Nios II Processor Booting From Altera Serial Flash (EPCQ)

2015.05.04

AN-736 Subscribe Send Feedback

The Altera® Nios® II processor is a soft processor that supports all Altera SoC and FPGA families. This document describes how to boot a Nios II processor from Altera EPCQ flash memory (EPCQx1, EPCQx4) using an Altera serial flash controller. The Altera Serial Flash Controller supports configuration and programming of EPCQx4, EPCQx1 and EPCS flash in FPGA design for Altera Arria® V, Arria 10, Cyclone V and Stratix V devices.

Prerequisites

You are required to have knowledge in instantiating and developing a system with a Nios II processor. Altera recommends that you go through the online tutorials and training materials provided on **Altera's website** before using this application note.

Related Information

- **AN 717: Nios II Gen 2 Hardware Development Tutorial**
  Refer to this document for a step-by-step procedure to build a Nios II Gen 2 soft core processor system.
- **Embedded Peripherals IP User Guide**
  Refer to this document for more information about the Altera serial flash controller.
- **Quad-Serial Configuration (EPCQ) Devices Datasheet**
  Refer to this document for more information about quad-serial (EPCQ) devices.

Definitions

The following table lists acronyms used in this application note along with their corresponding definitions.

Table 1: AN 736 Acronym Definitions

Acronym	Description
ACDS	Altera Complete Design Suite
API	Application Programming Interface
CFI	Compact Flash Interface
EPCQ	Altera Quad SPI flash device
HAL	Hardware Abstraction Layer

Acronym	Description
HEX	Hexadecimal file: this is an ASCII text file with the extension of .hex. It stores the initial memory values for a memory block.
I/O	Input/Output
Memcpy	Memory copy
OCRAM	On-chip RAM
RAM	Random Access Memory
SBT	Software Build Tools
SoC	System on a Chip
SOF	SRAM Object Files
UFM	User Flash Memory
XiP	Execute in Place

Description of the Altera Serial Flash Controller

The Altera Serial Flash Controller with Avalon interface allows Nios II processor systems to access an Altera EPCQ flash memory, which supports standard, quad and single- or dual-I/O mode. The Nios II processor SBT supports the Nios II booting from the Altera Serial Flash Controller. In addition, a Nios II hardware abstraction layer (HAL) driver is available for the Altera Serial Flash Controller that allows an application to read, write, or erase flash.

Scenarios for Booting Nios II from EPCQ Flash

Booting a Nios II processor from an EPCQ flash has a similar flow to booting a Nios II processor from a CFI flash.

The Nios II processor supports the following two boot options using EPCQ flash:

- The Nios II processor application executes in place from EPCQ flash.
- The Nios II processor application is copied from EPCQ flash to RAM using a boot copier.

Table 2: Summary of Nios II Processor Boot Options Using EPCQ Flash

Boot Option	Application Code Stored Location	Application Run-time Location	Method
Nios II processor application executes in place from EPCQ flash	EPCQ	EPCQ (XIP) + OCRAM (for writable data sections)	Using `alt_load()` function
Nios II processor application is copied from EPCQ flash to RAM using boot copier	EPCQ	OCRAM/ External RAM	Using a memcpy-based boot copier

Nios II Processor Application Execute-In-Place from EPCQ Flash

The execute-in-place option is suitable for Nios II processor applications which require limited on-chip memory usage.

The `alt_load()` function operates as a mini boot copier which initializes and copies only the writable memory sections to on-chip RAM (OCRAM). The code section (**.text**), which is a read-only section, remains in the Altera EPCQ flash memory region. Retaining the read-only section in EPCQ helps to minimize RAM use but may limit the code execution performance. The Nios II processor application is programmed into the EPCQ flash.

The Nios II processor reset vector points to the EPCQ flash to allow code execution after the system resets. If you are debugging the application using the source-level debugger, you must use a hardware breakpoint because the EPCQ cannot efficiently support random memory access.

Nios II Processor Application Copied from EPCQ Flash to RAM

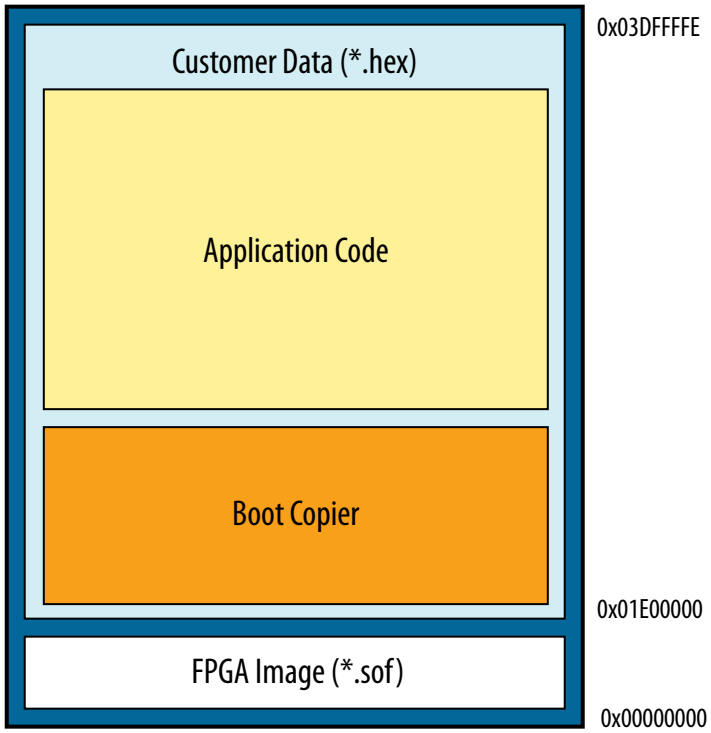
Using a boot copier to copy the Nios II application from EPCQ flash to RAM is suitable when multiple iterations of application software development and high system performance are required. Altera recommends this solution for Nios II processor booting from Altera Serial flash (EPCQ).

