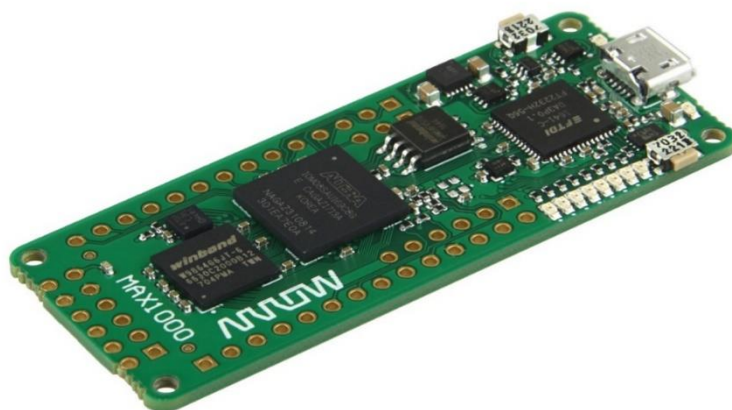


## MAX1000

# Internal Flash Lab



### Software and hardware requirements to complete all exercises

**Software Requirements:** Quartus® Prime Lite or Standard Edition version 18.0 or 18.1

**Hardware Requirements:** ARROW MAX1000 Board



## 1. Introduction

This tutorial provides comprehensive information to help you understand how to use the internal flash feature of the MAX10 and how to run it on your MAX1000 board. The Nios II processor is a soft intellectual property (IP) processor that you download (along with other hardware components that comprise the Nios II system) onto an Intel FPGA. At the end, you will know how to setup your Quartus project in order to boot your Nios processor from the internal flash memory.

**Lab Notes:** Many of the names that the lab asks you to choose for files, components, and other objects in this exercise must be spelled exactly as directed. This nomenclature is necessary because the pre-written software application includes variables that use the names of the hardware peripherals. Naming the components differently can cause the software application to fail. There are also other similar dependencies within the project that require you to enter the correct names.

## 2. Getting Started

The first objective is to ensure that you have all the necessary hardware items and software installed so that the lab can be completed successfully. Below is a list of items required to complete this lab:

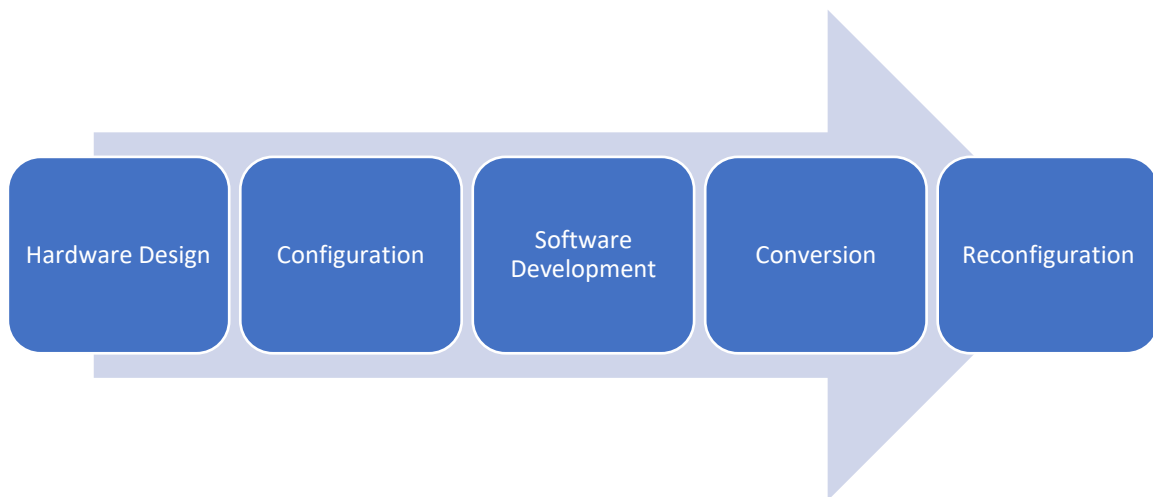
- MAX1000 Board (10M08SAU169C8G)
- USB Cable
- Lab files: Internal\_Flash\_lab\_template: Template files are required to complete the lab. Includes: nios\_subsystem.qsys, main.c
- Quartus Prime 18.0 Lite was used for this lab. Previous/newer versions should work (If no Quartus Prime is installed, refer to MAX1000 User Guide for instructions)
- Installed Arrow USB Drivers (If not, refer to MAX1000 User Guide for instructions)
- Personal computer or laptop running 64-bit Linux / Windows 7 or later with at least an Intel i3 core (or equivalent), 4GB RAM and 12 GB of free hard disk space
- A desire to learn!

### 3. Design Flow

MAX10 devices contain on-chip flash which segmented to two types:

- Configuration Flash Memory (CFM) to store hardware configuration settings for MAX10 FPGA.
- User Flash Memory (UFM) to store user software application.

You can boot and configure the Nios II soft core processor to execute code from the on-chip flash within the FPGA.



The above diagram shows the typical design flow for the system design with the internal flash memory. The system definition is done with Platform Designer. The Nios II IDE uses the system description to create a new project for the software application. The output of the FPGA design is a FPGA image that is used to configure the FPGA. The output of the software flow is an executable which runs on the Nios II processor. At the end, the Quartus puts these two files together in a single configuration image.

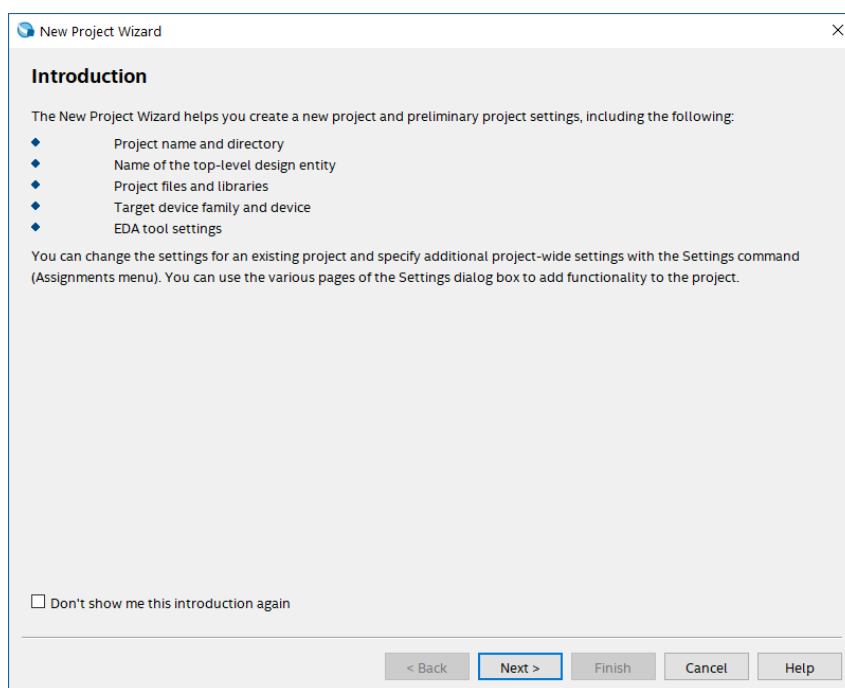
## 4. Project with MAX1000

### 4.1 Quartus Prime project

#### 4.1.1 Create a new Quartus Prime project

4.1.1.1 If not already open, from the Start menu or the Desktop, open the Quartus Prime 18.0 Lite software.

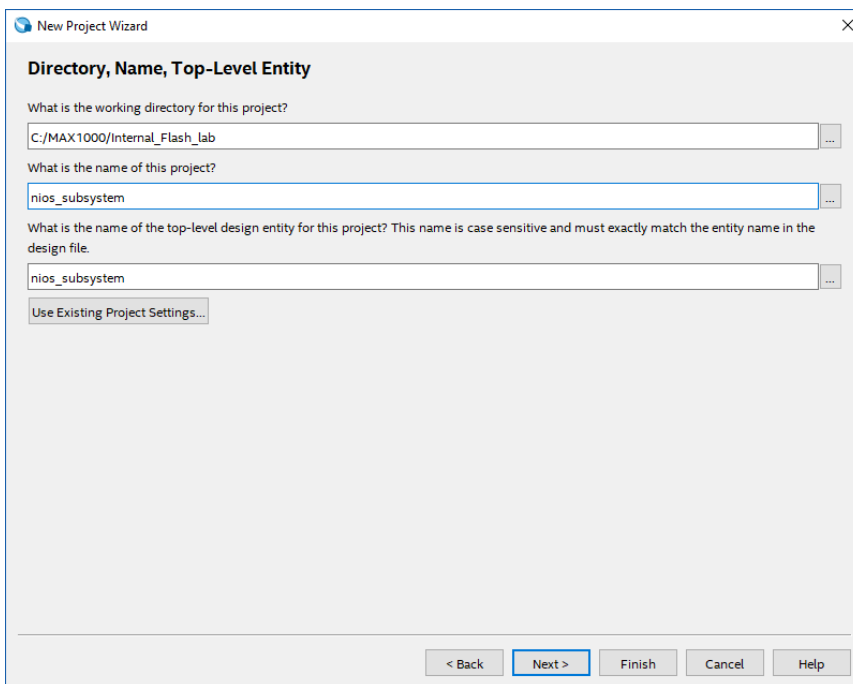
4.1.1.2 Create a new project using the New Project Wizard: **File → New Project Wizard.**



4.1.1.3 Click **Next**.

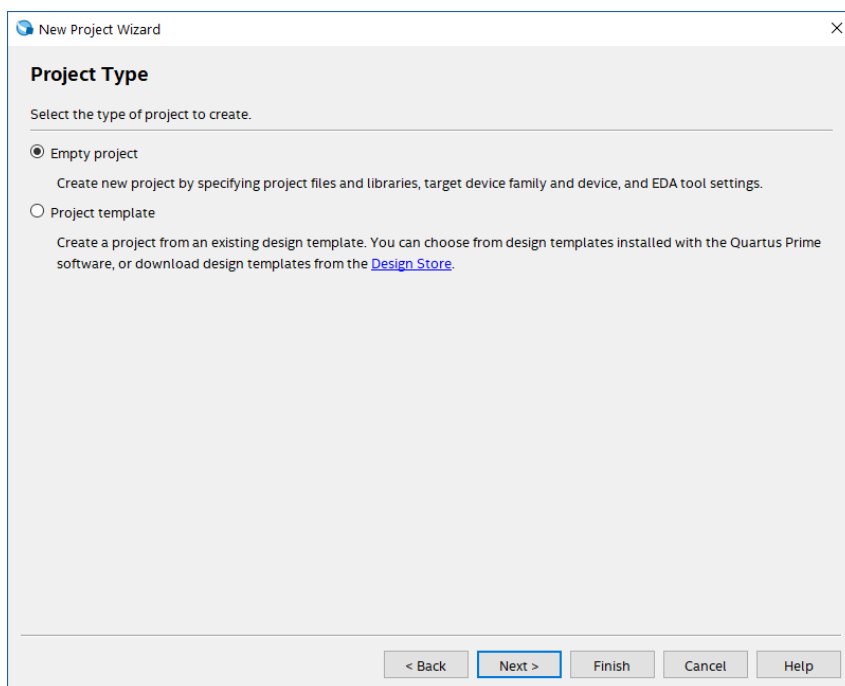
4.1.1.4 Configure the New Project Wizard directory, name and top-level entity information:


- Enter a directory in which you will store your Quartus project files for this design, for example, **C:/MAX1000/Internal\_Flash\_lab**
- Specify the name of the project: **nios\_subsystem**
- Specify the name of the top-level entity: **nios\_subsystem**

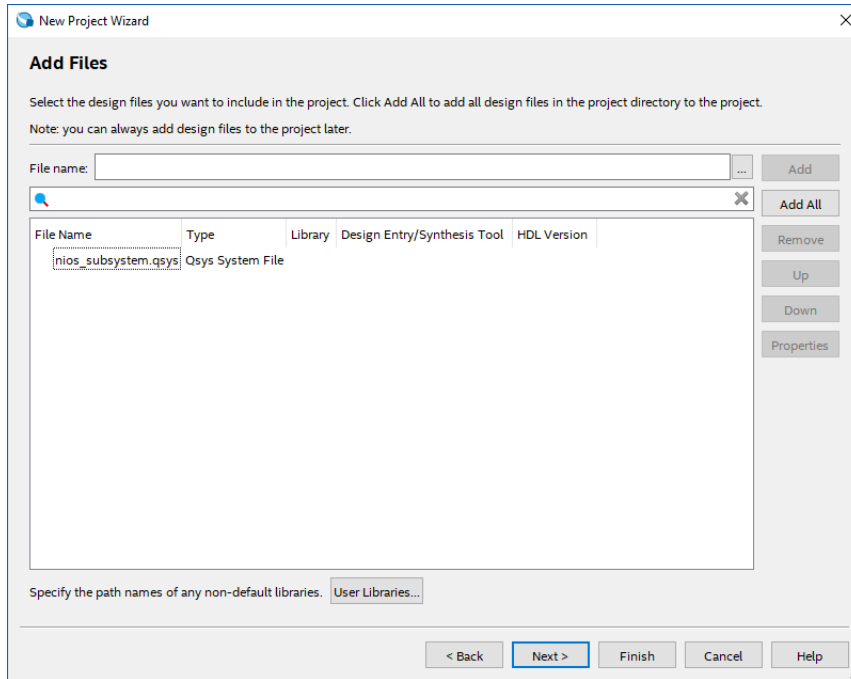


4.1.1.5 Click **Next**.

4.1.1.6 On the Project Type page, select “**Empty project**” and click **Next**.

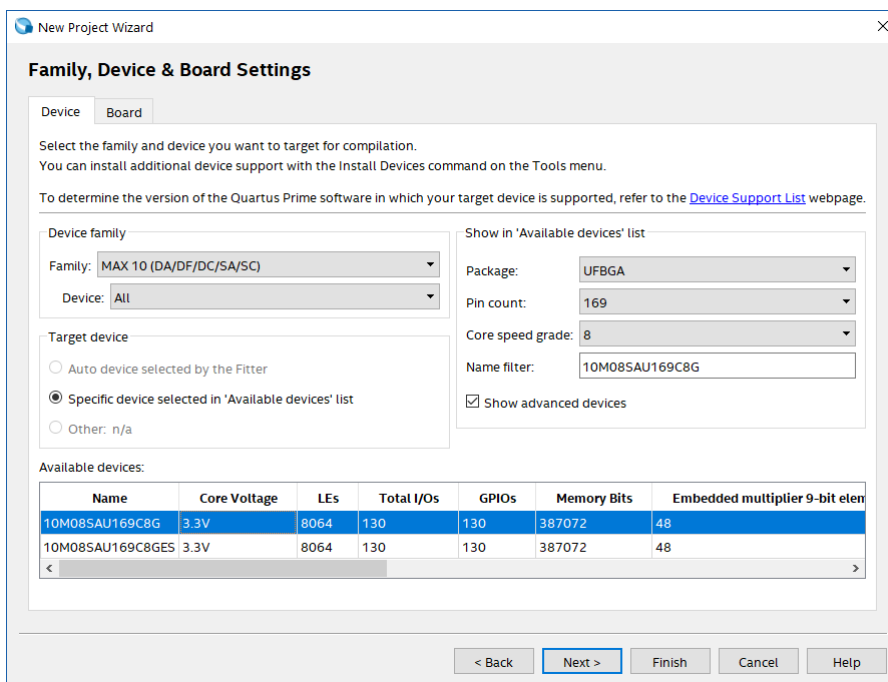


4.1.1.7 On the Add Files page, add source file to the project by clicking on the  button and browse into the lab files folder where you will locate the provided design files and add **nios\_subsystem.qsys**.



