Nios II Processor Booting Methods in MAX 10 FPGA Devices

2016.05.24

AN-730 Subscribe Send Feedback

Overview

This document describes the various boot or software execution options available with the Nios II processor and MAX 10 FPGAs.

You can configure the Nios® II processor to boot and execute software from different memory locations, including the MAX10 FPGA on-chip RAM and UFM.

MAX 10 FPGA devices contain on-chip flash that is segmented into two part:

- Configuration Flash Memory (CFM)—store the hardware configuration data for MAX 10 FPGAs.
- User Flash Memory (UFM)—stores the user data or software applications.

Abbreviations

Table 1: List of Abbreviations

Abbreviation	Description
App	Application
CFM	Configuration Flash Memory
DDR3	Double Data Rate type-3 synchronous dynamic random-access memory
EMIF	External Memory Interface
ERAM	Embedded Random Access Memory
HEX/**.hex**	Hexadecimal File[1]
memcpy	Memory copy
OCRAM	On-Chip RAM
POF	Programmer Object File
QSPI	Quad Serial Parallel Interface
RAM	Random Access Memory

[1] An ASCII text file with the extension of .hex that stores the initial memory values for a memory block.

Abbreviation	Description
SBT	Software Build Tools
SOF/**.sof**	SRAM Object File
SPI	Serial Parallel Interface
UART	Universal asynchronous receiver/transmitter
UFM	User Flash Memory
XIP	Execute In Place

Prerequisite

You are required to have the knowledge of instantiating and developing a 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 processor system.
- **Getting Started with the Graphical User Interface.**
  This document provides the details of the Nios II Software Build Tools using graphical user interface.

MAX 10 FPGA On-chip Flash Overview

The Altera On-chip Flash IP core supports the following features:

- Read or write accesses to UFM and CFM sectors using the Avalon MM data and control slave interface.
- Supports page erase, sector erase and sector write.
- Simulation model for UFM read / write accesses using various EDA simulation tool.

Table 2: On-chip Flash Regions in MAX 10 FPGA Devices

Flash Regions	Functionality
Configuration Flash Memory (sectors CFM0-2)	FPGA configuration file storage
User Flash Memory (sectors UFM0-1)	Nios II processor application and/or user data

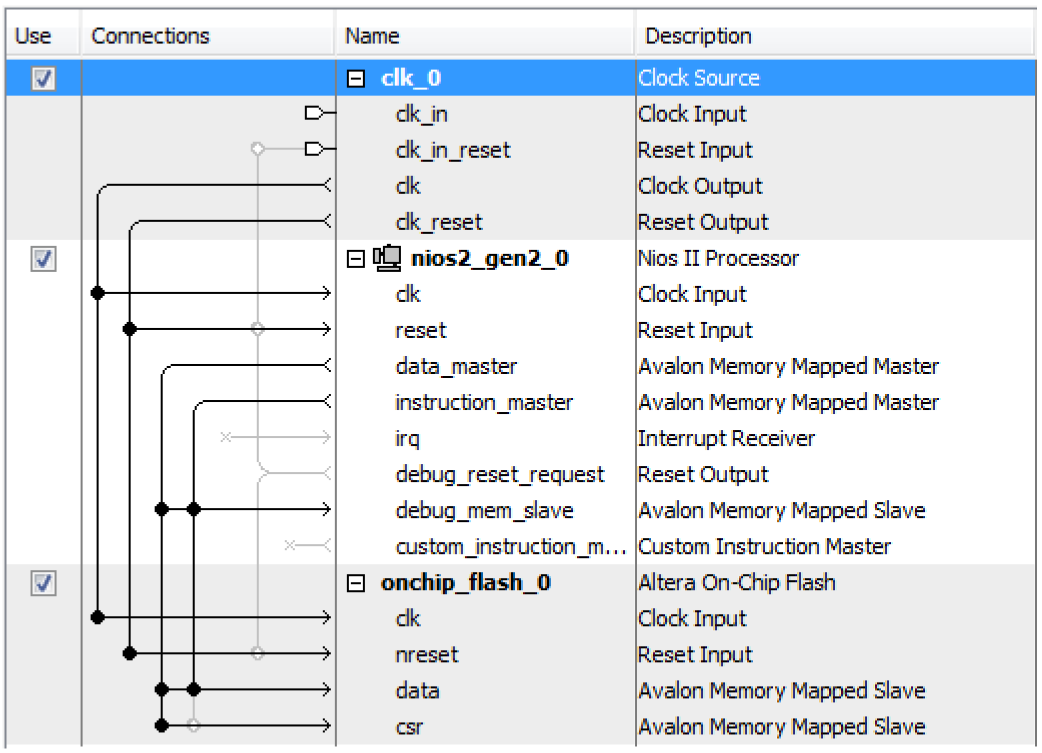
MAX 10 FPGA devices support several configuration modes and some of these modes allow CFM1 and CFM2 to be used as an additional UFM region. The following table shows the storage location of the FPGA configuration images based on the MAX 10 FPGA's configuration modes.

Table 3: Storage Location of FPGA Configuration Images

Configuration Mode	CFM2[2]	CFM1[2]	CFM0
Dual images	Compressed Image 2		Compressed Image 1
Single uncompressed image	UFM[3]	Uncompressed image	
Single uncompressed image with Memory Initialization	Uncompressed image (with pre-initialized on-chip memory content)		
Single compressed image with Memory Initialization	Compressed image (with pre-initialized on-chip memory content)		
Single compressed image	UFM[3]		Compressed Image

You must use the Altera On-chip Flash IP core to access to the flash memory in MAX 10 FPGAs. You can instantiate and connect the Altera On-chip Flash IP to the Nios II processor using the Qsys system design tool in the 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: Connection Example for The Altera On-chip Flash IP and the Nios II Processor



Use	Connections	Name	Description
✓		⊟ clk_0	Clock Source
		clk_in	Clock Input
		clk_in_reset	Reset Input
		clk	Clock Output
		clk_reset	Reset Output
✓		⊟ nios2_gen2_0	Nios II Processor
		clk	Clock Input
		reset	Reset Input
		data_master	Avalon Memory Mapped Master
		instruction_master	Avalon Memory Mapped Master
		irq	Interrupt Receiver
		debug_reset_request	Reset Output
		debug_mem_slave	Avalon Memory Mapped Slave
		custom_instruction_m...	Custom Instruction Master
✓		⊟ onchip_flash_0	Altera On-Chip Flash
		clk	Clock Input
		nreset	Reset Input
		data	Avalon Memory Mapped Slave
		csr	Avalon Memory Mapped Slave

Related Information

- MAX 10 FPGA Configuration User Guide

[2] Sector is NOT supported in 10M02 device.
[3] The CFM sector is configured as virtual UFM.

- MAX 10 User Flash Memory User Guide

MAX 10 FPGA On-chip Flash

The Altera On-chip Flash IP core can provide 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 **Table 4** for available sectors in each individual MAX 10 FPGA device.
- You can configure CFM2 as a virtual UFM by selecting "Single Uncompressed Image" configuration mode.
- You can configure CFM2 and CFM1 as a virtual UFM by selecting "Single Compressed Image" configuration mode.
- The size of each sector varies with the selected MAX 10 FPGA devices.

Table 4: UFM and CFM Sector Size

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

Device	Pages per Sector					Page Size (Kbit)	Maximum User Flash Memory Size (Kbit)[4]	Total Configuration Memory Size (Kbit)	OCRAM Size (Kbit)
	UFM1	UFM0	CFM2	CFM1	CFM0				
10M02	3	3	0	0	34	16	96	544	108
10M04	0	8	41	29	70	16	1248	2240	189
10M08	8	8	41	29	70	16	1376	2240	378
10M16	4	4	38	28	66	32	2368	4224	549
10M25	4	4	52	40	92	32	3200	5888	675
10M40	4	4	48	36	84	64	5888	10752	1260
10M50	4	4	48	36	84	64	5888	10752	1638

ERAM Preload Option

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

[4] The maximum possible value, which is dependent on the configuration mode you select.

All MAX 10 FPGA devices except for the MAX 10 10M02 supports dual FPGA configuration images. The ERAM preload is disabled when you select the dual configuration images mode.

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.

Selecting Single Compressed Image with Memory Initialization or Single Uncompressed Image with Memory Initialization configuration mode will enable the ERAM Preload option but will reduce the size of UFM available.

Nios II Processor Boot Options and Guidelines

There are 5 Nios II processor boot options available.

Table 5: Nios II Processor Boot Options

Boot Option	Application Code Storage	Application Runtime Location	Boot Method	Supported Configuration Mode
Option 1: Nios II processor application executes in-place from Altera On-chip Flash (UFM)	UFM	UFM (XIP) + OCRAM/ External RAM (for data)	alt_load () function	• Single Uncompressed Image • Single Compressed Image • Dual Compressed Images
Option 2: Nios II processor application copied from UFM to RAM using boot copier	UFM	OCRAM/ External RAM	memcpy-based boot copier	• Single Uncompressed Image • Single Compressed Image • Dual Compressed Images
Option 3: Nios II processor application executes in-place from OCRAM	OCRAM	OCRAM	No boot copier is required	• Single uncompressed image with Memory Initialization • Single compressed image with Memory Initialization
Option 4: Nios II processor application executes in-place from QSPI flash	QSPI flash	QSPI (XIP) + OCRAM/ External RAM (for data)	alt_load() function	• Single Uncompressed Image • Single Compressed Image • Dual Compressed Images
Option 5: Nios II processor application copied from QSPI flash to RAM using boot copier	QSPI flash	OCRAM/ External RAM	memcpy-based boot copier	• Single Uncompressed Image • Single Compressed Image • Dual Compressed Images

Boot Option 1 and Option 2

Boot Option 1: Nios II Processor Application Executes in-place from Altera On-chip Flash (UFM)

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 copies the data sections (**.rodata**, **.rwdata**, or **.exceptions**) from boot memory to RAM based on the BSP settings. The code section (**.text**), which is a read only section, remains in the Altera On-chip Flash memory region. This setup minimizes the RAM usage but may limit the code execution performance as access to the flash memory is slower than the on-chip RAM.

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

If you are debugging the application using the source-level debugger, you must use a hardware break-point to debug because the UFM does not support random memory access. Random memory access is required for soft break-point debug.

Figure 2: Boot Option 1 Block Diagram

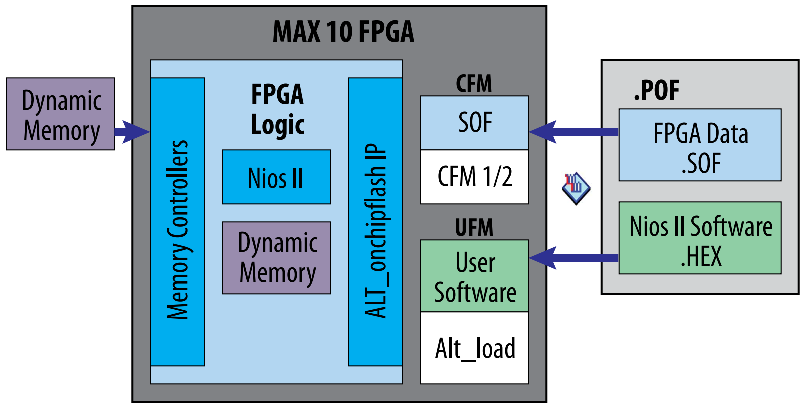


Table 6: RAM and ROM Size Requirement for Boot Option 1

You can manually determine the required RAM size for the Altera On-Chip Flash by referring to the initial part of the **.objdump** file, created when you build your application.

RAM Size Requirement	ROM Size Requirement
Equivalent to the dynamic memory space usage during run time which is the sum of the maximum heap and stack size.	Executable code must not exceed the size of the UFM.

Boot Option 2: Nios II Processor Application Copied from UFM to RAM using Boot Copier

Altera recommends this solution for MAX 10 FPGA 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 within the UFM at an offset which is the same address with the reset vector. The Nios II application is located next to the boot copier.

