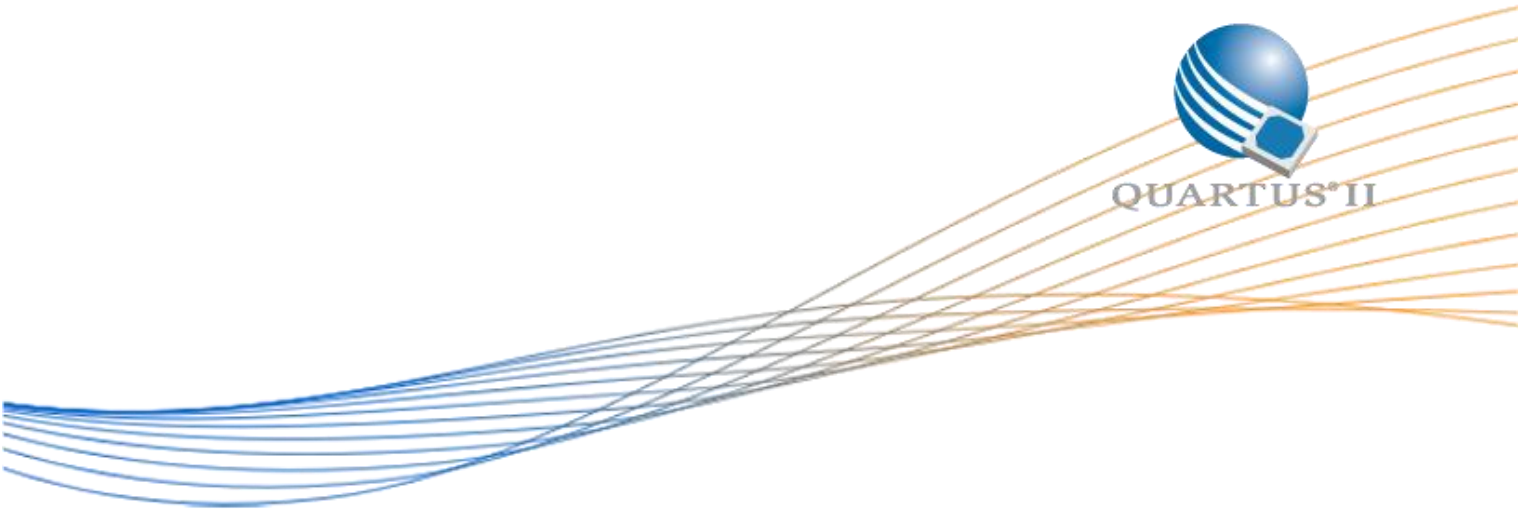


Using the On-Die Temperature Sensor in MAX10 FPGA using NIOS II

Contents

Overview 2

Theory of Operation 3

Steps to run this program: 4

Steps to recreate the output files : 6

Hardware 6

Software 6

Additional Information 9

JTAG Signals - Unconstrained Path violation 9

Merging the NIOS executable into the FPGA configuration file 9

7/15/2015	Initial Release	L. Landis
9/29/2016	Revised documentation and C code was updated for stability.	A. Bacye

Overview

This design example uses NIOS II and a Qsys interconnect to demonstrate how to read the on-die temperature sensor on the MAX10 FPGA Development 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.