4.1.1.8 Click **Next**.

4.1.1.9 Specify Family and Device Settings. Use pull-down menus to select MAX10 family or enter the part number in the Name Filter text box. The part number is **10M08SAU169C8G**.



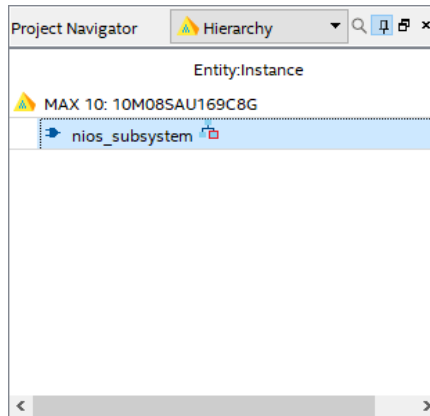
4.1.1.10 Click **Finish**.

## 4.2 Design entry

**Overview:** In this module you will generate the basic configuration file for the FPGA.

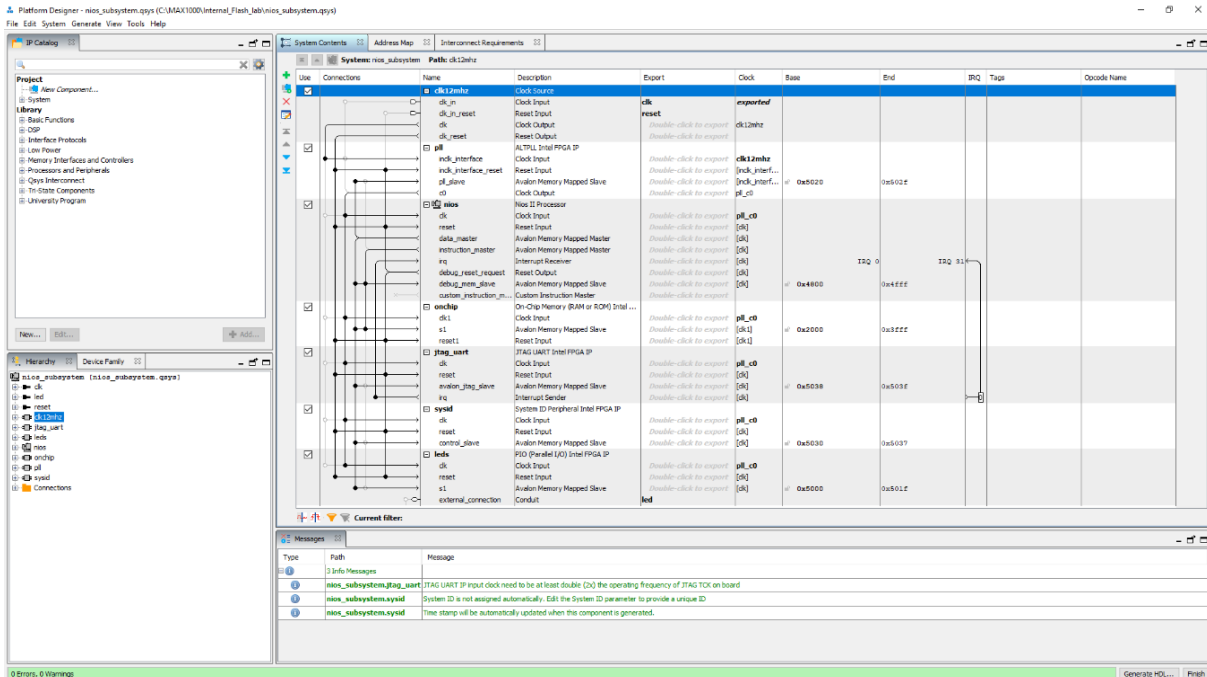
### 4.2.1 Platform Designer

#### 4.2.1.1 Double click on `nios_subsystem` top-level entity in Project Navigator.



#### 4.2.1.2 Platform Designer will open. The Nios processor system is already done, currently no need to change the design.

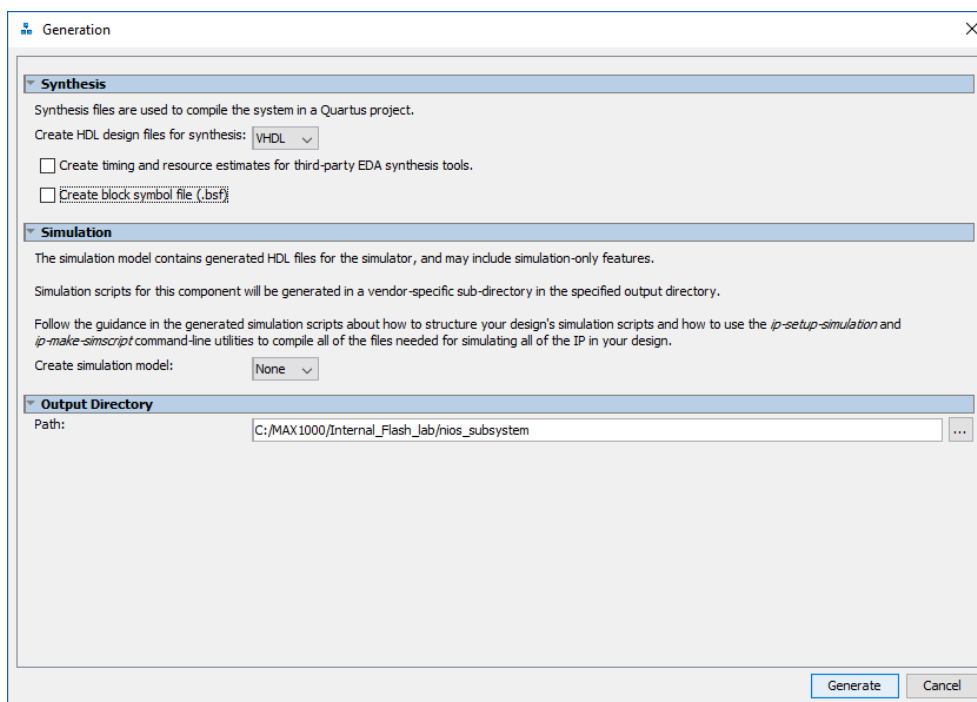
It is a simple design, which contains a 12MHz clock source, a pll which generates 50MHz for the system, a Nios II/e processor, on-chip RAM which will store the program code in the beginning, jtag-uart and system ID interface for the programming and a parallel I/O interface for the LEDs.



4.2.1.3 Select **Generate** → **Generate HDL...** from the menu or alternately click **Generate HDL...** button on the bottom right of the Platform Designer window.

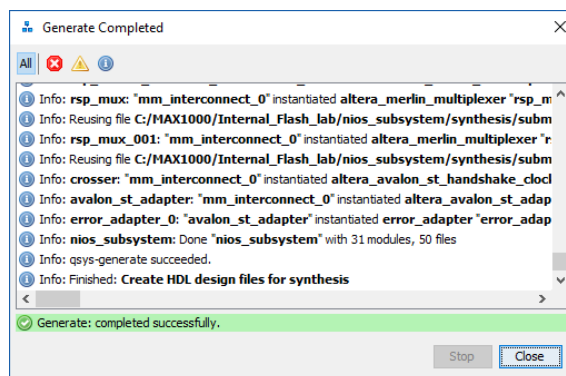
4.2.1.4 On the Generation window, enter the following information.

- Create HDL design files for synthesis: **VHDL**
- Uncheck Create timing and resource estimates for third-party EDA synthesis tools.
- Uncheck Create block symbol file (.bsf)
- Create simulation model: **None**



4.2.1.5 Click **Generate**.

4.2.1.6 When the generate process completed, click **Close**.



4.2.1.7 Click **Finish** button on the bottom right of the Platform Designer window.

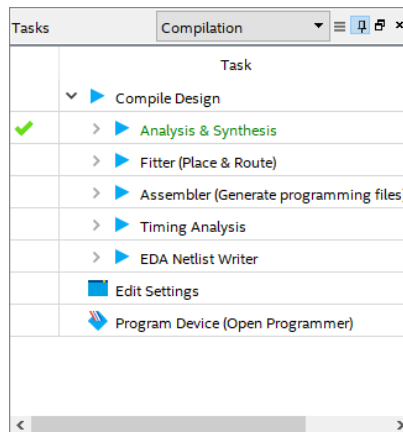


## 4.3 Compile design

### 4.3.1 Analysis and Synthesis

4.3.1.1 Run Analysis and Synthesis by clicking on button on the toolbars, or **Processing** → **Start** → **Analysis and Synthesis**.

There should be no errors. If there are errors, they should be fixed before continuing. If there are no errors the compilation task windows should look like this:



### 4.3.2 Pin Assignments

4.3.2.1 Open **Pin Planner** by clicking on button on the toolbars, or **Assignments** → **Pin Planner**.

Named	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservator
	altera_reserved_tck	Input				2.5 V (default)		12mA (default)			
	altera_reserved_tdi	Input				2.5 V (default)		12mA (default)			
	altera_reserved_tdo	Output				2.5 V (default)		12mA (default)	2 (default)		
	altera_reserved_tms	Input				2.5 V (default)		12mA (default)			
	clk_clk	Input				2.5 V (default)		12mA (default)			
	led_export[7]	Output				2.5 V (default)		12mA (default)	2 (default)		
	led_export[6]	Output				2.5 V (default)		12mA (default)	2 (default)		
	led_export[5]	Output				2.5 V (default)		12mA (default)	2 (default)		
	led_export[4]	Output				2.5 V (default)		12mA (default)	2 (default)		
	led_export[3]	Output				2.5 V (default)		12mA (default)	2 (default)		
	led_export[2]	Output				2.5 V (default)		12mA (default)	2 (default)		
	led_export[1]	Output				2.5 V (default)		12mA (default)	2 (default)		
	led_export[0]	Output				2.5 V (default)		12mA (default)	2 (default)		
	reset_reset_n	Input				2.5 V (default)		12mA (default)			



4.3.2.2 In the bottom table, type **PIN\_H6** in Location column of the clk\_clk.

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Tri-state Preservation
altera_reserved_tck	Input				2.5 V (default)		12mA (default)			
altera_reserved_tdi	Input				2.5 V (default)		12mA (default)			
altera_reserved_tdo	Output				2.5 V (default)		12mA (default)	2 (default)		
altera_reserved_tms	Input				2.5 V (default)		12mA (default)			
clk_clk	Input	PIN_H6	2	B2_NO	2.5 V (default)		12mA (default)			
led_export[7]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[6]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[5]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[4]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[3]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[2]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[1]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[0]	Output				2.5 V (default)		12mA (default)	2 (default)		
reset_reset_n	Input				2.5 V (default)		12mA (default)			

4.3.2.3 Repeat the previous step with the following assignments:

Node Name	Pin Location
altera_reserved_tck	PIN_G2
altera_reserved_tdi	PIN_F5
altera_reserved_tdo	PIN_F6
altera_reserved_tms	PIN_G1
led_export[0]	PIN_A8
led_export[1]	PIN_A9
led_export[2]	PIN_A11
led_export[3]	PIN_A10
led_export[4]	PIN_B10
led_export[5]	PIN_C9
led_export[6]	PIN_C10
led_export[7]	PIN_D8
reset_reset_n	PIN_E6

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Tri-state Preservation
altera_reserved_tck	Input	PIN_g2	1B	B1_NO	2.5 V (default)		12mA (default)			
altera_reserved_tdi	Input	PIN_f5	1B	B1_NO	2.5 V (default)		12mA (default)			
altera_reserved_tdo	Output	PIN_f6	1B	B1_NO	2.5 V (default)		12mA (default)	2 (default)		
altera_reserved_tms	Input	PIN_g1	1B	B1_NO	2.5 V (default)		12mA (default)			
clk_clk	Input	PIN_H6	2	B2_NO	2.5 V (default)		12mA (default)			
led_export[0]	Output	PIN_a8	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[1]	Output	PIN_a9	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[2]	Output	PIN_a11	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[3]	Output	PIN_a10	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[4]	Output	PIN_b10	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[5]	Output	PIN_c9	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[6]	Output	PIN_c10	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[7]	Output	PIN_d8	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
reset_reset_n	Input	PIN_e6	8	B8_NO	2.5 V (default)		12mA (default)			

4.3.2.4 Double click in the I/O Standard column for any pins to open a drop-down list and change the 2.5V (default) to the specific I/O standard.

