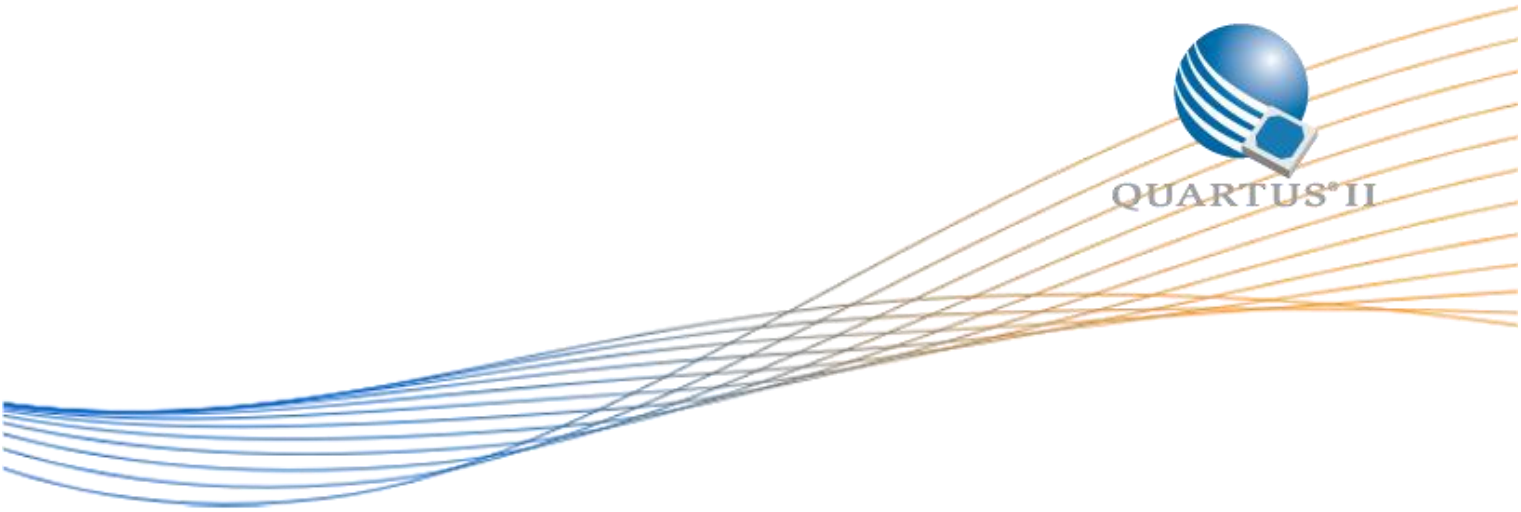


Using the On-Die Temperature Sensor in MAX10 FPGA using NIOS II

7/15/2015	Initial Release	L. Landis
4/14/2016	Removed master_image. C code was updated for stability.	L. Landis
9/28/2016	C Code portion for ADC sequencer was modified.	A. Bacye

This design example uses NIOS II and a Qsys interconnect to demonstrate how to read the on-die temperature sensor on the MAX10 FPGA Evaluation Kit. It also demonstrates a simple parallel IO read/write interface.

You can extend this program to use other ADC channels also by correspondingly modifying the ADC module in Qsys file. Refer to MAX10 ADC User Guide for further information.

This example performs the following:

I. Reads pushbuttons 1, 2 and 3 and displays them on LEDs 1, 2 and 3.
II. The ADC reads only the Temperature Sensing Diode (TSD) Channel. A sequencer is configured such that it reads the TSD channel 64 times per second. An average of the 64 samples is performed in the Nios II Processor.
III. A look up table is used to convert the ADC Raw value to Celsius and the Temperature in Celsius will be displayed in the console.

