Remote System Upgrade Lab (NEEK Board)

Author: Devon Andrade

Date: 8/11/2015

Revision: 1.0

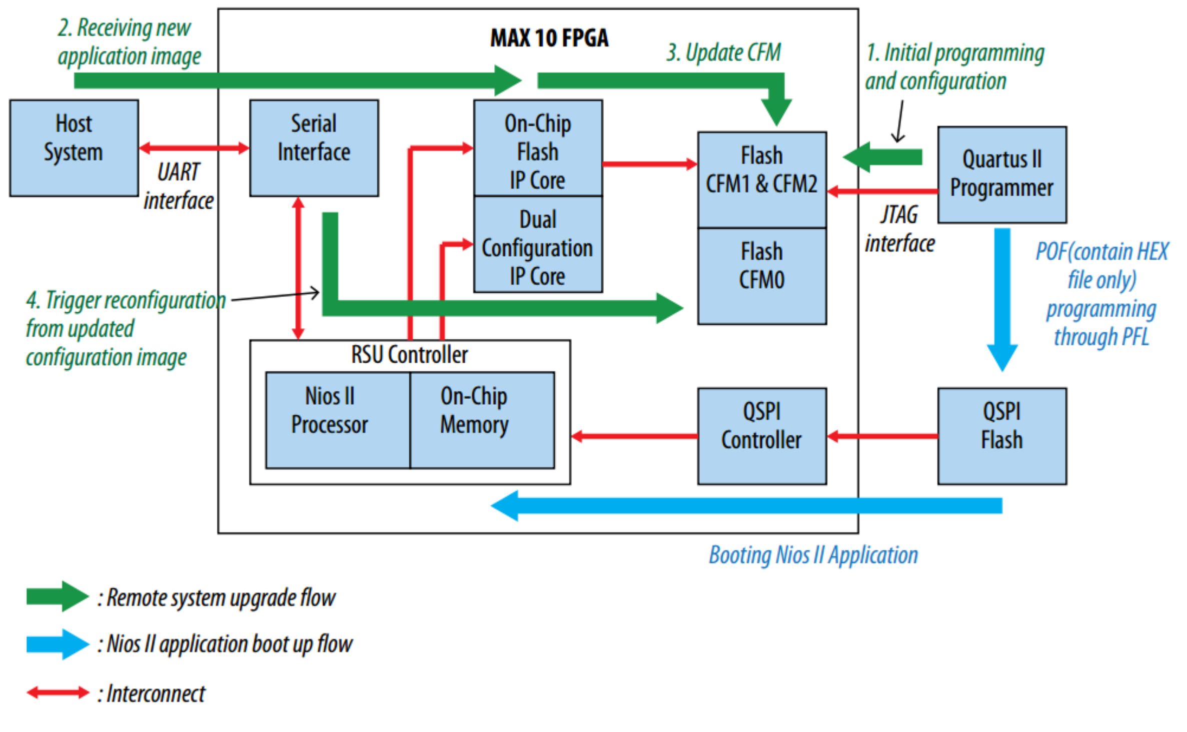
Table of Contents

Lab Introduction

This lab will walk you through creating and programming all of the files needed to perform a remote system upgrade on a Max 10 device. The steps you will take are as follows:

1. Create a default factory image.
2. Program the factory image.
3. Program the Nios II firmware into external QSPI flash.
4. Create an updated application image.
5. Generate a raw programming data (.rpd) file to remotely upgrade CFM1.
6. Setup the USB-To-UART system for transferring data from PC to FPGA.
7. Perform the remote system upgrade.
8. Corrupt an image and try to update with it (verifying the auto-reconfiguration feature).

The design used in this lab is based off of the design created in *AN741: Remote System Upgrade for Max 10 FPGA Devices over UART with the Nios II Processor* with some slight modifications to the source code to make it easier to read and use. As the name would suggest, this design focuses on using a Nios II processor to perform the actual remote upgrade. With that said, a designer could also choose to forego using a processor (e.g. to save resources) and instead design a custom component that receives raw programming data and then passes that data to the On-Chip Flash IP Core for storing into configuration flash. Read Appendix A for more information on the hardware design used in this lab. The block diagram below shows the basic remote system upgrade data flow:


2. Receiving new application image
MAX 10 FPGA
3. Update CFM
1. Initial programming and configuration
Host System
UART interface
Serial Interface
On-Chip Flash IP Core
Dual Configuration IP Core
Flash CFM1 & CFM2
Flash CFM0
Quartus II Programmer
JTAG interface
POF(contain HEX file only) programming through PFL
4. Trigger reconfiguration from updated configuration image
RSU Controller
Nios II Processor
On-Chip Memory
QSPI Controller
QSPI Flash
Booting Nios II Application
: Remote system upgrade flow
: Nios II application boot up flow
: Interconnect

The Nios II firmware that is programmed into the external flash is provided with the lab's source code (in the *software* directory) and can be used as a base for your own remote system upgrade design. Follow the steps in Appendix B to extract the software archive and if you plan on placing this software in external flash, make sure to follow the steps in Appendix C to generate the correct programming files.

Prerequisites

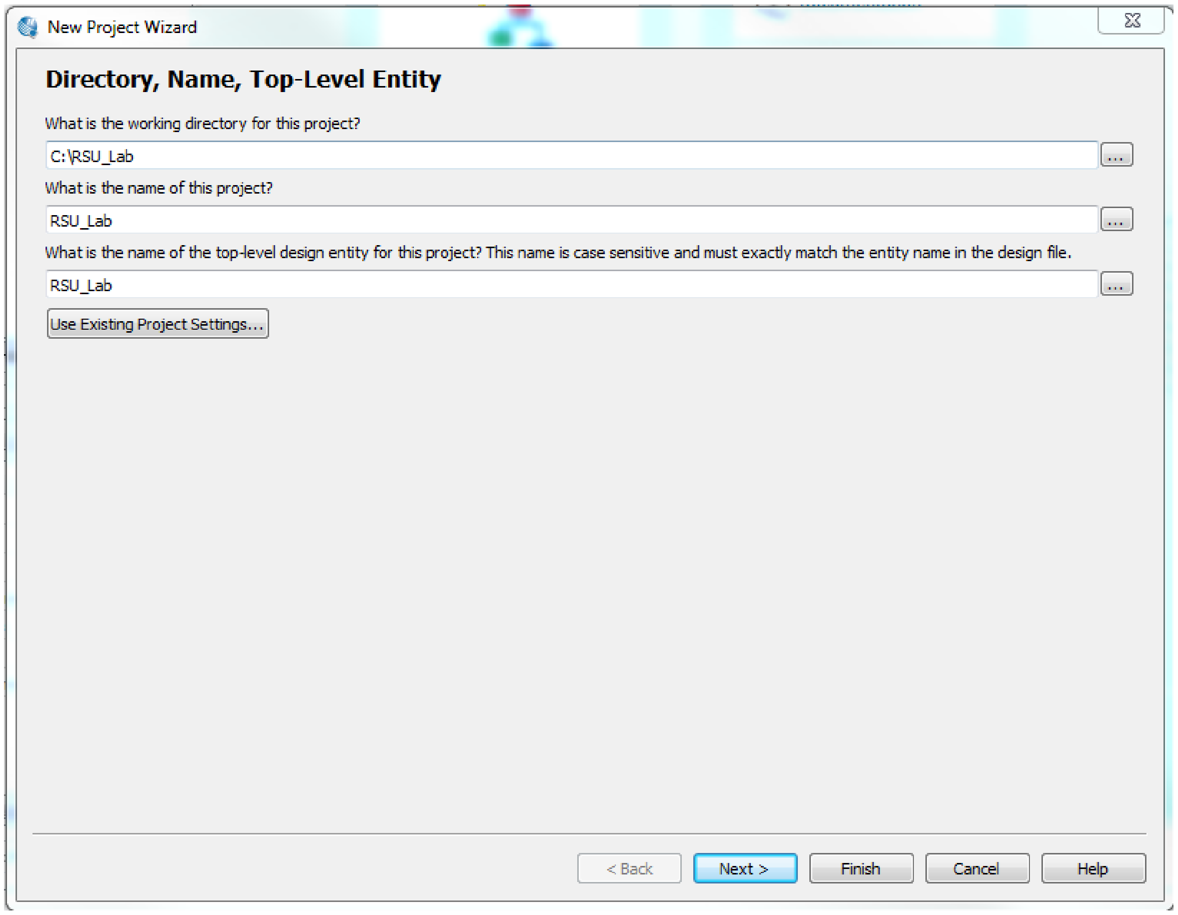
You will need the following software to complete this lab:

- Quartus II v15.0 with the Nios II Software Build Tools for Eclipse installed
- The HxD hex editor v1.7.7.0 (which can be downloaded here)

Getting the Lab's Source Code

The lab's source code is located on the Altera Design Store and can be located at this link: https://cloud.altera.com/devstore/platform/15.0.0/remote-system-upgrade-rsu-lab-neek-version/. Click the download button to download a *.par* file. This file contains the lab source code and will need to be unpacked by the Quartus II software. Follow these steps to unpack the source code:

1. In the Quartus II software, open the New Project Wizard (File menu).
2. Specify C:\RSU_Lab as the working directory for this project
3. Name the project "RSU_Lab". Your settings should be identical to the screenshot below.



Directory, Name, Top-Level Entity

What is the working directory for this project?

C:\RSU_Lab

What is the name of this project?