The Nios II SBT tool automatically adds the Nios II processor memcpy-based boot copier to the system when the executable file (**.elf**) is converted to the memory initialization file (**.hex**). The boot copier is located at the base address of the HEX data, followed by the application.

For this boot option, the Nios II processor starts executing the boot copier software upon system reset which copies the application from the EPCQ to the internal or external RAM. Once this is complete, the Nios II processor transfers the program control over to the application.

Figure 1: EPCQ Flash Layout When Using Boot Copier

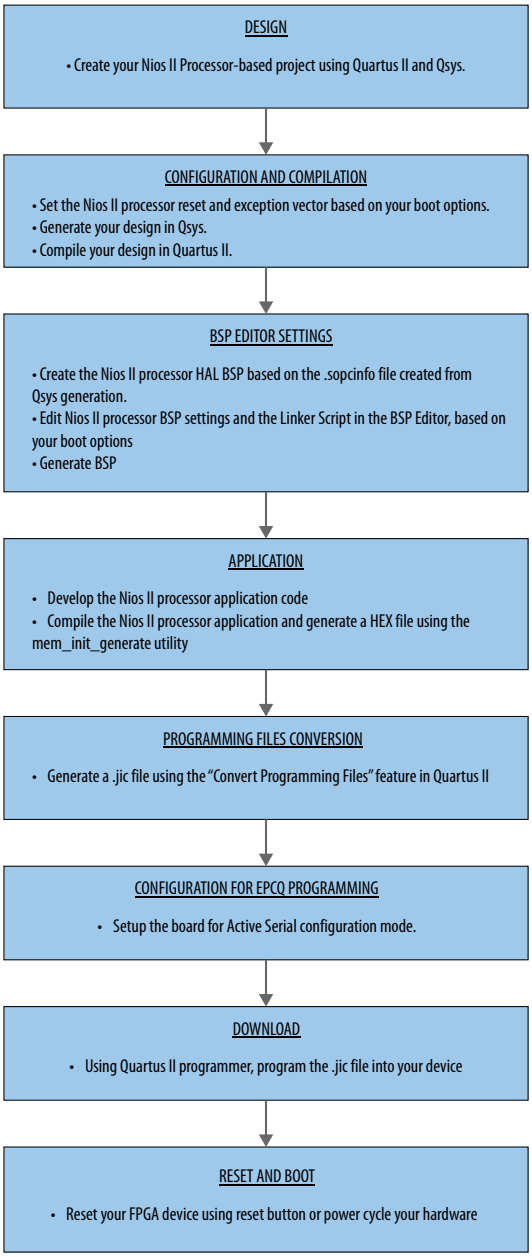
In this figure, the size of the FPGA image (***.sof**) is estimated to be less than 0x01E00000. Customer data (***.hex**) is set to start at address 0x01E000000 and the size of the ***.hex** file is assumed to be 0x01FFFFFE.



Nios II Processor Configuration and Boot Flow

This section highlights the design, configuration and BSP settings for the supported boot scenarios mentioned in this document.

Figure 2: Configuration and Booting Flow



Steps to Build a Bootable System

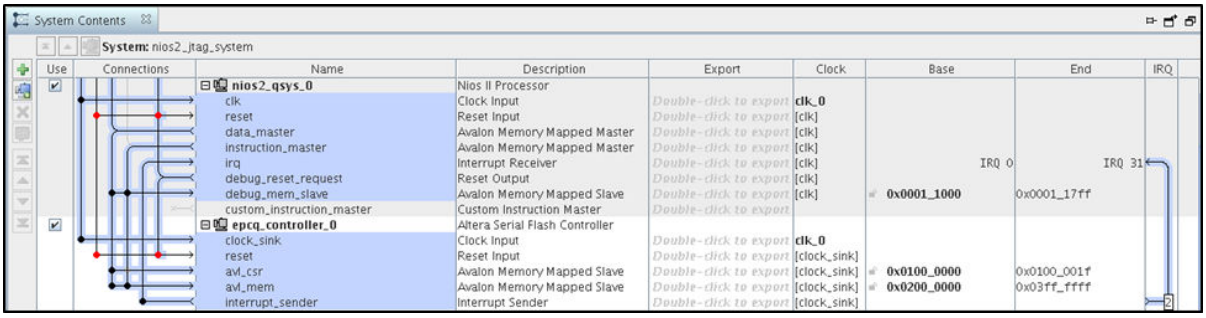
The following sections describe a step-by-step method for building a bootable system for a:

- Nios II processor application executing in place from EPCQ flash.
- Nios II processor application copied from EPCQ flash to RAM using a boot copier.

Design

1. Create your Nios II processor project using Quartus II and Qsys.
2. Make sure the Altera Serial Flash Controller IP is added in your Qsys system. Refer to the diagram below for the IP connection in Qsys.

Figure 3: Altera Serial Flash Controller IP Connections in Qsys



Configuration and Compilation

To proceed with configuration and compilation, you must double click on the Nios II Processor row in the **System Contents** window in Qsys to open the Nios II processor **Parameters** window.

From here, proceed to the "Reset and Vector Settings" section that is applicable to your boot method.

Reset and Exception Vector Settings for Nios II Execute-In-Place Method

1. Under **Reset Vector** in the **Vectors** tab, select EPCQ (**epcq_controller_0.avl_mem**) in the **Reset vector memory** drop-down menu and type the reset vector offset in the **Reset vector offset** entry box. The reset vector must be the base address of your application. In this example, it is 0x01E00000.

