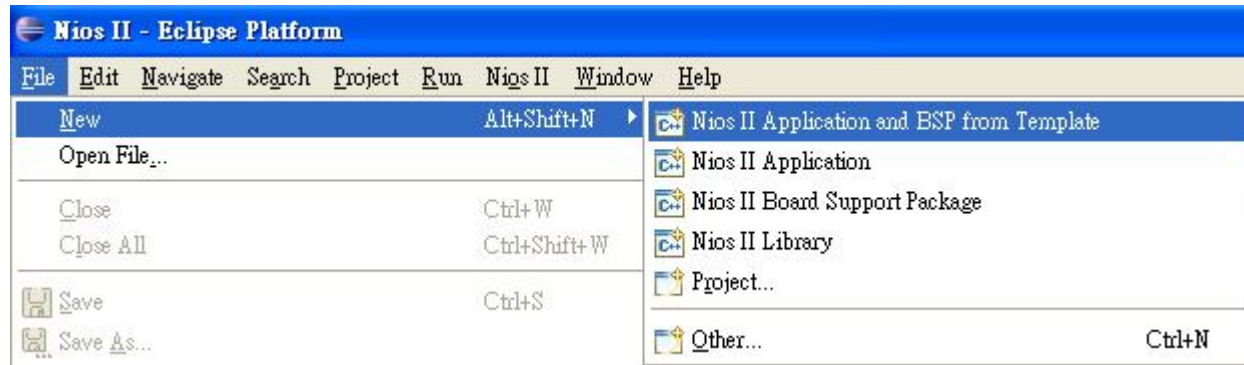


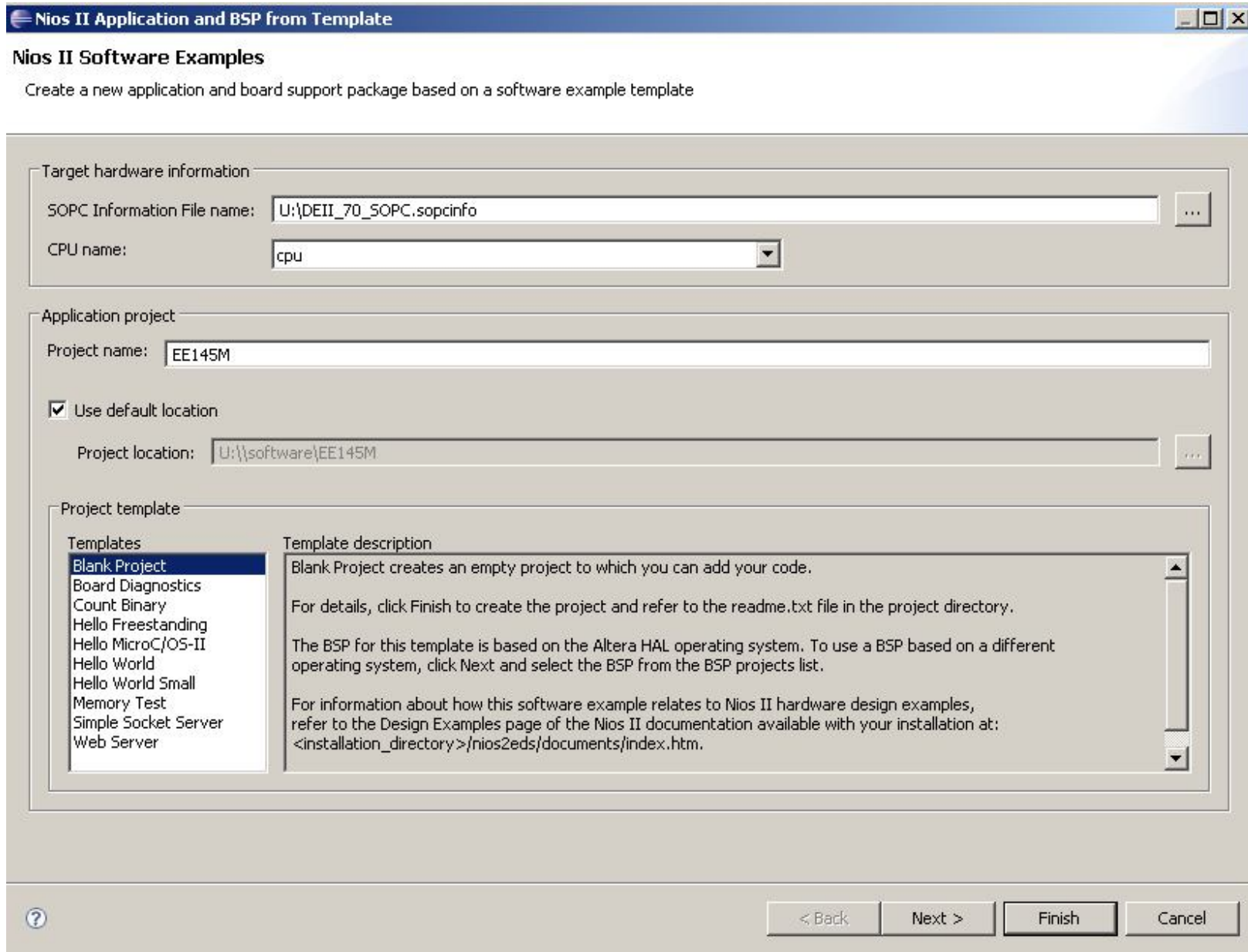
Creating a new Project

1. To create a new project, on Toolbar, click File -> New -> Nios II Application and BSP from Template