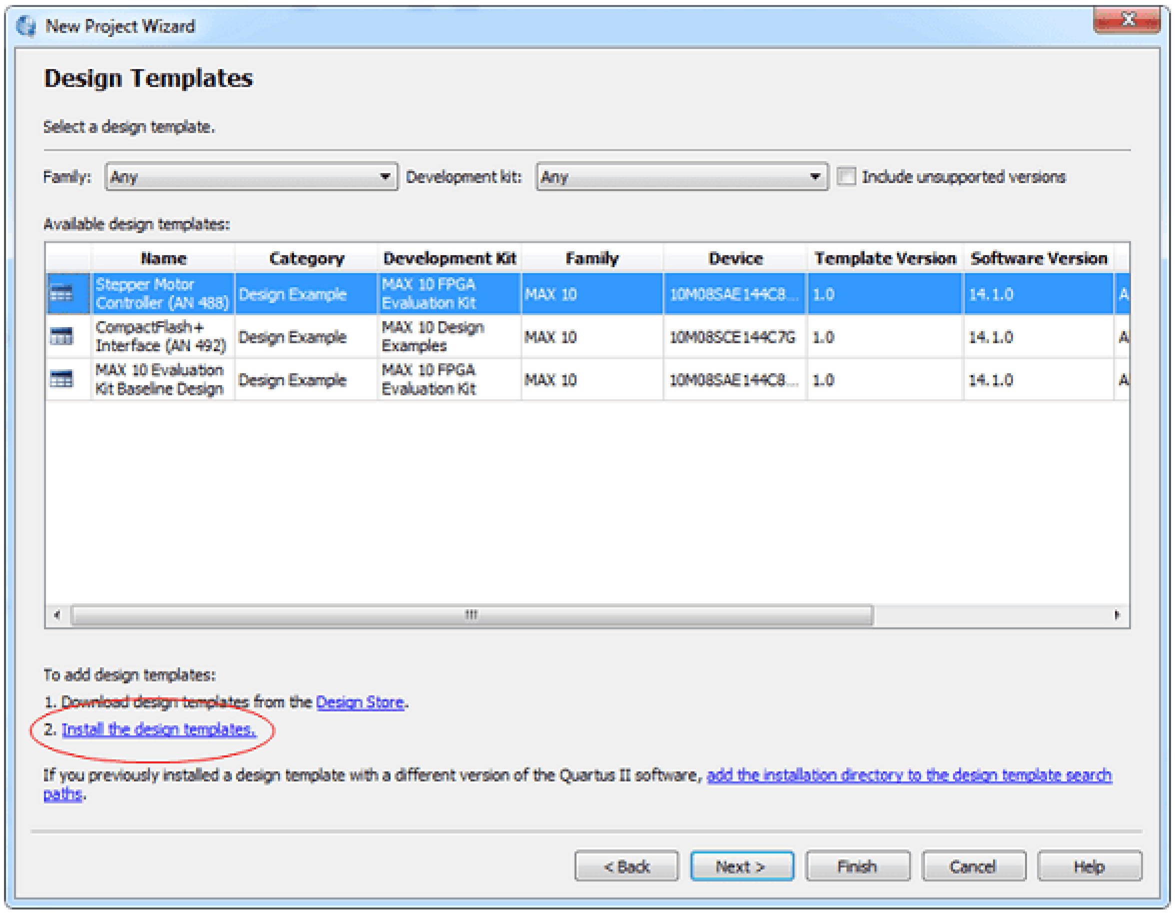
RSU_Lab

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

RSU_Lab

4. Click Next. If prompted to create a new directory, click Yes.
5. Select Project Template, and then click Next.

6. Click the “Install the design templates” link as shown below:



New Project Wizard

Design Templates

Select a design template.

Family: Any — Development kit: Any — ☐ Include unsupported versions

Available design templates:

Name	Category	Development Kit	Family	Device	Template Version	Software Version
Stepper Motor Controller (AN 488)	Design Example	MAX 10 FPGA Evaluation Kit	MAX 10	10M08SAE144C8...	1.0	14.1.0
CompactFlash+ Interface (AN 492)	Design Example	MAX 10 Design Examples	MAX 10	10M08SCE144C7G	1.0	14.1.0
MAX 10 Evaluation Kit Baseline Design	Design Example	MAX 10 FPGA Evaluation Kit	MAX 10	10M08SAE144C8...	1.0	14.1.0

To add design templates:

1. Download design templates from the Design Store.
2. Install the design templates.

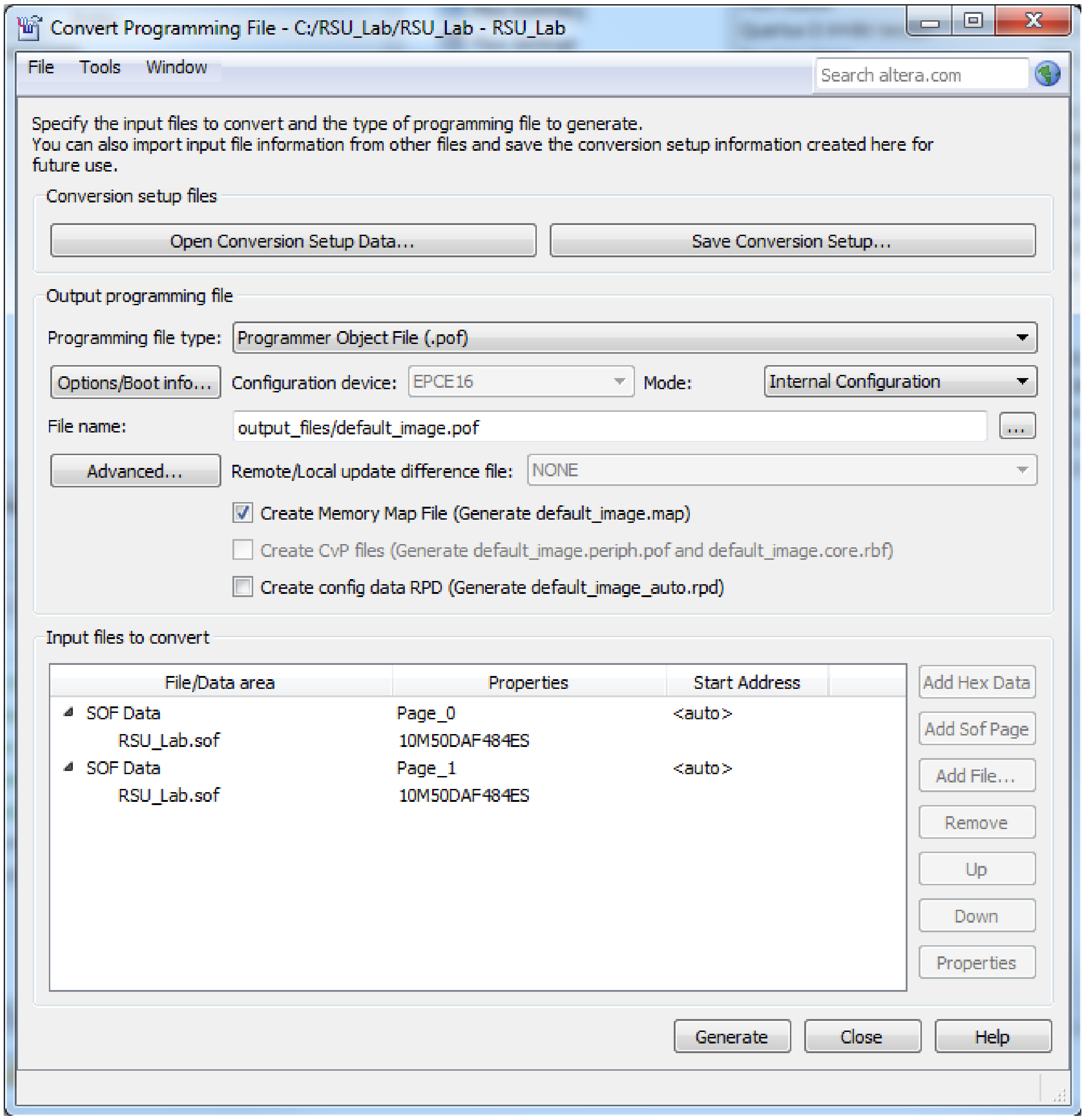
If you previously installed a design template with a different version of the Quartus II software, add the installation directory to the design template search paths.

7. Browse and open the Design template file (.par) that you downloaded previously.
8. Click OK at the Design template Installation dialog box.
9. Once the design template is installed, click on the Remote System Upgrade template that was just added to the templates list and then click Next.
10. Click Finish. The lab project should now be open.

Step 1: Create a Factory Image

Now that the firmware has been programmed, we will create a “factory” (or failsafe/default) image that will be programmed into the device initially. This factory image will reside within both CFM0 and CFM1 of the Max 10 internal flash memory at first. Later on we will remotely program an updated version of this image into CFM1 and then boot into that newer image. Refer to the Max 10 User Flash Memory User Guide for more information on the flash memory built in to the Max 10 family of chips.

1. From within Quartus, select Processing->Start Compilation and wait for the design to compile.
2. Once the design is done compiling, we need to convert the generated *sof* file into a *pof* file for flash programming. Click on File->Convert Programming Files.
3. Under “Mode” select “Internal Configuration.” This specifies we want to generate a programming file for the Max 10 internal configuration flash.
4. Under File name, type in “*output_files/default_image.pof*”.
5. Inside of the “Input files to convert” file list, click on the SOF Data item and then click on “Add File…” in the sidebar. Select *RSU_Lab.sof* in the output_files directory.
6. Click on “Add Sof Page” in the sidebar then repeat step 5 with this new Sof page. These two pages will be used to program CFM0 and CFM1 respectively with the same default image. Your settings should like the following:


Convert Programming File - C:/RSU_Lab/RSU_Lab - RSU_Lab
File
Tools
Window
Search altera.com

Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Open Conversion Setup Data...
Save Conversion Setup...

