SPIRIT-30 DEBUGGER

USER MANUAL

Sonitech International Inc
14 Mica Lane, Suite 208
Wellesley, MA 02181 , USA

TABLE OF CONTENTS

CHAPTER 1
STARTING UP

1.1 HARDWARE AND SOFTWARE REQUIREMENTS

The following hardware and software environment is necessary for SPIRIT-30 Debugger:

- IBM PC, XT, AT and compatibles with SPIRIT-30 board
- DOS 3.0 or later
- 640K memory
- Hercules graphics card or IBM compatible EGA card
- TMS320C30 Assembler and Linker

The following systems are optional depending on the applications:

- TMS320C30 'C' Compiler.

A recommended system for applications development and debugging is the following:

- IBM AT or compatible
- 40 MEG Hard drive
- 640K memory
- TMS320C30 'C' compiler

1.2 SPIRIT-30 DEBUGGER PACKAGE

SPIRIT-30 Debugger package consists of one 5.25 inch, 360K diskette.

1.3 INSTALLATION

STEP 1 : Backup your Diskettes

- Make a backup copy of the diskette in this package and store the original disk in a safe place. Disk copy can be done using the DOS command DISKCOPY (refer to your DOS manual for more information on this command).

STEP 2 : Board Installation

- Please follow the instructions in the SPIRIT-30 *Technical Reference Manual* included with the SPIRIT-30 board.

STEP 3: Debugger Software Installation

* Setting up the Environment

In order to use the software, certain commands must be placed in the AUTOEXEC.BAT file in the root directory on the hard disk. To do this:

1. Use a text editor to edit the file AUTOEXEC.BAT in the root directory on the hard disk.

2. Within the file, there should already be a line of the form:

```
PATH=XXXXXX
```

If this line is present, then add the string ;C:\SPIRIT30\EDSP to the end of this line, so that it appears:

```
PATH=XXXXXX;C:\SPIRIT30\EDSP
```

If there was no PATH command already in the file, then add the line:

```
PATH=C:\SPIRIT30\EDSP
```

3. Add the following line to your AUTOEXEC.BAT file:

```
SET SPIRIT_DIR=C:\SPIRIT30
```

Note: If you choose not to use the base directory shown above, i.e., C:\SPIRIT30, then use the name of your SPIRIT-30 base directory.

* Installation

To install the software on your hard drive, follow these steps:

1. Insert the disk labeled **SPIRIT-30 : EDSP and Debugger** diskette 1 in drive A and enter the command:

```
A:INSTALL
```

2. Now, follow the installation program prompts.

1.4 MEMORY DESCRIPTION AND ADDRESSING CONVENTIONS

The TMS320C30 on SPIRIT-30 board provides on-chip memory that includes two blocks of 1K x 32-bit RAM, which is refered to as 'internal RAM'. Apart from this, SPIRIT-30 has 32K x 32-bit words of dual access on-board memory which is refered to as 'external memory'.

Following is the convention used by Debugger to address SPIRIT-30 board and PC(host) memories:

NAME	DESCRIPTION	ADDRESS RANGE
EX	SPIRIT-30 board external memory	0x000000 to 0x007fff
M0	SPIRIT-30 board internal RAM bank-0	0x809800 to 0x809bff
M1	SPIRIT-30 board internal RAM bank-1	0x809c00 to 0x809fff
PC	IBMPC (host) memory	

CHAPTER 2
BASICS

2.1 PRELIMINARIES

The debugger executable code has two parts: PC-Debugger and DSP-Debugger. PC-Debugger runs on the IBMPC (host) whereas DSP-Debugger runs on the SPIRIT-30 board. **DSP-Debugger uses the SPIRIT-30 external memory locations C0H - 3ffH, and the TRAP0, TRAP1, TINT0, and INT0 vectors. Hence, the user programs cannot be loaded in the locations below 400H, and cannot make use of the above mentioned vectors.** A sample '.cmd' file is provided along with the package, which can be used to generate a valid '.out' file that can be loaded in the debugger.

In general, the DSP processors do not check and recover from invalid instructions, so the processor state is unknown if an invalid instruction is executed. Hence care must be taken to execute only valid instructions, otherwise unpredictable results may occur.

