CHAPTER 5

Using Other Commands



5.1 Display and Edit Commands



Commands:

```
U        -- Unassemble instructions or display source
R        -- Display or change registers
MAP   -- Display system memory map
D        -- Display memory in the most recently
            specified format
DB       -- Display memory in byte format
DW       -- Display memory in word format
DD       -- Display memory in double word format
E        -- Edit memory in the most recently
            specified format
EB       -- Edit memory bytes
EW       -- Edit memory words
ED       -- Edit memory double words
INT?  -- Display last interrupt number
? or H -- Display help information
VER   -- Display Soft-ICE version number
```



U

U -- Unassemble instructions or display source

Syntax :

U [address] [L[=]length]

length -- The number of instructions
to be unassembled

Comments:
The U command displays the instructions of the program being debugged.

If length is not specified, the length defaults to eight lines if available, or one less than the screen length.

If address is not specified, the command unassembles at address starting at the first byte after the last byte unassembled by a previous unassemble command. If the has been no previous unassemble command, the address defaults to the current CS:IP.

If the code window is visible, the instructions are displayed in the code window.

If source is loaded for the address range specified then source lines may be displayed depending on the current source mode.

Example:
U $-10
This command unassembles instructions beginning
10 hexadecimal bytes before the current address.

77

U .499

This command displays the current source file starting at line 499. The code window must be visible and in source mode.

78

R

R-- Display or change registers

Syntax:

R register-name [ [ = ]value] ]

register-name -- Any of the following:
AL, AH, AX, BL, BH,
BX, CL, CH, CX, DL,
DH, DX, DI, SI, BP,
SP, IP, CS, DS, ES, SS,
or FL

value -- If register-name is any name other than FL, value is a hex value or an expression. If register-name is FL, value is a series of one or more of the following flag symbols, each optionally preceded by a plus or minus sign:

- O (Overflow flag)
- D (Direction flag)
- I (Interrupt flag)
- S (Sign flag)
- Z (Zero flag)
- A (Auxiliary carry flag)
- P (Parity flag)
- C (Carry flag)

Comments:
The R command displays or changes register values.

If no parameters are supplied, all register and flag value are displayed, as well as the instruction at the current CS:IP address.

If register-name is supplied without a value, Soft-ICE displays the current value of the specified register and



prompts you for a new value. If register-name is FL, flags that are set are displayed as highlighted uppercase characters; flags that are cleared are displayed as non-highlighted lowercase characters. To retain the current value of a register, press ENTER.

If both register-name and value are supplied, the specified register's contents are changed to the value.

To change a flag value, use FL as the register-name, followed by the symbols of the flag whose values you want to toggle. To turn a flag on, precede the flag symbol with a plus sign. To turn a flag off, precede the flag symbol with a minus sign. The flags can be listed in any order.

Examples:

```
RAH 5
```

This command sets the AH register equal to 5.

```
R FL = OZP
```

This command toggles the O, Z, and P flag values.

R FL

This command displays the current flag values, and allows them to be changed.

RFL O + A-C

This command toggles the O flag value, turns on the flag value, and turns off the C flag value.



## MAP

MAP -- Display system memory map

Syntax :

MAP

Comments:
The MAP command displays the names, locations, and sizes of system memory components. The size is displayed in paragraphs. One paragraph is equivalent to 10 hexadecimal bytes.

The component that the CS:IP register currently points to is highlighted.

Use the MAP command when:
* A break point occurs and CS:IP is not in a known memory region.
* You want to get control within a resident program or system program. A range break point can be set based on the starting address and size reflected by MAP.
* You suspect a program or system component of writing over code outside of its memory space. MAP is used to obtain the memory address of the region to use with the CSIP command.
* You need to find out which resident program owns certain interrupt vectors.



Example:

MAP

The following is a sample display produced by the command:

```
Start Length
0000:0000 0040 Interrupt Vector Table
0040:0000 0030 ROM BIOS Variables
0070:0000 00FE I/O System
016E:0000 06B7 DOS
0842:0000 02CE DOS File Table & Buffers
A000:0000 5E00 System BUS
F000:0000 1000 ROM BIOS
```

Versions of DOS lower than 3.1 display program addresses instead of displaying the program names.





D, DB, DW, DD -- Display memory

Syntax:

D [size] [address] [L[ = ]length]
size -- B -- Byte
W -- Word
D -- Double Word

length -- The number of bytes to be displayed.

Comments:
The D command displays the memory contents of the specified address.

The contents are displayed in the format of the size specified. If no size is specified, the last size used will be displayed. The ASCII representation is also displayed for all forms.

If address is not specified, the command displays memory at the address starting at the first byte after the last byte displayed.

If length is not specified, it defaults to eight lines, or fewer if the window is smaller.

If the data window is visible, the data is displayed in the data window and the length is ignored.

Example:

```
DW DS:00 L=8
```

This command displays, in word format and in ASCII format, the value of the first eight bytes of the current data segment.





E, EB, EW, ED -- Edit memory

Syntax:

E [size ] address [data-list]
size -- B -- Byte
W -- Word
D -- Double Word

data-list -- list of data objects of the specified size (Bytes, Words or Double Words) or quoted strings separated by commas or spaces. The quoted string can begin with a single quote or a double quote.

Comments:
The E commands display the memory contents at the specified address, and allow you to edit the values.

These commands display the memory contents in ASCII format, and in the format of the size specified.

A memory editor is provided for quick memory updates. Memory can be edited by typing ASCII characters, or by typing byte, word, or double word values. If no size is specified, the last size used will be assumed. The memory Editing key strokes are:
@ -- Move cursor up
@ -- Move cursor down
@ -- Move cursor right
@ -- Move cursor left
SPACE -- Move cursor to next element



TAB -- Toggle between numeric and ASCII areas

ESC or

ENTER -- Exit memory editor

As values are input, the actual memory locations are updated. All numeric values are hex numbers. To toggle between the ASCII and numeric display areas, press the TAB key.

If the data window is visible, the data is edited in the data window, otherwise the data is edited in the command window.

The data display length defaults to 8 lines if in the command window, or to the size of the data window if it's visible.

If no parameters are supplied, the cursor moves into the data window if the data window if visible. If the data window is not visible, the data is edited in the command window at the last address displayed or edited.

Examples:
EB 1000:0
This command displays, in byte format, up to six lines containing both the numeric and the ASCII representation of the values of the data starting at location 1000:0000. Once the lines are displayed, you can edit the values.
EB 8000:0 "Hello",0D
This command replaces the values starting at locatio 8000:0000 with the string "Hello" followed by a carriage return.



