An Interactive Execution and Symbolic Debugging System

Leonard Shustek
Computation Research Group
Stanford Linear Accelerator Center
P.O. Box 4349
Stanford, California 94305
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>How To Run It</td>
<td>5</td>
</tr>
<tr>
<td>Notation Used In This Description</td>
<td>6</td>
</tr>
<tr>
<td>Leading and Executing Programs</td>
<td>7</td>
</tr>
<tr>
<td>Asynchronous Execution</td>
<td>8</td>
</tr>
<tr>
<td>A Note about &lt;dsname&gt;</td>
<td>8</td>
</tr>
<tr>
<td>Starting and Stopping Programs</td>
<td>9</td>
</tr>
<tr>
<td>Accessing a Program's Output</td>
<td>10</td>
</tr>
<tr>
<td>Looking at and Modifying Memory</td>
<td>11</td>
</tr>
<tr>
<td>DUMP</td>
<td>11</td>
</tr>
<tr>
<td>EDIT</td>
<td>11</td>
</tr>
<tr>
<td>FIND</td>
<td>12</td>
</tr>
<tr>
<td>Expressions</td>
<td>13</td>
</tr>
<tr>
<td>Address Expressions</td>
<td>14</td>
</tr>
<tr>
<td>Looking at and Modifying Variables</td>
<td>15</td>
</tr>
<tr>
<td>The DISPLAY command</td>
<td>15</td>
</tr>
<tr>
<td>Assignments</td>
<td>15</td>
</tr>
<tr>
<td>Modifying the Symbol Table</td>
<td>16</td>
</tr>
<tr>
<td>Disabling Subroutines</td>
<td>17</td>
</tr>
<tr>
<td>Calling Subroutines from DEMON</td>
<td>18</td>
</tr>
<tr>
<td>Controlling FORTRAN files</td>
<td>19</td>
</tr>
<tr>
<td>SET Commands</td>
<td>20</td>
</tr>
<tr>
<td>SHOU Commands</td>
<td>23</td>
</tr>
<tr>
<td>Miscellaneous Commands</td>
<td>26</td>
</tr>
</tbody>
</table>
Unattended Execution ........................................ 27
Running Non-FORTRAN jobs under DEMON ................. 28
Miscellaneous Notes ........................................... 28
   The Timebomb ............................................... 28
   The INISH file ............................................. 28
Appendix I - JCL for Running DEMON ....................... 29
Appendix II - Sample DEMON Session ....................... 31
Appendix III - History ....................................... 36
Appendix IV - Ideas for Future Changes .................... 38
DENON is an experimental interactive Debugging MONitor designed primarily to be used with FORTRAN programs at SLAC. Some of the facilities of DENON are:

1) FORTRAN, FORTRAN, or Assembly-language source programs residing on any WILBUR disk volume may be compiled by command. The SYSPRINT and/or SYSTEM output may be
   A) Listed on the terminal
   B) Sent to the printer
   C) Saved in a dataset to be accessed by WILBUR.

2) The resulting object modules may be link-edited with public or private load module libraries to form an executable load module.

3) The resulting load module, or any load module from an accessible load module library, may be loaded into memory for execution. The absolute csect loading map may be displayed.

4) Execution of the loaded program may be started, stopped, and restarted from the terminal.

5) Variables may be examined and set symbolically, using the original source program names. COMMON areas are accessible symbolically as well.

6) New variable names may be added to the symbol table, or existing variable names may be redeclared.

7) FORTRAN-like expressions involving standard functions, local variables, program variables, and constants may be immediately evaluated. More complex expressions may be written using IF-THEN-ELSE statements, FOR-TO-BY loops, and WHILE or UNTIL loops.

8) Breakpoints may be symbolically put into the program. When a breakpoint is encountered, execution may be suspended so that program variables may be examined and/or modified. Alternately, a string of commands may be associated with the breakpoint which will be executed whenever the breakpoint is encountered.
9) **FORTRAN files** may be assigned to the terminal, rather than to a file specified on a JCL card. Alternately, the I/O to any file may be echoed to the terminal in addition to normal processing by the operating system.

10) Most system **ABEND** conditions are announced at the terminal, as are program interrupts and **FORTRAN** execution-time error messages.

11) An optional log of all or part of the interactive session can be maintained.

12) Almost no restrictions are placed on the I/O characteristics of the executing program. In particular, it may use mountable disk or tapes, Unified Graphics devices, 2250 display scopes, real-time network devices, etc.

13) Programs may be executed in a "background" fashion so that the terminal remains unlocked and other **DENON** or **WILBUR/ORYLV** commands may be executed while the program runs. The program may be left running after you logoff.

14) **DENON** can be directed to automatically load a program and start execution without waiting for anyone to logon to the subsystem. You (or your representative) may then logon at any time, symbolically examine the progress of the program, and either stop it or allow it to continue to run unattended.
HOW TO RUN IT

The JCL necessary to run DEMON is currently in
WYL.CG.LJS.LIB@DEM0N ON CAT and also appears in Appendix I.
You may want to modify this JCL in the following ways:

1) Concatenate whatever load module libraries you wish
   the linkage editor to use to the LEHLIB DD card.

2) If you need to specify linkage editor 'INCLUDE'
cards, you can concatenate an appropriate 'DD *'
file to LEHLIB.

3) Add whatever DD cards will be necessary for your
   executing program. (PTRANSF cards for accessing
tape or disk datasets, TAPExx cards for use by
TPREAD, etc.)

4) Concatenate whatever load module libraries will be
   needed by your executing program to STEPLIB. (The
   Unified Graphics library, for example.)

If you intend to have a FORTRAN file always assigned to
the terminal, you need not include a DD card for it.

When the job has begun execution, you will be sent a
"READY" message and the subsystem name will appear in response
the "SHO SYS" command. The subsystem name will normally be
your initials followed by "DEM0N" (e.g. "LJSDEMON"). To enter
DEM0N, type the name of the subsystem as a WYLBUR command. The
prompt for DEM0N commands is '"' instead of the normal '"'.
You may return to WYLBUR at any time by typing "wylbur" as a
command to DEM0N.

