CHAPTER 7

Symbolic and Source Level Debugging

7.1 Introduction
7.2 Preparing for Symbolic or Source Debugging
7.2.1 Preparing for Symbolic Debugging Only
7.2.2 Preparing for Symbolic and Source Level Debugging
7.3 Reserving Memory for Symbols and Source File
7.4 Loading Programs and Symbol Files
7.5 Debugging With Symbols
7.6 Debugging With Source

7.1 Introduction

Soft-ICE can load programs, symbol tables and source files for enhanced debugging. Symbolic debugging allows you to set break points and reference variables with symbol names rather than specifying numeric addresses. Source level debugging allows you to step through your program at the source code level rather than assembly code level.

Symbol and source line number information is extracted from the link map file. The link map must be compatible with Microsoft's linker version 3.60 or greater.

Symbols and source files reside in extended memory. You must have sufficient extended memory for the symbols and source files. Source files are not paged from the disk as in many debuggers. This allows Soft-ICE to provide complete system debugging in source level, You can debug T&SR's interrupt routines and other systems level code at the source level.

Note:
You cannot use symbolic or source level debugging unless Soft-ICE has been loaded as a device driver in CONFIG.SYS.

7.2 Preparing for Symbolic or Source Debugging

Before debugging a program with symbols or source you must create a symbol file. This is a binary file that contains symbol and line number information in a format that Soft-ICE can understand. This file is created with the utility MSYM.EXE. MSYM.EXE reads in your link map to create a symbol file with the extension (.SYM).

7.2.1 Preparing for Symbolic Debugging Only

To prepare a program for symbolic debugging only, you must do the following steps:

1. Compile or assemble your program.
2. Link your program with the proper switches to create a .MAP file that contains a list of public symbols. If you are using Microsoft's linker, the /MA switch is the proper switch to use. This .MAP file must be identical to the .MAP file produced by Microsoft's linker, version 3.60 or greater.
3. Create a.SYM file by running MSYM.EXE. The syntax for using MSYM.EXE is:
   MSYM program-name [.extension]

If the extension is not supplied MSYM assumes the extension is.MAP. MSYM reads in a map file as in and writes out a symbol file as output. The symbol has the name program-name.SYM.

Note:
Before compiling or assembling your program you may want to make some additional symbols public. Only public symbols are supported with Soft-ICE symbolic debugging. The way to make a variable or a label public varies, depending upon which language you are using.

In 8086 assembly language, simply use the PUBLIC directive followed by the locally defined symbols you wish to make public. For example:

```
PUBLIC FOO, LOOP1, STATUS
```

In C language, all procedure names and static variables are defined outside a block are public.

For other languages, refer to your language manual for details.

7.2.2 Preparing for Symbolic and Source Level Debugging

To prepare a program for both symbolic and source debugging, you must do the following steps:

1. Compile or assemble each module that you wish debug at the source level with the appropriate switch to put line number information into the object files. With Microsoft languages you can use either the /Zi or the /Zd switches. You may not want to do this with all files, because the combined file sizes of the symbol file and all the source files compiled with these switches must fit into the

amount of extended memory you have reserved with the /SYM loading switch in CONFIG.SYS.

2. Link your program with the proper switches to create a.MAP file that contains source line numbers and a list of public symbols. If you are using Microsoft's linker, the /LI and /MA switches are the proper switches to use. This .MAP file must be identical to the.MAP file produced by Microsoft's linker, version 3.60 or greater.
3. Create a.SYM file by running MSYM.EXE. The syntax for using MSYM.EXE is:
   MSYM program-name [.extension]

If the extension is not supplied MSYM assumes the extension is.MAP. MSYM reads in a map file as input and writes out a symbol file as output. The symbol file has the name program-name.SYM.

7.3 Reserving Memory for Symbols and Source Files

Before loading programs, symbol files and source files you must reserve extended memory for them. Extended memory is reserved when you load Soft-ICE in CONFIG.SYS. Before reserving extended memory you may want to add up the file sizes of the .SYM file and all of the source files that you want to load. You must reserve at least this much extended memory.
You must use the /SYM loading switch when loading S-ICE.EXE. A sample line in CONFIG.SYS for loading Soft-ICE and reserving space for symbols and source files is:

```
DEVICE = S-ICE.EXE /SYM 1024
```

This example loads Soft-ICE into extended memory and reserves 1 megabyte of memory for symbols and source files. See section 6.3 (Loading Soft-ICE as a Loadable Device Driver) for more details on reserving memory.

7.4 Loading Programs and Symbol Files

The Soft-ICE utility LDR.EXE is used for loading programs, symbol files and source files. For symbolically debugging application programs and T&SR programs you will typically use LDR.EXE to load the program, symbols and source files in one step. For debugging loadable device drivers, ROMs and other system components you will typically use LDR.EXE to load the symbol file and source files only.
The syntax for LDR.EXE is:

```
LDR program-name | program-name.SYM |
                        program-name.extension
```

7.4.1 Loading Program, Symbols and Source

To load your program, symbols and source files in one step, you must use LDR.EXE in the form:

```
LDR program-name
```

Notice that program-name does not have a file extension. If no file extension is supplied, then LDR.EXE will do the following:

1. Load program-name.SYM into extended memory
2. Load source files into extended memory. This step is done only if source records exist in the .SYM file.
3. Load program-name.EXE into memory at the location it would have loaded if it had been loaded directly from the DOS prompt.
4. Bring up Soft-ICE with the instruction pointer at first instruction of your program. If it is a C program and source is loaded for the file containing , _MAIN, then the source for that file will be visible in the code window.

