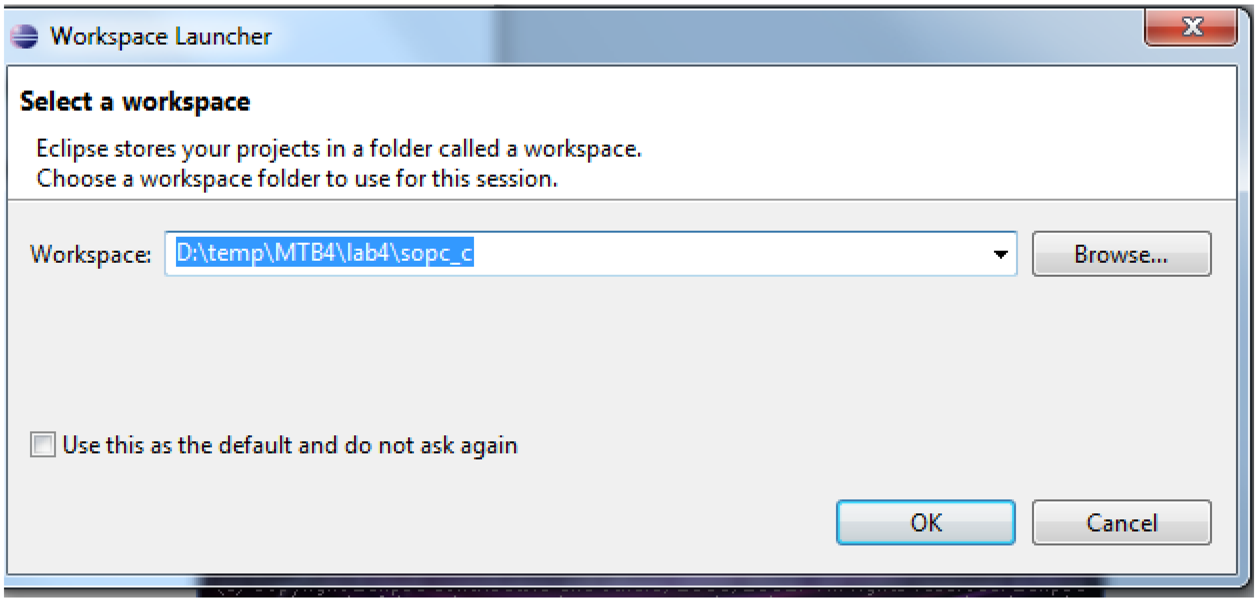
Tutorial Material for NIOS II SBT for Eclipse

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.



Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: D:\temp\MTB4\lab4\sopc_c

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 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).

