CHAPTER 4

Using Break Point Commands

4.1 Introduction
4.2 Setting Break Points
4.3 Manipulating Break Points

4.1 Introduction

Soft-ICE has break point capability that has traditionally only been available with hardware debuggers. The power and flexibility of the 80386 chip allows advanced break point capability without additional hardware.

Break points can be set on memory location reads and writes, memory range reads and writes, program execution and port accesses. Soft-ICE assigns a one-digit hexadecimal number (0-F) to each break point. This break-number is used to identify break points when you set delete, disable, enable, or edit them.

All of Soft-ICE's break points are sticky. That means they don't disappear automatically after they've been used; you must intentionally clear or disable them using the BC or the BD commands. Soft-ICE can handle 16 break points at one time. You can have up to ten break points of a single type except for break points on memory location (BPMs), of which you can only have four, due to restrictions of the 80386 processor.

Break points can be specified with a count parameter. The count parameter tells Soft-ICE how many times the break point should be ignored before the break point action occurs.

4.2 Setting Break Points

Commands:
BPM, BPMB, BPMW, BPMD -- Set break point on memory
access or execution
BPR -- Set break point on memory range
BPIO -- Set break point on I/O port access
BPINT -- Set break point on interrupt
BPX -- Set/clear break point on execution
CSIP -- Set CS:IP range qualifier
BPAND -- Wait for multiple break points
to occur

BPM, BPMB, BPMW, BPMD

BPM, BPMB, BPMW, BPMD -- Set break point on memory access or execution

Syntax:

```
  BPM[size]address[verb][qualifier value][C=count]
  size -- B, W, D
                  B -- Byte
                  W -- Word
                  D -- Double Word
                  The size is actually a range covered by
                  this break point. For example, if
                  double word is used, and the third
                  byte of the double is modified, then a
                  break point will occur. The size is
                  also important if the optional qualifier
                  is specified (see below).
 verb -- R, W, RW, or X
 qualifier -- EQ, NE, GT, LT, M
                  EQ -- Equal
                  NE -- Not Equal
                  GT -- Greater than
                  LT -- Less Than
                  M -- Mask
                  These qualifiers are only applicable to
                  the read and write break points.
 value -- A byte, word, or double word value,
                  depending on the size specified.
```

Comments:

The BPM commands allow you to set a break point on memory reads or writes or execution.

If a verb is not specified, RW is the default.

If a size is not specified, byte is the default.

All of the verb types except X cause the program to execute the instruction that caused the break point. The current CS:IP will be the instruction after the break point. If the verb type is X, the current CS:IP will be the instruction where the break point was set.

If R is specified, then the break point will occur on read access and on write operations that do not change the value of the memory location.

If the verb type is R, W or RW, executing an instruction at the specified address will not cause the break point action to occur.

Note:
If BPMW is used, the specified address must start on a word boundary. If BPMD is used, the specified address must point to a double word boundary.

Example:

```
  BPM 1234:SI W EQ 10 C=3
```

This command defines a break point on memory byte access. The third time that 10 hexadecimal is written to location 1234:SI, the break point action will occur.

```
  BPM CS:1235 X
```

This command defines a break point on execution. The break point action will occur the first time that the

instruction at address CS:1235 is reached. The current CS:IP will be the instruction where the break point was set.

```
BPMW DS:FOO W EQ M 0XXX XXXX XXXX XXX1
```

This command defines a word break point on memory write. The break point action will occur the first time that location DS:FOO has a value written to it that sets the high order bit to 0 and the low order bit to 1. The other bits can be any value.

```
BPM DS:1000 W GT 5
```

This command defines a byte break point on memory write. The break point action will occur the first time that location DS:1000 has a value written to it that is greater than 5.

BPR -- Set break point on memory range

Syntax:

```
   BPR start-address end-address [verb] [C=count]
   start-address,
   end-address -- start-address and end-address specify
                  memory range.
   verb -- R, W, RW, T or TW
```

Comments:

The BPR command allows you to set a break point across a range of memory.

All of the verb types except T or TW cause the program to execute the instruction that caused the break point. The current CS:IP will be the instruction after the break point.

There is no range break point on execution. If a range break point is desired on execution, R must be used. An instruction fetch is considered a read for range break points.

If a verb is not specified, W is the default.

The range break point will degrade system performance in certain circumstances. Any read or write within the 4K page that contains the break point range is analyzed by Soft-ICE. This performance degradation is usually not noticeable, however, degradation could be extreme in exception cases.

