NIOS II Processor Booting Methods In MAX 10 Devices

2015.01.23

AN-730 Subscribe Send Feedback

MAX® 10 device is the first MAX device series which supports Nios® II processor.

Overview

MAX 10 devices contain on-chip flash which segmented to two types:

- Configuration Flash Memory (CFM) to store hardware configuration settings for MAX 10 FPGA
- User Flash Memory (UFM) to store user software applications

You can boot and configure the Nios II soft core processor to execute code from the on-chip flash within the FPGA using the Altera On-chip Flash IP core.

This document describe the overview of the Altera On-chip Flash IP core and various booting methodology using the MAX 10 on-chip flash.

Abbreviations

Table 1: Abbreviations Use In the Document

Abbreviation	Description
RAM	Random Access Memory
OCRAM	On-Chip RAM
UFM	User Flash Memory
CFM	Configuration Flash Memory
SBT	Software Build Tools
HEX	Hexadecimal File This is an ASCII text file with the extension of .hex which stores the initial memory values for a memory block.
XIP	Execute In Place
POF	Programmer Object File
ERAM	Embedded Random Access Memory

© 2015 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Prerequisite

You are required to have the knowledge of instantiating and developing a system with Nios II processor based system. Altera recommends you to go through the online tutorials and training materials provided at **http://www.altera.com/education/edu-index.html** before using this application note.

Related Information

- **Nios II Gen2 Hardware Development Tutorial. A step by step procedure to build a Nios II Gen2 soft core processor system.**
- **Getting Started with the Graphical User Interface. This document provides the details of Nios II Software Build Tools using graphical user interface.**

MAX 10 On-chip Flash Overview

The MAX 10 On-chip Flash consists of two flash sectors with the functionality shown in the following table.

Table 2: On-chip Flash Sectors in MAX 10 Devices

Flash Sector	Functionality
Configuration Flash Memory (sectors CFM0-2)	FPGA configuration file storage
User Flash Memory (sectors UFM0-1)	User application or data storage

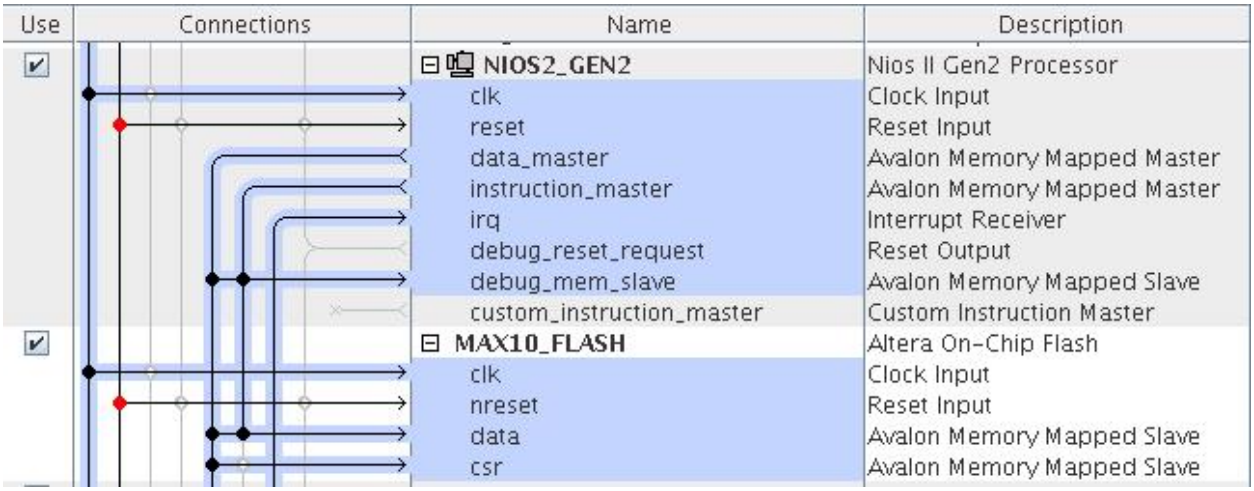
You can configure the CFM sector to use as UFM sector. The following table shows the storage location of the configuration images based on internal configuration mode options available in MAX 10 devices. You can refer to MAX 10 FPGA Configuration User Guide for more information related to the internal configuration modes supported in MAX 10 devices.

Table 3: Storage Location Of The Configuration Images Per Internal Configuration Mode Options

Internal Configuration Modes	CFM2[1]	CFM1[1]	CFM0
Dual images	Compressed Image 2		Compressed Image 1
Single uncompressed image	UFM	Uncompressed image	
Single uncompressed image with Memory Initialization	Uncompressed image (with ERAM preload)		
Single compressed image with Memory Initialization	Compressed image (with ERAM preload)		
Single compressed image	UFM		Compressed Image

You must use the Altera On-chip Flash IP core to access to the flash memory in MAX 10 devices. You can instantiate and connect the Altera On-chip Flash IP to the Nios II processor using the Qsys system in Quartus II software. The Nios II soft core processor uses the Avalon ® Memory-Mapped (Avalon-MM) interface to communicate with the Altera On-chip Flash IP.

Figure 1: Example of Connections for The Altera On-chip Flash IP and the Nios II Gen2 Soft Core Processor



Related Information

- MAX 10 FPGA Configuration User Guide

[1] This sector is NOT supported in 10M02 device.

- **MAX 10 User Flash Memory User Guide**

Altera On-chip Flash IP Architecture and Features

The Altera On-chip Flash IP core provides access to five flash sectors:

- UFM0
- UFM1
- CFM0
- CFM1
- CFM2

Important facts about UFM and CFM sectors:

- CFM sectors are intended for configuration (bitstream) data (*.pof) storage.
- You can store user data in the UFM sectors.
- Certain devices do not have a UFM1 sector. You can refer to UFM and CFM Array Size table for available sectors per MAX 10 devices.
- You can configure CFM2 as a virtual UFM using "Single Uncompressed Image" internal configuration mode.
- You can configure CFM2 and CFM1 as a virtual UFM using "Single Compressed Image" internal configuration mode.
- The sizes of each sector varies with the selected MAX 10 devices.

Table 4: UFM and CFM Array Size

This table lists the dimensions of the UFM and CFM arrays.

Device	Pages per Sector					Page Size (Kb)	Total User Flash Memory Size (Kb)	Total Configuration Memory Size (Kb)
	UFM1	UFM0	CFM2	CFM1	CFM0			
10M02	3	3	0	0	34	16	96	544
10M04	0	8	41	29	70	16	1248	2240
10M08	8	8	41	29	70	16	1376	2240
10M16	4	4	38	28	66	32	2368	4224
10M25	4	4	52	40	92	32	3200	5888
10M40	4	4	48	36	84	64	5888	10752
10M50	4	4	48	36	84	64	5888	10752

The Altera On-chip Flash supports the following features:

- Read or write accesses to UFM sectors using the Avalon MM data and control slave interface.
- Configure CFM sectors to UFM sectors using single uncompressed image or single compressed image.
- UFM initialization using HEX file.
- Simulation model for UFM read / write accesses various EDA simulation tool.

ERAM Preload Option

The FPGA configuration data may contain MAX 10 On-chip RAM or ERAM initialization data. The ERAM preload occurs during FPGA configuration before the device enters to user mode. The ERAM preload option allows initialization data for the On-chip RAM to be stored in the CFM sectors, this could be any type of application data, including Nios II soft core processor software.

All MAX 10 devices except for the MAX 10 10M02 and 10M04 devices can support dual FPGA configuration images. This however requires the ERAM preload to be set to OFF, in order to reduce FPGA configuration image size.

When the ERAM preload feature is set to OFF, features that require initialization of on chip RAM will not work. The ERAM preload option is set to OFF by default.

Nios II Processor Booting Options Using On-chip Flash

The Nios II processor supports following two boot options using on-chip flash:

- Option 1: Nios II processor application executes in-place from Altera On-chip Flash.
- Option 2: Nios II processor application copied from UFM to RAM using boot copier.

Table 5: Summary of Nios II Processor Boot Options

Boot Option	Application Code Stored Location	Reset Vector	Method
Option 1: Nios II processor application executes in-place from Altera On-chip Flash	UFM	UFM (XIP) + OCRAM (for data)	Using the alt_load () function
Option 2: Copy Nios II processor application from UFM to RAM using boot copier	OCRAM/ External RAM	UFM	Using default boot copier

Option 1: Nios II Processor Application Executes In-place From Altera On-chip Flash

This solution 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 OCRAM. The code section (.text), which is a read only section, remains in the Altera On-chip Flash memory region. This helps to minimize the RAM usage but may limit the code execution performance.

The Nios II processor application is programmed into the UFM sector. The Nios II processor reset vector points to the UFM sector in order to executes code after the system resets.

If you are debugging the application using the source-level debugger, you must use a hardware breakpoint because the UFM cannot efficiently support random memory access.

Option 2: Nios II Processor Application Copied From UFM To RAM Using Boot Copier

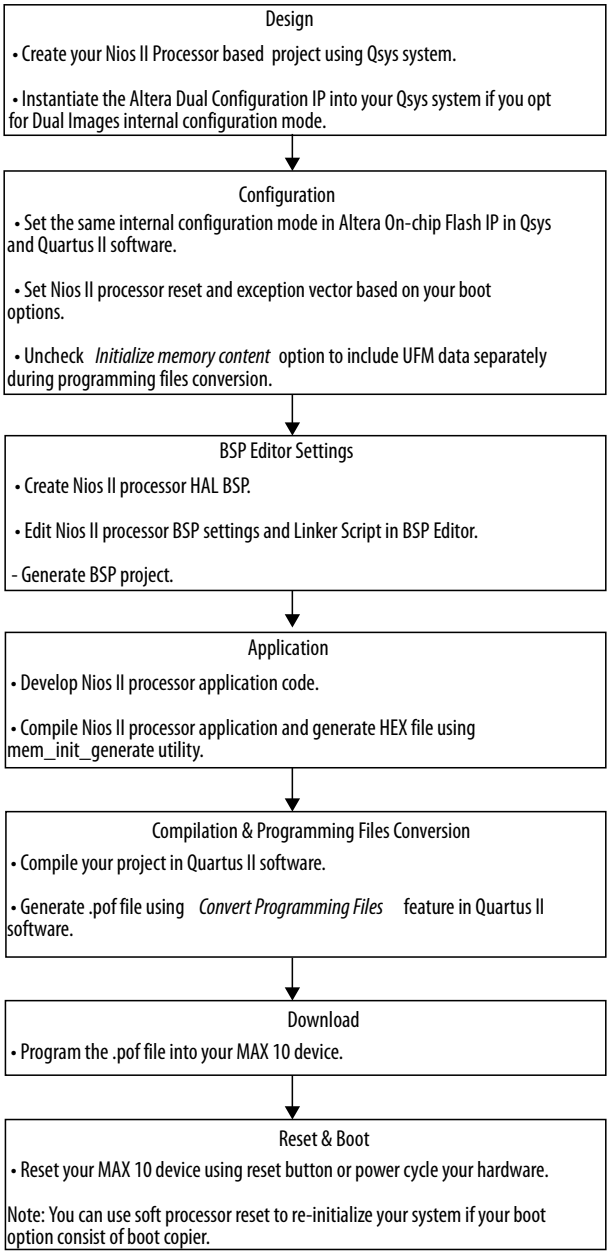
Altera recommends this solution for MAX10 Nios II processor system designs where there may be multiple iterations of application software development and when high system performance is required. The boot copier is located at the base address of the UFM, followed by the application.

