Soft-ICE 2.5 Addendum

CONTENTS

Introduction

This release memo is an addendum to the Soft-ICE 2.0 User's Guide. It describes the differences between the Soft-ICE 2.0 User's Guide and the Soft-ICE 2.5 release. When this memo is referring to enhancements or changes made to features that existed in the Soft-ICE 2.0 User's Guide, the memo headings will include, in parenthesis, the chapter number of the corresponding information in the Soft-ICE 2.0 User's Guide. Please read both the Soft-ICE 2.0 User's Guide and this release memo.

Product Description (1, 1)

There have been many features added to the Soft-ICE 2.5 release. The principal features are that Soft-ICE:

* integrates with BOUNDS-CHECKER.
* reads symbolic and source information directly from the .EXE header from Microsoft & Borland languages.
* has overlay support for Microsoft's LINK and Pocket Soft's .RTLink/Plus.
* can have two symbol tables loaded at the same time.
* allows 386 32-bit instruction dis-assembly and 32-bit register dump.
* is Microsoft C version 6 compatible.
* provides numeric processor dis-assembly.
* lets device drivers and T&SR programs load high
* includes some additional commands: BOUNDS, TABS, STACK, SERIAL.
* includes enhancements to some existing commands: R, FILE, SHOW, BPX, G.
* has VCPI support.

* allows remote debugging
* has 80486 support.
* allows customizes Soft-ICE window colors.

The Diskettes (2, 1),

A directory of a Soft-ICE 2.5 diskette will now show the following additional files:

```
\NEW\LH.
\NEW\LD.SYS
\NEW\ADDHI.EXE
\NEW\CE.EXE
\IOSIM.ASM
```

LH.EXE is a utility that loads high T&SRs. LD.SYS is a utility that loads high DOS loadable device drivers. ADDHI.EXE is a utility that adds high memory to DOS memory chain. CE.EXE is the CONFIG.SYS editor. IOSIM.ASM is an example of a user qualified break point. It will take a BPIO break point and log all the values that were written to or read from that port.

Loading Soft-ICE (2.2)

Follow the installation instructions in the Soft-ICE 2.0 User's Guide to copy all the files from the root directory of the distribution diskette to your Soft-ICE directory on your hard disk. In addition, copy the files from the /WW directory on the distribution diskette to your Soft-ICE directory on your hard drive; these files are new with the Soft-ICE 2.5 release.

Note

If you were previously using Soft-ICE 2.0 with the /EMM option on the S ICE command line in CONFIG.SYS, you need to run EMMSETUP. Since EMMSETUP writes configuration information directly into the S ICE.EXE file, this information was over-written when you loaded Soft ICE 2.5.

Using Soft-ICE with BOUNDS-CHECKER (New feature)

Introduction

BOUNDS-CHECKER gives you the protection of a protected mode operating system under MS-DOS. When your program is running, BOUNDS-CHECKER protects your program's CODE and all memory outside your program. When an MS-DOS system call or BIOS call or interrupt occurs, BOUNDS-CHECKER prevents the system software from corrupting your program. So BOUNDS-CHECKER can not only detect problems caused by your program, it can also determine if a T&SR or other program is clobbering you.

Each time you make a change to your program, run BOUNDS-CHECKER while testing the new code. Your program runs at full speed, and if you accidentally access out-of-bounds memory, BOUNDS-CHECKER pops up displaying the offending source line.

Using Soft-ICE in combination with BOUNDSCHECKER is very useful when the bug found by BOUNDS-CHECKER is not clearly self-explanatory. You may need to use Soft-ICE to look at data, to debug a little, or to rerun the program with Soft-ICE's back trace capability to determine why the out-of-bounds access occurred..

Loading BOUNDS-CHECKER to use with Soft-ICE 2.5

To use BOUNDS-CHECKER with Soft-ICE 2.5, you must first:

1. Install BOUNDS-CHECKER on your hard disk using the BOUNDS-CHECKER installation program (BCSETUP.EXE).
2. Replace the DEVICE=d:\path\BC.SYS line in your CONFIG.SYS file with DEVICE=d:\path\S-ICE.EXE.
   Use the same parameters that were on the BC.SYS command line. In addition, you may want to use the /TRA nnnn parameter to create a back trace buffer larger than 10K. You may also need to increase the size of your /SYM nnnn parameter to allow your source and your symbols to be loaded.

   Notes

   You do not need the /BC switch on the DEVICE=d:/path /S-ICE.EXE line in CONFIG.SYS as the BOUNDS-CHECKER manual states.
   You must have Soft-ICE version 2.5 or greater and BOUNDS-CHECKER version 1.1 or greater for them to coexist.

