California State University, Fresno
Lyles College of Engineering
Electrical and Computer Engineering Department

TECHNICAL REPORT

Assignment: Number 5
Experiment Title: HW/SW Co-Design of an Embedded System on FPGA
Course Title: ECE 178 (Embedded Systems)
Instructor: Dr. Reza Raeisi

Prepared by: Anthony Herrick
Student ID #300144976
Date Submitted: 10/25/2022

INSTRUCTOR SECTION

Comments: ____

Final Grade: ____________________

TABLE OF CONTENTS

Objective 4
Hardware Requirements 4
Software Requirements 4
Background 4
Project Overview 6
Project Procedure 7
Conclusion 26
Appendix A (Part 1 Code) 27
Appendix B (Part 2 Code) 29

LIST OF FIGURES

1. 7-Segment Display Address Spaces 5
2. 7-Segment Display Breakdown 6
3. Quartus Prime Programmer 7
4. Hardware Setup 8
5. Selecting Device 9
6. Adding the File to be Downloaded onto the Board 10
7. Completed Download onto the Board 11
8. NIOS II Application and BSP from Template 12
9. NIOS II Application and BSP From Template Settings 13
10. Hello_world.c Program 14
11. Project Files 15
12. Building the project 16
13. BSP File Properties 17
14. NIOS II BSP Properties 18
15. BSP Editor 19
16. Software Packages Settings 20
17. Drivers Settings 21
18. Linker Script Settings 22
19. Running As Nios II Hardware 23
20. Setting the Target Connection 24
21. Output from Hello_World 25

Objective

Upon the completion of this lab, one will have co-designed an embedded soft-core system and application development in an FPGA design environment. We will be using Intel's EDA design tool Qsys, in conjunction with the Quartus software, and Intel FPGA monitor.

Hardware Requirements

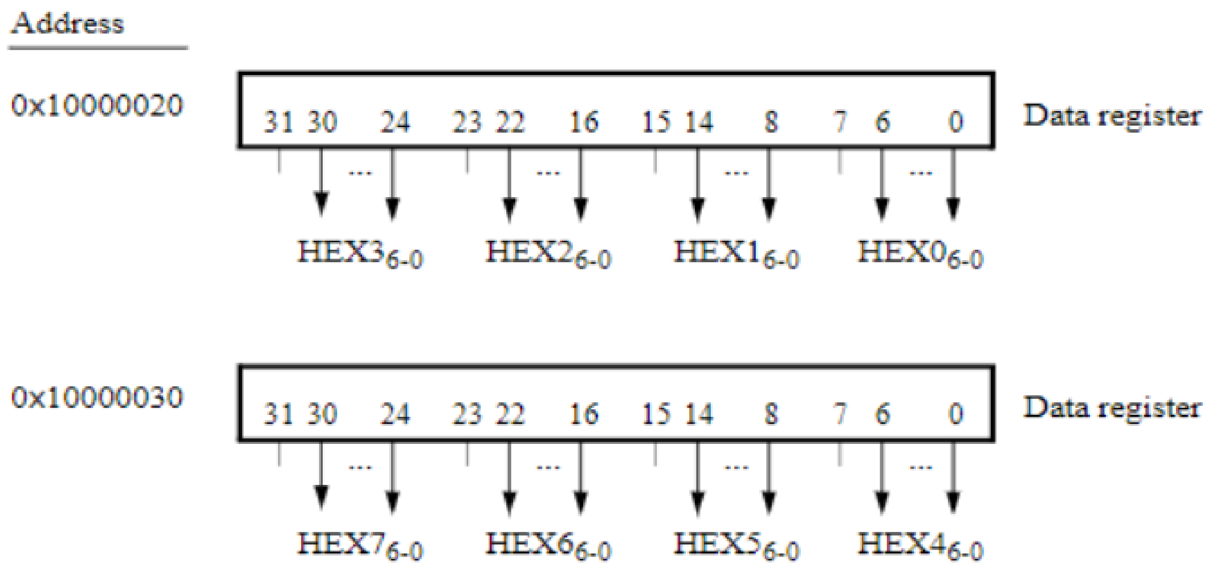
- Computer with Intel FPGA Monitor program 16.1
- Computer with Quartus Prime 16.1.
- DE2-115 Board
- A-B USB Cable

Software Requirements

- Intel FPGA monitor program 16.1 or greater
- Quartus Prime with Qsys, version 16.1.

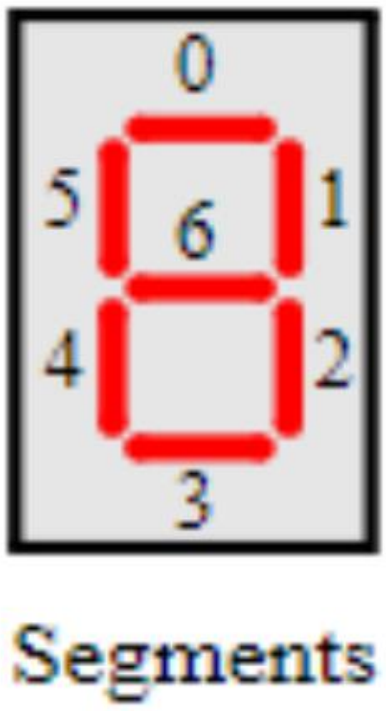
Background

In order to write to a 7-segment display, we have two address spaces, one is for the four most significant bits and the second for the four least significant bits. This is because the DE2-115 board has eight 7-Segment displays. From the design in the last lab, the address spaces for these were, 0x2020 and 0x2030. The typical address spaces for the DE2-115 board can be seen below.



1. 7-Segment Display Address Spaces

Now, from this, we also need to know how to manipulate the 7-Segments in the way that we want. So, being that the display is an active low device, in order to light up a certain display we must pass a 1 into the bit that we want to light up. Now, the segments are controlled individually, so, passing a 7 bit value of 0b1111000 will turn on the bits corresponding to bit zero, one, and two. How these bits correspond to the actual 7 segment display can be seen below.



2. 7-Segment Display Breakdown

With this knowledge we can begin this project.

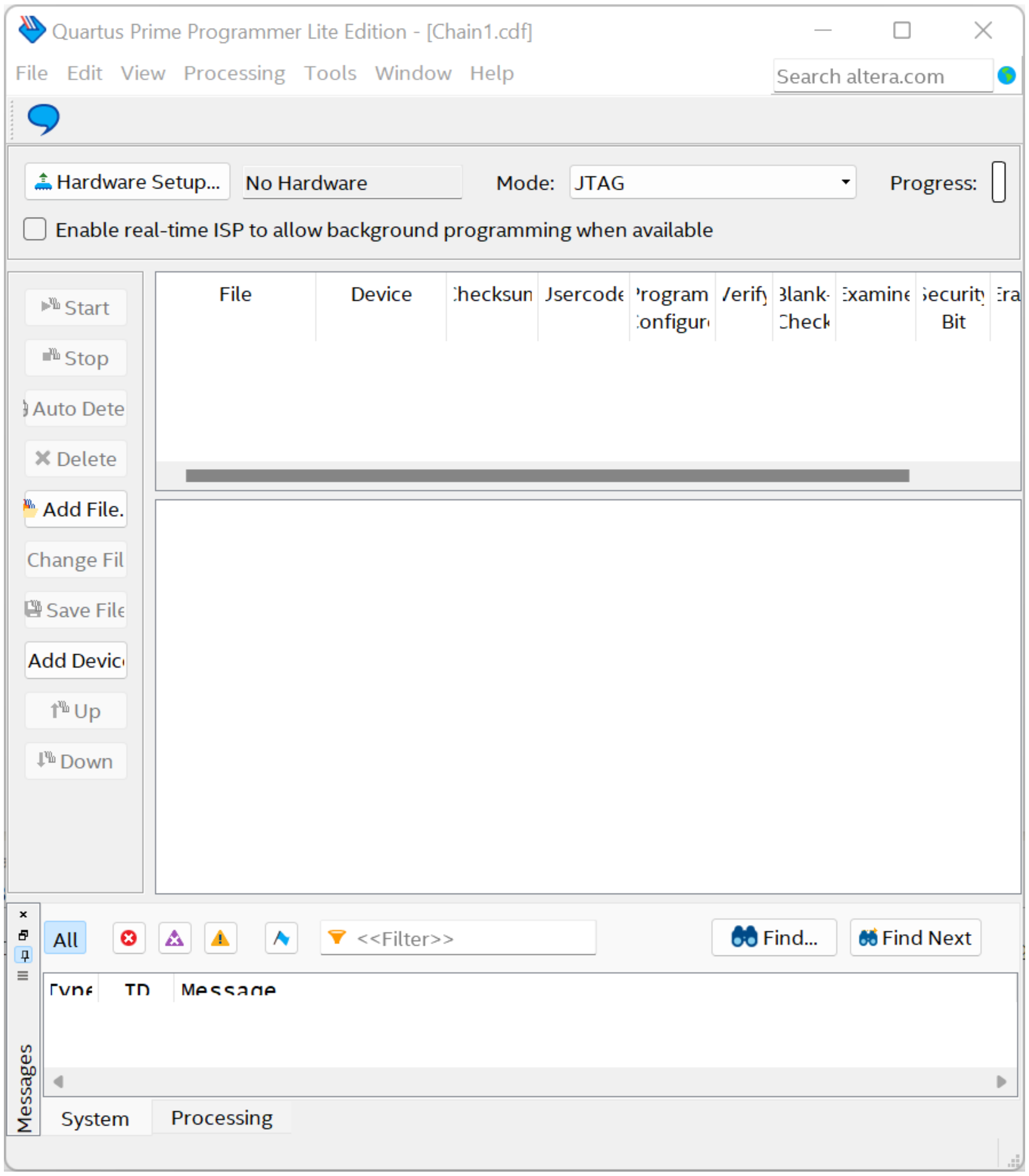
Project Overview

For this lab, we will be using the soft-core embedded system that was developed in a previous project in order to perform a variety of tasks on the DE2-115 board using a C/C++ runtime enviroment. So, to begin, we must first download our system onto the board, and set up a NIOS II Eclipse enviroment. Next, we create a program to request an option one through 6. Options one through four turn on that respective green LED, option 5 will scroll left and 6 scroll right. Lastly, we will create a program to represent the maximum of an 8 bit unsigned and 8 bit signed number as well as the minimum of a 8 bit signed number.

.

Project Procedure

To begin this lab, we first open the Quartus Prime Programmer. The programmer can be seen in the figure below.