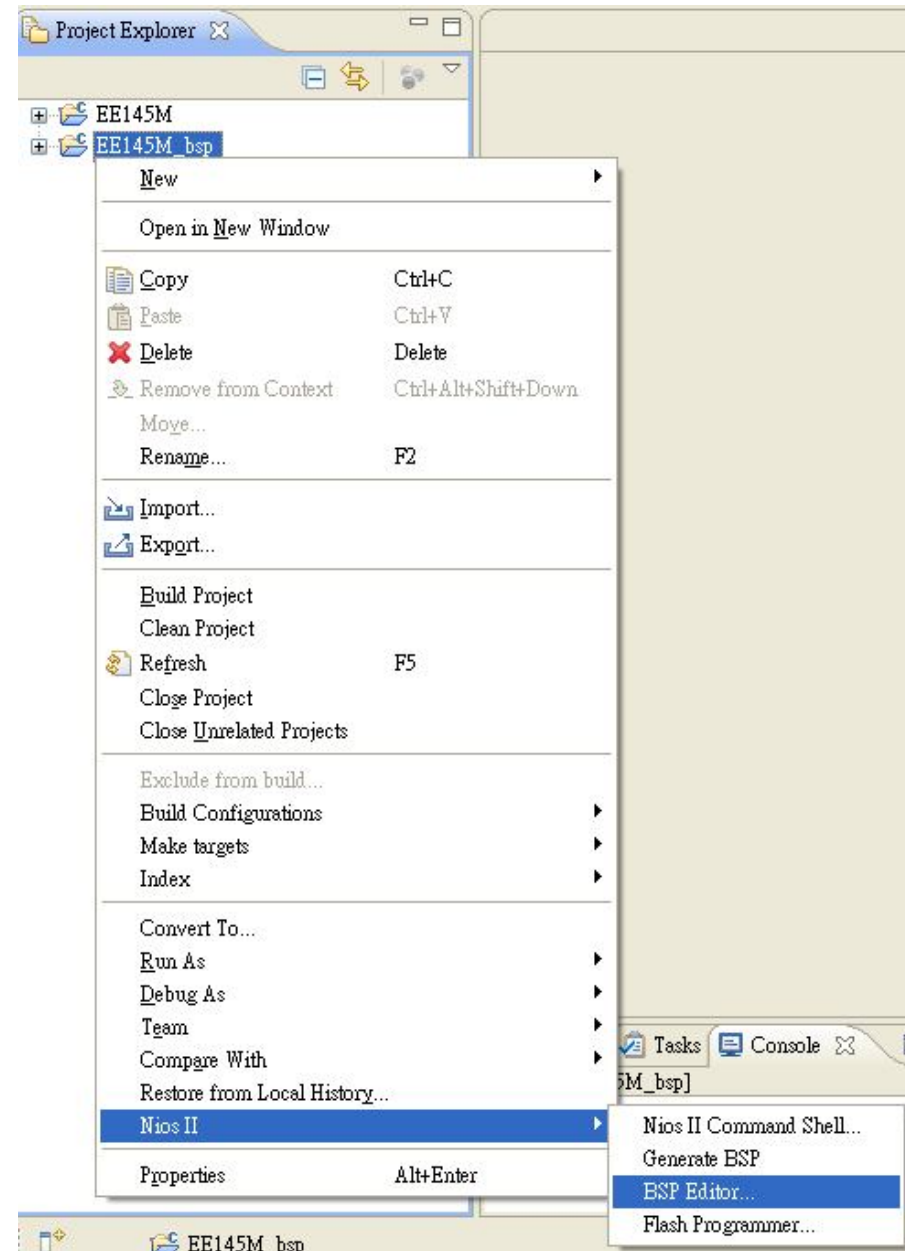


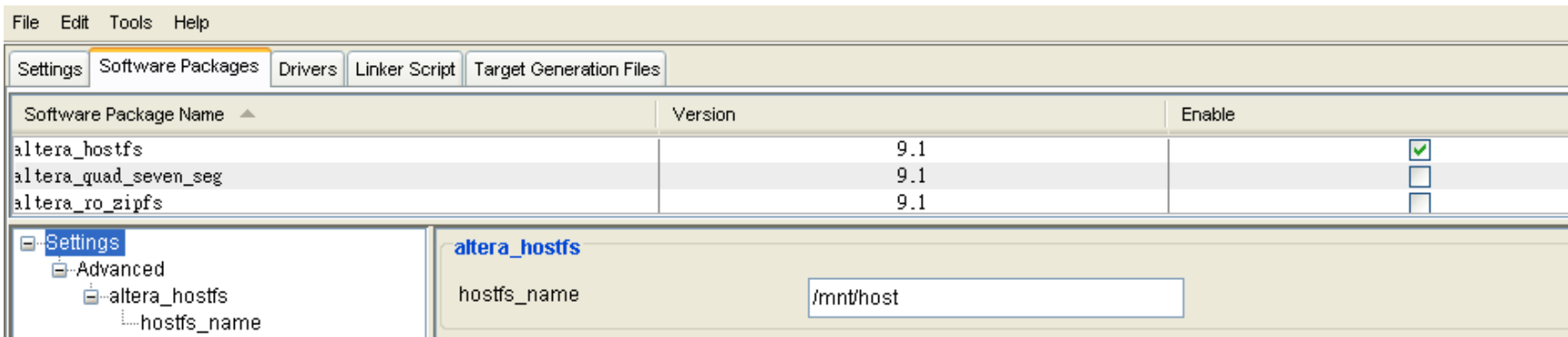
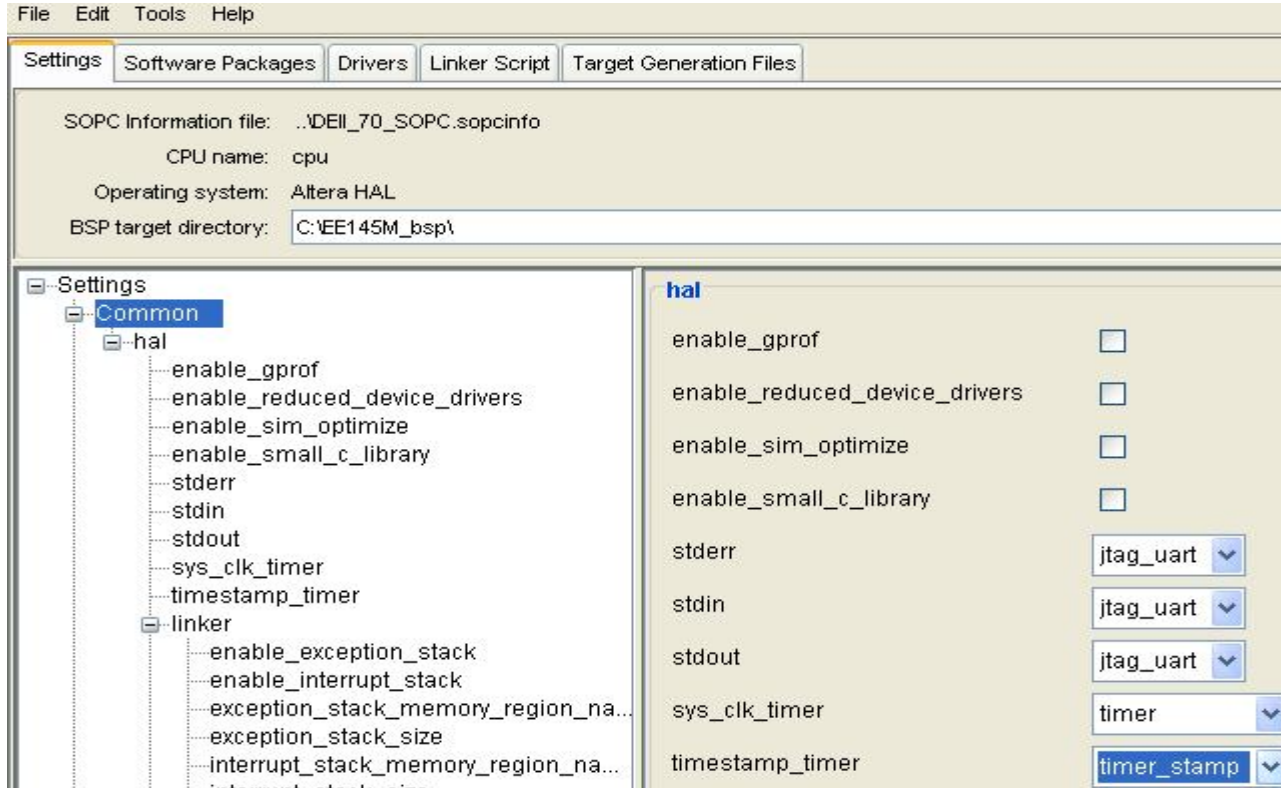
2. On the window that pops up, in the "SOPC Information File Name" field, browse and choose the *DEII_70_SOPC.sopcinfo* we provided.
3. For the "Project Name" field, type EE145M (or anything you desire).
4. Next, uncheck use default location and choose a folder in your U:/ drive
5. In "Project Template", choose Blank Project and press "Finish"