For this boot option, Nios II processor starts executing the boot copier software upon system reset to copy the application from the UFM sector to the internal/external RAM. Once this is complete, the Nios II processor transfers the program control over to the application.

Nios II Soft Core Processor Configuration And Booting Flow

This section will help to guide you through the design, configuration and BSP settings for the supported boot scenarios mentioned in this document.

Figure 2: Configuration and Booting Flow for Boot Option 1 and 2



Steps To Build A Bootable System With Boot Option 1 and 2

This is a step by step guide to build a bootable system using:

- Option 1: Nios II processor application execute in-place from Altera On-chip Flash
- Option 2: Nios II processor application copied from UFM to RAM using boot copier

The following table shows the required RAM size for each boot option:

Table 6: RAM Size Requirement For Each Boot Option

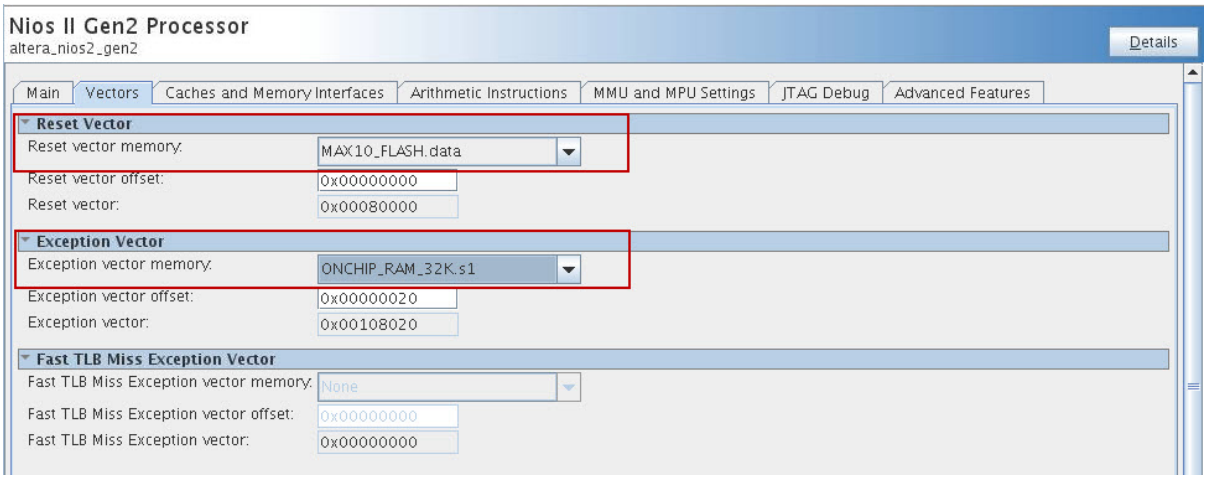
Boot Option	RAM Size Requirement
Option 1: Nios II processor application execute in-place from Altera On-chip Flash	Equivalent to the dynamic memory space usage during run time which is the sum of the maximum heap and stack size.
Option 2: Nios II processor application copied from UFM to RAM using boot copier	Equivalent to the executable code and dynamic memory size required by user program.

Using Single Uncompressed Image Internal Configuration Mode

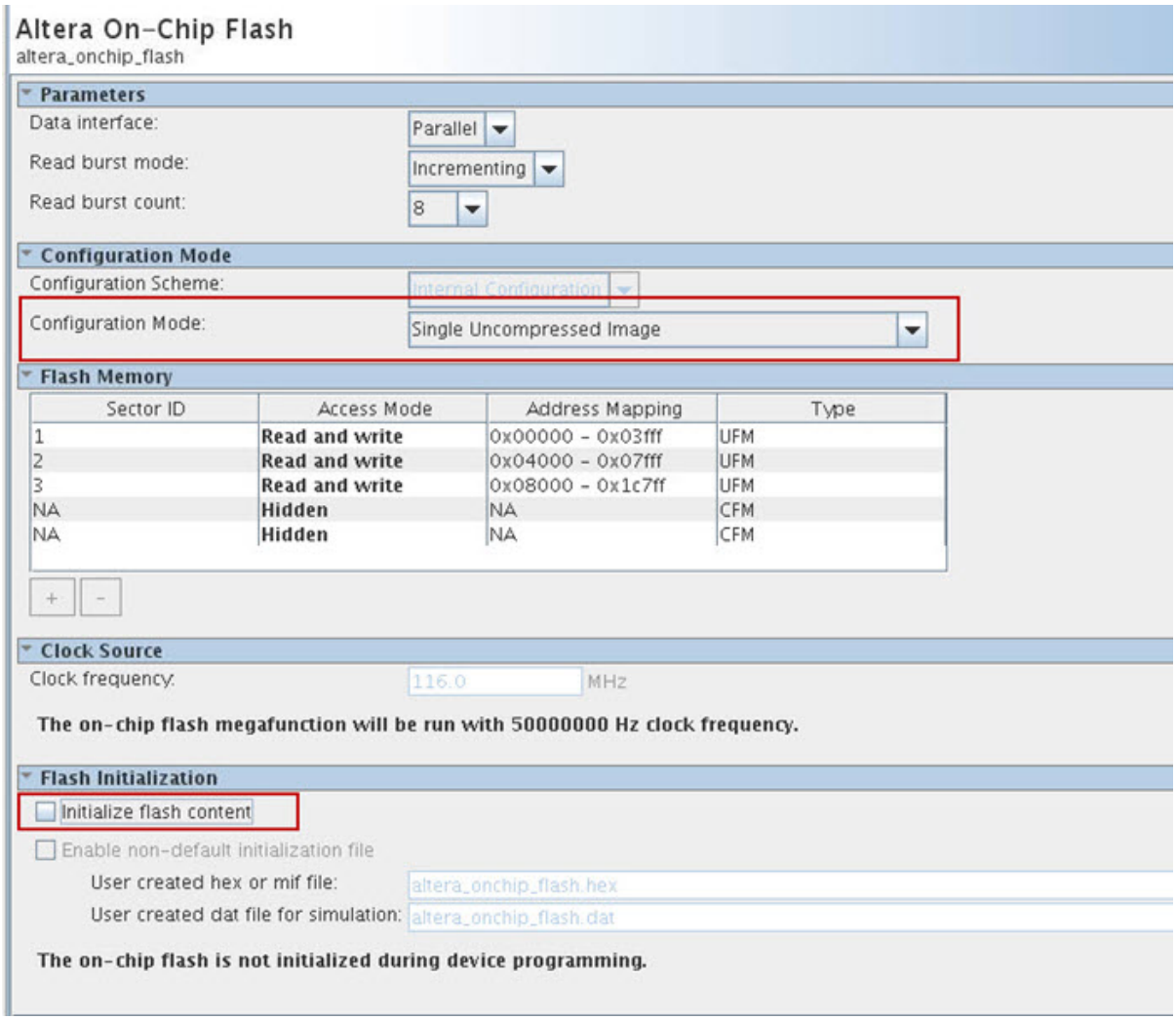
The following steps are applicable for internal configuration mode of single uncompressed image.

Qsys Settings

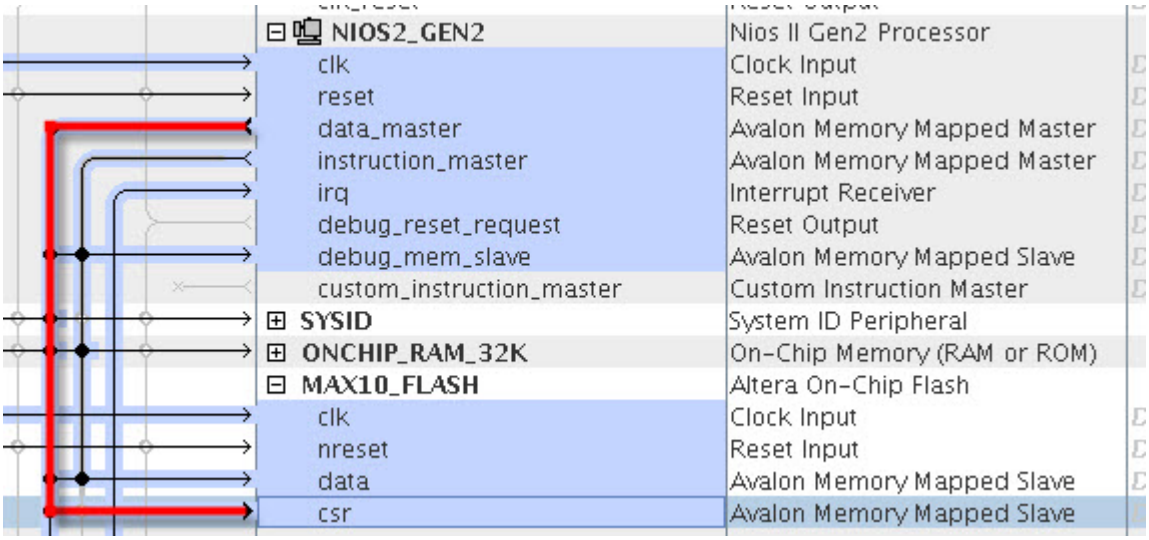
1. In Nios II Gen2 Processor parameter editor, set the **Reset vector memory:** to Altera On-chip Flash and **Exception vector memory:** to OCRAM. The following is a snapshot of an example where Altera On-chip Flash and OCRAM are named as MAX10_FLASH and ONCHIP_RAM32K respectively.



2. In Altera On-chip Flash IP parameter editor, set the **Configuration Mode:** to **Single Uncompressed Image**. Make sure the **Initialize flash content** option is left unchecked. Initializing the on-chip flash during device programming feature is currently not supported in boot option 1 and 2.