Design Overview

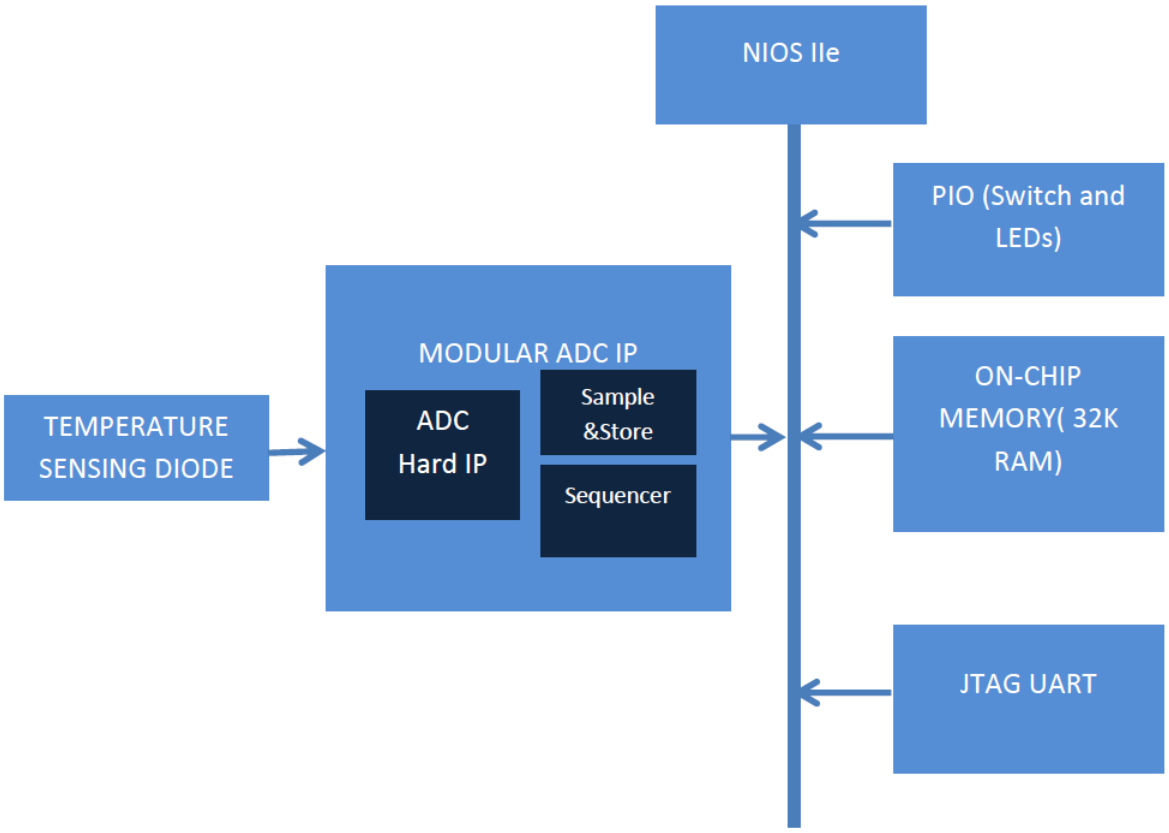


Figure 1 : Qsys System

The ADC reads the Temperature Sensing Diode (TSD) Channel. The sequencer is configured such that it reads the TSD channel in all the available 64 slots. In the Nios II processor, an average of all the 64 samples is taken every second. The ADC Raw data is then converted into Celsius value using a lookup table function and displayed in the Eclipse Console. Note that 64 samples are averaged to filter out spurious core noise behavior seen on the MAX10 FPGA Evaluation Kit since the ADC is highly sensitive to core noise and the ADC and core in this design share a common supply. A better filtered power supply would produce more consistent readings, but for this kit, averaging 64 measurements every second produces a fairly wide variation in results. Variation on the MAX 10 Eval Kit over 64 averaged values can range from ~ +/- 10C.

The NIOS II also reads the values from the (DIP) switches 1, 2 and 3 and displays them in the LEDs 1, 2 and 3 correspondingly. Note that there is a 1 second delay in the while loop, so expect a 1 second delay from flipping the switch to showing up on the LED.