2.2 SPIRIT-30 PROGRAM EXECUTION

Begin Debug session by typing "SPIRIT30" .

This will set the Debugger window environment and start the Debug session.

If you want to exit the Debug session select QUIT option by pressing 'Q'or 'q'.

Before we proceed further, let's take a minute to understand the Menu / Window selection strategy.

2.3 MENU / WINDOW STRATEGY

The main command line on the Debugger screen appears as shown below:

```
Display   Execute   Inspect   Register .... Quit
```

These options (Display, Execute, ..) are referred to as [Modules] and can be easily identified in this manual by square brackets. A module can be selected by pressing the corresponding highlighted letter. For example, [Inspect] module can be selected by pressing 'I' or 'i'.

Once a module is selected, it may give another list of sub-modules or functions to choose from. For example, selecting [Inspect] option would print the following command line:

```
  SetBP   Address   Edit   View_file
```

These options (SetBP, Address,..) are referred to as {functions}, and can be easily identified in this manual by curly brackets. A function can be selected by pressing the corresponding highlighted letter. For example, {Address} function can be selected by pressing 'A' or 'a'.

A typical function selection will open a window with several options and parameters. This is explained in more detail in the section WINDOW EXAMPLE.

The Debugger screen has six parts: Command line (top most line), Message line (bottom most line), 'Code display' or 'Inspect' window (top left), Command explanation window (bottom left), 'extended precision registers' window (top right), and 'cpu status' window (bottom right).

NOTATIONS:

The following notations are used throughout the documentation:

[module]
- [...] is module name.

{function}
- {...} is a function name.

<Parameters>
- <...> is a window parameter name.

Example:
<1 Source :> (window parameters in DISPLAY window)
<2 Address :>

'input'
- '...' is user input.

NOTE: all user input must follow 'ENTER' key.

KEYBOARD CONTROL:

The following are the key controls for the screen:

<ESC>
- Will exit from the present screen menu to the previous state.

<ENTER>
- All input for the <Parameters> must be followed by this key.

<BACK SPACE>
- To delete the inputs

WINDOW ERROR MESSAGE:

Error messages are displayed on the bottom of the screen. The following is the commonly observed error message in Debugger.

"Error in input.."
This means there is an error in the user input. If this error occurs, type the correct input.

The other frequently displayed messages are explained in Chapter 3.

2.4 DEBUGGER PARAMETERS

Following are the frequently used parameters in the Debugger:

<Source:>

This signifies the origin of the data in the memory. The valid options for <source> are:

(see MEMORY DESCRIPTION AND ADDRESSING CONVENTIONS for the following memories and their respective ranges)

'EX' - refers to the SPIRIT-30 external memory.

'M0' - refers to the SPIRIT-30 internal RAM bank-0.

'M1' - refers to the SPIRIT-30 internal RAM bank-1.

'PC' - refers to the memory on IBMPC.

'FI' - File, refers to IBMPC hard disk or floppy files.

'fi test.dat' - refers to source as file and filename as test.dat.

<Address :>

Address of the memory location. In case the <Source> is 'FI'(file), <Address> corresponds to offset of the file. (see section on MEMORY DESCRIPTION AND ADDRESSING CONVENTIONS for valid addresses)

<Datatype:>

These are valid inputs for data type :

'H'	-	Hex	80000000
'F'	-	IEEE float	0.0000000E+000
'I'	-	Integer	-2147483648
'U'	-	Unsigned Integer	3458718999
'T'	-	TI float	1.0000000E+000

<Filename:>

Files used in Debugger are program files and data files. In [Load] option, the valid filename must be DSP executable 'COFF' file prepared by TMS320C30 linker. The default extension for program files is '.out' and for data files is '.dat'. The data file should have the header or else the error message 'invalid file header info' will appear. See Appendix A-1 for the format of data file header.

2.5 WINDOW EXAMPLE

STEP 1: *Start the Debugger session by typing "SPIRIT30" and 'ENTER'.*

STEP 2: *Select the [Memory] module by pressing 'M' or 'm'.*

STEP 3: *Select {Block Fill} option* by typing 'B' or 'b'.

STEP 4: *Fill 100 DSP words in External memory with a value of 1324 starting at address 1200H.*

- Select item 1 <source> by typing '1' 'ENTER'.