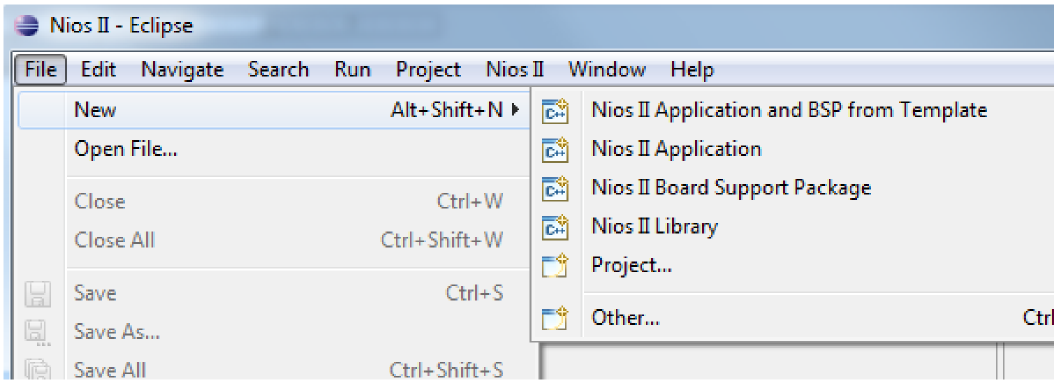
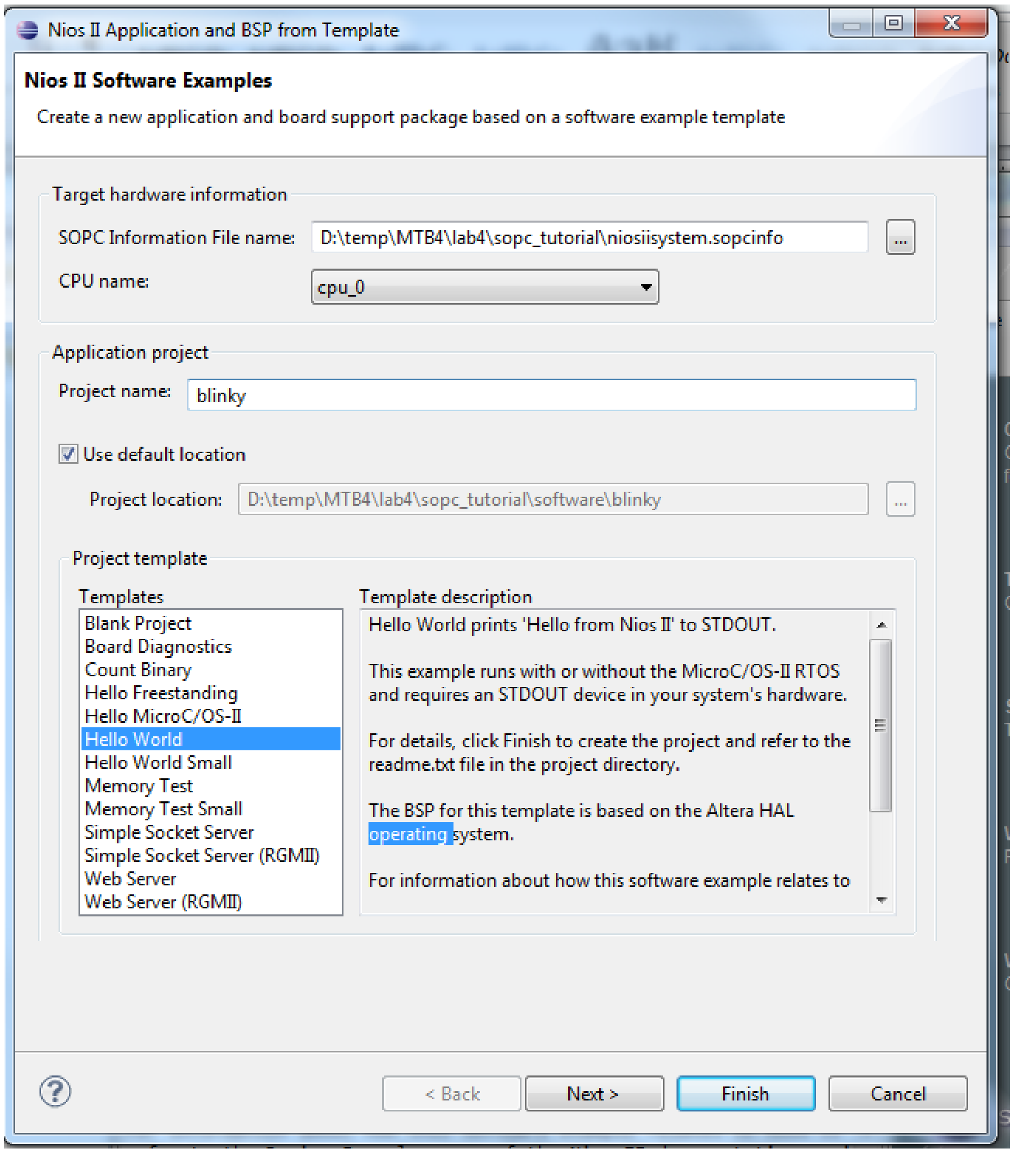


Figure 2. Create a new project



Nios II Software Examples

Create a new application and board support package based on a software example template

Target hardware information

SOPC Information File name: D:\temp\MTB4\lab4\sopc_tutorial\niosiisystem.sopcinfo

CPU name: cpu_0

Application project

Project name: blinky

[✓] Use default location

Project location: D:\temp\MTB4\lab4\sopc_tutorial\software\blinky

Project template

Templates:
- Blank Project
- Board Diagnostics
- Count Binary
- Hello Freestanding
- Hello MicroC/OS-II
- Hello World [✓]
- Hello World Small
- Memory Test
- Memory Test Small
- Simple Socket Server
- Simple Socket Server (RGMII)
- Web Server
- Web Server (RGMII)

Template description:

Hello World prints 'Hello from Nios II' to STDOUT.