Theory of Operation

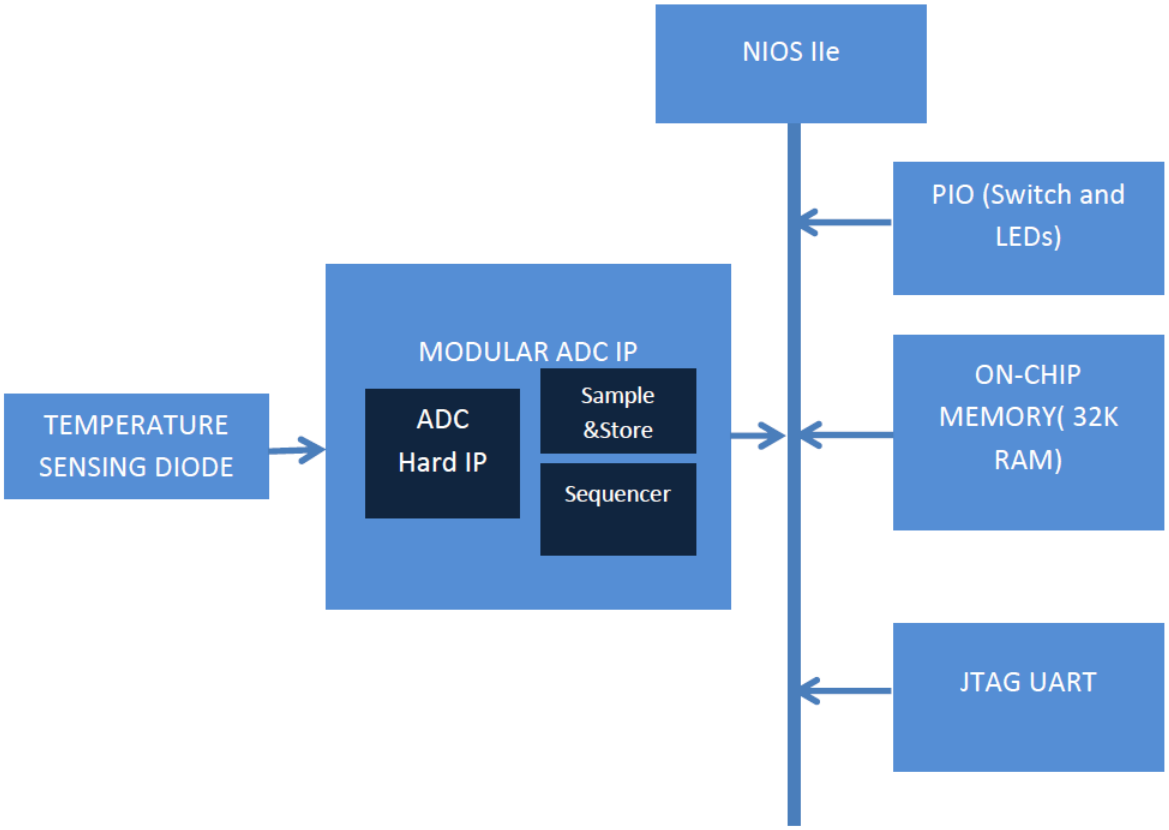


Figure 1 : Qsys System

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 Development 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 Development Kit over 64 averaged values can range from ~ +/- 10C.

The NIOS II also reads the values from the pushbuttons 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.