WHERE TO COMPLAIN TO

I would like to receive complaints, questions, and
suggestions about DEM0N. Use the WYLBUR suggest command,
DEM0N's suggest command (q.v.), or contact me directly:

Len Shustek (LJSSCG)
SLAC Computation Research Group
BIN 88  12359
NOTATION USED IN THIS DESCRIPTION

The notation used here to describe the commands is much like that of the WYLBUR manual:

1) Braces are used to enclose required alternatives, either vertically or horizontally. Required alternatives written on one line are separated by the "or" symbol ({}).
   
   [this|that]    [this]    [that]

2) Brackets are used to enclose one or more optional alternatives:

   [maybe] [this|that] [this] [that]

3) Angle brackets surround symbols for which a specific value must be substituted. The symbol is (usually) defined elsewhere.

   <unit no.> <dsnane>

4) Most keywords longer than 3 characters may be abbreviated by the first 3 characters.

5) The WYLBUR convention for quoted strings is followed, that is, strings can be enclosed by either ' or " , and the string delimiter may be doubled to represent a single such delimiter inside the string.
LOADING AND EXECUTING PROGRAMS

{ ASH }
{ COMPILc }
{ NORTFAN } [ <dsname> ] [ PARM 'string' [ REPLACE ] ] [ NOUNIT ]
{ LINK/LINK } { }
{ LOAD }

This is the basic command for invoking a language processor or loading a program.

ASH invokes the Level-II assembler (with the modifications to produce SYSTEM output)

COMPILE invokes the FORTRAN-I Extended compiler, as modified by J. Steffani to output the symbol table. "FORTRAN" is a synonym for "COMPILE".

NORTFAN invokes the NORTFAN preprocessor

LINK invokes the linkage editor

LOAD loads (but does not start executing) a load module. (No external symbols not resolved by the linkage editor are resolved by LOAD. This command loads a program into memory, but does not invoke the OS LOADER.)

The <dsname> specifies the name of the file which contains the program. Source files may be in either card format or WYLBUR edit format, and may be either sequential datasets or members of a library. ORVYL files may not be specified, so they should be moved beforehand. If the dsname is omitted or specified as 's', input is assumed to be from a "logical" predecessor, according to the following flow diagram:

NORTFAN --> [ COMPILE ] --> LINK --> LOAD
{ ASH }

The PARM string may be specified for all except LOAD, and specifies parameters to be passed to the processor. If REPLACE is specified no other parameters are used; if REPLACE is omitted the given parameters are concatenated with those normally supplied. (See also the SET/SHOW PARM commands.)

When the program terminates, a message giving the return code of the task will be printed.

Examples:

FORT=0 (The FORTRAN compile step terminated without errors)
GO=S80A (The GO step terminated with system abend code S80A)
Asynchronous Execution ("NOWAIT")

The keyboard normally is locked while a program is executing. Execution may be interrupted by using the ATTN key and then may be resumed by typing "GO". Some processors, however, (and perhaps even user programs) take a painfully long time to complete. To allow program execution to proceed while you do other work, specify NOWAIT. After the program has begun executing, the terminal will be unlocked and you may issue other commands. Any DEMON commands may be issued except those which would start another program. When the program terminates, DEMON will interrupt with the normal return code message.

While the program is executing you may also transfer to WYLBUR (by typing "WYLBUR"), in which case the message "DEM0N WAITS YOU" will be sent when the program is terminating. If you wish to stop the execution of the program at any time, use the STOP command. For other notes about running programs in this asynchronous fashion, see the section on "Unattended Execution".

Current implementation restriction: NOWAIT cannot be used with LOAD, which, however, usually does not take such much time.

A NOTE ABOUT <dsnname>

The <dsnname> used here and elsewhere in DEMON is specified exactly as in WYLBUR, except that the special symbols $, #, and ' are not allowed.

Examples: "SOURCE"
MACLIB"BEGIN
WIL.CG.LJS.FCTIN ON WORK01
"FOUNDER USER XYZ GROUP TH"
"ASPGEN"

Source files may be in either card format or in WYLBUR edit format. In the latter case, the file will be "unpressed" to card format before being sent to the language processor.

The volume specified may be CAT or CATLG, in which case the catalog is searched for volume information. If the volume is omitted, the default volume is used. (See "SET VOLUME"). The default volume is initially "CAT".
STARTING AND STOPPING PROGRAMS

`GO [ PARN 'string' ] [ WOWAIT ] [ TO <addrexpr> ]`

This is the command used to start the execution of a program loaded with the LOAD command, or to resume the execution of a program which has been suspended.

The PARN string is valid only if execution is being started for the first time, not when execution is resumed.

The NOWAIT option specifies that the terminal is to be unlocked for further commands while the program is running. See the discussion of asynchronous execution in the previous section for more details.

If the TO <addrexpr> option is used, execution is resumed at the specified address. In general this is a very dangerous operation unless you clearly understand the register usage of the code and have made any necessary changes to the registers. For a related but much safer alternative, see the description of the CALL command.

`STOP`

This command suspends the execution of an asynchronously running program. Execution may be resumed using "GO".
ACCESSING A PROGRAM'S OUTPUT

The command allows the manipulation of the output of the language processors or a FORTRAN load module.

LIST lists the output on the terminal.

PRINT sends the output to a SISOUT file to be printed.

SAVE saves the output in a dataset which can be accessed by ULYB. The dataset is created automatically. If it exists already, you will be asked for permission to replace it.

In general, the "OUTPUT" file is the main printed output of the program, and the "ERRORS" file is a condensed listing of diagnostic error messages. The following table shows the ddnames used for each of the processors:

<table>
<thead>
<tr>
<th></th>
<th>OUTPUT</th>
<th>ERRORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>P106P001</td>
<td>none</td>
</tr>
<tr>
<td>FORTRAN</td>
<td></td>
<td>SYSTEMH</td>
</tr>
<tr>
<td>ASHE</td>
<td>P106P001</td>
<td>none</td>
</tr>
<tr>
<td>ASHE</td>
<td></td>
<td>SYSTEMH</td>
</tr>
<tr>
<td>LKED</td>
<td>P106P001</td>
<td>none</td>
</tr>
<tr>
<td>LKED</td>
<td></td>
<td>SYSTEMH</td>
</tr>
<tr>
<td>GO</td>
<td>P106P001</td>
<td>none</td>
</tr>
<tr>
<td>GO</td>
<td></td>
<td>SYSTEMH</td>
</tr>
</tbody>
</table>

If the qualifier ("FORTRAN" etc.) is omitted, the output of the last program executed is used.

