DECA Linux Tutorial

For the MAX® 10 DECA FPGA Evaluation Kit

Version 15.1, Revision 1 **12/17/2015**

TABLE OF CONTENTS

DECA LINUX TUTORIAL

Overview: This tutorial walks through bringing up Linux on the DECA development kit. There are two main sections to cover. The first gets you to a Linux prompt by using pre-built binaries. The second walks through building these binaries from scratch.

This tutorial accompanies the **deca_linux_package.zip** files required to bring Linux up on DECA.

The contents of **deca_linux_package.zip** include:

FPGA_files

- **deca_linux_ghrd.zip**
- **deca_linux_ghrd.sof**
- **deca_linux_ghrd.pof**
- **deca_qpfl.sof**
- **max10_pfl_deca.qar**

deca_linux-socfpga_patch.zip

deca_buildroot_patch.zip

vmlinux_rootfs.pof

vmlinux.hex

rootfs.hex

quartus.ini

SECTION 1. USING LINUX ON DECA

1.1 Getting Started

This document assumes the reader knows how to use the Linux shell and GIT.

1.2 Hardware - Information

1.2.1 **Review the DECA Development Platform**

Review the components on the DECA board.

There are many components on the DECA board including the LEDs, CapSense Buttons, USB, Ethernet, HDMI and MIPI Interfaces. There is a gesture sensor on the back of the board and an Audio Line In/Out jack on the top.

We have simplified this FPGA design to include the main features required to support Linux. These I/O and interfaces are 10/100 Ethernet, DDR3 SDRAM, QSPI flash, LEDs, push-buttons, and switches

1.2.2 **DECA, User Manual**

A copy of the DECA User Manual can be found here:
http://www.alterawiki.com/uploads/7/7a/DECA_User_manual_rev1.pdf

1.2.3 **DECA, Wiki**

Other DECA tutorials and general information can be found at this AlteraWiki page:
http://www.alterawiki.com/wiki/DECA

1.3 Demonstrate Usage

1.3.1 Adding the Quartus.ini file to your install path

The **quartus.ini** variable is required to program the correct contents to the QSPI flash.

1.3.1.1 Copy **quartus.ini** to **altera/15.1/quartus/bin64**

1.3.2 Loading the Parallel Flash Loader design

You will need to configure the MAX 10 FPGA with the parallel flash loader design before programming the Quad SPI flash. The parallel flash loader design bridges the FPGA JTAG pins to the QSPI pins so that a standard USB-Blaster connection can program the QSPI flash. We will eventually be programming the QSPI with the root file system and the kernel image.