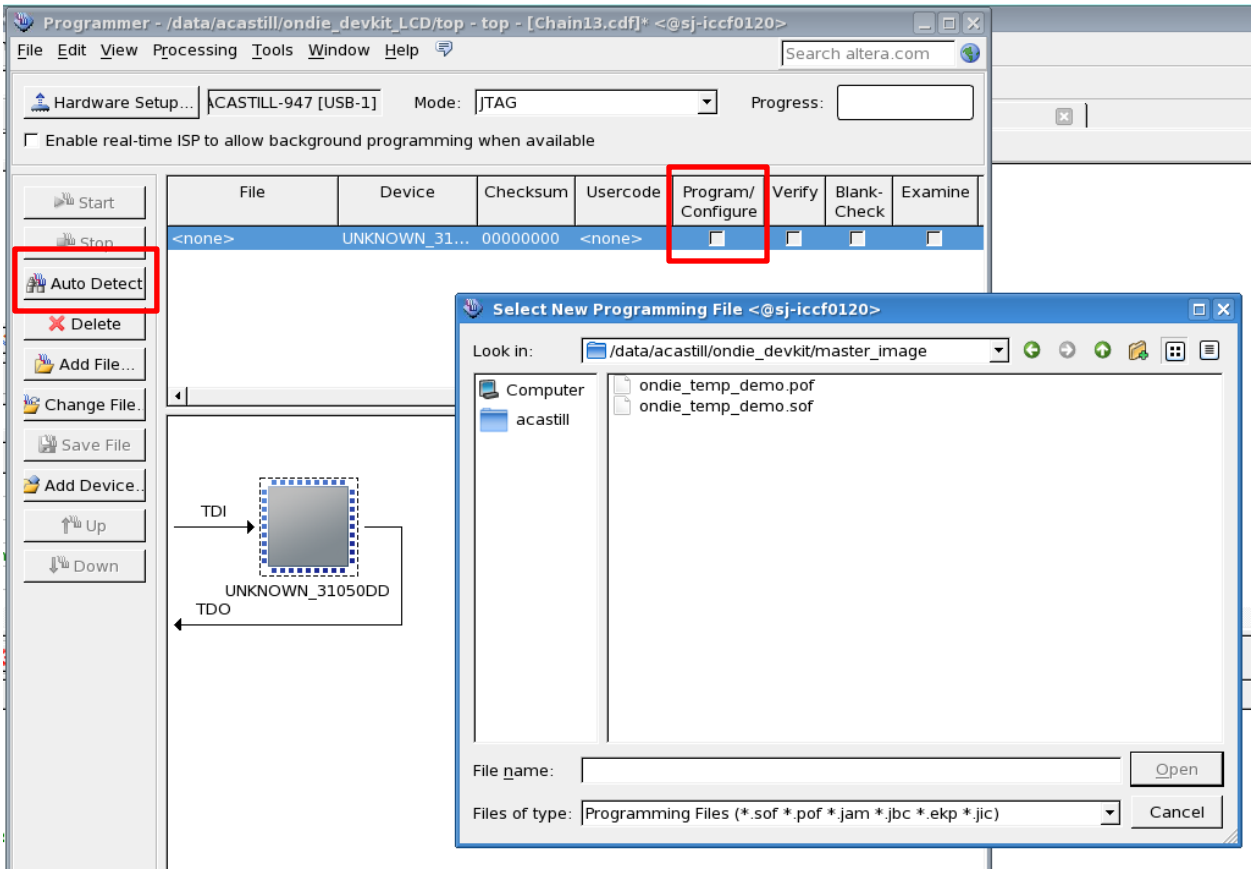
In order to run this example you will need the following software:

- 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 run this program:

After you extract the .qar, we have provided the configuration files for your convenience in the master_image folder so you can directly download them on the MAX 10 Dev Board without recompiling this design. The master_image folder includes the ondie_temp_demo.sof, ondie_temp_demo.pof and ondie_temp_demo.hex files. The ondie_temp_demo.hex file is generated from the .elf (executable linking format) file and is used to initialize the on-chip RAM.

1) Program either ondie_temp_demo.sof or ondie_temp_demo.pof into the MAX 10 Devboard using the Quartus II Programmer.
    a. Click on Auto-Detect, then double-click on <none> and navigate to the appropriate file
    b. Check box “Program/Configure” and hit start.



Notes:

- .sof is SRAM Object file and is volatile while .pof is loaded into the Flash and is non-volatile.
- Sometimes a V-TAP appears in the programmer when auto detected – this is normal

2) Make sure the reset of this example design on the Development Kit is in the "on" position which corresponds to a low state. This is assigned the 1st switch from the on the SW2 bank of switches, located underneath the board. The ON state is written (in very tiny letters on the DIP switch bank) and is the position located when the switch is pushed away from the bottom edge of the PCB.
3) Now, since we are using the Nios Terminal (using JTAG UART) to display the values, we need to open Nios II Command Shell to start displaying the values. To open the command shell :

   Start->All Programs->Altera->Nios II EDS -> Nios II Command Shell in Windows or
   <Nios II EDS install path>/nios2_command_shell.sh in Linux Platforms.

4) In the NIOS command shell, type the following command:
   nios2-terminal

The on-die temperature will be displayed in the command shell once per second.

If you press push buttons 1, 2, and 3, it will turn on LEDs 1, 2 and 3. To reset the design, flip the 1st switch on the SW2 switch bank found under the dev kit.

```
*******PIO and On-Die Temp 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 = 26
On-die temperature = 27
On-die temperature = 26
```

Steps to recreate the output files:

If you want to edit the project and create your own version, use the following procedure:

Hardware

The following steps describe how to setup a project in Quartus II software in order to program the MAX10 FPGA device with the ondie_devkit demo design.