INT?

INT? -- Display last interrupt number

Syntax:

INT?

Comments:
The INT? command displays the address and the number the last interrupt that happened.

Example:
INT?
An example of the display produced by the INT? command follows:
Last Interrupt: 16
At: 0070:0255

This example shows that the last interrupt generated in the system before the Soft-ICE window was brought up was an interrupt 16 hexadecimal, at location 0070:0255H. If the last interrupt that happened was a software interrupt, unassembling the code at 0070:0255H will show the interrupt instruction. If it was a hardware interrupt, unassembling the code will show the instruction that was executing when the hardware interrupt occurred.



## ? or H

? or H -- Display help information

Syntax:

< ? | H > [command | expression]

Comments:
The ? command and the H command both display help information.

If no parameters are specified, help displays short descriptions of all the commands and operators, one screen at a time. Press any key to continue, or press ESC to quit displaying help.

If command is specified, help displays more detailed information on the specified command, including the command syntax and an example.

If expression is specified, the expression is evaluated and the result is displayed in hexadecimal, decimal, and ASCII.

Examples:
? ALTKEY
This command displays information about the ALTKEY command, including its syntax and an example.
H 10 + 14*2
This command displays: 0038 00056 "8". These are the hexadecimal, decimal and ASCII representations of value of the expression "10 + 14*2".



## VER

VER -- Display Soft-ICE version number

Syntax:

VER

Example:
VER
This command displays the Soft-ICE version
and the Nu-Mega Technologies copyright message.



5.2 I/O Port Commands

Commands:
I or IB -- Input from byte I/O port
IW -- Input from word I/O port
O or OB -- Output to byte I/O port
OW -- Output to word I/O port



I, IB, IW

I, IB, IW -- Input from I/O port

Syntax:

I [size] port

Size -- B -- Byte
W -- Word

port -- A byte or word value

Comments:
The input from port commands are used to read and display a value from a hardware port. Input can be done From byte or word ports. If no size is specified, the default is byte.

Example:
I 21
This command displays the mask register for interrupt
controller one.

## O, OB, OW

O, OB, OW, -- Output to I/O port

Syntax:

O [size] port value
size -- B -- Byte
W -- Word

port -- A byte or word value
value -- A byte for a byte port or a word
for a word port

Comments:
The output to port commands are used to write a value to a hardware port. Output can be done to byte or word ports If no size is specified, the default is byte.

Example:
O 21 FF
This command masks off all the interrupts for interrupt controller one.



5.3 Transfer Control Commands

Commands:
X -- Exit from Soft-ICE window
G -- Go to address
T -- Trace one instruction
P -- Program step
HERE -- Go to current cursor line
GENINT -- Force an interrupt
EXIT -- Force exit of current DOS program
BOOT -- System boot (retain Soft-ICE)
HBOOT -- Hard system boot (total reset)



## X

X -- Exit from Soft-ICE window

Syntax:

X

Comments:
The X command exits the Soft-ICE window and restores control to the program that was interrupted to bring up Soft-ICE. The Soft-ICE window disappears. If any break points have been set, they become active.

Example:
X

93

G

G -- Go to address

Syntax:

G [=start-address] [break-address]

Comments:
The G command exits from the Soft-ICE window with a single one-time execution break point set. In addition, all sticky break points are armed.

Execution begins at the current CS:IP unless the start-address parameter is supplied. In that case execution begins at start-address. Execution continues until break-address is encountered, the window pop-up key sequence is used, or a sticky break point occurs.

The break-address must be the first byte of an instruction opcode.

When the specified break-address is reached, the current CS:IP will be the instruction where the break point was set.

The G command with no parameters behaves the same as the X command.

The non-sticky execution break point uses an 80386 break point register, unless all break point registers have been allocated to sticky break points. In that case, an INT 3 style break point is implemented. When this case occurs, the G and P commands will not work correctly in ROM. An error message will be displayed if this is attempted.

Example:
G CS:1234
This command sets a one time break point at CS:1234

# T

T -- Trace one instruction

Syntax:

T [=start-address] [count]

Comments:
The T command single steps one instruction by utilizing the single step flag.

Execution begins at the current CS:IP unless the start-address parameter is specified. If start-address is specified, CS:IP is changed to start-address prior to single stepping.

If count is specified then Soft-ICE single steps count time The TRACE command will continue until the count is exhausted or the Esc key is pressed, regardless of which break points are reached.

In source mode, the T command steps to the next source statement. If the current statement is a procedure or function call, and source exists for the routine being called, T steps into the call. If there is no source available for the called procedure or function, T steps over the routine.

Example :
T = 1284 3
This command single steps through three instruction
starting at memory location 1284.



# P

P -- Program step

Syntax:

P

Comments:
The P command is a logical program step. One instruction at the current CS:IP is executed unless the instruction is a call, interrupt, loop, or repeated string instruction. In those cases, the entire routine or iteration is completed before control is returned to Soft-ICE.

The P command uses a one-time execution break point. The non-sticky execution break point uses an 80386 break point register, unless all break point registers have been allocated to sticky break

points. In that case, an INT3 style break point is implemented. When this case occurs, the P and G commands will not work correctly in ROM. An error message will be displayed if this is attempted.

In source mode, the P command steps to the next source statement. If the current statement is a procedure or function call, the P command steps over the it.

Example:

```
P
```

This command executes one 'program step'.





HERE -- Go to current cursor line

Syntax:

```
HERE
```

Comments:
The HERE command executes until the program reaches the current cursor line. HERE is only available when the cursor is in the code window. If the code window is not visible or the cursor is not in the code window, use the G command instead.

The HERE command exits from Soft-ICE with a single one-time execution break point set. In addition, all sticky break points are armed.

Execution begins at the current CS:IP and continues until address of the current cursor position in the code window encountered, the window pop-up key sequence is used, a sticky break point occurs.

The non-sticky execution break point uses an 80386 break point register, unless all break point registers have been allocated to sticky break points. In that case, an INT 3 style break point is implemented. When this case occurs, the HERE command will not work correctly in ROM. An error message will be displayed if this is attempted.