Hardware: No Hardware   Mode: JTAG   Progress:

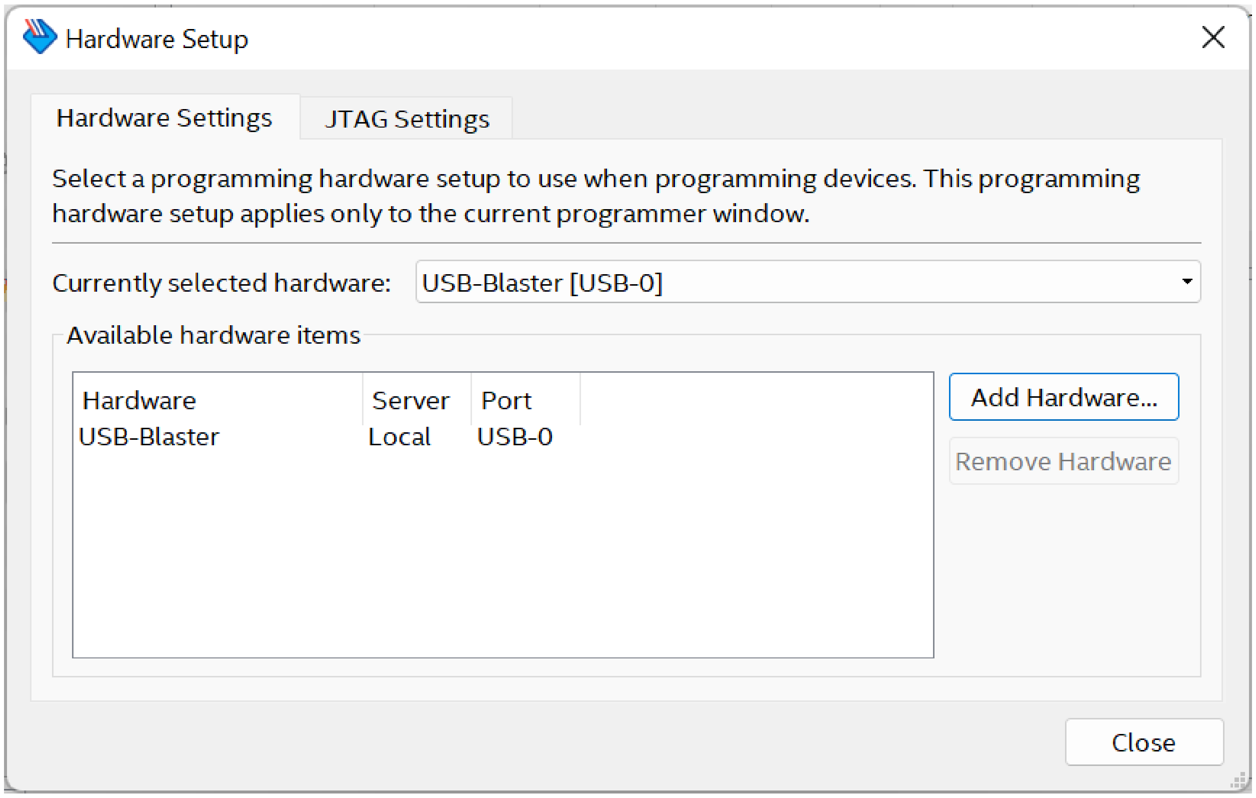
Enable real-time ISP to allow background programming when available

File	Device	hecksun	Jsercode	rogram onfigur	/erify	3lank-Check	Examine	Security Bit	Era

Type	ID	Message

3. Quartus Prime Programmer

Next, we enter the hardware setup. Here we select the USB-Blaster and click add hardware. This will add the USB-Blaster to our hardware list and begin to allow us to download the system onto the board. This window can be seen below.



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

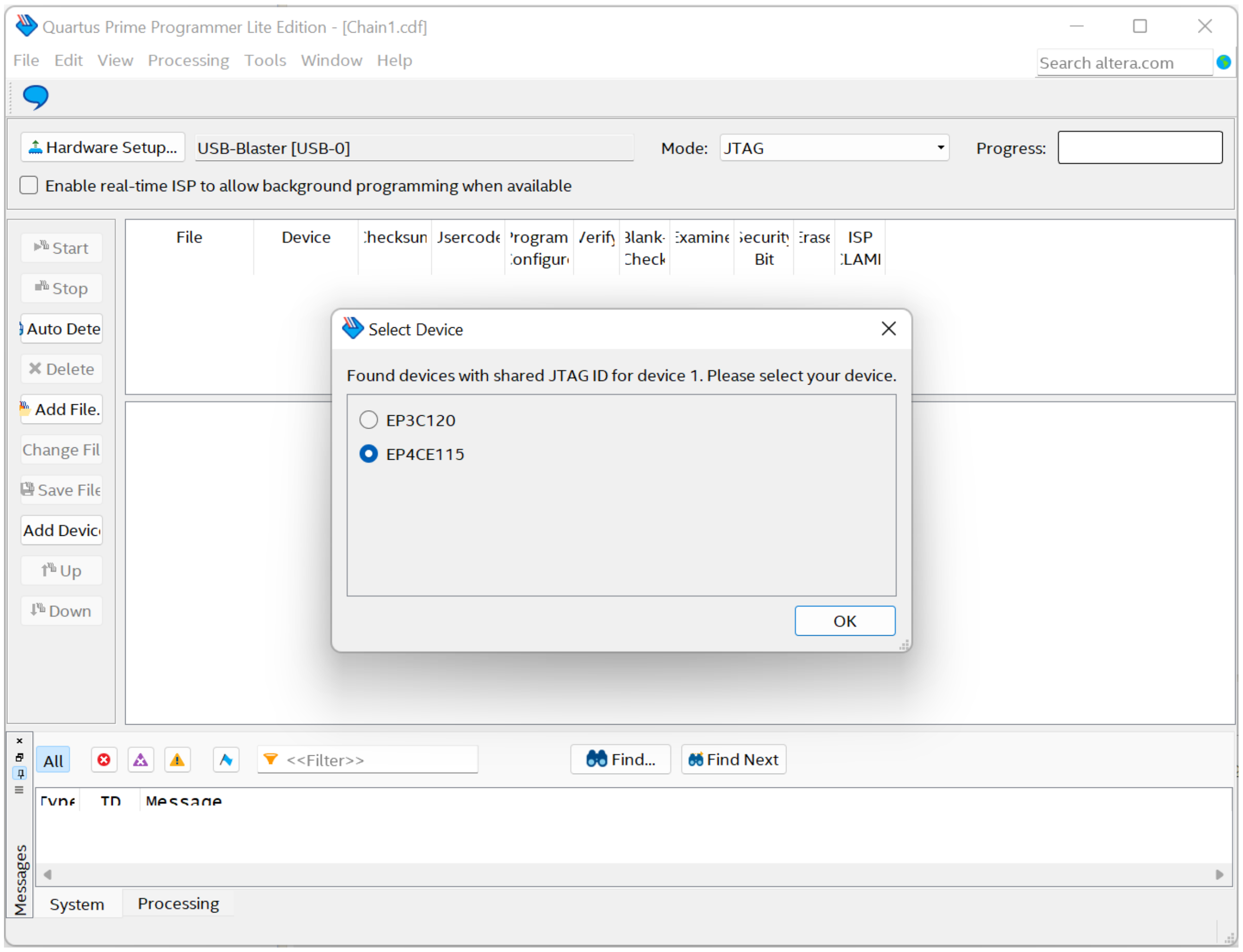
Currently selected hardware: USB-Blaster [USB-0]

Available hardware items

Hardware	Server	Port
USB-Blaster	Local	USB-0

4. Hardware Setup

Next after clicking auto detect, we are able to select our board, and in our case the device is the EP4CE115 for the DE2-115 board. So, seen below, we select the device and click okay.



USB-Blaster [USB-0]

Mode: JTAG

Progress:

Enable real-time ISP to allow background programming when available

Select Device

Found devices with shared JTAG ID for device 1. Please select your device.

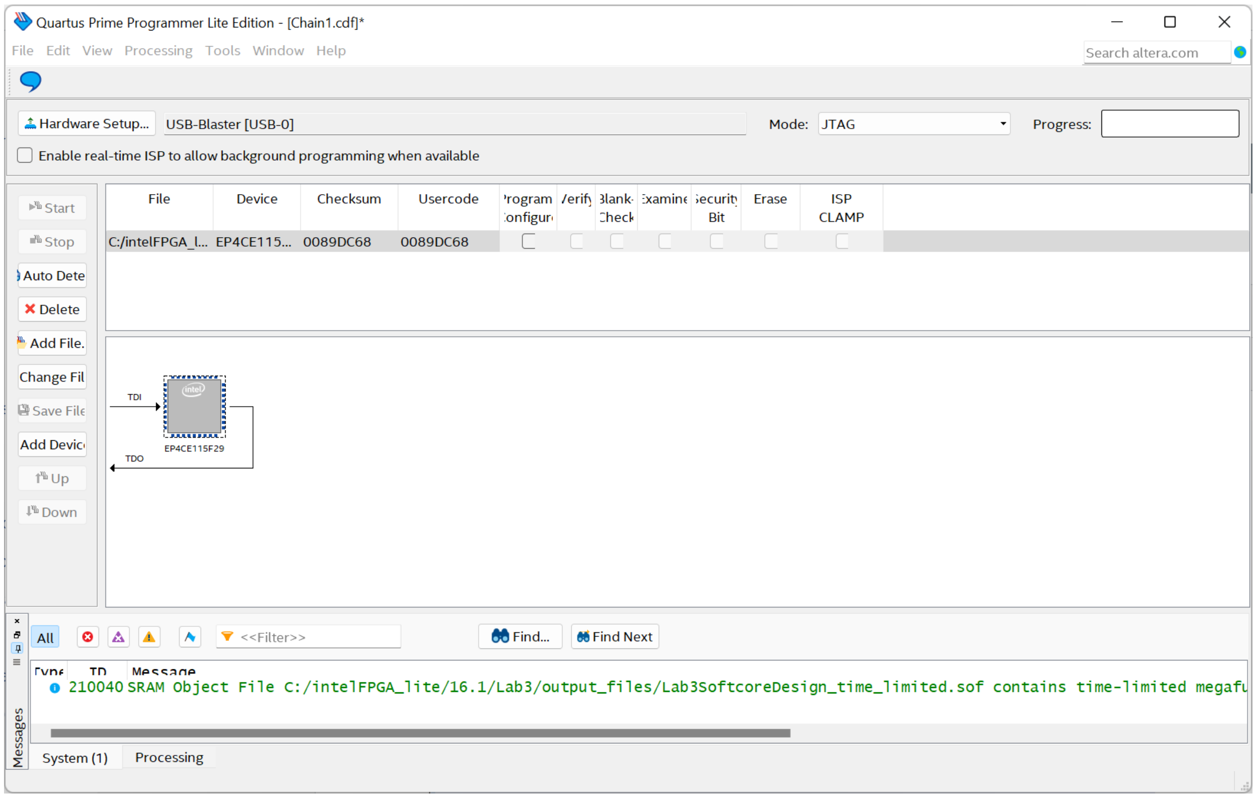
EP3C120 [✓ EP4CE115]

OK

Type ID Message

5. Selecting Device

Next, we must download the sof file onto the board. So, we double click in the file section and add the sof file from a previous lab, Lab 3.