Note – extract platform

i) Launch Quartus II software and open the project top.qpf using **File->Open Project**.

ii) Compile the Project by clicking the  button.

iii) Launch the Quartus II programmer from the Tools menu or alternatively by clicking the  button.

iv) Download the .sof file output_files/top.sof and program the device using the programmer as previously described.

Software

The following steps describe how to use Nios II Software Build Tools for Eclipse to perform the following tasks;

i) Create a software project (BSP and application) from a .sopcinfo file

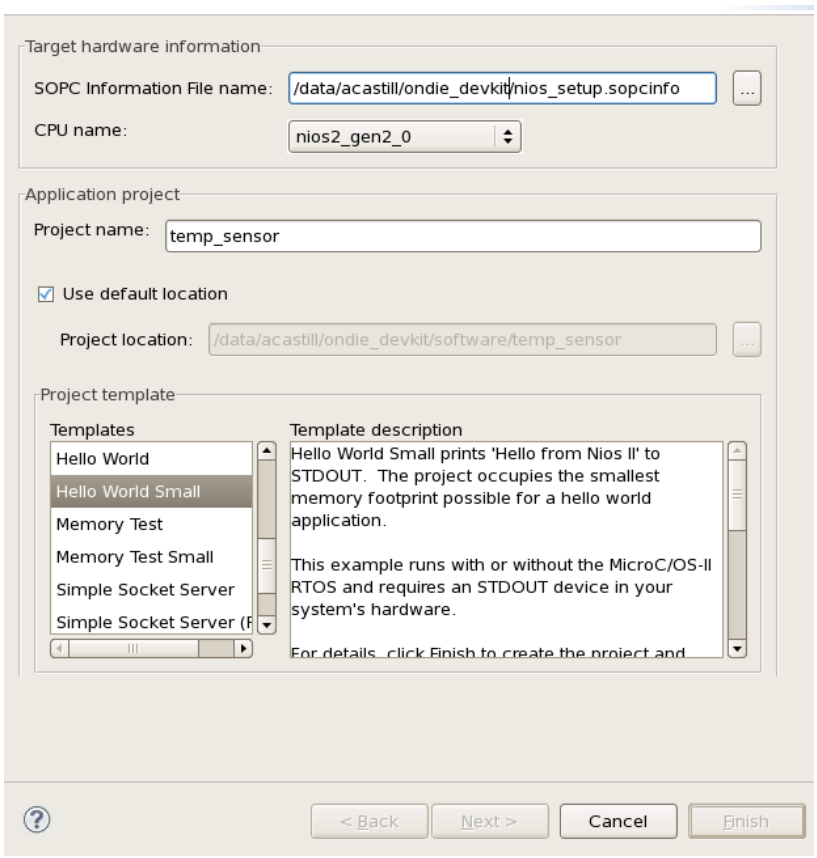
ii) Add source code (that is used to the output the temperature readings) to the project

iii) Download and run source code on the target processor (NIOS II)

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. The .sopcinfo file is

generated by the Qsys file – we have already provided this file for your convenience in the project folder.

a. Provide a name for the software project i.e. “temp_sensor”
b. Select Hello World Small for a template and hit “Finish”



Note: In the future, if you modify the Qsys file for your own design, make sure you are using the most recently generated .sopcinfo file. Check the timestamp for the file.