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

Figure 3: Boot Option 2 Block Diagram

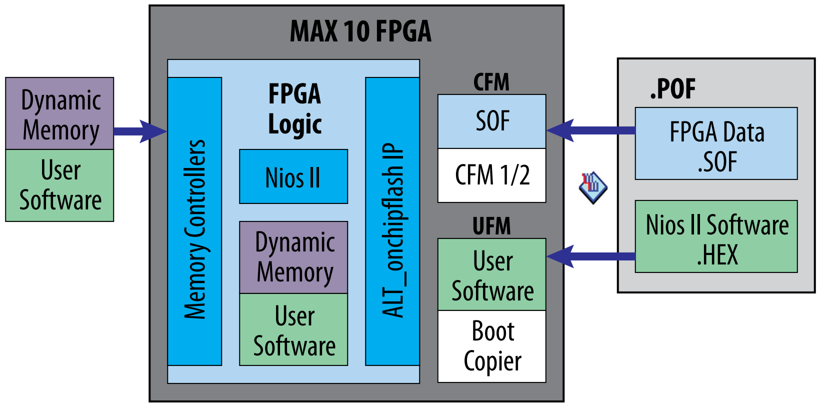
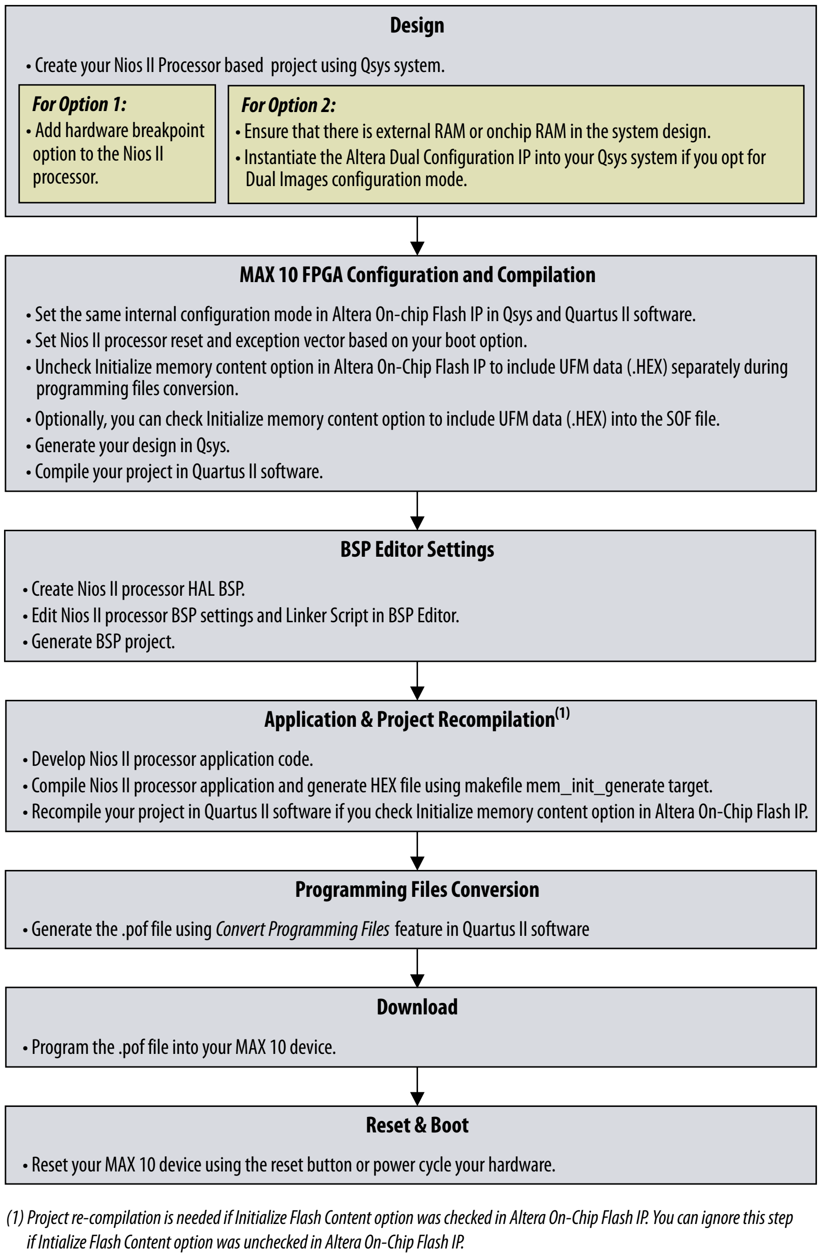


Table 7: RAM and ROM Size Requirement for Boot Option 2

- You can manually determine the required RAM size for the Altera On-Chip Flash by referring to the initial part of the **.objdump** file, created when you build your application.
- The OCRAM size is limited, you have to ensure that the size is sufficient for application execution.

RAM Size Requirement	ROM Size Requirement
Equivalent to the executable code and dynamic memory size required by user program.	Executable code and boot copier must not exceed the size of the UFM.

Figure 4: Configuration and Booting Flow for Option 1 and Option 2


Design
• Create your Nios II Processor based project using Qsys system.
For Option 1:
• Add hardware breakpoint option to the Nios II processor.
For Option 2:
• Ensure that there is external RAM or onchip RAM in the system design.
• Instantiate the Altera Dual Configuration IP into your Qsys system if you opt for Dual Images configuration mode.
MAX 10 FPGA Configuration and Compilation
• Set the same internal configuration mode in Altera On-chip Flash IP in Qsys and Quartus II software.
• Set Nios II processor reset and exception vector based on your boot option.
• Uncheck Initialize memory content option in Altera On-Chip Flash IP to include UFM data (.HEX) separately during programming files conversion.
• Optionally, you can check Initialize memory content option to include UFM data (.HEX) into the SOF file.
• Generate your design in Qsys.
• Compile your project in Quartus II software.
BSP Editor Settings
• Create Nios II processor HAL BSP.
• Edit Nios II processor BSP settings and Linker Script in BSP Editor.
• Generate BSP project.
Application & Project Recompilation(1)
• Develop Nios II processor application code.
• Compile Nios II processor application and generate HEX file using makefile mem_init_generate target.
• Recompile your project in Quartus II software if you check Initialize memory content option in Altera On-Chip Flash IP.
Programming Files Conversion
• Generate the .pof file using Convert Programming Files feature in Quartus II software
Download
• Program the .pof file into your MAX 10 device.
Reset & Boot
• Reset your MAX 10 device using the reset button or power cycle your hardware.
(1) Project re-compilation is needed if Initialize Flash Content option was checked in Altera On-Chip Flash IP. You can ignore this step if Intialize Flash Content option was unchecked in Altera On-Chip Flash IP.

Related Information

MAX 10 FPGA Configuration User Guide, section Remote System Upgrade in Dual Compressed Images.

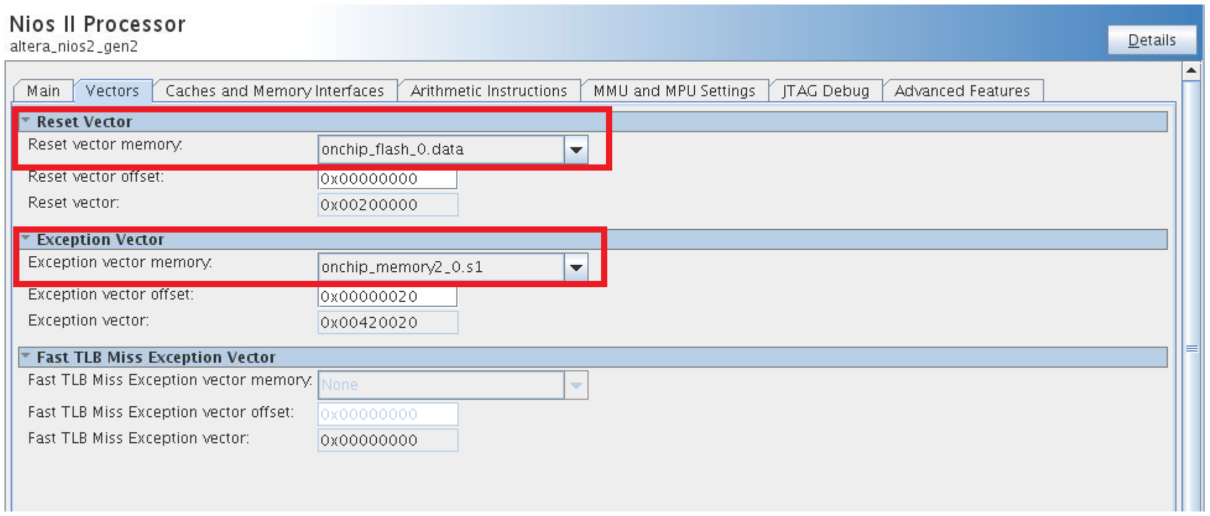
Single Uncompressed/Compressed Image Bootable System Guideline

Qsys Settings

1. In the Nios II Processor parameter editor, set the reset vector memory and exception vector memory based on the boot options below:

Boot Option	Reset vector memory:	Exception vector memory:
Option 1a[5][6]	Altera On-chip Flash	OCRAM/ External RAM
Option 1b[5]	Altera On-chip Flash	Altera On-chip Flash
Option 2	Altera On-chip Flash	OCRAM/ External RAM

Figure 5: Nios II Parameter Editor Settings Boot Option 1a and 2



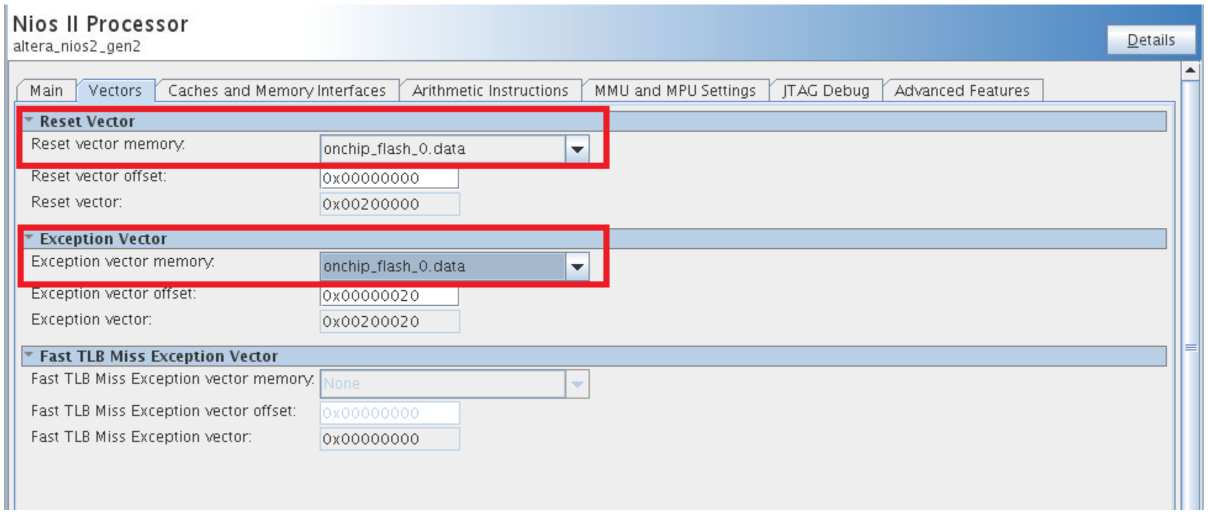
Reset vector memory: onchip_flash_0.data
Reset vector offset: 0x00000000
Reset vector: 0x00200000

Exception vector memory: onchip_memory2_0.s1
Exception vector offset: 0x00000020
Exception vector: 0x00420020

Fast TLB Miss Exception vector memory: None
Fast TLB Miss Exception vector offset: 0x00000000
Fast TLB Miss Exception vector: 0x00000000

[5] You can set the exception vector for Boot Option 1 to OCRAM or External RAM (Option 1a) or Altera On-chip Flash (option 1b) according to your design preference.
[6] Boot option 1a which sets exception vector memory to OCRAM or External RAM is recommended to make the interrupt processing faster.

