PR1ME Technical Update

Subject: Source Level Debugger,Rev.18 **Number:** 68

Revision: 0

Date: December 1980

Applicable Hardware: All CPUs

Applicable Software: PRIMOS

Documentation Impact: The Source Level Debugger Guide (IDR4033)

Abstract:

The Source Level Debugger (DBG) now supports filename suffix conventions.

Changes have been made to: Command line edit facility, ACTIONLIST command, HELP command, IF command.

The following commands have been added -- LOADSTATE (restores DGB information saved by SAVESTATE), SAVESTATE (saves certain debugging information for subsequent restoration), MACRO (allows definition of macros), and MACROLIST (lists all defined macros).

PR1ME **Computer, Inc.** **145 Pennsylvania Avenue, Framingham, Mass. 01701**
(617) 879-2960

PTU 68

REV. 18 SOURCE LEVEL DEBUGGER

The Source Level Debugger now supports the standard suffixes for filenames. The following other extensions and additions have been made at Rev. 18.

****

ADD TO PAGE 2-25

A Few Warnings Concerning DBG Usage

This section lists some restrictions on the use of DBG which may cause difficulties debugging some programs.

- Exit breakpoints may not be set in FTN program blocks which execute alternate returns. This restricts the use of exit breakpoints and tracepoints, the OUT command, entry/exit, statement, and value tracing. This restriction does not apply to F77 GOTO's, or to GOTO's executed by PL1$NL to labels created by MKLB$F.

- When debugging a program which closes file units indiscriminately (e.g., CLOSE ALL) the -FULL_INIT (-FI) option should be specified on the DBG command line. When quick initialization is performed, the debugger leaves one file unit open to perform initialization throughout the debugging session. If this unit is closed by the user program (or the user), the next attempt by DBG to initialize a block will fail.

- If an onunit for the condition ILLEGAL_INST$ or ANY$ has been created by an active user program block, this onunit will be invoked whenever any breakpoint is encountered (whether set by the user or DBG).

- If the user program has created onunits for ILLEGAL_INST$ or ANY$ which are compiled in debug mode, DBG commands which cause statement traps to be set (e.g., STEP, STEPIN, STRACE, VTRACE) should not be used, and breakpoints or tracepoints should not be set in those onunits. Doing so will cause infinite recursion as user onunits will be invoked repeatedly rather than the DBG handler for breakpoints (i.e., illegal instructions).

- When debugging a program which uses specific segments for some purpose (e.q., temporary storage) without allocating those segments in SEG, DBG's storage manager may use those segments for its storage. To avoid this problem, such segments may be marked as used with the A/SYMBOL command in SEG.

****

CHANGE PROGRAM BLOCKS - FORTRAN ON PAGE 3-2

Main programs are identified by $MAIN (or by name, if a name has been specified in a FORTRAN 'PROGRAM' statement),

****

ADD TO PAGE 4-2

All breakpoints and tracepoints with their attributes may be saved to a file for restoration at a later debugging session by using the SAVESTATE command.

****

CHANGE COMMAND LINE EDIT FACILITY ON PAGE 4-21

to modify the content of breakpoint action lists or macro command lists,

This line is the previous command line, for the RESUBMIT command, or the breakpoint action list, for BREAKPOINT -EDIT or the macro command list, for MACRO -EDIT.

****

CHANGE ACTIONLIST COMMAND ON PAGE 5-4

Controls the printing of action lists or macro command lists.

****

REPLACE THE HELP COMMAND ON PAGE 5-14

The HELP Command

The HELP command may be used to find the name of the most recent and up-to-date DBG documentation. It may also be used to display a list of all debugger commands, the syntax of any DBG command, or the definition of a command syntax symbol.

The format of the HELP command is:

```
HELP  [-LIST | <command-name> | <syntax-symbol>]
```

<command-name> is the name or abbreviation of a DBG command.

<syntax-symbol> is a symbol enclosed in angle brackets used in a command syntax description.

If the HELP command is issued with no arguments, the syntax of the HELP command and the name of the most recent DBG documentation are printed.

If LIST is specified, a list of DBG commands is printed. Command abbreviations are indicated by capital letters.

If HELP is followed by a <command-name>, the syntax of that command is displayed.