Quartus Prime Programmer Lite Edition - [Chain1.cdf]*

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress:

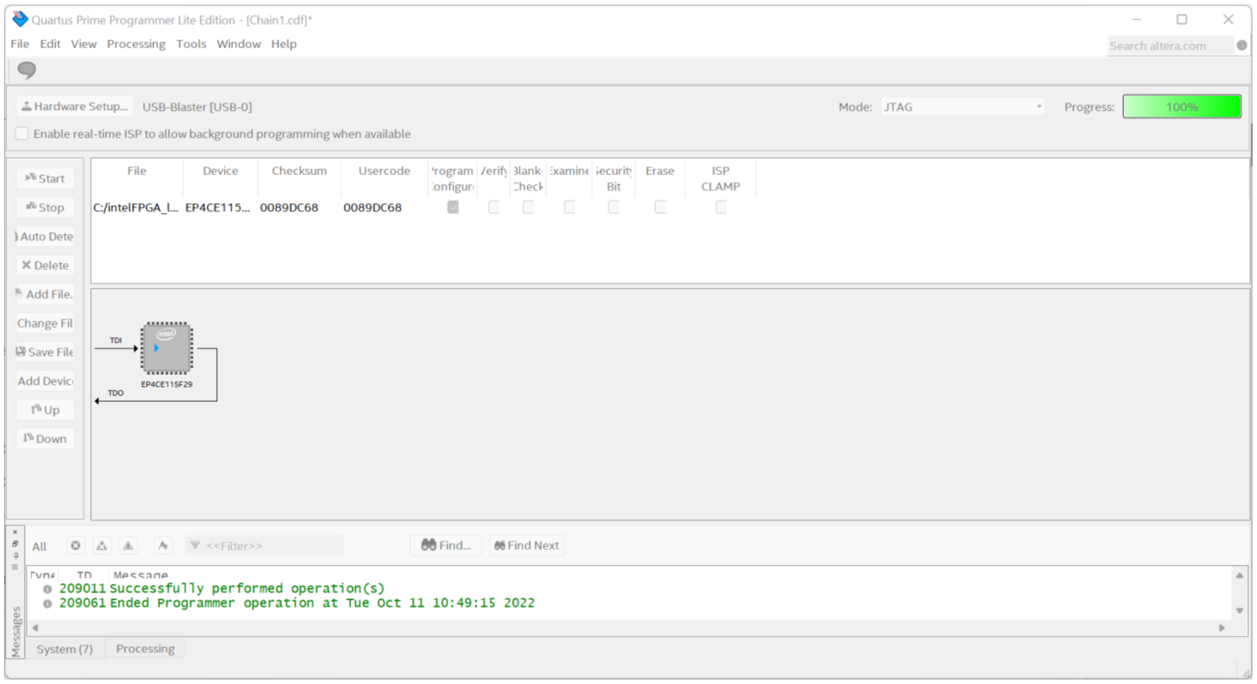
Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
C:/intelFPGA_l...	EP4CE115...	0089DC68	0089DC68	☐	☐	☐	☐	☐	☐	☐

```
Type  ID     Message
      210040 SRAM Object File C:/intelFPGA_lite/16.1/Lab3/output_files/Lab3SoftcoreDesign_time_limited.sof contains time-limited megafu
```

6. Adding the File to be Downloaded onto the Board

After adding the sof file, it is fairly simple, we must next check the program configuration box, and click start. This will download the system onto the DE2-115 board. A successful download can be seen in the figure below, marked by the progress bar in the top right being at 100%, and the successes in the console.

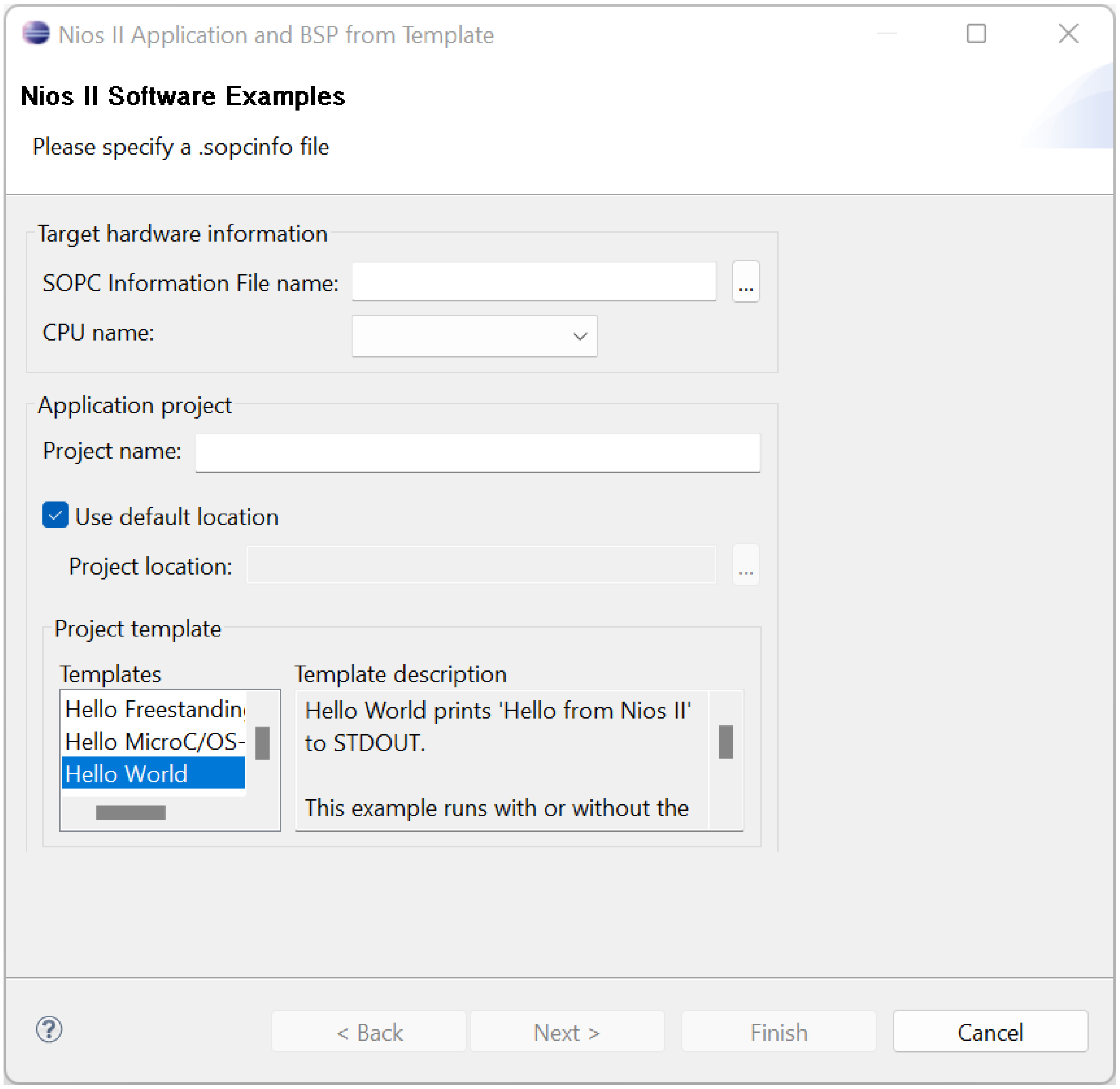

Quartus Prime Programmer Lite Edition - [Chain1.cdf]*
File Edit View Processing Tools Window Help
Hardware Setup... USB-Blaster [USB-0]
Mode: JTAG
Progress: 100%
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Dete
Delete
Add File.
Change Fil
Save File
Add Devic
Up
Down
TDI
TDO
EP4CE115F29
Find...
Find Next
System (7)
Processing

File	Device	Checksum	Usercode	Program/Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
C:/intelFPGA_l...	EP4CE115...	0089DC68	0089DC68	✓						

```
209011 Successfully performed operation(s)
209061 Ended Programmer operation at Tue Oct 11 10:49:15 2022
```

7. Completed Download onto the Board

After downloading the system on the board, we then launch NIOS II Software Build tools for Eclipse. This program can be found in the tools drop down menu in Quartus Prime. From here, we create a NIOS II Application and BSP from template, and the pop-out window can be seen below.


Nios II Application and BSP from Template

Nios II Software Examples

Please specify a .sopcinfo file

Target hardware information

SOPC Information File name:

CPU name:

Application project

Project name:

✓ Use default location

Project location:

Project template

Templates

Hello Freestandin
Hello MicroC/OS-
Hello World

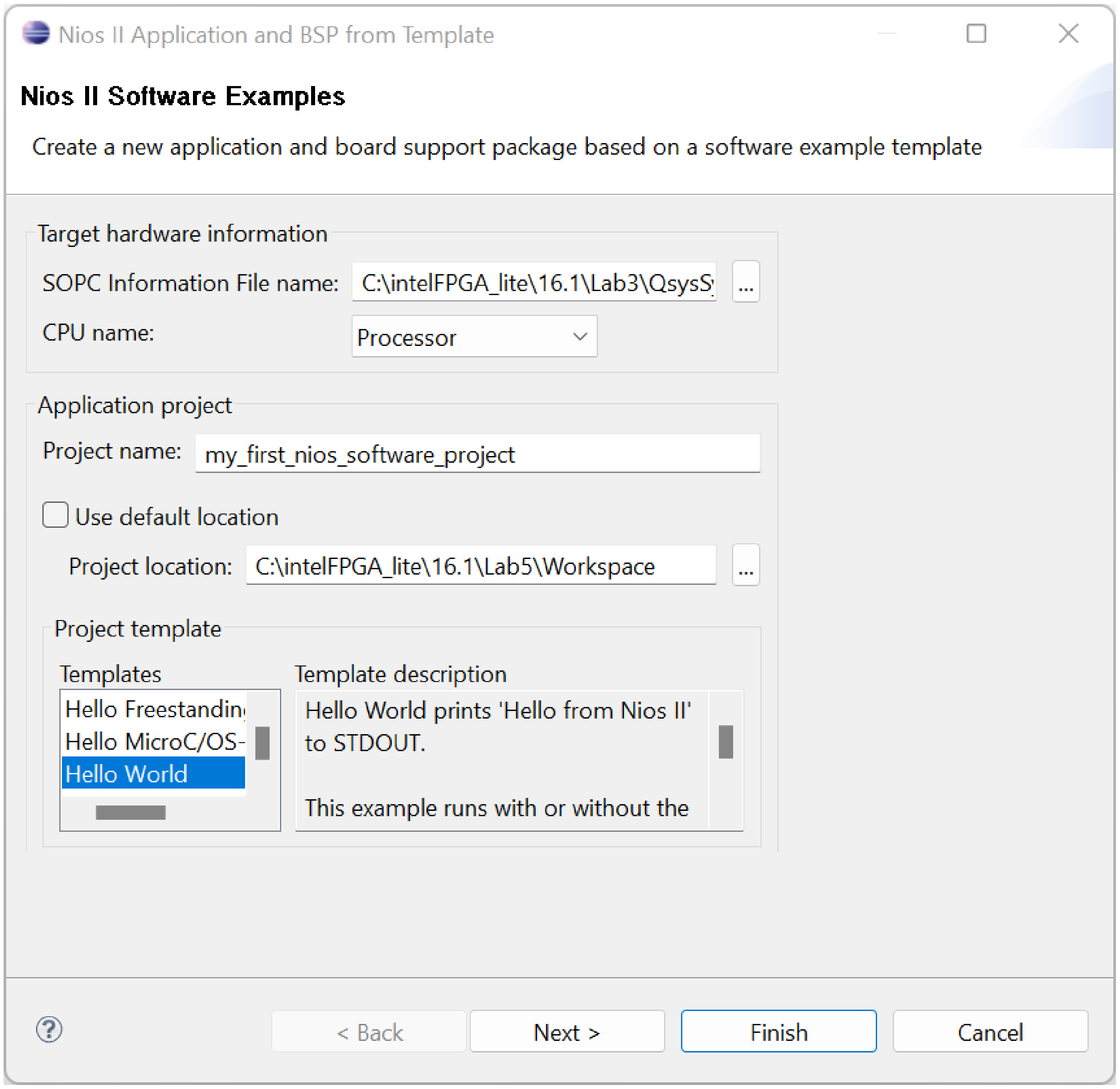
Template description

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

This example runs with or without the

8. NIOS II Application and BSP from Template

After setting the SOPC Info file to the correct one corresponding to our board, setting the CPU Name, as well as the project name, and location, we select the Hello World from the project templates. We can see this in the figure below.