Figure 6: Nios II Parameter Editor Settings Boot Option 1b



Reset Vector

Reset vector memory: onchip_flash_0.data

Reset vector offset: 0x00000000

Reset vector: 0x00200000

Exception Vector

Exception vector memory: onchip_flash_0.data

Exception vector offset: 0x00000020

Exception vector: 0x00200020

Fast TLB Miss Exception Vector

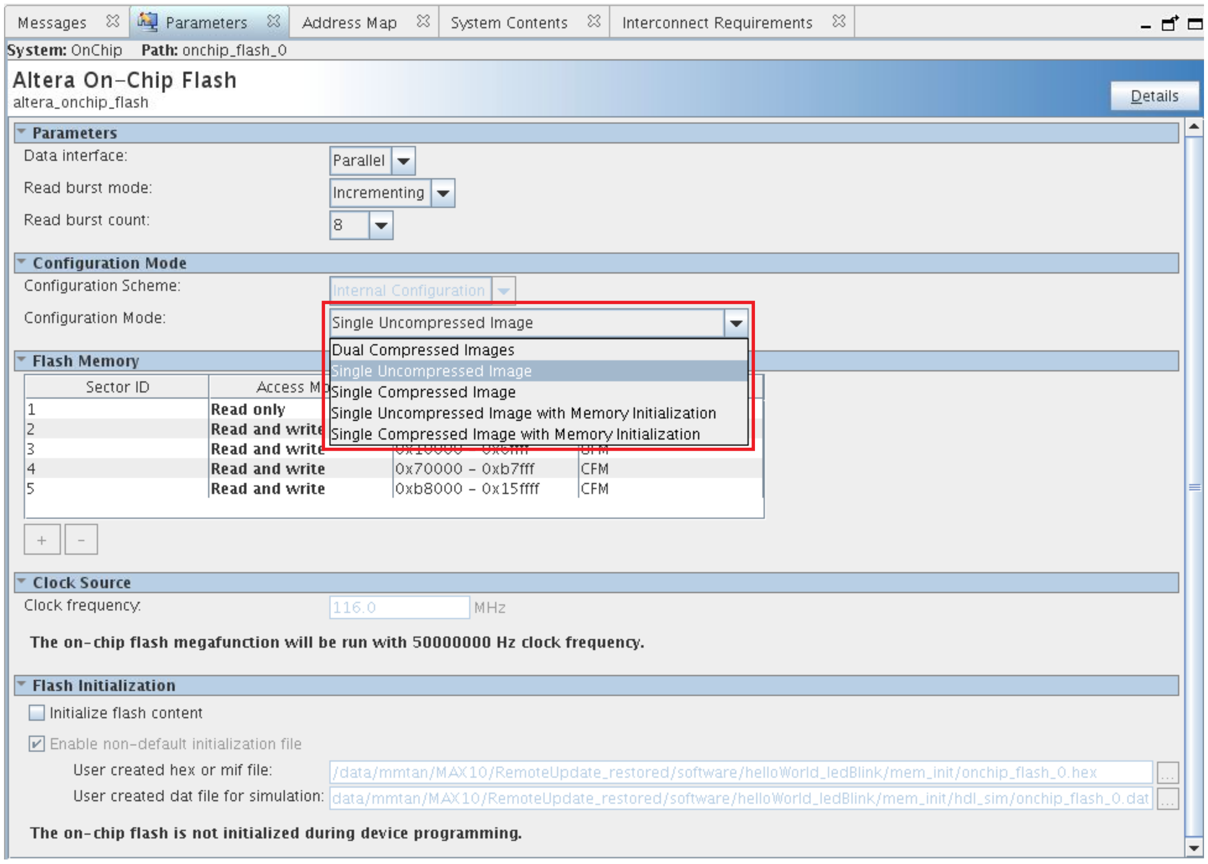
Fast TLB Miss Exception vector memory: None

Fast TLB Miss Exception vector offset: 0x00000000

Fast TLB Miss Exception vector: 0x00000000

2. In the Altera On-chip Flash IP parameter editor, set the **Configuration Mode** to **Single Uncompressed Image** or **Single Compressed Image**.

Figure 7: Configuration Mode Selection in Altera On-Chip Flash Parameter Editor



System: OnChip **Path:** onchip_flash_0

Altera On-Chip Flash
altera_onchip_flash

Parameters

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

Configuration Mode

Configuration Scheme: Internal Configuration
Configuration Mode: Single Uncompressed Image

- Dual Compressed Images
- Single Uncompressed Image
- Single Compressed Image
- Single Uncompressed Image with Memory Initialization
- Single Compressed Image with Memory Initialization

Flash Memory

Sector ID	Access M		
1	**Read only**	[illegible]	[illegible]
2	**Read and write**	[illegible]	[illegible]
3	**Read and write**	0x10000 – 0x6ffff	UFM
4	**Read and write**	0x70000 – 0xb7fff	CFM
5	**Read and write**	0xb8000 – 0x15ffff	CFM

Clock Source

Clock frequency: 116.0 MHz

The on-chip flash megafunction will be run with 50000000 Hz clock frequency.

Flash Initialization

☐ Initialize flash content
☑ Enable non-default initialization file

User created hex or mif file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex
User created dat file for simulation: data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.dat

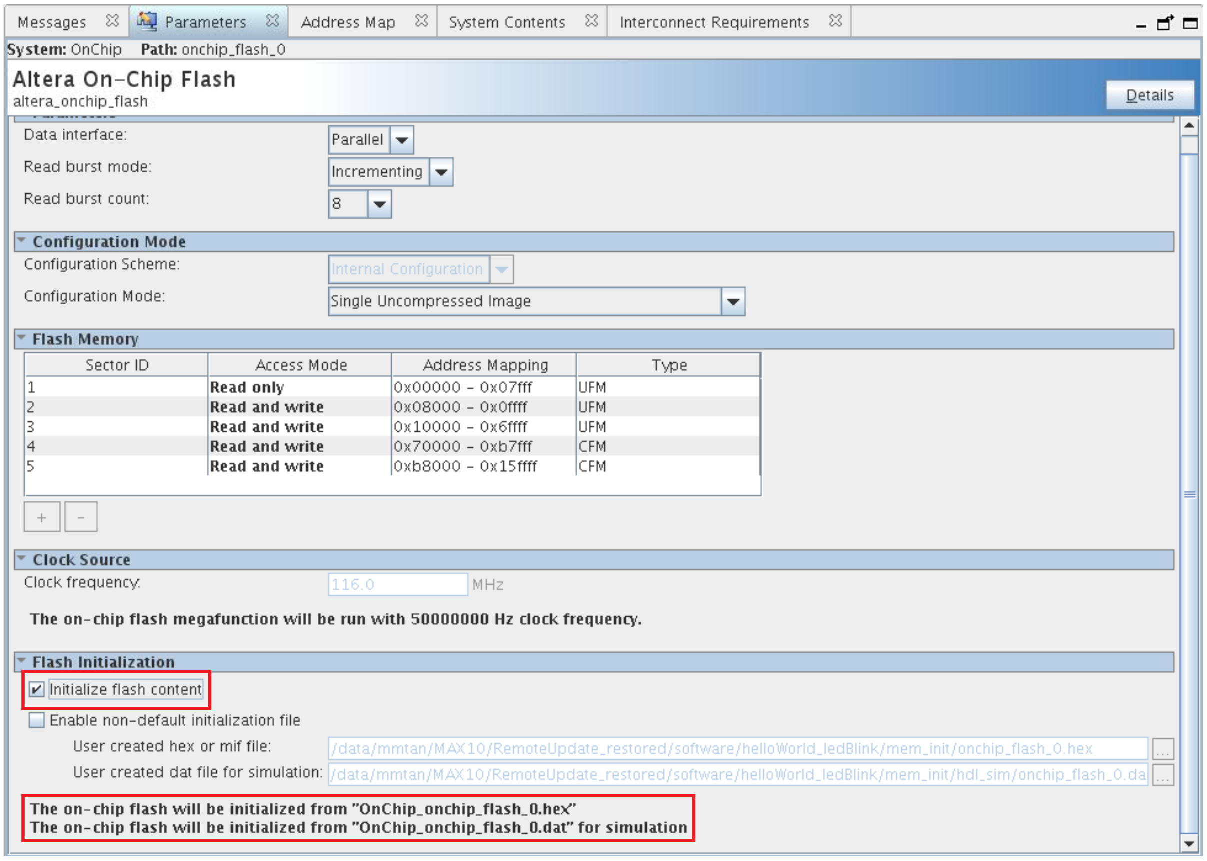
The on-chip flash is not initialized during device programming.

3. Refer to the following table for the options to program UFM data (HEX file) and settings required in Altera On-chip Flash IP.

Options to program UFM data	Method	Settings in Altera On-Chip Flash IP
Option 1: Initialize the UFM data in the SOF	Quartus II includes the UFM initialization data in the SOF during compilation. SOF recompilation is needed if there are changes in the UFM data.	1. Check **Initialize flash content** 2. If default path is used, add **meminit.qip** generated during “make mem_init_ generate” into Quartus II project. Refer to **Figure 9**. Make sure the generated HEX naming matches the default naming. 3. If non-default path is selected, enable the **Enable non-default initialization file** and specify the path of the HEX file. **Note:** For more information about Steps 2 and 3, refer to HEX File Generation section.
Option 2: Combine UFM data with a compiled SOF during programming files (POF) conversion [7]	UFM data is combined with the compiled SOF during the programming files conversion. SOF recompilation is NOT needed even if there are changes in the UFM data.	Uncheck **Initialize flash content**

[7] This is the recommended method for application developer. You are not required to recompile SOF file for application changes during development.

Figure 8: Initialize Flash Contents with Default Initialization File



System: OnChip **Path:** onchip_flash_0

Altera On-Chip Flash
altera_onchip_flash

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

Configuration Mode

Configuration Scheme: Internal Configuration
Configuration Mode: Single Uncompressed Image

Flash Memory

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

Clock Source

Clock frequency: 116.0 MHz

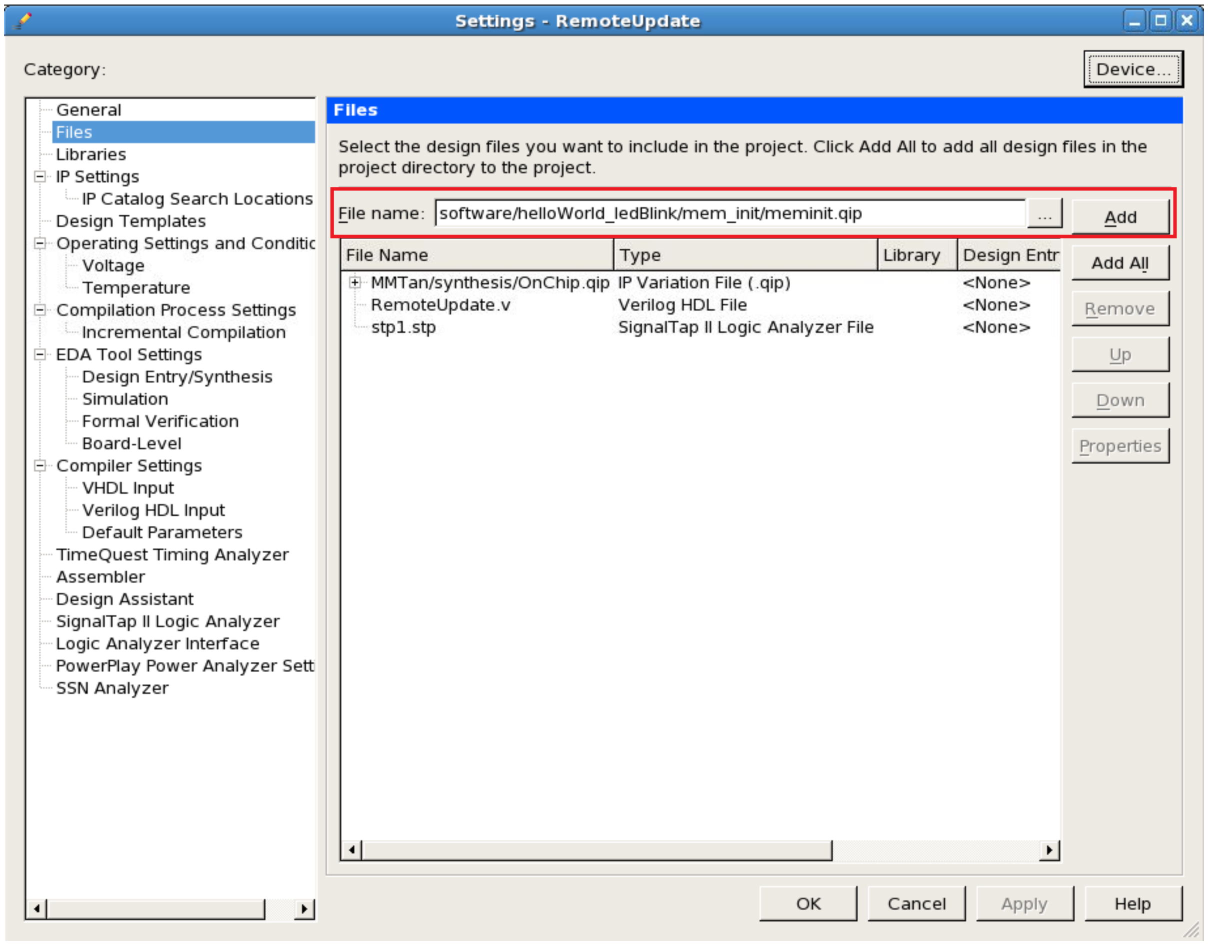
The on-chip flash megafunction will be run with 50000000 Hz clock frequency.