In order to run this example you will need the following software:

- Quartus II Programmer , Version 14.1 or Later
- The Altera Nios II Embedded Design Suite (EDS) (This is required since we are displaying the Temperature in the Nios Terminal)

- Quartus II , version 14.1 or Later (This is required only if you want to modify or recompile the example)

Steps **to recreate the output files :**
If you want to edit the project and create your own version, use the following procedure:

Hardware

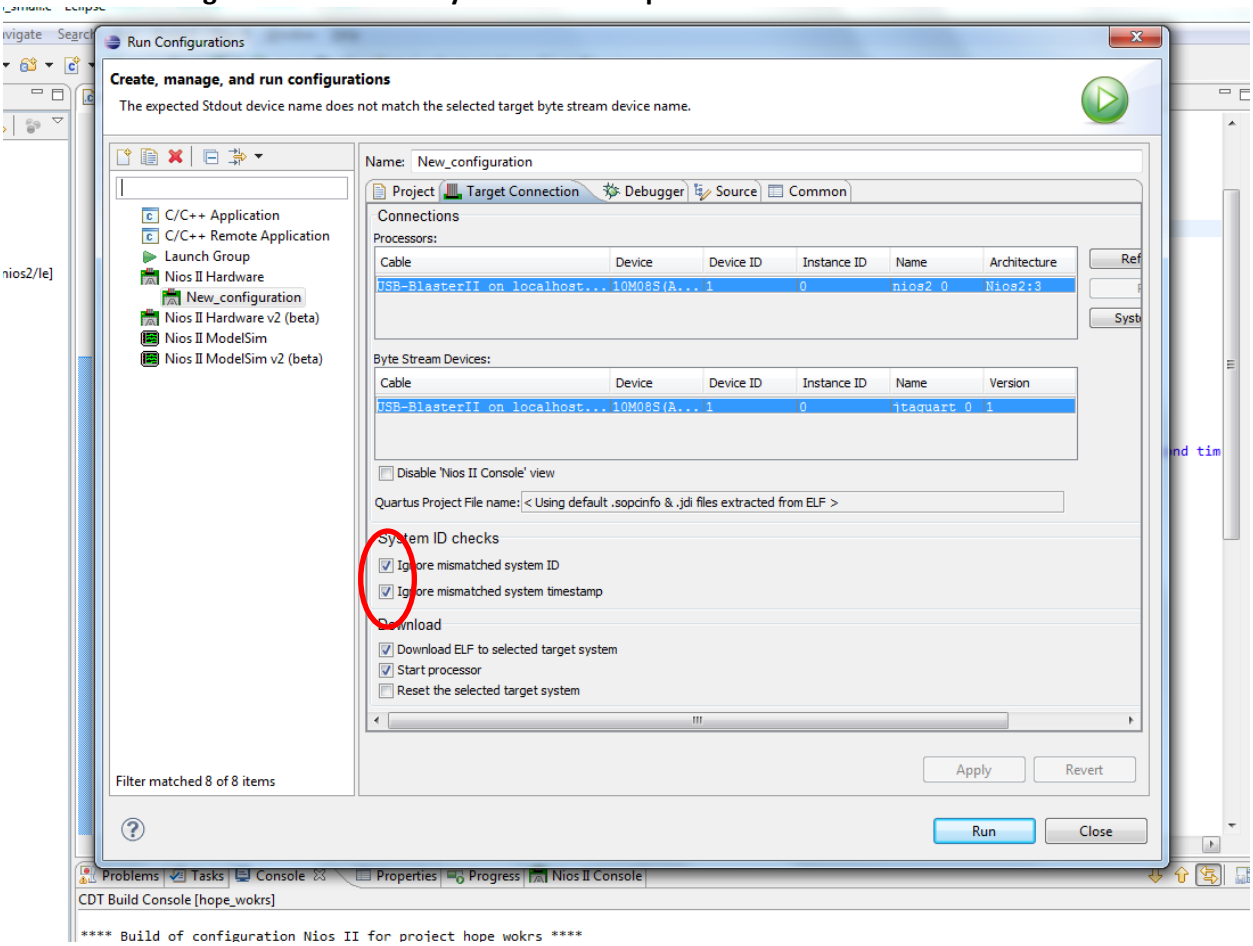
1) Launch Quartus II and open the top.qpf project (File -> Open Project -> top.qpf -> Open).
2) nios_setup.qsys is the Qsys system which contains all the modules shown in Figure 1 .
3) In case you make any modifications on the qsys file, make sure to reassign base addresses and interrupt numbers before generating the hardware description language (HDL).
4) Compile the modified project (Processing -> Start Compilation).
5) Program the ./output_files/top.sof to the board using Quartus II Programmer (Tools ->
6) Programmer - > Add File -> top.sof -> OK -> Start).
7) If you want get the .pof , use **Convert Programming Files** from File menu and follow a similar procedure to configure from Flash versus configure from RAM.