Nios II Application and BSP from Template

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: C:\intelFPGA_lite\16.1\Lab3\QsysS

CPU name: Processor

Application project

Project name: my_first_nios_software_project

☐ Use default location

Project location: C:\intelFPGA_lite\16.1\Lab5\Workspace

Project template

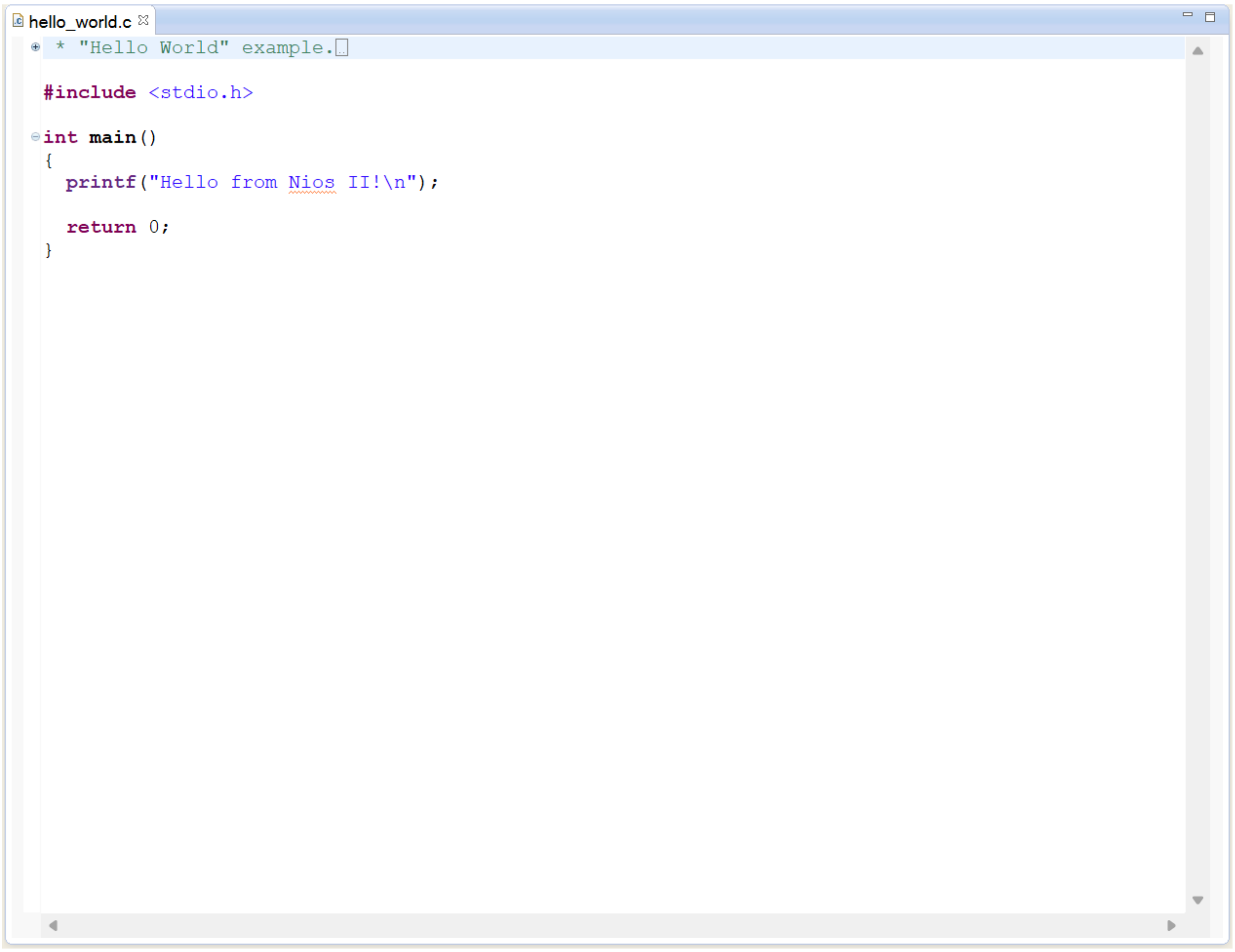
Templates: Hello Freestandin, Hello MicroC/OS-, Hello World

Template description: Hello World prints 'Hello from Nios II' to STDOUT.

This example runs with or without the

9. NIOS II Application and BSP From Template Settings

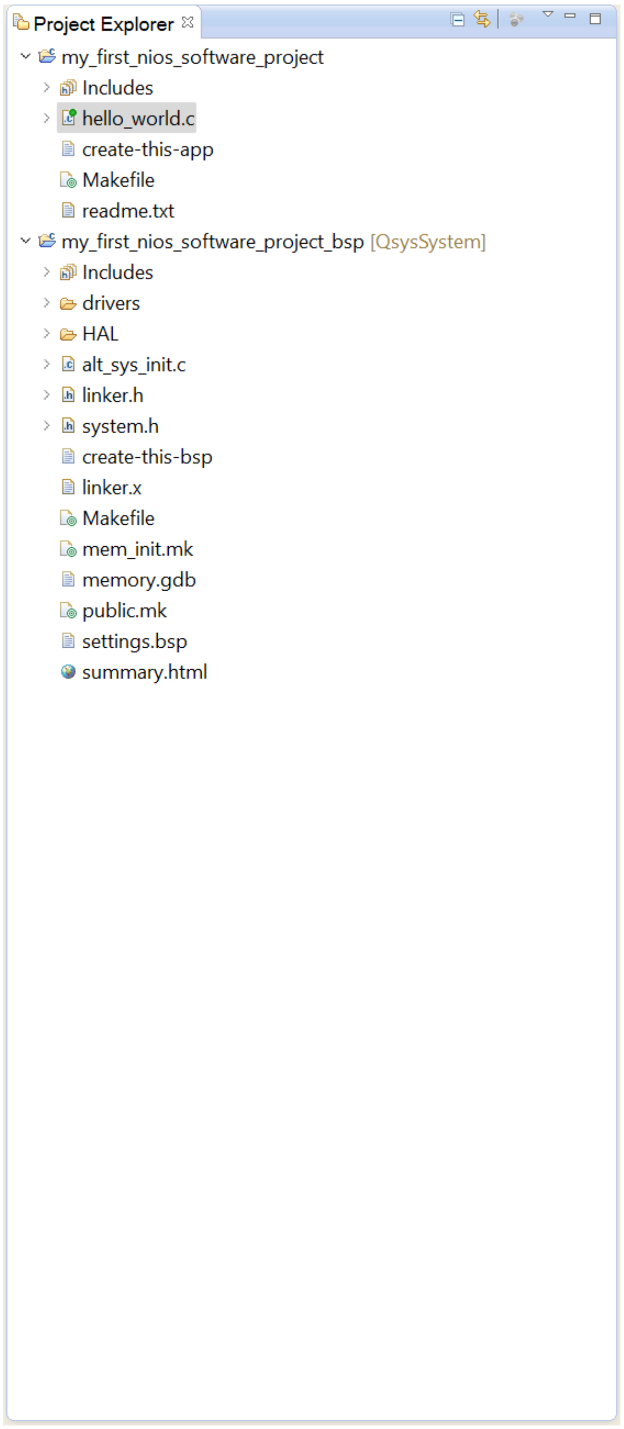
This will then create a simple hello world program as well as create the necessary files and folders for the project. The Program can be seen in the figure below.



```c
 * "Hello World" example.

#include <stdio.h>

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

  return 0;
}
```

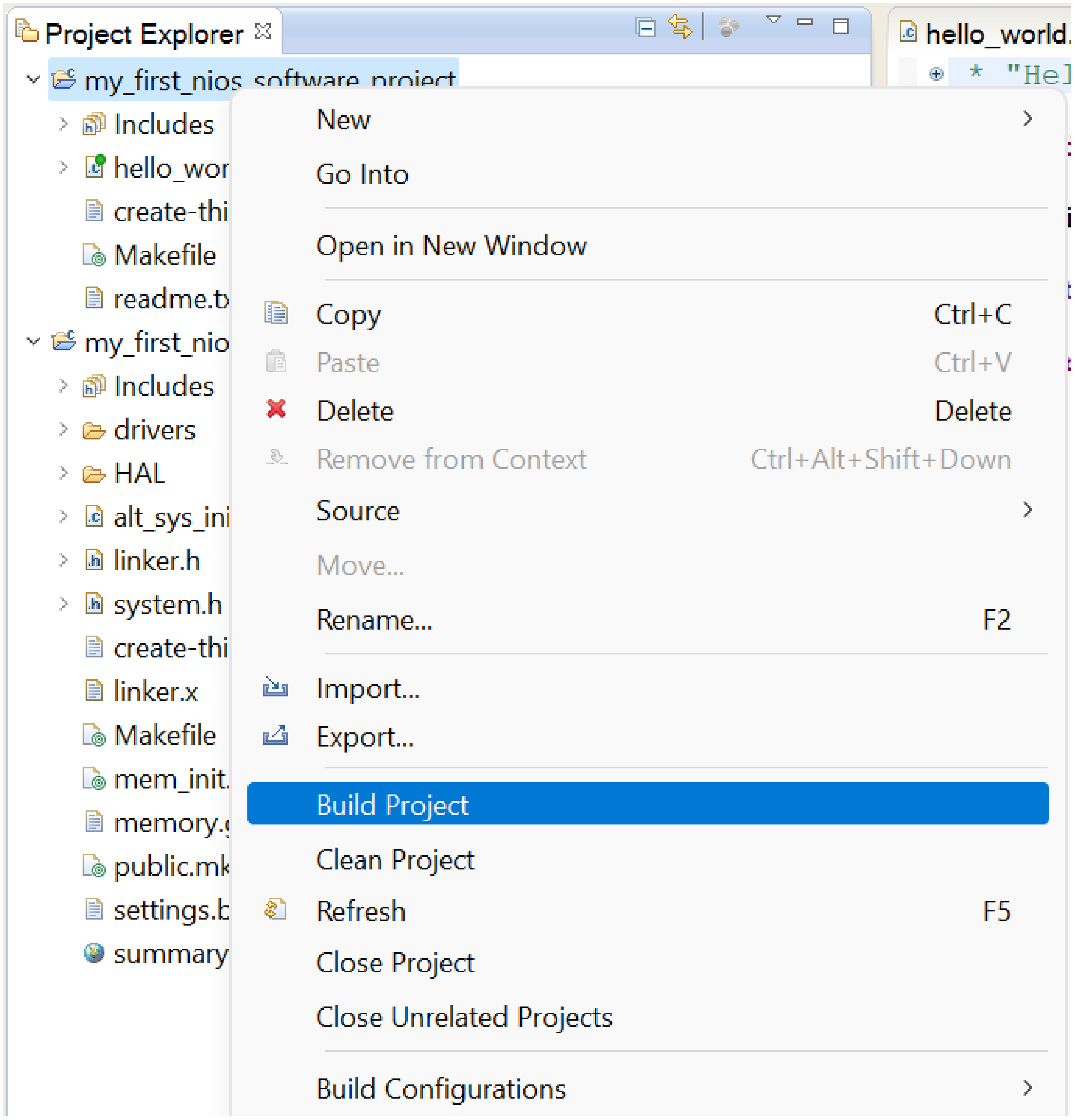
10. Hello_world.c Program

This is a very simple, program which just prints “Hello from Nios II” in the console. We can see the project breakdown in the project explorer below.