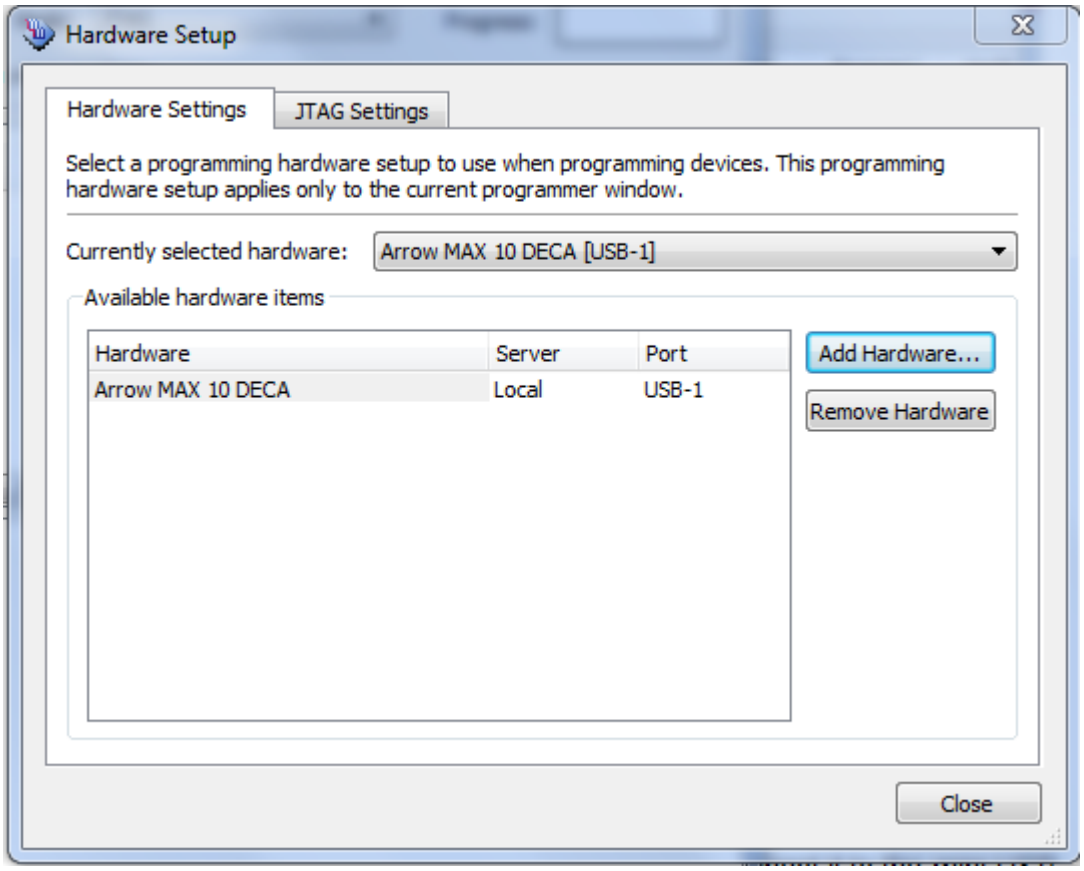
The following file is required for this section:

FPGA_files/deca_qpfl.sof	The parallel flash loader

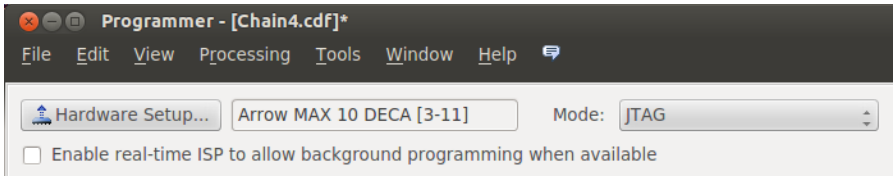
1.3.2.1 Connect your DECA board to your PC using a USB cable. Be sure to connect it to the mini-USB connector labeled **J10 UB2** (on the bottom right of the board).

1.3.2.2 Launch the Quartus Prime software in the host PC and open the Programmer via Tools → Programmer

1.3.2.3 In the Programmer window, click Hardware Setup and double-click the Arrow MAX10 DECA entry in the Hardware pane. The Currently selected hardware drop-down should now show Arrow MAX10 DECA [USB-1]. Depending on your PC, the USB port number may be different. Click Close.