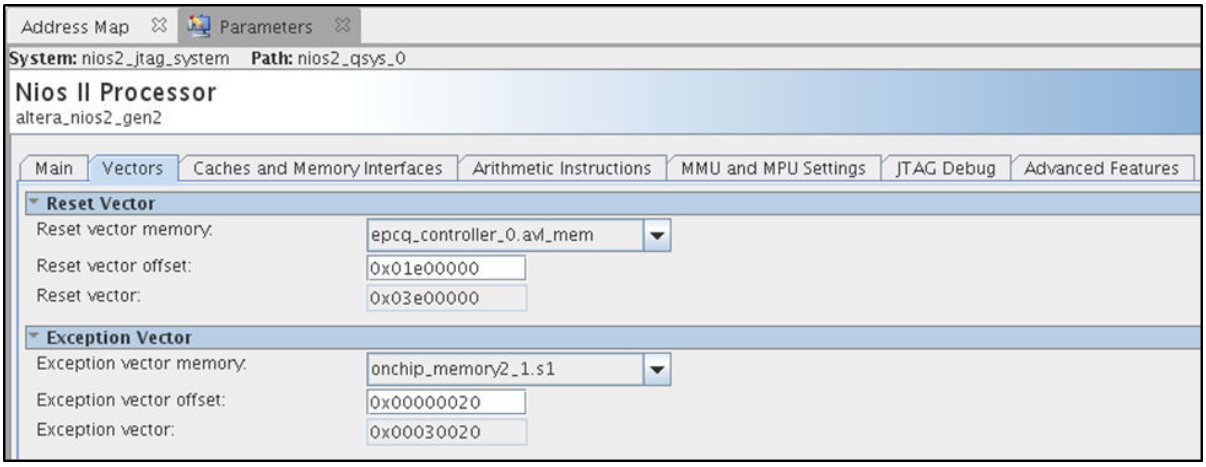
   Note: Reset vector offset varies according to your ***.sof** image size. The offset is the start address of the **.hex** file in EPCQ flash and it should point to a location after the *.sof file image. You must not configure the reset offset to 0x0 because the ***.sof** image is located at address 0x0 in the EPCQ flash. You must estimate the size of your ***.sof** image to identify the smallest address to which your reset vector can point. For example, a ***.sof** image size of 512 KB requires a reset vector at 0x00080000. Quartus II displays an overlapping error if the addresses are not set correctly.
2. Under **Exception Vector**, you may select EPCQ (**epcq_controller_0.avl_mem**) or OCRAM in the **Exception vector memory** drop-down menu. In this example, 0x20 is listed for the **Exception vector offset** entry.

   Note: The exception vector is not at offset 0x20 in all cases. The only exception is when the reset vector and the exception vector point to the same memory and no boot copier is present. In this

case, the exception vector offset must be 0x20 so the application can jump to the correct starting address.

Note: Setting the exception vector to OCRAM is recommended to make the interrupt processing faster.

Figure 4: Exception Vector Settings in Qsys



3. Select the **Finish** button. You will return to the Qsys **System Contents** tab.
4. Double click on the **Altera Serial Flash Controller IP** to open the **Altera Serial Flash Controller Parameter** editor.
5. Select the **Configuration device** type based on your hardware design and choose the desired **I/O mode**. Close the **Parameter Editor** and return to the **Qsys System Contents** tab.
6. Select **Generate HDL** to generate your Qsys design.
7. Compile your design in Quartus II.

Reset and Exception Vector Settings for Nios II Boot Copier Method

1. Under **Reset Vector** in the **Vectors** tab, select EPCQ (**epcq_controller_0.avl_mem**) in the **Reset vector memory** drop-down menu and type the reset vector offset in the **Reset vector offset** entry box. The reset vector must be the starting address of your application. In this example, it is 0x01E00000.

   Note: Reset vector offset varies according to your ***.sof** image size. It is the start address of the **.hex** file in EPCQ flash and points to somewhere after the *.sof file image. You must not configure the reset offset to 0x0 because the ***.sof** image is located at address 0x0 in the EPCQ flash. You must estimate the size of your ***.sof** image to identify the smallest address to which your reset vector can point. For example, a ***.sof** image size of 512 KB requires a reset vector at 0x00080000. Quartus II displays an overlapping error if the addresses are not set correctly.
2. Under **Exception Vector**, select **OCRAM/External RAM** in the **Exception vector memory** drop-down menu. In this example, 0x20 is listed for the **Exception vector offset** entry.

Note: The exception vector is not at offset 0x20 in all cases. When the reset vector and the exception vector point to the same memory and no boot copier is present, exception vector offset 0x20 must be used so the application can jump to the correct starting address.

3. Select the **Finish** button. You will return to the Qsys **System Contents** tab.
4. Double click on the **Altera Serial Flash Controller IP** to open the **Altera Serial Flash Controller Parameter** editor.
5. Select the **Configuration device** type based on your hardware design and choose the desired **I/O mode**. Close the **Parameter Editor** and return to the **Qsys System Contents** tab.
6. Select **Generate HDL** to generate your Qsys design.
7. Compile your design in Quartus II.

BSP Editor Settings

1. Open the Nios II SBT tool and create the Nios II processor HAL BSP based on the **.sopinfo** created from Qsys generation.
2. Right click on the BSP that you have created and select **Nios II** > **BSP Editor**.
3. Configure the **Settings.Advanced.hal.linker** and the **Linker Section** mappings in the **BSP Editor** based on your boot options. Refer to the table below.

Table 3: BSP Editor Mappings

Boot Option	BSP Editor: Linker Script	Settings.Advanced.hal.linker
Nios II processor application executes in place from EPCQ flash	• Set **.text** Linker Section to EPCQ (**epcq_controller_0_avl_mem**) • Set other Linker Sections (**.heap**, **.rwdata**, **.rodata**, **.bss**, **.stack**) to OCRAM	If the exception vector is set to OCRAM, enable the following settings in **Settings.Advanced.hal.linker**: • allow_code_at_reset • enable_alt_load • enable_alt_load_copy_rodata • enable_alt_load_copy_rwdata • enable_alt_load_copy_exceptions If the exception vector is set to EPCQ, enable the following settings in **Settings.Advanced.hal.linker**: • allow_code_at_reset • enable_alt_load • enable_alt_load_copy_rodata • enable_alt_load_copy_rwdata
Nios II processor application is copied from EPCQ flash to RAM using boot copier	Make sure all Linker Sections are set to OCRAM/External RAM.	Make sure all settings are left unchecked.

