

## Tutorial Material for NIOS II SBT for Eclipse

Robert Li

1. Before proceeding with this tutorial, please get the Quartus Prime project containing the NIOS II components ready and download it to the DE1-SoC board.
2. Launch NIOS II SBT for Eclipse by clicking the shortcut icon on the desktop, or by selecting “NIOS II Software Build Tools for Eclipse” under the Tools menu from Quartus, and specify a directory for your workspace (the workspace folder and the path to this folder must not contain spaces, otherwise NIOS II SBT for Eclipse will not work). See Figure 1.

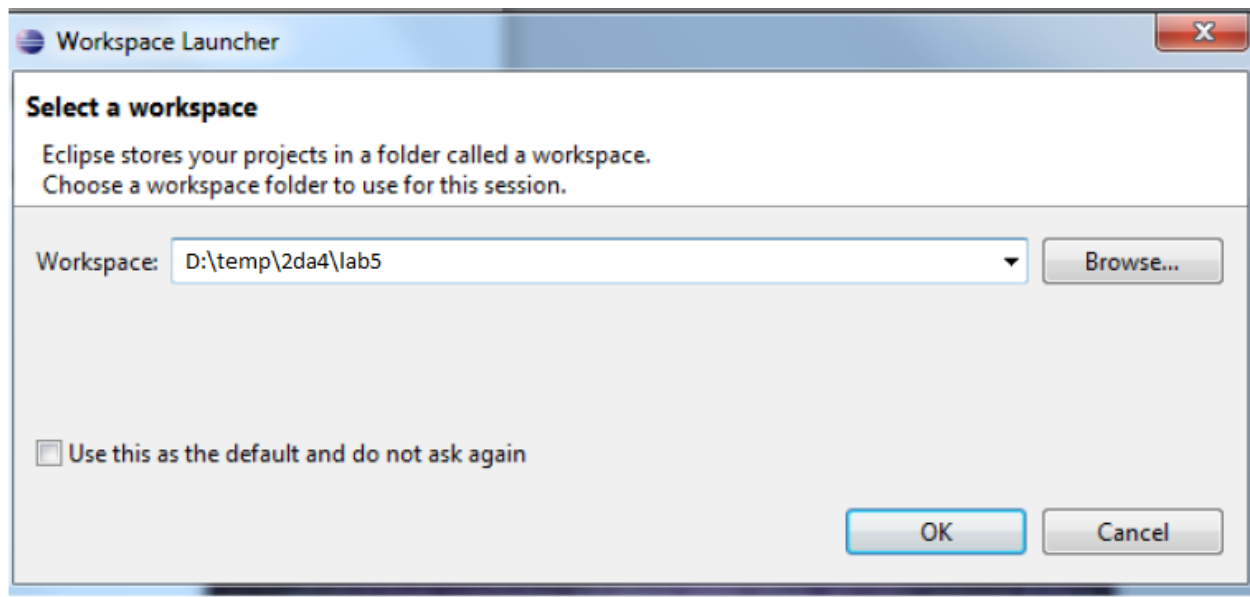


Figure 1. Workspace launcher

3. Create a new project by choosing: File | New | Nios II Application and BSP from Template. See Figure 2.
4. In the wizard window for creating the new project, specify the name of the SOPC information file, or by click the “...” button to browse the Quartus project folder and select the appropriate .sopcinfo file, then specify a project name. For the project location, it is better to use the default value, otherwise, the system may have difficulty creating the project. Select the Hello World template from the Project Template box, then click Finish. (Figure 3.)