The T and TW verbs enable back trace ranges on the specified range. They do not cause break points, but instead log instruction information that can be displayed later with the SHOW or TRACE commands. For more information on back trace ranges, see chapter 9.

Example:

```
  BPR B000:0 B000:1000 W
  This command defines a break point on memory range.
  The break point will occur if there are any writes to the
  monochrome adapter video memory region.
```

BPIO

BPIO -- Set break point on I/O port access

Syntax :

```
   BPIO port [verb] [qualifier value] [C=count]
   port -- A byte or word value
   verb -- R, W, or RW
               R -- Read (IN)
               W -- Write (OUT)
   qualifier -- EQ, NE, GT, LT, M
               EQ -- Equal
               NE -- Not Equal
               GT -- Greater Than
               LT -- Less Than
               M -- Mask
   value -- A byte or word value
```

Comments:

The BPIO command allows you to set a break point on I/O port reads or writes.

If value is specified, it is compared with the actual data value read or written by the IN or OUT instruction causing the break point. The value may be a byte or a word. If the I/O is to a byte port, then the lower 8 bits are used in the comparison.

The instruction pointer (CS:IP) will point to the instruction after the IN or OUT instruction that caused the break point.

If a verb is not specified, RW is the default.

Example:

```
 BPIO 21 W NE FF
```

This command defines a break point on I/O port
access. The break point will occur if the interrupt
controller one mask register is written with a value
other than FFH.

```
 BPIO 3FE R EQ M 11XX XXXX
```

This command defines a byte break point on I/O port
read. The break point action will occur the first time
that I/0 port 3FE is read with a value that has the two
high order bits set to 1. The other bits can be any value.

BPINT -- Set break point on interrupt

Syntax:

```
 BPINT int-number [ < AL | AH | AX >= value] [C = count]
 int-number -- Interrupt number from 0 - FF hex
 value -- A byte or a word value
```

Comments:

The BPINT command allows breaking on the execution of a hardware or a software interrupt. By optionally qualifying the AX register with a value, specific DOS or BIOS calls can be easily isolated.

If no value is specified, a break point will occur when the interrupt specified by int-number occurs. This interrupt can be a hardware, software, or internal interrupt.

The optional value is compared with the specified register (AH, AL, or AX) when the interrupt occurs. If the value matches the specified register, then the break point will occur.

When the break point occurs, if the interrupt was a hardware interrupt, the instruction pointer (CS:IP) will point to the first instruction within the interrupt routine. The INT? command can be used to see where execution was when the interrupt occurred. If the interrupt was a software interrupt, when the break point occurs, the instruction pointer (CS:IP) will point to the INT instruction causing the interrupt.

Example :

```
 BPINT 21 AH=4C
```

This command defines a break point on interrupt 21H
The break point will occur when DOS function call
4CH (terminate program) is called.

BPX

BPX -- Set/clear break point on execution

Syntax:

```
   BX [address] [C=count]
```

Comments:

The BPX command allows you to set or clear a point-and-shoot execution break point in source. When the cursor is in the code window the address is not required. The execution break point is set at the address of the current cursor location. If an execution break point has already been set at the address of the current cursor location, then the break point is cleared.

If the code window is not visible or the cursor is not in the code window then the address must be specified. If an offset only is specified then the current CS register value used as the segment.

Technical Note:
BPX uses an interrupt 3 style of break point unless the specified address is ROM. This is used instead of a break point register to make more execution break points available. If your circumstances require the use of a break point register for some reason (code not loaded yet for

example) you can set an execution break point with the BPM command.

Example:

```
  BPX.1234
   This sets an execution break point at source line 1234.
```

CSIP -- Set CS:IP range qualifier

Syntax:

```
   CSIP [OFF | [NOT] start-address end-address]
   NOT -- When NOT is specified, the break point
                 will only occur if the CS:IP pointer is
                 outside the specified range.
   OFF -- Turns off CS:IP checking
```

Comments:

The CSIP command causes a break point to be dependent upon the location of the instruction pointer when the break point conditions are met. This function is often useful when a program is suspected of accidentally modifying code outside of its boundaries.

When break point conditions are met, the CS:IP registers are compared with a specified range. If they are within the range, the break point is activated. To activate the break point when CS:IP is outside the range, use the NOT parameter.

When a CSIP range is specified, it applies to ALL break points that are currently active.

If no parameters are specified, the current CSIP range is displayed.
Example:

```
  CSIP NOT F000:0 FFFF:0
```

  This command causes the break points to occur only
  the CS:IP is NOT in the ROM BIOS when the break
  point conditions are met.