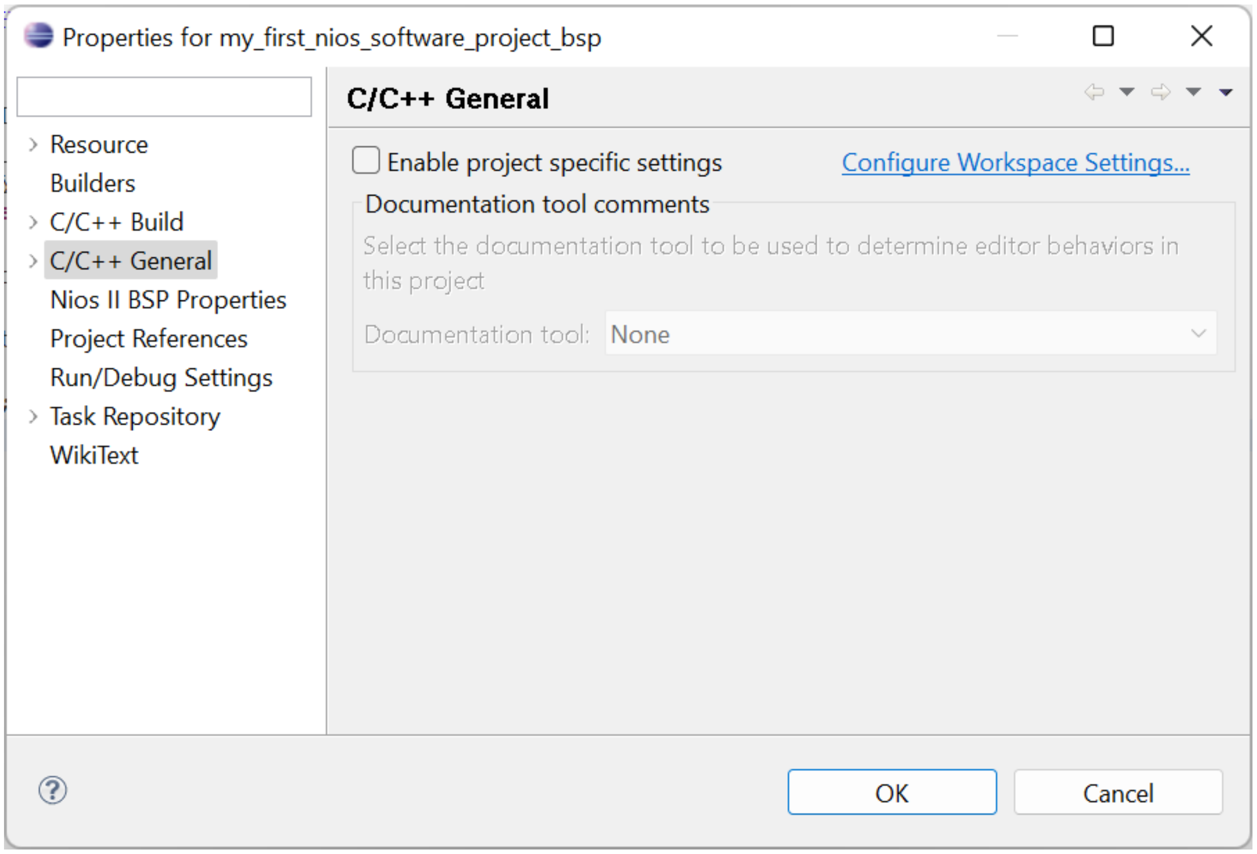
11. Project Files

From this, we can see the important files that we will be using in this program. System.h shows all of the values of our PIO devices, a well as other useful information about the system. However, in order to run the program a couple steps must be completed, seen in the figure below



12. Building the project

To run this project, we first right click the project and click build project. Then, we go into the BSP editor, by right clicking the BSP project, and clicking properties. This will then open the window seen below.


Properties for my_first_nios_software_project_bsp

- Resource
- Builders
- C/C++ Build
- C/C++ General
- Nios II BSP Properties
- Project References
- Run/Debug Settings
- Task Repository
- WikiText

C/C++ General

☐ Enable project specific settings   Configure Workspace Settings...

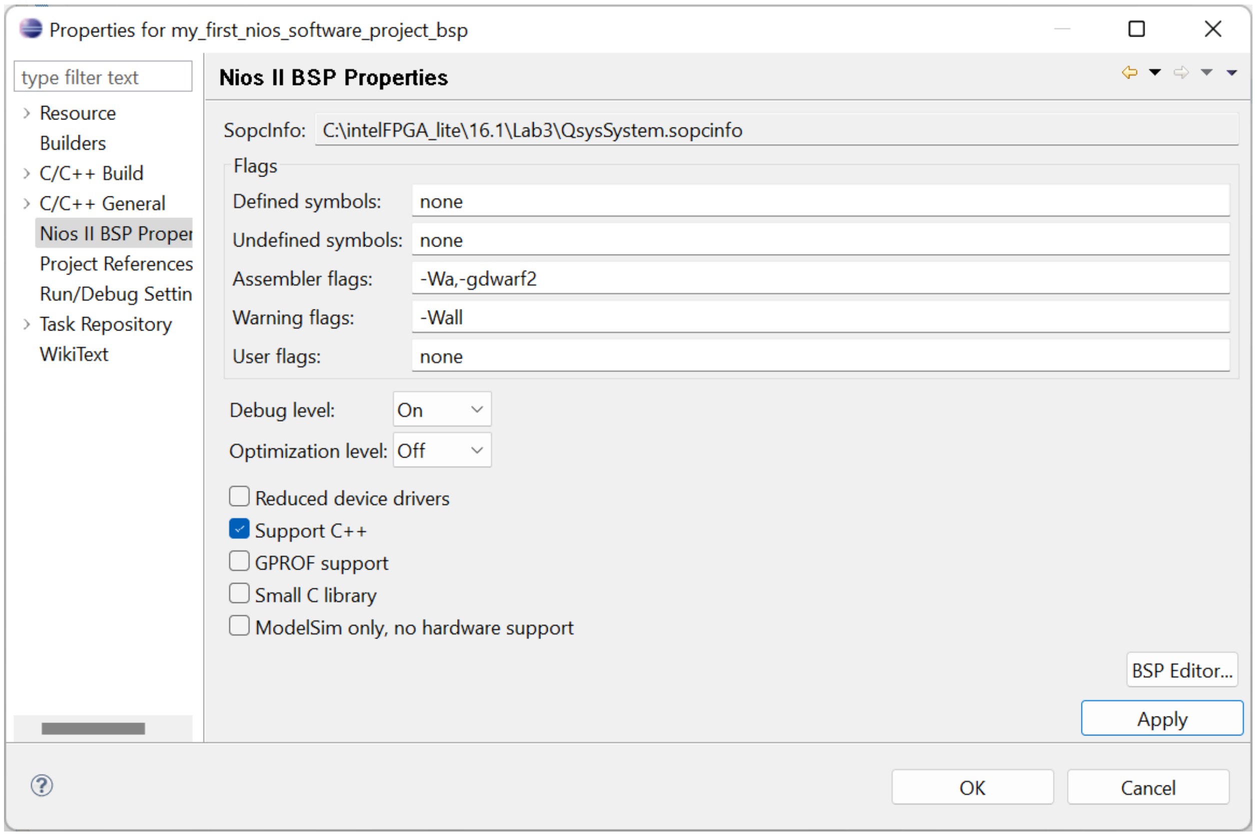
Documentation tool comments

Select the documentation tool to be used to determine editor behaviors in this project

Documentation tool: None

13. BSP File Properties

After opening this window, we select NIOS II BSP Properties from the table on the left. The following figure with the NIOS II BSP Properties then appears.


Properties for my_first_nios_software_project_bsp

Nios II BSP Properties

SopcInfo: C:\intelFPGA_lite\16.1\Lab3\QsysSystem.sopcinfo

Flags

Defined symbols:	none
Undefined symbols:	none
Assembler flags:	-Wa,-gdwarf2
Warning flags:	-Wall
User flags:	none

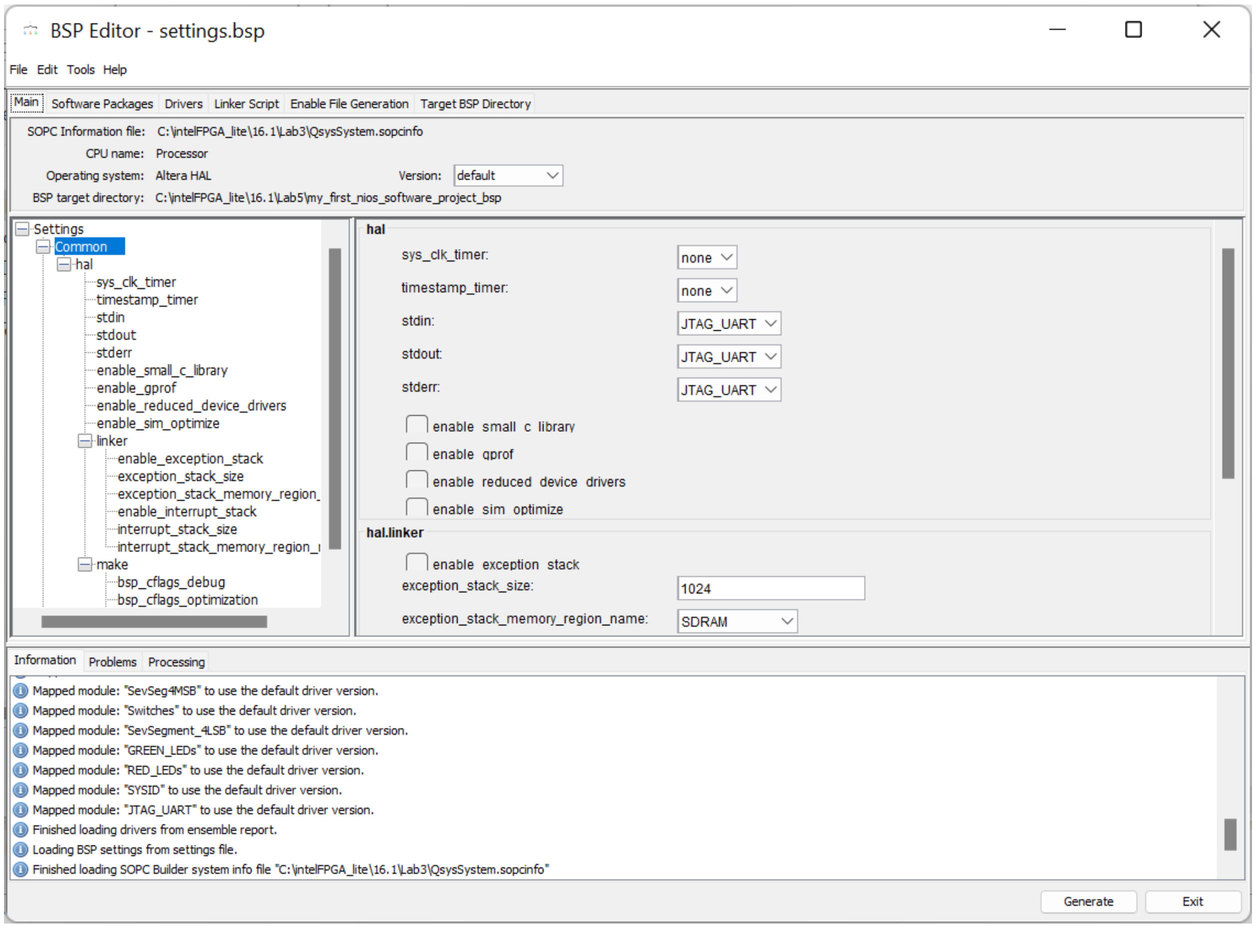
Debug level: On

Optimization level: Off

- [ ] Reduced device drivers
- [x] Support C++
- [ ] GPROF support
- [ ] Small C library
- [ ] ModelSim only, no hardware support

14. NIOS II BSP Properties

Here we can see the BSP properties for the project, by going into the BSP Editor we can see the settings more in depth.