Flash Initialization

[✓] Initialize flash content
[ ] Enable non-default initialization file
User created hex or mif file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex
User created dat file for simulation: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.da

The on-chip flash will be initialized from "OnChip_onchip_flash_0.hex"
The on-chip flash will be initialized from "OnChip_onchip_flash_0.dat" for simulation

Figure 9: Adding **meminit.qip** File in Quartus II

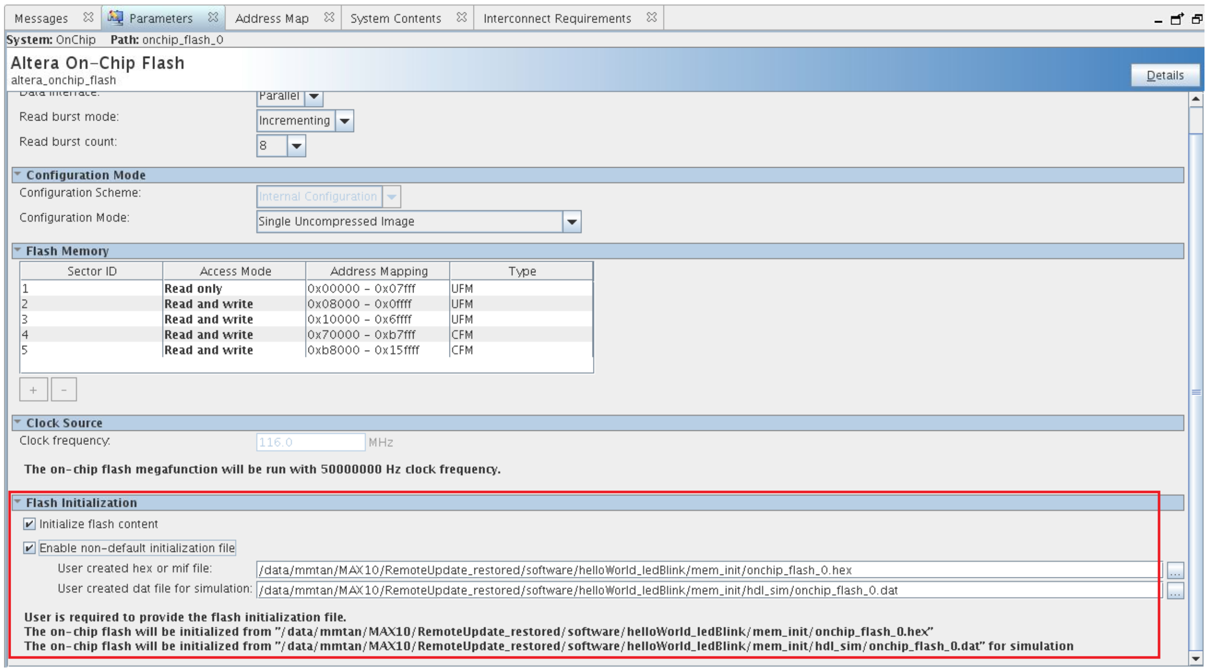


Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name: software/helloWorld_ledBlink/mem_init/meminit.qip

File Name	Type	Library	Design Entr
MMTan/synthesis/OnChip.qip	IP Variation File (.qip)		<None>
RemoteUpdate.v	Verilog HDL File		<None>
stp1.stp	SignalTap II Logic Analyzer File		<None>

Figure 10: Initialize Flash Content with Non-default Initialization File



Messages | Parameters | Address Map | System Contents | Interconnect Requirements

System: OnChip **Path:** onchip_flash_0

Altera On-Chip Flash
altera_onchip_flash

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

Configuration Mode
Configuration Scheme: Internal Configuration
Configuration Mode: Single Uncompressed Image

Flash Memory

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

Clock Source
Clock frequency: 116.0 MHz

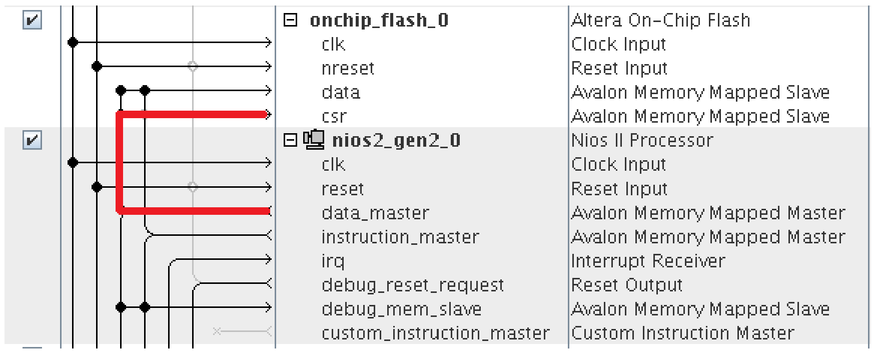
The on-chip flash megafunction will be run with 50000000 Hz clock frequency.

Flash Initialization
☑ Initialize flash content
☑ Enable non-default initialization file
User created hex or mif file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex
User created dat file for simulation: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.dat

User is required to provide the flash initialization file.
The on-chip flash will be initialized from "/data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex"
The on-chip flash will be initialized from "/data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.dat" for simulation

4. Ensure that the Altera On-chip Flash `csr` port is connected to the Nios II processor `data_master` to enable write and erase operations.

Figure 11: CSR Connection to Nios II `data_master`



onchip_flash_0	Altera On-Chip Flash
clk	Clock Input
nreset	Reset Input
data	Avalon Memory Mapped Slave
csr	Avalon Memory Mapped Slave
nios2_gen2_0	Nios II Processor
clk	Clock Input
reset	Reset Input
data_master	Avalon Memory Mapped Master
instruction_master	Avalon Memory Mapped Master
irq	Interrupt Receiver
debug_reset_request	Reset Output
debug_mem_slave	Avalon Memory Mapped Slave
custom_instruction_master	Custom Instruction Master

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

Related Information

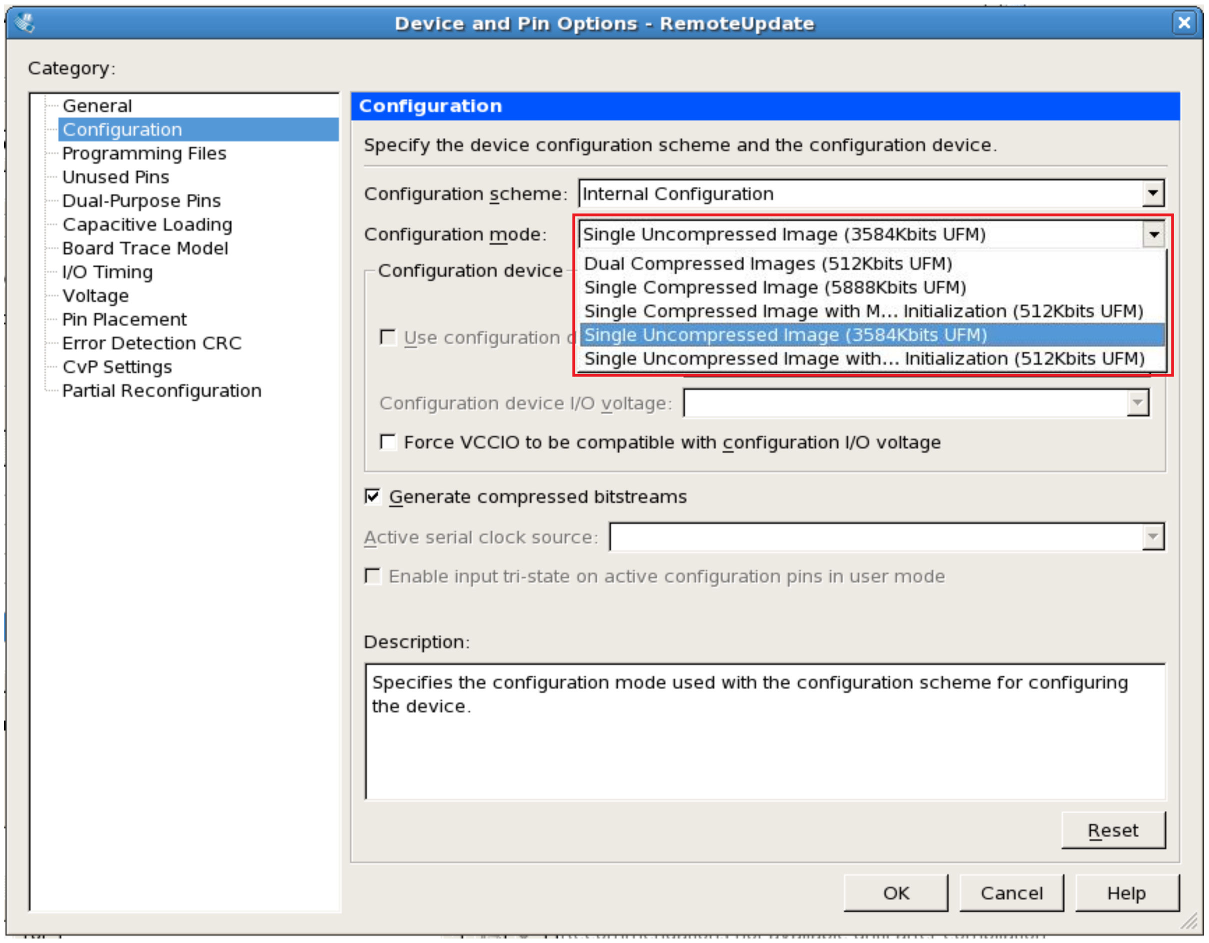
HEX File Generation on page 21

Quartus II Software Settings

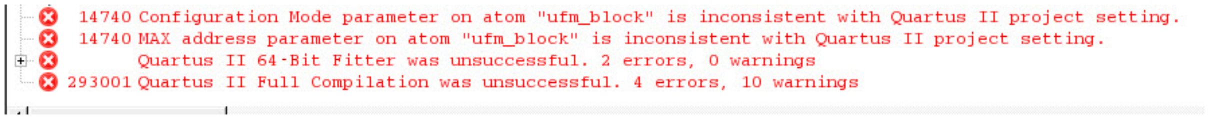
1. In the Quartus II software, click on **Assignment** -> **Device** -> **Device and Pin Options** -> **Configuration**. Set **Configuration mode** to **Single Uncompressed Image** or **Single Compressed Image**.[8]

[8] The size of UFM shown will vary according to your device selection.