Running Soft-ICE 2.5 with BOUNDSCHECKER

Run BOUNDS-CHECKER. When BOUNDSCHECKER pops up, if you want to enter Soft-ICE to do further debugging, select Options on the main menu, then select Soft-ICE. To re-enter BOUNDS-CHECKER, simply exit Soft-ICE with the hot key sequence or the X command.

If you don't have enough extended memory to run BOUNDS-CHECKER, you can save space by running BOUNDS-CHECKER with option /S in this form:

```
BC /S program-name
```

This stops source from loading up into extended memory for use by Soft-ICE. The disadvantage is that Soft-ICE will show line numbers, but will not show source code.

> Note
>
> Soft-ICE range break points and back trace ranges will be disabled while the BOUNDS-CHECKER is running.

The Soft-ICE BOUNDS Command

The new command, BOUNDS, is used for turning BOUNDS-CHECKing on and off from within SoftICE. This is useful if you want to stop to do some debugging from within a BOUNDS-CHECKER session, then return to BOUNDS-CHECKing after you have debugged a portion of the program.

The syntax of the BOUNDS command is:

```
BOUNDS [ON | Off]
```

BOUNDS OFF turns off BOUNDS-CHECKing, and BOUNDS-ON turns BOUNDS-CHECKing back on. If no parameters are specified, then the current state is displayed.

Overlay Support (New feature of BPX and G commands)

The Soft-ICE BPX break point will follow overlays produced by the Microsoft linker or.RTLink/Plus.

The BPX and G commands allow you to use break points in overlays. Other break point types do not follow overlays. To use BPX to set a break point in an overlay, you must type:

```
BPX routine_name
```

Other forms of BPX, such as using source line numbers or setting BPX using point-and-shoot, do not follow break points in overlays.

32 Bit Dis-assembly and Register Display (5.1 - R command)

Soft-ICE now displays 32 bit 80386 instructions properly. The Dis-assembly is always enabled. To enable 32 bit register display in the data window enter:

```
          R 32
```

This toggles between 16 bit and 32 bit registers.

STACK Command (New command)

Soft-ICE 2.5 now allows you to display the call stack. A call stack is a list of routines that were called to reach the current address. Using the call stack is especially useful when Soft-ICE pops up in a library routine. By using the call stack, you can quickly see the last routine in your program that had control before entering the library, even if the program is several levels deep into library calls.
The most recently called entry in the stack is displayed first in the command window.

The format of the call stack is:

```
     procedure(offset) [line-number]
```

If line-number is a '?' then no line number information was available for this procedure.

The STACK command can only be used if symbolic information is loaded.
If the module of an entry in the call stack was not compiled with debug information, no symbolic label will be displayed. Only a hexadecimal offset will be shown.

SHOW Command (5.6)

The SHOW command has been enhanced to allow you to dump large amounts of back traced instructions to the printer.
The new syntax for SHOW is:

```
    SHOW [B | start] [L length]
   B - start at beginning of buffer
   start - number of instructions back to begin
   length - number of instructions to display
```

If SHOW is used with the length argument, you can use CTRL P to dump the Dis-assembled source to the printer. If you don't specify B or start, it starts displaying at the current location.

FILE Command (5.10)

The FILE command has been enhanced. The new syntax for FILE is:

```
FILE [file-name |*]
```

FILE * displays all source files that have been loaded by LDR.EXE into extended memory.

To switch to a new file with the FILE command you no longer have to type the full path name or file extension. For example, to switch to file C:\SOURCE\FOO.C, pop up Soft-ICE and enter FILE FOO.

Preparing For Symbolic or Source Debugging (7,2)

Microsoft and Turbo Source/Symbolic Improvements

Soft-ICE 2.5 has made source and symbolic improvements for users of Microsoft or Turbo languages.

Soft-ICE can now get the symbolic and source information directly from the .EXE file if there is Microsoft CodeView compatible or Turbo Debug compatible debug information in the.EXE file. MSYM.EXE, the.MAP file, and the.SYM file are no longer needed if the debug records are present. With Microsoft, compile with /Zi and link with /CO. With Turbo, compile with /v and link with /v.

The /CO switch makes the linker append symbolic information to the end of your.EXE file. although this will make your.EXE file grow in size, this will not affect the amount of conventional memory required by your program.

Note

MSYM.EXE is useful when you are using a compiler that produces a Microsoft Link compatible .MAP file, but does not place Microsoft compatible debug information in the .EXE file.

Loading Programs and Symbol Files (7,4 and new TABLE command),

Multiple Symbol Tables

Soft-ICE 2.5 can now handle two symbol tables. This is useful when debugging a T&SR or DOS loadable device driver with an application, or debugging a shell with a child process.