Output programming file

Programming file type: Programmer Object File (.pof)

Configuration device: EPCE16   Mode: Internal Configuration

File name: output_files/default_image.pof

Remote/Local update difference file: NONE

[✓] Create Memory Map File (Generate default_image.map)

[ ] Create CvP files (Generate default_image.periph.pof and default_image.core.rbf)

[ ] Create config data RPD (Generate default_image_auto.rpd)

Input files to convert

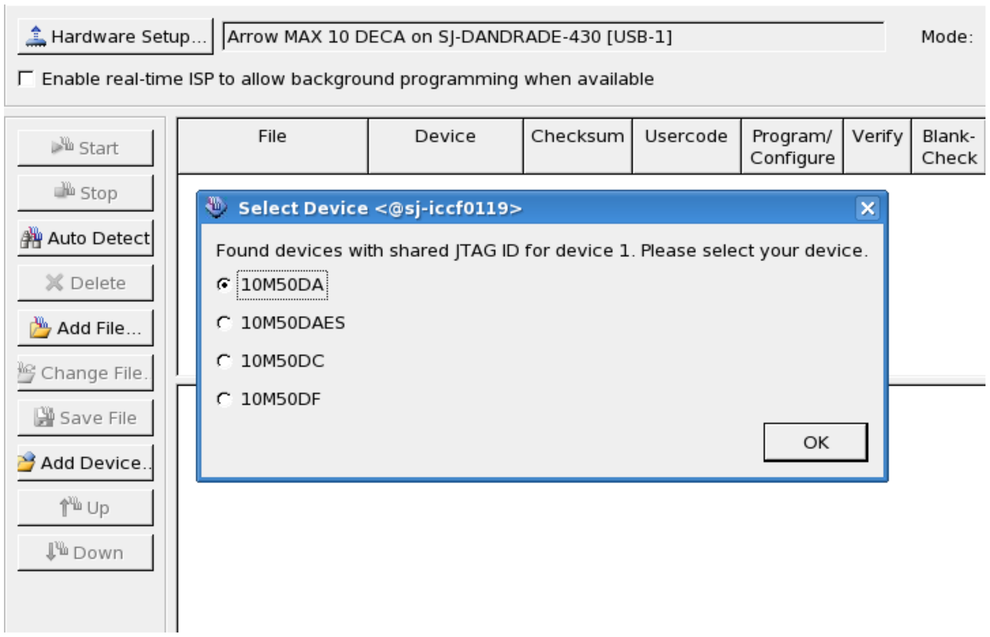
File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>
RSU_Lab.sof	10M50DAF484ES	
SOF Data	Page_1	<auto>
RSU_Lab.sof	10M50DAF484ES	

7. Click on “Generate” to create the programming object file.
8. Close the Convert Programming Files window.

Step 2: Program the Factory Image

Now that the default image is created, we need to program it into the internal configuration flash memory.

1. Set the CONFIG_SEL switch to “0” to inform the device to boot from configuration flash region 0 (CFM0).
2. Open up the Quartus II Programmer by going to Tools->Programmer.
3. If you haven’t already, connect your development board and make sure the programmer detects your board. If it says “No Hardware” in the upper left corner after plugging in your board, click on the “Hardware Setup…” button and select your development board from the “Currently selected hardware” drop down list.
4. Click on Auto Detect and in the pop-up that appears, select *10M50DA* and click OK.



Arrow MAX 10 DECA on SJ-DANDRADE-430 [USB-1]

☐ Enable real-time ISP to allow background programming when available

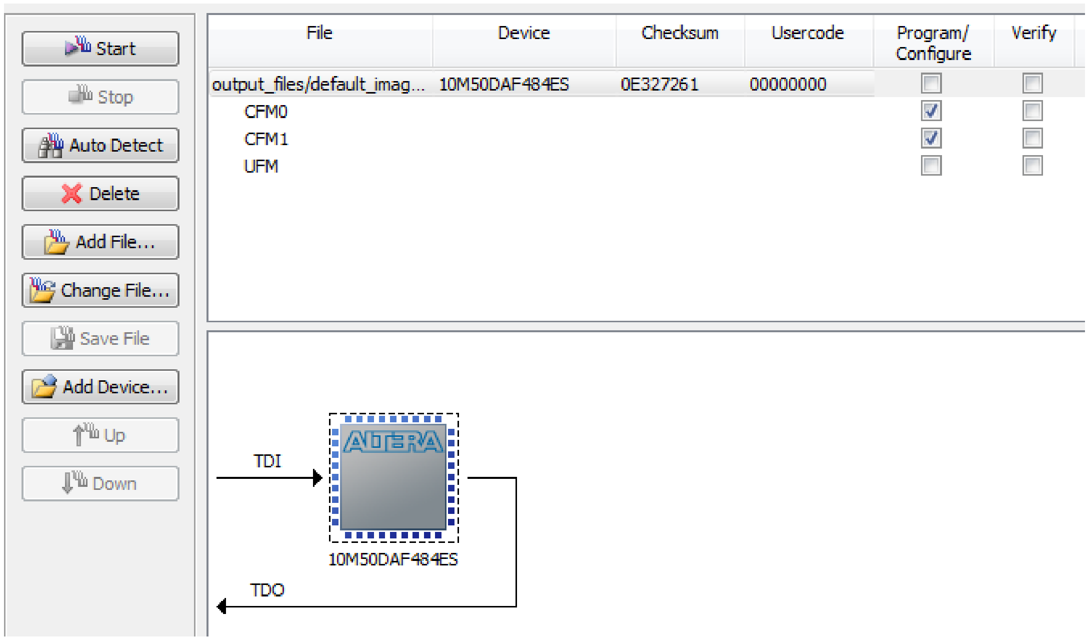
Select Device <@sj-iccf0119>

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

[✓ 10M50DA] 10M50DAES 10M50DC 10M50DF

5. Double click on “<none>” in the file list next to the entry that has the 10M50DA device. Select and open *output_files/default_image.pof* that we created in the last step.

6. Check both the CFM0 and CFM1 checkboxes under the “Program/Configure” header. Your programmer should like the image below:



File	Device	Checksum	Usercode	Program/ Configure	Verify
output_files/default_imag...	10M50DAF484ES	0E327261	00000000	☐	☐
CFM0				☑	☐
CFM1				☑	☐
UFM				☐	☐

7. Click on Start and wait for the device to finish programming. You should now see the LEDs blinking in a binary counter pattern.
8. Confirm that the default image is located in both configuration regions by changing the CONFIG_SEL switch and then turning the board off and on. The same LED blinking pattern should be displayed. Before continuing, change the same switch back to the “0” position and restart the board to make sure the image being booted is from CFM0.
9. Keep the programmer window open.

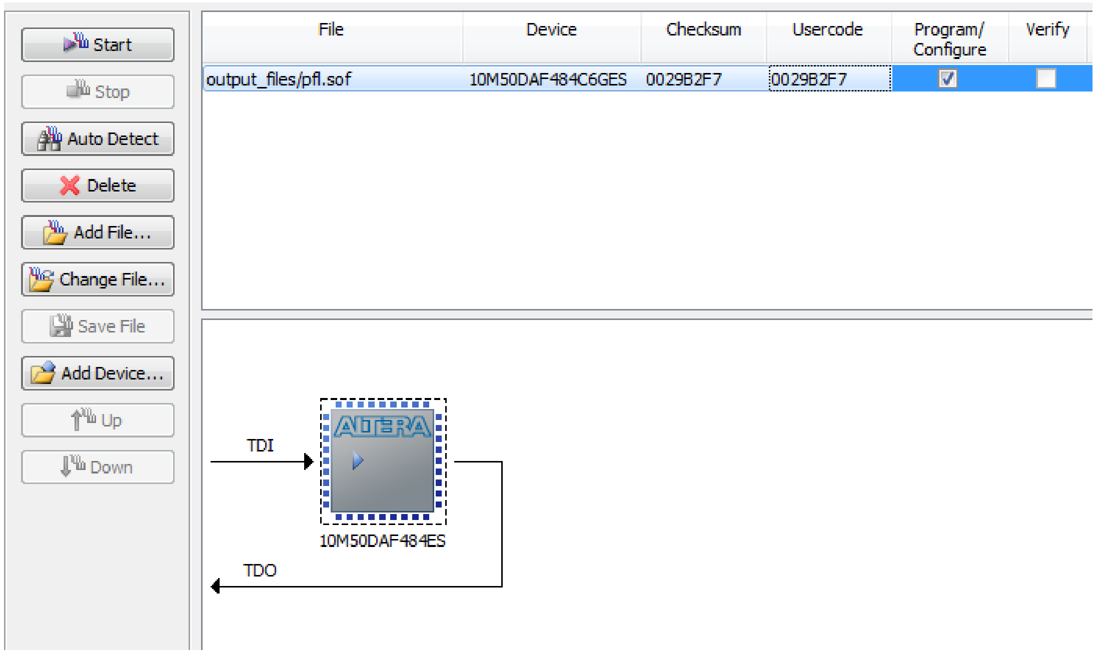
Step 3: Program the Nios II firmware into QSPI Flash

The firmware for the Nios II system responsible for performing the remote system upgrade will be stored within the QSPI flash. The firmware doesn't need to change with each revision (only a simple LED driver block is changing) which means we will only have to program the QSPI once. To program the external QSPI flash we first need to program in a parallel flash loader since the Max 10 doesn't have a hard QSPI controller built into it. A design with just a parallel flash loader has already been created for you and the programming file has been provided.

This firmware is being programmed into the external flash, but upon booting up, will be copied into on-chip memory. This will provide faster execution of the code while performing the remote update operation. For more information on the boot options available to Nios II developers, please reference "Boot Option 2" (in our case, we're using an external QSPI instead of the internal flash memory, but the concepts are the same) in AN730 (Nios II Processor Booting Methods in Max 10 FPGA Devices).

Follow the steps below to program the flash memory:

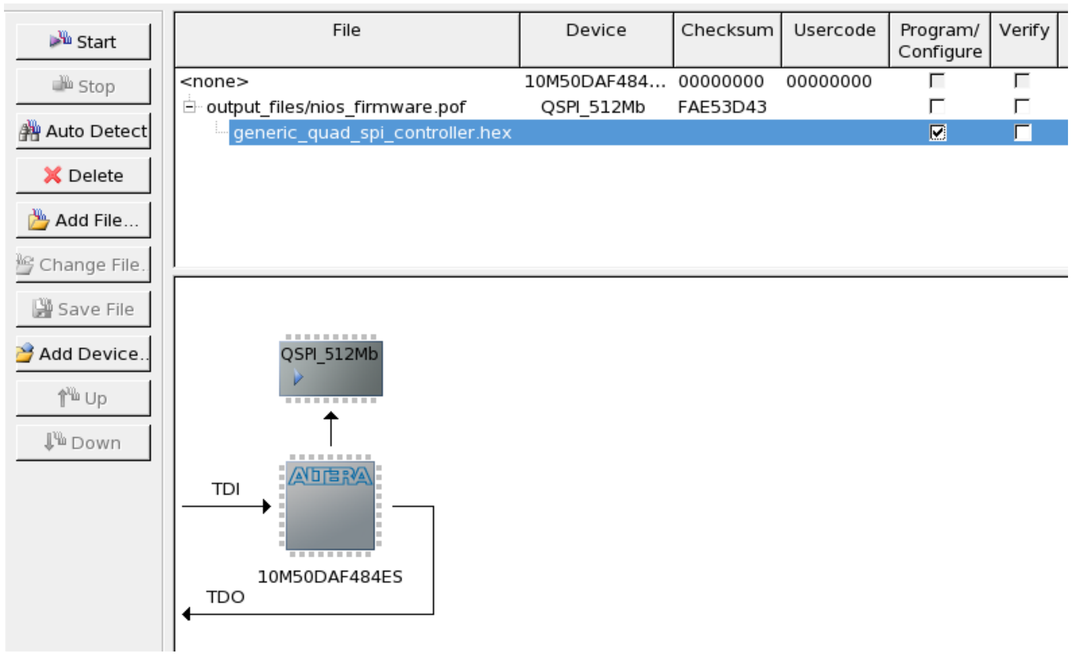
1. If the Programmer isn't already open, open it by going to Tools->Programmer from Quartus.
2. Select the 10M50DA device in the JTAG Chain viewer and then select "Change File…" in the left sidebar. Alternatively, you can just double click in the file list where it currently says "output_files/default_image.pof".
3. Inside of the "output_files" directory, select the *pfl.sof* file and click Open. This programming file was created from a Qsys system with nothing but a Parallel Flash Loader IP instantiated inside of it and a Quartus project that connected the Qsys system up to the QSPI flash pins on the board.
4. Check the "Program/Configure" checkbox for the programming file, then click "Start".