- Valid options for source will be displayed. Type 'EX' 'ENTER' to select external memory as source.

- Select item 2 <Start Address> by typing '2' 'ENTER' or 'down arrow' key.

- Type '1200' and hit 'ENTER' and the default value should get updated to 1200.

- Keep the default value for item 3 <Address Increment>.

- Select item 4 <Count> by typing '4' 'ENTER'.

- Type '100' for the <Count> value hit 'ENTER'. The <End Address> should get updated to '1263'.

- Select item 6 <Data Type> by typing '6' 'ENTER'.

- Type 'U' and hit 'ENTER' for unsigned integer type.

- Select item 7 <Data Value> by typing '7' 'ENTER'.

- Type '1324' for <Data Value> and hit 'ENTER'.

- keep the default values for all the other options.

- Select item 0 <Fill> by typing '0' 'ENTER'.This will fill the given value in the external board memory starting at address 1200H.

- Hit <ESC> to exit from this window.

STEP 5: *Check the memory contents to be sure the above Block Fill operation was successful.*

- Select [Display] option by typing 'D'or 'd'.

- Enter <Source> as 'ex' (be sure to hit 'ENTER')

- Give <Start Address> as '11F0'

- Give <Data Type> as 'U'

- On the screen you should see the External memory values in decimal. Hit <ALT> H to change the format to Hex; Hit F9 to see other data display options.

- Hit <Up or Down arrow or Page Up or Page Down> to scroll the screen. Hit <ALT> U to display in decimal. Compare the displayed data with the data you loaded in previous STEPs.

- Hit <ESC> twice to go back to the main menu.

2.6 A QUICK RUN

The objective of the following tutorial is to understand how to debug a program using single step and breakpoint features.

TUTORIAL

In this turorial, we will
- Load DSP executable 'COFF' file into SPIRIT-30 external memory.
- Inspect the disassembled code.
- Clear memory and reset registers.
- Execute the program in single step mode.
- Set breakpoints and execute the program in breakpoint mode.
- Remove breakpoints, run the program and halt the execution.

NOTE: Print a copy of the file test.asm (in your debugger directory) and follow the procedure as described below.

STEP 1: *Load program and Inspect the text*

NOTE: The programs must be above 400H in order to load and execute them in Debugger.

- Select [Load] option by pressing 'L', and load the file 'test.out' (starts at 400H). Observe the PC (in 'CPU Registers' window) changes to 400H, and the inspect window shows the first few lines of the '.text' starting from 400H.
- Choose [Inspect] option by typing 'I', and use arrow / Pg keys to look at the disassembled code of test.out.
- Hit <Esc> to go to the main menu.

STEP 2: *Clear memory*

- Press 'M' to choose [Memory] option.
- Go to {Single Step Fill} mode by typing 'S', and give 'EX' as 'source', '1000H' as 'Starting address', 'H' (hex) as data type. Press <RETURN> four times to clear memory locations 1000H - 1003H. The test program updates these locations, and so it is better to clear them before executing the program.
- Hit <Esc> twice to go to main menu of {Debug}.

STEP 3: *Reset Registers*

- Choose [Execute] option by typing 'E'.
- Press 'T'{reseT};This clears the extended-precision, and auxiliary registers.

STEP 4: *Single step execution*

- Press 'P'{steP}; This executes the instruction at 400H. Observe the following:
    - * AR1 (in 'CPU Registers' window) changes to 1000H, which indicates that the instruction at 400H has been executed.
    - * PC changes to 401H and the instruction at 401H in the Inspect window is highlighted, indicating that the instruction at 401H will be the next one to be executed.
- Press 'P' five times to single step through five more instructions.
    - * Observe the new values of the registers: PC = 406H, R1 = 1234H, R2 = 2468H, AR1 = 1002H.
    - * Observe the instruction at 406H in Inspect window is now highlighted.
    - * Look at the memory locations 1000H and 1001H:
        - Hit <ESC>;Type 'D' [Display]; Select 'EX' as source, '1000' as starting address, 'H' (Hex) as data format. Observe locations 1000H and 1001H have changed to 1234H and 2468H respectively.
        - Hit <ESC> twice to go the main menu of {Debug}.