1.3.2.4 Upon completion, the Hardware Setup field should look something like this:



1.3.2.5 Click “Auto Detect”. When requested, select the “10M50DA” device and click “OK”

1.3.2.6 Right click the MAX10 device and select “Edit → Change File” in the popup menu

1.3.2.7 Choose the file: **deca_qpfl.sof**

1.3.2.8 Check the Program/Configure checkbox

1.3.2.9 Click “Start” to start programming.

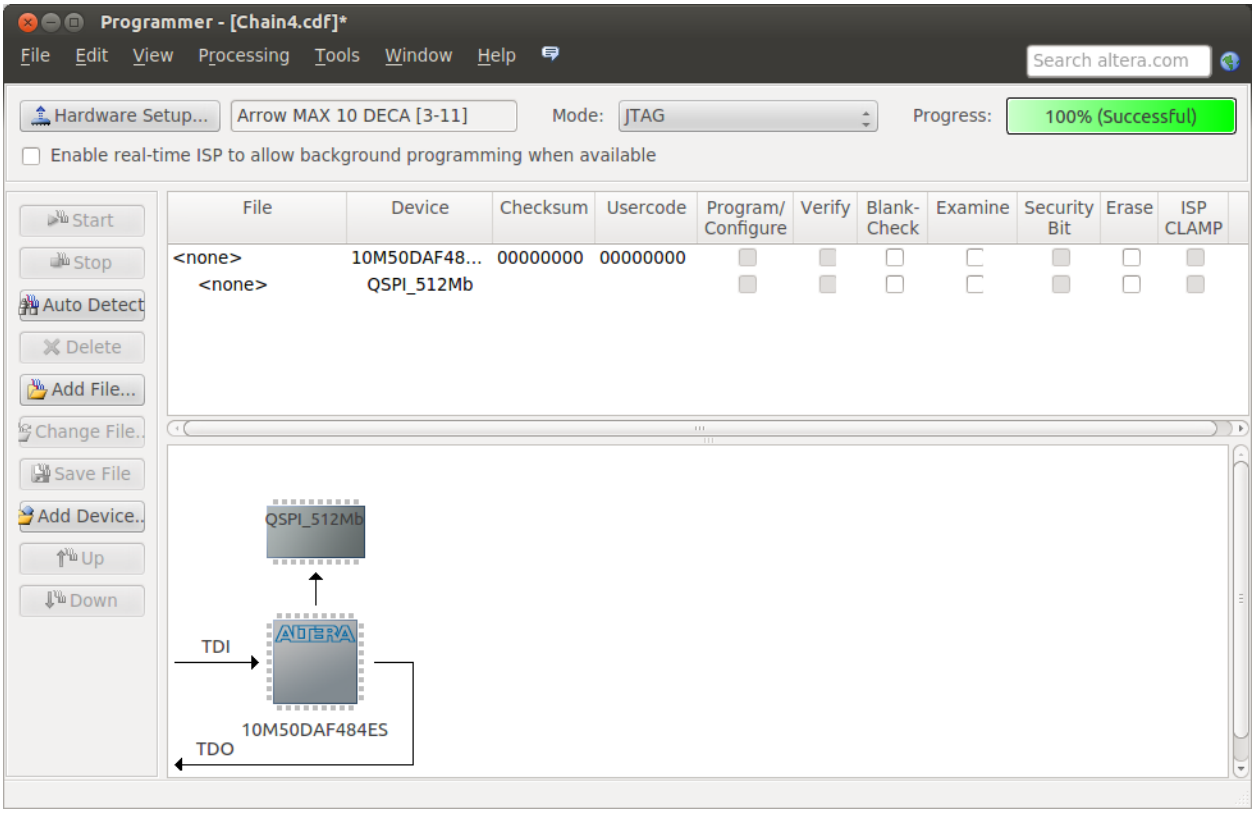
1.4 Program POF (linux image) into Quad SPI Flash

The following file is required to execute this procedure

vmlinux_rootfs.pof	The Linux image containing and the kernel image and the root filesystem image.

1.4.1.1 In the Programmer, Click “Auto Detect”. The Programmer will ask you to confirm if you want to overwrite the existing settings. Click “Yes”

1.4.1.2 Your Programmer window will look like this:



1.4.1.3 Right click the QSPI device and select “Edit → Change File” in the popup menu

1.4.1.4 Choose the file: **vmlinux_rootfs.pof**.

1.4.1.5 Check both **rootfs.hex** and **vmlinux.hex** boxes under Program/Configure

1.4.1.6 Click “Start” to start programming. This process will take approximately five minutes depending on the speed of your system.

