRief suppresses some error messages.

[WRITE_FILE *unit text*]
strip *text* of one layer of quotes and write *text* on the file open on *unit*. Return 0 if successful; otherwise, nonzero.

2.6.4. Miscellaneous Functions

[ABbrev -EXPand *text*]
returns *text* with the abbreviations expanded.

[CND_INFO *ctl_flag*]
return information on the most recent condition on the stack. *ctl_flag* can be -NAME, -CONTinue_SWitch, and -RETurn_PerMiT.

[DATE [*ctl*]]
return date/time according to *ctl*. *ctl* can be -FULL, -USA, -UFULL, -DAY, -MONTH, -YEAR, -TIME, -AMPM, -DOW, -CAL, -TAG, -FTAG, -VFULL, or -VIS.

[GET_VAR *expr*]
return the value of the variable named *expr* if it has been defined; otherwise, $UNDEFINED$.

[QUERY *text* [*default*] [-TTY]]

on the terminal, return *text* in quotes and followed with a question mark. -TTY forces input from the terminal.

[RESCAN *str*]
return the result of stripping one level of quotes from *str* and evaluating any function calls or variable references no longer appearing in quotes.

[RESPONSE *text* [*default*] [-TTY]]
on the terminal, print *text* in quotes and followed with a colon. -TTY forces input from the terminal.

2.7. Command Descriptions

The following notations may indicated for a command:

(CF) - Also a Command Function
(EX) - External Command
(EPF) - EPF Command
(IN) - Internal Command
(LO) - Command executable when logged out
(NR) - Not Released
(OIN) - Old Internal Command (uses leftmost substring)
(OP) - Operator Command
(P2) - Can be used in PRIMOS-II
(QT) - Qualified Tool
(SA) - System Administrator only
(*revno*) - New; released at *revno*

$$ *batch-command*

Flag a command to be passed on to the batch monitor. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

ABbrev [*pathname*] [*options*]

Invoke abbreviation preprocessor. (IN)
Options:

-Change $name_1$ ...[$name_n$]
-Change_Argument $name_1$ ...[$name_n$]
-Change_Command $name_1$ ...[$name_n$]
-Change_Name *oldname newname*
-CReate
-DeLete $name_1$ ...[$name_n$]
-HELP
-LIST [$name_1$...[$name_n$]]
-OFF | -ON
-STatus
-No_Query
-No_VeriFY | -VeriFY
-WILD
-Add *name rest-of-line*
-Add_Argument *name rest-of-line*
-Add_Command *name rest-of-line*
-EXecute *rest-of-line*
-EXPand *rest-of-line*
-Expand_Execute *rest-of-line*

Ref: *PRIMOS Commands Reference Guide* [49].

ADdisk [PROTECT] $pdev_1$ [...$pdev_9$ | -RENAME *packname*]
ADdisk $packname_1$ [...$packname_9$] -ON *node-name*

Add disks to system. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

Add_Remote_ID *user-id* [password] -ON *nodename*
[-PROJect *project-id*] [-PROMPT]

Specify id for slaves on remote machines. (IN)
Ref: *PRIMENET Guide* [45].

ADMIN_LOG *logname* [*log-type*] *subcommand*

Create, list, modify, purge or delete DSM logfiles. (EPF, 21.0)

Log-type:
-Private_LOG
-System_LOG [*node* | *nodegroup*]

Subcommands:
-CReate [*attributes*]
-MODify *attributes*
-PURGE [*age* | ALL]
-DELETE
-LIST
-Help
-USAGE

Attributes:
-CYClic | LINear
-MaXimum_SiZe *records*
-MiNimum_SiZe *records*
-Warning_Level *percent*
-RETain [*days*]
-Purge_TIMe *hh:mm*

AIDS

Invoke the PRIMEAIDS system. (EX, OBS)

AMlc [Tty | TRan | TTYHs | TRANHS | TTYNop | TTYUPC | TTYHUP | TT8BIT | ASD] *line* [*config* [*lword*]]

Set AMLC line characteristics. (IN,OP)

line, *config*, *lword* are octal. Command is obsolescent, use SET_ASYNC (20.2).

config	Line Speed
2033	110 BAUD
2113	134.5 BAUD
2213	300 BAUD
2313	1200 BAUD (default)
2413	9600 BAUD (programmable clock)
2513	75 BAUD (or by jumper or ICS JUMPER directive)
2613	150 BAUD (or by jumper or ICS JUMPER directive)
2713	1800 BAUD (or by jumper or ICS JUMPER directive)

lword:

Bit	Meaning when on	Octal	Hex
1	Half duplex	100000	8000
2	No LF after CR if half duplex	040000	4000
3	XOFF/XON recognition	020000	2000
4	XOFF received, output suspended	010000	1000
5	Buffered protocol, use bit 6 for sense	004000	0800
6	If set, send XOFF on ^DTR else send XOFF on DTR	002000	0400
7	Enable error detection, send NAK on parity or overflow	001000	0200
8	Reserved	000400	0100
9-16	User number (0 => assignable)	000377	01FF

Ref: *System Administrator's Guide, Vol. II: Communication Lines and Controllers* [63] and *Operator's Guide to System Commands* [35].

ARCHIVE [-LIST] *pathname* -MT *n* -VOLID *name* [*options*]

Archive disk files onto magnetic tape. (EPF)

-LIST indicates that *pathname* contains a list of objects. *Options*:

-INDEX [*pathname*]
-IndeX_Levels [*n*] (1 <= *n* <= 99)
-LeVels *n* (1 <= *n* <= 99)
-No_Query
-VeriFY
-Tty
-REMARK [*character-string*]

-DeLete
-OWNer *user_name*
-OVerWrite
-CAtalog_PAthname *pathname*
-Cam_To_Dam
-CAtalog_PAthname *pathname*
-Compatible_VersioN *rev*
-VALidate
-HELP [{USER | OPERATOR | *option* | EXAMPLE |
ERROR [*error#*] | ERROR_LIST | WILDCARDS | HELP}]

and the standard wildcard options. Ref: *Data Backup and Recovery Guide* [7].

ARCHIVE_RELEASE -VOLID *name* [*options*]

Release a tape generated with ARCHIVE for reuse. (EPF)
Options:

-MT *n* (0 <= *n* <= 7)
-REEL *n* (1 <= *n* <= 255)
-OWNer *user_name*
-CAtalog_PAthname *pathname*
-No_Query
-HELP [USER | OPERATOR | *option* | EXAMPLE |
ERROR [*error#*] | ERROR_LIST | WILDCARDS | HELP]

and the standard wildcard options. Ref: *Data Backup and Recovery Guide* [7].

ARCHIVE_RESTORE *object-pathname* [*target-pathname*] -MT *n* [*options*]

Restore files from an ARCHIVE tape to disk. (EPF)
Options:

-VOLID *name* [*name*]...
-INDEX [*pathname*]
-IndeX_Levels [*n*] (1 <= *n* <= 99)
-REEL *n* (1 <= *n* <= 255)
-Tty
-Cam_RBF
-Dam_RBF
-From_Logical_Tape *n*
-From_Save_Number *n*
-To_Logical_Tape *n*
-To_Save_Number *n*
-MAGSAV
-WRitten_After [*date*]
-From_Save_Number *n* (1 <= *n* <= 255)
-WRitten_Before [*date*]
-To_Save_Number *n* (1 <= *n* <= 255)
-No_Query
-VeriFY
-OWNer *user_name*
-CAtalog_PAthname *pathname*
-COMBine
-REPLACE
-HELP [USER | OPERATOR | *option* | EXAMPLE |
ERROR [*error#*] | ERROR_LIST | WILDCARDS | HELP]

and the standard wildcard options. Ref: *Data Backup and Recovery Guide* [7].

ASRCWD [*number*]

Set virtual ASR control word. (IN) Ref: *PRIMOS Commands Reference Guide* [49].

ASsign *device* [-WAIT]
ASsign DISK *pdev* [-PRIority_SELect]
ASsign ASYNC -LINE *n*
ASsign AMLC *protocol amlc-line config lword*

Assign peripheral device. (IN)

device: CArdr
Cenpr
CE2pr
CR*n* (*n*=0,1)
DIsk *pdev*
GS*n* (*n* = 0..3)
MG*n* (*n* = 0..3)
PBhist
PLot
PR*n* (*n*=0..3)
Ptr
PUnch
SMLC*nn* (*nn*=00..07)
MTX -ALias MT*ldev* (*ldev*=0..7)
MT*pdev* [-ALias MT*ldev*] [*mt-options*]
(*pdev*, *ldev*=0..7)

mt-options can be:

-TPID *id*
-MOUNT
{-RINGON | -RINGOFF}
{-7TRK | -9TRK}
-RETENSION
-DENSITY *bpi* (*bpi*=800, 1600, 3200, 6250)
-SPEED *spd* (*spd*=25, 100)

protocol, *config*, and *lword* are described under the AMLC command. Ref: *PRIMOS Commands Reference Guide* [49] and *Operator's Guide to System Commands* [35].

ATM

Enter Advanced Text Management Option Selection Menu of OAS. (EX)
Ref: *OAS Word Processing User's Guide.*

ATM_ADMIN

Maintain OAS document database. (EX, OBS)
Replaced by OA_ADMIN. Ref: *OAS System Administrator's Guide* [30].

Attach [*pathname*] [*passwd*] [*ldev*] [*key*]

Attach to UFD. (IN)

ldev:
100000 - search MFDs of all started devices (*default*).
177777 - search MFD of current device.
n=0..77 - search MFD on logical device *n*.

key:
0 - attach to UFD and set home (*default*).
1 - don't set home UFD after attach to subUFD.
2 - set home UFD after attach to subUFD.
177777 - attach to UFD and don't set home.

Ref: *PRIMOS Commands Reference Guide* [49].

AUTOPSY [*filename*]

Dump analyzer. (EX/EPF, QT)
Internal commands:

Clrmap
Clears out old maps so new ones may be read in.
CHkprt
Prints a description of the last check handled.
COmsearch *address*
Searches symbol table for the common block at <address>.
Dump *start_address end_word* [*user*]
Dumps specified region of memory in octal.
DAte Displays date header for current dump.
DDqb [*start* [*end*]] [-FREE] [-USED] [-MeTeRs]
Dumps Disk Queue Blocks.
DLcb [*start* [*end*]] [-LRU_list] [-Hash_Table]
Dumps Locate Control Blocks.
DSemaphore *address* [*user*]
Dumps semaphore at *address*.
Ecbsearch *address*
Searches symbol table for the procedure with the given ECB.
From *treename*
Reads a crash dump from *treename*
FSchk
Checks file system tables for consistency.
Help [*command* | *topic* | NEW]
Displays helpful information for the selected topic. NEW displays information on the latest AUTOPSY updates.
IPCDump
Dump the inter-process communications area.
Keyprt *keys modals*
Decodes keys and modals. See 3.11.
Lprnt
Displays status of all N1LOCKs. See 4.4.
LBsearch *address*
Searches symbol table for procedure with given LB.
LBNames *address*
Lists all procedures with specified LB.
LOcsearch *address*
Searches for the symbol nearest to *address*.
Map [*maptree1* [*maptree2*]]
Reads in maps (by default, from MAPS UFD).
Othsearch *address*
Searches for other symbol (not procedure or common) at *address*.
Pdump *user*
Displays PCB (and concealed stack) for *user*. See 3.15.
PAgchk
Checks memory maps for consistency. See 3.13.1.
PAUse
Exits AUTOPSY but leaves everything in place so you can restart.

PBsearch *address*
Searches for procedure closest to specified PB.
PMap *segno user*
Prints HMAP and LMAP for specified segment. See 3.13.1.
Quit Exits AUTOPSY.
RDump [SLAVE | AP]
Dumps absolute register set, either for master (default), slave, or AP board.
Read *treename mtunit*
Reads dump into *treename* from tape unit *mtunit*.
REAL [ON | OFF]
Use real memory as opposed to a read-in dump.
REStore *segno user*
Restores given user's segment into seg 4001 and invokes VPSD.
RPrnt [[-]LIve | [-]LAst] [[-]SLave] [-R, -REGno]
Displays registers for live or last process (live is default). If SLAVE is specified, displays registers for live or last on slave ISU.
Status {*user* | ALL | US}
Displays status for specified user, all processes, or only user processes.
SYmbol *symbol* [*symbol*...]
Returns information about given *symbol*s.
Trace *user* [*address*]
Traces stack for specified user, from current SB or *address*. Trace commands are:

Father
Move to the father of this frame.
Son Move to the son of this frame.

TTybuf *user* [-INput] [-OUTput] [-User_1_Message] [-CENtronics_1] [-CENtronics_2] [-CARD_reader] [-Paper_Tape_Reader] [-Paper_Tape_Punch] [-OCTal] [-CRLF]
Displays specified terminal buffer in format indicated (default is *user*'s input and output buffers in unformatted ASCII).
Unit {*address* | *offset*} [-uNFormatted]
Displays unit table entry at *address* (or *offset* from UTCOM$).
UOwned {*address* | *offset*} [-DISP]
Displays owner (user and unit number) of specified unit, with optional display of unit table entry.
UPtime
Gives time system was running in seconds.
UTbl *user* [-uNFormatted]
Displays unit table for specified user.
UTEntry *user* {unit | -CURrent | -HOME | -INITial} [-uNFormatted]
Displays specified unit table entry.
Vpsd
Enters VPSD.
! *Primos_command_line*
Executes argument as a PRIMOS command. (Must be internal or an EPF.)

AVAIL [*partition* | -LDEV *n* | *] [-NORM]

Show disk usage statistics. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

BACKUP [-LIST] *pathname* -MT *n* -VOLID *name* [*options*]

Backup files from disk to magnetic tape. (EPF)
-LIST indicate that *pathname* contains a list of objects. *Options:*

-INDEX [*pathname*]

-IndeX_Levels [*n*] (1 <= *n* <= 99)
-LeVels *n* (1 <= *n* <= 99)
-No_Query
-Cam_To_Dam
-Compatible_VersioN *rev*
-EXpiry_Date *date*
-NO_CATalog
-No_Spawn_Disk_Reader
-Spawn_Disk_Reader
-OVerWrite
-INCremental
-VALidate
-VeriFY
-Tty
-REMARK [*character-string*]
-HELP [USER | OPERATOR | *option* | EXAMPLE |
ERROR [error#] | ERROR_LIST | WILDCARDS | HELP]

and the standard wildcard options. Ref: *Data Backup and Recovery Guide* [7].

BACKUP_RELEASE -VOLID *name* [*options*]

Release a BACKUP tape for reuse. (EPF) *Options:*

-MT *n* (0 <= *n* <= 7)
-REEL *n* (1 <= *n* <= 255)
-No_Query
-HELP [USER | OPERATOR | *option* | EXAMPLE |
ERROR [*error#*] | ERROR_LIST | WILDCARDS | HELP]

and the standard wildcard options. Ref: *Data Backup and Recovery Guide* [7] and *Operator's Guide to System Backups* [34].

BACKUP_RESTORE *object-pathname* [*target-pathname*] -MT *n* [*options*]

Restore a file from a BACKUP tape to disk. (EPF)
Options:

-VOLID *name* [*name...*]
-RECOVER
-INDEX [*pathname*]
-IndeX_Levels [*n*] (1 <= *n* <= 99)
-REEL *n* (1 <= *n* <= 255)
-Tty
-Cam_RBF
-Dam_RBF
-From_Logical_Tape *n*
-To_Logical_Tape *n*
-MAGSAV
-WRitten_After [*date*]
-WRitten_Before [*date*]
-From_Save_Number *n* (1 <= *n* <= 255)
-To_Save_Number *n* (1 <= *n* <= 255)
-No_Query
-VeriFY
-COMBine
-REPLACE
-HELP [USER | OPERATOR | *option* | EXAMPLE |

ERROR [*error#*] | ERROR_LIST | WILDCARDS | HELP]

and the standard wildcard options. Ref: *Data Backup and Recovery Guide* [7] and *Operator's Guide to System Backups* [34].

BASIC [*pathname*]

BASIC language interpreter. (EX)
Ref: *Interpretive BASIC User's Guide* [20].

BASICV [*pathname*] [-MIN]

Virtual memory BASIC. (EX)
Ref: *BASIC/VM Programmer's Guide* [4].

BASINP *pathname*

Read BASIC program from paper tape. (EX)
Ref: *Interpretive BASIC User's Guide* [20].

BATCH {-DisPlay | -STatus | SYSTEM { -START | -STOP | -PAUSE | -CONTINUE}}

Invoke BATCH monitor. (EX)
Ref: *Operator's Guide to the Batch Subsystem* [31].

BATGEN {-STATUS | -DISPLAY [*queue*]}

Query BATCH queues. (EX)
Subcommands:

BLocK {*queue* | ALL}
UNBLocK {*queue* | ALL}
CAP {*queue* | ALL}
UNCAP {*queue* | ALL}
DisPlay [{*queue* | ALL}]
STatus
FIle [*pathname*]
Quit

Ref: *Operator's Guide to the Batch Subsystem* [31].

Binary *pathname*

Open file unit 3 for binary output. (IN) Ref: *PRIMOS Commands Reference Guide* [49].

BIND [epf-name] [commands] [-DeBuG] [-$PRVSTA] [-FOPT *level*] [-FULL_REV]

EPF linker. (EPF, 19.4)
Reference: **Programmer's Guide to BIND and EPFs**. BIND subcommands are:

LOad *list-of-options-and-pathnames*
Loads a binary or runfile into the EPF currently being built. Options are: -PAge, -FOrce, or -Force_Page.

LIbrary *list-of-options-and-pathnames*
Loads a binary file from LIB. Same options as LOad.

ReLoad *list-of-options-and-pathnames*
Reloads a binary into an existing EPF, replacing an old entry of the same name. Same options as LOad.

DYNT *list-of-names*
Creates a dynamic entry for the *list-of-names*.
SYmbol *name definition* [*size*]
Creates a symbol, *name*, at the location specified by *definition*. (Default size is 0.)
ALLOCate *name size*
Allocates *size* halfwords of storage for *name*.
MAp [*map-dest*] [*map-option*]
Creates a load map. (Default is a full map without flags on the terminal.)
MAIN *ecb-name*
Changes the main entrypoint to *ecb-name*. (Default is the first entry loaded.)
HELP [*command* | -LIST]
Gives help on a command or a list of commands.
Quit Quits BIND without creating or modifying an existing run file.
FILE [*epfname*]
File the EPF as *epfname* or as the current runfile name (either the same as the first binary file loaded or that given on the command line).
Common_Warning
Turns on common size mismatch checking. (Default)
No_Common_Warning
Turns off common size mismatch warnings. Will still give error for illegal redefinition.
Resolve_Deferred_Common
Allocates space for all deferred common blocks.
COMMENT *comment*
Inserts a comment into the EPF comment field. Takes the remainder of the line. Cannot be entered on the command line.
VERSION *string*
Sets the version stamp for this EPF to *string*.
ENtryname *list-of-names* | -ALL | -NONE
Add *list-of-names* as entrypoints to the current EPF library being built. -ALL implies that all successively loaded modules will have all entrypoints added; -NONE excludes subsequent entries from being added to the entrypoint list. (Default is EN -NONE)
LibMode *library-class* [-REGister]
Generate a library EPF of the given *library-class*.
ProgMode {-NORMAL | -REGister}
Generate a program EPF (default).
Initialize_DATA [-OCTal] *value*
Initialize all uninitialized static areas with *value*. Slows down program startup.
COMPRESS
Removes data unnecessary to program execution; saves file space.
WildCard [*file-type-options*] [*verify-option*]
Allow command line processing of wildcarded pathnames using @ and +. (Default is on with all file types but not RBF.)
No_WildCard
Disallow command line processing of wild cards.
ITeRation
Allow command line iteration using parentheses. (Default)
No_ITeRation
Disallow command line iteration.
TreeWalk
Allow tree-walking. (Default)
No_TreeWalk
Disallow tree-walking.
NameGenPos *position*
Perform equal name generation from the *position*th argument. (Default = 1)
No_Generation
Do not allow name generation.
Search_Rule_VeriFY
Causes BIND to print out the full path of each file it loads.

AKLMB
Allocates KLM block for serialization.

Ref: *Programmer's Guide to BIND and EPFs* [51] and *Advanced Programmer's Guide; Vol I: BIND and EPFs* [1].

BOOT_ATTACH

Used by BOOT_SAVE/BOOT_RESTORE. (EX, P2)

BOOT_CREATE [*pathname*] [-Help] [-MT[*n*]] [-No_Query]

Makes a boot tape. (EX)
Ref: *Operator's Guide to System Commands* [35].

BOOT_IMPCODE

Used by BOOT_SAVE/BOOT_RESTORE. (EX, P2)

BOOT_RESTORE

Restore files from BRMS/BACKUP tape under Primos 2. (EX, P2)

BOOT_SAVE

Save files to BRMS tape under Primos 2. (EX, P2)

BOOT_TREE

Used by BOOT_SAVE/BOOT_RESTORE. (EX, P2)

BUILD [*component*] [-No_commands] [-DeBuG] [-From *pathname*] [-Ignore_errors] [-Verbose] [-Keep_tempfiles] [-SDI] [var_1=$value_1$ ..] [-Help]

BUILD reads a description file and brings a program (programs) up to date. (EPF, QT)
Ref: *BUILD...* [40].

CARDSPOOL

Submit a job from the card reader to an RJE site. (EX)

CBL *filename* [*CE-options*]

Low intermediate ANSI-74 COBOL compiler. (EX)
See compiler options, section 2.7.1.

CBLDML [*input-pathname* [*output-pathname*] [*error-pathname*] | *options*]

COBOL Data Manipulation Language. (EX)
Options:

-Input *pathname*
-OUTput *pathname*
-ERROR *pathname*
-DYnamic
-No_Line_Number

Ref: *DBMS Data Manipulation Language Reference Guide.*

CBLSUBS *source* [-Output *pathname*] [-List *pathname*]

CBL DBMS subschema processor. (EX)
Ref: *DBMS Data Manipulation Language Reference Guide.*

CC {*pathname* | -SOURCE *pathname* | -INPUT *pathname*} [*options*]

C compiler. (EX)

-BINARY [*pathname* | YES | no]
-NOBIG | -big
-BIT8 | -nobit8
-NOCOMPATIBILITY | -compatibility
-COPY | -nocopy
-CONVERT | -noconvert
-ERRTTY | -noerrtty
-NOFRN | -frn
-LISTING [<pathname> | YES | no | tty]
-NOEXPLIST | -explist
-NOSILENT | -silent
-statistics
-debug
-64v
-noonunit
-NOANSI | -ansi
-xref
-xrefs
-CDBG | -spldbg
-production
-NOPOP | -pop
-INTL | -ints
-psi1
-psi2
-psi3
-NOVERBOSE | -verbose
-NOCHECKOUT | -checkout
-include *pathname*
-define *name* [1 | *value*]
-NO_STORE_OWNER_FIELD | -store_owner_field
-NOUNIX | -unix
-LBECB | -pbecb
-32IX

Ref: *C User's Guide* [5].

CDML

COBOL Data Manipulation Language. (EX)
Ref: *DBMS COBOL Subschema Guide.*

Change_PassWord [*old-password*]

Change login password. (IN)
A new password is then asked for twice with echo turned off. Ref: *PRIMOS Commands Reference Guide* [49].

CHap {*-usrno* | ALL} [*priority* [*timeslice*]]

Change user priority. (IN, OP)

priority = 0..3; UP, DOWN, LOWER, -IDLE, -SUSPEND, DEFAULT (*default*=1).
timeslice is in tenths-of-a-second (*default*=3).

Defaults taken only for ALL option, else unchanged. Ref: *PRIMOS Commands Reference Guide* [49] and *Operator's Guide to System Commands* [35].

Close {*pathname* | [-]ALL | -UNIT $unit_1$ [...$unit_n$] | $funit_1$ ... $funit_n$}

Close file unit(s). (IN)
[-]ALL closes all file units above unit 1; does not close the COMO unit. Use of *pathname* from the console will close the file for all users. Ref: *PRIMOS Commands Reference Guide* [49].

CLUP [-Userno *user-number*] [-FORCE]

Cleanup processor for ROAM, PRISAM and DBMS. (EX)
Ref: *ROAM Administrator's Guide* [53].

CMPF $path_1$ $path_2$ [... $path_5$] [*option*...]

Compare ASCII files. (EX)
Options:

-MINL [*n*] (default = 3)
-BRIEF
-REPORT *report-pathname*

Ref: *PRIMOS Commands Reference Guide* [49].

CN_RBF *old-pathname new-filename* [-ALL]

Change the name of an active or inactive ROAM file. (EX)
Ref: *ROAM Administrator's Guide* [53].

CName *oldpathname newfilename*

Change name of file. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

CNVTMA *infilename outfilename*

Convert load map for PMA. (EX, OBS)
Converts load map into format usable by PSD 'LS' command.

COBOL *pathname* [*option*...]
or
COBOL [*option*...] -I *pathname* [...*option*]

Invoke COBOL compiler. (EX, OBS)
Use CBL. Options can be:

-Binary [*pathname* | NO | YES]
Define binary file generation. Default: YES.
-EXPlist
Generate an expanded listing file.
-Input *pathname*
pathname is source program.

-Listing [*pathname* | NO | YES | TTY | SPOOL]
Define listing generation. Default: YES.
-NOEXPLIST
Do not generate expanded listing file.
-64RGenerate relative-addressed code.
-64VGenerate segmented-addressed code.

COminput *pathname* [*funit*] [-PAUSE | -CONTINue [*funit*] | -TTY | -Start | -End]

Change command input stream. (IN)

```
CO -S = S;CO -CONTIN
CO -E = CO -TTY
```

Ref: *PRIMOS Commands Reference Guide* [49].

COMM_CONTROLLER {-Help | -INIT | -LOAD | -SHutdown | -UpLine_Dump} [*options*]

Control a communications controller. (21.0, EPF)
Options:

-ALL
-Dest_Node_Address {*hh-hh-hh-hh-hh-hh* | *hh-hh-hh*} (hex)
-Dest_Node_Name *node-name*
-DEVice {ICS1 | ICS2 | ICS3 | LHC | LTS}
-Device_Address *device-number* (octal)
-No_Query
-PathName *pathname*
-PRotocol *protocol*

Ref: *Operator's Guide to System Commands* [35].

COMOutput [*pathname*] [-Continue | -Pause | -End | -Ntty | -Tty]

Control routing of terminal output. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

CONCAT [*outpathname*] [*option...*]

Concatenate files. (EX)
Options:

-APPend
-BANner [*line*]
-CLOse
-COMmand (cmd mode)
-DELETE
-EJEct
-HEAder
-INSert (insert mode)
-IUNit [*n*] (dflt=1)
-NODelete
-NHEader
-NRESetp
-OPEn
-OUNit [*n*] (dflt=2)
-OVErwrite
-RESetp
-TRUncate
-VERify

Insert Mode:

Prompt char: :
Enter 1 filename or pathname per line.
Exit to command mode with a blank or null line.

Command Mode:

Prompt char: >
/* ignores rest of line
exit CONCAT with Quit command
enter 1 command per line:

BANner [*line*]	NHEader
DELETE	NREsetp
EJEct	QUIt
HEAder	RESetp
INSert [*pathname*]	TITle [*title*]
NDElete	

Ref: *PRIMOS Commands Reference Guide* [49].

CONFIG {-DATA *config-filename* | *ntusr pagdev comdev* [*maxpag* [*altdev* [*namlc* [*nphan* [*nrusr* [*smlc*]]]]]]}

Configure system. (IN, OP)
The numeric form is obsolete as of rev 20.0. Config file directives:

ABBREV YES | NO
Enables abbreviation expansion.
ALTDEV *pdev* [*records*]
Specifies the alternate paging device. *Obsolete at Rev 21; use PAGING.*
AMLBUF *line* [*ibufsz* [*obufsz* [*dmqsz*]]]
Sets AMLC buffer sizes.
AMLCLK *baudrate*
Sets the baudrate for the programmable AMLC line (4).
AMLIBL [*buffer-size*]
Sets the size of the AMLC input tumble tables.
AMLTIM [*ticks* [*disctime* [*gracetime*]]]
Sets time intervals for event timers.
ASRATE *ctrl*
Sets the console baud rate.
ASRBUF *line* [*ibufsz* [*obufsz*]]
Sets the sizes of the console terminal buffers.
ASYNC JUMPER *speed5 speed6 speed7*
Set the speeds for the last three available baud rates.
COMDEV *pdev*
Indicates the physical disk device to find CMDNC0 on.
COMDVM *pdev*
Specifies the disk to use as the mirror for *comdev* (21.0).
CONFIG *ntuser pagdev comdev* [*maxpag*] [*altdev*] [*namlc*] [*npusr*] [*nrusr*] [*smlcon*]
One line simple configuration.
DISLOG YES | NO | *line-num*
Log out users if DTR drops. *line-num* new at 21.0.
DTRDRP
Drop DTR on logout.
ERASE {*char* | *octal-val*}
Set the system-wide erase character (default is ").
FILTER
Turn on the network PDN filter (allows connections only from known nodes).
FILUNT *rsvunt maxunt* [*tount*]
Specifies number of file units. Outdated; do not use.
GO End of configuration file.
ICS CARDS *device-addr config*
Check async LAC cards in ICS2 or 3.

ICS INPQSZ *queue-size*
Set size of ICS input queues.
ICS INTRPT [*interrupt-rate*]
Set async interupt rate for ICS controllers.
ICS JUMPER *speed5 speed6 speed7*
Set the speeds for the last three available baud rates. (Obsolete at 21.0; use ASYNC JUMPER.)
KILL {*char* | *octal-val*}
Set the system-wide kill character (default is '?')
LHC *number address*
Assign logical LHC number with physical board address. (21.0)
LOGBAD YES | NO
Log failed login attempts on the console.
LOGLOG YES | NO
Allow login-over-login.
LOGMSG YES | NO
Show all logins on console.
LOGREC *sys-logging-value*
Enable system logging. (Obsoleted at 21.0 by DSM)
LOTLIM *minutes-to-login*
Set login time limit.
LOUTQM *minutes-idle-til-logout*
Set inactivity time limit for automatic logout.
MAXPAG *num-pages*
Maximum number of memory pages. (Outdated; do not use.)
MEMHLT {YES | NO}
Halt on memory ECCU.
MIRROR
Enables disk mirroring. (21.0)
NAMLC *num-assign-line-buffers*
Allocate assignable amlc line buffers.
NET ON
Start up the network (obsolete as of 19.3).
NETREC *net-logging-value*
Number of records to use for net logging. (Obsolete at 21.0)
NLBUF *num-locate-buffers*
Configure number of locate buffers.
NPUSR *num-phantom-users*
Configure the number of phantom users.
NRUSR *number-remote-users*
Configure the number of remote (network) users.
NSEG *number-total-virtual-segs*
Set maximum number of virtual segments.
NSLUSR *number-slave-users*
Configure the number of NPX slaves.
NTSABF *line in-buff-size out-buff-size xoff-lag xon-lag*
Sets buffer sizes and xon/xoff threshholds for NTS assignable lines. (21.0)
NTSASL *num-assign-lines*
Reserves buffers for assignable NTS lines.
NTSBUF *line in-buf-size out-buff-size xoff-lag xon-lag*
Sets buffer sizes and xon/xoff threshholds for NTS lines. (21.0)
NTSUSR *num-users*
Set number of NTS terminal users. (21.0)
NTUSR *number-terminal-users*
Configure the number of terminal users.
NUSEG *number-user-segs*
Set the number of segments per user (obsolete as of 19.4; use EDIT_PROFILE)

NVMFS *number-vmfa-segs*
Allocate VMFA segments.
PAGDEV *pdev* [*records*]
Indicates the disk partition for paging. *Obsolete at Rev 21; use PAGING.*
PAGING $pdev_1$ [...$pdev_8$]
Specify paging devices. (21.0)
PAGINM $pdev_1$ [...$pdev_8$]
Specify paging device mirrors. (21.0)
PRATIO *alt-dev-ratio*
Sets the ratio of how often to page to the alternate paging device. *Obsolete at Rev 21; use PRATIO command*
PREPAG *number-prepage-pages*
Specify number of pages to pre-page. (Outdated; do not use.)
PRMENG *flags*
Turns on various engineering features. REMBUF *in-buf-size out-buf-size*
Sets the size of buffers to allocate for remote users.
RWLOCK *rwlock-value*
Sets the system default file read-write lock. (Outdated; do not use.)
SMLC {ON | DSC *line strap proc recv* | CNTRLR *ctrl-num dev-adr* | SMLC*nn ctrl-num line-num*}
Turns on the smlc driver. (Obsolescent as of rev 20.0, use SYNC directives.)
SYNC CNTRLR *ctrl-num* [*dev-adr*] [*protocol*]
Enables a sync line with a specified protocol.
SYNC DSC *line strap proc recv*
Specify data set control.
SYNC ON
Turn on sync line drivers.
SYNC SYNC*nn* [*ctrl-num* [*line-num*]]}
Map logical line number to a physical line on a given controller.
SYSNAM *system-name*
Set the system name.
TPDUMP {YES | NO}
Allow tape dump before abnormal shutdown.
TYPOUT {YES | NO}
Indicates whether to echo config directives on the terminal.
UPS *ups-number*
Indicates whether an uninteruptable power supply is in use.
VPSD
Wire VPSD into memory for debugging. (Obsoleted by Ring 0 debugger.)
WIRMEM
Print out the amount of wired memory.

Ref: *System Administrator's Guide: Vol I: System Configuration* [62].

CONFIG_DSM [*options*]

Builds and edits the DSM configuration file. (EPF, 21.0)

Options:
-TTP [TTY | PT45 | PST100 | PT200]
-No_Wait
-Help [-No_Wait]
-USAGE

Ref: *DSM User's Guide* [13]

CONFIG_NET [*pathname*] [-Help] [-TTP *terminal-type*]

Network configurator. (EX)
See the *Network Planning and Administration Guide*, [27].

CONFIG_NTS [*config-pathname*] [*options*]

Configure Network Terminal Support (NTS). (EPF)
Options:

-CReate
-DisPlay
-No_Wait
-Terminal_TyPe {PT45 | PT200 | PST100 | TTY}
-EDit
-Listing [*pathname*]
-SPOOL [*spool-options*]
-LANGuage *language*

CONFIG_UM [*selection-name*] *subcommand*

Configures DSM unsolicited message handling on a system. (EPF, 21.0)

Subcommands:
-SELect [-ON *node*]
-MODify [-ON *node*]
-CANcel [-ON *node*]
-LIST [-ON *node*] [-No_Wait]
-Help [-No_Wait]
-USAGE [-No_Wait]

Ref: *DSM User's Guide* [13].

TOOLS>CONVERT_AMLC_COMMANDS {*input_file output_file* |
-HELP | -INTERACTIVE}

Convert AMLC commands to SET_ASYNC commands. (CPL)
Ref: *System Administrator's Guide, Vol. II: Communication Lines and Controllers* [63].

COPY *pathname* [*new-pathname*]
[-Copy_All | -DTM | -PROTect | -QUOTA | -RWLock] [-Save_UFD]
[-DAM | -SAM | -CAM] [-DeLete] [-INCremental | -REPLACE]
[-FORCE] [-MERGE] [-ADD] [-MXL] [-NO_CMLV] [-NO_CHECK]
[-LeVels [*n*]] [-No_Query | -Query] [-RePorT] [-DEBUG]

Disk to disk copy utility. (EPF)
Ref: *PRIMOS Commands Reference Guide* [49].

COPY_DISK [-DO_VERIFY] [-NO_BADS] [-TTY]
[-NO_RAT] [-NO_CHECKSUM]

Copy disk. (EX, OP)
Ref: *Operator's Guide to System Backups* [34].

COPY_RBF *source-pathname dest-pathname* [-DeLete] [-PROtect] [-DAM]
[-CAM] [-Min_eXt_Len] [-RePorT]

Copy an RBF file. (EX)
Ref: *ROAM Administrator's Guide* [53].

CPL *filename*

Execute a CPL file. (IN)
See section 2.5. Ref: *CPL User's Guide* [6].

CPMPC *pathname* [-PRINT] [-CR*n*] (*n*=0,1)

Punch file on card punch. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

CRASH_AUDIT -MT *n* -DUMPFILE *pathname* -OUTFILE *pathname* -MAP *pathname*

Completes a partially written security audit after a system halt. (EX).
Ref: *System Administrator's Guide, Volume III: Security & Access* [64].

CReate *ufdname* [-PassWord] [-CAT *acat*] [-MAX *n*]

Create subUFD in current UFD. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

CREATK

Build multikeyed index files. (EX)
Ref: *MIDASPLUS User's Guide* [25].

CRMPC *pathname* [-PRINT] [-CR*n*] (*n*=0,1)

Read cards. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

CRSER *pathname*

Read from serial card reader. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

CSUBS *subschema-source* [-Output *pathname*] [-List *Pathname*]

Invoke COBOL DBMS subschema. (EX, OBS)

DATE [-FULL | -USA | -UFULL | -DAY | -MONTH | -YEAR | -TIME | -AMPM | DOW | -CAL | -TAG | -FTAG | -VFULL | -VIS]

Print date and time. (IN, LO)
Ref: *PRIMOS Commands Reference Guide* [49].

DBACP

Data Base Administrator Command Processor. (EX)
Subcommands:

ALlocate {FILES | KEYS} [[OF] [SCHEMA] *schema*]
ALlocate KEY [OF] [LOCK] *lock*
[OF SUBSCHEMA *schema*]
ALLOW {AI-RECoVery | BI-RECoVery |
TRANS-ROLLback | MULTIUSERS}
[[OF] [SCHEMA] *schema*]

CHANGE KEY [OF] [LOCK] *lock*
[[OF] [SCHEMA] *schema*]
CHANGE KEYS [[OF] [SCHEMA] *schema*]
CLEAR FILES [[OF] [SCHEMA] *schema*]
CLEAR LISTing [*filename*]
DELete {FILES | KEYS} [[OF] [SCHEMA] *schema*]
DELete KEY [OF] [LOCK] *lock* [[OF]
[SCHEMA] *schema*]
DELete [SCHEMA] *schema*
DELete SUBschema {*ss-name* |
ss-num} [[OF] [SCHEMA] *schema*]
DELete SUBschemaS [[OF] [SCHEMA] *schema*]
DISallow {AI-RECoVery | BI-RECoVery |
TRANS-ROLLback | MULTIUSERS}
[[OF] [SCHEMA] *schema*]
EXPAND {AREA | CALC [OF] [RECORD] | SET}
object-name
[[OF] [SCHEMA] *schema*]
EXPAND FILES [[OF] [SCHEMA] *schema*]
LOCK [SCHEMA] *schema*
MOVE {AREA | CALC [OF] [RECORD] | SET}
object-name
[[OF] [SCHEMA] *schema*]
PACK {AREA | CALC [OF] [RECORD] } *object-name*
[[OF] [SCHEMA] *schema*]
RENAME [SCHEMA] *schema*
SAVE LISTing [*filename*]
UNLOCK [SCHEMA] *schema*
VERify {AREA | CALC [OF] [RECORD] |
KEY [OF] [LOCK] | SET}
object-name [[OF] [SCHEMA] *schema*]
VERify {AREAS | CALCS | FILES | KEYS | SETS |
FILES | SUBSchemas}
[[OF] [SCHEMA] *schema*]
VERify SCHEMAS | [SCHEMA] *schema*
VERify SUBschema {*ss-name* | *ss-num*}
[[OF] [SCHEMA] *schema*]
VERify SUBschemaS [[OF] [SCHEMA] *schema*]

Ref: *DBMS Administrator's Guide* [8].

DBASIC [*pathname*]

Double Precision Arithmetic BASIC. (EX)
Ref: *Interpretive BASIC User's Guide* [20].

DBG *filename* [-COminput | -No_COminput]
[-VeriFY_Proc | -No_VeriFY_Proc]
[-No_VeriFY_Symbols | -VeriFY_Symbols]
[-Load_State *pathname*] [-FCN]
[-Full_Initialize | -Quick_Initialize]

Source level debugger. (EX)
Subcommands:

! *primos-command-line*
pass *primos-command-line* to the PRIMOS command processor.

* [*value*]
execute command line *value* times or until an error occurs.

:{[*language-name* [,*print-mode*]] *expression* | *print-mode expression*}
evaluate *expression*. *language-name* either PL1, PASCAL, CoBoL, vRPG, Cc, MODula-2(MOD), or FORTRAN, *print-mode* either Ascii, Bit, Decimal, Float, Hex, or Octal.

ActionList Suppress | Print
Control printing of action lists.

Again
Repeat last command.

ARGumentS [*program-blk-name* [*act-num*] | *alt-entry-id*]
display value of all arguments to specified program block.

BReaKpoint [*brkpt-id*] [*act-list*] [-AFter *val*] [-BeFore *val*] [-EVery *val*] [-COunt *val*] [-IGNore | -NIGNore] [-EDit] set and modify breakpoints.

CALL *variable* [(*arg-list*)]
call a subroutine or function from the debugger command level.

CLeaR [*brkpt-id*]
clear a breakpoint or tracepoint.

CLeaRALL [*prog-blk-name* [-DeSCend]] [-BRK | -TRA]
clear all breakpoints or tracepoints in the debugging environment or all breakpoints in a specified program block.

CmdLine
enter program command line arguments.

Continue
continue program execution following a breakpoint, condition signal, or single step operation.

ENVironment [*prog-blk-name* [*act-num*]] | -POP]
define the evaluation environment.

EnvList
print current evaluation environment and contents of the evaluation stack.

ETrace {ON | ARGS | OFF}
enable and disable entry and exit tracing.

GOTO [*prog-blk-name*\[*act-num*\]]*statement-id*
modify value of execution pointer and transfer control to a specific statement when execution resumes.

HELP [-LIST | -SYM_LIST | *command_name* | *syntax_symbol*]
print debugger syntax

IF *expression act-list* [ELSE *act-list*]
conditionally execute debugger commands.

IN continue program execution until next procedure is called.

IniCmdLine
enter initialization routine command line arguments (registered EPFs only).

INFO *prog-blk-name* | *alt-entry-id* | *statement-id*
print information about program block, alternate entry to a program block, or statement.

Init_LIBrary
Initializes an EPF library.

Init_LINKage
Initializes the EPF's linkage.

iPSD
enter IPSD.

LANGuage [PLI | ForTraN | PLP | PAScal | F77 | CoBoL | vRPG | Cc | MODula-2]
specify language for expression evaluation.

LET *variable* = *expression*
assign new variable to a variable defined by the program.

LIST [*brkpt-id*]
print attributes of one breakpoint or tracepoint.

LiSTAll [*prog-blk-name* [-DSC]] [-BRK | -TRA]
print list of breakpoints and tracepoints.
LoadState *filename*
restore DBG state contained in *filename*.
MACro {*macro-name* {*command-list* | -DeLete | -EDit} | -Change_Name *old-macro-name new-macro-name* | -ON | -OFF
control macro definition and execution.
MacroList [*macro-name*]
list *macro-name* and associated command list or list all macros.
MAIN [*prog-blk-name*]
define procedure called by RESTART or print name of main program.
OUT continue execution until program block specified by execution environment pointer returns.
PAuse
temporarily suspend debugging session and return to PRIMOS command level.
PMode *print-mode* var_1 [,var_2...]
set print mode of a variable.
PSD
enter IPSD (rev 22) or VPSD (< rev 22).
PSYMbol
print table containing names of special symbols and current character values.
Quit exit to PRIMOS command level and terminate debugging session.
ReSTart [*step-command*]
start or restart execution of program. *step-command* is either STEP, STEPIN, or IN command line.
ReSUbmit
edit and resubmit last command line entered.
SaveState *filename* [-MACros] [-BReakpoints] [-TRAcepoints]
save state of DBG session to *filename*.
SEGmentS
print list of segments in use.
SouRCe *source-command* [*arg*]
examine debug source files. *edit-command* can be Top, Bottom, BRief, Verify, Print, PPrint, Where, POint, Next, MODE, Locate, Find, Symbol, PSymbol, *, EX, and NAme.
STATUS
print status information.
Step [*value*]
resume program execution for *value* number of statements. Do not include statements within called procedures.
StepIn [*value*]
resume program execution for *value* number of statements. Include statements within called procedures.
STrace {Full | Quiet | OFF}
enable or disable statement tracing.
SYMbol *symbol-name char-val*
set value of DBG character symbol.
TraceBack [-FROMR *value* [-LR]] [-F *value*] [-TO *value*] [-REV] [-DBG] [-ONU] [-ADR]
print call/return and ownership information contained in stack frames.
TRAcepoint [*brkpt-id*] [-AFter *value*] [-BeFore *value*] [-EVery *value*] [-IGNore | -NIGNore] [-COunt *value*]
set tracepoint.
TYPE *expression*
evaluate expression and print attributes of result.
UnWatch {var_1 [,var_2...] | -ALL}
remove variable(s) from watch list.
UNWIND

release user program and debugger from procedure call; unwind stack and undefine execution pointer.

vPSD
enter VPSD (< rev 22) or IPSD (rev 22).

VTrace {Full | Entry_Exit | OFF}
enable or disable value tracing.

WAtch *var*$_1$ [,*var*$_2$...]
add variable(s) to watch list and enable value tracing.

WatchList
print names of variables currently on watch list.

WHere [*segno/wordno*]
print program location or value of execution pointer.

DBG edit subcommands allowed with RESUBMIT, BREAKPOINT, and MACRO:

D	delete character.
F	specify first character.
L	specify last character.
A *text*	append text to end of line.
I *text*	insert text following character under which "I" is positioned.
O *text*	overlay text beginning with character under which "O" is positioned.
Q	return to DBG command level.

DBG internal variables:

```
$COUNT
$COUNTERS
$MR
$RTN_FUNCTION_PTR
$RTN_FUNCTION_STRUCTURE
```

Ref: *Source Level Debugger User's Guide* [57].

DBUTL

Data Base dump UTiLity. (EX)
Subcommands:

ADir [*area-name* | *area-num*]
Area {*area-name* | *area-num*} [-History]
BRief
Bucket *start* [*end*] [Octal | Ascii] [Continue]
DBK {*area-num rec-num occ bucket* | *int1 int2 int3*}
Dump CALc [*rec-name* | *rec-num*] [-History]
Dump SHared
EDit {A | R | S | SC} [*ent-name* | *ent-num*]
  subcommands:
    DEN *entry-num*
    FILe
    Next [-] *num-lines*
    POint *line-num*
    Print *num-lines* [A | D | O]
    Quit
    Replace *data*
FIX {A | S} [*ent-name* | *ent-num*]
Help [*command*]
ID {A | D | I | L | R | S} {*ent-name* | *ent-num*}
IST [[*rec-name* | *rec-num*] {*item-name* | *item-num*}]
List *list-num*
MON
NBrief
NEnt *entry-point*
Node [*node-num* | [L | P | R | S | = | >
    [*num* | *]]] [A | O | D]
ODir *occ*
Output [*filename*] [TTY]
Quit
RAM *schema-name*
RDir [*rec-name* | *rec-num*]
Record *rec-num occ bucket* [Octal | Ascii]
    [Continue]
REWind {A | R | S | SC} [*ent-name* | *ent-num*]
ROAM *schema-name*
SChema *schema-name* [-History]
SDir [*set-name* | *set-num*]
SEt {*set-name* | *set-num*} [-History]
SST [[*rec-name* | *rec-num*] {*set-name* | *set-num*}]
VERify *interval*
WHere

Ref: *DBMS Administrator's Guide* [8].

DEFine_GVar {*pathname* [-CREATE] | -OFF}

Define global variable file. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

DELAy [*min* [*max* [*width*]]]

Set terminal delay characteristics. (IN)
Can issue prior to login. Defaults are: 6, 12, 72. Ref: *PRIMOS Commands Reference Guide* [49].

DELETE *pathname* [-No_Query | -Query] [-FORCE] [-RePorT] [-DEBUG]

Delete files or directories. (EPF)
DELETE supports the wildcard convention. Ref: *PRIMOS Commands Reference Guide* [49].

DELETE_RBF *pathname* [-No_Query] [-RePorT]

Delete an RBF file. (EX) Ref: *ROAM Administrator's Guide* [53].

DELETE_VAR id_1 [...id_n]

Delete global variables. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

DELSeg {*segno* [-TO *segno*] | ALL}

Delete segment(s). (IN)
segno > '4000 Ref: *PRIMOS Commands Reference Guide* [49].

DENOTE

DEsign NOTEbook generator (SPS) (EX, QT).

DEREMER *pathname* [-GRaMmar] [-FSA] [-DEBUG] [-EXTernals] [-EXTERNALS] [-CC | -PL1 | -PL1G | -SPL | -PLP] [-No_PARser] [-No_ACTions] [-No_SR_Conflicts] [-NOERRTTY]

Parser generator. (EX, QT)
Ref: PE-T 535 [38].

DEVice_ACLs {-ON | -OFF}

Enables/disables device ACLs in DEVICE*. (IN)
Ref: *System Administrator's Guide, Volume III: Security & Access* [64].

DIAG

Diagnostic utility for PRISAM. (EX)
Ref: *PRISAM User's Guide* [50].

DISCOVER [*options*]

DBMS and PRISAM query/update/report generation tool.(EPF)
Options:

-TESt
Don't abort comi file on error (NR)

-TIMe
Produce timing messages after each command (NR)

-INITialize *tree*
Initialize shared data structures from data in *tree* (default SYSTEM>DISCOVER.CONFIG) (20.2, 21.0) (ND)

-CLeanUP
Clean up after abort (usually not needed)

-Edit_Cmd_Line
Allow ECL-style command line editing (22.0)

Ref: *DISCOVER User's Guide* [11] and *DISCOVER Reference Guide* [10].

DISCOVER_TCB

Generate Terminal Control Blocks for DISCOVER screen interface. (EX)
Ref: *DISCOVER User's Guide* [11] and *DISCOVER Reference Guide* [10].

DIsks [NOT] *pdev*$_0$ [...*pdev*$_7$]

Specify assignable disks. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

DISPLAY_LOG *logname* [*options*]

Displays messages from a DSM log. (EPF, 21.0)

Options:
-Private_LOG
-System_LOG [*node-ID*]
out-file [-No_Query]
-ForMaT {BRIEF | FULL | *format-name*}
-NOHeader
-CENSUS
-No_Wait
-PRODuct *products*
-MeSsaGe_ID *message-types*
-NODE *nodenames*
-USER *usernames*
-SEVerity *severities*
-Logged_AFter [*date/time*]
-Logged_BeFore [*date/time*]
-REMARK *text*
-Help [-No_Wait]
-USAGE

Ref: *DSM User's Guide* [13].

DISTRIBUTE_DSM [*options*]

Distributes DSM configurations. (EPF, 21.0)

Options:
-TTP [TTY | PT45 | PST100 | PT200]
-No_Wait
-Help [-No_Wait]
-USAGE

Ref: *DSM User's Guide* [13].

DLGEN

Generate a downline load file (.DL) from .DDL files. (EX, QT)

DMSTK

See DUMPSTACK.

DPTCFG *config-pathname* [-O *outpathname*]

Configure file for DPTX. (EX, OP)
Config file commands can be:

DEFINE GROUP *n options* (*n*=1,32)
Options:
-PROTOCOL SP3270
EM3270
-LINE *n* (where *n*=0,1)
-ADDRESS *nn* (where *nn*=2-digit hex)
-DEVICE n_1 n_2
($n_1 \leq n_2$)

DEFINE DEVICE *n options* (*n*=1,32)
Options:
-NAME *32-char-name*
-ADDRESS *nn* (*nn*=2-digit hex)
-ENABLE [COMMAND],[BLOCK],[WRITE],[READ]
-USER *n*
-PRINTER [VFC] [PLATEN *nn*])

Ref: *Distributed Processing Terminal Executive Guide* [12].

DPTX {-ON | -DATA *pathname* | -OFF}

Enable DPTX terminals. (EX)
Ref: *Distributed Processing Terminal Executive Guide* [12].

DPTXMTR [-TOTals] [-FREQuency *min*]
DPTXMTR -QUEUE [-FREQuency *sec*]

DPTX communications line monitor. (IN)
Ref: *Distributed Processing Terminal Executive Guide* [12].

DROPDTR

Force dropping of Data Terminal Ready. (IN, LO only)

DuMP_Segment{ [$segment_1$...$segment_{10}$]
[-Range *start-segment end-segment*] | -HELP}

Specify user segments to be dumped for partial tape dump. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

DuMp_STacK [-ALL | -BRief | -FRames*n* | -FROM*n* |
-ON_Units]

Trace user command stack. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

DuMP_User {$username_1$ [...$username_{10}$] | -HELP}

Select which users will have their segments dumped to a partial tape dump. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

ED [*filename*]

Editor. (EX) No *filename* => new file. (*str* - text string) (/ = unique delimiter not in string)
Subcommands:

.CR. = INPUT TTY
Append *str*
Append to current line.
Bottom
Go to bottom of file.
BRief
Don't display changes.
Change/str_1/str_2[/] [*n*] [G]
Change str_1 to str_2 for first occurrence on line, for all occurrences if G present, for *n* lines if *n* present.
Delete [*n*]
Delete *n* (1) lines.
Delete TO *str*
Delete to line containing *str*.
DUnload *fname* [*n*]
Unload/delete *n* (1) lines.
DUnload *fname* TO *str*
Unload/delete up to (not incl) line containing *str* to *fname*.
Erase *char*
Make *char* the erase character (").
FILe [*fname*]
Write updated file to *fname*.
Find[(*column*)] *str*
Find line with *str* starting in *column*.
Gmodify *subcommands*
Modify line with *subcommands*:

A/*str*/ - Append
B*n* - Back *n* chars
C*c* - Copy up to (not inc) char *c*
D*c* - Delete up to char *c*
E*n* - Delete next *n* chars
F - Copy to end of line
I/*str*/ - Insert *str* at curr pos.
M*n* - Copy *n* chars
Nxx - Negate criteria of cmnd xx
O/*str*/ - Overlay at current position
R/*str*/ - Retype at current position
S - Reset to start of line

INPUT[(ASR) | (PRT) | (TTY)]
Input text from specified device.
Insert *str*
Insert line (.NULL. => input mode).
Kill *char*
Make *char* new kill character.
LInesz *n*
Set max line size to *n* chars.
LOAd *fname*
Insert contents of *fname*.
Locate *str*
Locate line containing *str*.
MODE *arg*
Set editor mode. *arg* can be:

PRUPPER, PRALL, PRLOWER,
PROMPT, NPROMPT,
COUNT, NCOUNT,
NUMBER, NNUMBER,
COLUMN, NCOLUMN.

Modify /*str*$_1$/*str*$_2$[/] [G] [*n*]
Copy *str*$_2$ on top of *str*$_1$ starting with first char.

MOVe *buf*$_1$ *buf*$_2$
Move contents of *buf*$_2$ to *buf*$_1$.

MOVe *buf1 str*
Move contents of *str* to *buf1*. Buffers are EDLIN (command line), INLIN (current line to be edited), STR.1, ..., STR.10.

Next [*n*]
Advance *n* (1) lines.

NFind *str*
Find line not starting with *str*.

OUTput {DISPLAY | TTY}
Send verification output to specified device.

Overlay *str*
Overlay line with *str*. Blank leaves current char, WILD becomes blank.

PAuse
Back to PRIMOS, restart with 'S'.

POint *n*
Go to line *n*.

Print [*n*]
Print *n* (1) lines

PSymbol
Print symbols.

PTabset
Use physical tab stops of terminal when printing.

PUnch [*n*] [ASR | PTP]
Punch n lines on indicated device.

Quit Exit without filing.

Retype *str*
Replace line with *str*.

SAVE
???

Symbol *name char*
Define *name* symbol. *name*: BLANK (#), CPROMPT ($), COUNTER ( ), DPROMPT (&), ERASE ("), ESCAPE (^), KILL (?), SEMICO (;), TAB (\), WILD (!).

TAbset *tab1*...
Set tab positions.

Top Go to top of file.

Unload *fname* [*n*]
Unload *n* line into *fname*.

Unload *fname* TO *str*
Unload lines up to (but not incl) *str* to *fname*.

Verify
Display all changed lines.

Where
Print current line number.

Xeq *buff*
Execute contents of buffer.

*[*n*] Repeat *n* (until bottom or forever) times.

Ref: *New User's Guide to EDITOR and RUNOFF* [28]

EDB {*inpathname* | -Ptr | -ASR} [*outpathname* | -Ptr | -ASR]

Binary editor. (EX)
Ref: *Advanced Programmer's Guide; Vol I: BIND and EPFs* [1]. Subcommands:

BRIEF
No names printed.
Copy {*name* | ALL | <SFL> | <RFL>}
Copies up to (but not incl) specified point.
ET Copies an EOT (end-of-tape) mark to the output file. OBSOLETE.
Find {*name* | ALL | <SFL> | <RFL>}
Position to object.
GENET [G]
Copy current routine and then write out an EOT. (G) indicates copying all files, each with an EOT appended. OBSOLETE.
Insert *pathname*
Insert *pathname* into the output file.
Newinf *pathname*
Open new input file after closing old one.
Omitet [G]
Copy current routine to output file, omiting any EOT. (G) causes this to occur for all routines. OBSOLETE.
OPEN *pathname*
Open new output file after closing old one.
Quit Close all files and exit to PRIMOS.
Replac *fname pathname*
Replace *fname* with *pathname*.
RFL Insert Reset Force Load flag.
SFL Insert Set Force Load flag.
TERSE
Print 1st name in blocks.
Top Top of input file.
VERIFY
Print all names.

Ref: *Advanced Programmer's Guide, Vol.3* [3]

EDit_ACcess *target acl* [-No_Query]

Modify existing access control list. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

Edit_Command_Line *options*

EMACS like command line editor. (21.0)
Options:

-ON
-OFF
-CASE_search
-Clean_COMO
-COL_major
-COMPonent
-Edit_COMI
-ENTRY
-Error_Brief ['*text*']
-Help
-INITialize

-No_CASE_search
-No_Clean_COMO
-No_Edit_Como
-No_OBey_ERkl
-No_SHOW_hidden
-No_STACK
-No_DTICK
-No_Wild_Tail
-OBey_ERkl
-Ready_Brief ['*text*']
-Restore_HISTory *filename*
-ROW_major
-Save_HISTory *filename*
-SHOW_hidden
-SIlent
-STACK
-STICK
-Warning_Brief ['*text*']
-Wild_Tail

Ref: *Software Release Document for PRIMOS Rev. 21* (DOC10001-4PA).

EDIT_EFU [*pathname*] [-DispLay] [-HELP [*internal-cmd*]]

Edit an SNA printer form. (EX)
Ref: *PRIME/SNA Operator's Guide* [56].

EDIT_PROFILE [*pathname*] [-PROJect *project-id*] [-DEBUG]

Users' system profile editor. (EX)
Subcommands:

Add_Project [*project_id* [-PA *pa_name*] [-CReate_pa] [-LIKE *like_reference*] [-PROFile] [-SIZE *entry_count*] [-No_Query]]
Add_User [*user_id* [-LIKE *like_reference*] [-No_Query] {-PROJect | -DeFauLT [*project_id*]} [-PROFile] [-PassWord [*password*]] [-SYStem] [-Verify_NS]
ATtach_Project [*project_id*]
Change_Project [*project_id* [-PROFile] [-LIST] [-SIZE *entry_count*]] [-PA [*pa_name*]] [-LIMits]
Change_System_Administrator [*sa_name*] [-ALL]
Change_System_Defaults [-Dynamic_Segments *number*] [-Static_Segments *number*] [-LEVels *number*] [-PROGrams *number*]
Change_User [*user_id* [-PROJect [*project_id*]] [-LIST] [-PassWord [*password*]] [-SYStem]]
Delete_Project [*project_id*]
Delete_User [*user_id* [-PROJect [*project_id*]]
DeTach_Project [*project_id*]
Force_Password {-ON | -OFF}
HELP [*command_name*]
List_Project [*project_id* [-PROFile] [-OUTput *filename*] [-TTY] [-APPend] {-USER *user_id* | -ALL}
List_System {[-USers] [-GRoups] [-PROJects] | -ALL} [-OUTput *filename*] [-TTY] [-APPend] [-DETail]
List_User [*user_id* {-PROJect [*project_id*] | -ALL}]
Minimum_PassWord_Length *length*
No_Null_Password [{-ON | -OFF}]
Quit
REbuild [-PROJect [*project_id*]] [-SIZE *entry_count*]
Set_Default_PRotection [-CoNVert]

System_Defaults [{-ON | -OFF}]
Verify_User {*user_id* | -ALL}

Ref: *System Administrator's Guide, Vol. III* [64].

Eligts *tenths*

Change user eligibility timeslice. (IN)
Ref: *Operator's Guide to System Commands* [35].

EMACS [*filename*] [*options*]

EMACS screen editor. (EX) Options can be:

-Terminal_TyPe *terminal_type*
-SPEED, -BPS *bps*
-Height *num_lines*
-Width *num_columns*
-NOXOFF | -XOFF
-DEBUG
-LIBRARY
-INITIALIZE_EMACS, -IX {BUILDING | *treename*}
-ECHO_CPL
-HELP
-SUI | -SUIX
-User_LIBrary *pathname*
-No_User_LIBrary

Ref: *EMACS Reference Guide* [14]

EVENT_LOG [-NET] [-ON | -OFF]

Enable/disable event logging for system or network. (EPF, OP, OBS)
Replaced by DSM at 21.0. Ref: *Operator's Guide to System Monitoring* [36].

Expand_Search_Rules [*object*] [*options*]

Returns the full pathname of an object found by a search rule. (IN)
Options:

-Access_CATegory
-DIRectory
-FILE
-REFerencing_DIR *pathname*
-SuFfiX *suffix*
-DIRectory
-List_NAMe *listname*
-SEGment_DIRectory
-Help

Ref: *Advanced Programmer's Guide, Volume II* [2]

F77 *pathname* [*CE-option...*]

Invoke FORTRAN 77 compiler. (EX)
See Compiler options, 2.7.1. Ref: *FORTRAN 77 Reference Guide* [15].

F77DML [*input-pathname* [*output-pathname*] [*error-pathname*] | *options*]

F77 data manipulation language processor. (EX)
Options:

-Input *pathname*
-OUTput *pathname*
-ERROR *pathname*
-DYnamic
-No_Line_Number

Ref: *DBMS Data Manipulation Language Reference Guide.*

F77SUBS *source* [-Output *pathname*] [-List *pathname*]

F77 DBMS subschema processor. (EX)
Ref: *DBMS Data Manipulation Language Reference Guide.*

FAP

FORMS administrative processor. (EX)
Ref: *FORMS Programmer's Guide* [17].

FAU

File access utility for PRISAM. (EX) Ref: *PRISAM User's Guide* [50].

FDL *pathname* [*options*]

FORMS definitive language. (EX)
Options:

-INPUT {*pathname* | TTY}
-BINARY {*pathname* | YES | NO}
-LISTING {*pathname* | YES | NO | TTY | SPOOL}
-ERRLIST
-ERRTERM
-EXPLIST
-IOFLIST
-MACLIST
-OBJLIST
-REPLIST

Ref: *FORMS Programmer's Guide* [17].

FDML [*input-pathname* [*output-name*] [*error-pathname*] | *options*]

FORTRAN DML preprocessor. (EX)
Options:

-Input *pathname*
-OUTput *pathname*
-ERROR *pathname*
-DYnamic
-No_Line_Number

Ref: *DBMS Data Manipulation Reference Reference Guide.*

FED [-PROFILE *pathname*]

Forms EDitor. (EX)
Ref: *FED User's Guide* [16].

FILMEM [ALL]

Zero memory. (EX)
Note: 'ALL' => '100 - '77777 excluding PRIMOS II zeroed. Ref: *PRIMOS Commands Reference Guide* [49].

FILVER [*pathname*$_1$ [*pathname*$_2$]]

Compare binary files. (EX)
Prompts if no names entered. Ref: *PRIMOS Commands Reference Guide* [49].

FIND_RING_BREAK [-Help] [-Input *filename*]

Find ring break in local network. (EX)
Ref: *PRIMENET Guide* [45].

BATCHQ>FIXBAT [-DAYS *n*] [-QUIET] [-STARTUP {SAVE | SPOOL | DELETE | NOLOG}]

Check and fix the batch queue database integrity. (OP)
Ref *Operator's Guide to the Batch Subsystem* [31].

FIXRAT [*options*]

Fix record availability table for rev 18 disks. (EX, OBS)
Prompts:

```
FIX DISK?   (Enter Yes or No.)
UFD COMPRESSION.  (Enter Yes or No.)
PHYSICAL DISK=  (Enter pdev-see Disk Addresses.)
```

If 'OPTIONS' specified:

```
TYPE DIRECTORIES TO LEVEL  (Enter level.)
AUTO TRUNCATE DIRECTORIES NESTED TOO DEEPLY
                          (Enter Yes or No.)
TYPE FILENAMES  (Enter Yes or No.)
TYPE FILE CHAINS   (Enter Yes for disk addrs.)
```

WARNING: Do not use on Rev. 19.0 or later disks! Use FIX_DISK.

FIX_DISK {-DISK | -PDEV} *pdev* [*options*]

Disk maintenance utility. (EX)

-FIX Fix the disk

-ufd_CoMPRession
Compress the disk

-LEVEL dec Lowest level in which UFD names are printed

-MAX_nested_level dec
The max depth that UFDs are allowed to be nested

-Auto_Truncation
Automatically truncates UFDs nested too deeply

-List_File The file names are printed

-No_Quota The partition is not a quota partition

-COMmand_DEVice
The disk being fixed is the command disk

-CONVERT_19 Convert the disk to a rev 19 style disk

-CONVERT_20 Convert the disk to a rev 20 style disk

-CONVERT_21 Convert the disk to a rev 21 style disk

-DUFE Delete all unknown file entries (default)

-SUFE Save all unknown file entries

-INTeractive Interactively fix the DSKRAT

-List_BadSpots List badspots and remapping records

-TRUncate Truncate the file on error

-ADd_BADSpot oct[, oct,]
Add badspot(s) to disk

-number_of_retries dec
Modify the number of retries

-numrty dec Modify the number of retries

-FAST Perform fix checking last two data records

-CHECK Determine if partition has been shutdown properly

-All_Controller Convert a rev 21 disk to be used on all controllers

-Intelligent_Controller
Convert a rev 21 disk to be used only on intelligent disk controllers

-Dump_DBS Dump the DBS file of a rev 21 disk

-Disk_Type Specify the disk model type for a -CONVERT_21 option

-Override_Default_Interleave
Override default allocation interleave

-Restore_Default_Interleave
Restore default allocation interleave

-MINimum_extent_SIZe
New value for the minimum extent size on partition

-MAXimum_extent_SIZe
New value for the maximum extent size on partition

Ref: *Operator's Guide to File System Maintenance* [32].

FSUBS *source-filename* [-Output *pathname*] [-List *pathname*]

Invoke FORTRAN DBMS subschema. (EX)
Ref: *DBMS Data Description Language Reference Guide* [9].

FTGEN

FTS system administrator utility. (EX)
Ref: *Network Planning and Administration Guide* [27].

FTN [-Input] *pathname* [*options*]

FORTRAN-66 compiler. (EX)
Input/output options can be:

-Binary [*pathname* | YES | NO] Specify binary file creation.
-ERRlist Generate errors-only listing.
-ERRty Print errors on terminal.
-Explist Generate expanded listing.
-Listing [*pathname* | NO | SPOOL | TTY | YES] Specify listing file creation.
-NOErrtty Don't print errors on terminal.
-NOXref Do not generate a cross reference listing.
-XREFL Produce a cross reference, implies -L YES.
-Xrefs Produce an abbreviated cross reference listing.

Memory usage options can be:

-BIG Handle arrays larger than one segment.
-Debase Conserve loader base areas.
-DYnm Dynamically allocate storage for local variables.
-Fp Generate floating point skip instruction.
-FRn Floating round before store.
-INTS Assume integers are INTEGER*2.
-Intl Assume integers are INTEGER*4.
-NOBIG Don't allow arrays to span segment boundaries.
-NOFp Don't generate floating point skip instruction.
-NOFRn Don't perform floating round before store.
-Pbecb Allocate entry control block in procedure frame.
-SAve Statically allocate storage for local variables.
-64v Generate 64V-mode code.
-64R Generate 64R-mode object code.
-32r Generate 32R-mode code.

Compiler operation options can be:

-Opt Perform conservative do-loop optimization.
-STdopt Perform standard optimizations.
-Uncopt Perform unconditional do-loop optimization.

Debugging options can be:

-DCLVAR Flag all undeclared variables.
-DEBUg Allow full use of DBG.
-NODclvar Don't flag undeclared variables.
-NODEBUG Allow no use of DBG.
-Notrace Don't generate code for trace output.
-PRODUCTION
Allow production mode debugging.
-Spo System programmer option.
-Trace Generate code for trace output.

Device codes for Input, Listing, Binary:

```
0 - None              4 - Line Printer
1 - ASR               5 - Magtape
2 - PTR/PTP           6 - Cassette
3 - Card reader       7 - Disk
```

Ref: *FORTRAN Reference Guide* [18].

FTOP {-HELP [*subject*] | *server-option* | *manager-option*}

File transfer service operator command. (EX, OP)

Server-option:
-Abnd_SrVr *server-name*
-Abrt_SrVr_Link *server-name link-number*
-List_SrVr_Sts [*server-name*]
-STart_SrVr *server-name*
-SToP_SrVr *server-name*
Manager-option:
-STaRt_MnGr [*manager-name*]
-SToP_MnGr

Ref: *PRIMENET Guide* [45].

FTR {*source-file* [*dest-file*] [-Dstn_Site *sitename*]
[-Dstn_User *username*] [-DEVice *dev-name*] [-HOLD]
[-LOG *pathname*] [-NAme *ext-name*]
[-Src_Site *sitename*] [-Src_User *username*]
| {-ABORT | -CANCEL | -DISPLAY | -HOLD | -RELEASE}
requestname
| -STATUS [*requestname*]
| -MODIFY *requestname* [*FTR-options*]

File Transfer Request. (EX)
Ref: *PRIMENET Guide* [45].

FUTIL [-NORM]

File system utility. (EX)
Subcommands:

Attach *pathname* ('*' => home ufd)
CLEAN *prefix* [*level*]
Copy *file* [*newname*] [,*file* [*newname*]]...
COPYDam *file* [*newname*] [,*file* [*newname*]]...
COPYSam *file* [*newname*] [,*file* [*newname*]]..
CReate *ufdname* [*owner* [*nonowner*]]
DELETE *file* [,*file*]...
FOrce ON or OFF
Listf [*level*] [First] [LISTFIL] [PROtect] [Size]
[Type] [Date] [Rwlock] [PAsswd]
LISTSave *filename* [*options as for Listf*]
Protect *file* [*owner* [*nonowner*]]
Quit
Scan *file* [*options as for Listf*]
SRwloc *file lockno*
To *pathname*
TRECpy *ufd* [*newname*] [,*ufdname* [*newname*]]...
TREDEL *ufdname* [*ufdname*]...

TREPro *ufdname* [*owner* [*nonowner*]]
TRESrw *ufdname lockno*
UFDCpy
UFDDEL
UFDPro [*owner* [*non-owner*]]
UFDSrw *lockno level*
lockno:
0 use system read/write lock (SYS)
1 n readers or 1 writer (W/NR)
2 n readers and 1 writer (1WNR)
3 n readers and n writer (NWNR)

Ref: *PRIMOS Commands Reference Guide* [49].

GENERATE_CATALOG -MT *n* [*options*]

Generate/validate a BRMS tape catalog. (EX)
Options:

-CAtalog_PAthname *pathname*
-No_Query
-OWNer *user-id*
-REEL *n*
-VALidate
-VOLID *volume-id*
-HELP [*subject*]

Ref: *Operators Guide to System Backups* [34].