Node Name	Pin I/O Standard
altera_reserved_tck	3.3-V LVTTTL
altera_reserved_tdi	3.3-V LVTTTL
altera_reserved_tdo	3.3-V LVTTTL
altera_reserved_tms	3.3-V LVTTTL
clk_clk	3.3-V LVTTTL
led_export[0]	3.3-V LVTTTL
led_export[1]	3.3-V LVTTTL
led_export[2]	3.3-V LVTTTL
led_export[3]	3.3-V LVTTTL
led_export[4]	3.3-V LVTTTL
led_export[5]	3.3-V LVTTTL
led_export[6]	3.3-V LVTTTL
led_export[7]	3.3-V LVTTTL
reset_reset_n	3.3-V Schmitt Trigger

Top View - Wire Bond  
MAX10 - 10M08SAU169C8G

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
altera_reserved_tck	Input	PN_G2	18	B1_NO	3.3-V LVTTTL		8mA (default)			
altera_reserved_tdi	Input	PN_F5	18	B1_NO	3.3-V LVTTTL		8mA (default)			
altera_reserved_tdo	Output	PN_R5	18	B1_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
altera_reserved_tms	Input	PN_A1	18	B1_NO	3.3-V LVTTTL		8mA (default)			
clk_clk	Input	PN_H6	2	B2_NO	3.3-V LVTTTL		8mA (default)			
led_export[0]	Output	PN_A8	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[1]	Output	PN_A9	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[2]	Output	PN_A11	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[3]	Output	PN_A10	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[4]	Output	PN_B10	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[5]	Output	PN_C9	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[6]	Output	PN_C10	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[7]	Output	PN_D8	6	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
reset_reset_n	Input	PN_E6	6	B8_NO	3.3 V Sc... Trigger		8mA (default)			

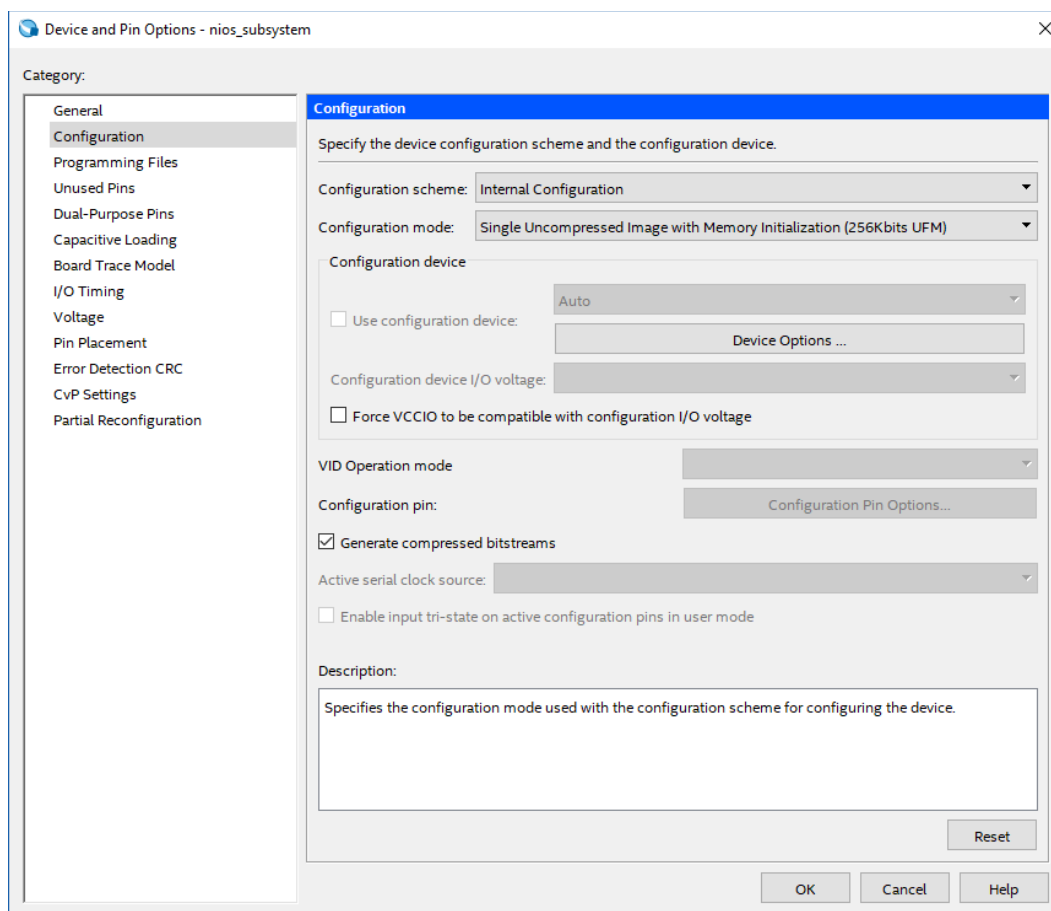
4.3.2.5 Close the Pin Planner, the settings are automatically saved.

### 4.3.3 Compiling the Design

4.3.3.1 Open the device settings window from **Assignments** → **Device...** and click **Device and Pin Options**.

4.3.3.2 Click to the **Configuration** category.

4.3.3.3 Set configuration mode to **Single Uncompressed Image with Memory Initialization (256kbits UFM)**.

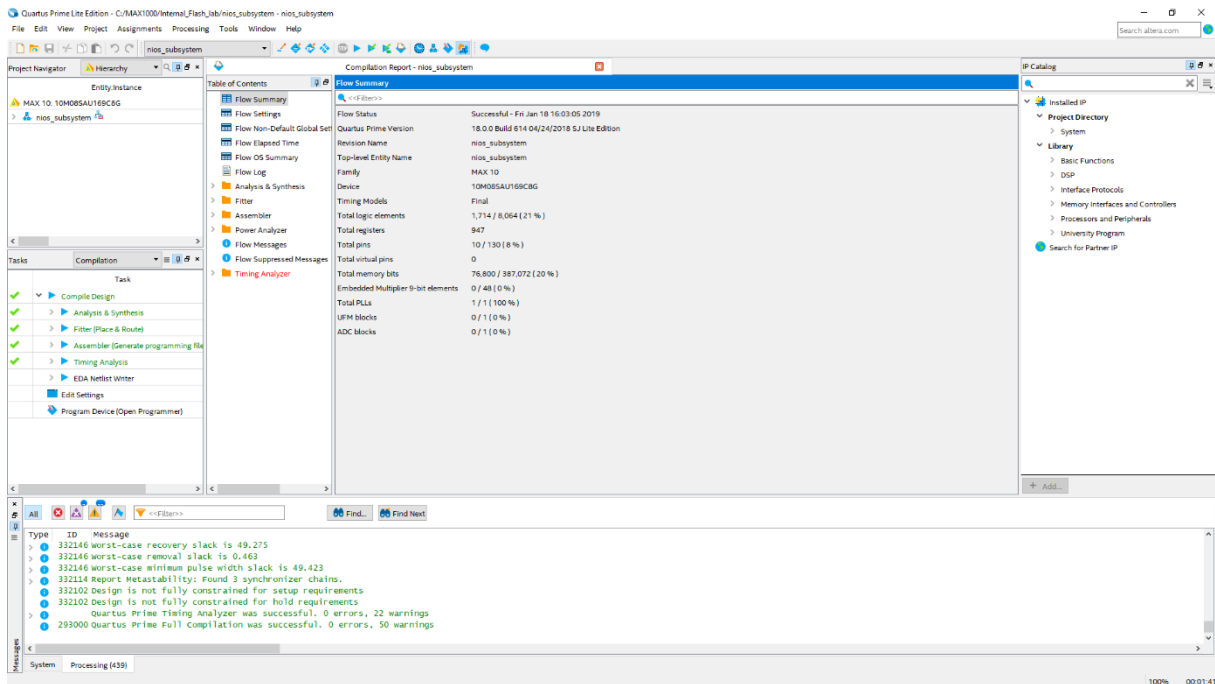


4.3.3.4 Press **OK** to close Device and Pin Options window. Press again **OK** to close Device window.

4.3.3.5 Start Compilation by clicking on  button on the toolbars, or **Processing** → **Start Compilation**.

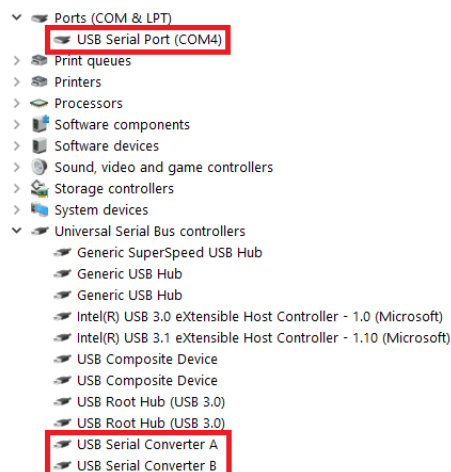


There should be no errors. If there are errors, they should be fixed before re-compiling. The 100% in the lower right corner or a green checkmark next to the Compile Design in the Compilation task window indicates that the compilation was successful.

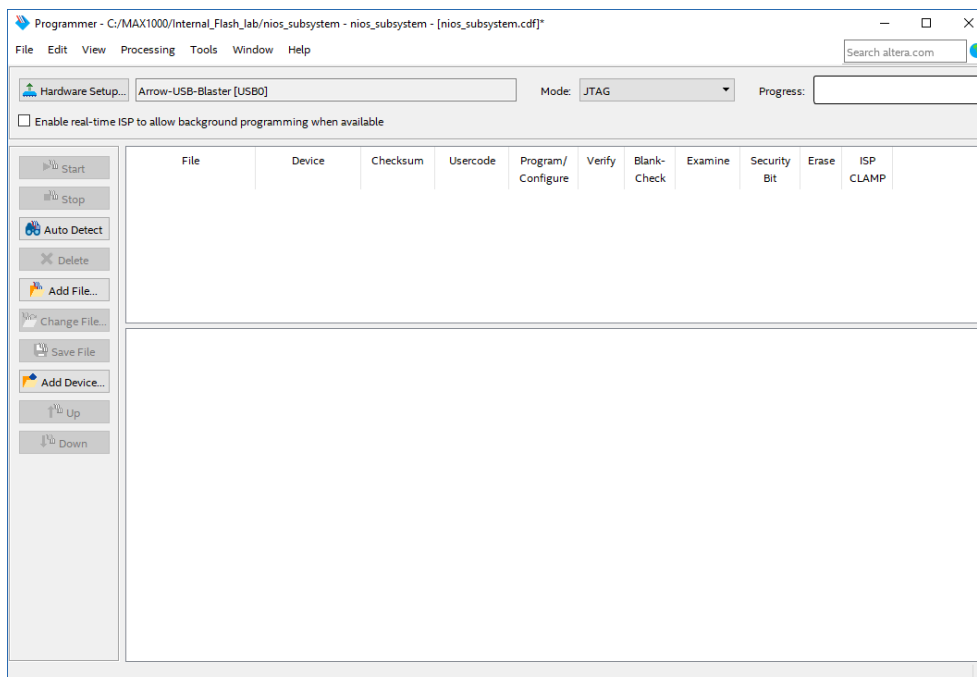


## 4.3.4 Configuration

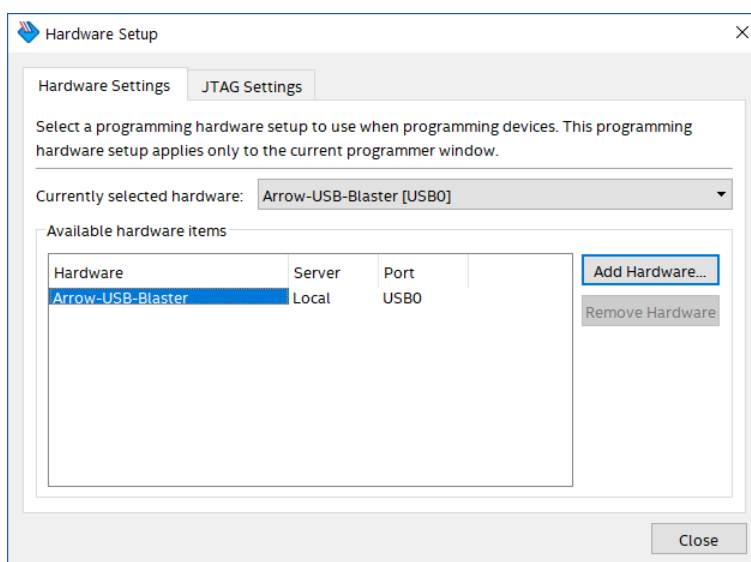
4.3.4.1 Connect your MAX1000 board to your PC using a USB cable. Since the Arrow USB Blaster should be already installed, the Window's Device Manager should display the following entries are highlighted in red (port number may differ depending on your PC):



4.3.4.2 Open the Quartus Prime Programmer from **Tools** → **Programmer** or double click on Program Device (Open Programmer) from the Task window.