Software

1) Open **Tools->Nios II Software build tools** from Quartus II. This launches the NIOS II Eclipse IDE where you can modify your C Code.
2) Select the workspace for Nios II Eclipse. Then select **File->New->Nios II application and BSP from Template**
3) In the Window which opens, select the nios_setup.sopcinfo file in your Quartus project folder. The .sopcinfo file contains information about the Qsys system, each module instantiated in the project, and parameter names and values contained in the project.
4) Give the name to your nios project as temp_sensor, select hello world small as the project template and click finish. You can see that temp_sensor and temp_sensor_bsp is created on your workspace.
5) Now we have to add ondie_temp.c to the project
    a. In the Project Navigator window, select hello_world_small.c under temp_sensor. Right click to delete.
    b. Right click temp_sensor and select “Import” -> “File System”
    c. Navigate to <project_name>/software. Click OK.
    d. Select ondie_temp.c and click Finish. The file is now imported to the project.
6) Let us turn off the compiler optimization, since compiler optimization may cause some problems while running the program.
    a. Right click on the project (temp_sensor) and select **Properties.** Select **Nios II Application Properties** and Change the **Optimization level** to off. Click OK.
    b. Now, Right click on the project_bsp (temp_sensor_bsp) and select **Properties.** Select **Nios II BSP Properties** and Change the **Optimization level** to off. Click OK.
7) Right Click on the temp_sensor and Select **Build Project** to build the project. The initial build may take some time.

8) Once the build is finished, to run the project, right click on the project and select Run As -> Run Configurations.
9) Double click on Nios II Hardware, and new configuration opens on the right panel. Make sure you select the project name as temp_sensor and .elf file as temp_sensor.elf.
10) Select Target Connection Tab. Then Check the following two check boxes

Ignore mismatched system ID
Ignore mismatched system timestamp



11) Double check that your device is showing in Connections under both Processors and Byte Stream Devices. Refresh connections if needed.
12) Click Apply and Run

You will observe the following output in the console:

```
*******PIO and On-Die Temperature Sensor example********
Change Switches 1, 2 and 3 to change LEDs 1, 2 and 3
The value of ADC Channel connected to Temperature Sensing Diode is collected every second and is averaged over 64 Samples
---------------------------------------------------------------------------------------------
On-die temperature = 25
```

The On-die Temperature will be displayed every second in the console.

NOTE: Accuracy of the TSD is +/- 10 C. Refer to MAX10 FPGA Device Datasheet for more information.

Additional Information

JTAG Signals - Unconstrained Path violation

Many in-system debugging tools use the JTAG interface in Altera FPGAs. When you debug your design with the JTAG interface, the JTAG signals TCK, TMS, TDI, and TDO are implemented as part of the design. Because of this, the TimeQuest analyzer flags these signals as unconstrained when an unconstrained path report is generated. TCK is constrained by default while the other 3 signals need to be constrained in the .sdc file manually.

To remove the unconstrained path Timing Violations, add the following constraints in your .sdc file.