BPAND -- Wait for multiple break points to occur

Syntax:

```
 BPAND list | * | OFF
  list -- A series of break-numbers
               separated by commas or spaces
  * -- ANDs together all break points
```

Comments:

The BPAND command does a logical AND of two or more break points, activating the break point only when conditions for all break points are met.

Sometimes conditions arise when you don't want a break point to occur until several different conditions are met. The BPAND command allows specifying two or more break points that must occur before the action is generated. This function allows more complex break point conditions to be set.

Each time the BPAND command is used, the specified break point numbers are added to the list until BPAND OFF is used.

You can tell which of the break-numbers are ANDed together by listing the break points with the BL command. The break points that are ANDed together will have an ampersand (&) after their break-number.

Once break points have been ANDed together, each remains ANDed until it is cleared, or until BPAND is turned off.

Example:

```
BPAND 0,2,3
```

This command causes the conditions of the break points 0, 2, and 3 to be logically tied together. The break occurs only when the conditions of all three are met. For example, if the conditions of break points 2 and 3 have both been met at least once, but the conditions of break point 0 have not been met at all yet, then the action will not occur until break point 0 conditions are met.

4.3 Manipulating Break Points

Soft-ICE provides several commands for manipulating break points. Manipulation commands allow listing, modifying, deleting, enabling, and disabling of break points. Break points are identified by break-numbers which are hexadecimal digits from 0 to F. The break point manipulation commands are:

BD -- Disable break points
BE -- Enable break points
BL -- List break points
BPE -- Edit break point
BPT -- Use break point as a template
BC -- Clear break points

BD -- Disable break points

Syntax:

```
  BD list | *
  list -- A series of break-numbers separated
                  by commas or spaces
  * -- Disables all break points
```

Comments:

The BD command is used to temporarily deactivate break points. The break points can be reactivated with the BE (Enable break points) command.

You can tell which of the break-numbers are disabled by listing the break points with the BL command. The break points that are disabled will have an asterisk (*) after their break-number.

Example:

```
  BD 1,3
```

This command temporarily disables break points 1 and 3.

BE -- Enable break points

Syntax:

```
   BE list | *
   list -- A series of break-numbers separated
                  by commas or spaces
   * -- Enables all break points
```

Comments:

The BE command is used to reactivate break points that were deactivated by the BD (Disable break points) command.

Note that a break point is automatically enabled when it first defined.

Example:

```
  BE 3
```

This command enables break point 3.

BL -- List break points

Syntax:

```
  BL
```

Comments:

The BL command displays all break points that are currently set. For each break point, BL lists the break-number, break point conditions, break point state, and count.

The state of a break point is either enabled or disabled. If the break point is disabled, an asterisk (*) is displayed after its break-number. If an enabled break point was used in a BPAND command, an ampersand (&) is displayed after its break-number. The break point that most recently caused an action to occur is highlighted.

The BL command has no parameters.

Example:

```
  BL
```

This command displays all the break points that have been defined. A sample display, which shows four break points, follows:

```
    0) BPMB 1234:0000 W EQ 0010 C=03
    1) BPR B000:0000 B000:1000 W C=01
    2) BPIO 0021 W NE 00FF C=01
    3) BPINT 21 AH=4C C=01
```

Note that in this example, break point 1 is preceded with an asterisk (*), showing that it has been disabled.

BPE -- Edit break point

Syntax:

```
  BE break-number
```

Comments:

The BPE command loads the break point description into the edit line for modification. The command can then be edited using the editing keys, and re-entered by pressing the ENTER . This command offers a quick way to modify the parameters of an existing break point.

Example:

```
 BPE 1

  This command moves a description of break point 1
  into the edit line and removes break point 1. Pressing
  the ENTER key will cause the break point to be
  re-entered.
```

BPT -- Use break point as a template

Syntax:

```
  BT break-number
```

Comments:

The BPT command uses an existing break point description as a template for a new break point.

A description of the existing break point is loaded into the edit line. The break point referenced by break-number is not altered. This command offers a quick way to create a new break point that is similar to an existing break point.

Example:

```
 BPT 3

  This command moves a template of break point 3 into
```

the edit line. When the ENTER key is pressed, a new break point is added.

BC -- Clear break points

Syntax:

```
   BC list | *
   list -- A series of break-numbers separated
                    by commas or spaces
   *    -- Clears all break points
```

Comments:

The BC command is used to permanently delete one or more break points.

Example:

```
  BC *
```

This command clears all break points.

PAGE 74 IS EMPTY