Figure 5: Settings.Advanced.hal.linker for Execute-In-Place Method if Exception Vector is Set to OCRAM

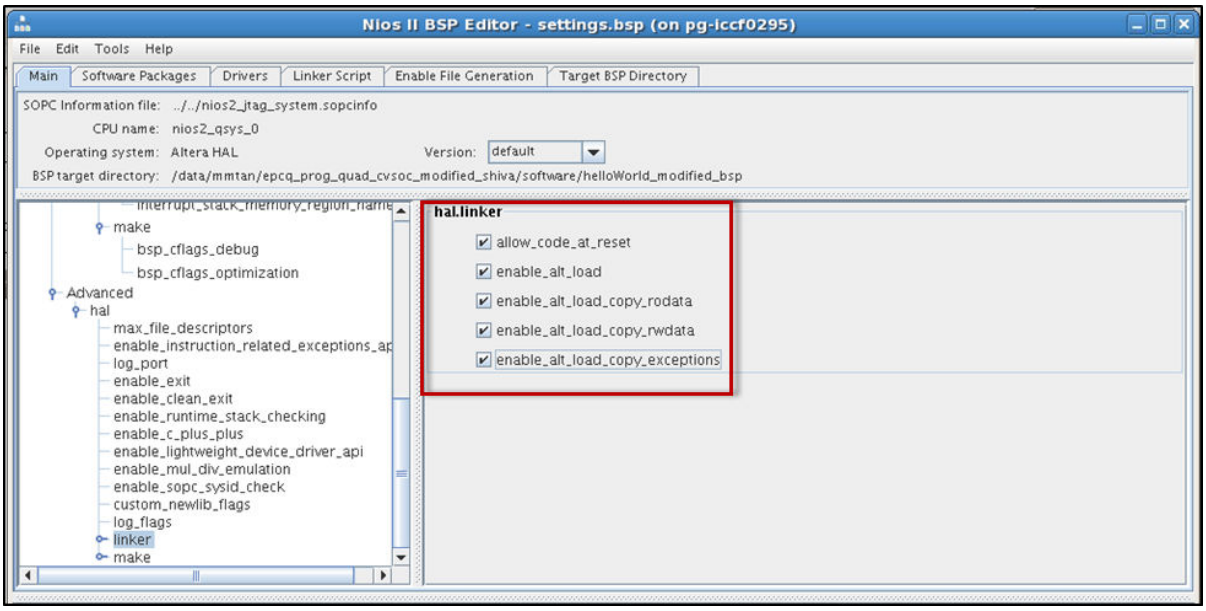


Figure 6: Settings.Advanced.hal Linker for Execute-in-Place Method if Exception Vector is Set to EPCQ