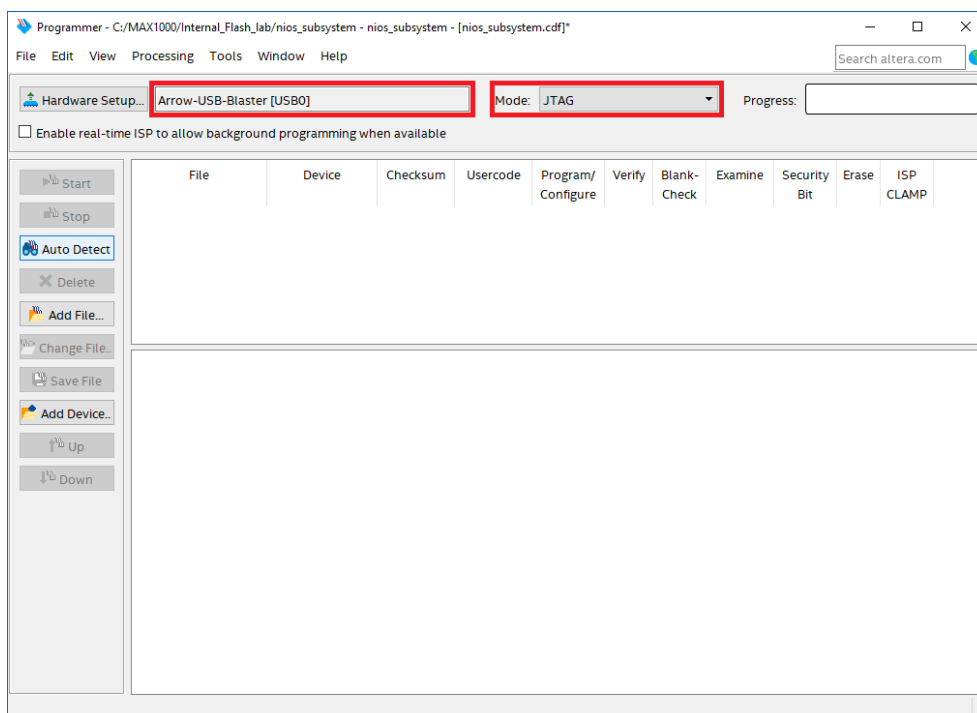


4.3.4.3 Click **Hardware Setup...** and double click **Arrow-USB-Blaster** entry in the Hardware Setup tab. The Currently selected hardware should now show Arrow-USB-Blaster [USB0] (depending on your PC, the USB port number may variant).



4.3.4.4 Click **Close**.

4.3.4.5 Make sure the hardware setup is Arrow-USB-Blaster [USB0] and the mode is JTAG. Click **Auto Detect**.



4.3.4.6 If the configuration has been added by default, you can skip the following steps and continue with the 4.3.4.11 point.

4.3.4.7 Select **10M08SA** device and click **OK** in the pop-up window.

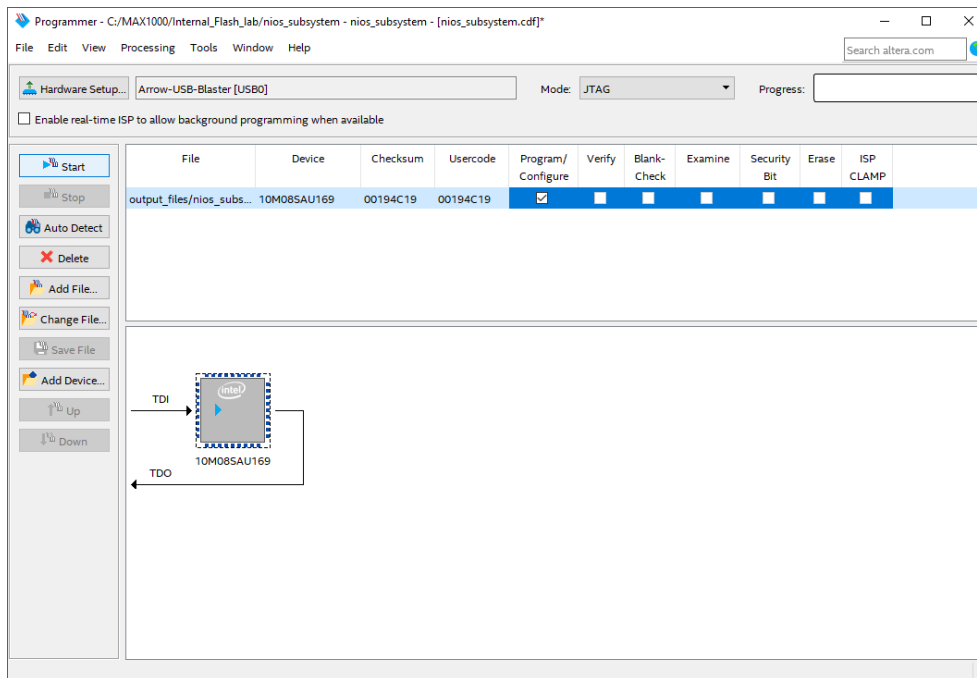


4.3.4.8 Click **Change File...** or double click <none> to choose the programming file.

4.3.4.9 Navigate to <project\_directory>/output\_files/ and select the **nios\_subsystem.sof** file.

4.3.4.10 Click **Open**.

4.3.4.11 Make sure the Programmer shows the correct file and the correct part in the JTAG chain and check the Program/Configure checkbox.



4.3.4.12 Click **Start** to program the board. When the configuration is complete, the Progress bar should show 100% (Successful).





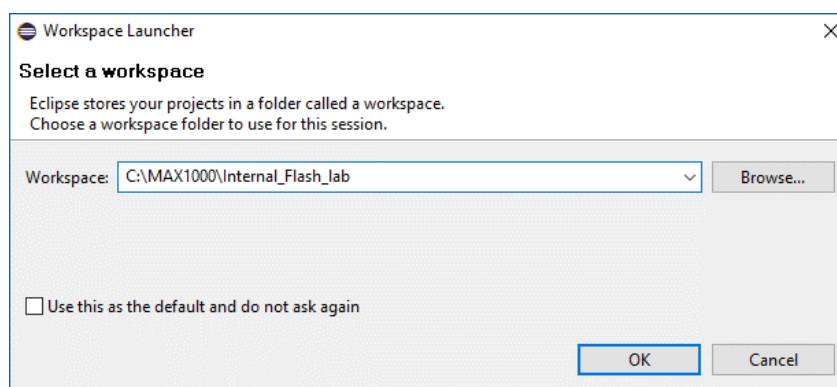
## 4.4 Software Design

**Overview:** In this section, you will use the Nios II Software Build Tools (SBT) for Eclipse to quickly create a board support package (BSP) and a C software application to run on the Nios II processor.

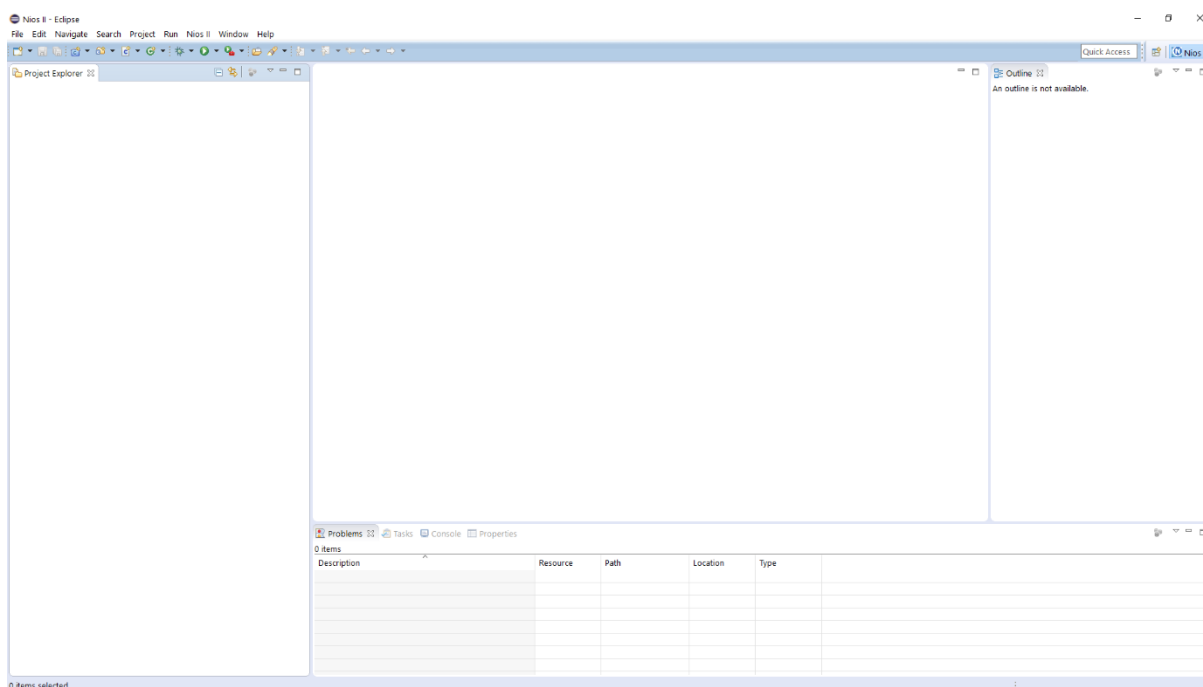
### 4.4.1 Create a new software project

4.4.1.1 From the main Quartus Prime window, start STB from **Tools → Nios II Software Build Tools for Eclipse**.

4.4.1.2 The Eclipse Workspace Launcher will open. Click **Browse...** and choose the directory of your project. In this case it was C:\MAX1000\Internal\_Flash\_lab.

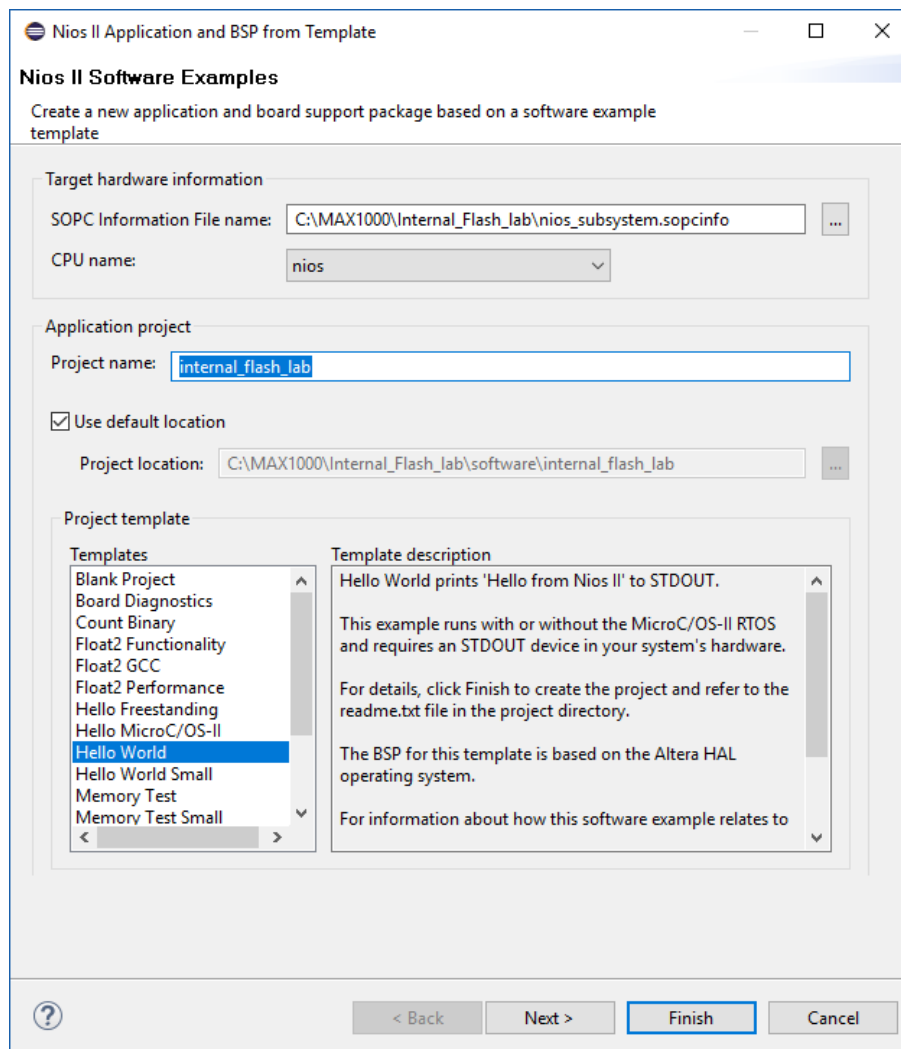


4.4.1.3 Click **OK** and the Eclipse will open.



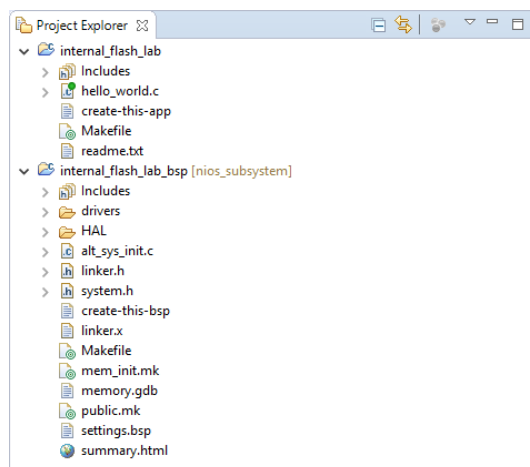
#### 4.4.1.4 Select **File** → **New** → **Nios II Application and BSP from Template**.

4.4.1.5 Click to select the **nios\_subsystem.sopcinfo** from your project directory and name the project **internal\_flash\_lab**. Make sure you select **Hello World** from the Templates.



4.4.1.6 Click **Finish**.

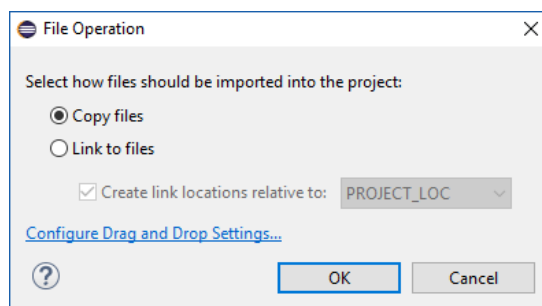
4.4.1.7 Eclipse will create two directories in the workspace, one for the application project and one for the BSP. The application directory (internal\_flash\_lab) contains a hello\_world.c file while the BSP directory (internal\_flash\_lab\_bsp) contains software drivers, a system.h, header file and other software infrastructure.



4.4.1.8 The C source file have been provided for you in this lab. Right click on hello\_world.c and **delete** it. In the Delete Resources window click **OK**.

4.4.1.9 From Windows Explorer, navigate to your main project directory. There you will find a file named **main.c** that you will need to copy to this project.

4.4.1.10 Select main.c file and drag it into the internal\_flash\_lab directory in Eclipse. Select Copy files option in the pop-up and click **OK**.