Note: Both your SOPC information file and the project location should be in U:/ drive, and the path must not contain spaces.



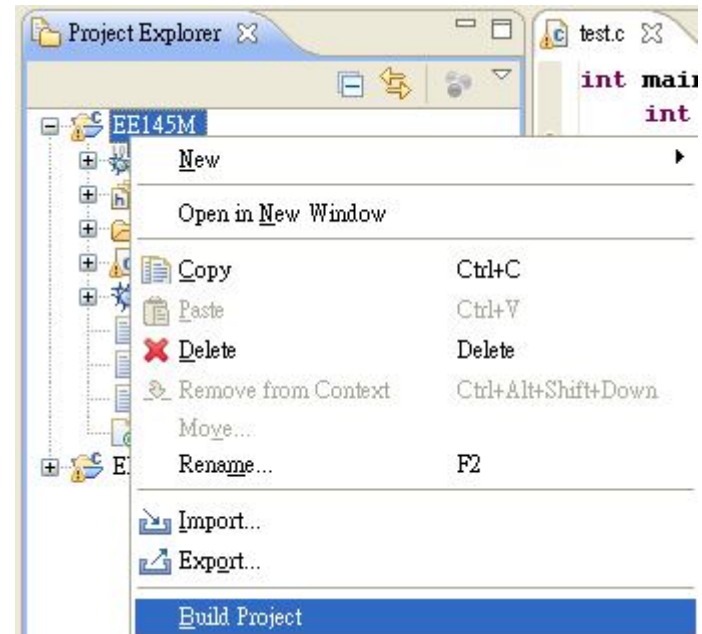
6. After the project is created, in the project explorer menu, right click on the EE145M_bsp folder, choose Nios II and BSP Editor.
7. Under **Setting->Common**, for the “timestamp_timer” field, choose *timer_stamp*
8. Under **Software Packages**, check “altera_hostfs” to enable writing file to computer
Note: To enable writing file to computer, you must be running the program in debug mode. See *Writing Collected Data to File* Section for more information.
9. Save the BSP and press **Generate** on the lower right corner.
10. Wait for it to finish and close the window.
11. You are now ready to start your first program.