To load a separate symbol table or a separate program with symbols use the Soft-ICE TABLE command. TABLE 1 uses symbol table number one, TABLE 2 uses symbol table number 2.

To use two symbol tables, do the following:

1. Use LDR to load your first program and symbolic information.
2. Pop up Soft-ICE.
3. Enter TabLE 2.
4. Exit Soft-ICE.
5. Use LDR.EXE to load the second symbol table.

Both sets of symbolic information are now loaded into extended memory and you are currently viewing the second set of symbolic information. Use the TABLE 1 and TABLE 2 commands to toggle between which set of symbolic information you are currently viewing.

To view your first program's symbolic information, pop up Soft-ICE if it's not up already, and enter:

```
TABLE 1
```

To view your second program's symbolic information, pop up Soft-ICE if it's not up already, and enter:

```
TABLE 2
```

If you enter TABLE without any parameter, it will tell you which set of symbolic information is currently being viewed.

> Note
> When you re-load table 1 by entering TABLE 1 then loading with LDR, table 2 is invalidated.

Tabs Control (New TABS command)

Soft-ICE 2.5 allows you to control tab expansion size of source files. Previous versions of Soft-ICE assumed tabs of every 8. The syntax for the TABS command is:

```
TABS [2 | 4 | 8]
```

If no parameter is specified then the current tabs setting is displayed. An interesting use of the TABS command is to see more than 78 characters of source on a single line. To do this enter TABS 2.

Remote Debugging (New feature and new SERIAL command)

Soft-ICE is capable of displaying all of the information from the command window over a serial port. The hot key is still activated via the system keyboard but once Soft-ICE is popped up, both the system and the remote keyboard will be active. To activate remote debugging use the following sequence:

1) Set the BAUD rate with the DOS MODE command to the same baud rate as the remote terminal.

2) Within Soft-ICE, set PRN to the correct serial port.

   Example: PRN COM1

3) Within Soft-ICE, enter SERIAL ON. At this point, you may enter information on either keyboard, and the command window output will go to both screens.

4) You will probably want to get rid of your Code, Data and Register windows as these will not be displayed across to the remote terminal.

5) If you do not want the Soft-ICE screen up on the host machine then turn ALTSCR ON from within Soft-ICE.

486 Support (New Feature)

Soft-ICE 2.5 now has 80486 support. Previous versions did not, due to anomalies with the 80486 processor. Also 486 instructions will be disassembled correctly

Special Configuration Options (6.4.1)

Soft-ICE color support

Soft-ICE 2.5 now allows you to specify the screen colors for the Soft-ICE windows.

To set colors for the different windows in Soft-ICE you must use the COLORS directive in the S-ICE.DAT file. The syntax of the COLORS command is:

```
COLORS = "nnH,nnH,nnH,nnH,nnH,nnH, nnH,nnH,nnH,nnH,nnH,nnH
```

where nn is a HEX number. There are four sets of three numbers. Each grouping of three affects the colors of a particular window. The ordering for the four Soft-ICE windows is:

```
COLORS register-window, data-window, code-window, command-window
```

Each grouping of three HEX numbers controls the normal attribute, highlight attribute and reverse attribute for the respective window. The actual number is the value that is placed in the attribute field in the video frame. The high order nibble is the background color and the low order nibble is foreground color. The colors are:

0 - black 1 - blue 2 - green

3 - cyan 4 - red 5 - magenta 6 - brown 7 - gray
8-0F are intense versions of the above.

An example of the colors command is:

```
COLORS = "47H,4EH,7EH, 07H,OFH,70H, 17H,lFH,71H,30H,3FH,71H"
```

Expanded Memory Support (8)

The expanded memory manager has been enhanced in Soft-ICE 2.5, and utilities are now provided to load device drivers and T&SR programs into extended memory. (The expanded memory manager is enabled with the /EMM command when S-ICE.EXE is placed in your CONFIG.SYS.)

EMMSETUP.EXE Changes

Running EMMSETUP.EXE (the expanded memory manager setup program) now requires a command line parameter that specifies the name of the expanded memory manager file. The syntax for running EMMSETUP is:

```
EMMSETUP file-name
```

The file-name parameter should be S-ICE.EXE. EMMSETUP can also be used with MagicCV release 3.0, in which case the parameter should be NUMEGA.SYS. This parameter is required

because EMMSETUP writes the configuration information directly into the driver file.
EMMSETUP now has the option of enabling memory blocks for loading high of device drivers and T&SR programs. You must select this feature on EMMSETUP's initial screen.