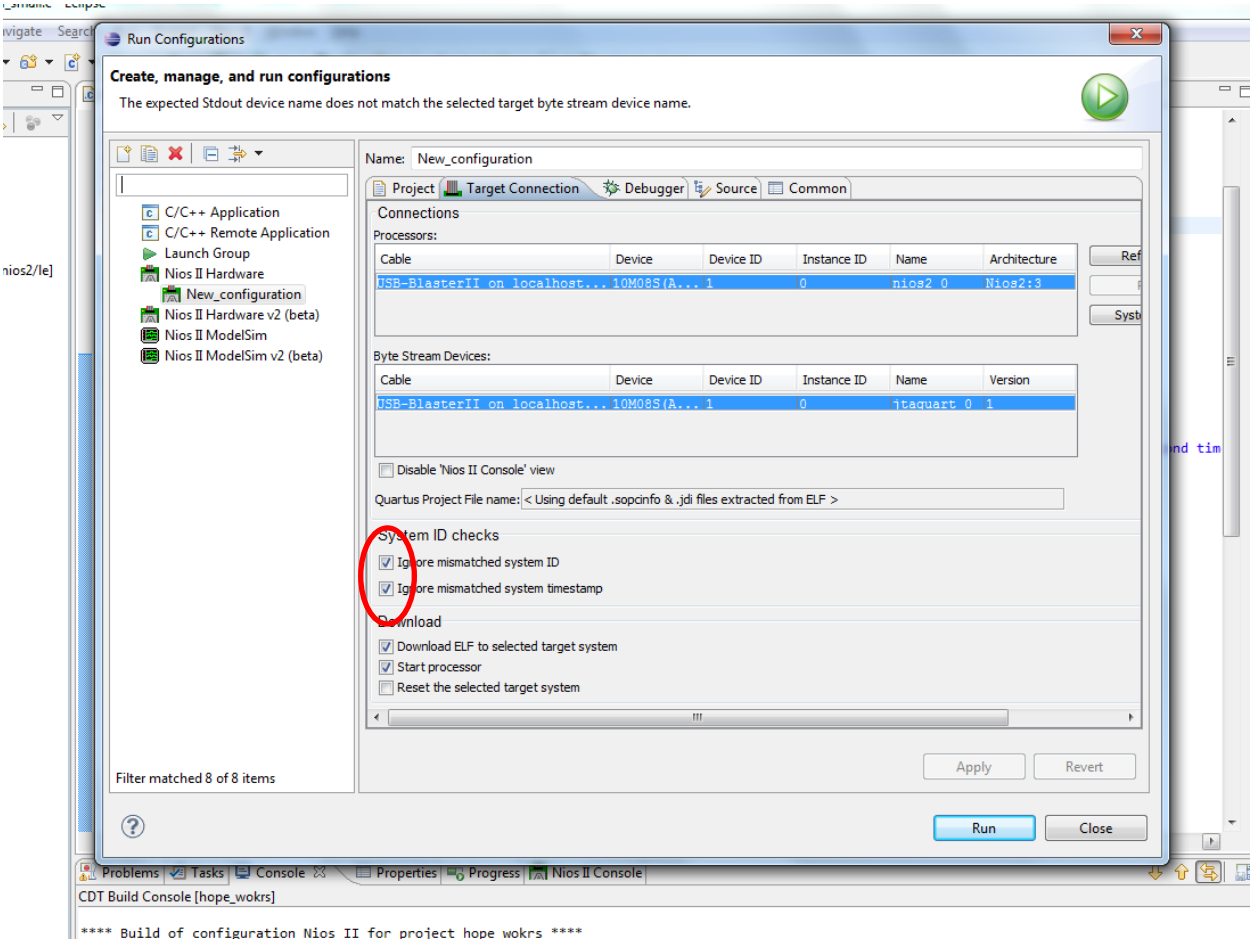
4) Now we have to add ondie_temp_demo.c to the project
   a. In the Project Navigator window, select hello_world_small.c under temp_sensor. Right click to delete.
   b. Right click temp_sensor and select “Import” -> “File System”
   c. Navigate to <project_name>/software. Click OK.
   d. Select ondie_temp_demo.c and click Finish. The file is now imported to the project.
5) 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.
6) Compile the project by selecting Build All from the Projects Menu or alternatively Ctrl+B.

a. It is often better practice to do **Project → Clean** to refresh and build your files.
b. Once successfully built, a .elf file should appear under the project folder.

7) Program the FPGA device by launching Quartus II programmer from the menu and downloading top.sof found in the output folder of the project.
8) Load the executable to the processor by right clicking on the project folder and selecting **Run-As → Run Configurations.**

9) Double click on Nios II Hardware, and new configuration opens on the right pane.
   a. Make sure you select the project name as temp_sensor and .elf file as temp_sensor.elf

10) In the window the appears, select Target connections tab. Check the options:
   a. **Ignore mismatched system ID** and **Ignore mismatched system timestamp**.
   b. If you don't see the connection specified, click Refresh Connections.
   c. Click Apply followed by Run.