BSP Editor - settings.bsp
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Generate
Exit
Information Problems Processing

SOPC Information file: C:\intelFPGA_lite\16.1\Lab3\QsysSystem.sopcinfo
CPU name: Processor
Operating system: Altera HAL Version: default
BSP target directory: C:\intelFPGA_lite\16.1\Lab5\my_first_nios_software_project_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_
        enable_interrupt_stack
        interrupt_stack_size
        interrupt_stack_memory_region_
      make
        bsp_cflags_debug
        bsp_cflags_optimization
```

hal

sys_clk_timer:	none
timestamp_timer:	none
stdin:	JTAG_UART
stdout:	JTAG_UART
stderr:	JTAG_UART
☐ enable small c library	
☐ enable gprof	
☐ enable reduced device drivers	
☐ enable sim optimize	

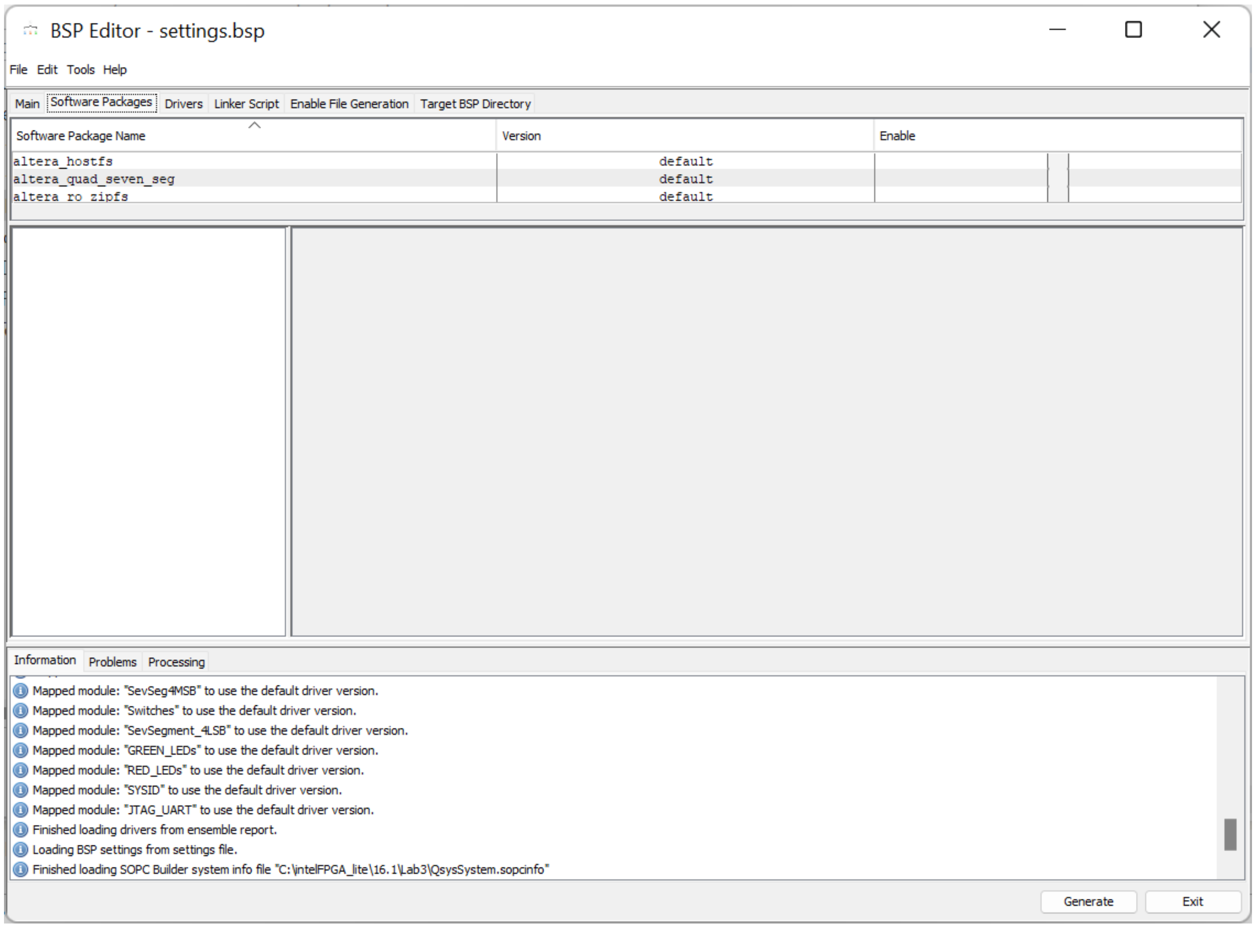
hal.linker

☐ enable exception stack	
exception_stack_size:	1024
exception_stack_memory_region_name:	SDRAM

```
Mapped module: "SevSeg4MSB" to use the default driver version.
Mapped module: "Switches" to use the default driver version.
Mapped module: "SevSegment_4LSB" to use the default driver version.
Mapped module: "GREEN_LEDs" to use the default driver version.
Mapped module: "RED_LEDs" to use the default driver version.
Mapped module: "SYSID" to use the default driver version.
Mapped module: "JTAG_UART" to use the default driver version.
Finished loading drivers from ensemble report.
Loading BSP settings from settings file.
Finished loading SOPC Builder system info file "C:\intelFPGA_lite\16.1\Lab3\QsysSystem.sopcinfo"
```

15. BSP Editor

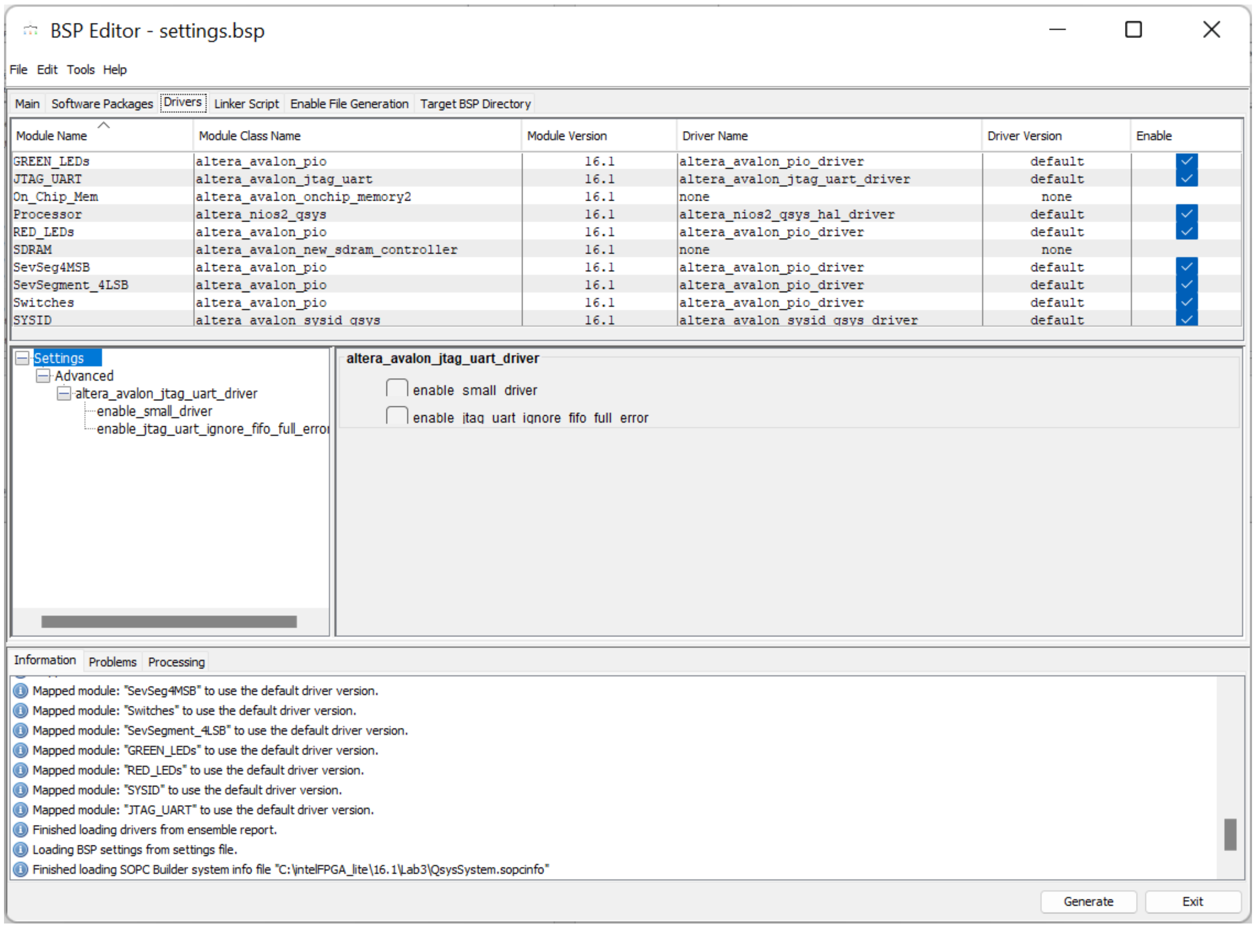
In the BSP Editor main, we can see the HAL settings along with other properties, we can also go into the different tabs in the editor in order to see things like the Software Packages seen in the figure below, drivers immediately following that, and lastly the Linker Script following that.


BSP Editor - settings.bsp
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Information Problems Processing
Generate
Exit

Software Package Name	Version	Enable
altera_hostfs	default	
altera_quad_seven_seg	default	
altera_ro_zipfs	default	