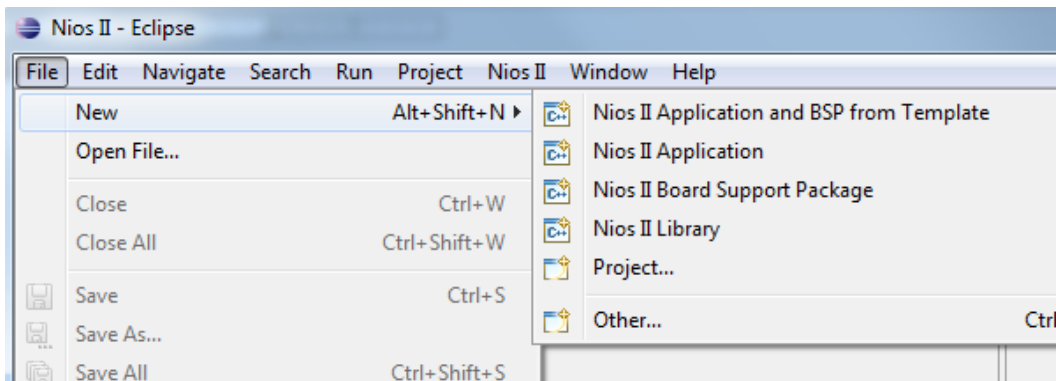


Figure 2. Create a new project

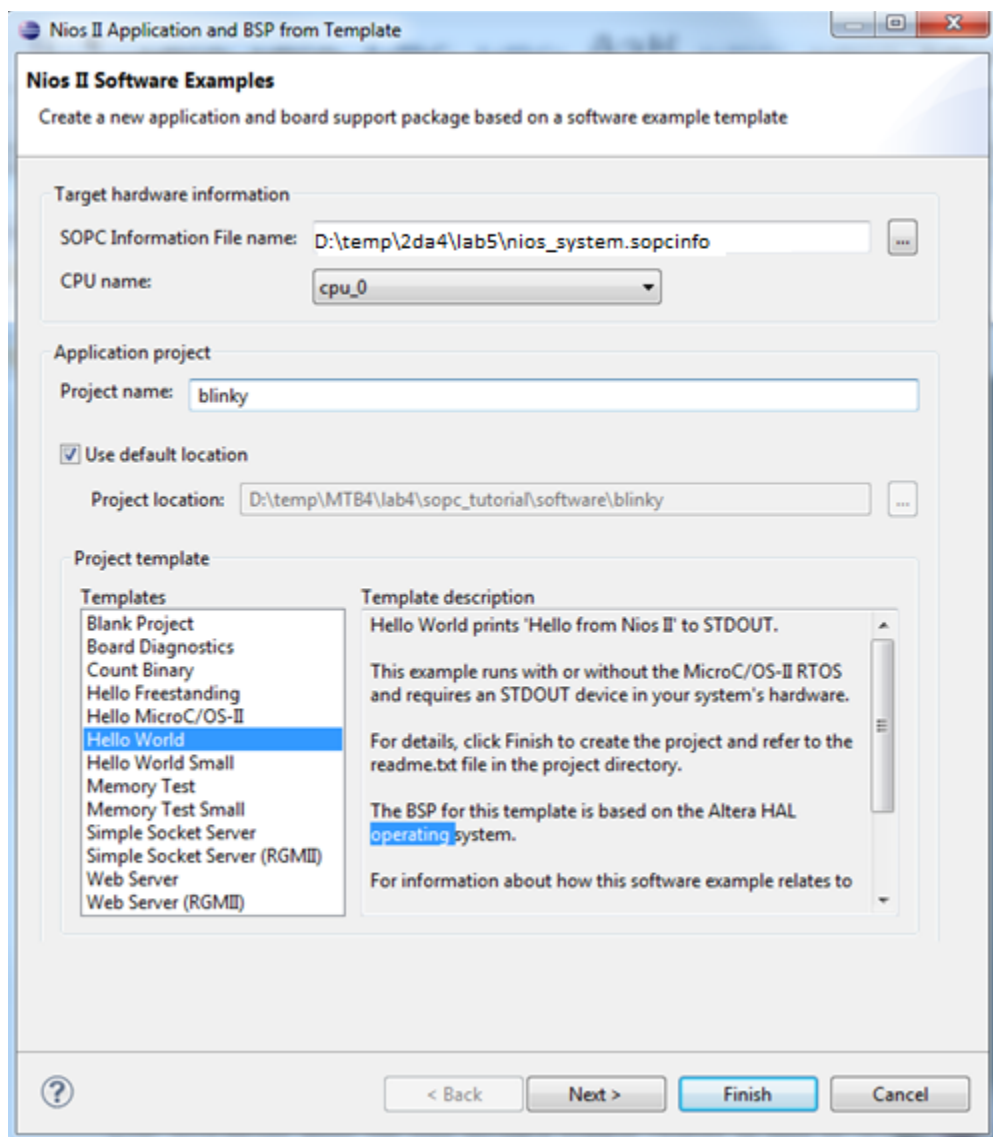


Figure 3. Wizard window for creating a new project

5. Nios II SBT for Eclipse automatically generates a project. The project has two folders or two sub projects. In our example, the two folders are `blinky` and `blinky_bsp` in the Project Explorer window. A `hello_world.c` file is in the `blinky` folder. Another important header file, `system.h`, is in the `blinky_bsp` folder (Figure 4).