"SQUASH" means "change all strings of more than one blank to a single blank".

If an arbitrary <ddname> is given in lieu of either OUTPUT or ERRORS, the specified file is used.

Remember that an executing program may buffer some of its output in memory, and that a file currently open for output may not have a valid end-of-file marker anywhere.
LOOKING AT AND MODIFYING MEMORY

DUMP <addrexpr> [ HEX|CHAR|ADDRESS ] [ LINES <expr> ]

This dumps memory in HEX or CHARACTER format (or both if no option is supplied). The <addrexpr> is a constant or expression which is used as the starting address. (More will be said about expressions later.) LINES specifies the number of lines to dump (or until the ATTN key is pressed, if LINES is omitted.)

The ADDRESS format of the dump tries to interpret each consecutive fullword as an address, and gives as much symbolic information about that address as it can.

Examples:
-dump main.rouvec
314090 40B17217 F7D1CF79 40B17217 F7D1CF79 * ...7J...7J.
3140A0 406F2BDC 549B943A 40C90F6A A22168C2 * ?...B. I...HE
314050 413243F6 A8885A31 4DC90F6A 42ABAC4B *...6yh..I......**
3140C0 411...

-dump grapln.txtstr char lines 2
470334 *'THIS BRILLIG AN'*
470344 *O THE SLITHRY TO*

EDIT <addrexpr> [ LINES <expr> ] [ HEX|CHAR ]

The command allows memory to be modified in either hex or character format. It is analogous to the WILBUR EDIT command, in that the line is printed, and changes are indicated by typing underneath the characters to be changed. You may make hex changes to the hex part of the dump, and character changes to the character part. The WILBUR EDIT command convention for blanks (" ") creates a blank unless it is already blank) is followed. LINES specifies how many lines (16 bytes each) are to be edited; the default is 1.

Example:
- ed ex main.abc
143348 00000001 01020304 47F0F00C 5010C1BA *.........00...A.*
EDITS? 2
143348 00000002 01020304 47F0F00C 50C2C2C2 *.........00..BBB*
EDITS?
FIND [ 'string' ] [ START <addrespr> ] [ HEX|CHAR ]
[ hexhex ] [ LINES|COUNT <expr> ]

This command searches memory for a particular character string or hex constant. If the START parameter is omitted it starts from the beginning of the loaded program. The hex constant (hexhex) must be an even number of hex digits.

When the specified string or hex constant is found, the 16 bytes starting at that location are displayed as if a DUMP command had been given. The HEX and CHAR options can be used to restrict the display to only the hex part or only the character part. Unless LINES is specified, the search will continue until all occurrences of the object are found; LINES specifies the maximum number of DUMPed lines and therefore the maximum number of occurrences of the string or hex constant that will be found.

To facilitate modification of the thing found, the local variable DA ("Dump Address") will be set to the address of the last occurrence printed. This variable may then, for example, be used in an EDIT command.

Example:

- find 'electrons' lines 1
  459322 C5D3C5C3 13D9D6D5 2A0D4C1 8640C2C5 *ELECTRONS MAY BE*
- edi da char
  459322 *ELECTRONS MAY BE*
  EDITS? posi
  459322 *POSITRONS MAY BE*
  EDITS?
EXPRESSIONS

An expression ("<expr>") or "<address expr>"") may be part of many DEMON commands. These expressions are FORTRAN-like, and may use any of the operators \( \times, \div, \% \) with the usual precedence rules applied.

The following basic elements may be part of expressions:

1) Decimal constants: 27 -3.6 22.34E-6

2) Hex constants: 01h43a (Must have a leading zero and no decimal point.)

3) Local variables. These are simple (unqualified) identifiers of up to eight characters. A variable is created by assigning it a value. Local variables are created within DEMON, and are not the same as variables in the program.

4) Program variables. These are variables in the program being run, and they are written in the following form:

   \(<\text{subroutine name}+>\cdot<\text{variable name}> [(<\text{subscript expr}>)]\)

When used in place of a subroutine name, the character "+" means "same as last time".

EXAMPLES: MAIN.J
           *.J
           MAIN.VECT(2)
           MAIN.VECT(MAIN.J)
           *.VECT(J) (J is a local variable)

Program variables will be known if the program was compiled under DEMON, or if the program was compiled in batch using FORTRAN or the TEST option of the assembler. If this was not the case, program variables may be declared using the DEMON command "DECLARE" (q.v.).

5) Machine (general purpose) registers, indicated by "a" followed by a constant, variable, or parenthesized expression whose value is between 0 and 15.

EXAMPLES: a2
           a1
           a(I-1)
6) Indirect through a memory location, indicated by ">>" followed by a constant, variable, or parenthesized expression.

EXAMPLES:  
>0142a4  
>>(4>16))  
>begin.startadr

7) Functions. Most of the FORTRAN mathematical functions are available. Arguments may be arbitrary expressions of any mode. The following additional functions may be used:

OUT(expr) outputs the value of the expr to the terminal
LOC(var) returns the address (location) of the variable
INF(var) prompt for an expression which will be assigned to the variable "var".

DEMON also allows more complicated expressions of the type accepted by the ABACUS interactive expression evaluator. Some of the available expression types are:

IF <lexpr> THEN <expr> ELSE <expr>
FOR <var> [= <expr>] [TO <expr>] [BY <expr>] <expr>
LOOP ... WHILE <lexpr> ... UNTIL <lexpr> ... REPEAT
DO <expr>; <expr> ... END

Documentation for ABACUS in available as Computation Group Technical Memo CGTM 181, available in CGB room 321. All ABACUS-like expressions are valid in DEMON except user-defined functions.

ADDRESS EXPRESSIONS

An "address expression" ( <addressexpr> ) is an expression in which the address rather than the value of a program variable is used. In addition, address expressions may use the following elements which are invalid for ordinary arithmetic expressions:

1) Unqualified subroutine names (e.g. MAIN )

2) FORTRAN statement label references of the form
   <subroutine name>.$<constant>

EXAMPLES:  
MAIN.$34  
*.#1007
LOOKING AT AND MODIFYING VARIABLES

The DISPLAY Command

The DISPLAY command is used to display the value of an expression as described above. It may be abbreviated by "D", and many expressions may be given at once.