STEP 5: *Set Breakpoints*

- Press 'B' to go to [Breakpoint] option.
- Type 'A'{Add}, and enter '403' as address, to set breakpoint at 403H
- Type 'A'{Add}, and enter either '407' or 'LOOP' to set breakpoint at LOOP. Observe Inspect window - breakpoint number (#2) will be displayed against the instruction at 407H.
- Press 'B'{Breakpoint display} to display the breakpoints and press any key.
- Hit <ESC> to go to the main menu.

STEP 6: *Reset Memory and Registers*

- Repeat Steps 2 and 3.

STEP 7: *Execution in Breakpoint mode*

- Press 'R'{Restart} or 'S'{Start} with address '400'; This loads 400H into PC and starts execution. Observe the following:
    - * "Halted" appears on top right end of the screen indicating that the program has stopped executing at a break point.
    - * PC changes to 403H, indicating that the instructions 400H through 402H have been executed.
    - * Observe the new values of registers: R1 = 1234H, AR1 = 1000H.
    - * Observe the memory location 1000H has changed to 1234H (follow the steps given at the end of Step 4, and press 'E' to come back to Execute option).
- Type 'G'{Go} to continue execution from where it stopped before.Observe the registers and memory. The program would have halted at the second breakpoint (407H).
- Press 'G'{Go}; This resumes execution at 407H, executes the loop once, comes back to 407H and halts. Observe R4 has been incremented. If you press 'G' again, loop will be executed one more time, and R4 gets incremented.

STEP 8: *Halting the executing program*

- Hit <ESC> to go to the main menu.
- Press 'B'[Breakpoint] to go to Breakpoint option and type 'R'{Remove All}; This removes all break points.
- Hit <ESC>; Press 'E'[Execute].
- Press 'T'{reseT} to clear the registers.
- Type 'R'{Restart}; This loads PC with 400H and starts execution. Since there are no breakpoints, the program will be looping. "Press ESC to halt" appears on top right end of the screen.
- Hit <ESC> to halt the program. "Halted" appears on the screen. Observe the registers. R4 has a value, which is the number of times loop has executed before you halted the program.

CHAPTER 3
DESCRIPTION

3.0 DESCRIPTION

The following is the description of Debugger modules. Debugger has 9 major modules with some of the modules having several functions. The following is a list of Modules and each are described in detail in this chapter.

[Display] Module

[Execute] Module functions
- {Restart}
- {Go}
- {Start}
- {SteP}
- {ReseT}
- {SetPC}

[Inspect] Module functions
- {SetBP}
- {Address}
- {Edit}
- {View_file}

[Register] Module

[Memory] Module functions
- {Block fill}
- {Single step fill}

[Breakpoint] Module functions
- {Add}
- {Delete}
- {Remove-all}
- {Brkpt-display}

[Load] Module

[DOS] Module

[Quit] Module

[Display] Module

NAME: [Display]

OPERATION:

Show contents of SPIRIT-30/pc memory or data file in various data types.

PARAMETERS:

<Source> Refer to section on DEBUGGER PARAMETERS.
<Address> Refer to section on DEBUGGER PARAMETERS.
If an invalid (out of range) address is given, the closest valid address is used for the display.
<Data Type> Refer to section on DEBUGGER PARAMETERS.
If the source is a file then this input is ignored and file is displayed in the data type of the file (as given in file header)
<Filename> Refer to section on DEBUGGER PARAMETERS.
This input is meaningful if source is of type file.

MESSAGES:

F9 - see available data types.
Shows how to change the current display data type. Opens a window having the following information.

Press	To change display data type to
ALT-F	IEEE floating point
ALT-H	Hex
ALT-I	Integer (32 bit long)
ALT-U	Unsigned integer (32 bit long)
ALT-T	TI floating point

See also: DEBUGGER PARAMETERS.

[Execute] module:

NAME: {Restart} (Restart execution)

OPERATION:

This option allows you to restart execution (of a DSP executable 'COFF' code) from the start of the program.

PARAMETERS:

NONE.

COMMENTS:

This option does not modify any CPU register except the PC, which is loaded with the address of the start of program.