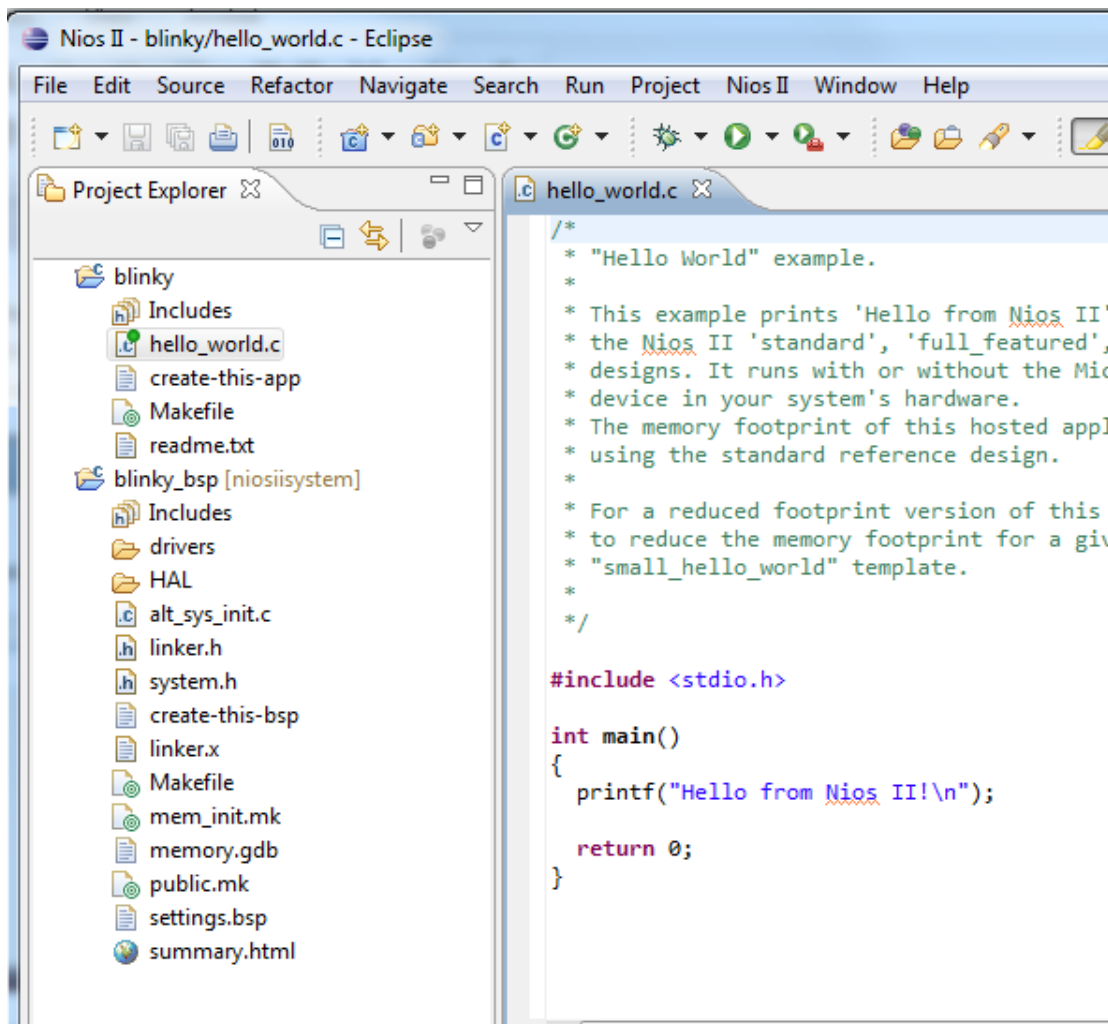


Figure 4. Project Explorer

6. Build the `blinky` project, or run the project. Building the project can be accomplished either by right clicking the `blinky` project in the Project Explorer window and then selecting the Build Project command in the context menu, or by selecting the Build Project command from the Project menu. Running the project can be accomplished either by right clicking the `blinky`

project in the Project Explorer window and then selecting the Run As | NIOS II Hardware command in the context menu, or by selecting the blinky project first in the Project Explorer window, and then selecting the Run As | NIOS II Hardware command from the Run menu. The Run As | NIOS II Hardware command also builds the project if it has not been built before.

7. At this point, the project may not be built successfully **if the on-chip memory is not large enough** (for example, if less than 32K). The printf() function actually takes up 64KB since it requires all the libraries from stdio.h. (See Figure 5)

