- SNOOPY -

INTERFACE PROGRAM

between VAX or microVAX and SSP

TRIUMF VERSION

Original SLAC version by

BRIGITTE M. BRICAUD

SLAC

JUNE 10, 1986

This document modified for TRIUMF version of SNOOPY by

Suzannah Daviel

TRIUMF

4004 Wesbrook Mall, Vancouver, B.C., Canada

Computer mail address:

SUZ@TRIUMFCL

February 27th, 1987

CHAPTER 1

INTRODUCTION

The SLAC Scanner Processor (or SSP) is a general-purpose, high-speed, programmable FASTBUS master (see reference 1) used originally by the SLAC Mark II group. It provides a general and powerful means of moving and processing data in a FASTBUS system.

The instruction set of the SSP is a subset of IBM assember with some extensions for accessing FASTBUS. At SLAC, the Mark II and SLD groups have been writing code for the SSP in IBM Fortran, IBM Assembly language or SLAC VAX Cross Assembly language. (In order to write code in IBM FORTRAN, or IBM assembly language directly, an IBM machine is needed to compile the source code; where an IBM machine is not available, a cross assembler is necessary). The object modules produced are translated into microinstructions that are executed by the SSP processor. Load modules can be translated, with or without a main program. They are linked with special code including an error handler (forming an "SSP system") and downloaded to the SSP.

The SSP transfers data to the Host for further processing.

The program SNOOPY (written by Brigitte Bricaud, SLAC) runs on the host (VAX or microVAX) and provides a means of testing and debugging SSP code.

1.1 HISTORY OF TRIUMF VERSION

The SSP has been chosen for data acquisition by the collaboration for

---

REFERENCE 1

"The SLAC scanner processor: a FASTBUS module for data collection and processing", H.Brafman, T.Glanzman, A.J.Lankford, J. Olsen and L.Paffrath, IEEE Trans. Nuclear Science, Vol NS-32, No. 1, February 1985, 336-340.

the BNL787 experiment. SSP modules have been in use both at BNL and TRIUMF (by members of the BNL787 group and others).

At TRIUMF, one of the SLAC methods of programming the SSP was implemented. The SLD group's VAX cross assembler and SSP debugger program SNOOPY, SSP system software from the MKII group and SSP diagnostic programs, were all obtained from SLAC.

SNOOPY was modified to produce versions that run with host-to-FASTBUS interfaces available at TRIUMF, i.e. the CERN CFI (CERN FASTBUS INTERFACE) and the TRIUMF IORFI, by R. Poutissou and A. Daviel. Only the CFI version will be discussed further (and referred to as TRIUMF SNOOPY).

Extra features were added to TRIUMF SNOOPY (such as a breakpoint handler and additional symbolic debugging aids), some of which existed in the FORTH SSP programs at SLAC. This work was done by A.J. Daviel.

Meanwhile, at BNL, the MPS group also chose to use the SSP for their experiment. One of their members, R. Hackenburg, developed a VMS MACRO cross assembler, and an SSP "operating system" and error handler (referred to as the "BNL system" or BNLSYS) based on the SLAC system (see BNLSSP:SSP_PROGRAMMER.MEM for details). He also developed a number of FASTBUS access routines, including BSI (Buffered Segment Interconnect) routines that permit access to other FASTBUS segments through a chain of SSPs.

This "BNL method" formed an alternative to the SLAC cross assembler and system (the "SLAC method"). The BNL787 collaboration decided to adopt the BNL method for BNL787 SSP code.

Therefore, support for SSP load files produced by the BNL method, and access to other segments using BSI routines was added to the TRIUMF SNOOPY by R. Poutissou, while retaining the support for files produced using the SLAC method (such as SLAC SSP diagnostic code).

1.2 TRIUMF SNOOPY

The TRIUMF version of SNOOPY is installed at TRIUMF on the microVAX (KAON::) and at BNL on the VAX750 (BNL787::). In summary, TRIUMF SNOOPY can download and support code written using any of the following languages:

- o SLAC VAX Cross Assembly language
- o BNL VAX MACRO cross-assembly language

    (produced on a VAX or microVAX)

- o IBM Assembly language
- o IBM Fortran

(produced on an IBM machine running VM)

NOTE

The source codes of the first three methods are incompatible; although all use the IBM assembler instruction set, the format is quite different. The formats of the load files are also incompatible.

TRIUMF SNOOPY has useful symbolic debugging aids and a breakpoint handler. It also supports SSPs connected by a cable segment using the BSI method developed at BNL (see 787DOC:SSP_PROGRAMMER.MEM for details). This may preclude support in SNOOPY for the SLAC SSP cable segment method; this has not been investigated at TRIUMF.

This document describes the features of TRIUMF SNOOPY, henceforth referred to as "SNOOPY".

CHAPTER 2

SSP REGISTERS

A summary of the SSP registers is listed here for convenience. For more information about the SSP hardware, see 787DOC:SSPDOC_T.FORLIS.

2.1 SSP PROGRAM STATUS WORD (PSW)

The initial PSW is stored in data memory 0 prior to starting the processor and dumped to data memory 5 by the SSP Linker.