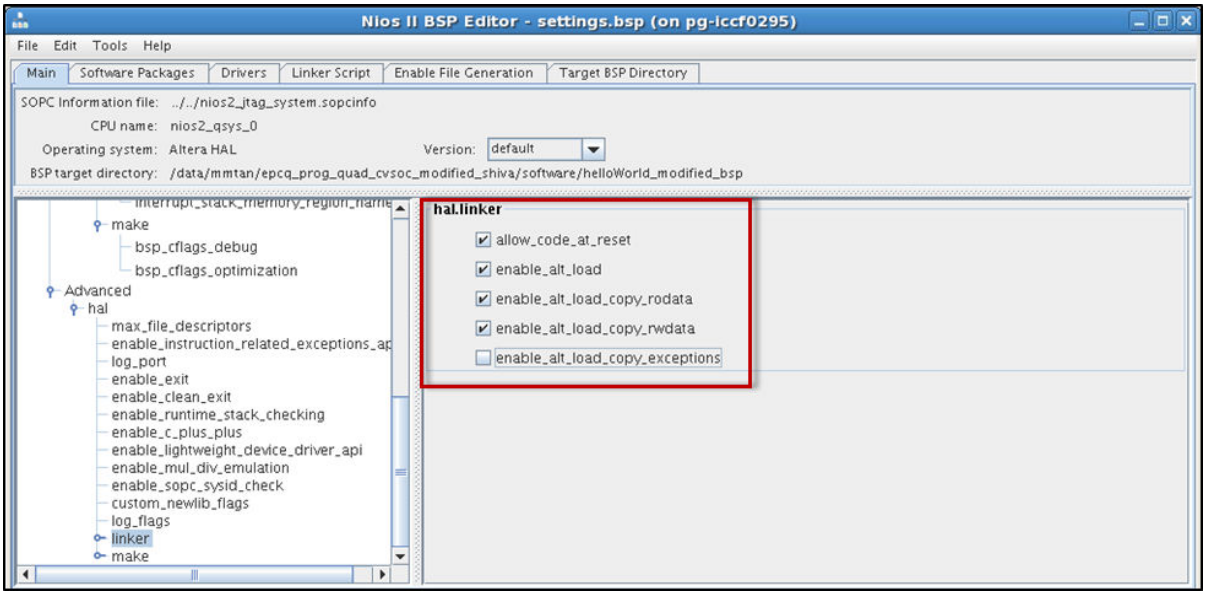


Figure 7: Settings.Advanced.hal Linker for Boot Copier Method

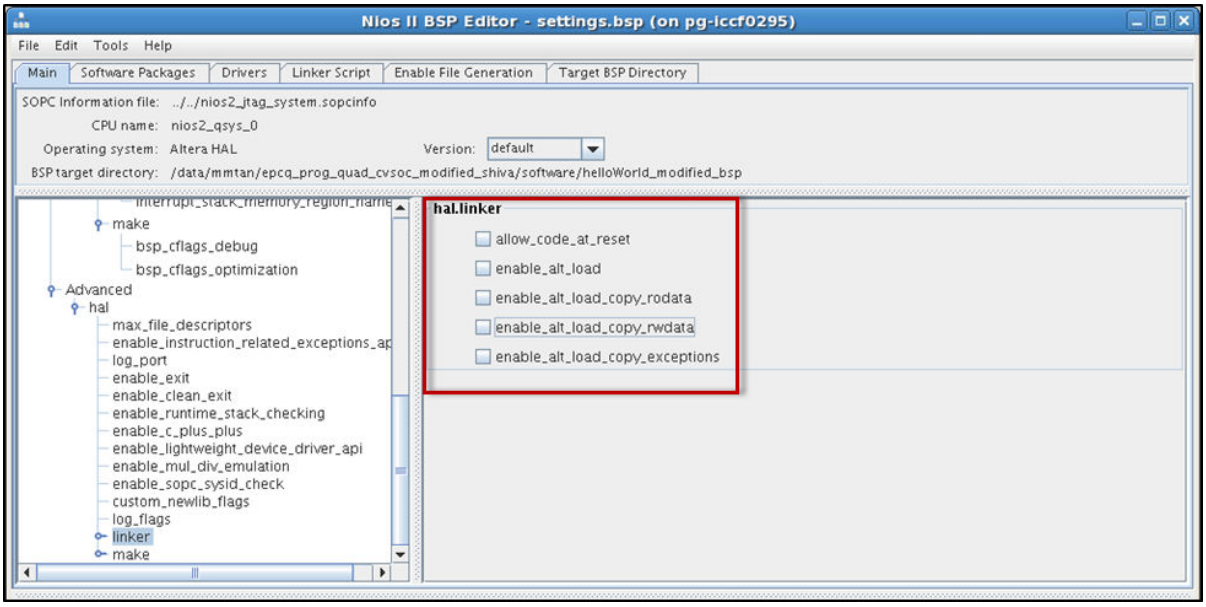


Figure 8: Linker Section Mappings for Execute-In-Place Method

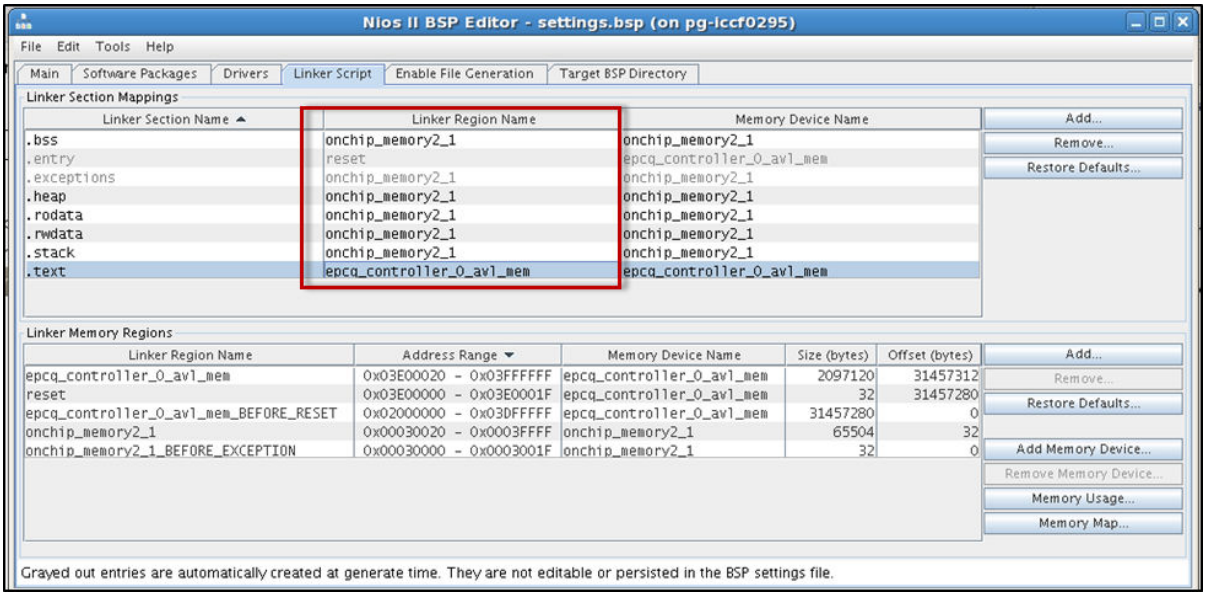
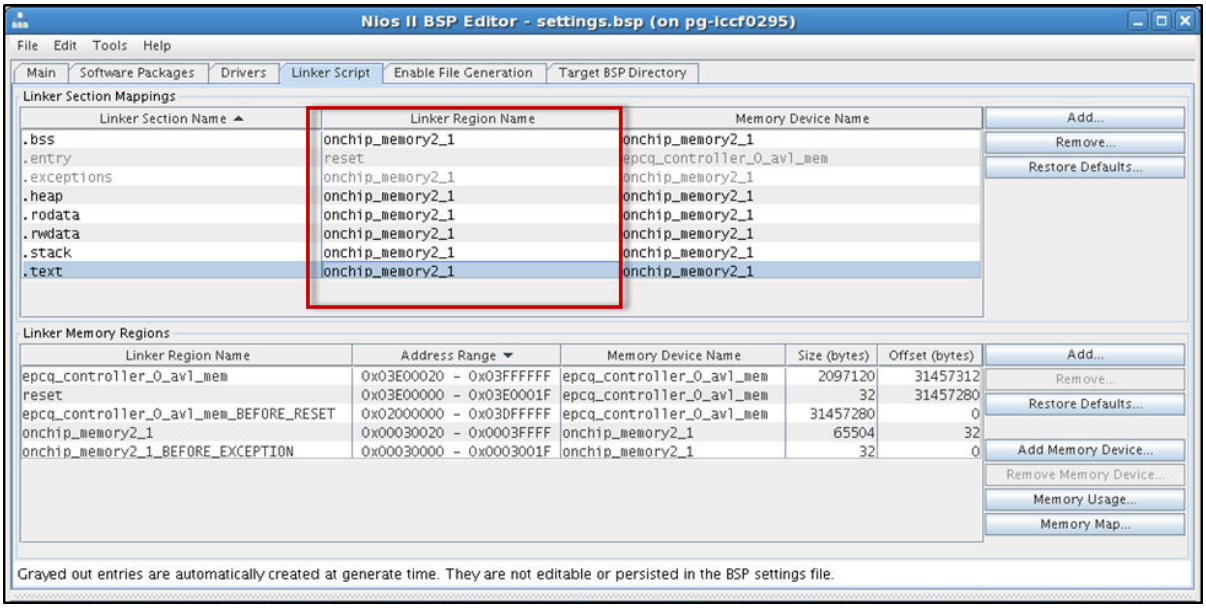