1.5 Program FPGA’s CFM

The following file is required to execute this procedure

FPGA_files/deca_linux_ghrd.pof	The FPGA bitstream file required to program the CFM (Configuration Flash Memory) of the MAX 10 FPGA

1.5.1.1 Right click the MAX10 device and select “Edit → Change File” in the popup menu

1.5.1.2 Browse to the folder where the material was copied and select: **deca_linux_ghrd.pof**

1.5.1.3 In the Programmer window, under Program/Configure, Check the MAX10’s CFM0 box

1.5.1.4 Click “Start” to start programming

1.6 Boot with Linux

In order to connect to the Linux system, login with this username and password:

Username	**root**
Password	**deca**

1.6.1 **Connect using the console**

1.6.1.1 In a terminal, execute the following command

```
/altera/15.0/nios2eds/nios2_command_shell.sh
```

1.6.1.2 Then connect the DECA board to the computer using a USB cable connected to the J10 connector. Immediately after, execute the following command in the previously opened terminal.

```
nios2-terminal
```

Launching the Nios II terminal is required because the JTAG UART is a blocking device where Linux will stop launching if the uart’s TX buffer is full. This causes the kernel to hang because the printk messages aren’t being flushed out of the uart’s FIFO. On the development kit, we resolve this issue by launching the nios2-terminal to display the printk messages and keep the TX buffer from filling.

1.6.1.3 The nios2-terminal will display all data received from the JTAG UART. It first display the boot traces and after that, you will find the prompt for the user to login.

1.6.1.4 Go ahead and log in using the username and password described above.

There are some drawbacks to using the JTAG_UART as the driver does not support many of the control characters one is accustomed to when using the Linux prompt. Options such as the up-arrow, or the tab key to complete a filename are not available when using this interface. As such, we suggest users connect to the board using SSH

1.6.2 Connect using SSH

1.6.2.1 In a linux terminal, execute the following command. Note that you will need to use the IP address assigned by the DHCP server to the DECA board. You can find the assigned IP address by using the **`ifconfig`** command before launching **`ssh`**

```
ssh –l root 192.168.0.123
```

If you have a Windows machine, you can still run SSH. The easiest way to do this is from the Nios II 15.1 Command Shell. Other SSH tools are available for download but fall beyond the scope of this tutorial.

1.7 Accessing GPIO

The DECA board uses 12 GPIOs. Eight of them are connected to LED, two to push buttons and two to switches. The BSP allow accessing them very easily using the standard Linux way.

1.7.1 Turning on a LED

1.7.1.1 In a terminal, execute the command

```
echo 1 > /sys/class/leds/led2/brightness
```

Note: You can replace **`led2`** using **`led3`** to **`led7`**. Note that **`led0`** and **`led1`** are reserved as they are used to indicate power state and a heartbeat for Linux.

1.7.1.2 To turn off the led, you need to use the above command, but replace the **`1`** with a **`0`**

1.7.2 Reading the button's states or the switches' position

1.7.2.1 In a terminal, execute these commands to read a gpio input

```
echo 244 > /sys/class/gpio/export

cat /sys/classes/gpio/gpio244/value
```

The value displayed by the output of cat is the status of the push-button or switch. The table below maps which number is associated to each device:

GPIO	Number
Button 0	**246**
Button 1	**247**
Switch 0	**244**
Switch 1	**245**

1.7.2.2 To release the input, you will use the following command:

```
echo 244 > /sys/classes/gpio/unexport
```

THIS ENDS THE USING LINUX ON DECA SECTION OF THIS TUTORIAL

SECTION 2. REBUILDING THE SOURCE FILES

2.1 Building the FPGA Design

Creating the Nios II Embedded System and designing the FPGA fall beyond the scope of this tutorial. To learn more about creating an embedded system with Nios II, refer to the DECA Wiki page:

http://www.alterawiki.com/wiki/DECA#Lab_3:_Embedded_Systems_Lab

Having said that, you can safely add components to the Qsys system, regenerate it, recompile the FPGA design, and use the new FPGA configuration files to either configure to the FPGA directly via JTAG (via the SOF File) or program the MAX 10 FPGA's configuration flash memory (CFM).

2.2 Building QSPI flash file with Nios II Linux MMU kernel image

In order to run Linux on DECA, you need two binary files. These files are the kernel image and the root file system image. To create these two binaries, you will need to install the development tools required on a Linux machine. It is possible to use a virtualization tool such as VisualBox or VMware running on a Windows Computer.