Examples:

- disp main.i, main.a
  33 1.2947
  d 1/actev.energy(actev.i)
  1.4283458-08

If the name of an array in the program is given, all elements of the array will be displayed. If an array element is named and followed by '...', elements of the array starting at the named element will be displayed. In both cases the name and subscript will also be displayed, and consecutive identical values will be denoted by 'i...j'.

Examples:

- d main.rowvec
  ROWVEC(1)=3  ROWVEC(2)=5  ROWVEC(3...100)=0
- d main.a(10)...
  A(10...19)=12.4  A(20)=0

ASSIGNMENTS

Assignments may be written as FORTRAN assignment statements, or they may be proceeded by the keyword LET if the target conflicts with a DEMON command. The target may be a DEMON local variable (which will be created if it doesn't already exist), or a program variable.

Examples:

- main.i=0
- firstval=2**3 (subr.z)
- let d=47
- cvt = >16
MODIFYING THE SYMBOL TABLE

DECLARE <csect>.<var> [(dimension)] AT <addrEXPR> [TYPE <type>]

This command defines a program variable, that is, a
symbolic name for a datum or a location in the program. If the
COMPILE or ASH command was used to process the source program,
most program variables, labels, and statement numbers will
automatically have been defined, and this command can be used
to augment or modify those symbols.

The <csect> is the name of the program with which the
variable is to be associated. The variable will usually be
located within the <csect>.

The <var> is the name (up to 8 characters) to be defined.
If the name already exists in the <csect>, you will be asked
for permission to redefine it. The dimension is optional, and
can be used to declare a (one-dimensional) array.

The <addrEXPR> is an address expression (q.v.) which
specifies the location of the variable. A warning is issued if
it is not in the <csect>, or if it is outside the program.

The <type> is the type and length of the variable.
Currently the following are allowed:

REAL  REAL*8  REAL*4  R  R*4  R*8
INTEGER INTEGER*8  INTEGER*4  I  I*4  I*2
LOGICAL LOGICAL*8  LOGICAL*4  L  L*4  L*1
LABEL

The FORTRAN convention is obeyed if TYPE is omitted: TYPE
R*4 is used unless the first letter is I through N, for which
TYPE I*4 is used. Character types may eventually be added.

The DECLARE statement is useful in defining
"equivalences", that is, alternate names for variables with
different types. This is sometimes necessary for FORTRAN
variables in COMMON, since the compiler is wont to discard all
but one of the equivalenced names of variables in COMMON. All
the names (and types) of equivalenced variables not in COMMON
are normally available automatically.

EXAMPLES

DECLARE MAIN.INTX AT MAIN.I  (default type is INTEGER)

DECLARE SORT.VEC(20) AT COBBLK+20 TYPE I*2

(COBBLK is a COMMON block)
A NOTE ABOUT SYMBOL TABLES IN LOAD MODULES

Both the FORTRANI compiler and the Assembler (with the 'TEST' option specified) produce object decks with special symbol table cards. If these object decks are linked with the 'TEST' option, the symbol table information will be retained with the load module and thus be available to DEMON. The presence of that information does not affect normal batch use of the load module, nor does it increase the memory requirements. It does somewhat increase the disk space required for the load module.

DISABLING SUBROUTINES

DUMMY <csect> [,<csect>]...

This command can be used to cause a subroutine to be a NOP, that is, to immediately return to the caller. (It places a DB 14 instruction at the start of the routine.)

Example:

dummy rtput,rtget,rtinit

The DUMMY command can be used to prevent execution of subroutines even though they will be called by routines being tested. You must, of course, beware that the execution of a dummyed routine is not required for proper initialization of variables used elsewhere.
CALLING SUBROUTINES FROM DEMON

CALL name [ (arg [,arg]... ) ]

The CALL command allows you to call a subroutine or function directly from the terminal. This can be used to test individual subroutines without having to write special test main programs, or to retry sections of a program without reloading. It is much safer than the GO TO command in the FORTRAN environment because it (1) will initialize the run-time package if a main program has not been executed, and (2) will set up the registers according to the normal entry conventions.

The argument list is optional, but if the subroutine expects any they should be supplied. Arguments may be any of the following:

(1) An integer constant (passed as an 1#4 fullword)
(2) A floating-point constant (passed as 2#8 but also acceptable to 1#4 variables)
(3) An expression in parentheses (passed as 2#8)
(4) A quoted string of any length
(5) A program variable within any subroutine, main program, or common block
(6) A DEMON local variable (passed as 2#8)

For cases (5) and (6), any modification of the argument by the subroutine will be reflected in a change to the variable when the subroutine is finished. If you have CALLED a function, the function value will be in the program variable whose name is the same as the function name (i.e. DISPLAY ABC.ABC will display the value of function ABC after it has executed).

Examples:
call trkfind
call simul(main.anat,100,100,.0001)
call ugetxt("large",x,y,"deus ex machina",15,main.elen)
CONTROLLING FORTRAN FILES

ASSIGN <unit. no.> [ LCG|NOLOG ]
[ TERMINAL|NOTERMINAL ]

This command controls the assignment of FORTRAN files to the terminal, and controls whether I/O to the file should be logged on the terminal. The <unit. no.> is the FORTRAN unit (file) number used in the READ or WRITE statement.

TERMINAL indicates that all I/O operations for the file are to be diverted to the terminal. The usual FTnnFO01 ddcard is not needed or used.

NOTERMINAL indicates that all I/O operations are to be performed using the file specified on the usual FTnnFO01 ddcard.

LOG indicates that all I/O operations are to be logged on the terminal; that is, a copy of lines written or read to the file is to be typed on the terminal.

NOLOG indicates that I/O operations are not to be logged on the terminal.

There is one special unit number (100) used to represent the file on which the FORTRAN execution-time routines write error messages.

The initial file assignment are:

5 TERMINAL NOLOG
6 NOTERMINAL LCG (FT06FO01 is used)
100 NOTERMINAL LCG (FT06FO01 is used)
others NOTERMINAL NOLOG (FTnnFO01 is used)
**SET BREAK**

`SET BREAK <addrespr> [ ACTIVE|INACTIVE]`

`[ EXECUTE <commands or expressions> ]`

A breakpoint is set in the current program. If the breakpoint already exists, it will be modified to the state indicated. (In this case the `<addrespr>` can be the breakpoint number - as printed by the SHOW BREAK command - rather than the address of the breakpoint.)