This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware.

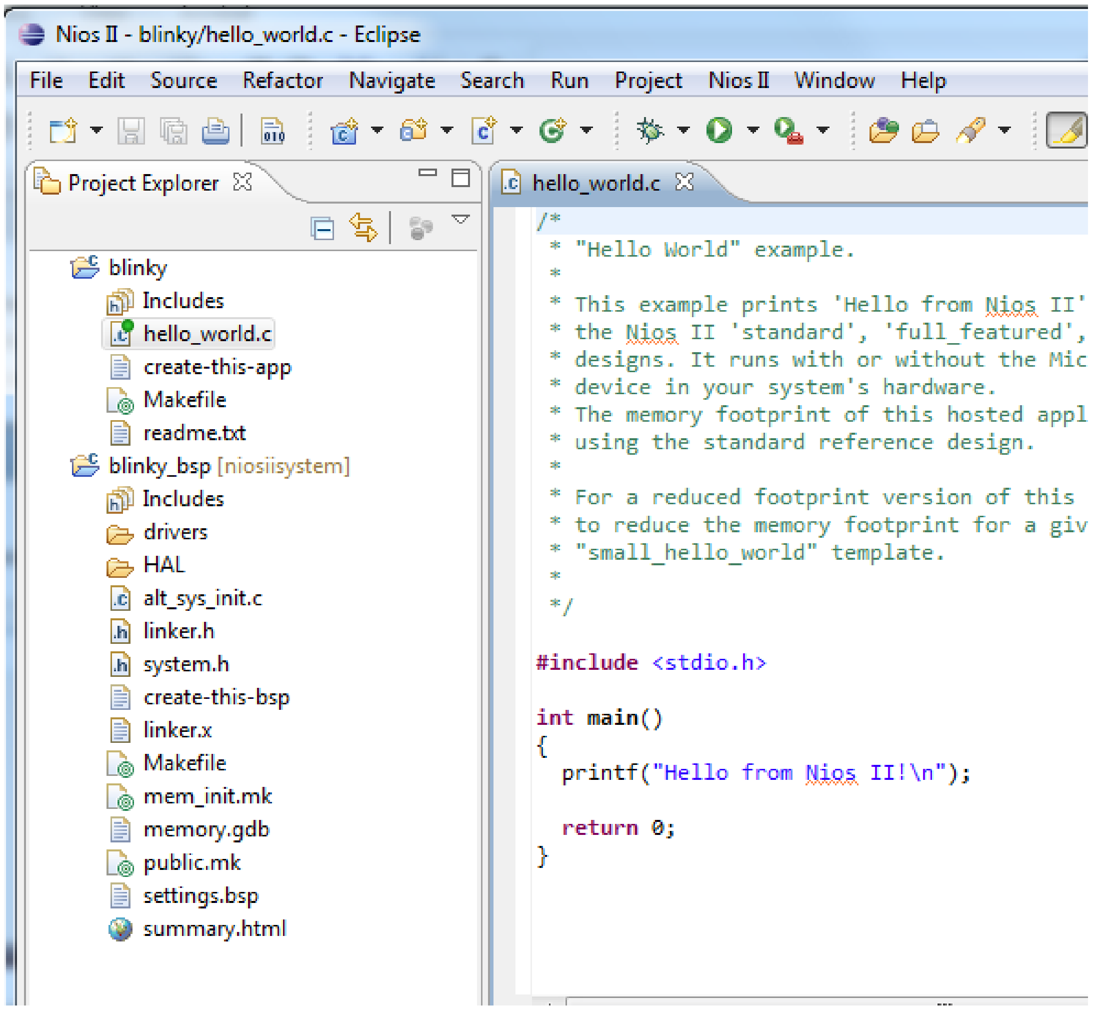
For details, click Finish to create the project and refer to the readme.txt file in the project directory.