INTR	EXCP	TOUT	SS	BREQ	EG	ASAK	MS	RD	WAIT	EX	NPA	SMSK	OMSK	CC	PC (program counter)

bits 0:11 - PC = program counter
12:13 - CC = IBM encoded condition code

14 - OMSK = IBM overflow mask
15 - SMSK = IBM system mask
16 - NPA = Next PROM address (multiple cycle)

17:18 - EX = 0 if divide by zero
1 if overflow
2 if unsupported opcode

19 - WAIT = WT being asserted at time of error
20 - RD = 0 if write in progress at time of error
1 if read in progress at time of error

21:22 - MS = Mode select bits 0,1 at time of error

23 - ASAK = ASAK lock present at time of error
24 - EG = Address cycle in progress at time of error
25 - BREQ = Arbitration cycle in progress at time of error

26:28 - SS = Slave status at time of I/O error

29 - TOUT = Timeout during FASTBUS operation
30 - EXCP = Exception (see EX bits description)
31 - INTR = External interrupt (attached as FASTBUS slave)

2.2 SSP CONTROL STATUS REGISTER SPACE (CSR)

CSR (0000,0000) = Mfg. ID. and control/status bits

Bit#	Power-on value	Meaning on READ when bit set	Meaning on WRITE when bit set
0	0		
1	0	Crate selected for RB	Select crate for RB
2	0	RUN enabled	Enable RUN state
3	0	RB asserted	Assert RB
4	0	Crate selected for SR	Select crate for SR
5	0	SR asserted	Assert SR
6	0	Single step enabled	Enable single step
7	0	GKUP (maintain GK)	Enable GKUP
8	0	Crate seg. selected	Select crate segment
9	0	SR enabled (cable)	Enable SR (cable)
10	0	Timeout inhibited	Enable TO inhibit
11	0		Start processor
12	?	CLASS device	>Set 'N' (CLASS bits)
13	?	defined on 4 bits	>
14	?	used in broadcast	> if enabled
15	?	case #1	>
16	0	Mfg. ID. (106 hex)	Enable SET CLASS bits
17	1	Mfg. ID.	Select cable for RB
18	1	Mfg. ID.	Disable RUN state
19	0	Mfg. ID.	Lower RB
20	0	Mfg. ID.	Disable SR (cable)
21	0	Mfg. ID.	Lower SR
22	0	Mfg. ID.	Disable single step
23	0	Mfg. ID.	Disable GKUP
24	1	Mfg. ID.	Select cable segment
25	0	Mfg. ID.	Disable SR (cable)
26	0	Mfg. ID.	Disable TO inhibit
27	0	Mfg. ID.	
28	0	Mfg. ID.	
29	0	Mfg. ID.	
30	0	Mfg. ID.	
31	0	Mfg. ID.	

CSR (0000,0008) = Arbitration level

2.3 SSP PROGRAM MEMORY SPACE

The following locations in PM space are reserved by the hardware:

CSR (4000,0000) through CSR (4000,0FFF) = 4K words
Reserved locations

FASTBUS CSR Space address	CPU program memory address	Contents
4000-0000	0	Exception handler trap addr
4000-0001	1	Reserved
.	.	.
.	.	.
.	.	.
4000-007F	7F	Reserved

2.4 SSP DATA MEMORY SPACE

2.4.1 SLAC reserved DM

The SLAC reserved locations in DM are listed below (see SLAC MK/II FASTBUS NOTE #9).

SLAC MKII SSP Data Memory reserved locations

SSP DM byte address (hex)	Contents
00	Initial PSW and error PSW
04	Updated Memory address register (MAR)
08	Error PSW
0C	Interrupt handler address (if >0)
10	Error return address (if > 0)
14	Original PSW
18	Error handler return code
1C	Program done flag
20	HALT data (for WCSR0)
24	Error handler control word
28	6 words user defined
40	Register dump area (16 words)
80	Total entries to error handler
84	Total divide-by-zero errors
88	Total overflow errors
8C	Total unsupported opcodes
90	Total block-count-exhausted
94	Total I/O errors: arbitration
98	Total I/O errors: address
9C	Total I/O errors: data
A0	7 words reserved for error handler
BC	Total external interrupts
C0	No. times SSP IPLed at PM80
C4	PM location of most recent halt
C8	13 words reserved
FC	reserved for HALT macro
100	reserved for SNOOPY debugger (16 words)
140	SSP interrupt control block (16 words)
180	SSP interrupt message block (16 words)

2.4.2 BNLSYS reserved DM

Reserved locations in DM for the BNL system are shown here. See 787DOC:SSP_PROGRAMMER.MEM for more details.

SSP Data memory address (hex)	Contents
00	SSP starting PSW
04	Updated MAR
08	Saved error or interrupt PSW.
10	User-specified error-return address.
18	Error code
1C	Halt code
20	Datum for the HALT instruction.
2C	Datum for the PAUSE instruction.
30	Current contents of register 13
3C	The number of DM words in this SSP.
40	Register dump area (16 words)
80	Total errors counter
84	Divide-by-zero errors
88	Overflow errors
8C	Unsupported opcodes
90	Block-count-exhausted
94	Arbitration time-outs
98	Primary address time-outs
9C	I/O cycle time-outs
A0	Wrap-around errors
A4	Unknown time-outs
A8	Slave busy (SS=1) errors
AC	Slave done (SS=2) errors
B0	SS=3,4,5 (unknown code)
B4	Slave data errors (SS=6,7)
B8	Unknown Fastbus errors
C4	Address of instruction following last
CC	Word-address of lowest free PM word.
D4	Lowest free DM location
D8	Highest free DM location
E0	4 words reserved for breakpoint
F0	BSI DM base address
F4	BSI PM system start address.
F8	System save register area (6 words)
140	Reserved 16 words (internal control block)
180	Reserved 16 words (internal message block)
1C0	PA of SSP on the crate segment
1C4	PA of SSP on the cable segment
1C8	Reserved until DM200