```
Mapped module: "SevSeg4MSB" to use the default driver version.
Mapped module: "Switches" to use the default driver version.
Mapped module: "SevSegment_4LSB" to use the default driver version.
Mapped module: "GREEN_LEDs" to use the default driver version.
Mapped module: "RED_LEDs" to use the default driver version.
Mapped module: "SYSID" to use the default driver version.
Mapped module: "JTAG_UART" to use the default driver version.
Finished loading drivers from ensemble report.
Loading BSP settings from settings file.
Finished loading SOPC Builder system info file "C:\intelFPGA_lite\16.1\Lab3\QsysSystem.sopcinfo"
```

16. Software Packages Settings



BSP Editor - settings.bsp

Module Name	Module Class Name	Module Version	Driver Name	Driver Version	Enable
GREEN_LEDs	altera_avalon_pio	16.1	altera_avalon_pio_driver	default	✓
JTAG_UART	altera_avalon_jtag_uart	16.1	altera_avalon_jtag_uart_driver	default	✓
On_Chip_Mem	altera_avalon_onchip_memory2	16.1	none	none	
Processor	altera_nios2_qsys	16.1	altera_nios2_qsys_hal_driver	default	✓
RED_LEDs	altera_avalon_pio	16.1	altera_avalon_pio_driver	default	✓
SDRAM	altera_avalon_new_sdram_controller	16.1	none	none	
SevSeg4MSB	altera_avalon_pio	16.1	altera_avalon_pio_driver	default	✓
SevSegment_4LSB	altera_avalon_pio	16.1	altera_avalon_pio_driver	default	✓
Switches	altera_avalon_pio	16.1	altera_avalon_pio_driver	default	✓
SYSID	altera_avalon_sysid_qsys	16.1	altera_avalon_sysid_qsys_driver	default	✓

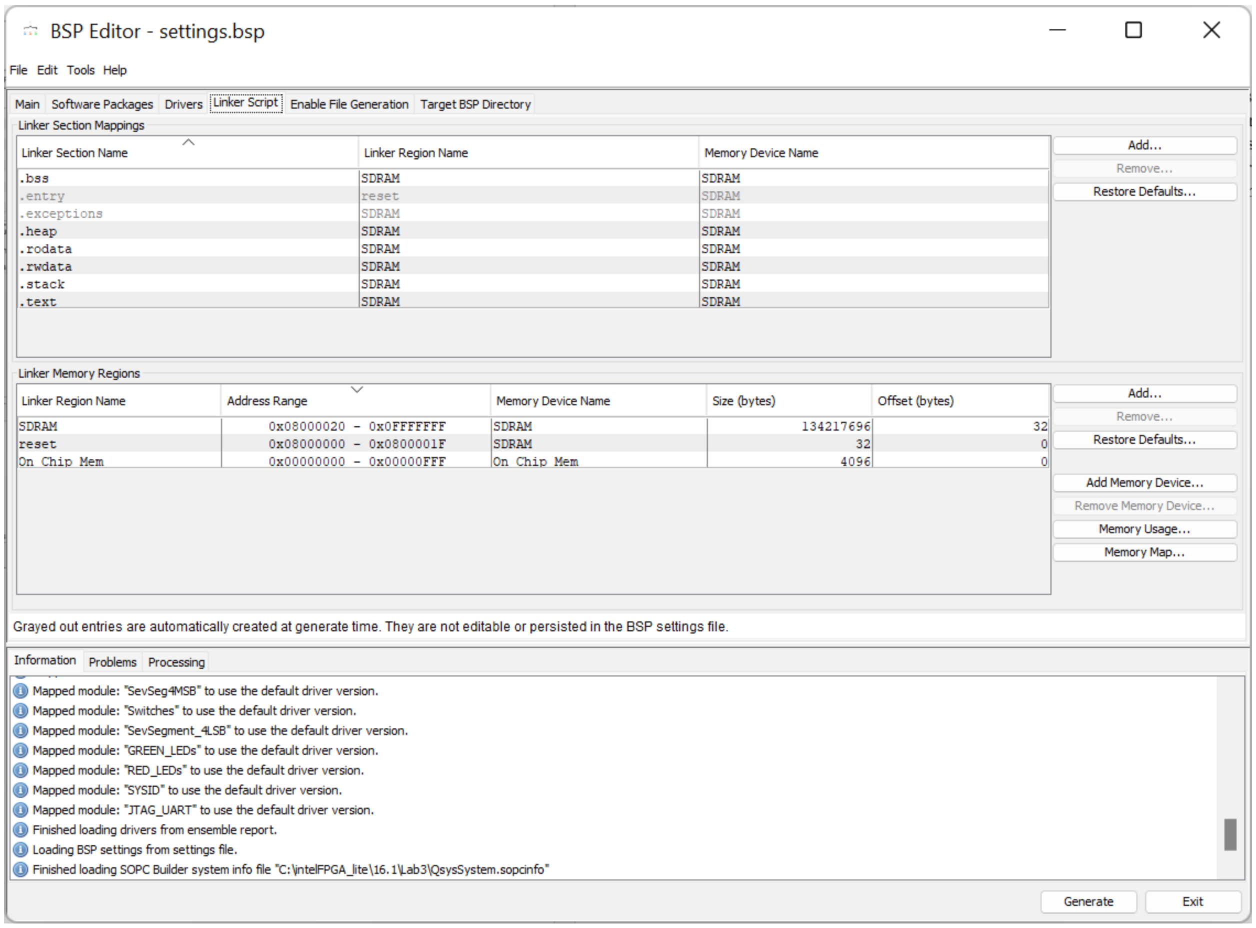
altera_avalon_jtag_uart_driver

- [ ] enable small driver
- [ ] enable jtag uart ignore fifo full error

```
Mapped module: "SevSeg4MSB" to use the default driver version.
Mapped module: "Switches" to use the default driver version.
Mapped module: "SevSegment_4LSB" to use the default driver version.
Mapped module: "GREEN_LEDs" to use the default driver version.
Mapped module: "RED_LEDs" to use the default driver version.
Mapped module: "SYSID" to use the default driver version.
Mapped module: "JTAG_UART" to use the default driver version.
Finished loading drivers from ensemble report.
Loading BSP settings from settings file.
Finished loading SOPC Builder system info file "C:\intelFPGA_lite\16.1\Lab3\QsysSystem.sopcinfo"
```

File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Settings
Advanced
altera_avalon_jtag_uart_driver
enable_small_driver
enable_jtag_uart_ignore_fifo_full_error
Information Problems Processing
Generate Exit

17. Drivers Settings


BSP Editor - settings.bsp
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Add... Remove... Restore Defaults...
Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...
Information Problems Processing
Generate Exit

Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	SDRAM	SDRAM
.entry	reset	SDRAM
.exceptions	SDRAM	SDRAM
.heap	SDRAM	SDRAM
.rodata	SDRAM	SDRAM
.rwdata	SDRAM	SDRAM
.stack	SDRAM	SDRAM
.text	SDRAM	SDRAM

Linker Memory Regions

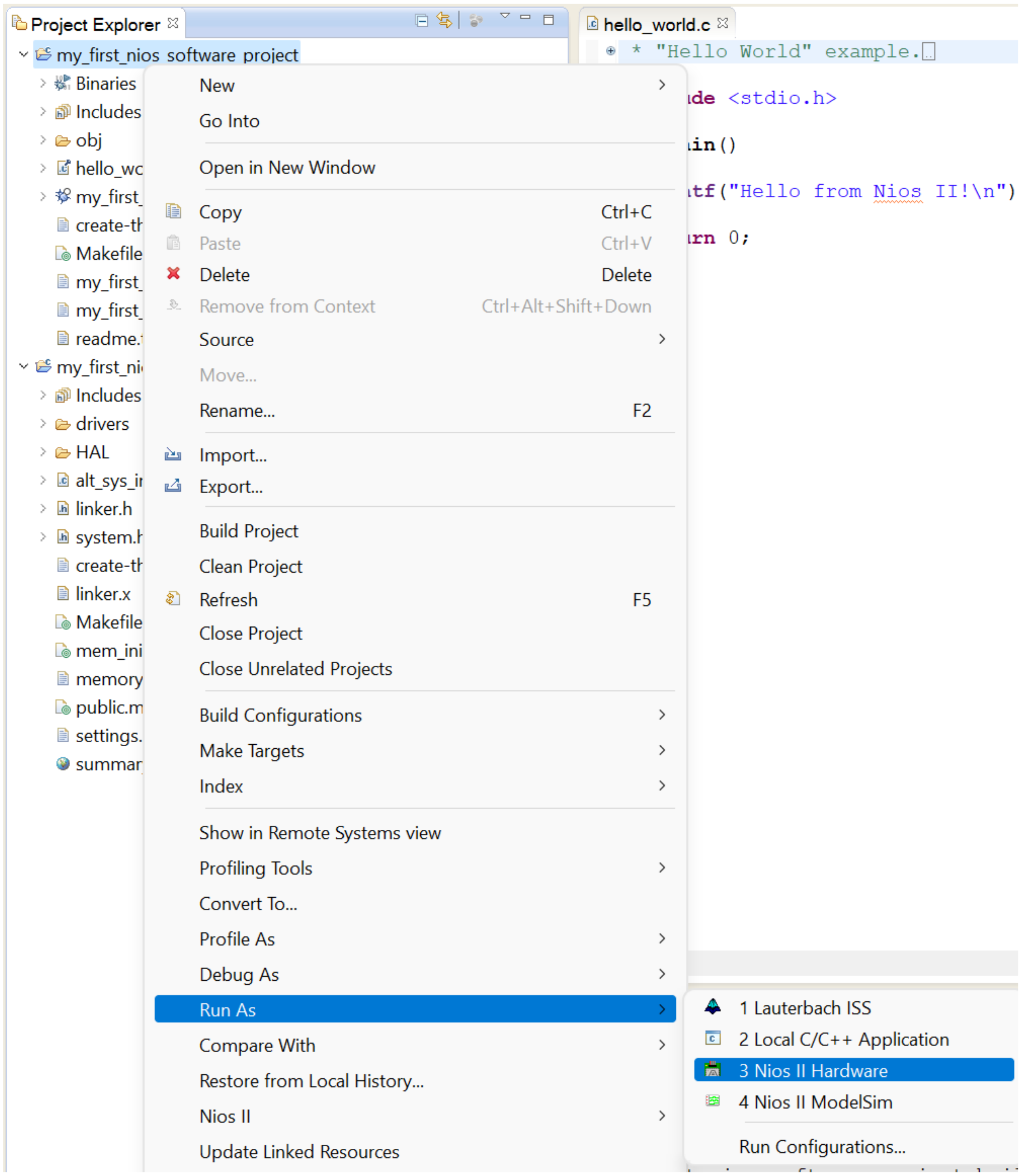
Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
SDRAM	0x08000020 - 0x0FFFFFFF	SDRAM	134217696	32
reset	0x08000000 - 0x0800001F	SDRAM	32	0
On Chip Mem	0x00000000 - 0x00000FFF	On Chip Mem	4096	0

Grayed out entries are automatically created at generate time. They are not editable or persisted in the BSP settings file.

