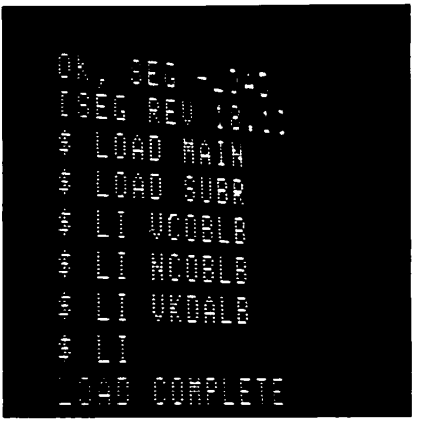
[redacted]

Prime Computer, Inc.

Loading and Debugging Rev. 18



FDR5094

TABLE OF CONTENTS

The Programmer's Companion is a new series of pocket-size, quick reference guides to Prime software products

Published by Prime Computer Inc
Technical Publications Department
500 Old Connecticut Path
Framingham MA 01701

First Printing May 1981

Credits

Research and copy
Alice Landy

Design and production
William I Agush

Typesetting
COMPSET, Inc

Printing and Binding
Commonwealth Publishing

Covers
MARK-BURTON

LOADERS

Prime supplies two loaders to create executable run files from compiled programs LOAD loads R-mode code gener ated by PMA FORTRAN, or RPGII SEG loads V-mode or I-mode code, generated by COBOL, FORTRAN, FOR-TRAN77, PASCAL, or PL/I, Subset G

For complete details on these loaders, see the LOAD and SEG Reference Guide

▶ LOAD

Invokes the Linking Loader for R-mode code When in-voked prints a $ prompt and waits for subcommands

LOAD SUBCOMMANDS

ATTACH [directory] [password] [disk] [key]
Attaches to specified directory

AUTOMATIC base-length
Inserts base area of specified length at end of routine if >'300 locations have loaded since last base area

CHECK [symbol-name] [offset-1] . . . [offset-9]
Checks value of current PBRK against symbol or number, **symbol-name** is an 8 character symbol de-fined in the symbol table **offset-1** thru **9** are summed to form an address or offset from symbol name Numbers preceded by "-" are negative

COMMON address
Moves top/starting COMMON location to **address.**

DC [END]
Defers definition of COMMON block **END** turns off DC

ENTIRE pathname
Saves entire state of loader as runfile along with temporary file for building overlays

ERROR n
Determines action taken in case of load errors

n	Meaning
0	SZ errors treated as multiple indirect, others act as n=1

1 Display multiple indirects on TTY but continue LOAD, abort load of file for all other errors
2 Abort to PRIMOS

EXECUTE [a] [b] [x]
Starts execution with specified register values

F/ { LIBRARY / LOAD } pathname [parameters]
Forceloads all modules in object file, specified by **pathname**. See LOAD for **parameters**.

INITIALIZE [pathname] [parameters]
Initializes LOADER and optionally does a LOAD See LOAD for **parameters**.

LIBRARY [filename] [loadpoint]
Loads the specified file from the library UFD Default = FTNLIB

LOAD pathname [parameters]
Loads the object module specified by **pathname.** (Searches first for **pathname.BIN**, then for plain **pathname.**) The **parameters** may be entered in three formats

1 **loadpoint [setbase-1] . . . [setbase-8]**
2 ***[setbase-1] . . . [setbase-9]**
3 **symbol [setbase-1] . . [setbase-9]**

In form 1 **loadpoint** is the starting location of the load In form 2 the load starts at the current PBRK location (*) In form 3 the load address can be stated symbolically (**symbol**) The remaining numeric parameters (**setbase-1**, etc) specify the size of linkage areas to be inserted before and after modules during loading If the last parameter is '177777, the loader requests more setbase values

MAP [pathname] [option]
Generates a load-state map If **pathname** is omitted, the map is displayed at the user terminal

Option	
0	Load state, base area, symbol storage map symbols sorted by address (default)
1	Load state only
2	Load state and base area
3	Unsatisfied references only
4	Same as 0
5	System programmer map
6	Undefined symbols sorted alphabetically
7	All symbols, sorted alphabetically
10	Special symbol map for PSD (in a file)

MODE { **D32R** | **D64R** | **D16S** | **D32S** | **D64V** }

Specifies address resolution mode for next load module (32K relative is default) If used, MODE must precede other LOAD commands