CHAPTER 3

RUNNING SNOOPY

SNOOPY is written in VAX FORTRAN, and is linked with the CERN CFI library routines (in CFILIB:), and the BNLSYS standard routines (in BNL$SSP:). All the SNOOPY code is found in the directory FBSN:

The CFI version of SNOOPY can run using a FASTBUS CFI card connected to the VAX (or microVAX) either

- via a UFI (UNIBUS to FASTBUS Interface) or QFI (QBUS to FASTBUS interface) respectively
  or

- via a CFI CAMAC interface

Where hardware for both these possibilities is available, two link files are provided,

- o FBSN:SNOOPY_U.COM creates a UFI/QFI version

- o FBSN:SNOOPY_C.COM creates a CAMAC version

NOTE

> The term "CFI" throughout this document is used to mean a CERN FASTBUS INTERFACE in general (that is, via a QFI, UFI or CAMAC coupler). The type of FASTBUS interface used is both irrelevent and virtually transparent to the SNOOPY user.

To run SNOOPY, enter :

```
$SNOOPY
```

This will run the default version for the system (normally the UFI/QFI version).

First, SNOOPY identifies itself by typing :

```
     Welcome to SNOOPY *** Version 8.04 *** 3-Mar-1987
```

Then, it prompts you to set the parameters which define the Interface:

```
     Enter SSP Segment (crate) number (CR= 0) :
     Enter SSP Geographical Address (CR= 9) :
```

and waits for an answer.

If the parameters are set to the correct value, enter <CR>. If you want to change a parameter, enter the new value, followed by <CR>.

The message:

```
     UFI/CFI initialized and FB software reset
```

should then appear.

SNOOPY then prompts you to enter a command :

```
     Command :
```

Enter the name of the command you want to execute. If the string does not match a valid command, SNOOPY prints the message "Invalid syntax" and waits for a new command.

CHAPTER 4

SNOOPY'S COMMANDS

4.1 DEFINITIONS

The following definitions will be used throughout:

- All answers must be terminated by pressing the Return key which is symbolized by <CR>.
- dddddd represents a Decimal value.
- hhhhhh represents a Hexadecimal value.
- cccccc represents an Ascii character string.
- [abcd] represents an optional parameter.

4.2 LIST OF COMMANDS

```
@FNAME[.CMD] ........................... Execute a command file
BEGIN  ................................. Re-start SNOOPY
RESET  ................................. Reset FASTBUS interface
CSR0   ................................. Display SSP CSR0 content
HELP   ................................. Display list of commands
DIR    [directory] ..................... Display list of SSP files
STOP   ................................. Exit SNOOPY
QUIT   ................................. Exit SNOOPY
DUMP   ERRORS .......................... Dump system error counters
DUMP   FILE filename[.SSP] ............. Dump SSP disk file
DUMP   COMMON /LIST .................... Dump list of commons
DUMP   COMMON name [start [nwrds]] .... Dump contents of COMMON
DUMP   PM | DM [start [nwrds]] ........ Dump contents of PM or DM
       Dump options : none ............ Output to Terminal
                      (DISK ........... Output to Disk
                      (PRINT .......... Output to Printer
WRITE  PM | DM [start [nwrds [value] .. Write a pattern into PM or DM
LOAD   filename[.SSP] [(VERIFY ........ Load program from disk
EXEC   filename[.SSP] ................. Load program and execute
SAVE   filename[.SSP] ................. Save program onto disk
EDIT   ................................ Examine, deposit, execute,
                                                 dump registers
```

4.3 @COMMAND FILE

A "command file" is a VAX disk file created by any editor. It contains variable-size records, but the maximum length of each record is limited to 128 characters. Each record must contain one single command, that may be followed by a comment separated from the command itself by a semi-colon character ";". "Blank records" or records beginning with ";" are ignored; they are treated as comments. Mutiple spaces between words of a command are ignored.

The file is defined by its name, with a default type = .CMD; it MUST be specified in the command line after the character "@", can be adjacent to it or separated from it by a space.

When Edit commands are specified in a command file, their format is slightly different, and will be described in Section 5.4.

4.4 BEGIN

This command allows you to re-define the parameters of the Interface (see above). This command is NOT ALLOWED in a command file.

4.5 RESET

This command allows you to initialize the interface.

4.6 CSR0

This command displays the contents of the SSP Control Register (CSR0) on the Terminal :

```
SSP control register = hhhhhhhh
```

4.7 HELP

This command displays the list of commands on the Terminal.

4.8 DIR [DIRECTORY]

This command displays on the Terminal a list of the SSP binary load files (i.e. *.SSP) available under the directory specified in the command line. If a directory is specified, it may be a straight name or a logical name, and it must not contain any embedded blanks. If no directory is specified, the default directory is searched.

4.9 STOP AND QUIT

These commands terminate the execution of the SNOOPY program and return control to the VAX operating system.

4.10 DUMP

This command has the following different possibilities :

4.10.1 DUMP ERRORS

Dump contents of error counters. The reserved Data memory locations DM80 to DMB8 are dumped. These contain the system error counters for both the SLAC and BNL systems.

4.10.2 DUMP FILE

DUMP FILE filename[.SSP] (option

e.g. DUMP FILE 787EXAMPLES:TEST.SSP

Dump contents of a program saved on disk. If "filename" is not specified, you will be prompted.

