SECTION III - Advanced Topics

CHAPTER 11

Advanced Features

11.1 Using Soft-ICE with other Debuggers

Soft-ICE was designed to work well with other debuggers. Each debugger offers different features, and therefore can require special treatment. This section will describe some ways to use several debuggers effectively.

11.1.1 Debuggers that Use DOS

Many debuggers use DOS and ROM BIOS to perform their display and keyboard I/O. Special consideration must be taken when using these debuggers with Soft-ICE (e.g., DEBUG, SYMDEB, and CODEVIEW), because DOS and ROM BIOS are not fully re-entrant. If a break point occurs while code is executing in DOS or BIOS, a re-entrancy problem can occur.

Soft-ICE provides optional re-entrancy warning, which is activated with the WARN command. When WARN mode is on, Soft-ICE checks for DOS or ROM BIOS re-entrancy before generating the ACTION that wakes up the host debugger. When a re-entrancy problem is detected, Soft-ICE displays a warning message and offers you the choice of continuing to execute the code or returning to Soft-ICE.

Note that Soft-ICE itself does not use DOS or ROM BIOS calls in its debugging commands. This means that you can use Soft-ICE any time, without the worry of re-entrancy problems.

For more information on the WARN command, see section 5.4.

11.1.2 ACTION Command with other Debuggers

Different debuggers use different methods of activation For a description of these methods see section 13.1.

If you want to return to your debugger after a break point reached, you must change the ACTION (see section 5.4) to work with your debugger.

In most cases, the action that should be taken after a break point is reached is INT3. For instance, DEBUG and SYMDEB will work best with ACTION set to INT3.

If INT3 doesn't work with your debugger, try INT1 or NMI. CODEVIEW works best with ACTION set to NMI.

11.1.3 Special Considerations

When a break point is set, you must be careful not to set off the break point unintentionally. For instance, if you set a memory break point at 0:0, then use your debugger to dump memory location 0:0, Soft-ICE will be triggered. If ACTION is set to go to your debugger, then your debugger will be triggered by itself. Since some debuggers cannot be re-entrant, this could be a fatal problem. This problem can also occur with other debugging functions, such as editing or unassembling.

For this reason, it is a good practice to disable the Soft-ICE break points once Soft-ICE has helped you get to the point where you want to look around with your debugger.

11.1.4 Using Soft-ICE with CODEVIEW

Soft-ICE works best with CODEVIEW when CODEVIEW is either in Assembler mode or Mixed mode. When CODEVIEW is in Source mode with higher-level languages it does not always break correctly.
It is always best to use ACTION NMI when you want Soft-ICE to wake up CODEVIEW.

11.1.5 Debuggers that Use 80386 Break Point Registers

The 80386 has 4 break point registers that are available for use by debuggers. Soft-ICE uses these for its memory byte, word and double word break points. If the debugger you are using Soft-ICE with uses these debug registers there will be a conflict. There are two ways to handle this problem.

1. Disable the use of 80386 break point registers in the debugger you are using Soft-ICE with. Check the documentation of your other debugger for a description of how to do this.
2. Some debuggers automatically use the break point registers if they detect an 80386 processor with no method of turning them off (some versions of SYMDEB do this). For these debuggers do the following:
   * Bring up the Soft-ICE window before you start the other debugger.
   * Turn on Soft-ICE's break mode with the BREAK command (you may want to do this in the INIT statement of S-ICE.DAT if you are doing this frequently).
   * Start up your other debugger.
   * You may now pop up the Soft-ICE window and turn the Soft-ICE break mode off if desired.

11.2 User-Qualified Break Points

Occasionally you may have the need for a very specific set of break point conditions. If the special conditions require qualifying register values or memory values, you can write a break point qualification routine.

Soft-ICE contains a very general mechanism for calling user-written break point qualification routines: the ACTION command. When you use the ACTION command, Soft-ICE can route all break points through special interrupt vector. However, before break points can be routed, the qualification routine must be placed in memory, and the interrupt vector must be pointing to the qualification routine.

All registers are identical to the values when the Soft-ICE break point occurred. It is the responsibility of the qualification routine to save and restore the registers. If your qualification routine detects a match of break point conditions, it can do a variety of activities. Some examples of useful activities that a routine can do when a match is found are:

* store information for later
* send the information directly to a printer or serial terminal
* issue an INT 3 instruction to bring up Soft-ICE
  The command 13HERE must be turned on in order for the INT 3 to bring up Soft-ICE (see section 5.4).

If conditions do not match, the qualification routine in should execute an IRET instruction. To summarize:

1. Create a break point qualification routine in your code space, or anywhere in free memory. The routine must preserve registers. After comparing the desired conditions, the routine can execute either an INT 3 to bring up Soft-ICE, or an IRET to continue.
2. Point an unused interrupt vector to your qualification routine. This can be done either within your code or from Soft-ICE.

3. In Soft-ICE, set ACTION to the interrupt- number that was used to point to your qualification routine.
4. In Soft-ICE, set 13HERE on. This is necessary to bring up Soft-ICE after the conditions have been met.
5. Set the Soft-ICE general break point conditions. When any of these break point conditions are met, your qualification routine will be called.

11.2.1 Example of a User-Qualified Break Point

This section contains an example of a user-qualified break point that compares for the conditions of U = 3, BX = 4 and CX = 5 when a break point goes off.