HDXSTAT

Display status of half duplex network. (EX)
Ref: *PRIMENET Guide* [45].

HELP [*command-name* | *topic-name*]

Access on-line information about commands. (EPF) Ref: *PRIMOS Commands Reference Guide* [49].

HISTORY

Generate program history (SPS). (EX, QT)

HPSD

High PSD. (EX)
SA, EA = 147760, 156552. Start of initial P counter = 150000. For internal commands, see PSD.

IDBMS [-CONFIG]

Initialize DBMS. (EX, OP).
Ref: *DBMS Administrator's Guide* [8].

INFO

Enter INFORMATION. (EX)

INFORM

INstruction FORMatter for PLP programs(SPS). (EX, QT)

BATCHQ>INIT [-ReSeT_Queues] [-ADMINistrator *user*]

Initialize the BATCH data base. (EX, OP)
The -ADMIN option may be used several times on the command line. Ref: *Operator's Guide to the Batch Subsystem* [31].

Initialize_Command_Environment

Reinitializes user command environment. (IN, 19.4)
Ref: *Programmer's Guide to BIND and EPFs* [51].

Input *pathname*

Open file unit 1 for input. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

IPSD, IPSD0, IPSD16

Enter I/IX mode symbolic debugger. See PSD for commands.

IROAM [-COLDSTART]

Unwind incomplete transactions and initialize ROAM shared memory. (EX)
User 1 or .ROAM_ADMIN group only. Ref: *ROAM Administrator's Guide* [53].

JOB [*pathname*] | [*job-id*] [*option*]

Submit batch job. (EX)
Monitor job options can be:

-STATUS
-DISPLAY

Job control options can be:

-ABORT
-CANCEL
-CHANGE

Operator options can be:

-ABORT
-CANCEL
-HOLD
-RELEASE
-RESTART

Submit job options can be:

-ACCT *information*
-ARGS *cplargs*
-CPL
-CPTIME *seconds*
NONE
-ETIME *minutes*
NONE
-FUNIT *number*
-HOME *pathname*
-PRIORITY *value*

-QUEUE *queuename*
-RESTART YES
NO

Ref: *PRIMOS Commands Reference Guide* [49] and *Operator's Guide to the Batch Subsystem* [31].

KBUILD

Build keyed-index file. (EX)
Ref: *MIDASPLUS User's Guide* [25].

KIDDEL

Delete records in keyed-index file. (EX)
Ref: *MIDASPLUS User's Guide* [25].

LABEL MT*n* [*options*]

Create magnetic tape label. (EX)
Options:

-TYPE A | B | E
{-VOLume | -VOLid | -VOLser} *vol-id* (1-6 chars)
-OWNer *owner* (1-14 chars)
-ACCESS *access* (1 char)
-HELP
-INIT
-OVERWRITE

Types are A (ASCII - ANSI), B (BCD - IBM) or E (EBCDIC - IBM). Ref: *Magnetic Tape User's Guide* [24].

LATE

Defer command execution. (EX) Prompts for time in the form HHMM. Ref: *PRIMOS Commands Reference Guide* [49].

LD [*pathname*] [*options*]

List file characteristics. (EX)
Options:

-BRief
-CATegory_Protected [*acat-name*]
-DeFauLT_Protected
-DETail
-DTA
-DTB
-DTC
-DTM
-HELP
-No_Column_Headers
-No_HEader
-No_SORT
-No_Wait
-PROtect
-ReVerse
-SinGLe_COLumn

-SIZE (uses 1K record size)
-SORT_dtA
-SORT_dtB
-SORT_dtC
-SoRT_Dtm, -SORTM
-SORT_Name, -SRTN
-SPECific_Protected
-WIDE

and wildcard options. Ref: *PRIMOS Commands Reference Guide* [49].

LEM {*rbf-filename>subfile-number* | BIFILE}

List extent map of a CAM file. (EX)
Ref: *ROAM Administrator's Guide* [53].

LISP [-INPUT_FILE *source_pathname*
[-OUTPUT_FILE *output_pathname*
[-ERROUT *error_pathname*]]]
[-DYNAMIC *number_of_segments* -RESERVED *number_of_segments*]

Invokes the Prime Common LISP Interpreter/Compiler (EPF).
Ref: *PRIME Common LISP Language Reference Manual* [22] and *PRIME Common LISP Environment Reference Manual* [21].

Listf

List files in current UFD. (IN, 19.4-CPL/EPF)
Ref: *PRIMOS Commands Reference Guide* [49].

Listing *pathname*

Open file unit 2 for listing output. (IN, 19.4-CPL)
Ref: *PRIMOS Commands Reference Guide* [49].

List_ACcess [*object*]

List access rights. (IN)
Ref: *PRIMOS Commands Reference Guide* [49] and *Prime User's Guide* [47].

LIST_ASSIGNED_DEVICES [*device-names* |
-USER [*user-names* | *user-numbers*]]
[*general SIM options*]

Lists assigned devices on the system. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_ASYNC [*line-numbers*
-USER [*user-names* | *user-numbers*]]
[*general SIM options*]

Displays the status of any or all asynchronous lines. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_CATALOG

List contents of archive or backup tape(BRMS). (EX)
Ref: *Operator's Guide to System Backups* [34].

LIST_COMM_CONTROLLERS [*general SIM options*]

Displays information on comms controllers on the network. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_CONFIG [*directive-names*] [*general SIM options*]

Displays the various values of system variables. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_DISKS [*disk-names*] [-USERS] [-LOCAL] [-REMOTE] [*general SIM options*]

Displays information for local and remote disks. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

X.LIST_DISKS [*disk-name*] [-ON *system*] [-SIZE] [-SYStem *system* | -LOCAL] [-DETail]

List disk status. (EPF, NR)

List_DuMP [-HELP]

List the current values for a partial tape dump. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

List_Epf [$pathname_1$...$pathname_8$] [-ACtive | -Not_Active] [-Not_Mapped] [-PRoGram] [-LIbrary] [-SEGmentS] [-Command_Processing] [-Epf_Data] [-DETail] [-No_Wait] [-Help]

Display information about EPF mapped in. (IN, 19.4)
Ref: *Programmer's Guide to BIND and EPFs* [51].

List_Group

List ACL groups. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

LIST_LAN_NODES [*lan-names*] [-HOST] [-LTS] [*general SIM options*]

Displays all nodes on LAN300 networks. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*

-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_LHC_STATUS [*options*]

Show status of LHC300 controllers. (EPF)
Options:

-Dest_Node_Name *node-name*
-Dest_Node_Address *node-address* (hex pairs)
-Dest_LHC_number *lhc-number*
-Lan_Name *lan-name*
-Help
-PERFormance
-CONNection *connection-type*
-ManaGeMenT
-ALL
-No_Wait

Ref: *PRIMENET Planning and Configuration Guide* [46] and *NTS Planning and Configuration Guide* [29].

List_Library_ENTries [$pathname_1 ... pathname_8$] [-ACtive | -Not_Active] [-No_Wait] [-Help] [-ENtryname $entry_1 ... entry_8$] [-Not_Mapped]

List entrypoints in EPF libraries. (IN, 19.4)
Ref: *Programmer's Guide to BIND and EPFs* [51].

List_Limits

List the limit of segments and program invocations/levels authorized. (IN, 19.4)
Ref: *Programmer's Guide to BIND and EPFs* [51].

LIST_LTS_STATUS [i<options>]

Show status of LAN Terminal Servers. (EPF)
Options:

-Dest_Node_Name *node-name*
-Dest_Node_Address *node-address* (hex pairs)
-Help
-PERFormance
-CONNection *connection-type*
-ManaGeMenT
-ALL
-No_Wait

Ref: *NTS Planning and Configuration Guide* [29].

LIST_MEMORY [*user-names* | *user-numbers*] [-TYPE *user-types*] [*general SIM options*]

Displays memory usage per user process. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]

-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

List_Mini_Commands [*command_match*]

Display the available commands at mini-command level. (IN, 19.4)
command_match is a command name that may contain wildcards. Ref: *Programmer's Guide to BIND and EPFs* [51].

LIST_PRIMENET_LINKS [*node-names* | PDN *names*]
[-LINK *link-devices*]
[*general SIM options*]

Displays the status of PRIMENET links. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_PRIMENET_NODES [*node-names*]
[-LINK *link-devices*]
[*general SIM options*]

Displays all PRIMENET configured remote nodes. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_PRIMENET_PORTS [*port-numbers*]

[-USER *user-names* | *user-numbers*]
[*general SIM options*]

Displays a system's port assignments. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

List_Priority_ACcess [*disk-name*]

Show any priority acls on a disk. (IN)
Ref: *PRIMOS Commands Reference Guide* [49] and *System Administrator's Guide, Vol. III* [64].

LIST_PROCESS [*user-names* | *user-numbers*]
[-PROJect *project-groups*]
[-TYPE *user-types*]
[-DETail]
[*general SIM options*]

Displays the environment of a specified user. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

List_Quota [*pathname*] [-BRief]

Show quota and current usage on a directory. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

LIST_RBF *treename* [-DETAIL] [-SIZE]

List attributes of an RBF file. (EX)

List_Remote_ID [-ON *nodename*]

List all of the remote IDs for this user. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

List_Search_Rules

List all of the search rules in effect for user. (IN, 19.4)
Ref: *Programmer's Guide to BIND and EPFs* [51].

List_Segment [*segno*$_1$...*segno*$_8$] [-STatic] [-DYnamic] [-BRief] [-No_Wait] [-Help] [-NAME]

Show segments in use for user. (IN, 19.4)
Ref: *Programmer's Guide to BIND and EPFs* [51].

LIST_SEMAPHORES [*semaphore-numbers*] [-USER *user-numbers* | *user-names*] [-TYPE {NAMED | NUMBERED}] [*general SIM options*]

Displays the value of all in-use semaphores. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_SYNC [*line-numbers*] [*general SIM options*]

Displays the configuration of all enabled synchronous lines. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_TAPE

List the contents of an archive/backup tape(BRMS). (EX)
Ref: *Operator's Guide to System Backups* [34].

LIST_UNITS [*user-names* | *user-numbers*]
[-PATHNAME *pathname-prefix*]
[*general SIM options*]

Displays information relating to files, units and attach points. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LIST_USERS [*wild-user-name*] [-USers] [-SerVers] [-PHantoms]
[-BATCH] [-SLaves] [-ALL_Disks]
[-PROJects] [-Disks *disk-name*]

Display the current users on a system. (EPF)
Ref: *PRIMOS Commands Reference Guide* [49].

LIST_VAR [*wild-card-name...*]

List CPL global variables. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

LIST_VCS [*VC-ID-numbers*]
[-USER *user-names* | *user-numbers*]
[-NODE *node-names*]
[-LINK *link-devices*]
[-PORT *port-numbers*]
[*general SIM options*]

Displays the state of virtual circuits. (EPF, 21.0)

General SIM options:
-Help [-No_Wait]
-USAGE
-ON *node* | *nodegroup*
-Private_LOG *pathname* [-Ntty]
-System_LOG *pathname* [-Ntty]
-No_Wait
-FREQ *integer*
-TIMES *integer*
-START *date/time*
-STOP *date/time*

Ref: *DSM User's Guide* [13].

LOAD

R- and S-mode linker. (EX)
Subcommands:

ATtach [*ufd*] [*password*] [*ldisk*] [*key*]
key=0=>don't set home, 1=>set home.
AUtomatic [*n*]
Linkareas of length *n* around module. n = 0 turns feature off.
CHeck [*symbol*] [$par_1...par_9$]
COmmon *address* Set COMMON TOP - 1
DC [END]
ENtire *pathname*
ERror [*num*]; *num* = 0, 1, or 2
EXecute [*a*] [*b*] [*x*] Uses START entry
FOrceload *pathname* [*addr*] [*linkstart*] [*linkrange*]
F/ Force prefix for FO, LO, LI commands.
HArdware *definition*

177700 Must be zero
000040 1=>Prime 400 instruction set
000020 Unused
000010 1=>Double prec. fl. pt.
000004 1=>Single prec. fl. pt.
000002 1=>Prime 300 instruction set
000001 1=>High speed arithmetic

INitialize *pathname* [*addr*] [*linkstart*] [*linkrange*]
Resets everything and loads *pathname*.
LIbrary [*pathname*] [*addr*]
Loads binary from LIB; default is LIB>FTNLIB.
LOad *pathname* [$barea_1$]...[$barea_8$] *pathname* [$barea_1$]...[$barea_9$] *pathname symbol* [$barea_1$]...[$barea_9$]
MAp [*pathname*] [*option*]
Create a load map. Default *pathname* is $F. *option* = 0=>full map, 1=>load state, 2=>load state and link info, 3=>unresolved references, 4=>same as 0, 5=>system programmer map, 6=>sorted unresolved references, 7=>sorted full map, 10=>symbol map for PSD.
MOde [D32R | D64R | D16S | D32S | D64V | D32I]
P/ Page boundary prefix for FO, LO, LI commands.
PAuse
PBrk [*symbol*] [$par_1...par_9$] * *par1* [$par_2...par_9$]
QUit Back to PRIMOS
RR
SAve *pathname* [*a* [*b* [*x* [*keys*]]]]
SEtbase [*linkstart*] [*linklen*] * [end of sector] (*=>current sector)
SS *symbol*
SYmbol *symbol oldsym* [$par_1...par_6$] *symbol addr* [$par_2...par_6$] *symbol* * [$par_1...par_3$]
parameters can contain + and - signs
SZ [NO | YES]
VIrtualbase *linkstart tosector*
XPunge [*y*] [*z*] *y*: 0=>all but undefined symbols, 1=>all but undefined and COMMON. *z*: 0=>all defined base areas, 1=>all but sector 0, 2=>return all.

Ref: *SEG and LOAD Reference Guide* [23].

LOGIN [*username* [-ON *nodename*]] [-PROJect *project*]

Login to system. (LO)
Ref: *PRIMOS Commands Reference Guide* [49].

LOgout [-*usrno* | ALL]

Logout user. (IN, OP)
usrno must have same login name as user unless issued by System user. Ref: *PRIMOS Commands Reference Guide* [49].

LOGPRT [*outfile*]
[LOGLST | Tty] [-Help]
[-From [*mmddyy hhmm*]]
[-Input *pathname*]
[-Type {Cold | Warm | Timdat | CHecks | Disk | DSKnam
Overfl | Shutdn | CHK300 | Par300 | Mod300
TYPE10-TYPE15 | REMARK | POWERF]
[-Spool]
[-COntin]
[-DBug]
[-Census]
[-Remark]
[-Dump]
[-Delete]
[-PURGE]

Print LOGREC. (EX, OP, OBS)
Prompts for input pathname, default (just .CR.) is CMDNC0>LOGREC. Replaced by PRINT_SYSLOG and PRINT_NETLOG at rev 20.0. Ref: *Operator's Guide to System Monitoring* [36].

LON [-ON | -OFF]

Control logout notification. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

LOOk [-*usrno*] [*segno* [*access* [*mapseg*]]]

Map segment to user 1. (IN, OP)
Defaults are 1 6000 200 4001. Ref: *Operator's Guide to System Commands* [35].

LOOPBACK [*source-options*] *dest-options*
[-LAN_NAME *lan-name*] [-HELP]

Check network integrity. (EX)
Source-options:

-Src_Node_Name *node-name*
-Src_LHC_number *lhc-number*
-Src_Node_Address *node-address*
-Src_Lbk_Layer {NMSR | NME}

Dest-options:

-Dest_Node_Name *node-name*
-Dest_LHC_number *lhc-number*
-Dest_Node_Address *node-address*
-Dest_Lbk_Layer {NMSR | NME}

MAGNET [-SILENT] [-USER | -OPeRator] [-OVERWRITE]

Transfer data to and from tape. (EX)
Ref: *Magnetic Tape User's Guide* [24].

MAGRST [-7TRK] [-TTY] [-QUERY] [-Cam_RBF] [-Dam_RBF]

Magtape restore. (EX)

Example:
TAPE UNIT: 0 - 7
ENTER LOGICAL TAPE NUMBER:
logical tape number or 0 if positioned
READY TO RESTORE:
Yes
No
Partial
$I [*filename*] [*level*] (turn on indexing)
$A *ufd* [*passwd*] [*ldev*] [*key*] (attach to ufd)
NW [*filename*] [*level*] (index only)
TREE NAME:
pathname per line. End with null line.

Note: does not save DTA or DTC. Ref: *PRIMOS Commands Reference Guide* [49] and *Magnetic Tape User's Guide* [24].

MAGSAV [-7TRK] [-INC] [-UPDT] [-VAR] [-P300] [-Cam_To_Dam] [-Save_UFD] [-TTY] [-No_Acl] [-NO_RBF] [-REV19]

Magtape save. (EX)

TAPE UNIT: *n* (where $0 \leq n \leq 7$)
ENTER LOGICAL TAPE NUMBER: 0 -- tape already positioned
-or- *n* -- *nth* logical tape
TAPE NAME: *6-character name*
DATE: *mm dd yy* or
CR for today's date
REV NO: *an arbitrary integer*
NAME OR COMMAND:
pathname to save an object (file or dir)
$A *ufd* [*passwd*] [*ldev* [*key*]]
attach to ufd
$Q terminate tape and return to PRIMOS
$R terminate tape, rewind, and return to PRIMOS
$I [*filename*] [*level*] print index to indicated level
$UPDT [ON | OFF] set dumped switch (OFF default)
$INC [ON | OFF] include only items with a set dumped switch
OLD [ON | OFF] create old partition format
$VALID [ON | OFF] check for conformance to new file name rules
MFD save entire disk (must be attached to MFD)
* save current directory

Ref: *PRIMOS Commands Reference Guide* [49] and *Magnetic Tape User's Guide* [24].

MAIL [*filename*] [*username* [*username*...] | !*file*]
[*send-options*] [*options*]

Send and receive PDNMail (EPF, NR)
Options:

-List List headers.
-Help
Display usage info.
-Into Force interactive mode.
-Delete_Mailbox
Deletes the user's mailbox.
-Set_Forward_to *user-address*
Forward mail to another user/node.
-Cancel_Forwarding
Cancel mail forwarding.
-Xmail
Do not collect X.MAIL.
-Alias *filename*
Use *filename* for user aliases.
-To *user-addresses*
Users to send mail.
-File *filename*
File to send as mail.
-Subject *subject*
Set subject field.
-CErtify
Certify receipt of mail.
-CC *user-addresses*
Send 'carbon copies' to *user-addresses.*

User-address is of the form:

Local user (same machine): *username*
User at other company machine: *username@machine-name*
User at x.mail site: *username@xmail-site*.XMAIL

Ref: *PDN Mailer User's Guide* [41]

X.MAIL [*user* [*pathname*] [-ON *nodename*]] [*options*]

Send and receive mail. (EX, NR)
To send mail, *user* must be specified. Terminate mail with a $ or a ctrl-C. To print mail, use no options. *Options*:

-Check
Reports mail availability
-List List headers only
-LFirst
List headers and first lines
-Append *pathname*
Append current mail to a file *pathname*
-Spool *formtype*
Spool mail to printer using *formtype*
-Held
List all held mail by username

-SRTM
Sort by amount held.

-NSRT
Sort by name.

-ON *node*
Perform action on *node*.
-PORT *n*
Use x.25 port *n*.
-Users
Provide list of users.
-NW
Don't paginate.
-NQ Don't query about mail being read.

MAKE -DiSK *pdev* -PARTition *name* -Disk_Type *disk_type* [*options*]

Format disk. (EX)
options:

-Disk_Type *disk_type*
Specifies what kind of disk. Valid types are:

SMD	80MB or 300MB removable
CMD	cartridge module device
68MB	68 megabyte fixed media
158MB	158 megabyte fixed media
160MB	160 megabyte fixed media
600MB	600 megabyte fixed media
MODEL_4475	315 megabyte fixed media
MODEL_4711	60 megabyte fixed media (rev 21)
MODEL_4715	89 megabyte fixed media (rev 21)
MODEL_4714	120 megabyte fixed media (rev 21)
MODEL_4719	258 megabyte fixed media (rev 21)
MODEL_4735	496 megabyte fixed media (rev 21, pickeral)
MODEL_4845	770 megabyte fixed media (rev 21, beluga)
FLOPPY	floppy disk (diskette, OBSOLETE as of 21)

-SPLIT [*#-of-paging-records*]
Make part of the partition for paging. If number of paging records is not given, MAKE will print the total number available and ask for number of paging records.
-PRE_rev19 Create a pre-rev 19 partition.*
-BADspot_LEVel *bad-spot-checking-level*
Checking level can be from 0 to 4 inclusive. If level 0 is specified, no checking is done. Level 4 gives the best checking. The default is 1 for SMD or CMD, 4 for fixed media disks.
-BAUD_rate *valid-baud-rate*
Set initial baud rate of system console. Valid baud rates are: 110, 300, 1200, or 9600. The default is 300.
-NO_INIT Do not initialize the file system part of the disk. Unless this is specified, the records are initialized.
-ForMaT Write hardware formats on the disk. Use this only if the disk has never been used on a Prime system.
-map_UNCORR
Map out only records with uncorrectable errors. Default is map out all records with any error-uncorrectable or correctable. Use of this option is not recommended.
-Query_BADSpots
Query user for known bad spots on disk.

-NEW_DiSK Suppress the attempt to read the old badspot file.
-CoPY_badspots_by_NAMe *partition*
Copy the badspots from the disk specified by the name *partition*.
-CoPY_badspots_by_DEVice *copy-pdev*
Copy the badspots from the disk specified by the device *copy-pdev*.
-DiSK_REVision {18 | 19 | 20 | 21}
Specify which revision of disk to make. Most recent rev is assumed. (20.0)
-Override_Default_Interleave
Override default interleave. (21.0)
-NO_FLaw_MaP
Disable the usage of the flaw map. (21.0)
-All_Controller
Create a compatible disk, for all controllers (21.0)
-Intelligent_Controller
Create a mirrorable, dynamic badspotting disk (21.0)
-ROBust Create a robust partition. (22.0)
-MIN_extent_SIZe
Specify minimum extent size for a CAM file; default is 64 for robust partition, 16 for normal. (22.0)
-MAX_extent_SIZe
Specify maximum extent size for a CAM file; default is 256 for robust partition, 32 for normal. (22.0)

Ref: *Operator's Guide to File System Maintenance* [32].

MAXSch *n*

Set scheduling constant. (IN, OP)
Default value is 3. Ref: *Operator's Guide to System Commands* [35].

MAXusr [*number* | ALL]

Limit number of logged-in users. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

MCLUP

Midas cleanup utility. (EX, OP, OBS)
Ref: **MIDAS Reference Guide**.

MDUMP

Utility for recovering MIDAS files. (EX)
Ref: *MIDASPLUS User's Guide* [25].

MED_SPOOL

Spool a MEDUSA plot file. (EX)

MEDCONFIG [*project-name*]

Medusa system configurator. (CPL, EX)

MEDUSA [*workstation-directory*]

Medusa graphics design program. (CPL, EX)

Message [-*usrno* | *username* | ALL] [[-]NOW] [-FORCE]
[-ON *nodename*]
[-DEFER | -REJECT | -ACCEPT |
-STATUS [*usernum* | *user-id* | ME]]

Send message to user(s) or system. (IN)
Enter message on next line. Ref: *PRIMOS Commands Reference Guide* [49] and *Operator's Guide to System Commands* [35].

Mirror_OFF $pdev_1$ $pdev_2$ [*options*]

Shuts off disk mirroring. (IN, OP)
Options:

-SHUT_BOTH
-SHUT_PRIMARY
-SHUT_SECONDARY
-FORCE

Ref: *Operator's Guide to File System Maintenance* [32].

Mirror_ON $pdev_1$ $pdev_2$ [*options*]

Turns on disk mirroring. (IN, OP)
Options:

-NO_QUERY
-PRIority_SELect
-HELP

Ref: *Operator's Guide to File System Maintenance* [32].

MODULA *pathname* [*CE-options*]

Modula-2 compiler.(NR)
See the compiler options, 2.7.1. Ref: *Modula-2 Reference Guide* [26].

MONITOR_NET [*options*]

Monitor Primenet. (EX)
Options:

-Ring [D]
-Sync [*line-number*]
-Virtual
-PERIOD *seconds*
-LANGuage *language*
-TIMES *repeat-count*
-Reset_Day
-Reset_Hour
-Input *filename*
-OUTput *filename*
-TRace
-Terminal_TyPe *terminal*
-Help
-RePorT *filename*

Ref: *PRIMENET Guide* [45].

MONITOR_RING

Monitor ring network. (EX, OBS)
Obsolete as of 19.4. Use MONITOR_NET. Ref: *PRIMENET Guide* [45].

MPACK

MIDASPLUS file packing utility. (EX)
Ref: *MIDASPLUS Reference Guide* [25].

MPLUSCLUP [-USER *user-number* | -ALL]

MIDASPLUS cleanup utility. (EX)
Ref: *MIDASPLUS Reference Guide* [25].

MRGF p_1 p_2 [...p_5] -OUTF p [-MINL [n]] [-BRief] [-FORCE] [-REPORT *pathname*]

Merge ASCII files. (EX) MRGF edit commands:

A insert all differing lines in p_1
B insert all differing lines in p_2
C insert all differing lines in p_3
D insert all differing lines in p_4
E insert all differing lines in p_5
A*n* insert line *n* of p_1
E*n* insert line *n* of p_5
PA print all differing lines in p_1
PE print all differing lines in p_5
PA*m,n*
print lines *m* thru *n* of p_1
PE*m,n*
print lines *m* thru *n* of p_5
OOPS
undo previous editing for this discrepancy
GO terminate editing and continue MRGF
Quit terminate editing, close all files, and exit from MRGF

Ref: *PRIMOS Commands Reference Guide* [49].

MTDENS MT*n* [-6250BPI | -1600BPI]

Set magnetic tape density. (EX, P2)
PRIMOS II only.

MTRESUME MT*n* [-Logical_Tape *ltn*] {*pathname* [-CoMmand_line_OPTions *options*] | -INDEX *n* [-Page_Length *lines*] [-No_Wait] | -Help}

Execute (resume) a command from magtape. (IN)
Ref: *Operator's Guide to System Commands* [35].

NCOBOL [same options as COBOL]

Nonshared old COBOL compiler. (EX, P2, OBS)

Net {-ASSIGN *line* | -START *line* [-SITE *nodename*] |
-STOP {*line* | *nodename*} | -UNASSIGN *line*}

Control half-duplex network. (IN, OP) Ref: *Network Planning and Administration Guide* [27].

NETCFG [-NOCHECK] [-DSC]

Configure PRIMENET. (EX, OBS)
Prompts for the following about the RING, IPC, SMLC and PDN network types: name, PDN address, ID, slave #, line #, enable FAM, permit remote FAM to start disks, enable remote login. Use CONFIG_NET after 19.2. Ref: *PRIMENET Guide* [45].

NETLINK [*options*]

Network linker. (EX)
Prompts with an @. NETLINK subcommands are:

C *address*	Connect to an address
COntinue	Continue a currently active circuit.
D	Disconnect the currently active circuit.
HELP	Invoke the help function.
NC *address*	Connect without reverse charging.
PAuse	Exit to PRIMOS but allow returns.
Quit	Exit to PRIMOS command level.

Ref: *PRIMENET Guide* [45].

NETLOG

Convert NETREC file to an ASCII file. (EX, OBS)

NETLVL

Change severity level of network errors. (IN, OP, OBS)

NSED [*filename*]

Non-shared EDitor. See ED for commands. (EX, P2)

NTS_ASSOCIATE [-LINE *primos_line_number*] [-Lts_NAME *lts_name*
-Lts_LINE *lts_line_number*] [-PERManent]

Associates an LTS line number with a PRIMOS line number for assignment. (EX, 21.0)

NTS_LINE -CoMmanD

Sets the NTS line the user is logged on to into LTS command mode. (EX, 21.0)

NTS_LIST_ASSOCIATE [-LINE *primos_line_number* |
-Lts_NAME *lts_name* [-Lts_LINE *lts_line_number*]]

Lists NTS assigned line associations. (EX, 21.0)

NTS_UNASSOCIATE [-LINE *primos_line_number* |
-Lts_NAME *lts_name* -Lts_LINE *lts_line_number*]

Removes an association between an NTS line and an assignable PRIMOS line. (EX, 21.0)

NUMBER

Number or renumber a BASIC file. (EX)
Prompts for pathnames and starting and increment numbers. Ref: *PRIMOS Commands Reference Guide* [49].

OAS

Enter into Master Function Selection of OAS. (EX)

OA_ADMIN

Enter into System Administrator Function Selection of OAS. (EX)
Ref: *OAS Administrator's Guide* [30].

OA_TERM

Downline load PT65 (Ontel) for OAS. (EX)

Open [*pathname*] *unit key*

Open file on specified unit. (IN)

key	Description
1	Read
2	Write
3	R/W
4	Close
5	Delete
6	Exist
7	Rewind
10	Truncate
+0	File is in current directory
+100	File is entry in segdir open on *funit*
+1000	Change open mode of *funit*
+0000	New SAM
+2000	New DAM
+4000	New SAM segment
+6000	New DAM segment
+10000	New UFD

pathname optional only for Rewind and Truncate. Ref: *PRIMOS Commands Reference Guide* [49].

OPRpri [1 | 0]

Set operator privilege. (IN, OP, OBS)

Not required after 21.0. Ref: *PRIMOS Commands Reference Guide* [49] and *Operator's Guide to System Commands* [35].

ORigin

Attach user to origin (login) directory. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

OSLOG

Control OS logging facility.

OWLDSC [-FAST] [-NOLOCK] [-REPORT]

Owl interface program for DPTX. (EX)
Ref: *Distributed Processing Terminal Executive Guide* [12].

PASCAL *filename* [*CE-options*]

Invoke Pascal compiler. (EX)
Ref: *Pascal Reference Guide* [37]. See Compiler options, 2.7.1 for options.

PASSWD [*owner-password* [*non-owner-password*]]

Set passwords on current UFD. (IN)
If not given, passwords are blanks (no password). Ref: *PRIMOS Commands Reference Guide* [49].

PassWord_DIRs {-ON | -OFF}

Sets the ability to create password directories. (OP, IN)
Ref: *System Administrator's Guide, Volume III: System Access and Security* [64]

PHantom *pathname* [*funit* | *CPL-arguments*]

Start phantom user. (IN)
Funit only valid for *cominput* files. If using a cominput file, file should end with 'LOgout' command. Ref: *PRIMOS Commands Reference Guide* [49].

PHYRST [-UNMOD] [-TTY] [-NO_BADS] [-SPEED {25 | 100}]

Physical disk restore. (IN, OP)

```
UNIT NO: n | Quit  (for tape drive n)
LOGICAL TAPE: n  (for nth logical tape)
CORRECT TAPE? YES | NO
RESTORE ALL PARTITIONS TO ORIGINAL POS? YES | NO
RESTORE PARTITION xxxxxxxx? YES | NO
AS PARTITION: CR | pdevno
PARAMETERS OK? YES | NO
```

Ref: *Operator's Guide to System Backups* [34].

PHYSAV [-UNMOD] [-TTY] [-COMDEV] [-SPEED {25 | 100}]

Physical disk save. (IN, OP)

```
UNIT NO: n  (for tape drive n)
LOGICAL TAPE: n  (for nth logical tape)
**COMMENT** up to 80 char comment
PHYS. DEV. NO: physical-device-number-to-be-saved
USE THE RAT? YES | NO
40MB DISK? YES | NO
PARAMETERS OK?  YES | NO
```

Ref: *Operator's Guide to System Backups* [34].

PL1 *pathname* [*CE-options*]

Full PL/I compiler. (EPF)
Uses same options as PL1G, see 2.7.1. Ref: *PL1 Reference Guide* [42].

PL1G *pathname* [*CE-options*]

PL/I subset G. (EPF)
See Compiler options, 2.7.1, for options. Ref: *PL/I Subset G Reference Guide* [43].

PLIB

Random EDMS utility. (EX)

PLOT

PRIMEAIDS plot utility. (EX)

PLP *filename* [*options*]

PLP compiler. (EX, QT)

Pm

Print user register vector. (IN)
Displays starting address(SA), ending address(EA), program counter(P), register contents for: A, B, X, keys(K), procedure base(PB), stack base(SB), linkage base(LB), and the temporary base(XB). Ref: *PRIMOS Commands Reference Guide* [49].

PMA [-Input] *ipath* [-B *btree*] [-L *ltree*] [1/*a-reg*] [2/*b-reg*] [3/*x-reg*]

Prime Macro Assembler. (EX)
Options:

```
Errlist  Errors-only listing
EXplist  Expanded listing

A-REG          ON-OPTION            OFF-OPTION

1     100000   Unused               --
2     040000   Errlist              EXplist
3     020000   EXplist              Errlist
4-7   017000   Unused               --
8-10  000700   Input device (default = 7)
11-13 000070   Listing device (default = 7)
14-17 000007   Binary device (default = 7)

Device codes (Input, Listing, Binary):

0 - None            4 - Line Printer
1 - ASR             5 - Magtape
2 - PTR/PTP         6 - Cassette
3 - Card Reader     7 - Disk

B-REG (PRIMOS IV BUILD):

11-13 000020   64-user version
      000000   16-user version
16    000001   Large 16-user version
```

Ref: *Assembly Language Programmer's Guide* [44].

POWER

Invokes the POWERPLUS data management facility. (EX)

PRATIO {*value-pagdev*$_0$ [...*value-pagdev*$_7$] | -DISPLAY}

PRATIO
Sets or displays the ratio for the paging devices. (IN, OP)
Ref: *Operator's Guide to File System Maintenance* [32].

PRerr

Print ERRVEC and last error message. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

PRIMIX [*UNIX-command*]

Enter UNIX subsystem. (EPF)

PRIMOS [*primos-directory*]

Start Primos from Primos II. (OP, P2 only, OBS)
If *primos-directory* is given, that will replace the default for all subsequent executions.

X.PRINT *pathname...*
[-HEader | -No_HEader]
[-Line_Length *n*]
[-No_Wait]
[-Page_Length *n*]

[-No_Line_Numbers]
[-Line_Numbers]
[-SeLect [*from*] [*to*]]
[-LIMit *from to*]
[-No_Case]

Print file contents to terminal. (EPF, NR)
X.PRINT internally supports the wildcard convention.

PRINT_KSR

Print a file from SPOOLQ to keyboard printer (OAS). (EX, OBS)

PRINT_NETLOG [*output-file* | TTY] [*options*]

Convert a NETREC file to an ASCII file. (EX, OBS)
Replaced with DISPLAY_LOG at 21.0. *Options:*

-Census
    Reports totals of entries.
-COntinue
    Continue processing after a bad entry.
-DEBug
    Read entries from terminal for testing.
-Delete
    Delete output file when done.
-DUmp
    Display entries in octal.
-From [*mmddyy* [*hhmm*] | TODAY]
    Earliest entry to be processed.
-Help
    Print option usage.
-Input *Pathname*
    Input file to be used.
-PURGE
    Empty contents of event log file.
-Remark *text*
    Enter text into input file.
-Spool
    Spools the output file.
-Type $type_1$ ... $type_n$
    Process only these types.

Ref: *PRIMENET Guide* [45].

PRINT_SCS [*pathname*] [*options*]

Print a file containing SCS data streams (SNA). (EX)
Options:

-AS *alias*
-AT <destination>
-COPies *n*
-DeFer *hh:mm*
-DeLete
-DISK {*disk-name* | *ldev-number*}
-ForMaT {NONE | PAGE}

-FOrm *type*
-NOHead
-No_Page_HeaDeR

Ref: *Remote Job Entry Phase II User's Guide* [52].

Print_Security_LOG -LOGFILE *pathname* [*options*]

Displays a report from a security audit file. (OP, 21.0)
Options:

-USERS *userid-list*
-NUMBER_OBJECT *num-obj-list*
-TEXT_OBJECT *text-obj-list*
-EVENTS [FILE SYSTEM] [SYSTEM] [PRIV_OPS] [ATTACHES]
-EVENT_TYPES [SUCCESS] [NO_ACCESS] [FAILURE]
-NO_WAIT
-NO_HEADER
-HELP

Ref: *System Administrators Guide, Volume III: System Access & Security* [64].

PRINT_SYSLOG

Converts LOGREC files to ASCII. (EX, OBS)
Replaced by DISPLAY_LOG at 21.0. Ref: *Operator's Guide to System Monitoring* [36].

PRMPC *pathname*

Print file on line printer (PR0). (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

PROP *phantom-name option...*
-STATUS

Control spooler phantoms. (EX, OP)
Environment options can be:

-CREATE -MODIFY
-DELETE -STATUS
-DISPLAY -COMPRESS

Phantom options can be:

-ABORT -BACK *n*
-CONTINUE -DROP
-HANG {NOW | FINISH | IDLE}
-RESTART -START
-STOP {NOW | FINISH | IDLE}

-CREATE and -MODIFY subcommands (obsolete at 21.0):

COMOUT [ON | OFF]
DEST *synonym*
DEVICE [PR0 | PR1 | PR2 | PR3 | CENPR |
CE2PR | PLOT | AMLC *n*]
FILE
FORM *synonym*
HEADER [0 | 1 | 2]
LARGE [*n*] (*default:* 30)
LENGTH [*n*] (*default:* 38)

LIMIT [*n*] (*default:* 3000)
LINES [*n*] (*default: LENGTH+13*)
LOWER [*n*] (*default:* 0)
MESSAGE *text*
PAPER [*name*] (*default:* ' ')
PLOT [ON | OFF]
PRINT [ON | OFF]
QUIT
UNDEST *synonym*
UNFORM *synonym*
UPCASE [ON | OFF]
UPPER [*n*] (*default:* 63)
WIDTH [*n*] (*default:* 180)

Ref: *Operator's Guide to the Spooler Subsystem* [33].

PROtec *pathname* [*owner* [*nonowner*]]

Set protection on file. (CPL, OLD)
0-No access(*default*), 1-Read, 2-Write, 3-R/W, 4-Delete/Truncate, 5-D/T/R, 6-D/T/W, 7-All.
Default on file creation equals 7 0. Ref: *PRIMOS Commands Reference Guide* [49].

PROTECT *pathname* [*owner-code* [*non-owner-code*]] [-RePorT]

Sets protection rights for password protected objects. (EPF)
Codes are:

NIL - no access(*default*)
R - read
W - write
D - delete
RW - read and write
RD - read and delete
WD - write and delete
RWD - all

Ref: *PRIMOS Commands Reference Guide* [49].

PRSER *pathname*

Print file on serial line printer. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

PRTDSC $station_1$ [...$station_n$]

Printer emulation program. (EX, OP)
Ref: *Distributed Processing Terminal Executive Guide* [12].

PRVER *pathname*

Print file on Versatec. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

PSD [*token*...]

Prime Symbolic Debugger. (EX)
(NOTE: VPSD has: segment, base register operations, does not have: symbols, trace.)

TERMINATORS for 'A'

.CR.	*+1
,	*+1
^	*-1 (uparrow)
.n	*+n
.-n	*-n
@	Effective address
\	Back to last
(	To contents of *
)	Back to last defined (
=	EA + contents, no update of *
/	Return, do not close *
?	return, do not close *
!	Return, close *

MODES

:A	ASCII
:B	BINARY
:H	HEXADECIMAL
:O	OCTAL
:S	SYMBOLIC
:D	DECIMAL
:P	AP
:L	LONG OCTAL INTEGER

Expressions: Locations can be expressions including:

* (current location)
[+]*number-in-current-mode*
>*number-relative-to-relocation-constant*

Subcommands:

Access *loc*
    Access location.
Breakpoint *loc*
    Set breakpoint (up to 10).
BR Print base registers.
Copy *from to new-addr*
    Copy block of memory to new location.
Define *sym val*
    Define symbol.
Dump *from to* [*ncol*] [*mode*]
    Dump contents of memory.
Effective *from to match* [*mask*]
    Search for effective address.
EXecute
    Execute segmented program.
FAddress *fld-addr-reg-no*
    Access field address register.
FLength *fld-len-reg-no*
    Access field address register.
Fill *from to val*
    Fill memory block with *val.*
GO [*count*] [*a* [*b* [*x* [*k*]]]]
    Continue at breakpoint.
Jumptrace [*start* [*a* [*b*]]]
    Execute obj prog and produce diagnostic listing.
Keys *value*
    Set keys to value.

LB *sn wn*
Set link base.
List *loc*
list location.
LS Load symbols (unit 1).
MAp Print load map symbols.
MO {D16S | D32R | D64R | D64V | D32S | D32I}
Set address mode.
Monitor [*start* [*a* [*b*]]] *addr*
Trace obj prog for mem ref instr.
Not-equal *from to nmatch* [*mask*]
Negative search.
Open *fname unit key*
Open unit.
PATCH *loc1 loc2*
Patch instr in *loc2* into *loc1*.
Print Print brkpt, contents, a, b, x, keys, relocation.
PRoceed [*newbrk* [*a* [*b* [*x* [*k*]]]]]
Set new brkpt and resume execution.
Quit Quit.
RElocate *reloc-val*
Set relocation constant.
Run [*loc* [*a* [*b* [*x* [*keys*]]]]]
Run program.
SB *sn wn*
Set stack base.
Search *from to match* [*mask*]
Search memory block.
SN *sn*
Set segment number.
SY 0 Symbol mode off.
SY 1 Symbol mode on.
Trace [*addr* [*a* [*b* [*val* [-1 *interval*]]]]]
Trace program.
Update *loc val*
Update location.
Verify *from to copy-addr*
Verify block of memory.
VErsion
Print version, restart address.
Where
Display brkpts and proceed counts.
X *reloc-val*
Set relocation constant.
XB *sn wn*
Set X base.
XR *val*
Set X register.
YR *val*
Set Y register.
Zero [*brk-loc*]
Remove brkpt (current)

Ref: *Assembly Language Programmer's Guide* [44].

PSD20

PSD for 16K PRIMOS II. (EX, P2)
See PSD.

PST100DSC

IBM 3277 emulation program for PST100 (DPTX). (EX, OBS)
Replaced by PTDSC.

PTDSC

IBM 3277 emulator for PST100 or PT200 (PDTX). (EX)
Ref: *Distributed Processing Terminal Executive Guide* [12].

PTELE

Access OAS telephone inquiry function. (EX)

PT45DSC

IBM 3277 emulator for PT45. (EX)
Ref: *Distributed Processing Terminal Executive Guide* [12].

PT46DSC

IBM 3277 emulation for PT46(DPTX). (EX)
Ref: *Distributed Processing Terminal Executive Guide* [12].

RDY [-LONG | -BRief] [-ON | -OFF] [-Ready_Long *text*] [-Ready_Brief *text*] [-Error_Long *text*] [-Error_Brief *text*] [-Warning_Long *text*] [-Warning_Brief *text*]

Choose prompt messages. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

REeNter

Re-enter a subsystem after quitting. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

REFORM

Representation formatter for files with STROMA constructs. (EX, QT)

ReLeaSe_level [-ALL | -TO *n* | -LeVels *n*]

Release one or more stack levels. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

REMote PERMIT [*option*]

Set remote access to local files. (IN, OBS)
Obsolete as of 19.3. Option can be:

node $pdev_1$ [...$pdev_9$]
node **-ALL**
-NET $pdev_1$ [...$pdev_9$]
-NET -ALL

REMove_EPF [*pathname*] [-ACtive | -Not_Active] [-Help] [-VeriFY | -No_VeriFY] [-QUERY | -No_Query] [-PRoGram | -LIbrary] [-Force]

Unmap EPF from user workspace. (IN, 19.4)
Ref: *Programmer's Guide to BIND and EPFs* [51].

Remove_Priority_ACcess *disk-name*

Removes priority ACLs from *disk-name*. (IN, OP, SA)
Ref: *Operator's Guide to System Commands* [35].

Remove_Remote_ID -on *nodename*

Removes a remote ID established by ARID. (IN)

REN

Re-enter subsystem after quit. (IN)
See also: REENTER.

REPLY {-*usernum* -TAPE {GO | ABORT | *pdn* | RESEND} | -TAPE RESEND | -ALL -RESEND | -*usernum* -RESEND | -REPEAT *seconds*}

Reply to a tape drive request. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

Reset_DuMP [-HELP]

Resets partial tape dump parameters to their default values. (OP, IN)
Defaults are:

0 to 1777 (ring 0 PRIMOS)
6000 to 6003 for all logged-in users
4000 to 7777 for the process that was using the CPU at the time

Ref: *Operator's Guide to System Commands* [35].

RESTATE

Representation converter(SPS). (EX, QT)

RESTor *pathname*

Restore external program. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

RESTORE_RBF *src-pathname* [*dest-pathname*] [-No_Query] [-PROtect] [-DAM] [-CAM] [-Min_eXt_Len] [-RePorT]

Activate an inactive RBF file. (EX)

Resume *pathname* [*arguments...*] [*p* [*a* [*b* [*x* [*k*]]]]]

Run an EPF, CPL or static mode program. (IN)
P, a, b, x and *k* are only valid for static mode programs. Ref: *PRIMOS Commands Reference Guide* [49].

RESUS *subcommand*

Remote Systems User facility. (EPF, 21.0)

Subcommands:
-ENABLE
-DISABLE [-FORCE]
-START [-ON *node-name*]
-STOP
-STATUS [-ON *node-group*]
-Help [-No_Wait]
-USAGE

Ref: *DSM User's Guide* [13].

REVERT_PASSWORD

Change current directory from ACL to password. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

RJ1004, RJ200UT, RJ7020, RJX80, RJGRTS, RJHASP

Submit job to remote site (EX, OLD)
Replaced by RJOP. Ref: *Remote Job Entry Guide.*

RJOP

RJE operator command processor. (EX)
Ref: *Remote Job Entry Phase II User's Guide* [52].

RJQ *pathname* [-TO] {*queuename* | *sitename*} [*options*]
RJQ {-LIST | -CANCEL | -RESET} [OWN | ALL | RJ*xxxx*] [*options*]

Send a job to a remote mainframe via RJE system. (EX)
Options for submittal:

-WITH *protocol*
-DeFer *time*
-No_Copy
-DeLete
-No_Translate
-AS *internal-name*
-DEVice {CR[*n*] | LP[*n*] | CP[*n*]}
-VFC [NONE | FTN]
-Keep_Request
-LU *lu_port_name*
-MEDSUB *medium/subaddress*

Options for all others:

-TO *queuename*
-WITH *protocol*

-DeFer *time* (-LIST only)

Ref: *Remote Job Entry Phase II User's Guide* [52].

RLS

Release stack history. See RELEASE_LEVEL.(IN)
Ref: *PRIMOS Commands Reference Guide* [49].

RO_TRACE_EVENTS *filename* [-SYSTEM | -USER *userno*] [-ON {*option*$_1$ *option*$_2$ ... | -ALL}] [-OFF [*option*$_1$ *option*$_2$ ... | -ALL]] [-REMOTE_NODE *node*] [-DEBUG] [-No_Query] [-DISPLAY] [-HELP]

Display ROAM actions taken by entire system or a user. (EX)

ROSAU

ROAM system administrator utility. (EX)
Ref: *ROAM System Administrator's Guide* [53].

ROUTL [-DumpFILE *treename*] [-No_INVoKe]

Interactive tool to examine ROAM shared memory. (EX)

RPG *filename* [-SEQCHK | -NOSEQCHK] [-BANNER | -NOBANNER] [-OBDATA | -NOOBDATA] [-STATUS | -NOSTATUS] [-ERRTTY | -NOERRTTY] [-LISTING] [-BINARY]

Report Program Generator (RPG II). (EX)
Obsolete after rev 20.0. Use VRPG.

RSTERM [-READ] [-WRITE]

Empty terminal I/O buffers. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

RUNOFF [*pathname*]

Text formatter. (EX)
Notes:

- When embedded in text, all runoff command lines begin with a period; when issued at command level, runoff commands do not begin with a period.
- In the table below, some runoff command actions are followed by brk, ejt, and/or deflt to indicate the command causes a break, ejects a page, and/or is the default. Also, if the runoff command has a default value, that value is specified.

(*str* = text string) Subcommands:

.NULL.	Start processing (from command mode).
* *str*	Comment line.
+ *str*	Enter verbatim string.
/-/-/-/	/Left/Center/Right/ strings.
> *str*	Center string.
Adjust	Enter adjust/fill modes (brk, deflt).

BLank *char* Define blank substitute character (.NULL.).
BMargin *n* Set bottom margin (brk, ejt, 5).
Break Break (start new line).
CMargin *n* Set column margin (brk, ejt, 5).
Column Set number of columns (brk, ejt, 1).
DDown *str* Down Decimal level.
DDSup *str* Down decimal level, no decimal number.
DEfine *sym str*
Define symbol value.
DIndent *lev before after*
Set decimal indents. 0 => all levels
DLevel *n* Go to decimal level *n* (1).
DLImit *n* Set highest decimal level to appear in Table of Contents (all).
DNext *str* Next heading on current decimal level.
DNSup *str* Next heading on current decimal level, suppress number.
DReset *n* Reset number on decimal level *n*.
DSkip *lev before after*
Set decimal heading skip values. 0 => all; -1 => eject before
DUp [*n*] Go up *n* decimal levels (1).
EEven Eject to next even numbered page.
EFooter /-/-/-/ Define even-page footer.
EHeader /-/-/-/
Define even-page header.
Eject Page eject (brk, ejt).
EOdd Eject to next odd numbered page.
ERase *char* Define cmnd mode erase char.
ERRgo Continue on error.
FILe *fn* Specify output file.
Fill Enter fill mode.
FLoat *fn* Floating insert of *fn*.
FOoter /-/-/-/ Define footer for all pages.
FRom *n* First page number to output.
Header /-/-/-/ Define header for all pages.
HYphen *char* Define phantom hyphen char (.RUBOUT.).
Indent *n* Indent left margin (5).
INDEX *str* Write *str* and page number to index.
INSert *fn* [(*parms*]
Insert *fn*.
INSert *unit* Insert from *unit*.
IXfile *fn* Define index file (16).
Kill *char* Define command line kill char (?).
Length Specify physical page length (brk, ejt, 66).
NAdjust Leave adjust mode (brk).
NEed *n* Eject if < *n* lines (1).
NERrgo Stop on error encountered (deflt).
NFILE No output to file.
NFill Leave fill and adjust modes (brk).
NIXfile Stop output to index file.
NParagraph No paragraph indentation (deflt).
NPAUse No pause between pages (deflt).
NPERforate No perforation marks (deflt).
NTty No output to TTY (deflt).
OFooter /-/-/-/ Define odd-page footer.
OHeader /-/-/-/
Define odd-page header.
PAGen *n* Set page number (1).
Paragraph [*m*] [*n*]
Start paragraph, indent*m*, skip *n*.

PAUse Pause between output pages.
PERforate Print perforation marks.
PIcture *n* Leave *n* lines together (1).
PUrge Force in outstanding floats.
Quit Exit RUNOFF (brk, ejt).
RBar [ON] Start revision bars.
RBar [OFF] Stop revision bars.
REturn *n* Return to prev input file (0).
Rindent *n* Indent right margin (5).
RUndent *n* Undent right margin (0).
Skip *n* Skip *n* lines (brk, 1).
SM *n* Specify side margins (brk, ejt, 7).
SO *n* Print *n*th source line # (1).
SPace *n* Specify single/double, etc. spacing (1).
STop Conditional .QUIT/.RETURN.
SYchar *char* Define symbol delimiter (%).
Tab *char n1* ...
Set tab character and stops.
TMargin *n* Specify top margin (brk, ejt, 7).
TO *n* Specify last page to print (32767).
TOFc *fn lim* Specify table of contents file.
TOFc [*opt*] Close, stop, start table ofcontents for *opt*=omitted, 0, 1.
TTOfc *str* Enter string in table of contents.
TTy Output to TTY.
UNDEFine *sym*
Undefine symbol.
Undent *n* Undent left margin.
WIDOw *n* Specify allowable widow size (0).
Width *n* Specify paper width (brk, ejt, 85).
{{*str*}} Underline *str*.

Ref: *New User's Guide to EDITOR and RUNOFF* [28].

RWLOCK *pathname lock* [-RePorT]

Set file read/write lock. (EX)
lock may be: SYS - system default; EXCL - N readers OR 1 Writer; UPDT - N readers AND 1 writer; NONE - N readers and N writers. Ref: *PRIMOS Commands Reference Guide* [49].

SAve *pathname* [*sa* [*ea* [*pc* [*a* [*b* [*x* [*keys*]]]]]]]

Save memory image. (IN)
Do not use SAve with 64V or 32I segmented files or EPFs. Ref: *PRIMOS Commands Reference Guide* [49].

SAVE_RBF *src-pathname dest-pathname* [-PROtect] [-DAM] [-CAM] [-Min_eXt_Len] [-RePorT]

Make a backup copy of an RBF. (EX)
Ref: *ROAM Administrator's Guide* [53].

SCHDEC [[-SCHEMA] *schema-name* [-LISTING] *out-file*]

Invoke DBMS schema decompiler. (EX)
Ref: *DBMS Administrator's Guide* [8].

SCHED [*pathname*]

Alter definition of database. (EX)
Ref: *DBMS Administrator's Guide* [8].

SCHEMA *pathname* [-OUTPUT*pathname*] [-LIST *pathname*] [-DAM]

Invoke DBMS DDL compiler. (EX)
Ref: *DBMS Data Description Language Reference Guide* [9].

SCRIBE *pathname* [*options*]

SCRIBE document formatter. (EPF)
Options:

-Agile Generate output for an Agile printer.

-Diablo Generate output for a Diablo printer.

-DEVIce *name* Generate output for the device *name*.

-DOCument *name*
Produce output in file *name*.

-DOVER Generate output for a Dover printer.

-DRAFT [*value*]
Set the variable *draft* to *value* (default for *value* is 1).

-File Generate output for the device *file*.

-Gsi Generate output for a GSI photocomposer.

-GIGI Generate output for a GIGI.

-HypVocab Create a lexicon showing the hyphenation points of each word in the document.

-HYD Create a lexicon showing each hyphenation decision.

-IMPrint, -IMPTINT10
Create output for an Imagen Imprint10 laser printer.

-KEEPfiles Don't delete temporary files.

-Lpt Create output for a Line PrinTer.

-LA36 Create output for a DEC LA36.

-LGP1 Create output for an LGP1.

-PAGEDFILE Create output for a PagedFile.

-Quiet Don't print error messages on the terminal.

-Voc, -VOCABulary
Generate sorted word listings in a .LEX file.

-Words, -WORDCOUNT
Count the number of words in the document.

-X, -X9700 Create output for a Xerox X9700.

SECurity_MONitor [*options*] [-HELP]

Enables/disable audit collection. (IN, OP)
Ref: *System Administrator's Guide, Volume III: System Access and Security* [64].

SECurity_STatus [*options*] [-HELP]

Display status of system events being audited. (IN, OP)
Ref: *System Administrator's Guide, Volume III: System Access and Security* [64].

SEG [*pathname* [1/1] | -LOAD]

Segmented loader. (EX)
Giving *pathname* executes that segdir. 1/1 causes the segdir to be loaded and execution is passed to VPSD (See PSD for commands). -LOAD causes SEG to go into the LOAD subprocessor. Subcommands:

DELETE [*filename*]
deletes runfile.
HElpprint list of SEG commands.
LOad [*pathname*]
define runfile and invoke loader for creation.
LOad * [*pathname*]
define runfile and invoke loader for appending.

ATtach [*UFDname*] [*password*] [*ldisk*] [*key*]
attach to UFD.
A/SYmbol *sname* [*segtype*] *segno size*
define a symbol in memory and reserve space for it using absolute segment numbers.
COmmon [ABS] *segno*
relocate COMMON using absolute segment numbers.
COmmon REL *segno*
relocate COMMON using relative segment assignment.
D/IL, D/LOad, D/LIbrary, D/FOrceload, D/PL or D/RL
load using previous parameters. D/ and F/ may be combined.
EXecute
save load to disk and execute program.
F/xx [*filename*] [*addr psegno lsegno*]
forceload all routines in object file.
IL [*addr psegno lsegno*]
load impure FORTRAN library.
INitialize [*pathname*]
initialize and restart loader.
LIbrary [*pathname*] [*addr psegno lsegno*]
load library file.
LO [*pathname*] [*addr psegno lsegno*]
load object file.
MAp [*filename*] *option*
generate load map.
MIxup [ON | OFF]
mixes procedure and static data.

MV moves portion of loaded file. Will prompt for info.
NSCW
Do not generate warnings for smaller redefinition of common blocks.
Operator {0 | 1}
relax/impose high-level restrictions
PL [*addr psegno lsegno*]
load pure FORTRAN library.
P/xx [*filename*] [*option*] [*psegno lsegno*]
load on a page boundary.
QUit return to PRIMOS command level.
REturn
return to SEG command level.
RL *pathname* [*addr psegno lsegno*]
reload a routine.
R/SYmbol *sname* [*segtype*] *segno size*
define symbol in memory and reserve space for relative segment assignment.
SAve [*a* [*b* [*x*]]]
save load to disk.
SCW
Emit warnings for smaller redefinition of common blocks (default).
SE *segno len*
create base area for desectorization.
Split {*segno addr* | *addr* | *addr ssegno saddr esegno*}
break data into data and procedure portions
SS *sname*
save symbol.
STack *size*
change stack size.
SYmbol [*sname*] *segno addr*
define a symbol at specific location in memory.
S/xx [*filename*] *addr psegno lsegno*
load a specific absolute segment.
XP *dsymbol dbase*
expunge symbols from symbol table and delete base information.

MAp {*filename1* | *} [*filename2*] [*option*]
prints specified load map. *option*:0=full map (default), 1=extent map only, 2=extent map and base areas, 3=undefined symbols only, 4=full map, 5=system programmer's map, 6=undefined symbols sorted, 7=full map sorted, 10=symbols by ascending addr, 11=symbols sorted.
MOdify [*filename*] or SA [*filename*]
invoke modification subprocessor.

NEw *filename*
write new copy of runfile to disk.
PAtch
modify save range of existing segment.
REturn
return to SEG command level.
SK {*ssize* | *segno addr*}
alter stack size and/or location.
STart *segno addr*
change program execution start address.
WRite
write all segments to disk.

PArams [*filename*]
display parameters of runfile.

PSd invoke VPSD debugging utility.
Quit return to PRIMOS command level.
RESTore [*pathname*]
bring runfile into user memory.
RESUme [*pathname*]
restore runfile and begin execution.
SHare [*pathname*]
create R mode runfiles for segments below '4001.
SIngle [*pathname*] *segno*
create R mode file image of single segment.
TIme [*pathname*]
print time and date of last runfile modification.
VERSION
display SEG version number.

Ref: *SEG and LOAD Reference Guide* [23].

SEtime -*mmddyy* -*hhmm*

Set date and time. (IN, OP)
Must be issued before user logins possible (unless machine has newer CP (2000, 4000, 6000 and 9000) series). Ref: *PRIMOS Commands Reference Guide* [49] and *Operator's Guide to System Commands* [35].

SETMod {-User | -Operator | -Noassign}

Control tape drive assignment. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

Set_ACcess *target* [*acl* [-No_Query] | -LIKE *reference* | -CATegory *acat-name*]

Set access rights to an object. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

SET_ASYNC {-LINE *n* [-TO *m*] [*options*] | -DisPlay | -Help}

Set async line configurations (EPF, 21.0)
Options:

-DEFault
-SYStem
-ASsiGNable {NO | yes}
-Char_Length {5 | 6 | 7 | 8}
-[NO_]Data_Set_Control
-[NO_]Data_Sense_Enable
-Data_Set_Sense {HIGH | low}
-[NO_]DISLOG
-[NO_]ECHO
-[NO_]ERRor_DETection
-[NO_]Line_Feed
-[NO_]LOOP_line
-PARity {NONE | odd | even}
-PROtocol [*name*]
-[NO_]REVerse_XOFF
-SPEED [*value*]
-[NO_]Speed_Detect
-STOP_bits {1 | 2}

-USER_number *x*
-[NO_]XOFF

Ref: *System Administrator's Guide, Vol.II: Communication Lines and Controllers* [63].

SET_DELETE *pathname* [-PROtect | -No_PROtect]

Set/reset protection from deletion. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

Set_Priority_ACcess *disk-name acl*

Put a priority ACL on a disk. (IN, OP)
Ref: *Operator's Guide to System Commands* [35].

Set_Quota *pathname* [-Max *n*]

Set maximum number of records allowed on a directory. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

SET_RBF *pathname* [-AIrcv] [-BIrcv] [-No_AIrcv] [-No_BIrcv]
[-TRans_rollback] [-No_TRans_rollback] [-AICHK] [-No_AICHK]
[-USAGE {TRANS | NON-TRANS}] [-LOCK] [-No_LOCK] [-Write_Access]

Set the attributes of a ROAM file. (EX)
Ref: *ROAM Administrator's Guide* [53].

Set_Search_Rules [*pathname* | -DeFauLT] [-No_System]
[-List_NAMe *search-rule-name*] [-Help]

Use set of rules to search for objects. (IN)
Ref: *Programmer's Guide to BIND and EPFs* [51].

SET_TIME [-ON *list-of-remote-nodes*]

Set system clock from remote machine. (EX, NR, OP)

Set_Time_Info {-TIMEZONE *timezone-offset* | -HELP |
-DLST {NO | YES [*start-date end-date dslt-offset*]}

Set time zone information. (OP, IN)
Ref: *Operator's Guide to System Commands* [35].

SET_VAR *gvar-name* [:=] *value*

Set the value of a global variable. (EX)
See section on global variables. Ref: *PRIMOS Commands Reference Guide* [49].

SHAre [*pathname*] *segno* [*access*]

Specify shared segment. (IN, OP)
Omitted *pathname* => change access only. *access*: 000-no access, 200-read access, 600-read/execute access(*default*), 700-read, write, execute access. segno < '4000. Note: 'OPRpri 1' must be issued before this command can be used on revs prior to 21.0. Caution should be used when sharing OS segments. Ref: *Operator's Guide to System Commands* [35].

SHutdn {*pdev*$_1$ ... *pdev*$_n$ | ALL }[-FORCE]
SHutdn *pdev* [-RENAME] *packname*
SHutdn *packname*$_1$ [...*packname*$_n$] -ON *nodename* [-FORCE]

Shutdown disk(s) or system. (IN, OP)
-FORCE is used for malfunctioning disk drives. Ref: *Operator's Guide to System Commands* [35].

SIZE {*pathname* [-NORM] | -HELP}

Print size of file. (EX)
Uses 1024 word record size. -NORM option causes size to be given in 440 word records. Size on UFDs, segdirs, and acats show number of entries. Ref: *PRIMOS Commands Reference Guide* [49].

SLIST [*pathname*]

Print file to terminal. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

SNA_3270 [*config-pathname*] -START [-ENTRY_ID *entry-id*] [*options*]
-STOP [*stop-action*] [*options*]
-STATUS [-NO_WAIT] [*options*]
-MESSAGE_LEVEL *msg-level*
-AUTO_STOP *delay-time*
-NO_AUTO_STOP

Administrator command to invoke and control PRIME/SNA interactive. (EPF, 19.4, OP)

[*config-pathname*]
Server configuration pathname.
-START
Start the interactive server.
-Entry_ID*entry-id*
Interactive configuration name.
-STOP [*stop-action*]
Stop the server. Quit, Idle, Finish, Now
-STATus -No_Wait -MeSsaGe_Level *msg-level* Brief, Medium, Detailed -Auto_STOP *delay-time*
Number of minutes to delay automatic logout of Interactive when no user sessions are active.
-No_Auto_STOP
-Remote_System *rsname*
Remote system name in Server configuration file.
-Remote_System_group *rsgname*
Remote system group name in Server configuration file.

Ref: *PRIME/SNA Operator's Guide* [56]

SNA_3270_CONFIG [*config-pathname*] [-ENTRY_ID *entry-id*] [*options*]

Create/edit SNA interactive config files. (EPF, 19.4)
Options:

{-CReate | -EDit | -DisPlay |
-Listing [*listing-pathname*] |
-SPOOL [*spool-options*]}

-Entry_ID *entry-id*
-No_Wait
-Terminal_TyPe *terminal-type*

Ref: *PRIME/SNA Administrator's Guide* [55]

SNA_PRINT [-NO_LOG]

Start and control SNA interactive printer emulation. (EPF, OP, 19.4)
Internal commands:

START *printer-name* [*spool-options*]
STOP [*printer-name*] [-NOW]
STATus [*printer-name*]
DisPlay
Quit
CANCEL *printer-name*
{PA1 | PA2} *printer-name*
Help

Ref: *PRIME/SNA Operator's Guide* [56]

SNA_SERVER [*config-pathname*] -START [*options*]
-STOP [*stop-action*] [*options*]
-STATUS [-NO_WAIT] [*options*]
-MESSAGE_LEVEL *msg-level* [*options*]
{-STATISTICS [*stats-file*] | -NO_STATISTICS}

Invoke and control the SNA server. (EPF, OP, 19.4)
options:

-LINE *lname*
-LINE_GROUP *lgname*
-REMOTE_SYSTEM *rsname*
-REMOTE_SYSTEM_GROUP *rsgname*

Ref: *PRIME/SNA Operator's Guide* [56]

SNA_SERVER_CONFIG [*config-pathname*] [*options*]

Create/edit SNA server config file. (EPF, 19.4)

{-CReate | -EDit | -DisPlay |
-Listing [*listing-pathname*] |
-SPOOL [*spool-options*]}
-No_Wait
-Terminal_TyPe *terminal-type*

Ref: *PRIME/SNA Administrator's Guide* [55]

SORT [-BRief] [-SPace] [-MErge] [-TAG | -NONTAG]

Sort files. (EX)
Prompts for: input filename, output filename, number of pairs of starting and ending columns, input pairs of starting and ending columns, reverse order, and data type. If -MERGE: number of files to be merged followed by input files, one per line.

Multiple input files, file types, and record length information may be specified by using the following keywords. These keywords may be used to bypass the standard dialog. Enter in any order on single line:

-INPUTFILE *name*
-OUTPUTFILE *name*
-KEYS *n*
-INTYPE COMPRESSED
UNCOMPRESSED
FIXED
VARIABLE
-OUTTYPE *type*
-INLENGTH *n*
-OUTLENGTH *n*
-START *n*
-END *n*
-DESCENDING
-TYPE *code*

Ref: *PRIMOS Commands Reference Guide* [49].

SPAC

See Set_Priority_ACl.

SPL *pathname* [*CE-options*]

SPL compiler. (EX, QT)
Ref: *SPL Reference Guide*, [39].

SPOOL [*pathname*] [*options*]
SPOOL -MODify *request-number options*
SPOOL -CANcel {PRT*n* | *n* | -ALL} [*options*]
SPOOL -LISt [*options*]

Print queue manager. (EX)
NB: Pre-rev 21 spoolers will not print files from rev 21 spool queues. File submital *options*:

-ALIas *name*	Replace user name in header (21.0)
-AS *name*	Replace file name in header
-AT *name*	Same as -ATT
-ATTribute *names*	Specify device attributes (21.0)
-COB	Cobol format (21.0)
-COPies *n*	No of copies (1 - 99)
-DEFer *hh:mm*	Defer printing to given time
-DISk *name*	Place request in named pre-rev 21 queue
-FORm *name*	Same as -ATT
-FTN	Fortran format
-HeaDeR *text*	Use text as page header (21.0)
-LNUmbers	Print with line numbers
-NOCopy	Print from original file location (21.0)
-NOEject	Suppress form feed at end
-NOFormat	Inhibit all format actions
-NOHeader	Print without header page
-NOP	Suppress overprinting (was -CRLF) (21.0)
-NotiFY	Notify user on completion
-NPH	Suppress page header in paginate mode (21.0)
-ON *node*	Place request in queue on named node (21.0)
-OPEn	Open file in spool queue
-PLOt *n*	Plot raster size(words/scan)

-PROc *name* PostScript procedure name (21.0)
-RUSh Priority listing (Administrator only)
-SFI Suppress file info in banner (21.0)
-TRUncate Truncate long lines
-TO *page_number*
Stop at *page_number* (21.0)
-TUNit *n* File unit for -OPEN (default 2)

-MODIFY *options*:

any of the above excluding: -NOCOPY, -OPEN, -TUNIT

-NODefer
Cancel deferral

-NORush
Cancel rush priority

-CANCEL *options*:

-DISk *disk-name*
Cancel request on pre-21 queue

-ON *node*
Cancel request in queue on named node

-LISt *options*:

request-number -USEr name
Only requests for named user

-ATTribute *names*
Only requests with named attributes

-BRief
Short form report

-DETail
More detailed report

-FULl
Extended form report

-DISk *name*
Report pre-rev 21 queue on named partition

-ON *node*
Report queue on named node

-ALL Report on all known queues

-NoWait
Suppress --More-- prompt

Ref: *Operator's Guide to the Spooler Subsystem* [33].

SPSS [-INPUT] [-LISTING] [-SIZE] [-PAGESIZE] [-EDIT] [-PRINTBACK] [-MAXERROR]

Statistics Package for the Social Sciences. (EX, OBS)

SPSSX *input-file*

SPSS-X statistics package. (EX, JM)

SPY

MIDASPLUS information utility. (EX)
Ref: *MIDASPLUS User's Guide* [25].

SQ

See Set_Quota.

Start [*token...* [*p* [*a* [*b* [*x* [*k*]]]]]]

Start execution. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

START_DSM [*options*]

Starts DSM on the system. (EPF, 21.0)

Options:
-Multi_Node
-Help [-No_Wait]
-USAGE

Ref: *DSM User's Guide* [13].

START_LSR

Start the login server. (OP, EPF).
Ref: *Operator's Guide to System Commands* [35].

START_NET [*config-filename*] [-NODE *node-name*] [-NT] [-CACHE]

Start network. (EX, OP)
Node-name need not be given if SYSNAM config directive used (21.0). Ref: *PRIMENET Guide* [45].

START_NTS [*config_pathname*]

Start the network terminal service server. (OP, EPF)

STARTUp [PROTECT] [*nodename*] *dvno ...*

Startup disk(s). (IN, P2)

STATus [ALl | COmm | DEvice | DIsks | ME | NEtwork | PRoject | SEmaphores | SYstem | UNits | USers | HDWR]

Print user or system status. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

STATUS_DSM [options]

Displays status of DSM configurations on nodes. (EPF, 21.0)

Options:
-TTP [TTY | PT45 | PST100 | PT200]
-No_Wait
-Help [-No_Wait]
-USAGE

Ref: *DSM User's Guide* [13].

STOP_DSM [-Help [-No_Wait] | -USAGE]

Shuts down DSM on the system by logging out the DSM processes. (EPF, 21.0)
Ref: *DSM User's Guide* [13].

STOP_LSR

Logs out the login server. (OP, EPF)
Ref: *Operator's Guide to System Commands* [35].

STOP_NET

Stop network processing. (EX, OP)
Ref: *PRIMENET Guide* [45].

STOP_NTS

Stops the NTS server. (OP, EPF)

SVcsw [1 | 0]

Set user SVC switch. (IN)
1 => bounce (except class 5), 0 => don't bounce. Ref: *PRIMOS Commands Reference Guide* [49].

SYSLOG

Convert LOGREC file to ASCII file. (EX, OBS)

TA_ADMIN

Start transport agent admin utility for OAS mail. (EX, OBS)

TAP

1-sector octal-mode debugger. (EX, OBS)

TCF {-Host *hname* | -Autoport *n*} -Terminal *tname* [option]

DPTX Transparent connect facility. (EX)
Option can be:

-Quit [*q-string*] (default: TCF$QUIT)
-PA *n* (*n*=1, 2, or 3)
-PF *n* (*n*=1..12)
-TR

Ref: *Distributed Processing Terminal Executive Guide* [12].