Labels for DM and PM locations (if available - see LOAD command) will also be reloaded on a DUMP FILE command, and a message will then be printed, e.g. :

```
 Loading symbols from
787EXAMPLES:TEST7_K7.MAP
```

4.10.3 DUMP COMMON /LIST

DUMP COMMON /LIST (option

e.g. DUMP COMMON /LIST (PRINT

Dump list of COMMONs used by the program that has been loaded into the SSP. If no program has been previously loaded, SNOOPY types the message

"Invalid command - no COMMONs in table",

and returns to Command mode.

Note that in the case of code produced by the BNL method, the COMMONS are the same as MARKER BLOCKS.

4.10.4 DUMP COMMON name

DUMP COMMON name [start_location [number_of_words]] (option

e.g. DUMP COMMON FRED FFFF 10 (PRINT

Dump contents of COMMON "name" defined in the program that has been loaded into the SSP. If no program has been previously loaded, SNOOPY types the message

"Invalid command - no COMMONs in table",

and returns to Command mode.

If "name" is not specified, you will be prompted.

- o Start_location = BYTE offset from Common starting address.
- o Maximum number of words = common block size in words.
- o Parameters must be entered in Hexadecimal.
- o Default for "start_location" is Common starting address.
- o Default for "number_of_words" is Common size.

Note that in the case of code produced by the BNL method, the COMMONS are the same as MARKER BLOCKS.

4.10.5 DUMP PM | DM

DUMP PM [start_location [number_of_words]] (option

e.g. DUMP PM 200 500 (DISK

Dump contents of a selected portion of Program or Data memory.

- o Start_location = WORD address (PM) - BYTE address (DM).
- o Maximum number of words = 4096 (PM) - 32768 (DM) for small memory SSP.
- o Parameters must be entered in Hexadecimal.
- o Default for "start_location" is 0.
- o Default for "number_of_words" is (maximum - start).

4.10.6 DUMP Options

The following options are supported for DUMP:

- none = output is directed to Terminal - this is the default.
- DISK = output is directed to Disk - filename = "SSPhhhhh.DMP".
- PRINT = as DISK above (file can be printed later)

4.11 WRITE

This command writes a pattern into Program or Data memory.

1. WRITE PM [start_location [number_of_words [pattern]]]

   e.g. WRITE PM 100 50 E0E0E0E0

o start_location = WORD address.
o maximum number of words = 4096 (1000(hex)).
o pattern = value to be written.

2.
```
WRITE DM [start_location [number_of_words [pattern]]]

        e.g. WRITE DM 100 50 E0E0E0E0
```

o start_location = BYTE address.
o maximum number of words = 32768 (8000(hex))
o pattern = value to be written.

For both WRITE PM and DM :

- Default for "start_location" is 0.
- Default for "number_of_words" is (maximum - start).
- Default for "pattern" is 0.
- Parameters must be entered in Hexadecimal.

Then, SNOOPY types :

```
Write PROGRAM [DATA] MEMORY starting at hhhhhh(hex) ...
...  for hhhhh(hex) words with pattern = hhhhhhhh

Enter Ok(=CR) or Quit :
```

The Program or Data memory will NOT be modified, if you answer "q(uit)" or "Q(UIT)" to this prompt.

After execution, SNOOPY returns to command mode.

4.12 LOAD OR EXEC

e.g. LOAD 787EXAMPLES:TEST7_K7

These commands download a program, previously linked by the VAX SSP Linker and saved on disk, into the SSP memory. The program is not executed unless the EXEC command is given. If "filename" is not specified, you will be prompted :

```
Disk filename (default type =.SSP) :
```

If you enter "<CR>" only, the command is aborted.

After the file has been loaded, a number of messages appear, stating the type of load file (BNL or SLAC type), the size of the SSP memory, the lowest free location in DM (in bytes) and in PM (in bytes and in words).

For example, for a BNL file (i.e. one created using the BNL method) the following messages may appear on the screen:

```
 Loading symbols from
787EXAMPLES:TEST7_K7.MAP
 This is a BNL .SSP file
Downloading a  32K DM SSP.
SSP program: TEST7_K7
The SSP system is: BNLSYS  , version: 4.00 .
Lowest free DM location:  00013774
Lowest free PM location:  3618 ( D86).
Number of marker words in use:   767
```

After the program has been loaded into the SSP memory,

- o if the "LOAD" command is used with the option (VERIFY, SNOOPY compares the contents of the SSP Program and Data memory with the values read from the disk file and types an error message if they do not match.

- o if the "EXEC" command is used, the processor is started and the program executed.

Then, SNOOPY types "*" and enters the Edit mode.

4.12.1 DM,PM Labels

In the case of loading a file created with the BNL method, SNOOPY will look (in the same directory) for a file with the same name and a .MAP extension (i.e. a map file). If this exists, SNOOPY will read in the values of certain global symbols, making some assumptions to decide which are DM labels and which are PM labels.

The following message will then be printed:

```
 Loading symbols from
787EXAMPLES:TEST7_K7.MAP
```

These symbols will then appear as labels in listings created with the edit commands "DM" or "PM", e.g.

```
*dm200
 $COMMAND          DM00000200= 54494E49     1414090313 "INIT"
 $MASTER_ID        DM00000204= 00000204            516 "...."
 $TRIGGER          DM00000208= 00000208            520 "...."
 $PR_EVENT         DM0000020C= 0000020C            524 "...."
 $SPILL            DM00000210= 00000210            528 "...."
 $RUN              DM00000214= 00000214            532 "...."
                   DM00000218= 00000000              0 "...."
                   DM0000021C= 00000000              0 "...."
*pm97
                   PM00000097= 50D00030   ST       13,48
 M_LOAD_CMD        PM00000098= 58300200   L        3,512
                   PM00000099= 5930D04C   C        3,76(0,13)
                   PM0000009A= 477000AC   BC       7,172
```

4.13 SAVE

This command saves a program previously loaded into the SSP memory onto disk. This command is NOT ALLOWED in a command file.