ACTIVE breakpoints cause some action to be taken when program execution reaches the corresponding address. If there is no EXECUTE string, then execution of the program is suspended, and commands may be entered at the terminal. The GO command can be used to resume execution of the program. If the EXECUTE string is non-null, it will be interpreted as a string of commands and/or expressions (separated by semicolons) to be executed when the breakpoint is encountered. The command "PAUSE" may be executed from a breakpoint EXECUTE string to cause the program to be suspended, and commands may be entered from the terminal.

INACTIVE breakpoints cause no action until they have been set ACTIVE by a SET BREAK command.

**EXAMPLES:**

```
SET BREAK MAIN.012
SET BREAK SORT EXEC IF MAIN.J EQ 100 THEN PAUSE
SET BREAK 2 INACTIVE
SET BREAK 1 EXEC IF SORT.VEC(1) LT 0 THEN SET BRR 2 ACT
SET BRR SEARCH.01 EXEC DI SEARCH.IPES
SET BRR LOOKUP.010071 EXEC LOOKUP.I=LOOKUP.I+1
```

**SET NOBREAK**

`SET NOBREAK <addrespr>`

Remove a breakpoint. The `<addrespr>` can be the address at which the breakpoint was set, or the breakpoint number.

**Examples:**

```
SET NOBREAK MAIN.02
SET NOB3 3
```
SET HEX

Output all numerical quantities in hex, rather than decimal. (This does not affect numbers entered as input; hex numbers must still be entered with a leading zero.)

SET DECIMAL|NONE

Return to outputting numerical quantities in decimal.

SET VOLUME vvvvvv

Use "vvvvv" as the default volume. The initial default volume is "CAT".

SET DIGITS n

Display numerical quantities to "n" significant figures, where 1<n<17. This applies both to integers and floating point numbers.

SET FUZZ nnnn

Set the comparison tolerance for floating-point arithmetic comparisons to nnnn. This is used in ABACUS-like logical tests. The fuzz is initially about 1E-14.

SET NODE [ FORTRAN|NOFORTRAN ]

FORTRAN mode is the default. In NOFORTRAN mode, certain internal patches necessary to support breakpoints and file assignment to the terminal will not be attempted. Addresses are also formatted with the CSECT offset to match the assembly listing in addition to the offset from the nearest label.

SET MEMBER memname

Set the default memname to be used in the LOAD command to memname. The initial default member is MAIN.
SET NAME (PORT|PORT|ASH|LKED|GO) "string"

Change the name of a processor to "string". This can be used, for example, to change the assembler from M-level to G-level by saying "SET NAME ASH ASHG"

SET PARM (PORT|ASH|LKED|GO) 'string' [REPLACE]

The default parameters for the processor are modified. If 'REPLACE' is specified, the string entirely replaces the existing parameters, otherwise the string is appended to the end of the existing parameters. Remember that if 'REPLACE' is used, the TEST option must be specified for symbol table information to be available.

SET PRINT (ON|OFF)

The printed session log is either started or stopped. The session log is a SYSOUT file which contains a record of everything typed on the terminal. The print log is initially ON.

SET PROMPT x

The prompt character for DRECN commands is set to "x".

SET XXX

Any SET commands which cannot be recognized are sent to MILTER.

Example: SET NOCOMM
"SHOW" COMMANDS

SHOW BREAKPOINTS

Displays the address and state of all current breakpoints.

SHOW CSECT(MAP [ LIKE xxx ] [ HOSTS ])

Display the csect (subroutine) loading map. The address and length of each csect is printed, and the address only of each entry point is printed.

LIKE xxx prints only csects whose names begin with "xxx"

HOSTS causes any routines which look like standard FORTRAN library routines not to be printed.

SHOW VARS [ IN <csect> ] [ LIKE xxx ] [ VALUE=<expr> ]

[ VALUE ] [ UNDEFINED ]

The names and types of the variables in the named csect are displayed. If the IN <csect> phrase is omitted, the last-accessed csect is used.

LIKE xxx causes only variables whose names begin with "xxx" to be listed.

VALUE=<expr> causes the names of variables which have the specified value to be printed. Array elements will be indicated by subscript, for example "A(1,3,5...9)".

VALUE causes the name and value of all variables to be displayed.

UNDEFINED (or VALUE=UNDEFINED) causes the name of all undefined variables to be displayed.

SHOW LOCALS

Displays the names of all local variables (those within DEMON) which have been created by assignment commands.
SHOW TRACEBACK

Displays the current location and subroutine-call traceback for the program currently executing.

SHOW VOLUME

Displays the current default disk volume.

SHOW VOLS;VOLUMES  [ LIKE xxx ]

Displays the volumes which can be accessed by DEMON for "COMPILE", "SAVE", and related commands.

SHOW UNITS

Display the current FORTRAN unit assignments.

SHOW PROGRAM
SHOW PGM
SHOW ENTRY

Displays the name, starting address, entry address (if different from the starting address), and length of the current program.

SHOW MEMBER

Displays the default membername to be used in the LOAD command.

SHOW DDNAMES  [ LIKE xxx ]

Lists the ddnames of the JCL cards supplied for this run.
LIKE xxx causes only ddnames starting with "xxx" to be listed.

SHOW ADDRESS <addrexpr>

Displays the value of the <addrexpr> in formatted form, that is, with as much symbolic information as possible.
SHOW RETCODE
Displays (again) the return code of the program which has terminated.

SHOW REGS
SHOW GPERS
Displays all fixed-point registers in formatted form.

SHOW PERS
SHOW FPERS
Displays all floating point registers.

SHOW PERS
Display the comparison tolerance for floating-point operations.

SHOW VERSION
Indicates which version of DEMON is being used.

SHOW PARS [HORT|PORT|ASH|LEED|GO]
Displays the default parameter string for the given processor.

SHOW PRINT
Display the current status of the session log.

SHOW XXXX
Any "SHOW" commands which cannot be recognized are sent to HILTRAN.

Examples: SHOW USERS
           SHOW TIN
MISCELLANEOUS COMMANDS

CLEAR (BREAKPOINTS)

BREAKPOINTS: clear all breakpoints in the current program.