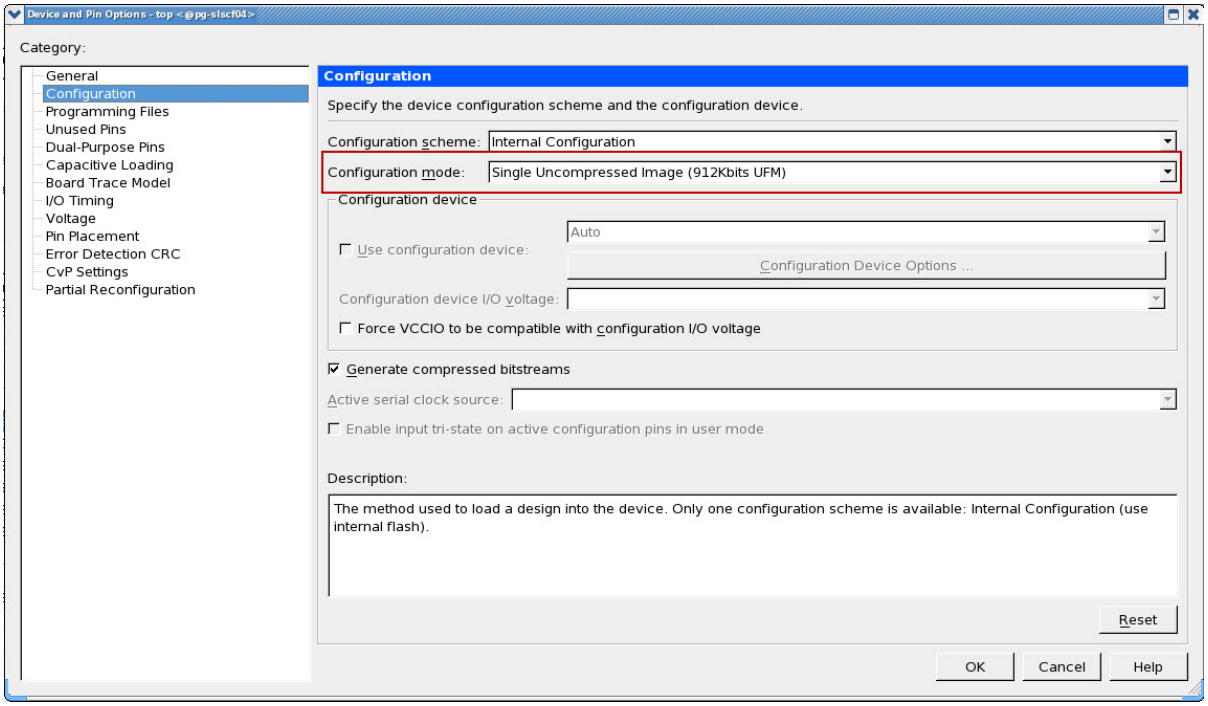
3. Make sure the Altera On-chip Flash CSR port is connected to the Nios II Gen2 processor data master to enable write and erase operations.



4. Click **Generate HDL**, the **Generation** dialog box appears.
5. Specify output file generation options, and then click **Generate**.

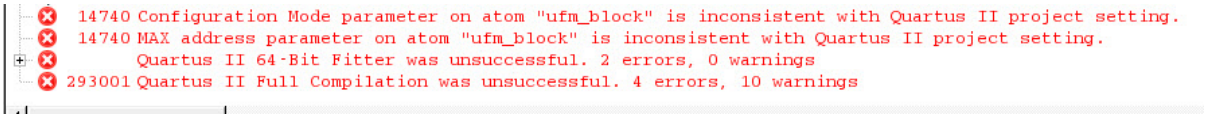
Quartus II Software Settings

1. In Quartus II software, click on **Assignment** -> **Device** -> **Device and Pin Options** -> **Configuration**. Set **Configuration mode:** to **Single Uncompressed Image (912Kbits UFM)**[2].



[2] 912Kbits UFM shown in the following snapshot is the size of the UFM sector specific to 10M08SAE144C8GES device. The size of UFM sector will vary according to your device selection.

If the configuration mode setting in Quartus II software and Qsys parameter editor is different, the Quartus II project compilation will fail with the following error message.

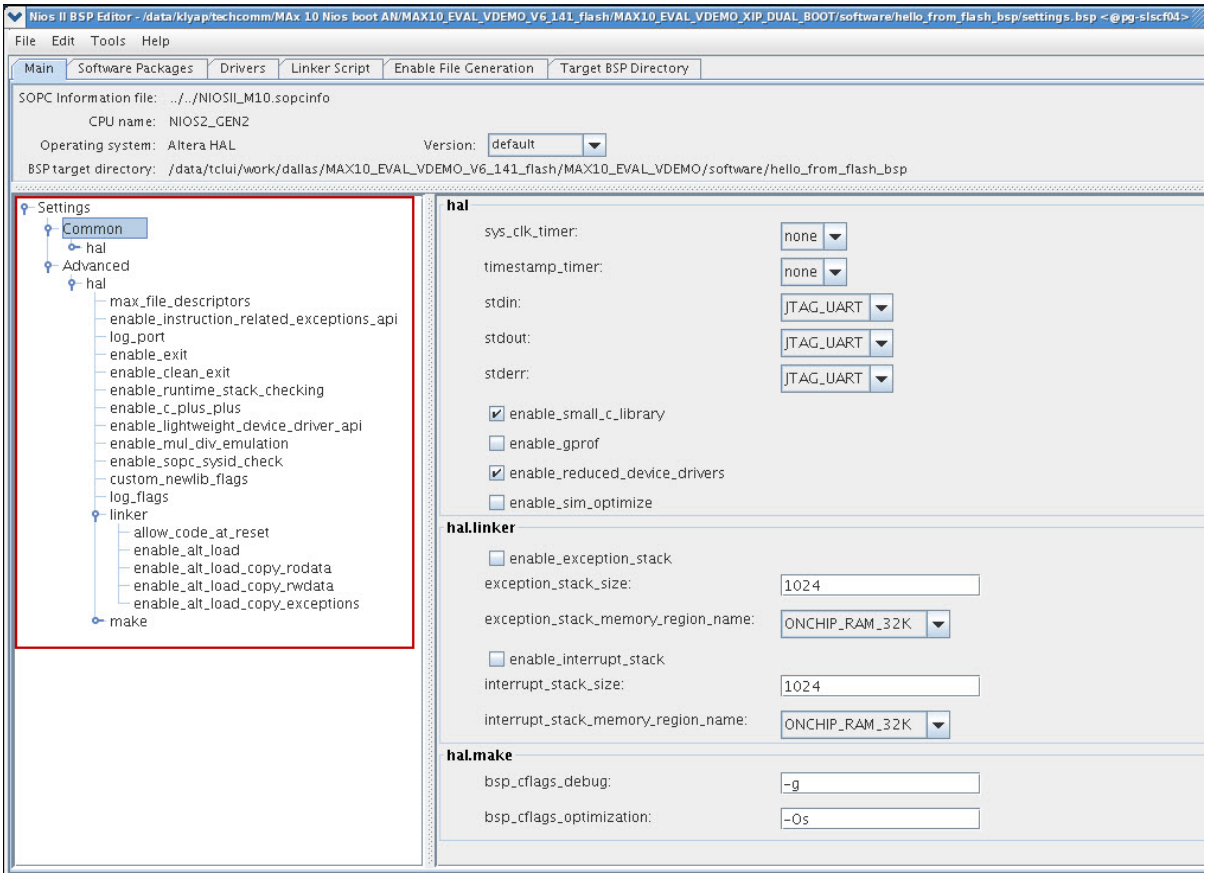


2. Click **OK** to exit the **Device and Pin Options window**.
3. Click **OK** to exit the **Device** window.
4. Click **Start Compilation** to compile your project and generate the .sof file.

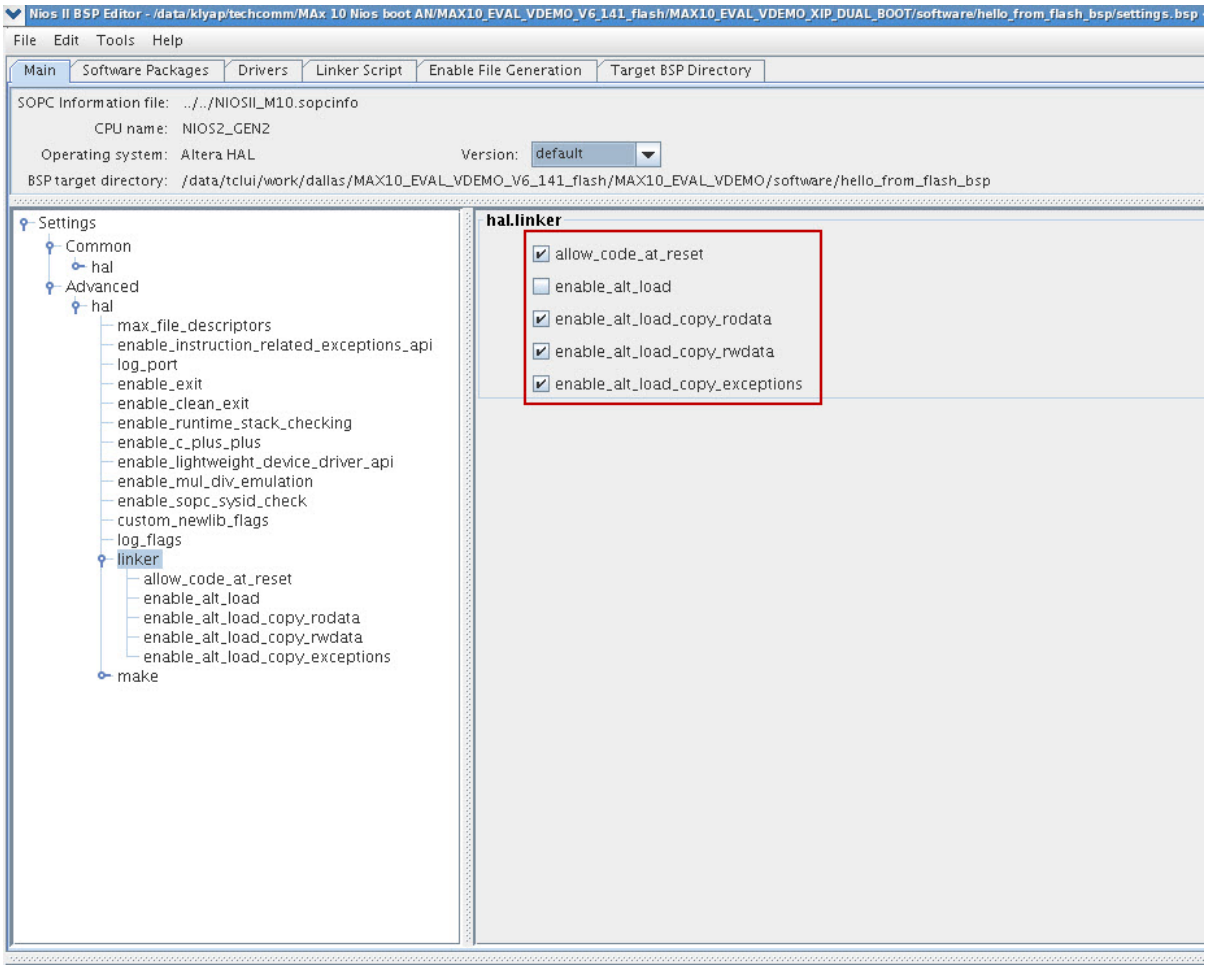
BSP Editor Settings

You must edit the BSP editor settings according to the selected Nios II processor boot options.