NOTE

> Regardless of the format of the original load file, the new file has the same format as an SSP file created by the VAX SSP Linker.

If "filename" is not specified, you will be prompted :

```
     Disk filename (default type =.SSP) :
```

If you enter "<CR>" only, the command is aborted.

Then, SNOOPY will prompt you for the parameters that define the program:

```
     Program load address (word) :
     Program length (words) :
     Data Load address (byte) :
     Data constants length (bytes) :
     Program start address (word) :
     Data start address (byte) :
     Error handler address (word) :
     Number of common blocks :
```

All parameters must be entered in hexadecimal. If the number of common blocks is not 0, you will be prompted to define them :

Common name :
Common address (byte) :
Common length (bytes) :

SNOOPY then reads the SSP Program memory and Data memory, saves the program as a disk file and returns to Command mode.

4.14 EDIT

This is the Edit mode of SNOOPY which allows the user to:

- o Examine CSR space, Data space, Program or Data memory, one word at a time. Program memory instructions are de-assembled and displayed in symbolic form.
- o Deposit into CSR space, Data space, Program or Data memory, one word at a time.
- o Search for a specified pattern in Program or Data memory. Program instructions are de-assembled and displayed in symbolic form.
- o Execute a program in Continuous Run Mode.
- o Display contents of PSW and CSR0.
- o Dump or Display contents of the SSP general registers.

SNOOPY tells you that it is entering the Edit Mode by typing a star on the left margin of the next line:

```
*
```

Then, it waits for a command.

Each Edit command is defined by two letters that can be followed by optional parameter(s), then by <CR>.

The edit commands are described in Chapter 5.

CHAPTER 5

THE SNOOPY EDITOR

The Snoopy editor is invoked from SNOOPY with the EDIT command (see above).

5.1 LIST OF COMMANDS

The individual edit commands are listed below:

```
BKhhh ........ Set a breakpoint at PM address "hhh".
               "hhh" is a word address, and must not be 0.

CBhhh ........ Cancel a breakpoint at PM address "hhh".

CShhhhhhhh ... Examine and/or Deposit into CSR space.
               "hhhhhhhh" is a WORD address - default=0.

DMhhhhhhhh ... Examine and/or Deposit into Data memory.
               "hhhhhhhh" is a BYTE address - default=0.

DShhhhhhhh ... Examine and/or Deposit into Data space.
               "hhhhhhhh" is a WORD address - default=0.

PMhhh ........ Examine and/or Deposit into Program memory.
               "hhh" is a WORD address - default=0.

SPhhhhhhhh ... Search for pattern "hhhhhhhh" in Program memory, or
or SP"cccc ... search for ASCII string "cccc" in Program memory

SDhhhhhh ..... Search for pattern "hhhhhh" in Data memory or
or SD"cccc.... search for ASCII string "cccc" in Data memory

SB............ Show all breakpoints

PR............ Proceed from a breakpoint

QU ........... Exit from Edit mode.

ST............ Step from a breakpoint

GO ........... Start processor as it is.

GOhhh ........ Start execution of program at address hhh.
               Program counter field in DM0 is overwritten.
               "hhh" is a WORD address and NOT 0.

SH ........... Display contents of PSW and CSR0.

DR ........... Dump and display contents of general registers.

TR ........... Display contents of general registers.

XT ........... Exit from Edit mode.
```

5.2 PM AND DM

PM and DM in direct Edit mode - examine and deposit into Program or Data memory respectively.

SNOOPY opens the location defined in the command, prints its contents, and waits for input.

If you don't want to change this value, just enter <CR> and SNOOPY will display the contents of the following location, If you want to modify it, simply enter the new value and terminate by <CR>.

An ASCII character string can be deposited, by preceding the string with a double quote ("), e.g. "ASCI. All lower-case input is converted to upper-case.

The following example illustrates opening a PM location, changing it, and verifying the change:

```
*PM97<CR>                                               (open PM location)
         PM00000097= 50D00030   ST  13,48      <CR> (no change, open next)
LABEL_1  PM00000098= 58300200   L   3,512      41300000<CR> (change)
         PM00000099= 5930D04C   C   3,76(0,13) <CR> (no change, open next)
         PM0000009A= 477000AC   BC  7,172      PM97<CR> (open PM97)

         PM00000097= 50D00030   ST  13,48      <CR> (verify)
LABEL_1  PM00000098= 41300000   LA  3,0        <CR> (verify)
         PM00000099= 5930D04C   C   3,76(0,13) <CR> (verify)
```

The following example illustrates opening a DM location, changing it to an ASCII string, and verifying the change:

```
*DM200
$COMMAND   DM00000200= 00000200 512        "...."   "init (update)<CR>
$MASTER_ID DM00000204= 00000204 516        "...."   <CR> (no change)
$TRIGGER   DM00000208= 00000208 520        "...."   DM200 (open DM200)

$COMMAND   DM00000200= 54494E49 1414090313 "INIT"   <CR> (verify)
```

- When examining Program memory, the contents are displayed in Hexadecimal and as assembly language instructions.
- When examining Data memory, the contents are displayed in hexadecimal, decimal and as ASCII characters.
- If a location cannot be opened for reading, SNOOPY will display "FFFFFFFF".