The message "Press ESC to halt" on the top right corner of the screen shows that the program is executing. Hit ESC key to halt the execution. After the program halts all the CPU registers and the inspect window are updated.

See also: [Load] module

[Execute] module:

NAME: {Go} (Continue execution)

OPERATION:

This option allows you to continue execution of a DSP program from the current PC.

PARAMETERS:

NONE.

COMMENTS:

The message "Press ESC to halt" on the top right corner of the screen shows that the program is executing. Hit ESC key to halt the execution. After the program halts all the CPU registers and the inspect window are updated.

See also: [Load] module

[Execute] module:

NAME: {Start} (Start execution)

OPERATION:

This option allows you to Start execution of a DSP program from the specified address.

PARAMETERS:

<Address>: Address (in hex) or 'label name' (string of characters) from which execution needs to be started.

COMMENTS:

This option loads the PC with the specified address, and executes the DSP program. The message "Press ESC to halt" on the top right corner of the screen shows that the program is executing. Hit ESC key to halt the execution. After the program halts all CPU registers and inspect window are updated.

MESSAGES:

If an invalid address/label name is entered, error message will be displayed on the message line. When this occurs, type the correct address/label name .

See also: [Load] module

[Execute] module:

NAME: {steP} (Single Step)

OPERATION:

This option allows you to single step an instruction of a DSP program from the current PC.

PARAMETERS:

NONE.

COMMENTS:

If you try to single step B*cond*D or DB*cond*D (delayed branch) instruction, the three instructions following the delayed branch will also be executed.

After executing one instruction all the CPU registers and the inspect window are updated.

MESSAGES:

TRAP*cond* instruction cannot be single stepped. An error message will be displayed if you try to single step this instruction. To solve the problem, control the execution using breakpoints.

See also: [Load] module, [Breakpoint] {Add} function

[Execute] module:

NAME: {reseT} (Reset Registers)

OPERATION:

This option allows you to clear the extended-precision (R0-R7) registers, auxiliary (AR0-AR7) registers, index registers (IR0-IR1) and status register (ST).

PARAMETERS:

NONE.

MESSAGES:

A message will be displayed stating which registers had been cleared.

[Execute] module:

NAME: {SetPC} (Modify PC value)

OPERATION:

This option allows you to modify PC value.

PARAMETERS:

<Value>: Number (in hex) / 'label name'.

[Inspect] module:

NAME: {SetBP} (Set a breakpoint)

OPERATION:

This option allows you to set a breakpoint at the specified address.

In [Inspect] module, you can use the Arrow keys or Pgup,Pgdn keys to scroll the inspect window.

PARAMETERS:

<Address>: Address (in hex) or 'Label name' (string of characters) at which breakpoint is to be set.

COMMENTS:

See [Breakpoint] {Add} function.

MESSAGES:

See [Breakpoint] {Add} function.

[Inspect] module:

NAME: {Address} (Inspect code from the address)

OPERATION:

This option allows you to inspect disassembled code from the specified address.

In [Inspect] module, you can use the Arrow keys or Pgup,Pgdn keys to scroll the inspect window.

PARAMETERS:

<Address>: Starting address (hex) / 'label name' of the code to be displayed in the inspect window.

COMMENTS:

Depending on the Address, it will read the instructions from external or internal memory of SPIRIT-30. See section 1.4 for valid address range.

MESSAGES:

An error message will be displayed if an invalid address is entered. If the error occurs, type the correct address or hit ESC to skip this option.

[Inspect] module:

NAME: {Edit} (Edit a memory location)

OPERATION:

This option allows you to edit specified memory location.

In [Inspect] module, you can use the Arrow keys or Pgup,Pgdn keys to scroll the inspect window.

PARAMETERS:

<Address>: Address (in hex) or 'Label name' (string of characters) of the location which needs to be modified.

<Value>: New value (in hex).

MESSAGES:

An error message will be displayed if an invalid address is entered. If the error occurs, type the correct address or hit ESC to skip this option.

[Inspect] module:

NAME: {View_file} (type an ascii file)

OPERATION:

This option allows you to look at an ascii file.

In [Inspect] module, you can use the Arrow keys or Pgup,Pgdn keys to scroll the inspect window.

PARAMETERS:

<Filename>: Name of the ascii file.