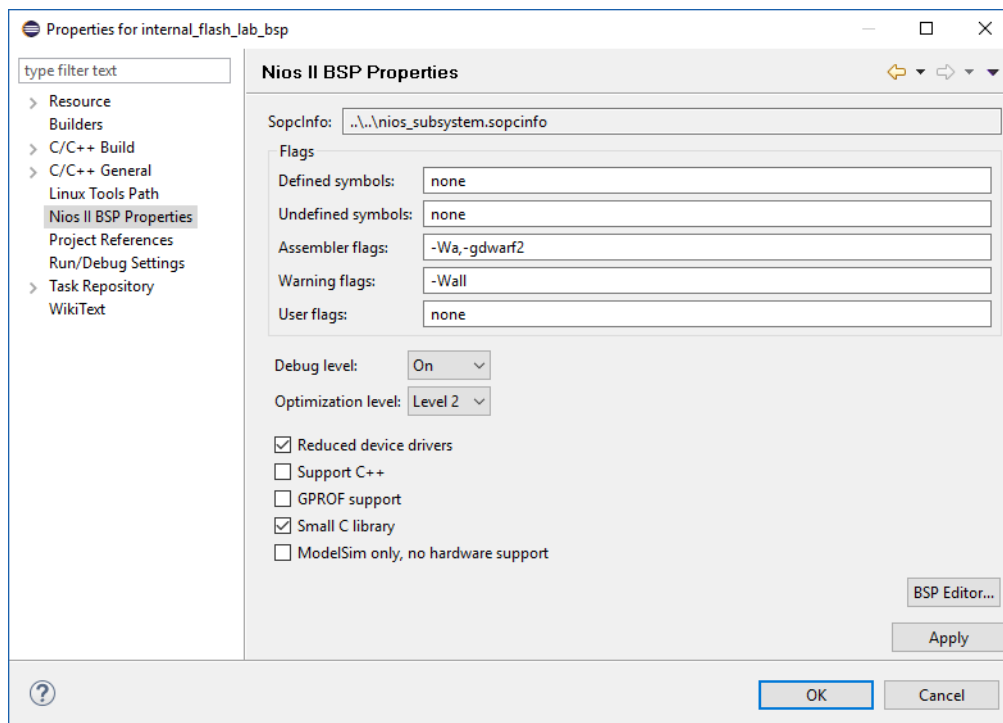
You should see the new file appear under the internal\_flash\_lab project in the Project Explorer.

## 4.4.2 Build the software

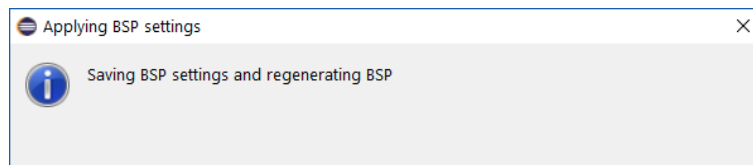
4.4.2.1 Right click on the internal\_flash\_lab\_bsp project and select **Properties** from the pop-up menu.

4.4.2.2 In the Properties window select the **Nios II BSP Properties** tab. It may take a moment to load the settings.

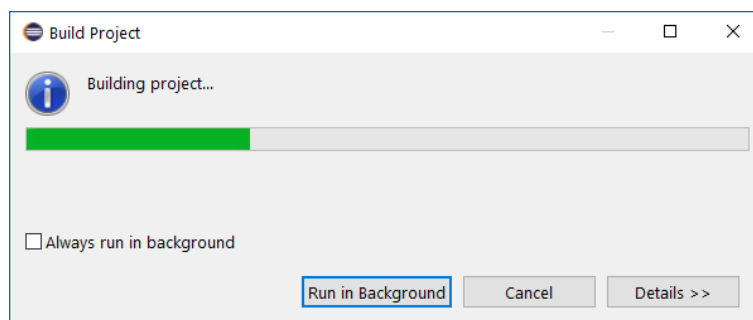
4.4.2.3 To keep the software footprint small so it fits our device, open the drop-down menu for Optimization level and change it to **Level 2**. Enable **Reduced device drivers** and **Small C library** options. As there is no C++ code, uncheck the Support C++ option.



4.4.2.4 Click **Apply**, and when it finished with Applying BSP Settings click **OK** to close Properties window.



4.4.2.5 Right click on the internal\_flash\_lab\_bsp project and select **Build Project** from the pop-up menu.



4.4.2.6 When it finished, repeat the previous step for the internal\_flash\_lab application project.

```

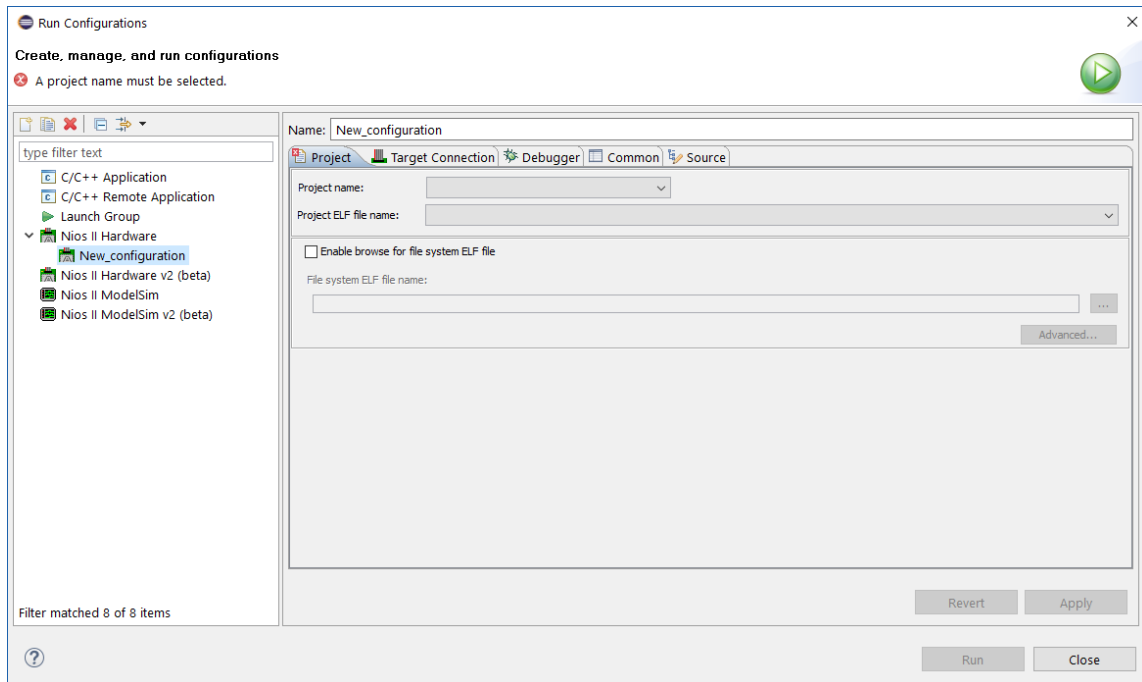
Problems Tasks Console Properties
CDT Build Console [internal_flash_lab]
nios2-elf-gcc -xO -I'./internal_flash_lab_bsp//linker.x' -msys-crt0='
Info: Linking internal_flash_lab.elf
nios2-elf-g++ -T'./internal_flash_lab_bsp//linker.x' -msys-crt0='
nios2-elf-insert internal_flash_lab.elf --thread_model hal --cpu_na
Info: (internal_flash_lab.elf) 4360 Bytes program size (code + init
Info: 3108 Bytes free for stack + heap.
Info: Creating internal_flash_lab.objdump
nios2-elf-objdump --disassemble --syms --all-header --source intern
[internal_flash_lab build complete]

16:25:32 Build Finished (took 4s.620ms)

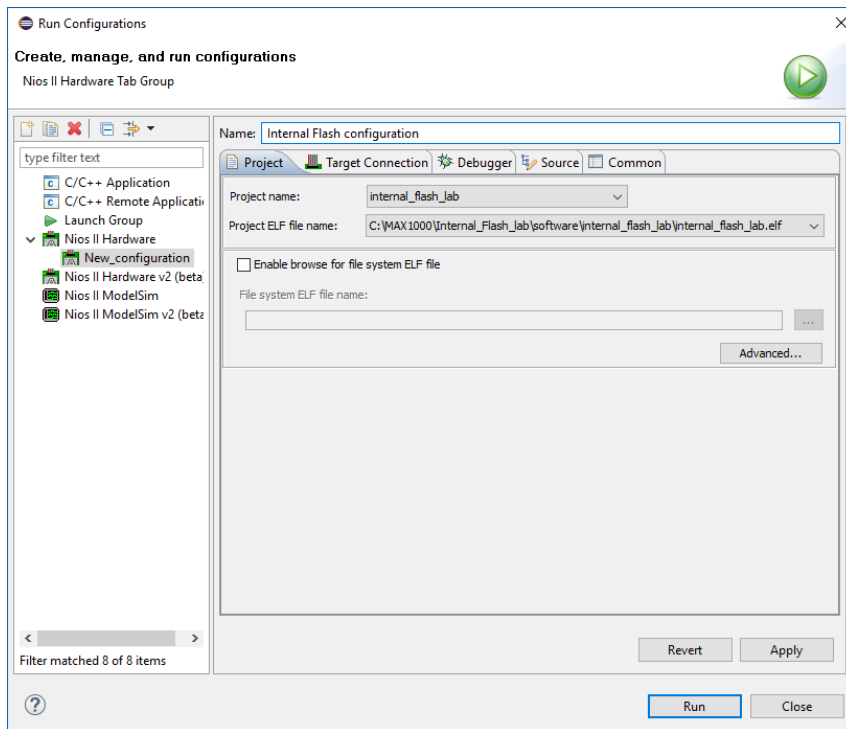
```

### 4.4.3 Run the application

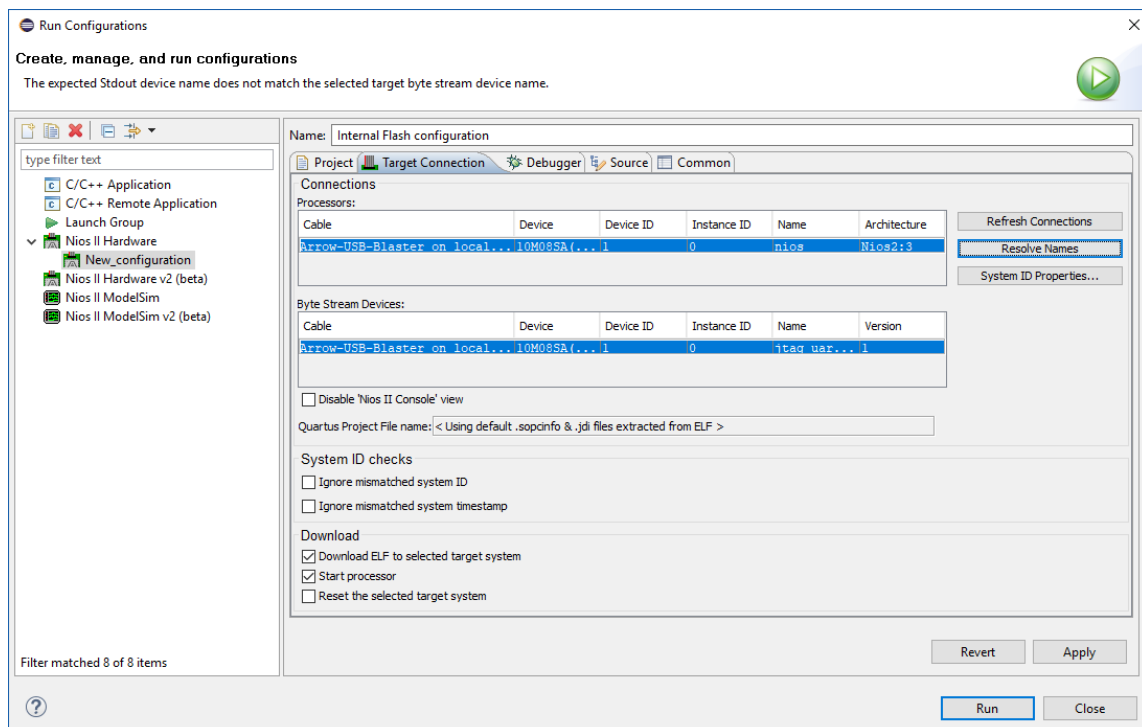
4.4.3.1 Select internal\_flash\_lab and go to **Run → Run Configurations...** and double click to **Nios II Hardware** to add a new configuration.



4.4.3.2 Rename it to **Internal Flash configuration** and on the Project tab select **internal\_flash\_lab** from the drop-down menu for the Project name.



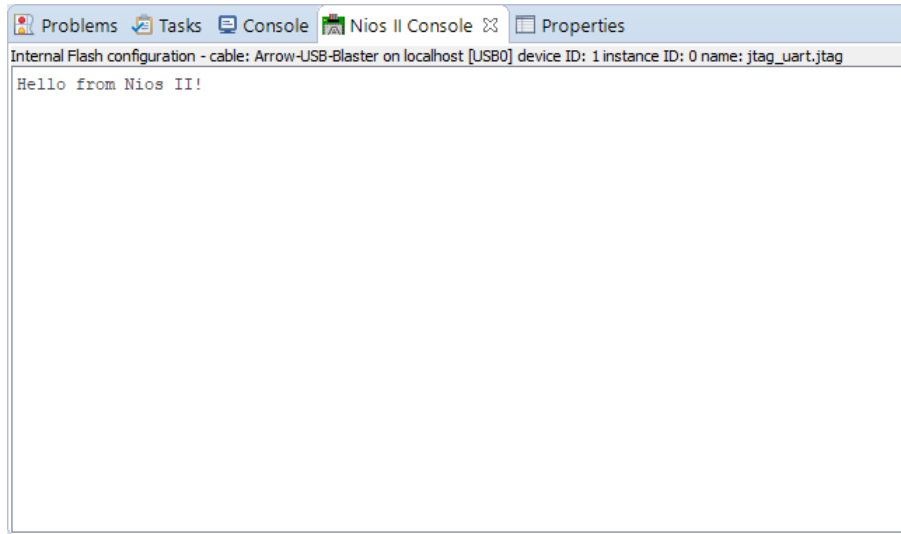
4.4.3.3 Click on the **Target Connection** tab and click **Refresh Connections** button. The configured MAX1000 board should appear.