5.3 CS AND DS

Examine and deposit into CSR or Data space respectively.

These commands work similarly to PM,DM commands (see above).
Valid addresses for CS option (CSR space) are :

```
CS0 = SSP control and status register ...... R/W
CS8 = Arbitration level .................... W/O
CS400000000-CS400000FFF = Program memory ... R/W
CS800000000-CS80001FFFC = Data memory ...... R/W
```

NOTE

In the case of the command "CS" where "hhhhhhhh"=0 (examine CSR0), SNOOPY returns to Edit mode immediately after the location has been examined or modified. If you want to modify it, SNOOPY checks if bit RB [=8(hex)] is set, and if it is, prompts you

Bit RB set - replace (Yes/No) ?

The location will be modified ONLY if you answer "Yes" to this prompt.

5.4 EDIT IN COMMAND FILE

CS,DS,PM,DM in Indirect Edit mode (i.e. inside a command file)
- Examine or deposit into CSR space, Data space, Program or Data memory.

When in Edit mode inside a command file, you can either read the contents of a location (examine) OR modify it (deposit). Each record contains one single Edit command, which has the following format :

```
Examine : "CShhhhhh"   ===>   SNOOPY types CShhhhhh=hhhhhhhh
          "DShhhhhh"   ===>   SNOOPY types DShhhhhh=hhhhhhhh
          "PMhhhhhh"   ===>   SNOOPY types PMhhhhhh=hhhhhhhh
          "DMhhhhhh"   ===>   SNOOPY types DMhhhhhh=hhhhhhhh

Deposit : "CShhhhhh=hhhhhhhh"   ===>   SNOOPY writes.
          "DShhhhhh=hhhhhhhh"   ===>   SNOOPY writes.
          "PMhhhhhh=hhhhhhhh"   ===>   SNOOPY writes.
          "DMhhhhhh=hhhhhhhh"   ===>   SNOOPY writes.
```

SNOOPY recognizes that the command is a "write" (deposit) command by looking for the character "=" inside the command line. If this character is not present, then SNOOPY assumes that the command is a "read" (examine) command and displays the contents of the location on the screen.

NOTE

- The same restrictions described in section 5.3 apply to CSR space.
- NO space character is allowed between the "=" sign and the parameter that follows it. If there is one, SNOOPY will write (deposit) "0" into the specified location.
- If a location cannot be opened for reading, SNOOPY will display "FFFFFFFF".

5.5 SP,SD - SEARCH FOR PATTERN

Search for pattern in Program or Data memory,

    e.g. SDFFF* EA0 F00

These commands can be followed by two optional parameters "start_add end_add", which define the zone of memory to be scanned.

The pattern is either

- a hexadecimal number of up to eight digits, which may contain a "*" as a wild-card at either end of the string,

  e.g. FFF*, E0, FFFFFFFF, *A43, ababab, *F3*

  Note that strings such as AA*B are not supported.

- or a 4-character ASCII string preceded by a double quote (")
  e.g. "ASCI, "12bg

All lower-case characters are converted to upper-case before the search is begun.

The action of the search command is best illustrated by the following table. The search pattern in column 3 of the table is compared with the memory contents in columns 1 or 2. If the search pattern matches the memory contents, the result is TRUE, a match has been found and the memory location and contents will be output. If the result is FALSE, either the pattern did not match, or an illegal pattern was specified.

SEARCH FOR PATTERN

Memory Contents hexadecimal	ASCII	Search for pattern	Result
544E5645	EVNT	544E5645	TRUE (exact match)
544E5645	EVNT	*E5645	TRUE (wildcard)
544E5645	EVNT	544E*	TRUE (wildcard)
544E5645	EVNT	*44E5*	TRUE (wildcard)
544E5645	EVNT	"EVNT	TRUE (exact match)
544E5645	EVNT	"EV*	FALSE (unsupported)
544E5645	EVNT	544*645	FALSE (unsupported)
544E5645	EVNT	544XX645	FALSE (unsupported)
544E5645	EVNT	544	FALSE (no match)
00000034	....	34	TRUE (exact match)
00012345	....	123	FALSE (no match)

Examples of the search command are shown below. The carriage returns following each command have been omitted for clarity.

1. Search DM for the ASCII string "INIT" in the range DM100 to DM500

```
*SD"init 100 500
     DM00000200= 54494E49     1414090313 "INIT"
```

2. Search DM in the range DM100 to DM500 for hexadecimal numbers whose 3 least significant digits are "E49".

```
*SD*e49 100 500
     DM00000200= 54494E49     1414090313 "INIT"
     DM00000260= 4D494E49     1296649801 "INIM"
     DM000004A4= 49494E49     1229540937 "INII"
```

3. Search PM in the range PM80 to PM90 for a hexadecimal value whose two most significant digits are "58"

```
sp58* 80 90
   PM00000080= 58D00014   L        13,20
   PM00000084= 58F000F0   L        15,240
   PM0000008C= 5860D084   L        6,132(0,13)
   PM0000008E= 58C0D048   L        12,72(0,13)
   PM00000090= 58A0D090   L        10,144(0,13)
```

4. Search PM in the range PM80 to PM90 for a hexadecimal value whose containing the digits "C0"

```
sp*C0* 80 90
   PM00000088= 41C00634   LA       12,1588
   PM0000008B= 98EBC000   LM       14,11,0(12)
   PM0000008E= 58C0D048   L        12,72(0,13)
   PM0000008F= 50C00268   ST       12,616
```

SNOOPY decodes the pattern to be searched for, reads the Program or Data memory in blocks of 4096 words, according to the zone of scanning that was specified with the command, and displays all locations that contain a matching value.