COMMENTS:

This option works similar to the DOS command "more < Filename".

[Register] module:

NAME: [Register]

OPERATION:

This option allows you to modify any of the CPU registers.

PARAMETERS:

<Register name>: Name of the register (string of characters).
<Datatype (h/f)>: 'h' for hex, 'f' for TI float (R0-R7)
<Value Exp>: exponent in hex (R0-R7)
<Value Man>: mantissa in hex (R0-R7)
<Value>:
For R0-R7 give TI float number,
For any other register give a hex number.

COMMENTS:

You cannot modify the registers RS, RE and RC. An error message will be displayed if you try to modify any of the above registers.

MESSAGES:

An error message will be displayed if an invalid register name or value is entered. If the error occurs, type the correct input or hit ESC to skip this option.

[Memory] Module:

NAME: {Block fill} (Fill block of memory)

OPERATION:

Fills the selected memory locations with given value of given data type.

PARAMETERS:

<1 Source>
Valid source inputs are 'ex','m0','m1','pc'. (see section SOURCE under DEBUGGER PARAMETERS)

<2 Start address>
Address in hex (see section 1.4 for valid address range)

<3 Address increment>
Address increment in +/- integer is the number of words skipped before two consecutive words are filled. A value of '1' will cause a contiguous block to be filled with address increment and value of '-1' will cause a contiguous block to be filled with address decrement.

<4 count>
Count in decimal is the number of words to be filled. If you are not sure of the count but know the end address you want to reach, you can skip this option and go to the next option. However, if count is given, end address will get updated accordingly.

<5 End address>
End address in hex is the end of the memory block to be filled. Option 4 (count) updates the end address, however, if end address input is given then count is updated accordingly.

<6 Data type>
Data type format of the data value to be filled. (see section DATATYPE under DEBUGGER PARAMETERS for valid data types)

<7 Data value>
Data value to be filled in above data type format.

<8 Data increment>
Data increment value in above data type format will increment data value at each address increment as follows: data = data + data_increment;

<9 Address fill width>
Address fill width in decimal is the number of memory words to be filled at every address increment. Width of '2' will fill the data at current address and next address with same data. Address fill width must not be greater than address increment.

<0 Fill>
Choose this to perform block fill.

Example:

1 source	:'ex'
2 start address	:'1000'
3 address inc	:'3'
4 count	:'5'
5 end address	:'1038'
6 data type	:'i'
7 data value	:'125'
8 data increment	:'10'
9 addr fill width	:'2'

Memory Map

```
0fff -> xxxxxxxxxxxx
1000 -> 125
1004 -> 125
1008 -> xxxxxxxxxxxx
100C -> 135
1010 -> 135
1014 -> xxxxxxxxxxxx
102C -> 165
```

[Memory] Module

NAME: {Single step fill}

OPERATION:

Fills a selected SPIRIT-30 memory location in specified data type. The fill address is highlighted and location is filled with the value entered.

PARAMETERS:

<Source> Refer to section on DEBUGGER PARAMETERS.
Error message is displayed if source is file.

<Address> Refer to section on DEBUGGER PARAMETERS.

<Data Type> Refer to section on DEBUGGER PARAMETERS.

Memory locations are displayed in the specified data type.

MESSAGES:

F9 - see available data types.Tells how to change the input and display data type by opening a window having following information..

Press	To change data type to
ALT-F	IEEE floating point
ALT-H	Hex
ALT-I	Integer (32 bit long)
ALT-U	Unsigned integer (32 bit long)
ALT-T	TI floating point

See also: DEBUGGER PARAMETERS.

[Breakpoint] module:

NAME: {Add} (Set a breakpoint)

OPERATION:

This option allows you to set a breakpoint at the specified address.

PARAMETERS:

<Address>: Address (in hex) or 'Label name' (string of characters) at which breakpoint is to be set.

COMMENTS:

Inspect window is updated and the breakpoint number is printed against the instruction at which breakpoint is set.

MESSAGES:

Message "Error in input.." is displayed if an invalid address is entered. If this error occurs, type the correct address or hit ESC to skip this option.

Message "Can't set anymore breakpoints" is displayed if total number of breakpoints exceeds 16. If this occurs, delete a breakpoint using [Breakpoint] {Delete} function and then set a new breakpoint.