Building Project

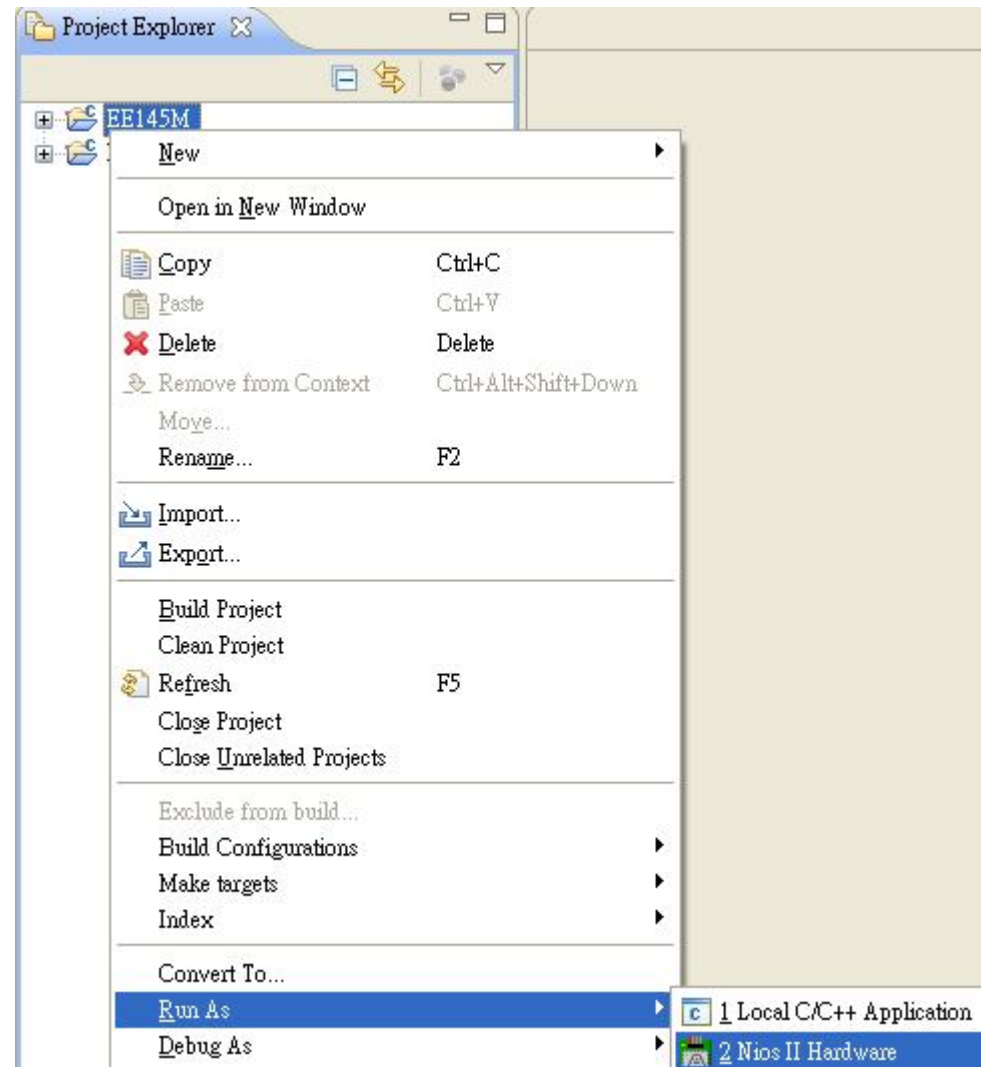
1. When you are finish writing your program, you must build the project before you can run it. First, under Project Explorer, right click on the project folder (EE145M)
2. Choose "Build Project".
3. When the build is successfully finished, the console will inform you that build complete by displaying [EE145M build complete], as shown below.
4. If build unsuccessful, refer to the error and fix your program.



```
Problems Tasks Properties Console
C-Build [EE145M]
Info: Linking EE145M.elf
nios2-elf-g++ -T'../EE145M_bsp/linker.x' -msys-crt0='../EE145M_bsp/obj/HAL/src/crt0.o' -msys-lib=hal_bsp -L../EE145M
-mcustom-fpu-cfg=60-2 -Wl,-Map=EE145M.map -OO -g -Wall -mhw-div -mcustom-fpu-cfg=60-2 -mhw-mul -mno-hw-mulx -o
EE145M.elf obj/test.o -lm
nios2-elf-insert EE145M.elf --cpu_name cpu --simulation_enabled false --id 1851523791 --sidp 0xc411f8 --timestamp 124
--stderr_dev jtag_uart --stdin_dev jtag_uart --stdout_dev jtag_uart --sopc_system_name DEII_70_SOPC --quartus_project
Info: (EE145M.elf) 76 KBytes program size (code + initialized data).
Info: 32692 KBytes free for stack + heap.
Info: Creating EE145M.objdump
nios2-elf-objdump --disassemble --syms --all-header --source EE145M.elf >EE145M.objdump
[EE145M build complete]
```

Download and Run Program

1. Under Project Explorer, right click on the project folder (EE145M)
2. Choose Run As -> Nios II Hardware



Debugging Program

Setting Breakpoints

By default, the program will have a break point at the main function, alongside with a watch for all variables. However there may be cases where you want to add additional breakpoint.