When the Program memory is scanned, the instructions that match the pattern are de-assembled and displayed in symbolic form.

By default, the entire memory is scanned. After 4096 words have been scanned, SNOOPY prompts you to ask if you want to continue or return to Edit mode.

NOTE

In the case of a pattern of hexadecimal digits (i.e. one not preceded by a double quote (")), any non-hexadecimal digit is unacceptable, and such a string is not supported. The following are examples of unacceptable strings:

FF*BE, INIT, FFOEA

However, no error message will indicate that the string is unacceptable. The result will be the same as if the string has not been found.

5.6 BREAKPOINTS

The breakpoint utility is designed for debugging programs, and will preserve the registers and PC (program counter). Since there is no single SSP instruction which does this, it is necessary to use three instructions per breakpoint. This leads to complications.

It is possible to set breakpoints and to step through code using SNOOPY, subject to certain restrictions. The breakpoint process works by replacing 3 lines of PM code with breakpoint code, saving the original code in the host. Therefore a listing of the PM obtained when a breakpoint has been set will not show all the original code. When a breakpoint is reached, the processor halts, and a dump of the registers is performed.

The breakpoint commands are:

BKhhh	set a breakpoint at PMhhh
CBhhh	cancel a breakpoint at PMhhh
SB	show all breakpoints
PR	proceed from a breakpoint
ST	step from a breakpoint

5.6.1 BK - Set a breakpoint

A breakpoint may be set anywhere in PM except for the following restrictions:

- o Control must not be transferred to the second or third instruction of a breakpoint sequence. Thus a breakpoint must never span a label to which a branch instruction is expected to go.
- o Breakpoints MUST be spaced apart by at least 3 PM instructions (since each breakpoint consists of 3 instructions) UNLESS they are entered in reverse order (i.e. break instructions BK96,BK95,BK93 will work, but BK93,BK95,BK96 will not).

5.6.2 CB - Cancel breakpoint

This command will cancel a breakpoint that has previously been set using the "BK" command.

5.6.3 SB - Show breakpoints

This will list the positions in PM of all breakpoints that have previously been set using the "BK" command.

5.6.4 PR - Proceed, ST - Step

It is ESSENTIAL to use "PR" to PROCEED from a breakpoint. "GO" would restart from the address given by bits 0-11 of DM0, skipping two instructions.

Similarly, "ST" (step) will only work from an existing breakpoint. NEVER step over a breakpoint that you have already put in. If you are approaching a breakpoint using "ST", either cancel it ("CB"), or proceed to it ("PR").

It is not possible to step from the starting address of the program (i.e. bits 0-11 of DM0). To achieve this effect, set a breakpoint at the beginning of the program (usually PM80), then "GO", then step, i.e.

```
*BK80
*GO
*ST
```

If breakpoints or steps have been used, and it is desired to restart the program from the beginning, you must restart the processor with "GO80" not "GO", since using breakpoints or steps overwrites the initial contents of DM0.

Either stepping or proceeding (from a breakpoint) deletes the initial breakpoint; thus a breakpoint can only be used once. For example, when stepping around a loop, the breakpoint disappears after you have reached it. So it must be replaced each time (when you have stepped more than 3 instructions away).

The "ST" command will step through a BC (branch on condition) instruction, decoding the condition codes in order to decide which branch to take.

"ST" will step over BAL and BALR (branch and link) instructions, which are used for subroutine calls. It will therefore have the effect of a " STEP " rather than " STEP/INTO " in VAX debugger terminology - the subroutine will be called and executed; stepping will be resumed after the return.

"ST" does not decode BCR, BCT, BCTR, BXH or BXLE (in fact any of the branch instructions except BC). So stepping through these is to be avoided.

5.6.5 Error message

The message

" Error - must step from a break; break not found "

when attempting to step or proceed usually indicates a problem caused by

- Placing a breakpoint over a label to which control has branched.
- Stepping over a breakpoint
- Stepping over a branch instruction (except BC)
- Illegally overlapping breakpoints

These problems are discussed above. Recovery from this usually means reloading the program and starting again.

5.6.6 Example

The following example of an editing session using SNOOPY demonstrates setting and cancelling breakpoints, and the use of the Step, Proceed and Go (ST,PR and GO) commands. The carriage returns following each command are omitted for clarity.

1. Show the original contents of PM90-PM92 using the "PM" command

```
*pm90
          PM00000090= 58A0D090   L        10,144(0,13)
          PM00000091= 50A004EC   ST       10,1260
          PM00000092= 41A00000   LA       10,0
```

2. Set a break point at PM90, dump the contents of PM90-PM92 to show that the original instructions have been replaced by the 3-instruction breakpoint code.

```
bk90                     ( set a breakpoint at PM90 )
*pm90
          PM00000090= 50E000E0   ST       14,224
          PM00000091= 09E00000   BALR0    14,0
          PM00000092= 47F00064   BC       15,100
```

3. Set a break point at PM97 in the same way

```
          PM00000097= 50D00030   ST       13,48
          PM00000098= 58300200   L        3,512
          PM00000099= 5930D04C   C        3,76(0,13)
bk97                                ( set a breakpoint at PM97)
*pm97
          PM00000097= 50E000E0   ST       14,224
          PM00000098= 09E00000   BALR0    14,0
          PM00000099= 47F00064   BC       15,100
```