4.4.3.4 Click **Apply** and **Run**.



4.4.3.5 After a few second, the Nios II Console should open at the bottom of the Eclipse and the LEDs should run on the MAX1000 board.

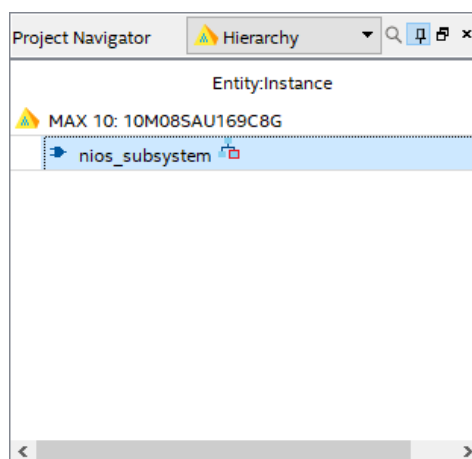


## 4.5 Program and boot On-Chip Flash

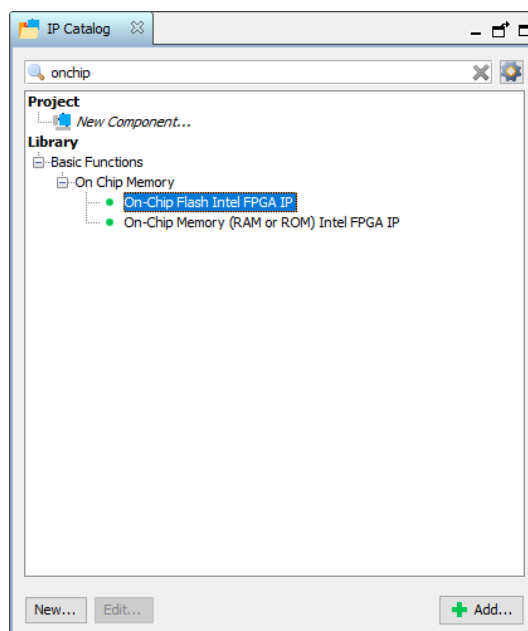
**Overview:** In this module you will modify the existing design to ensure that the Nios processor is able to boot from the internal Flash.

### 4.5.1 Hardware modification

4.5.1.1 In Quartus Prime, double click on **nios\_subsystem** top-level entity in Project Navigator. The Platform Designer will open.

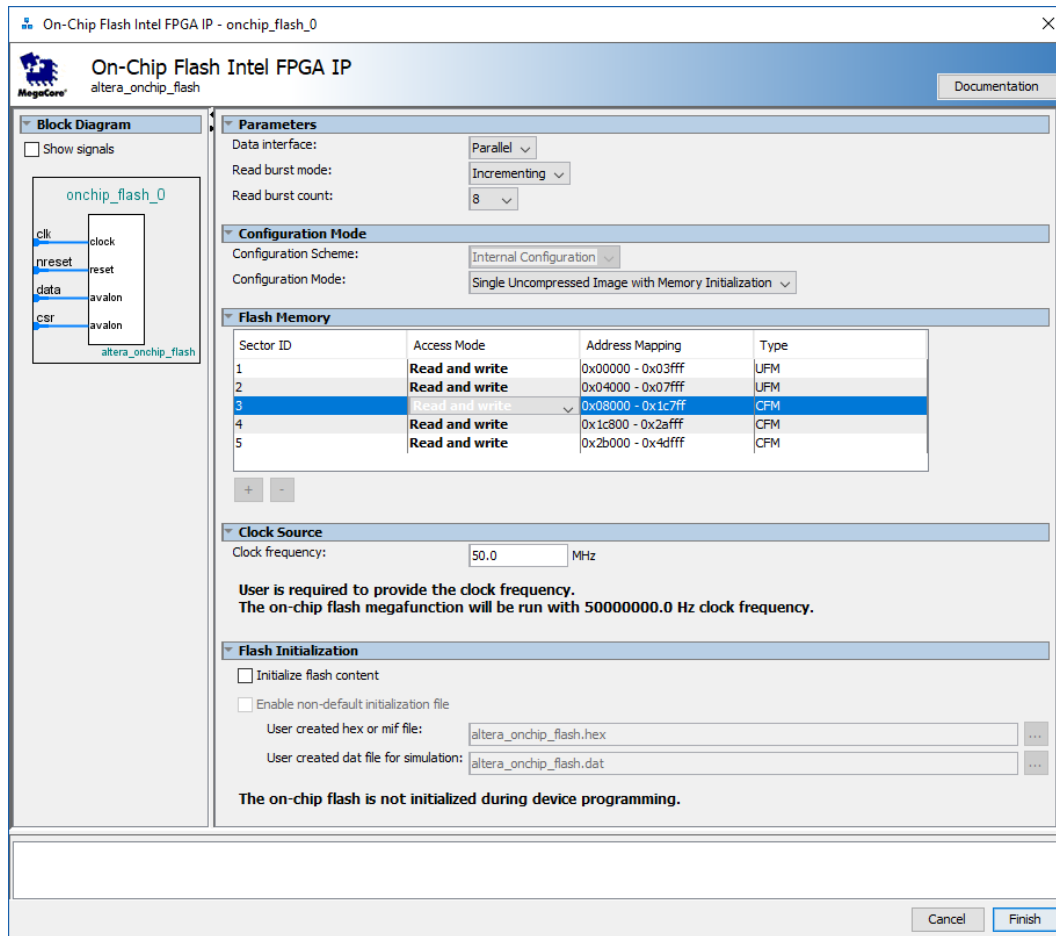


4.5.1.2 In the search bar of the IP Catalog, type “onchip,” and add **On-Chip Flash Intel FPGA IP**.





4.5.1.3 Change the Configuration Mode to **Single Uncompressed Image with Memory Initialization**, under Flash Memory change the Access Mode to **Read and write** for each Sector and set the Clock frequency to **50 MHz**.



4.5.1.4 Accept the defaults for the remaining fields and click **Finish** to add the component to the system.

4.5.1.5 Rename the new onchip\_flash\_0 component to **onchip\_flash**.

4.5.1.6 In the Connections column, hover over the connections and you will then be able to fill in dots to make the connections by clicking them.

4.5.1.7 Make the following connections:

Component A		Component B
onchip_flash   clk	↔	pll   c0
onchip_flash   data	↔	nios   data_master
onchip_flash   data	↔	nios   instruction_master
onchip_flash   csr	↔	nios   data_master

4.5.1.8 Automatically create global reset by selecting **System → Create Global Reset Network** from the menu.

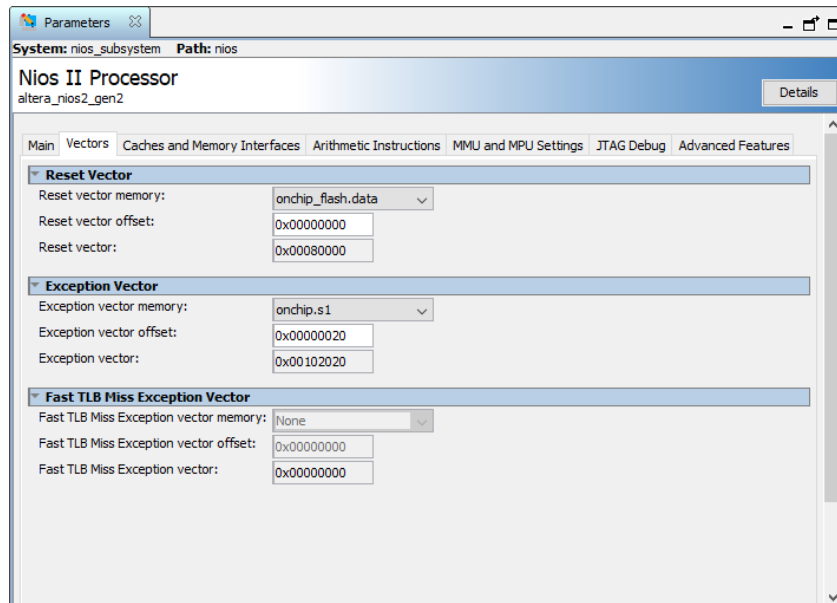
4.5.1.9 Automatically assign base addresses for the peripherals by selecting **System → Assign Base Addresses** from the menu.

4.5.1.10 Verify that your system is the same as below:

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name	
<input checked="" type="checkbox"/>		clk12mhz	Clock Source	clk12mhz	exported						
		dk_in	Clock Input	Double-click to export							
		dk_in_reset	Reset Input	Double-click to export							
		dk	Clock Output	Double-click to export		clk12mhz					
		dk_reset	Reset Output	Double-click to export							
<input checked="" type="checkbox"/>			pll	ALTRPLL Intel FPGA IP	Double-click to export						
		indk_interface	Clock Input	Double-click to export		clk12mhz					
		indk_interface_reset	Reset Input	Double-click to export		[dk]					
		pll_slave	Avalon Memory Mapped Slave	Double-click to export		[dk]	# 0x0010_5020	0x0010_502f			
		c0	Clock Output	Double-click to export		pll_c0					
<input checked="" type="checkbox"/>		nios	Nios II Processor	Double-click to export							
	dk	Clock Input	Double-click to export		pll_c0						
	reset	Reset Input	Double-click to export		[dk]						
	data_master	Avalon Memory Mapped Master	Double-click to export		[dk]						
	instruction_master	Avalon Memory Mapped Master	Double-click to export		[dk]						
	irq	Interrupt Receiver	Double-click to export		[dk]			IRQ 0	IRQ 31		
	debug_reset_request	Reset Output	Double-click to export		[dk]						
	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export		[dk]	# 0x0010_4800	0x0010_4fff				
	custom_instruction_m...	Custom Instruction Master	Double-click to export		[dk]						
<input checked="" type="checkbox"/>		onchip	On-Chip Memory (RAM or ROM) Intel ...	Double-click to export							
	dk1	Clock Input	Double-click to export		pll_c0						
	s1	Avalon Memory Mapped Slave	Double-click to export		[dk1]	# 0x0010_2000	0x0010_3fff				
	reset1	Reset Input	Double-click to export		[dk1]						
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART Intel FPGA IP	Double-click to export							
	dk	Clock Input	Double-click to export		pll_c0						
	reset	Reset Input	Double-click to export		[dk]						
	avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export		[dk]	# 0x0010_5040	0x0010_5047				
	irq	Interrupt Sender	Double-click to export		[dk]						
<input checked="" type="checkbox"/>		sysid	System ID Peripheral Intel FPGA IP	Double-click to export							
	dk	Clock Input	Double-click to export		pll_c0						
	reset	Reset Input	Double-click to export		[dk]						
	control_slave	Avalon Memory Mapped Slave	Double-click to export		[dk]	# 0x0010_5038	0x0010_503f				
<input checked="" type="checkbox"/>		leds	PID (Parallel I/O) Intel FPGA IP	Double-click to export							
	dk	Clock Input	Double-click to export		pll_c0						
	reset	Reset Input	Double-click to export		[dk]						
	s1	Avalon Memory Mapped Slave	Double-click to export		[dk]	# 0x0010_5000	0x0010_501f				
	external_connection	Conduit	Double-click to export		[dk]						
<input checked="" type="checkbox"/>		onchip_flash	On-Chip Flash Intel FPGA IP	Double-click to export							
	dk	Clock Input	Double-click to export		pll_c0						
	nreset	Reset Input	Double-click to export		[dk]						
	data	Avalon Memory Mapped Slave	Double-click to export		[dk]	# 0x0008_0000	0x0008_dfff				
	csr	Avalon Memory Mapped Slave	Double-click to export		[dk]	# 0x0010_5030	0x0010_5037				

4.5.1.11 Double click on the Nios II component **nios**. The Nios II Processor parameter editor will open.

4.5.1.12 Click on Vectors tab and set Reset Vector to **onchip\_flash.data**.



#### 4.5.1.13 Review message window for remains errors.

At this point should be no remaining errors in the message window. If there are, please refer again to the previous steps to resolve them.

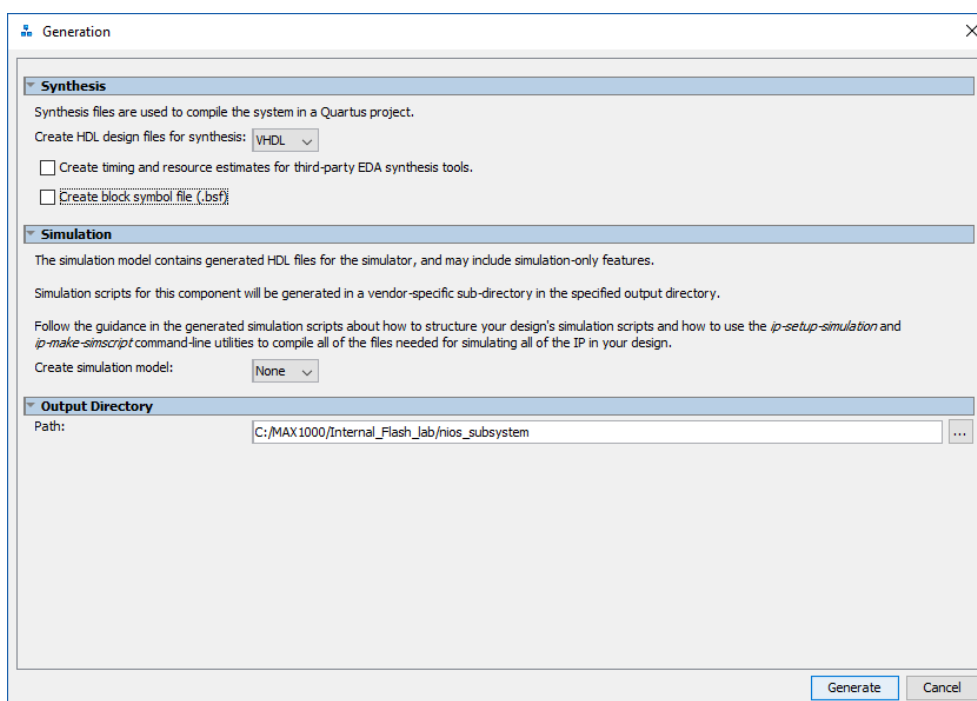
#### 4.5.1.14 Save your design.

