Hitachi America, Ltd. TN-0150
Application Engineering
TechNote
Jennifer Ediyanto

Running S-records Using GDB

The H8/300 GNU Debugger (GDB) can debug both COFF and Motorola S-records format files produced by the GNU C Compiler or Linker. The GDB can connect to the H8/300 Series Evaluation Board and the Simulator. This paper will provide an example in debugging the S-records format absolute file using the GDB Evaluation Board interface.

There is a demo program called **Flash.c** and a linker command file called **Flash.ld** on the distribution disk. The demo program and the linker command file can be found in the H8300 directory under the DEMO sub-directory.

Compiling the Demo Program

Copy the Flash.c and Flash.ld files in the DEMO directory to the BIN directory. We need to compile the Flash.c program to produce the s-records absolute file. The following command line will compile and produce the absolute file called **Flash.x**:

```
C:\h8300\bin>  GCC  -o  Flash.x  -O -g  -Xlinker  -M  -Tflash.ld  Flash.c  >  Flash.map
```

where

GCC	name of the compiler
-o	compiler switch that specifies output file name
-O	compiler switch that optimizes the source code
-g	compiler switch to include the debugging information
-Xlinker	compiler switch that passes all switches after -Xlinker to the linker
-M	linker switch to generate the linker map file
>	redirect the output map file to a user-defined file, i.e., Flash.map
-T	linker switch that directs the linker to read commands from Flash.ld file

The generated s-records file, Flash.x, does not contain information for the program starting address. We can determine the program starting address from the generated map file, Flash.map. The starting address is the address for **_main** (i.e., **0x8020**) in the Flash.map file.

Debugging the Demo Program

The following are DOS mode commands to initiate between the H8/300 series Eval Board and COM2:

```
C:\h8300\bin>  mode  com2:9600,n,8,1,p      <enter>
com2:9600,n,8,1,p
C:\h8300\bin>  asynctsr  2     <enter>
asynctsr installed
```

We can invoke the GDB Debugger by calling the GDB name. The GDB will prompt us with the prompt **(GDB)**. The following commands start and run the GDB:

```
C:\h8300\bin>  GDB
(gdb)     target  hms  com2
Connected to remote H8/300 HMS system
(gdb)     load  flash.x
.text:    0x8000 .. 0xf814 *******************
```

```
(gdb)   set  $pc=0x8020
(gdb)   disassem 0x8020 0x8030
Dump of assembler code from 0x8020 to 0x8030:
(gdb)   b *0x8050
Breakpoint 1 at 0x8050
(gdb)   c
Continuing.
Breakpoint 1, 0x8050 in ?? ()
(gdb)   si
Single stepping the program
(gdb)   <enter>
Continue single stepping the program
(gdb)   set $pc=0x8020
Reset the program
(gdb)   clear *0x8050
Deleted breakpoint 1
(gdb)   c
continuing.
```

At this point, we can see the lights on the Eval Board are flashing one after another. To stop the program, press the **RESET** button on the Eval Board. The RESET button will:

- interrupt our program
- return to the GDB command prompt after our program finishes normally. The communication protocol provides no other way for GDB to detect program completion.

We can use the **HELP** command at any time in the GDB to find out more about GDB commands. To end the GDB, type the **QUIT** command. We use some of the GDB commands in this tutorial, which are:

• target hms com#	specify cross-debugging to the Hitachi Eval Board
• load	download the program to the board
• set $pc=*addr*	set the program counter equal to the specified address, that is the starting address of the program (i.e., _main address).
• c	execute the program
• b **addr*	set breakpoint at the address
• clear **addr*	clear breakpoint at the address
• si	single step
• disassem *addr addr*	disassemble a block of memory