The BSP for this template is based on the Altera HAL operating system.

For information about how this software example relates to

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).

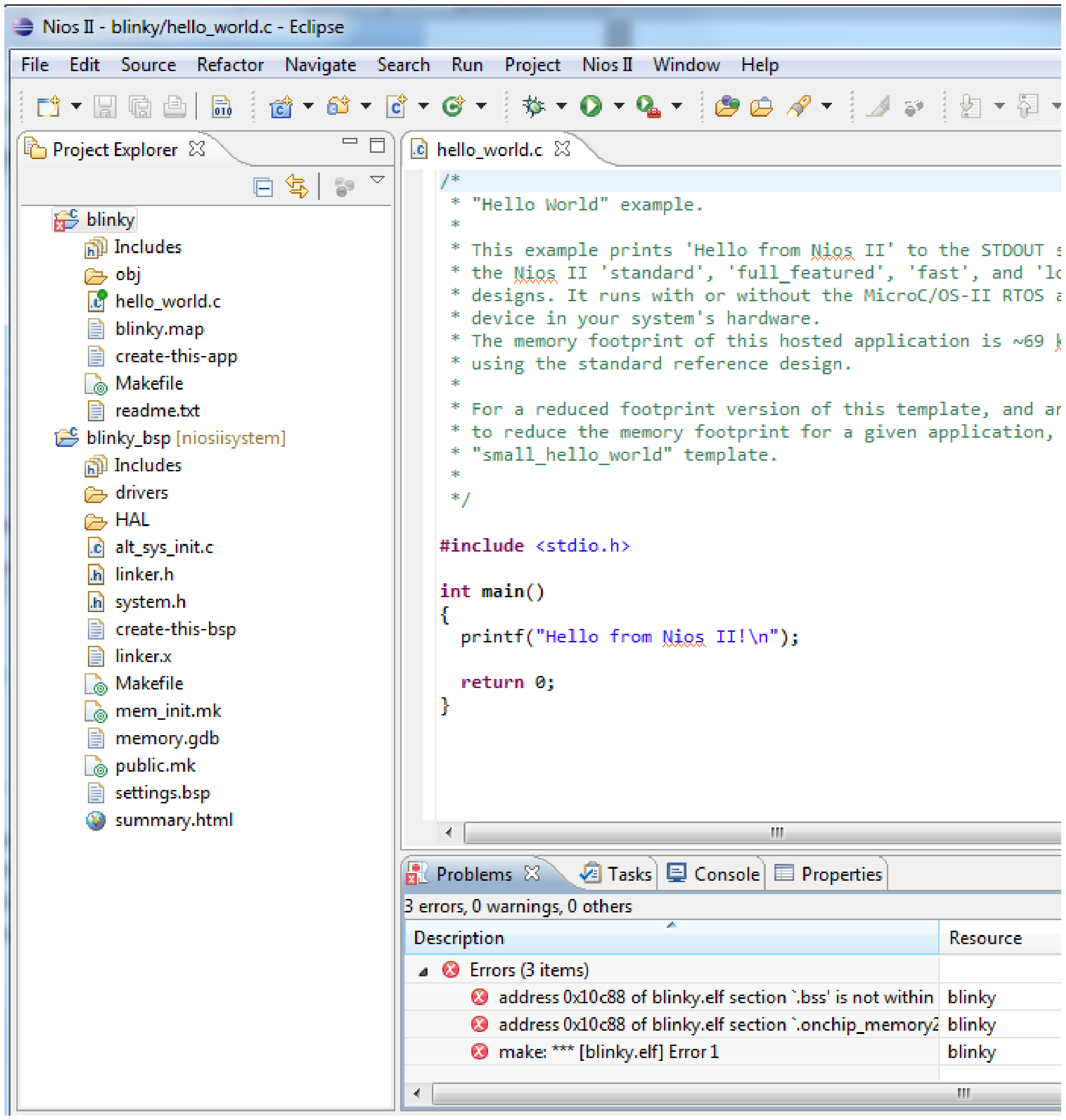


```
/*
 * "Hello World" example.
 *
 * This example prints 'Hello from Nios II'
 * the Nios II 'standard', 'full_featured',
 * designs. It runs with or without the Mic
 * device in your system's hardware.
 * The memory footprint of this hosted appl
 * using the standard reference design.
 *
 * For a reduced footprint version of this
 * to reduce the memory footprint for a giv
 * "small_hello_world" template.
 *
 */

#include <stdio.h>

int main()
{
  printf("Hello from Nios II!\n");

  return 0;
}
```