4. Set a break point at PM9C - show breakpoints - cancel a breakpoint

```
bk9C                           ( set a breakpoint at PM9C )
*sb                            ( show breakpoints)
Break # 1 at 00000090
Break # 2 at 00000097
Break # 3 at 0000009C
*cb9F                          ( cancel a breakpoint at 9F )
Break not found                ( no such breakpoint )
*cb9c                          ( cancel a breakpoint at 9C )
*sb                            ( show breakpoints )
Break # 1 at 00000090          ( breakpoint at 9C has gone )
Break # 2 at 00000097
```

5. Start the SSP - and stop at first breakpoint

```
*dm0                           ( display contents of DM0 )
 $PSW0            DM00000000= 00008080          32896 "...."

go               ( start the SSP at contents of DM0 )
*sh              ( Display Break and PSW and CSR0 )
                 ( - have stopped on the break point at PM90)

break at  90  58A0D090   L        10,144(0,13)
 Show:   Retcod=00000000 - PSW=00008093 - CSR0=0106F110
```

6. Step then proceed to next break point

```
*st                               ( step from breakpoint)
break at  91  50A004EC   ST       10,1260  ( stops at PM91)
*st                               ( step from breakpoint)
break at  92  41A00000   LA       10,0     ( - stops at PM92)
*sb                            ( show breakpoints )
Break # 1 at 00000092
Break # 2 at 00000097
*pr               ( proceed from a breakpoint to next
                                        breakpoint )
*sh               ( Display Break and PSW and CSR0 )
break at  97  50D00030   ST       13,48
Show: Retcod=00000000 - PSW=0000809A - CSR0=0106F110
```

7. Step, then proceed to end of program

```
*sb                            ( show breakpoints )
Break # 1 at 00000097
*st                            ( step to PM98 )
break at  98  58300200   L        3,512
*st                            ( step PM99 )
break at  99  5930D04C   C        3,76(0,13)
```

```
    *pr    (proceed to next breakpoint
           or end of program if no more breakpoints)
    *dm0                         ( display contents of DM0 )
     $PSW0            DM00000000= 00008099         32921 "...."
```

8. Restart processor, and run program wihout breakpoints

```
    go80    ( use GO80 command as DM0 does not contain 8080 hex )

    *sh
     Show:   Retcod=00000000   -   PSW=00008080   -   CSR0=0106F132
```

5.6.7 Breakpoint code in PM

Five lines of special code are also downloaded into the SSP starting at the PM address 64 (given by PM_P_BRKADD_1 in BNLSYS). Two locations in DM starting at E0 (given by DM_P_BRKPT in BNLSYS) are also used to store R14. The register dump area is at DM40 (given by DM_P_RDUMP in BNLSYS).

These particular locations are acceptable for both the SLAC and BNL SSP "operating systems", so that the SNOOPY breakpoint handler will work with both SLAC and BNL SSP files.

5.7 GO - START PROCESSOR

If the command "GO<CR>" is given, the processor is started as it is; SNOOPY assumes that DM0 has been previously loaded with the starting address of the program.

If the command "GOhhhhhh<CR>" is given, SNOOPY overwrites the 12 low bits of DM0, (the Program Counter field), with the specified address and starts the processor.

Then, SNOOPY re-enters the Edit mode.

5.8 SH - SHOW

"SH" displays contents of PSW and CSR0. If the processor has halted at a breakpoint (see above) the PM address of the current breakpoint is also displayed, and the original contents of the PM location (i.e. the instruction which the processor will execute after the next step or proceed command is given).

SNOOPY displays on the Terminal (if not halted at a breakpoint)

```
Show : CSR0 = hhhhhhhh - PSW = hhhhhhhh - Retcod = hhhhhhhh

where CSR0   = Control Status Register (CSR space location 0).
      PSW    = Processor Status Word = contents of DM0.
      Retcod = Contents of DM18 (word location 6).
               This is the return code value of a program
               after it has been executed. This parameter
               has no meaning otherwise.
```

and re-enters the Edit mode. If the SSP is halted at a breakpoint, SNOOPY will display (for example)

```
     break at  97  50D00030   ST       13,48
     Show: Retcod=00000000 - PSW=0000809A - CSR0=0106F110
```

Here the SSP is halted at a breakpoint at PM97; the next instruction it will execute (on a step "ST" or proceed "PR" command) is also shown.

5.9 DR - DUMP REGISTERS

"DR" dumps and displays contents of General registers.

The SSP contains 16 general registers, whose contents are dumped into Data memory locations DM40 through DM7C, (i.e. starting at DM_P_RDUMP in BNLSYS) and then displayed on the Terminal.

Locations PM10-PM11-PM12 are saved, then loaded with the DUMP program, and the processor is started. Then, SNOOPY reads locations DM40 through DM7C and displays them. After having restored the program memory locations, SNOOPY re-enters the Edit mode.

5.10 TR - TYPE REGISTERS

This command is equivalent to the DR command, except that SNOOPY assumes that the contents of the general registers have already been dumped into locations DM40 through DM7C (i.e. starting at DM_P_RDUMP in BNLSYS). It reads these locations, displays them on the Terminal, then re-enters the Edit mode.

5.11 XT AND QU - EXIT FROM EDIT MODE

These commands exit from the SNOOPY editor, and return control to the command mode.