TDOS64

Run PRIMOS II emulator under PRIMOS. (EX, OBS)

TEMPLATE

File construction utility(SPS). (EX, QT)

TERM *options*

Set terminal characteristics. (EX)
Options:

-DISPLAY
-ERASE *char*
-KILL *char*
-BREAK {ON | OFF}
-HALF [LF | NOLF]
-FULL
-NOXOFF
-XOFF [-HALF]

Ref: *PRIMOS Commands Reference Guide* [49].

Time

Print time statistics. (IN)
Rev 18: HH'MM logged in, MM'SS CPU time, MM'SS I/O time. Ref: *PRIMOS Commands Reference Guide* [49].

TIMER

OAS timer facility. (EX)

TP [-BE [-ID *workstation*]] [-HELP] [-Log_Input *pathname*] [-NO_OVERPRINT | -NO_OVP] [-RESTORE_CONFIG] [-SAVE_CONFIG] [-SCript *pathname*] [-TERM | -TTP {PT200 | PST100 | PT45}]

Start PRIMEWAY subsystem. (CPL)

TP_EXO

Invoke execute-only version of PRIMEWAY. (CPL)

TPLINK -ON {*mc* | :*addr*} -ID *ws-id* [*options*]

Start PRIMEWAY network utility. (EX)
Options:

-PORT *port*
-HELP

TPBE

Transaction Processing Business Environment. (CPL)

TRACE_RO [SYSTEM | USER] [-USERNO *user-no*] [-ON] [-OFF] [-STATUS]

Display ROAM actions taken by entire system or a user. (EX)

TRAMLC {TRANSMIT | RECEIVE} *filename line* [T]

Amlc I/O. (EX)
Ref: *PRIMOS Commands Reference Guide* [49].

Transfer_LOG *options*

Backs up and restores audit logs. (IN, OP)
Ref: *System Administrator's Guide, Volume III: System Access & Security* [64]

TRANSPORT [-LIST] *pathname* -MT *n* [*options*]

Copy files from disk to tape for transport to another Prime(BRMS). (EX)
Options:

-Cam_To_Dam
-Compatible_VersioN *rev*
-HELP [*argument*]
-INDEX [*pathname*]
-IndeX_Levels [*n*]
-LeVels [*n*]
-No_Query
-REMARK [*character-string*]
-SAVE_Protection
-Tty
-VALidate
-VeriFy
-VOLID *volume-id*

and wildcard date (after/before) options. Ref: *Data Backup and Recovery Guide* [7].

TRANSPORT_RELEASE -MT *n* [*options*]

Release a transport tape for reuse(BRMS). (EPF)
Options:

-VOLID *name*
-No_Query
-REEL *n*
-HELP [*argument*]

Ref: *Data Backup and Recovery Guide* [7].

TRANSPORT_RESTORE *source-path* [*target-path*] -MT *n* [*options*]

Copy file from tape to disk(BRMS). (EX)
Options:

-VOLID *name* [*name...*]
-RECOVER
-INDEX [*pathname*]
-IndeX_Levels [*n*] (1 <= *n* <= 99)
-REEL *n* (1 <= *n* <= 255)
-Tty
-Cam_RBF
-Dam_RBF
-From_Logical_Tape *n*
-To_Logical_Tape *n*

-MAGSAV
-WRitten_After [*date*]
-WRitten_Before [*date*]
-From_Save_Number *n* (1 <= *n* <= 255)
-To_Save_Number *n* (1 <= *n* <= 255)
-No_Query
-VeriFY
-COMBine
-REPLACE
-HELP

Ref: *Data Backup and Recovery Guide* [7].

TYPE *text*

Print text at terminal. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

ULOAD

Loader for code from Z8080MA, Z8KMA, and Z80MA. (EX, QT)

Unassign {*device* | [-ALIAS] MT*n* [-UNLOAD] | ASYNC -LINE *n* [-TO *m*]}

Release peripheral device. (IN)
See ASSIGN for *device* types. Can unassign device held by another user if issued from console. Ref: *PRIMOS Commands Reference Guide* [49].

UPCASE *inpathname* [*outpathname*]

Translate file to upper case. (EX)
If *outpathname* is not given, output will go on file open on unit 2. Ref: *PRIMOS Commands Reference Guide* [49].

USAGE [-ALL] [-BRIEF] [-DEBUG] [-DISK] [-FREQ *sec*] [-ON *nodename*] [-TIMES *n*] [-USER]

Display system resource utilization. (EX)
Ref: *Operator's Guide to System Monitoring* [36].

USErs

Print current number of users. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

USRasr *usrno*

Connect system ASR to user. (IN, OP)
Must type USRASR in full if *usrno* is not logged in. USRASR 1 returns console to normal.
Ref: *PRIMOS Commands Reference Guide* [49].

VISTA

Invoke DBMS/QUERY (Obsolete, use DISCOVER). (EX, 19.4)

VPSD[16]

Virtual mode PSD. (EX)
Supports V-mode. For internal commands, see PSD.

VRPG *pathname* [*CE-options*]

RPG II V-mode compiler. (EX)
See Compiler options, 2.7.1, for options. Ref: *RPG II V-Mode Compiler Reference Guide* [54].

Vrtssw [*sense-switch-setting*]

Set virtual sense switches. (IN)
Ref: *PRIMOS Commands Reference Guide* [49].

WORD [*document-name*] [*options*]

Invoke PRIMEWORD word processor. (EX)
Options:

-CREATE [*format_type*]
Creates a document with the name optionally with PRIMEWORD format type specified.
-PRINT [*n* | -DISK *new-name* | -MENU | -VIEW]
Prints the document named.
-SPELL [-MENU | -OUTPUT *new-name*]
Checks spelling of a document.
-GGLOSS glossary_name
Starts execution of a PRIMEWORD Global Glossary item.
-GLOSS glossary_name> [-STEP]
Starts execution of a PRIMEWORD Local Glossary item.
-NOEXIT
Proceeds to PRIMEWORD Main Menu after finishing the specified functions.

Ref: *PRIMEWORD Administrator's Guide*, [48].

WP_ADMIN

Word Processor administrator. (EX, OBS)
Replaced by OA_ADMIN.

WPS

Word processing system. (EX)

WS1004, WS200UT, WS7020, WSX80, WSGRTS, WSHASP

Control RJE workstations. (EX, OBS)
Replaced by RJOP.

Z80MA

Z80 cross assembler. (EX, QT)

Z8KMA

Z8000 cross assembler. (EX, QT)

2.7.1. Standard compiler options

These options apply to the common backend compilers.

Usage: *compiler_name input-file* [*options*]

compiler_name may be one of: CBL, F77, MODULA, PASCAL, PL1, PL1G, SPL, VRPG.

input-file
Source program name. If "TTY", then take source from terminal.

-32I Generates 32I mode code.

-32IX
Generates 32I mode with general register relative instructions.

-64V
Generates 64V mode code.

-ALLerrors
Override the limit of 100 fatal diagnostics. (CBL)

-Allow_PREconnection
Allow use of pre-opened listing or binary output files.

-No_Allow_PREconnection
Negation of -Allow_PREconnection.

-BANner
Prints column index banner before each non-comment line. (VRPG)

-No_BANner
Negation of -BANner.

-BIG
Handle segment-spanning data properly when unclear from program itself.

-No_BIG
Negation of -BIG.

-Binary *tree*
Specifies binary object file.

-No_Binary
Negation of -Binary *tree*.

-CALCindex
Calculate index offsets when referenced instead of when SET. (CBL)

-No_CALCindex
Negation of -CALCindex.

-CLUster
Cluster routines for optimization. (SPL, PL1, PL1G, PASCAL, F77)

-COMP
Use full hardware capacity (15 or 31 bits) of COMP fields. (CBL)

-No_COMP
Negation of -COMP.

-Compiler_DATa [*tree*]
Specifies path to non-standard compiler data.

-Conformant_Arrays
Used for ISO conformance. (PASCAL)

-No_Conformant_Arrays
Negation of -Conformant_Arrays (PASCAL)

-COPy
Copies, not originals, of constants are passed by reference. (SPL, PL1, PL1G, PASCAL, MODULA, CBL)

-No_COPy
Negation of -COPy.

-CORrMap
Insert into listing a map of CORRESPONDING matches. (CBL)

-No_CORrMap
Negation of -CORrMap.

-D_STateMenT
Interpret statements with a "D" in column 1 as compilable source text. (F77)
-No_D_STateMenT
Interpret statements with a "D" in column 1 as comments. (F77)
-DCIvar
Flags undeclared variables. (F77)
-No_DCIvar
Negation of -DCIvar.
-DeBuG
Generates code for full debugger (DBG) functionality.
-No_DeBuG
Negation of -DeBuG.
-DO1, -DO
Performs one-trip DO-loops according to FORTRAN IV standard. (F77)
-No_DO1, -NDO
Negation of -DO1, -DO.
-DYnm
Allocates local storage dynamically, opposite of -SAve. (F77)
-EntryTRaCe
Generate runtime code to display PROGRAM-ID & DATE-COMPILED. (CBL)
-No_EntryTRaCe
Negation of -EntryTRaCe.
-ERRList
Produces an errors-only listing.
-No_ERRList
Negation of -ERRList.
-ERRorFile
Create a file of diagnostics called <source file name>.CBL.ERROR. (CBL)
-No_ERRorFile
Negation of -ERRorFile.
-ERRTty
Outputs error messages to user terminal.
-No_ERRTty
Negation of -ERRTty.
-Escape34
Convert from IBM to Prime RPG. (VRPG)
-No_Escape34
Negation of -Escape34.
-EXPlist
Expands program listing to include assembler-like output.
-No_EXPlist
Negation of -EXPlist.
-Extended_Character_Set
Prime-Extended-Character-Set. (SPL, F77)
-EXTernal
Allows object file to be linked with other Pascal procedures and functions. (PASCAL)
-No_EXTernal
Negation of -EXTernal.
-FIPS *dec*
The decimal number signals the FIPS syntax level to check. (CBL)
-FORCEbinary
Forces binary even if fatal diagnostics were issued. (CBL)
-No_FORCEbinary
Negation of -FORCEbinary.
-FRN
Better accuracy of single-precision floating-point calculations. (SPL, PL1, PL1G, PASCAL, F77, MODULA, CBL)

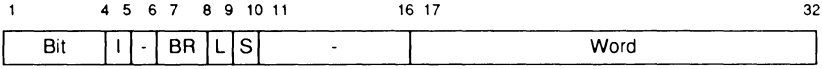
-No_FRN
Negation of -FRN.
-FTN_Entry
All procedures passed as actual parameters are to be passed in the FTN way. (F77)
-No_FTN_Entry
Negation of -FTN_Entry.
-Full_Help
Most detailed online help from the compiler.
-Full_OPTimize
Optimize as much as possible.
-Help
Produces usage information and option list.
-HEXaddress
Addresses in the listing file are printed in hexadecimal notation. (CBL)
-No_HEXaddress
Negation of -HEXaddress.
-Input *tree*
Specifies source input file.
-INTL
Makes INTEGER default to INTEGER*4. (F77)
-INTS
Makes INTEGER default to INTEGER*2. (F77)
-LCase
Distinguishes lower and uppercase characters in the source program.
-Listing [*tree*]
Generate source listing. Write it to *tree* if specified.
-No_Listing
Negation of -Listing [*tree*].
-LOGL
Makes LOGICAL default to LOGICAL*4. (F77)
-LOGS
Makes LOGICAL default to LOGICAL*2. (F77)
-MAIN id
Specifies the main entrypoint of the program, useful with -CLUster. (SPL, PL1, PL1G, PASCAL, F77)
-MAp
Produce a listing with a map of data and procedure names.
-No_MAp
Negation of -MAp.
-MAPSort
Same as -MAP except names are sorted alphabetically. (CBL)
-MAPWide [*dec*]
Same as -MAP except use *dec* character lines instead of 80 (108 assumed if *dec* not given).
-MaX_GRowth_percent *dec*
Specify optimization space growth limits. (SPL, PL1, PL1G, PASCAL, F77, MODULA, CBL)
-MaX_Inline_Statements *dec*
Sets threshold procedure size for inline expansion. (SPL, PL1, PL1G, PASCAL, F77, MODULA, VRPG)
-MAXErrors *dec*
Specifies the max number of errors to display before compilation abort.
-NEsting
Adds nesting level numbers in program listing. (SPL, PL1, PL1G, PASCAL, MODULA, VRPG)
-No_NEsting
Negation of -NEsting.
-OFFset
Print locations of each executable statement in listing.

-No_OFFset
Negation of -OFFset.
-Old_SEmantic
Allows non-standard semantics. (PASCAL)
-OLDio
Allow only I/O constructs allowed by the previous COBOL compiler. (CBL)
-No_OLDio
Negation of -OLDio.
-OPTimize *dec*
Specifies the level of optimization to perform.
-OPTimization_Selection *char*
Specify specific optimization to perform or not.
-OVerFlow
Enables integer exception checking. (F77, MODULA, PASCAL, PL1, PL1G, SPL)
-No_OVerFlow
Negation of -OVerflow.
-PBECB
Load Entry Control Blocks (ECBs) into the procedure frame. (SPL, PL1G, F77, MODULA)
-No_PBECB
Negation of -PBECB.
-PreFiX *char*
Prefix the argument to the source file. Used to define modes. (SPL)
-PRODuction
Generates code for partial debugger functionality.
-No_PRODuction
Negation of -PRODuction.
-PROFile
Generates code that will produce execution profile information.
-No_PROFile
Negation of -PROFile.
-RAnge
Generates runtime code that checks subscript ranges.
-No_RAnge
Negation of -RAnge.
-Range_NonFatal
Generate non-fatal run-time code to check subscript ranges. (CBL)
-RMARGin
Extend Area B of each source line to column 160. (CBL)
-SAve
Allocates local storage statically, opposite of -DYnm. (F77)
-SEQchk
Checks columns 3-5 for proper sequencing. (VRPG)
-No_SEQchk
Negation of -SEQchk.
-Set_DeFaulTs
Set compiler defaults in a compiler data file.
-SIGnalerrors
Abort at runtime if any overflow errors. (CBL)
-No_SIGnalerrors
Negation of -SIGnalerrors.
-SIlent *severity-limit*
Suppress reporting of errors of specified or lower severities.
-SLACKbytes
Issue diagnostic whenever compiler-generated FILLER is inserted. (CBL)
-No_SLACKbytes
Negation of -SLACKbytes.
-Source *tree*
Specifies source input file.

-SPACE
Prefer space over time in optimization.
-STANdard
Warning of variances from language standard. (PASCAL, F77, MODULA)
-No_STANdard
Negation of -STANdard.
-STATistics
Displays compilation statistics at terminal.
-No_STATistics
Negation of -STATistics.
-STATUS
Displays statement types on terminal as parsed. (VRPG)
-No_STATUS
Negation of -STATUS.
-Store_Owner_Field
Store identity of called routines in stack.
-No_Store_Owner_Field
Negation of -Store_Owner_Field.
-SYNtaxmsg
Print "syntax checking suspended/resumed" messages. (CBL)
-No_SYNtaxmsg
Negation of -SYNtaxmsg.
-SYM *tree*
Specify directory into which Modula-2 Definition Symbol files go. (MODULA)
-TIME
Prefer time over space in optimization.
-TPs
Generate errors for Transaction Processing System. (CBL)
-TRUNCdiags
Issue diagnostics for truncated result. (CBL)
-No_TRUNCdiags
Negation of -TRUNCdiags.
-TTrace
Specifies that the Time Trace routines are to be called on entry to procedures and begin blocks.
-NTTrace
Specifies that the Time Trace routines will not be called.
-UPcase
Map source program to uppercase (except quoted literals).
-XRef
Produce listing with cross reference of data/procedure names.
-No_XRef
Negation of -XRef.
-XREFS
Specifies that the xref map contain only the symbols that are actually used. (SPL, F77, MODULA)
-XRefSort
Like -XREF except sorted alphabetically. (CBL)

3. ARCHITECTURE

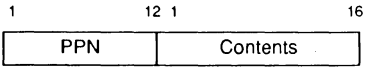
3.1. Argument Pointer (AP)



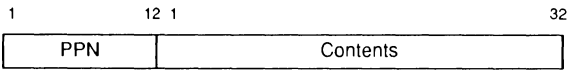
Field	Description	Octal	Hex
Bit	Bit Number	170000	F000
I	Indirect	004000	0800
BR	Base Register: 00 - PB 01 - SB 10 - LB 11 - XB	001400	0300
L	Last AP in argument list	000200	0040
S	Store this argument	000100	0020
Word	Word displacement from Base Register		

3.2. Cache entries

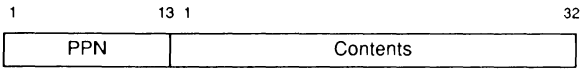
450, 250-II, 550-II, 2250:



750, 850, 2350, 2450, 2550, 9650:



2755, 4150, 6350, 6550, 9750, 9950, 9955:



3.3. Checks

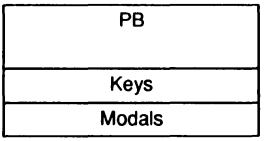
Header	Handler	Type of check
4/200	4/204	Fail, Environment on 9650, 9750, 9950, 9955.
4/270	4/274	Memory parity(ECCU, ECCC)
4/300	4/304	Machine check(MCHK)
4/310	4/314	Missing memory module(MMOD)
4/320	4/324	Recoverable machine check (9955)

On entry to fault handler, mode=64V, MCM=0 for all but ECCC, for which MCM = MCM-at-check - 1, and recoverable machine check(MCM = 2).

MMOD interrupts any other check in progress.

MCHK and ECCU interrupt ECCU in progress if MCM = 2 (QUIET).

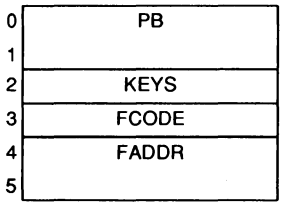
3.3.1. Check header



3.4. Concealed Stack/Queue

Valid only between time of fault and subsequent CALF instruction.

PCB+74 FIRST
PCB+75 NEXT
PCB+76 LAST



PB, KEYS are those of procedure causing the fault.

3.5. Diagnostic Status Word (DSW)

Register mapping:

DSWRMA R34
DSWSTAT R35
DSWPB R36
DSWPARITY R27

3.5.1. DSWSTAT

3.5.1.1. 6350, 6550

DSWSTATH:

Bit	Description	Octal	Hex
1	Check immediate	100000	8000
2	Machine check	040000	4000
3	Memory parity (ECC)	020000	2000
4	Missing memory module	010000	1000
5	E (Execute) unit parity error	004000	0800
6	IS (Instruction/Storage) unit reported parity error	002000	0400
7	CS (Control Store) unit reported parity error	001000	0200
8	MC (Memory Controller) reported error	000400	0100
9	ECCU - ECC Uncorrectable (if bit 3 on)	000200	0080
10	ECCC - ECC Correctable (if bit 3 on)	000100	0040
11	Reserved	000040	0020
12	RCM parity error reported by CS board	000020	0010
13-14	RP backup count (subtract from DSWPB)	000014	000C
15	Check occurred during DMx operation	000002	0002
16	Check occurred during I/O	000001	0001

DSWSTATL:

Bit	Description	Octal	Hex
1-7	ECCC Syndrome[1]: 000 No error 001 CB0 057 16 147 8 002 CB1 061 18 150 9 004 CB2 062 19 153 12 007 1 064 21 155 14 010 CB3 067 24 156 15 020 CB4 070 25 160 17 040 CB5 073 28 163 20 043 4 075 30 165 22 045 6 076 31 166 23 046 7 100 CB6 171 26 051 10 141 2 172 27 052 11 142 3 174 29 054 13 144 5 177 32	177000	FE00
8	Low order address bit of module in error	000400	0100
9	DSWRMA is invalid	000200	0080
10	Recoverable machine check	000100	0040
11	Hard error (permanent error that should be fixed)	000040	0020
12	Unused	000020	0010
13	Internal microcode error. Algorithm code in DSWRMA	000010	0008
14	Processor in dual configuration	000004	0004
15	Slave processor reported error	000002	0002
16	Memory bus that had the error (=bit 14 of address)	000001	0001

[1]MB - Multibit; RP - Righ Parity; CB*n* - Check Bit *n*

3.5.1.2. 9750, 9950, 9955

DSWSTATH:

Bit	Description	Octal	Hex
1	Check immediate	100000	8000
2	Machine check	040000	4000
3	Memory parity (ECC)	020000	2000
4	Missing memory module	010000	1000
5	E1 board parity error	004000	0800
6	F or S unit reported parity error	002000	0400
7	I unit reported parity error	001000	0200
8	Memory controller reported parity error	000400	0100
9	ECCU - ECC Uncorrectable (if bit 3 on)	000200	0080
10	ECCC - ECC Correctable (if bit 3 on)	000100	0040
11	Control store reported parity error	000040	0020
12	RCM parity error reported by CS board	000020	0010
13-14	RP backup count (subtract from DSWPB)	000014	000C
15	Check occurred during DMx operation	000002	0002
16	Check occurred during I/O	000001	0001

DSWSTATL:

Bit	Description	Octal	Hex
1-7	ECCC Syndrome[2]:	177000	FE00
	000 No error		
	001 CB0 057 16 147 8		
	002 CB1 061 18 150 9		
	004 CB2 062 19 153 12		
	007 1 064 21 155 14		
	010 CB3 067 24 156 15		
	020 CB4 070 25 160 17		
	040 CB5 073 28 163 20		
	043 4 075 30 165 22		
	045 6 076 31 166 23		
	046 7 100 CB6 171 26		
	051 10 141 2 172 27		
	052 11 142 3 174 29		
	054 13 144 5 177 32		
8	Low order address bit of module in error	000400	0100
9	DSWRMA is invalid	000200	0080
10	9955: recoverable mach check	000100	0040
11-16	Unused	000077	003F

[2]MB - Multibit; RP - Righ Parity; CB*n* - Check Bit *n*

3.5.1.3. 2250, 2550, 9650

DSWSTATH:

Bit	Description	Octal	Hex
1	Check immediate	100000	8000
2	Machine check	040000	4000
3	Memory parity (ECC)	020000	2000
4	Missing memory	010000	1000
5-7	Machine Check Code (Valid if bit 8=1): 0 none 1 Peripheral Data (BPD) parity 2 Memory Data (BMD) parity 3 Cache Data (RCD) 4 Peripheral Addr (BPA) parity 5 STLB parity 6 Memory Address (BMA) parity 7 A-board parity	007000	0E00
8	Not control unit (RCM) parity	000400	0100
9	ECCU - ECC Uncorrectable error	000200	0080
10	ECCC - ECC Correctable error	000100	0040
11	RP backup count (BUP) is invalid	000040	0020
12-14	RP backup count (subtract from DSWPB)	000034	001C
15	Check occurred during DMx	000002	0002
16	Check occurred during I/O operation	000001	0001

DSWSTATL:

Bit	Description	Octal	Hex
1-5	ECCC Syndrome[3] 000xxx MB 100xxx MB 004xxx MB 104xxx 7 010xxx MB 110xxx MB 014xxx 15 114xxx 3 020xxx MB 120xxx MB 024xxx 14 124xxx 2 030xxx 13 130xxx 1 034xxx 9 134xxx CB2 040xxx MB 140xxx 8 044xxx MB 144xxx 6 050xxx MB 150xxx 5 054xxx 12 154xxx CB5 060xxx 16 160xxx 4 064xxx 11 164xxx CB4 070xxx 10 170xxx CB3 074xxx RP,CB1 174xxx No error	174000	F800
6	OP -- Overall Parity	002000	0400
7	Unused	001000	0200
8	Low order address bit of module in error	000400	0100
9	DSWRMA contents invalid	000200	0080
10-16	Unused	000177	007F

[3]MB - Multibit; RP - Righ Parity; CB*n* - Check Bit *n*

3.5.1.4. All other 50 series

DSWSTATH:

Bit	Description	Octal	Hex
1	CI - Check Immediate	100000	8000
2	MC - Machine Check	040000	4000
3	MP - Memory Parity (ECC)	020000	2000
4	MM - Missing Memory	010000	1000
5-7	Machine Check Code (Valid if bit 8=1): 0 Peripheral Data (BPD) Output 1 Peripheral Addr (BPA) Input 2 Memory Data (BMD) Output 3 Cache Data (RCD) 4 Peripheral Addr (BPA) Output 5 RDX-BPD Input 6 Memory Address (BMA) 7 Register File (RF)	007000	0E00
8	Not RCM Parity (P500, XCS)	000400	0100
9	ECCU -- ECC Uncorrectable Error	000200	0080
10	ECCC -- ECC Correctable Error	000100	0040
11	BUP Invalid -- RP Backup Count Invalid	000040	0020
12-14	RP Backup Count -- Sub from DSWPB	000034	001C
15	Check During DMX	000002	0002
16	IO Bus -- DMX, PIO, µ-code check	000001	0001

DSWSTATL:

Bit	Description	Octal	Hex
1-5	ECCC Syndrome[4] 000xxx MB 100xxx MB 004xxx MB 104xxx 7 010xxx MB 110xxx MB 014xxx 15 114xxx 3 020xxx MB 120xxx MB 024xxx 14 124xxx 2 030xxx 13 130xxx 1 034xxx 9 134xxx CB2 040xxx MB 140xxx 8 044xxx MB 144xxx 6 050xxx MB 150xxx 5 054xxx 12 154xxx CB5 060xxx 16 160xxx 4 064xxx 11 164xxx CB4 070xxx 10 170xxx CB3 074xxx RP,CB1 174xxx No error	174000	F800
6	OP -- Overall Parity	002000	0400
7	Unused	001000	0200
8	Low order address bit of module in error	000400	0100
9	RMA Invalid	000200	0080
10	Unused	000100	0040
11-16	U-Verify Test Number	000077	003F

[4]MB - Multibit; RP - Righ Parity; CB*n* - Check Bit *n*

3.5.2. DSWPARITY

3.5.2.1. 6350, 6550

Bit	Description	Octal	Hex
1	CS: I/O parity error	100000	8000
2	CS: BPD high side, left parity error	040000	4000
3	CS: BPD high side, right parity error	020000	2000
4	CS: BPD low side, left parity error	010000	1000
5	CS: BPD low side, right parity error	040000	0800
6	PIOS: BPA parity error	020000	0400
7	PIOS: BPD parity error	010000	0200
8-10	CS: RCC parity error: FRCCPE(*n*+1)	034000	0700
11	Unused?	000040	0020
12	CS: Decode net high side parity error	000020	0010
13	CS: Decode net low side parity error	000010	0008
14-16	E: parity error 0 No error 1 No erroR 2 BAH parity error 3 BAL parity error 4 BAE parity error 5 BBH parity error 6 BBL parity error 7 BBE parity error	000007	0007
17	MC: lost error	100000	8000
18	Memory address shift control: 0 8 mbyte slot decode 1 16 mbyte slot decode	040000	4000
19	Memory array number 1 reported error	020000	2000
20	Memory array number 2 reported error	010000	1000
21	Memory array number 3 reported error	004000	0800
22	Memory array number 4 reported error	002000	0400
23	Memory array number 5 reported error	001000	0200
24	Memory array number 6 reported error	000400	0100
25	Memory array number 7 reported error	000200	0080
26	Memory array number 8 reported error	000100	0040
27	MC: BB parity error	000040	0020
28	MC: BD parity error	000020	0010
29	MC: BIP in parity error	000010	0008
30	MC: BIP out parity error	000004	0004
31	MC: memory time-out error	000002	0002
32	MC: CIT error	000001	0001

Bit	Description	Octal	Hex
1	IS: BDH left parity error	100000	8000
2	IS: BDH right parity error	040000	4000
3	IS: BDL left parity error	020000	2000
4	IS: BDL right parity error	010000	1000
5-8	Unused	074000	0F00
9	Fatal cache parity error	002000	0080
10	IS: Branch Cache recoverable error	001000	0040
11	IS: cache data parity error: Element B, even data, low byte	000040	0020
12	IS: cache data parity error: Element B, odd data, low byte	000020	0010
13	IS: cache data parity error: Element A, even data, low byte	000010	0008

Bit	Description	Octal	Hex
14	IS: cache data parity error: Element A, odd data, low byte	000004	0004
15	IS: cache index parity error: Element B, low byte	000002	0002
16	IS: cache index parity error: Element B, high byte	000001	0001
17	IS: cache data parity error: Element A, even data, high byte	100000	8000
18	IS: cache data parity error: Element A, odd data, high byte	040000	4000
19	IS: cache data parity error: Element B, even data, high byte	020000	2000
20	IS: cache data parity error: Element B, odd data, high byte	010000	1000
21	IS: cache index parity error: Element A, low byte	004000	0800
22	IS: cache index parity error: Element A, high byte	002000	0400
23	IS: STLB parity error: Element B, physical addr, low byte	001000	0200
24	IS: STLB parity error: Element A, physical addr, low byte	000400	0100
25	IS: STLB parity error: Element B, access bits	000200	0080
26	IS: STLB parity error: Element B, process ID	000100	0040
27	IS: STLB parity error: Element B, virtual address tag	000040	0020
28	IS: STLB parity error: Element A, physical addr, high byte	000020	0010
29	IS: STLB parity error: Element B, physical addr, high byte	000010	0008
30	IS: STLB parity error: Element A, access bits	000004	0004
31	IS: STLB parity error: Element A, process ID	000002	0002
32	IS: STLB parity error: Element A, virtual address tag	000001	0001

3.5.2.2. 9750, 9950, 9955

Bit	Description	Octal	Hex
1	RCC parity error	100000	8000
2	I/O parity error	040000	4000
3-8	Parity error code: If RCC then: 3-5 034000 Encoding of error bits 1-8 6 002000 Logical OR of bits 1-8 7 001000 Error bit 9 8 000400 0 If I/O then: 3 020000 Left byte of BPA or BPD 4 010000 Right byte of BPA or BPD 5 004000 CPU detected BPD parity error 6 002000 CPU detected BPA parity error 7 001000 Cntrlr detected BPD par error 8 000400 Cntrlr detected BPA par error	037400	3F00
9	Unused	000200	0080
10	E1 board detected BBH left byte error	000100	0040
11	E1 board detected BBH right byte error	000040	0020
12	E1 board detected BBL left byte error	000020	0010
13	E1 board detected BBL right byte error	000010	0008
14	E1 board detected BAH right byte error	000004	0004

Bit	Description	Octal	Hex
15	E1 board detected BAL right byte error	000002	0002
16	E1 board detected BAE right byte error	000001	0001
17	BD parity error (memory control unit) 20 010000 BDH left byte 21 004000 BDH right byte 22 002000 BDL left byte 23 001000 BDL right byte	100000	8000
18	Latched memory data parity error (MCU) 20 010000 LMDH left byte 21 004000 LMDH right byte 22 002000 LMDL left byte 23 001000 LMDL right byte	040000	4000
19	Latched memory addr parity error (MCU) 20 010000 MCADDR High byte 21 004000 MCADDR Low byte 22 002000 MCADDR Extended byte 23 001000 Unused	020000	2000
24	ECCU detected by memory control unit	000400	0100
25	I-unit error	000200	0080
26-28	I-unit error location: xxx00x No error xxx02x Unused xxx04x Unused xxx06x Decode net, right byte xxx10x Decode net, left byte xxx12x Base register file high xxx14x Base register file low xxx17x Index register file	000160	0070
29	F or S unit error	000010	0008
30-32	F or S unit error bits: 9955: 0 No error 1 LPID out of STLB in error 2 LBPA out of STLB in error 3 LBVA out of STLB in error 4 ARR out of STLB in error 5 Cache index 6 Cache data, high side 7 Cache data, low side 9750, 9950: 0 PID or STLB control bits 1 LBPA out of STLB in error 2 Cache index, right 16 bits 3 Cache index, left 16 bits 4 Cache data, high side 5 Cache data, low side 6 LBVA out of STLB in error 7 Branch cache parity error	000007	0007

3.5.2.3. 2550, 9650

Bit	Description	Octal	Hex
1	Last memory operation was a wide word	100000	8000
2	Last memory operation was interleaved	040000	4000
3	STLB parity error	020000	2000
4	Cache index parity error	010000	1000
5	Cache data even parity error	004000	0800
6	Cache data odd parity error	002000	0400
7	BMD backplane parity error	001000	0200
8	BPD backplane parity error	000400	0100
9	RFH left byte parity error	000200	0080
10	RFH right byte parity error	000100	0040
11	RFL left byte parity error	000040	0020
12	RFL right byte parity error	000020	0010
13	RFH parity error during late cycle	000010	0008
14	RFL parity error during late cycle	000004	0004
15	BMD or BPD par err read into A-board	000002	0002
16	BMA or BPA par err read into A-board	000001	0001
17-32	Reserved.	177777	FFFF

3.5.2.4. 750, 850

Bit	Description	Octal	Hex
1	RPA parity error, type 1	100000	8000
2	RPA parity error, type 2	040000	4000
3	Burst-mode DMx parity error	020000	2000
4	DMx parity on output if 1, on input if 0.	010000	1000
5-7	J board parity error: 0 - peripheral reports BPD error(output) 1 - base register file high 2 - memory reports BMD error (write) 3 - prefetch buffer address 4 - peripheral reports BPA error (output) 5 - base register file low 6 - memory reports BMA error 7 - prefetch buffer instruction	007000	0E00
8	RCM parity error, if no board error	000400	0100
9	ECCC error (uncorrectable)	000200	0080
10	Prefetch board error	000100	0040
11	BPA input parity error	000040	0020
12	RDX parity error	000020	0010
13	Register file parity error	000010	0008
14	REA parity error	000004	0004
15	DMx cycle parity error	000002	0002
16	AP board parity error	000001	0001
17	C board parity error	100000	8000
18	BMD input parity error, even word	040000	4000
19	BMD input parity error, odd word	020000	2000
20	Missing memory module	010000	1000
21	BMA parity error	004000	0800
22	RMA was incremented at time of error	002000	0400
23	BMA15 indicator	001000	0200
24	BMA16 indicator	000400	0100
25	ECCU error on cache miss	000200	0080
26	ECCC error on cache miss	000100	0040
27	Cache index parity error	000040	0020
28	Cache data odd word parity error	000020	0010
29	Cache data even word parity error	000010	0008

Bit	Description	Octal	Hex
30	Cache cycle purpose: 0 - prefetch 1 - execute	000004	0004
31-32	Unused	000003	0003

3.5.3. DSWRMA
Memory address register. Valid on: ECCU, ECCC, recoverable error (9955), or missing memory.

3.5.3.1. 6350, 6550
32 bit virtual address. On ECCC, ECCU or MISMOD: DSWRMAH = PPN of failing physical memory location; DSWRMAL = 0.

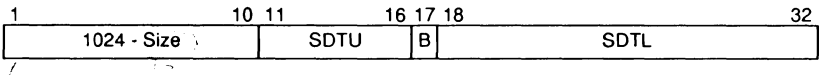
3.5.3.2. 9955
32 bit virtual address. Cleared on cache parity error.

3.5.3.3. 9750, 9950
Bits 1-13 of 23 bit physical address.

3.5.3.4. All other 50 series
32 bit virtual address.

3.5.4. DSWPB
Extended program counter. Always valid.

3.6. Descriptor Table Address Register (DTAR)



Bit	Description	Octal	Hex
Size	Number of SDWs in table	177700	FFC0
SDTU	Bits 1-6 of Physical Addr of Table.	000077	003F
B	Same as bit 18.	100000	8000
SDTL	Bits 7-21 of table physical address (Bit 22 taken as zero.)	077777	7FFF

3.7. Entry Control Block (ECB)

0	Procedure	
1	Base	
2	Stack frame size	
3	Stack root segment	(0 - use current)
4	Disp of arglist in s.f.	
5	Number of arguments	
6	Link	
7	Base	
'10	Keys	

Locations '11 through '17 are set to 0.

3.8. Faults

Locations in current register set:
FCODEH: 26H
FADDR: 27

Fault	#	Offset	Vect	FCODEH	FADDR	Ring Saved	PB
RXM	0	0	62	-	addr	current	backed
PROCESS	1	4	63	ABFLAGS	-	0	current
PAGE	2	10	64	-	addr	0	backed
SVC	3	14	65	-	-	current	backed
UII	4	20	66	cur PBL	addr	current	backed
SEMAPHORE	5	24	67	undfl 0 ovfl 1	sem addr	0	backed
ILL	10	40	72	cur PBL	addr	current	backed
ACCESS	11	44	73	code	addr	0	backed
ARITH	12	50	74	code	addr	current	current
STACK	13	54	75	code	addr	0	backed
SEGMENT	14	60	76	code	addr	0	backed
POINTER	15	64	77	code	ptr addr	current	backed

Code at offset is usually a HLT instruction or a CALF to a fault handling routine.

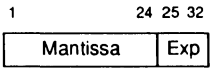
3.8.1. Fault table entry

1	CALF instruction
2	pointer to ECB
3	of fault handler
4	Unused

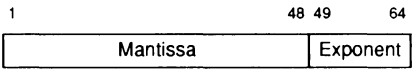
3.9. Floating Point formats

3.9.1. Memory formats

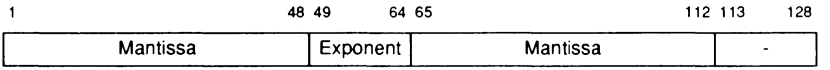
Single precision:



Double precision:

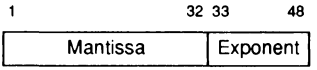


Quad precision:

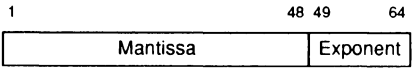


3.9.2. Register formats

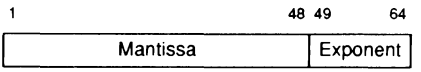
Single precision(2250, 550-II, 650, older machines):



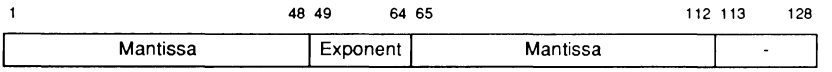
Single precision(750, 850, 9950):



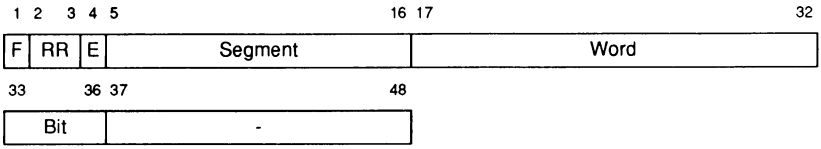
Double precision:



Quad precision:



3.10. Indirect Pointers (IP)



Bit	Description	Octal	Hex
F	Fault bit: 1 - Missing Pointer	100000	8000
RR	Ring Number (00-11)	060000	6000
E	Extension bit: 1 - Word 3 is present with bit offset	010000	1000
Segment	Segment Number	007777	0FFF
Word	Halfword offset	177777	FFFF
Bit	Bit offset within half word; present only if E is set.	170000	F000

As an effective address in a base reg, F and E are ignored.

3.11. KEYS, MODALS

(CRS 24 RFILE 124,164 CRASH 50,150)

KEYSH (Keys):

S, R modes:

Bit	Description	Octal	Hex
1	C Bit(CBIT)	100000	8000
2	Double Precision(DBL)	040000	4000
3	Reserved	020000	2000
4-6	Addressing Mode: x00xxx 16S (0) x02xxx 32S (1) x04xxx 64R (2) x06xxx 32R (3) x10xxx 32I (4) x14xxx 64V (6)	016000	1C00
7	Floating exception (FEX) 0: set CBIT & fault 1: set CBIT	001000	0200
8	Integer exception(IEX) 0: set CBIT 1: set CBIT & fault	000400	0100
9-16	Visible shift count.	000377	00FF

V, I modes:

Bit	Description	Octal	Hex
1	C bit(CBIT)	100000	8000
2	Reserved, must be zero.	040000	4000
3	Link bit (L, LINK)	020000	2000
4-6	Addressing mode: x00xxx 16S (0) x02xxx 32S (1) x04xxx 64R (2) x06xxx 32R (3) x10xxx 32I (4) x14xxx 64V (6)	016000	1C00
7	Floating exception (FEX) 0: set CBIT & fault 1: set CBIT	001000	0200
8	Integer exception(IEX) 0: set CBIT 1: set CBIT & fault	000400	0100

Bit	Description	Octal	Hex
9	Less than condition code (LT, CCLT)	000200	0080
10	Equal condition code (EQ, CCEQ)	000100	0040
11	Decimal exception(DEX)	000040	0020
12	ASCII-8	000020	0010
13	Floating Round if set	000010	0008
14	P850	000004	0004
15	ID (In Dispatcher)	000002	0002
16	SD (Save Done)	000001	0001

KEYSL (Modals):

Bit	Description	Octal	Hex
1	ENB (1: intpts enabled; 0: intpts disabled)	100000	8000
2	VIM (1: vectored int. mode; 0: std int. mode)	040000	4000
3-8	Unused, must be zero.	037400	3F00
9-11	CRS: xxx00x Reg File 2 xxx04x Reg File 3	000340	01E0
12	MIO (1: mapped I/O; 0: unmapped I/O)	000020	0010
13	PXM (1: process exch enabled)	000010	0008
14	SEG (1: segmentation enabled)	000004	0004
15-16	MCM (Machine Check Mode): 0 No reporting 1 Memory Parity, uncorrected (ECCU) 2 Quiet; all unrecovered errors 3 Record; report all errors	000003	0003

3.12. Modals

See KEYS (KEYSL).

3.13. Page maps

3.13.1. HMAP, LMAP

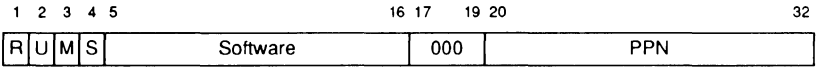
Locations of pagemaps:

Rev	Address
< 19.2	4/4000
19.2	segments 401-420
> 19.2	segments 601-620

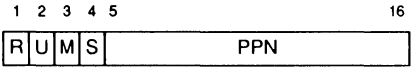
HMAP, LMAP interleaved in 64-word chunks, thus 128 words/segment in system.

HMAP (Hardware Map):

9950, 9955:



Other:



Mnem	Description		Octal	Hex
R	If set, page is resident		100000	8000
U	If set, page has been used		040000	4000
M	If reset, page has been modified		020000	2000
S	If set, page is shared (inhibits cache)		010000	1000
Software	Reserved for software:		007777	0FFF
PPN	Physical Page Number High order 13/12 bits of physical page address.	9950: Other:	017777 007777	1FFF 0FFF

If non-resident, bits 3,5 software defined:

3,5 024000 Page status:
- 000000 Not resident, copy on disk
- 020000 Not resident, no copy on disk
- 004000 In transition, coming in
- 024000 In transition, going out

LMAP (Software Map -- HMAP+'100):

Bit	Description	Octal	Hex
1-2	Lock counter (0 - unlocked)	140000	C000
3	First Time (just paged in)	020000	0200
4	Use alternate paging device	010000	1000
5-16	Record index (1 val/8 pages)	007777	0FFF

3.14. MMAP entry

1	16	17	48
Page Status		Unused	

PAGE STATUS < 0 page is not to be used.
PAGE STATUS = 0 page is available.
PAGE STATUS > 0 page is in use; PAGE STATUS points to an HMAP entry.

3.15. Process Control Block (PCB)

Word	Description	Word	Description
0	Level (in ready list)	40	GR7
1	Link (next PCB in list)	41	"
2	Wait list SN (0 = ready)	42	FP0
3	Wait list word number	43	"
4	Abort flags	44	"
5	Last Reg set used	45	"
6	Reserved	46	FP1
7	Reserved	47	"
10	Elapsed timer (Low)	50	"
11	Elapsed timer (High)	51	"
12	DTAR2H	52	PBH
13	DTAR2L	53	PBL
14	DTAR3H	54	SBH
15	DTAR3L	55	SBL
16	Interval timer	56	LBH
17	Reserved	57	LBL
20	SAVE MASK	60	XBH
21	KEYS	61	XBL
22	GR0	62	Fault vector, RING 0
23	"	63	"
24	GR1	64	Fault vector, RING 1
25	"	65	"
26	GR2	66	Reserved
27	"	67	"
30	GR3	70	Fault vector, RING 3
31	"	71	"
32	GR4	72	Page fault vector
33	"	73	"
34	GR5	74	Concealed stack, First
35	"	75	Concealed stack, Next
36	GR6	76	Concealed stack, Last
37	"	77	Reserved

Word 5 has the following format for 850s:

Bit	Description	Octal	Hex

Bit	Description	Octal	Hex
1-4	Restrict process from ISU: 0000 - no restrictions 0100 - bar from this ISU 1000 - bar from other ISU	170000	F000
5	Reserved	004000	0800
6-7	Last ISU 01 - this ISU 10 - other ISU	003000	0600
8	Register have been saved in PCB	000400	0100
9-11	Last register set used. (same as modals CRS)	000340	00E0
12	Reserved	000020	0010
13-16	Process is lock to ISU: 0000 - neither 0100 - this ISU 1000 - other ISU	000017	000F

3.16. Queue Control Block (QCB)

1 2 4 5 16

Top Pointer		
Bottom Pointer		
V	000	High Order Address
Size Mask		

V - Virtual (not physical) queue

3.17. READY LIST

PPA current level/current PCB
PPB next level/next PCB
LEVEL*n* first PCB on LEVEL*n*
LEVEL*n+1* last PCB on LEVEL*n*
PCB+0 level this PCB is on
PCB+1 next pcb, 0 if last

Ready list in Segment 4, starting at 4/600:

Lev	PCB on level
600	CLOCK PROCESS
602	SMLC PROCESS
604	AMLC PROCESS
606	MPC, MP2 PROCESSES
610	VERSATEC PROCESS
612	IPC PROCESS

Lev	PCB on level
614	RINGNET PROCESS
616	SPARE1, SPARE2 PROCESSES
620	SUPERVISOR PROCESS (USER 1)
622	PRIORITY 3 USER PROCESSES
624	PRIORITY 2 USER PROCESSES
626	PRIORITY 1 USER PROCESSES (NORMAL LEVEL)
630	PRIORITY 0 USER PROCESSES
632	BACKSTOP PROCESS

3.18. Registers

Register file allocations.
2550, 9650:

Register File	Absolute Loc	Use
RF0	'0-'37	Microcode (set 1 for 9650)
RF1	'40-'77	DMA channels (32)
RF2	'100-'137	User register set 2
RF3	'140-'177	User register set 3
RF4	'200-'237	User register set 4
RF5	'240-'277	User register set 5
RF6	'300-'337	User register set 6
RF7	'340-'377	User register set 7
RF8	'400-'437	User register set 8
RF9	'440-'477	User register set 9
RF10		Microcode set 2 for 9650

All others:

Register File	Absolute Loc	Use
RF0	'0-'37	Microcode (set 1 for 9750, 9950, 9955)
RF1	'40-'77	DMA channels (32)
RF2	'100-'137	User register set 2
RF3	'140-'177	User register set 3
RF4	'200-'237	User register set 4 (9750, 9950, 9955)
RF5	'240-'277	User register set 5 (9750, 9950, 9955)
RF6	'300-'337	User register set 6 (9750, 9950, 9955)
RF7	'340-'377	Microcode set 2 for 9750, 9950, 9955

Microcode registers:

9750, 9950, 9955:

Reg num	Contents	Crash addr	Reg num	Contents	Crash addr
0	TR0	300	300	DGR0 (STLBRF1)	200
1	TR1	302	301	DGR1 (STLBRF2)	202
2	TR2	304	302	DGR2 (RDMX1)	204
3	TR3	306	303	DGR3	206
4	TR4	310	304	DGR4	210
5	TR5	312	305	DGR5	212
6	TR6	314	306	DGR6, RSSAV(9955)	214
7	TR7	316	307	DGR7	216
10	TR8,FR032	320	310	DGR10	220
11	TR9	322	311	DGR11	222
12	TR10,FR132	324	312	DGR12	224
13	TR11	326	313	DGR13, FF80(9955)	226
14	REOIV, UCSADDR	330	314	DGR14	230
15	RDSAVE	332	315	DGR15	232
16	CFF00, C00FF	334	316	DGR16	234
17	RATMP	336	317	DGR17	236
20	RMASAVE	340	320	MINUS1	240
21	342		321	ONE32	242
22	PARREG1	344	322	KMASK, IUART	244
23	PARREG2	346	323	C3FF, C3F	246
24	PARREG3	350	324	C8000	250
25	PBSAVE	352	325	C0D0D, CB0B0	252
26	SYSREG1	354	326	C9C00, C0080	254
27	DSWPARITY	356	327	CB1E0, -	256
30	PSWPB	360	330	C6666	260
31	PSWKEYS	362	331	C10K, ACK2	262
32	PLA, PPA	364	332	FERRET6	264
33	PLB, PPB	366	333	FERRET5	266
34	DSWRMA	370	334	FERRET4	270
35	DSWSTAT	372	335	FERRET3	272
36	DSWPB	374	336	FERRET2	274
37	RSAVPTR	376	337	FERRET1	276

2550, 9650:

Reg num	Contents	Crash addr	Reg num	Contents	Crash addr
0	TR0	1000	500	DEC0	1100
1	TR1	1002	501	DEC1	1102
2	TR2	1004	502	DEC2	1104
3	TR3	1006	503	DEC3	1106
4	TR4	1010	504	DEC4	1110
5	TR5	1012	505	DEC5	1112
6	TR6	1014	506	DEC6	1114
7	TR7	1016	507	DEC7	1116
10	RDMX1	1020	510	DEC10	1120
11	RDMX2	1022	511	RECC3	1122
12	USCADDR, REOIV	1024	512	TMRSAVE	1124
13	RSGT1	1026	513	CTRLBYTE, QFDIDX	1126
14	RSGT2	1030	514	CMDBYTE, SCR14L	1130
15	RECC1	1032	515	EXP32	1132
16	RECC2	1034	516	SSN	1134
17	TEMPCACH	1036	517	SWITCHES, PICSTAT	1136
20	ONE32	1040	520	WWADTR	1140
21	PBSAVE	1042	521	ADRREG2	1142
22	RDMX3	1044	522	ADRREG	1144
23	RDMX4	1046	523	LIGHTS, INTVEC	1146
24	C377	1050	524	QPTR, BYTFLG	1150
25	MINUS1	1052	525	WSLFLG	1152
26	LREGSET, CHKREG	1054	526	RDMX5	1154
27	DSWPARITY	1056	527	UMASK1, SCR27L	1156
30	PSWPB	1060	530	UMASK2, SCR30L	1160
31	PSWKEYS	1062	531	URDRXH, SCR31L	1162
32	PPA	1064	532	BFR04	1164
33	PPB	1066	533	DSSW	1166
34	DSWRMA	1070	534	RSTLB1	1170
35	DSWSTAT	1072	535	RSTLB2	1172
36	DSWPB	1074	536	RSTLB3	1174
37	RSAVPTR	1076	537	RSTLB4	1176

Other 50 series:

Rfile addr	Contents	Crash addr
0	TR0	300
1	TR1	302
2	TR2	304
3	TR3	306
4	TR4	310
5	TR5	312
6	TR6	314
7	TR7(PB)	316
10	RDMX1	320
11	RDMX2	322
12	USCADDR (750,850)/REOIV	324
13	RSGT1	326
14	RSGT2	330
15	RECC1	332
16	RECC2	334
17	-/RATMPL	336
20	ZERO/ONE	340
21	PBSAVE	342
22	RDMX3	344
23	RDMX4	346
24	C377	350
25	MINUS1/MINUS2	352
26	WWADTR	354
27	DSWPARITY(>750)	356
30	PSWPB	360
31	PSWKEYS	362
32	PPA/PCBA	364
33	PPB/PCBB	366
34	DSWRMA	370
35	DSWSTAT	372
36	DSWPB	374
37	RSAVPTR	376

RFIL ADDR = Address in Register File
CRASH ADDR = Disp in hardware register save area.

TR7	PB at machine halt
PSWPB	PB at last interrupt
PSWKEYS	Keys at last interrupt
PPA	Current level/current PCB
PPB	Next level/next PCB
RSAVPTR	Reg save area ptr. 0: regs saved.
'33	850 only: 41004 - this ISU; 102010 - other ISU

Register set

Reg num	I-mode	V-mode	R-mode	Rel crash addr
0	GR0			0
1	GR1			2
2	GR2	L,A/B	A(1)[5]/B(2)	4
3	GR3	E		6
4	GR4			10
5	GR5	-/Y	-/S(3)	12

[5](n) indicate P300 address mapping

Reg num	I-mode	V-mode	R-mode	Rel crash addr
6	GR6			14
7	GR7	-/X	-/X(3)	16
10	FAR0,FAC0L	FAR0	('13/-)	20
11	FLR0	FLR0		22
12	FAR1	FAR1	(4/5)	24
13	FLR1	FLR1	(6/-)	26
14	PB	PB	PB	30
15	SB	SB	SB('14/'15)	32
16	LB	LB	('16/'17)	34
17	XB	XB	XB	36
20	DTAR3	DTAR3	DTAR3('10/-)	40
21	DTAR2	DTAR2	DTAR2	42
22	DTAR1	DTAR1	DTAR1	44
23	DTAR0	DTAR0	DTAR0	46
24	KEYS/MODALS	KEYS/MODALS	KEYS/MODALS	50
25	OWNER	OWNER	OWNER	52
26	FCODE	FCODE	FCODE('11/-)	54
27	FADDR	FADDR	FADDR(-/'12)	56
30	TIMER	TIMER	TIMER	60
31				62
32				64
33				66
34				70
35				72
36				74
37				76

See the *System Architecture Reference Guide* [65] for additional information.

3.19. RSAV format

Registers as saved/restored by the RSAV/RRST instructions.

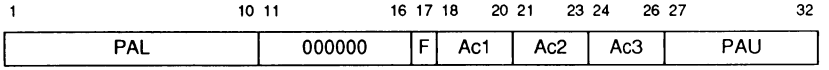
0	SAVE MASK
1	FRN1
2	
3	FR1
4	
5	FRN0
6	
7	FR0
10	
11	GR7
12	
13	GR6
14	
15	GR5
16	
17	GR4
20	
21	GR3
22	
23	GR2
24	
25	GR1
26	
27	GR0
30	
31	XB
32	

Save Mask:

Mnem	Description	Octal	Hex
1-4	Unused	170000	F000
5	FRN1	004000	0800
6	FR1	002000	0400
7	FRN0	001000	0200
8	FR0	000400	0100
9	GR7	000200	0080
10	GR6	000100	0040
11	GR5	000040	0020
12	GR4	000020	0010
13	GR3	000010	0008
14	GR2	000004	0004
15	GR1	000002	0002
16	GR0	000001	0001

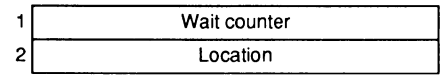
The XB is always saved.

3.20. Segment descriptor word (SDW)



Mnem	Description	Octal	Hex
PAL	Bits 7-22 of physical addr of a PMT or HMAP entry; bits 17-22 must be zero	177700	FFC0
F	Fault, 1 = No segment or missing pagemap	100000	8000
Ac1	Access controls for Ring 1: 000 No access 001 Gate 010 Read 011 Read/Write 100 Reserved 101 Reserved 110 Read/XEQ 111 R/W/XEQ	070000	7000
Ac2	Access controls for Ring 2 (not used)	007000	0E00
Ac3	Access controls for Ring 3	000700	01C0
PAU	Bits 1-6 of physical addr of a PMT or HMAP entry.	000077	003F

3.21. Semaphores



Word	Description
Wait counter	Number of outstanding waits: 0 - empty list <0 - uneventful notifies >0 - PCBs waiting
Location	Location of first PCB in OWNERH segment

3.22. Stack frame

0	0:PCL 1:CALF
1	SN of stack root
2	PB for
3	return
4	Caller's
5	SB
6	Caller's
7	LB
10	Caller's KEYS
11	PBCL
12	FCODE if CALF[6]
13	FADDR if CALF
14	
15	
16	Reserved
17	

3.23. Stack Headers

Stack root header (word 0 of stack segment)

0	Free pointer
1	
2	First extension
3	

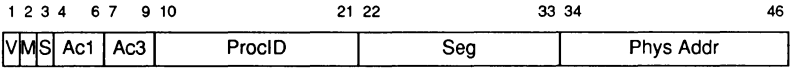
Stack extension header (word 0 of stack segment)

0	0
1	0
2	Next extension
3	

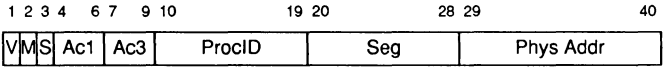
[6]Start of automatic storage if PCL. See section 4.11

3.24. STLB

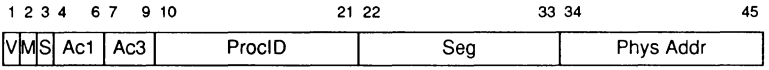
9750, 9950, 9955:



2550, 9650:



All others:



Mnem	Description
V	STLB has valid data.
M	Page has been modified.
S	This memory is shared.
Ac1	Ring 1 access rights.
Ac3	Ring 3 access rights.
ProcID	ID of the process referencing this memory.
Seg	Segment number of the virtual address.
Phys Addr	The physical page address.

STLB

DTAR segment page - Physical page #

Cache PFN found 4 HI

4. PRIMOS

4.1. ABORT FLAGS

PCB+4, ABSAVE at 6000/20

Bit	Description	Octal	Hex
MINALM	One minute abort flag	100000	8000
SMLALM	SLMC alarm	040000	4000
LGIALM	Login alarm	010000	1000
WRMALM	Warm start alarm	004000	0800
MSGALM	User 1 message alarm	002000	0400
CHKALM	Check alarm	001000	0200
SWIALM	Software interrupt alarm	000200	0080
IOALM	I/O completed alarm	000100	0040
IOMALM	IO$MSG alarm	000080	0020
DISALM	Disconnect alarm	000010	0008
TMOALM	Timeout alarm	000004	0004
TSEALM	Time slice end (firmware)	000001	0001

4.2. EPF formats

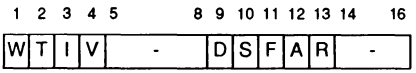
VCIB:

0	0	
1	-1	
2	Type	Version
3	Resume segs	
4	Linkage areas	
5	Debugger segs	
6	CIB	
7	ERP	

Type	Description
1	Program.
3	Process class library (initialized once per process).
5	Program class library (initialized once per program invocation).
6	Registrable program (may be registered).
8	Registrable process class library.
10	Registrable program class library.

CIB:

0	Version	Size
1	Starting	
2	ECB ERP	
3	LTD list	
4	ERP	
5	LIB	
6	ERP	
7	-	
10	Map	
11	ERP	
12	DBG info	
13	ERP	
14	Merge segs	
15	Merge info	
16	ERP	
17	CP Flags	
20	Name gen pos	
21	-	
22	-	
23	AIB	
24	ERP	
25	Module body	
26	ERP	
27	Basic-86 info	
30	ERP	



Bit	Description	Octal	Hex
W	Wildcarding enabled.	100000	8000
T	Treewalking enabled.	040000	4000
I	Iteration enabled.	020000	2000
V	Verification of wildcaring enabled.	010000	1000
D	Directories.	000200	0080
S	Segment directories.	000100	0040
F	Files.	000040	0020
A	Access categories (ACATs).	000020	0010
R	Recovery based file types only.	000010	0008

4.3. FIGCOM

Starts at 14/700

Name	Dflt	Definition
LOUTQM	1000	Inactive minutes to auto logout
LOTLIM	3	Inactive logout time during login
DONSTP	0	Phantom restart flag for warmstart 0 => Logout phantoms on warm start 1 => Continue phantoms on warm start
CSSEGS	-	Number of concealed(?) stack segs.
DEFERA	242	Default erase character (")
DEFKIL	277	Default kill character (?)
PRI500	-	1 => P500
VERSIO	-	PRIMOS revision id (char(16)var)
NLGPRT	1	1 => Inhibit login messages at console
LOGOVR	0	1 => Can't login while logged in (login-over-login)
LRQUOT	10000	LOGREC quota (obsolete at 21.0)
DMQMSK	-	= 157777 to disable DMQ-AMLC logic
CPUID	-	CPU model number
INSTLB	-	1 => Can use LIOT and PTLB instructions
APCNFG	-	1 => P850 cpu in use
UPSSW	-1	UPS config -1 => No UPS 0 => Halt on warm-start >0 => n seconds delay after warmstart
CPUREV	-	CPU U-CODE REV
STAMP	-	((15)bin)
RWLOCK	1	System read/write lock: 0 - 1 reader or 1 writer 1 - n readers or 1 writer 3 - n readers and 1 writer 5 - n readers and n writers
ABBRSW	1	Abbreviation enabled flag 1 => Abbrevs enabled 0 => Abbrevs disabled
SLVRUN	-	1 => P850 slave CPU running
DTRDRP	0	1 => Drop DTAR at logout
ZCPU	-	1 => can use ZMOV and ZFIL
STTMCP	-	1 => can use STTM
MAPREV	-	Rev of page map format
RGSETS	-	Number of user reg sets
RGSET0	-	Number of ucode reg set
ECCTRL	-	Memory controller ECC
BCLOCK	-	Battery clock with cpu
SENSOR	-	Environmental sensors
MEMHLT	1	Halt on ECCU.
DISPCH	-	'in dispatcher' bit always set.
LOGBAD	0	Monitor failed logins.
DEFMEL	-	Default minimum extent length
R0DBG_ON	0	Ring 0 debugger configuration flag
SUSPEND_SLAVE	-	Debugger variable
TPDUMP	-	Take tape dumps

FIGCME:

Name	Dflt	Definition
HIOSEG	-	Highest I/O sindow segment

PRMENG	0	Prime engineering flags 100000 AD & SH usable by Sys Admin 040000 Use long login prompts 037777 reserved
STBCLK	-	PRIMOS can set the battery clock
MIRROR	0	Mirroring enabled

4.4. LOCKS, LCKCOM

Locks are semaphores used to control acccess to serially reusable resources. Located in LCKCOM (SEG 6), source file N1LOCK.

Lock	Description
FSLOK	File system lock
UFDLOK	Directory lock
BLKLOK	Block I/O lock (21.0)
SDLOK	Segment directory locks
TRNLOK	Transaction locks
UTLOK	Unit table lock
RATLOK	DISKRAT lock
DEVLCK	Device lock
NSSLCK	NSS database lock (21.0)
NETLCK	Network lock
NMMLCK	Network memory mapping lock
SLCLCK	SMLC lock
MOVLCK	Segment movement lock
EPFLCK	REPF database lock
SEGLCK	GETSN$/RTNSG$ lock
PAGLCK	Page fault lock
LOCSEM	Mutual excl. semaphore for locate
SEMSEM	Mutual excl. semaphore for name semaphores
BCBLOK	ROAM BCB lock
WCBLOK	ROAM WCB lock
SAL_SEM	Semaphore for system class storage
LON$SEM	Logout notification semaphore
IPC$SEM	IPC mutual excl. lock
LON$STA	Logout notification area pointer
SYST_HCB	Heap control block for system class storage
LOCUSR	Owner of LOCSEM
SEMUSR	Owner of SEMSEM
BCBUSR	Owner of BCBSEM
WCBUSR	Owner of WCBSEM
SAL_USER	Owner of SAL_SEM
ISCSEM	Mutual excl. semaphore for ISC database
ISCUSR	Owner of ISCSEM
REGSEM	Semaphore for ISC registration database
REGUSR	Owner of REGSEM
SHRSEM	SHSEGT lock (ISC/SNA shared segments)
SHRUSR	Owner of SHRSEM

4.5. PTUSEG

PTUSEG(2,KSEG) (SEG 14)

PTUSEG(1,N) Owner of Page Map N
PTUSEG(2,N) Segment Number for Page Map N

4.6. PUDCOM

Starts at 6000/0.

Name	Dflt	Definition
0	PGFFRE	Next frame pointer (ptr)
2	PGFEXT	Stack extension pointer (ptr)
4	-	Reserved
6	PGFSPB	Saved return PB on page fault stack (ptr)
10	CUSR	Current user number
11	PCBUSR	PCB index
12	UTLBPT	Pointer to unit table (ptr)
14	VRTSSW	Virtual sense switches
15	LITE	User virtual lights
16	DSKUSE	Current disk request
17	INHPRF	Inhibit-process-fault counter
20	ABSAVE	Saved abort flags; see 4.1.
21	LOKOWN	N1-locks owner table
22	OWNFS	Owner count for FSLOK
23	R3QUIT	Ring 3 quit inhibit count (> 0 - quits inhibited)
24	R1QUIT	Ring 1 quit inhibit count (> 0 - quits inhibited)
25	PRVL	Master privilege word 000010 0008 NSS priviledged (NSSPRV) 000004 0004 SNA priviledged (SNAPRV) 000001 0001 Priviledged (PRVBIT)
26	ASRCWD	ASR controls
27	COMSWI	Command input switch 100000 On (1 - on) 060000 Read state 00 - read left character next 10 - read right character next 01 - tab expans. in progress read left character next 11 - tab expans. in progress read right character next 010000 Last character was LF 007400 Unused 000377 Character saved
30	COMUNI	Command input unit
31	COUSWI	Command output switch
32	COUPTR	COULIN character pointer
33	COULIN	Command output buffer (char(20))
45	ERRVEC	Error vector ((9) bin)
56	SWITYP	Pending software interrupts (See 4.10)
57	MSGCTL	Message control 100000 Global message pending 040000 Personal message pending 020000 Rejecting messages 010000 Deferred messages only 001000 Message was sent by a process
60	MYPDB	Pointer to process data block (ptr)
62	HMAPSK	Pointer to stack page map (ptr)
64	BUFNEW	Index of current locate buffer
65	XSAVE	Temp save for X in fault entrances
66	LSAVE	Temp save for L in process fault (bin(31))
70	RHDBUF	Record header buffer ((16)bin)
110	USRETM	Remainder of eligibility time-quantum
111	USRTS	Default user timeslice
112	CURRTS	Current user timeslice
113	LOUTCK	Auto logout clock
114	CPLKCT	Master CPU preference count
115	WTTIME	WAIT$T time
116	IOUSED	I/O time used

120	TIMDOG	Real time watchdog timer (bin(31))
122	CPUDOG	CPU time watchdog timer (bin(31))
124	SLNODE	Slave's master's node id
125	SLMAST	Slave's master's user number
126	FLAGBT	Various flag bits 100000 8000 Named semaphore sync. 040000 4000 Software interrupt not allowed 020000 2000 More than 256 DTAR 2 segments 010000 1000 ECCU logout pending 004000 0800 Semaphore aborted switch 002000 0400 User is doing a fork
127	SWIDEF	Software interrupts already deferred (See 4.10)
130	CURPRI	Current priority level
131	LIDATE	Date of last login (FS format)
132	LITIME	Time of last login (FS format)
133	TABCNT	Tab expansion count (USED BY C1IN$)
134	R0QUIT	Ring 0 quit enable count (> 0 - quits enabled)
135	R0SWIN	Ring 0 software interrupt enable flags (See 4.10)
136	R1SWIN	Ring 1 software interrupt enable flags (See 4.10)
137	R3SWIN	Ring 3 software interrupt enable flags (See 4.10)
140	TRNLH	Transaction lock hash address
141	SDLKH	Offset of hashed SDLOK
142	CPLIM	CPU time limit (bin(31))
144	LOGLIM	Login time limit (bin(31))
146	ORGUTP	Original user type (used by NLOGIN)
147	AWEFLG	Asyncronous write error list/flag
150	SDW3	DTAR 3 SDWs ((0:76) bin(31)) (varies rev to rev)
*	SDW2	DTAR 2 SDWs ((0:191) bin(31)) (varies rev to rev)
*	PGFSF	Start of 1st page fault frame ((64) bin)
1300	PUDEND	End of PUDCOM

4.7. SEGMENT USAGE BY PRIMOS

SEGMENT	CONTENTS
0	I/O MAP SEGMENT
1	LOCATE BUFFER SEG
2,3	TEMP SEG - INTERUSER MOVES
4	CHECKS, TRAPS, PX, ETC.
5	RING 0 GATES
6	RING 0 KERNAL CODE, LINKAGE
7	LOW SPEED I/O BUFFERS
10	FILE SYSTEM DATA STRUCTURES
11	FILE SYSTEM CODE, LINKAGE
12	NETWORK CODE, LINKAGE
13	COMMAND LOOP SEGMENT 1
14	COLD & WARM START, SDW0,1, etc.
15	SECOND SEG FOR KERNAL CODE, LINKAGE
21	USED BY DMQ BUFFER
22	PAGE MAP SEGMENT
23	SMLC COPY SEGMENT
24	SMLC COPY SEGMENT
25	SMLC COPY SEGMENT
26	SMLC COPY SEGMENT
27	NETWORK BUFFERS
30	NETWORK QUEUES/BHA's
31	NETWORK
32	COMMANDLOOP SEGMENT 2

33 LOGICAL PAGE MAP SEGMENT
34 NAMED SEMAPHORE SEGMENT
35 INTERPROCESS COMMUNICATION SEGMENT
36 SECOND AMLC BUFFER SEGMENT
37 SCL DATABASES
40 UNIT TABLE ENTRIES SEGMENT
6000 Page fault stack and PUDCOM (ring 0)
6001 NPX data
6002 Ring 3 data (CLDATA)
6003 Per user unwired ring 0 stack

4.8. Shared Segments

Segment	Contents
2000	ED (Editor)
2001-2003	DBMS
2004-2011	SPSS, Scicards
2012	DBMS
2013	BASICV
2014	Shared libraries (obsolete): *Area* — *Product* 100-277 — COBOL LIB 300-377 — MIDAS LIB 1000-37777 — COBOL LIB 40000-177777 — MIDAS LIB
2015	DPTX
2016	COBOL
2017	BASIC/VM
2020	MIDASPLUS writable shared segment (177762-177777)
2021	FORMS library
2022-2023	PLP
2024-2025	POWERPLUS
2026-2027	FTS
2030-2037	Reserved for customers. In-house only: 2030-2032 Software Tools 2033 STAR TREK; LISP; IFPS 2034 X.ED; IFPS 2035 X.SE 2036 Lyne Smith memorial segment 2037 OS experiments (Bootleg)
2040-2042	DBG
2043	SPSS, Scicards
2044	DSM
2045-2056	-
2057-2065	OAS (until rev 6.0)
2066-2067	-
2070	DBMS
2071	OAS (until rev 6.0)
2072	SPSS, Scicards
2073-2077	DISCOVER
2100	EDMS
2101	OAS (until rev 6.0)
2102-2114	EDMS
2115	DBG
2116-2121	SPL (until rev 20.2)
2122-2125	MIDASPLUS

Segment	Contents
2126-2127	FTS
2130-2137	MEDUSA
2140	EDMS, BP99
2141-2150	-
2151-2153	FED
2154-2161	CBL
2162-2163	EDMS, BP99
2164-2166	SPICE
2167	SPOOL
2170-2177	Reserved for Customers, SCICARDS 2170-2171 INFO 2172-2175 MDS 2176 IFPS 2177 X.ED
2200-2203	ROAM
2204-2207	PRISAM
2210-2215	ESCAPE34, TAPS
2216	TAPS
2217-2220	ROAM
2221	-
2222	MAGLIB (until 21.0)
2223-2224	ROAM
2225	DBMS
2226	ESCAPE34
2227	PRISAM
2230-2267	PRIMEWAY
2270-2276	INFORMATION
2277	PRISAM/DISCOVER
2300-2317	Reserved for customers
2310-2317	THEMIS
2320-2321	MIDASPLUS
2322	-
2323	PRIMEWAY
2324-2327	C (CC)
2330-2337	INFORMATION
2340	INFORMATION
2341-2347	EMACS
2350-2355	INFORMATION/PDGS
2356-2367	PDGS
2370-2372	(Terminal simulator - Steve Sohn)
2370-2376	MEDUSA
2377	SNA
2400-2427	PDMS
2430-2442	THEMIS
2443	EDMS
2444-2447	-
2450-2467	PRIMEWAY
2470-2475	INFORMATION/CONNECTION
2476	SNA RJE
2477	-
2500-2521	ORACLE
2522-2534	-
2535	CBL
2536-2547	-
2550-2556	C
2557-2564	PDGS
2565-2567	-
2570-2573	ESCAPE34
2574-2575	
2576	DBG
2577	-
2600-2601	ROAM/DDM
2602-2665	-

Segment	Contents
2666-2765 6001	EPFs Per user linkage segment: Allocated Product 0-32777 FORMS 33000-40777 VCOBLB (obsolete) 41000-66777 MIDAS (obsolete) 67000-67767 SPOOL (obsolete at 21) 67770-67777 BATCH 70000-105777 FORMS 106000-112777 ED 113000-117777 NPX 120000-131777 ABBREV 132000-177777 FORMS (V-FTNLIB < 19.4)
6006	Per user data: Allocated Product 0-37777 FTS (QPAKS) 40000-70000 MIDASPLUS 70001-77777 - 100000-177777 ROAM/DDM
6007	Per user data: Allocated Product 0-47777 ROAM 50000-122777 PRISAM 123000-137777 - 140000-177777 MAGLIB (until 21.0)
6010 6011	ORACLE, EMACS, PRIMEWAY Per user data: 0-177777 ROAM

4.9. Semaphore allocation

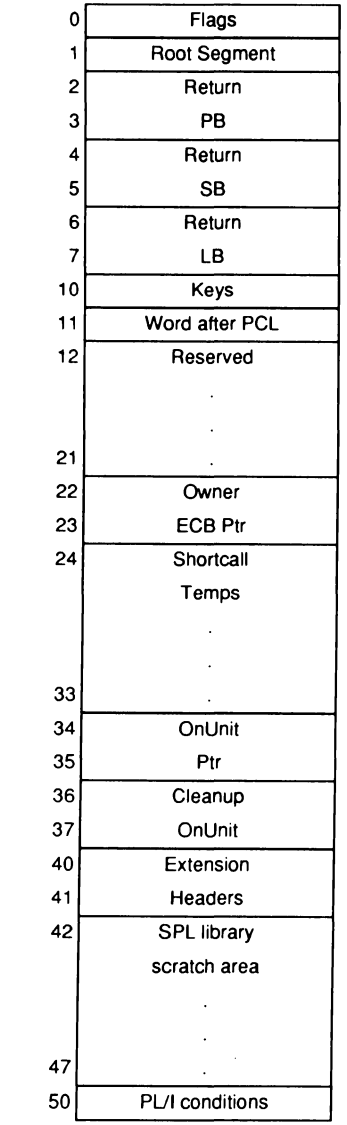
-1 to -10 DBMS
-12 to -15 QPAKS
-16 MIDAS
-63 to -64 SPOOL

4.10. Software interrupt flags

Bit	Description	Octal	Hex
QUTINT	Terminal quit	100000	8000
CPUINT	CPU watchdog timer	040000	4000
TIMINT	Real time watchdog timer	020000	2000
LOGINT	Forced logout	010000	1000
LONINT	Logout notification	004000	0800

Bit	Description	Octal	Hex
CPSINT	Cross process signalling	002000	0400
IPCMWI	IPC message waiting	001000	0200
WRMINT	Warmstart software interrupt	000400	0100
PXCPSINT	Primix cps interupt	000200	0080

4.11. Software Stack Frame



Bit	Description	Octal	Hex
I	Inhibit crawlout-backup of PB	100000	8000
C	This is a condition frame	040000	4000
D	Cleanup has been done for this frame	020000	2000
X	Extension to frame exists	010000	1000
U	User Procedure	004000	0800
S	Stack conains valid condition bits	002000	0400
L	This is a library procedure	001000	0200
E	ECB contains valid condition bits	000400	0100
FF	Fault frame indicator (if '01'b)	000003	0003

4.12. SVC interlude

```
ENTRY DAC     **
      SVC
      OCT     CODE
 1    100000  1 => interlude call
 2    040000  1 => bounce
3,4   030000  Unused
5-16  007777  SVC number
```

4.13. UPCOM

User Profile COMmon.