File	Device	Checksum	Usercode	Program/ Configure	Verify
output_files/pfl.sof	10M50DAF484C6GES	0029B2F7	0029B2F7	✓	

5. Once the programming is done, click on the Auto Detect button. Click "Yes" when it asks if you want to overwrite any existing settings. This is because the programmer discovered the QSPI flash thanks to the previously programmed parallel flash loader.

6. Double click on the “<none>” in the file list next to the QSPI_512Mb device (it should be the second “<none>”) and select the *nios_firmware.pof* file from inside of the *output_files* directory. Click Open.
7. The programmer analyzed the *pof* file and found the *hex* file containing the Nios II firmware. To program only the firmware and not the entirety of the flash, select the “Program/Configure” checkbox next to the *generic_quad_spi_controller.hex* file only as show below. Click “Start” and wait for the flash to program (this will take up to a few minutes).

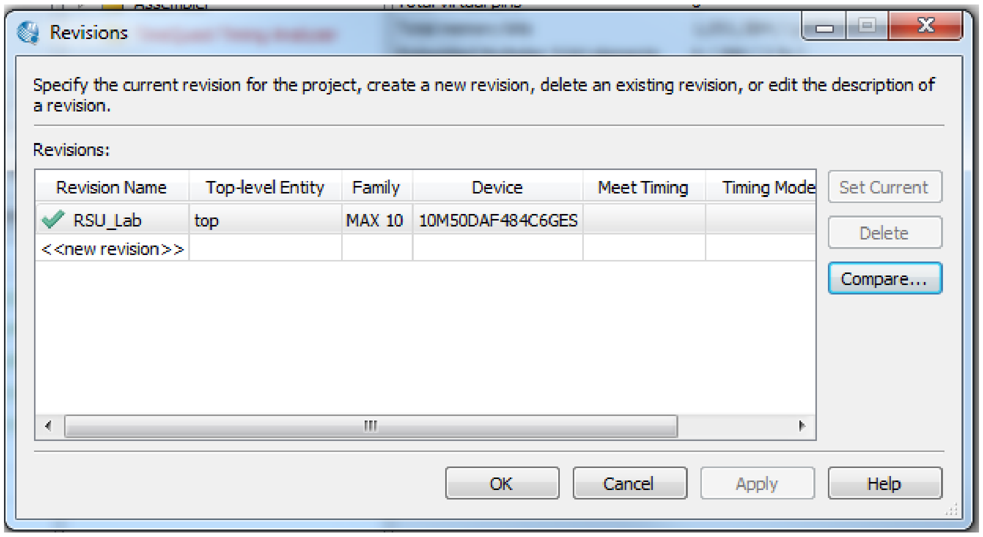


File	Device	Checksum	Usercode	Program/ Configure	Verify
<none>	10M50DAF484...	00000000	00000000	☐	☐
output_files/nios_firmware.pof	QSPI_512Mb	FAE53D43		☐	☐
generic_quad_spi_controller.hex				☑	☐

Step 4: Create an Updated Application Image

Now that the factory image has been programmed in, we need to create a new, updated version of our design to remotely upgrade with. The only difference between this new version and the old one will be a different LED driver. This updated version will drive the LEDs in a pattern similar to the lights found on a certain Knight Rider vehicle from a popular 80's television series of the same name. To make this new version, we will use the Quartus Revisions feature and swap the file responsible for driving the LEDs with a newer file.

1. Within Quartus click on Project->Revisions.



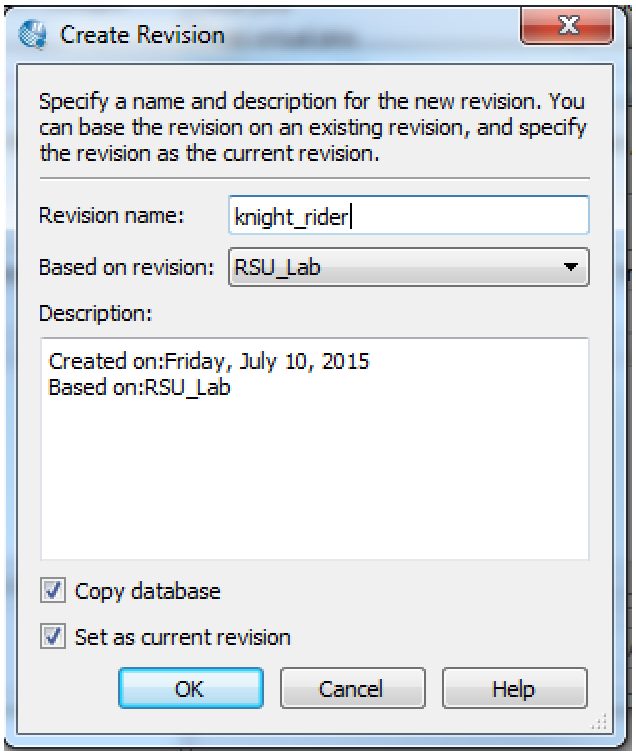
Specify the current revision for the project, create a new revision, delete an existing revision, or edit the description of a revision.

Revisions:

Revision Name	Top-level Entity	Family	Device	Meet Timing	Timing Mode
✓ RSU_Lab	top	MAX 10	10M50DAF484C6GES		
<<new revision>>					

2. Inside of the Revisions dialog, double click on "<<new revision>>".

3. For the revision name, type in “knight_rider” and leave the rest of the settings at the default. Click OK.



Specify a name and description for the new revision. You can base the revision on an existing revision, and specify the revision as the current revision.

Revision name: knight_rider

Based on revision: RSU_Lab

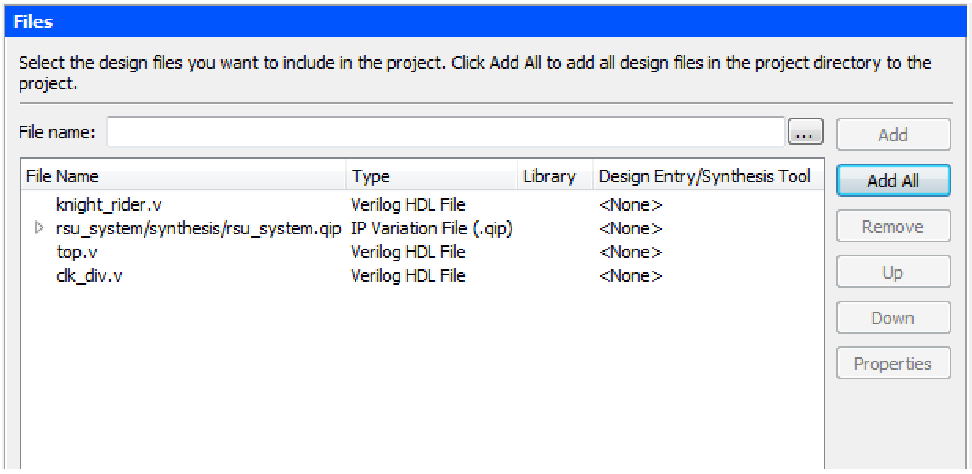
Description:

Created on:Friday, July 10, 2015
Based on:RSU_Lab

[✓] Copy database

[✓] Set as current revision

4. If a dialog appears asking if you want to save the chain.cdf file, select “No”.
5. After Quartus has finished creating the new revision, click OK inside of the Revisions dialog box.
6. Once Quartus has switched to the new revision, click on Project->Add/Remove Files in Project.
7. Click on the *binary_leds.v* file and click “Remove” in the sidebar.
8. Next to the File name text box, click on the “…” button and browse for the “knight_rider.v” file. Click Open, and then click Add in the sidebar.



Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool
knight_rider.v	Verilog HDL File		<None>
rsu_system/synthesis/rsu_system.qip	IP Variation File (.qip)		<None>
top.v	Verilog HDL File		<None>
clk_div.v	Verilog HDL File		<None>

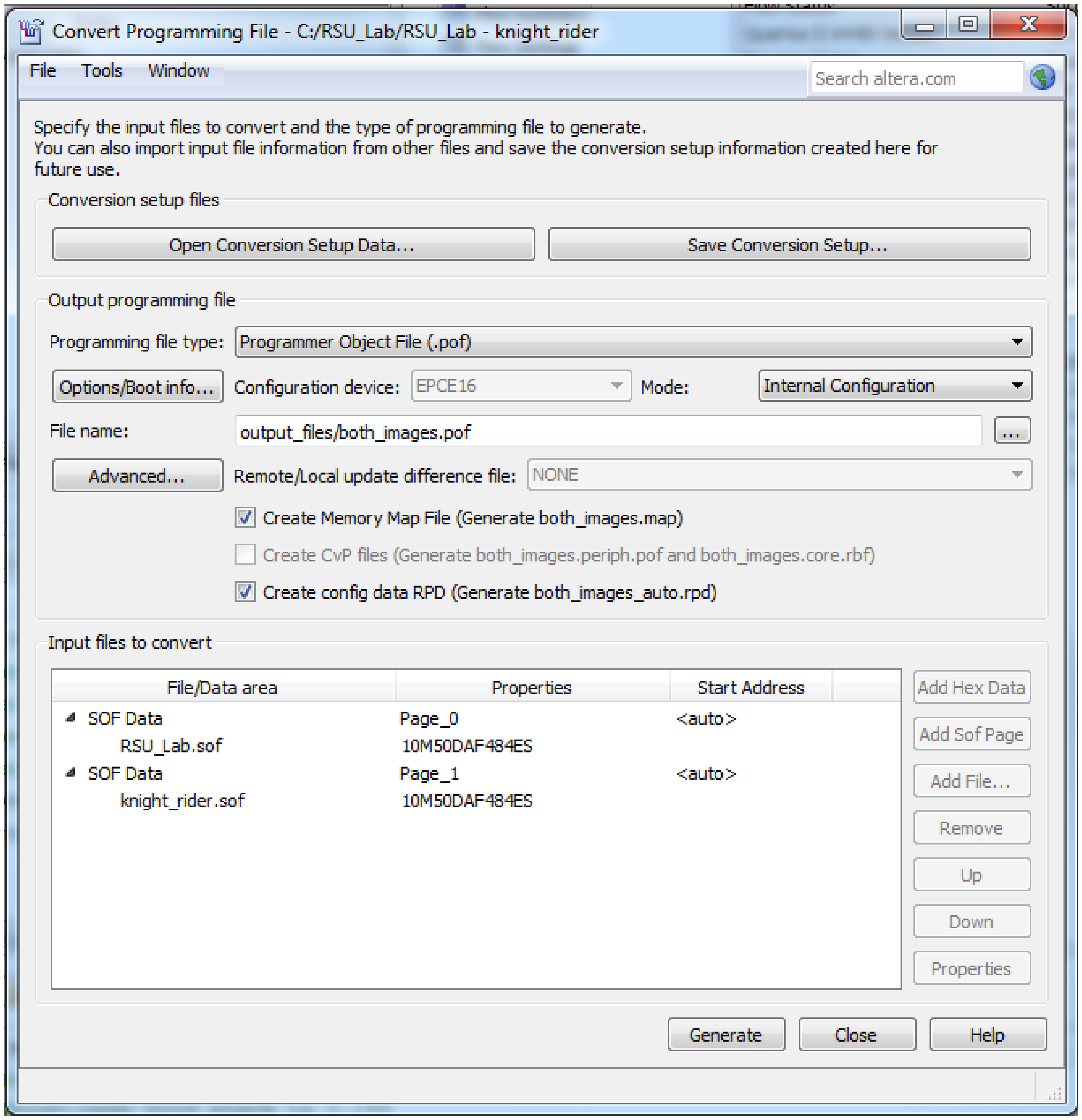
9. Click OK to close the Settings dialog.
10. Start compilation by clicking on Processing->Start Compilation.

Step 5: Generate an .rpd File for Remote Upgrading

To remotely upgrade the design we need to have a Raw Programming Data (.rpd) file. This file can be generated alongside a *.pof* file in the Convert Programming Files dialog window. In the current version of Quartus (v15.0), you can’t generate an .rpd file from an individual Sof file if you are using the Dual Configuration mode. Instead, you have to create an .rpd that contains both the default/factory image (CFM0) and the newer application image (CFM1) then use a hex editor to copy out just the newer application image into its own file that will be used to remotely upgrade. This will be improved upon in a future release of Quartus.

1. From within Quartus, click on File->Convert Programming Files.
2. Under “Mode” select “Internal Configuration”.
3. Under “File name”, type in *output_files/both_images.pof*.
4. Make sure both the “Create Memory Map File” and “Create config data RPD” checkboxes are selected.
5. In the “Input files to convert” section select the auto-generated SOF Data (Page_0/CFM0) and Add the “RSU_Lab.sof” file as described in the “Create a Factory Image” section.
6. Add a new Sof page (Page_1/CFM1) and add the *knight_rider.sof* to this page. We will later edit out the raw programming data created from this *.sof* file.

7. Make sure your settings are identical to the image below then click Generate.



Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Output programming file

Programming file type: Programmer Object File (.pof)

Configuration device: EPCE16

Mode: Internal Configuration

File name: output_files/both_images.pof

Remote/Local update difference file: NONE

[✓] Create Memory Map File (Generate both_images.map)

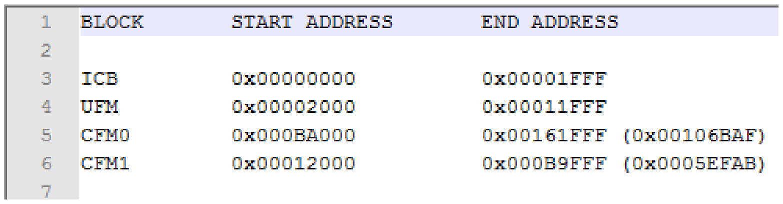
[ ] Create CvP files (Generate both_images.periph.pof and both_images.core.rbf)

[✓] Create config data RPD (Generate both_images_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>
RSU_Lab.sof	10M50DAF484ES	
SOF Data	Page_1	<auto>
knight_rider.sof	10M50DAF484ES	

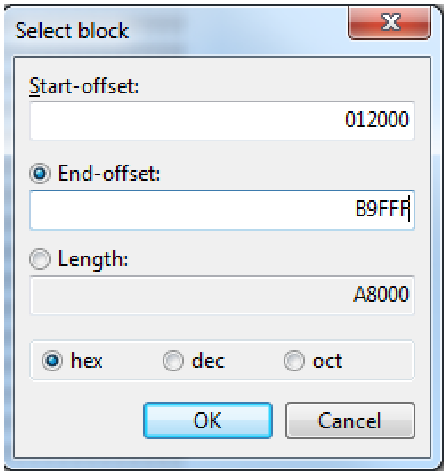
8. Open up the HxD hex editor tool and click on File->Open.
9. Open the *C:\RSU_Lab\output_files\both_images_auto.rpd* file that was previously generated.
10. To know where the new application image is located inside of this *.rpd* file, you can open up *C:\RSU_Lab\output_files\both_images.map* in a text editor. In this case, the application image is located between addresses 0x12000 and 0xB9FFF while the factory image is stored from 0xBA000 to 0x161FFF.



```
BLOCK       START ADDRESS       END ADDRESS

ICB         0x00000000          0x00001FFF
UFM         0x00002000          0x00011FFF
CFM0        0x000BA000          0x00161FFF (0x00106BAF)
CFM1        0x00012000          0x000B9FFF (0x0005EFAB)
```

11. Inside of the hex editor, click on Edit->Select Block.

12. For the Start-offset, put “12000”.
13. For the End-offset, put “B9FFF” and click OK.


Select block

Start-offset: 012000

[✓ End-offset:] B9FFF

Length: A8000

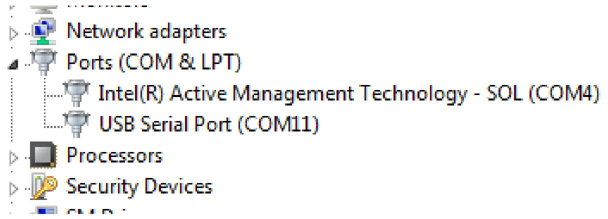
[✓ hex] dec oct

14. Click Edit->Copy.
15. Create a new file by going to File->New.
16. Paste the copied contents into the new file by click on Edit->Paste insert.
17. Click OK at the dialog confirming the file size changing.
18. Save this new file as *knight_rider.rpd* in the *C:\RSU_Lab\output_files* folder. This is the file we will use to remotely upgrade image 1 (CFM1) in the Max 10 device.

Step 6: Setup the USB-To-UART System

In this Remote System Upgrade design we are sending the new application image over a serial UART connection. The Max 10 Development Kit has an onboard USB-To-UART converter. Follow the steps below to connect the converter to the board and to determine which COM port it represents in Windows.

1. Connect a mini-USB cable to the UART1 connector on the dev kit board and your PC.
2. Open up Device Manager by opening up the start menu, searching for “Device Manager”, and clicking on the first result.
3. Under “Ports (COM & LPT)” there should be a USB Serial Port (or similar) device with a COM number next to it. For instance, the serial port below is utilizing COM11:



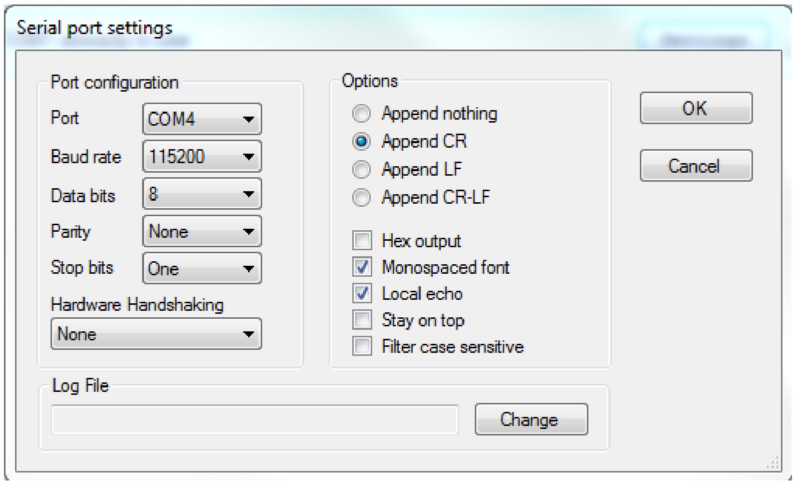
```
Network adapters
Ports (COM & LPT)
    Intel(R) Active Management Technology - SOL (COM4)
    USB Serial Port (COM11)
Processors
Security Devices
```

4. Remember this COM port number for later.

Step 7: Perform the Remote System Upgrade

Everything is now setup to actually perform a remote system upgrade. The following steps will walk you through updating the CFM1 flash regions (boot image 1) with the newly generated knight rider application image.