Figure 12: Configuration Mode Selection in Quartus II Software


Device and Pin Options - RemoteUpdate
Category:
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC
CvP Settings
Partial Reconfiguration
Configuration
Specify the device configuration scheme and the configuration device.
Configuration scheme: Internal Configuration
Configuration mode: Single Uncompressed Image (3584Kbits UFM)
Dual Compressed Images (512Kbits UFM)
Single Compressed Image (5888Kbits UFM)
Single Compressed Image with M... Initialization (512Kbits UFM)
Single Uncompressed Image (3584Kbits UFM)
Single Uncompressed Image with... Initialization (512Kbits UFM)
Configuration device
Use configuration d
Configuration device I/O voltage:
Force VCCIO to be compatible with configuration I/O voltage
Generate compressed bitstreams
Active serial clock source:
Enable input tri-state on active configuration pins in user mode
Description:
Specifies the configuration mode used with the configuration scheme for configuring the device.
Reset
OK
Cancel
Help

Note: 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.



```
14740 Configuration Mode parameter on atom "ufm_block" is inconsistent with Quartus II project setting.
14740 MAX address parameter on atom "ufm_block" is inconsistent with Quartus II project setting.
      Quartus II 64-Bit Fitter was unsuccessful. 2 errors, 0 warnings
293001 Quartus II Full Compilation was unsuccessful. 4 errors, 10 warnings
```

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.

Related Information

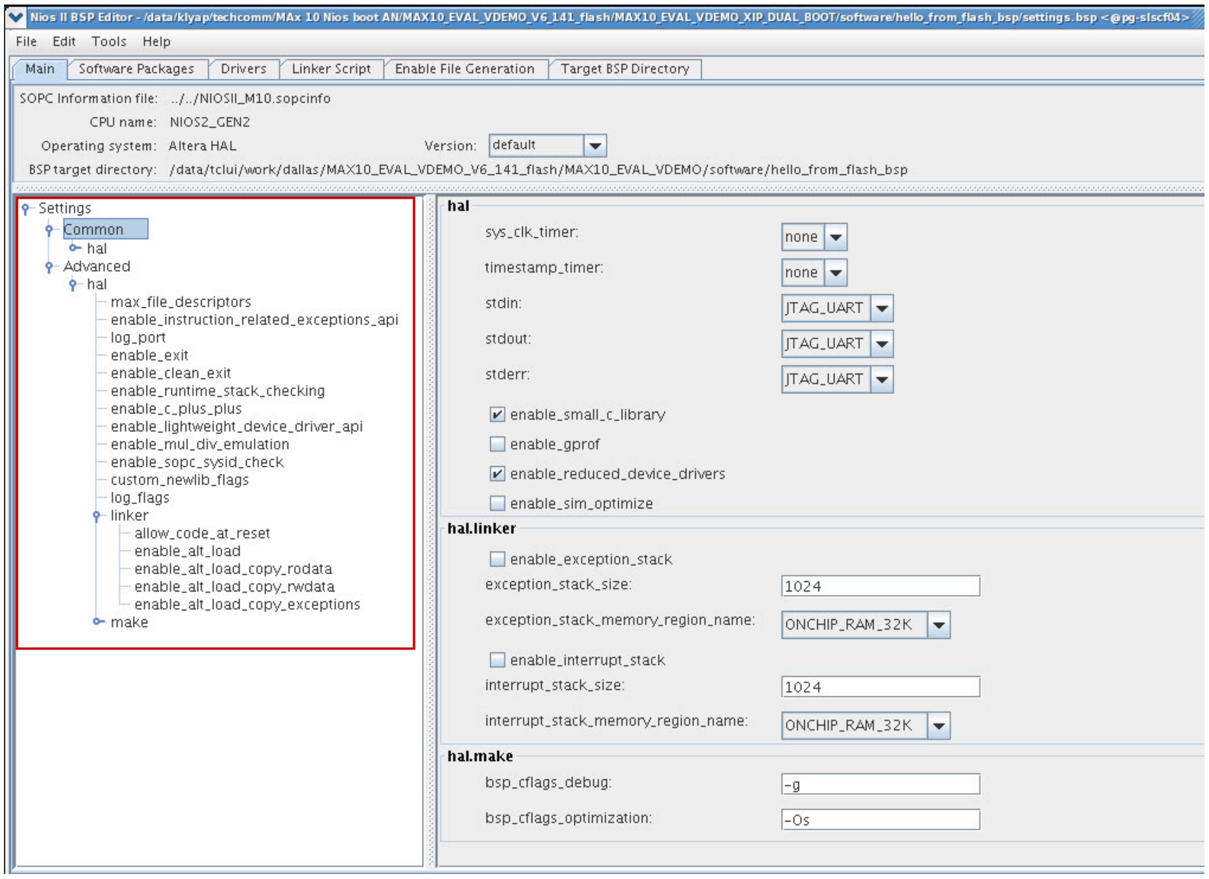
HEX File Generation on page 21

BSP Editor Settings

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

1. In the Nios II SBT tool, right click on your BSP project in the **Project Explorer** window. Select **Nios II** > **BSP Editor...** to open the **Nios II BSP Editor**.
2. In Nios II BSP Editor, click on **Advanced** tab under **Settings**.
3. Click on **hal** to expand the list.
4. Click on **linker** to expand the list.

Figure 13: BSP Editor Settings


Nios II BSP Editor - /data/klyap/techcomm/MAx 10 Nios boot AN/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO_XIP_DUAL_BOOT/software/hello_from_flash_bsp/settings.bsp <@pg-slscf04>
File Edit Tools Help
Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory

SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL   Version: default
BSP target directory: /data/tclui/work/dallas/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO/software/hello_from_flash_bsp

```
Settings
  Common
    hal
  Advanced
    hal
      max_file_descriptors
      enable_instruction_related_exceptions_api
      log_port
      enable_exit
      enable_clean_exit
      enable_runtime_stack_checking
      enable_c_plus_plus
      enable_lightweight_device_driver_api
      enable_mul_div_emulation
      enable_sopc_sysid_check
      custom_newlib_flags
      log_flags
      linker
        allow_code_at_reset
        enable_alt_load
        enable_alt_load_copy_rodata
        enable_alt_load_copy_rwdata
        enable_alt_load_copy_exceptions
      make
```

hal

sys_clk_timer:	none
timestamp_timer:	none
stdin:	JTAG_UART
stdout:	JTAG_UART
stderr:	JTAG_UART

- [✓] enable_small_c_library
- [ ] enable_gprof
- [✓] enable_reduced_device_drivers
- [ ] enable_sim_optimize

hal.linker

- [ ] enable_exception_stack

exception_stack_size:	1024
exception_stack_memory_region_name:	ONCHIP_RAM_32K

- [ ] enable_interrupt_stack

interrupt_stack_size:	1024
interrupt_stack_memory_region_name:	ONCHIP_RAM_32K

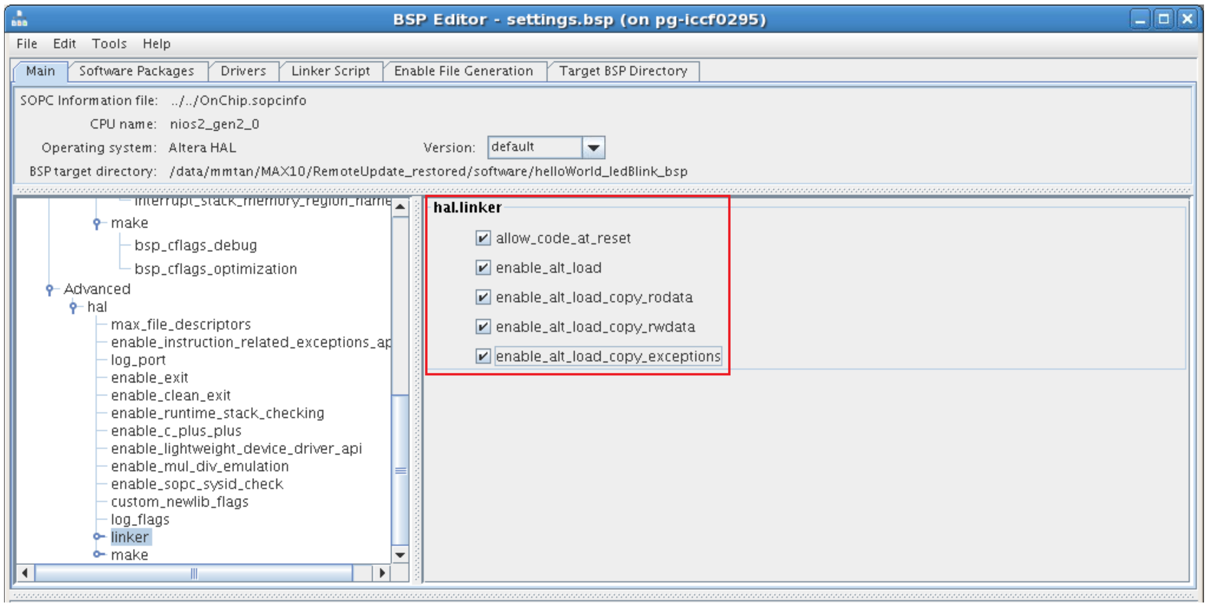
hal.make

bsp_cflags_debug:	-g
bsp_cflags_optimization:	-Os

5. Based on the boot option used, do one of the following:

- For boot option 1a, if exception vector memory is set to OCRAM or External RAM, enable the following:
  - **allow_code_at_reset**
  - **enable_alt_load**
  - **enable_alt_load_copy_rodata**
  - **enable_alt_load_copy_rwdata**
  - **enable_alt_load_copy_exceptions**
- For boot option 1b, if exception vector memory is set to Altera On-chip Flash, enable the following:
  - **allow_code_at_reset**
  - **enable_alt_load**
  - **enable_alt_load_copy_rodata**
  - **enable_alt_load_copy_rwdata**
- For boot option 2, leave all the hal.linker settings unchecked.

Figure 14: Advanced.hal.linker Settings for Boot Option 1a


BSP Editor - settings.bsp (on pg-iccf0295)

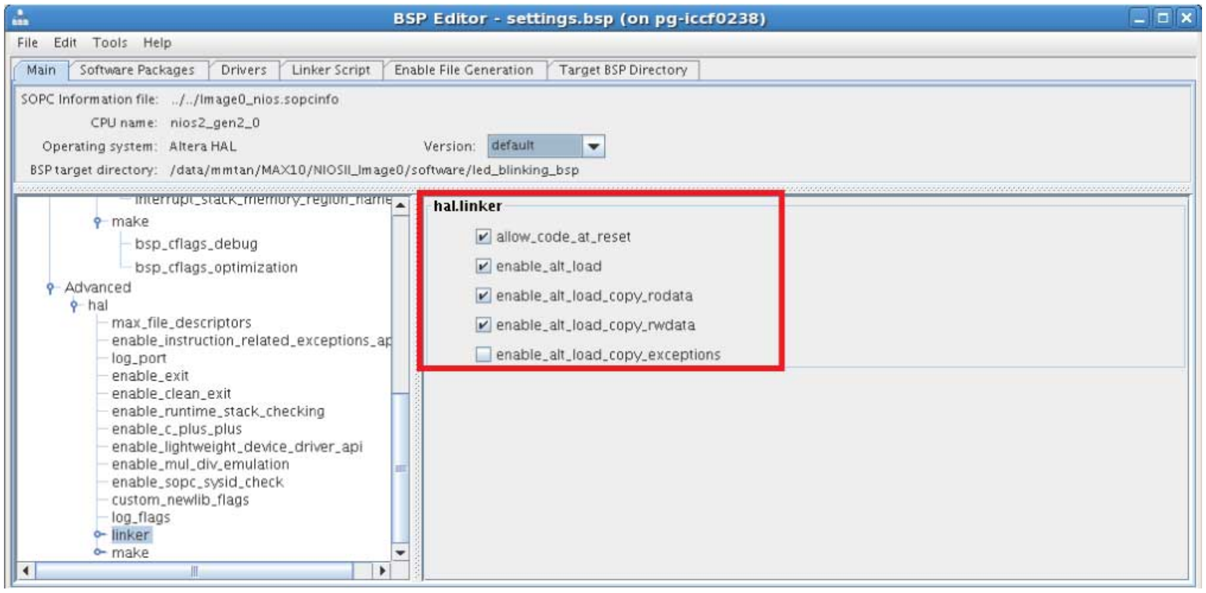
File Edit Tools Help

Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory

SOPC Information file: ../../OnChip.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL Version: default
BSP target directory: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink_bsp

hal.linker
- ✓ allow_code_at_reset
- ✓ enable_alt_load
- ✓ enable_alt_load_copy_rodata
- ✓ enable_alt_load_copy_rwdata
- ✓ enable_alt_load_copy_exceptions