Offset	Name	Description
0	projid	Project id (char (32))
20	comnum	Number of command levels
21	epfinvoc	Number of EPF invocations
22	nstat2	Number static dtar2 segs
23	ndynm2	Number of dynamic dtar2 segs
24	dynsgs	Start of dynamic dtar segment ranges((0:3) bin)
30	difns(16)	Remote id information nodnam_ptr Pointer to the node name in nnt of this entry (ptr) user_id The user id for this node (char(32) var) password The password for this user id (char(16) var) project_id The project to login under (char(32) var)
1350	difns_count	Number of entries in use
1351	vcdata(16)	State info for each active NPX virtual circit vcid Virtual circuit id for use with IPCF node Node number spare Save it for namtab ptr vcstat Virtual circuit status words for IPCF ((2) bin) ns Npx message sequence number to send next (mod 8) nr Npx message seq number last received over this VC alocnt Allocation count for this slave slavno For slave's id (char(6)) flags: 000004 firstime: shared by R$ALOC R$CALL R$BGIN 000002 receive_posted: set when there is 1 rcv pending 000001 inprog: set while a RPCL is still pending
1651	npxvc	NPX VC active in TRNRCV
1652	npxany	Store any_handler entry (entry variable)
1654	upend	End of upcom

4.14. USRCOM

```
SEG 10/(USRNO-1)*'505 ...

+1    UNITAB:       0 ≤ UNIT ≤ '21
         +UNIT*17 +0  VSTAT
                  +1  VBRA(2)
                  +3  VDVNO
                  +4  VDCRA(2)
                  +6  VDRWP
                  +10 VCRA(2)
                  +12 VRWP
                  +13 VPRIV
                  +14 VPOPRA(2)
                  +16 VPOPRW
+417 CUFD:
      +0  CURRENT UFD NAME
      +20 CFDBRA(2)
      +22 CFDDEV
      +23 CFDPOP(2)
      +25 CFDOWN
      +26 CFDLEN
      +27 CFDPRA(2)
      +31 CFDPRW

+451 HOMUFD:
      +0  HOME UFD NAME
      +20 HOMBRA(2)
      +22 HOMDEV
      +23 HOMPOP(2)
      +25 HOMOWN
      +26 HOMLEN
      +27 HOMPRA(2)
      +31 HOMPRW

+503 LOGNAM(3)
```

4.15. VQUTM

```
1    1 MIN.        1 MINUTE (UPDATE, LOGEV2, ETC.)
2    16.5 MSEC     PUNCH DIM
3
4    112 MSEC      ASR DIM
5    100 MSEC      TENTH SECOND (STIMER QUEUES)
6    --            UNUSED
7    --            UNUSED
8    --            UNUSED
9    1/2           HALF SECOND DISK TIMEOUT
10   1/2 SEC       SMLCEX ALARM
11   10 SEC        NETWORK GROSS TIMER
12   1 SEC         IPC PROTOCOL TIMER
13   1/2 SEC       REMOTE USER POLL
14   4 SEC         FOUR SECONDS
```

5. File System

The following describes the internal formats of all disk records for both the old and new file system partitions. Where possible, field names are the same as those used by the internal file system routines.

5.1. Diskrat Formats

Beginning Record Address(bra) = 2

5.1.1. 21

Rev 21:

0	Len (= 31)
1	Rec_size
2	Disk_size
3	
4	Heads
5	Spec_bits
6	Cylinders
7	Disk_vers
8	Npertk
9	block_alloc
10	disk_model
11	dts
12	
13	first_free
14	
15	DBS_address
16	
17	Reserved
	.
	.
30	.
31	RAT
	.
n	.

Field	Description
Len	Diskrat header length
Rec_size	Physical record size (448 or 1040)
Disk_size	Number of records in partition
Heads	Number of heads in partition
Spec_bits	See section 5.1.3.
Cylinders	Number of cylinders
Disk_vers	Disk version
Npertk	Number of sectors/track
block_alloc	Block allocation method 1 2 16 D | Interleave D Allocation direction: 0 forward 1 reverse Interleave Interleave direction: 3 forward 1 reverse
Disk_model	Disk model type
Dts	Date/time shut down (for mirroring)
First_free	First record after RMA
DBS_address	Pointer to DBS (Dynamic Bad Spot): 1 8 9 16 cylinder head | sector
RAT	Record Availability Table (Disk_size/16) (one bit/record)

5.1.2. Rev 19 and 20

Rev 20:

0	Len (= 11)
1	Rec_size
2	Disk_size
3	
4	Heads
5	Spec_bits
6	Cylinders
7	Disk_vers
8	Npertk
9	Reserved
10	
11	RAT
	.
n	.

Rev 19:

0	Len (= 8)
1	Rec_size
2	Disk_size
3	
4	Heads
5	Spec_bits
6	Cylinders
7	Disk_vers
8	RAT
	.
n	.

Field	Description
Len	Diskrat header length
Rec_size	Physical record size (448 or 1040)
Disk_size	Number of records in partition
Heads	Number of heads in partition
Spec_bits	See section 5.1.3.
Cylinders	Number of cylinders
Disk_vers	Disk version
Npertk	Number of sectors/track
RAT	Record Availability Table (Disk_size/16) (one bit/record)

5.1.3. RAT specifier bits

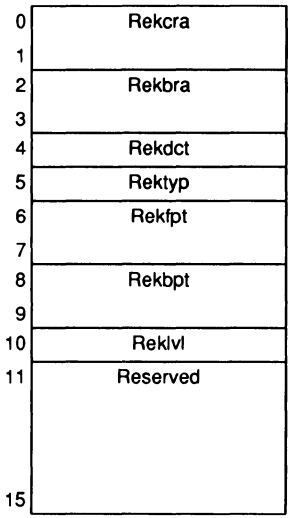
1	14	15	16
-	C	D	

Field	Description	Octal	Hex
C	Crash; disk not shut down previous time	000002	0002
D	DOS modified or permanently broken	000001	0001

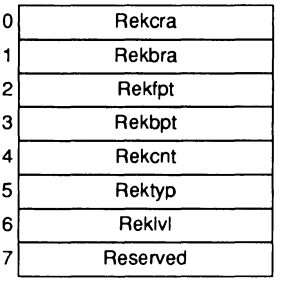
5.2. Record Header Formats

NOTE: record header formats are the same for all partitions. The format of a record header is a function of the physical record size.

1040-Word Records:



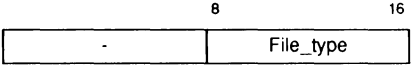
448-Word Records:



Field	Description
Rekcra	Record address of this record
Rekbra	Beginning Record Address (BRA of directory if first record)
Rekdct	Number data words in this record
Rektyp	Type of this file
Rekfpt	RA next sequential record (0 if last)
Rekbpt	RA of previous record (0 if first)
Reklvl	Index level for dam files
Rekcnt	Number data words in this record

5.2.1. Rektyp

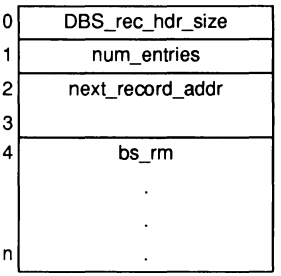
Rektyp is valid only in the first record of a file.



Field	Description	Octal	Hex
File type	File type: 0 - SAM file 1 - DAM file 2 - SAM segment directory 3 - DAM segment directory 4 - Directory 5 - ACL directory 6 - ACAT 7 - CAM file	000377	00FF

If the file is the record zero bootstrap (BOOT) or the disk record availability table (DSKRAT or volume name) and the disk has a 1040 record size (Storage Module), bit 1 (:100000) of FILTYP will be set.

5.2.2. DBS Record Headers



Field	Description
DBS_rec_hdr_size	Size of DBS Record_header
DBS_rec_hdr_size	Size of DBS Record Header
num_entries	Number of DBS entries in record
next_record_addr	Pointer to next record 1 8 9 16 cylinder head | sector
bs_rm	Array of badspot/remap matched pairs 1 2 8 9 16 BScylinder BShead | BSsector A | RMcylinder RMhead | RMsector BScylinder, BShead, BSsector Address of badspot A Already mapped by controller RMcylinder, RMhead, RMsector Address of remap

5.3. UFD Header and Entry Formats

5.3.1. UFD header formats

Rev 20, 21:

0	ECW
1	Owner_password
3	
4	Non_owner_password
6	
7	Reserved
8	Max_quota
9	
10	Dir_used
11	
12	Tree_used
13	
14	Rec_time_prod
15	
16	Prod_dtm
17	
18	Free_pos
19	Hash_version
20	Hash_tbl_size
21	Hash_table
	.
	.
n	.

Rev 19:

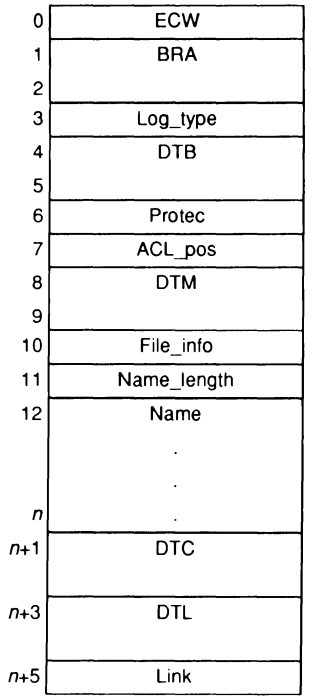
0	ECW
1	Owner_password
3	
4	Non_owner_password
6	
7	Reserved
8	Max_quota
9	
10	Dir_used
11	
12	Tree_used
13	
14	Rec_time_prod
15	
16	Prod_dtm
17	
18	Reserved
22	

Field	Description
ECW	Entry control word. See 5.3.3.
Owner_password	Owner password (6 chars)
Non_owner_password	Non-owner password (6 chars)
Max_quota	Maximum quota
Dir_used	Quota used in this directory
Tree_used	Quota used in entire tree including subdirectories
Rec_time_prod	Record-time product
Prod_dtm	DTM-record product (FS date format)
Free_pos	Free pointer
Hash_version	Version of hash function
Hash_tbl_size	Number of entries in hash table
Hash_table	The hash table

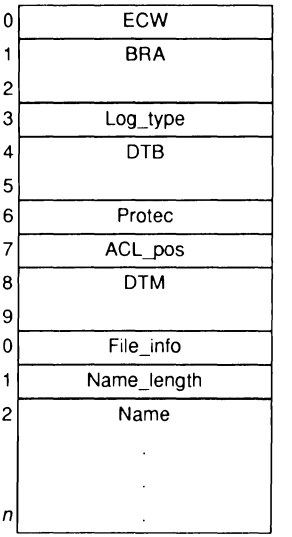
5.3.2. UFD Entry Formats

5.3.2.1. File entries

Rev 20:



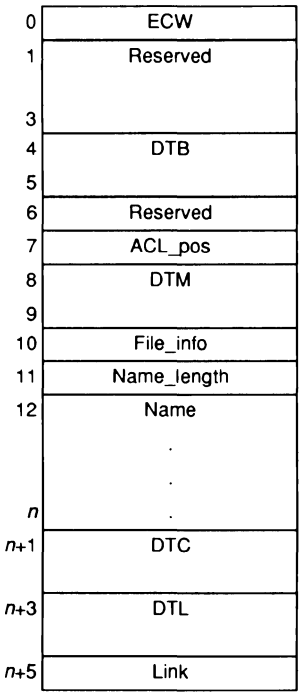
19.0:



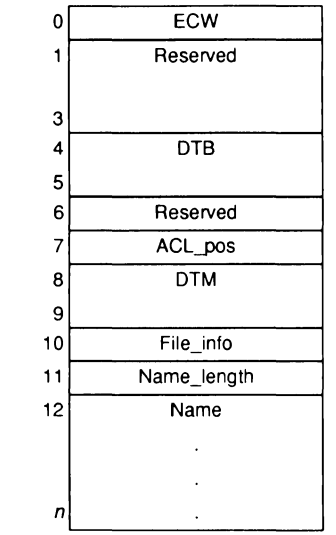
Field	Description
ECW	Entry Control Word
BRA	Beginning Record Address
Log_type	Logical type
DTB	Date/Time Backed-up
Protec	Protection keys
ACL_pos	ACL position
DTM	Date/Time Modified
File_info	See sect 5.3.2.4
Name_length	Length of name
Name	Name of file entry (32 characters)
DTC	Date/Time Created
DTL	Date/Time Last accessed
Link	Link to next entry on chain

5.3.2.2. ACAT entries

Rev 20:



Rev 19:



Field	Description
ECW	Entry Control Word
DTB	Date/Time Backed-up
ACL_pos	ACL position
DTM	Date/Time Modified
File_info	See sect 5.3.2.4
Name_length	Length of name
Name	Name of file entry (32 characters max)
DTC	Date/Time Created
DTL	Date/Time Last accessed
Link	Link to next entry on chain

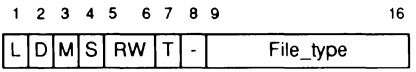
5.3.2.3. DBS entries

Rev 21:

0	file_hd_size
1	version
2	DBS_entry_size
3	number_recs
4	num_badspots
5	num_remaps
6	control_bits
7	reserved

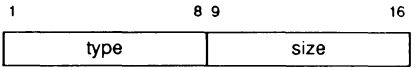
Field	Description
File_hd_size	Size of DBS file header
Version	Version number
DBS_entry_size	Size of a DBS entry
number_recs	Number of records in file
num_badspots	Number of badspots in file
num_remaps	Number of remap records
control_bits	Various flags: 1 14 15 16 - P C P Primos modified this last C The controller modified this last

5.3.2.4. File information bits



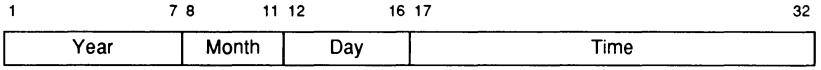
Field	Description	Octal	Hex
L	Long RAT header	100000	8000
D	Dumped; file has been backed up	040000	4000
M	File has been modified under DOS	020000	2000
S	Special file	010000	1000
RW	Read/Write lock: 00 - system default 01 - n readers | 1 writer 10 - n readers & 1 writer 11 - n readers & n writers	006000	0C00
T	Truncated by FIX_DISK	001000	0200
File type	File type: 0 - SAM file 1 - DAM file 2 - SAM segment directory 3 - DAM segment directory 4 - Directory 5 - ACL directory 6 - ACAT 7 - CAM file	000377	00FF

5.3.3. Entry Control Word (ECW)



Field	Description	Octal	Hex
type	Type of entry: 0 - old dir header 1 - directory header 2 - vacant entry 3 - file entry 4 - access category (named ACL) 5 - ACL 6 - directory index block	177400	FF00
size	Size of the entry	000377	00FF

5.4. File system date format



Field	Description	Octal	Hex
Year	Year + 1900 (100-127 = 2000-2027)	177000	FE00
Month	Month (1 = Jan)	000740	01E0
Day	Date	000037	001F
Time	Quad-seconds since midnight		

6. Subroutine Libraries

For additional information, see the *Subroutines Reference Guide* [58] , [59] , [60] , [61] or the *Primenet Guide* [45]. Volume and page numbers follow the documented routines (P indicates *Primenet Guide* reference). Any routine not marked is **not** released. Use of unreleased routines must be cleared with the owning group.

6.1. System routines - Supervisor Calls

Primos ring 0 gates and ring 3 entries.

ABSW () returns(bin) III-2-3
```
if substr(ab$sw$() = 1) then abbrevs enabled;
```
Returns the cold start setting of the global abbrev enable switch.

AC$CAT (char(128)var, char(32)var, bin) II-2-3
```
call ac$cat(object_pathname, acat_name, code);
```
Add a file to an access category.

AC$CAT0 (char(32)var, char(32)var, bin)
```
call ac$cat0 (object_name, category_name, code);
```
Place an object into an access category.

AC$CHG (char(128)var, ptr, bin) II-2-5
```
call ac$chg(object_pathname, addr(acl_struct), code);
```
Modify an existing ACL.

AC$DEV (bit(1), bin)
```
call ac$dev(on_or_off, code);
```
Enable/Disable device ACLs.

AC$DFT (char(128)var, bin) II-2-7
```
call ac$dft(object_pathname, code);
```
Set default protection.

AC$DFT0 (char(32)var, bin)
```
call ac$dft0(object_name, code);
```
Protect an object with default access rights.

AC$LIK (char(128)var, char(128)var, bin) II-2-9
```
call ac$lik(target_object, reference_object, code);
```
Protect one file like another.

AC$LST (char(128)var, ptr, bin, char(128), bin, bin) II-2-11
```
call ac$lst(object_pathname, addr(acl_struc), max_acl_entries,
     acl_name, acl_type, code);
```
Read an ACL.

AC$LST0 (char(32)var, ptr, bin, char(128)var, bin, bin);
```
call ac$lst0(object_name, logical_acl_ptr, max_entry_count,
             acl_name, acl_type, code);
```
Return the contents of an ACL in logical format.

AC$RVT (bin) II-2-13
```
call ac$rvt(code);
```
Convert current ACL dir to password dir.

AC$SET (bin, char(128)var, ptr, bin) II-2-15
```
call ac$set(key, object_pathname, addr(acl_struc), code);
```
Create or replace an ACL.

AC$SET0 (bin, char(32)var, ptr, bin)
```
call ac$set0(key, object_name, acl_ptr, code);
```
Create an ACL for an object, given the object's entryname.

ACCOM$ (bit(16) aligned, fixed bin, char(*) var)
```
call accom$ (switch, unit, action);
```
Saves or restores cominput switch and file unit. *action* is "save" or "restore".

ADBSP$ (bin, bin(31), bin)
```
call adbsp$(work_pdev, physical_badspot, error_code);
```
Add a badspot to the Rev 21 (or greater) disk specified.

ADISK$ (ptr, bin, bin)
```
call adisk$(struc_ptr, list_max, code)
```
Return a list of the locally ASSIGNED disks.

AD_CMD (char(256)var, bin)
```
call ad_cmd(cam_args, com_status);
```
ADDISK command.

ALC$RA (fixed bin(31), ptr options(short)) III-4-16
```
call alc$ra (words_to_allocate, rtn_function_ptr);
```
Allocate space in process class storage for return function data.

ALLOC (fixed bin(31))
```
call alloc(size);
```
Allocates *size* bytes on the callers stack.

ALOC$S entry (fixed bin (15), ptr, bit (1) aligned) options (shortcall (4)); III-4-3
```
call aloc$s (size_to_allocate, pointer_to_space, contiguous);
```
Allocates *size_to_allocate* half-words on stack.

ALS$RA (char(*), fixed bin(31), fixed bin(31)); III-4-21
```
call als$ra(function_result_str, char_size_of_str,
            rtn_function_addr);
```
Allocate space and set return data for return function.

AMLC$ (char(32) nonvarying, (3) fixed bin, bit(16) aligned, fixed)
```
call amlc$ (protocol, line_config_lword, arg_flag, status);
```
Set the line configuration for an amlc line.

AMT$DTR3 (bin) returns(bin(31))
```
DTAR3_storage_used = amt$dtr3(code);
```
Find amount of DTAR 3 storage used by the caller.

APPEND (char(*)var, bin, char(*)var[, bin[, bin]]) returns(bit(1))
```
full = append(string, max_size_of_string, new,
   new_start, new_length);
```
Appends *new* to end of *string*.

APROTO (bin, char(6),code)
```
call aproto(line, protocol, code)
```
Select protocol for an async line. (OBSOLETE. Removed at 22.0)

APSFX$ (char(128)var, char(128)var, char(32)var, bin) II-4-4
```
call apsfx$(in_pathname, out_pathname, suffix, code);
```
Append a suffix to a pathname. code = -1 -> suffix already present.

AR$ALC entry (ptr, fixed bin (31)) returns (ptr)
```
storage_ptr = ar$alc (area_ptr, size_to_allocate);
```
Allocates storage in area previously defined by AR$IN.

AR$FRE (ptr, ptr)
```
call ar$fre (area_ptr, storage_ptr);
```
Frees storage from a previously defined area.

AR$IN (ptr, fixed bin (31))
```
call ar$in (area_ptr, area_size);
```
Initializes area for use by the area manager package.

AR$SIZ (ptr, ptr) returns(bin(31))
```
area_size =  ar$siz(area_ptr, block_ptr);
```
Return size of allocated area.

ARW$ALC (ptr, bin(31), bin) returns(ptr)
```
alloc_ptr = arw$alc(area_ptr, size, code);
```
Allocate in area.

ARW$FRE (ptr, ptr, bin);
```
call arw$fre(area_ptr, block_ptr, code);
```
Free block in area.

ARW$IN (ptr, bin(31), bin);
```
call arw$in(area_ptr, area_size, code);
```
Initialize area header.

ARW$SIZ (ptr, ptr, bin) returns(bin(31))
```
block_size = arw$siz(area_ptr, block_ptr, code);
```
Return size of allocated area.

AS$GET (bin, bin, ptr, bin, bin)
```
call as$get(line_number, version, par_list_ptr, par_list_len,
            error_code)
```
Returns async line information.

AS$LIN (bin, bin)
```
call as$lin(line_number, error_code);
```
Returns the current user's line number.

ASNDE$ (bin, char(80), bin, bin)
```
call asnde$(key, line, state, code);
```
Assign disk and other peripheral devices except magtape.

ASNLN$ (bin, bin, char(*), bin, bin, bin) IV-8-21
```
call asnln$(key, amlc_line, protocol, amlc_config, lword, code);
```
Assign AMLC line. Key = 0 - unassign; 1 - assign; 2 - unassign all.

ASNMT$ (bit(16), char(256) var, bin(15))
```
call asnmt$ (no_msgs, user_assign_cmd_line, return_status);
```
Assign magnetic tape drive.

ASSUR$ (bin) returns(bit(1)) III-2-17
```
enough_time = assur$(desired_mseconds);
```
Allow a user process to assure it has a certain amount of cpu time left.

AT$ (bin, char(128)var, bin) II-3-3
```
call at$(key, path_name, code);
```
Attach by pathname. Key = K$SETH, K$SETC.

AT$ABS (bin, char(32)var, char(39)var, bin) II-3-6
```
call at$abs(key, partition_name, dir_name, code);
```
Attach to top-level dir on given partition. Dir_name includes password. Key = K$SETH, K$SETC.

AT$ANY (bin, char(39)var, bin) II-3-8
```
call at$any(key, dir_name, code);
```
Attach to top-level dir. See AT$ABS for notes.

AT$ANY0 (bin, char(39)var, bin)
```
call at$any0(key, dir_name, code);
```
Do an old-style attach scan.

AT$HOM (bin) II-3-10
```
call at$hom(code);
```
Return to home dir.

AT$INV (bin, bin)
```
call at$inv(key, code);
```
Invalidates specified attach point(s). Key = K$KURA, K$HOMA, K$INIA, K$ALL.

AT$LDEV (bin, bin, char(39)var, char(32)var, bin) II-3-11
```
call at$ldev(key, ldev, dir_name, partition, code);
```
Attach to top-level directory given the ldev of the partition.

AT$OR (bin, bin) II-3-13
```
call at$or(key, code);
```
Return to origin dir. Key = K$SETH, K$SETC.

AT$REL (bin, char(39)var, bin) II-3-15
```
call at$rel(key, dir_name, code);
```
Attach relative to current dir. Key = K$SETH, K$SETC.

AT$TMP returns(bit(1))
```
swap_completed = at$tmp();
```
Save or restore the current attach point.

ATCH$$ (char(32), bin, bin, char(6), bin, bin) (svc = 1500) II-A-2
```
call atch$$(ufd_name, name_len, ldisk_num, password, key);
```
Attach to UFD. (Obsolete; use AT$, AT$ABS, AT$ANY, AT$HOM, AT$OR, AT$REL)

ATLIST (fixed bin, (12) char(32) var, fixed bin, char(32), fixed bin, char(6), fixed bin, fixed bin)
```
call atlist (key, disk_list, disk_count, dir_name,
             dir_name_len, pass~word, found_index, code);
```
Search a list of dirs on a given system (NPX only).

ATSHR$ (bin(31), bin, ptr, ptr, bin)
```
call atshr$(unique_seg_id, req_accesses, true_seg_ptr,
            dtar2_seg_ptr, code);
```
Attach to a segment allocated by gtshr$.

AU$CUR (bin, char(256), bin)
```
call au$cur(user, dest, code);
```
Access current log entry for a given user.

AU$DRN (bin, bin)
```
call au$drn(context, code)
```
Shut down an AUSLOG phantom.

AU$GET (ptr, bin, bin)
```
call au$get(dest, npage, code);
```
Return copy of current log buffers for LOGANAL utility of AUSLOG.

AU$START (bin, bin)
```
call au$start(return_phantom, code);
```
Start up the AUSLOG utility phantom.

AU$STAT (bin, char(128)var)returns(bit(16))
```
status = au$stat(user, file);
```
Show current status of AUSLOG phantom.

AU$STRT0 (bin, bin)
```
call au$strt0(rtn_ph, code);
```
Start up the OS_LOG utility phantom.

AU$TSK (bin, bin, char(160)var, bin, char(32), bin)
```
call au$tsk(type, task_type, command, status, caller_id,
      cpl_taskno);
```
Assembles AUSLOG login/logout message types before logging.

AU$WRT (bin, bin, bin)
```
call au$wrt(log_file, prwf_rtn_code, status);
```
Write to AUSLOG log file & wait for a data buffer.

BATCH$ (char(*), fixed bin, fixed bin, char(*), fixed bin, fixed bin, fixed bin)
```
call batch$ (filename, name_length, unit, user_name,
            user_name_length, user_num, status)
```
Spawns a phantom under any user id. Priviledged. (OBSOLETE; use SPAWN$)

BCKUPB$ (ptr)
```
call bckupb$(target_sb);
```
Back Up Return PB For Ring 0 Restart.

BD$ATT (char(*6), bin, bin, bin)
```
call bd$att(name, length, dev, code);
```
Block device 'ATTACH' subroutine. (DPTX)

BD$DET (bin, bin)
```
call bd$det(device, code);
```
Block device detach subroutine. (DPTX)

BD$INF (bin, bin, char(*), bin, (10)bin, bin)
```
call bd$inf(device, key, buffer, buf_len, stat_protocol, code);
```
Block device information & status subroutine. Key = k$infn, k$infd, k$infs. (DPTX)

BD$INP (bin, bin, char(*), bin, (10)bin, bin, bin)
```
call bd$inp(device, key, buffer, buf_len, status_protocol, code,
          wait_period);
```
Block device input subroutine. Key = k$wait, k$nowa, k$watt. (DPTX)

BD$LST (bin, char(6), bin, (*)bin, bin, bin)
```
call bd$lst(key, name, name_len, data_buffer, data_len, code);
```
Block device interface description routine. Key = k$infd, k$infn, k$ltat, k$lpat, k$ptat, k$patd, k$bsys. (DPTX)

BD$OUT (bin, bin, char(*), bin, (10)bin, bin)
```
call bd$out(device, key, buffer, len, status_protocol, code);
```
Block device output subroutine. Key = k$xmtf, k$xmtd, kmrk, krawd. (DPTX)

BD$SET (bin, bin, bin)
```
call bd$set(device, key, code);
```
Block device attribute-setting subroutine. Key = k$inwt, k$iwof, k$tabt, k$rabt, k$spdi, k$spdo, k$rsmi, k$rsmo, k$pa2p, k$pa2q. (DPTX)

BIN$SR (char(*) var, fixed bin, ptr, ptr, ptr, fixed bin)
```
call bin$sr(entry, entry_size, start_ptr, end_rel, spot_ptr,
            code);
```
Binary search on ordered table (one segment restriction).

BM$GET (bin, char(*), bin, bin, bin);
```
call bm$get(key, buffer, buffer_length, chars_returned,
      return_code);
```
Gets data from the user's TFLIO input buffer to the user buffer.

BM$MOD (bin, char(1), bin)
```
call bm$mod(key, eot_char, return_code);
```
Switch user's terminal line bewteen character and block mode.

BM$QRY (bin)
```
call bm$qry(mode);
```
Queries whether a user's terminal is in block mode.

BM$RDY returns (bin)
```
buf_sem_count = bm$rdy();
```
Returns the count field of a user's terminal buffer semaphore.

BM$SCAN (char(*), bin, char(1), bin)
```
call bm$scan(buffer, length_of_scan_area, scan_char, offset);
```
Search for a single character in a string from a offset.

BREAK$ (bin) (svc = 0507) III-3-50
```
call break$(value);
```
Inhibits or enables quits. Value = 1 to inhibit breaks.

BSCMAN (fixed bin(15), fixed bin(15), (256) fixed bin(15))
```
call bscman (error_line, option, protocol_table);
```
Iniate the bisynchronous communications. Handler for the IBM 3270 protocols (DPTX).

C1IN (char) (svc = 0601) III-3-5
```
call c1in(character);
```
Get one char (right justified) from terminal or command file.

C1IN$ (char(2), bit(1), bit(1)) III-3-7
```
call c1in$(retchar, echo, termonly);
```
Single character command input.

C1NE$ (char(2)) III-3-9

```
call cl1ne$(rtn_char)
```
Input single character with no echo.

CALAC$ (char(128)var, ptr, char(80), char(80)var, bin) II-2-17 returns(bit(1))
```
have_access = calac$(pathname, addr(id_struc), access_required,
              access_gotten, code);
```
Calculate access available.

CALAC$0 (char(32)var, ptr, char(47)var, char(47)var, bin) returns (bit (1))
```
have_access = calac$0(name, id_ptr, access_needed,
 access_gotten, code);
```
Calculate accesses available on a named object.

CALFC_ (ret_pb, not_in_range) options (shortcall (4))
```
call calfc_(PB_in_question, not_in_range);
```
Does magnitude check on a return PB to see if it is within the ring 3 pointer fault table.

CAT$DL (char(128)var, bin) II-2-19
```
call cat$dl(acat_pathname, code);
```
Delete an access category.

CAT$DL0 (char(32)var, bin)
```
call cat$dl0(category_name, code);
```
Delete an access category.

CE$BRD returns(bin) II-6-3
```
maximum_command_env_breadth = ce$brd();
```
Return maximum command level breadth for this user.

CE$DPT returns(bin) II-6-4
```
maximum_command_env_depth = ce$dpt();
```
Return maximum command level depth for this user.

CF$EXT (bin, bin(31), bin(31), bin)
```
call cf$ext(unit, req_peof, act_peof, code);
```
Moves physical end of file for a contiguous file.

CF$REM (bin, bin, bin, bin)
```
call cf$rem(unit, buffer, length, code);
```
Returns a copy of the on-disk extent map.

CF$SME (bin, bin, bin)
```
call cf$sme(unit, min_ext_len, code);
```
Sets minimum extent length for a contiguous file.

CFI returns(bit(16))
```
char_avail = cfi();
```
Program to check if there is a character in the terminal buffer.

CH$FX1 (char (*) var, fixed bin(15) [, fixed bin(15)]) III-6-3
```
call ch$fx1 (string_to_convert, result [, nonstandard_code]);
```
Convert character varying string to fixed bin(15).

CH$FX2 (char(*) var, fixed bin(31) [, fixed bin(15)]) III-6-5
```
call ch$fx2 (string_to_convert, result [, nonstandard_code]);
```
Convert character varying string to fixed bin(31).

CH$HX2 (char (*) var, fixed bin (31) [, fixed bin (15)]) III-6-7
```
call ch$hx2 (string_to_convert, result [, nonstandard_code]);
```
Convert character varying string to fixed bin(31) as hex.

CH$MOD (fixed bin, fixed bin, fixed bin) II-4-6
```
call ch$mod (key, unit, code);
```
Change the open mode of an open file. Key = K$READ, K$WRIT, K$RDWR.

CH$OC2 (char(*) var, fixed bin(31) [, fixed bin(15)]) III-6-9
```
call ch$oc2 (string_to_convert, result [, nonstandard_code]);
```
Convert character varying string to fixed bin(31) as octal.

CHBK$$ (bin, bin(31), bin, bin)
```
call chbk$$(key, uri, unit, status);
```
Routine to check status of asynchronous writes.

CHG$PW (char(16)var, char(16)var, code) III-2-18
```
call chg$pw(old_password, new_password, code);
```
Change login password.

CHG$SA (char(32) var, fixed bin)
```
call chg$sa (new_administrator_id, code);
```
Changes the user id of the system administrator. Priviledged.

CIRLOG (bin, bin, bin, bin, bin, bin, bin, bin, bin)
```
call cirlog(entry_type, subroutine_name, arg1, arg2, arg3,
            arg4, arg5, arg6, arglen);
```
Debug routine for NPX.

CKDYN$ (char(32)var, bin) III-2-4
```
call ckdyn$(routine_name, code)
```
Check for the existance of a dynamic entrypoint.

CKNDNM (char(32)var, bin, bin)
```
call ckndnm(node_name, vcix, code);
```
Subroutine to check the validity of node name on the name table.

CL$FN0 (char(32)var, bin)
```
call cl$fn0(entryname, code);
```
Close an open file by name.

CL$FNR (char(128)var, 1, 2 bin, 2 (*)bit(16), bin, bin) II-4-7
```
call cl$fnr(pathname, rtn_list, first_file_unit, code);
```
Close a file by name and return a bit varying indicating closed units.

CL$FR0 (char(32)var, 1, 2 bin, 2 (*) bit (16), bin, bin)
```
call cl$fr0(entryname, rtn_list, first_file_unit, code);
```
Close a file by name and return a bit varying indicating closed units.

CL$GET (char(*)var, bin, bin) III-3-10
```
call cl$get(buffer, max_buffer_len, code);
```
Read a line of text from terminal or command file.

CL$GET_EV (bin, entry, bin)
```
call cl$get_ev(key, routine, code)
```
Command loop get entry variable. Key = K$COMMAND_PROCESSOR, K$COMMAND_LINE_READER, K$COMMAND_PROMPT

CL$PAR (bit(16) aligned, char(*) var, char(*) var, bin, 1 ..., bin, bin)
```
call cl$par (keys, source_str, token_str, token_str_size, info,
             next_ch, status);
```
Parse *source_str* according to basic command line rules.

CL$PIX (bit(16), char(*)var, ptr, bin, char(*)var, ptr, bin, bin, bin, ptr) II-6-5
```
call cl$pix(keys, caller_name, addr(picture), pixel_size,
      input_line, addr(com_line_struct), pic_error_loc,
      bad_index, code);
```
Parse command line. See Subroutine Ref Guide for code values.

CL$SET_EV (bin, entry, bin)
```
call cl$set_ev (key, routine, code);
```
Command line set entry variable. Same keys as CL$GET_EV.

CLO$FN (char(128) var, fixed bin) II-4-9
```
call clo$fn(pathname, code);
```
Close an open file by name.

CLO$FU (fixed bin, fixed bin) II-4-10
```
call clo$fu(unit, code);
```
Close an open file by unit.

CLRLV$
```
call clrlv$;
```
clear the existing command level.

CMD_POST (ptr options(short))
```
call cmd_post_invk (epf_smt_ptr);
```
Perform post-program invocation cleanup.

CMD_PRE_ (bin)
```
call cmd_pre_ (code);
```
Perform pre-program invocation initialization.

CMLV$E III-5-5
```
call cmlv$e;
```
Call a new command level with error prompt. (see comlv$)

CMREA$ (char(80), bin, (59)bin, bin, bin)
```
call cmrea$(com_line, com_state, ucmpar, maxlen, code);
```
Old style command line parser.

CNAM$$ (char(*), bin, char(*), bin, bin) (svc = 1515) II-4-11
```
call cnam$$(old_name, old_name_len, new_name, new_name_len,
      code);
```
Change the name of a file.

CNIN$ (char(*), bin, bin) (svc = 0604) III-3-13
```
call cnin$(buffer, char_count, rtn_char_count);
```
Input char_count characters.

CNSIG$ (bin) III-7-19
```
call cnsig$(code);
```
Call more on-units; continue to signal condition.

CO$GET (bin, bin) III-3-52

```
call co$get (comoutput_funit, command_stream_sw);
```
Retrieve the comoutput unit number and value of COUSWI.

COM$AB (char(1024)var, bin, bin) III-2-20
```
call com$ab(command, command_size, code);
```
Interlude to invoke command abbreviation processor.

COMANL (svc = 0600) III-3-15
```
call comanl;
```
Read a line of text. (Obsolete; use CL$GET)

COMI$$ (char(32), bin, bin, bin) (svc = 1516) III-3-53
```
call comi$$(file_name, file_name_len, file_unit, code);
```
Switch between terminal and command file for input.

COMLV$ III-5-6
```
call comlv$;
```
Call a new command level.

COMO$$ (bit(16), char(32), bin, bin, bin) (svc = 1523) III-3-55
```
call como$$(key, file_name, file_name_len, reserved, code);
```
Change terminal output to terminal or file. Key bits: :1 - TTY off; :2 - TTY on; :10 - file off; :20 - file on; :40 - append if file on, close if file off; :100 - truncate if file on.

CP$ (char(*) var, bin, bin, 1, 2 bit(1), 2 bit(1), 2 bit(14), ptr, ptr) II-6-9
```
call cp$(command_line, status, com_status, flags,
       local_variable_ptr, rtn_function_ptr);
```
Invoke the user's currently specified command processor.

CP$ITR(char(1024)var, entry, bin, bit(5), bit(3), bit(1), bin)
```
call cp$itr (com_line, executer, eq_position, default_types,
             exp_wild, vfy_default, status)
```
Command language iteration handler.

CP$TW (char(128)var, ptr, bit(1), entry(char(*)var)var, bin, bin, char(128)var, bit(1), char(32))
```
call cp$tw (wildpath, a_optp, exp_wildcards, executer, status,
    a_level, a_tame_path, tame_wild, a_tree_wildcard);
```
Perform command language Treewalk Iteration.

CP$WLD (char(128), var, ptr, entry, fixed)
```
call cp$wld (wildcard_path, options_ptr, executer, status);
```
Invoke *executer* for every file in *wildcard_path.*

CPS$ (fixed bin(15),fixed bin(15))
```
call cps$(user, code);
```
Invoke cross process signal on-unit set up by another user process.

CPS$CN (fixed bin(15))
```
call cps$cn(key);
```
Enable or disable registered cross process signal condition.

CPS$NA (char(*) var)
```
call cps$cn(name);
```
Retrieve name of the on-unit currently registered for cross process signalling.

CPS$RC (fixed bin(15),fixed bin(15))
```
call cps$(user,code);
```
Check on receipt of a cross process signal by another user.

CPS$RG (char(32) var, (*) bin, bin, bin)
```
call cps$rg(name,acl,len,key);
```
Register process with the cross process signalling mechanism.

CPS$SN ((128) bin, bin, bin)
```
call cps$sn(usl,len,rtrnlen);
```
Gets list of users who have signalled during disabled CPS state.

CPS$ST (fixed bin(15))
```
call cps$st(code);
```
Check CPS status of a process.

CPUID$ (ptr, bin) III-2-5
```
call cpuid$ (cpuid$_struc_ptr, ercode);
```
Return the CPU Id and microcode revision numbers.

CRAWL_ (entry, entry, ptr, ptr, fixed bin, fixed bin)
```
call crawl_ (crawl_fim, crawl_fim_backup, crawl_frame,
             regs_frame, cs_depth, defer_crawl);
```
Crawlout from an inner ring and rejoin signl$ or fim_.

CREA$$ (char(32), bin, char(6), char(6), bin) (svc = 1501) II-A-5
```
call crea$$(file_name, fiel_name_len, owner_pw, non_owner_pw,
      code)
```
Create sub-UFD of same type as containing UFD (ACL or non-ACL).

CREPW$ (char(32), bin, char(6), char(6), bin) II-A-7
```
call crepw$(new_dir_name, dir_name_len, owner_pw, non_owner_pw,
      code)
```
Create a password dir.

CSTAK$ (fixed bin, 1 ..., bit(1) aligned, ptr)
```
call cstak$ (depth, cs_data, eog, pb_value);
```
Manipulate/examine the calling process' concealed stack.

CUCPY$ (bin)
```
call cucpy$(ldev);
```
Perform a catch up copy on a mirrored pair of disks.

CV$DQS (bin(31), bin(31)) III-6-12
```
call cv$dqs(binary_date, quadseconds);
```
Convert binary date to quadseconds.

CV$DTB (char(128)var, bin(31), bin) III-6-13
```
call cv$dtb(ascii_date, binary_date, code);
```
Convert formatted date to binary.

CV$FDA (bin(31), bin, char(21)) III-6-15
```
call cv$fda(binary_date, day_of_week, ascii_date)
```
Convert binary date to ISO format.

CV$FDV (bin(31), bin, char(28)var) III-6-17
```
call cv$fdv(binary_date, day_of_week, ascii_date)
```
Convert binary date to visual format.

CV$QSD (bin(31), 1, 2, 3 bit(7), 3 bit(4), 3 bit(5), 2 bin) III-6-19
```
call cv$qsd(quad_secs, fs_date);
```
Convert quadseconds since January 1, 1901 to date.

DATE$ returns(bin(31)) III-2-8
```
binary_date = date$();
```
Return current date/time in binary format.

DATE_A (char(1024) var, bin, char(32) var, ptr, char(1024) var, bin)
```
call date_af (arguments, code, af_name, vcb_ptr, answer,
              result_max);
```
CPL date function.

DB$MOD (bit(1) aligned, ptr)
```
call db$mod (dbg_in_use, dbg_onunit_ptr);
```
Set/reset debugger-mode switch and static on-unit.

DBG$BR (bin, bit(1))
```
call dbg$br(fault_fr_hdr, do_signal);
```
Notify the ring zero debugger in the event of a breakpoint.

DELAY (bin, bin, bin)
```
call delay (min, max, margin)
```
Define delay times for printing characters after new line.

DELAY_ (char(*) var, fixed, char(*) var)
```
call delay_ (com_args, com_status, com_name);
```
Processes command arguments for DELAY command.

DET$GET (char(128)var, bin, char(32)var, char(1024)var, bin, bin)
```
call det$get(et_path, error_code, error_name, message,
     msg_size, code);
```
Get msg from a Diagnostic Error Table.

DH3270
```
call dh3270;
```
Initiate the data handler for IBM 3270 terminals (DPTX).

DIR$CR (char (128) var, ptr, bin) II-4-15
```
call dir$cr(dir_path, attribute_ptr, code);
```
Create a directory.

DIR$CR0 (char(32)var, ptr, bin);
```
call dir$cr0(dir_name, attribute_ptr, code);
```
Create a directory.

DIR$LS (bin, bin, bit(1), bit(4), ptr, bin, ptr, bin, bin, bin, (4)bin, bin(31), bin(31), bin) II-4-17
```
call dir$ls(dir_unit, dir_type, initialize, rtn_file_types,
     addr(wild_card_array), wca_len, addr(rtn_struc),
     max_entries, entry_size, rtn_num_entries, num_types,
     before_binary_date, after_binary_date, code)
```
Search directory.

DIR$RD (bin, bin, ptr, bin, bin) II-4-23
```
call dir$rd(key, dir_unit, addr(buffer), buf_len, code)
```
Read dir entries. Key = K$INIT, K$READ.

DIR$SE (bin, 1, 2 bit(13), 2 bit(2), 2bit(1), bit(1), ptr, ptr, bin, bin, bin, (*)bin, bin, bin) II-4-27
```
call dir$se(dunit, dtype, rewind, sel_ptr, arg_outptr, outnum,
                 outlen, out_filled_in, totals, max_type, code);
```
Retrieve info about selected entries in a given directory.

DIRSER (bin, bin, bit(1), ptr, bin, ptr, bin, ptr, bin, bin, bin, (4) bin, bin, bin)
```
call dirser(dunit, dtype, rewind, sel_ptr, sel_len_rem,
      wild_ptr_rem, wild_len_rem, arg_outptr, outnum, outlen,
      out_filled_in, totals, max_type, code);
```
Remote interlude to DIR$SE (NPX only).

DKGEO$ (bin, ptr, bin) [returns(bin)] IV-5-18
```
ldev = dkgeo$ (pdev, geo_ptr, code);
```
Register disk geometry with the disk driver.

DL$CMGCE (1, 2 bin, 2 bin, 2 bin, 1, ..., bin)
```
call dl$cmgce(search_list, status, error_code);
```
Gate to get CCPAT entry for a specified controller.

DL$CMGCI (1, 2 bin, 2 bin, 2 bin, 2 bin, bin, bin)
```
call dl$cmgci(search_list, cntl_index, error_code);
```
Get controller index for a specified LAN or ICS controller.

DL$CMGLS (1, 2 bin, 2 bin, 1, ..., bin);
```
call dl$cmgls(search_list, status, error_code);
```
Get pcc_load_parms state for specified controller index.

DL$CMGRE (1, 2 bin, 2 bin, 1, 2 bin, 2 (7)bin, 2 (16)bin, bin)
```
call dl$cmgre(search_list, status, error_code);
```
Get a PCCRSTDT entry for a sepcific ICS controller.

DL$CMUCE (1, ..., bin)
```
call dl$cmuce(update_list, error_code);
```
Update CCPAT entry for a specified controller index.

DL$CMULS (1, ..., bin)
```
call dl$cmuls(update_list, error_code);
```
Update PCC_LOAD_PARMS state for any controller.

DL$CMURE (1, 2 bin, 2 bin, 2 bin, 2 (7)bin, 2 (16)bit(16), bin)
```
call dl$cmure(update_list, error_code);
```
Update a PCCRSTDT entry for a sepcific ICS controller.

DL$ICAIO (1, 2 bin, 2 (*)bin, 2 bin, 2 bin, 2 bin(31), 2 bin, (*)bin)
```
call dl$icaio(alloc_list, error);
```
Allocating SEG0 area and phantom interrupt code.

DL$ICASY (1, 2 bin, 2 (*)bin, 1 (*), 2 bin, 2 bin, 2 bin)
```
call dl$icasy(start_list, error);
```
Start ASYNC support for an ICS controller.

DL$ICDIO (1, 2 bin, 2 (*)bin, (*)bin);
```
call dl$icdio(deallocate_list, error);
```
Deallocating SEG0 area and phantom interrupt code.

DL$ICDLL (1, 2 bin, 2 (*)bin, 2(*)bin, 2 char(128)var, 1(*), 2 bin, 2 bin, 2 bin)
```
call dl$icdll(load_list, error);
```
Down line load a DMT file into specified ICS controllers.

DL$ICISV (1, 2 bin, 2 (*)bin, 1(*), 2 bin, 2 (*)bin, 1 (*), 2 bin, 2 bin, 2 bin)
```
call dl$icisv(verify_list, status, error);
```
Issue self verify to a specified ICS controllers.

DL$ICNCR (1, 2 bin, 2 (*)bin, (*)bin)
```
call dl$icncr(reset_list, error_code);
```
Perform normal reset for ICS controllers.

DL$ICSCR (1, 2 bin, 2 (*)bin, (*)bin)
```
call dl$icscr(reset_list, error_code);
```
Perform special reset for ICS controllers.

DL$ICSDC (1, 2 bin, 2 (*)bin, 1 (*), 2 bin, 2 bin, 2 bin)
```
call dl$icsdc(shut_list, error);
```
Shut down a specified ICS controller.

DL$ICSRT (1, 2 bin, 2 (*)bin, 1 (*), 2 bin, 2 bin, 2 bin)
```
call dl$icsrt(start_list, error);
```
Starting IPQNM for given ICS controllers.

DL$LHDLL (1, 2 bin, 2 (*)bin, char(128)var)
```
call dl$lhdll(load_list, error);
```
Initiate downline load of an LHC controller.

DL$LHISV (1, 2 bin, 2 (*)bin, 1 (*), 2 bit(16), 2 bit(16), 1 (*), 2 bin, 2 bin)
```
call dl$lhisv(verify_list, status, error);
```
Initiate self verify of an LHC controller.

DL$LHLSP (1, 2 bin, 2 (*)bin, 2 ptr, 2 bin, 1 (*), 2 bin, 2 bin)
```
call dl$lhlsp(start_list, error);
```
Perform load start packet function for the LAN controller.

DL$LHNCR (1, 2 bin, 2 (*)bin, (*)bin)
```
call dl$lhncr(reset_list, error_code);
```
Normal reset of a LAN controller

DL$LHULD (1, 2 bin, 2 bin, 2 char(128)var, 1, 2 bin, 2 bin)
```
call dl$lhuld(dump_list, error);
```
Initiate upline dump from a LAN controller.

DMP$LS (bin, bin, bin, bin)
```
call dmp$ls(index, low_seg, high_seg, err_code);
```
Display entries from the DMP_SEG array for partial tape dump.

DMP$LU (bin, char(32), bin)
```
call dmp$lu(index, user_name, err_code);
```
Display entries from the DMP_USR array for partial tape dump.

DMP$RS (bin)
```
call dmp$rs(err_code);
```
Reset the DMP_SEG and DMP_USR arrays to the default values.

DMP$SG (bin, bin, bin)
```
call dmp$sg(low_seg, high_seg, err_code);
```
Add new entries to DMP_SEG array for partial tape dump.

DMP$US (char(32), bin)
```
call dmp$us(user_name, err_code);
```
Routine to add new entries to DMP_USR array for partial tape dump.

DOSSUB (char(80), bin)

```
call dossub(command_line, code);
```
Old internal command processor.

DPT$QM (1, 2 (8)bin, 2 (8)bin, 2 (8)bin, 2 (8)bin, 2 (32)bin, 2 (32)bin, 2 (32)bin, 2 (32)bin, 2 bin, 2 bin, 2 bin, bin)
```
call dpt$qm(queue_length, code);
```
Queue monitor subroutine for DPTX queues.

DPT$ST (bin, bin, (*, 19)bin, bin)
```
call dpt$st(key, line, info, code);
```
Retrieve ring 0 information for DPTX monitor.

DPTINI (bin, bin)
```
call dptini (file_unit, code);
```
Initialize all of the DPTX databases.

DPTOFF (bin)
```
call dptoff (code);
```
Deallocates all of the DPTX databases and shuts down the DPTX phantoms.

DS$ACC (char(32)var, ptr, bin)
```
call ds$acc(node_name, sptr, code);
```
Return Primenet nodal access configuration.

DS$ASY (bin, bin, ptr, bin)
```
call ds$asy(key, line_no, sptr, code);
```
Retrieve asynchronous line information.

DS$AVL (ptr, bin, bin)
```
call ds$avl(list_p, ldev, code);
```
Return disk size and date of last backup.

DS$CFG (ptr, bin(31), bin)
```
call ds$cfg(loc_ptr, storage_size, code);
```
Return config directive values.

DS$COM (ptr, bin)
```
call ds$com(bufptr, code);
```
List communications controller status.

DS$ENV (bin, ptr, bin)
```
call ds$env(user_no, lptr, code);
```
Return general information about a user's process environment.

DS$HST (bin, char(32)var, pointer, bin)
```
call ds$hst(version_num, lan_name, user_bufr_p, error_code);
```
Retrieve configured HOST information from the NSS database.

DS$LAN (pointer, bin);
```
call ds$lan(user_bufr_p, error_code);
```
Retrieve configured LAN information from the NSS database.

DS$LTS (bin, char(32)var, pointer, bin)
```
call ds$lts(version_num, lan_name, user_bufr_p, error_code);
```
Retrieve LTS information from the NSS database.

DS$LTU (bin, char(32)var, char(16)var, bin, bin)

```
call ds$ltu(user_num, LAN_name, LTS_name, LTS_line, error_code);
```
Retrieve LTS User Information from Primos User Number.

DS$PNC (bin, ptr, bin)
```
call ds$pnc(pnc_id, sptr, code);
```
Return the IDs of all configured nodes on a specified ring.

DS$POR (ptr, bin(31), bin)
```
call ds$por(sptr, size, code);
```
Return the system port assignments.

DS$RECHK (1, ..., bin)
```
call ds$rechk(resus_switch_data, return_code);
```
Interrogate the REmote System USer switch.

DS$REENA (1, ..., bin)
```
call ds$reena(r3, return_code);
```
Enable REmote System USer switch.

DS$RERD (char(1), bin)
```
call ds$rerd(in_char, return_code);
```
Read character from User-1 output queue.

DS$RERST (bin)
```
call ds$rerst(return_code);
```
Reset REmote System USer switch

DS$RESSW (char(32), bin, char(32), bin, bin)
```
call ds$ressw(user_id, user_no, user_node, synchroniser_id,
       return_code);
```
Set REmote System USer switch.

DS$REWR (char(1), bin)
```
call ds$rewr(out_char, return_code);
```
Put character on REmote System USer input queue.

DS$RNG (bin, bin, ptr, bin)
```
call ds$rng(pnc_id, ring_id, sptr, code);
```
Return the status of a specified ring node

DS$SYN (ptr, bin)
```
call ds$syn(sptr, code);
```
Return synchronous line configuration.

DS$UNI (bin, bin, bin, char(128)var, ptr, bin)
```
call ds$uni(key, user_no, unit_no, full_path, struc_ptr, code);
```
Returns data on specified user's open file unit, attach point or open file.

DUPLX$ (bin) returns(bin) (svc = 0705) III-3-57
```
prev_config = duplx$(new_term_config)
```
Set terminal configuration word (bits 1-8 only).

DY$SGS returns(bin) III-4-25
```
maximum_private_dynamic_segs = dy$sgs();
```
Returns maximum number of dynamic segments for this user.

EBK$$ (bin, bin(31), bin(31), bin)

```
call ebk$$(unit, leof, peof, code);
```
Returns physical and logical eof for file open in block mode.

EF$RELOCATE (ptr, ptr) returns(ptr)
```
real_address = ef$relocate(smt_ptr, erp);
```
Relocate an ERP for an EPF.

EM3270 (bin, (1920)bin)
```
call em3270 (line_number, virtual_buffer_temporary);
```
Initiate the emulator handler for the IBM 3270 terminals (DPTX).

ENCRYP, ENCRYPT$ (char(16) var) returns (char(16)) III-6-24
```
encrypted_password = encrypt$ (unencrypted_password);
```
Encrypts login passwords (one-way).

ENT$RD (bin, char(32)var, ptr, bin, bin) II-4-35
```
call ent$rd(dir_unit, entry_name, addr(rtn_struc), rs_len, code);
```
Read a dir entry by a given name.

ENTDIR$ (char(128)var, char(128)var, char(32)var, bin) returns(bit(1))
```
attached = entdir$(xpathname, pathname, entry_name, code);
```
Attach to parent of object in pathname, return entry name of object.

EPFAL, EPFALLC(ptr, fixed bin) II-5-3
```
call epf$allc(smtp, status);
```
Allocate EPF linkage.

EPF$CHAIN (ptr, ptr, bin, bin, struc, struc)
```
call epf$chain(smt_ptr, data_ptr, data_len, code, com_state,
               com_proc_flags);
```
Support routine to allow B86 EPFS to do chaining.

EPF$CINF (char(128)var, (8)char(32) var, ptr, bin, ptr, bin)
```
call epf$cinf(epf_treename, alias_list, smt_ptr,
      epf_database, error_aliases_ptr, status);
```
Copy EPF information to the registered EPFs database.

EPFCP, EPFCPF (ptr, struc, bin, bin) II-5-5
```
call epf$cpf (smtp, com_state_structure, status);
```
Get command processor flags from an EPF.

EPF$DBG (pointer, bin, pointer, bin)
```
call epf$dbg(smtp, requested_version, epf_dbg_info_ptr,
      status);
```
Obtain EPF information from the PRIMOS command environment for DBG.

EPFDEL, EPFDL (ptr, bin) II-5-7
```
call epf$del(smtp, status);
```
Terminate EPF, de-allocating storage.

EPF$GETI (char(32)var, bin, ptr, char(32)var, bin)
```
call epf$geti(epf_name, epf_database, smt_ptr, bad_lib,
      code);
```
Get information (in an SMT) about a registered EPF.

EPF$GTLI (char(32)var, bin, bin, ptr, ptr, char(32)var, bin)

```
call epf$gtli(epf_name, epf_database, num_smts,
      current_SMTs_ptr, smt_ptr, bad_lib, code);
```
Get limited information about a registered EPF.

EPF$INFO (ptr, struc, bin)
```
call epf$info(smtp, epf_info, status);
```
Return info about a desired epf file.

EPF$INIT (bin, ptr, bin) II-5-9
```
call epf$init (key, smtp, status) options(nocopy);
```
Initialize EPF static data.

EPF$INVK (ptr, bin[, char(*) var, bin, 1, ..., ptr]) II-5-11
```
call epf$invk(smtp, status, com_args, com_status, com_state,
              flags, rtn_function_ptr);
```
Start execution of an EPF.

EPF$LENT (char(32)var, char(32)var, ptr, bin)
```
call epf$lent(entryname, libname, liberp, code);
```
Search registered EPF libraries in order for specific entrypoint.

EPFMAP, EPFMP (bin, bin, bin, bin) returns(ptr) II-5-15
```
smt_pointer = epf$map (key, vmfa_unit, access_rights, status);
```
Map an EPF file to virtual memory. Key = K$COPY, K$DBG.

EPF$NF. See EPF$INFO.

EPF$NT. See EPF$INIT.

EPF$REG (1, 2 char(128)var, 2 (8)char(32)var, 2 char(1024)var, 2 bit(1), ptr, bin)
```
call epf$reg (register_info, error_aliases_ptr, status);
```
Register an EPF.

EPF$RELC, EPF$RL (ptr, ptr) returns (ptr)
```
real_virtual_address = epf$relc(epf_relative_ptr, smt_ptr);
```
Relocate EPF realtive pointer(ERP).

EPFRN, EPFRUN(bin, bin, bin [, char(*) var, bin, struc, struc, ptr]) II-5-19
```
smtp = epf$run (key, src_unit, status [, com_args, com_status,
               com_state, flags, rtn_function_ptr]);
```
Run an EPF.

EPF$SMAL
```
call epf$smal;
```
Permit linking to in-use static mode library.

EPF$SMDL
```
call epf$smdl;
```
Disallow linking to in-use static mode library.

EPF$SRCH (ptr, char(32)var, ptr)
```
call epf$srch(epf_smt_ptr, faulted_entryname, lib_entry_erp);
```
Search an EPF library to resolve a faulted entrypoint.

EPF$UREG (bin, char(128)var, bin, bin, bin)
```
Call epf$reg(remove_key, epf_pathname, epf_smt, first_proc_seg,
             epf_database, status);
```
Un-register EPF.

EPF$VK. See EPF$INVK.

EPF_ERR (fixed bin(15), char(1024) var)
```
call epf_err(err_code, info_str);
```
Print diagnostic error message on terminal.

EPF_RL (ptr)
```
call epf_rl (epf_smt_ptr);
```
Pop volatile EPF smt data for program and library EPFs.

EQUAL$ (char(32)var, char(32)var, char(32)var, bin) II-4-37
```
call equal$(obj_name, pattern, generated, code);
```
Generate (equal) name from a source name and a pattern.

ERKL$$ (bin, char(1), char(1), code) (svc = 1524) II-3-60
```
call erkl$$(key, erase_char, kill_char, code)
```
Read or set erase and kill chars. Chars are right justified, zero filled. Key = K$WRIT, K$READ.

ERRPR$ (bin, bin, char(*), bin, char(*), bin) (svc = 1402) III-3-30
```
call errpr$(key, error_code, message, message_len, file_name,
     file_name_len)
```
Interpret a return code. Key = K$NRTN, K$SRTN, K$IRTN.

ERTXT$ (bin, char(1024)var) III-2-9
```
call ertxt$(error_code, error_text);
```
Return the text of a specified error code.

EVAL_A (char(*) var, bit(1) aligned, ptr, ptr, fixed bin, fixed bin, fixed bin)
```
call eval_a (expression, op_switch, local_var_ptr,
          global_var_ptr, expr_size, error_code, com_status)
```
Evaluate all CPL vars in a character string.

EX$CLR III-7-35
```
call ex$clr;
```
Diable signalling of the EXIT$ condition upon program termination.

EX$RD (bin) III-7-36
```
call ex$rd (transmit_exit_setting);
```
Return value of the TRANSMIT_EXIT counter.

EX$SET III-7-37
```
call ex$set;
```
Enable signalling of EXIT$ on program termination.

EXIT (svc = 0105) III-5-7
```
call exit
```
Return to PRIMOS.

EXTR$A (char (*) var, char (*) var, bin, char (32) var, bin) II-4-39
```
call extr$a (full_path, parent_path, max_length, entryname,
          code);
```
Return parent tree and entryname from treename.

EXTRAC (bin, pointer, bin, char(*) varying, bin);
```
call extrac(caller_key, xp, xtype, xstr, xarglen);
```
Extract a spare data field from a string.

FATAL$ (bin);
```
call fatal$(code);
```
Fatal error handler.

FIL$D0 (char(32)var, bin)
```
call fil$d0(obj_name, code);
```
Delete a file or directory.

FIL$DL (char(128)var, bin) II-4-41
```
call fil$dl(object_pathname, code);
```
Delete a file.

FIND$BKT (ptr, char (32) var, bin) returns (ptr);
```
data_address = find$bkt (table_address, name, code);
```
Search a standard hash table for a bucket address.

FIND_U, FIND_UID (char(32) var, bin, ptr, ptr, bin(31), bin) returns (bit(1))
```
id_found = find_uid (user_id, vf_unit, addr(vf_header),
                     addr(uvf_entry), entry_pos, code);
```
Search system validation files for an entry.

FINFO$ (bin, ptr, bin) II-4-43
```
call finfo$(unit, addr(info_struc), code);
```
Return information about specified file unit.

FNCHK$ (bin, char(*)var) returns(bit(1)) II-4-45
```
name_ok = fnchk$(key, file_name)
```
Check a filename for valid format. Key = K$UPRC, K$WLDC, K$NULL, K$NUM.

FNONU$ (ptr, char(32) var, ptr, ptr, ptr) returns(bit(1));
```
cond_was_found = fnonu$(frame_ptr, condition_name,
                 onunit_or_last_ptr, catch_all_ptr, spec_ptr);
```
Find on-unit in specified stack frame.

FNSID$ (fixed bin, ptr, fixed bin, fixed bin);
```
call fnsid$ (key, addr(remote_id), max_entries, code);
```
Search and add entries to user's remote id database. (NPX) Key = KADD, KLIST, K$SRCH.

FORCEW (bin, bin [, bin]) (svc = 0115) II-4-47
```
call forcew(key, file_unit [, code])
```
Force write to disk immediately. Key = 0.

FORK$ (char(8), bin) returns(bit(1))
```
I_am_child = fork$(unique_id, code);
```
Creates a child process from within a program.

FPLEN$ (bin(31))
```
length = fplen$(Free_poll_id);
```
Return the length of the free pool queue.

FRE$RA (ptr) III-4-23
```
call fre$ra (rtn_function_ptr);
```
De-allocate space used for return info from command functions.

FRK$CP returns(bit(1));
```
foo = frk$cp;
```
Address copy routine for Forked processes.

G$METR (bin, ptr, bin, bin, bin, bin)
```
call g$metr(key, bufptr_arg, buf_size, user_arg, revision,
      code);
```
Get metering data of various sorts and flavors.

GEM$PB (bin, bin, bin, bin, bin, [bin, bin, ..., bin, bin])
```
call gem$pb(sec_code, eventid, nwords, len1, arg1,
      [len2, arg2, ..., len6, arg6]);
```
Probe to monitor ring3 activities.

GEM$R3 returns(bit(1))
```
monitoring_enabled = gem$r3();
```
Indicates whether ring 3 monitoring is enabled.

GEM$ST (bin, pointer, bin)
```
call gem$st (assign_buffer, addr(init_structure), code);
```
Control procedure for General Event Monitor (GEM).

GEM$WT (bin, pointer);
```
call gem$wt (lost_count, buffer_pointer);
```
Gate routine to wait for and dump General Event Monitor buffers.

GET$DPT (bin)
```
call get$dpt(program_session_deapth);
```
Get the depth of the program_session.

GET$DTR3 (bin, bin(31), bin) returns(ptr)
```
block_pointer = get$dtr3(storage_type, block_size, code);
```
Allocates given amount of storage in DTAR 3 according to storage type.

GETAT$ (1, 2 bin, 2 bin, 2 bin, 2 bin, 2 bin);
```
call getat$(system_defaults)
```
Reads system defaults and passes them to Edit_Profile.

GETID$ (ptr, bin, bin) II-2-21
```
call getid$(addr(id_struc), max_groups, code);
```
Get full user id.

GETREG ((*)bin)
```
call getreg(svec)
```
Sets tvec from svec.

GETSN$ (bin, bin, bin, (*)bin, bin, bin);
```
call getsn$(key, start_segno, num_segs, segno_array,
            num_segs_found, code);
```
Allocates a set of dynamic segments. Key = KUP, KDOWN, KUPC, KDWNC.

GET_REPL (bit(1)) returns(bit(2));
```
repy = get_reply(all_options);
```
Fetch a yes/no/null/next reply from command input stream.

GINFO ((6)bin, bin) (svc = 0112) III-2-10
```
call ginfo(xer_vec, xer_vec_len)
```
Return operating system info (PRIMOS II).

GMETR$ (bin, ptr,bin, bin, bin)
```
call gmetr$(key, addr(buffer), buf_size, code, user_number)
```

Get metering data of various sorts and flavors. Key = GM_SYS, GM_FS, GM_INT, GM_USER, GM_MEM, GM_DISK.

GNUSR$ (bin)
```
call gnusr$(network_user_number);
```
Gets the network process' user number.

GPAS$$ (char(32), bin, char(6), char(6), bin) (svc = 1504) II-2-23
```
call gpas$$(ufd_name, ufd_name_len, owner_pw, non_owner_pw,
            code);
```
Return passwords of sub-UFD.

GPATH$ (bin, bin, char(128), bin, bin, bin) II-4-49
```
call gpath$(key, file_unit, path_name, path_name_len,
      rtn_path_len, code)
```
Find pathname for file unit or current home or attach point. Key = K$UNIT, K$CURA, K$HOMA, K$INIA.

GT$PAR (bit(16), char(*)var, char(*)var, char(*)var, char(*)var, bin, struc, bin) III-6-27
```
call gt$par(key, white, quote, break, source_str, token_str,
            token_str_size, info, next_char);
```
Parse a character string into tokens separated by white space, quotes, and break characters.

GTSHR$ (bin(31), bin, ptr, ptr, bin)
```
call gtshr$(unique_seg_id, req_accesses, seg_to_share,
      dtar2_seg_ptr, code);
```
Map a DTAR2 segment onto a DTAR0 segment.

GV$GET (char(32)var, char(*)var, bin, bin) II-6-12
```
call gv$get(gvar_name, gvar_value, gvar_value_len, code);
```
Retrieve value of a global variable.

GV$SET (char(32)var, char(*)var, bin) II-6-14
```
call gv$set(gvar_name, gvar_value, bin);
```
Set the value of a global variable.

HASH_U, HASH_UID (char(32) var, bin) returns (fixed bin);
```
table_index = hash_uid (user_id, table_size);
```
Performs the current hashing function on the passed user ID.

HS$DRAIN
```
call hs$drain;
```
Drain the caller's per-user semaphore.

HS$NTFY (bin, bin)
```
call hs$ntfy(user_number, code);
```
Notify the specified user's per-user semaphore.