TO XXX text

Send a message to XXX. (Same as normal WYLBUR "TO" command)

WYLBUR

HILTHE

Return to WYLBUR/HILTHE temporarily. DEMON will not terminate. You may return to DEMON by typing its name as a WYLBUR/HILTHE command.

LOGOFF

LOGON

Terminate this session and DEMON. You will be logged off from WYLBUR/HILTHE as well. If a program is running asynchronously, DEMON will not be terminated and you may logon again later.

LOGOFF DEMON

Logoff from the DEMON subsystem, but not from WYLBUR and HILTHE. DEMON will not be terminated and you (or someone else) may logon again later. The fuse on the DEMON timebomb (q.v.) is still lit, however.

GO AWAY

EXIT

Terminate DEMON and return to WYLBUR.

SUGGEST text

Send a suggestion to LJS (about DEMON, presumably). The text may be anything up to 130 characters long.
UNATTENDED EXECUTION

Although DEMON was primarily designed to be a fully interactive system, there are cases in which it is desirable to have DEMON load and execute programs without requiring anyone to logon to the subsystem. At any time later during the execution of the program, you can then sign on to the subsystem, examine the progress of the program, and use any of the debugging facilities within DEMON. You may then either sign off the subsystem and allow the program to continue, or terminate the program.

To use DEMON in this mode, the normal JCL is used except that:

1) PARM='NOHAIL' is supplied on the EXEC card which starts DEMON, and

2) The DD card //INISH should specify a file which contains the DEMON commands necessary to start the program. A simple example of such a file is:

    //INISH DD *
    load wyl.ec.pub.loadmods$ap1 on cat go

The subsystem name started for an unattended DEMON run will be the full jobname rather than your initials followed by "DEM0N". As the job owner you may signon to the subsystem at any time; others may sign on (one at a time only!) if they can supply the magic password. The password is initially "INATHDS", but may be specified ("MAGIC-xxxx") in the EXEC card para. The subsystem name may also be changed to other than the jobname by specifying "NAME-xxx" in the EXEC card para.
RUNNING NON-PORTERAN JOBS UNDER DEMON

Although DEMON was primarily designed for FORTRAN-based jobs, it can often be useful with non-FORTRAN jobs. Assembler-language jobs are particularly suited, since among all the language processors only it, in addition to FORTRAN, produces symbol table information in the object deck. However any load module, with or without symbol table information, can be executed under DEMON.

Before loading a non-FORTRAN program, you should issue the command "SET MODE NOFORTRAN". This prevents DEMON from trying to find the standard entry points of the FORTRAN execution-time package, and causes it to issue a SPIN for the loaded program. In order to allow breakpoints to work, the program should not issue a SPIN of its own unless it passes the invalid operation interrupt up to the previous SPIN exit routine. (Tip: after loading, you can use the FIND command to find the SPIN (GAON) and zap it to a NOP. I often use this technique to run DEMON under DEMON.)

MISCELLANEOUS NOTES

1) The timebomb

DEMON will issue warning messages at five minute intervals if you never sign on, or if it has been idle for 10 or more minutes. After two warnings, it will self-destruct. The timebomb is defused if a program is running asynchronously.

2) The INISH file

A file of initial DEMON commands may be included with the JCL under the ddbname 'INISH'. This can be used to "personalize" DEMON by changing defaults. There should be only one command on a line, which is treated exactly as if the line had been typed at the terminal. The file containing the initial commands may be in the input stream ("DD *") or in a disk dataset (a member of your UNIV8 library, for instance).

The INISH file is also used to supply commands for running DEMON without any interaction with the terminal; see the section on "UNATTENDED EXECUTION".
APPENDIX I - JCL FOR RUNNING HORMON

See the section on "How To Run It" for possible modifications to this JCL.

// JOB
//*MAIN CLASS=1,SYSTEM=SYA
//DEMON HENC PCN=DEMON,REGION=320K,COND=(5,12)
//*
//* DEMON JCL
//*
//* THIS IS IN WYL.CG.LJS.ISSDEMON ON CAT
//* DOCUMENTATION FOR DEMON IS IN WYL.CG.LJS.DOC6DEMON ON CAT
//*
//* THIS JCL IS SUBJECT TO CHANGE, SO IF YOU MAKE A LOCAL COPY,
//* BE SURE TO KEEP IT UP-TO-DATE.
//*
//*STPLIBS DD DSNAME=WYL.CG.LJS.DEFMON,DISP=SHR (DEMON)
// DD DSN=WYL.CG.LJS.THREADS,DISP=SHR (FORTX, LEXBX)
// DD DSN=WYL.CG.AJC.HORTM,DISP=SHR (HORTM)
//PPRTOUT DD SYSPRT=A (FOR 'PRRT' COMMAND)
//PPRTOUT DD SYSPRT=A (FOR 'SRT PRINT ON' LOG)
//SYSPRT DD SYSPRT=A (NOT USED BY ANYTHING, USUALLY)
//SUGGESTS DD DSN=WYL.CG.LJS.HOSUGG,DISP=SHR
//INIT DD DUMMY INITIAL DEMON COMMANDS
//WYLO01 DD UNIT=DISK,DISP=SHR
//*
//* FORTRAN JCL
//*
//*FORTRAN DD UNIT=SYS0,SPACE=(LOK, (10,10)) (SYSIN)
//*FORTRN DD UNIT=SYS0,SPACE=(LOK, (50,25)) (SYSPRINT)
// DD DBC=BLKSIZE=3425
//*FORTRU DD SYSPRT=A (SYSBATCH)
//*FORTRN DD UNIT=SYS0,SPACE=(LOK, (10,5)) (SYSTEM)
//*SYSTEM DD UNIT=SYS0,SPACE=(LOK, (10,5)),DBC=BLKSIZE=1093
//*SYSPRT DD UNIT=SYS0,SPACE=(LOK, (75,25))
//*SYSPRT DD UNIT=SYS0,SPACE=(LOK, (50,25))
//*SYSPRT DD UNIT=SYS0,SPACE=(LOK, (50,25))
//*SYSPRT DD UNIT=SYS0,SPACE=(LOK, (20,10))
// DD DBC= (RECFM=FB,LBCE=80,BLKS=600,DUPNO=2)
//*
// FORTRAN JCL (THIS IS FCB FORTRAN, RESERVED WORD VERSION)
//*
//*HT01P001 DD DSN=WYL.CG.AJC.HAC.HACRESERVE,DISP=SHR RESERVED WORD
//*HT05P001 DD UNIT=SYS0,SPACE=(LOK, (20,10))
//*HT06P001 DD UNIT=SYS0,SPACE=(LOK, (20,10))
// DD DBC=(RECFM=FB,LBCE=133,BLKSIZE=3458)
//*HT07P001 DD UNIT=SYS0,SPACE=(LOK, (20,10))
// DD DBC=(RECFM=FB,LBCE=80,BLKSIZE=3520)
// ASH JCL
//
// ASHLIB DD DSN=SYST.HCLIB,DISP=SRE  (SYSLIB)
// DD DSN=SYST3.HCLIB,DISP=SRE
// ASIN DD UNIT=SYSDA,SPACE=(TEK,(20,10))  (SYSLIN)
// ASKPRINT DD UNIT=SYSDA,SPACE=(TEK,(20,10)), (SYSPRNT)
//       DCB=BLKSIZE=3509
// ASHPUN DD SYSCUR-T (SYSPUNCH)
// ASTTBR DD UNIT=SYSDA,SPACE=(TEK,(20,10)), (SYSTBIN)
//       DCB=BLKSIZE=1573
//
// LKED JCL
//
// LKEDOBJ DD UNIT=SYSDA,SPACE=(TEK,(50,25))  (SYSLIN)
//        DCB=BLKSIZE=3200
// LKEDMOD DD UNIT=SYSDA,SPACE=(TEK,(30,10,2)), (SYSLMOD)
//        DSN=SYSLIB(MAIN)
// LKEDLIB DD DSN=YL.CLI.LJS.LOADLIBS,DISP=SRE  (SYSLIB)
//        DD DSN=SYST1.FILIB,DISP=SRE
//        DD DSN=SYST3.FILIB,DISP=SRE
// LKEDPRT DD UNIT=SYSDA,SPACE=(TEK,(20,10))  (SYSPRT)
//        DCB=BLKSIZE=12160
// LKEDTBN DD UNIT=SYSDA,SPACE=(TEK,(10,5))  (SYSTBN)
//
// USER PROGRAM JCL
//
// PT06F001 DD UNIT=SYSDA,SPACE=(TEK,(20,10))
// PT07F001 DD UNIT=SYSBA,SPACE=(TEK,(20,10))
// PT05F001 WILL BE NOT NEEDED IF IT IS TO BE LEFT
// Assigned to the terminal.
// SYSUDUMP DD SYSCOUT=A
APPENDIX II - SAMPLE SESSION