The EMMSETUP configuration memory map now has more choices. You can choose F for page frame, and H for high memory areas. To enable expanded memory you must have 4 and only 4 contiguous F's above 640K. To load high device drivers or T&SRs you must place H's in UN-occupied memory blocks above 640K.

Note

> If you want to load high device drivers & T&SR programs, but do not want EMM (expanded) memory, then make sure there are no E's or F's in the memory map.

Loading High Of Resident Programs (New feature)

The LH.EXE utility allows loading certain resident programs into available memory blocks between 640K and I megabyte. Before using LH.EXE you must reserve memory for loading high using EMMSETUP.EXE. This is done by placing an 'H' in each memory block above 640K that you wish to have as a load high area.
To load a resident program high enter:

```
     LH program-name [program parameters]
```

If there is a high memory block large enough to hold the program, the program will be loaded into it.

If no program-name follows LH on the command line, a memory map is displayed of the DOS loadable device drivers and resident programs loaded high along with available memory.

Note

> You can not load all resident programs with LH.EXE. You must experiment to see which programs can be loaded high.

Loading High Of MS-DOS Loadable Device Drivers (New feature)

The LD.SYS utility allows loading certain MS-DOS loadable device drivers into available memory blocks between 640K and I megabyte. Before using LD.SYS you must reserve memory for loading high using EMMSETUP.EXE. This is done by placing an 'H' in each memory block above 640K that you wish to have as a load high area.

To load an MS-DOS loadable device driver high, you must place the following line in your CONFIG.SYS file:

```
DEVICE = \path\LD.SYS device-name [parameters]
     path - Path containing LD.SYS
     device-name - Name of DOS Loadable
            device driver including path
```

If there is a high memory block large enough to hold the device driver, the program will be loaded into it when you boot.

To display a memory map of DOS loadable device drivers and resident programs loaded high use the LH utility with no parameters from DOS.

Note

You can not load all DOS loadable device drivers high. You must experiment to see which drivers can be loaded high. Make sure you have a boot disk handy While experimenting.

Adding High Memory to MS-DOS (New feature)

The ADDHI.EXE utility allows you to add high memory areas to the DOS pool of free memory. Before using ADDHI.EXE you must reserve memory for adding high using EMMSETUP.EXE. This is done by placing an 'H' in each memory block above 640K that you wish to have as a add high area.

VCPI Support (New feature)

VCPI (Virtual Control Program Interface) is automatically enabled when you use the /EMM switch on the S-ICE.EXE line in CONFIG.SYS. VCPI support lets you run VCPI applications that use DOS extenders when Soft-ICE is loaded. It does not allow you to debug these applications in protected mode. VCPI conforming applications include Lotus 123 version 3.0 and Autocad.

VCPI support does NOT enable Soft-ICE to run with other VCPI control programs, such as Quarterdeck's QEMM and Qualitas's 386MAX.

CONFIG.SYS Editor (New feature)

CONFIG EDIT (CE.EXE) is an on-the-fly text editor for CONFIG.SYS. CONFIG EDIT is useful if you have to make occasional changes to your CONFIG.SYS. It is especially useful if you suspect that a driver in CONFIG.SYS may hang the system. It is advisable to use CONFIG EDIT when installing SoftICE in your CONFIG.SYS for the first time.
Install CONFIG EDIT by placing CE.EXE as the first DEVICE = line in your CONFIG.SYS file. For example:

```
   DEVICE = /S-ICE /CE.EXE
```

When your system boots, you will hear a tone. After the tone, you have a short time to press any key. If you press a key CE will take over and allow you to edit CONFIG.SYS.

When you have edited your CONFIG.SYS file, you may exit CE by pressing one of the following keys:

F1 Pressing F1 exits and changes
CONFIG.SYS for this boot only. The
changes are not permanent.

F10 Pressing F10 exits and changes
CONFIG.SYS for this boot and
subsequent boots.

ESC Pressing ESC exits with no changes

CE can also be run From the DOS command line. This is for a quick look or quick changes to CONFIG.SYS. Simply enter CE from the DOS command line.
The /Q switch (Quiet) will disable the initial sound made by CE. when it is installed in CONFIG.SYS.

Back Door Commands (New feature)

Soft-ICE 2.5 contains commands for controlling SoftICE from an MSDOS program. A program can take advantage of powerful break points for special debugging jobs or hardware simulation projects.

These calls all have the following calling sequence:

```
    MOV AH,09
    MOV AL,SUB-FUNCTION
    MOV SI,'FG'
    MOV DI,'JM'
    INT 3
```

The sub-functions are available:

AL value	Description
10H	Display information in the Soft-ICE window.
11H	Do a Soft-ICE command.
12H	Get break point information.
13H	Set Soft-ICE break point.
14H	Remove a Soft-ICE break point.