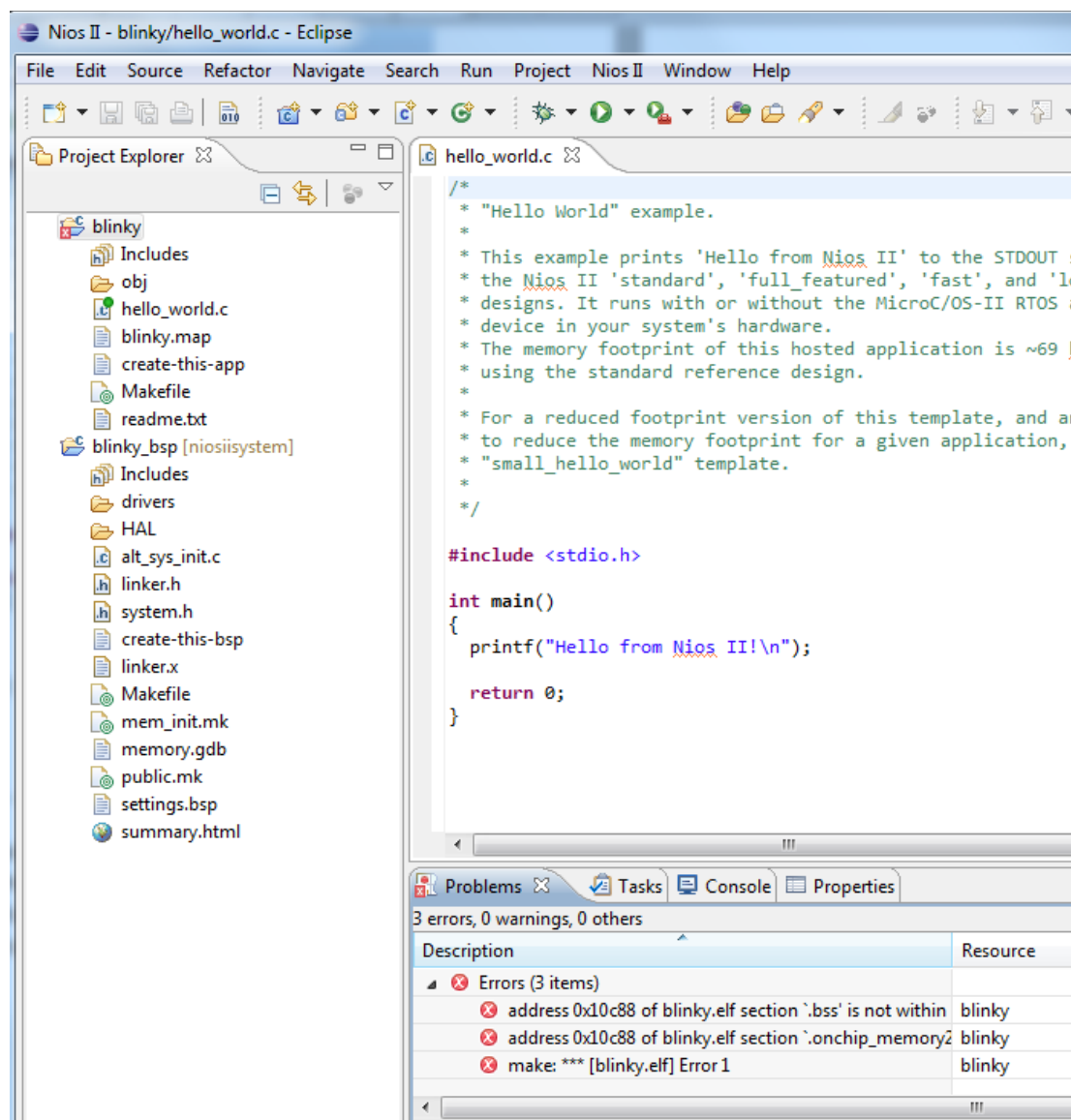


Figure 5. Project may not be built because of printf()

In case the project cannot be built because of the function `printf()`, we either can remove the two lines involving `stdio.h` and `printf()`, if we do not need to use `printf()`, or we can increase the size of the on-chip memory (for example, increasing to 128K for DE1-SoC boards) by modifying the SOPC system. **In case we need to use the `printf()` function and do not have large enough on-chip memory, we can change settings to let the NIOS II SBT use the small C library. Details for this will be given later (in step 11).**

8. Since we do not actually need the `printf()` function in this tutorial, we can remove the two lines involving `"stdio.h"` and `"printf()"`, and start a brand new main function.

In our tutorial, we intend to blink the LEDs. In order to do this, we need to know the address of the LEDs on the Avalon interconnect.

From the Project Explorer, navigate to `blinky_bsp>system.h`. We will find everything we need about our system in this file. A defined constant `LEDS_BASE` is the base location of the LEDs we are trying to toggle. We can use `LEDS_BASE` just like any other pointer in C. **NOTE:** you will need to cast it to an appropriate 8-bit data type to ensure proper functionality. The `hello_world.c` file should now look like Figure 6 below.

```
12  * to reduce the memory footprint for a given
13  * "small_hello_world" template.
14  *
15  */
16
17  #include "system.h"
18  #include <unistd.h>
19
20  int main()
21  {
22      char * LEDs=(char *)LEDS_BASE;
23
24      while (1) {
25          *LEDs=0xFF;
26          usleep(1000*1000);
27          *LEDs=0x00;
28          usleep(1000*1000);
29      }
30      return 0;
31  }
32
```

Figure 6. C program to blink LEDs

9. Build this project and run the project through “Run As | NIOS II Hardware”, as described in Step 6.
10. If the Run Configuration window appears after trying to run the application as NIOS II Hardware, most likely it is due to a target connection problem. Open the Target Connection tab in the Run Configuration window, click the Refresh Connection button one or more times until the USB connection is shown under the cable section, then click the Apply button and then the Run button. If the Refresh Connection button cannot be seen, make the Run Configuration window larger. The button is at the right edge of the window (Figure 7).