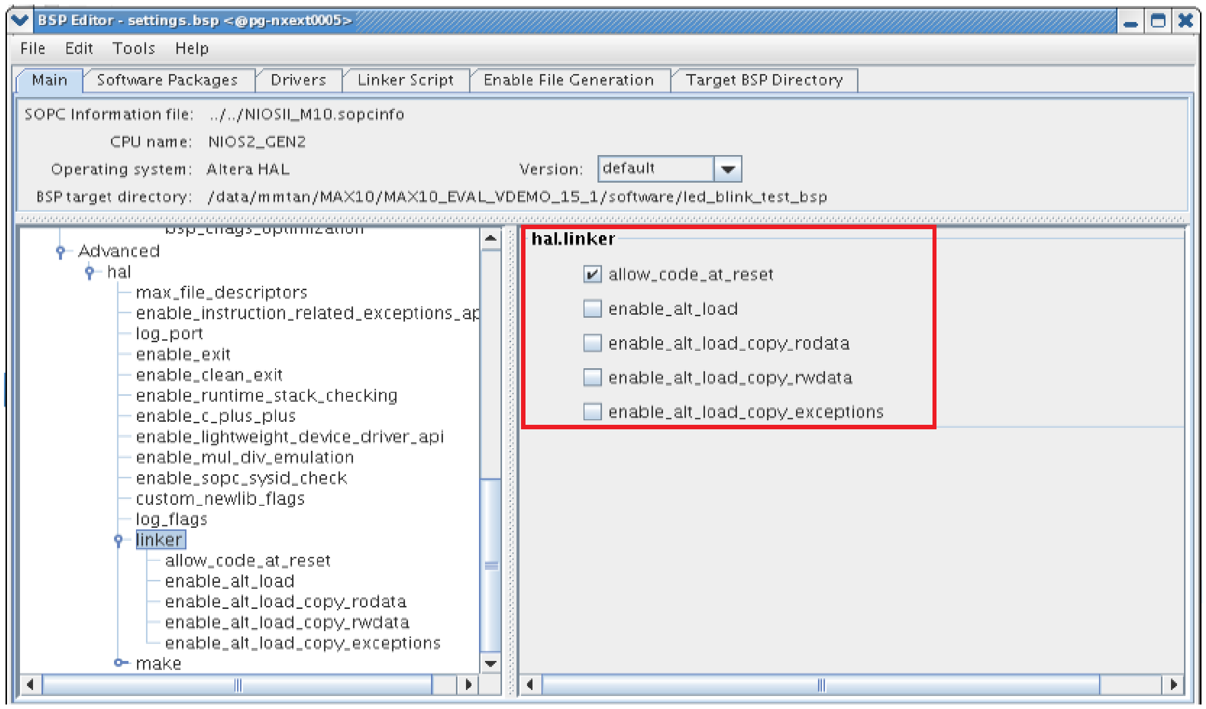
Figure 15: Advanced.hal.linker Settings for Boot Option 1b


BSP Editor - settings.bsp (on pg-iccf0238)

SOPC Information file: ../../image0_nios.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL
Version: default
BSP target directory: /data/mmtan/MAX10/NIOSII_image0/software/led_blinking_bsp

hal.linker
- [✓] allow_code_at_reset
- [✓] enable_alt_load
- [✓] enable_alt_load_copy_rodata
- [✓] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

Figure 16: Advanced.hal.linker Settings for Boot Option 2


BSP Editor - settings.bsp <@pg-nxext0005>

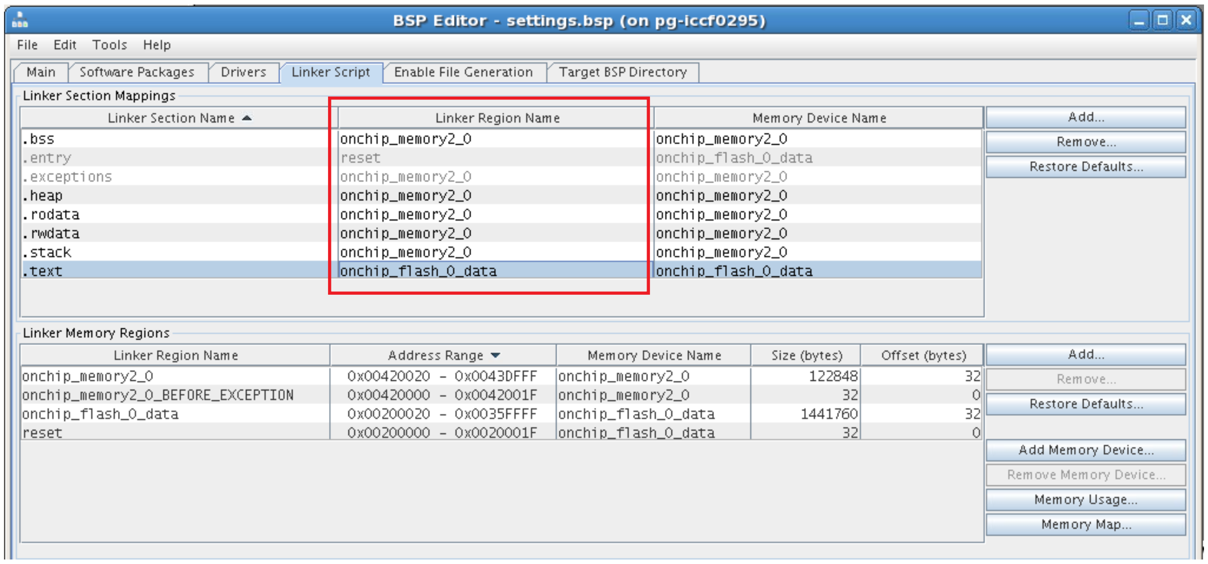
SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL
Version: default
BSP target directory: /data/mmtan/MAX10/MAX10_EVAL_VDEMO_15_1/software/led_blink_test_bsp

hal.linker
- [✓] allow_code_at_reset
- [ ] enable_alt_load
- [ ] enable_alt_load_copy_rodata
- [ ] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

6. Click on **Linker Script** tab in the Nios II BSP Editor.
7. Based on the boot option used, do one of the following:

- For boot option 1a and 1b, 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 Memory (OCRAM) or external RAM.
- For boot option 2, set all of the items in the **Linker Section Name** list to Altera On-chip Memory (OCRAM) or external RAM.

Figure 17: Linker Region Settings for Boot Option 1a and 1b


BSP Editor - settings.bsp (on pg-iccf0295)
File Edit Tools Help
Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory
Add... Remove... Restore Defaults...
Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...

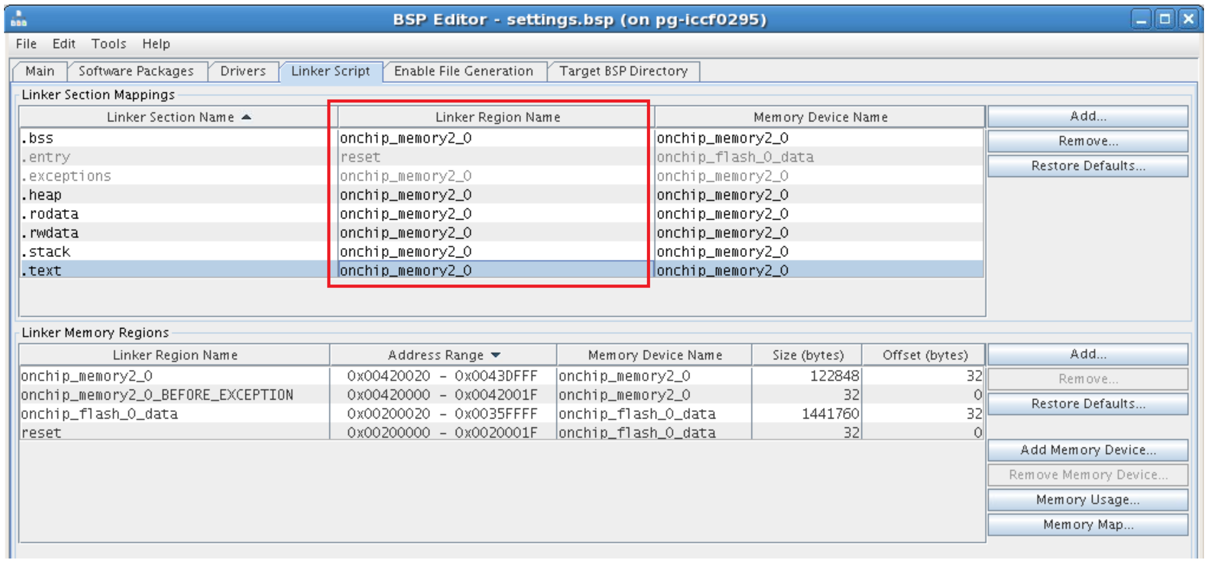
Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	onchip_memory2_0	onchip_memory2_0
.entry	reset	onchip_flash_0_data
.exceptions	onchip_memory2_0	onchip_memory2_0
.heap	onchip_memory2_0	onchip_memory2_0
.rodata	onchip_memory2_0	onchip_memory2_0
.rwdata	onchip_memory2_0	onchip_memory2_0
.stack	onchip_memory2_0	onchip_memory2_0
.text	onchip_flash_0_data	onchip_flash_0_data

Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_memory2_0	0x00420020 - 0x0043DFFF	onchip_memory2_0	122848	32
onchip_memory2_0_BEFORE_EXCEPTION	0x00420000 - 0x0042001F	onchip_memory2_0	32	0
onchip_flash_0_data	0x00200020 - 0x0035FFFF	onchip_flash_0_data	1441760	32
reset	0x00200000 - 0x0020001F	onchip_flash_0_data	32	0

Figure 18: Linker Region Settings for Boot Option 2


BSP Editor - settings.bsp (on pg-iccf0295)
File Edit Tools Help
Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory
Add... Remove... Restore Defaults...
Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...

Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	onchip_memory2_0	onchip_memory2_0
.entry	reset	onchip_flash_0_data
.exceptions	onchip_memory2_0	onchip_memory2_0
.heap	onchip_memory2_0	onchip_memory2_0
.rodata	onchip_memory2_0	onchip_memory2_0
.rwdata	onchip_memory2_0	onchip_memory2_0
.stack	onchip_memory2_0	onchip_memory2_0
.text	onchip_memory2_0	onchip_memory2_0

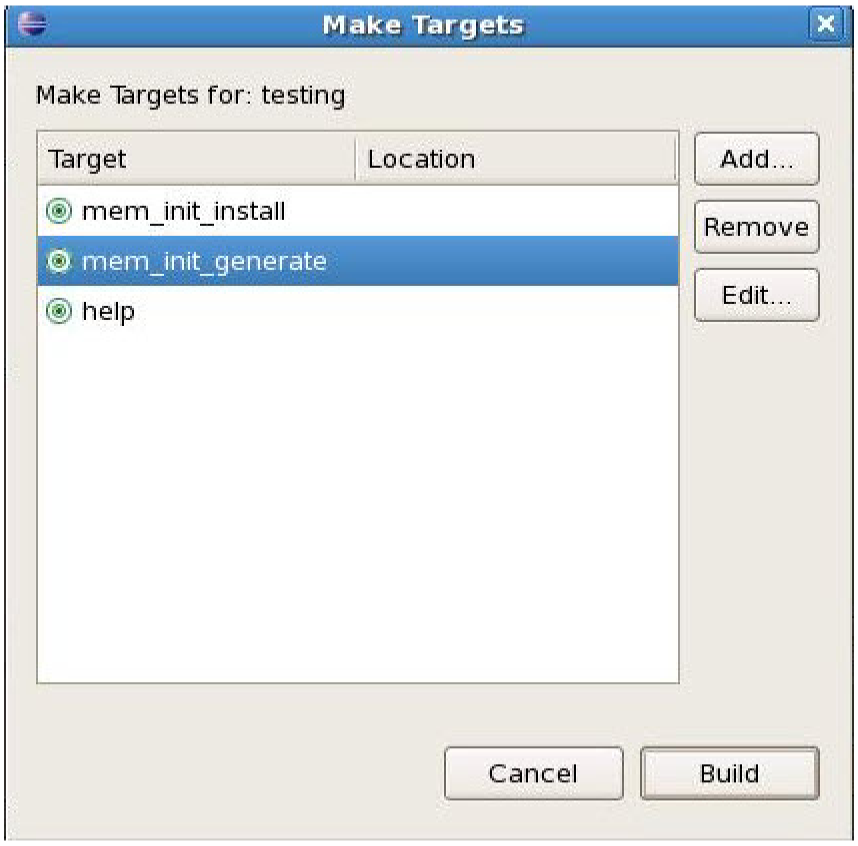
Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_memory2_0	0x00420020 - 0x0043DFFF	onchip_memory2_0	122848	32
onchip_memory2_0_BEFORE_EXCEPTION	0x00420000 - 0x0042001F	onchip_memory2_0	32	0
onchip_flash_0_data	0x00200020 - 0x0035FFFF	onchip_flash_0_data	1441760	32
reset	0x00200000 - 0x0020001F	onchip_flash_0_data	32	0

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.