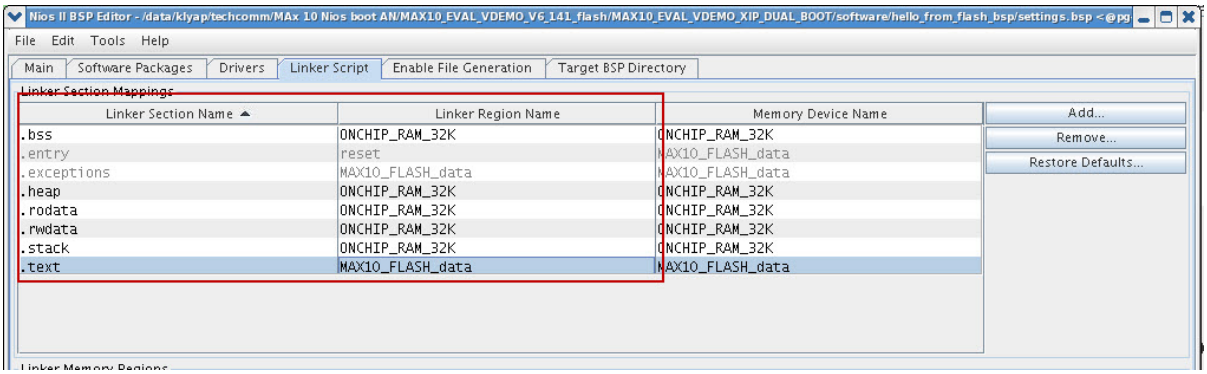
1. In Nios II BSP Editor, click on **Advanced** tab under **Settings**.
2. Click on **hal** to expand the list.
3. Click on **linker** to expand the list.



4. For boot option 1, you must enable **allow_code_at_reset**, **enable_alt_load_copy_rodata**, **enable_alt_load_copy_rwdata**, and **enable_alt_load_copy_exceptions**. For boot option 2, you must leave the **hal.linker** settings as default.

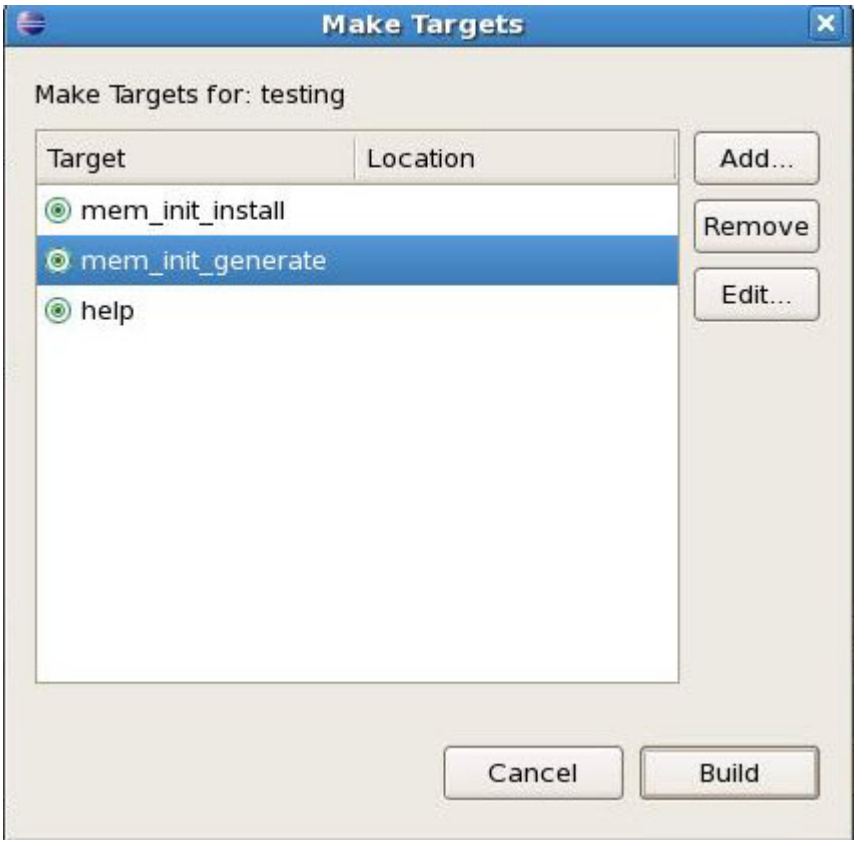


5. Click on **Linker Script** tab in the Nios II BSP Editor.
6. For boot option 1, set the **.text** item in the **Linker Section Name** to the Altera On-chip Flash in the **Linker Region Name**. Set the rest of the items in the **Linker Section Name** list to the Altera On-chip RAM. For boot option 2, set all of the items in the **Linker Section Name** list to Altera On-chip RAM.



HEX File Generation

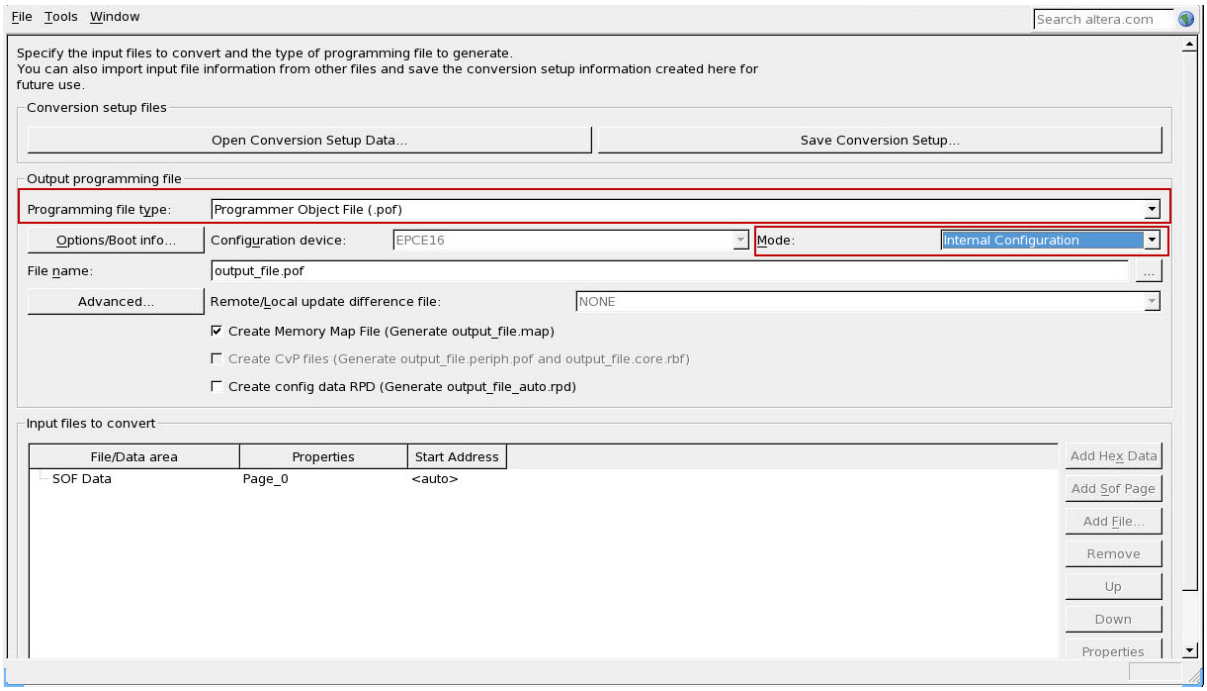
1. In the Nios II SBT tool, right click on your project in the Project Explorer window.
2. Click **Make Targets** -> **Build…**, the Make Targets dialog box appears. You can also press shift + F9 to trigger the Make Target dialog box.
3. Select **mem_init_generate**.
4. Click **Build** to generate the HEX file.



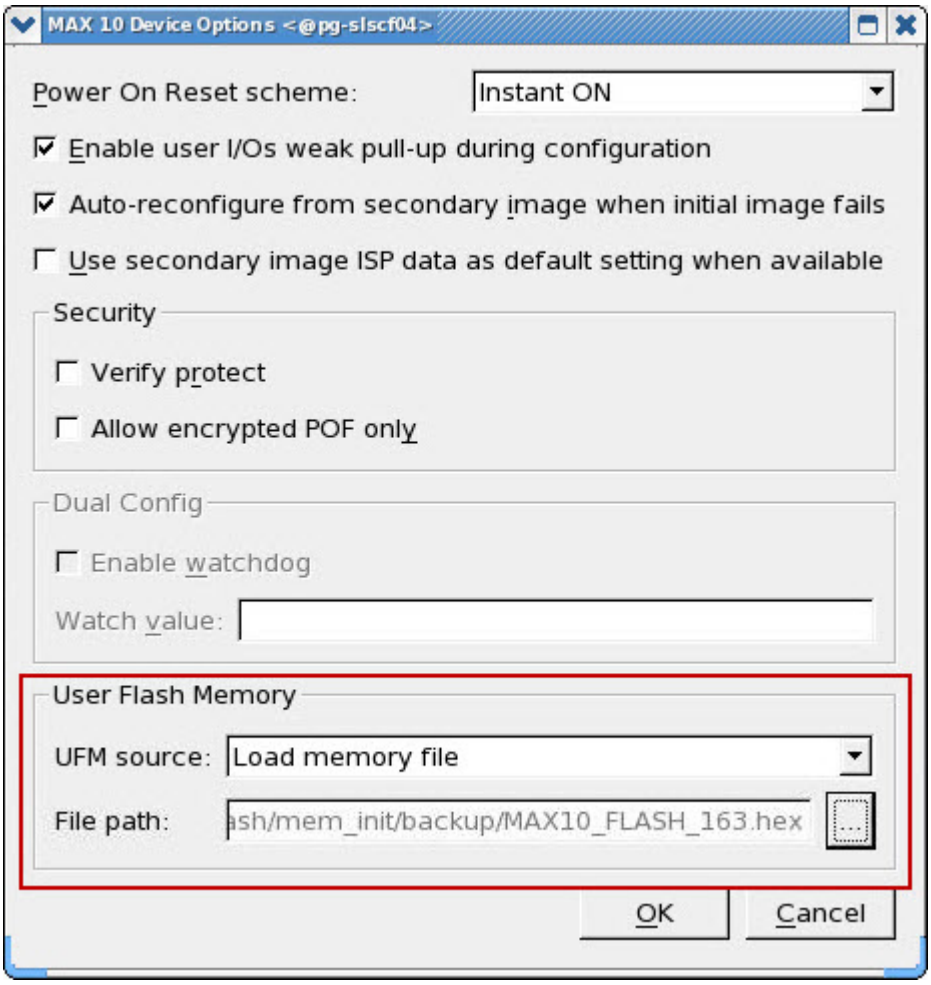
5. For these boot options, the "mem_init_generate" macro will create two HEX files; **on_chip_ram.hex** and **on_chip_flash.hex**.