Figure 9: Linker Section Mappings for Boot Copier Method

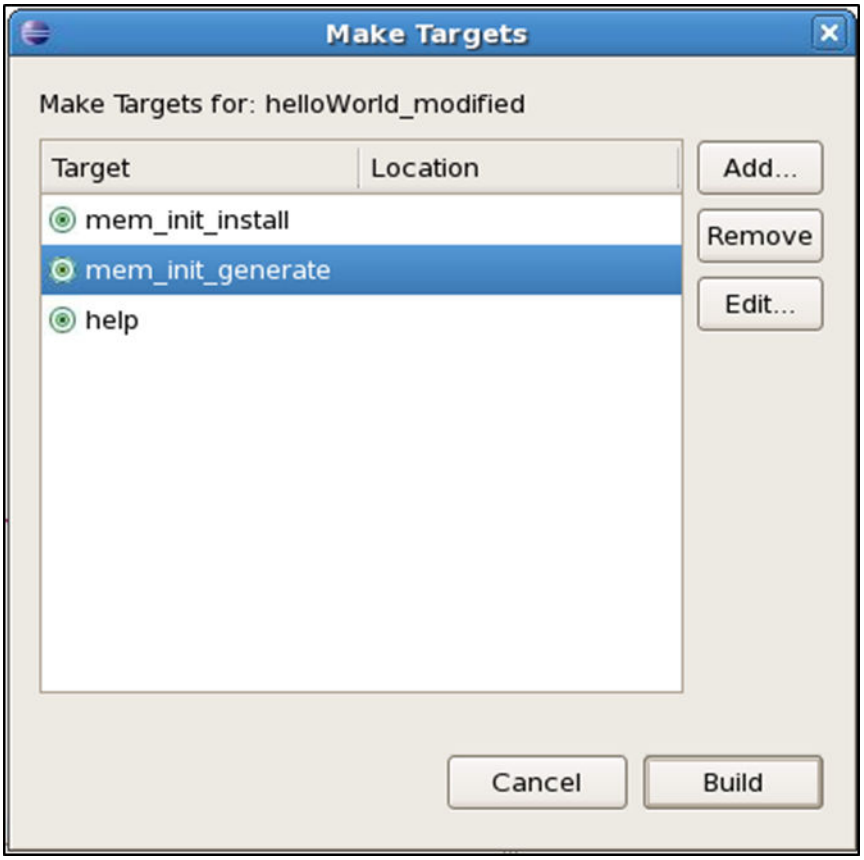


4. Select the **Generate** button to generate the BSP. Make sure the BSP generation is successful.
5. Click on the **Exit** button to close the BSP editor.

Application

1. In the Nios II SBT window, select **File** > **Nios II Application** to create a new application. Develop your Nios II application. Alternatively, you can use a template for the application creation.
2. Point your application to the BSP location that you have created in the previous section.
3. Once the application development is done, right click on the project in **Project Explorer** and select **Build Project**. A ***.elf** file is create under the project folder.
4. Next, generate the HEX file using the **mem_init_generate** utility. Right click on the project in **Project Explorer** and select **Make Targets** > **Build**. You will see the **Make Targets** window as shown below.