This is an example of a simple DEMON session, with running commentary.

? use demo on wyl18
? p

1. INTEGER TABLE(25)
2. C
3. DO 1 I=1,25
4. 1 TABLE(I)=I**2
5. C
6. CALL TEST(TABLE,6)
7. CALL TEST(TABLE,2)
8. CALL TEST(TABLE,8)
9. CALL TEST(TABLE,4#)
10. CALL TEST(TABLE,5#)
11. STOP
12. END
13. C
14. SUBROUTINE TEST(TABLE,VAL)
15. INTEGER TABLE(25),VAL
16. CALL SEARCH(TABLE,25,VAL,IPOS)
17. WRITE(6,2) VAL,IPOS
18. 2 FORMAT (1X,I8,' IS AT POSITION ' ,I4)
19. RETURN
20. END
21. C
22. SUBROUTINE SEARCH(TABLE,N,VAL,IPOS)
23. INTEGER TABLE(1),VAL
24. INTEGER IPOS,N,LOW,HIGH
25. LOW=1
26. HIGH=N
27. IPOS=S
28. 1 IF (HIGH.LT. LOW) RETURN
29. IPOS=(LOW+HIGH)/2
30. IF (VAL.LT. TABLE(IPOS)) GO TO 4
31. IF (VAL.GT. TABLE(IPOS)) GO TO 5
32. RETURN
33. 4 HIGH=IPOS
34. GO TO 1
35. 5 LOW=IPOS+1
36. GO TO 1
37. END

List the program we will debug.

This is a binary search algorithm
with a minor bug planted in it.
This algorithm, incidentally, is infamous
for being conceptually very simple but
likely to be programmed incorrectly the
first time by almost everyone.

(Maint: the bug is here!)

Get the JCL for DEMON
We need no addition, so just run it.
After a (hopefully short) wait...

LJSDEMON is the subsystem name
WELCOME! THIS IS THE VERSION OF 8/6/18/77
- compile demo on wy1919
  P.ORT-S
  - list errors
FORTRAN I EXTENDED COMPILER ENTERED
*STATISTICS* SOURCE STATEMENTS = 11, PROGRAM SIZE = 525, SUBPROGRAM NAME = MAIN
*STATISTICS* NO DIAGNOSTICS GENERATED
***** END OF COMPILATION *****
*STATISTICS* SOURCE STATEMENTS = 7, PROGRAM SIZE = 352, SUBPROGRAM NAME = TEST
*STATISTICS* NO DIAGNOSTICS GENERATED
***** END OF COMPILATION *****
*STATISTICS* SOURCE STATEMENTS = 19, PROGRAM SIZE = 456, SUBPROGRAM NAME = SEARCH
*STATISTICS* NO DIAGNOSTICS GENERATED
***** END OF COMPILATION *****

- link
  LIRED-S
- load
  - sho map nosys
MAIN #4925CS $283
TEST #4927C3 $166
SEARCH #492928 $1C3
- go
WRITE(6):  6 IS AT POSITION 3
WRITE(6):  2 IS AT POSITION 1
MAIN IS SUSPENDED AT $492A48 (SEARCH,#1+#38)
- sho trace
$492A48 (SEARCH,#1+#38)
SEARCH
TEST, ISN 4
MAIN, ISN 7
  - sho var in search
N I#4
LOW I#4
VAL I#4
HIGH I#4
IPOS I#4
SEARCH N#4
#1 LAB
#4 LAB
#5 LAB
  - sho var in search lik i
IPOS I#4
  - sho var val
N$25
LOW#G
VAL#G
HIGH=#1
IPOS=1
SEARCH-UNDEF
  - sho var val=1
LOW
HIGH
IPOS

Friendly, isn't it?
Start compiling the program
In luck, no errors.
If there were, this would show them.