Figure 19: Selecting mem_init_generate in Make Targets



5. The "mem_init_generate" macro will create two HEX files; **<on_chip_ram.hex>** and **<on_chip_flash.hex>**. The **<on_chip_ram.hex>** will be used for boot option 3 and **<on_chip_flash.hex>** is used for boot option 1 and 2.

   Note:
   - The **mem_init_generate** target also generates a Quartus II IP file (**meminit.qip**). Quartus II software will refer to the **meminit.qip** for the location of the initialization files.
   - All these files can be found under "<project_folder>/software/<application_name>/mem_init" folder.
6. Recompile your project in Quartus II software if you check **Initialize memory content** option in Altera On-Chip Flash IP. This is to include the software data (.HEX) into the SOF file.

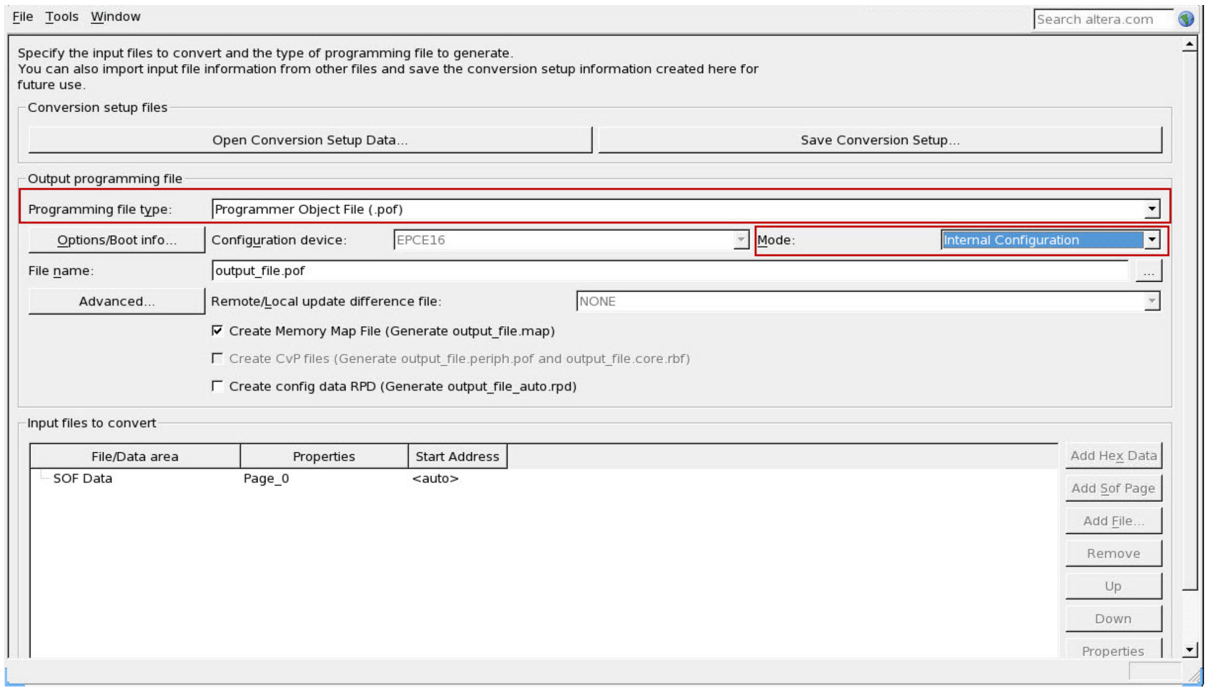
Related Information

- Qsys Settings on page 8
- Quartus II Software Settings on page 15

Programmer Object File (.pof) Generation

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

Figure 20: Convert Programming File Settings



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_file.pof

Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)

☐ Create CvP files (Generate output_file.periph.pof and output_file.core.rbf)

☐ Create config data RPD (Generate output_file_auto.rpd)

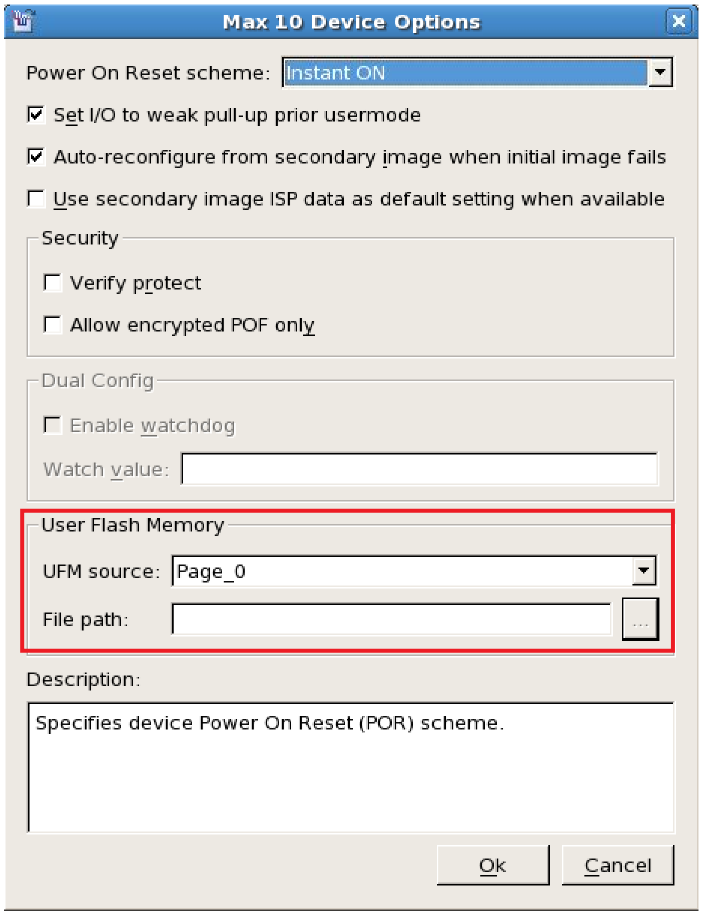
Input files to convert

File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>

4. Click on **Options/Boot info...**, the MAX 10 Device Options dialog box appears.
5. Based on the **Initialize flash content** settings in the Altera On-chip Flash IP, do one of the following:

- If **Initialize flash content** is checked, the UFM initialization data was included in the SOF during Quartus II compilation. Select **Page_0** for **UFM source:** option. Click **OK** and proceed to next step.
- If **Initialize flash content** is not checked, choose **Load memory file** for **UFM source:** option. Browse to the generated Altera On-chip Flash HEX file (**on_chip_flash.hex**) in the **File path:** and click **OK**. This will add UFM data separately to the SOF file during the programming file conversion.

Figure 21: Setting Page_0 for UFM Source if Initialize flash content is Checked



Power On Reset scheme: Instant ON

☑ Set I/O to weak pull-up prior usermode

☑ Auto-reconfigure from secondary image when initial image fails

☐ Use secondary image ISP data as default setting when available

Security

☐ Verify protect

☐ Allow encrypted POF only

Dual Config

☐ Enable watchdog

Watch value:

User Flash Memory

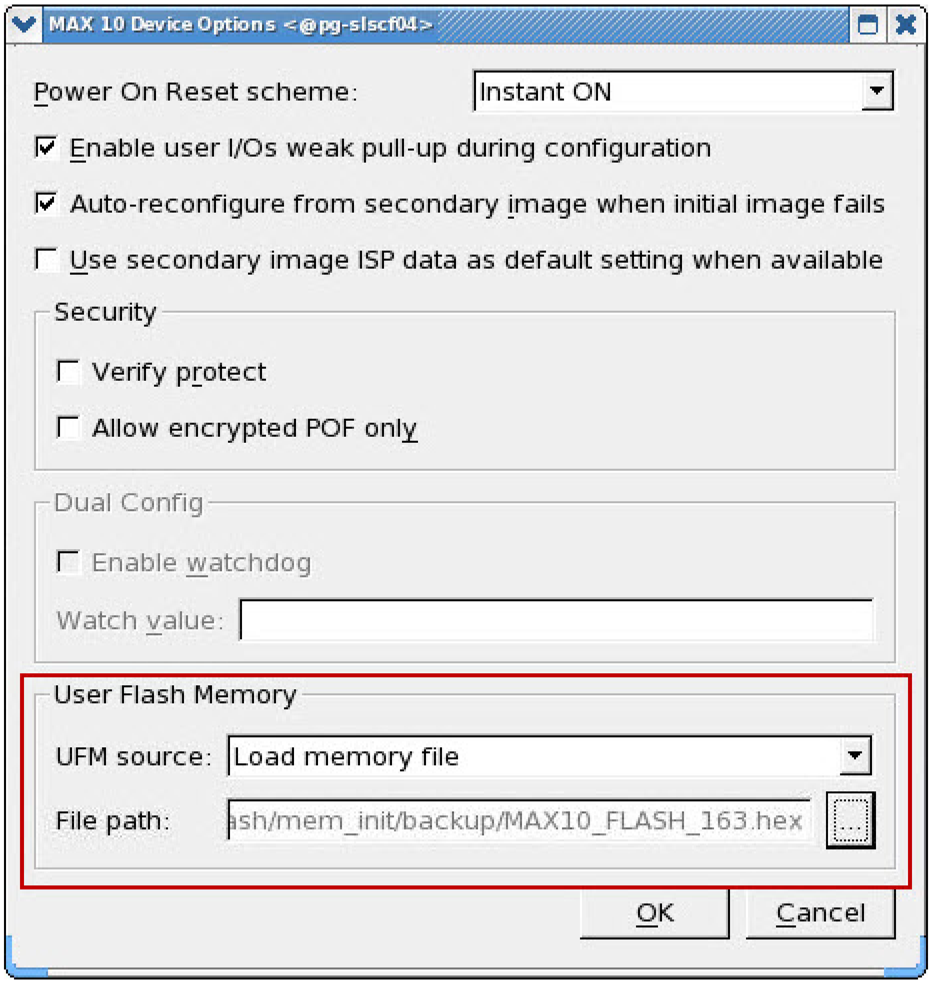
UFM source: Page_0

File path:

Description:

Specifies device Power On Reset (POR) scheme.