P/ { **LIBRARY** | **LOAD** } **[pathname] [parameters]**

Begins loading at next page boundary See LOAD for **parameters.**

PAUSE

Leaves loader to execute internal PRIMOS command Return via START

PBRK { **[symbol-name] [offset-1] . . . [offset-9]** | *** offset-1 [offset-2] . . . [offset-9]** }

Sets a program break to value of **symbol-name** plus **offset** or a number * treats sum of numbers as offset from current PBRK Offsets 2 through 9 may be negative

QUIT
Deletes temporary file closes map file (if loader opened it) and returns to PRIMOS

RR
Reset the save range prior to ENTIRE save when building overlays

SAVE [pathname] A-register B-register X-register
Saves the loaded memory image including all initialized COMMON areas into the file named **pathname** If pathname is a simple filename, it is in the current directory If **pathname** is not given, LOAD creates a filename from the name (without suffix) of the first file loaded, plus the SAVE suffix

The low high, start and keys parameters are obtained from the loader and saved as well (there is no way to set them)

A-register	Initial value of A register
B-register	Initial value of B register
X-register	Initial value of X register

SETBASE { **[base-start] [base-range]** | * **base-range** }
Defines starting location and size of base area * is current value of PBRK

SS symbol-name
Save symbol Exempts specified **symbol** from action of XPUNGE

SYMBOL symbol name { old-name [offset-1] [offset 9] | address [offset-2] [offset-9] | offset 1 [offset 2] [offset-9] }
Establishes locations in the memory map for COMMON blocks or for relocation points for the course of the load Also may be used to satisfy references Symbols follow rules for filenames but are limited to 8 characters

The first form equates two symbols or equates the new symbol to an offset from the old The second form equates a symbol to an octal value The third form equates a symbol to the current PBRK plus the sum of the numeric parameters

SZ { **YES** | **[NO]** }
Permits/prohibits links in sector zero

VIRTUALBASE base-start to-sector
Copies base sector to corresponding locations in **to-sector** Used for building RTOS modules

XPUNGE dsymbol dbase
Expunges symbol from symbol table and deletes base information

dsymbol	Action
0	Delete all defined symbols including COMMON area
1	Delete all defined symbols leaving COMMON areas

dbase	Action
0	Retain all base information
1	Retain only sector zero information
2	Delete all base information

▶ **SEG [pathname] [-LOAD]**

Invokes a utility for loading modifying running and sharing segmented (V mode or I-mode) programs

If the -LOAD option is given SEG enters the loader automatically and requests LOAD subcommands SEG also takes the base name of the first file loaded adds the SEG suffix, and creates the segment directory under this name

DELETE [pathname]
Deletes a saved SEG runfile, if **pathname** is omitted deletes the established runfile

HELP
Prints a list of SEG commands at user's terminal

LOAD
Synonym for **VLOAD**

MAP { [runfile] [output-file] [map-option] / * }

Prints a loadmap of **runfile** or current loadfile (*) at terminal or into **output-file**

0	Full map (default)
1	Extent map only
2	Extent map and base areas
3	Undefined symbols only
4	Full map (identical to 0)
5	System programmer's map
6	Undefined symbols alphabetical order
7	Full map sorted alphabetically
10	Symbols by ascending address
11	Symbols alphabetically

MODIFY [pathname]
Invokes MODIFY subprocessor to create a new run file or modify an existing runfile

MODIFY SUBPROCESSOR COMMANDS

NEW pathname
Writes a new copy of SEG runfile to disk

PATCH segno baddr taddr
Adds a patch (loaded between **baddr** and **taddr**) to an existing runfile and saves it on disk

RETURN
Writes runfile to disk and returns to SEG command level

SK { **ssize** / **segno addr** / **ssize 0 esegno** / **ssegno addr esegno** }

Specifies stack size (**ssize**) and location **esegno** specifies an extention stack segment

START segno addr
Changes address of the starting ECB

WRITE
Writes all segments above 4000 of current runfile to disk

End of MODIFY subprocessor

PARAMS [pathname]
Displays the parameters of a SEG runfile

PSD
Invokes VPSD debugging utility

QUIT
Returns to PRIMOS command level and closes all open files

RESTORE [pathname]
Restores a SEG runfile to memory for examination with VPSD

RESUME [pathname]
Restores runfile and begins execution

SAVE [pathname]
Synonym for MODIFY

SHARE [pathname]
Converts portions of SEG runfile corresponding to segments below 4001 into R mode-like runfiles