Programmer Object File (.pof) Generation

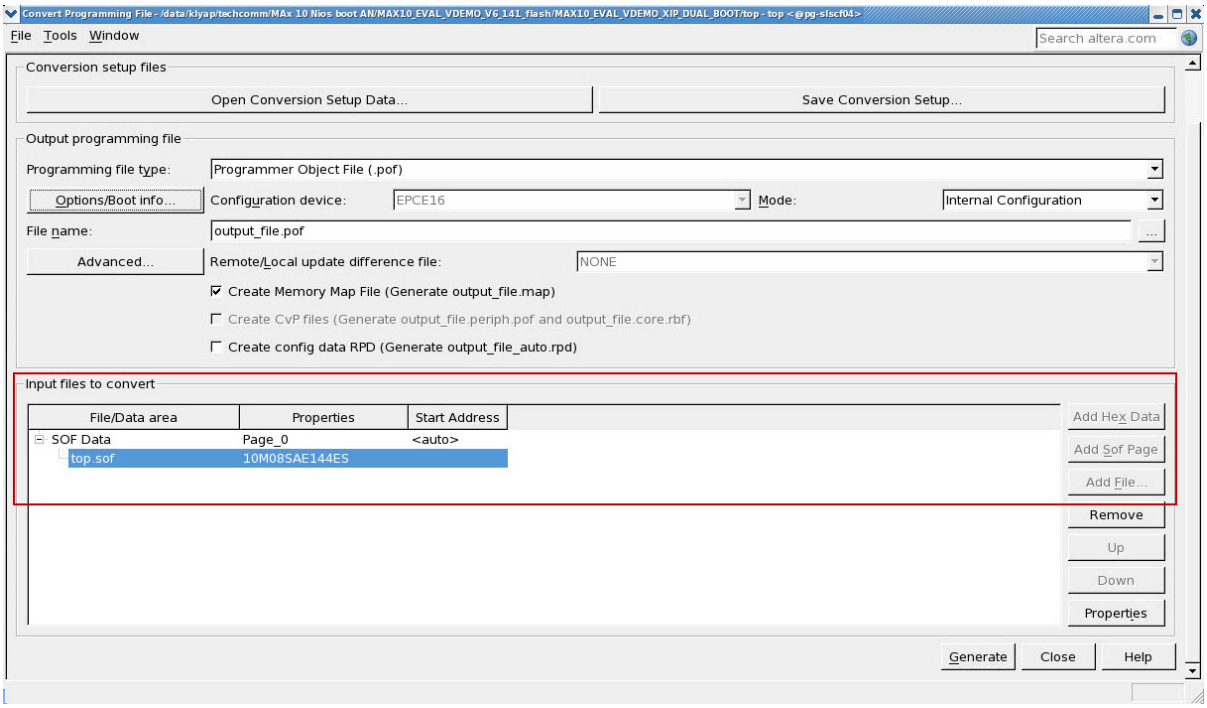
1. In Quartus II, click on **Convert Programming Files** from the **File** tab.
2. Choose **Programmer Object File** as **Programming file type:**.
3. Set **Mode** to **Internal Configuration**.



4. Click on **Options/Boot info...**, the MAX 10 Device Options dialog box appears.
5. Choose **Load memory file** for **UFM source:** option.
6. Browse to the generated Altera On-chip Flash HEX file (on_chip_flash.hex) in the **File path:**. Click **OK**.



7. In the Convert Programming File dialog box, at the **Input files to convert** section, click **Add File...** and point to the generated Quartus II .sof file.



8. Click **Generate** to create the .pof file.
9. Program the .pof file into your MAX 10 device.

Using Dual Compressed Images Internal Configuration Mode

The following steps are applicable for internal configuration mode of dual compressed images.

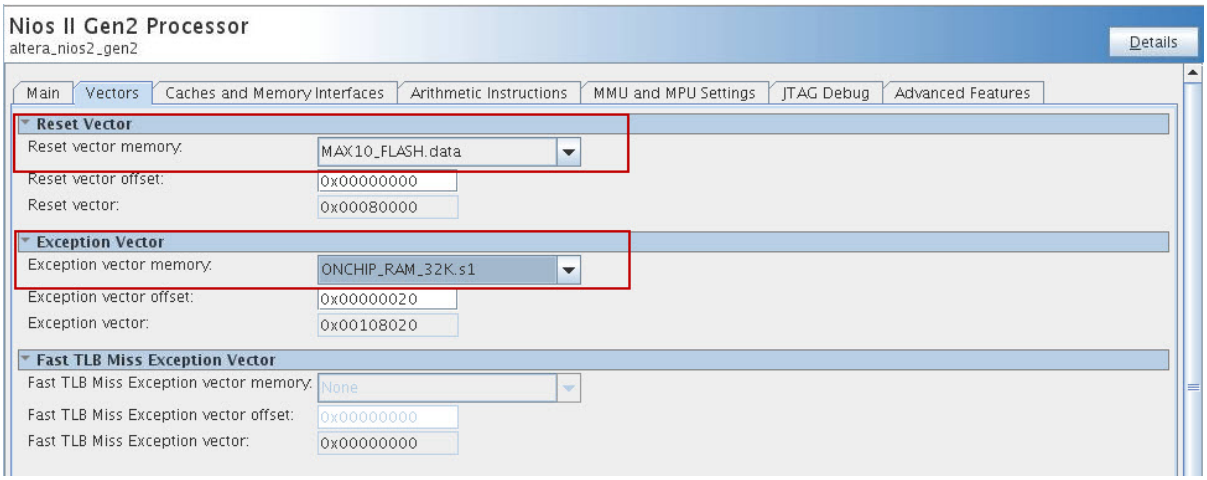
The On-chip RAM pre-initialization (ERAM preload) feature is not supported in this mode.

Related Information

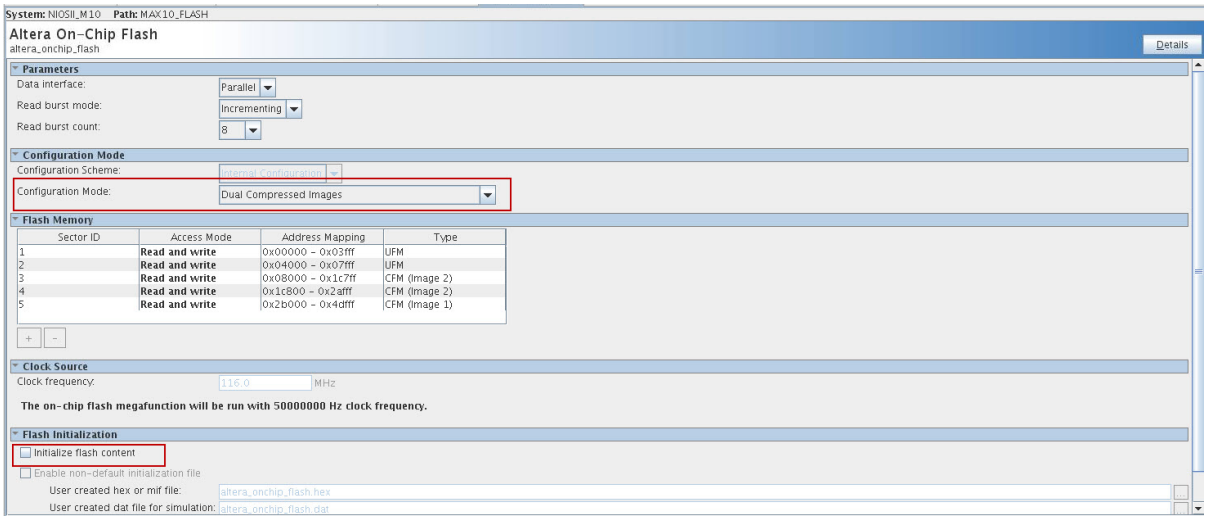
For information on Remote System Upgrade in Dual Compressed Images, please refer to MAX 10 FPGA Configuration User Guide, section Remote System Upgrade in Dual Compressed Images.

Qsys Settings

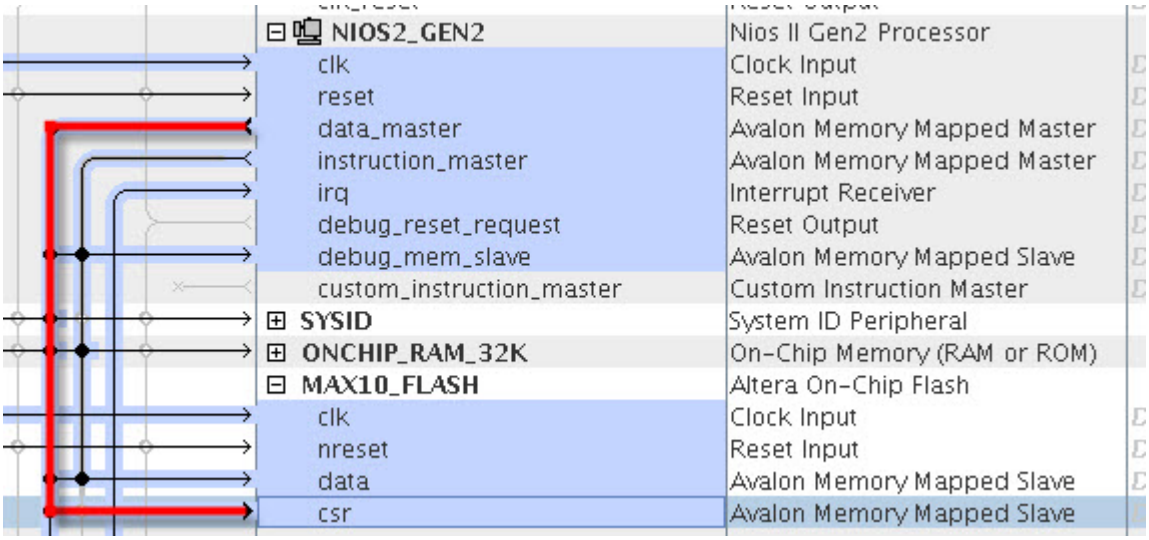
1. In Nios II Gen2 Processor parameter editor, set the **Reset vector memory:** to **MAX10_FLASH_data**and **Exception vector memory:** to **ONCHIP_RAM_32K.s1** for Nios II Gen2 processor.