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.



```
/*
 * "Hello World" example.
 *
 * This example prints 'Hello from Nios II' to the STDOUT s
 * the Nios II 'standard', 'full_featured', 'fast', and 'lo
 * designs. It runs with or without the MicroC/OS-II RTOS a
 * device in your system's hardware.
 * The memory footprint of this hosted application is ~69 k
 * using the standard reference design.
 *
 * For a reduced footprint version of this template, and ar
 * to reduce the memory footprint for a given application,
 * "small_hello_world" template.
 *
 */

#include <stdio.h>

int main()
{
  printf("Hello from Nios II!\n");

  return 0;
}
```

3 errors, 0 warnings, 0 others

Description	Resource
Errors (3 items)	
address 0x10c88 of blinky.elf section `.bss' is not within	blinky
address 0x10c88 of blinky.elf section `.onchip_memory2	blinky
make: *** [blinky.elf] Error 1	blinky

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

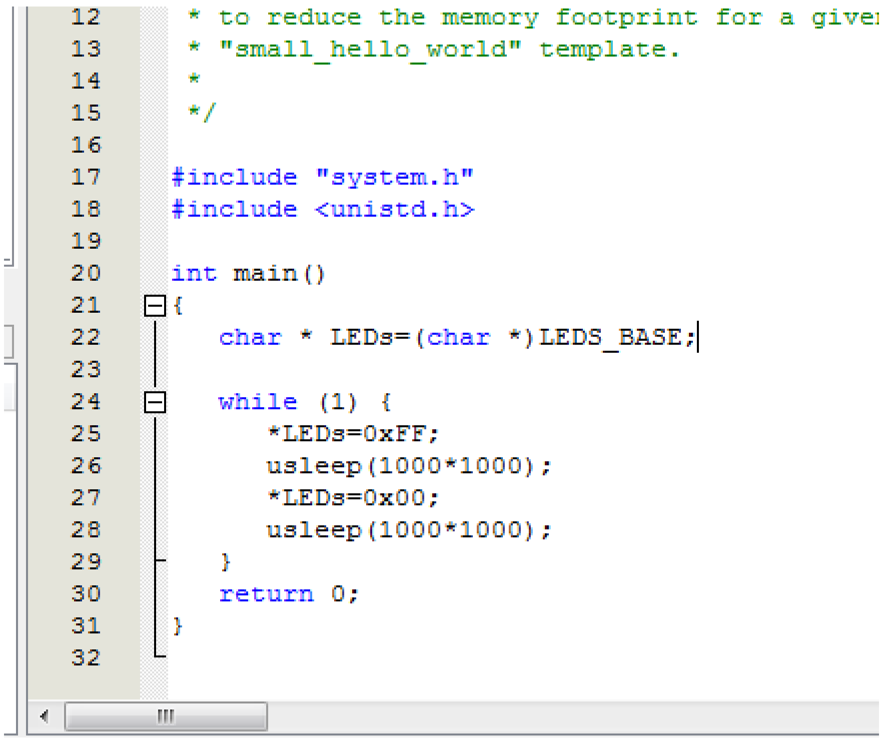
7. At this point, the project may not be built successfully **if the on-chip memory is not large enough** (for example, if less than 32K). The printf() function actually takes up 64KB since it requires all the libraries from stdio.h. (See Figure 5)

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.