SINGLE [pathname] segno
Creates an R mode-like runfile for any segment

TIME [pathname]
Prints time and date of last runfile modification

VERSION
Displays SEG version number

VLOAD { **pathname** | * **[pathname]** }

Defines the runfile name and invokes the virtual loader to create a new runfile or append to an existing runfile if * is specified

VLOAD SUBPROCESSOR COMMANDS

ATTACH [ufd-name] [password] [ldisk] [key]
Attaches to directory

AUTOMATIC base-area-size
Automatically places base areas between procedures

A/SYMBOL symbolname [segtype] segno size
Defines a symbol in memory and reserves space for it using absolute segment numbers

COMMON { **ABS** / **[REL]** } **segno**
Relocates COMMON using absolute or relative segment numbers

D/ { **IL** / **LOAD** / **LIBRARY** / **PL** / **RL** }
Continues a load using parameters of previous load command

EXECUTE [a] [b] [x]
Saves loaded image on disk and executes program

F/ { **IL** / **LOAD** / **LIBRARY** / **PL** / **RL** } **[filename] [addr psegno lsegno]**
Forces loading of all routines in an object file

IL [addr psegno lsegno]
Loads impure FORTRAN library IFTNLIB

INITIALIZE
Initializes and restarts the VLOAD subprocessor. Used for aborting a bad load or beginning a new load after a SAVE

LIBRARY [filename] [addr psegno lsegno]
Loads a library file (PFTNLB and IFTNLB if no **filename** specified) LIBRARY may be combined with D/, F/, P/, and S/

LOAD [pathname] [addr psegno lsegno]
Loads object file

MAP [pathname] option
Generates load map (see SEG-level MAP command)

MIX { [ON] / OFF }
Mixes procedure and data in segments and permits loading of linkage and common areas in procedure segments Not reset by INITIALIZE

MV [start-symbol move-block desegno]
Moves portions of the load file Used primarily in creation of shared libraries

OPERATOR option
Enables or removes system privileges

PL [addr psegno lsegno]
Loads pure FORTRAN library PFTNLB

P/ { IL / LOAD / LIBRARY / PL / **RL** } **[filename] option [psegno] [lsegno]**

Loads on a page boundary The options are **PR** = procedure only, **DA** = link frames only, **none** = both procedure and link frames

QUIT
Returns to PRIMOS command level

RETURN
Returns to SEG command level

RL pathname [addr psegno lsegno]
Replaces a binary module in an established runfile

R/SYMBOL symbolname [segtype] segno [size]
Defines a symbol in memory and reserves space for it using relative segment assignment (Default = data segment)

SAVE [a] [b] [x]
Saves the results of a load on disk

SETBASE segno length
Creates base area for desectorization

SPLIT { **segno addr** | **addr** | **addr ssegno saddr esegno** }

Splits segment into data and procedure portions

SS symbol-name

Saves symbol prevents XPUNGE from deleting symbol name

STACK size

Sets minimum stack size

SYMBOL [new-symbol-name] { old-symbol-name | segno [wordno] } [addr]

Defines a symbol at specific location in a segment

SZ segno { **YES** | **NO** }

Allows or disallows use of sector zero base areas in procedure segments

S/ { IL | LOAD | LIBRARY | PI | RL } **[filename] addr psegno lsegno**

Loads an object file in specified absolute segments

XP dsymbol dbase

Expunges symbol from symbol table and deletes base information

dsymbol	Action
0	delete all defined symbols, including COMMON area
1	delete only entry points, leaving COMMON area

dbase	Action
0	delete all base information
1	retain only sector zero information
2	retain all base information

End of VLOAD subprocessor

SOURCE LEVEL DEBUGGER (DBG)

To use the source level debugger, compile the program with the -DEBUG option Load the program normally and run it When it finishes (or fails) invoke the debugger by typing DBG filename, and monitor its execution with DBG commands A summary of source level debugger commands follows For a complete description of these commands, refer to the Source Level Debugger Reference Guide