1. Open up the *RemoteTerminal.exe* software located in the *C:\RSU_Lab\TerminalSoftware* folder. This program will be used to interact with the Nios II firmware and to send the application image over UART. In reality, any terminal program that supports sending files should work fine. This terminal software is derived from the Termie open source project. **Note: if you are getting errors when trying to run this software from a network drive, first copy and paste the RemoteTerminal.exe file to your local drive before running.**
2. Click on the Settings button and change the "Port" to the COM port you noted down in the previous section. Leave the rest of the settings at their defaults. Your settings should like the following (besides the COM port):

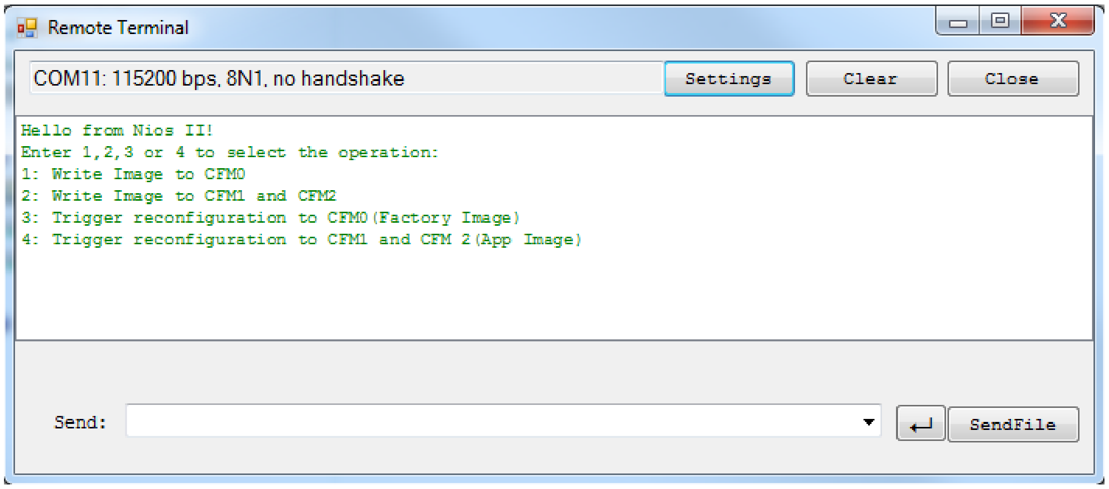


Serial port settings

Port configuration
- Port: COM4
- Baud rate: 115200
- Data bits: 8
- Parity: None
- Stop bits: One
- Hardware Handshaking: None

Options
- Append nothing
- [✓ Append CR]
- Append LF
- Append CR-LF
- Hex output
- [✓ Monospaced font]
- [✓ Local echo]
- Stay on top
- Filter case sensitive

3. Press the KEY0 button to reset the Nios II processor. Inside the terminal window you should now be presented with a menu.

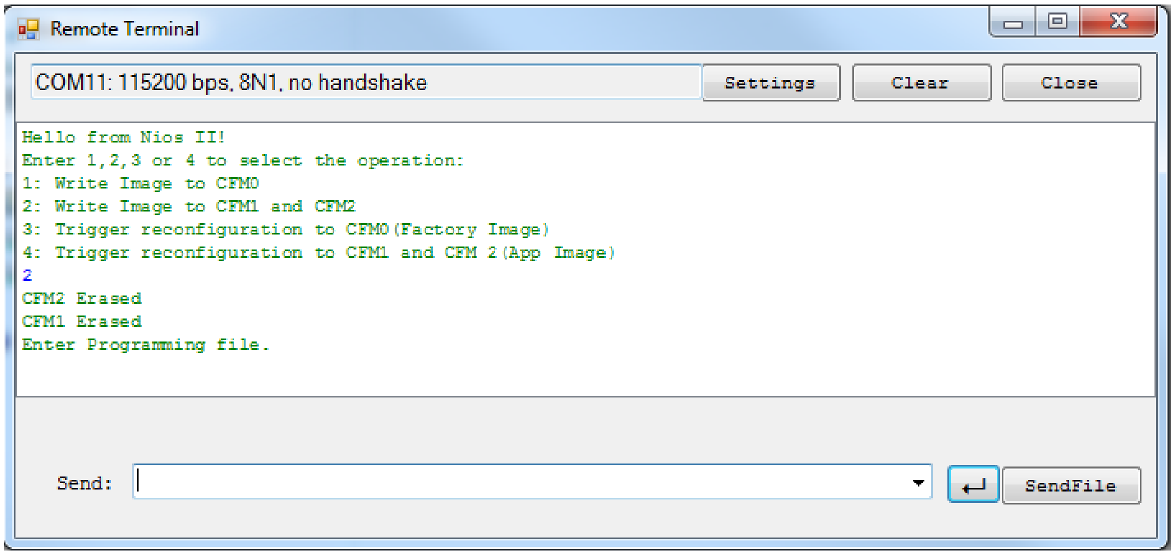


COM11: 115200 bps, 8N1, no handshake

```
Hello from Nios II!
Enter 1,2,3 or 4 to select the operation:
1: Write Image to CFM0
2: Write Image to CFM1 and CFM2
3: Trigger reconfiguration to CFM0(Factory Image)
4: Trigger reconfiguration to CFM1 and CFM 2(App Image)
```

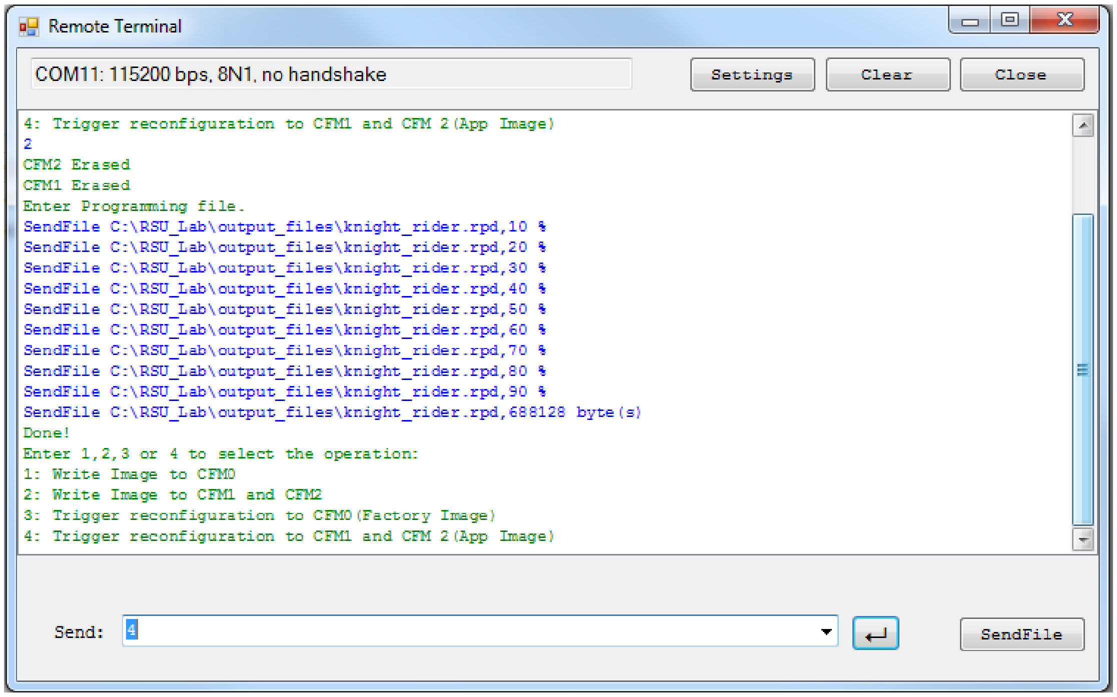
4. Enter "2" in the Send textbox and press enter (this will start the "Write Image to CFM1 and CFM2" operation).

5. Once “Enter Programming file.” appears in the terminal, click on the “SendFile” button and browse for the *knight_rider.rpd* file in the *C:\RSU_Lab\output_files* directory.



```
COM11: 115200 bps, 8N1, no handshake

Hello from Nios II!
Enter 1,2,3 or 4 to select the operation:
1: Write Image to CFM0
2: Write Image to CFM1 and CFM2
3: Trigger reconfiguration to CFM0(Factory Image)
4: Trigger reconfiguration to CFM1 and CFM 2(App Image)
2
CFM2 Erased
CFM1 Erased
Enter Programming file.
```

6. The remote upgrading process will now begin. This process will take a few minutes, and in this time the terminal will become unresponsive. The terminal will update after every 10% of the file that it sends.
7. Once the terminal has sent 100% of the file and the menu has reappeared, send “4” to reboot (reconfigure) the device into the new application image that is stored in CFM1. The board should now be showing the new LED pattern. An alternative to reconfiguring from software would be to change the CONFIG_SEL switch to the “1” position and restarting the board.

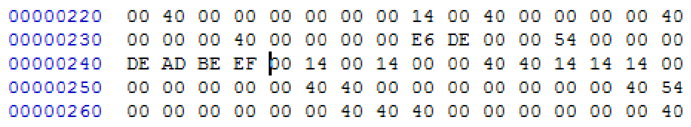


```
COM11: 115200 bps, 8N1, no handshake

4: Trigger reconfiguration to CFM1 and CFM 2(App Image)
2
CFM2 Erased
CFM1 Erased
Enter Programming file.
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,10 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,20 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,30 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,40 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,50 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,60 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,70 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,80 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,90 %
SendFile C:\RSU_Lab\output_files\knight_rider.rpd,688128 byte(s)
Done!
Enter 1,2,3 or 4 to select the operation:
1: Write Image to CFM0
2: Write Image to CFM1 and CFM2
3: Trigger reconfiguration to CFM0(Factory Image)
4: Trigger reconfiguration to CFM1 and CFM 2(App Image)

Send: 4
```

Step 8: Corrupted Images and Auto-Reconfiguration

The Max 10 device provides failsafe booting features. If one of the images fails to boot up (fails CRC or other security checks) then the other image is automatically booted up. In this section we are going to demonstrate this feature of the Max 10 device by corrupting the *knight_rider.rpd*, remote upgrading to this corrupted version, and then trying to reconfigure into the corrupted image. Upon boot up the new image will fail the CRC check and the device will fall back to the default image.