2. In Altera On-chip Flash IP parameter editor, set the **Configuration Mode:** to **Dual Compressed Images**. Make sure the **Initialize flash content** option is left unchecked. Initializing the on-chip flash during device programming feature is currently not supported in boot option 1 and 2.



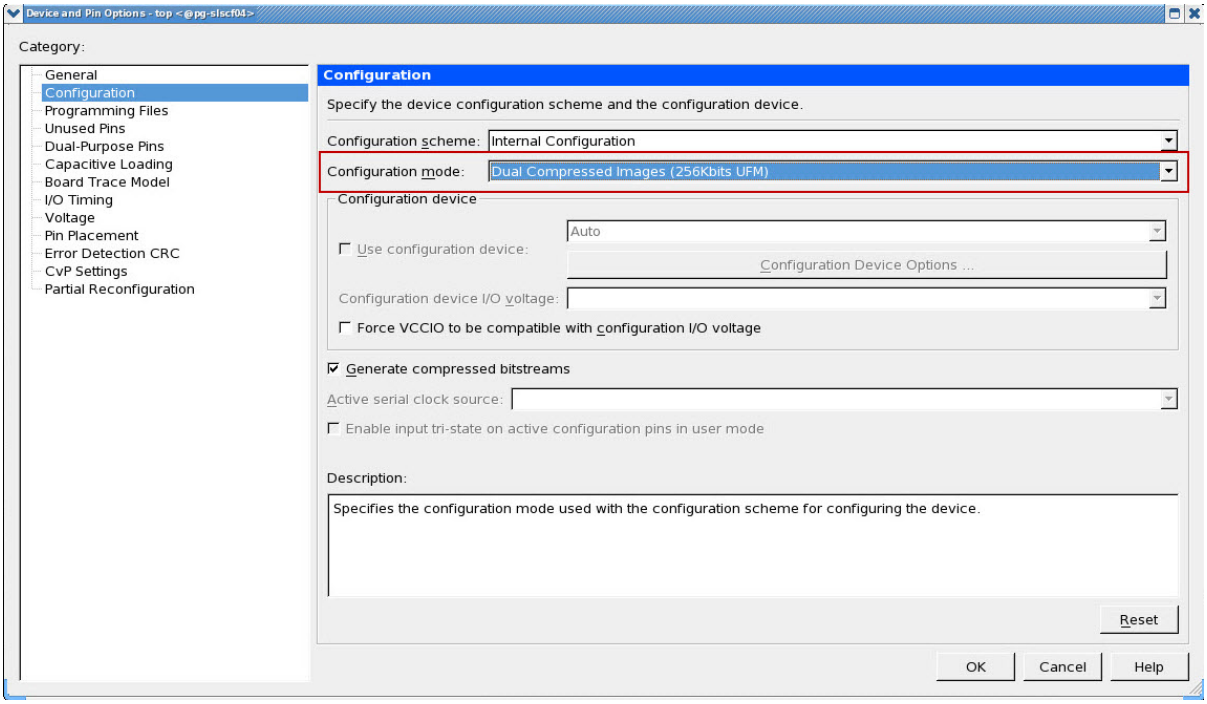
3. From the IP catalog, instantiate the Altera Dual Configuration IP core. Quartus II software will prompt an error message during system compilation if this IP is not instantiated.
4. Make sure the Altera On-chip Flash CSR port is connected to the Nios II Gen2 processor data master to enable write and erase operations.



5. Click **Generate HDL**, the **Generation** dialog box appears.
6. Specify output file generation options, and then click **Generate**.

Quartus II Software Settings

1. In Quartus II software, click on **Assignment** -> **Device** -> **Device and Pin Options** -> **Configuration**. Set **Configuration mode:** to **Dual Compressed Images (256Kbits UFM)**[3].



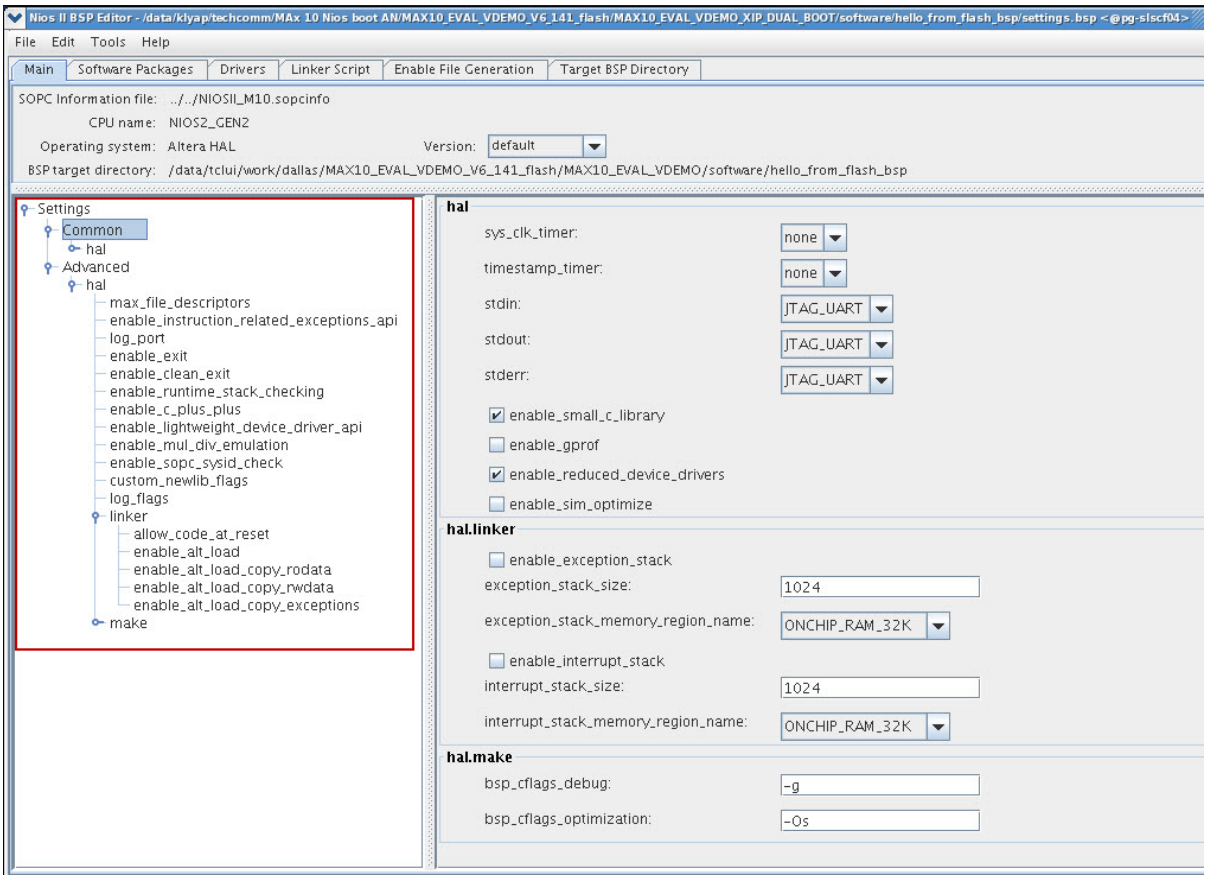
2. Click **OK** to exit the **Device and Pin Options window**.
3. Click **OK** to exit the **Device** window.
4. Click **Start Compilation** to compile your project and generate the .sof file.

[3] 256Kbits UFM shown in the following snapshot is the size of the UFM sector specific to 10M08SAE144C8GES device. The size of UFM sector will vary according to your device selection.