HS$WAIT returns(bin)
```
notified = hs$wait();
```
Wait on the caller's per-user semaphore.

I$GCLB (ptr, ptr)
```
call i$gclb(callers_sb, callers_lb);
```
Get EXIT_LB and EXIT_SB from CLDATA. (INFORMATION only).

I$ON Nonstandard.
```
FAR0 = addr(condition name (char(*)var))
FAR1 = addr(on-unit_ecb)
GR2  = Specifier ptr (0 => null())
GR5H = 1 if snap option is on, else 0.
JSXB I$ON
```
Make PL/I on-unit.

ICE$ III-5-8
```
call ice$;
```
Initialize command environment.

ICMTB_
Internal command table. Not a procedure.

ICPL_ (char(*) var, char(*) var, bin, bin, 1, 2 bit(1), 2 bit(1), 2 bit(14), ptr);
```
call icpl_(arg_source, args, com_status, src_unit,
           flags, rtn_function_ptr);
```
Invoke CPL interpreter on given file, processing suffix.

ICS2CT (bin, bin, bin)returns(bin)
```
success = ics2ct(key, device_address, data);
```
Allow OTA and INA from eagle monitor to controller. Key = 1 (INA), 2 (OTA).

IDCHK$ (bin, char(*)var) returns(bit(1)) III-2-22
```
id_ok = idchk$(key, id);
```
Check an id for valid format. Key = K$UPRC, K$WLDC, K$NULL, K$GRP.

IG$ABUF (ptr, ptr, bin)
```
call ig$abuf(lcptr, xcb, status);
```
Add a buffer to RTNQ. Simulates buffers being returned by controller.

IG$AWIR (ptr, bin)
```
call ig$awir(address, status);
```
Ring3 gate to wire a page.

IG$COLD (bin, bin, bin, bin)
```
call ig$cold(device, num_connections, num_windows, status);
```
Initialize database for a controller. If first time called for any controller, initialize over-all IGUANA database.

IG$DEQ (bin(31), ptr, bin, bin)
```
call ig$deq(lcid, buffer, size, status);
```
Dequeue an item (command or XCB) from INQ.

IG$ENQ (bin(31), ptr, bin, bin)
```
call ig$enq(lcid, buffer, size, status);
```
Enqueue an item (command or XCB) on an OUTQ.

IG$FIND (bin, bin, bin, bin, bin(31), bin(31), bin(31), bin)
```
call ig$find(device, lcn, q_array, buf_size, i_sem, o_sem, lcid,
             status);
```
Find a specific per-connection database for a particular controller

IG$GBUF (bin(31), ptr, bin, bin);
```
call ig$gbuf(lcid, buffer, size, status);
```
Obtain a buffer from RTNQ.

IG$RBUF (bin(31), ptr, bin)
```
call ig$rbuf(lcid, buffer, status);
```
Add a buffer to BUFQ. Becomes available for controller to write input data and put on INQ.

IG$RMV (bin(31), bit(16), ptr, bin, bin)
```
call ig$rmv(lcid, bit_mask, buffer, size, status);
```
Routines to remove buffers from all queues of a connection.

IG$SWIR (ptr, bin)
```
call ig$swir(address, status);
```
Ring3 gate to unwire a page.

IG$WAIT (bin(31), bin)
```
call ig$wait(lcid, sem);
```
Ring3 gate to wait on a semaphore (input done, output done).

IN$LO returns(bit(1)) III-2-23
```
in_grace_period = in$lo();
```
Return state of PPMD.IN_GRACE_PERIOD (i.e., force logout in progress).

INIT$3 (bin, bin, char (*) var, bin)
```
call init$3 (key, user_num, login_comline, cpl_unit);
```
Initialize ring 3 environment.

INSON$ (bin)
```
call inson$(key)
```
Initialize static on units. Key = 0 (ring 0), 3 (ring 3), 2 (both).

INTCM_ (char(32) var, entry, bit(3), bin, bit(1), bit(5)) returns (bit(1));
```
is_internal = intcm_ (command_name, entry_var, exp_wildcards,
                      eq_position, vfy_default, default_types);
```
Fetch local command table entry if any , else check system's table.

IO$GET_MSG (1, 2 bin, 2 bin, 1, 2 bin, 2 ..., 1, ..., bin)
```
call io$get_msg(wait_info, msg_sender, message, status);
```
Return a stored I/O related message for DSM/SM to log.

IO$PUT_MSG (1, 2 bin, 2 bin, 1, ..., bin)
```
call io$put_msg(wait_info, message, status);
```
Put a I/O related message into the queue of message for DSM/SM to log.

IOA$ (char(*), bin, [arg1, ... , arg99]) III-3-32
```
call ioa$(control_string, control_string_len
        [, arg1, ... , arg99]);
```
Write formatted string to terminal. See D for control string format.

IOA$ER (char(*), bin, [arg1, ... , arg99]) III-3-38
```
call ioa$er(control_string, control_string_len
        [, arg1, ... , arg99]);
```
Write formatted string to terminal after forcing on terminal output. See D for control string format.

IOA$RS (char(*), bin, bin, char(*), bin, [arg1, ... , arg99]) III-6-32
```
call ioa$(rtn_string, rtn_str_size, rtn_str_rtned_len,
      control_string, control_string_len [, arg1, ... , arg99]);
```
Return formatted string according to control string. See D for control string format.

IOAFM$ ((101)ptr (long), char(*), bin, bin)
```
call ioafm$(arg_pointers, buffer, buffer_max_size, rtn_len);
```
Process control format string. (IOA$)

IPC$C(bin, bin)
```
call ipc$o(mbx_id,code);
```
Close a IPC mailbox using the mbx_id specified.

IPC$CA
```
call ipc$ca;
```
Close all mailboxes the current user owns.

IPC$CM (bin, bin, bin)
```
call ipc$cm(mode_key, mbx_id, code);
```
Change mailbox access mode from read/write to specified mode.

IPC$GU (bin, bin, ptr, bin, bin, bin);
```
call ipc$gu(key, mbx_id, buf_ptr, buf_size, returned_size, code);
```
Get the desired mailbox user ID specified by key. Key = K$READ, K$WRIT, K$RDWR, K$MINE.

IPC$NC (bin, bin)
```
call ipc$nc (mbx_id, code);
```
Close a IPC mailbox with notification using the mbx_id specified.

IPC$O(bin, bin, char(128) var, bin, bin);
```
call ipc$o(access_key, notification_key, pathname, mbx_id, code);
```
Open an IPC mailbox for specified access using pathname. Access_key = K$READ, K$WRIT, K$RDWR. Notification_key = K$NFIN, K$NFSN.

IPC$O0 (bin, bin, char(*)var, bin, [1, 2 char(6), 2 bin, char(*)var], bin)
```
call ipc$o0(access_key, notification_key, entry_or_pathname,
     mbx_id, [uusrid, my_node], code);
```
Open an IPC mailbox for specified access using entryname for access control. Access_key = {k$read, k$writ, k$rdwr}; notification_key = {k$nfin, k$nfsm}.

IPC$R (bin, bin, ptr, bin, bin, bin, bin);
```
call ipc$r(read_key, mbx_id, buf_ptr, buf_size, msg_size,
          mbx_send_uid, code);
```
Receive a message from specified IPC mailbox waiting if specified. Read_key = K$READ, K$RDWT.

IPC$RA (bin, bin, ptr, bin, bin, bin, bin, bin);
```
call ipc$ra(read_key, buf_ptr, buf_size, mbx_id, msg_size,
            mbx_send_uid, code);
```
Receive a message from any IPC mailbox owned by the user. Read_key = K$READ, K$RDWT.

IPC$SA (bin, ptr, bin, bin);
```
call ipc$sa(mbx_id, msg_ptr, msg_size, code);
```
Send a message to any IPC user attach to specified mailbox.

IPC$SB (bin, ptr, bin, bin);
```
call ipc$sb(mbx_id, msg_ptr, msg_size, code);
```
Send a message to all IPC users attach to specified mailbox.

IPC$SS (bin, bin, ptr, bin, bin);

```
call ipc$ss(mbx_id, mbx_uid, msg_ptr, msg_size, code);
```
Send a message to a specific IPC user.

IPC$SSA (bit(1), bin, ptr, bin, bin)
```
call ipc$ssa(mbx_id, msg_ptr, msg_size, code);
```
Send a message to any IPC user attach to specified mailbox and notify the caller.

IPC$SSB (bit(1), bin, ptr, bin, bin)
```
call ipc$ssb(mbx_id, msg_ptr, msg_size, code);
```
Send a message to all IPC users attach to specified mailbox and notify caller.

IPC$ST (bin, bin, bin, bin);
```
call ipc$st(key, mbx_id, value, code);
```
Return various IPC statuses determined by user specified key. Key = K$NMSG, K$MROM, K$ROOM, K$NUSR, (K$NFYS).

IS$AB (bin, bin, bin) returns(ptr)
```
call is$ab(session_id, buffer_length, code)
```
AllocateBuffer - allocate an ISC data buffer.

IS$AS (bin, ptr, ptr, ptr, bin)
```
call is$pas(SessionID, ConnectMessage, ConfigInfo_ptr,
            SessionSyncs, ReturnCode);
```
AcceptSession - accept an ISC Session Request.

IS$CE (bin, bin)
```
call is$ce(session_id, code);
```
ClearException - clear an an outstanding exception

IS$EPFUS, IS$EPU (char(128)var, bin, bin) returns(bit(1));
```
in_use = is$epfus(target_tree, target_type, code);
```
Determine if an EPF is in use.

IS$FB (bin, ptr, bin)
```
call is$fb(session_id, buffer, code);
```
FreeBuffer - free an ISC data buffer.

IS$GE (bin, bin, ptr, bin)
```
call is$ge(session_id, exception_raised, message, code);
```
GetException - get details of an outstanding exception.

IS$GRQ (ptr, ptr, ptr, bin, ptr, bin)
```
call is$grq(TargetLLN, ConnectMessage, ConfigInfo_ptr, SessionID,
            AuthInfo_ptr, ReturnCode);
```
GetSessionRequest - get an incoming session request.

IS$GRS(bin, ptr, ptr, bin, ptr, bin)
```
call is$grs(SessionID, TargetLLN, AuthInfo_ptr, ResponseCode,
            ConnectMessage, ReturnCode);
```
GetSessionResponse - get response to ISC session request.

IS$GSA (bin, ptr, ptr, ptr, bin)
```
call is$gsa(SessionID, ConfigInfo_ptr, SessionSyncs_ptr,
            AuthInfo_ptr, ReturnCode);
```
GetSessionAttributes - provide attributes of a session.

IS$GSO (bin, ptr, bin, bin)

```
call is$gso(ArrayLength, SessionsOwned_ptr, SessionCount
      ReturnCode);
```
GetSessionsOwned - get a list of sessions owned by caller.

IS$GSS (bin, ptr, bin)
```
call is$gss(SessionID, StatusInfo_ptr, ReturnCode);
```
GetSessionStatus - provide session status information.

IS$PAS (bin, ptr, ptr, ptr, ptr, bin)
```
call is$pas(SessionID, ConnectMessage, ConfigInfo_ptr,
      InternalAuthInfo_ptr, SessionSyncs, ReturnCode);
```
AcceptSession - accept an ISC Session Request (privileged process).

IS$PRS (ptr, ptr, ptr, ptr, bin, ptr, bin)
```
call is$prs(TargetLLN, ConnectMessage, ConfigInfo_ptr,
      AuthInfo_ptr, SessionID, SessionSyncs, ReturnCode);
```
RequestSession - request an ISC session (privileged process).

IS$PTS (bin, bin, ptr, ptr, bin)
```
call is$pts(SessionId, ReasonCode, Message,
      InternalAuthInfo_ptr, ReturnCode);
```
TerminateSession - terminate an ISC session

IS$R (char(12), bin, bin)
```
call is$r(ServerUID, SessionRequestPending, ReturnCode);
```
RegisterProcessAsServer - register as an ISC Server.

IS$RE (bin, bin, bin)
```
call is$re(SessionId, ExceptionRaised, ReturnCode);
```
RaiseException - raise an exception on an ISC session.

IS$RM (bin, ptr, bit(1), bin)
```
call is$rm(SessionId, Message, IsExpedited, ReturnCode);
```
ReceiveMessage - receive a message on an ISC session.

IS$RS (ptr, ptr, ptr, bin, ptr, bin)
```
call is$rs(TargetLLN, ConnectMessage, ConfigInfo_ptr,
          SessionID, SessionSyncs, ReturnCode);
```
RequestSession - request an ISC session.

IS$SM (bin, ptr, bit(1), bin)
```
call is$sm(SessionId, Message, IsExpedited, ReturnCode);
```
SendMessage - send a message on an ISC session.

IS$STA (bin, ptr, bin)
```
call is$sta(SessionID, StatisticsInfo_ptr, ReturnCode);
```
GetSessionStatistics - provide ISC session statistics.

IS$TS (bin, bin, ptr, bin)
```
call is$ts(SessionId, ReasonCode, Message, ReturnCode);
```
TerminateSession - terminate an ISC session.

IS$U (bin)
```
call is$u(ReturnCode);
```
UnregisterProcessAsServer - unregister as an ISC Server.

ISACL$ (bin, bin)returns(bit(1)) II-2-25

```
is_acl_directory = isacl$(file_unit, code);
```
Get directory type (ACL or non-ACL).

ISFEPF () returns(bit(1))
```
parent_is_epf = isepf();
```
Determine if parent is an EPF.

ISPRIV$,ISPRV$ (bit(16), bin, char(128)var, char(32)var) returns (bit(1))
```
user_is_priv = ispriv$(privilege_definition, user_type,
               operation, ck_group);
```
Check user privilege.

ISREM$ (bin, char(128)var, bin, char(32)var, bin)returns(bit(1)) II-4-52
```
file_is_remote = isrem$(key, filename, unit, system_name, code);
```
Return information on remoteness of a filesystem object. Key = k$name, k$unit.

ISUREM$(bin, char(32)var, bin) returns(bit(1))
```
unit_is_remote = isurem$(unit, sysnam, code);
```
Return information on remoteness of a filesystem object open on a unit.

JOB$0 (bin, bin, bin, (entry_length) bin, (entry_length) bin, bin, bin)
```
call job$0(key, queue_index, priority, old_entry, new_entry,
           entry_length, code);
```
Operate on batch queue control file in a secure manner. (JOB only)

JOB$1 (bin, ptr, bin)
```
call job$1(key, addr(qinfo), addr(job_entry), code);
```
Queue control gate for BATCH subsystem.

KLM$ES (struc, bin)
```
call klm$es(klm_struc, code);
```
Return serialization information on an EPF.

KLM$MV (ptr, bin)
```
call klm$mv(klm_ptr, status);
```
Move klm info from invokers buffer into level class storage.

KLM$PR (bin)
```
call klm$pr(code);
```
Output copyright notice.

KLM$RT (struc, bin)
```
call klm$rt(klm_struc, code);
```
Return klm infomation.

KTRAN$ (char(*) var, bin) returns (bin)
```
hash_key = ktran$(name, modulus);
```
Provides simple hash on *name*.

LDISK$ (bin, char(32) var, ptr, bin, bin) II-4-54
```
call ldisk$ (key, system_name, addr(disk_list), max_entries,
     code);
```
Return information on the system's disk list. Key = KALL, KLOCL, KREM, KSYS.

LDSKU$ (fixed bin, (128) bit(1), fixed bin)
```
call ldsku$ (logical_device, user_list, code);
```
Returns bit-encoded list of users using a specified logical device.

LGINI$
```
call lgini$(key, code)
```
Turn on and off OS and network logging.

LIBTBL - Library tables; not a routine.

LIMIT$ (1, 2 bit(8), 2 bit(8), bin(31), bin, bin) III-8-36
```
call limit$(key, limit, reserved, code);
```
Set/read cpu, realtime, and login time limits. KeyL = 1 (read), 2 (set). KeyH = 1 (cpu sec), 2 (login min), 5 (cpu watchdog sec), 6 (real-time watchdog min), 7 (real-time watchdog sec).

LIST$CMD (char(32) var, bin) II-6-16
```
call list$cmd (wildcard_match, status);
```
List internal mini-level commands by wildcard match.

LIST$EN (char(128) var, (8) char(32) var, bin, bin, ptr, bin)
```
call list$en (pathname, entrynames, num_total, num_found,
              rtn_list_ptr, error);
```
Return library entrynames in an EPF library.

LN$SET(pointer, bin)
```
call ln$set(smtp, status);
```
Sets a library already mapped in into a user's search list.

LOGIN$ (char(256) var, fixed bin)
```
call login$ (com_args, com_status);
```
Parsing and routing routine for the LOGIN command.

LOGO$$ (bin, bin, char(*), bin, bin(31), bin) III-2-24
```
call logo$$(key, user_number, user_name, user_name_len, reserved,
            code)
```
Log out a process or user. Key = -1 - all; 0 - self; 1 - user_number; 2 - user_name.

LOGOU$
```
call logou$;
```
Initial processor for the LOGOUT command.

LON$CN (bin) III-5-20
```
call lon$cn(key);
```
Enable or disable logout notification. Key = 0 - off; 1 - on.

LON$PR(bin, (6)bin)
```
call lon$pr(code, msginfo);
```
Print phantom logout notification message.

LON$R (ptr, bin, bit(1), bin) III-5-21
```
call lon$r(addr(message), message_len, more_waiting, code);
```
Retrieve logout info.

LOV$SW returns (bit (1));
```
login_over_login_not_allowed = lov$sw();
```
Checks to see if login over login is allowed.

LSR$DLAY (bin, bin, bin, bin, bin)
```
call lsr$dlay(min, max, margin, who, status);
```
Set slope of delay curve for terminal of specified user.

LSR$ERR (char(*), bin, bin)
```
call lsr$err(message, message_length, status);
```
Gives the Login Server a way to log to the console.

LSR$GETC (bin, char(1), bin)
```
call lsr$getc(line_number, retchar, status);
```
Special Login Server gate to let it get characters from its lines.

LSR$GLSE (bin(31), bin, bin)
```
call lsr$glse(Timeout, NewEvent, Status);
```
Routine to return Login Server Event.

LSR$GTLL ((*)bin, bin, bin, bin)
```
call lsr$gtll(WhichLines, ArraySize, HowMany, Status);
```
Get list of loginable "lines" (buffer indices).

LSR$GTLO (bin, bin, char(*)var, bin)
```
call lsr$gtlo(Who, Why, Command, Status);
```
Manage logout information for the Login Server.

LSR$GTNM (bin, bin)
```
call lsr$gtnm(NewMaxusr, Status);
```
Retrieve the maxusr value for the Login Server.

LSR$GTPR (bin, bin, bin)
```
call lsr$gtpr(who, newprocess, status);
```
Obtain a process number for use with a given line.

LSR$KLSR (bin)
```
call lsr$klsr(status)
```
Post a suicide event for the Login Server.

LSR$SLSR (bin)
```
call lsr$slsr(code);
```
Procedure to start up the Login Server.

LSR$SRLI (bin, bin, struc, bin)
```
call lsr$srli(Who, ProcessNo, Attr, Status);
```
Start up a user process to be used for logged-out user going remote.

LSR$STPR (bin, bin, struc, bin)
```
call lsr$stpr(Who, ProcessNo, Attr, Status);
```
Routine to start up a local user's process.

LSR$TNOA (bin, char(*), bin)
```
call lsr$tnoa(user, string, count);
```
Login Server terminal output (Login Server only).

LSR$TRBC (bin, bin, bin)
```
call lsr$trbc(line, toWhom, Status);
```
Transfer line (buffer) control from one process to another.

LSR$USRA (char(80), bin, bin, bin)
```
call lsr$usra(line, status, for_whom, code);
```
USRASR command processer for Login Server.

LUDEV$ (bin, ptr, bin, bin);

```
call ludev$(user, addr(rtn_struc), max_devs, code);
```
List a user's assigned devices.

LUDSK$ (fixed bin, ptr, fixed bin, fixed bin); II-4-57
```
call ludsk$ (user, addr(disk_list), max_entries, code);
```
Returns list of all disks currenty in use by a given user.

LUID$ (bin, bin(31), bin)
```
call luid$(unit, uid, code);
```
Return a unique ID consisting of the ldev and BRA.

LV$GET (ptr, char(32)var, char(1024)var, bin, bin) II-6-18
```
call lv$get(vcbp_arg, var_name, var_value, var_size, code);
```
Get local variable.

LV$SET (ptr, char(32)var, char(1024)var, bin) II-6-20
```
call lv$set(vcb_ptr, variable, value, code);
```
Set local user variables.

M2SMA$(bin, bin) returns(bin)
```
runit = m2sma$(unit, code);
```
Returns the master-to-slave mapping for the remote file unit.

MAXUS$ (char(80), bin)
```
call maxus$(line, status);
```
Carry out the MAXUSR operator command.

MESSG$ (char(32), char(*), bin, bin, bin)
```
call messg$(user_name, comline, message, msg_code, code);
```
Handle message command.

MGSET$ (bin, bin) III-9-5
```
call mgset$(key, code);
```
Set receiving state for messages. Key = K$ACPT, K$DEFR, K$RJCT.

MIR_OFF_CMD$ (char(*)var, bin)
```
call mir_off_cmd$(CommandArgs, CommandStatus);
```
Process MirrorOff command.

MIR_ON_CMD$(char(*)var, bin)
```
call mir_on_cmd$(CommandArgs, CommandStatus);
```
Process MirrorOn command.

MKLB$F (int*2, real*8) III-7-20
```
call mklb$f(fortran_label, rtn_pl1_label)
```
Make PL/I compatible label in fortran program.

MKON$F (int*2(*), int*2, external) III-7-21
```
call mkon$f(condition_name, condition_name_len, routine)
```
Create an on-unit in FTN.

MKON$P (char(*), bin, entry) III-7-23
```
call mkon$p(condition_name, condition_name_len, handler);
```
Create an on-unit in F77 or PL1G.

MKONU$ (char(*)var, entry) options(shortcall 20) III-7-25
```
call mkonu$(condition_name, handler);
```
Create an on-unit in PMA, SPL, or PLP.

MKONX$ (char(*) var, entry, ptr, bit(16)) options (shortcall(18))
```
call mkonx$ (condition_name, onunit_proc, specifier, flags);
```
Make PL/I on-unit.

MKSH1$ (bin(31), bin, bin, ptr, bin)
```
call mksh1$(unique_seg_id, req_accesses, limiting_accesses,
      dtar2_seg_ptr, code);
```
Make a pure DTAR 2 shared area.

MKSON$ (entry, fixed bin);
```
call mkson$ (sou_routine, code);
```
Make a static on-unit in either ring 0 or ring 3.

MM$MLPA (bin, bin)
```
call mm$mlpa(segment, status);
```
Make an out of bounds last page available.

MM$MLPU (bin, bin)
```
call mm$mlpu(segment, status);
```
Make the last page of a segment unavailable.

MOVB(ptr, ptr, bin)
```
call movb(from, to, number_of_bytes);
```
Moves words ((number_of_bytes + 1)/2) from area pointed to by *from* to area pointed to by *to*.

MOVEW$ (ptr, ptr, bin) III-6-34
```
call movew$(from, to, count);
```
Move *count* words from area pointed to by *from* to that pointed at by *to*.

MOVWDS (ptr, ptr, fixed bin(31))
```
call movwds(from, to, number_of_words);
```
Moves *number_of_words* from *from* to *to*.

MSG$ (bin, char(*), bin, char(*), bin, bin, char(*), bin, bin, char(*), bin, (131) bin)
```
call msg$ (key, from_name, from_user_num, to_name,
           to_user_num, name_len, from_system_name,
           system_name_len, time_sent, text, text_len,
           error vector);
```
Send message using specified banner information. NPX only.

MSG$ST (bin, bin, char(*), bin, char(*), bin, bin) III-9-3
```
call msg$st(key, user_num, system_name, system_name_len,
            user_name, uname_len, status);
```
Return receiving state of a user. Key = K$READ, 2 (read by user_num).

N$AADR (char(16)var, char(32)var, bin)
```
call n$addr(address, name, code);
```
Add a node "addr block" to the network database.

N$AHCB (char(32)var, char(32)var, bin, bin, bin, bin, bin, bin, bin, bin, bin, bit(16), bin, bin, bin, bin)
```
call n$ahcb(node_name, pdn_name, pdn_flag, maxvc, window,
            packet_size, block_type, line_no, fdx_flag,
            slccon, prdsc, hcbid, lapflg, i_am_dte, code);
```
Add an HCB block and a linedef block to the database.

N$ANAM (char(32)var, bit(16), bit(16), bit(16), char(32)var, char(32)var, char(32)var, bin, bin)

```
call n$anam(node_name, netbits, rltbits, fambits,
            npxpsw, ihdxpas, ohdxpas, nodtype, code);
```
Add a node "name block" to the network database.

N$APDN (bin, bin, bin, char(32)var, char(4)var, char(6)var, char(6)var, char(6)var, char(6)var, bin)
```
call n$apdn(iti_typ, addr_typ, thru_key, pdn_name, dnic,
            creq_fctys, cacpt_fctys, rlt_fctys, rlg_fctys, code);
```
Add a "pdn block" to the network database.

N$APTH (char(16)var, char(32)var, bit(16), bin, char(32)var, bin, bit(1), bin)
```
call n$apth(address, name, access, hcbid, gate_name, pthid,
                    path_online, code);
```
Add a "path block" to the network database.

N$ASAD (char(16)var, char(32)var, bin)
```
call n$asad(passed_addr, pdn_name, code);
```
Add an address to a source address chain.

N$CHCB(char(32)var, bit(3), bin, ptr, bin)
```
call n$chcb(name, pnet, line, buffer_ptr, code);
```
Modify an existing host block.

N$HONE(char(32)var, bit(3), bin, ptr, bin)
```
call n$hone(name, pnet, line, buffer_ptr, code);
```
Return description of one host-block (packet-level).

N$INIT (bin);
```
call n$init(code);
```
Initialize all the network databases.

N$IPDN (bin);
```
call n$ipdn(code);
```
Fill the PDN table with known pdn values.

N$LALL (ptr, bin)
```
call n$lall(buffer_ptr, error_code);
```
Gathers statistics for all primenet synchronous lines.

N$LCFG (bin, ptr, bin)
```
call n$lcfg(line_num, buffer_ptr, error_code);
```
Gathers configuration statistics for one primenet synchronous line.

N$LDYN (bin, ptr, bin)
```
call n$ldyn(line_num, buffer_ptr, error_code);
```
Gathers dynamic statistics for one primenet synchronous line.

N$NETS (bin, bin, bin);
```
call n$nets(my_ring_id, ring_block_size, code);
```
Do final network configuration and setup.

N$PNC (bin, pointer, bin, pointer) returns(bin)
```
status = n$pnc(pnc_number, traffic_buf, traffic_buf_size,
               trace_buffer);
```
Gather pnc statistics data.

N$RTRC (bit(1), bin);

```
call n$rtrc (on_off_flag, error_code);
```
Turn network ring tracing on/off.

N$SPME (char(32)var, bin, bin, bin, char(16)var, bin, bin);
```
call n$spme(my_name, maxvc, window, packet, comp_addr,
            hcbid, code);
```
Add all the "myself specific" data to the network databases.

N$VALL (ptr, bin)
```
call n$vall(buffer_ptr, error_code);
```
Gathers data for all virtual circuits.

N$VONE (bin, ptr, bin)
```
call n$vone(vcid, buffer_ptr, error_code);
```
Gathers statistics for one virtual circuit.

NETPRC
```
call netprc;
```
Network process running in ring 0.

NETSET (bin);
```
call netset(error_code);
```
Checks authorization of user starting network & init network segments.

NEWLV$ ();
```
call newlv$;
```
Pushes a new command level.

NPX$RL returns(entry(ptr)var);
```
entry_point = npx$rl();
```
Called by SLAVE_CK to retrieve the entry point of any_handler.

NPX$SL (entry(ptr));
```
call npx$sl(entry_point);
```
Called by SLAVE to store its any_handler in ring 0 data base.

NPXPRC (bin, *, *, *)
```
call npxprc(key, arg1, arg2, arg3);
```
Call random NPX routine. Key = CVTNAM(6), CVTNUM(7), RTICK(12), LOGMES(15), WNAME(17), RFMREV(18), CLUP$R(20), CLSBYN(21), RR0PW(24), CHKR0P(25), RGROUP(26), WGROUP(27), LOGIN(28), LOG21(29), LOG22(30), LOG23(31), LOG24(32), LOG25(33), USRTYP(34) (also HBWAIT(22), LOGO5(23), XLWAIT(35), LOG26(36), LOG44(44), LOG45(45))

NS$CRHOS(char(16)var, bin)
```
call ns$crhos(host_name, error_code);
```
Create a host on an extant LAN.

NS$CRLAN (char(32)var, bit(2), bit(2), (*) char(16)var, bin)
```
call ns$crlan(lan_name, unconfig_lts_ok, media_type,
              ntwk_mgmt_host, error_code);
```
Create a LAN node in the NSS database.

NS$CRLHC (char(32)var, char(16)var, bit(8), bin, bin, char(6), bin)
```
call ns$crlhc(lan_name, host_name, function, lhc_number,
              dev_addr, mac_addr, error_code);
```
Create an LHC on an extant host and LAN.

NS$CRLTS (char(32)var, char(16)var, bit(8), char(6), bin)
```
call ns$crlts(lan_name, lts_name, function, mac_addr,
              error_code);
```
Create an LTS on an extant LAN.

NS$DLTSA (char(6), bin)
```
call ns$dltsa(mac_addr, error_code);
```
Delete an LTS by address.

NS$DLTSN (char(16)var, bin)
```
call ns$dltsn(lts_name, error_code);
```
Delete an LTS by name.

NS$FLAG (bit(1), bin, bit(1), bin)
```
call ns$flag(write, flag_no, value, code);
```
Read or write NSS client visible flag.

NS$FLFUN (bit(8), bin);
```
call ns$flfun(functions, error_code);
```
Flush a function from the NSS database.

NS$RHA (char(6), pointer, bin)
```
call ns$rha(mac_addr, host_rec_p, error_code);
```
Read host description by address.

NS$RHI (bin, char(*)var, char(10), pointer, bin)
```
call ns$rhi(key, name, handle, host_rec_p, error_code);
```
Read host and LHC descriptions.

NS$RLA (char(6), pointer, bin)
```
call ns$rla(mac_addr, lts_rec_p, error_code);
```
Read LTS description by address.

NS$RLI (bin, char(32)var, char(10), pointer, bin)
```
call ns$rli(key, name, handle, lts_rec_p, error_code);
```
Read LTS description.

NS$RNI (bin, char(32)var, char(10), pointer, bin)
```
call ns$rni(key, lan_name, handle, lan_rec_p, error_code);
```
Read LAN description.

NS$SEC (bin)
```
call ns$sec(code);
```
Ensure that caller is user 1 or ACL group member.

NS$SFUNA (bin, char(6), bit(8), bin)
```
call ns$sfuna(key, mac_addr, function, error_code);
```
Set the function of an LHC or LTS based upon MAC address.

NS$SFUNI (bin, char(16)var, bin, bit(8), bin)
```
call ns$sfuni(key, host_name, lhc_number, function,
              error_code);
```
Set the function of an LHC based upon host name and lhc number.

NS$SSTAA (char(6), bit(3), bin)
```
call ns$sstaa(mac_addr, state, error_code);
```
Set the state of an LHC or LTS based upon MAC address.

NS$SSTAI (char(16)var, bin, bit(3), bin)
```
call ns$sstai(host_name, lhc_number, state, error_code);
```
Set the state of an LHC based upon host name and lhc number.

NS$XAN (char(6), char(16)var, bin, bin)
```
call ns$xan(mac_addr, name, lhc_number, error_code);
```
Translate an address to a name (and, for hosts, an LHC number).

NS$XNA (char(16)var, bin, char(6), bin)
```
call ns$xna(name, lhc_number, mac_addr, error_code);
```
Translate a name to an address.

NT$AS (bin, bin, char(16)var, bin, bit(1), bin)
```
call nt$as(primos_line, media_type, lts_name,
           lts_line, permanent, error_code);
```
Associate an LTS line with a Primos line number.

NT$CHECK (bit(16), bin)
```
call nt$check(lhc_list, error_code);
```
Check for required LHCs configured and downline loaded.

NT$CM
```
call nt$cm
```
NTS connection manager (part 1).

NT$CMODE (bin)
```
call nt$cmode(status);
```
Force the NTS terminal line of the current process back to LTS command mode.

NT$INIT (char(128)var, bin)
```
call nt$init(config_name, error_code);
```
Initialize NTS database.

NT$LTS (bin, bin, char(16)var, bin, char(6), bin)
```
call nt$lts(primos_line, media_type, lts_name,
            lts_line, mac_address, error_code);
```
Return NTS line connection info.

NT$NNAME (char(128)var, bin)
```
call nt$nname(config_pathname, error_code);
```
Return NTS config file pathname.

NT$RAS (bin, bin, char(32)var, char(6), bin, bit(1), bin, char(32)var, bin)
```
call nt$ras(line, user_no, user_name, lts_address, lts_line,
            permanent, as_user_no, as_user_name, error_code);
```
Read an entry from the NTS associate table.

NT$START (bin, bin)
```
call nt$start(lhc_number, error_code);
```
Start NTS.

NT$STOP (bin, bin)
```
call nt$stop(lhc_number, error_code);
```
Stop NTS.

NT$UAS (bin, bin, char(16)var, bin, bin)

```
call nt$uas(primos_line, media_type, lts_name,
            lts_line, error_code);
```
Dissociate an LTS line from a Primos line number.

OERRTN (bin, bin, bin, char(*), bin, char(*), bin);
```
call oerrtn(alt_val, alt_rtn, code, text, text_len, name,
      name_len);
```
Old style error handling.

OPEN$B (bin, char(*) var, bin, bin, bin) returns(bin(31));
```
char_pos = open$b(open_key, tree, unit, type, code);
```
Open a branch by tree name (nonstandard).

OPN$SR (char(32)var, char(128)var, char(128)var, bin, bit(5), char(128)var, bin, bin, bin);
```
call opn$sr(search_list, referencing_dir, file_path, open_mode,
            types, found_path, out_unit, out_type, code);
```
Open file using a search list. (Obsolete; will be removed; Use OPSR$).

OPN$SRSF (char(32)var, char(128)var, ptr, bin, bin, bit(5), char(128), bin, char(32)var, char(128)var, bin, bin, bin);
```
call opn$srsf(search_list, file_path, suffix_list_ptr,
 n_suffixes, open_mode, types, referencing_dir, suffix_index,
 file_basename, found_path, out_unit, out_type, code);
```
Open file using a search rule and suffices. (Obsolete, will be removed; use OPSRS$).

OPSR$ (char(32)var, char(128)var, bit(16), bin, char(128)var, bin, bin, char(128)var, bin)
```
call opsr$(list_name, referencing_dir, valid_types, open_key,
           file_path, unit, out_type, found_path, code);
```
Open a file system object using a search list.

OPSRS$ (char(32)var, char(128)var, bit(16), bin, char(128)var, bin, bin, bin, ptr, char(32)var, bin, char(128)var, bin)
```
call opsrs$(list_name, referencing_dir, valid_types, open_key,
      file_path, unit, out_type, n_suffices, suffix_list_ptr,
      basename, suffix_index, found_path, code);
```
Open an object using search rules and suffix processing.

PA$DEL (char(32)var, bin) II-2-27
```
call pa$del(partition_name, code);
```
Delete a priority ACL.

PA$LST (char(128)var, ptr, bin, bin) II-2-28
```
call pa$lst(object_pathname, addr(acl_struc), max_acl_entries,
      code);
```
Read a priority ACL.

PA$LST0 (char(32)var, ptr, bin, bin)
```
call pa$lst0(object_name, logical_acl_ptr, max_entry_count,
      code);
```
Return the contents of a priority ACL in logical format.

PA$SET (char(32)var, ptr, bin) II-2-30
```
call pa$set(partition_name, addr(acl_struc), code);
```
Set a priority ACL.

PAR$RV (char(32)var, bin) returns(bin) II-4-59
```
rev_no = par$rv(partname, code);
```
Returns the partition rev. stamp of a named disk partition

PBH$GD ((1024)bin31), 1, 2 bin, 2 like pbhcom, bin)
```
call pbh$gd(arg_counters, arg_struc, code);
```
Get data for PB histogram.

PBH$ON (bin, bin, (max_num_segs)bin(12), bin)
```
call pbh$on(arg_user_number, arg_num_segs, arg_seg_numbers,
            code);
```
PB Histogram Facility Startup/Access entries.

PHANT$ (char(*), bin, bin, bin, bin) III-10-8
```
call phant$(file_name, file_name_len, file_unit, user_num, code);
```
Start a phantom (Obsolete; use PHNTM$).

PHDBG (ptr, bin, bin)
```
cal phdbg(free_store_area_ptr, length, code);
```
Returns addresses of common area for protocol handler. (RJE)

PHNTM$ (bit(16), char(32), bin, bin, bin, bin, char(128), bin) III-5-23
```
call phntm$(cpl_flag, file_name, file_name_len, file_unit,
      user_num, code, cpl_args, cpl_args_len)
```
Start a phantom.

PID$CK (1, 2 char (6), 2 fixed bin) returns (bit(1) aligned);
```
id_is_valid = pid$ck (target_uusrid);
```
Validates process unique id.

PID$GET (char(8))
```
call pid$get(unique_id);
```
Get the PID of the current process.

PK2LDV (char(*) var, bin, bin, bin);
```
call pk2ldv(packname, packlen, node, ldev)
```
Convert disk pack name, node number into a logical device number.

PMSG$
```
call pmsg$;
```
Print messages on the caller's terminal.

PNM$CHK (bin, char(32)var, bin, bin)
```
call pnm$chk(lhc_nbr, lan300_name, dev_addr, error_code)
```
Performs the consistency check for Ethernet Host Controller.

PNM$RLHB (bin, ptr, bin)
```
call pnm$rlhb(lhctbl_number, lhctbl_info, return_code);
```
Access data from the LHCTBL Data Structure.

PNM$RNMB (bin, bin, pointer, bin)
```
call pnm$rnmb(action_code, data_from_nmdb, sem_addr,
      return_code);
```
Access data from the Ring0 Network Management Data Structure.

PNM$SEC (bin)
```
call pnm$sec(code);
```
Security check for Network Management gates.

PNM$WLHB (bin, bin, ptr, bin)

```
call pnm$wlhb(action_code, lhctbl_number, lhctbl_info,
      return_code);
```
Update the LHCTBL data structure.

PNM$WNMB (bin, (2)bin, bin)
```
call pnm$wnmb(action_code, data_for_nmdb, return_code);
```
Update the Network Management Ring0 data structure

PRERR (bin) (svc = 0111) III-10-9
```
call prerr(user);
```
Print name and/or message from user's ERRVEC (obsolete).

PRI$RV (char(16)var) III-2-12
```
call pri$rv(primos_rev);
```
Returns the Primos rev. stamp of the currently running operating system.

PRIO$CH (bin, bin, bin)
```
call prio$ch(pdev_index, pratio, err_code);
```
Routine to change PRATIO values.

PRIO$PD (bin, bin)
```
call prio$pd(pdev_count, err_code);
```
Routine to return the number of paging partitions on the system.

PRIO$ST (bin, bin, bin, bin)
```
call prio$st(pdev_index, pratio, ldev, err_code);
```
Routine to return a specific pratio value.

PRJID$ (char(32)var) III-2-26
```
call prjid$(project_id);
```
Return project ID of current user.

PRVSB_ (ptr, bit(1), bit(1), bin) returns (ptr)
```
prev_ptr = prevsb_ (curr_ptr, crawl_flag, fix, cs_depth);
```
Find previous stack frame given pointer to current one.

PRWF$$ (bin, bin, ptr, bin, bin(31), bin, bin) (svc = 1506) II-4-61
```
call prwf$$(key, file_unit, addr(buffer), num_words, position,
     num_words_transferred, code);
```
Postion, read or write to a file. Key = (K$READ, K$WRIT, K$POSN, K$TRNC, K$RPOS) + (K$PRER, K$POSR, K$PREA, K$POSA) + (K$CONV, K$FRCW)

PTIME$ returns(bin(31)) III-2-27
```
process_time = ptime$();
```
Returns process time since logged in.

PTRAP$,PTRAP (= P3TRAP)
```
CALF PTRAP
```
FIM for restricted mode (RXM) and illegal instruction (ILL).

PWCHK$ (bin, char(*)var) returns(bit(1)) III-2-28
```
password_ok = pwchk(key, password)
```
Check a password for valid format. Key = K$UPRC, K$NULL.

PWDIR$ (bit(1), bin)
```
call pwdir$(on_or_off, code);
```
Enable/Disable creation of password directories.

PX$BIRTH (bin, bin, char(34), bin)
```
call px$birth(my_id, parent_id, command, status_code);
```
Record the birth of a Primix process.

PX$CREA (char(128) var, ptr, bin)
```
call px$crea(dirname, info, code);
```
Special version of dir$cr -- presets ACL.

PX$CREA0 (char(32) var, bin)
```
call px$crea0(dirname, code);
```
Special version of dir$cr -- presets ACL (ring 0 part, sets ACL).

PX$CWAIT (bin, bin)
```
call px$cwait(user_id, status_code);
```
Primix PM support for pause system call.

PX$DEATH (bin, bin, ptr, bin, bit(1), bin)
```
call px$death(my_id, child_status, snode_ptr, snode_count,
              parent_wait, status_code);
```
Record the death of a Primix process.

PX$DUMP (bin, bin, ptr, bin)
```
call px$dump(my_id, expected_version, ptr_dump_table,
     status_code);
```
Primix dump/who/write/wall commands support.

PX$EXEC (bin, char(34), bin)
```
call px$exec(my_id, command, status_code);
```
Record the name of the command being executed for Primix.

PX$INIT (bin, bin, bin);
```
call px$init(ver_num, lisc_number, status_code);
```
Initialize Primix.

PX$MXUSR (bin, bin);
```
call px$mxusr(max_users, status_code);
```
Handles the SET_PRIMIX_USERS command.

PX$PAUSP (bin, bin)
```
call px$pausp(user_id, status_code);
```
Primix PM support for pause system call.

PX$PDATA (bin, bin, bin, (*)bin, bin)
```
call px$pdata(user_id, expected_version, buf_size, buffer,
     status);
```
Return Primix process data for the indicated user.

PX$RDSIG (bin, bin, bin, (*)bin(31), bin)
```
call px$rdsig(user_id, num_expected, num_returned, signals,
     status_code);
```
Return current Primix signal.

PX$SGACT (bin, bin(31), bin) returns(ptr)
```
action = px$sgact(pid, signal, status_code);
```
Return current response to a Primix signal.

PX$SGSYS (bin, bin(31), ptr, ptr, bin);

```
call px$sgsys(user_id, signal, action, prev_action, status_code);
```
Primix PM support for the Signal System call function.

PX$SHDWN (bin)
```
call px$shdwn(code);
```
Shut down Primix.

PX$SIGNL (bin, bin(31), bin, bin)
```
call px$signl(user_id, signal_num, target, status_code);
```
Signal a process for Primix PM support.

PX$SRCH (bin, char(128)var, bin, bin, bin);
```
call px$srch(action+ref+newfil, filename, funit, type, code);
```
Special version of srch$$ for creating items with preset ACL.

PX$SRCH0 (char(32)var, bin)
```
call px$srch0(filename, code);
```
Special version of srch$$ for creating items with preset ACL.

PX$SVTIM (bin, bin(31), bin(31), bin)
```
call px$svtim(key, cpu, io, code);
```
ates the CPU and I/O time for the forked process.

PX$SYNC (bin, bin)
```
call px$sync(user_id, status_code);
```
Primix PM support for fork synchronization.

PX$UNSYNC (bin, bin)
```
call px$unsync(user_id, status_code);
```
Primix PM support for fork synchronization.

PX$WAITP (bin, bin, bin, ptr, bin, bin)
```
call px$waitp(user_id, child_status, child_id, file_info_ptr,
          file_info_count, status_code);
```
Primix PM support for wait system call.

Q$READ (char(128)var, (8)bin(31), bin, bin, bin) II-4-68
```
call q$read(path_name, quota_info, quota_info_len, dir_type,
      code)
```
Read quota information.

Q$READ0 (char(32)var, (8)bin(31), bin, bin, bin)
```
call q$read0(dir_name, output_structure, max_entries, dir_type,
      code);
```
Read quota information for current directory.

Q$SET (bin, char(128)var, bin(31), bin) II-4-71
```
call q$set(key, path_name, max_quota, code);
```
Set quota maximum. key = K$SMAX.

QUIT$ (bit(16) aligned) III-3-62
```
call quit$ (pending_quit);
```
Determine if there are any pending quits. pending_quit = 0 if none.

QUOTE_ (char(*) var, char(*) var, bin, bin);
```
call quote_ (input_string, output_string, output_size, status);
```
Quote a given string.

R$ALLC(ptr, fixed bin) returns(ptr);
```
smt_pointer = r$allc(smtp, status);
```
Allocate linkage for an EPF. Obsolete after 19.3; use EPF$ALLC.

R$ALO1 (char(8), bin)returns(bin);
```
alloc_count = r$alo1(slave_id, code);
```
This routine increment the ALOCNT by 1.

R$ALOC(fixed bin);
```
call r$aloc(remote_node);
```
Allocate an index to a slot in VCDATA for a node number. (NPX)

R$BGIN (bin, char(8), char(*), bin, (*8392)bin, bin(31), bin, variable);
```
call r$bgin(key, slave_id, subr_name, subr_name_len, buffer,
            buffer_len, code [, arg1, arg1len, arg1key, ... ,
            arg15, arg15len, arg15key]);
```
The user callable interface to NPX for synchronous and asynchronous RPCL. Key = 0 (2 - called by R$CALL).

R$CALL (bin, bin, char(*), bin, bin, variable);
```
call r$call(key, rnode, subroutine_name, subroutine_namlen,
            rcode, arg1, arg1len, arg1key, arg2, arg2len,
            arg2key, ...);
```
Perform remote procedure call. Key = 0, K$FUNC.

R$CKNT (char(32)var, bin);
```
call r$cknt(node_name, code);
```
Subroutine to check the validity of the supplied node name.

R$CPF (ptr, bit(3), fixed bin, bit(1), bit(4));
```
call r$cpf(smtp, expand_wildcards, eq_position, vfy_default,
           match_type_default);
```
Get command processor flags from an EPF. Obsolete after 19.3.

R$CVT(char(32), bin) returns(bin);
```
nodenum = r$cvt(node_name, node_name_length);
```
Convert node name to the corresponding node number. Obsolete after 19.3; use NPXPRC.

R$DEL (ptr);
```
call r$del(smtp);
```
Delete an EPF from a user's address space. Obsolete after 19.3.

R$END (bin, char(8), bin, bin, bin) returns(bin(31));
```
func_rtn = r$end(key, slave_id, buffer, time, code);
```
The asynchronous remote procedure call-end, check slave's task.

R$INFO (bin, ptr, ptr, ptr, bin, (*) bin, bin, (*) bin)
```
call r$info(status, smtp, dbg_info, starting_ecbp,
            n_fil_segnos, fil_segnos, n_link_segnos,
            link_segnos);
```
Returns information about an EPF. Obsolete after 19.3.

R$INIT (fixed, ptr, fixed);
```
call r$init (key, smt_ptr, status);
```
Initialize an EPF's linkage. Obsolete after 19.3; use EPF$INIT.

R$INVK (ptr, [bin, char(*) var, bin, char(*) var, ptr, char(*) var, bin]);

```
call r$invk (smt_ptr [, status, com_args, com_status, com_name,
             vcb_ptr, result, result_max]);
```
Invoke an EPF. Obsolete after 19.3; use EPF$INVK.

R$MAP (fixed, fixed, fixed, fixed);
```
call r$map (key, vmfa_funit, access_rights, status);
```
Map in the procedure image of an EPF. Obsolete after 19.; use EPF$MAP.

R$MYNM (char(32)var);
```
call r$nymn(system_name);
```
Return name of local node.

R$RELC (ptr, ptr) returns (ptr);
```
relocated_ptr = r$relc(erp, smt_ptr);
```
Relocate relative pointers in an EPF. Obsolete after 19.3.

R$RLS(fixed bin(15));
```
call r$rls(xrnode);
```
Decrement slave allocation count. (NPX)

R$RUN (bin, bin, bin, char(*) var, bin, char(*) var, ptr, char(*) var, bin) returns (ptr);
```
smt_ptr = r$run (kry, src_unit, status, com_args, com_status,
                 com_name, vcb_ptr, result, result_max);
```
Runs an EPF. Obsolete after 19.3; use EPF$RUN.

R$SLID (char(32)var, char(8), bin);
```
call r$slid(node_name, slave_id, code);
```
Subroutine to convert node name to slave id if the VC is secured.

R$SLST (struc, bin, bin)
```
call r$slst(slave_list, slave_list_size, error_code);
```
Return a list of a user's active slaves.

R$SYSN (char(32)var, char(8), bin);
```
call r$sysn(slave_id, node_name, code);
```
Subroutine to return the system name for a given *slave_id*.

R$WAIT ((*)bin)
```
call r$wait(buffer);
```
Wait for a call request and initialize user profile.

R$WHER (bin, char(*) var, bin, bin);
```
call r$wher(key, filname, unit, code);
```
Returns the location of a file (local or remote). Obsolete.

R0$ABUF (bin, bin, bin)returns(bit(1))
```
success = r0$abuf(number, avail, code);
```
Allocates "reserved buffers" for R0AM users.

R0$BI (bin, bin(31), ptr, bin, bin)
```
call r0$bi(bi_unit, bi_address, buffer_ptr, bi_fileid, code);
```
Writes before images for R0AM.

R0$CHK (bin) returns(bin)
```
status = r0$chk(key);
```
Checks if R0AM ring zero is initialized.

R0$FBUF (bin, bin, bin, bin, bit(1), bit(1), bin)
```
call r0$fbuf(user, num_free, num_avail, num_freed, release_flag,
      had_none, code);
```
De-allocates "reserved buffers" for R0AM users.

R0$INI (bin, bin)
```
call r0$ini(num_buffers, code);
```
Initializes R0AM ring zero data structues.

R0$PUR (bin, bin)
```
call r0$pur(fileid, code);
```
Purges the specified file from the R0AM buffer pool.

R0$RBUF (bin, bin, ptr, bin, bin);
```
call r0$rbuf(key, priority, buffer_ptr, file_id, code);
```
Releases R0AM buffer(s).

R0$RW (bit(16), bin, bin(31), bin, bin, bin, ptr, ptr, bin)
```
call r0$rw(key, unit, address, length, access, fileid,
                user_buf_ptr, shared_buf_ptr, code)
```
R0AM ring zero buffer manager.

R0$RWM (bit(16), bin, bin(31), bin, bin, ptr, ptr, bin);
```
call r0$rwm(key, unit, xpage_num , access, fileid,
          user_buf_ptr, shared_buf_ptr, code);
```
R0AM ring zero buffer manager.

R0BASE (ptr);
```
call robase(r0_first_ptr);
```
Get a pointer to the first frame on the ring 0 stack.

R3FALT
Ring 3 fault table.

RBK$$ (bin, bin(31), ptr, bin, bin)
```
call rbk$$(unit, logical_block, buffer_ptr, words_read, code);
```
Logical Block i/o block read routine.

RCINF$ (bit(16), ptr, bin)
```
call rcinf$(pdev, info_structure_ptr, code);
```
Return information about disk controller.

RD$CED, RD$CE_DP (bin) II-6-22
```
call rd$ce_dp (program_session_depth)
```
Return to the current depth of the command env. program session.

RDEN$$ (bin, bin, (*)1, 2 bin, 2 char(32), 2 (7)bin, bin, bin, bin(31) or char(32), bin, bin) (svc = 1507) A-9
```
call rden$$(key, file_unit, buffer, buffer_len, rtn_buffer_len,
     file_name, name_len, code)
```
Position and read from a UFD. (Obsolete; use DIR$RD and ENT$RD)

RDLIN$ (bin, char(*), bin, bin) (svc = 1525) II-4-74
```
call rdlin$(file_unit, buffer, buffer_len, code);
```
Read a specified number of characters. buffer_len is size in words.

RDTK$$ (bin, (8)bin, char(*), bin, bin) (svc = 1517) III-3-16

```
call rdtk$$(key, info, token, token_len, code);
```
Parse a command line (Obsolete; use CL$PIX or CL$PAR).

RDTK$P (bin, (8) bin, char(*), bin, char(*) var, bin, bin)
```
call rdtk$p (key, info, buffer, buflen, com_line, com_state,
              code);
```
Parse a command line.

READY$ (bit(16) aligned, fixed bin) III-2-29
```
call ready$ (format_sw, error_code);
```
Print the ready message on the terminal.

RECYCL (svc = 0505)
```
call recycl
```
Pass control to next user.

REMEPF$ (bin, char(*) var, bin) II-5-22
```
call remepf$(key, epf_treename, status);
```
Remove an EPF from a user's environment. Key = KFRC_DEL, KNO_FRC_DEL.

REST$$ ((9)bin, char(32), bin, bin) (svc = 1520) III-5-13
```
call rest$$(r_vector, file_name, file_name_len, code)
```
Read an R-mode runfile.

RESU$$ (char(32), bin) (svc = 1521) III-5-15
```
call resu$$(file_name, file_name_len)
```
Restore and execute an R-mode runfile.

RIPC$C (bin, char(*)var, bin, bin)
```
call ripc$c(uid, node, mbx_id, code);
```
Close a IPC mailbox using the mbx_id specified (Remote; NPX).

RIPC$GU (bin, char(*)var, bin, bin, ptr, bin, bin, bin)
```
call ripc$gu(uid, node, key, mbx_id, buf_ptr, buf_size,
               returned_size, code);
```
Get the desired mailbox user ID specified by key (Remote; NPX).

RIPC$NF (ptr, bin)
```
call ripc$nf(receiver_ptr, code);
```
Interrupt a specified IPC user by mailbox user ID (remote; NPX).

RIPC$O (bin, bin, char(*)var, bin, char(8), char(*)var, bin, bin)
```
call ripc$o(access_key, notification_key, entryname, mbx_uid,
              uusrid, my_node, remote_mbx_id, code);
```
Open an IPC mailbox for specified access using entryname for access control (remote version for NPX).

RIPC$R (bin, char(*)var, bin, ptr, bin, bin, bin, bin)
```
call ripc$r(uid, node, mbx_id, buf_ptr, buf_size, msg_size,
            mbx_send_uid, code);
```
Receive a message from specified IPC mailbox waiting if specified (NPX only).

RIPC$SA (bin, char(*)var, bit(1), bin, ptr, bin, ptr, bin, bin, bin)
```
call ripc$sa(uid, node, nfy_self, mbx_id, msg_ptr, msg_size,
              addr(recvr_list), max_recvr, num_recvr, code);
```
Send a message to any IPC user attach to specified mailbox (NPX only).

RIPC$SB (bin, char(*)var, bit(1), bin, ptr, bin, ptr, bin, bin, bin)
```
call ripc$sb(uid, node, nfy_self, mbx_id, msg_ptr, msg_size,
             addr(recvr_list), max_recvr, num_recvr, code);
```
Send a message to all IPC users attach to specified mailbox (NPX only).

RIPC$SS (bin, char(*)var, bin, bin, ptr, bin, ptr, bin)
```
call ripc$ss(uid, node, mbx_id, mbx_uid, msg_ptr, msg_size,
      receiver_ptr, code);
```
Send a message to a specific IPC user (NPX only).

RIPC$ST (bin, char(*)var, bin, bin, bin, bin)
```
call ripc$st(uid, node, key, mbx_id, value, code);
```
Return various IPC statuses determined by user specified key (NPX only).

RJ$ATT (bin, ptr, ptr, ptr, (2)bin);
```
call rj$att(key, addr(line_info), addr(device_info),
            addr(other_info), errvec);
```
Allow process to attach for line.

RJ$DET (bin, bin, (2)bin)
```
call rj$det(key, line, errvec);
```
Disable the line. Key = 0 if drop DTR.

RJ$INF (bin, ptr, (3)bin)
```
call rj$inf(worker_id, addr(rtn_info), errvec);
```
Return control information from the protocol handler.

RJ$INP (bin, ptr, ptr, bin, bin, (3)bin)
```
call rj$inp(worker_id, addr(rtn_info), addr(buffer), buffer_len,
            msg_type, errvec);
```
Receive a block of data from the RJI.

RJ$MSG (bin, bin, char(80)var)
```
call rj$msg(type, num, string);
```
Return RJE message.

RJ$OUT (bin, ptr, ptr, (2)bin)
```
call rj$out(key, addr(info), addr(buffer), errvec);
```
Queue a block of data for transmission.

RJ$SET (bin, bin, bin, (2)bin)
```
call rj$set(line, key, param, errvec);
```
Send request to protocol handler.

RJDBG (ptr,bin, bin)
```
call rjdbg(com_block_ptr, length, code);
```
Debug gate returns pointer to RJI common blocks for worker RJI.

RJMNIT (bin, ptr, bin)
```
call rjmnit(line, ptr_to_structure, return_code);
```
Ring 0 code required to run the Monit facility.

RJPROC (bin, bin)
```
call rjproc(chap_level, code);
```
Main driver for RJE emulator process.

RLSLV$

```
call rlslv$;
```
Restore a command environment level.

RMSGD$ (char(*), bin, bin, char(*), bin, bin, char(*), bin) III-9-7
```
call rmsgd$(sender_uname, suname_len, sender_unum, system_name,
            system_name_len, time_sent, message, msg_len);
```
Receive a deferred message. Time_sent in minutes past midnight.

ROM$CN (char(32), bin, char(32), bin, bin [, bin])
```
call rom$cn (old_name, old_namel, new_name, new_namel, code,
      open);
```
Changes the name of an RBF file.

ROM$D0 (char(32)var, bin)
```
call rom$d0(obj_name, code);
```
Delete a ROAM file in current directory.

ROM$DL (char (128) var, fixed bin);
```
call rom$dl (obj_path, code);
```
Delete a ROAM file.

RPL$ (char(*) var, char(*) var, char(*) var, bit(1), bin) II-5-24
```
call rpl$(source_path, target_path, rpl_path, no_query, code);
```
Replace one EPF with another.

RPL$CN (char(*) var, char(*) var, bit(1), bin)
```
call rpl$cn(target_tree, rpl_tree, no_query, code);
```
Change the name of an open EPF.

RRECL$ (struc, (3)ptr, (3)bin, bin(31), bit(16), bin)
```
call rrecl$(nch, buf_ptrs, buf_lens, rec_adr, pdev, code);
```
Handle READ requests for ASSIGNED disks.

RSEGAC$ (bin, (2)bin) III-2-13
```
have_access = rsegca$ (segno, access);
```
Function which returns per ring access to the segment if segment is in use.

RTIME$ (1, 2 bin(32), 2 bin)
```
call rtime$(rt_data);
```
Return real-time as 48 bit value in PIC counts.

RTN$DTR3 (ptr, bin)
```
call rtn$dtr3(block_ptr, code);
```
Return storage allocated from DTAR 3 segments through GET$DTR3.

RTNSG$ (bin, bin, bin, bin)
```
call rtnsg$ (segment_number, code [, user, epf_delete_ok]);
```
Returns segments to the system. -1 - all static mode; -2 all static & 6002; -3 all user segs; -4 all user and 6002.

RVON$F (int*2(*), int*2) III-7-28
```
call rvon$f(condition_name, condition_name_len)
```
Revert an on-unit (F77 or FTN).

RVONU$ (char(*)var) III-7-29
```
call rvonu$(condition_name);
```
Revert an on-unit (PL1G, SPL, PLP or PMA)

RVSON$ (entry, fixed bin)
```
call rvson$ (static_on_unit, code);
```
Remove a static on unit.

S$ATRB (1, 2 bin, 2 bit(1), 2 bit(1), 2 bit(1), 2 bin, 2 bit(1), 2 bin, 2 bit(1), 2 bin, 2 bit(1), 2 bin, bin)
```
call s$atrb(attr, status);
```
Sets up default attributes (in memory copy) for the system.

S$ATRG (1, 2 bin, 2 bit(1), 2 bit(1), 2 bit(1), 2 bin, 2 bit(1), 2 bin, 2 bit(1), 2 bin, 2 bit(1), 2 bin, 1, 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(1), bin)
```
call s$atrg (attr, legalr, status);
```
Range checks for attributes.

SAL$SYS1 (bin(31), bin) returns(ptr)
```
block_ptr = sal$sys1(block_size, ercode);
```
System Class Storage Allocator.

SAL_HP(bin, ptr, bin(31), bin)
```
block_ptr = sal_heap(storage_class, hcb_ptr, block_size, ercode);
```
Allocate heap storage.

SANAM$ (char(32) var)
```
call sanam$(system_administrator_id);
```
Returns user id of system administrator.

SATR$$ (bin, char(32), bin, var, bin) (svc = 1510) II-4-76
```
call satr$$(key, object_name, object_name_len, attributes, code);
```
Set or modify a file's attributes. Key = K$PROT, K$DTIM, K$DMPB, K$RWLK, K$SDL.

SAVE$$ ((9) bin, char(32), bin, bin) (svc = 1522) III-5-17
```
call save$$(rmode_vector, file_name, file_name_len, code);
```
Save an R-mode runfile.

SC$CLR (bit(32), bin)
```
call sc$clr(key, code);
```
Disable the signalling of the synchronous conditions.

SC$CLR0 (bit(16), bin)
```
call sc$clr0(key, code);
```
Disable ring 0 synchronous conditions.

SC$PRB - see SEC$PROB.

SC$RD (bit(32), (*)bin)
```
call sc$rd(key, signal_status);
```
Return the value of synchronous condition flags.

SC$RD0 (bit(16), bin)
```
call sc$rd0(key, status);
```
Read the status of ring 0 synchronous conditions.

SC$RST0 (bit(16), bin)
```
call sc$rst0(key, code);
```
Reset ring 0 synchronous condition status.

SC$SET (bit(32), bin)

```
call sc$set(key, code);
```
Enable the signalling of the synchronous conditions.

SC$SET0 (bit(16), bin)
```
call sc$set0(key, code);
```
Enable ring 0 synchronous conditions.

SCH$RD (fixed bin, fixed bin, fixed bin);
```
call sch$rd (key, value, code);
```
Scheduler variable read subroutine.

SCH$ST (fixed bin, fixed bin, fixed bin);
```
call sch$st (key, value, code);
```
Scheduler variable set subroutine.

SEARCH_C, SEARCH_CASELESS_HASH_TABLE$, SRCH$CHT (ptr, char (32) var, bin) returns (ptr);
```
data_address = search_caseless_hash_table$(table_address, name,
                                                    code);
```
Seach a standard hash table regardless of case.

SEARCH_H, SEARCH_HASH_TABLE$, SRCH$HTB (ptr, char (32) var, bin) returns(ptr);
```
data_address = search_hash_table$ (table_address, name, code);
```
Search a standard hash table for *name*.

SEC$AUD
```
call sec$aud
```
AUDITOR gate. Buffer drain process for Security Auditing Facility (privileged).

SEC$MON (ptr, bin)
```
call sec$mon(sec_ptr, code);
```
Start up or change status of SECURITY_MONITOR (privileged).

SEC$PROB, SC$PRB (char(80)var, bin, bin, bin, bin, bin, bin, char(*)var)
```
call sec$prob(program, event_group, event_number, ev_type, code,
            obj_len, obj_type, obj_arg);
```
Record an event in the security audit trail.

SEC$ST (ptr, bin)
```
call sec$st(stat_ptr, code);
```
Ring 0 gate to implement SECURITY_STATUS command (privileged).

SEGAC$ (ptr, fixed bin, fixed bin)
```
call segac$ (segment_pointer, access, code);
```
Changes access of a segment.

SEM$CL (bin, bin) III-8-17
```
call sem$cl(sem_num, code);
```
Close named semaphore.

SEM$DR (bin, bin) III-8-19
```
call sem$dr(sem_num, code);
```
Drain semaphore.

SEM$NF (bin, bin) III-8-21
```
call sem$nf(sem_num, code);
```
Notify semaphore.

SEM$OP (char(32), bin, bin, (*)bin, bin) III-8-23
```
call sem$op(file_name, file_name_len, sem_num, ids, code);
```
Open a semaphore by name.

SEM$OU (bin, bin, (*)bin, bin, bin) III-8-23
```
call sem$ou(file_unit, sem_num, ids, init_val, code);
```
Open a semaphore by file unit.

SEM$ST (bin, bin, bin, bin, bin, (128) bin, bin)
```
call sem$st (key, sem_nmbr_in, sem_nbr_out, wait_count,
             proc_cnt, proc_nbr, status);
```
Return status of a semaphore.

SEM$TN (bin, bin(31), bin(31), bin) III-8-27
```
call sem$tn(sem_num, first_wait_msec, other_wait_msec, code);
```
Set timer for numbered semaphore.

SEM$TS (bin, bin) returns(bin) III-8-29
```
sem_value = sem$ts(sem_num, code);
```
Test counter for semaphore.

SEM$TW (bin, bin, bin) III-8-31
```
call sem$tw(sem_num, time_in_tenths, code);
```
Timed wait for named semaphore. code = 1 -> timed out.

SEM$WT (bin, bin) III-8-33
```
call sem$wt(sem_num, code);
```
Wait on a semaphore.

SET$OR (bin, bin)
```
call set$or(key, code);
```
Set initial attach point (origin).

SETRC$ (bin) III-5-9
```
call setrc$(error_code);
```
Set static mode error code.

SETREG ((4)bin, bin)
```
call setreg(tvec, parflg)
```
Set svec from tvec and parflg.

SFR$SYS1 (ptr, bin)
```
call sfr$sys1(block_ptr, ercode);
```
Frees Space From System Class Storage.

SFR_CFSC(fixed bin, ptr, fixed bin)
```
call sfr_cfsc(storage_class, hcb_ptr, code);
```
Completely free allocated storage for a level.

SFR_HP(fixed bin, ptr, ptr, fixed bin)
```
call sfr_heap(storage_class, hcb_ptr, block_ptr, ercode);
```
Free heap storage.

SGD$DL (bin, bin) II-4-82
```
call sgd$dl (segdir_unit, code);
```
Delete an entry from a segment directory.

SGD$EX (bin, bin, bin)
```
call sgd$ex(unit, type, code);
```
Check the existence of a segment directory entry.

SGD$OP (bin, bin, bin, bin, bin) returns (bin) II-4-84
```
open_unit = sgd$op(key, segunit, unit, type, code);
```
Open a segment directory entry. Key = k$read, k$writ, k$rdwr, k$vmr.

SGDR$$ (bin, bin, bit(16), bit(16), code) (svc = 1512) II-4-86
```
call sgdr$$(key, file_unit, entry1, entry2, code);
```
Position, read or modify a segment directory.

SGNL$F (int*2(*), int*2, int*4, int*2, int*4, int*2, int*2) III-7-30
```
call sgnl$f(condition_name, cname_len, loc(stack_frame), sf_len,
     loc(aux_info), ai_len, flags);
```
Signal a condition from FTN or F77.

SHARE$ (char(32)var, bin, bin, bin)
```
call share$(entryname, segment_no, access, code);
```
Share a segment with specified access and file (privileged).

SHRLIB (bin, (16)bin, bin);
```
rtn_package_num = shrlib(package_mnumber, ecb, code);
```
Install shared library. Restricted.

SH_CMD (char(256)var, bin)
```
call sh_cmd(com_args, com_status);
```
Process the SHUTDN command.

SID$GT(fixed bin(15)) III-2-30
```
call sid$gt(sid);
```
Get the spawner's id in a phantom process.

SIGNL$ (char(*)var, ptr, bin, ptr, bin, bit(16)) III-7-32
```
call signl$(condition, addr(stack_frame), sf_len, addr(aux_info),
     ai_len, action);
```
Signal a condition (PL1G, SPL, PLP or PMA).

SINFO$ (bin, bin, bin, bin)
```
call sinfo$ (action, info_st, echo_st, code);
```
Set and check values of INFO_STATUS for PRIME INFORMATION.

SLAVE((4200)bin, bin)
```
call slave(buf(1), vcix);
```
Slave message handler (NPX).

SLAVER
```
call slaver;
```
Root slave processor (NPX).

SLEEP$ (bin(31)) III-8-39
```
call sleep$(milliseconds);
```
Suspend process.

SLEP$I (bin(31)) III-8-40
```
call slep$i(interval);
```
An interruptable SLEEP$.

SMSG$ (bin, char(*), bin, bin, char(*), bin, char(79), bin, (4+*)bin) III-9-9
```
call smsg$(key, user_name, uname_len, user_num, system_name,
           system_name_len, message, message_len, error_vector);
```
Send a message to another user. Key = 0 - deferred; 1 - immediate.

SMT_QFR (ptr)
```
call smt_qfr(smt_ptr);
```
Unthread an entry from the smt_list for active EPFs.

SNA$CF (bin)
```
call sna$cf(code);
```
Get dynamic Segments for SNA Server Wired and Unwired FS Classes.

SNA$CL (bin, bin)
```
call sna$cl(segnum, code);
```
Get Dynamic segments for LU6.2 free storage class.

SNA$CRFP (bin, bin, bin, bin, bin)
```
free_pool_id = sna$crfp(key, count, size, fs_class, code);
```
Create a free pool (interlude to crfp).

SNACRQ (bin, bin, ptr) returns(ptr)
```
event_qcb_ptr = sna$crq$(fs_class, length, semaphore);
```
Create event queue routine. *Length* must be 2^{k-1}.

SNA$CX (bin, bin)
```
call sna$cx(segnum, code);
```
Get Dynamic segments for PRIME/SNA RJE free storage class.

SNA$DEQA (ptr, bin, ptr, bin) returns(bin)
```
status = sna$deqa(lccb_addr, command, bha_ptr, qflag);
```
Dequeue a command or data block from either queue.

SNA$DEQE (ptr) returns(bin)
```
result = sna$deqe(event_qcb);
```
Dequeue from top of event queue.

SNA$DLQ (ptr, bin, bin)
```
call sna$dlq(qcb_ptr, free_storage_class, code);
```
Routine to delete a queue, returning it's storage to the free list.

SNA$ENQA (ptr, bin, ptr, bin) returns(bin)
```
status = sna$enqa(lccb_ptr, command, bha_ptr, qflag);
```
Enqueue a command or data block on the specified queue.

SNA$FLSH (bin, bin)
```
call sna$flsh(fs_class, code);
```
Flush free storage.

SNA$FREE (ptr)
```
call sna$free(block_ptr);
```
Return a block to its free pool.

SNA$GETB (ptr) returns(ptr)
```
BHA_ptr = sna$getb(fpid);
```
Unconditional get block from free pool.

SNA$GETC (ptr) returns(ptr)
```
BHA_ptr = sna$getc(fpid_threshold);
```
Conditional get block from free pool.

SNA$IADM (bin, bin, bin, char(*), bin, bin, bin)
```
call sna$iadm(log, trace, stats, stats_file, auto_stop,
      stop_time, return_code);
```
Administration control request.

SNA$IAIN (char(*), ptr, ptr, bin)
```
call sna$iain(config_path, config_ptr, rem_sys_ptr, return_code);
```
Create and send a START_3270 LECB to the LU Manager.

SNA$ICLS (bin)
```
call sna$icls(return_code);
```
Close established Mate-Manager connection

SNA$IGD (char(*), bin, bin)
```
call sna$igd(dev_name, time, return_code);
```
Build and send a GET_DEVICE LECB to the LU Manager.

SNA$IGE (ptr, bin, bin, bin)
```
call sna$ige(lecb_ptr, event_type, time_limit, return_code);
```
Retrieve a message for a LU Mate from the LU Manager.