Figure 10: Make Targets Window



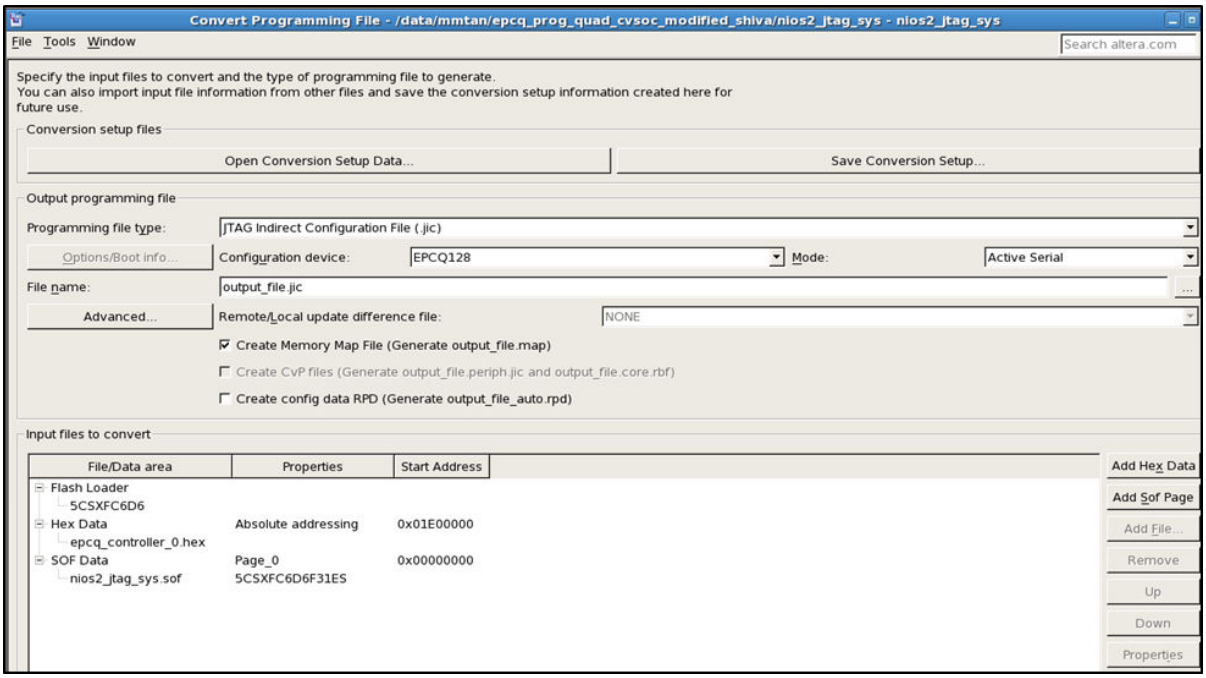
5. Select **mem_init_generate** and select the **Build** button. Make sure **epcq_controller_0.hex** is created under the **mem_init** folder.

Programming Files Conversion

1. In Quartus II, go to **File** > **Convert Programming Files** to open the **Convert Programming File** tool.
2. Under **Output programming file** section, set the following items:

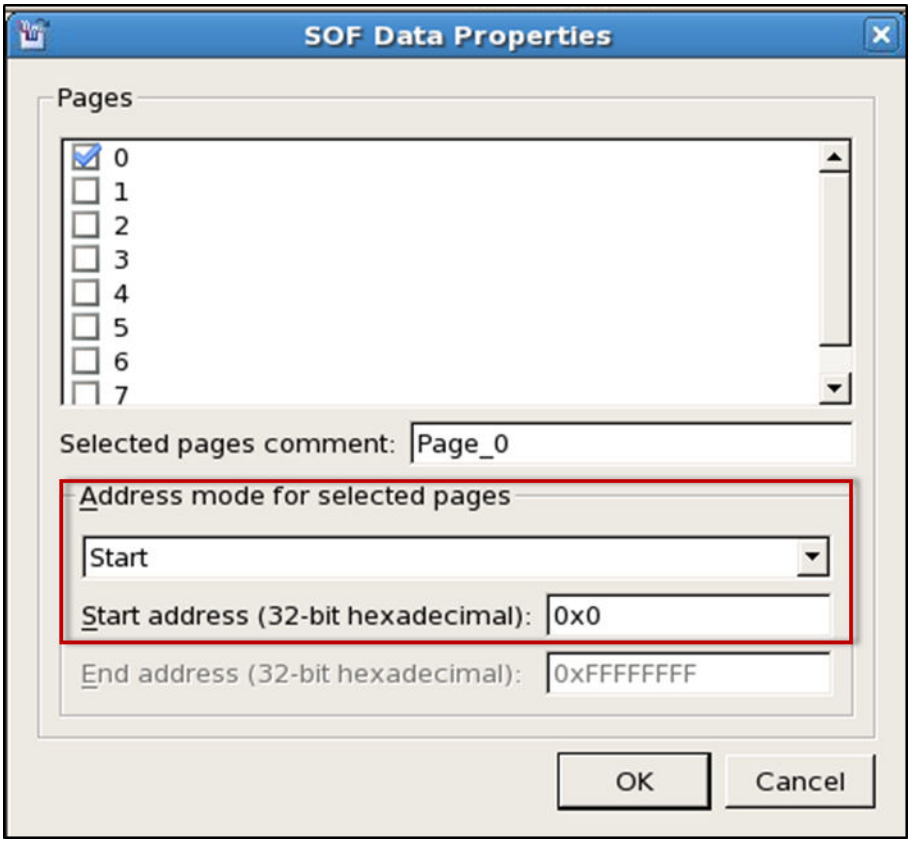
a. **Programming file type:** JTAG Indirect Configuration File (**.jic**)
b. **Configuration device:** Select according to your EPCQ type
c. **Mode:** Active Serial
d. **File name:** You may select your preference path for the output file (**.jic**). By default this is generated under your project root directory.
e. Keep the default settings for **Create Memory Map File** and **Create config data RPD**.

Figure 11: Convert Programming File Settings



3. Under **Input files to convert** section:
   a. Select **Flash Loader**, then **Add Device** to select the FPGA device that you are using. Select **OK** when you are done.
   b. Select **SOF Data**, then **Add File** to choose the ***.sof** file generated by Quartus II compilation.
   c. Click on the ***.sof** file that you have just added. Next, select **Properties** and enable **Compression**.
   d. Click on **SOF Data** and select **Properties**. The **SOF Data Properties** window opens. Under **Address mode for selected pages**, select **Start** and set the start address (32-bit hexadecimal). If using a Cyclone® V SoC development board, set the start address to 0x0 to avoid a "size exceeds memory capacity" error.