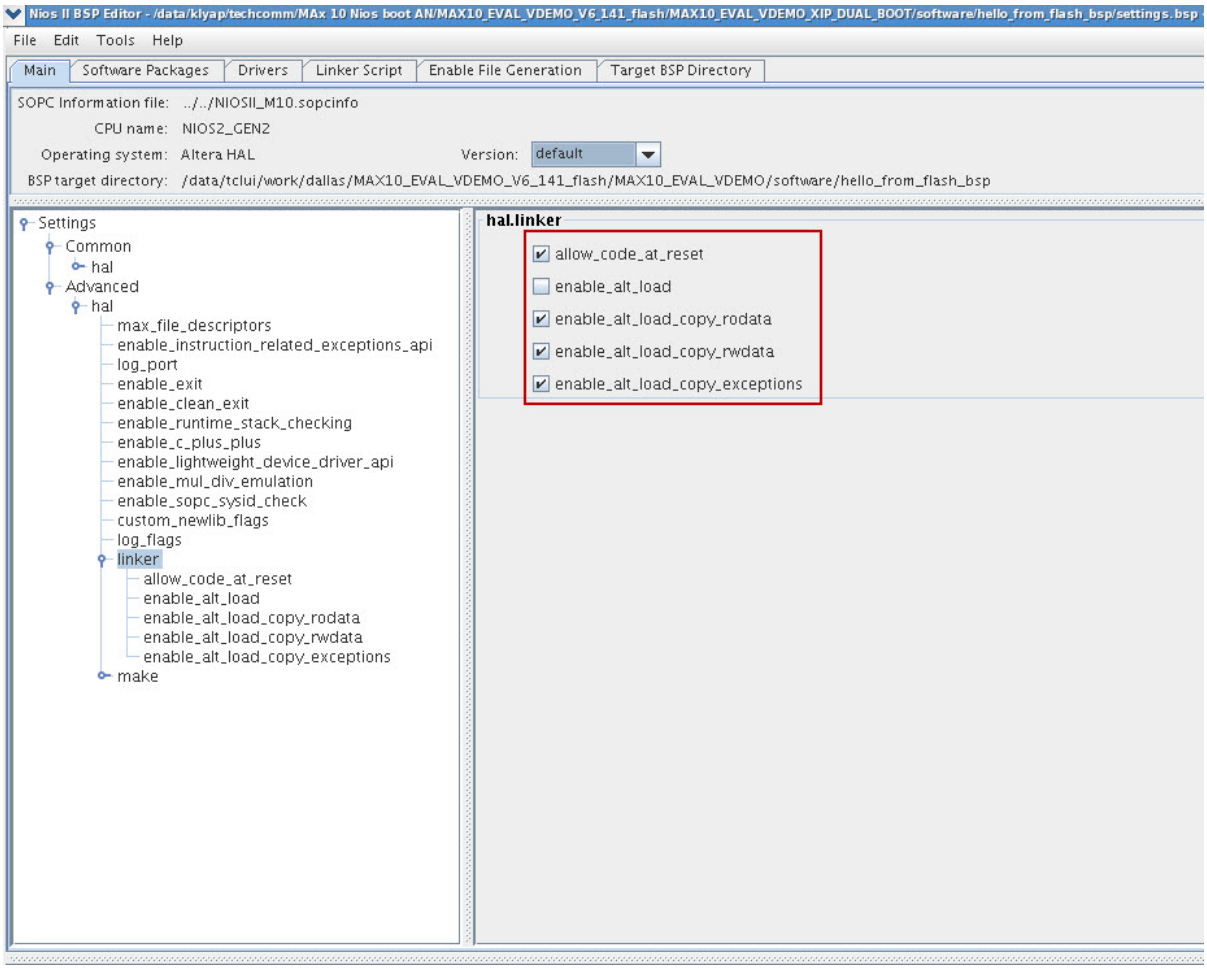
BSP Editor Settings

You must edit the BSP editor settings according to the selected Nios II processor boot options.

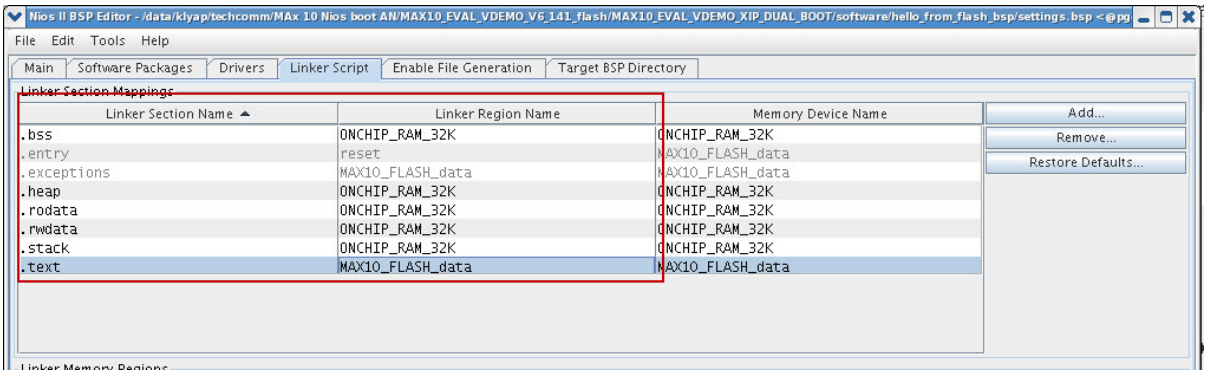
1. In Nios II BSP Editor, click on **Advanced** tab under **Settings**.
2. Click on **hal** to expand the list.
3. Click on **linker** to expand the list.



4. For boot option 1, you must enable **allow_code_at_reset**, **enable_alt_load_copy_rodata**, **enable_alt_load_copy_rwdata**, and **enable_alt_load_copy_exceptions**. For boot option 2, you must leave the **hal.linker** settings as default.

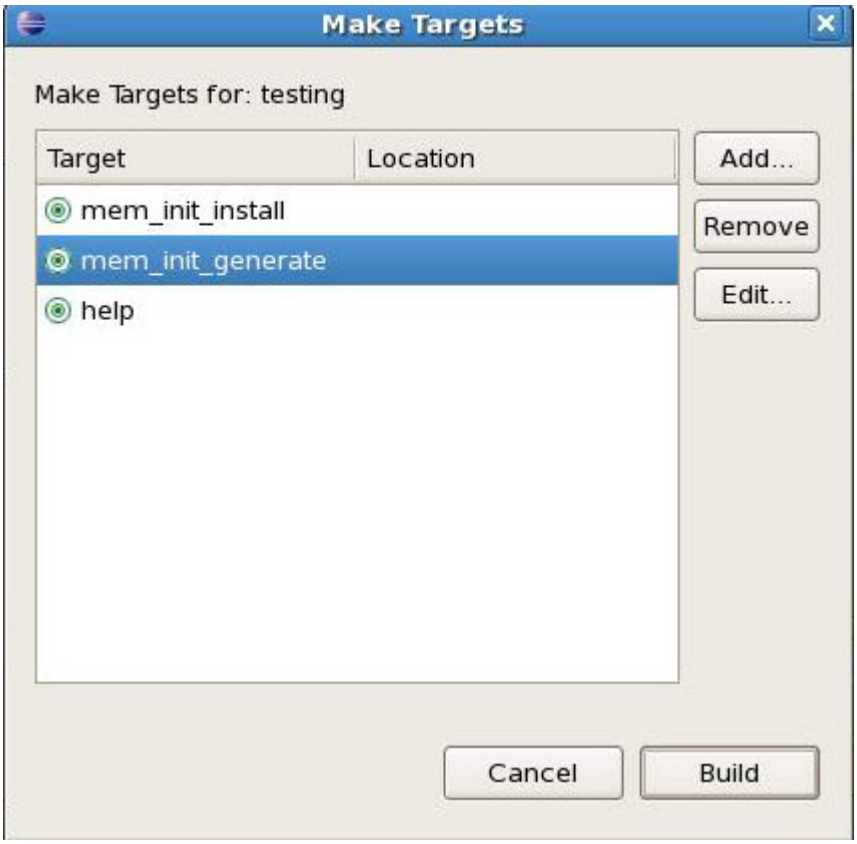


5. Click on **Linker Script** tab in the Nios II BSP Editor.
6. For boot option 1, set the **.text** item in the **Linker Section Name** to the Altera On-chip Flash in the **Linker Region Name**. Set the rest of the items in the **Linker Section Name** list to the Altera On-chip RAM. For boot option 2, set all of the items in the **Linker Section Name** list to Altera On-chip RAM.



HEX File Generation

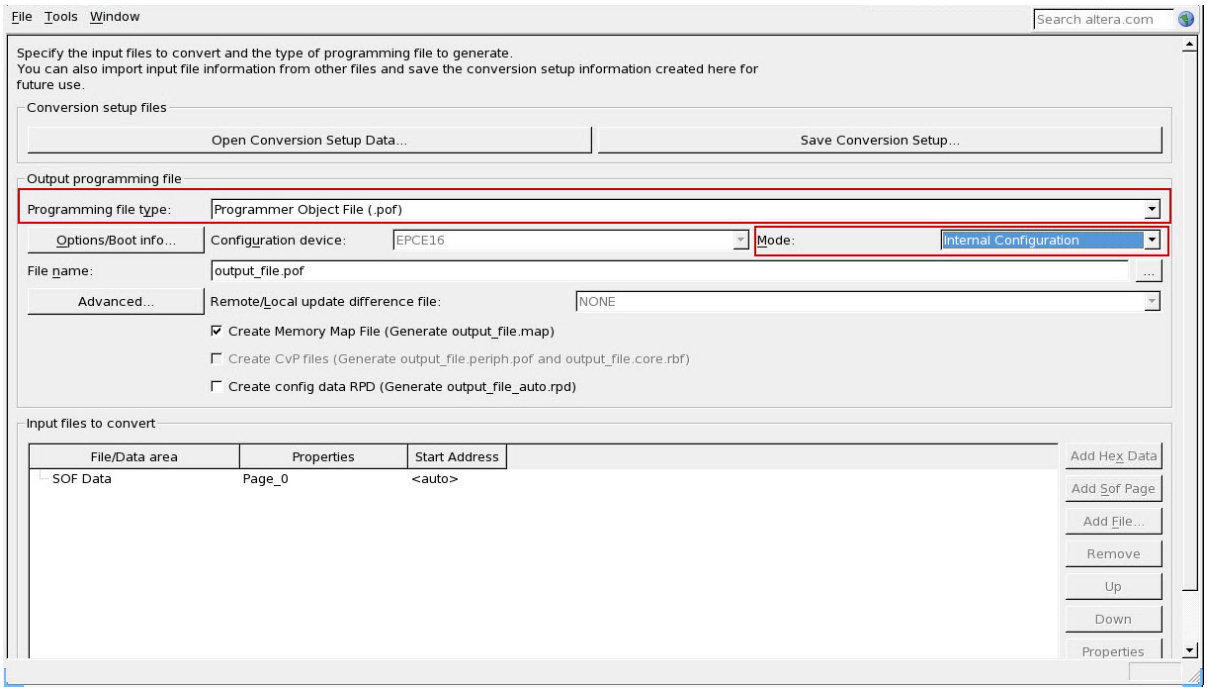
1. In the Nios II SBT tool, right click on your project in the Project Explorer window.
2. Click **Make Targets** -> **Build…**, the Make Targets dialog box appears. You can also press shift + F9 to trigger the Make Target dialog box.
3. Select **mem_init_generate**.
4. Click **Build** to generate the HEX file.