▶ **' primos-command-line**

Passes verbatim the text which follows the "'" to the PRIMOS command processor for execution

▶ **: [language-name / print-mode] expression**

Evaluates an **expression**

▶ *** value**

Executes the command line either indefinitely, until an error occurs, or a specific number of times

▶ **ACTIONLIST { SUPPRESS / PRINT }**

Controls the printing of actionlists

▶ **ARGUMENTS { program-block-name [\ activation-number] / alternate-entry-identifier }**

Displays the values of all the arguments to a given program block

▶ **BREAKPOINT [breakpoint-identifier] [action-list]**
[-AFTER value] [-BEFORE value]
[-EVERY value] [-COUNT value]
[-EDIT] [-IGNORE / -NIGNORE]

Sets and modifies breakpoints

▶ **CALL variable [(argument-list)]**

Calls a subroutine or function from the debugger command level

▶ **CLEAR [breakpoint-identifier]**

Clears a breakpoint

▶ CLEARALL [program block-name [DESC END]] { BREAK / TRACE }

Clears either all breakpoint or tracepoints in the debugging environment or all breakpoints or tracepoints in a specified program block

▶ **CMDLINE**

Calls the PRIMOS subroutine COMANL to read a line into the static command line buffer

▶ **CONTINUE**

Continues program execution following a breakpoint, condition signal or single step operation

▶ **DBG program-name [option-1 [option-2. . .]]**

Invokes the symbolic debugger

▶ ENVIRONMENT { **program-block-name [activation-number]** / **-POP** }

Defines the evaluation environment

▶ **ENVLIST**

Prints the current evaluation environment and the contents of the evaluation environment stack

▶ **ETRACE** { ON / ARGS / OFF }

Enables and disables entry and exit tracing

▶ **HELP**

Prints the name of the most recent and up-to-date DBG documentation

▶ **GOTO [program-block-name ['activation-number]] statement-identifier**

Modifies the value of the execution environment pointer, transferring control to a specific statement when program execution is resumed with either the CONTINUE command or a single step command

▶ **IF expression action-list** [ELSE **action-list**]

Conditionally executes one or more debugger commands, contingent upon the result of an expression evaluation.

▶ **IN**

Continues program execution until the next procedure is called.

▶ **INFO** { **program-block-name** | **alternate-entry-identifier** | **statement-identifier** }

Prints information about a procedure, alternate entry to a procedure, or a statement.

▶ **LANGUAGE** { **PLI** | **FORTRAN** }

Specifies a language for expression evaluation.

▶ **LET variable = expression**

Assigns a new value to any variable defined by the procedure.

▶ **LIST [breakpoint-identifier]**

Prints the attributes of one breakpoint or tracepoint.

▶ LISTALL [program-block-name [-DESCEND]] { -BREAKPOINTS | -TRACEPOINTS }

Prints a list of breakpoints and tracepoints.

▶ **MAIN [program-block-name]**

Defines the procedure called by RESTART or prints the name of the main program.

▶ **OUT**

Continues program execution until the procedure specified by the execution environment pointer at the time the OUT command was given returns.

▶ **PAUSE**

Temporarily suspends the debugging session, returning to PRIMOS command level.

▶ **PMODE print-mode variable-1 [, variable-2...]**

Sets the print-mode of a variable.

▶ **PSYMBOL**

Prints a table containing the names of the special symbols and their current character values.

▶ **QUIT**

Exits to PRIMOS command level terminating the debugging session

▶ **RESTART [step-command]**

Starts or restarts execution of the program

▶ **RESUBMIT**

Allows the user to edit and resubmit for execution the last command line entered using the DBG command line editor

▶ **SEGMENTS**

Prints a list of segments which are in use

▶ **SOURCE source-command [argument]**

Examines source files during a debugging session

▶ **STATUS**

Prints information about the debugging environment of the program

▶ **STEP [value]**

Resumes program execution for **value** number of statements and then returns to DBG command level

▶ **STEPIN [value]**

Resumes program execution for value number of statements

▶ **STRACE** { FULL | QUIET | OFF }

Enables or disables statement tracing

▶ **SYMBOL symbol-name character-value**

Modifies the value of a debugger character symbol

▶ **TRACEBACK** [-FRAMES value [-FARTHEST-RECENT]] [-FROM value] [-TO value] [REVERSE] [-DGB] [-ONUNITS] [ADDRESSES]