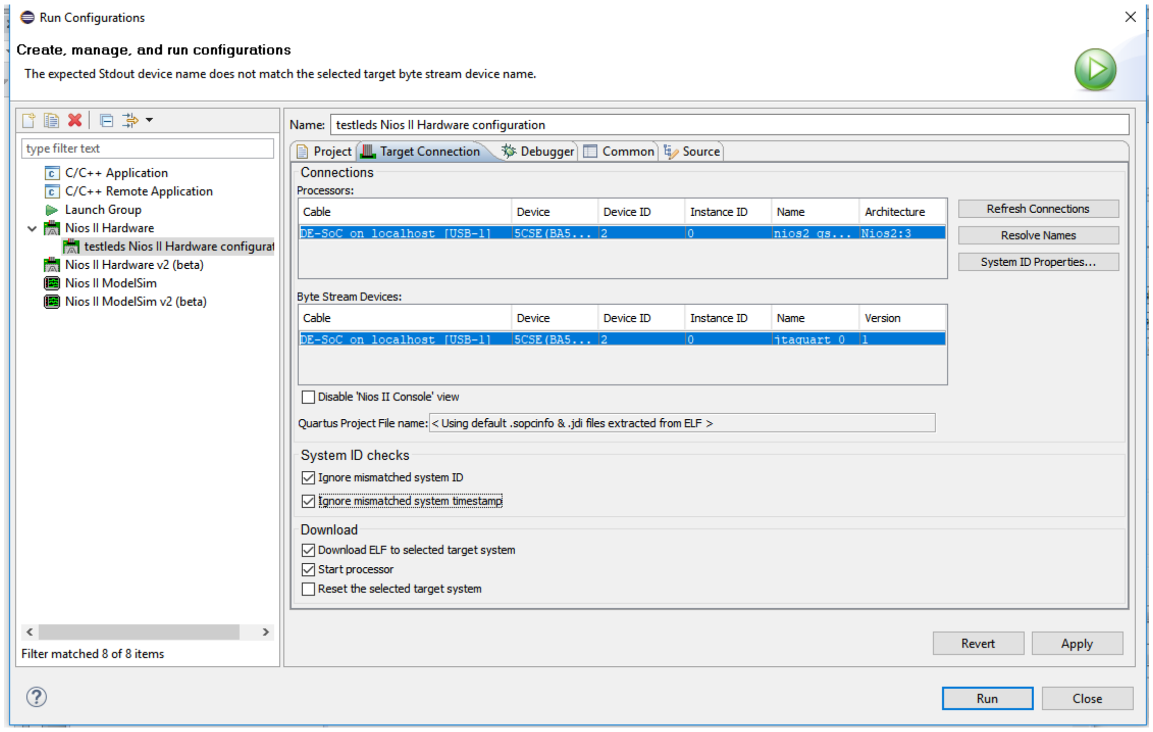


```
 * to reduce the memory footprint for a give
 * "small_hello_world" template.
 *
 */

#include "system.h"
#include <unistd.h>

int main()
{
    char * LEDs=(char *)LEDS_BASE;

    while (1) {
        *LEDs=0xFF;
        usleep(1000*1000);
        *LEDs=0x00;
        usleep(1000*1000);
    }
    return 0;
}
```

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).



The expected Stdout device name does not match the selected target byte stream device name.

Name: testleds Nios II Hardware configuration

Processors:

Cable	Device	Device ID	Instance ID	Name	Architecture
DE-SoC on localhost [USB-1]	5CSE(BA5...	2	0	nios2 qs...	Nios2:3

Byte Stream Devices:

Cable	Device	Device ID	Instance ID	Name	Version
DE-SoC on localhost [USB-1]	5CSE(BA5...	2	0	jtaguart 0	1

☐ Disable 'Nios II Console' view

Quartus Project File name: < Using default .sopcinfo & .jdi files extracted from ELF >