SNA$IOPN (bin)
```
call sna$iopn(code);
```
Open connection between mate and manager.

SNA$IRD (bin(31), bin)
```
call sna$ird(device_id, return_code);
```
Build and send a RETURN_DEVICE LECB to the LU Manager.

SNA$IRS (bin(31), bin)
```
call sna$irs(session_id, return_code);
```
Build and send a RECOVER_SESSION LECB to the LU Manager.

SNA$ISS (bin(31), char(*), ptr, bin)
```
call sna$iss(session_id, suspend_text, ssib_ptr, return_code);
```
Build and send a SUSPEND_SESSION LECB to the LU Manager.

SNA$IST (bin, bin)
```
call sna$ist(status_type, return_code);
```
Build and send a CHECK_STATUS LECB to the LU Manager.

SNA$ISTA (bin, char(*), bin)
```
call sna$ista(type, name, return_code);
```
Administration status request.

SNA$ISTP (bin, char(*), bin, bin)
```
call sna$istp(key, name, type, return_code);
```
Administration stop request.

SNA$IWR (bin(31), bin, ptr, bin, bin, bin)
```
call sna$iwr(sess_id, writeflag, bufptr, datalen,
      vb_versno, return_code);
```
Build and send a WRITE_DATA LECB to the LU Manager.

SNA$LCDL (ptr, bin, bin)
```
call sna$lcdl(lccb_addr, stat1, stat2);
```
Delete a logical connection for the IPQNM routines.

SNA$LCIN (struc)
```
call sna$lcin(lcarray);
```
Initialize a logical connection for the IPQNM routines.

SNA$NTFY (bin)
```
call sna$ntfy(user);
```
Interlude to x$ntfy for SNA.

SNA$PH (char(*), char(*), bin, bin)
```
call sna$ph(service_name, cpl_args, user_no, code);
```
Create an SNA Service for an SNA Administrator.

SNAP$0 (char(32)var) returns(ptr);
```
ecb_ptr = snap$0(name);
```
Snap a dynamic link into ring zero (i.e. a gate).

SNCHE$ (bin, char(32)var, bin, bin)
```
call snche$(keys, name, position, code);
```
Check a system name for validity, return specific errors information.

SNCHK$ (bin, char(32)var) returns(bit(1))
```
name_ok = snchk$(key, name);
```
Check a system name for validity.

SOR0$ (ptr)
```
call sor0$ (cfh_ptr);
```
Invoke the list of ring 0 static on-units.

SOR3$ (ptr)
```
call sor3$ (cfh_ptr);
```
Invoke list of ring 3 static on-units.

SOUR3_ (ptr)
```
call sour3_ (list_ptr);
```
Return pointer to the ring3 static on-units.

SP$MGR (bin, char(32)var, struc, struc, bin(31), bin, bin)
```
call sp$mgr(key, node, queue_entry, template, rqst_no,
     data_file_unit, code);
```
Spool queue manager.

SPAS$$ (char(6), char(6), bin) (svc = 1513) II-2-32
```
call spass(owner_pw, non_owner_pw, code);
```
Set passwords of current UFD.

SPAWN$ (1, 2 bit(13), 2 bit(1), 2 bit(1), 2 bit(1), ptr, char(32) var, bin, char(256) var, bin, bin)
```
call spawn$ (key_structure, addr(spawn_data_struc), filename,
             unit, cpl_args, user_num, status);
```
Spawn a process. Priviledged.

SR$ABSDS, SR$ABS (char(128)var, char(32)var, bin)
```
call sr$absds(rule, list, code);
```
Absolutely disable an optional search rule.

SR$ADDB, SR$ADB (ptr, char(128)var, char(128)var, bin)
```
call sr$addb(arg_list_ptr, old_rule, new_rule, code);
```
Add a search rule to a list before an existing rule.

SR$ADDE, SR$ADE (ptr, char(128)var, char(128)var, bin)
```
call sr$adde(arg_list_ptr, old_rule, new_rule, code);
```
Add a search rule to a list after an existing rule.

SR$CREAT, SR$CRE (char(32)var, ptr, bin)
```
call sr$creat(search_list_name, list_ptr, code);
```
Create a search list by name and open it.

SR$DEL (char(32)var, bin);
```
call sr$del(search_list_name, code);
```
Delete an existing search rule.

SR$DSABL, SR$DSA (char(128)var, char(32)var, bin)
```
call sr$dsabl(rule, list, code);
```
Disable an optional search rule.

SR$ENABL, SR$ENA (char(128)var, char(32)var, bin)
```
call sr$enabl(rule, list, code);
```
Enable an optional search rule.

SR$EXSTR, SR$EXS (char(128)var, bin, char(32)var, bit(1)) returns (bit(1))
```
rule_exists = sr$exstr(rule, req_type, list, case_sensitive);
```
Check a search list for the existence of a specific rule.

SRFR_LS, SRFRL (ptr);
```
call sr$fr_ls(obj_ptr);
```
Free storage used by search rule.

SR$INIT, SR$INI (bin)
```
call sr$init(code);
```
Set search lists for ALL template files in the search rules directory.

SR$LIST, SR$LIS (ptr, bin)
```
call sr$list(arg_output_ptr, code);
```
Return a list of all search list names in this process.

SR$NEXTR, SR$NEX (ptr, ptr, char(128)var, ptr, char(128)var) returns(ptr);
```
next_ptr = sr$nextr(list_ptr, prev_rule_ptr, referencing_dir,
                    locator_ptr, search_place);
```
Fetch the next search rule from a given search list.

SR$OPEN (char(32)var, ptr, bin);
```
call sr$open(search_list_name, list_ptr, code);
```
Find search list specified by name and "open" it. Obsolete.

SR$READ, SR$REA (ptr, ptr, bin);
```
call sr$read(list_ptr, arg_output_ptr, code);
```
Return a list of all search rules of a given search list, printable.

SR$REM (ptr, char(128)var, bin);
```
call sr$rem(arg_list_ptr, the_rule, code);
```
Remove a search rule from a list.

SR$SETL, SR$SET (ptr, ptr);
```
call sr$setl(rule_ptr, locator_ptr);
```
Set the locator value in a given search rule.

SR$SSR (char(128)var, char(32)var, bit(1), char(128)var, bin, bin)
```
call sr$ssr(template_path, list_name, overwrite, error_path,
          error_line, code);
```
Set search rules from a template file.

SR$TEMPL (char(128)var, ptr, char(32)var, bit(1), bit(1), char(128)var, bin, bit(1), bin)
```
call sr$templ(template_file, list_ptr, real_list_name,
              set_up_dflt, dflt_override, error_pathname,
              error_line_number, rec_call, code);
```
Process a search list template file. Obsolete.

SR$UPDT (char(32)var, ptr, bin);
```
call sr$updt(arg_old_list_name, new_list_ptr, code);
```
Install (update) a new copy of a possibly existing search list. Obsolete.

SRCH$$ (bin, char(32), bin, bin, bin, bin) (svc = 1511) II-4-94
```
call srch$$(key, file_name, file_name_len, file_unit, file_type,
          code)
```
Open, close, delete or verify existance of a file. key = (K$READ, K$WRIT, K$RDWR, K$CLOS, K$DELE, K$EXST) + (K$IUFD, K$ISEG, K$CACC, K$GETU) + (K$NSAM, K$NDAM, K$NSGS, K$NSGD)

SRCH$CHT. See SEARCH_CASELESS_HASH_TABLE$.

SRCH$HTB. See SEARCH_HASH_TABLE$.

SRSFX$ (bin, char(*)var, bin, bin, bin, char(32)var, char(32)var, bin, bin) returns(bin(31)) II-4-103
```
char_pos = srsfx$(key, path_name, file_unit, file_type,
     num_suffixes, suffix_list, base_name, suffix_used, code)
```
Search for a file with any set of suffices. Key same as SRCH$$.

SRWREC (bin, bin, bin, bin, bin(31), bin, bin);
```
call srwrec(key, pbav, nwv, nch, rel_addr, device_num, alt_rtn);
```
SVC handler for RREC, WREC SVC.

SS$ERR III-5-11
```
call ss$err;
```
Signal SUBSUS_ERR$ if not interactive.

ST$SGS returns(bin) III-4-26
```
maximum_private_static_segs = st$sgs();
```
Return maximum number of static segments allowed for this user.

STD$CP (char(*) var, bin, bin, 1, 2 bit(1), 2 bit(1), 2 bit(14), ptr, ptr);
```
call std$cp (command_line, status, com_status, flags,
            local_variable_ptr, rtn_function_ptr);
```
Standard command processor.

STKOV$
```
      CALF    STKOV$     /* PMA only
```
Stack overflow handler.

STK_EX (ptr);

```
call stk_ex (full_stack_ptr);
```
Automatic stack extender.

STPNC (ptr, bin, ptr, bin) returns(bin);
```
status = stpnc(error_buffer, err_buf_size, trace_buffer,
          zero_flag);
```
Routine to gather PNC statistics data.

STR$AL(bin, bin, bin, bin) returns(ptr) III-4-5
```
block_ptr = str$al(storage_type, block_size, base_wd, status);
```
Temporary storage allocator. Check for new calling sequence.

STR$AP (bin(31)) returns(ptr) III-4-7
```
block_ptr = str$ap(block_size);
```
Process class storage allocator.

STR$AS(bin(31), bin) returns(ptr) III-4-8
```
block_ptr = str$as(block_size, err_code);
```
Subsystem process class storage allocator.

STR$AU (bin(31)) returns(ptr) III-4-10
```
block_ptr = str$au(block_size);
```
User program class storage allocator.

STR$FP (ptr) III-4-11
```
call str$fp(block_ptr);
```
Frees space from process class storage.

STR$FR(bin, ptr, bin) III-4-12
```
call str$fr(key, block_ptr, status);
```
Free allocated storage (by STR$AL). Check for changed calling sequence.

STR$FS (ptr, bin) III-4-13
```
call str$fs(block_ptr, bin);
```
Frees space from subsystem process class storage.

STR$FU (ptr) III-4-14
```
call str$fu(block_ptr);
```
Frees space from user program class storage.

STRBL (bin, ptr, bin) returns(bin)
```
node_status = strbl(my_node, target_buffer, zero_flag);
```
Routine to move the ring break information to a ring 3 buffer.

STUFF (ptr, bin, char(253) var, bin)
```
call stuff(addr(msg), type, str, str_len);
```
Put subfield data into spare data field of a message

SUSR$ returns(bit(1)) III-2-31
```
is_user_1 = susr$();
```
Returns whether or not caller is user 1.

SW$INT (bin, 1, 2 bin, 2 bit(16), 2 bit(16), 1, 2 bin, 2 bit(16), 2 bit(16), bin [, bin])[returns(bin)/* ring 0];
```
call sw$int(key, selection, value, ercode [, outer_ring]);
already_deferred = sw$int(key, selection, value, ercode
                          [, outer_ring]);
```

Software interrupt enable control module. Key = KON, KOFF, K$RDON, K$RDOF, K$READ, K$ALON, K$ALOF, K$RAON, KR$RAOF, K$RDAL.

SW$ON (1, 2 fixed bin, 2 bit(16), 2 bit(16));
```
call sw$on(selection);
```
Turns on the specified software interrupts for ring 3.

SW$RAOF, SW$RAO (1, 2 fixed bin, 2 bit(16), 2 bit(16));
```
call sw$raof(value);
```
Reads and then turn off all present interrupts for ring 3.

SW$RST
```
call sw$rst;
```
Reset ring 0 software interrupt enable mechanism.

SWFBK_
```
call swfbk_;
```
Invoke QUIT condition in ring 3 with pb backup.

SWFIM_
```
call swfim_;
```
Invoke QUIT condition in ring 3.

SYN$CHCK (bin, bin, bin, bin)
```
call syn$chck(SyncNum, NumberOfNotices, NumberOfWaiters, Status);
```
Returns the number of outstanding notices or number of waiters on an event synchronizer.

SYN$CREA (bin, bin, bin)
```
call syn$crea(InitialNoticeCount, SyncNum, Status);
```
Create an event synchronizer for this server.

SYN$DEST (bin, bin)
```
call syn$dest(SyncNum, Status);
```
Destroy a synchronizer belonging to this server.

SYN$GCHK (bin, bin, bin, bin, bin)
```
call syn$gchk(GroupNum, PriorityLevel, NumberOfNotices,
              NumberOfWaiters, Status);
```
Returns the number of outstanding notices or number of waiters on an event group.

SYN$GCRE (bin, bin, bin)
```
call syn$gcre(PriorityLevels, GroupNum, Status);
```
Create an event group for this server.

SYN$GDST (bin, bin)
```
call syn$gdst(GroupNum, Status);
```
Destroy the event group after first removing any event synchronizers from the group.

SYN$GLST (bin, (*)bin, bin, bin)
```
call syn$glst(GroupListSize, GroupList, GroupCount, Status);
```
Returns the numbers of the event groups belonging to this server (process).

SYN$GRTR (bin, bin, bin, bin, ptr, bin)
```
call syn$grtr(GroupNum, PriorityLevel, WhatHappened,
              SyncNum, ForClientUse, Status);
```
Retrieve a notice from an event group if at least one has been posted.

SYN$GTWT (bin, bin(31), bin, bin, ptr, bin)
```
call syn$gtwt(GroupNum, WaitTime, WhatHappened,
              SyncNum, ForClientUse, Status);
```
Timed wait for a notice to be posted to an event group.

SYN$GWT (bin, bin, ptr, bin)
```
call syn$gwt(GroupNum, SyncNum, ForClientUse, Status);
```
Wait on an event group until a notice has been posted to it.

SYN$INFO (bin, ptr, bin)
```
call syn$info(SyncNum, SyncInfoPtr, Status);
```
Returns information about an event synchronizer.

SYN$LIST (bin, (*)bin, bin, bin)
```
call syn$list(SyncListSize, SyncList, SyncCount, Status);
```
List event synchronizers belonging to this server (process).

SYN$LSIG (bin, bin, (*)bin, bin, bin)
```
call syn$lsig(GroupNum, SyncListSize, SyncList, SyncCount,
      Status);
```
Returns a list of the synchronizers currently in an event group.

SYN$MVTO (bin, bin, bin, ptr, bin)
```
call syn$mvto(GroupNum, SyncNum, PriorityLevel, ForClientUse,
      Status);
```
Move an event synchronizer into an already existing event group.

SYN$POST (bin, bin)
```
call syn$post(SyncNum, Status);
```
Post a notice to an event synchronizer.

SYN$REMV (bin, bin)
```
call syn$remv(SyncNum, Status);
```
Remove a synchoronizer from whatever group it is in.

SYN$RTRV (bin, bin, bin)
```
call syn$rtrv(SyncNum, WhatHappened, Status);
```
Retrieve a notice on an event synchronizer if at least one has been posted.

SYN$TMWT (bin, bin(31), bin, bin)
```
call syn$tmwt(SyncNum, WaitTime, WhatHappened, Status);
```
Timed wait on an event synchronizer.

SYN$WAIT (bin, bin)
```
call syn$wait(SyncNum, Status);
```
Wait on an event synchronizer until a notice is returned.

T$AMLC (bin, ptr, bin, bin, (2)bin [, bin, bin]) (svc = 0513) IV-8-23
```
call t$amlc(line, addr(buffer), buf_char_count, key, status_vec,
      [buf_start_char, code]);
```
Communicate with AMLC driver. See Subroutine Ref Guide for keys.

T$CMPC (bin, ptr, bin, bin, (2)bin) (svc = 0512) IV-7-28
```
call t$cmpc(unit, addr(buffer), num_words, inst, status);
```
Input from MPC card reader.

T$GPPI (bin, bin, bin, bin, (4096)bin, bin)

```
call t$gppi(unit, key, data1, data2, array, code);
```
General purpose parallel interface routine.

T$GS (bin, bin, bin, bin/ptr, bin, bin)
```
call t$gs(unit, key, function, buffer, buf_len, non_std_code);
```
Driver for Vector General graphics terminals.

T$LMPC (bin, ptr, bin, bin, (2)bin) (svc = 0511) IV-7-6
```
call t$lmpc(unit, addr(buffer), num_words, inst, status);
```
Move data to MPC line printer.

T$MG (bin, bin, bin, ptr, bin, (3)bin)
```
call t$mg(unit, key, aux_data, addr(buffer), buf_len, stat_vec);
```
Driver for SOC-Megraphic 7000 interface.

T$MT (bin, ptr, bin, bin, (2)bin) (svc = 0510) IV-7-37
```
call t$mt(unit, addr(buffer), num_words, inst, status);
```
Raw data mover for tape drive.

T$PMPC (bin, ptr, bin, bin, (2)bin) (svc = 0515) IV-7-34
```
call t$pmcp(unit, addr(buffer), num_words, inst, status);
```
Raw data mover for card reader.

T$SLC1 (bin, bin, ptr, bin);
```
call t$slc1(key, line, addr(block), block_len);
```
Control block interpreter for HSSMLC, MDLC, and LYNX controllers.

T$VG (bin, ptr, bin, bin, (2) bin) (svc = 0514) IV-7-16
```
call t$vg(phys_unit, addr(buffer), num_words, inst, status)
```
Interface to Versatec printer.

TA$ (char(*) var, bin, bin, char(32), bin, bit(16), bin) returns(bin);
```
outc = ta$(line, state, key, entry_name, entry_name_length,
          attach_switch, code);
```
Attach to directory. Obsolete; use AT$.

TERM$I (bin)
```
call term$i(key)
```
SET/reset terminal parameters for use with the INFORMATION product. Key = 1 (enter INFO), 2 (leave INFO). Obsolete.

TEXTO$ (char(32), bin, bin, bin) III-10-15
```
call texto$(file_name, file_name_len, actual_len, text_ok)
```
Check a filename for valid format. Text_ok is a fortran logical.

TI$MSG (bin, bin, bin, bin) III-2-32
```
call ti$msg (user, connect_minutes, cpu_seconds, io_seconds);
```
Print accumulated time message (for logout message).

TIMDAT (1, 2 (3)char(2), 2 (9)bin, 2 char(32), bin) (svc = 0502) III-2-34
```
call timdat(time_date_stuff, time_date_stuff_len)
```
Return system and user information.

TL$SGS returns(bin) III-4-27
```
max_segno_in_dtar2 = tl$sgs();
```
Return highest segment number allowed in dtar 2.

TM3270 ((3) bin, (3) bin, bin)
```
call tm3270 (delays, polling_periods, code);
```
Initiate the Traffid Manager for IBM 3270 terminals. (DPTX)

TMR$CANL (bin, bit(1), bin)
```
PROCEDURE tmr$canl(Timer: TimerNumber;
    VAR Expired: plp_boolean; VAR Status: TimerStatusCode);
```
Cancel the pending timer identified by Timer. (Timer)

TMR$CREA (bin, bin, bin)
```
PROCEDURE tmr$crea(WhichKind: KindOfTimer;
      VAR NewTimer: TimerNumber; VAR Status: TimerStatusCode);
```
Create a timer private to the calling server. (Timer)

TMR$DEST (bin, bit(1), bin)
```
PROCEDURE tmr$dest(Timer: TimerNumber;
 VAR Expired: plp_boolean; VAR Status: TimerStatusCode);
```
Destroy a timer. (Timer)

TMR$GINF, TMR$NF (struc)
```
PROCEDURE tmr$ginf(VAR CurrentTimeInfo: PermTimeInfo);
```
Returns the PermTimeInfo. (Timer)

TMR$GTIM, TMR$GT (struc)
```
PROCEDURE tmr$gtim(VAR CurrentTime: AbsoluteTime);
```
SystemTime is returned in Universal Time. (Timer)

TMR$GTMR (bin, struc, bin)
```
PROCEDURE tmr$gtmr(Timer: TimerNumber; VAR Info: TimerInfo;
                   VAR Status: TimerStatusCode);
```
Returns information on the timer specified. (Timer)

TMR$LIST(bin, (0:15)bin, bin, bin)
```
PROCEDURE tmr$list(TimerListSize: SHORT_CARDINAL;
   VAR TimerList: TimerListArray;
   VAR NumberOfTimers: SHORT_CARDINAL;
   Status: TimerStatusCode);
```
Returns the timer numbers belonging to this server in TimerList. (Timer)

TMR$LOCALCONVERT (struc, struc)
```
PROCEDURE TMR$LocalConvert(LocalTime: LocTime;
     VAR UnivTime: CARDINAL_64);
```
Converts the local time provided to Universal Time.

TMR$NF. See TMR$GINF.

TMR$PROC
```
call tmr$proc;
```
The timer process. (TimerMDK)

TMR$SABS (bin, bin, struc, bit(1), bin)
```
PROCEDURE tmr$sabs(Timer: TimerNumber; Sync: EventSyncNumber;
      ExpirationTime: AbsoluteTime; VAR Expired: plp_boolean;
      VAR Status: TimerStatusCode);
```
Sets the timer to expire at the absolute time specified. (Timer)

TMR$SINT (bin, bin, bin(31), bit(1), bin)

```
PROCEDURE tmr$sint(Timer: TimerNumber;
   Sync: EventSyncNumber; ExpirationInterval: IntervalTime;
   VAR Expired: plp_boolean; VAR Status: TimerStatusCode);
```
Sets the timer to expire after the interval specified. (Timer)

TMR$SREP (bin, bin, bin(31), bin)
```
PROCEDURE tmr$srep(Timer: TimerNumber; Sync: EventSyncNumber;
 ExpirationIntervals: IntervalTime; VAR Status: TimerStatusCode);
```
Sets a repetitive timer to expire every ExpirationIntervals. (Timer)

TMR$STI (char(*)var, bin)
```
PROCEDURE TMR$STI (xline: ComLineString;
                   VAR status: TimerStatusCode);
```
Implements the SET_TIME_INFO operator command. (TimerMDK)

TMR$STIM (struc, bin);
```
PROCEDURE tmr$stim(NewSysTime: AbsoluteTime;
              VAR Status: TimerStatusCode);
```
Sets the system time. Changes will not affect interval timers. (Timer)

TMR$UNIVCONVERT (struc, struc)
```
PROCEDURE TMR$UnivConvert(UnivTime: CARDINAL_64;
     VAR LocalTime: LocTime);
```
Converts the Universal time value, UnivTime, to local time in LocTime format. (TimeLibrary)

TNCHK$ (bin, char(128)var) returns(bit(1)) II-4-109
```
path_name_ok = tnchk$(key, path_name)
```
Check pathname for valid format. Key = K$UPRC, K$WLDC, K$NULL.

TNOU (char(*), bin) (svc = 0702) III-3-40
```
call tnou(string, string_size);
```
Output characters and newline to terminal.

TNOUA (char(*), bin) (svc = 0703) III-3-41
```
call tnoua(string, string_size);
```
Output characters to terminal.

TP$CON (bin)
```
call tp$con(code);
```
Reconnect user process to a terminal line.

TP$DIS (bin)
```
call tp$dis(code);
```
Disconnect the terminal from this process making it assignable.

TRNRCV (bin, bin, bin, bin, bin);
```
call trnrcv(key, vcix, mitlen, buffer, code);
```
Transmits and receives messages between master and slave processes.

TSRC$$ (bin, char(128), bin, bin, bin, bin) II-A-17
```
call trsc$$(key, path_name, file_unit, chr_pos, type, code)
```
Open, close, delete or find file. (Obsolete; use SRSFX$)

TTY$CNT returns(bin)
```
num_chars = tty$cnt();
```
Ring 3 interlude for Tf$cnt - returns # of characters in user's IRB.

TTY$IN returns(bit(1)) III-3-63
```
characters_waiting = tty$in();
```
Check if there are any characters in the tty input buffer for user.

TTY$RS (bit(16), bin) III-3-65
```
call tty$rs(key, code);
```
Routine to clear a process's I/O buffers. Key: bit 1 - output buffer; bit 2 - input buffer.

U$TERM (bit(1)) returns(bit(1))
```
previous_state = u$term(enable_terminal_output);
```
Enable/disable terminal output from a child process.

UID$BT (bit (48) aligned) III-6-39
```
call uid$bt (unique_bit_string);
```
Return unique bit string.

UID$CH (bit (48) aligned, char (13)) III-6-40
```
call uid$ch (unique_bit_string, character_string);
```
Return a unique character sequence based on a unique bit string.

UNITS$ (bin, bin) II-4-112
```
call units$(num_unit, max_unit);
```
Get the current unit number bounds.

UNLKF$
```
call unlkf$;
```
Unlock all N1 locks owned by the calling process.

UNO$GT((128) bin, bin, bin) III-2-36
```
call uno$gt(ids, lenids, numids);
```
Return all ids for the current user.

UNWND$ (label) returns (bit(1))
```
unwind_ok = unwnd$(target_of_nl_goto);
```
Prepare the stack for nonlocal-goto-induced unwinding.

UPDATE (bin, bin)
```
call update(key, 0);
```
Update current UFD (Primos II). Key = 1.

USER$ (bin, bin) III-2-15
```
call user$(current_user_num, num_users);
```
Return process number and total user count.

USNMT$ (bit(16), char(256) var, bin)
```
call usnmt$ (no_msgs, user_unassign_cmd_line, return_status);
```
Unassigns magnetic tape drive. (DOSSUB only)

USRAS$ (char(256) var, fixed bin)
```
call usras$ (com_args, com_status);
```
Process USRASR command.

UTYPE$ (bin) III-2-38
```
call utype$(user_type);
```
Return type of current process.

VALID$ (char(32)var, bin) returns(bit(1)) III-2-41

```
id_found = valid$ (name, code);
```
Validates name passed vs. user's composite ID (user ID plus groups).

VINIT$ (bin, bin, (*)bin, bin, (*)bin, (*)bin, (*)bin, bin);
```
call vinit$ (key, unit, segment_numbers, number_of_segments,
             window, access, segment_length, code);
```
Map in a DAM file using initial VMFA.

WARM$I (ptr, bin)
```
call warm$i(data_ptr, code);
```
Handle warm start setup for INFORMATION.

WBK$$ (bin, bin(31), ptr, bin, bin)
```
call wbk$$(unit, logical_block, buffer_ptr, num_words, code);
```
Logical Block i/o block write routine.

WILD$ (char(32)var, char(32)var, bin) returns (bit(1)aligned) II-4-113
```
match = wild$ (wildcard_name, entry_name, code);
```
Compare *entry_name* against *wildcard_name* for containment.

WRECL$ (1, 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(1), 2 bit(6), (3)ptr, (3)bin, bin(31), bit(16), bin);
```
call wrecl$(nch, buf_ptrs, buf_lens, rec_adr, pdev, code);
```
Write record to assigned disk.

WRL$ (ptr, fixed bin);
```
call wrl$ (list_ptr, entries);
```
Return a pointer to the caller's list of static on-units.

WTLIN$ (bin, char(*), bin, bin) (svc = 1526) II-4-115
```
call wtlin$(file_unit, buffer, buffer_len, code)
```
Write a given number of ASCII chars. Buffer_len is in words.

X$ASGN (bin, bin, bin) P-14-4
```
call x$asgn(subprocess, count, code);
```
Assign primitive for general users.

X$CLRA P-14-23
```
call x$clra;
```
Routine that can be used to clear all connections a user owns.

X$FRPL (bin, ptr, bin)
```
call x$frpl(version, buffer_ptr, status_code);
```
Gathers size information for all Primenet free pools.

X$GVVC (bin, bin, bin) P-14-25
```
call x$gvvc(vcid, user, code);
```
Pass control of a virtual circuit to another user

X$LHCS (bin, ptr, bin)
```
call x$lhcs(version, buffer_ptr, status_code);
```
Gathers traffic information for Ethernet Primenet.

X$LTRC (bin, ptr, bin)
```
call x$ltrc(version, buffer_ptr, status_code);
```
Gathers traffic information for Ethernet Primenet nodes.

X$PRTQ (bin, ptr, bin)
```
call x$prtq(version, buffer_ptr, status_code);
```
Gathers length information concering the Primenet protocol queues.

X$RCV (bin, char(*), bin, bin) P-14-18
```
call x$rcv(vcid, buffer, buffer_size, state);
```
Provide receive buffers for X.25 packet input.

X$RSET (bin, bin, bin)
```
call x$rset(vcid, why, status)
```
Allow a user to cause a reset on one of his virtual circuits.

X$RT (bin, bin, char(32)var, char(32)var, bin, bin, bin, char(32)var, bin)
```
call x$rt(key, option_key, src_item, dest_item, path,
          ret_rt_class, ret_path, ret_item, code);
```
Ring 0 support for route through configuration information. Key = xknam, xkadr (only for xr$name). Option_key = xr$me, xr$name, xr$path, xr$scradr.

X$RTI returns(bit(1))
```
made_it = x$rti();
```
Set up this process to run as the route-through server.

X$STAT (bin, bin, (255)bin, bin, (255)bin, bin, bin, bin) P-14-29
```
call x$stat(key, narray_or_vcn, array1, ar1_len, array2, ar2_len,
            code, time);
```
Routine to return status information to user space.

X$TRAN (bin, bin, char(*), bin, bin) P-14-16
```
call x$tran(vcid, buffer_type, buffer, buffer_count, state);
```
X.25 transmit primitive.

X$UASN (bin) P-14-23
```
call x$uasn(subprocess);
```
Unassign primitive for general users.

X$VCLT (bin, bin, 1, 2 bin, 2 bin, 2 bin, 2 (*)bin, bin)
```
call x$vclt(user_id, vc_list_size, vc_list, error_code);
```
Return a list of a user's active VCs.

X$WAIT (bin) returns(bin) P-14-24
```
timer_expired = x$wait(tenths);
```
Timed wait for network event.

XLACPT (bin, bin, char(*), bin, char(4), bin, char(*), bin, bin) P-14-14
```
call xlacpt(key, vcid, fclty, fcltyn, prid, pridn,
            udata, udatan, state);
```
Accept pending x.25 connection. Key = 1, xkmdb, xkfct, xk$svc.

XLASGN (bin, char(16)var, char(16)var, char(4)var, char(32)var, bin, bin, bin, char(16)var, char(16)var, bin, bin)
```
call xlasgn(key, adr, subadr, prid, udata, port, gfi, vcn,
      src_adr, src_sadr, count, code);
```
Extended declaration of interest in incoming calls

XLCLR (bin, bin, bin, char(*), bin, char(*), bin, bin, bin)
```
call xlclr(key, vcid, why, fclty, fcltyn,
           udata, udatan, xtra3, state);
```
Clear an X.25 virtual circuit. Key = 1.

XLCLRA (bin)
```
call xlclra(key);
```
Clear either 'USER' or 'SYSTEM' VCs. Key = xkuvc, xksvc.

XLCONN (bin, bin, bin, char(*), bin, char(*), bin, char(4), bin, char(*), bin, (2)bin [, char(*) bin, bin, bin, char(*), bin, char(*), bin]) P-14-6
```
call xlconn(key, vcid, port, addr, addr_len, fclty, fclty_size,
            pr_id, pr_id_size, udata, udata_size, state[,
            rtn_udata, rtn_udata_len, r_u_rtn_cnt, more_key,
            src_addr, src_addr_len, pnet, pnet_len);
```
Request a virtual circuit connection. Key = (xkany, xkrte, xksyn, xkrng, xk$pdn) + (xk$fct, xk$mbd) + (xk$adr, xk$nam) + [xk$fam, xk$rlg] + [xk$rtd]. (X.25)

XLGA$ (bin, bin, bin, bin, bin, bin, char(15), bin, bin, char(15), bin, bin, char(4), bin, bin, char(*), bin, bin, bin)
```
call xlga$(key, vcid, port, gfi, vcn, cmnd,
           faddr, faddrm, faddrl, taddr, taddrm, taddrl,
           fcty, fctym, fctyl, prid, pridm, pridl,
           udata, udatam, udatal, state);
```
Get all of the fields in a call accept packet. Key is ignored.

XLGC$ (bin, bin, bin, bin, bin, bin, (8)bin, bin, bin, (8)bin, bin, bin, (32)bin, bin, bin, (2)bin, bin, bin, (63)bin, bin, bin, (2)bin)
```
call xlgc$(key, vcid, port, gfi, vcn, cmnd, faddr, faddrm,
      faddrl, taddr, taddrm, taddrl, fcty, fctym, fctyl,
      prid, pridm, pridl, udata, udatam, udatal, state);
```
Get all of the fields in a connect request packet. Key = 0, xk$reg.

XLGCON (bin, bin, bin, char(*), bin, bin, char(*), bin, bin, char(4), bin, bin, char(*), bin, bin, bin) P-14-11
```
call xlgcon(key, vcid, port, sadr, sadrn, sadrbc,
            fclty, fclyn, fclybc, prid, pridn, pridbc,
            udata, udatn, udatbc, state);
```
Get information about pending call requests. Key = xknam, xkadr.

XLGI$ (bin, bin, bin, bin, bin, bin, (*)bin, bin, bin, (*)bin, bin, bin, (*)bin, bin, bin, (*)bin, bin, bin, (*)bin, bin, bin, bin)
```
call xlgi$(key, vcid, port, GFI, VCN, command, calling_addr,
  calling_addr_len, calling_addr_rtn_len, called_addr,
  called_addr_len, called_addr_rtn_len, facilities,
  facil_len, facil_rtn_len, proto_id, prid_len,
  prid_rtn_len, user_data, user_data_len, user_data_rtn_len,
  result_state)
```
Get all of the fields in an extended CLEAR INDICATION.

XLGVVC (bin, bin, bin, (8)bin, bin, bin, (8)bin, bin, (8)bin, bin, (32)bin, bin, (2)bin, bin, (62)bin, bin, bin) P-14-25
```
call xlgvvc(key, vcid, user, adr, adrn, port, fadr, fadrn, tadr,
            tadrn, fcty, fctyn, prid, pridn, udata, udatan, code)
```
Pass control of a virtual circuit to another user. key = xkusr, xkprt, xk$adr.

XLUASN (bin, char(16)var, char(16)var, char(4)var, char(32)var, bin, bin, bin, char(16)var, char(16)var, bin)
```
call xluasn(key, adr, sadr, prid, udata, port, gfi, vcn, src_adr,
            src_sadr, code);
```
Unassign an extended declaration.

XMTRCV (bin, char(8), bin(31), (*)bin, bin, bin)
```
call xmtrcv(caller_key, slave_id, xmit_len, buffer, time, rcode);
```
Transmits and receives messages to and from slaves in one operation under quit protection.

6.2. Spool library

Spool routines are in the shared spool library or (at 21.0) SPOOL_LIBRARY.RUN.

SPOOL$ (bin, char(*), bin, (29)bin, (*)bin, bin, bin)
```
call spool$(key, filename, namelen, info, buffer, buflen, code);
```
Insert a file in spooler queue.

6.3. Application Library

Binary routines are in NVAPPLB.BIN; dynts in VAPPLB.BIN; runtime library is APPLICATIONS_LIBRARY.RUN. Mainly used in FORTRAN programs. It is recommended that the appropriate system routines be used instead of application library routines where possible. R-mode binaries are found in APPLIB.BIN.

CASE$A (int*2, char*, int*2) returns logical
```
[valid_length =] case$a (key, string, length)
```
Converts case from lower to upper or upper to lower. Key = A$FUPP, A$FLOW.

CLOS$A (int*2) returns logical
```
[closed_ok =] clos$a (file_unit)
```
Closes the file open on *file_unit.*

CMDL$A (int*2, int*2(*), int*2, char*, int*2, int*2, int*4, int*2) returns logical
```
[command_ok =] cmdl$a (key, kwlist, kwindx, optbf, bfln, opt,
                       val, kwinfo)
```
Parses a command line.

CNVA$A (int*2, char*, int*2, int*4) returns logical
```
[conversion_ok =] cnva$a (numkey, name, namlen, val)
```
Convert an ASCII digit string to its bnary value for octal, decimal and hex. numkey = (ADEC, ABIN, AOCT, AHEX)

CNVB$A (int*2, char*, int*2, int*4) returns int*2
```
[int_2_val =] cnvb$a (numkey, val, name, namlen)
```
Convert a bindary number to an ASCI string.

CSTR$A (char*, int*2, char*, int*2) returns logical
```
strings_equal = cstr$a (astring, alen, bstring, blen)
```
Compares two strings for equality.

CSUB$A (char*, int*2, int*2, int*2, char*, int*2, int*2, int*2) returns logical
```
substrings_match = csub$a (a, alen, afc, alc, b, blen, bfc, blc)
```
Compare two substrings for equality.

CTIM$A (int*4) returns real*8
```
seconds = ctim$a (cputim_in_centiseconds)
```
Returns elapsed CPU time.

DATE$A (char*) returns real*8
```
mm_dd_yy = date$a (date)
```
Returns today's date.

DELE$A (char*, int*2) returns logical
```
success = dele$a (name, namlen)
```
Delete a file.

DOFY$A (char*) returns real*8
```
yr_ddd = dofy$a (dofy)
```
Returns the day of the year (DDD).

DTIM$A (int*4) returns real*8
```
time_in_seconds = dtim$a (disktim)
```
Returns disk time in centiseconds.

EDAT$A (char*) returns real*8
```
dd_mm_YY = edat$a (edate)
```
Returns date in European (military) form.

ENCD$A (char*, int*2, int*2, real*8) returns logical
```
success = encd$a (array, width, dec, val)
```
Encodes a real number into a string in F*width.dec* format.

EXST$A (char*, int*2) returns logical
```
exists = exst$a (name, namlen)
```
Indicates whether a file exists.

FDAT$A (int*2, char*) returns real*8
```
mm_dd_yy = fdat$a (datemod, date)
```
Converts file date to string.

FEDT$A (int*2, char*) returns real*8
```
mm_dd_yy = edt$a (datemod, date)
```
Converts file date to string, European style.

FILL$A (char*, int*2, char*1)
```
CALL FILL$A (name, namlen, char)
```
Fill a buffer with char.

FSUB$A (char*, int*2, int*2, int*2, char*1) returns logical
```
success = fsub$a (string, len, fchar, lchar, filchar)
```
Fills a substring with a character.

FTIM$A (int*2, char*) returns real
```
realtimemod = ftim$a (timemod, time)
```
Converts a file time to string or real.

GCHR$A (char*, int*2) returns int
```
character = gchr$a (farray, fchar)
```
Extracts a character from a string.

GEND$A (int*2) returns logical
```
success = gend$a (unit)
```
Position a file to EOF.

JSTR$A (int*2, char*, int*2) returns logical
```
success = jstr$a (key, string, len)
```
Justify a string (left, right, or center). Key = (A$RGHT, A$LEFT, A$CNTR).

LSTR$A (char*, int*2, char*, int*2, int*2, int*2) returns logical

```
found = lstr$a (a, alen, b, blen, fcp, lcp)
```
Locates one string within another.

LSUB$A (char*, int*2, int*2, int*2,char*, int*2, int*2, int*2, int*2) returns logical
```
found = lsub$a (a, alen, afc, alc, b, blen, bfc, blc, fcp, lcp)
```
Locates one substring within another.

LTOK$A (char*, int*2, int*2, char*, int*2, int*2, int*2, int*2, int*2(*)) returns logical
```
found = ltok$$a (a, alen, afc, alc, b, blen, bfc, blc, hcp, lcp,
    ndel)
```
Locates character substrings as tokens. (V-mode only)

LWC$$A (char*, int*2, int*2)
```
call lwc$$a(string, position, length)
```
Translates a substring to lowercase. (V-mode only)

MCHR$A (char*, int*2, char*, int*2) returns int
```
char_moved = mchr$a (tarray, tchar, farray, fchar)
```
Moves a character from one array to another array.

MOVE$A (char*, int*2, char*, int*2, int*2)
```
call move$a (fstr, fpos, tstr, tpos, len)
```
Move a string to another. (V-mode only)

MSTR$A (char*, int*2, char*, int*2) returns int
```
number_moved = mstr$a (astring, alen, bstring, blen)
```
Move a string to another.

MSUB$A (char*, int*2, int*2, int*2, char*, int*2, int*2, int*2) returns int
```
number_moved = msub$a (a, alen, afc, alc, b, blen, bfc, blc)
```
Move a substring to another.

NLEN$A (char*, int*2) returns int*2
```
length = nlen$a (name, namlen)
```
Returns actual length of the string.

OPEN$A (int*2, char*, int*2, int*2) returns logical
```
success = open$a (opnkey+typkey+untkey, name, namlen, unit)
```
Opens a file.

OPVP$A (char*, int*2, int*2, char*, int*2, int*2, int*2, int*2) returns logical
```
success = opvp$a (msg, msglen, opnkey+typkey+untkey, name,
  namelen, unit, wtime, retrys)
```
Prompts a user for a file name and opens it with retries and verification.

OPNP$A (char*, int*2, int*2, char*, int*2, int*2) returns logical
```
success = opnp$a (msg, msglen, opnkey+typkey+untkey,
          name, namlen, unit)
```
Prompts user for a file name and opens it.

OPNV$A (int*2, char*, int*2, int*2, int*2, int*2) returns logical
```
success = opnv$a (opnkey+typkey+untkey, name, namlen,
          unit, verkey, wtime, retrys)
```
Opens a file with retries and verification.

POSN$A (int*2, int*2, int*4) returns logical
```
success = posn$a (poskey, unit, pos)
```
Positions in a file.

RAND$A (int*4) returns real
```
random_number = rand$a (seed)
```
Generates a pseudo-random number.

RNAM$A (char*, int*2, int*2, char*, int*2) returns logical
```
success = rnam$a (msg, msglen, namkey, name, namlen)
```
Prompts user for a name.

RNDI$A (int*4) real
```
random_number = rndi$a (seed)
```
Initializes the random number generator.

RNUM$A (char*, int*2, int*2, int*4) returns logical
```
success = rnum$a (msg, msglen, numkey, val)
```
Prompts user for a number and returns it.

RPOS$A (int*2, int*4) returns logical
```
success = rpos$a (unit, pos)
```
Returns the absolute position of a file.

RSTR$A (char*, int*2, int*2) returns logical
```
success = rstr$a (string, len, cnt)
```
Rotates a string.

RSUB$A (char*, int*2, int*2, int*2, int*2) returns logical
```
sucess = rsub$a (string, len, fchar, lchar, cnt)
```
Rotates a substring.

RWND$A (int*2) returns logical
```
success = rwnd$a (unit)
```
Rewinds a file.

SSTR$A (char*, int*2, int*2, int*2)returns logical
```
success = sstr$a (string, len, cnt, filchr)
```
Shifts a string.

SSUB$A (char*, int*2, int*2, int*2, int*2, int*2) returns logical
```
success = ssub$a (string, len, fchar, lchar, cnt, filchar)
```
Shifts a substring.

TEMP$A (int*2, char*, int*2, int*2) returns logical
```
success = temp$a (typkey+untkey, name, namlen, unit)
```
Creates a temporary file and opens it.

TIME$A (char*) returns real
```
real time = time$a (time)
```
Returns the time of day.

TREE$A (char*, int*2, int*2, int*2) returns logical
```
is_a_treename = tree$a (name, namlen, fstart, flen)
```
Checks a treename for validity.

TRNC$A (int*2) returns logical
```
success = trnc$a (unit)
```
Truncates a file at its current position.

TSCN$A (int*2, int*2(*), int*2(*), int*2, int*2, int*2, int*2, int*2) returns logical

```
success = tscn$a (key, units, entry, maxsiz, entsiz, maxlev,
          lev, code)
```
Scans a tree.

TYPE$A (int*2, char*, int*2) returns logical
```
is_valid = type$a (key, string, len)
```
Checks a string for being a valid type.

UNIT$A (int*2) returns logical
```
unit_open = unit$a (unit)
```
Determines if a file unit is open.

UPC$$A (char*, int*2, int*2)
```
call upc$$a (string, postition, length)
```
Translates a substring to uppercase. (V-mode only)

YSNO$A (char*, int*2, int*2) returns logical
```
answer_is_yes = ysno$a (msg, msglen, defkey)
```
Prompts a user and returns true if answer is yes.

6.4. DBMS routines

PRISAM routines.

Z$ABRT (bin, bin)
```
call z$abrt (user_tranid, scode);
```
Abort an active transaction, and remove any updates from the file.

Z$CLOS (bin, bin)
```
call z$clos(uniq_file_id, code);
```
Close an open PRISAM file.

Z$DDL (bin, pointer, bin)
```
call z$ddl (file_id, info_ptr, scode);
```
Return DDL information. (Data structure layout subject to change without notice - this routine intended for DISCOVER support).

Z$DELE (bin, bin)
```
call z$dele (file_id, scode);
```
Delete the current record of a PRISAM file.

Z$ENDT (bin, bin)
```
call z$endt (user_tranid, scode);
```
End an active transaction, and commit any updates to the file.

Z$FIND (bin, bin, char(*), bin, bin, char(*), bin, bin, bin)
```
call z$find(uniq_file_id, funct, key_buff, key_len, key_num,
          found_key_buff, found_key_len, reserved, code);
```
Find (not read) a record in a PRISAM file and make it the current record. funct = PFST, PLST, PEQU, PGRT, P$GRE.

Z$INSR (bin, char(*), bin, bin(31), bin, bin)
```
call z$insr(uniq_file_id, rec_buff, rec_len, rec_num, reserved,
     code);
```
Insert a new record into a PRISAM file.

Z$KDEL (bin, char(*), bin, bin, bin)

```
call z$kdel(uniq_file_id, key_buff, key_len, key_num, scode);
```
Delete a record by key match.

Z$KUPD (bin, char(*), bin, bin(31), bin, bin)
```
call z$kupd(uniq_file_id, rec_buff, rec_len, rec_num, reserved,
     code);
```
Replace the record which the keys in the record presented uniquely identify.

Z$KYST (bin, (*num_items* * 2)bin, bin, bin, 1, 2 char(30), 2 bin, 2 bin, 2 bin, 2 bin, 2 bin, 2 bin, 2 bin, 2 bin, bin)
```
call z$kyst(file_id, key_info, num_items, info_len, key_found,
          code);
```
Return key_num and information about a key.

Z$OPEN (bin, char(*), bin, bin, bin, bin, bin)
```
call z$open (open_key, pathname, pathname_size, tran_key,
           file_org, uniq_file_id, code);
```
Open an existing PRISAM file. open_key = (ONWT, OWAT) + (OFSH, OPRO, O$EXC) + (O$RDO, OEXO, OUPD). tran_key = ONCK, ONTM, O$TRM. file_org = O$IND, O$REL.

Z$READ (bin, bin, char(*), bin, char(*), bin, bin, char(*), bin, bin, bin, bin)
```
call z$read (uniq_file_id, funct, rec_buff, rec_len, key_buff,
  key_len, found_key_buff, found_key_len, recsize, reserved,
  code);
```
Read a record from a PRISAM file and make it the current record. funct = PFST, PLST, PEQU, PGRT, PGRE, PNXT, PNXE, PNXG, PPCD, PCUR.

Z$STRT (bin, bin, bin(31), bin)
```
call z$strt(key, user_tranid, roam_tranid, scode);
```
Start a transaction. key = (TRTV, TUPD) + (TCLR, TNCL)

Z$UPDT (bin, char(*), bin, bin, bin)
```
call z$updt(uniq_file_id, rec_buff, rec_len, reserved, code);
```
Replace the current record with a user-supplied record.

- 32s [illegible]

xxS [illegible] addressing
xXR limited to one segment
Store instr. flush the pipeline

64v
- Full range of Virtual mem
- Procedure base (PB) register used with Program Counter for Procedure Frame References
- link base (LB) register used for links frame ref
- Stack base (SB) register used for stack frame
- XC For external and Common ref
- NO Pipeline Flush

32I
- 8 general purpose registers (64v more) dedicated by
- Two floating point Reg (more than 64v)

32Ix

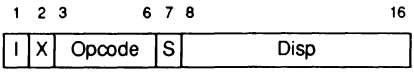
7. INSTRUCTION SET

7.1. Instruction formats

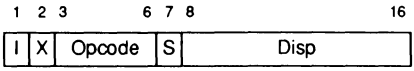
Further information may be found in the *Instruction Sets Guide* [19].

7.1.1. S, R, and V mode

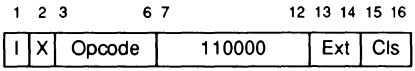
S & R - memory reference:



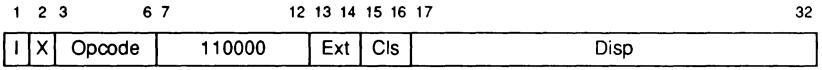
V - memory reference, short:



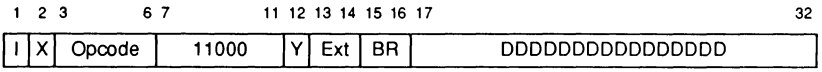
R - memory reference, long:



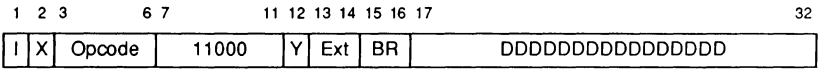
or

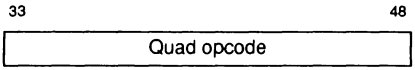


V - memory reference, long:

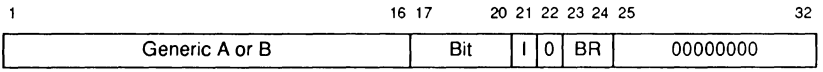


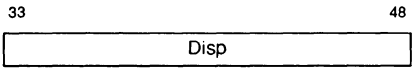
or



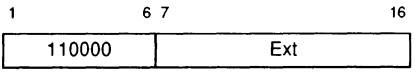


Generic AP:

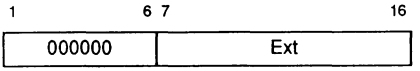




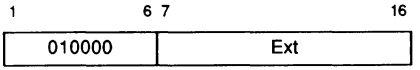
S, R & V - generic A:



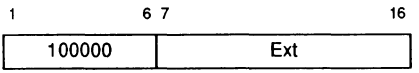
S, R & V - generic B:



S, R & V - shift:



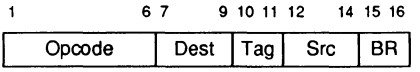
S, R & V - skip:



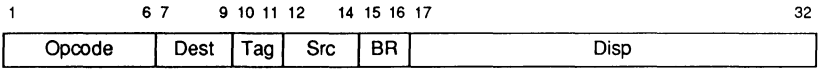
BR - base register
Bit - bit number
Cls - class bits
Disp - displacement
Ext - extended opcode
I - indirect
Opcode - opcode
Quad opcode - quad extended opcode
X - index
Y - use Y register for indexing

7.1.2. I mode

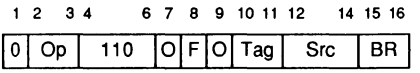
General memory reference:



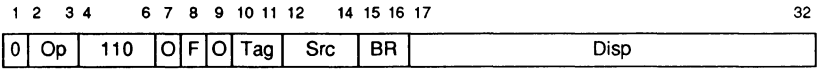
or



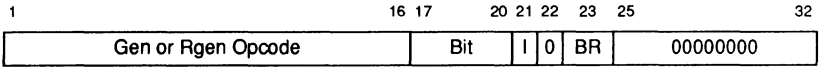
Special memory reference:

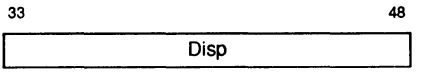


or

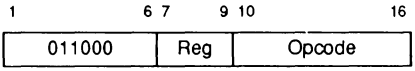


Generic AP:

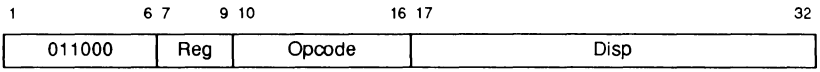




Register generic:



Register generic branch:



BR - base register
Bit - bit
Disp - displacement
F - floating point register
O, Opc, Opcode - opcode
Dst - destination register
Src - source register
Reg - general register
Tag - tag modifier

7.2. Machine Instructions

The 'type' column indicates the format and/or function of the operation as follows.

AP	Three-word operation, the last two words of which are an AP address pointer.
BR	Two-word operation, the second word of which is a word number within the current procedure segment to which to branch.
CON	Single-word control operation.
DA	Decimal arithmetic operation.
FE	Field and edit operation.
FLD	Single-word field operation.
FOP	Single-word floating-point operation.
FSK	Single-word floating-point skip operation.
IG	Single-word integrity operation.
IO	Single-word input/output operation.
LOG	Single-word logicize operation.
MGR	Memory reference/general register to register operation.
MOD	Single-word mode operation.
MR	Memory-reference operation.
OPR	Single-word miscellaneous operation.
PIO	Programmed input/output operation.
QAD	Quad floating point.
RAP	Register A P.
RGN	Register generic.
SH	Single-word shift operation.
SKP	Single-word skip operation.
VM	Virtual memory operation.

The 'C' column indicates the effect of the operation on the C-bit and the L-bit as follows.

-	C and L are unchanged by the operation.
1	C is unchanged, L is carry.
2	C is overflow, L is carry.
3	C is overflow, L is indeterminate.
4	C is shift extension, L is indeterminate.
5	C is a result of op, L is indeterminate.
6	C and L are indeterminate.

7 C and L are loaded by the operation.
8 C is cleared, L is indeterminate.
9 C is a result of op, L is unchanged.

The 'cc' column indicates the effect of the operation on the condition codes as follows.

- Cond. codes are unchanged.
1,4 Cond. codes result of arith op or compare.
5 Cond. codes indeterminate.
6 Cond. codes loaded by operation.
7 Cond. codes indicate result of operation.

The 'Modes' column indicates in which addressing modes the operation is available as follows.

S Available in 16S and 32S modes.
R Available in 32R and 64R modes.
V Available in 64V mode.
I Available in 32I mode (and 32IX).
* Restricted to Ring 0 execution.

Notes following instruction description in parentheses:

32IX I-mode extended instruction. Will not run on all machines.
Long: xxxxxx
Long form of instruction.
Pxxx For Prime xxx model only.
R Register to register form available.
RI Register to register and immediate forms available.

Mnem	OpCode	Typ	C	cc	Modes	Description
A	004---	MR	2	1	I	Add. R + [EA]32 => R. (RI)
A1A	141206	GEN	2	1	SRV	Add 1 to A. A + 1 => A.
A2A	140304	GEN	2	1	SRV	Add 2 to A. A + 2 => A.
ABQ	060134	AP	-	7	I	Add to bottom of Q. CCEQ -> FULL.
ABQ	141716	AP	-	7	V	Add to bottom of Q. CCEQ -> FULL.
ACA	141216	GEN	2	1	SRV	Add CBIT to A. CBIT + A => A.
ACP	132---	MGR	-	-	I	Add character pointer. (32IX, RI only, see SCC)
ADD	-14---	MR	2	1	SRV	Add. A + [EA]16 => A. (Long -15400)
ADL	-15414	MR	2	1	V	Add long. L + [EA]32 => L.
ADLL	141000	GEN	2	1	V	Add LINK to L.
ADLR	060014	RGN	-	7	I	Add LINK to R.
AH	024---	MR	2	1	I	Add halfword. RH + [EA]16 => RH. (RI)
AIP	172---	MGR	2	1	I	Add indirect pointer. (32IX)
ALFA	001301	FLD	6	-	V	Add L to FAR.
ALL	0414--	SH	4	-	SRV	A left logical.
ALR	0416--	SH	4	-	SRV	A left rotate.
ALS	0415--	SH	3	-	SRV	A left shift (arith).
ANA	-06---	MR	-	-	SRV	And. AND(A, [EA]16) => A. (Long: -07400)
ANL	-07414	MR	-	-	V	And long. AND(L, [EA]32) => L.
AOA	141206	GEN	2	1	SRV	OBSOLETE. Add 1 to A. A + 1 => A. (Use A1A)
ARFA	060161	FLD	6	-	I	Add R to FAR. FAR + R => FAR.
ARGT	000605	CON	6	5	VI	Argument transfer (used with PCL).
ARL	0404--	SH	4	-	SRV	A right logical.
ARR	0406--	SH	4	-	SRV	A right rotate.
ARS	0405--	SH	4	-	SRV	A right shift (arith).
ATQ	060135	AP	-	7	I	Add to top of queue. RH => Q. CCEQ -> FULL.

Mnem	OpCode	Typ	C	cc	Modes	Description
ATQ	141717	AP	-	7	V	Add to top of queue. A => Q. CCEQ -> FULL.
BCEQ	141602	BR	-	-	VI	Branch on Condition Code .EQ.
BCGE	141605	BR	-	-	VI	Branch on Condition Code .GE.
BCGT	141601	BR	-	-	VI	Branch on Condition Code .GT.
BCLE	141600	BR	-	-	VI	Branch on Condition Code .LE.
BCLT	141604	BR	-	-	VI	Branch on Condition Code .LT.
BCNE	141603	BR	-	-	VI	Branch on Condition Code .NE.
BCR	141705	BR	-	-	VI	Branch on CBIT reset.
BCS	141704	BR	-	-	VI	Branch on CBIT set.
BDX	140734	BR	-	-	V	Branch on decremented X.
BDY	140724	BR	-	-	V	Branch on decremented Y.
BEQ	140612	BR	-	4	V	Branch on A .EQ. 0.
BFEQ	020122	BR	-	4	I	Branch on FAC .EQ. 0.
BFEQ	141612	BR	-	4	V	Branch on FAC .EQ. 0.
BFGE	020125	BR	-	4	I	Branch on FAC .GE. 0.
BFGE	141615	BR	-	4	V	Branch on FAC .GE. 0.
BFGT	020121	BR	-	4	I	Branch on FAC .GT. 0.
BFGT	141611	BR	-	4	V	Branch on FAC .GT. 0.
BFLE	020120	BR	-	4	I	Branch on FAC .LE. 0.
BFLE	141610	BR	-	4	V	Branch on FAC .LE. 0.
BFLT	020124	BR	-	4	I	Branch on FAC .LT. 0.
BFLT	141614	BR	-	4	V	Branch on FAC .LT. 0.
BFNE	020123	BR	-	4	I	Branch on FAC .NE. 0.
BFNE	141613	BR	-	4	V	Branch on FAC .NE. 0.
BGE	140615	BR	-	4	V	Branch on A .GE. 0.
BGT	140611	BR	-	4	V	Branch on A .GT. 0.
BHD1	020144	BR	-	-	I	Branch on RH dec by 1. RH - 1 => RH.
BHD2	020145	BR	-	-	I	Branch on RH dec by 2. RH - 2 => RH.
BHD4	020146	BR	-	-	I	Branch on RH dec by 4. RH - 4 => RH.
BHEQ	020112	BR	-	4	I	Branch on RH .EQ. 0.
BHGE	020115	BR	-	4	I	Branch on RH .GE. 0.
BHGT	020111	BR	-	4	I	Branch on RH .GT. 0.
BHI1	020140	BR	-	-	I	Branch on RH incr by 1. RH + 1 => RH.
BHI2	020141	BR	-	-	I	Branch on RH incr by 2. RH + 2 => RH.
BHI4	020142	BR	-	-	I	Branch on RH incr by 4. RH + 4 => RH.
BHLE	020110	BR	-	4	I	Branch on RH .LE. 0.
BHLT	020114	BR	-	4	I	Branch on RH .LT. 0.
BHNE	020113	BR	-	4	I	Branch on RH .NE. 0.
BIX	141334	BR	-	-	V	Branch on incremented X ^= 0.
BIY	141324	BR	-	-	V	Branch on incremented Y ^= 0.
BLE	140610	BR	-	4	V	Branch on A $\leq$ 0.
BLEQ	140702	BR	-	4	V	Branch on L $=$ 0.
BLGE	140615	BR	-	4	V	Branch on L $\geq$ 0.
BLGT	140701	BR	-	4	V	Branch on L $>$ 0.
BLLE	140700	BR	-	4	V	Branch on L $<$ 0.
BLLT	140614	BR	-	4	V	Branch on L $\neq$ 0.
BLNE	140703	BR	-	4	V	Branch on L <> 0.
BLR	141707	BR	-	-	VI	Branch on LINK reset.
BLS	141706	BR	-	-	VI	Branch on LINK set.
BLT	140614	BR	-	4	V	Branch on A .LT. 0.
BMEQ	141602	BR	-	-	VI	Branch on mag-cond L,CC .EQ. (BCEQ)
BMGE	141706	BR	-	-	VI	Branch on mag-cond L,CC .GE. (BLS)
BMGT	141710	BR	-	-	VI	Branch on mag-cond L,CC .GT.
BMLE	141711	BR	-	-	VI	Branch on mag-cond L,CC .LE.
BMLT	141707	BR	-	-	VI	Branch on mag-cond L,CC .LT. (BLR)
BMNE	141603	BR	-	-	VI	Branch on mag-cond L,CC .NE. (BCNE)
BNE	140613	BR	-	4	V	Branch on A .NE. 0.
BRBR	02004-	BR	-	-	I	Branch on R bit reset.
BRBS	02000-	BR	-	-	I	Branch on R bit set.
BRD1	020134	BR	-	-	I	Branch on R dec by 1. R - 1 => R.
BRD2	020135	BR	-	-	I	Branch on R dec by 2. R - 2 => R.
BRD4	020136	BR	-	-	I	Branch on R dec by 4. R - 4 => R.
BREQ	020102	BR	-	4	I	Branch on R .EQ. 0.
BRGE	020105	BR	-	4	I	Branch on R .NE. 0.

Mnem	OpCode	Typ	C	cc	Modes	Description
BRGT	020101	BR	-	4	I	Branch on R .LE. 0.
BRI1	020130	BR	-	-	I	Branch on R incr by 1. R + 1 => R.
BRI2	020131	BR	-	-	I	Branch on R incr by 2. R + 2 => R.
BRI4	020132	BR	-	-	I	Branch on R incr by 4. R + 4 => R.
BRLE	020100	BR	-	4	I	Branch on R .LT. 0.
BRLT	020104	BR	-	4	I	Branch on R .GT. 0.
BRNE	020103	BR	-	4	I	Branch on R .GE. 0.
C	142---	MR	1	1	I	Compare R with [EA]32. (RI)
CAI	000411	IO	-	-	SRVI*	OBSOLETE. Clear active interrupt.
CAL	141050	GEN	-	-	SRV	Clear left byte of A.
CALF	000705	AP	6	5	SRVI	Proc call from faulting proc.
CAR	141044	GEN	-	-	SRV	Clear right byte of A.
CAS	-22---	MR	1	1	SRV	Skip 0,1,2 if A >,=,< [EA]16. (Long: -23400)
CAZ	140214	GEN	1	1	SRV	Skip 0,1,2 if A >,=,< 0.
CEA	000111	GEN	-	-	SR	Compute effective address. EA => A.
CGT	060026	RGN	6	5	I	Computed go to.
CGT	001314	GEN	6	5	V	Computed go to.
CH	162---	MR	1	1	I	Compare RH with [EA]16. (RI)
CHS	060040	RGN	-	-	I	Change sign of R. ^R(1) => R(1).
CHS	140024	GEN	-	-	SRV	Change sign of A. ^A(1) => A(1).
CLS	-23414	MR	1	1	V	Skip 0,1,2 if L >,=,< [EA]32.
CMA	140401	GEN	-	-	SRV	One's complement A. ^A => A.
CMH	060045	RGN	-	-	I	Complement RH. ^RH => RH.
CMR	060044	RGN	-	-	I	Complement R. ^R => R.
CR	060056	RGN	-	-	I	Clear R. 0 => R.
CRA	140040	GEN	-	-	SRV	Clear A. 0 => A.
CRB	140015	GEN	-	-	SRV	Clear B. 0 => B.
CRB	140014	GEN	-	-	SRV	OBSOLETE. Clears B & LSW of DFAC(6). (Use CRB)
CRBL	060062	RGN	-	-	I	Clear R left byte. 0 => R(1-8).
CRBR	060063	RGN	-	-	I	Clear R right byte. 0 => R(9-16).
CRE	141404	GEN	-	-	V	Clear E. 0 => E.
CREP	-21410	MR	-	-	R	OBS. Call re-ent. proc. P+1 => [S+1]16, EA => P.
CRHL	060054	RGN	-	-	I	Clear RH. 0 => RH.
CRHR	060055	RGN	-	-	I	Clear R right halfword. 0 => R(17-32).
CRL	140010	GEN	-	-	SRV	Clear L. 0 => L.
CRLE	141410	GEN	-	-	V	Clear L and E. 0 => L, 0 => E.
CSA	140320	GEN	5	-	SRV	Copy sign of A. A(1) => CBIT,0 => A(1).
CSR	060041	RGN	5	-	I	Copy & save sign. R(1) => C, 0 => R(1).
CXCS	001714	IG	-	-	VI*	OBSOLETE. Control extended control store.
D	144---	MR	3	5	I	Divide. (R,R+1)/[EA]32 => R; REM => R + 1. (RI)
DAD	-14---	MR	2	1	SR	Dbl. add. (A,B)+[EA]32 => A,B w/hole. (DP,Long: -15400)
DBL	000007	MOD	-	-	SR	Enter double-prec mode.
DBLE	060106	FOP	-	-	I	Convert single to double fltg pt.
DCP	060160	RGN	-	-	I	Decrement character pointer. (32IX)
DFA	0352--	MR	3	5	I	Dbl fltg add. DFR + [EA]64 => DFR. (RI)
DFAD	-15410	MR	3	5	RV	Dbl fltg add. DFAC + [EA]64 => DFAC.
DFC	0152--	MR	-	1	I	Dbl fltg compare DFR to [EA]64. (RI)
DFCM	060144	FOP	3	5	I	Dbl fltg complement. -DFAC => DFAC.
DFCM	140574	FOP	3	5	RV	Dbl fltg complement. -DFAC => DFAC.
DFCS	-23410	MR	6	5	RV	Skip 0,1,2 if DFAC >,=,< [EA]64.
DFD	0742--	MR	3	5	I	Dbl fltg divide. DFAC/[EA]64 => DFAC. (RI)
DFDV	-37410	MR	3	5	RV	Dbl fltg divide. DFAC/[EA]64 => DFAC.
DFL	0142--	MR	-	-	I	Dbl fltg load. [EA]64 => DFAC. (RI)
DFLD	-05410	MR	-	-	RV	Dbl fltg load. [EA]64 => DFAC.
DFLX	-33410	MR	-	-	V	Load dbl fltg index. 4*[EA]16 => X. (No X)

Mnem	OpCode	Typ	C	cc	Modes	Description
DFM	0552--	MR	3	5	I	Dbl fltg multiply. DFAC * [EA]64 => DFAC. (RI)
DFMP	-35410	MR	3	5	RV	Dbl fltg multiply. DFAC * [EA]64 => DFAC.
DFS	0542--	MR	3	5	I	Dbl fltg subtract. DFAC - [EA]64 => DFAC. (RI)
DFSB	-17410	MR	3	5	RV	Dbl fltg subtract. DFAC - [EA]64 => DFAC.
DFST	0342--	MR	-	-	I	Dbl fltg store. DFAC => [EA]64.
DFST	-11410	MR	-	-	RV	Dbl fltg store. DFAC => [EA]64.
DH	164---	MR	3	5	I	Divide halfword. R/[EA]16 => RH; RM => RL. (RI)
DH1	060130	RGN	2	1	I	Decr RH by 1. RH - 1 => RH.
DH2	060131	RGN	2	1	I	Decr RH by 2. RH - 2 => RH.
DIV	-36---	MR	3	5	SR	Divide. (A,B)31/[EA]16 => A; REM => B. (Long: -37400)
DIV	-36---	MR	3	5	V	Divide. L/[EA]16 => A, REM => B. (Long: -37400)
DLD	-04---	MR	-	-	SR	Double load. [EA]32 => A,B. (DP) (Long: -05400)
DM	1540--	MR	-	1	I	Decr memory. [EA]32 - 1 => [EA]32.
DMH	1740--	MR	-	1	I	Decr memory halfword. [EA]16 - 1 => [EA]16.
DR1	060124	RGN	2	1	I	Decr R by 1. R - 1 => R.
DR2	060125	RGN	2	1	I	Decr R by 2. R - 2 => R.
DRN	040300	FOP	3	5	VI	Double round from quad.
DRNM	140571	FOP	3	5	VI	Double round from quad to -infinity.
DRNP	040301	FOP	3	5	VI	Double round from quad to +infinity.
DRNZ	040302	FOP	3	5	VI	Double round from quad to 0.
DRX	140210	GEN	-	-	SRV	Decrement X and skip if 0.
DSB	-16---	MR	2	1	SR	Dbl subtract. (A,B)-[EA] => A,B w/hole. (DP, Long: -17400)
DST	-10---	MR	-	-	SR	Double store. (A,B) => [EA]32. (DP,Long: -11400)
DVL	-37414	MR	3	5	V	Divide long. (L,E)/[EA]32 => L; REM => E.
DXA	000011	MOD	-	-	SRVI	OBSOLETE. Enter 16K sectored mode. (Use E16S)
E16S	000011	MOD	-	-	SRVI	Enter P300 16K sectored mode.
E32I	001010	MOD	-	-	SRVI	Enter P500 32I mode.
E32R	001013	MOD	-	-	SRVI	Enter P300 32K relative mode.
E32S	000013	MOD	-	-	SRVI	Enter P300 32K sectored mode.
E64R	001011	MOD	-	-	SRVI	Enter P300 64K relative mode.
E64V	000010	MOD	-	-	SRVI	Enter P400 64K virtual mode.
EAA	-03404	MR	-	-	R	Eff. addr to A. EA => A.
EAFA	001300	AP	-	-	VI	Eff. addr to FAR.
EAL	-03404	MR	-	-	V	Eff. addr to L. EA => L.
EALB	1144--	MR	-	-	I	Eff. addr to LB. EA => LB.
EALB	-27410	MR	-	-	V	Eff. addr to LB. EA => LB.
EAR	146---	MR	-	-	I	Eff. addr to R. EA => R.
EAXB	1344--	MR	-	-	I	Eff. addr to XB. EA => XB.
EAXB	-25410	MR	-	-	V	Eff. addr to XB. EA => XB.
EIO	070---	MR	-	7	I*	Execute EA as I/O inst. CCEQ -> success.
EIO	-31404	MR	-	7	V*	Execute EA as I/O inst. CCEQ -> success.
EMCM	000503	IG	-	-	SRVI*	OBSOLETE. Enter machine check mode.
ENB	000401	IO	-	-	SRVI*	Enable interrupts.
ENBL	000401	IO	-	-	SRVI*	Enable interrupts (local). (P850)
ENBM	000400	IO	-	-	SRVI*	Enable interrupts (mutual). (P850)
ENBP	000402	IO	-	-	SRVI*	Enable interrupts (process). (P850)

Mnem	OpCode	Typ	C	cc	Modes	Description
ENTR	-03414	MR	-	-	R	OBSOLETE. Enter recursive proc stack.
EPMJ	000217	VM	-	-	SR	OBSOLETE. Enter page mode & jump (P300).
EPMX	000237	VM	-	-	SR	OBS. Enter page mode & jump to microcode (P300).
ERA	-12---	MR	-	-	SRV	Exclusive or. XOR(A, [EA]16) => A.
ERL	-13414	MR	-	-	V	Exclusive or long. XOR(L, [EA]32) => L.
ERMJ	000701	VM	-	-	SR	OBSOLETE. Enter restricted mode & jump (P300).
ERMX	000721	VM	-	-	SR	OBS. Enter restr'd mode & jump to ucode (P300).
ESIM	000415	MOD	-	-	SRVI*	OBSOLETE. Enter standard interrupt mode.
EVIM	000417	MOD	-	-	SRVI*	OBSOLETE. Enter vectored interupt mode.
EVMJ	000703	VM	-	-	SR	OBSOLETE. Enter virtual mode & jump (P300).
EVMX	000723	VM	-	-	SR	OBS. Enter virtual mode & jump to ucode (P300).
EXA	000013	MOD	-	-	SRVI	OBSOLETE. Enter 32K sectored mode. (Use E32S)
FA	0350--	MR	3	5	I	Fltg add. FAC + [EA]32 => FAC. (RI)
FAD	-15404	MR	3	5	RV	Fltg add. FAC + [EA]32 => FAC.
FC	0150--	MR	-	1	I	Fltg compare FAC with [EA]32. (RI)
FCDQ	140571	FOP	-	-	VI	Fltg convert dbl to quad. (P9950)
FCM	060100	FOP	3	5	I	Fltg complement. -FAC => FAC.
FCM	140530	FOP	3	5	RV	Fltg complement. -FAC => FAC.
FCS	-23404	MR	6	5	RV	Skip 0,1,2 if FAC >,=,< [EA]32. (RI)
FD	0740--	MR	3	5	I	Fltg divide. FAC / [EA]32 => FAC.
FDBL	140016	FOP	-	-	V	Fltg convert single to dbl. FAC => DFAC.
FDV	-37404	MR	3	5	RV	Fltg divide. FAC / [EA]32 => FAC.
FL	0140--	MR	-	-	I	Fltg load. [EA]32 => FAC. (RI)
FLD	-05404	MR	-	-	RV	Fltg load. [EA]32 => FAC.
FLOT	140550	FOP	6	5	R	Convert int to fltg. Flot(A,B)=>FAC w/ hole.
FLT	060105	FOP	6	5	I	Convert int to fltg. Flot(R) => FAC.
FLTA	140532	FOP	6	5	V	Convert int to fltg. Flot(A) => FAC.
FLTH	060102	FOP	6	5	I	Convert half word int to fltg pt.
FLTL	140535	FOP	6	5	V	Convert long to fltg. Flot(L)=>FAC
FLX	-33404	MR	-	-	RV	Load fltg index. 2*[EA]16 => X. (No X)
FM	0550--	MR	3	5	I	Fltg multiply. FAC * [EA]32 => FAC. (RI)
FMP	-35404	MR	3	5	RV	Fltg multiply. FAC * [EA]32 => FAC.
FRN	060107	FOP	3	5	I	Fltg round.
FRN	140534	FOP	3	5	RV	Fltg round up.
FRNM	060146	FOP	3	5	I	Fltg round towards - infinity.
FRNM	040320	FOP	3	5	V	Fltg round towards - infinity.
FRNP	060145	FOP	3	5	I	Fltg round towards + infinity.
FRNP	040303	FOP	3	5	V	Fltg round towards + infinity.
FRNZ	060147	FOP	3	5	I	Fltg round towards zero.
FRNZ	040321	FOP	3	5	V	Fltg round towards zero.
FS	0540--	MR	3	5	I	Fltg subtract. FAC - [EA]32 => FAC. (RI)
FSB	-17404	MR	3	5	RV	Fltg subtract. FAC - [EA]32 => FAC.
FSGT	140515	FSK	-	1	RV	Fltg skip if .GT. 0.
FSLE	140514	FSK	-	1	RV	Fltg skip if .LE. 0.
FSMI	140512	FSK	-	1	RV	Fltg skip if .LT. 0.
FSNZ	140511	FSK	-	1	RV	Fltg skip if .NE. 0.
FSPL	140513	FSK	-	1	RV	Fltg skip if .GE. 0.
FST	0340--	MR	3	5	I	Fltg store. FAC => [EA]32.
FST	-11404	MR	3	5	RV	Fltg store. FAC => [EA]32.
FSZE	140510	FSK	-	1	RV	Fltg skip if .EQ. 0.
HLT	000000	CON	-	-	SRVI*	Halt computer operation.
I	102---	MR	-	-	I	Interchange R with [EA]32. (R)

Mnem	OpCode	Typ	C	cc	Modes	Description
IAB	000201	GEN	-	-	SRV	Exchange A and B. A => B & B => A.
ICA	141340	GEN	-	-	SRV	Interchange bytes of A.
ICBL	060065	RGN	-	-	I	Exchange bytes. 0 => RH(1-8) => RH(9-16).
ICBR	060066	RGN	-	-	I	Exchange bytes. 0 => RH(9-16) => RH(1-8).
ICHL	060060	RGN	-	-	I	Interchange halfwords. RH => RL, 0 => RH.
ICHR	060061	RGN	-	-	I	Interchange halfwords. RL => RH, 0 => RL.
ICL	141140	GEN	-	-	SRV	Exchange bytes of A & clr left.
ICP	060167	RGN	-	-	I	Increment character pointer. (32IX)
ICR	141240	GEN	-	-	SRV	Exchange bytes of A & clr right.
IH	122---	MR	-	-	I	Interchange RH with [EA]16. (R)
IH1	060126	RGN	2	1	I	Incr halfword by 1. RH + 1 => RH.
IH2	060127	RGN	2	1	I	Incr halfword by 2. RH + 2 => RH.
ILE	141414	GEN	-	-	V	Exchange L and E. L => E & E => L.
IM	1140--	MR	-	1	I	Incr memory. [EA]32 + 1 => [EA]32.
IMA	-26---	MR	-	-	SRV	Exchange memory and A. (LONG: -27400)
IMH	1340--	MR	-	1	I	Incr halfword. [EA]16 + 1 => [EA]16.
INA	130---	PIO	-	-	SR*	Input to A.
INBC	001217	AP	6	5	VI*	Interrupt ntfy LIFO, clear active interrupt.
INBN	001215	AP	6	5	VI*	Interrupt ntfy LIFO.
INEC	001216	AP	6	5	VI*	Interrupt ntfy FIFO, clear active interrupt.
INEN	001214	AP	6	5	VI*	Interrupt ntfy FIFO.
INH	001001	IO	-	-	SRVI*	Inhibit interrupts.
INHL	001001	IO	-	-	SRVI*	Inhibit interrupts (local). (P850)
INHM	001000	IO	-	-	SRVI*	Inhibit interrupts (mutual). (P850)
INHP	001002	IO	-	-	SRVI*	Inhibit interrupts (process). (P850)
INK	060070	RGN	-	-	I	Input keys to RH.
INK	000043	GEN	-	-	SR	Input P300 keys into A.
INT	060103	FOP	3	5	I	Convert fltg to int. INT(FAC) => R.
INT	140554	FOP	3	5	R	Convert fltg to int. INT(FAC) => A,B w/ hole.
INTA	140531	FOP	3	5	V	Convert fltg to int. INT(FAC) => A.
INTH	060101	FOP	3	5	I	Convert fltg to halfword. INT(FAC) => RH.
INTL	140533	FOP	3	5	V	Convert fltg to int long. INT(FAC) => L.
IR1	060122	RGN	2	1	I	Incr R by 1. R + 1 => R.
IR2	060123	RGN	2	1	I	Incr R by 2. R + 2 => R.
IRB	060064	RGN	-	-	I	Interchange bytes. RH(1-8) <=> RH(9-16).
IRH	060057	RGN	-	-	I	Interchange halves. RH <=> RL.
IRS	-24---	MR	-	-	SRV	Inc, replace, and skip if zero. (Long: -25400)
IRTC	000603	CON	7	6	VI*	Interrupt return, clear active intrpt.
IRTN	000601	CON	7	6	VI*	Interrupt return.
IRX	140114	GEN	-	-	SRV	Increment X and skip if 0.
ITLB	000615	CON	6	5	VI*	Invalidate STLB entry, L, R2 = VADDR.
JDX	-33410	MR	-	-	R	Decrement X & jump if not zero. (No X)
JEQ	-05414	MR	-	-	R	OBSOLETE. Jump if A .EQ. 0, EA => P.
JGE	-17414	MR	-	-	R	OBSOLETE. Jump if A .GE. 0, EA => P.
JGT	-13414	MR	-	-	R	OBSOLETE. Jump if A .GT. 0, EA => P.
JIX	-33414	MR	-	-	R	Increment X & jump if not zero. (No X)
JLE	-11414	MR	-	-	R	OBSOLETE. Jump if A .LE. 0, EA => P.
JLT	-15414	MR	-	-	R	OBSOLETE. Jump if A .LT. 0, EA => P.
JMP	1342--	MR	-	-	I	Jump. EA => P.
JMP	-02---	MR	-	-	SRV	Jump (uncond). EA => PB,P. (Long: -03400)
JNE	-07414	MR	-	-	R	OBSOLETE. Jump if A .NE. 0, EA => P.
JSR	166---	MR	-	-	I	Jump to subr. P => RH, EA32 => P.