Linked the program with standard libraries
No unresolved references, etc.
So load it.
Show the absolute loading map
(but omit the standard system routines)

Start execution
Look good!
So does this.
But nothing happens for a while so I hit ATTN
Where are we?

What variables are accessible by name

(# means a FORTRAN statement number)
The list could be restricted
or the values shown
(Notice that the routine name was omitted)
(so it means "SAME as before")

"show all variables whose value is 1"
- declare search_table(25) at main_table.type integer

Array params aren't in subroutines, so add it
It is outside the SEARCH subroutine.
Display the value of the created symbol
It displays all array values

We could also begin in the middle
Back to debugging: set a breakpoint
and start up again
A breakpoint hit.
Continue.
Again. Seems like we've got the loop.
Yup.
Change it to display some variables
without stopping, and try again.

Hmm. Seems to be a problem.

Time to hit ATT.
Now I see what the problem is, so
back to WYLBUR to fix the source code.

A small change.

Save it back.

Return to DEMON.
Recompile, but this time don't wait.
(Our old program was still loaded)
While it compiles, we can fiddle around
(Or even go to WYLBUR).

Compile is done.
Start the link editor.
More thumb twiddling. What DDCARDS are there?

Done.
Load it.
So for broke.
Made it!

Normal termination.

Let's fool around with some breakpoints.

Set another

but make it inactive.

Now reexecute with breakpoint #1 active
but doing a "display" rather than stopping.

For dump lovers...
You can always reload if things get messed up, and start again.

The variables in the main program...

They can be changed at will.

And displayed.

Subroutines can be called directly from DEMON.

Forgot about the breakpoint.

Disable it and continue.

(This output also goes to PSYCH1)

NORMAL return from called routine

Back to WYLBUR; the DEMON job is finished.
APPENDIX III - HISTORY

Most of DEMON was written in an intense 3-week period in late January and early February of 1975, primarily due to the instigation of Marty Briedenbach in SLAC Experimental Group C. It received occasional use for the next two years, during which time I made no changes except to put the year into the version date printed when you sign.

In early 1977 some substantial and many minor changes were made in response to comments received in those two years. Starting with that set of changes (and hopefully from now on) the changes introduced with each new version will be listed here.

******** CHANGES IN THE VERSION OF 14 JAN 77 ********

1. Expressions must be preceded by "D" ("DISPLAY"). This removes a number of syntactic funny.

2. Programs may be run "asynchronously", i.e. you may do other things (go to WILBUR, for example) while your program is compiling.

3. DEMON can be told to start your job without waiting for you to logon, so it can be used to run production jobs that can then be monitored at any time.

4. Arrays are displayed in a nicer format from the DISPLAY command.

5. SHOW VARS will also search for variables with certain values only.

6. FIND sets a variable which remembers where the thing was found.

7. (For assembler freaks) Register notation has been changed to $ to agree with ORVIL.

8. Users other than the job's owner may logon (one at a time only!) if a password is supplied.

9. The print log of the terminal session is open by default.

10. The prompt character may be changed.

11. The DUMMY command will cause subroutines to return immediately.
12. The command WILBUR can be abbreviated by "W".
13. The assembler used for ASH is ASHE rather than ASHE.
14. Pseu...dophases (like "FORT") are used instead of 
"RC" for the returncode message.
15. A SPIE is issued for non-FORTRAN programs.
16. An INPUT function was added.
17. Undefined variables are printed as UNDEF rather than 
strange numbers.
18. Commands which wipe cut a program which has not yet 
terminated will ask "OK TO CLEAR PGH?".
19. FIND has a LINES option so that it can be made to stop 
after the first hit.
20. "SHOW PBFNS" or "SHOW PFRS" display floating point 
registers.
21. LOGICAL and LOGICAL*1 added as valid types for 
DECLARE.

************ CHANGES IN THE VERSION OF 12 FEB 77 ************
1. The CALL command has been added to allow subroutines 
with arguments to be called from the terminal.
2. The DISPLAY and LST commands are now valid from within 
breakpoint strings.
3. The SHOW VARIABLES command now has a VALUE option which 
causes the value as well as the name of all variables 
which satisfy the other options (IN csect, LIKE xxy) to 
be displayed. Also SHOW VAR UNDEF will display the 
names of all undefined variables.
4. The EDIT command now saves the original data so that 
ATTN can be used to abort the edit and revoke any 
changes made. (This corresponds more precisely to the 
WILBUR EDIT command.)
5. The command LOGOFF DENON will allow you to logoff from 
DENON only. You will still be connected to 
WILBUR/ONYX/HILTON, and DENON will wait for someone 
else to logon.
6. Various new abbreviations (in addition to the normal 
3-character-or-more rule) and synonyms have been added.
APPENDIX IV - IDEAS FOR CHANGES TO DEMON

(This list is not ordered, nor should appearance on this list be taken as a commitment to implement any item.)

1) GO should be allowed to have an optional \texttt{<addr>expr} indicating that a branch should be taken. This should be used with care, since FORTRAN programmers generally don't know what assumptions the compiler has made about register contents at various points in the program.

2) Some information about CPU time used should be available - both for DEMON and for the executing program. If possible, send warning messages if the executing program appears to be in a loop.

3) There should be a "command retry" mode analogous to ULYSSE's.

4) There should be an interpretive execution mode, whereby memory locations could be monitored for stores and/or fetches. (This is not as far-fetched as it may sound!)

5) Allow concatenated input for MONT, COMPILE, ASH, and LINK. Perhaps something of the form \texttt{COMPILE &STUFF AND MONTCHAP ON ULYUS4 AND * where \texttt{*} means "take input from the terminal".}

6) It has been pointed out that the automatic setting of the default CSECT (what \texttt{"*"} means) is both confusing and inconvenient. There should perhaps be a \texttt{SET CSECT ccccccc} command which specifies the default CSECT.

8) (This also applied to ABACUS) I have been thinking about ways to implement a macro facility which will also serve as a vehicle for user-defined functions. For DEMON, it would provide a "cataloged procedure"-like mechanism, whereby a single command would cause a series of DEMON commands to be executed. (\texttt{"FONTECCLG" might be convenient, for example.) Anyone with ideas on this subject is encouraged to communicate them.

L. Shustek 2/16/75
4/08/75
1/14/77
2/12/77