Example:

```
HERE
  This example sets an execution break point at the
  current cursor position, then exits from Soft-ICE and
  begins execution at the current CS:IP.
```

Default Function Key: F7

GENINT

GENINT -- Force an interrupt

Syntax:

GENINT INT1 | INT3 | NMI | interrupt-number

interrupt-number -- a number in the range 00 - FF

Comments:
The GENINT command forces an interrupt to occur. This function can be used to hand off control to another debugger when using Soft-ICE with another software debugger. It can also be used to test interrupt routines.

The GENINT command simulates the processing sequence of a hardware interrupt or an INT instruction. It pushes the flags, the CS register, and the IP register, then changes the value of the CS and IP registers to the value of the interrupt vector table entry corresponding with the specified interrupt number.

Example:
GENINT NMI
This forces a non-maskable interrupt. This will give control back to CodeView if Soft-ICE is being used as an assistant to CodeView.



EXIT

EXIT -- Force exit of current DOS program

Syntax:

EXIT [R] [D]
R -- Restore the interrupt vector table
D -- Delete all break points

Comments:
The EXIT command attempts to abort the current program by forcing a DOS exit function (INT 21H, function 4CH) This command will only work if the DOS is in a state where it is able to accept the exit function call. If this call is made from certain interrupt routines, or other times when the DOS is not ready, the system may behave unpredictably.

This function does NOT do any system resetting other than the interrupt table when the R option is used. This means that BIOS variables, video modes and other systems level data are not restored.

Using the R option will cause the interrupt vectors to be restored to whatever they were the last time they were saved. Soft-ICE saves the interrupt vectors when it is loaded, when a program is loaded with LDR.EXE, and when the VECS S command is used.

Note:
To re-start a program that has been loaded with the Soft-ICE program loader (LDR.EXE) do the following:

```
EXIT R
LDR prog.EXE
```

The EXIT command will restore the interrupt table to the values it contained before the program was loaded, then



exit to the command processor. By running the LDR utility and specifying the .EXE suffix, the program is loaded back in without re-loading symbols and source. The symbols and source will remain in memory.

Caution:
The EXIT command should be used with care. Since Soft-ICE can be popped up at any time, a situation can occur where the DOS is not in a state to accept an exit function call. Also, the EXIT command does not do any program specific resetting. For instance, the EXIT command does not reset the video mode. If your program has placed the video BIOS and hardware in a particular video mode, it will stay in that mode after the EXIT command.

Example:

```
EXIT R
  Restores the interrupt table and exits the current
  program. The R option should be used if exiting from
  a program loaded with the Soft-ICE program loader
  LDR.EXE.
```



## BOOT

BOOT -- System boot (retain Soft-ICE)

Syntax:

```
BOOT
```

Comments:
The BOOT command resets the system and retains Soft-ICE. BOOT is required to debug boot

sequences, DOS loadable drivers, and non-DOS operating systems.

BOOT is implemented with an Interrupt 19H ROM BIOS call. In some instances memory may be corrupted to the point where Interrupt 19 will not work. If this occurs, bring up Soft-ICE and use the HBOOT command.

For BOOT to work properly, Soft-ICE should be installed as a loadable driver in CONFIG.SYS before any other device drivers. This is so Soft-ICE can restore the original system state as accurately as possible.

Example:
 BOOT
  This command makes the system reboot. Soft-ICE
  remains resident.



HBOOT

HBOOT -- Hard system boot (total reset)

Syntax:

 HBOOT

Comments:
The HBOOT command resets the entire system. Soft-ICE is not retained in the reset process. HBOOT is sufficient unless an adapter card requires a power-on reset. In those rare cases, the machine power must be recycled.

Example :
 HBOOT
  This command makes the system reboot. Soft-ICE
  must be reloaded.



5.4 Debug Mode Commands

Commands:
  ACTION  -- Set action after break point is reached
  WARN  -- Set DOS/ROM BIOS re-entrancy
    warning mode
  BREAK  -- Break out any time
  I3HERE  -- Direct Interrupt 3's to Soft-ICE

## ACTION

ACTION -- Set action after break point is reached

Syntax:

ACTION [INT1 | INT3 | NMI | HERE | int-number]

int-number -- Any valid interrupt number (0-FFH).
Use this option only if a user-supplied break point qualification routine has taken over that interrupt vector (see section 11.2).

Comments:
The ACTION command determines where control is given when break point conditions have been met. In most cases, the desired action is INT3 or HERE, INT3 is typically used if Soft-ICE is being used with a host debugger, HERE is used when it is desired to return to Soft-ICE when break point conditions have been met, INT1 and NMI are alternatives for certain debuggers that will not work with the INT3 option. For instance, CODEVIEW works best with ACTION set to NMI.

Use int-number if there is a user-supplied break point qualification routine installed. Using int-number without having a user-supplied break point qualification routine installed causes an error. For more information, see section 11.2,'User-Qualified Break Points'.

If no parameter is supplied with the ACTION command, the current action is displayed.

The default action is HERE.



Example:
ACTION HERE
This command specifies that control will return to Soft-ICE when break point conditions have been met.



## WARN

WARN -- Set DOS/ROM BIOS re-entrancy warning mode

Syntax:

WARN [ON | OFF]

Comments:
The WARN command is provided for using Soft-ICE with debuggers that use DOS and ROM BIOS. Many debuggers use DOS and ROM BIOS for screen output and for receiving keystrokes. Since DOS and ROM BIOS are not fully re-entrant, these debuggers may not work properly if break point occurs while the DOS or ROM BIOS is executing.

If WARN ON is set, and ACTION is not HERE, then control will come to Soft-ICE before the actual action occurs. The system displays the current CS:IP and gives you the choice of continuing or returning to Soft-ICE. Generally, you should choose to return to Soft-ICE to continue your debugging. Only continue with the host debugger if you know your debugger will not cause DOS or ROM BIOS to be re-entered.

WARN mode should be turned on to use Soft-ICE with DEBUG, SYMDEB, and CODEVIEW.

If no parameter is specified, the current state of WARN is displayed.

The default is WARN mode OFF.

Example:
WARN ON
This command turns on DOS/ROM BIOS re-entrancy
warning mode.





BREAK -- Break out any time

Syntax:

BREAK [ON | OFF]

Comments:
The BREAK command allows popping up the Soft-ICE window when the system is hung with interrupts disabled. Break mode can be used for the entire debugging session, or it can be turned on and off when it is required.

Break mode degrades system performance slightly. This performance degradation must be weighed against the necessity of breaking out of a hung program. A user may want to have break mode on all

the time, even though performance is degraded, because the program could hang at any time.