If a <syntax-symbol> follows HELP, the definition of that symbol is printed.

Examples:

```
> HELP
For help, refer to IDR4033 Source Level Debugger Reference Manual.

HELP  [-LIST | <command-name> | <syntax-symbol>]

> HELP -LIST
    !                  *                  :
    ActionList         ARGumentS          BReaKpoint
    CALL               CLeaR              CLeaRAll
    CmdLine            Continue           ENVironment
    EnvList            ETrace             GOTO
    HELP               IF                 IN
    INFO               LANGuage           LET
    LIST               LiSTAll            LoadState
    MACro              MacroList          MAIN
    OUT                PAuse              PMode
    PSYMbol            Quit               ReSTart
    ReSUbmit           SaveState          SEGmentS
    SouRCe             STATUS             Step
    StepIn             STrace             SYMbol
    TraceBack          TRAcepoint         TYPE
    UnWatch            UNWIND             vPSD
    VTrace             WAtch              WatchList
    WHere

> HELP INFO
INFO  {<program-block-name> \  |
       <alternate-entry-identifier>  |
       <statement-identifier>}
```

```
> HELP <STATEMENT-IDENTIFIER>
<STATEMENT-IDENTIFIER>:
<source-line> |
<source-line>+<statement-offset> |
<source-line> (<insert-line>) |
<source-line> (<insert-line>+<statement-offset>) |
<statement-label> |
<statement-label>+<line-offset> |
<statement-label>+<line-offset>+<statement-offset>
```

****

CHANGE IF COMMAND ON PAGE 5-15

IF is generally used in conjunction with a breakpoint action list or a macro command list, but may be used alone if desired.

****

ADD TO PAGE 5-2Ø

The LOADSTATE Command

The LOADSTATE command is used to restore the DBG information contained in a file created by the SAVESTATE command at a previous debugging session.

The format of this command (abbreviated LS) is:

```
LoadState <file-name>
```

Where:

<File-name> is a tree name describing a file created by the SAVESTATE command.

When the LOADSTATE command is given, commands are read from <file-name> and executed by DBG before returning to command level. Should an error occur while processing a command, terminal output is resumed, and an error message is printed, but the remaining commands in <file-name> are executed.

Example:

```
> LOADSTATE DBG.SS.Ø2/27
```

The SAVESTATE Command

The SAVESTATE command may be used to save certain debugging information for restoration at a subsequent invocation of DBG.

The format of the SAVESTATE command (abbreviated SS) is:

```
SaveState <file-name>
```

Where:

<File-name> is a tree name describing a file.

When this command is issued, the following are saved: all tracepoints and breakpoints with associated action lists and attributes, and all user-defined macros. DBG commands to restore this information are written to <file-name> in user-readable form.

The DBG state may be restored later from this file using the LOADSTATE command, or the file may be used to remind the programmer of patches made to correct the program. The user may modify files created by SAVESTATE command using the editor.

Note: the END-SAVE pseudo-command appears at the end of all SAVESTATE files. It triggers certain cleanup operations following a LOADSTATE command.

Example:

```
> SAVESTATE DBG.SS.Ø2/27
```

The MACRO Command

Using the MACRO command, the user can define a macro name which may be used in place of one or more debugger commands.

The format of the command (abbreviated MAC) is:

```
MACro <macro-name> {<command-list> | -DeLete | -EDit}
```

<Macro-name> is a user-supplied name.

<Command-list> is one or more debugger commands enclosed in square brackets ("[]").

A macro is defined with the MACRO command by entering a <macro-name> followed by a <command-list>. The macro is then entered into a debugger table known as the MACRO LIST. Thereafter, whenever <macro-name> is entered at DBG command level, the debugger commands in <command-list> will be executed, with supplied parameters, if any.

A <macro-name> is removed from the macro list by supplying the argument -DELETE.

The -EDIT argument indicates that the <command-list> associated with this <macro-name> is to be edited using the DBG command line edit facility.

All currently defined macros may be displayed using the MACROLIST command. Macros may be saved to a file using the SAVESTATE command.