Mnem	OpCode	Typ	C	cc	Modes	Description
JST	-20---	MR	-	-	SRV	Jump & store. P => [EA]16, EA+1 => P. (Long: -021400)
JSX	-73414	MR	-	-	RV	Jump & save in X. P => X, EA => P. (No X)
JSXB	1542--	MR	-	-	I	Jump & set XB. P => XB, EA => P.
JSXB	-31410	MR	-	-	V	Jump & set XB. PB => XB, EA => PB.
JSY	-30---	MR	-	-	V	Jump & save in Y. P => Y, EA => P. (Long: -031400)
L	002---	MR	-	-	I	Load R. [EA]32 => R. (RI)
LCC	112---	MGR	-	-	I	Load character via char pointer. (32IX, RI)
LCEQ	060153	LOG	-	-	I	Load RH if EQ. CCEQ => RH.
LCEQ	141503	LOG	-	-	V	Load A if EQ. CCEQ => A.
LCGE	060154	LOG	-	-	I	Load RH if GE. CCGE => RH.
LCGE	141504	LOG	-	-	V	Load A if GE. CCGE => A.
LCGT	060155	LOG	-	-	I	Load RH if GT. CCGT => RH.
LCGT	141505	LOG	-	-	V	Load A if GT. CCGT => A.
LCLE	060151	LOG	-	-	I	Load RH if LE. CCLE => RH.
LCLE	141501	LOG	-	-	V	Load A if LE. CCLE => A.
LCLT	060150	LOG	-	-	I	Load RH if LT. CCLT => RH.
LCLT	141500	LOG	-	-	V	Load A if LT. CCLT => A.
LCNE	060152	LOG	-	-	I	Load RH if NE. CCNE => RH.
LCNE	141502	LOG	-	-	V	Load A if NE. CCNE => A.
LDA	-04---	MR	-	-	SRV	Load A. [EA]16 => A. (Long: -05400)
LDAR	110---	MR	-	5	I(*)	Load addressed register.
LDC	060162	FLD	-	7	I	Load char to RH.
LDC	001302	FLD	-	7	V	Load char to A via FAR.
LDL	-05414	MR	-	-	V	Load long. [EA]32 => L.
LDLR	-13404	MR	-	5	V(*)	Load long from addressed reg.
LDX	-72---	MR	-	-	SRV	Load X. [EA]16 => X. (No X, Long: -73414)
LDY	-73404	MR	-	-	V	Load Y. [EA]16 => Y. (No X)
LEQ	060003	LOG	-	4	I	Load RH if R = 0. (R = 0) => RH.
LEQ	140413	LOG	-	4	SRV	If A .EQ. 0, 1 => A, else 0 => A.
LF	060016	LOG	-	4	I	Logicize false. 0 => RH.
LF	140416	LOG	-	5	SRV	Logicize false. 0 => A.
LFEQ	060023	LOG	-	4	I	Load RH if FAC = 0. (FAC = 0) => RH.
LFEQ	141113	LOG	-	4	V	If FAC .EQ. 0, 1 => A, else 0 => A.
LFGE	060024	LOG	-	4	I	Load RH if FAC >= 0. (FAC >= 0) => RH.
LFGE	141114	LOG	-	4	V	If FAC .GE. 0, 1 => A, else 0 => A.
LFGT	060025	LOG	-	4	I	Load RH if FAC > 0. (FAC > 0) => RH.
LFGT	141115	LOG	-	4	V	If FAC .GT. 0, 1 => A, else 0 => A.
LFLE	060021	LOG	-	4	I	Load RH if FAC <= 0. (FAC <= 0) => RH.
LFLE	141111	LOG	-	4	V	If FAC .LE. 0, 1 => A, else 0 => A.
LFLI	001303	FLD	-	-	SRVI	Load FLR immediate.
LFLT	060020	LOG	-	4	I	Load RH if FAC < 0. (FAC < 0) => RH.
LFLT	141110	LOG	-	4	V	If FAC .LT. 0, 1 => A, else 0 => A.
LFNE	060022	LOG	-	4	I	Load RH if FAC <> 0. (FAC <> 0) => RH.
LFNE	141112	LOG	-	4	V	If FAC .NE. 0, 1 => A, else 0 => A.
LGE	060004	LOG	-	4	I	Load RH if R >= 0. (R >= 0) => RH.
LGE	140414	LOG	-	4	SRV	If A .GE. 0, 1 => A, else 0 => A.
LGL	0414--	SH	4	5	SRV	OBSOLETE. A left logical. (Use ALL)
LGR	0404--	SH	4	5	SRV	OBSOLETE. A right logical. (Use ARL)
LGT	060005	LOG	-	4	I	Load RH if R > 0. (R > 0) => RH.
LGT	140415	LOG	-	4	SRV	If A .GT. 0, 1 => A, else 0 => A.
LH	022---	MR	-	-	I	Load halfword. [EA]16 => RH. (RI)
LHEQ	060013	LOG	-	4	I	Load RH if RH = 0. (RH = 0) => RH.
LHGE	060004	LOG	-	4	I	Load RH if RH >= 0. (RH >= 0) => RH.
LHGT	060015	LOG	-	4	I	Load RH if RH > 0. (RH > 0) => RH.
LHL1	010---	MR	-	-	I	Load halfwd shifted by 1. LS([EA]16,1) => RH. (R)

Mnem	OpCode	Typ	C	cc	Modes	Description
LHL2	030---	MR	-	-	I	Load halfwd shifted by 2. LS([EA]16,2) => RH. (R)
LHL3	072---	MR	-	-	I	Load halfwd shifted by 3. LS([EA]16,3) => RH. (R)
LHLE	060011	LOG	-	4	I	Load RH if RH <= 0. (RH <= 0) => RH.
LHLT	060000	LOG	-	4	I	Load RH if RH < 0. (RH < 0) => RH.
LHNE	060012	LOG	-	4	I	Load RH if RH <> 0. (RH <> 0) => RH.
LIOT	000044	AP	6	5	VI*	Load IOTLB. L, R2 -> target virt addr.
LIP	152---	MGR	-	-	I	Load indirect pointer. (32IX)
LLE	060001	LOG	-	4	I	Load RH if R <= 0. (R <= 0) => RH.
LLE	140411	LOG	-	4	SRV	If A .LE. 0, 1 => A, else 0 => A.
LLEQ	141513	LOG	-	4	V	If L. EQ. 0, 1 => A, else 0 => A.
LLGE	140414	LOG	-	4	V	If L. GE. 0, 1 => A, else 0 => A.
LLGT	141515	LOG	-	4	V	If L. GT. 0, 1 => A, else 0 => A.
LLL	0410--	SH	4	-	SRV	Long left logical.
LLLE	141511	LOG	-	4	V	If L. LE. 0, 1 => A, else 0 => A.
LLLT	140410	LOG	-	4	V	If L. LT. 0, 1 => A, else 0 => A.
LLNE	141512	LOG	-	4	V	If L. NE. 0, 1 => A, else 0 => A.
LLR	0412--	SH	4	-	SRV	Long left rotate.
LLS	0411--	SH	3	5	SRV	Long left shift. (SR -> B(1) ignored)
LLT	060000	LOG	-	4	I	Load R if R < 0. (R < 0) => R.
LLT	140410	LOG	-	4	SRV	If A .LT. 0, 1 => A, else 0 => A.
LMCM	000501	IG	-	-	SRVI*	Leave machine check mode.
LNE	060002	LOG	-	4	I	Load R if R <> 0. (R <> 0) => R.
LNE	140412	LOG	-	4	SRV	If A .NE. 0, 1 => A, else 0 => A.
LPID	000617	CON	-	-	VI*	Load process ID from A(1-12), R2(1-12).
LPMJ	000215	VM	-	-	SR	OBSOLETE. Leave page mode & jump (P300).
LPMX	000235	VM	-	-	SR	OBS. Leave page mode & jump to microcode (P300).
LPSW	000711	AP	7	6	VI*	Load PSW (SN,WN,KEYS,MODALS).
LRL	0400--	SH	4	-	SRV	Long right logical.
LRR	0402--	SH	4	-	SRV	Long right rotate.
LRS	0401--	SH	4	-	SRV	Long right shift. (SR -> B(1) ignored)
LT	060017	LOG	-	5	I	Logic set true. 1 => R.
LT	140417	LOG	-	5	SRV	Logicize true. 1 => A.
LWCS	001710	IG	-	-	VI	OBSOLETE. Load writable control store.
M	104---	MR	-	-	I	Multiply. R * [EA]32 => (R,R+1). (RI)
MDEI	001304	IG	-	-	VI*	OBSOLETE. Mem diag enable interleave.
MDII	001305	IG	-	-	VI*	OBSOLETE. Mem diag inhibit interleave.
MDIW	001324	IG	-	-	VI*	OBSOLETE. Mem diag write interleave. L => [E].
MDRS	001306	IG	-	-	VI*	OBSOLETE. Mem diag read syndrome bits.
MDWC	001307	IG	-	-	VI*	OBSOLETE. Mem diag load write control reg.
MH	124---	MR	3	5	I	Multiply halfword. RH * [EA]16 => R. (RI)
MIA	150---	MR	-	-	I	OBSOLETE. Microcode execute A.
MIA	-25404	MR	-	-	V	OBSOLETE. Microcode execute A.
MIB	170---	MR	-	-	I	OBSOLETE. Microcode execute B.
MIB	-27404	MR	-	-	V	OBSOLETE. Microcode execute B.
MPL	-35414	MR	-	-	V	Multiply long. L * [EA]32 => L,E.
MPY	-34---	MR	3	-	V	Multiply. A * [EA]16 => A,B. (Long: -35400)
MPY	-34---	MR	3	-	SR	Multiply. A * [EA]16 => (A,B)31. (Long: -35400)
N	006---	MR	-	-	I	And. AND(R, [EA]32) => R. (RI)
NFYB	001211	AP	6	5	VI*	Notify on sem at AP. LIFO Q.
NFYE	001210	AP	6	5	VI*	Notify on sem at AP. FIFO Q.

Mnem	OpCode	Typ	C	cc	Modes	Description
NH	026---	MR	-	-	I	And halfword. AND(RH, [EA]16) => RH. (RI)
NOP	000001	GEN	-	-	SRVI	No operation.
NOP	101000	SKP	-	-	SRV	No operation (faster on certain machines).
NRM	000101	GEN	-	-	SR	OBSOLETE. Normalize A,B as on P300.
O	046---	MR	-	-	I	Or. OR(R, [EA]32) => R. (RI)
OCP	030---	PIO	-	-	SR*	Output control pulse.
OH	066---	MR	-	-	I	Or halfword. OR(RH, [EA]16) => RH. (RI)
ORA	-07410	MR	-	-	V	Or. OR(A, [EA]16) => A.
OTA	170---	PIO	-	-	SR*	Output from A.
OTK	060071	RGN	7	6	I	Output keys from RH. [RH] => KEYS.
OTK	000405	GEN	7	6	SR	Output A to P300 KEYS & S. (TAK in V-mode)
PCL	1142--	MR	6	5	I	Procedure call.
PCL	-21410	MR	6	5	V	Procedure call.
PID	060052	RGN	-	-	I	Pos for int divide. R => R+1; w/ sign extend.
PID	000211	GEN	-	-	SR	Pos for divide. A => L w/ sign ext. & hole.
PIDA	000115	GEN	-	-	V	Pos for int divide. A => L w/ sign extend.
PIDH	060053	RGN	-	-	I	Pos RH for div. RH => RL; RH(1) => RH(2-16).
PIDL	000305	GEN	-	-	V	Pos for long divide. L => E w/ sign extend.
PIM	060050	RGN	3	5	I	Pos after int multiply. (R+1) => R.
PIM	000205	GEN	-	-	SR	Pos after mult. B(2-16) => A(2-16)
PIMA	000015	GEN	3	5	V	Pos after mult. L => A.
PIMH	060051	RGN	3	5	I	Pos RH after int multiply. RL => RH.
PIML	000301	GEN	3	5	V	Pos after mult long. (L,E) => L.
PRTN	000611	CON	7	6	VI	Procedure return.
PTLB	000064	MOD	6	5	VI*	Purge TLB (non-IO). L, R2, R3. (CRE first)
QFAD	0754--	QAD	3	5	I	Quad fltg add. QAC + [EA]112 => QAC.
QFAD	-13410	QAD	3	5	V	Quad fltg add. QAC + [EA]112 => QAC. (Ext: 2)
QFC	1156--	QAD	-	7	I	Quad floating compare QAF to [EA]112. (RI)
QFCM	140570	QAD	3	5	VI	Quad fltg complement. -QAC => QAC
QFCS	-13410	QAD	6	5	V	Skip 0,1,2 if QAC >,=,< [EA]128. (Ext: 6)
QFDV	1154--	QAD	3	5	I	Quad fltg divide. QAC / [EA]112 => QAC.
QFDV	-13410	QAD	3	5	V	Quad fltg divide. QAC / [EA]112 => QAC. (Ext: 5)
QFLD	0750--	QAD	-	-	I	Quad fltg load. [EA]112/128 => QAC.
QFLD	-13410	QAD	-	-	V	Quad fltg load. [EA]112/128 => QAC. (Ext: 0)
QFLX	-33414	QAD	-	-	V	Quad fltg load index. [EA]*8 => X,Y. (No X)
QFMP	1152--	QAD	3	5	I	Quad fltg multiply. QAC * [EA]112 => QAC.
QFMP	-13410	QAD	3	5	V	Quad fltg mpy. QAC * [EA]112 => QAC. (Ext: 4)
QFSB	0756--	QAD	3	5	I	Quad fltg subtract. QAC - [EA]112 => QAC.
QFSB	-13410	QAD	3	5	V	Quad fltg sub. QAC - [EA]112 => QAC. (Ext: 3)
QFST	0752--	QAD	-	-	I	Quad fltg store. QAC => [EA]128.
QFST	-13410	QAD	-	-	V	Quad fltg store. QAC => [EA]128. (Ext: 1)

Mnem	OpCode	Typ	C	cc	Modes	Description
QINQ	140572	QAD	3	5	VI	Convert quad to integer.
QIQR	140573	QAD	3	5	VI	Convert quad to integer rounded.
RBQ	060133	AP	-	7	I	Remove from bottom of Q. emp -> 0 => RH, CCEQ
RBQ	141715	AP	-	7	V	Remove from bottom of Q. emp -> 0 => A, CCEQ
RCB	140200	GEN	9	-	SRVI	Reset CBIT. 0 => CBIT.
RMC	000021	IG	-	-	SRVI*	Reset machine check flag.
RMP	000021	IG	-	-	SRVI*	OBSOLETE. Reset machine check flag. (Use RMC)
ROT	050---	MR	4	-	I	Rotate. Shift(R,[EA]16) => R.
RRST	000717	AP	-	-	VI	Restore registers (GEN, FLT, XB).
RSAV	000715	AP	-	-	VI	Save registers (GEN, FLT, XB).
RTN	000105	GEN	-	-	SR	OBSOLETE. Return from P300 recur proc.
RTQ	060132	AP	-	7	I	Remove from top of Q. empty -> 0 => RH, CCEQ
RTQ	141714	AP	-	7	V	Remove from top of Q. empty -> 0 => A, CCEQ
RTS	000511	MOD	-	-	VI*	Reset time slice with A, R2.
S	044---	MR	2	1	I	Subtract. R - [EA]32 => R. (RI)
S1A	140110	GEN	2	1	SRV	Subtract 1 from A. A - 1 => A.
S2A	140310	GEN	2	1	SRV	Subtract 2 from A. A - 2 => A.
SAR	10026-	SKP	-	-	SRV	Skip if A(n) reset.
SAS	10126-	SKP	-	-	SRV	Skip if A(n) set.
SBL	-17414	MR	2	1	V	Subtract long. L - [EA]32 => L.
SCA	000041	GEN	-	-	SR	OBSOLETE. Load P300 shift count into A.
SCB	140600	GEN	5	-	SRVI	Set CBIT. 1 => CBIT.
SCC	132---	MGR	-	-	I	Store character via char pointer. (32IX)
SEQ	100040	SKP	-	-	SRV	OBSOLETE. Skip if A .EQ. 0. (Use SZE)
SGE	100400	SKP	-	-	SRV	OBSOLETE. Skip if A .GE. 0. (Use SPL)
SGL	000005	MOD	-	-	SR	Enter single-precision mode.
SGT	100220	SKP	-	-	SRV	Skip if A .GT. 0.
SH	064---	MR	2	1	I	Subtract halfword. RH - [EA]16 => RH. (RI)
SHA	032---	MR	4	-	I	Arithmetic shift. Shift(R,[EA]16) => R.
SHL	012---	MR	4	-	I	Logical shift. Shift(R,[EA]16) => R.
SHL1	060076	RGN	4	-	I	Shift halfword left 1. LS(RH, 1) => RH.
SHL2	060077	RGN	4	-	I	Shift halfword left 2. LS(RH, 2) => RH.
SHR1	060120	RGN	4	-	I	Shift halfword right 1. RS(RH, 1) => RH.
SHR2	060121	RGN	4	-	I	Shift halfword right 2. RS(RH, 2) => RH.
SKP	100000	SKP	-	-	SRV	Skip one word.
SKS	070---	PIO	-	-	SR*	Skip if condition set.
SL1	060072	RGN	4	-	I	Shift halfword left 1. LS(RH, 1) => R.
SL2	060073	RGN	4	-	I	Shift halfword left 2. LS(RH, 2) => R.
SLE	101220	SKP	-	-	SRV	Skip if A .LE. 0.
SLN	101100	SKP	-	-	SRV	Skip if A bit 16 set.
SLT	101400	SKP	-	-	SRV	OBSOLETE. Skip if A .LT. 0. (Use SMI)
SLZ	100100	SKP	-	-	SRV	Skip if A bit 16 .EQ. 0.
SMCR	100200	SKP	-	-	SRV	Skip if machine check reset.
SMCS	101200	SKP	-	-	SRV	Skip if machine check set.
SMI	101400	SKP	-	-	SRV	Skip if A .LT. 0.
SMK	170020	PIO	-	-	SR*	OBSOLETE. Set interrupt masks. (P100-P300)
SNE	101040	SKP	-	-	SRV	OBSOLETE. Skip if A .NE. 0. (Use SNZ)
SNR	10024-	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch N reset.
SNS	10124-	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch N set.
SNZ	101040	SKP	-	-	SRV	Skip if A .NE. 0.
SOA	140110	GEN	2	1	SRV	OBSOLETE. Subtract 1 from A. A - 1 => A (Use S1A)
SPL	100400	SKP	-	-	SRV	Skip if A .GE. 0.

Mnem	OpCode	Typ	C	cc	Modes	Description
SPN	100200	SKP	-	-	SRV	OBSOLETE. Skip if machine check reset. (Use SMCR)
SPS	101200	SKP	-	-	SRV	OBSOLETE. Skip if machine check set. (Use SMCS)
SR1	060074	RGN	4	-	I	Shift halfword right 1. RS(RH, 1) => R.
SR1	100020	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 1 reset.
SR2	060075	RGN	4	-	I	Shift halfword right 2. RS(RH, 2) => R.
SR2	100010	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 2 reset.
SR3	100004	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 3 reset.
SR4	100002	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 4 reset.
SRC	100001	SKP	-	-	SRV	Skip if CBIT reset.
SS1	101020	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 1 set.
SS2	101010	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 2 set.
SS3	101004	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 3 set.
SS4	101002	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 4 set.
SSC	101001	SKP	-	-	SRV	Skip if CBIT set.
SSM	060042	RGN	-	-	I	Set sign minus. 1 => R(1).
SSM	140500	GEN	-	-	SRV	Set sign of A minus. 1 => A(1).
SSP	060043	RGN	-	-	I	Set sign plus. 0 => R(1).
SSP	140100	GEN	-	-	SRV	Set sign of A plus. 0 => A(1).
SSR	100036	SKP	-	-	SRV*	OBSOLETE. Skip if all sense switches reset.
SSS	101036	SKP	-	-	SRV*	OBSOLETE. Skip if all sense switches set.
SSSN	040310	GEN	6	5	VI	Store system serial number => [XB]16 halfwords.
ST	042---	MR	-	-	I	Store. R => [EA]32.
STA	-10---	MR	-	-	SRV	Store A. A => [EA]16. (Long: -11400)
STAC	001200	AP	-	7	V	Store A if B = [EA]16 (-> CCEQ).
STAR	130---	MR	-	5	I(*)	Store addressed register.
STC	060166	FLD	-	7	I	Store character from RH.
STC	001322	FLD	-	7	V	Store char from A via FAR.
STCD	060137	RGN	-	7	I	Store cond. IF R+1 = [EA]32, R => [EA]32.
STCH	060136	RAP	-	7	I	Store cond. halfwd. IF RL=[EA]16, RH=>[EA]16.
STEX	060027	RAP	6	5	I	Stack extend by R.
STEX	001315	GEN	6	5	V	Stack extend. Extent in L.
STFA	001320	AP	-	-	VI	Store FAR.
STH	062---	MR	-	-	I	Store halfword. RH => [EA]16.
STL	-11414	MR	-	-	V	Store long. L => [EA]32.
STLC	001204	AP	-	7	V	Store L if E = [EA]32 (-> CCEQ).
STLR	-07404	MR	-	5	V(*)	Store long into register(EA).
STPM	000024	MOD	-	-	VI*	Store processor model via XB.
STTM	000510	MOD	6	5	VI	Store process timer at XB. (48 bit)
STX	-32---	MR	-	-	SRV	Store X. X => [EA]16. (No X, Long: -33400)
STY	-73410	MR	-	-	V	Store Y. Y => [EA]16. (No X)
SUB	-16---	MR	2	1	SRV	Subtract. A - [EA]16 => A. (Long: -17400)
SVC	000505	CON	-	-	SRVI	Supervisor call.
SZE	100040	SKP	-	-	SRV	Skip if A .EQ. 0.
TAB	140314	GEN	-	-	V	Transfer A to B. A => B.
TAK	001015	GEN	7	6	V	Transfer A to KEYS.
TAX	140504	GEN	-	-	V	Transfer A to X. A => X.
TAY	140505	GEN	-	-	V	Transfer A to Y. A => Y.
TBA	140604	GEN	-	-	V	Transfer B to A. B => A.
TC	060046	RGN	3	1	I	Two's complement R. -R => R.
TCA	140407	GEN	2	1	SRV	Two's complement A. -A => A.

Mnem	OpCode	Typ	C	cc	Modes	Description
TCH	060047	RGN	3	1	I	Two's complement RH. -RH => RH.
TCL	141210	GEN	2	1	V	Two's complement L. -L => L.
TCNP	1754--	MGR	-	1	I	Test for C null pointer. (32IX, R)
TFLL	001323	FLD	-	-	V	Transfer FLR to L.
TFLR	060163	FLD	-	-	I	Transfer FLR to R.
TKA	001005	GEN	-	-	V	Transfer KEYS to A.
TLFL	001321	FLD	-	-	V	Transfer L to FLR.
TM	1150--	MR	-	1	I	Test memory. ([EA]32::0) => CC.
TMH	1350--	MR	-	1	I	Test memory halfword. ([EA]16::0) => CC.
TRFL	060165	FLD	-	-	I	Transfer R to FLR.
TSTQ	060104	AP	-	7	I	Test queue. # items => RH. empty -> CCEQ.
TSTQ	141757	AP	-	7	V	Test queue. # items => A. empty -> CCEQ.
TXA	141034	GEN	-	-	V	Transfer X to A. X => A.
TYA	141124	GEN	-	-	V	Transfer Y to A. Y => A.
VIRY	000311	IG	5	6	SRVI*	OBSOLETE. Execute verificaton routine.
WAIT	000315	AP	-	-	VI*	Wait on semaphore at AP.
WCS	001600	IG	-	-	RVI*	OBSOLETE. WCS entrances. UII on no WCS.
X	1146--	MR	-	-	I	Exclusive OR. XOR(R, [EA]32) => R. (RI)
XAD	001100	DA	3	1	VI	Decimal add. FAR1 + FAR0 => FAR1.
XBTD	001145	DA	3	5	VI	Convert binary to decimal.
XCA	140104	GEN	-	-	SRV	Exchange & clear A. A => B, 0 => A.
XCB	140204	GEN	-	-	SRV	Exchange & clear B. B => A, 0 => B.
XCM	001102	DA	-	1	VI	Decimal compare.
XDTB	001146	DA	3	5	VI	Convert decimal to binary.
XDV	001107	DA	3	5	VI	Decimal divide. FAR1 / FAR0 => FAR1.
XEC	-03410	MR	-	-	RV	Execute instruction at EA.
XED	001112	DA	-	-	VI	Edit numeric field.
XH	1346--	MR	-	-	I	Excl. OR halfword. XOR(RH, [EA]16) => RH. (RI)
XMP	001104	DA	3	1	VI	Decimal multiply. FAR1 * FAR0 => FAR1.
XMV	001101	DA	3	1	VI	Decimal move.
XVRY	001113	IG	6	5	VI*	OBSOLETE. Verify XIS board. (P500)
ZCM	001117	CS	6	7	VI	Compare char fields.
ZED	001111	CS	-	-	VI	Edit char field.
ZFIL	001116	CS	6	5	VI	Fill string with char. (A(9-16), R2(9-16))
ZM	106---	MR	-	-	I	Zero memory. 0 => [EA]32.
ZMH	126---	MR	-	-	I	Zero memory halfword. 0 => [EA]16.
ZMV	001114	CS	6	5	VI	Copy char field, space fills.
ZMVD	001115	CS	6	5	VI	Copy equal length char fields.
ZTRN	001110	CS	-	-	VI	Copy and translate char string.

7.3. Instruction Set Grouped by Function

7.3.1. Address Pointer Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
ABQ	060134	AP	-	7	I	Add to bottom of Q. CCEQ -> FULL.
ABQ	141716	AP	-	7	V	Add to bottom of Q. CCEQ -> FULL.
ATQ	060135	AP	-	7	I	Add to top of queue. RH => Q. CCEQ -> FULL.

Mnem	OpCode	Typ	C	cc	Modes	Description
ATQ	141717	AP	-	7	V	Add to top of queue. A => Q. CCEQ -> FULL.
CALF	000705	AP	6	5	SRVI	Proc call from faulting proc.
EAFA	001300	AP	-	-	VI	Eff. addr to FAR.
INBC	001217	AP	6	5	VI*	Interrupt ntfy LIFO, clear active interrupt.
INBN	001215	AP	6	5	VI*	Interrupt ntfy LIFO.
INEC	001216	AP	6	5	VI*	Interrupt ntfy FIFO, clear active interrupt.
INEN	001214	AP	6	5	VI*	Interrupt ntfy FIFO.
LIOT	000044	AP	6	5	VI*	Load IOTLB. L, R2 -> target virt addr.
LPSW	000711	AP	7	6	VI*	Load PSW (SN,WN,KEYS,MODALS).
NFYB	001211	AP	6	5	VI*	Notify on sem at AP. LIFO Q.
NFYE	001210	AP	6	5	VI*	Notify on sem at AP. FIFO Q.
RBQ	060133	AP	-	7	I	Remove from bottom of Q. emp -> 0 => RH, CCEQ
RBQ	141715	AP	-	7	V	Remove from bottom of Q. emp -> 0 => A, CCEQ
RRST	000717	AP	-	-	VI	Restore registers (GEN, FLT, XB).
RSAV	000715	AP	-	-	VI	Save registers (GEN, FLT, XB).
RTQ	060132	AP	-	7	I	Remove from top of Q. empty -> 0 => RH, CCEQ
RTQ	141714	AP	-	7	V	Remove from top of Q. empty -> 0 => A, CCEQ
STAC	001200	AP	-	7	V	Store A if B = [EA]16 (-> CCEQ).
STFA	001320	AP	-	-	VI	Store FAR.
STLC	001204	AP	-	7	V	Store L if E = [EA]32 (-> CCEQ).
TSTQ	060104	AP	-	7	I	Test queue. # items => RH. empty -> CCEQ.
TSTQ	141757	AP	-	7	V	Test queue. # items => A. empty -> CCEQ.
WAIT	000315	AP	-	-	VI*	Wait on semaphore at AP.

7.3.2. Branch Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
BCEQ	141602	BR	-	-	VI	Branch on Condition Code .EQ.
BCGE	141605	BR	-	-	VI	Branch on Condition Code .GE.
BCGT	141601	BR	-	-	VI	Branch on Condition Code .GT.
BCLE	141600	BR	-	-	VI	Branch on Condition Code .LE.
BCLT	141604	BR	-	-	VI	Branch on Condition Code .LT.
BCNE	141603	BR	-	-	VI	Branch on Condition Code .NE.
BCR	141705	BR	-	-	VI	Branch on CBIT reset.
BCS	141704	BR	-	-	VI	Branch on CBIT set.
BDX	140734	BR	-	-	V	Branch on decremented X.
BDY	140724	BR	-	-	V	Branch on decremented Y.
BEQ	140612	BR	-	4	V	Branch on A .EQ. 0.
BFEQ	020122	BR	-	4	I	Branch on FAC .EQ. 0.
BFEQ	141612	BR	-	4	V	Branch on FAC .EQ. 0.
BFGE	020125	BR	-	4	I	Branch on FAC .GE. 0.
BFGE	141615	BR	-	4	V	Branch on FAC .GE. 0.
BFGT	020121	BR	-	4	I	Branch on FAC .GT. 0.
BFGT	141611	BR	-	4	V	Branch on FAC .GT. 0.
BFLE	020120	BR	-	4	I	Branch on FAC .LE. 0.
BFLE	141610	BR	-	4	V	Branch on FAC .LE. 0.
BFLT	020124	BR	-	4	I	Branch on FAC .LT. 0.
BFLT	141614	BR	-	4	V	Branch on FAC .LT. 0.
BFNE	020123	BR	-	4	I	Branch on FAC .NE. 0.
BFNE	141613	BR	-	4	V	Branch on FAC .NE. 0.
BGE	140615	BR	-	4	V	Branch on A .GE. 0.
BGT	140611	BR	-	4	V	Branch on A .GT. 0.

Mnem	OpCode	Typ	C	cc	Modes	Description
BHD1	020144	BR	-	-	I	Branch on RH dec by 1. RH - 1 => RH.
BHD2	020145	BR	-	-	I	Branch on RH dec by 2. RH - 2 => RH.
BHD4	020146	BR	-	-	I	Branch on RH dec by 4. RH - 4 => RH.
BHEQ	020112	BR	-	4	I	Branch on RH .EQ. 0.
BHGE	020115	BR	-	4	I	Branch on RH .GE. 0.
BHGT	020111	BR	-	4	I	Branch on RH .GT. 0.
BHI1	020140	BR	-	-	I	Branch on RH incr by 1. RH + 1 => RH.
BHI2	020141	BR	-	-	I	Branch on RH incr by 2. RH + 2 => RH.
BHI4	020142	BR	-	-	I	Branch on RH incr by 4. RH + 4 => RH.
BHLE	020110	BR	-	4	I	Branch on RH .LE. 0.
BHLT	020114	BR	-	4	I	Branch on RH .LT. 0.
BHNE	020113	BR	-	4	I	Branch on RH .NE. 0.
BIX	141334	BR	-	-	V	Branch on incremented X ^= 0.
BIY	141324	BR	-	-	V	Branch on incremented Y ^= 0.
BLE	140610	BR	-	4	V	Branch on A $\leq$ 0.
BLEQ	140702	BR	-	4	V	Branch on L = 0.
BLGE	140615	BR	-	4	V	Branch on L $\geq$ 0.
BLGT	140701	BR	-	4	V	Branch on L > 0.
BLLE	140700	BR	-	4	V	Branch on L $\leq$ 0.
BLLT	140614	BR	-	4	V	Branch on L < 0.
BLNE	140703	BR	-	4	V	Branch on L <> 0.
BLR	141707	BR	-	-	VI	Branch on LINK reset.
BLS	141706	BR	-	-	VI	Branch on LINK set.
BLT	140614	BR	-	4	V	Branch on A .LT. 0.
BMEQ	141602	BR	-	-	VI	Branch on mag-cond L,CC .EQ. (BCEQ)
BMGE	141706	BR	-	-	VI	Branch on mag-cond L,CC .GE. (BLS)
BMGT	141710	BR	-	-	VI	Branch on mag-cond L,CC .GT.
BMLE	141711	BR	-	-	VI	Branch on mag-cond L,CC .LE.
BMLT	141707	BR	-	-	VI	Branch on mag-cond L,CC .LT. (BLR)
BMNE	141603	BR	-	-	VI	Branch on mag-cond L,CC .NE. (BCNE)
BNE	140613	BR	-	4	V	Branch on A .NE. 0.
BRBR	02004-	BR	-	-	I	Branch on R bit reset.
BRBS	02000-	BR	-	-	I	Branch on R bit set.
BRD1	020134	BR	-	-	I	Branch on R dec by 1. R - 1 => R.
BRD2	020135	BR	-	-	I	Branch on R dec by 2. R - 2 => R.
BRD4	020136	BR	-	-	I	Branch on R dec by 4. R - 4 => R.
BREQ	020102	BR	-	4	I	Branch on R .EQ. 0.
BRGE	020105	BR	-	4	I	Branch on R .NE. 0.
BRGT	020101	BR	-	4	I	Branch on R .LE. 0.
BRI1	020130	BR	-	-	I	Branch on R incr by 1. R + 1 => R.
BRI2	020131	BR	-	-	I	Branch on R incr by 2. R + 2 => R.
BRI4	020132	BR	-	-	I	Branch on R incr by 4. R + 4 => R.
BRLE	020100	BR	-	4	I	Branch on R .LT. 0.
BRLT	020104	BR	-	4	I	Branch on R .GT. 0.
BRNE	020103	BR	-	4	I	Branch on R .GE. 0.

7.3.3. Control Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
ARGT	000605	CON	6	5	VI	Argument transfer (used with PCL).
HLT	000000	CON	-	-	SRVI*	Halt computer operation.
IRTC	000603	CON	7	6	VI*	Interrupt return, clear active intrpt.
IRTN	000601	CON	7	6	VI*	Interrupt return.
ITLB	000615	CON	6	5	VI*	Invalidate STLB entry, L, R2 = VADDR.
LPID	000617	CON	-	-	VI*	Load process ID from A(1-12), R2(1-12).
PRTN	000611	CON	7	6	VI	Procedure return.
SVC	000505	CON	-	-	SRVI	Supervisor call.

7.3.4. Character String Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
ZCM	001117	CS	6	7	VI	Compare char fields.
ZED	001111	CS	-	-	VI	Edit char field.
ZFIL	001116	CS	6	5	VI	Fill string with char. (A(9-16), R2(9-16))
ZMV	001114	CS	6	5	VI	Copy char field, space fills.
ZMVD	001115	CS	6	5	VI	Copy equal length char fields.
ZTRN	001110	CS	-	-	VI	Copy and translate char string.

7.3.5. Decimal Arithmetic

Mnem	OpCode	Typ	C	cc	Modes	Description
XAD	001100	DA	3	1	VI	Decimal add. FAR1 + FAR0 => FAR1.
XBTD	001145	DA	3	5	VI	Convert binary to decimal.
XCM	001102	DA	-	1	VI	Decimal compare.
XDTB	001146	DA	3	5	VI	Convert decimal to binary.
XDV	001107	DA	3	5	VI	Decimal divide. FAR1 / FAR0 => FAR1.
XED	001112	DA	-	-	VI	Edit numeric field.
XMP	001104	DA	3	1	VI	Decimal multiply. FAR1 * FAR0 => FAR1.
XMV	001101	DA	3	1	VI	Decimal move.

7.3.6. Field Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
ALFA	001301	FLD	6	-	V	Add L to FAR.
ARFA	060161	FLD	6	-	I	Add R to FAR. FAR + R => FAR.
LDC	060162	FLD	-	7	I	Load char to RH.
LDC	001302	FLD	-	7	V	Load char to A via FAR.
LFLI	001303	FLD	-	-	SRVI	Load FLR immediate.
STC	060166	FLD	-	7	I	Store character from RH.
STC	001322	FLD	-	7	V	Store char from A via FAR.
TFLL	001323	FLD	-	-	V	Transfer FLR to L.
TFLR	060163	FLD	-	-	I	Transfer FLR to R.
TLFL	001321	FLD	-	-	V	Transfer L to FLR.
TRFL	060165	FLD	-	-	I	Transfer R to FLR.

7.3.7. Floating-point Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
DBLE	060106	FOP	-	-	I	Convert single to double fltg pt.
DFCM	060144	FOP	3	1	I	Dbl fltg complement. -DFAC => DFAC.
DFCM	140574	FOP	3	5	SRV	Dbl fltg complement. -DFAC => DFAC.
DRN	040300	FOP	3	5	VI	Double round from quad.
DRNM	140571	FOP	3	5	VI	Double round from quad to -infinity.
DRNP	040301	FOP	3	5	VI	Double round from quad to +infinity.
DRNZ	040302	FOP	3	5	VI	Double round from quad to 0.
FCDQ	140571	FOP	-	-	V	Fltg convert dbl to quad. (P9950)
FCM	060100	FOP	3	1	I	Fltg complement. -FAC => FAC.
FCM	140530	FOP	3	5	RV	Fltg complement. -FAC => FAC.
FDBL	140016	FOP	-	-	V	Fltg convert single to dbl. FAC => DFAC.
FLOT	140550	FOP	6	5	R	Convert int to fltg. Flot(A,B) => FAC w/hole.

Mnem	OpCode	Typ	C	cc	Modes	Description
FLT	060105	FOP	-	-	I	Convert int to fltg. Flot(R) => FAC.
FLTA	140532	FOP	3	5	V	Convert int to fltg. Flot(A) => FAC.
FLTH	060102	FOP	-	-	I	Convert half word int to fltg pt.
FLTL	140535	FOP	8	5	R	Convert long to fltg. Flot(L) => FAC.
FRN	060107	FOP	3	1	I	OBSOLETE. Fltg round. (FRN)
FRN	140534	FOP	3	5	RVI	Fltg round up.
FRNM	060146	FOP	3	5	I	Fltg round towards - infinity.
FRNM	040320	FOP	3	5	V	Fltg round towards - infinity.
FRNP	060145	FOP	3	5	I	Fltg round towards + infinity.
FRNP	040303	FOP	3	5	V	Fltg round towards + infinity.
FRNZ	060147	FOP	3	5	I	Fltg round towards zero.
FRNZ	040321	FOP	3	5	V	Fltg round towards zero.
INT	060103	FOP	-	-	I	Convert fltg to int. INT(FAC) => R.
INT	140554	FOP	3	5	SR	Convert fltg to int. INT(FAC) => A,B w/ hole.
INTA	140531	FOP	3	5	V	Convert fltg to int. INT(FAC) => A.
INTH	060101	FOP	-	-	I	Convert fltg to halfword. INT(FAC) => RH.
INTL	140533	FOP	3	5	V	Convert fltg to int long. INT(FAC) => L.

7.3.8. Floating-point Skip Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
FSGT	140515	FSK	-	1	RV	Fltg skip if .GT. 0.
FSLE	140514	FSK	-	1	RV	Fltg skip if .LE. 0.
FSMI	140512	FSK	-	1	RV	Fltg skip if .LT. 0.
FSNZ	140511	FSK	-	1	RV	Fltg skip if .NE. 0.
FSPL	140513	FSK	-	1	RV	Fltg skip if .GE. 0.
FSZE	140510	FSK	-	1	RV	Fltg skip if .EQ. 0.

7.3.9. Generic Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
A1A	141206	GEN	2	1	SRV	Add 1 to A. A + 1 => A.
A2A	140304	GEN	2	1	SRV	Add 2 to A. A + 2 => A.
ACA	141216	GEN	2	1	SRV	Add CBIT to A. CBIT + A => A.
ADLL	141000	GEN	2	1	V	Add LINK to L.
AOA	141206	GEN	2	1	SRV	OBSOLETE. Add 1 to A. A + 1 => A. (Use A1A)
CAL	141050	GEN	-	-	SRV	Clear left byte of A.
CAR	141044	GEN	-	-	SRV	Clear right byte of A.
CAZ	140214	GEN	1	1	SRV	Skip 0,1,2 if A >,=,< 0.
CEA	000111	GEN	-	-	SR	Compute effective address. EA => A.
CGT	001314	GEN	6	5	V	Computed go to.
CHS	140024	GEN	-	-	SRV	Change sign of A. ^A(1) => A(1).
CMA	140401	GEN	-	-	SRV	One's complement A. ^A => A.
CRA	140040	GEN	-	-	SRV	Clear A. 0 => A.
CRB	140015	GEN	-	-	SRV	Clear B. 0 => B.
CRB	140014	GEN	-	-	SRV	OBSOLETE. Clears B & LSW of DFAC(6). (Use CRB)
CRE	141404	GEN	-	-	V	Clear E. 0 => E.
CRL	140010	GEN	-	-	SRV	Clear L. 0 => L.
CRLE	141410	GEN	-	-	V	Clear L and E. 0 => L, 0 => E.
CSA	140320	GEN	5	-	SRV	Copy sign of A. A(1) => CBIT,0 => A(1).
DRX	140210	GEN	-	-	SRV	Decrement X and skip if 0.
IAB	000201	GEN	-	-	SRV	Exchange A and B. A => B & B => A.
ICA	141340	GEN	-	-	SRV	Interchange bytes of A.

Mnem	OpCode	Typ	C	cc	Modes	Description
ICL	141140	GEN	-	-	SRV	Exchange bytes of A & clr left.
ICR	141240	GEN	-	-	SRV	Exchange bytes of A & clr right.
ILE	141414	GEN	-	-	V	Exchange L and E. L => E & E => L.
INK	000043	GEN	-	-	SR	Input P300 keys into A.
IRX	140114	GEN	-	-	SRV	Increment X and skip if 0.
NOP	000001	GEN	-	-	SRVI	No operation.
NRM	000101	GEN	-	-	SR	OBSOLETE. Normalize A,B as on P300.
OTK	000405	GEN	7	6	SR	Output A to P300 KEYS & S. (TAK in V-mode)
PID	000211	GEN	-	-	SR	Pos for divide. A => L w/ sign ext. & hole.
PIDA	000115	GEN	-	-	V	Pos for int divide. A => L w/ sign extend.
PIDL	000305	GEN	-	-	V	Pos for long divide. L => E w/ sign extend.
PIM	000205	GEN	-	-	SR	Pos after mult. B(2-16) => A(2-16)
PIMA	000015	GEN	3	5	V	Pos after mult. L => A.
PIML	000301	GEN	3	5	V	Pos after mult long. (L,E) => L.
RCB	140200	GEN	9	-	SRVI	Reset CBIT. 0 => CBIT.
RTN	000105	GEN	-	-	SR	OBSOLETE. Return from P300 recur proc.
S1A	140110	GEN	2	1	SRV	Subtract 1 from A. A - 1 => A.
S2A	140310	GEN	2	1	SRV	Subtract 2 from A. A - 2 => A.
SCA	000041	GEN	-	-	SR	OBSOLETE. Load P300 shift count into A.
SCB	140600	GEN	5	-	SRVI	Set CBIT. 1 => CBIT.
SOA	140110	GEN	2	1	SRV	OBSOLETE. Subtract 1 from A. A - 1 => A (Use S1A)
SSM	140500	GEN	-	-	SRV	Set sign of A minus. 1 => A(1).
SSP	140100	GEN	-	-	SRV	Set sign of A plus. 0 => A(1).
SSSN	040310	GEN	6	5	VI	Store system serial number => [XB]16 halfwords.
STEX	001315	GEN	6	5	V	Stack extend. Extent in L.
TAB	140314	GEN	-	-	V	Transfer A to B. A => B.
TAK	001015	GEN	7	6	V	Transfer A to KEYS.
TAX	140504	GEN	-	-	V	Transfer A to X. A => X.
TAY	140505	GEN	-	-	V	Transfer A to Y. A => Y.
TBA	140604	GEN	-	-	V	Transfer B to A. B => A.
TCA	140407	GEN	2	1	SRV	Two's complement A. -A => A.
TCL	141210	GEN	2	1	V	Two's complement L. -L => L.
TKA	001005	GEN	-	-	V	Transfer KEYS to A.
TXA	141034	GEN	-	-	V	Transfer X to A. X => A.
TYA	141124	GEN	-	-	V	Transfer Y to A. Y => A.
XCA	140104	GEN	-	-	SRV	Exchange & clear A. A => B, 0 => A.
XCB	140204	GEN	-	-	SRV	Exchange & clear B. B => A, 0 => B.

7.3.10. Integrity Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
CXCS	001714	IG	-	-	VI*	OBSOLETE. Control extended control store.
EMCM	000503	IG	-	-	SRVI*	OBSOLETE. Enter machine check mode.
LMCM	000501	IG	-	-	SRVI*	Leave machine check mode.
LWCS	001710	IG	-	-	VI	OBSOLETE. Load writable control store.
MDEI	001304	IG	-	-	VI*	OBSOLETE. Mem diag enable interleave.
MDII	001305	IG	-	-	VI*	OBSOLETE. Mem diag inhibit interleave.

Mnem	OpCode	Typ	C	cc	Modes	Description
MDIW	001324	IG	-	-	VI*	OBSOLETE. Mem diag write interleave. L => [E].
MDRS	001306	IG	-	-	VI*	OBSOLETE. Mem diag read syndrome bits.
MDWC	001307	IG	-	-	VI*	OBSOLETE. Mem diag load write control reg.
RMC	000021	IG	-	-	SRVI*	Reset machine check flag.
RMP	000021	IG	-	-	SRVI*	OBSOLETE. Reset machine check flag. (Use RMC)
VIRY	000311	IG	5	6	SRVI*	OBSOLETE. Execute verificaton routine.
WCS	001600	IG	-	-	RVI*	OBSOLETE. WCS entrances. UII on no WCS.
XVRY	001113	IG	6	5	VI*	OBSOLETE. Verify XIS board. (P500)

7.3.11. Input/Output Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
CAI	000411	IO	-	-	SRVI*	OBSOLETE. Clear active interrupt.
ENB	000401	IO	-	-	SRVI*	Enable interrupts.
ENBL	000401	IO	-	-	SRVI*	Enable interrupts (local). (P850)
ENBM	000400	IO	-	-	SRVI*	Enable interrupts (mutual). (P850)
ENBP	000402	IO	-	-	SRVI*	Enable interrupts (process). (P850)
INH	001001	IO	-	-	SRVI*	Inhibit interrupts.
INHL	001001	IO	-	-	SRVI*	Inhibit interrupts (local). (P850)
INHM	001000	IO	-	-	SRVI*	Inhibit interrupts (mutual). (P850)
INHP	001002	IO	-	-	SRVI*	Inhibit interrupts (process). (P850)

7.3.12. Logicize Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
LCEQ	060153	LOG	-	-	I	Load RH if EQ. CCEQ => RH.
LCEQ	141503	LOG	-	-	V	Load A if EQ. CCEQ => A.
LCGE	060154	LOG	-	-	I	Load RH if GE. CCGE => RH.
LCGE	141504	LOG	-	-	V	Load A if GE. CCGE => A.
LCGT	060155	LOG	-	-	I	Load RH if GT. CCGT => RH.
LCGT	141505	LOG	-	-	V	Load A if GT. CCGT => A.
LCLE	060151	LOG	-	-	I	Load RH if LE. CCLE => RH.
LCLE	141501	LOG	-	-	V	Load A if LE. CCLE => A.
LCLT	060150	LOG	-	-	I	Load RH if LT. CCLT => RH.
LCLT	141500	LOG	-	-	V	Load A if LT. CCLT => A.
LCNE	060152	LOG	-	-	I	Load RH if NE. CCNE => RH.
LCNE	141502	LOG	-	-	V	Load A if NE. CCNE => A.
LEQ	060003	LOG	-	4	I	Load RH if R = 0. (R = 0) => RH.
LEQ	140413	LOG	-	4	SRV	If A .EQ. 0, 1 => A, else 0 => A.
LF	060016	LOG	-	4	I	Logicize false. 0 => RH.
LF	140416	LOG	-	5	SRV	Logicize false. 0 => A.
LFEQ	060023	LOG	-	4	I	Load RH if FAC = 0. (FAC = 0) => RH.
LFEQ	141113	LOG	-	4	V	If FAC .EQ. 0, 1 => A, else 0 => A.
LFGE	060024	LOG	-	4	I	Load RH if FAC >= 0. (FAC >= 0) => RH.
LFGE	141114	LOG	-	4	V	If FAC .GE. 0, 1 => A, else 0 => A.
LFGT	060025	LOG	-	4	I	Load RH if FAC > 0. (FAC > 0) => RH.
LFGT	141115	LOG	-	4	V	If FAC .GT. 0, 1 => A, else 0 => A.
LFLE	060021	LOG	-	4	I	Load RH if FAC <= 0. (FAC <= 0) => RH.

Mnem	OpCode	Typ	C	cc	Modes	Description
LFLE	141111	LOG	-	4	V	If FAC .LE. 0, 1 => A, else 0 => A.
LFLT	060020	LOG	-	4	I	Load RH if FAC < 0. (FAC < 0) => RH.
LFLT	141110	LOG	-	4	V	If FAC .LT. 0, 1 => A, else 0 => A.
LFNE	060022	LOG	-	4	I	Load RH if FAC <> 0. (FAC <> 0) => RH.
LFNE	141112	LOG	-	4	V	If FAC .NE. 0, 1 => A, else 0 => A.
LGE	060004	LOG	-	4	I	Load RH if R >= 0. (R >= 0) => RH.
LGE	140414	LOG	-	4	SRV	If A .GE. 0, 1 => A, else 0 => A.
LGT	060005	LOG	-	4	I	Load RH if R > 0. (R > 0) => RH.
LGT	140415	LOG	-	4	SRV	If A .GT. 0, 1 => A, else 0 => A.
LHEQ	060013	LOG	-	4	I	Load RH if RH = 0. (RH = 0) => RH.
LHGE	060004	LOG	-	4	I	Load RH if RH >= 0. (RH >= 0) => RH.
LHGT	060015	LOG	-	4	I	Load RH if RH > 0. (RH > 0) => RH.
LHLE	060011	LOG	-	4	I	Load RH if RH <= 0. (RH <= 0) => RH.
LHLT	060000	LOG	-	4	I	Load RH if RH < 0. (RH < 0) => RH.
LHNE	060012	LOG	-	4	I	Load RH if RH <> 0. (RH <> 0) => RH.
LLE	060001	LOG	-	4	I	Load RH if R <= 0. (R <= 0) => RH.
LLE	140411	LOG	-	4	SRV	If A .LE. 0, 1 => A, else 0 => A.
LLEQ	141513	LOG	-	4	V	If L. EQ. 0, 1 => A, else 0 => A.
LLGE	140414	LOG	-	4	V	If L. GE. 0, 1 => A, else 0 => A.
LLGT	141515	LOG	-	4	V	If L. GT. 0, 1 => A, else 0 => A.
LLLE	141511	LOG	-	4	V	If L. LE. 0, 1 => A, else 0 => A.
LLLT	140410	LOG	-	4	V	If L. LT. 0, 1 => A, else 0 => A.
LLNE	141512	LOG	-	4	V	If L. NE. 0, 1 => A, else 0 => A.
LLT	060000	LOG	-	4	I	Load R if R < 0. (R < 0) => R.
LLT	140410	LOG	-	4	SRV	If A .LT. 0, 1 => A, else 0 => A.
LNE	060002	LOG	-	4	I	Load R if R <> 0. (R <> 0) => R.
LNE	140412	LOG	-	4	SRV	If A .NE. 0, 1 => A, else 0 => A.
LT	060017	LOG	-	5	I	Logic set true. 1 => R.
LT	140417	LOG	-	5	SRV	Logicize true. 1 => A.

7.3.13. Memory reference/General register to register

Mnem	OpCode	Typ	C	cc	Modes	Description
ACP	132---	MGR	-	-	I	Add character pointer. (32IX, RI only, see SCC)
AIP	172---	MGR	2	1	I	Add indirect pointer. (32IX)
LCC	112---	MGR	-	-	I	Load character via char pointer. (32IX, RI)
LIP	152---	MGR	-	-	I	Load indirect pointer. (32IX)
SCC	132---	MGR	-	-	I	Store character via char pointer. (32IX)
TCNP	1754--	MGR	-	1	I	Test for C null pointer. (32IX, R)

7.3.14. Mode Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
DBL	000007	MOD	-	-	SR	Enter double-prec mode.
DXA	000011	MOD	-	-	SRVI	OBSOLETE. Enter 16K sectored mode. (Use E16S)
E16S	000011	MOD	-	-	SRVI	Enter P300 16K sectored mode.
E32I	001010	MOD	-	-	SRVI	Enter P500 32I mode.
E32R	001013	MOD	-	-	SRVI	Enter P300 32K relative mode.
E32S	000013	MOD	-	-	SRVI	Enter P300 32K sectored mode.
E64R	001011	MOD	-	-	SRVI	Enter P300 64K relative mode.
E64V	000010	MOD	-	-	SRVI	Enter P400 64K virtual mode.
ESIM	000415	MOD	-	-	SRVI*	OBSOLETE. Enter standard interrupt mode.

Mnem	OpCode	Typ	C	cc	Modes	Description
EVIM	000417	MOD	-	-	SRVI*	OBSOLETE. Enter vectored interupt mode.
EXA	000013	MOD	-	-	SRVI	OBSOLETE. Enter 32K sectored mode. (Use E32S)
PTLB	000064	MOD	6	5	VI*	Purge TLB (non-IO). L, R2, R3. (CRE first)
RTS	000511	MOD	-	-	VI*	Reset time slice with A, R2.
SGL	000005	MOD	-	-	SR	Enter single-precision mode.
STPM	000024	MOD	-	-	VI*	Store processor model via XB.
STTM	000510	MOD	6	5	VI	Store process timer at XB. (48 bit)

7.3.15. Memory-reference Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
A	004---	MR	2	1	I	Add. R + [EA]32 => R. (RI)
ADD	-14---	MR	2	1	SRV	Add. A + [EA]16 => A. (Long -15400)
ADL	-15414	MR	2	1	V	Add long. L + [EA]32 => L.
AH	024---	MR	2	1	I	Add halfword. RH + [EA]16 => RH. (RI)
ANA	-06---	MR	-	-	SRV	And. AND(A, [EA]16) => A. (Long: -07400)
ANL	-07414	MR	-	-	V	And long. AND(L, [EA]32) => L.
C	142---	MR	1	1	I	Compare R with [EA]32. (RI)
CAS	-22---	MR	1	1	SRV	Skip 0,1,2 if A >,=,< [EA]16. (Long: -23400)
CH	162---	MR	1	1	I	Compare RH with [EA]16. (RI)
CLS	-23414	MR	1	1	V	Skip 0,1,2 if L >,=,< [EA]32.
CREP	-21410	MR	-	-	R	OBS. Call re-ent. proc. P+1 => [S+1]16, EA => P.
D	144---	MR	3	5	I	Divide. (R,R+1)/[EA]32 => R; REM => R + 1. (RI)
DAD	-14---	MR	2	1	SR	Dbl. add. (A,B)+[EA]32 => A,B w/hole. (DP, Long: -15400)
DFA	0352--	MR	3	5	I	Dbl fltg add. DFR + [EA]64 => DFR. (RI)
DFAD	-15410	MR	3	5	RV	Dbl fltg add. DFAC + [EA]64 => DFAC.
DFC	0152--	MR	-	1	I	Dbl fltg compare DFR to [EA]64. (RI)
DFCS	-23410	MR	6	5	RV	Skip 0,1,2 if DFAC >,=,< [EA]64.
DFD	0742--	MR	3	5	I	Dbl fltg divide. DFAC/[EA]64 => DFAC. (RI)
DFDV	-37410	MR	3	5	RV	Dbl fltg divide. DFAC/[EA]64 => DFAC.
DFL	0142--	MR	-	-	I	Dbl fltg load. [EA]64 => DFAC. (RI)
DFLD	-05410	MR	-	-	RV	Dbl fltg load. [EA]64 => DFAC.
DFLX	-33410	MR	-	-	V	Load dbl fltg index. 4*[EA]16 => X. (No X)
DFM	0552--	MR	3	5	I	Dbl fltg multiply. DFAC * [EA]64 => DFAC. (RI)
DFMP	-35410	MR	3	5	RV	Dbl fltg multiply. DFAC * [EA]64 => DFAC.
DFS	0542--	MR	3	5	I	Dbl fltg subtract. DFAC - [EA]64 => DFAC. (RI)
DFSB	-17410	MR	3	5	RV	Dbl fltg subtract. DFAC - [EA]64 => DFAC.
DFST	0342--	MR	-	-	I	Dbl fltg store. DFAC => [EA]64.
DFST	-11410	MR	-	-	RV	Dbl fltg store. DFAC => [EA]64.
DH	164---	MR	3	5	I	Divide halfword. R/[EA]16 => RH; RM => RL. (RI)
DIV	-36---	MR	3	5	SR	Divide. (A,B)31/[EA]16 => A; REM => B. (Long: -37400)

Mnem	OpCode	Typ	C	cc	Modes	Description
DIV	-36---	MR	3	5	V	Divide. L/[EA]16 => A, REM => B. (Long: -37400)
DLD	-04---	MR	-	-	SR	Double load. [EA]32 => A,B. (DP) (Long: -05400)
DM	1540--	MR	-	1	I	Decr memory. [EA]32 - 1 => [EA]32.
DMH	1740--	MR	-	1	I	Decr memory halfword. [EA]16 - 1 => [EA]16.
DSB	-16---	MR	2	1	SR	Dbl subtract. (A,B)-[EA] => A,B w/hole. (DP, Long: -17400)
DST	-10---	MR	-	-	SR	Double store. (A,B) => [EA]32. (DP,Long: -11400)
DVL	-37414	MR	3	5	V	Divide long. (L,E)/[EA]32 => L; REM => E.
EAA	-03404	MR	-	-	R	Eff. addr to A. EA => A.
EAL	-03404	MR	-	-	V	Eff. addr to L. EA => L.
EALB	1144--	MR	-	-	I	Eff. addr to LB. EA => LB.
EALB	-27410	MR	-	-	V	Eff. addr to LB. EA => LB.
EAR	146---	MR	-	-	I	Eff. addr to R. EA => R.
EAXB	1344--	MR	-	-	I	Eff. addr to XB. EA => XB.
EAXB	-25410	MR	-	-	V	Eff. addr to XB. EA => XB.
EIO	070---	MR	-	7	I*	Execute EA as I/O inst. CCEQ -> success.
EIO	-31404	MR	-	7	V*	Execute EA as I/O inst. CCEQ -> success.
ENTR	-03414	MR	-	-	R	OBSOLETE. Enter recursive proc stack.
ERA	-12---	MR	-	-	SRV	Exclusive or. XOR(A, [EA]16) => A.
ERL	-13414	MR	-	-	V	Exclusive or long. XOR(L, [EA]32) => L.
FA	0350--	MR	3	5	I	Fltg add. FAC + [EA]32 => FAC. (RI)
FAD	-15404	MR	3	5	RV	Fltg add. FAC + [EA]32 => FAC.
FC	0150--	MR	-	1	I	Fltg compare FAC with [EA]32. (RI)
FCS	-23404	MR	6	5	RV	Skip 0,1,2 if FAC >,=,< [EA]32. (RI)
FD	0740--	MR	3	5	I	Fltg divide. FAC / [EA]32 => FAC.
FDV	-37404	MR	3	5	RV	Fltg divide. FAC / [EA]32 => FAC.
FL	0140--	MR	-	-	I	Fltg load. [EA]32 => FAC. (RI)
FLD	-05404	MR	-	-	RV	Fltg load. [EA]32 => FAC.
FLX	-33404	MR	-	-	RV	Load fltg index. 2*[EA]16 => X. (No X)
FM	0550--	MR	3	5	I	Fltg multiply. FAC * [EA]32 => FAC. (RI)
FMP	-35404	MR	3	5	RV	Fltg multiply. FAC * [EA]32 => FAC.
FS	0540--	MR	3	5	I	Fltg subtract. FAC - [EA]32 => FAC. (RI)
FSB	-17404	MR	3	5	RV	Fltg subtract. FAC - [EA]32 => FAC.
FST	0340--	MR	3	5	I	Fltg store. FAC => [EA]32.
FST	-11404	MR	3	5	RV	Fltg store. FAC => [EA]32.
I	102---	MR	-	-	I	Interchange R with [EA]32. (R)
IH	122---	MR	-	-	I	Interchange RH with [EA]16. (R)
IM	1140--	MR	-	1	I	Incr memory. [EA]32 + 1 => [EA]32.
IMA	-26---	MR	-	-	SRV	Exchange memory and A. (LONG: -27400)
IMH	1340--	MR	-	1	I	Incr halfword. [EA]16 + 1 => [EA]16.
IRS	-24---	MR	-	-	SRV	Inc, replace, and skip if zero. (Long: -25400)
JDX	-33410	MR	-	-	R	Decrement X & jump if not zero. (No X)
JEQ	-05414	MR	-	-	R	OBSOLETE. Jump if A .EQ. 0, EA => P.
JGE	-17414	MR	-	-	R	OBSOLETE. Jump if A .GE. 0, EA => P.
JGT	-13414	MR	-	-	R	OBSOLETE. Jump if A .GT. 0, EA => P.
JIX	-33414	MR	-	-	R	Increment X & jump if not zero. (No X)
JLE	-11414	MR	-	-	R	OBSOLETE. Jump if A .LE. 0, EA => P.
JLT	-15414	MR	-	-	R	OBSOLETE. Jump if A .LT. 0, EA => P.
JMP	1342--	MR	-	-	I	Jump. EA => P.
JMP	-02---	MR	-	-	SRV	Jump (uncond). EA => PB,P. (Long: -03400)
JNE	-07414	MR	-	-	R	OBSOLETE. Jump if A .NE. 0, EA => P.
JSR	166---	MR	-	-	I	Jump to subr. P => RH, EA32 => P.