Unlike other debuggers that can also be brought up at any time, Soft-ICE does not require an external switch. When BREAK is on, the Soft-ICE window can be brought up at any time by pressing the current key sequence.

If no parameter is specified, the current state of BREAK is displayed.

The default is BREAK mode OFF.

Example:
BREAK ON
  This command turns on break mode. This means that
  the Soft-ICE window can be brought up at any time,
  even if interrupts are disabled.



13 HERE

13HERE -- Direct Interrupt 3's to Soft-ICE

Syntax:

13HERE [ON | OFF]

Comments:
The 13HERE command lets you specify that any Interrupt 3 will bring up the Soft-ICE window. This feature is useful for stopping your program in a specific location.

To use this feature, place an INT 3 into your code at the location where you want to stop. When the INT 3 occurs, it will bring up the Soft-ICE window. At this point, you can use the R IP command to change your instruction pointer to the instruction after the INT 3, then you can continue debugging.

If no parameter is specified, the current state of 13HERE is displayed.

The default is 13HERE mode OFF.

Example:
13HERE ON
  This command turns on 13HERE mode. Any INT 3's
  generated after this point will bring up the Soft-ICE
  window.

5.5 Utility Commands

Commands:
- A -- Assemble code
- S -- Search for data
- F -- Fill memory with data
- M -- Move data
- C -- Compare two data blocks



# A

A -- Assemble code

Syntax:

A [address]

Comments:
The Soft-ICE assembler allows you to assemble instructions directly into memory. The assembler supports the basic 8086 instruction set with the 80186 and 80286 real address mode extensions. Numeric co-processor instructions and 80386 specific instructions, registers and addressing modes can NOT be assembled.

The A command enters the Soft-ICE interactive assembler. An address is displayed as a prompt for each assembly line After an assembly language instruction is typed in and ENTER is pressed, the instructions are assembled into memory at the specified address. Instructions must be entered with standard Intel format. Press ENTER at an address prompt to exit assembler mode.

If the address range in which you are assembling instructions is visible in the code window, the instructions will change interactively as you assemble.

The Soft-ICE assembler supports the standard 8086 family mnemonics, however there are some special additions :
- * The DB mnemonic is used to define bytes of data directly into memory. The DB command is followed by a list of bytes and/or quoted strings separated by spaces or commas.
- * The RETF mnemonic represents a far return.
- * WORD PTR and BYTE PTR are used to determine data size if there is no register



argument, for example: MOV BYTE PTR

ES:[ 1234],1.
* Use FAR and NEAR to explicitly assemble far and near jumps and calls. If FAR or NEAR is not specified then all jumps and calls are near.
* Operands referring to memory locations should placed in square brackets, for example: MOV AX,[1234].

Example:
A CS:1234
This command prompts you for assembly instruction then assembles them beginning at offset 1234H with the current code segment. Press ENTER at the address prompt after entering the last instruction.



## S

S -- Search for data

Syntax:

S address L length data-list
data-list -- list of bytes or quoted strings separated by commas or spaces. .A quoted string can begin with a single quote or a double quote.

length -- length in bytes

Comments:
The S command searches memory for a series of bytes or characters that matches the data-list. The search begins at the specified address and continues for the length specified. The address of each occurrence found in the range is displayed.

Example:
S DS:SI+10 L CX 'Hello',12,34
This command searches for the string 'Hello' followed by the bytes 12H and 34H starting at offset SI+10 in the current data segment and ending CX bytes later.



## F

F -- Fill memory with data

Syntax:

F address L length data-list
data-list -- list of bytes or quoted strings separated
by commas or spaces. A quoted string
can begin with a single quote or a
double quote.

length -- length in bytes

Comments:
The F command fills memory with the series of bytes or characters specified in the data-list. Memory is filled starting at the specified address and continuing for the specified length, repeating the data-list if necessary.

Example:
F 8000:0 L 100 'Test'
This command fills memory starting at 8000:0 for a
length of 100H bytes with the string 'Test'. The string
'Test' is repeated until the fill length is exhausted.



M

M -- Move data

Syntax:

M start-address L length end-address
length -- length in bytes

Comments:
The M command moves the specified number of bytes from the start-address in memory to the end-address in memory.

Example:
M 1000:0 L 200 2000:0
This command moves 200H bytes from memory
location 1000:0 to memory location 2000:0.

C

C -- Compare two data blocks

Syntax:

C address1 L length address2
length -- length in bytes

Comments:
The C command compares the memory block specified by address1 and the length with the memory block specified address2 and the length.
When a byte from the first data block does not match a byte from the second data block, both bytes are displayed, along with their addresses.

Example:
C 5000:100 L 10 6000:100
This command compares the 10H bytes starting at memory location 5000:100 with the 10H bytes starting at memory location 6000:100.



5.6 Specialized Debugging Commands

Commands:
SHOW -- Display instructions from history buffer
TRACE -- Enter trace simulation mode
XT -- Single step in trace simulation mode
XP -- Program step in trace simulation mode
XG -- Go to address in trace simulation mode
XRSET -- Reset back trace buffer
VECS -- Save/restore/compare interrupt vectors
SNAP -- Take snap shot of memory block
EMMMAP -- Display EMM allocation map



SHOW

SHOW -- Display instructions from history buffer

Syntax:

SHOW [B | start]
B -- This tells the show command to start

the display with the oldest instruction
in the back trace buffer.

start -- The number of instructions back from
the buffer end (last instruction
captured) to begin display.

Comments:
The SHOW command displays instructions from the back trace history buffer. If source is available for the instructions then the display is in mixed mode, otherwise only code is displayed.

SHOW allows scrolling through the back trace buffer with the up, down, Pageup and PaqeDn keys. To exit from SHOW you must press the Esc key.

Preceding the address of each instruction is the buffer entry number. This number shows how deep into the buffer you are displaying. The higher the number, the deeper you are into the buffer.

Note:
Before using the SHOW command, instructions must have been logged with a back trace range. See chapter 9 for more information on back trace ranges.



Hints:
It is often useful to have the code window visible with the actual code of the region you are displaying from the back trace buffer. When you compare the actual instruction flow to code, displayed jumps and calls are usually less confusing.

Using SHOW in conjunction with the TRACE command will allow you to see the instructions in the back trace history buffer from two different points of view.

Example:
SHOW 40
This example will displays starting with the 40th
instruction back in the back trace buffer.