### 4.5.2 Generate the Platform Designer System

4.5.2.1 Select **Generate** → **Generate HDL...** from the menu or alternately click **Generate HDL...** button on the bottom right of the Platform Designer window.

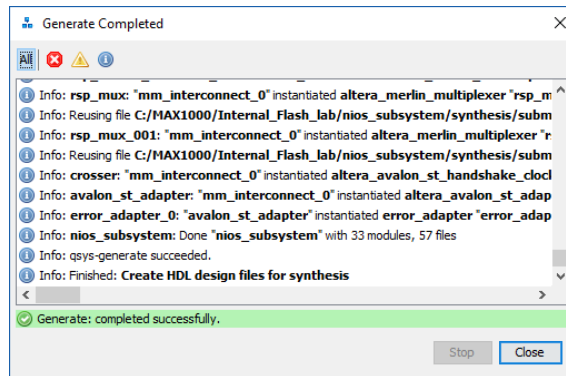
4.5.2.2 On the Generation window, enter the following information.

- Create HDL design files for synthesis: **VHDL**
- Uncheck Create timing and resource estimates for third-party EDA synthesis tools.
- Uncheck Create block symbol file (.bsf)
- Create simulation model: **None**




4.5.2.3 Click **Generate**.

4.5.2.4 When the generate process completed, click **Close**.

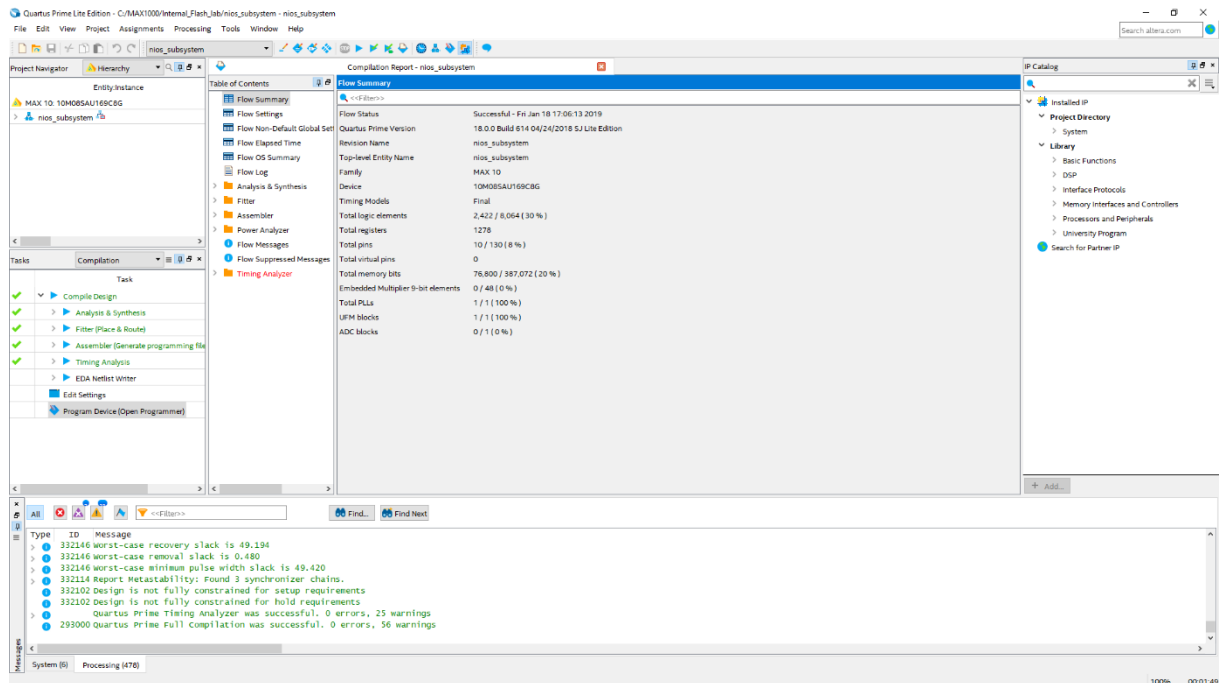


4.5.2.5 Click **Finish** button on the bottom right of the Platform Designer window.

## 4.5.3 Recompile

4.5.3.1 Start Compilation by clicking on  button on the toolbars, or **Processing** → **Start Compilation**.

There should be no errors. If there are errors, they should be fixed before re-compiling. The 100% in the lower right corner or a green checkmark next to the Compile Design in the Compilation task window indicates that the compilation was successful.

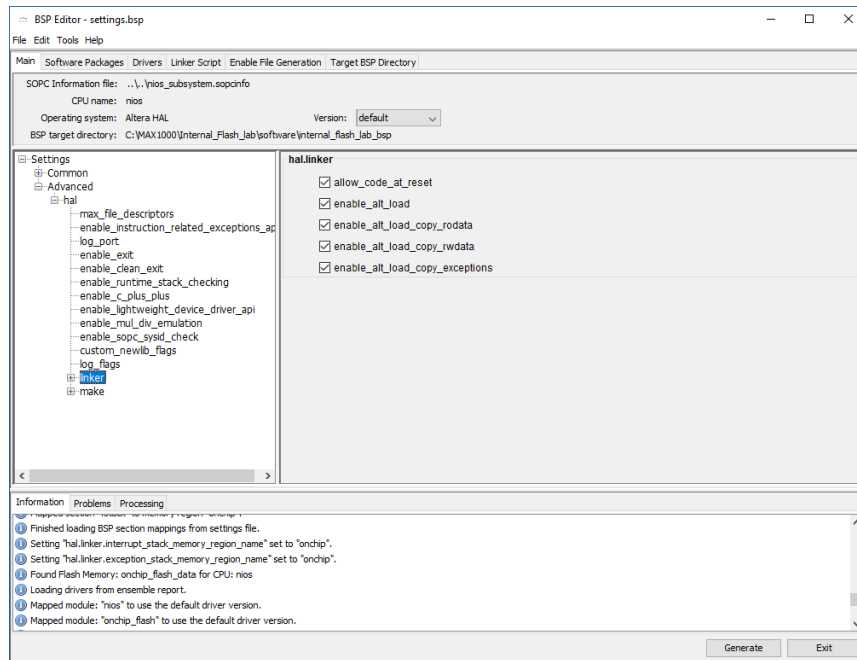


## 4.5.4 Software reconfiguration

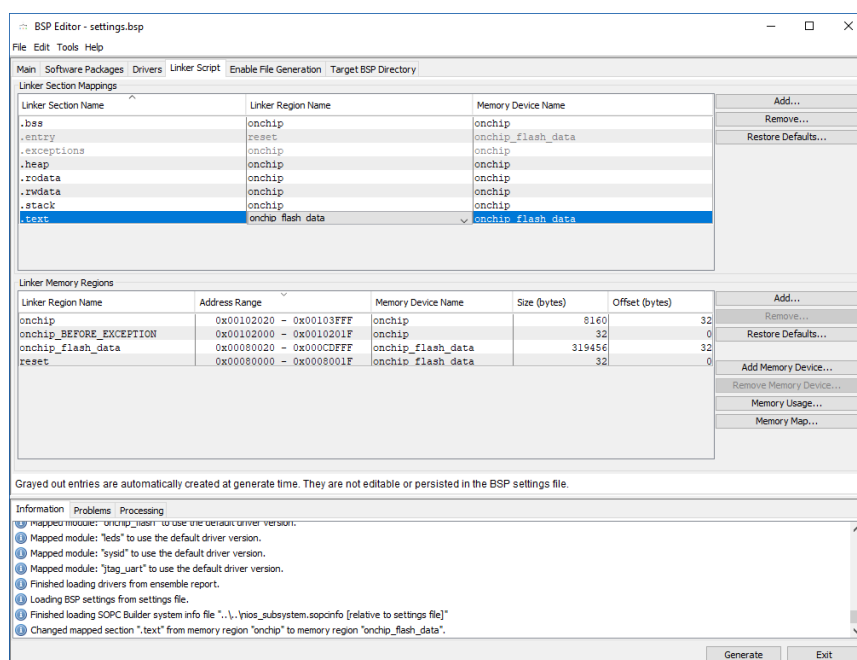
4.5.4.1 In the Eclipse, right click on `internal_flash_lab_bsp` and choose **Nios II → Generate BSP**.  
With this step you update the BSP files for the modified Nios system.

4.5.4.2 Right click on `internal_flash_lab_bsp` and select **Nios II → BSP Editor**.

4.5.4.3 On the left side of BSP Editor select **Settings/Advanced/hal/linker** and set all options.



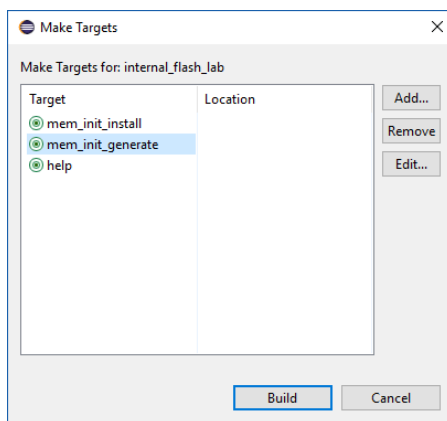
4.5.4.4 Click on the **Linker Script** tab and set `onchip_flash_data` from the drop-down menu for the Linker Region Name of `.text`.



4.5.4.5 Click **Generate** and when finished click **Exit**.

4.5.4.6 Right click on `internal_flash_lab` and choose **Make Targets** → **Build...**

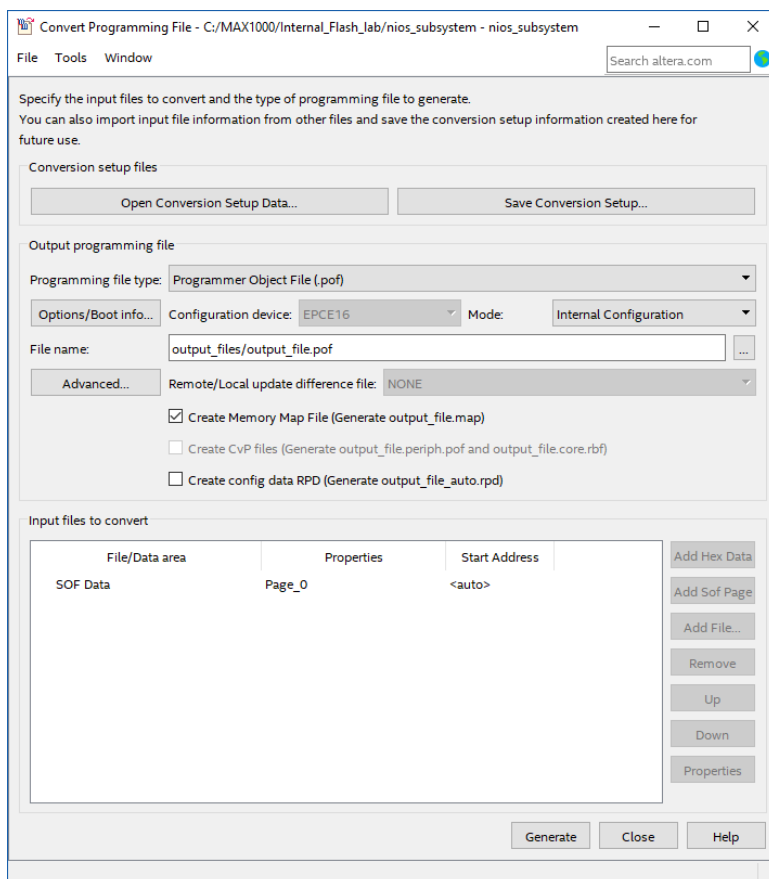
4.5.4.7 In the Make Targets window select `mem_init_generate` and click **Build**.



## 4.5.5 Configuration image


4.5.5.1 In Quartus Prime, go to **File** → **Convert Programming File**.

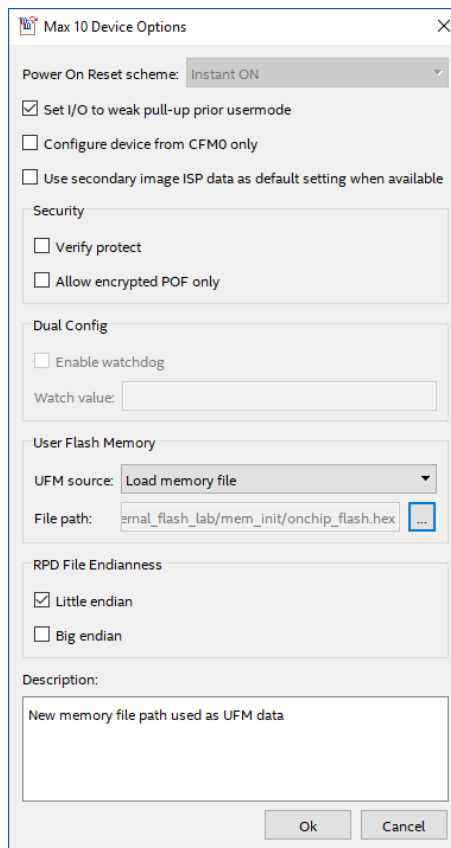
4.5.5.2 In the Convert Programming File window set Mode to **Internal Configuration**.