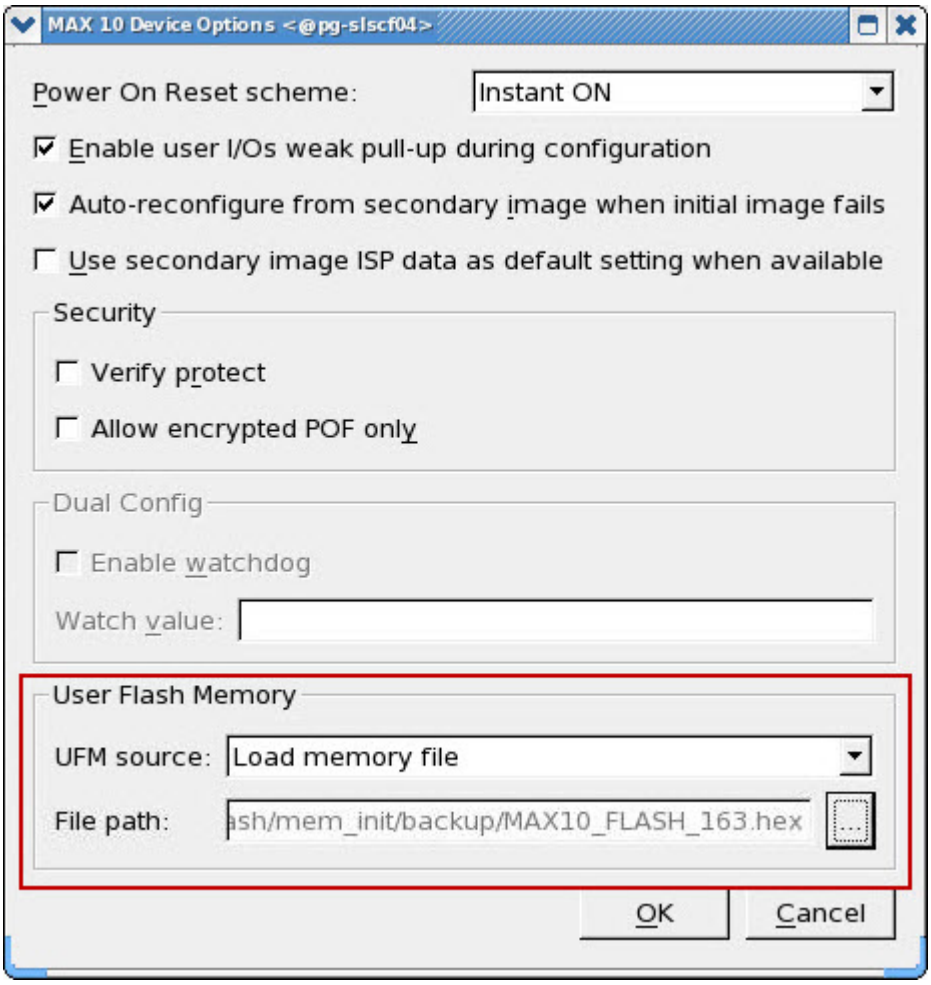
5. For these boot options, the "mem_init_generate" macro will create two HEX files; **on_chip_ram.hex** and **on_chip_flash.hex**.

Programmer Object File (.pof) Generation

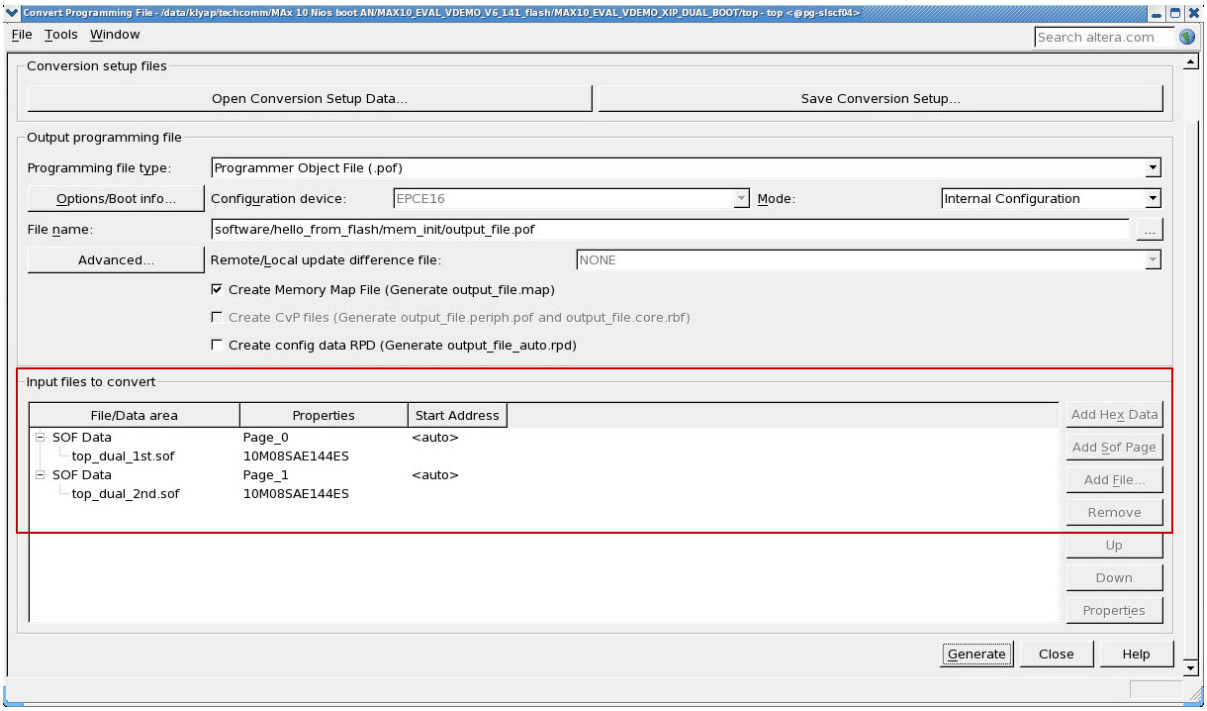
1. In Quartus II, click on **Convert Programming Files** from the **File** tab.
2. Choose **Programmer Object File** as **Programming file type:**.
3. Set **Mode** to **Internal Configuration**.



4. Click on **Options/Boot info...**, the MAX 10 Device Options dialog box appears.
5. Choose **Load memory file** for **UFM source:** option.
6. Browse to the generated Altera On-chip Flash HEX file (on_chip_flash.hex) in the **File path:**. Click **OK**.



7. In the Convert Programming File dialog box, at the **Input files to convert** section, click **Add File...** and point to the first generated Quartus II .sof file to add the .sof file at page 0.
8. Click on **Add Sof Page** to create additional page for .sof file. This creates SOF data Page_1 automatically. Click **Add File...** and point to the second generated Quartus II .sof file to add the .sof file at page 1.



9. Click **Generate** to create the .pof file.
10. Program the .pof file into your MAX 10 device.

Summary of Nios II Processor Vector Configurations and BSP Settings

The following table shows a summary of Nios II processor reset and exception vector configurations, and BSP settings.

Table 7: Summary of Nios II Processor Vector Configurations and BSP Settings

Boot Option	Reset Vector Configuration	Exception Vector Configuration	BSP Settings
Option 1:Nios II processor application execute in-place from Altera On-chip Flash	UFM	On-chip RAM	Enable **allow_code_at_reset** Enable **enable_alt_load_copy_rodata** Enable **enable_alt_load_copy_rwdata** Enable **enable_alt_load_copy_exceptions** **Linker Section Name: .text** set to Altera On-chip Flash. **Linker Section Name: .bss** set to Altera On-chip RAM. **Linker Section Name: .heap** set to Altera On-chip RAM. **Linker Section Name: .rodata** set to Altera On-chip RAM. **Linker Section Name: .rwdata** set to Altera On-chip RAM. **Linker Section Name: .stack** set to Altera On-chip RAM.
Option 2: Nios II processor application copied from UFM to RAM using boot copier	UFM	On-chip RAM	All **hal.linker** settings leave as default. **Linker Section Name: .bss** set to Altera On-chip RAM. **Linker Section Name: .heap** set to Altera On-chip RAM. **Linker Section Name: .rodata** set to Altera On-chip RAM. **Linker Section Name: .rwdata** set to Altera On-chip RAM. **Linker Section Name: .stack** set to Altera On-chip RAM. **Linker Section Name: .text** set to Altera On-chip RAM.

Booting Elements

All Nios II processor boot options introduced in this application note uses the below booting elements to create the necessary boot files:

- The boot copier
- The Nios II SBT "make mem_init_generate" utility
- The Convert Programming Files feature

Nios II Processor Boot Copier

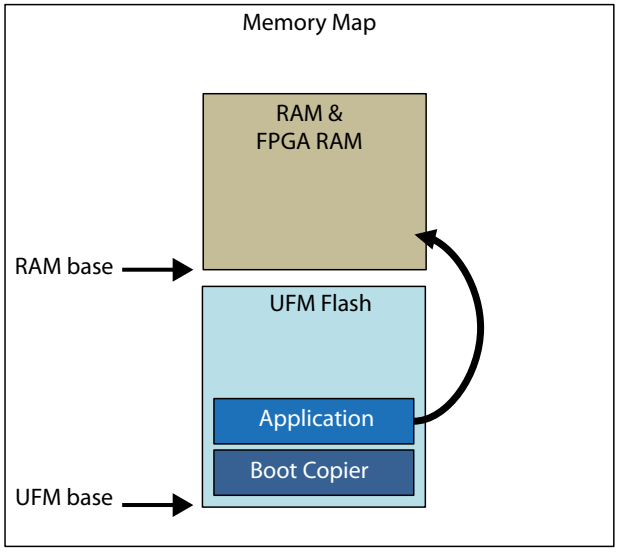
The Nios II processor default boot copier has the following features:

- Supports UFM flash memory
- Locating software application in the memory
- Unpacks and copies software application image to RAM
- Automatically switches to application code in RAM after copy completes

The Nios II processor boot copier is use to support boot option 2 (Nios II soft processor application copied from UFM to RAM using boot copier). The boot copier is automatically append into the HEX file during memory initialization file generation. When you download your .pof file into the FPGA, the boot copier will be placed at the beginning of the UFM sector while the software application will be placed at the end of the boot copier.

The function of the boot copier is to copy the software application to a user desired location such as RAM. Once the copying is done, the boot copier will pass the system control to the application.

Figure 3: Memory Map of An Example System Using Default Boot Copier



Nios II SBT "make mem_init_generate" Command

The Nios II SBT application Makefile “make mem_init_generate” target is responsible for generating memory initialization files using various file conversion tools. This includes a HEX file for the UFM data,

a HEX file for initialization of the on chip RAM in the SOF and a DAT file for initializing the on chip flash model for simulation.

When required, the Nios II SBT tools automatically adds the Nios II processor default boot copier to the system when the executable file (.elf) is converted to memory initialization file (.hex). This operation take place whenever the .text section is located in a different memory that the reset vector points to, which indicates a code copy is required. The file conversion happens during execution of “make mem_init_generate” command.

The "make mem_init_generate" command generates different HEX file content based on the specified boot options:

- For boot option 1, the generated HEX file contains ELF loadable section.
- For boot option 2, the generated HEX file contains the boot copier and the ELF payload.

The Convert Programing Files Option

You can use the Convert Programming Files option in Quartus II software to convert programming files from one file format to another. This tool is used for combining a .sof and a HEX file into a single .pof file for programming into the Altera On-chip Flash.

Document Revision History

This table lists the revision history for this application note.

Table 8: Document Revision History

Date	Version	Changes
January 2015	2015.01.23	Initial release.