System ID checks

☑ Ignore mismatched system ID

☑ Ignore mismatched system timestamp

Download

☑ Download ELF to selected target system

☑ Start processor

☐ Reset the selected target system

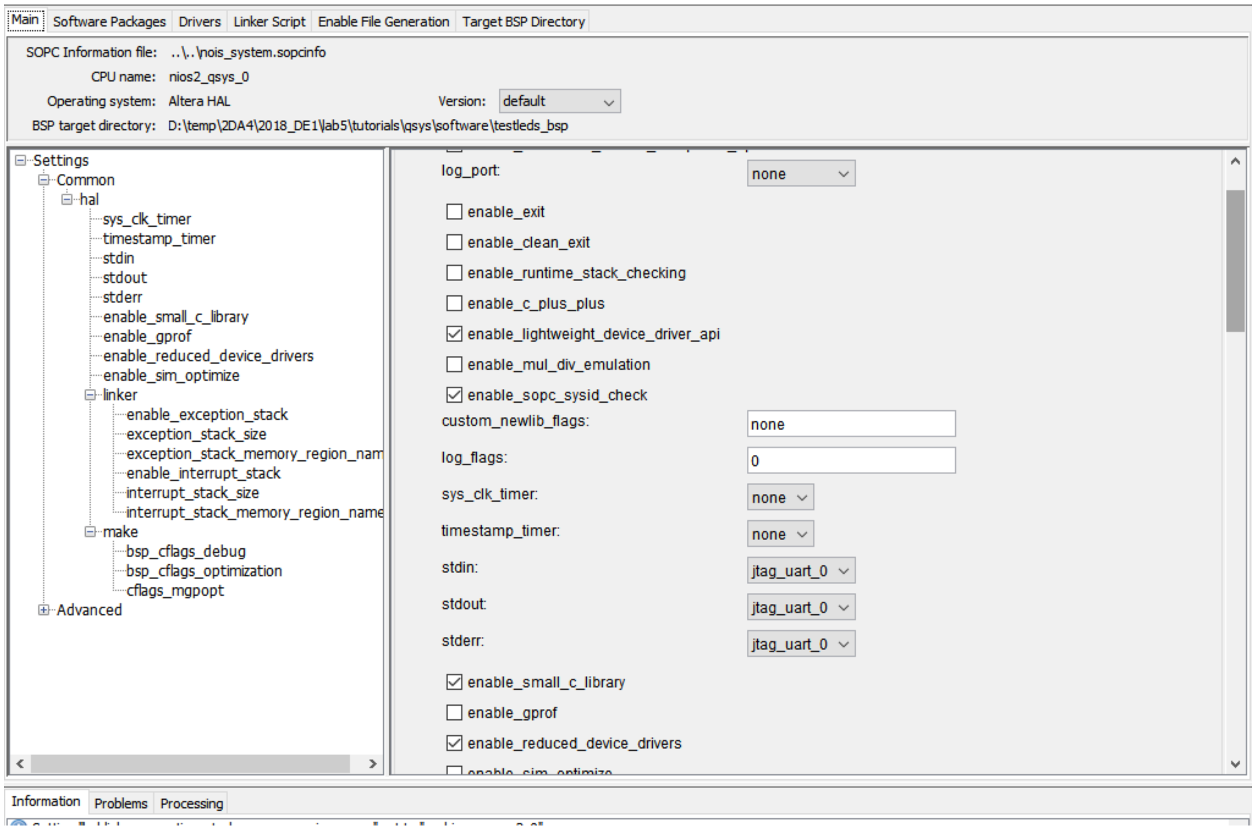
Figure 7. Run Configuration window

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.



SOPC Information file: ..\..\nois_system.sopcinfo
CPU name: nios2_qsys_0
Operating system: Altera HAL   Version: default
BSP target directory: D:\temp\2DA4\2018_DE1\lab5\tutorials\qsys\software\testleds_bsp

```
Settings
  Common
    hal
      sys_clk_timer
      timestamp_timer
      stdin
      stdout
      stderr
      enable_small_c_library
      enable_gprof
      enable_reduced_device_drivers
      enable_sim_optimize
      linker
        enable_exception_stack
        exception_stack_size
        exception_stack_memory_region_nam
        enable_interrupt_stack
        interrupt_stack_size
        interrupt_stack_memory_region_name
      make
        bsp_cflags_debug
        bsp_cflags_optimization
        cflags_mgpopt
  Advanced
```

log_port: none

☐ enable_exit
☐ enable_clean_exit
☐ enable_runtime_stack_checking
☐ enable_c_plus_plus
☑ enable_lightweight_device_driver_api
☐ enable_mul_div_emulation
☑ enable_sopc_sysid_check

custom_newlib_flags: none
log_flags: 0
sys_clk_timer: none
timestamp_timer: none
stdin: jtag_uart_0
stdout: jtag_uart_0
stderr: jtag_uart_0

☑ enable_small_c_library
☐ enable_gprof
☑ enable_reduced_device_drivers

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.