4.5.5.3 Click on **Options/Boot info...**

4.5.5.4 In the MAX10 Device Options window set **Load memory file** from the drop-down menu for UFM source.

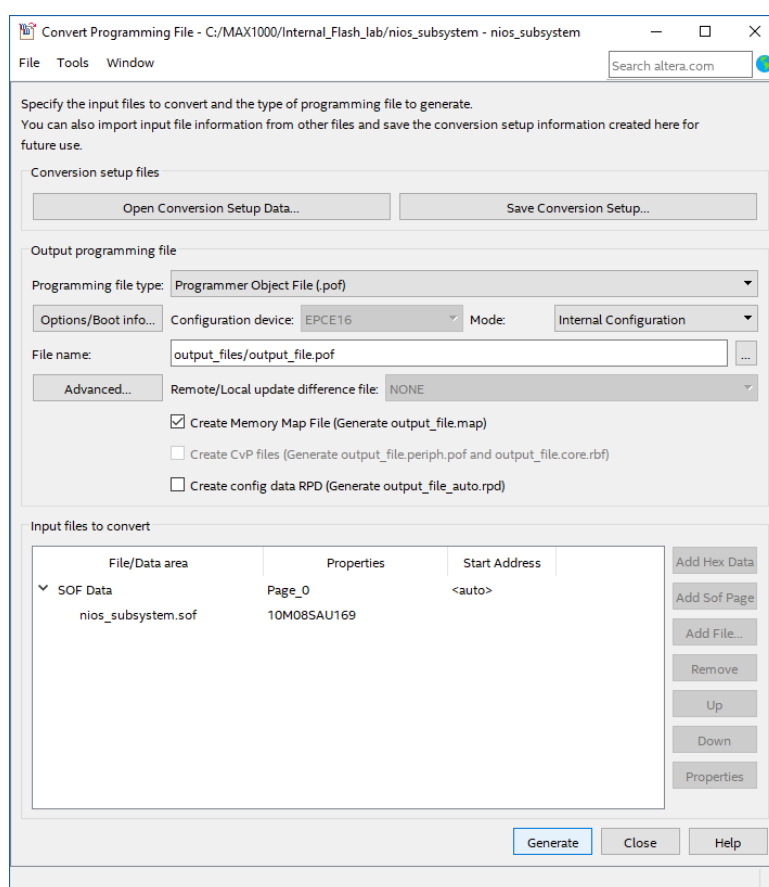
4.5.5.5 Click on the  button for the File path and browse through the memory initialization directory:  
**<project\_directory>/software/internal\_flash\_lab/mem\_init** and open **onchip\_flash.hex**



4.5.5.6 Click **OK**.

4.5.5.7 Under the Input files to convert Click on **SOB DATA** and then click on **Add File...**

4.5.5.8 Go to: <project\_directory>/output\_files/ and open nios\_subsystem.sof.



4.5.5.9 Click **Generate**.

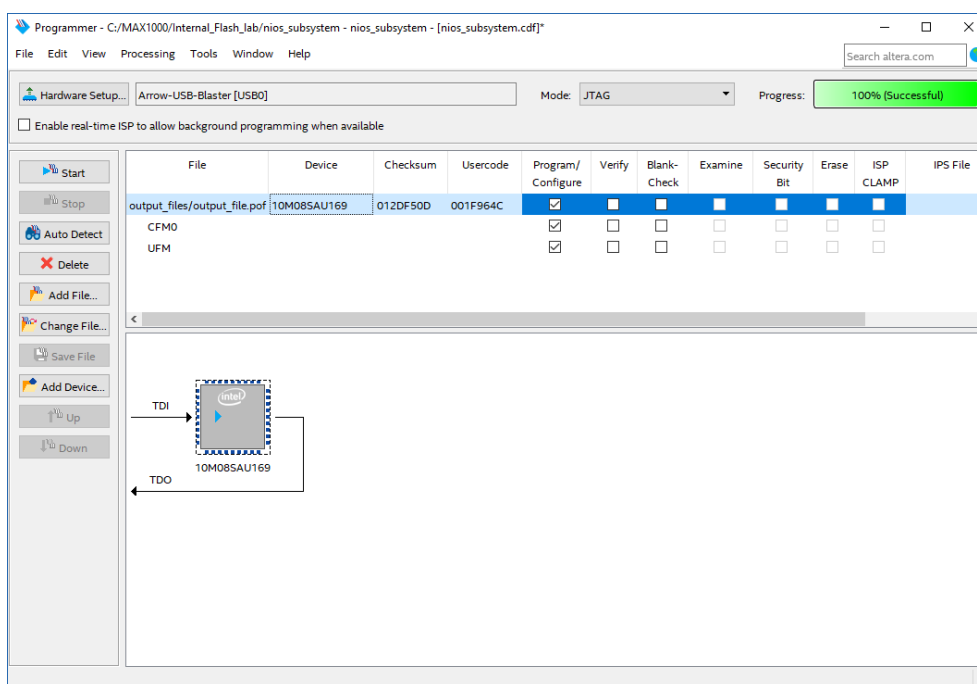
4.5.5.10 Click **OK** in the pop-up window and **close** the Convert Programming File window.

4.5.5.11 Open the Programmer window.

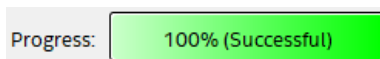
4.5.5.12 Click on **Change File...** button and open **output\_file.pof**.



#### 4.5.5.13 Check the Program/Configure checkbox for the .pof file, CFM0 and UFM.



4.5.5.14 Click **Start** to program the board. When the configuration is complete, the Progress bar should show 100% (Successful).



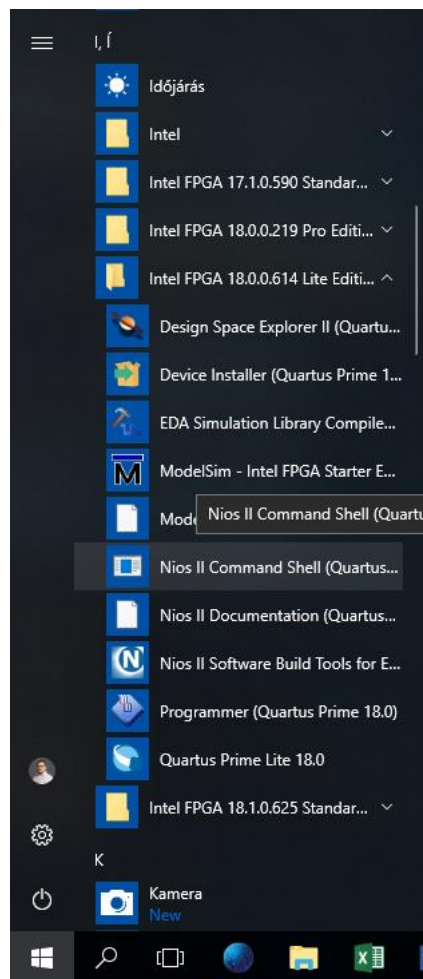
After this point, the FPGA configuration file and the program code for Nios soft processor are stored in the internal Flash memory of MAX10. When you power-on this board, this configuration file will load automatically.

### 4.5.6 Testing the project

4.5.6.1 **Close** Programmer, Quartus Prime and Nios II Eclipse.

4.5.6.2 Remove MAX1000 board from your PC in order to power-off the device. In this way we can check the non-volatile configuration.

4.5.6.3 Open Nios II Command Shell from the **Start menu** → **All Programs** → **Intel FPGA 18.0.0.614 Lite Edition**. This command line tool is useful for a range of activities, from board and system-level debugging to programming an FPGA configuration file.



4.5.6.4 Connect again your MAX1000 board to your PC.

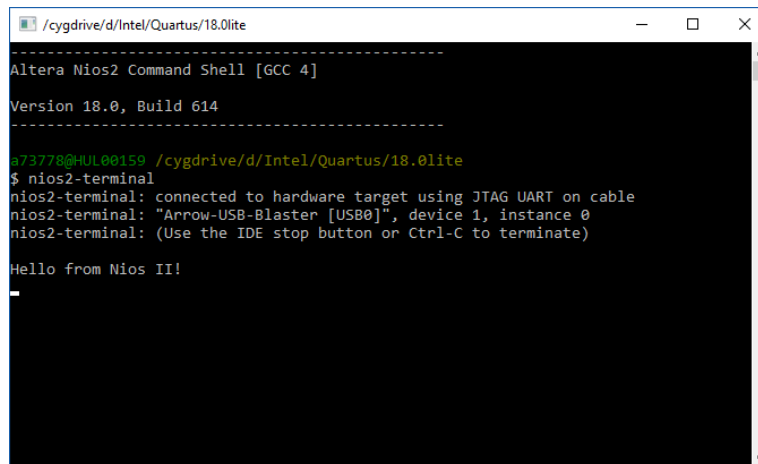
4.5.6.5 In the terminal window type the following command and press enter:

**nios2-terminal**

This command establishes contact with stdin, stdout, and stderr in the Nios II processor subsystem and allows you to provide input and monitor them. These standard streams are routed through the JTAG UART module within this system.



4.5.6.6 In the terminal window you should see 'Hello from Nios II!' that we saw in the Eclipse Nios terminal. Every time, when you press the Reset button this sentence is printed.



```
-----  
Altera Nios2 Command Shell [GCC 4]  
-----  
Version 18.0, Build 614  
-----  
a73778@HUL00159 /cygdrive/d/Intel/Quartus/18.0lite  
$ nios2-terminal  
nios2-terminal: connected to hardware target using JTAG UART on cable  
nios2-terminal: "Arrow-USB-Blaster [USB0]", device 1, instance 0  
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)  
  
Hello from Nios II!  
-
```

**CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE INTERNAL  
FLASH LAB!**



## 5 Revision History

Version	Change Log	Date of Change
V1.0	Initial Version	18/01/2019



## 6 Legal Disclaimer

### ARROW ELECTRONICS

#### EVALUATION BOARD LICENSE AGREEMENT

By using this evaluation board or kit (together with all related software, firmware, components, and documentation provided by Arrow, "Evaluation Board"), You ("You") are agreeing to be bound by the terms and conditions of this Evaluation Board License Agreement ("Agreement"). Do not use the Evaluation Board until You have read and agreed to this Agreement. Your use of the Evaluation Board constitutes Your acceptance of this Agreement.

#### PURPOSE

The purpose of this evaluation board is solely intended for evaluation purposes. Any use of the Board beyond these purposes is on your own risk. Furthermore, according the applicable law, the offering Arrow entity explicitly does not warrant, guarantee or provide any remedies to you with regard to the board.

#### LICENSE

Arrow grants You a non-exclusive, limited right to use the enclosed Evaluation Board offering limited features only for Your evaluation and testing purposes in a research and development setting. Usage in a live environment is prohibited. The Evaluation Board shall not be, in any case, directly or indirectly assembled as a part in any production of Yours as it is solely developed to serve evaluation purposes and has no direct function and is not a finished product.

#### EVALUATION BOARD STATUS

The Evaluation Board offers limited features allowing You only to evaluate and test purposes. The Evaluation Board is not intended for consumer or household use. You are not authorized to use the Evaluation Board in any production system, and it may not be offered for sale or lease, or sold, leased or otherwise distributed for commercial purposes.

#### OWNERSHIP AND COPYRIGHT

Title to the Evaluation Board remains with Arrow and/or its licensors. This Agreement does not involve any transfer of intellectual property rights ("IPR) for evaluation board. You may not remove any copyright or other proprietary rights notices without prior written authorization from Arrow or it licensors.

#### RESTRICTIONS AND WARNINGS

Before You handle or use the Evaluation Board, You shall comply with all such warnings and other instructions and employ reasonable safety precautions in using the Evaluation Board. Failure to do so may result in death, personal injury, or property damage.

You shall not use the Evaluation Board in any safety critical or functional safety testing, including but not limited to testing of life supporting, military or nuclear applications. Arrow expressly disclaims any responsibility for such usage which shall be made at Your sole risk.

#### WARRANTY

Arrow warrants that it has the right to provide the evaluation board to you. This warranty is provided by Arrow in lieu of all other warranties, written or oral, statutory, express or implied, including any warranty as to merchantability, non-infringement, fitness for any particular purpose, or uninterrupted or error-free operation, all of which are expressly disclaimed. The evaluation board is provided "as is" without any other rights or warranties, directly or indirectly.

You warrant to Arrow that the evaluation board is used only by electronics experts who understand the dangers of handling and using such items, you assume all responsibility and liability for any improper or unsafe handling or use of the evaluation board by you, your employees, affiliates, contractors, and designers.





#### **LIMITATION OF LIABILITIES**

In no event shall Arrow be liable to you, whether in contract, tort (including negligence), strict liability, or any other legal theory, for any direct, indirect, special, consequential, incidental, punitive, or exemplary damages with respect to any matters relating to this agreement. In no event shall arrow's liability arising out of this agreement in the aggregate exceed the amount paid by you under this agreement for the purchase of the evaluation board.

#### **IDENTIFICATION**

You shall, at Your expense, defend Arrow and its Affiliates and Licensors against a claim or action brought by a third party for infringement or misappropriation of any patent, copyright, trade secret or other intellectual property right of a third party to the extent resulting from (1) Your combination of the Evaluation Board with any other component, system, software, or firmware, (2) Your modification of the Evaluation Board, or (3) Your use of the Evaluation Board in a manner not permitted under this Agreement. You shall indemnify Arrow and its Affiliates and Licensors against and pay any resulting costs and damages finally awarded against Arrow and its Affiliates and Licensors or agreed to in any settlement, provided that You have sole control of the defense and settlement of the claim or action, and Arrow cooperates in the defense and furnishes all related evidence under its control at Your expense. Arrow will be entitled to participate in the defense of such claim or action and to employ counsel at its own expense.

#### **RECYCLING**

The Evaluation Board is not to be disposed as an urban waste. At the end of its life cycle, differentiated waste collection must be followed, as stated in the directive 2002/96/EC. In all the countries belonging to the European Union (EU Dir. 2002/96/EC) and those following differentiated recycling, the Evaluation Board is subject to differentiated recycling at the end of its life cycle, therefore: It is forbidden to dispose the Evaluation Board as an undifferentiated waste or with other domestic wastes. Consult the local authorities for more information on the proper disposal channels. An incorrect Evaluation Board disposal may cause damage to the environment and is punishable by the law.