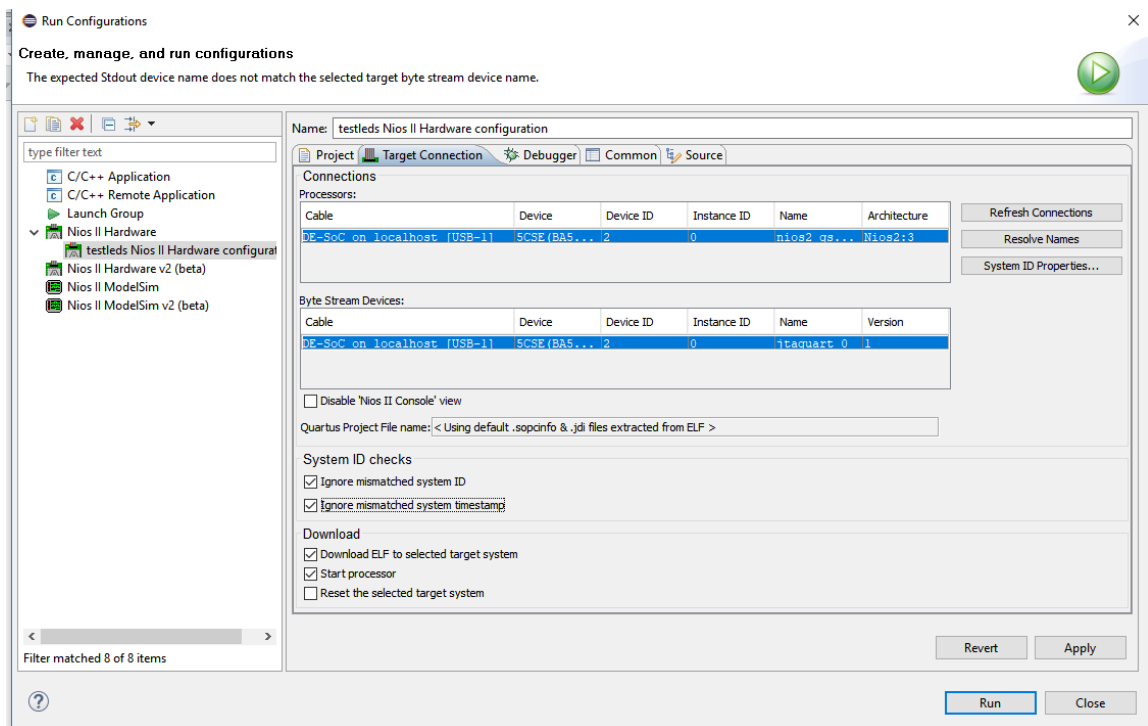


Figure 7. Run Configuration window

11. In case the `printf()` function is needed in the project and there is not enough on-chip memory (DE1-SoC boards have enough on-chip memory), we can change settings to let the NIOS II SBT use the small C library to reduce the code footprint. The setting can be changed by following these steps:

- 1) Right click the project in the Project Explorer window and select “NIOS II | BSP Editor...” in the context menu.
- 2) If a window opens asking to run the BSP Editor in the background, let the BSP Editor run in the background.
- 3) In the BSP Editor window, select the Main tab. In the left side of the window, select Settings, at the top part of the right side of the window, uncheck options: a) enable\_c\_plus\_plus, b) enable\_clean\_exit, c) enable\_exit. Also check options: a) enable\_lightweight\_device\_driver\_api, b) enable\_reduced\_device\_drivers, c) enable\_small\_c\_library, and leave everything else unchanged (Figure 8).
- 4) Then press the Generate button at the bottom of the BSP Editor and then Exit. Re-build the project to use the printf() function.