The documented procedure has been tested on two different Linux distributions: "Ubuntu 12.04" and "CentOS 6.7".

The following sections will guide you through the installation of Linux operating system, the installation of the OS dependent required packages, and wrap up with the installation of the development tools.

2.3 Install Linux (host computer)

You can choose to use either CentOS 6.7 (Step 2.3.1), or Ubuntu 12.04 (Step 2.3.3)

2.3.1 **CentOS 6.7 (64 bit)**

2.3.1.1 Install CentOS 6.7 selecting the "Software Development Workstation" install.

2.3.2 **Install 32 bit library**

2.3.2.1 In a terminal, execute the following command as "root".

```
yum install glibc.i686 gtkz.i686 alsa-lib.i686 libXt.i686
```

2.3.3 Ubuntu 12.04 (64 bit)

2.3.3.1 Install Ubuntu 12.04 using the default configuration.

2.3.4 Install 32 bits libraries

2.3.4.1 In a terminal, execute the following commands as “root”.

```
apt-get update

apt-get install libgtk2.0-0:i386 libxtst4:i386 gtk2-engine-murrine:i386
lib32stdc++6 libxt6:i386 libdbus-glib-1-2:i386 libasound2:i386
```

2.4 Install XULRunner 1.9.2

2.4.1.1 Download the file **`xulrunner-3.6.26.en-US.linux-i686.tar.bz2`** from the page

https://developer.mozilla.org/en-US/docs/Mozilla/Projects/XULRunner/1.9.2

2.4.1.2 Execute the following commands as “root”.

```
cd /opt

mkdir xulrunner

cd xulrunner

tar –xf /folder/xulrunner-3.6.26.en-US.linux-i686.tar.bz2

mv xulrunner 1.9.2

cd 1.9.2

./xulrunner –register-global
```

2.5 Install Sourcery CodeBench Lite 2013.05-43 for NIOS II GNU/Linux

2.5.1.1 Download the file **`sourceryg++-2013.05-43-nios2-linux-gnu.bin`** from the page:

https://sourcery.mentor.com/GNUToolchain/release2499

2.5.1.2 Execute the following commands as “root”.

```
chmod 777 sourceryg++-2013.05-43-nios2-linux-gnu.bin

./sourceryg++-2013.05-43-nios2-linux-gnu.bin
```

2.6 Install Quartus 15.1.0

2.6.1.1 Visit http://dl.altera.com to download the Quartus Prime Lite Edition for MAX 10.

2.7 Compile the root file system

2.7.1 Get sources

2.7.1.1 The provided patch contains the following new files.

- **configs/**
  - **deca_defconfig**
- **env/**
  - **nios2_create_hex.sh**
  - **nios2_make.sh**

2.7.1.2 Before executing the procedure, please download the following file.

deca_buildroot_patch.zip	The patch to apply in order to support the DECA board

2.7.1.3 In a terminal, navigate to the directory where the repository needs to be cloned and execute the following command.

```
git clone http://git.buildroot.net/git/buildroot.git buildroot
```

Note: The last instance of **buildroot** in the command can be replaced with any other folder name you want to use locally.

2.7.1.4 Using GIT, create a new local branch from tag “2015.02”.

2.7.1.5 See the Appendix – Applying patch for more information on how to apply the patch.

2.7.2 Compile

2.7.2.1 In a terminal, navigate to the project directory and execute the following command.

```
./env/nios2_make.sh deca_defconfig

./env/nios2_make.sh
```

2.7.2.2 Output files:

output/images

- **rootfs.cpio**
- **rootfs.jffs2**
- **rootfs.tar**

2.8 Generate HEX file

2.8.1.1 In a terminal, navigate to the project directory and execute the following command.

```
./env/nios2_create_hex.sh
```

Output files:

- **output/images/rootfs.hex**

2.9 Compile the Linux kernel

There are two options you can choose to take from here:

2.9.1 Option A – Get source from private repository

2.9.1.1 In a terminal, navigate to the directory where the repository needs to be cloned and execute the following command.

```
git clone https://github.com/ArrowDECA/linux_for_deca deca-linux
```

Note: The parameter: **deca-linux** in the command can be replaced with any other folder name you want to use locally.