1. Reopen HxD and open up the *C:\RSU_Lab\output_files\knight_rider.rpd* file created previously if it isn't already open.
2. At address 0x240 (Search->Goto, enter 240 and click OK) change the bytes to random garbage values, and File->Save As this file as *knight_rider_corrupt.rpd*.



```
00000220  00 40 00 00 00 00 00 00 14 00 40 00 00 00 00 40
00000230  00 00 00 40 00 00 00 00 E6 DE 00 00 54 00 00 00
00000240  DE AD BE EF 00 14 00 14 00 00 40 40 14 14 14 00
00000250  00 00 00 00 00 40 40 00 00 00 00 00 00 00 40 54
00000260  00 00 00 00 00 00 40 40 40 00 00 00 00 00 00 40
```

3. Open the terminal software if it isn't already open. If you're just now opening up the terminal software, press the KEY0 button to reset the Nios processor and re-display the menu.
4. Follow all of the steps in the section "Perform the Remote System Upgrade" section except send the corrupted image instead.
5. When you attempt to reconfigure into the newly corrupted image, the board should instead fall back to the factory image. Even if you attempt to boot into image 1 using the CONFIG_SEL switch, the default image (counting binary with the LEDs) will always be shown. From this point on, you can perform another Remote System Upgrade but instead send the non-corrupted version of the application image. This feature makes sure that you can't brick your Max 10 device while trying to update it. If the image gets corrupted while trying to update the device, the FPGA will just reboot into a working image at which point the upgrade process can be tried again.

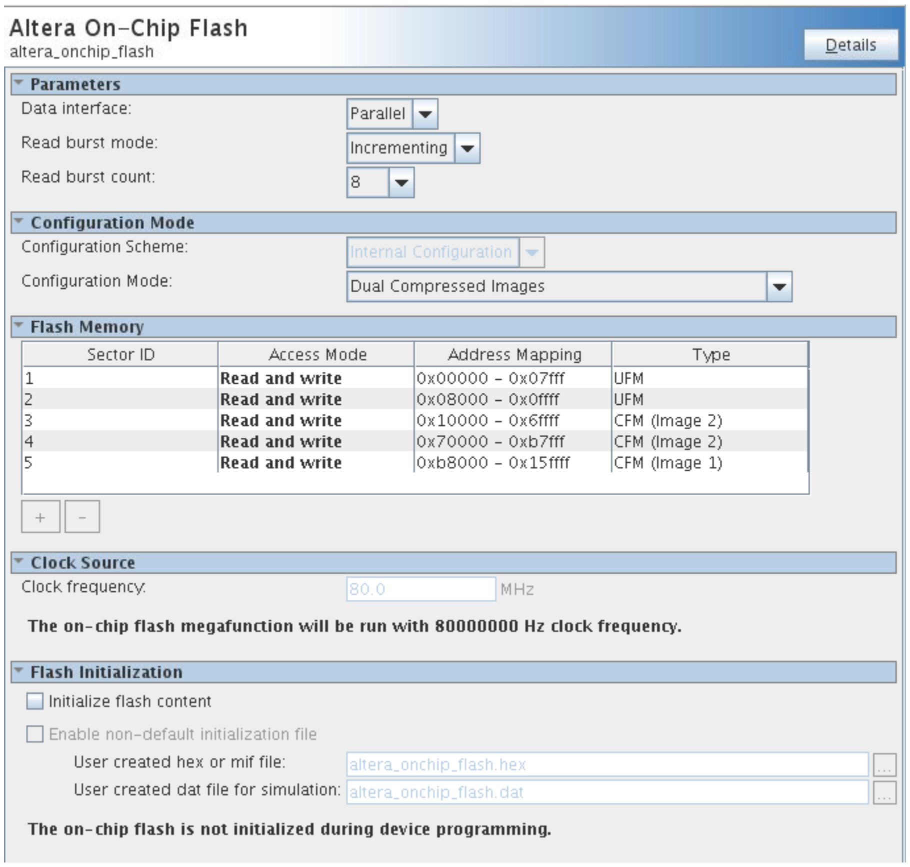
Appendices

Appendix A: Hardware Design Guidelines

When creating the remote system upgrade design, there are a few IP cores that you are going to have to use to complete the design. These IP cores include the On-Chip Flash IP Core and the Dual Configuration IP Core. In the design provided with this lab there's also a Nios II Processor and a UART IP Core but these IP cores are specific to this example remote system upgrade design. There's nothing stopping you from creating a design that has a custom IP core that receives data and immediately sends that data to the On-Chip Flash IP core to write that data into flash without the intervention of a processor. In fact, with the smaller Max 10 devices, this approach could save a substantial amount of logic resources—albeit at the cost of having to develop a custom component. Below you will find descriptions of the two required IP cores, their recommended Qsys settings, and Quartus device and pin options required for dual configuration.

On-Chip Flash IP Core

The Altera On-Chip Flash IP core functions as an interface to perform read, write, or erase operations on the CFM and UFM. The parameter editor this IP core in Qsys shows the access modes available for each of the five sectors (UFM0/1, and CFM0/1/2). Generally, it is recommended to set all of the CFM sectors to the "Hidden" access mode to prevent accidental modification of the configuration bitstream, but when creating a remote system upgrade design, you will need to set whichever CFM regions you want to update to the "Read and Write" access mode. Also, don't forget to set the Configuration Mode to "Dual Compressed Images" (you will also need to set this mode in Quartus, but that is explained in a later section). Below are the settings used in this lab:



Altera On-Chip Flash
altera_onchip_flash

Details

Parameters

Data interface:	Parallel
Read burst mode:	Incrementing
Read burst count:	8

Configuration Mode

Configuration Scheme:	Internal Configuration
Configuration Mode:	Dual Compressed Images

Flash Memory

Sector ID	Access Mode	Address Mapping	Type
1	Read and write	0x00000 - 0x07fff	UFM
2	Read and write	0x08000 - 0x0ffff	UFM
3	Read and write	0x10000 - 0x6ffff	CFM (Image 2)
4	Read and write	0x70000 - 0xb7fff	CFM (Image 2)
5	Read and write	0xb8000 - 0x15ffff	CFM (Image 1)

Clock Source

Clock frequency: 80.0 MHz

The on-chip flash megafunction will be run with 80000000 Hz clock frequency.

Flash Initialization

☐ Initialize flash content

☐ Enable non-default initialization file

User created hex or mif file: altera_onchip_flash.hex

User created dat file for simulation: altera_onchip_flash.dat

The on-chip flash is not initialized during device programming.

Dual Configuration IP Core

You can use the Altera Dual Configuration IP core to access the remote system upgrade block in MAX 10 FPGA devices. The Altera Dual Configuration IP core allows you to trigger reconfiguration once the new image has been downloaded. The only parameter available to the user in Qsys is the clock speed, although it is recommended to run this IP at 80MHz (which is the maximum). In reality, the clock being sent to the remote system upgrade circuitry is half the speed you enter into this parameter (with 40MHz being the maximum).

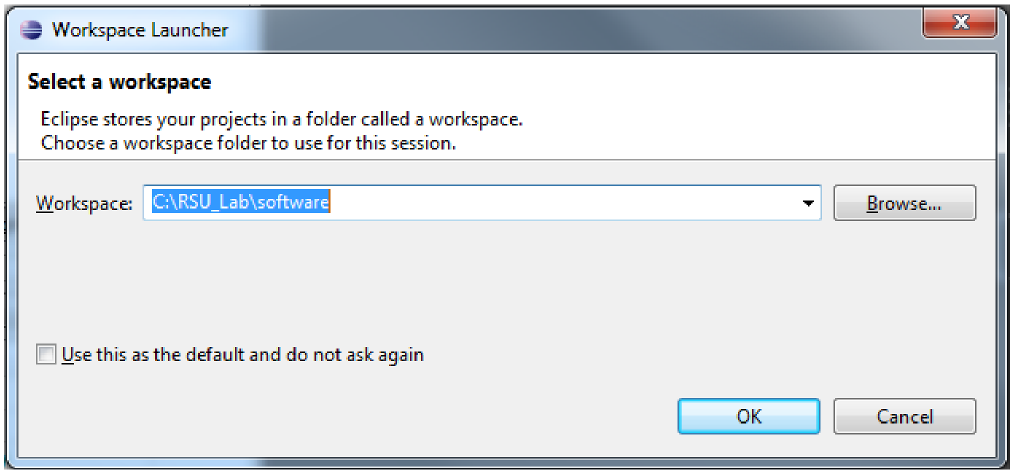
Quartus Device and Pin Options

Even though all of the IP used to perform the remote system upgrade will most likely be generated from within Qsys, there are still parameters that need to be set in Quartus—that if ignored, will cause an error in the fitter stage of compilation. These settings can be accessed by clicking on Assignments->Device then at the dialog that appears, clicking on “Device and Pin Options.” On the General tab, if it isn’t already checked, you might want to check the “Auto-restart configuration after error” option. This will automatically try to boot up a different image if one image fails to boot up (for instance, because of a CRC error). On the Configuration tab, make sure to select the exact same Configuration Mode that you selected in the On-Chip Flash IP parameter editor. If you fail to do so, the fitter will throw an error. If you would like to use the CRC error detection features, then make sure to review the settings in the Error Detection CRC category. All of these settings are described in detail in the Max 10 Handbook.

Appendix B: Extracting the Software Archive

The following steps will walk you through restoring the included example software project. This is the same project used to create the programming files (see Appendix C) for the Nios II firmware used in this lab. This software contains both an application project and a BSP project meant to run on a 10M50 device. The meat of the example is contained within the *remote_update.c* file inside of the RSU_firmware application project.