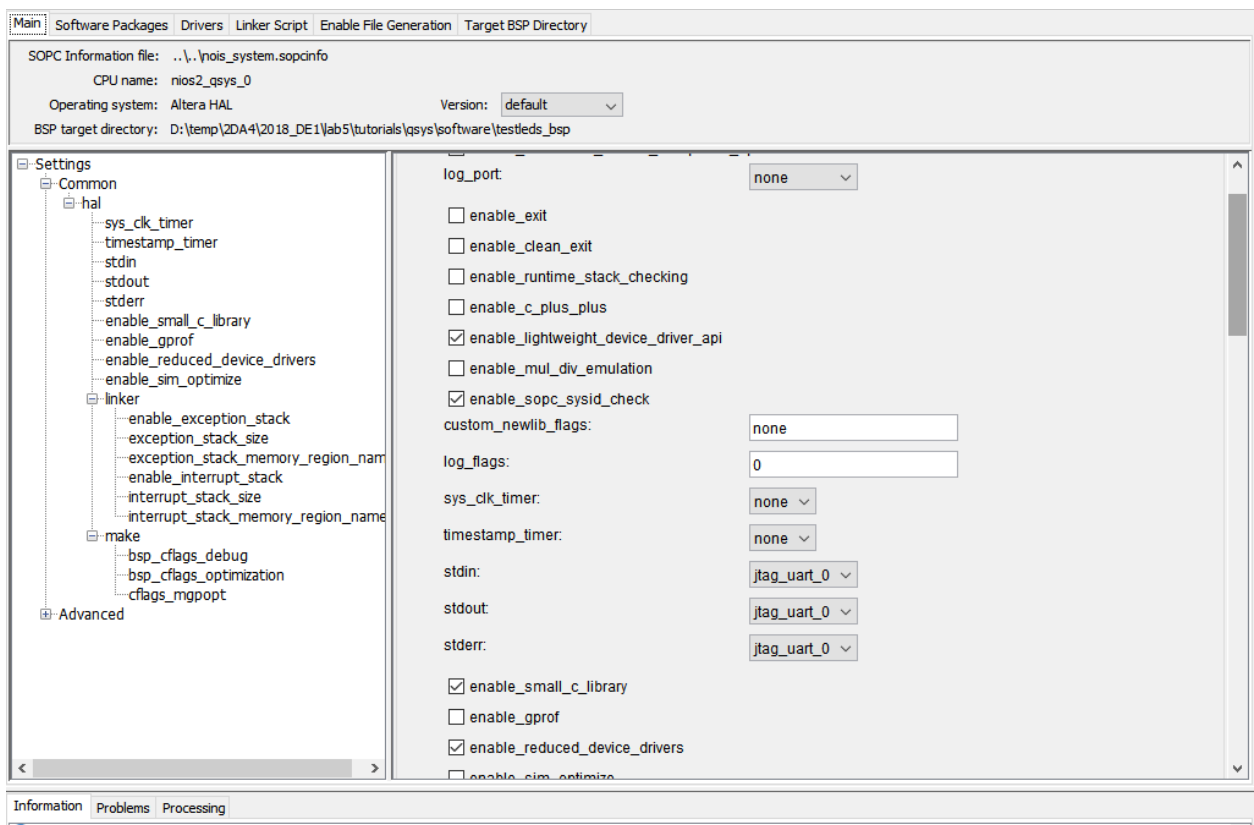


Figure 8. Settings to make NIOS II SBT for Eclipse use the small C library

12. In the project development process, you may need to modify the NIOS II system or Quartus project, and re-compile the Quartus project and download it to the DE1-SoC board. **Each time** after a new version of the Quartus project is downloaded to the DE1-SoC board, you need to re-generate the BSP library for NIOS SBT to make your NIOS SBT project work properly . To re-generate the BSP library, in the NIOS SBT's Project Explorer window, right click the \_bsp folder (or the \_bsp sub project), then select NIOS II | Generate BSP.