11) Click Apply and Run

You will observe the following output in the console:

```
*******PIO and On-Die Temp Sensor example********
Push Buttons 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 = 26
On-die temperature = 27
On-die temperature = 26
```

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]
```

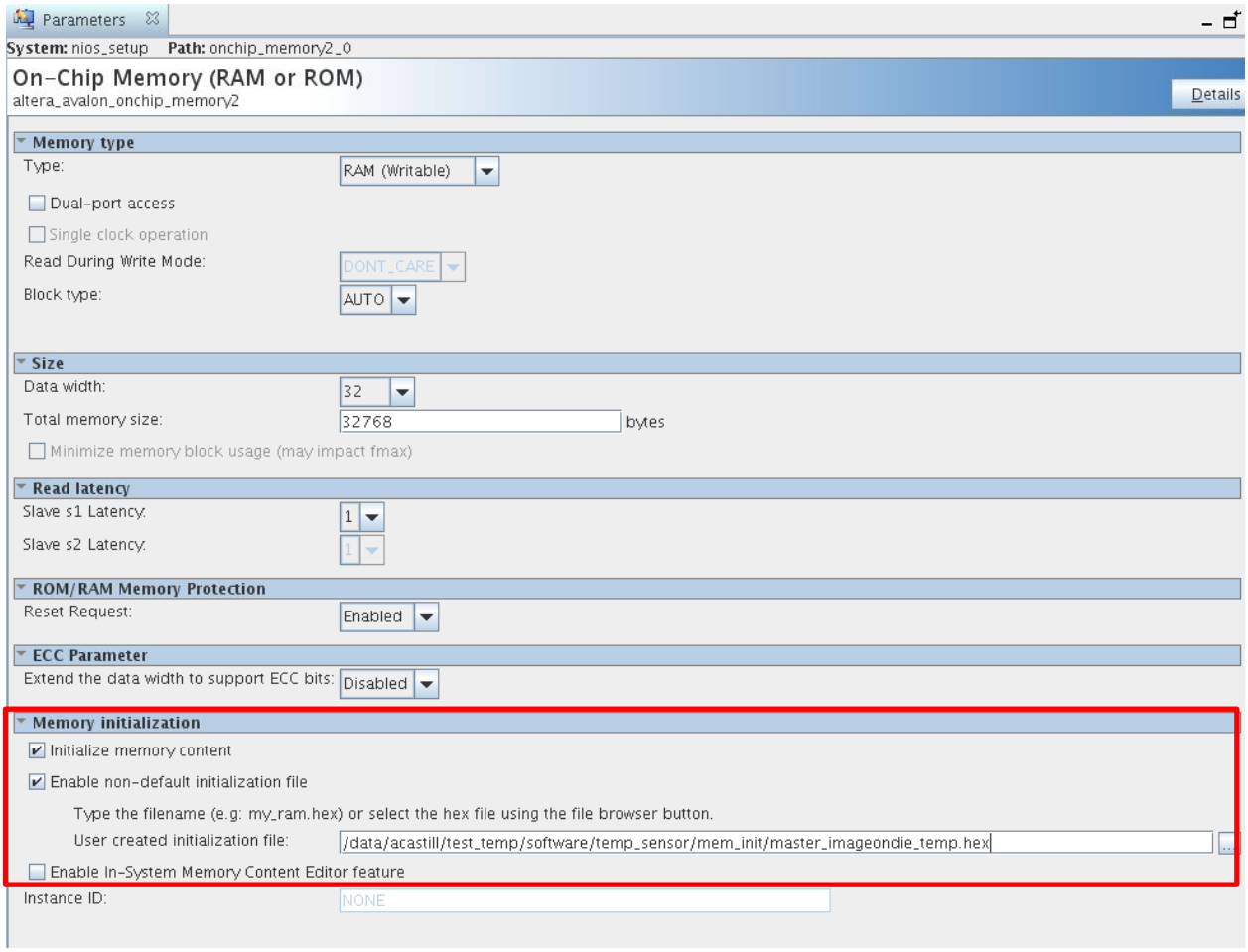
Merging the NIOS executable into the FPGA configuration file

The master_image directory shipped with the design example .sof and .pof files have