## TRACE

TRACE -- Enter trace simulation mode

Syntax:

TRACE [start] | [OFF]
start -- The number of instructions back from

the buffer end (last instruction
captured) to begin trace simulation

OFF -- Exit trace simulation mode.

Comments:
The TRACE command allows you to replay instructions from the instruction back trace history buffer just as if they were being executed for the first time. To use trace simulation mode you must have the code window visible After entering trace simulation mode you use the XT, XP and XG commands to trace through the instructions in the buffer.

To exit trace simulation mode type TRACE OFF.

TRACE with no parameters specified displays whether trace simulation mode is on or off.

Note:
Before using the TRACE command, instructions must have been logged with a back trace range. See chapter 9 for more information on back trace ranges.

Hints:
Trace simulation mode is most useful when the code window is visible. It is often useful to use TRACE in conjunction with the SHOW command. This allows the



instructions in the back trace history buffer to be viewed simultaneously in two different forms.

Example:
TRACE 40
This example enters trace simulation mode starting 40 instructions back from the last instruction logged. It will remain in trace simulation mode until TRACE OFF is entered.



## XT

XT -- Single step in trace simulation mode

Syntax:

XT [R]
R -- Single step in reverse direction.

Comments:

The XT command single steps through the instruction back trace history buffer. This command acts like the T command for normal debugging. Note that the registers do NOT change while stepping in trace simulation mode except CS and IP,

The XT instruction allows you to replay instructions from the back trace history buffer,

Note:
Before using XT you must be in trace simulation mode. See chapter 9 and the TRACE command in this section for more information on back trace ranges.

Hint:
If you are using XT frequently, like any other Soft-ICE command it can be assigned to a function key.

Example:
XT
This command single steps one instruction in trace
simulation mode.

121

XP

XP -- Program step in trace simulation mode

Syntax:

XP

Comments:
The XP command does a logical program step through the instruction back trace history buffer. This command acts like the P command for normal debugging. Note that the registers do NOT change while stepping in trace simulation mode except CS and IP.

The XP instruction allows you to replay instructions from the back trace history buffer.

Note:
Before using XP you must be in trace simulation mode. See chapter 9 and the TRACE command in this section for more information on back trace ranges.

Hint:
If you are using XP frequently, like any other Soft-ICE command it can be assigned to a function key.

Example:
XP
This command executes one program step in trace
simulation mode.

XG -- Go to an address in trace simulation mode

Syntax:

X [R] address
R -- Search for address in reverse direction.

address -- Address to go to in the back trace
history buffer.

Comments:
The XG command moves the instruction pointer to the next occurrence of the specified address in the back trace history buffer. If R is specified preceding the address, then the instruction pointer is moved to the previous occurrence the specified address in the back trace buffer.

The address must be the first byte of an instruction opcode.

The XG is analogous to the G command in normal debugging.

Note:
Before using XG you must be in trace simulation mode. See chapter 9 and the TRACE command in this section for more information on back trace ranges.

Example:
XG 273:1030
This command moves the instruction pointer to the
next instance of the instruction at address 273:1030.



XRSET -- Reset back trace history buffer

Syntax:

XRSET

Comments:
The XRSET command resets the back trace history buffer. This command should be executed before setting a back trace range if there is unwanted instruction information in the back trace buffer.

Example:
XRSET
This command resets the back trace buffer.



## VECS

VECS -- Save/restore/compare interrupt vectors

Syntax:

VECS [C|S|R]
C -- Compare current table with stored table
S -- Save current interrupt table to buffer
R -- Restore interrupt table from buffer

Comments:
The VECS command allows you to save and restore the interrupt table to an internal Soft-ICE buffer. The actual table can also be compared to the stored table with the differences displayed.

When the C option is used to compare the current interrupt vector table with the stored copy the output is in the following format:

address old-vector new-vector

Each vector that has changed is displayed.

The interrupt vector table is initially stored when Soft-ICE is loaded. It is also automatically stored when a program loaded with LDR.EXE. Only one copy of the interrupt vector table is stored, so each time VECS S is executed, previous copy of the interrupt table is overwritten.

If no parameters are specified, the entire interrupt vector table is displayed.



Example:
VECS C
This command compares the actual interrupt vector table with one that had been previously stored in the Soft-ICE internal VECS buffer.

# SNAP

SNAP -- Take snap shot of memory block

Syntax:

SNAP [C | S | R] address1 address2
C -- Compare buffer with address range
S -- Save address range to buffer
R -- Restore buffer to address range

Comments:
The SNAP command takes a snap shot of a memory block for later comparison. The S option copies a block of memory to a buffer in extended memory. The C option displays differences between the buffer in extended memory and the actual memory specified by the address range. The R option copies the buffer in extended memory to the address range in conventional memory.

When the C option is used to compare the buffer with the address range the output is in the following format:

address old-data new-data

Each byte that has changed is displayed.

The address is usually not necessary for the C and R options. If the address is not specified, the address from the last time SNAP was entered with a specified address used.

Notes:
To use the SNAP command you must have specified the /TRA XXXX switch on the S-ICE.EXE line in CONFIG.SYS.



The SNAP command saves data in the back trace history buffer. If you are using back trace then you will have a conflict with SNAP. Specifically, SNAP will overwrite back trace information if you do a SNAP S when instruction history is in the back trace buffer. Conversely, if you have saved a region with SNAP, then enabling a back trace range will overwrite the SNAP buffer.

Example:
SNAP S 2000:0 4000:0
This command stores the data block from 2000:0 to 4000:0 in the Soft-ICE back trace buffer.

EMMMAP

EMMMAP -- Display EMM allocation map

Syntax:

EMMMAP

Comments:
The EMMMAP command displays each physical page that is available for EMM memory and the pages that are currently mapped in.

Note:
The Soft-ICE EMM feature must be enabled to use this function. See chapter 8 for more information on enabling EMM capability.

Example:
EMMMAP
This example displays the current EMM allocation in in the following form.
Phy page Seg address Handle/Page
00 D000 FFFF
01 D400 0001/0000
02 D800 0001/0001
03 DC00 0001/0002
In this example, physical page 0 is located at D000 and is unmapped. Physical page 1 is located at D400 and has handle 1, page 0 mapped into it. Physical page 2 is located at D800 and has handle 1, page I mapped into it. Physical page 3 is located at DC00 and has handle page 2 mapped into it.