Figure 12: SOF Data Properties

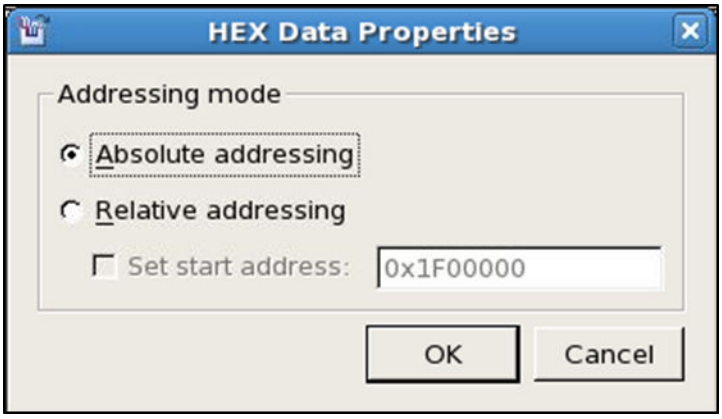


e. Select **OK** to close the window.

f. Select **Add Hex Data** and the **Hex Data Properties window** appears. Select the **Absolute addressing** button and browse to the HEX file location. Click **OK** when you are done.

Note: You may select **Relative addressing** if you would like to set a relative address to the reset vector offset (0x01E00000) you configured earlier. For example, setting a start address of 0x01F00000 for the relative addressing mode changes the start address to 0x3D00000.

Figure 13: HEX Data Properties



g. Select **Generate** to generate the ***.jic** programming file.

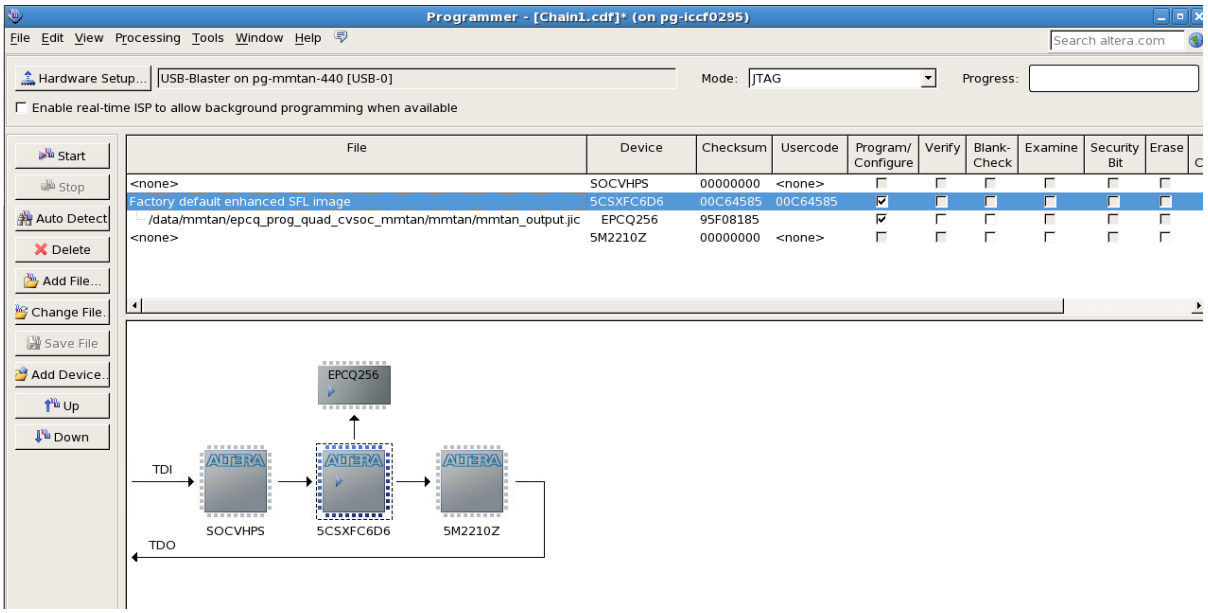
Configuration for EPCQ Programming

1. Ensure that the FPGA device's Active Serial (AS) pin is routed to the EPCQ flash. This routing allows the flash loader to load into the EPCQ flash and configure the board correctly.
2. If using a Cyclone V SoC development board, ensure the `MSEL` pin setting on the board is configured for AS programming.

Download

1. Open the Quartus II programmer and make sure JTAG was detected under the **Hardware Setup**.
2. Select **Auto Detect** and choose the FPGA device according to your board.
3. Right click on the selected FPGA device and select **Edit** > **Change File**. Next, select the generated ***.jic** file.
4. Select the **Program/ Configure** check boxes for FPGA and EPCQ devices. Click on **Start** to start programming.

Figure 14: Quartus II Programmer Window When Programming EPCQ Flash on the Cyclone V SoC Development Board



5. Make sure the download is successful.

Reset and Boot

1. Reset your FPGA device using the reset button or power cycle your hardware.

EPCQ HAL Driver

The Nios II hardware abstraction layer (HAL) driver has a generic application programming interface (API) for flash devices.

This API provides users read and write access to the EPCQ flash through the Altera Serial Flash controller for ACDS 15.0 and onwards. Use driver version 15.0 (or higher) for EPCQ software application development.

Nios II Processor Booting Elements

Memcpy-based Nios II Boot Copier

The Nios II processor memcpy-based boot copier has the following features:

- Supports EPCQ, CFI and Altera on-chip flash (UFM) flash memories
- Can locate software application in flash
- Unpacks and copies software application image to Random Access Memory (RAM)
- Automatically switches to application code in RAM after copy completes

The Nios II processor boot copier is used when the Nios II soft processor application is copied from EPCQ flash to RAM. The memcpy-based boot copier is automatically appended into the **.hex** file by the Nios II SBT tool when the executable file (**.elf**) is converted to a memory initialization file (**.hex**) using the `make mem_init_generate` target. This operation takes place whenever the **.text** linker section is located in a different memory than where the reset vector points, which indicates a code copy is required. The function of the boot copier is to copy the software application to a user desired location such as RAM. Once the copy is completed, the boot copier passes the system control to the application.

Related Information

AN 730: Nios II Processor Booting Methods in MAX 10 Devices
Refer to this information for more information about Nios II processor booting elements.

Revision History for Nios II Booting From Altera Serial Flash (EPCQ)

Date	Version	Changes
May 2015	2015.05.04	Initial Release