2.9.1.2 Using GIT, create a new local branch from tag “DECA_Linux.1.0.0”.

2.9.2 Option B – Get source from public repository and apply downloaded patch

2.9.2.1 The provided path contains the following new files

- **arch/nios2/**
  - **boot/**
    - **.gitignore**
    - **dts/**
      - **deca_devboard.dts**
  - **configs/**
    - **deca_defconfig**
  - **kernel/**
    - **.gitignore**
- **env/**
  - **nios2_create_hex.sh**
  - **nios2_make.sh**

2.9.2.2 The provided patch also contains changes to the following files

- **.gitignore**
- **arch/nios2/boot/compressed/**
  - **console.c**

- **drivers/tty/serial/Kconfig**

2.9.2.3 The below procedure requires the following file

deca_linux-socfpga_patch	The patch to apply in order to support the DECA board

2.9.2.4 In a terminal, navigate to the directory where the repository needs to be cloned and execute the following command.

```
git clone https://github.com/altera-opensource/linux-socfpga.git deca-linux
```

2.9.2.5 Using GIT, create and checkout a local branch from tag “ACDS15.0_REL_GSRD_RC2”.

2.9.2.6 See the Appendix – Applying patch for more information on how to apply the patch.

2.9.3 Compile

2.9.3.1 Verify is the Altera installation folder is the correct one in file “make.sh”.

```
./env/nios2_make.sh deca_defconfig

./env/nios2_make.sh vmlinux
```

Output files:

- **vmlinux**

2.9.4 Generate HEX file

2.9.4.1 In a terminal, navigate to the project directory and execute the following command.

```
./env/nios2_create_hex.sh
```

Output files:

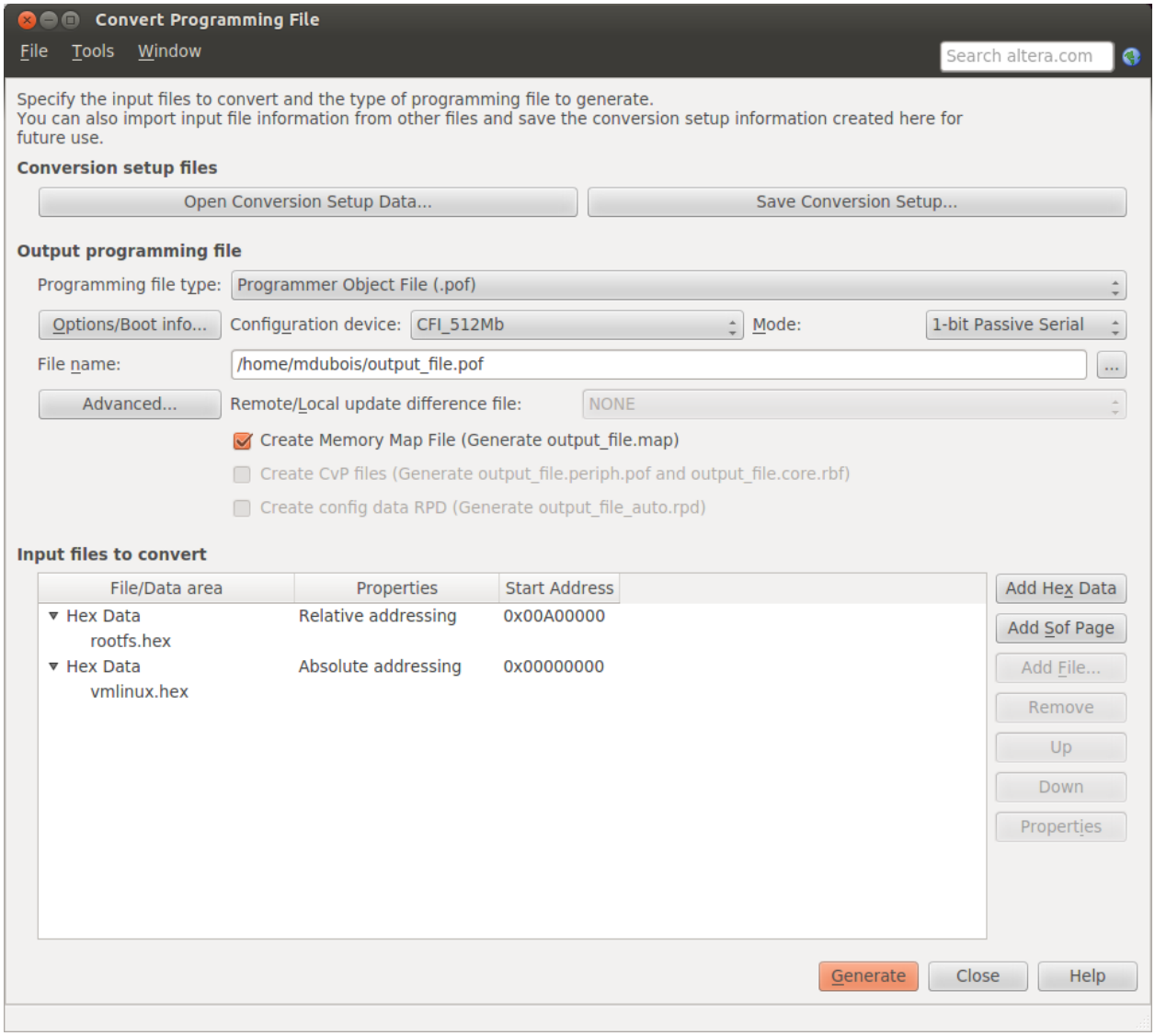
- **vmlinux.flash**
- **vmlinux.hex**

2.10 Creating the QSPI flash file from both HEX files

These files are prebuilt to save you from regenerating these:

rootfs.hex	The root file system image
vmlinux.hex	The kernel image (including the nios2 boot loader)

2.10.1.1 Start “Quartus Prime” and, in the file menu, select “Convert Programming Files…”. This opens the “Convert Programming File” window.



2.10.1.2 Set the “Programming file type” to “Programmer Object File (.pof)”;

2.10.1.3 Set the “Configuration device” to “CFI_512Mb”;

2.10.1.4 Set the “Mode” to “1-bit Passive Serial”;

2.10.1.5 Set the “File name” as needed to generate the file in the folder of your choice;

2.10.1.6 Remove any data entry already present in the “Input files to convert” list;

2.10.1.7 Click “Add Hex Data”

2.10.1.8 Select “Relative addressing”;

2.10.1.9 Enable the “Relative address” and enter “00a00000”;

2.10.1.10 Navigate and chose the previously created file “rootfs.hex”;

2.10.1.11 Click “OK”.

2.10.1.12 Click “Add Hex Data”

2.10.1.13 Select “Absolute addressing”;

2.10.1.14 Navigate and chose the previously created file “vmlinux.hex”;

2.10.1.15 Click “OK”.

2.10.1.16 Click “Generate”

SECTION 3. APPENDIX – APPLYING PATCH

Downloaded archives contains the GIT formatted patches and the final version of modified files. Use one of the two following procedures.

3.1.1 **Procedure A - Apply the patch using GIT**

3.1.1.1 Copy the folder “patches” out from the downloaded archive (uncompressing it);

3.1.1.2 Open a terminal navigate to the project folder and execute the following command

```
git apply /folder/*
```

Note: Don’t forget to replace “folder” with the name of the folder where the patch resides.

3.1.2 **Procedure B - Apply the patch using file copy**

3.1.2.1 Copy all the files from the folder “files” of the downloaded archive to the project folder.

SECTION 4. SUPPORT

There are many Linux based resources available at RocketBoards.org. For DECA support, please contact deca@arrow.com.