Macros may be defined which expect parameters following <macro-name> on the command line. These parameters are specified by a sequence of the form %i% in the command list, where _i_ is a positive integer. The sequence %i% in <command-list> indicates that <command-list> should be expanded prior to execution by replacing each occurrence of %i% with the _i_th token following <macro-name> on the command line. Suppose the macro RS has been defined as follows:

```
> MACRO RS [RESTART %1%;  SOURCE PRINT]
```

The command line RS STEP would be expanded as:

```
RESTART STEP;  SOURCE PRINT
```

If fewer than _i_ tokens have been supplied on the command line, %i% is replaced by the null string. For example, entering RS with no arguments would result in the following expansion:

```
RESTART ;  SOURCE PRINT
```

If %i% is the highest numbered parameter, or if the macro expects no parameters, tokens beyond the _i_th token are appended to the expanded command list. For example, the command line RS STEP 5 would be expanded as:

```
RESTART STEP;  SOURCE PRINT 5
```

Tokens are delimited by spaces. However, a string which contains spaces may be entered as one token by enclosing the string in single quotes. For example, the command line RS 'STEP 5' would expand to:

```
RESTART STEP 5;  SOURCE PRINT
```

Tokens which contain literal single quotes must also be quoted, and all literal single quotes included in a token must be doubled. Consider a macro V which has been defined as:

```
> MACRO V [:  'VALUE = ' || %1%]
```

then the following expansions would result from the given input:

Input	Expansion
V '''STRING'''	: 'VALUE = ' || 'STRING'
V 'STRING'	: 'VALUE = ' || STRING
V '''STRING OF A''''S'''	: 'VALUE = ' || 'STRING OF A''S'

If a token has been supplied which has no corresponding parameter, it is not used in the expansion. For example, if the macro RS2 is defined as:

```
> MACRO RS2 [RESTART %1%;  SOURCE PRINT %3%]
```

typing RS2 STEP 2 3 would result in the expansion:

```
RESTART STEP;  SOURCE PRINT 3
```

The sequence %% is interpreted as a single literal percent sign (%). If the characters between two paired percent signs cannot be converted to an integer, the entire sequence is copied literally.

The expanded macro may be printed prior to its execution by entering the ACTIONLIST PRINT command.

Examples:

1) To create a macro which expects two parameters:

```
> MACRO PN [: ADR(%1%)->PRES(%2%).NAME]
```

2) To execute the macro defined in example 1:

```
> PN 3 I
NAME = 'John Tyler'
```

3) To edit the macro PN:

```
> MACRO PN -EDIT
   : ADR(%1%)->PRES(%2%).NAME
: A; : STATE(ADR(%1%)->PRES(%2%).SN)
   : ADR(%1%)->PRES(%2%).NAME; : STATE(ADR(%1%)->PRES(%2%).SN)
:
```

4) To execute the edited macro, and view its expansion before execution:

```
> ACTIONLIST PRINT
> PN 3 I

<1> : ADR(3)->PRES(I).NAME; : STATE(ADR(3)->PRES(I).SN)
NAME = 'John Tyler'
STATE = 'Virginia'
```

5) To delete the macro:

```
> MACRO PN -DELETE
```

The MACROLIST Command

The MACROLIST command may be used to list all currently defined macros with their associated command lists, or to display a specific macro and its command list.

The format of the command (abbreviated ML) is:

```
MacroList [<macro-name>]
```

Where:

<Macro-name> is the name of a macro which has been defined by the user.

If the MACROLIST command is given with no arguments, all macros on the macro list are printed with their associated command lists.

If a <macro-name> follows the MACROLIST command, the command list for <macro-name> only is printed.

Examples:

1) To list all currently defined macros:

```
> MACROLIST
BALW     [BRK %1% [%2%; WHERE]]
BCC      [BREAKPOINT %1%; CLEAR; CONTINUE]
CMP      [LET I=1; IF '1'B [IF A(%1%,I)=N [: I]; LET I=I+1;*8]]
DEC      [:FORTRAN, DECIMAL :%1%]
RS       [RESTART %1%; SOURCE PRINT]
STS      [STEP; SOURCE EX]
```

2) To display only the command list for the macro BCC:

```
> MACROLIST BCC
BCC      [BREAKPOINT %1%; CLEAR; CONTINUE]
```