First, we create the qualification routine. For the purposes of this example, we will assemble the command directly into memory with the Soft-ICE interactive assembler. For this example we will arbitrarily assemble the routine at location 9000:0H. The following statements are entered into Soft-ICE:

```
A 9000:0
9000:0 CMP AX,3
9000:3 JNE 10
9000:5 CMP BX,4
9000:7 JNE 10
9000:A CMP CX,5
9000:D JNE 10
9000:F INT3
9000:10 IRET
```

Now that the routine is in memory, you must point an interrupt vector to the routine.

For this example, we arbitrarily pick INT 99H. To place 9000:0H in the INT 99H vector enter:

```
ED 0:99*4 9000:0
```


Set the ACTION command so that Soft-ICE will call your break point qualification routine on every break point.

```
ACTION 99
```

Set 13HERE on so the qualification routine can activate Soft-ICE when the conditions occur.

```
13HERE ON
```

Now you need to set the break points. For this example, we are just interested when the registers are: U = 3, BX = 4, CX = 5 in a specific program, and we do not want any further qualification. To do this, use a range break point on memory read:

```
BPR segment:starting-offset segment:ending-offset
```

This will cause your break point qualification routine to be called after every instruction is executed in the specified memory range. When the register conditions do not match, then the IRET instruction is executed. When the conditions finally match the specified qualifications, the INT 3 is executed and Soft-ICE is popped up.

When Soft-ICE pops up, the instruction pointer will be pointing at the INT3 in your qualification routine (9OOO:FH in our example). To get to the instruction after the one that caused the break point, you must change the instruction pointer to point to the IRET instruction (F000: 10H in the example) and single step one time. This is accomplished with the following Soft-ICE commands

```
RIP IP + 1
T
```

After your break conditions have gone off, remember to change the ACTION command back to ACTION HERE that subsequent break points do not go through your qualification routine.

11.3 The Window in Graphics Mode

The screen is switched to text mode when Soft-ICE is invoked. If the screen was in

graphics mode or 40-column mode, the graphics display is not visible while the window is up. For users who must see the graphics display while debugging, three features are provided. The first feature allows the Soft-ICE window to display on a second monitor (see the ALTSCR command, section 5.9). The second feature allows you to restore the screen while you are doing P or T instruction step commands (see the FLASH command, section 5.9). The third feature allows you to restore the program screen temporarily (see the RS command, section 5.9).

If Soft-ICE does not seem to be following your program into graphics mode, try turning WATCHV on (see section 5.9 for details).

11.4 Expanded Memory Debugging Features

A range break point or a break point on memory that is set in an EMM mappable area will stay at that address no matter which EMM page is mapped in.

When debugging EMM programs, the EMMMAP command may also be very useful. See section 5.6 for more information.

The D, E, S, F, and C commands can be used to view or modify any allocated EMM handle page. The page does not have to be currently mapped in. The syntax of these commands is similar to that of the commands when being used for non-EMM pages, except for the following:

* In the D, E, S, and F commands, the address portion of the command must be specified in the following way:
  Hhandle# Ppage# offset

  where handle is a number specifying which EMM handle to use, page is a number specifying which EMM page to use, and offset is a number from 0 to 4000H, specifying the offset from the beginning of the page.
  Example:

  DB H1 P3 0

  This command will dump bytes from page 3 of handle 1, starting at offset 0.

* The C command must be specified in the following way:

  C Hhandle# Ppage# offset1 L length offset2

where handle and page are the same as above.
offset1 is a number from 0 to 4000H, specifying the offset from the beginning of the page, where the first data block to be compared is located.
offset2 is a number from 0 to 4000H, specifying the offset from the beginning of the page, where the second data block to be compared is located.
Example:

C H2 P4 00 L10 1000

This command will compare the first 10 bytes of memory located at offset 0 of page 4 of handle 2 with the first 10 bytes of memory located at offset 1000 of page 4 of handle 2.

Note:
Subsequent uses of the D, E, S, F, and C commands will continue to use the handle and page last specified. To get back to conventional memory, use one of the above

commands with a segment specified in the address field, for example:
D 0:0

11.5 Extended Memory Debugging Features

The D, E, S, F, and C commands can be used to view or modify extended memory. Extended memory reserved by Soft-ICE can not be displayed. The syntax of these commands is similar to that of the commands when being used for conventional memory:

* In the D, E, S, and F commands, the address portion of the command must be specified in the following way:
  M megabyte address
  where megabyte is a number specifying which megabyte to use, and address specifies the address in the specified megabyte.
  Example:
  DB M 2 0:0
  This command will dump bytes from start of the megabyte starting at linear address 200000H.
* The C command must be specified in the following way:

C M megabyte address1 L length address2
where megabyte and address1 are the same as above.
address2 specifies the address in the specified
megabyte, where the second data block to be
compared is located.

Example:

C M 3 1000:2000 L10 3000:4000

This command will compare the first 10 bytes of
memory located at 1000:2000 with the first 10 bytes
of memory located at 3000:4000.

Note:
Subsequent uses of the D, E, S, F, and C commands will continue to use the last megabyte specified. To get back to megabyte 0 (conventional memory), use one of the above commands with 0 specified as the megabyte, for example:

D M 0

Page 220 is BLANK