Message "Can't set breakpoint at this instruction" is displayed if you try to set a breakpoint
--- At, or one instruction after RPTS
--- At RPTB
--- At, or one to three instructions after B*cond*D
--- At CALL*cond* <address>
--- At DB*cond* <address>
--- At, or one to three instructions after DB*cond*D
--- At location between 0H and 400H.

WARNING:

No error message is displayed if you set a breakpoint at any of the instructions following RPTB and belonging to its block. However, unpredictable results may occur if you try to execute the program.

[Breakpoint] module:

NAME: {Delete} (Delete a breakpoint)

OPERATION:

This option allows you to delete a currently set breakpoint.

PARAMETERS:

<Brkpt #>: Number of the breakpoint (integer) to delete.

MESSAGES:

Message "error .. no breakpoint set .." will be printed if an invalid breakpoint number is entered. If this error occurs, type the correct number or hit ESC to skip this option.

[Breakpoint] module:

NAME: {Remove-all} (Remove all breakpoints)

OPERATION:

This option allows you to delete all the currently set breakpoints.

PARAMETERS:

NONE.

MESSAGES:

Message "No breakpoints set" will be printed if there are no existing breakpoints. Hit any key to go out of this option.

[Breakpoint] module:

NAME: {Brkpt-display} (Display all breakpoints)

OPERATION:

This option allows you to display the numbers and addresses of the currently existing breakpoints.

PARAMETERS:

NONE.

MESSAGES:

Message "No breakpoints set" will be printed if there are no existing breakpoints. Hit any key to go out of this option.

[Load] module:

NAME: [Load] (Load a file)

OPERATION:

This option allows you to load a DSP executable 'COFF' file into SPIRIT-30 external memory.

PARAMETERS:

<Filename>: Name of the executable file to be loaded.

COMMENTS:

This option expects a valid DSP executable 'COFF' file prepared by the TMS linker. It also expects all the code sections to be above 400H (except Vectors). A part of the debugger resides in SPIRIT-30 exernal memory from C0H-3ffH. So the load option doesn't allow user program to corrupt the debugger code. **The debugger uses INT0, TIMER0, TRAP0 and TRAP1 vectors on SPIRIT-30, and user cannot make use of these vectors.**

MESSAGES:

Message "Error loading file" is displayed if there is an error reading the file or if the given file is not a valid 'COFF' file. If this error occurs, type the correct filename or hit ESC to skip this option.

Message "Invalid loading address: link your program with proper '.cmd' file" is displayed if the given file has sections with loading address between C0H and 3ffH. To solve this problem, come out of the debugger using Quit option, relink your program with correct '.cmd' file, and restart debug session.

[DOS] module:

NAME: [DOS] (execute DOS commands)

OPERATION:

This option allows you to execute DOS commands and resume debug session.

COMMENTS:

This option executes command.com, which lets you give DOS commands. To resume debug session, type "exit".

[Quit] module:

NAME: [Quit] (Quit debug session)

OPERATION:

This option allows you to quit the debug session.

COMMENTS:

This option confirms before quitting the session. Press 'ENTER' if you wish to quit. If you wish to continue the session press ESC.

APPENDIX

A-1 Data file header information

The standard header for all data files is the following:

```
-----------------------------------
DATASET  file.dat
VERSION  0
SIGNAL
DATE
TIME  Mon July 10 09:30:29 1989
DATA_TYPE  FLOAT
NUM_SAMPS 10
INTERVAL 0
MAX_VALUE 0
MIN_VALUE 0
VERT_UNITS 0
HORZ_UNITS 0
COMMENTS
DATA
0.000000E+000
5.887877E+002
9.232038E+002
6.890812E+003
1.988828E+002
4.181121E+005
-5.980890E+004
-9.828283E-002
-9.992834E+009
2.309208E-007
-----------------------------------
```

There are 14 lines of header information. Debugger, however uses only header lines 6 and 7. Their meanings are described below:

DATA_TYPE

This header can be one of the following:

[f] FLOAT	- ascii floating point
[h] HEX	- hex format
	- a blank header is interpreted as a FLOAT

NUM_SAMPS

The number of lines of data in this file after the last header, DATA.