1. Open up the Nios II Software Build Tools by clicking on Tools->Nios II Software Build Tools for Eclipse.
2. When the workspace selection window appears, browse to *<design-example-installation-directory>\software* and hit OK. For instance, if you extracted the example into *C:\RSU_Lab*, then the following image shows where your workspace will be located:

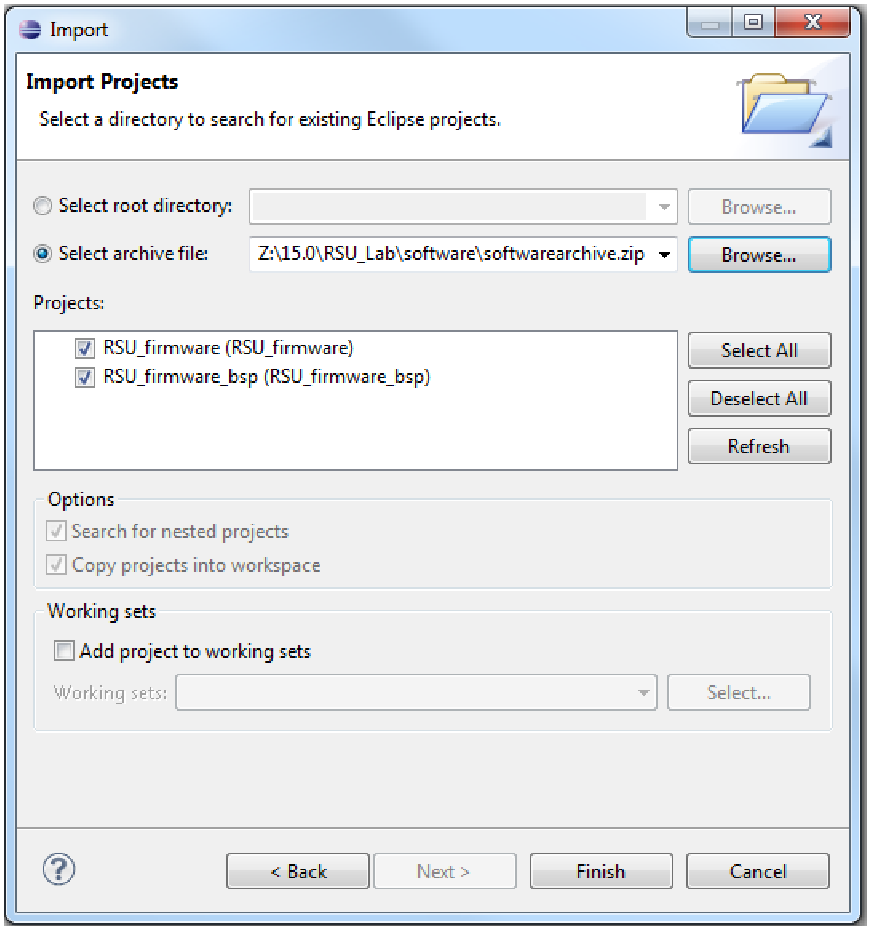


Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\RSU_Lab\software

3. Next, go to File->Import… and then select General->Existing Projects into Workspace. Click Next.

4. At this window, select the “Select archive file” option and browse to the “softwarearchive.zip” file within the software folder. Click Finish.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: [ ] Browse...

[● Select archive file:] Z:\15.0\RSU_Lab\software\softwarearchive.zip Browse...

Projects:

- [✓] RSU_firmware (RSU_firmware)
- [✓] RSU_firmware_bsp (RSU_firmware_bsp)

Select All / Deselect All / Refresh

Options

- [✓] Search for nested projects
- [✓] Copy projects into workspace

Working sets

- [ ] Add project to working sets

Working sets: Select...

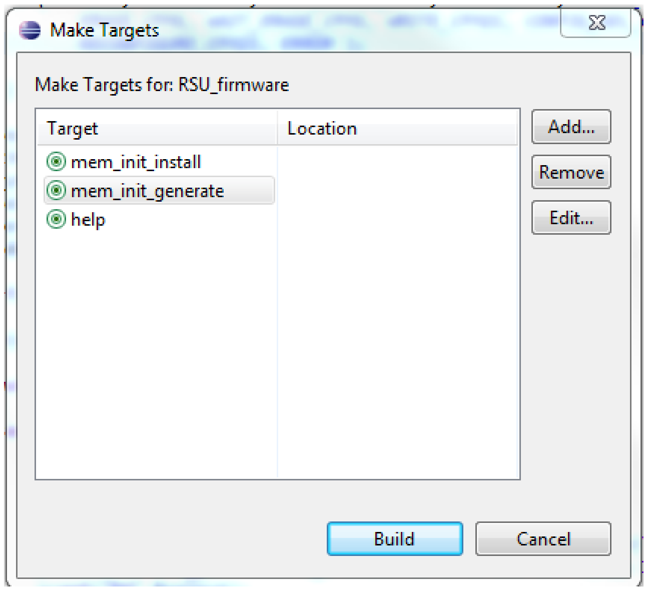
< Back | Next > | Finish | Cancel

5. Once the projects have been imported you can open up the example code by opening the *remote_update.c* file within the RSU_firmware application project.
6. Build the project by going to Project->Build All
7. Refer to Appendix C to build the project and create a *.pof* file for programming into flash memory.

Appendix C: Generating Programming Files for the Nios II Firmware

The process required to take a Nios II eclipse project and turn it into a *.pof* file for programming can be less than obvious. This section will guide you through taking the remote upgrade software and generating the programming files that were used to program the external QSPI flash memory at the beginning of the lab.

1. Follow the steps in Appendix B to un-archive and open the software project.
2. Build the project by clicking Project->Build All. This process will generate an *.elf* executable file which can be used to program the device over JTAG. To program flash memory, we will need a *.hex* file.
3. To convert the *.elf* executable into a *.hex* file, right click on the RSU_firmware project and under the Make Targets menu, select Build...
4. At the dialog box that appears, select mem_init_generate and click Build. This process will create a different *.hex* file for each memory device used within the system. For this design, that would be the on-chip flash, on-chip memory, and the quad-spi external flash. The hex file *generic_quad_spi_controller.hex* contains both the code for this design as well as a boot copier that will copy the code from flash to the on-chip memory at bootup.


Make Targets

Make Targets for: RSU_firmware

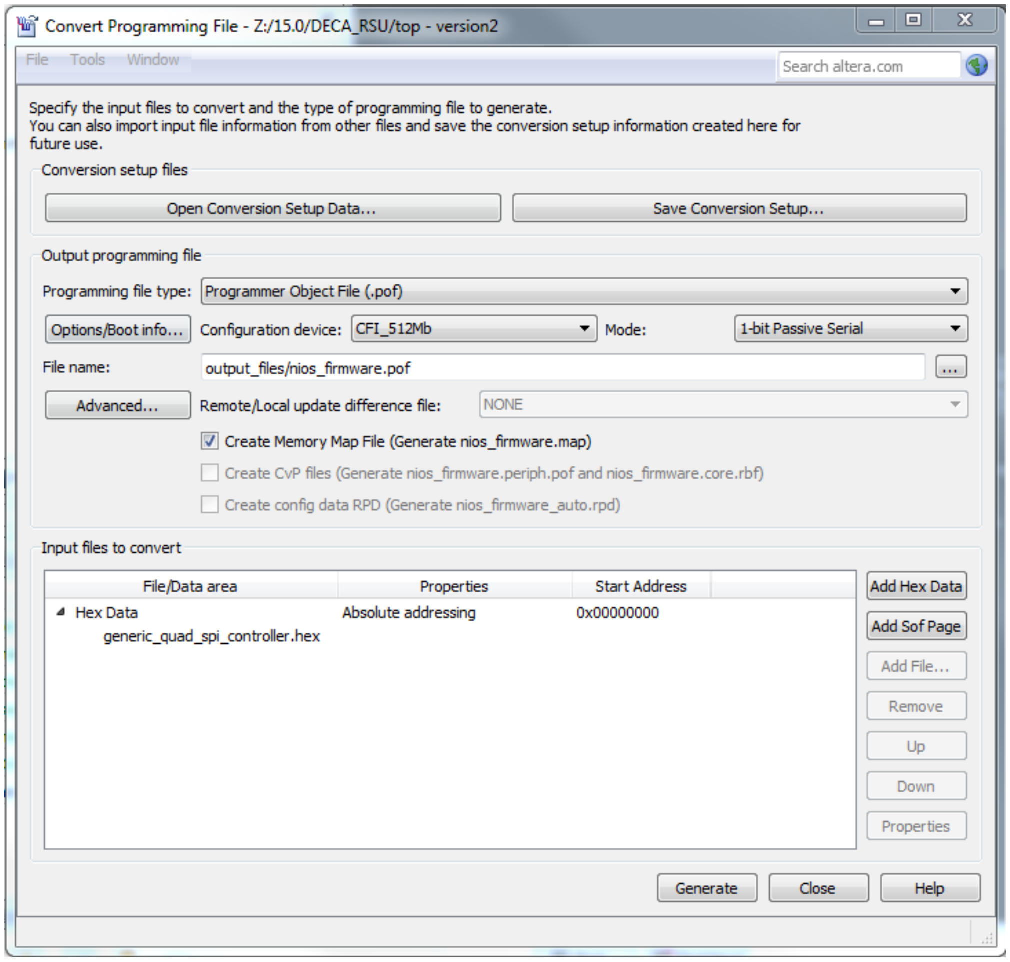
Target	Location
mem_init_install	
mem_init_generate	
help	

5. Switch back to Quartus and open up Files->Convert Programming Files...

   NOTE: A quartus.ini file is needed to generate a correct *.pof* file. The lab source code already has this file in the project directory, but for your own projects, copy and paste it from the lab's source to the Quartus II tool directory or your own project directory. If the quartus.ini file is copied into the project directory, open the Quartus project in Quartus II before opening the Convert Programming File GUI. The only thing that should be inside of the quartus.ini file is a single line containing:

```
PGMIO_SWAP_HEX_BYTE_DATA=ON
```

6. Under "Configuration Device" select "CFI_512Mb".
7. Change the File name to *output_files/nios_firmware.pof*.
8. Click on the "SOF Data" that was auto-generated and click on Remove in the sidebar.
9. Click on the Add Hex Data button in the sidebar.
10. Click on the "…" button under Hex File, and open up *<RSU_Lab_Directory>\software\RSU_firmware\mem_init\generic_quad_spi_controller.hex*.
11. Leave the rest of the settings in this dialog as the defaults, and click OK.
12. Ensure that your settings look like the following and then click Generate.



Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Output programming file

Programming file type: Programmer Object File (.pof)

Configuration device: CFI_512Mb

Mode: 1-bit Passive Serial

File name: output_files/nios_firmware.pof

Remote/Local update difference file: NONE

[✓] Create Memory Map File (Generate nios_firmware.map)

[ ] Create CvP files (Generate nios_firmware.periph.pof and nios_firmware.core.rbf)

[ ] Create config data RPD (Generate nios_firmware_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
Hex Data	Absolute addressing	0x00000000
generic_quad_spi_controller.hex		

13. Once the *.pof* file is generated, you can follow the steps in Step 3 to program this file into the external QSPI flash.