The following paragraphs give more detailed information about these subfunctions.

AL = 10H --Display Information In the Soft-ICE window.

This is useful for diagnostic writes - especially from within interrupt routines and other areas that may have reentrancy concerns.

Input: DS:DX - > Zstring of text characters to be
displayed

The Zstring can be a maximum of 100 characters and can contain carriage returns (0DH).

AL = 11H -- Do a Soft-ICE command.

This allows you to generate a Soft-ICE command from your program. This is used for all non-break point commands. To set Soft-ICE break points from your program see AL = 13H below.

Input: DS:DX - > Zstring that contains a Soft-ICE
command.

The Zstring can be a maximum of 100 characters. Each Soft-ICE command in the string should end with a carriage return (0DH).

AL = 12H -- Get break point Information.

Returns the break point number of the last break point set and the last break point that went off.

This is useful when setting break points from hardware control or doing hardware simulation.

Returns: DH - entry number of last break point that
went off
DL - type of last break point that went off
BH - entry number of last break point set
BL - type of last break point set

The entry number is the same as is displayed in the BL command.

The types are: 0 - BPM (break point register types)
1 - I/O
2 - INT
3 - BPX (int 3 style BP)
4 - Reserved
5 - Range

AL = 13H -- Set Soft-ICE break point.

Use this command to set Soft-ICE break points from program control.

Input: DS:DX - pointer to break point structure

Returns: ax = error code
bx = break point number

; Very little parameter value checking is done, but the following ; errors are returned.

OK EQU 0 BP_TABLE_FULL EQU 3 MEM_LlM_ERR EQU 6 IO_LlM_ERR EQU 7
RANGE_LlM_ERR EQU 9 DUP_ERR EQU l6 ;duplicate break point

; Break point structure

```
;bp_entry       struc
;bp_type                 db ?
;bp_addrl       dd ?
;bp_addr2       dd ?
;bp_addr3       dd ?
;bp_mode                 db ?
;bp_mode2       db ?
;bp_size                 db ?
;bp_cnt                  db ?
;bp_state       db ?
;bp_entry       ends
```

; The following break point types are allowed:

```
MEM_LOC                  equ 0  ;Memory Iocation
    break point (BPM).
MEM_RANGE                equ 1  ;Memory range
    break point (BPR).
IO                       equ 3  ;I/0 break point
```

```
    (BPIO).
INT_BP                  equ 4  ;Interrupt break
    point (BPINT).
X_BP            equ 5  ;Execution break
    point (BPX).

; Here are the possible break point modes and sizes.

; Break point modes

READ_MODE          equ 01
WRITE_MODE         equ 02
EX_MODE            equ 04

; Break point sizes

BYT             equ 0
WRD             equ 1
DBL             equ 3

; The following paragraphs give information on how to fill the break point structure
```

```
; for each break point type.

; Setting memory location break points

  bp_type = MEM_LOC
  bp_addr1 = address of break point
  bp_mode = one of following:
  READ_MODE
  WRITE_MODE
  EX_MODE or WRITE_MODE
  EX,MODE (execute break point)
  bp,size = one of following:
  BYT
  WRD
  DBL
  bp,cnt = Number of instances before
  breakpoint occurs

; All unused fields should be 0.

; Setting memory range break points
    bp_type = MEM_RANGE
```

```
     bp_addrl = lower range limit
     bp_addr2 = upper range limit
     bp_mode = one of following:
     READ_MODE
```

```
   WRITE_MODE
   READ_MODE or WRITE_MODE
   bp_cnt = Number of instances before breakpoint occurs

; All unused fields should be 0.

; Setting I/O break points

    bp_type = I/O
    word ptr bp_addr1 = I/O address
    bp_mode = one of following:
    READ_MODE
    WRITE_MODE
    READ_MODE or WRITE_MODE
    bp_cnt = Number of instances before
    breakpoint occurs

; All unused fields should be 0.

; Setting interrupt break points
```

```
    bp_type = INT_BP
    bp,addr1 = Interrupt #
    bp,addr2 = Optional value to check
    bp,mode = register to check
            0 - no value checking
            1 - check AL
            2 - check AH
            3 - check AX

;Setting execution break points

   bp_type = X_BP
   bp,addr1 = address of break point
   bp,addr2 = overlay number (0 = root)

AL = 14H -- Remove Soft-ICE break point.
```

Input: BX = Break point number

Returns: BX = ??? when set

Page 40 is blank