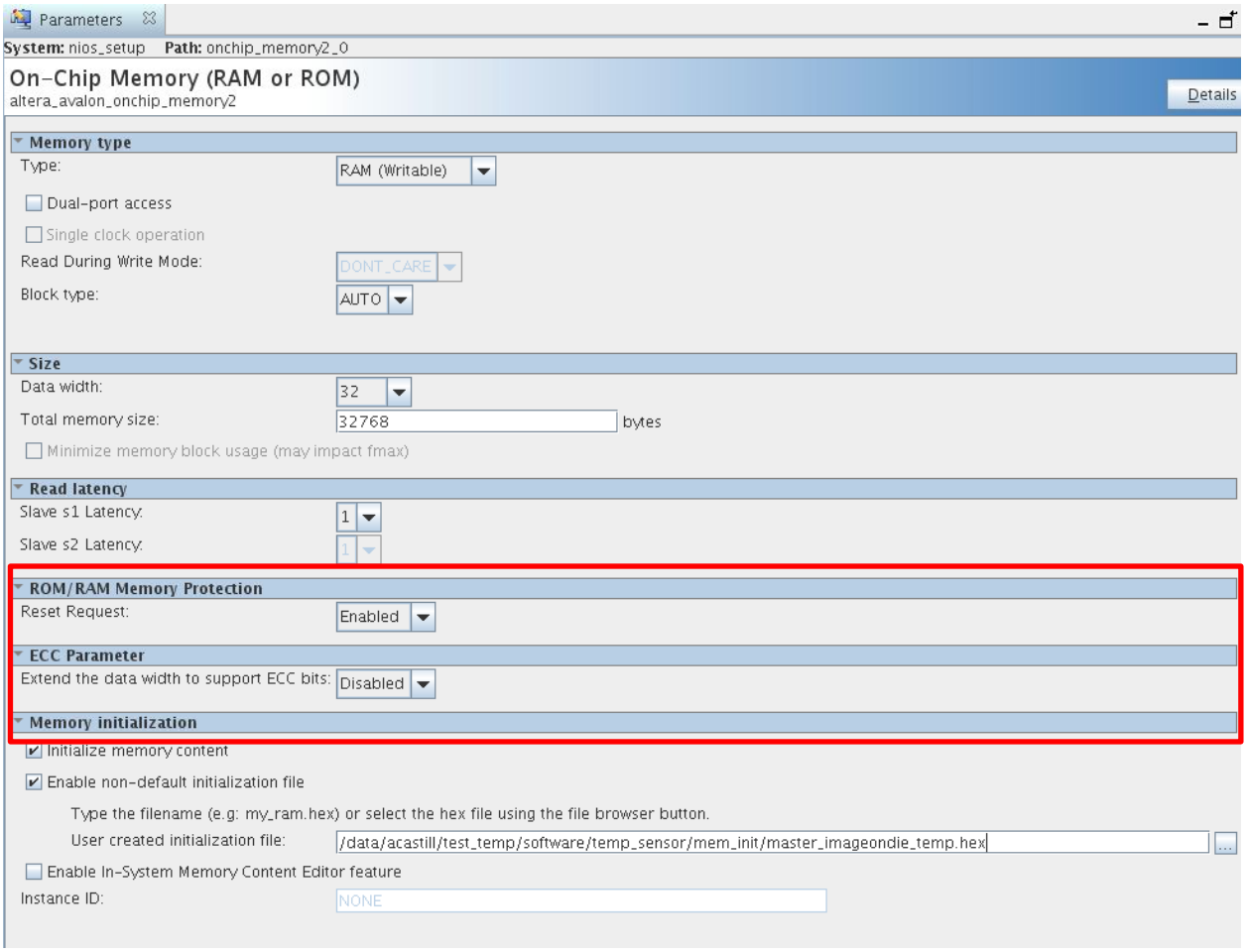
```
#cut all paths to and from altera_reserved_tck
set_clock_groups -exclusive -group [get_clocks altera_reserved_tck]
#constrain the TDI port
set_input_delay -clock altera_reserved_tck 20 [get_ports altera_reserved_tdi]
#constrain the TMS port
set_input_delay -clock altera_reserved_tck 20 [get_ports altera_reserved_tms]
#constrain the TDO port
set_output_delay -clock altera_reserved_tck 20 [get_ports altera_reserved_tdo]
```

Merging the NIOS executable into the FPGA configuration file

This section will allow you to merge the .elf executable into the .sof once your software is stable. In Eclipse, right click the project and select Make Targets. Click on mem_init_generate. You will generate a .hex file that is in the location where your software build is located (e.g):