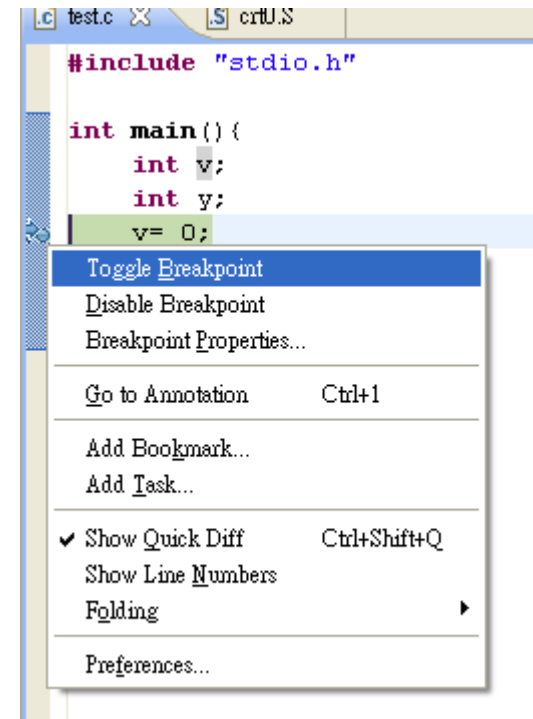
1. To enable breakpoint at a certain line of code, right click on the bar next to the line of code of interest, select "Toggle Breakpoint"
2. To disable breakpoint, right click on the breakpoint and select "Disable Breakpoint"