5.7 Windowing Commands

Commands:
WR -- Toggle register window
WC -- Toggle/set size of code window
WD -- Toggle/set size of data window
EC -- Enter/exit code window
. -- Locate current instruction

Three window types may be created with Soft-ICE:

register, data, and code. Any of these windows can be toggled on or off at any time. The data and code windows can be of variable size; the register window is fixed in size. The windows always remain in a fixed order. Starting from the top of the screen, the order is register window, data window, then code window.

130

WR

WR -- Toggle register window

Syntax:

WR

Comments:
The command makes the register window visible if not currently visible. If the register window is currently visible, WR removes the register window.

The register window displays the 8086 register set and the processor flags.

Default Function: F2

131

WC

WC -- Toggle/set size of code window

Syntax:

WC [window-size]
window-size -- a decimal number between one and 21.

Comments:
If window-size is not specified, this command toggles the code window. If it was not visible it is made visible, and if it was visible it is removed.

If window-size is specified the code window is resized, or it was not visible it is made visible with the specified size.

Note:
If you wish to move the cursor to the code window use the EC command. See description of the EC command for more details.

Example:
WC 12
If no code window is present, then a code window 12 lines in length is created. If the code window is currently on the screen, it is resized to 12 lines.

132

## WD

WD -- Toggle/set size of data window

Syntax:

WD [window-size]
window-size -- a decimal number between one and 21.

Comments:
If window-size is not specified, this command toggles the data window. If it was not visible it is made visible, and if it was visible it is removed.

If window-size is specified the data window is resized, or it was not visible it is made visible with the specified size.

Example:
WD 1
If no data window is present then a data window of one line is created. If the data window is currently on the screen, it is resized to one line.

133

## EC

EC -- Enter/exit code window

Syntax:

EC

Comments:
The EC command toggles the cursor location between the code window and the command window. If the cursor was in the command window it is moved to the code window, and if the cursor was in the code window it is moved to the command window.

When the cursor is in the code window several options become available that make debugging much easier. The options are:

* Point-and-shoot break points
  Point-and-shoot break points are set with the BP command. If no parameters are specified with the BPX command an execution break point is set at the location of the cursor position in the code window. The cursor must be on a line that contains code (place the code window in mixed mode if you are unsure). The default function key assignment for BPX is F9.

* Go to cursor line
  You can set a temporary break point at the cursor and go with the HERE command. The cursor must be on a line that contains code (place the code window in mixed mode if you are unsure). The default function key assignment for HERE is F7.



* Scrolling the code window
  The code window can be scrolled only while the cursor is in the code window. The scrolling keys (UP arrow, DOWN arrow, PageUp and PageDown) are redefined while the cursor is in code window. When the cursor is in the code window the scrolling keys do the following:
  up -- Scroll code window up one line
  down -- Scroll code window down one
  pageup -- Scroll code window up one window
  pageDn -- Scroll code window down one window

Note:
The code window must be visible for the EC command to work.
Default Function Key: F6



## .

. -- Locate current instruction

Syntax:
.

Comments:
When the code window is visible, the . command makes the current source line or current instruction visible.



5.8 Debugger Customization Commands

Commands:

| | |
|---|---|
| PAUSE | -- Pause after each screen |
| ALTKEY | -- Set alternate key sequence to invoke Soft-ICE |
| FKEY | -- Show and edit function keys |
| BASE | -- Set/display current radix |
| CTRL-P | -- Toggle log session to printer |
| Print-Screen | -- Print contents of screen |
| PRN | -- Set printer output port |



## PAUSE

PAUSE -- Pause after each screen

Syntax:

PAUSE [ON | OFF]

Comments:
PAUSE controls screen pause at the end of each page. If PAUSE is ON, you are prompted to press any key before information is scrolled off the window. The prompt is displayed in the status line at the bottom of the window.

If no parameter is specified, the current state of PAUSE is displayed.

The default is PAUSE mode ON.

Example:

PAUSE ON
This command specifies that subsequent window display commands will cause the screen to wait for you to press a key before scrolling new information off the window.

ALTKEY

ALTKEY -- Set alternate key sequence to invoke Soft-ICE

Syntax:

ALTKEY [ALTletter] | [CTRLletter] | [SYSREQ]
letter - Any letter (A - Z)

Comments:
The ALTKEY command allows the key sequence for popping up Soft-ICE to be changed. The key sequence be changed to CTRL + letter, ALT + letter, or the SysRq key.

Occasionally you may be using a program that conflicts with the CTRL D key sequence that brings up the Soft-ICE window. One way to circumvent this possible problem is to use the ALTKEY command to change the key sequence. Another way is to add the SHIFT key to the current sequence. Soft-ICE does not respond to this key sequence and allows it to go through to your program. For example if a resident program you are using is brought up with the CTRL D key sequence, try using the key sequence CTRL SHIFT D to bring up your resident program. On some keyboards, you must press ALT and the prtsc key simultaneously to generate a system request. Care must be taken so the screen is not printed accidentally.

If no parameter is specified, the current key sequence state is displayed.

The default key sequence is CTRL D.



Example:
ALTKEY ALT Z
This command specifies that the key sequence ALT Z will now be used to pop up the Soft-ICE window.



FKEY

FKEY -- Show and edit function keys

Syntax:

FKEY [function-key-name string]
function-key-name -- F1, F2... F12

string -- The string consists of any valid Soft-ICE commands and the special character ^ (caret) and ; (semicolon). A ^ is placed in the string to make a command invisible. A ; is placed in the string to denote a carriage return.

Comments:
The FKEY command is used from the command line to assign a function key to a command string. Function key can be assigned to any command string that can be typed into Soft-ICE.

If no parameters are specified, then the current function key assignments are displayed.

To unassign a specified function key, use the FKEY command with these parameters: a function-key-name followed by a null string.

The function keys can also be pre-initialized in the definition file S-ICE.DAT. For more information on function key definitions in the definition file, refer to section 6.4.

Using carriage return symbols in a function key assignment string allows you to assign a function key a series of commands. A carriage return is represented by a ; (semicolon).



If you put ^ (shift 6) in front of a function key definition, the subsequent command will be invisible. The command will function as normal, but all information displayed in the command window (including error messages) is suppressed. The invisible mode is useful when a command changes information in a window (code, register or data) but you do not want to clutter the command window,

When a function key is made invisible with ^, the function key can be used in the middle of typing in other command without affecting their operation. For example, if you are using the default assignment for F2, you can toggle the register window with F2 even if you are partially through typing in your next command.