Mnem	OpCode	Typ	C	cc	Modes	Description
JST	-20---	MR	-	-	SRV	Jump & store. P => [EA]16, EA+1 => P. (Long: -021400)
JSX	-73414	MR	-	-	RV	Jump & save in X. P => X, EA => P. (No X)
JSXB	1542--	MR	-	-	I	Jump & set XB. P => XB, EA => P.
JSXB	-31410	MR	-	-	V	Jump & set XB. PB => XB, EA => PB.
JSY	-30---	MR	-	-	V	Jump & save in Y. P => Y, EA => P. (Long: -031400)
L	002---	MR	-	-	I	Load R. [EA]32 => R. (RI)
LDA	-04---	MR	-	-	SRV	Load A. [EA]16 => A. (Long: -05400)
LDAR	110---	MR	-	5	I(*)	Load addressed register.
LDL	-05414	MR	-	-	V	Load long. [EA]32 => L.
LDLR	-13404	MR	-	5	V(*)	Load long from addressed reg.
LDX	-72---	MR	-	-	SRV	Load X. [EA]16 => X. (No X, Long: -73414)
LDY	-73404	MR	-	-	V	Load Y. [EA]16 => Y. (No X)
LH	022---	MR	-	-	I	Load halfword. [EA]16 => RH. (RI)
LHL1	010---	MR	-	-	I	Load halfwd shifted by 1. LS([EA]16,1) => RH. (R)
LHL2	030---	MR	-	-	I	Load halfwd shifted by 2. LS([EA]16,2) => RH. (R)
LHL3	072---	MR	-	-	I	Load halfwd shifted by 3. LS([EA]16,3) => RH. (R)
M	104---	MR	-	-	I	Multiply. R * [EA]32 => (R,R+1). (RI)
MH	124---	MR	3	5	I	Multiply halfword. RH * [EA]16 => R. (RI)
MIA	150---	MR	-	-	I	OBSOLETE. Microcode execute A.
MIA	-25404	MR	-	-	V	OBSOLETE. Microcode execute A.
MIB	170---	MR	-	-	I	OBSOLETE. Microcode execute B.
MIB	-27404	MR	-	-	V	OBSOLETE. Microcode execute B.
MPL	-35414	MR	-	-	V	Multiply long. L * [EA]32 => L,E.
MPY	-34---	MR	3	-	V	Multiply. A * [EA]16 => A,B. (Long: -35400)
MPY	-34---	MR	3	-	SR	Multiply. A * [EA]16 => (A,B)31. (Long: -35400)
N	006---	MR	-	-	I	And. AND(R, [EA]32) => R. (RI)
NH	026---	MR	-	-	I	And halfword. AND(RH, [EA]16) => RH. (RI)
O	046---	MR	-	-	I	Or. OR(R, [EA]32) => R. (RI)
OH	066---	MR	-	-	I	Or halfword. OR(RH, [EA]16) => RH. (RI)
ORA	-07410	MR	-	-	V	Or. OR(A, [EA]16) => A.
PCL	1142--	MR	6	5	I	Procedure call.
PCL	-21410	MR	6	5	V	Procedure call.
ROT	050---	MR	4	-	I	Rotate. Shift(R,[EA]16) => R.
S	044---	MR	2	1	I	Subtract. R - [EA]32 => R. (RI)
SBL	-17414	MR	2	1	V	Subtract long. L - [EA]32 => L.
SH	064---	MR	2	1	I	Subtract halfword. RH - [EA]16 => RH. (RI)
SHA	032---	MR	4	-	I	Arithmetic shift. Shift(R,[EA]16) => R.
SHL	012---	MR	4	-	I	Logical shift. Shift(R,[EA]16) => R.
ST	042---	MR	-	-	I	Store. R => [EA]32.
STA	-10---	MR	-	-	SRV	Store A. A => [EA]16. (Long: -11400)
STAR	130---	MR	-	5	I(*)	Store addressed register.
STH	062---	MR	-	-	I	Store halfword. RH => [EA]16.
STL	-11414	MR	-	-	V	Store long. L => [EA]32.
STLR	-07404	MR	-	5	V(*)	Store long into register(EA).
STX	-32---	MR	-	-	SRV	Store X. X => [EA]16. (No X, Long: -33400)
STY	-73410	MR	-	-	V	Store Y. Y => [EA]16. (No X)
SUB	-16---	MR	2	1	SRV	Subtract. A - [EA]16 => A. (Long: -17400)

Mnem	OpCode	Typ	C	cc	Modes	Description
TM	1150--	MR	-	1	I	Test memory. ([EA]32::0) => CC.
TMH	1350--	MR	-	1	I	Test memory halfword. ([EA]16::0) => CC.
X	1146--	MR	-	-	I	Exclusive OR. XOR(R, [EA]32) => R. (RI)
XEC	-03410	MR	-	-	RV	Execute instruction at EA.
XH	1346--	MR	-	-	I	Excl. OR halfword. XOR(RH, [EA]16) => RH. (RI)
ZM	106---	MR	-	-	I	Zero memory. 0 => [EA]32.
ZMH	126---	MR	-	-	I	Zero memory halfword. 0 => [EA]16.

7.3.16. Programmed I/O Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
INA	130---	PIO	-	-	SR*	Input to A.
OCP	030---	PIO	-	-	SR*	Output control pulse.
OTA	170---	PIO	-	-	SR*	Output from A.
SKS	070---	PIO	-	-	SR*	Skip if condition set.
SMK	170020	PIO	-	-	SR*	OBSOLETE. Set interrupt masks. (P100-P300)

7.3.17. Quad Floating Point Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
QFAD	0754--	QAD	3	5	I	Quad fltg add. QAC + [EA]112 => QAC.
QFAD	-13410	QAD	3	5	V	Quad fltg add. QAC + [EA]112 => QAC. (Ext: 2)
QFC	1156--	QAD	-	7	I	Quad floating compare QAF to [EA]112. (RI)
QFCM	140570	QAD	3	5	VI	Quad fltg complement. -QAC => QAC
QFCS	-13410	QAD	6	5	V	Skip 0,1,2 if QAC >,=,< [EA]128. (Ext: 6)
QFDV	1154--	QAD	3	5	I	Quad fltg divide. QAC / [EA]112 => QAC.
QFDV	-13410	QAD	3	5	V	Quad fltg divide. QAC / [EA]112 => QAC. (Ext: 5)
QFLD	0750--	QAD	-	-	I	Quad fltg load. [EA]112/128 => QAC.
QFLD	-13410	QAD	-	-	V	Quad fltg load. [EA]112/128 => QAC. (Ext: 0)
QFLX	-33414	QAD	-	-	V	Quad fltg load index. [EA]*8 => X,Y. (No X)
QFMP	1152--	QAD	3	5	I	Quad fltg multiply. QAC * [EA]112 => QAC.
QFMP	-13410	QAD	3	5	V	Quad fltg mpy. QAC * [EA]112 => QAC. (Ext: 4)
QFSB	0756--	QAD	3	5	I	Quad fltg subtract. QAC - [EA]112 => QAC.
QFSB	-13410	QAD	3	5	V	Quad fltg sub. QAC - [EA]112 => QAC. (Ext: 3)
QFST	0752--	QAD	-	-	I	Quad fltg store. QAC => [EA]128.
QFST	-13410	QAD	-	-	V	Quad fltg store. QAC => [EA]128. (Ext: 1)
QINQ	140572	QAD	3	5	VI	Convert quad to integer.
QIQR	140573	QAD	3	5	VI	Convert quad to integer rounded.

7.3.18. Register AP Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
STCH	060136	RAP	-	7	I	Store cond. halfwd. IF RL=[EA]16, RH=>[EA]16.
STEX	060027	RAP	6	5	I	Stack extend by R.

7.3.19. Register Generic Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
ADLR	060014	RGN	-	7	I	Add LINK to R.
CGT	060026	RGN	6	5	I	Computed go to.
CHS	060040	RGN	-	-	I	Change sign of R. ^R(1) => R(1).
CMH	060045	RGN	-	-	I	Complement RH. ^RH => RH.
CMR	060044	RGN	-	-	I	Complement R. ^R => R.
CR	060056	RGN	-	-	I	Clear R. 0 => R.
CRBL	060062	RGN	-	-	I	Clear R left byte. 0 => R(1-8).
CRBR	060063	RGN	-	-	I	Clear R right byte. 0 => R(9-16).
CRHL	060054	RGN	-	-	I	Clear RH. 0 => RH.
CRHR	060055	RGN	-	-	I	Clear R right halfword. 0 => R(17-32).
CSR	060041	RGN	5	-	I	Copy & save sign. R(1) => C, 0 => R(1).
DCP	060160	RGN	-	-	I	Decrement character pointer. (32IX)
DH1	060130	RGN	2	1	I	Decr RH by 1. RH - 1 => RH.
DH2	060131	RGN	2	1	I	Decr RH by 2. RH - 2 => RH.
DR1	060124	RGN	2	1	I	Decr R by 1. R - 1 => R.
DR2	060125	RGN	2	1	I	Decr R by 2. R - 2 => R.
ICBL	060065	RGN	-	-	I	Exchange bytes. 0 => RH(1-8) => RH(9-16).
ICBR	060066	RGN	-	-	I	Exchange bytes. 0 => RH(9-16) => RH(1-8).
ICHL	060060	RGN	-	-	I	Interchange halfwords. RH => RL, 0 => RH.
ICHR	060061	RGN	-	-	I	Interchange halfwords. RL => RH, 0 => RL.
ICP	060167	RGN	-	-	I	Increment character pointer. (32IX)
IH1	060126	RGN	2	1	I	Incr halfword by 1. RH + 1 => RH.
IH2	060127	RGN	2	1	I	Incr halfword by 2. RH + 2 => RH.
INK	060070	RGN	-	-	I	Input keys to RH.
IR1	060122	RGN	2	1	I	Incr R by 1. R + 1 => R.
IR2	060123	RGN	2	1	I	Incr R by 2. R + 2 => R.
IRB	060064	RGN	-	-	I	Interchange bytes. RH(1-8) <=> RH(9-16).
IRH	060057	RGN	-	-	I	Interchange halves. RH <=> RL.
OTK	060071	RGN	7	6	I	Output keys from RH. [RH] => KEYS.
PID	060052	RGN	-	-	I	Pos for int divide. R => R+1; w/ sign extend.
PIDH	060053	RGN	-	-	I	Pos RH for div. RH => RL; RH(1) => RH(2-16).
PIM	060050	RGN	3	5	I	Pos after int multiply. (R+1) => R.
PIMH	060051	RGN	3	5	I	Pos RH after int multiply. RL => RH.
SHL1	060076	RGN	4	-	I	Shift halfword left 1. LS(RH, 1) => RH.
SHL2	060077	RGN	4	-	I	Shift halfword left 2. LS(RH, 2) => RH.
SHR1	060120	RGN	4	-	I	Shift halfword right 1. RS(RH, 1) => RH.
SHR2	060121	RGN	4	-	I	Shift halfword right 2. RS(RH, 2) => RH.
SL1	060072	RGN	4	-	I	Shift halfword left 1. LS(RH, 1) => R.
SL2	060073	RGN	4	-	I	Shift halfword left 2. LS(RH, 2) => R.
SR1	060074	RGN	4	-	I	Shift halfword right 1. RS(RH, 1) => R.
SR2	060075	RGN	4	-	I	Shift halfword right 2. RS(RH, 2) => R.
SSM	060042	RGN	-	-	I	Set sign minus. 1 => R(1).
SSP	060043	RGN	-	-	I	Set sign plus. 0 => R(1).

Mnem	OpCode	Typ	C	cc	Modes	Description
STCD	060137	RGN	-	7	I	Store cond. IF R+1 = [EA]32, R => [EA]32.
TC	060046	RGN	3	1	I	Two's complement R. -R => R.
TCH	060047	RGN	3	1	I	Two's complement RH. -RH => RH.

7.3.20. Shift Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
ALL	0414--	SH	4	-	SRV	A left logical.
ALR	0416--	SH	4	-	SRV	A left rotate.
ALS	0415--	SH	3	-	SRV	A left shift (arith).
ARL	0404--	SH	4	-	SRV	A right logical.
ARR	0406--	SH	4	-	SRV	A right rotate.
ARS	0405--	SH	4	-	SRV	A right shift (arith).
LGL	0414--	SH	4	5	SRV	OBSOLETE. A left logical. (Use ALL)
LGR	0404--	SH	4	5	SRV	OBSOLETE. A right logical. (Use ARL)
LLL	0410--	SH	4	-	SRV	Long left logical.
LLR	0412--	SH	4	-	SRV	Long left rotate.
LLS	0411--	SH	3	5	SRV	Long left shift. (SR -> B(1) ignored)
LRL	0400--	SH	4	-	SRV	Long right logical.
LRR	0402--	SH	4	-	SRV	Long right rotate.
LRS	0401--	SH	4	-	SRV	Long right shift. (SR -> B(1) ignored)

7.3.21. Skip Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
NOP	101000	SKP	-	-	SRV	No operation (faster on certain machines).
SAR	10026-	SKP	-	-	SRV	Skip if A(n) reset.
SAS	10126-	SKP	-	-	SRV	Skip if A(n) set.
SEQ	100040	SKP	-	-	SRV	OBSOLETE. Skip if A .EQ. 0. (Use SZE)
SGE	100400	SKP	-	-	SRV	OBSOLETE. Skip if A .GE. 0. (Use SPL)
SGT	100220	SKP	-	-	SRV	Skip if A .GT. 0.
SKP	100000	SKP	-	-	SRV	Skip one word.
SLE	101220	SKP	-	-	SRV	Skip if A .LE. 0.
SLN	101100	SKP	-	-	SRV	Skip if A bit 16 set.
SLT	101400	SKP	-	-	SRV	OBSOLETE. Skip if A .LT. 0. (Use SMI)
SLZ	100100	SKP	-	-	SRV	Skip if A bit 16 .EQ. 0.
SMCR	100200	SKP	-	-	SRV	Skip if machine check reset.
SMCS	101200	SKP	-	-	SRV	Skip if machine check set.
SMI	101400	SKP	-	-	SRV	Skip if A .LT. 0.
SNE	101040	SKP	-	-	SRV	OBSOLETE. Skip if A .NE. 0. (Use SNZ)
SNR	10024-	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch N reset.
SNS	10124-	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch N set.
SNZ	101040	SKP	-	-	SRV	Skip if A .NE. 0.
SPL	100400	SKP	-	-	SRV	Skip if A .GE. 0.
SPN	100200	SKP	-	-	SRV	OBSOLETE. Skip if machine check reset. (Use SMCR)
SPS	101200	SKP	-	-	SRV	OBSOLETE. Skip if machine check set. (Use SMCS)
SR1	100020	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 1 reset.
SR2	100010	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 2 reset.

Mnem	OpCode	Typ	C	cc	Modes	Description
SR3	100004	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 3 reset.
SR4	100002	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 4 reset.
SRC	100001	SKP	-	-	SRV	Skip if CBIT reset.
SS1	101020	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 1 set.
SS2	101010	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 2 set.
SS3	101004	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 3 set.
SS4	101002	SKP	-	-	SRV*	OBSOLETE. Skip if sense switch 4 set.
SSC	101001	SKP	-	-	SRV	Skip if CBIT set.
SSR	100036	SKP	-	-	SRV*	OBSOLETE. Skip if all sense switches reset.
SSS	101036	SKP	-	-	SRV*	OBSOLETE. Skip if all sense switches set.
SZE	100040	SKP	-	-	SRV	Skip if A .EQ. 0.

7.3.22. P300 Virtual Memory Operations

Mnem	OpCode	Typ	C	cc	Modes	Description
EPMJ	000217	VM	-	-	SR	OBSOLETE. Enter page mode & jump (P300).
EPMX	000237	VM	-	-	SR	OBSOLETE. Enter page mode & jump to ucode (P300).
ERMJ	000701	VM	-	-	SR	OBSOLETE. Enter restricted mode & jump (P300).
ERMX	000721	VM	-	-	SR	OBS. Enter restr'd mode & jump to ucode (P300).
EVMJ	000703	VM	-	-	SR	OBSOLETE. Enter virtual mode & jump (P300).
EVMX	000723	VM	-	-	SR	OBS. Enter virtual mode & jump to ucode (P300).
LPMJ	000215	VM	-	-	SR	OBSOLETE. Leave page mode & jump (P300).
LPMX	000235	VM	-	-	SR	OBS. Leave page mode & jump to microcode (P300).

8. Operational Procedures

8.1. Front Panel Controls

Switch	Function
POWER	turns power on/off
KEY LOCK	locks/unlocks next 3 switches
MASTER CLEAR	initialize system
REMOTE ENABLE	permits remote access
REMOTE PRIVILEGE	selects remote privilege level
MULTI STREAM	select multiple stream mode (both ISUs)(P850)
ISU 1	select Instruction Stream Unit 1(P850)
ISU 2	select Instruction Stream Unit 2(P850)

8.2. Standard VCP Procedures

8.2.1. Cold start

1. Turn on power to equipment: supervisor terminal, CPU, disk drives, other peripherals.
2. For all machines(from first partition on drive 0, first controller):

   ```
   SYSCLR
   ```

   For Primos:

   ```
   BOOT 14114
   ```

   or, for Primos II:

   ```
   BOOT 10114
   ```

3. For other than first partition:

   ```
   SYSCLR
   ```

   For Primos:

   ```
   BOOT 4114
   PHYSICAL DEVICE=physical_device_number
   ```

   or, for Primos II:

   ```
   BOOT 114
   PHYSICAL DEVICE=physical_device_number
   ```

4. Add '20 to the device number for first partition on second controller ('27). See section 8.3 for other boot switch settings.
5. To bring up Primos from Primos II:

   ```
   PRIMOS [directory_containing_Primos]
   ```

   Option need not be specified if booting from same directory as last time.

For other boot options or devices see the Boot Device Table, section 8.3.

8.2.2. Warm Start

If a warm start is desired to reset a controller while the CPU is still running, hit the ESCAPE key twice to access the VCP/CP and then type STOP.

1. For 50 series machines, type in:

```
SYSCLR
RUN
RUN
```

2. For 9000 and 2000 machines (CP), type in:

```
WARMstart
```

8.2.3. Tape Dump

For 9000s, type:

```
TAPEdump unit
```

For 50 series, type:

Drive 0	*Drive 1*	*Drive 2*	*Drive 3*
`SYSCLR`	`SYSCLR`	`SYSCLR`	`SYSCLR`
`RUN 775`	`RUN 776`	`A 7`	`A 7`
		`775`	`775`
		`/`	`/`
		`SS 2`	`SS 3`
		`RUN`	`RUN`

8.3. Boot Device Settings

8.3.1. Booting from SMDs

H	D	E	F	P	0	unit	1	C	1 100	Storage Module
0						unit	-	B	T	-	101	Magtape
-											111	Pal Boot

Field	*Description*	Octal	Hex
H	Bypass CONFIG file; prompt for COMDEV & PAGING	100000	8000
D	Enable the ring 0 debugger	040000	4000
E	Enter the debugger during coldstart	020000	2000
F	Boot from first partition on drive 0, controller 0	010000	1000
P	Continue boot to PRIMOS	004000	0800
unit	Drive unit number	000600	0180
C	Controller number	000060	0030

Field	*Description*	Octal	Hex
Rel	Relocate boot file to ending address of: 00 - end of physical memory 01 - 16K 10 - 32K 11 - 48K	000600	0180
A	Suppress auto-start of program	000100	0040
B	Halt to allow baud rate change	000040	0020
T	Drive type: 0 - 9-track 1 - 7-track	000020	0010

8.4. Formatting disks: MAKE

To make a new disk from scratch (never on a Prime), use:

MAKE -PART *partition-name* -DISK *pdev* -DT *device-type*
-FMT -NO_INIT -NEW_DISK

To remake an existing pack, use:

MAKE -PART *partition-name* -DISK *pdev* -DT *drive-type*
-NO_INIT

Common device types are:

SMD	80 Mb or 300 Mb removable packs.
MODEL_4475	315 Mb fixed media (dark brown front, Century Data).
MODEL_4735	500 Mb fixed media (pickeral).
MODEL_4845	770 Mb fixed media (beluga).

Commonly needed options:

-SPLIT [*number-of-paging-records*
Split the disk into paging and file system parts. If not supplied, MAKE will ask for the number of paging records.
-IC Make the disk for an intelligent controller (ICOP mode controller). Uses dynamic badspotting.
-AC Make the disk compatible with all controllers. Can not use mirroring.
-ROBUST
Make the partition a robust partition (CAM files only).

For further info, see MAKE in the commands chapter (2.7).

8.5. Disk maintianance: FIX_DISK

To check a disk for damage but do no correcting:

FIX_DISK -DISK *pdev*

To quickly fix quotas or robust partitions (fast mode), use:

FIX_DISK -DISK *pdev* -FIX -FAST

Otherwise, do a normal disk repair:

FIX_DISK -DISK *pdev* -FIX -DUFE -CMPR

For further options, see FIX_DISK in the commands chapter (2.7)

8.6. Adding & changing user configurations: EDIT_PROFILE

To invoke EDIT_PROFILE, enter:

EDIT_PROFILE

Then, to add a user, enter the underlined commands at the appropriate prompts:

> AU *username* -PW *initial-password*
Groups: *system-wide-groups*
Default login project: *default-project*
Password lifetime in days: *number-of-days*
> AU *username* -PROJ *default-project* -PROF
Groups: *project-related-groups*
Initial attach point: *<partition>directory-path*
Create/change user attributes? Y
Number of command levels: *number-of-command-levels*
Number of live program invocations per command level: *number-of-invocations*
Number of private, dynamic segments: *number-of-segments*
Number of private, static segments: *number-of-segments*
> Q

then attach to the partition given and create the user's directory (this may be a subdirectory within another directory):

A *<partition>*MFD
CR *directory-path*
SAC *directory-path user*:ALL $REST:LUR

The ACLs may be changed, the above is a typical setting.

Users may be changed by using the CU command in EDIT_PROFILE and deleted using the DU sub-command. For more information on EDIT_PROFILE see its entry in the commands chapter (2.7).

8.7. VCP Commands

Access {*address* | *register*}[*modes*]
Subcommands:

return
Access next location.

^ Access previous location.

number
Replace this location with *number*.

/ Return to VCP.

AWARMOFF

Don't warmstart on power return. (UPS, 9000 series)

AWARMON

Warmstart on power restore. (UPS, 9000 series)

BOOT *device-number*

Boot the CPU.

BOOTD

Boot CPU to PRIMOS II. (9000 series)

BOOTP

Boot CPU to PRIMOS. (9000 series)

Copy *start end to*

Copy memory block. Copies area between *start* and *end* to area starting at *to*.

DATE

Display the date. (9000 series)

DIRectory [:0 | :1]

Display VCP floppy disk directory contents. (9000 series) Default is last drive displayed.

DISPLAY *address*

Display virtual memory contents. (Only when PRIMOS is running.)

DISPLAYC *address*

Continuously display virtual memory contents. (Only when PRIMOS is running.)

DOS

Restart PRIMOS II after interruption. (9000 series)

Dump {*register* | *start end*}[*modes*]

Display the register or block of memory according to modes (see A).

FETCH

Display data according to previously set sense and data switches.

Fill *start end number*

Fill block of memory from *start* to *end* with *number*.

HALT

Stop the CPU. (9000 series)

HELP

Display list of DP commands. (9000 series)

HISTORY

Invoke history disk editor. (6000 series)
Subcommands ("HST>" prompt):

P *n* Print next *n* entries.

N *n* Move *n* entries from current and print it.

^ Print previous entry.

E Go to last entry.

W Write a comment (max 256 chars). Terminate with '$'.

F Format the history disk floppy. Erases all data.

Q Exit to CP mode.

LDNET [*filename*]

Load a decode net file. (9000 and 6000 series).

Lights

Display the current value of the lights register. Abbreviation may only be used on 9000 series.

LightsC

Display the lights register whenever it changes. Abbreviation may only be used on 9000 series.

listREV

List CPU type, part number and required rev for each CPU board. (9000 series)

MO ABS

Enter absolute addressing mode.

MO BRIEF

Enter limited diagnostic message mode. (9000 series)

MO FULL

Enter full diagnostic message mode. (9000 series)

MO MAP

Enter mapped addressing mode. (Default condition.)

MO RFABS

Enter absolute register set addressing mode.

MO RFCRS

Enter current register set addressing mode.

MO RFH

Specify that high-order half of register is to be modified.

MO RFL

Specify that low-order half of register is to be modified.

MO ST

Place the terminal in supervisor terminal mode.

MO USER

Place the terminal in user terminl mode. (2250, 9000 series)

RCP [*address*]

Run without entering supervisor terminal mode.

REMPWD

Set password on remote port. (9000 series)

RUN [*address*]

Start the CPU running.

SD *number*

Set data switches. Except on 9000 series, this number is destroyed by any successive command that uses a number.

SEtime -*mmddyy* -*hhmmw* [-D]

Set the date and time for the DP. (9000 series) *mmddyy* is month(01-12), day(01-31), year(00-99). *hhmmw* is hour(00-23), minute(00-59), and day of week(1-7, 1 = Sunday). -D enables automatic daylight savings time change(last Sunday of April to last Sunday of October).

SPINDOWN

Instruct 68MB and 158MB (Winchester) drives to spin down. (2250) Must be issued before powering down 2250.

SS

Set sense switches.

STORE

Store specified data according to previously set sense and data switches.

Sysclr

Perform limited master clear. Resets CPU and I/O controllers. (Abbreviation valid only on 9000 series.)

SYSOUT {BUFF | IGN | INT}

Controls output to supervisor terminal. Output is either buffered (BUFF), ignored (IGN) or interleaved with interactive mode (INT). (2000 and 9000 series only.)

TAPEdump *unit*

Causes CPU to dump the current memory image on to the tape on drive *unit*. 9000 series only.

TRACE [*number*]

Single steps CPU for *number* of instructions. (2000 and 9000 series only.)

VIRY

Perform complete system master clear. Resets VCP, CPU and I/O controllers. Verifies VCP and CPU.

VPSD

Enter wired VPSD. VPSD directive must have been in CONFIG, Primos must have been running and machine must be halted. (2000 and 9000 only.) Old 50 series equivalent:

```
SYSCLR
RUN 600
```

WARMstart

Attempt warmstart of Primos. (9000 series.) Other machine equivalent:

```
SYSCLR
RUN
RUN
```

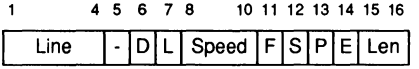
9. Peripheral I/O

9.1. Addresses

Addr	Device	Addr	Device
00	Polling	40	PRIMAD (AIS)
01	Paper Tape Reader	41	Digital Input 1 (DIS)
02	Paper Tape Punch	42	Digital Input 2
03	Unit Record Controller 1	43	Digital Output 1 (DOS)
04	STTY	44	Digital Output 2
05	Unit Record Controller 2	45	Disk Ctrlr (was AOS)
06	Interproc. Channel (IPC)	46	Disk Ctrlr (was CPI)
07	Primenet Node Controller 1	47	PNC 2
10	ICS2 1 or ICS1	50	HSSMLC 1
11	ICS2 2 or ICS1	51	HSSMLC 2 or MDLC
12	Floppy disk	52	AMLC 3 or ICS1
13	Magtape Controller 2	53	AMLC 2
14	Magtape Controller 1	54	AMLC 1
15	AMLC 5 or ICS1	55	MACI Autocall
16	AMLC 6 or ICS1	56	SMLC
17	AMLC 7 or ICS1	57	--
20	Panel, Real Time Clock	60	Gen. Purp. IF Board
21	Disk option B' (4002)	61	GPIB
22	Disk Controller 3	62	GPIB
23	Disk Controller 4	63	GPIB
24	Disk Controller (was WCS)	64	GPIB
25	Disk Controller (was 4000)	65	GPIB
26	Disk Controller 1	66	GPIB
27	Disk Controller 2	67	GPIB
30	IOC 1 (Parallel I/O)	70	GPIB Test
31	IOC 2	71	ADAGE GP/400 IF
32	AMLC 8 or ICS1	72	--
33	Versatec	73	--
34	Versatec	74	--
35	AMLC 4 or ICS1	75	--
36	ICS1 1	76	--
37	ICS1 2	77	I/O Bus Test

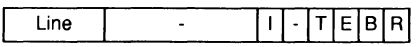
9.2. AMLC

9.2.1. OTA 01 -- Set Line Configuration



Field	Description	Octal	Hex
Line	Line Number	170000	F000
D	Data Set Control	002000	0400
L	Loop Line	001000	0200
Speed	Line Speed:	000700	01C0
	0 - 110 BAUD		
	1 - 134.5		
	2 - 300		
	3 - 1200		
	4 - Programed Clock		
	5 - Strap 1 (75)		
	6 - Strap 2 (150)		
	7 - Strap 3 (1800)		
F	ICS: reverse flow control; AMLC: Unused	000040	0020
S	Stop bits: 0 - 1, 1 - 2	000020	0010
P	Parity: 1 - Disable Parity	000010	0008
E	Parity: 0 - Odd Parity, 1 - Even	000004	0004
Len	Character Length:	000003	0003
	0 - 5 Bits		
	1 - 7 Bits		
	2 - 6 Bits		
	3 - 8 Bits		

9.2.2. OTA 02 -- Set Line Control



Field	Description	Octal	Hex
Line	Line Number	170000	F000
I	Interrupt: 1 - Char at a time	000040	0020
T	Transmit: 1 - Enable	000010	0008
E	Echo Back: 1 - Enable	000004	0004
B	Receive: 1 - Off, Report Break	000002	0002
R	Receive: 1 - Enable	000001	0001

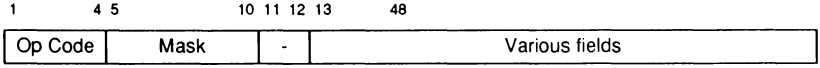
9.3. ASR

BAUD	OPTION-A	SOC CTL 1	SOC CTL 2
110	110	27	740**
300	1010	76	340**
1200	2010	373	340**
9600	3410	3735	340**

** = number of delays used by BOOT, PRIMOS

9.4. DISK CONTROLLERS

9.4.1. Disk Channel Program Definitions

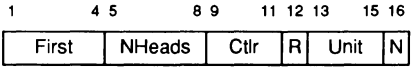


Mnem	Op Code	Ex time (µsec)	Command	Field Num	Field use
DHLT	0	6	Halt		
SFORM	2	-	Format	13-16	Rec Size
				23-32	Track Addr
				33-40	# Records
				44-48	Head Addr
SSEEK	3	7.5	Seek	17	Restore
				18	Clear
				23-32	Track Addr
DSEL	4	7.5	Select	29-32	MHD
SREAD	5	-	Read	13-16	Rec Size
				17-20	Offset
				21	SR
				23-32	Track Addr
				33-40	Rec Addr
				44-48	Head Addr
SWRITE	6	-	-	13-16	Rec Size
				23-32	Track Addr
				33-40	Rec Addr
				44-48	Head Addr
DSTALL	7	210	Stall		
DSTAT	9	9	Input Status	17-32	Mem Addr
SSTOR	A	9	Store	16	Diag Addr
				17-32	Mem Addr
DOAR	B	9	Input OAR	16	Mem Addr
SLOAD	C	9	Load	16	Diag Addr
				17-32	Mem Addr
SDMA	D	6	Channel Address	13-16	Chain
				17-32	Chan Addr
DINT	E	6+CPU	Interrupt	17-32	Vect Addr
DTRAN	F	6	Transfer	17-32	Trans Addr

Bit	Description	Octal	Hex
5	If 0, do not execute inst if: If 1, execute inst if:	004000	0400
6	No function. Reserved for "selected diskfile is write protected."	002000	0200
7	Last read or write record inst caused a DMA overrun, check error, controller parity error or header check failure (status word bits 2,4,5, or 6 set).	001000	0100
8	Selected MHD is seeking.	000400	0080
9	Selected diskfile has an error condition (status word bits 14, 15, or 16 are set).	000200	0040
10	For dual port operation only. Selected diskfile is busy servicing the "other" controller.	000100	0020

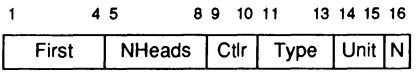
9.5. Disk Device Numbers (PDEV)

Rev 21.0:



Field	Description	Octal	Hex
First	(Offset to First Head)/2	170000	F000
NHeads	(Number of Heads)/2	007400	0F00
Ctlr	Controller: 0 - ('24) 1 - ('26) 2 - ('25) 3 - ('22) 4 - ('45) 5 - ('27) 6 - ('46) 7 - ('23)	000340	00E0
R	Reserved. Must be 1.	000020	0010
Unit	Unit (Inc. bit 16 for Diskette)	000016	000E
N	LSB of Number Heads	000001	0001

Pre-rev 21.0:



Field	Description	Octal	Hex
First	(Offset to First Head)/2	170000	F000
NHeads	(Number of Heads)/2	007400	0F00
Ctlr	Controller: 0 - 1 ('26) 1 - 3 ('22) 2 - 2 ('27) 3 - 4 ('23)	000300	00C0
Type	Type of Controller: 0 - 4000 MHD 1 - 4000 FHD 2 - Diskette 3 - 4003 8 Sectors/Track 4 - 4003 FHD 5 - 4003 32 Sectors/Track 6 - 4004 Storage Module 7 - Reserved	000070	0038
Unit	Unit (Inc. bit 16 for Diskette)	000006	0006
N	Diskette: Low Bit of Unit Storage Module: LSB of Number Heads	000001	0001

9.6. Disk Errors

9.6.1. Diskette Controller

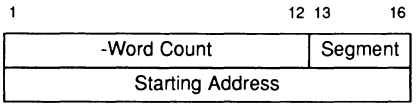
Field	Description	Octal	Hex
-	Bad Record Identifier	177777	FFFF
-	Device Not Ready	177776	FFFE
1	Normal End of Instruction	100000	8000
2	Sector Not Found	040000	4000
3	Checksum Error on Sector ID	020000	2000
4	Track Error (head misposition)	010000	1000
5	Bad OTA or Not Ready	004000	0800
6	Deleted Data Mark Read	002000	0400
7	DMx Overrun	001000	0200
8	Chksum err, Write Prot. Violation, Inoperable on Write or Format	000400	0100
9-15	Unused	000376	00FE
16	Not Ready	000001	0001
-	Redundant Int. (Warm Start)	000000	0000

9.6.2. Storage Module (4004 Controller)

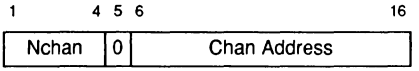
Field	Description	Octal	Hex
-	Bad Record Identifier	177777	FFFF
-	Device Not Ready (DOS)	177776	FFFE
-	Memory Parity Error During DMx	177775	FFFD
-	No controller	177774	FFFC
-	Hung controller	177773	FFFB
1	Bit 1 Always On	100000	8000
2	DMA Overrun	040000	4000
3	Write Protect	020000	2000
4	Read Check	010000	1000
5	Data Parity Error	004000	0800
6	Header Check	002000	0400
7-10	Unused	001700	0360
11	Busy(Dual Port Only)	000040	0020
12	Unused	000020	0010
13	Seeking	000010	0008
14	Illegal Seek	000004	0004
15	Select Error	000002	0002
16	Not Ready (hardware)	000001	0001
-	Redundant Int. (Warm Start)	000000	0000

9.7. DMx control words

9.7.1. DMA

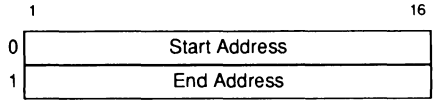


OTA '14dd:

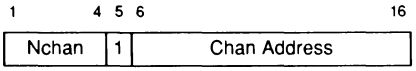


Nchan = Number of channels - 1.

9.7.2. DMC

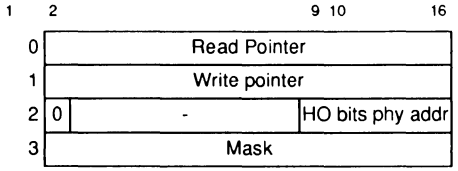


OTA '14dd:



Nchan = Number of channels - 1.

9.7.3. DMQ



Mask = length (of Queue) - 1.
length = 2^K $(4 \leq K \leq 10)$
(Queue must be on 2^K boundary.)

INPUT: End of Range if no room.
OUTPUT: EOR if empty (not w/last entry).

9.7.4. DMT

Device Defined.

9.8. Magtape

9.8.1. Command Bit Definitions

Field	Description	Octal	Hex
1	Select Transport (bits 9-12)	100000	8000
2	0=>File Operation, 1=>Record Op	040000	4000
3	0=>Read/Write Op, 1=>Spacing Op	020000	2000
4	1=>9-Track Read and Correct	010000	1000
5	0=>Binary, 1=>BCD (7-track only)	004000	0800
6	0=>7-Track Transport, 1=>9-Track	002000	0400
7	Unused	001000	0200
8	1=>2 Characters per Word	000400	0100
9	1=>Forward Motion (bits 10,11=0)	000200	0080
10	1=>Reverse Motion (bits 9,11,12=0)	000100	0040
11	1=>Rewind (bits 9,10,12=0)	000040	0020
12	1=>Write Order	000020	0010
13	Select Transport 0	000010	0008
14	Select Transport 1	000004	0004
15	Select Transport 2	000002	0002
16	Select Transport 3	000001	0001

9.8.2. Magtape Commands

Octal	Hex	Description
100000	8000	Select Transport (7 and 9 track)
000040	0020	Rewind to BOT (7 and 9 track)
022100	2440	Backspace File Mark, 9-track
020100	2040	Backspace File Mark, 7-track
062100	6440	Backspace Record, 9-track
060100	6040	Backspace Record, 7-track
022220	2490	Write File Mark, 9-track
020220	2090	Write File Mark, 7-track
062200	6480	Forward Space Record, 9-track
060200	6080	Forward Space Record, 7-track
022200	2480	Forward Space File Mark, 9-track

Octal	Hex	Description
020200	2080	Forward Space File Mark, 7-track
042220	4490	Write Record One Char/Word, 9-track
042620	4590	Write Record Two Char/Word, 9-track
042200	4490	Read Record One Char/Word, 9-track
042600	4580	Read Record Two Char/Word, 9-track
052200	5480	Read/Correct Record One Char/Word, 9-track
052600	5580	Read/Correct Record Two Char/Word, 9-track
040220	4090	Write Binary Record One Char/Word, 7-track
040620	4190	Write Binary Record Two Char/Word, 7-track
044220	4890	Write BCD Record One Char/Word, 7-track
044620	4990	Write BCD Record Two Char/Word, 7-track
040200	4080	Read Binary Record One Char/Word, 7-track
040600	4180	Read Binary Record Two Char/Word, 7-track
044200	4880	Read BCD Record One Char/Word, 7-track
044600	4980	Read BCD Record Two Char/Word, 7-track
140000	C000	Return controler ID
100020	8010	Erase 3 inch gap (vers. 2 or 3 controller)
100040	8020	Unload; rewind and plac offline (2, 3)
100060	8030	Set density to 800 bpi (2 only)
100100	8040	Set density to 1600 bpi (2, 3)
100120	8050	Set density to 6250 bpi (3 only)
100140	8060	Enable front panel density select (3)
100160	8070	Set speed to 25 IPS (future)
100200	8080	Set speed to 100 IPS (future)
043500	4740	Read record backwards (3 only)

9.8.3. Magtape Status

Field	Description	Octal	Hex
1	Parity Error	100000	8000
2	Runaway Tape	040000	4000
3	CRC Error	020000	2000
4	LRC Error	010000	1000
5	Low DMx Range	004000	0800
6	Permanent Error	002000	0400
7	Read-After-Write (RAW) Error	001000	0200
8	File Mark Detected	000400	0100
9	Ready	000200	0080
10	Online	000100	0040
11	End of Tape Detected	000040	0020
12	Rewinding	000020	0010
13	Beginning of Tape (at Load Point)	000010	0008
14	Tape is Write-Protected	000004	0004
15	DMx Overrun	000002	0002
16	Rewind Complete	000001	0001

Normal Completion: 000300 or 000304 (00C0 or 00C4)

9.9. PROGRAMMED I/O (PIO)

9.9.1. OCP -- Output Control Pulse
03FFDD FF=Function, DD=Device Address

9.9.2. SKS -- Skip on Condition
07CCDD CC=Condition, DD=Device Address

9.9.3. INA -- Input to A-Register
13FFDD FF=Function, DD=Device Address

No skip for device '20 Always skips if status register input.

9.9.4. OTA -- Output from A=Register
17FFDD FF=Function, DD=Device Address

No skip if device '20.

9.9.5. Standard Functions

FF	OCP	SKS	INA	OTA
00		Ready	Data Reg	
01		Not Busy		
02				
03				
04		Not Interrupting		
05				
06				
07				
10				
11			Input ID	
12	Normal Mode			
13	Diagnostic Mode			
14	Ack Interrupt			DMx Channel
15	Set Int Mask			
16	Reset Int Mask			Int Vect Addr
17	Initialize			

9.10. RS-232-C pin-outs

Pin	Abbrev	Description	Source
1	FG	Protective (frame) ground	
2	TxD	Transmitted data	DTE
3	RxD	Received data	DCE
4	RTS	Request To Send	DTE
5	CTS	Clear To Send	DCE
6	DSR	Data Set Ready	DCE
7	SG	Signal ground	
8	CD	Data Carrier Detect	DCE
9	-	Reserved for test	
10	-	Reserved for test	
11	-	Unassigned	
12	SCD	Sec. Carrier Detect	DCE
13	SCTS	Sec. Clear to Send	DCE
14	STxD	Sec. Transmitted Data	DTE
15	TxC	Trans. Signal Element Timing	DCE
16	SRxD	Sec. Received Data	DCE
17	RxC	Rec. Signal Element	DTE
18	-	Unassigned	
19	SRTS	Sec. Request to Send	DTE
20	DTR	Data Terminal Ready	DTE
21	SQ	Data Signal Quality	
22	RI	Ring Indicator	DCE
23	-	Data Rate Selector	
24	ETxC	Trans. Signal Element Timing	DTE
25	-	Unassigned	

Appendix A
ASCII character set

F Valid file name character
R Reserved conmand line character
^ Control key depressed

Octal	Octal Left	Hex	Dec	Char	Octal	Octal Left	Hex	Dec	Use
000	0000	00	0	**NUL ^@**	200	1000	80	128	
001	0004	01	1	**SOH ^A**	201	1004	81	129	
002	0010	02	2	**STX ^B**	202	1010	82	130	
003	0014	03	3	**ETX ^C**	203	1014	83	131	
004	0020	04	4	**EOT ^D**	204	1020	84	132	
005	0024	05	5	**ENQ ^E**	205	1024	85	133	
006	0030	06	6	**ACK ^F**	206	1030	86	134	
007	0034	07	7	**BEL ^G**	207	1034	87	135	
010	0040	08	8	**BS ^H**	210	1040	88	136	
011	0044	09	9	**HT ^I**	211	1044	89	137	
012	0050	0A	10	**LF ^J**	212	1050	8A	138	
013	0054	0B	11	**VT ^K**	213	1054	8B	139	
014	0060	0C	12	**FF ^L**	214	1060	8C	140	
015	0064	0D	13	**CR ^M**	215	1064	8D	141	
016	0070	0E	14	**LS1 ^N**	216	1070	8E	142	
017	0074	0F	15	**LS0 ^O**	217	1074	8F	143	
020	0100	10	16	**DLE ^P**	220	1100	90	144	
021	0104	11	17	**DC1 ^Q**	221	1104	91	145	
022	0110	12	18	**DC2 ^R**	222	1110	92	146	
023	0114	13	19	**DC3 ^S**	223	1114	93	147	
024	0120	14	20	**DC4 ^T**	224	1120	94	148	
025	0124	15	21	**NAK ^U**	225	1124	95	149	
026	0130	16	22	**SYN ^V**	226	1130	96	150	
027	0134	17	23	**ETB ^W**	227	1134	97	151	
030	0140	18	24	**CAN ^X**	230	1140	98	152	
031	0144	19	25	**EM ^Y**	231	1144	99	153	
032	0150	1A	26	**SUB ^Z**	232	1150	9A	154	
033	0154	1B	27	**ESC ^[**	233	1154	9B	155	
034	0160	1C	28	**FS ^**	234	1160	9C	156	
035	0164	1D	29	**GS ^]**	235	1164	9D	157	
036	0170	1E	30	**RS ^^**	236	1170	9E	158	
037	0174	1F	31	**US ^_**	237	1174	9F	159	
040	0200	20	32	*Space*	240	1200	A0	160	
041	0204	21	33	**!**	241	1204	A1	161	R
042	0210	22	34	**"**	242	1210	A2	162	R
043	0214	23	35	**#**	243	1214	A3	163	F
044	0220	24	36	**$**	244	1220	A4	164	F
045	0224	25	37	**%**	245	1224	A5	165	R
046	0230	26	38	**&**	246	1230	A6	166	FR
047	0234	27	39	**'**	247	1234	A7	167	R
050	0240	28	40	**(**	250	1240	A8	168	R
051	0244	29	41	**)**	251	1244	A9	169	R
052	0250	2A	42	*****	252	1250	AA	170	F
053	0254	2B	43	**+**	253	1254	AB	171	R
054	0260	2C	44	**,**	254	1260	AC	172	R
055	0264	2D	45	**-**	255	1264	AD	173	FR
056	0270	2E	46	**.**	256	1270	AE	174	F
057	0274	2F	47	**/**	257	1274	AF	175	F
060	0300	30	48	**0**	260	1300	B0	176	F
061	0304	31	49	**1**	261	1304	B1	177	F
062	0310	32	50	**2**	262	1310	B2	178	F
063	0314	33	51	**3**	263	1314	B3	179	F
064	0320	34	52	**4**	264	1320	B4	180	F
065	0324	35	53	**5**	265	1324	B5	181	F
066	0330	36	54	**6**	266	1330	B6	182	F

Octal	Octal Left	Hex	Dec	Char	Octal	Octal Left	Hex	Dec	Use
067	0334	37	55	7	267	1334	B7	183	F
070	0340	38	56	8	270	1340	B8	184	F
071	0344	39	57	9	271	1344	B9	185	F
072	0350	3A	58	:	272	1350	BA	186	R
073	0354	3B	59	;	273	1354	BB	187	R
074	0360	3C	60	<	274	1360	BC	188	
075	0364	3D	61	=	275	1364	BD	189	R
076	0370	3E	62	>	276	1370	BE	190	
077	0374	3F	63	?	277	1374	BF	191	
100	0400	40	64	@	300	1400	C0	192	R
101	0404	41	65	A	301	1404	C1	193	F
102	0410	42	66	B	302	1410	C2	194	F
103	0414	43	67	C	303	1414	C3	195	F
104	0420	44	68	D	304	1420	C4	196	F
105	0424	45	69	E	305	1424	C5	197	F
106	0430	46	70	F	306	1430	C6	198	F
107	0434	47	71	G	307	1434	C7	199	F
110	0440	48	72	H	310	1440	C8	200	F
111	0444	49	73	I	311	1444	C9	201	F
112	0450	4A	74	J	312	1450	CA	202	F
113	0454	4B	75	K	313	1454	CB	203	F
114	0460	4C	76	L	314	1460	CC	204	F
115	0464	4D	77	M	315	1464	CD	205	F
116	0470	4E	78	N	316	1470	CE	206	F
117	0474	4F	79	O	317	1474	CF	207	F
120	0500	50	80	P	320	1500	D0	208	F
121	0504	51	81	Q	321	1504	D1	209	F
122	0510	52	82	R	322	1510	D2	210	F
123	0514	53	83	S	323	1514	D3	211	F
124	0520	54	84	T	324	1520	D4	212	F
125	0524	55	85	U	325	1524	D5	213	F
126	0530	56	86	V	326	1530	D6	214	F
127	0534	57	87	W	327	1534	D7	215	F
130	0540	58	88	X	330	1540	D8	216	F
131	0544	59	89	Y	331	1544	D9	217	F
132	0550	5A	90	Z	332	1550	DA	218	F
133	0554	5B	91	[	333	1554	DB	219	R
134	0560	5C	92	\	334	1560	DC	220	R
135	0564	5D	93	]	335	1564	DD	221	R
136	0570	5E	94	^	336	1570	DE	222	R
137	0574	5F	95		337	1574	DF	223	F
140	0600	60	96	`	340	1600	E0	224	R
141	0604	61	97	a	341	1604	E1	225	
142	0610	62	98	b	342	1610	E2	226	
143	0614	63	99	c	343	1614	E3	227	
144	0620	64	100	d	344	1620	E4	228	
145	0624	65	101	e	345	1624	E5	229	
146	0630	66	102	f	346	1630	E6	230	
147	0634	67	103	g	347	1634	E7	231	
150	0640	68	104	h	350	1640	E8	232	
151	0644	69	105	i	351	1644	E9	233	
152	0650	6A	106	j	352	1650	EA	234	
153	0654	6B	107	k	353	1654	EB	235	
154	0660	6C	108	l	354	1660	EC	236	
155	0664	6D	109	m	355	1664	ED	237	
156	0670	6E	110	n	356	1670	EE	238	
157	0674	6F	111	o	357	1674	EF	239	
160	0700	70	112	p	360	1700	F0	240	
161	0704	71	113	q	361	1704	F1	241	
162	0710	72	114	r	362	1710	F2	242	
163	0714	73	115	s	363	1714	F3	243	
164	0720	74	116	t	364	1720	F4	244	
165	0724	75	117	u	365	1724	F5	245	
166	0730	76	118	v	366	1730	F6	246	

Octal	Octal Left	Hex	Dec	Char	Octal	Octal Left	Hex	Dec	Use
167	0734	77	119	**w**	367	1734	F7	247	
170	0740	78	120	**x**	370	1740	F8	248	
171	0744	79	121	**y**	371	1744	F9	249	
172	0750	7A	122	**z**	372	1750	FA	250	
173	0754	7B	123	{	373	1754	FB	251	R
174	0760	7C	124	|	374	1760	FC	252	R
175	0764	7D	125	}	375	1764	FD	253	R
176	0770	7E	126	~	376	1770	FE	254	R
177	0774	7F	127	**DEL**	377	1774	FF	255	

Appendix B
Conversion tables

B.1. Octal-Decimal Conversion Table

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14	15
20	16	17	18	19	20	21	22	23
30	24	25	26	27	28	29	30	31
40	32	33	34	35	36	37	38	39
50	40	41	42	43	44	45	46	47
60	48	49	50	51	52	53	54	55
70	56	57	58	59	60	61	62	63
100	64	65	66	67	68	69	70	71
110	72	73	74	75	76	77	78	79
120	80	81	82	83	84	85	86	87
130	88	89	90	91	92	93	94	95
140	96	97	98	99	100	101	102	103
150	104	105	106	107	108	109	110	111
160	112	113	114	115	116	117	118	119
170	120	121	122	123	124	125	126	127
200	128	129	130	131	132	133	134	135
210	136	137	138	139	140	141	142	143
220	144	145	146	147	148	149	150	151
230	152	153	154	155	156	157	158	159
240	160	161	162	163	164	165	166	167
250	168	169	170	171	172	173	174	175
260	176	177	178	179	180	181	182	183
270	184	185	186	187	188	189	190	191
300	192	193	194	195	196	197	198	199
310	200	201	202	203	204	205	206	207
320	208	209	210	211	212	213	214	215
330	216	217	218	219	220	221	222	223
340	224	225	226	227	228	229	230	231
350	232	233	234	235	236	237	238	239
360	240	241	242	243	244	245	246	247
370	248	249	250	251	252	253	254	255
400	256	257	258	259	260	261	262	263
410	264	265	266	267	268	269	270	271
420	272	273	274	275	276	277	278	279
430	280	281	282	283	284	285	286	287
440	288	289	290	291	292	293	294	295
450	296	297	298	299	300	301	302	303
460	304	305	306	307	308	309	310	311
470	312	313	314	315	316	317	318	319
500	320	321	322	323	324	325	326	327
510	328	329	330	331	332	333	334	335
520	336	337	338	339	340	341	342	343
530	344	345	346	347	348	349	350	351
540	352	353	354	355	356	357	358	359
550	360	361	362	363	364	365	366	367
560	368	369	370	371	372	373	374	375
570	376	377	378	379	380	381	382	383
600	384	385	386	387	388	389	390	391
610	392	393	394	395	396	397	398	399
620	400	401	402	403	404	405	406	407
630	408	409	410	411	412	413	414	415
640	416	417	418	419	420	421	422	423
650	424	425	426	427	428	429	430	431
660	432	433	434	435	436	437	438	439
670	440	441	442	443	444	445	446	447
700	448	449	450	451	452	453	454	455
710	456	457	458	459	460	461	462	463

	0	1	2	3	4	5	6	7
720	464	465	466	467	468	469	470	471
730	472	473	474	475	476	477	478	479
740	480	481	482	483	484	485	486	487
750	488	489	490	491	492	493	494	495
760	496	497	498	499	500	501	502	503
770	504	505	506	507	508	509	510	511
1000	512	513	514	515	516	517	518	519
1010	520	521	522	523	524	525	526	527
1020	528	529	530	531	532	533	534	535
1030	536	537	538	539	540	541	542	543
1040	544	545	546	547	548	549	550	551
1050	552	553	554	555	556	557	558	559
1060	560	561	562	563	564	565	566	567
1070	568	569	570	571	572	573	574	575
1100	576	577	578	579	580	581	582	583
1110	584	585	586	587	588	589	590	591
1120	592	593	594	595	596	597	598	599
1130	600	601	602	603	604	605	606	607
1140	608	609	610	611	612	613	614	615
1150	616	617	618	619	620	621	622	623
1160	624	625	626	627	628	629	630	631
1170	632	633	634	635	636	637	638	639
1200	640	641	642	643	644	645	646	647
1210	648	649	650	651	652	653	654	655
1220	656	657	658	659	660	661	662	663
1230	664	665	666	667	668	669	670	671
1240	672	673	674	675	676	677	678	679
1250	680	681	682	683	684	685	686	687
1260	688	689	690	691	692	693	694	695
1270	696	697	698	699	700	701	702	703
1300	704	705	706	707	708	709	710	711
1310	712	713	714	715	716	717	718	719
1320	720	721	722	723	724	725	726	727
1330	728	729	730	731	732	733	734	735
1340	736	737	738	739	740	741	742	743
1350	744	745	746	747	748	749	750	751
1360	752	753	754	755	756	757	758	759
1370	760	761	762	763	764	765	766	767
1400	768	769	770	771	772	773	774	775
1410	776	777	778	779	780	781	782	783
1420	784	785	786	787	788	789	790	791
1430	792	793	794	795	796	797	798	799
1440	800	801	802	803	804	805	806	807
1450	808	809	810	811	812	813	814	815
1460	816	817	818	819	820	821	822	823
1470	824	825	826	827	828	829	830	831
1500	832	833	834	835	836	837	838	839
1510	840	841	842	843	844	845	846	847
1520	848	849	850	851	852	853	854	855
1530	856	857	858	859	860	861	862	863
1540	864	865	866	867	868	869	870	871
1550	872	873	874	875	876	877	878	879
1560	880	881	882	883	884	885	886	887
1570	888	889	890	891	892	893	894	895
1600	896	897	898	899	900	901	902	903
1610	904	905	906	907	908	909	910	911
1620	912	913	914	915	916	917	918	919
1630	920	921	922	923	924	925	926	927
1640	928	929	930	931	932	933	934	935
1650	936	937	938	939	940	941	942	943
1660	944	945	946	947	948	949	950	951
1670	952	953	954	955	956	957	958	959
1700	960	961	962	963	964	965	966	967
1710	968	969	970	971	972	973	974	975
1720	976	977	978	979	980	981	982	983
1730	984	985	986	987	988	989	990	991

	0	1	2	3	4	5	6	7
1740	992	993	994	995	996	997	998	999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Appendix C
Powers of Two

Positive powers of two

n	2^n
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	1 31072
18	2 62144
19	5 24288
20	10 48576
21	20 97152
22	41 94304
23	83 88608
24	167 77216
25	335 54432
26	671 08864
27	1342 17728
28	2684 35456
29	5368 70912
30	10737 41824
31	21474 83648
32	42949 67296
33	85899 34592
34	1 71798 69184
35	3 43597 38368
36	6 87194 76736
37	13 74389 53472
38	27 48779 06944
39	54 97558 13888
40	109 95116 27776
41	219 90232 55552
42	439 80465 11104
43	879 60930 22208
44	1759 21860 44416
45	3518 43720 88832
46	7036 87441 77664
47	14073 74883 55328
48	28147 49767 10656
49	56294 99534 21312
50	1 12589 99068 42624
51	2 25179 98136 85248
52	4 50359 96273 70496
53	9 00719 92547 40992
54	18 01439 85094 81984
55	36 02879 70189 63968
56	72 05759 40379 27936
57	144 11518 80758 55872
58	288 23037 61517 11744
59	576 46075 23034 23488
60	1152 92150 46068 46976

n	2^n
61	2305 84300 92136 93952
62	4611 68601 84273 87904
63	9223 37203 68547 75808
64	18446 74407 37095 51616

Negative powers of two

n	2^{-n}
0	1.0
1	0.5
2	0.25
3	0.125
4	0.0625
5	0.03125
6	0.01562 5
7	0.00781 25
8	0.00390 625
9	0.00195 3125
10	0.00097 65625
11	0.00048 82812 5
12	0.00024 41406 25
13	0.00012 20703 125
14	0.00006 10351 5625
15	0.00003 05175 78125
16	0.00001 52587 89062 5
17	0.00000 76293 94531 25
18	0.00000 38146 97265 625
19	0.00000 19073 48632 8125
20	0.00000 09536 74316 40625
21	0.00000 04768 37158 20312 5
22	0.00000 02384 18579 10156 25
23	0.00000 01192 09289 55078 125
24	0.00000 00596 04644 77539 0625
25	0.00000 00298 02322 38769 53125
26	0.00000 00149 01161 19384 76562 5
27	0.00000 00074 50580 59692 38281 25
28	0.00000 00037 25290 29846 19140 625
29	0.00000 00018 62645 14923 09570 3125
30	0.00000 00009 31322 57461 54785 15625
31	0.00000 00004 65661 28730 77392 57812 5
32	0.00000 00002 32830 64365 38696 28906 25
33	0.00000 00001 16415 32182 69348 14453 125
34	0.00000 00000 58207 66091 34674 07226 5625
35	0.00000 00000 29103 83045 67337 03613 28125
36	0.00000 00000 14551 91522 83668 51806 64062 5
37	0.00000 00000 07275 95761 41834 25903 32031 25
38	0.00000 00000 03637 97880 70917 12951 66015 625
39	0.00000 00000 01818 98940 35458 56475 83007 8125
40	0.00000 00000 00909 49470 17729 28237 91503 90625
41	0.00000 00000 00454 74735 08864 64118 95751 95312 5
42	0.00000 00000 00227 37367 54432 32059 47875 97656 25
43	0.00000 00000 00113 68683 77216 16029 73937 98828 125
44	0.00000 00000 00056 84341 88608 08014 86968 99414 0625
45	0.00000 00000 00028 42170 94304 04007 43484 49707 03125
46	0.00000 00000 00014 21085 47152 02003 71742 24853 51562 5
47	0.00000 00000 00007 10542 73576 01001 85871 12426 75781 25
48	0.00000 00000 00003 55271 36788 00500 92935 56213 37890 625
49	0.00000 00000 00001 77635 68394 00250 46467 78106 68945 3125
50	0.00000 00000 00000 88817 84197 00125 23233 89053 34472 65625
51	0.00000 00000 00000 44408 92098 50062 61616 94526 67236 32812 5
52	0.00000 00000 00000 22204 46049 25031 30808 47263 33618 16406 25
53	0.00000 00000 00000 11102 23024 62515 65404 23631 66809 08203 125
54	0.00000 00000 00000 05551 11512 31257 82702 11815 83404 54101 5625
55	0.00000 00000 00000 02775 55756 15628 91351 05907 91702 27050 78125
56	0.00000 00000 00000 01387 77878 07814 45675 52953 95851 13525 39062 5
57	0.00000 00000 00000 00693 88939 03907 22837 76476 97925 56762 69531 25
58	0.00000 00000 00000 00346 94469 51953 61418 88238 48962 78381 34765 625
59	0.00000 00000 00000 00173 47234 75976 80709 44119 24481 39190 67382 8125
60	0.00000 00000 00000 00086 73617 37988 40354 72059 62240 69595 33691 40625
61	0.00000 00000 00000 00043 36808 68994 20177 36029 81120 34797 66845 70312 5
62	0.00000 00000 00000 00021 68404 34497 10088 68014 90560 17398 83422 85156 25
63	0.00000 00000 00000 00010 84202 17248 55044 34007 45280 08699 41711 42578 125

n	2^{-n}
64	0.00000 00000 00000 00005 42101 08624 27522 17003 72640 04349 70855 71289 0625

Appendix D
IOA$ usage

Declarations for ioa$, ioa$rs and arguments:

```
dcl ioa$                  entry options(variable);
dcl ioa$rs                entry options(variable);
dcl control               char(*);  /* control string */
dcl control_length        bin; /* length of control string */
dcl output_buffer         char(*); /* buffer for ioa$rs */
dcl output_buffer_size    bin; /* length of output buffer */
dcl rtn_buffer_length     bin; /* chars put into buffer */

call ioa$(control, control_length [, arg1, ..., arg99]);
call ioa$rs(output_buffer, output_buffer_size,
      rtn_buffer_length, control, control_length,
      [ arg1, ..., arg99]);
```

Conversion string format:

%[*fw*][.*s*][:*prec*][z][r]*type*

fw field width (default 1)
s scaling factor (default 0)
prec precision (values: 0, 1, 2, 3; default 1)
z character z, zero fill (default is blank fill)
r character r, reverse justification (default is right justify)

item	*type*	*fw*	*s*	*prec*	z	r	Notes
literal %	%	-	-	-	-	-	
decimal	d	o	o	0,1,2,3	o	o	1
octal	o	o	o	0,1,2,3	o	o	1
hex	h	o	o	0,1,2,3	o	o	1
fixed real	f	o	o	1,2	o	o	2
float real	e	o	o	1,2	o	o	2
logical	l	o	-	-	note 3	o	4
word	w	o	-	-	-	o	5
ASCII (non-var)	a	o	-	-	-	o	6, 7, 9
ASCII (non-var)	c	o	-	-	-	o	6, 8, 9
ASCII (var)	v	o	-	-	-	o	8, 9
pointer	p	o	-	-	-	o	9
filler	x	o	-	-	o	-	10
new line	/	o	-	-	-	-	10
form feed	^	o	-	-	-	-	10
reposition	y	o	-	-	-	-	11
repeat group	(	o	-	-	-	-	10, 12
end repeat	)	-	-	-	-	-	
terminate	$	-	-	-	-	-	
terminate w/newline	.	-	-	-	-	-	

- not applicable, usually ignored.
o optional

Notes:

1 Integer precision values are:

0 fixed bin(16,0)unsigned; SHORT_CARDINAL
1 fixed bin(15,0)signed; SHORT_INTEGER
2 fixed bin(31,0)signed; LONG_INTEGER

3 fixed bin(32,0)unsigned; LONG_CARDINAL

2 Floating precision values are

1 float bin(27);
2 float bin(48);

3 z implies TRUE/FALSE as opposed to T/F
4 works on bit(16)aligned (FORTRAN LOGICAL)
5 same as :0zo
6 takes two arguments: char(*), fixed bin(15,0) (string, length)
7 strips trailing blanks
8 doesn't strip trailing blanks
9 default justification is left
10 *fw* is repeat count (default 1)
11 *fw* is argument number (default 1)
12 repeat groups cannot nest

Ref: *Subroutines Reference Guide, Vol. III* [60].

Appendix E
References

[1] Burley, J. C.
Advanced Programmer's Guide, Vol I: BIND and EPFs.
Technical Report DOC 10055-1, Prime Computer, Inc., 1985.

[2] Burley, J. C. and Bruns, L. E.
Advanced Programmer's Guide, Vol II: File System.
Technical Report DOC 10056-1, Prime Computer, Inc., 1985.

[3] Burley, J. C. and Landy, A.
Advanced Programmer's Guide, Vol III: Command Environment.
Technical Report DOC 10057-1, Prime Computer, Inc., 1985.

[4] Unknown.
BASIC/VM Programmer's Guide.
Technical Report FDR 3058-101B, Prime Computer, Inc., 1981.

[5] Desmond, Ellen S.
C User's Guide.
Technical Report DOC 7534-3LA, Prime Computer, Inc., 1987.

[6] Morrow, Glenn.
CPL User's Guide.
Technical Report DOC 4302-3, Prime Computer, Inc., 1987.

[7] Turnbull, Ian K.
Data Backup and Recovery Guide.
Technical Report DOC 10129-1LA, Prime Computer, Inc., 1987.

[8] Karp, Joan.
DBMS Administrator's Guide.
Technical Report DOC 6292-192P, Prime Computer, Inc., 1985.

[9] Karp, Joan.
DBMS Data Description Language Reference Guide.
Technical Report DOC 5717-181L, Prime Computer, Inc., 1985.

[10] Kingsbury, B. & Wilson, A. C.
DISCOVER Reference Guide.
Technical Report DOC 7798-192L, Prime Computer, Inc., 1986.

[11] Unknown.
DISCOVER User's Guide.
Technical Report Unknown, Prime Computer, Inc., 19xx.

[12] Unknown.
Distributed Processing Terminal Executive Guide.
Technical Report IDR 4035, Prime Computer, Inc., 1981.

[13] Hassall, Peter and Wells, John.
DSM User's Guide.
Technical Report DOC 10061-1LA, Prime Computer, Inc., 1987.

[14] Shepp, Marion.
EMACS Reference Guide.
Technical Report DOC 5026-2LA, Prime Computer, Inc., 1988.

[15] Unknown.
FORTRAN 77 Reference Guide.
Technical Report IDR 4029, Prime Computer, Inc., 1980.

[16] Ward, Paul.
FED User's Guide.
Technical Report DOC 4940-191L, Prime Computer, Inc., 1983.

[17] Unknown.
FORMS Programmer's Guide.
Technical Report PDR 3040, Prime Computer, Inc., 1979.

[18] Lewis, Anthony.
FORTRAN Reference Guide.
Technical Report FDR 3057-101B, Prime Computer, Inc., 1980.

[19] Hammond, M. & Landy, A.
Instruction Sets Guide.
Technical Report DOC 9474-1, Prime Computer, Inc., 1985.

[20] Unknown.
Interpretive BASIC User's Guide.
Technical Report IDR 1813, Prime Computer, Inc., c1977.

[21] Prime.
PRIME Common LISP Environment Reference Manual.
Technical Report MAN 10120-1LA, Prime Computer, Inc., 1987.

[22] Prime.
PRIME Common LISP Language Reference Manual.
Technical Report MAN 10119-1LA, Prime Computer, Inc., 1987.

[23] Ladd, Anne P.
SEG and LOAD Reference Guide.
Technical Report DOC 3524-192, Prime Computer, Inc., 1983.

[24] Alley, Stephen.
Magnetic Tape User's Guide.
Technical Report DOC 5027-2LA, Prime Computer, Inc., 1986.

[25] Unknown.
MIDASPLUS User's Guide.
Technical Report IDR 4558, Prime Computer, Inc., 1981.

[26] Johnson, E. Andrew.
Modula-2 Reference Guide.
Technical Report PE-T 1265, Rev. 1, Prime Computer, Inc., 1987.

[27] Shores, Andrew.
Network Planning and Administration Guide.
Technical Report DOC 7532-3LA, Prime Computer, Inc., 1987.

[28] Dern, Daniel.
New User's Guide to EDITOR and RUNOFF.
Technical Report FDR 3104-101A, Prime Computer, Inc., 1981.

[29] Shores, Andrew.
NTS Planning and Configuration Guide.
Technical Report DOC 10159-1LA, Prime Computer, Inc., 1987.

[30] Unknown.
OAS System Administrator's Guide.
Technical Report DOC 6781-030L, Prime Computer, Inc., 1985.

[31] Perry, Elizabeth Hanes.
Operator's Guide to the Batch Subsystem.
Technical Report DOC 9302-3LA, Prime Computer, Inc., 1986.

[32] Gove, George.
Operator's Guide to File System Maintenance.
Technical Report DOC 9300-4LA, Prime Computer, Inc., 1986.

[33] Rose, Tom.
Operator's Guide to the Spooler Subsystem.
Technical Report DOC 9303-2LA, Prime Computer, Inc., 1986.

[34] Alley, Stephen.
Operator's Guide to System Backups.
Technical Report DOC 9301-1LA, Prime Computer, Inc., 1986.

[35] Forbes, J., Landy, A., Miles, C.
Operator's Guide to System Commands.
Technical Report DOC 9304-2LA, Prime Computer, Inc., 1986.

[36] Zegarra, Sonya.
Operator's Guide to System Monitoring.
Technical Report DOC 9299-3LA, Prime Computer, Inc., 1986.

[37] Hasse, Camilla B.
Pascal Reference Guide.
Technical Report DOC 4303-4LA, Prime Computer, Inc., 1987.

[38] Spector, D.
The DEREMER Parser Generator.
Technical Report PE-T 535, Prime Computer, Inc., 1987.

[39] Lacroix, R. P. Andre.
SPL Reference Guide.
Technical Report PE-T 1121, Rev. 1, Prime Computer, Inc., 1984.

[40] Rand, D.
BUILD: a Tool for Program Building.
Technical Report PE-T 1283, Rev 4, Prime Computer, Inc., 1987.

[41] Ullmann, R.
PDN Mailer User's Guide.
Technical Report PE-T 1325, Prime Computer, Inc., 1986.

[42] Xenakis, J. & Haase, C.
PL/I Reference Guide.
Technical Report DOC 5041-1LA, Prime Computer, Inc., 1986.

[43] Unknown.
PL/I Subset G Reference Guide.
Technical Report IDR 4031, Prime Computer, Inc., c1981.

[44] Bruns, Len.
Assembly Language Programmer's Guide.
Technical Report DOC 3059-2LA, Prime Computer, Inc., 1987.

[45] Venne, A. et al.
PRIMENET Guide.
Technical Report DOC 3710-193L, Prime Computer, Inc., 1985.

[46] Shores, Andrew.
PRIMENET Planning and Configuration Guide.
Technical Report DOC 7532-3LA, Prime Computer, Inc., 1987.

[47] Seybold, John.
Prime User's Guide.
Technical Report DOC 4130-4LA, Prime Computer, Inc., 1985.

[48] Unknown.
PRIMEWORD Administrator's Guide.
Technical Report DOC 11033-1LA, Prime Computer, Inc., 19xx.

[49] Carbonneau, William.
PRIMOS Commands Reference Guide.
Technical Report DOC 3108-6LA, Prime Computer, Inc., 1987.

[50] Walsh, R. & Paris, J.
PRISAM User's Guide.
Technical Report DOC 7999-3LA, Prime Computer, Inc., 1986.

[51] Burley, J. C., et al.
Programmer's Guide to BIND and EPFs.
Technical Report DOC 8691-1, Prime Computer, Inc., 1985.

[52] Ryan, David.
Remote Job Entry Phase II User's Guide.
Technical Report DOC 6053-4LA, Prime Computer, Inc., 1987.

[53] Munro, Andrew.
ROAM Administrator's Guide.
Technical Report DOC 7345-3LA, Prime Computer, Inc., 1987.

[54] McKenzie, Charles D.
RPG II V-Mode Compiler Reference Guide.
Technical Report DOC 5040-2LA, Prime Computer, Inc., 1985.

[55] Ryan, David.
PRIME/SNA Administrator's Guide.
Technical Report DOC 8908-3LA, Prime Computer, Inc., 1987.

[56] Ryan, David.
PRIME/SNA Operator's Guide.
Technical Report DOC 8909-3LA, Prime Computer, Inc., 1987.

[57] Cioto, Paul.
Source Level Debugger User's Guide.
Technical Report, Prime Computer, Inc., 1985.

[58] Breithaupt, J.
Subroutines Reference Guide; Vol I.
Technical Report DOC 10080-1, Prime Computer, Inc., 1986.

[59] Breithaupt, J.
Subroutines Reference Guide; Vol II.
Technical Report DOC 10081-1, Prime Computer, Inc., 1986.

[60] Breithaupt, J.
Subroutines Reference Guide; Vol III.
Technical Report DOC 10082-1, Prime Computer, Inc., 1986.

[61] Breithaupt, J.
Subroutines Reference Guide; Vol IV.
Technical Report DOC 10083-1, Prime Computer, Inc., 1986.

[62] Neilson, Peter A. and Forbes, B. Jacki.
System Administrator's Guide: Vol. I: System Configuration.
Technical Report DOC 10131-1LA, Prime Computer, Inc., 1987.

[63] Conrad, Lois Anne.
System Administrator's Guide, Vol. II: Communication Lines and Controllers.
Technical Report DOC 10132-1LA, Prime Computer, Inc., 1987.

[64] Frost, Dick.
System Administrator's Guide, Volume III: System Access and Security.
Technical Report DOC 10133-1LA, Prime Computer, Inc., 1987.

[65] Hammond, M. & Landy, A.
System Architecture Reference Guide.
Technical Report DOC 9473-1, Prime Computer, Inc., 1985.

Index