```
Mapped module: "SevSeg4MSB" to use the default driver version.
Mapped module: "Switches" to use the default driver version.
Mapped module: "SevSegment_4LSB" to use the default driver version.
Mapped module: "GREEN_LEDs" to use the default driver version.
Mapped module: "RED_LEDs" to use the default driver version.
Mapped module: "SYSID" to use the default driver version.
Mapped module: "JTAG_UART" to use the default driver version.
Finished loading drivers from ensemble report.
Loading BSP settings from settings file.
Finished loading SOPC Builder system info file "C:\intelFPGA_lite\16.1\Lab3\QsysSystem.sopcinfo"
```

18. Linker Script Settings

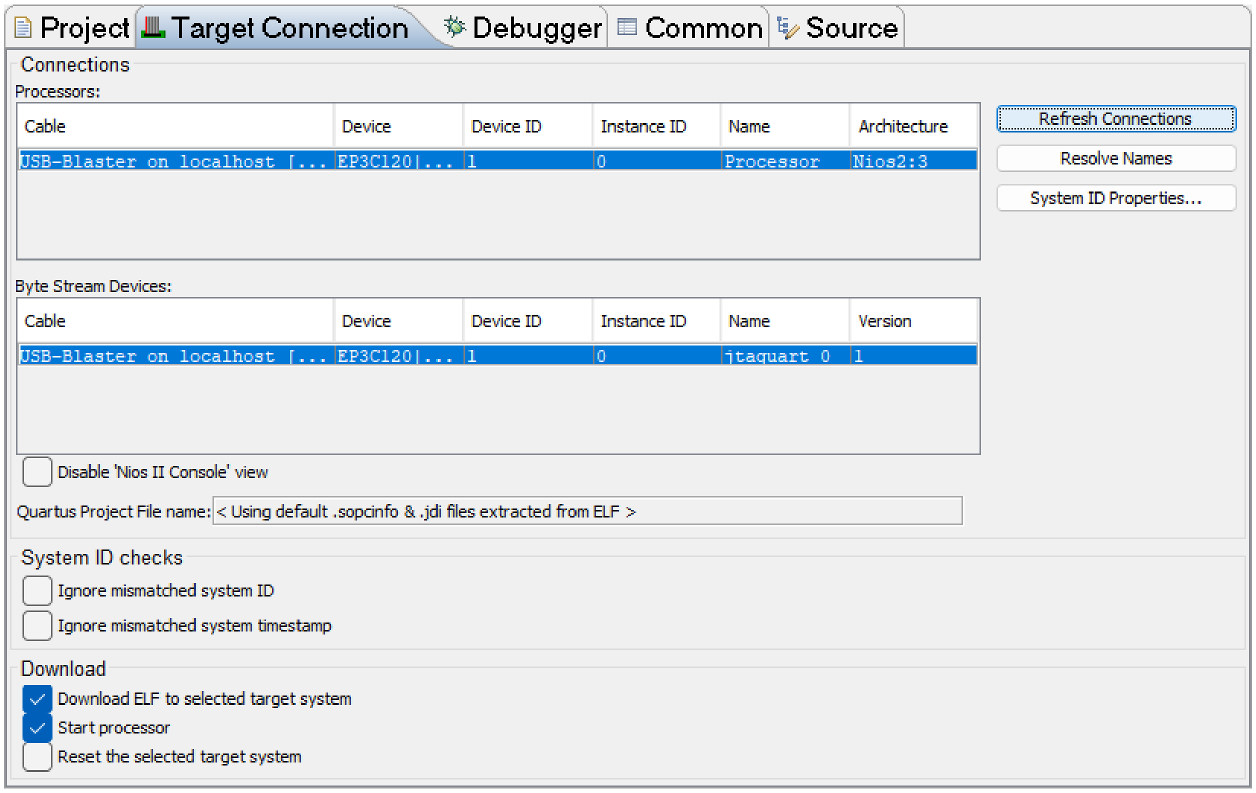
Here we can see all the different settings we have in the BSP. However, for the purpose of this project, we will be leaving all of these settings default. Next, to run our program, we right click the project, Run as, then NIOS II Hardware. Seen in the figure below.



```
 * "Hello World" example.
ude <stdio.h>
ain()
ntf("Hello from Nios II!\n")
urn 0;
```

19. Running As Nios II Hardware

After selecting Run as Nios II Hardware, a window will appear to set up the connection of the USB blaster. Here we refresh the connections and select the USB-Blaster. This is shown in the figure below.



Connections

Processors:

Cable	Device	Device ID	Instance ID	Name	Architecture
USB-Blaster on localhost [...	EP3C120|...	1	0	Processor	Nios2:3

Byte Stream Devices:

Cable	Device	Device ID	Instance ID	Name	Version
USB-Blaster on localhost [...	EP3C120|...	1	0	jtaguart_0	1

- [ ] Disable 'Nios II Console' view

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

System ID checks

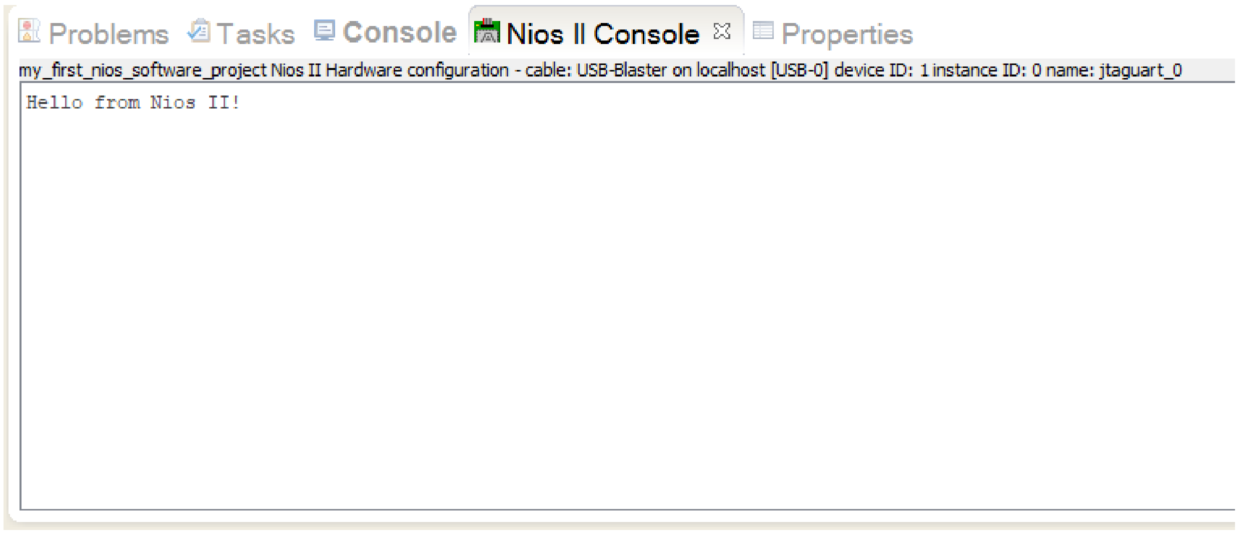
- [ ] Ignore mismatched system ID
- [ ] Ignore mismatched system timestamp

Download

- [✓] Download ELF to selected target system
- [✓] Start processor
- [ ] Reset the selected target system

20. Setting the Target Connection

After the Target connection is set, run as Nios II Hardware again, and the program will run, when the program runs the following is displayed in the console.



```
my_first_nios_software_project Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
Hello from Nios II!
```

21. Output from Hello_World

So, now that this is done, we create a program for part 1 which is to request a user input of 1 through 6. An input of one through four will turn on that respective LED, where an input of 5 to 6 will scroll left and scroll right respectively. The code for this part can be seen in appendix A.

The approach that was used for Part 1, was to first print the options that the user had to the console. They will select an option, options one through four will simply output 1, 2, 4, 8 to the Green LEDs PIO base address respectively. Option 5 will set the value to out – (out/2). This is so that, if you are at 8, the 4th LED is on, then scroll right will set your new value to output to 4, which is the third bit essentially creating a walking ones.

Part 2 was to display the min and max of 8 bit unsigned values as well as the max of an 8 bit unsigned value to the LEDs, as well as its decimal equivalent to the seven segment displays. The code for this part can be seen in Appendix B.

We accomplished this task using a display and a delay function. The display function first checks if the value is negative, if it is it will set the fourth segment to be a – sign, and take the absolute value of the negative value. We then use a lookup table for the seven segment display in order to display the decimal digits, using modulo 10 we are able to convert the binary to decimal digits. After it is displayed, it writes the value to the LEDs and delays before beginning the next number.

Conclusion

This lab accomplished its goal of an introduction to using C/C++ to program an FPGA board using Eclipse Software Build tool. In this lab, we were able to create two programs using Eclipse to accomplish the goals that we set out to. The value from this lab was astronomical going forward, as C/C++ is a lot closer to industry standard than assembly which has been all that was previously used in this class and other classes.

Appendix A (Part 1 Code)

```
#include <stdio.h>
#include <system.h>
#include <alt_types.h>
#include "altera_avalon_pio_regs.h"

int main()
{
  printf("Options:\n1. Turn on the first green LED\n2. Turn on 
the second green LED\n");
  printf("3. Turn on the third green LED\n3\4. Turn on the 
fourth green LED\n");
  printf("5. Scroll Left\n6. Scroll Right\n");
  int* userin = 0;
  int option = 0;
  alt_u8 out = 00000000;
  while(1){
       printf("\nSelect an Option:");
       scanf("%d", userin);
       option = *userin;
       switch(option){
       case 1:
            out = 1;
            IOWR(GREEN_LEDS_BASE, 0, out);
            break;

       case 2:
            out = 2;
            IOWR(GREEN_LEDS_BASE, 0, out);
            break;

       case 3:
            out = 4;
            IOWR(GREEN_LEDS_BASE, 0, out);
            break;

       case 4:
            out = 8;
            IOWR(GREEN_LEDS_BASE, 0, out);
            break;

       case 5:
            if (out == 8)
```

```
                 out = 1;
            else
                 out = out + out;
            IOWR(GREEN_LEDS_BASE, 0, out);
            break;

       case 6:
            if (out == 1)
                 out = 8;
            else
                 out = out - (out/2);
            IOWR(GREEN_LEDS_BASE, 0, out);
            break;

       default:
            return 0;
       }


  }

  return 0;
}
```

Appendix B (Part 2 Code)

```
#include <stdio.h>
#include <system.h>
#include <alt_types.h>
#include "altera_avalon_pio_regs.h"

int main(){
	alt_u32 outSevenSeg =0;
	int Max8Bitu = 255;
	int Max8Bits = 127;
	int Min8Bits = -127;
	outSevenSeg = 4294967295;

	IOWR(SEVSEG4MSB_BASE, 0, outSevenSeg);


	while(1){
		Display(Max8Bitu);
		IOWR(RED_LEDS_BASE, 0, Max8Bitu);
		delay();
		Display(Max8Bits);
		IOWR(RED_LEDS_BASE, 0, Max8Bits);
		delay();
		Display(Min8Bits);
		IOWR(RED_LEDS_BASE, 0, (0b000000000011111111 &
Min8Bits));
		delay();
	}





	printf("complete");
	return 0;
}

void Display(int Value){
	int FourthSeg;
	if (Value < 0){
		Value = abs(Value);
		FourthSeg = 0b0111111;
	}
	else
		FourthSeg = 0b1111111;
	int SEVENSEGLUT[9] = {0b01000000, 0b01111001, 0b00100100,
0b00110000, 0b00011001, 0b00010010, 0b00000010, 0b01111000,
0b00000000, 0b00011000};
	int R;
	int Count = 0;
	int toDisplay = 0;
	int shiftVal = 0;

	while (Value != 0){
	R = Value % 10;
	toDisplay = toDisplay + (SEVENSEGLUT[R] << shiftVal);
	Value = (Value - R) / 10;
	shiftVal = shiftVal + 8;
	}
	toDisplay = toDisplay + (FourthSeg << shiftVal);
	IOWR(SEVSEGMENT_4LSB_BASE, 0, toDisplay);
	return;
}

void delay(){
	int delay = 0;
		while(delay < 2000000)
		{
			delay++;
		}

return;
}
```