Prints the call/return and ownership information contained in all or selected stack frames

▶ **TRACEPOINT** **[breakpoint-identifier] [-AFTER value] [-BEFORE value] [-EVERY value] [-COUNT value] [-IGNORE] [-NIGNORE]**

Sets and modifies tracepoints, and converts breakpoints to tracepoints

▶ **TYPE expression**

Evaluates expression and prints the attributes of the resultant value

▶ **UNWATCH variable-1 [, variable-2...]**

Removes one or more variables from the watch list

▶ **UNWIND**

Releases all activations of the user program and debugger from the procedure call/return stack and causes the execution environment pointer to become undefined

▶ **VPSD**

Enters the debugger's copy of VPSD, the 64V mode Prime Symbolic Debugger

▶ **VTRACE** { **ON** | **OFF** }

Enables or disables value tracing, while retaining the variables in the watch list

▶ **WATCH variable-1 [,variable-2...]**

Adds one or more variables to the watch list and automatically enables value tracing

▶ **WATCHLIST**

Prints the names of the variables currently on the watch list

▶ **WHERE [segment-number word-number]**

Prints either a program location or the value of the execution environment pointer

PRIMOS DEBUGGERS (PSD, VPSD)

For S and R mode programs load the object program, using the PRIMOS commands LOAD or RESTORE and then choose which version you need Since the debug utility is resident in user memory with your program, make sure your program is not overlaid For V-Mode programs load using SEG and then run by saying SEG *filename* 1/1 Type the VE command to VPSD to obtain its restart address

Terminating long operations: To terminate long operations such as DUMP, type CTRL-P to return to PRIMOS command level

Restarting: Restart at the version's starting address To determine this value type a VERSION command to print the starting location

PSD/VPSD input/output formats: The format is established by ending any command with a colon followed by a single letter as in A 1000 O

A	ASCII
:B	Binary
:D	Decimal
:H	Hexadecimal
:O	Octal
:S	Symbolic
:L	Long integer (V PSD only)
:P	AP

Expressions: only + and − operators No literals

Symbol use: global symbols if the LS procedure has been used or any symbols defined within PSD Symbolic input is only legal in access mode—i e , 'S 100 200 SA1A is not legal Constants entered in S mode are octal

Command Line Operands: These may be constants, constant expressions or symbols The format of a constant is.

[:format] [>] { **±digits** / **ASCII-constant** } **[:format]**

> = relocatable mode

Access Mode Terminators:

CR	Move to next location
↑	Move to current location -1
?	Exit from access mode, do not change contents of current location
.n (CR)	Move to current location +n
.-n (CR)	Move to current location −n
@	Move to location addressed by instruction in current location
(	Move to location addressed by instruction in current location without following indirects
/	Return to the address of the last @
)	Return to the address of the last (
=	Calculate and print effective address and its contents
!	Set current location to new contents and exit from access mode

Subcommands: *(input rust letters in upper-case ONLY)*

▶ **ACCESS address**
Accesses **address** in the current segment and waits for keyboard input in the following form
[:format-symbol] [value] [:new-format] terminator

▶ **BR** **(VPSD)**
Prints the contents of the procedure base, stack base, link base and temporary base registers

▶ **BREAKPOINT location**
Sets a breakpoint at the specified **location.**

▶ **COPY blockstart block-end target**
Copies a block of memory to a new location starting at **target.**

▶ **DEFINE symbol value** **(PSD)**
Assigns a **value** to an alphanumeric **symbol.**

▶ **DUMP block-start block-end [words-per-line]**
Prints the contents of a block of memory on the terminal or, optionally to a file previously OPENed with no parameters

▶ **EFFECTIVE block-start block-end address [mask]**
Searches for an instruction with the specified effective **address** in the specified block, under an optional **mask.** The current values of the register are used

▶ **EXECUTE** **(VPSD)**
Executes a segmented program by passing control to SEG

▶ **FILL block-start block-end constant**
Fills a memory block with the specified **constant.**

▶ **FA f-register** **(VPSD)**
Accesses field address register 0 or 1 To change contents, enter new value in octal and terminate with ' ' or CR CR advances to the next register in the sequence FA0 FL0, FA1 FL1, while "^" backs up one A ' ?' returns to VPSD command mode