7.4.2 Loading Only Symbols and Source Files

If you wish to load only symbols and source files (for debugging a loadable device driver for example) you must use LDR.EXE in the form:

```
LDR program-name.SYM
```

Notice that the.SYM extension is specified. This will load the .SYM file and source files into extended memory. When symbols are loaded by this method your program or device driver symbols are assumed to be referenced from 0:0. Since this is rarely the case you will need to use the Soft-ICE command SYMLOC to locate the symbols. See

the description of the SYMLOC command in section 5.10 for a complete description. An example of loading a symbol file called DRIVER.SYM is:

```
LDR DRIVER.SYM
```

7.4.3 Loading a Program With No Symbols or Source

To load a program file without loading the associated symbol file you must use LDR.EXE

in the form:

```
LDR program-name.extension
```

Notice that the file extension is present. Typically the file extension will be.EXE or.COM. When a file extension specified LDR.EXE will load the program and bring up Soft-ICE with the instruction pointer at the first instruction of the program. An example of loading a program with symbols and source is:

```
LDR TEST.EXE
```

Notes:
LDR.EXE saves a copy of the interrupt vector table automatically when it loads your program. This is equivalent to doing a VECS S command. If you are going to exit your program before it runs to completion, you can do an EXIT R to exit the program and restore the interrupt vector table.

Using LDR.EXE to load only the program-name.EXE is often useful for restarting your program while in the middle of a source level debugging session. To restart, the EXIT R command to abort the current session. Then use LDR.EXE to reload your.EXE file. The symbols: source do not have to be loaded since they remain in extended memory.

If LDR.EXE gives you the message "Out of space loading symbol information", this means that you did not reserve enough extended memory with the /SYM loading switch in CONFIG.SYS.

If LDR.EXE does not find your source files on the same directory as the program you are loading, LDR.EXE will prompt you for the path names where it can find the source files. If you have source files on several directories or are loading a program frequently this becomes cumbersome. You can eliminate the need for prompting by using the DOS environment variable SRC. LDR.EXE uses this environment variable to find source files before prompting the user. The syntax for setting the environment variable from the DOS prompt is:

```
SET SRC = directory;directory;...;directory
```

Each of the specified directories will be searched before the user is prompted.

Limitations:
Soft-ICE supports symbols for only one program at a time. If you load a new .SYM file, the existing one is overwritten.
Soft-ICE does not follow overlays or Microsoft Windows segment movement.
Soft-ICE recognizes public symbols and line numbers only. It does not support local variables.

7.5 Debugging With Symbols

After you have loaded your program and.SYM file you can begin debugging your program symbolically. In general a symbol can be used in any command in place of an address.

Symbols are also used by several Soft-ICE commands when addresses are displayed. For example, the U command displays symbol names of labels and procedures as it encounters them.
There are two commands that are helpful when you are symbolically debugging:

* SYM -- Use the SYM command to get a listing of symbol names and values, or to change the value a symbol.
* SYMLOC -- Use the SYMLOC command to relocate the base of all of your symbols. You would need to use the SYMLOC command when:
  1. Loading symbols for a loadable device driver
  2. Loading symbols for a T&SR that has already been loaded
  3. Your program moves itself to a location other than its original location.

See section 5. 10 for a complete description of these commands.

7.6 Debugging With Source

When source files are loaded, Soft-ICE allows you to view and step through your source code as you are debugging. Soft-ICE offers two different modes of source level debugging: mixed mode and source mode. Use the SRC command to switch between modes.
Mixed mode shows source lines and the assembly language produced by those source lines intermixed on the display. Mixed mode is useful when you must debug at the assembly level, but use the source lines for reference. Mixed mode is allowed whether the code window visible or not.

Source mode strictly shows source lines on the display. Source level debugging requires the code window to be visible.

7.6.1 Using Line Numbers

Line numbers can be used in place of addresses in several commands. To differentiate a line number from an actual address, place a . (period) in front of the number. For

example, to set an execution break point at source line 45 type:

```
BPX .450
```

7.6.2 Using Source Mode in the Code Window

The code window must be visible to enter source mode. If not visible, use the WC command to make it visible. Once you are in source mode you can use Soft-ICE commands switch to a different source file, view source at any location in the file, scroll through the file, search for strings in the file, and set break points in the file. For a complete description of the following commands see their command descriptions in chapters 4 and 5. The following list is a brief overview of commands that are useful when debugging source code:

* Make the code window visible (if it is not already) with the WC command.
* Toggle between source, mixed, and code modes with the SRC command. To toggle modes enter:

```
SRC
186
```

* Place a source file in the code window (if it is n@ already) with the FILE command. For example change from the current file to file MAIN.C enter:

```
FILE MAIN.C
```

* Display source at a specific location within the source file with the U command. To change the view to a specific line number or memory address use the U command. You can specify actual addresses or line numbers as a parameter to the command. For example, to view source in the code window starting at source line 450 enter:

```
U .450
```

* Locate the current instruction in the code wind@ with the . (period) command.
* Search for a specific character string with the S@ command. For example, to search for the string "Hello World" starting at line 100 in the current source file enter:

```
SS 100 "Hello World"
```

* Move the cursor to the code window (if it is not already) with the EC command.
* Scroll the source with the keys up, down, PaqeUp, PageDn.
* Set point-and-shoot break points with the BPX command. Simply place the cursor on the source line that you wish to break on, then enter:

BPX

Page 188 is blank