Note :
Soft-ICE now has a definition file named S-ICE.DAT. You can place function key assignments in this file so that function keys will be automatically assigned when Soft-ICE is loaded. The syntax for assigning a function key in the configuration file is:

```
function-key-name = "string"
```

When assigning function keys to a command string in S-ICE.DAT, the string must be enclosed in double quotes.

Command line examples:

```
FKEY F2 ^WR;
```

This example will assign the toggle register window command to the F2 key. The ^ makes the function invisible, and the ; ends the function with a carriage return. The F2 key will toggle the register window on or off, and can even be evoked while typing in another command.



FKEY F1 "G CS:120; R; G CS:"
This example shows that multiple commands can be assigned to a single function key and that partial commands can be assigned for the user to complete. After this command is entered, pressing the F1 key will cause the program to execute until location CS:120 is reached, display the registers, then start the G command for the user to complete.

FKEY F1 WD 3;D DS:100;
This example will assign a series of commands to the F1 key. The function is visible, and ends with a carriage return. The F1 key will make the data window three lines long and dump data starting at DS:100.

S-ICE.DAT example:
F1 = "WR;WD 2;WC 10;"
If this line is placed in S-ICE.DAT, when Soft-ICE is loaded it will assign the string to the F1 key. When F1 is pressed while in Soft-ICE, it will toggle the register window, create a data window of length 2 and a code window of length 10. For more information about assigning function key definitions in S-ICE.DAT, refer to chapter 6.



## BASE

BASE -- Set/display current radix

Syntax:

BASE [10 | 16]

Comments:
The BASE command sets the current radix to base 10 or base 16. Base 10 is of limited use in the

narrow window because of window width limitations. It also limits the amount of information displayed in some commands in the wide mode.

When the current radix is base 10, all numbers and addresses typed into and displayed by Soft-ICE are in decimal, When the current radix is base 16, all numbers and addresses typed into Soft-ICE are in hexadecimal except:

* source line numbers
* screen coordinates and sizes in the WIN command

These exceptions are always typed in and displayed as decimal numbers.

The default radix is base 16.

Example:
BASE 16
This example sets the current radix to base 16.

144

CTRL-P

CTRL-P --- Toggle log session to printer

Syntax:

CTRL-P

Comments:
When the CTRL key followed by the P key is pressed, all subsequent information displayed in the command window is also sent to the printer. To turn the log to printer mode off, type CTRL followed by P again.

When you are sending a lot of information to the printer using CTRL-P, you may want to turn the PAUSE command OFF to allow information to scroll off the window without pressing a key.

145

Print-Screen

Print-Screen - Print contents of screen

Syntax:

Print-Screen

Comments:

Depressing the print-screen key does a screen dump to printer. All information from the screen is sent the printer.

If you wish to print the memory map or help information is usually much faster to use CTRL-P than Print-Screen. This is because Print-Screen prints every character on the screen including borders.



## PRN

PRN --- Set printer output port

Syntax:

PRN [LPTx | COMx]
x -- a decimal number between 1 and 4.

Comments:
The PRN command allows you to send output from the CTRL-P and Print-Screen commands to a different printer port.

If no parameters are supplied, PRN displays the currently assigned printer port.

Example:
PRN COM 1
This command causes the CTRL-P and Print-Screen command output to go to the COM 1 port.



5.9 Screen Control Commands

Commands:
FLASH -- Restore screen during P and T
FLICK -- Screen flicker reduction
WATCHV -- Set watch video mode
RS -- Restore program screen
CLS -- Clear window
ALTSCR -- Change to alternate screen
WIN -- Change size of Soft-ICE window



## FLASH

FLASH -- Restore screen during P and T

Syntax:

FLASH [ON | OFF]

Comments:
The FLASH command lets you specify whether the screen will be restored during any Trace and Program step commands. If you specify that the screen is to be restored it is restored for the brief time period that the P or T command is executing. This feature is needed to debug sections of code that access video memory.

If the P command executes across a call or an interrupt, the screen will always be restored, because the routine being called may write to the screen.

If no parameter is specified, the current state of FLASH is displayed.

The default is FLASH mode OFF.

Example:
FLASH ON
This command turns on FLASH mode. The screen will
be restored during any subsequent P or T commands.



## FLICK

FLICK -- Screen flicker reduction

Syntax:

FLICK [ON | OFF]

Comments:
Certain types of video cards require waiting for horizontal or vertical retrace before outputting characters. If the video writes are made arbitrarily, flickering will appear while displaying characters. If flickering occurs on your screen while using the Soft-ICE window, you should turn FLICK on.

With some EGA cards, colors will not be restored properly when you exit from Soft-ICE. This is a problem with virtualizing EGA video. The port 3DA is a video port used for two purposes. The first is old CGA software polling 3DA for hsync and vsync. This allows them to have flicker free output on some old CGA controller cards. The second is that it is used to reset a palette latch on EGA cards. Soft-ICE has an algorithm to avoid having to constantly watch this port, which would slow down old programs that think they are on a CGA. However, there can occasional be circumstances where this

algorithm does not work. If you are using Soft-ICE on an EGA screen and you notice that the colors are not restored correctly, then turn FLICK ON and Soft-ICE will watch the 3DA port, fixing the problem.

When FLICK mode is ON, screen update will be slower.

If no parameter is specified, the current state of FLICK is displayed.

The default is FLICK mode OFF.



Example:

```
FLICK ON
This command turns on FLICK mode. This causes
Soft-ICE to wait for the horizontal or vertical retrace
before outputting characters.
```



## WATCHV

WATCHV -- Set watch video mode

Syntax:

```
WATCHV [ON | OFF]
```

Comments:
The WATCHV command allows you to specify how Soft-ICE should watch the video ports. Normally, Soft-ICE only watches video ports after an INT 10 instruction has been executed that switches to a non-character video mode. Some programs do not use INT 10 to switch modes. In these cases, if WATCHV is OFF, Soft-ICE may have trouble saving and restoring the screen properly. Turning WATCHV ON will cause Soft-ICE to watch the video ports all the time.

Turn WATCHV ON if you notice that Soft-ICE is not handling your screen properly, or if the cursor is not being restored properly. Turning WATCHV ON may have a performance impact in certain video modes.

If no parameter is specified, the current state of WATCHV is displayed.

The default is WATCHV mode OFF.

Example:

WATCHV ON
This command turns on WATCHV mode. This causes Soft-ICE to watch additional video ports for the purpose of virtualization.