A "(" enters access mode displaying segment, word number and (in ASCII) the contents of the first location in the field CR advances to the next location in the field, and ^" backs up one To alter contents, type two ASCII characters before terminator A ")" returns to FA mode

▶ **FL f-register** **(VPSD)**
Accesses field length register 0 or 1 To change contents enter new value in 32-bit unsigned octal, and terminate with "!" or CR CR and "^' display next and previous registers, respectively, in the sequence FL0, FA1 FL1, FA0 A ? returns to VPSD command mode

▶ **GO [count] [a] [b] [x] [kevs]** **(PSD)**
Proceeds from the current breakpoint **Count** is the number of times to execute breakpoint location

▶ **JUMPTRACE [start-add] [a] [b]** **(PSD)**
Executes the object program and produces a diagnostic printout prior to execution of JMP, JST or HLT instruction SVC's are not traced Printout is

Location: instruction A= B= X= K= R=

▶ **KEYS value**
Sets CPU status keys to octal **value.**

▶ **LB segno wordno** **(VPSD)**
Loads the link base register with a segment number and word number

▶ **LIST address**
Prints the contents of **address** in the current output format *(Does not move pointer)*

▶ **LS** **(PSD)**
Enables the use of load map symbols Load program and use MAP 10 option to create symbol file, restore program and invoke PSD open PSD symbol file on any unit for reading, give LS command and close unit

▶ **MAP** **(PSD)**
Prints load map symbols and definitions

▶ **MO** { **D16S** | **D32S** | **D32R** | **D64R** | **D64V** }
Sets address mode

▶ **MONITOR [start-add] [a] [b] address** **(PSD)**
Traces the object program for a memory reference instruction whose effective address equals **address.**

▶ **NOT-EQUAL block-start block-end n-match [mask]**
Searches memory block for words not equal to **n-match** under an optional **mask** (a 16-bit logical AND)

▶ **OPEN filename file-unit key**
Opens a file to be used either as a DUMP output file or symbol table input file **Filename** must be ≤ six characters **Key** is the same as for PRIMOS OPEN

▶ **PRINT**
Prints CPU/PSD parameters in octal as follows
p: breakpoint a b x keys relcon

▶ **PROCEED [address] [a] [b] [x] [keys]**
Removes the current breakpoint, optionally sets a new breakpoint at **address**, and resumes execution

▶ **QUIT**
Returns to PRIMOS, (or SEG for SEG's VPSD)

▶ **RELOCATE value**
Sets a new **value** for the access mode relocation counter

▶ **RUN [start-add] [a] [b] [x] [keys]**
Runs the executable program starting at **start-add.**

▶ **SB segno wordno** **(VPSD)**
Loads the stack base register with a segment number and a word number

▶ **SEARCH block-start block-end match-word [mask]**
Searches memory block for words equal to **word** under an optional **mask.**

▶ **SN segno**
Sets a segment number for all commands where only a word number is entered such as UPDATE, DUMP, etc

▶ **SYMBOL** { 1 / 0 } **(PSD)**
Controls the use of symbols in address typeout. 1 =symbols 0 = no symbols

▶ **TRACE [start-add] [a] [b]** { p-val [0] / -1 interval }
Dynamically traces program by interpretive execution of each instruction and diagnostic printout. **P-val** causes printout only when program counter = **p-val**. P-val 0 means printout the first time program counter = p-val and every instruction thereafter **-1 interval** means printout every **interval** instructions. HLT instructions always cause printout followed by return to command mode

▶ **UPDATE location contents**
Puts **contents** into **location** and prints the old and new contents

▶ **VERIFY block-start block-end copy**
Verifies block of memory by comparing it with another block starting at **copy.** Locations that do not match are displayed as
location block-contents copy-contents

▶ **VERSION**
Prints the version number and restart address of the utility as an aid in restarting

▶ **WHERE**
Lists all currently installed breakpoints and their remaining proceed counts A proceed count of 1 is not listed

▶ **XB segno wordno** **(VPSD)**
Loads temporary base register with a segment number and word number

▶ **XR value** **(VPSD)**
Loads the X register with **value.**

▶ **YR value**
Loads value into the Y index register (P350 and up)

▶ **ZERO [location]**
Removes breakpoint at specified **location** or the breakpoint at the current program-counter location