the NIOS executable incorporated in them. The prior steps detailing how to build your hardware and software show you how to compile the hardware without the image loaded in the .sof or .pof file. In those steps, you load the NIOS executable from Eclipse through the Run → NIOSII Hardware step.

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):

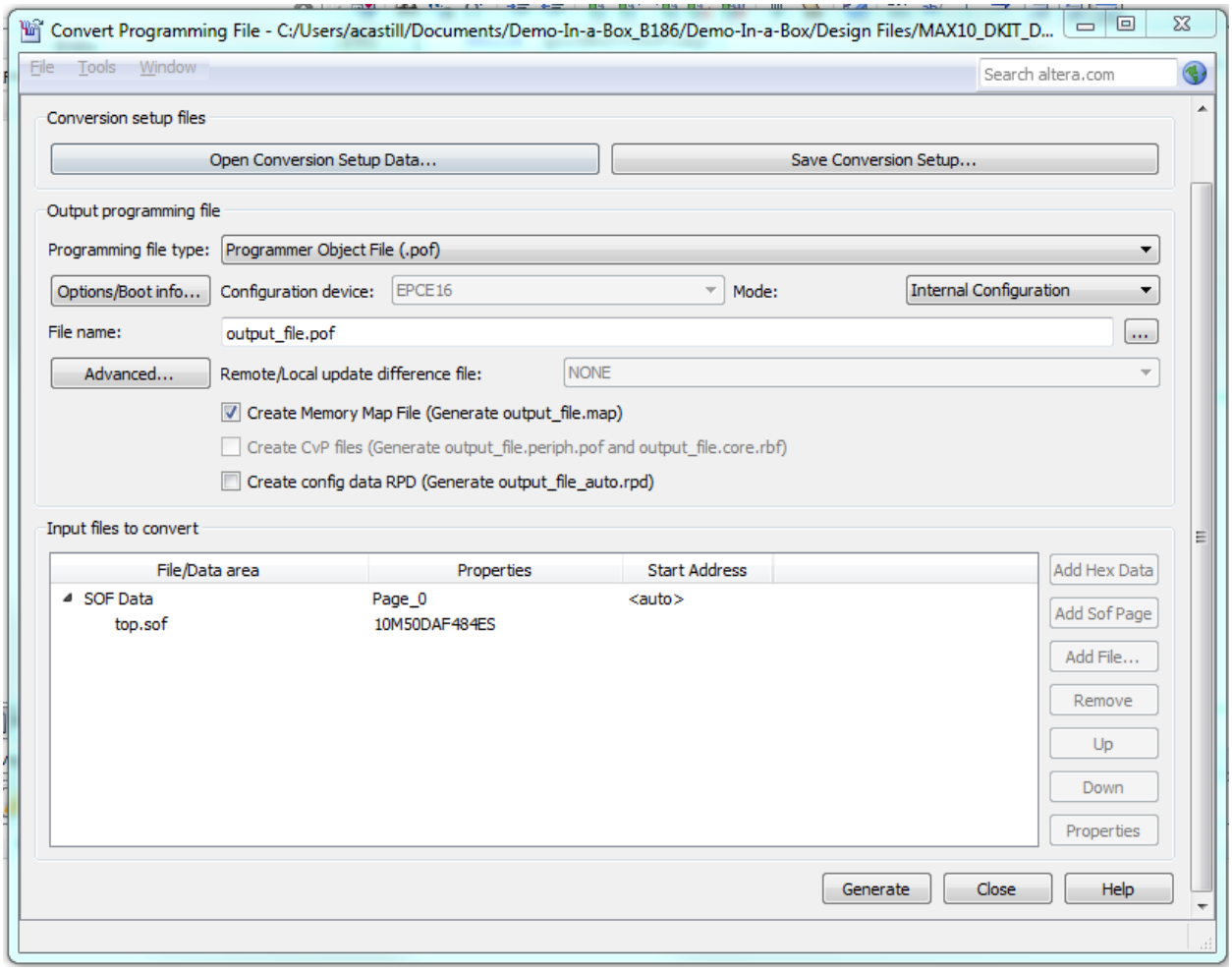
./ software/temp_sensor/mem_init/master_imageondie_temp.hex

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.