RS

RS -- Restore program screen

Syntax:

RS

Comments:
The RS command allows you to restore the program screen temporarily. The Soft-ICE window disappears until any key is pressed.

This feature is useful when debugging graphic programs that update the screen frequently. When Soft-ICE is brought up, it returns to text mode. Using the RS command temporarily restores the graphics screen.

Example:
RS



CLS

CLS -- Clear window

Syntax:

CLS

Comments:
The CLS command clears the Soft-ICE window and moves the prompt and the cursor to the upper left-hand corner the window.

Example:
CLS



ALTSCR

ALTSCR -- Change to alternate screen

Syntax:

ALTSCR [ON | OFF]

Comments:
The ALTSCR command allows you to redirect the Soft-ICE output from your default screen to the alternate screen. This feature is useful, for instance, when you want to debug a graphics program without having to switch between the Soft-ICE window and the graphics display.

ALTSCR requires the system to have two monitors attached. The alternate monitor should be in a character mode, which is the default mode for monitors.

The default is ALTSCR mode OFF.

Example:
ALTSCR ON
This command redirects screen output to the alternate monitor.





WIN -- Change size of Soft-ICE window

Syntax:

WIN [N | W] [start-row length [start-column]]
N -- When N is specified, the window will be set to the narrow width: 46 characters.

W -- When W is specified, the window will be set to full screen width.

start-row -- Number from 0 to 17 specifying row where window display starts.

length -- Number from 8 to 25 specifying how many lines tall you want the window to be.

start-column-- Column position of the left side of narrow window. The start-row and start-column specify the upper left hand corner of the narrow window. The start-column is ignored if applied to the wide window.

Comments:
The WIN command allows you to modify the width and height of the Soft-ICE display window.

If no parameters are specified, this command toggles the window between wide and narrow screen display modes.

If the WIN command is specified with only the N or the W parameter, the window size will be changed to the requested width at the current height.



If the number of lines plus the starting row number is larger than 25, the window length goes to the bottom of the screen.

The default is WIN mode narrow.

Examples:

WIN N 4 9 30
This command causes the window display to start at row 4 and column 30, and to be 9 rows tall and 46 characters wide.

WIN
This command toggles the window display width from its current state (either wide or narrow) to the opposite state.

WIN W 10 8
This command causes the window display to start at row 10, and to be 8 rows tall and go the width of the screen.



## 5.10 Symbol and Source Line Commands

Commands:

| | |
|---|---|
| SYM | -- Display/set symbol |
| SYMLOC | -- Relocate symbol base |
| SRC | -- Toggle between source, mixed and code |

FILE -- Change/display current source
SS -- Search current source file for string



SYM

SYM -- Display/set symbol

Syntax:

SYM [symbol-name [value]]
symbol-name -- A valid symbol name. The symbol
name can end with an * (asterisk).
This allows searching if only the first
part of the symbol name is known.
The , (comma) character can be used
as a wild card character in place of
character in the symbol-name.

value -- This is a word value that is used if you
want to set a symbol to a specific value.

Comments:
The SYM command allows displaying and setting of symbols. If SYM is entered with no parameters all symbols are displayed. The value of each symbol is displayed next to the symbol name.

If a symbol name is specified with no value then the symbol name and value are displayed. If the symbol name was not found then nothing is displayed.

The SYM command is often useful for finding a symbol name when you can only remember a portion of the name Two wild card methods are available for locating symbols. If symbol-name ends with an *, then all symbols that match the actual characters typed prior to the * will be displayed regardless of their ending characters. If a , is used in place of a specific character in symbol-name, that character is a wild card character.

If value is specified, all symbols that match symbol-name are set to the
value. All symbols have word values.



Examples:
SYM FOO*
All symbols that start with FOO are displayed.
SYM FOO* 6000

All symbols that start with FOO are given the value 6000.





SYMLOC -- Relocate symbol base

Syntax:

SYMLOC segment-address

Comments:
The SYMLOC command relocates the segment components of all symbols relative to the specified segment address. This function is necessary when debugging loadable device drivers or other programs that can not be loaded directly with LDR.EXE.

When relocating for a loadable device driver, use the value of the base address of the driver as found in the MAP command. When relocating for an .EXE program, the value is 10H greater than that found as the base in the MAP command. When relocating for a .COM program, use the base segment address that is found in the MAP command.

The MAP command will display at least two entries for each program. The first is typically the environment and the second is typically the program. The base address of the program is the relocation value.

Example:

```
SYMLOC 1244 + 10
```

This will relocate all segments in the symbol table relative to 1244. The + 10 is used to relocate a TSR that was originally a .EXE file. If it is a .COM file the + 10 is not necessary.





SRC -- Toggle between source, mixed and code

Syntax:

```
SRC [?]
```

Comments:
The SRC command toggles between source mode, mixed mode and code mode in the code

window.

If SRC ? is entered, the current state is displayed.

Example:
```
SRC
This command changes the current mode of the code
window. If the mode was source, it becomes mixed.
the mode was mixed, it becomes code. If the mode was
code, it becomes source.
```

Default-Function Key: F3



## FILE

FILE -- Change/display current source file

Syntax:

```
FILE {file-name]
```

Comments:
If a file-name is specified, that file becomes the current file and the start of the file is displayed in the code window. If no name is specified, the name of the current source file (if any) is displayed.

The FILE command is often useful when setting a break point on a line that has no associated public symbol. Use file to bring the desired file into the code window, use the SS command to locate the specific line, move the cursor the specific line, then type BPX to set the break point.

Note:
Only source files that have been loaded into extended memory with LDR.EXE are available with the FILE command.

Example:
```
FILE MAIN.C
If MAIN.C had been loaded with LDR.EXE, this
command brings it up in the code window starting with
line 1.
```



## SS

SS -- Search current source file for string

Syntax:

SS [line-number] [' string']
line-number -- a decimal number

string -- a character string surrounded by quotes
The quotes can be either single quotes
or double quotes.

Comments:
The SS command searches the current source file for the specified character string. If there is a match, the line that the string was located in will be displayed as the top line in the code window.

The search starts at the specified line number. If no line number is specified the search starts at the top line displayed in the code window.

If no parameters are specified, the search continues for the previously specified string.

Note:
The code window must be visible and in source mode before using the SS command.

Example:
SS 1 'if (i = = 3)'
The current source file is searched starting at line 1 for the string 'if (i = = 3)'. The line containing the next occurrence of the string becomes the top line displayed in the code window.