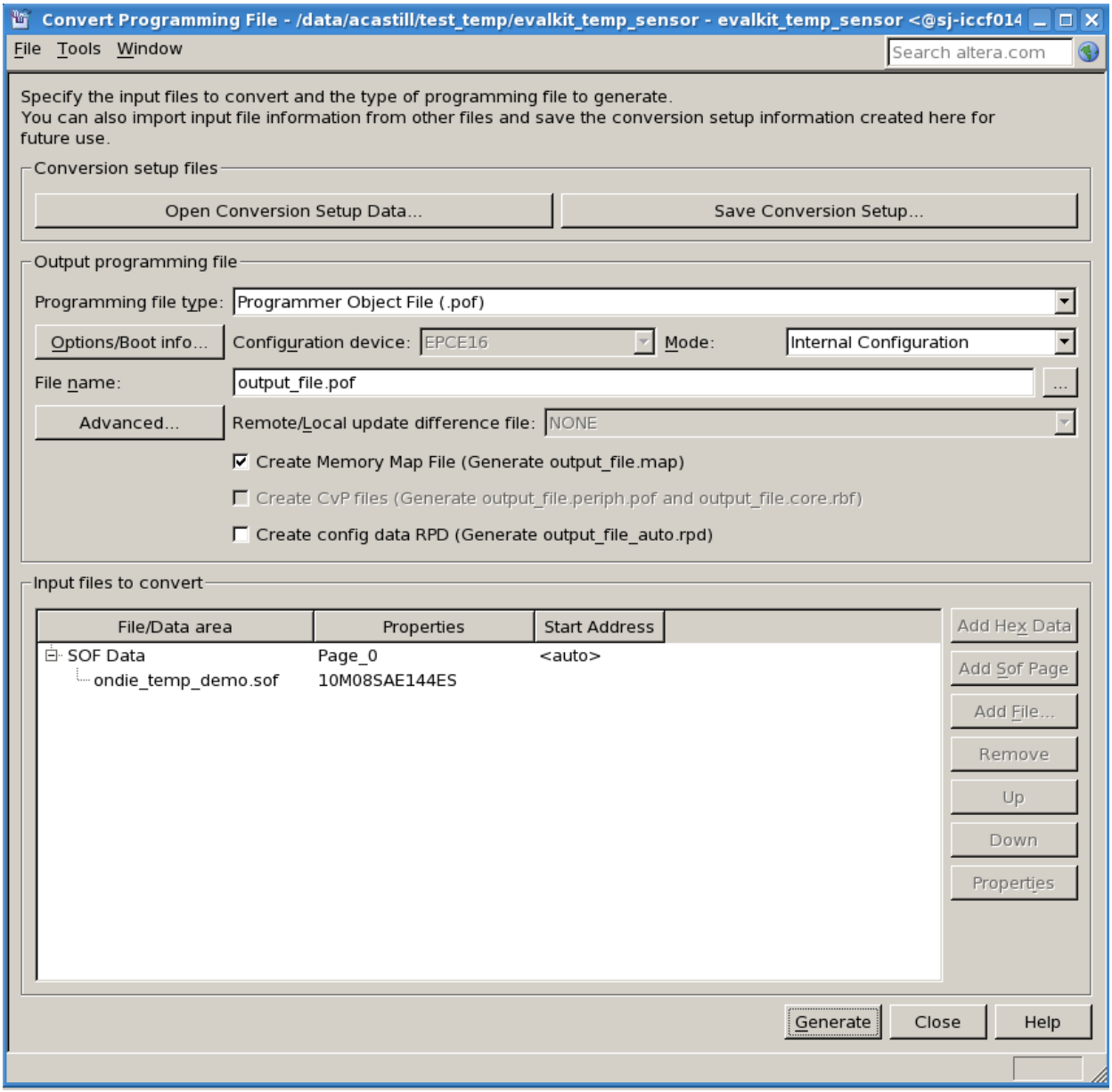
./software/temp_sensor/mem_init/master_imageondie_temp.hex

Next, you need to return to Qsys and change the onchip_memory2_0 component by double clicking it to open the parameters tab. The initialize memory content and enable non-default initialization file needs to on. Enter the location where the .hex file is located.



Next, click Generate HDL →Generate. Return to Quartus and click compile and you will generate a new .sof file with the NIOS executable included. Run the programmer and download the new .sof. Run the demonstration as described in the previous steps.

To generate a .pof file for non-volatile demonstrations, in Quartus run File →Convert Programming Files. Change mode to Internal Configuration. Highlight SOF data, click add file and select output_files/top.sof.



Click ‘Generate’, and you will see the output_files/top.pof file to download with the programmer. The demonstration will continue to run even after power cycling the development kit.