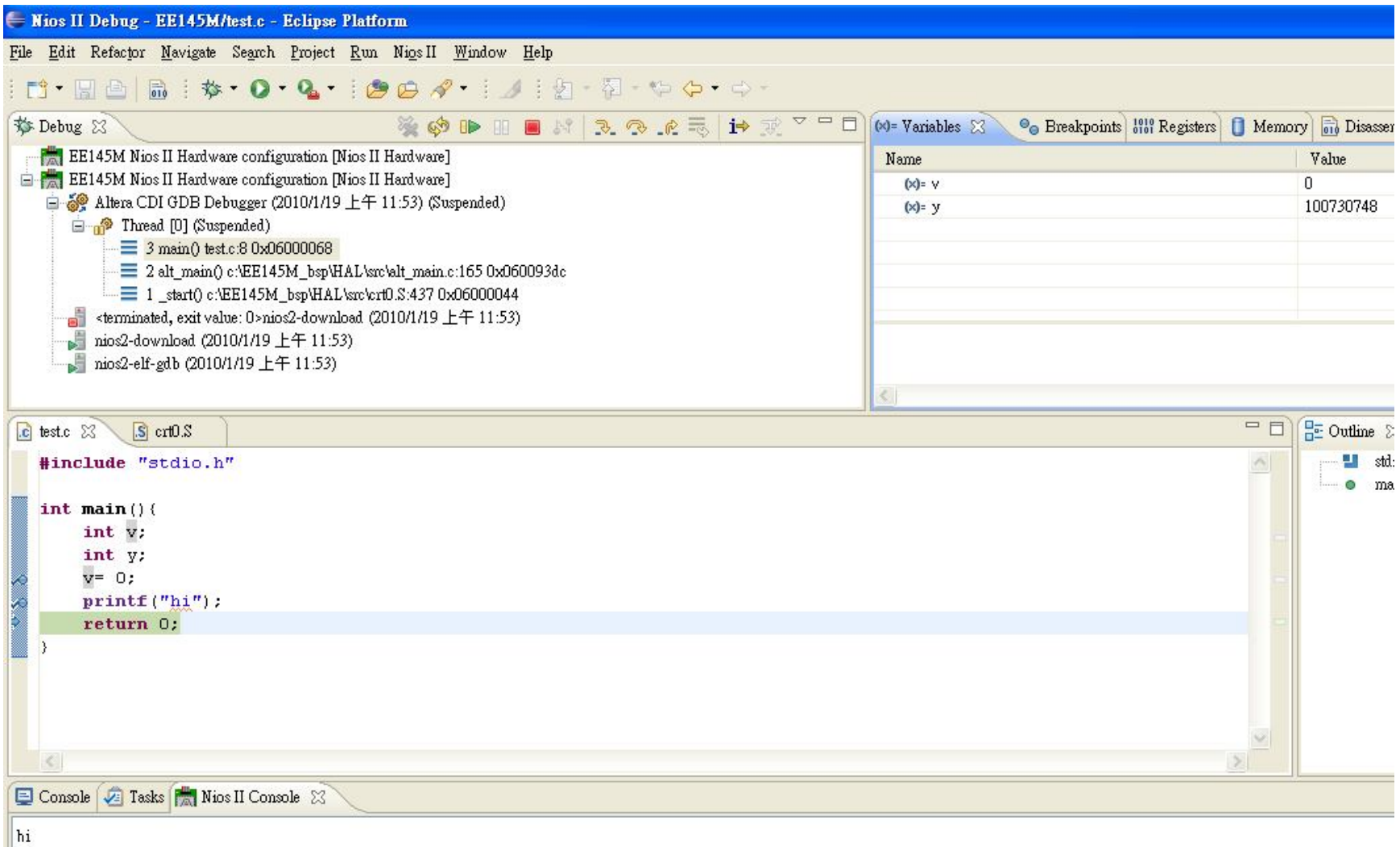
Run Program in Debug Mode

1. Under Project Explorer, right click on the project folder (EE145M)
2. Choose Debug As -> Nios II Hardware
3. This should open up the "Debug Perspective"

Stepping Through Program

1. By default, the program will stop at the first line of main().
2. The current line you are at is highlighted by a light blue color.
3. To run the next line of code, click "Run -> Step Over" (F6)
4. To go into the code of function call in the next line of code, click "Run -> Step Into" (F5)
5. If you want to continue running the program, click "Run -> Resume" (F8)
6. To stop the program at any time, click "Run -> Terminate" (Ctrl + F2)
7. On the upper right window, you can monitor the value of your variable and memory, and enable/disable any breakpoint





Writing Collected Data to File

If you follow the instruction in **Creating a new Project** section, your system can write file back to the computer.

1. First, you must open a file handle as usual in other C program:

```
    fopen(<file name>, <open operation>);
```

2. The file name should be preceded by the mount-point, which is /mnt/host/

Ex: To open a file with name test.txt for writing operation:

```
    fopen("/mnt/host/test.txt", "w");
```

3. To write information to the file, use *fprintf*(<file handle>, <format string>, <variables>...)
4. To read information from file, use *fscanf*(<file handle>, <format string>, <pointer to variables>....)
5. Once you are finish writing the file, use *fclose*(<file handle>) to close the file.

Note: This only work if you run the program in debug mode.

Example:

```
FILE* result = fopen("/mnt/host/data/result.txt", "w");           //open result.txt in directory data in write mode
FILE* data = fopen("/mnt/host/data/testData.txt", "r");          //open testData.txt in directory data in reading mode
int val = 0;
fscanf(data, "%d", &val);                                       //read the next integer from testData.txt and store in variable val
fprintf(result, "%d", val*val);                                  //write the square of val into result.txt
fclose(data);
fclose(result);
```

Using LCD Display

1. Include API file relate to LCD Display

```
#include "De2_70_media_PC_145M.h"
```

2. Open a file handler to the LCD Display

```
FILE* lcdHandle = fopen (LCD_NAME, "w")
```

3. Clearing the LCD Display

```
fprintf(lcdHandle, "%s",VT100ClearScreen);
```

4. Writing string to the LDC Display

```
fprintf(lcdHandle,<formatting string>,<arguments>);
```

Note: The LCD Display can only display 2 lines of text at max. Please clear out the display before displaying more text

Example:

```
Int x = 100;
```

```
FILE* lcdHandle = fopen (LCD_NAME, "w")
```

```
fprintf(lcdHandle, "%s",VT100ClearScreen);
```

```
fprintf(lcdHandle,"Testing %s",x);
```

```
fclose (lcdHandle);
```

```
//open a file handle to LCD display
```

```
//clear the screen
```

```
//Display Testing 100 on the LCD display
```

```
//close the file handle
```