Figure 22: Setting Load Memory File for UFM Source if Initialize flash content is not Checked



Power On Reset scheme: Instant ON

☑ Enable user I/Os weak pull-up during configuration

☑ Auto-reconfigure from secondary image when initial image fails

☐ Use secondary image ISP data as default setting when available

Security

☐ Verify protect

☐ Allow encrypted POF only

Dual Config

☐ Enable watchdog

Watch value:

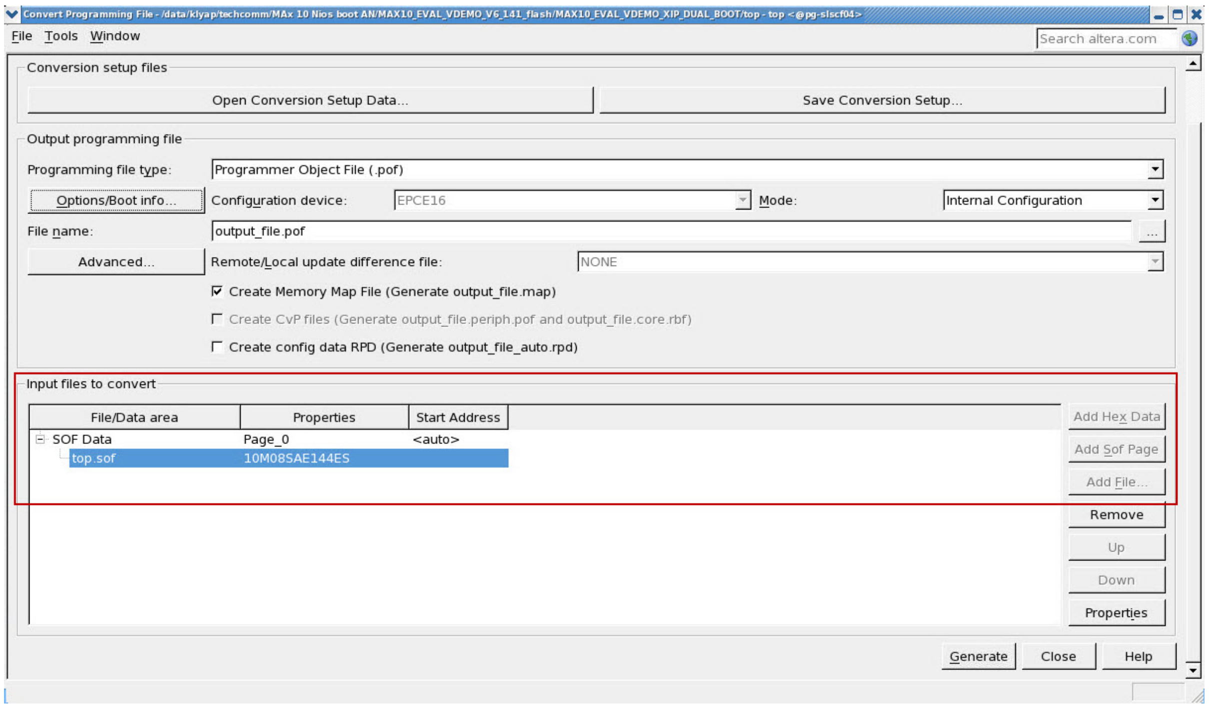
User Flash Memory

UFM source: Load memory file

File path: ash/mem_init/backup/MAX10_FLASH_163.hex

6. 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.

Figure 23: Input Files to Convert in Convert Programming Files


Convert Programming File - /data/klyap/techcomm/MAx 10 Nios boot AN/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO_XIP_DUAL_BOOT/top - top <@pg-slscf04>

Conversion setup files

Output programming file

Programming file type: Programmer Object File (.pof)

Configuration device: EPCE16

Mode: Internal Configuration

File name: output_file.pof

Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)

☐ Create CvP files (Generate output_file.periph.pof and output_file.core.rbf)

☐ Create config data RPD (Generate output_file_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>
top.sof	10M08SAE144ES	

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

Dual Compressed Images Bootable System Guideline

Qsys Settings

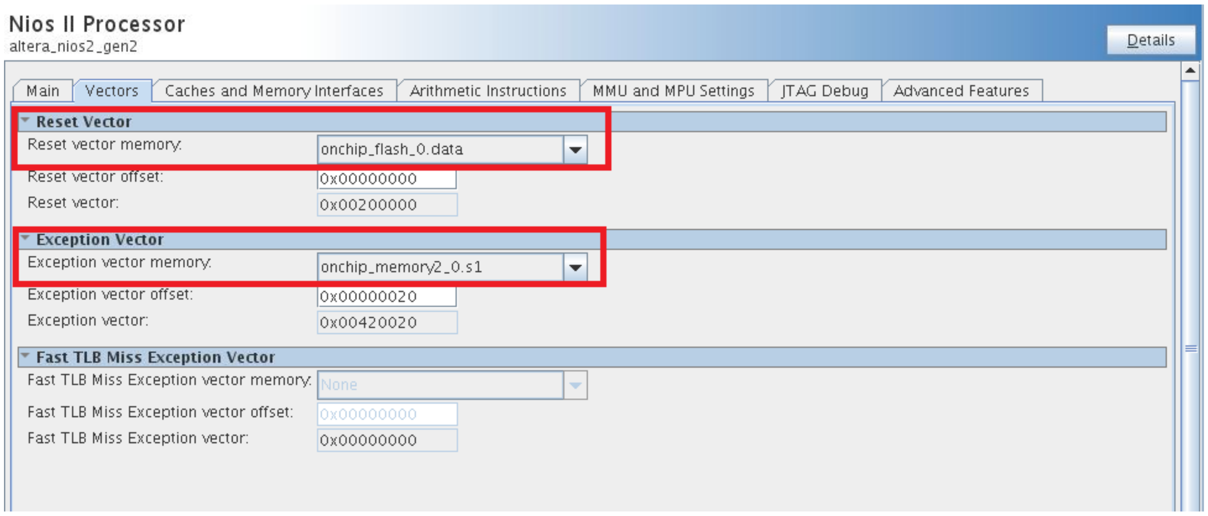
1. In the Nios II Processor parameter editor, set the reset vector memory and exception vector memory based on the boot options below:

Boot Option	Reset vector memory:	Exception vector memory:
Option 1a[9][10]	Altera On-chip Flash	OCRAM/ External RAM
Option 1b[9]	Altera On-chip Flash	Altera On-chip Flash
Option 2	Altera On-chip Flash	OCRAM/ External RAM

[9] You can set the exception vector for Boot Option 1 to OCRAM or External RAM (Option 1a) or Altera On-chip Flash (option 1b) according to your design preference.

[10] Boot option 1a which sets exception vector memory to OCRAM or External RAM is recommended to make the interrupt processing faster.

Figure 24: Nios II Parameter Editor Settings Boot Option 1a and 2

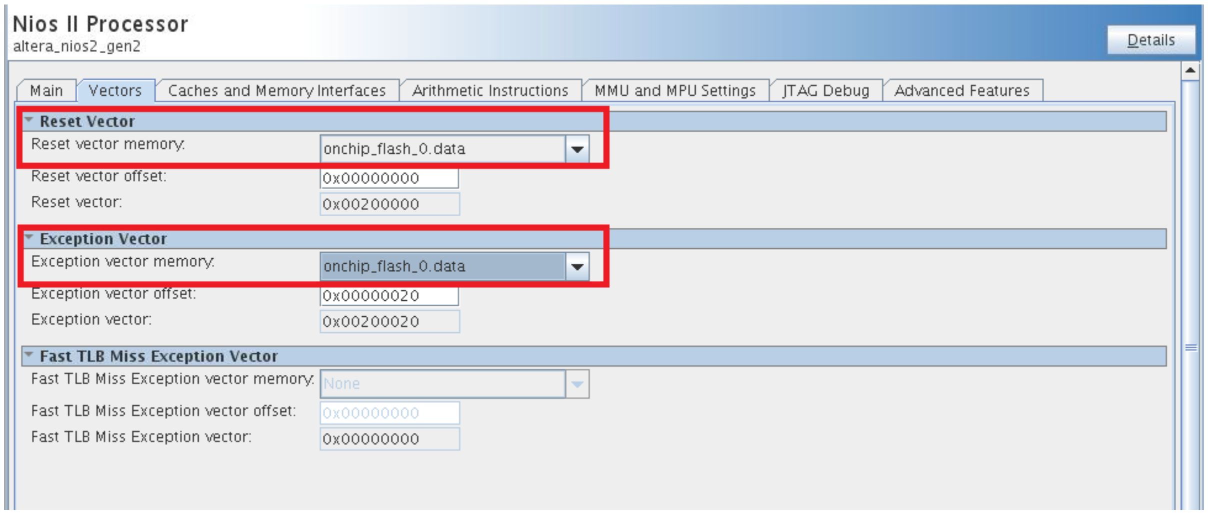


Reset Vector
Reset vector memory: onchip_flash_0.data
Reset vector offset: 0x00000000
Reset vector: 0x00200000

Exception Vector
Exception vector memory: onchip_memory2_0.s1
Exception vector offset: 0x00000020
Exception vector: 0x00420020

Fast TLB Miss Exception Vector
Fast TLB Miss Exception vector memory: None
Fast TLB Miss Exception vector offset: 0x00000000
Fast TLB Miss Exception vector: 0x00000000

Figure 25: Nios II Parameter Editor Settings Boot Option 1b



Reset Vector
Reset vector memory: onchip_flash_0.data
Reset vector offset: 0x00000000
Reset vector: 0x00200000

Exception Vector
Exception vector memory: onchip_flash_0.data
Exception vector offset: 0x00000020
Exception vector: 0x00200020

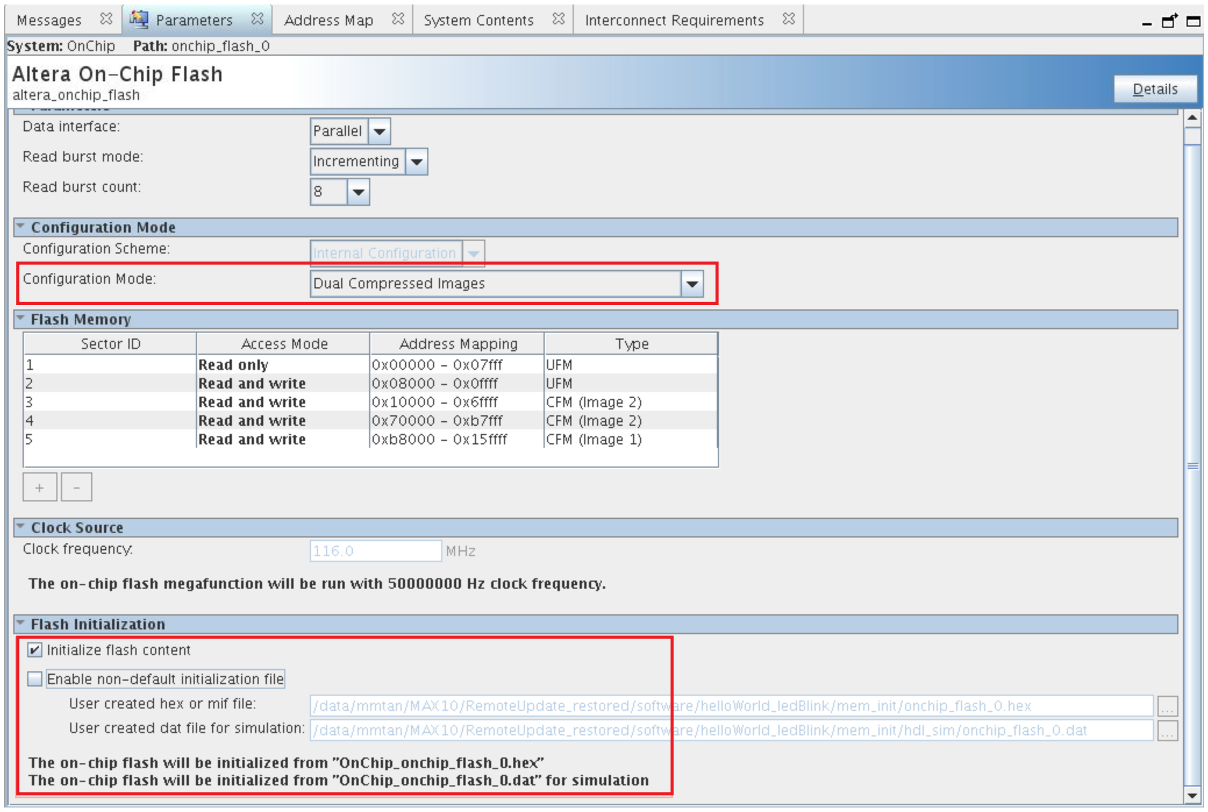
Fast TLB Miss Exception Vector
Fast TLB Miss Exception vector memory: None
Fast TLB Miss Exception vector offset: 0x00000000
Fast TLB Miss Exception vector: 0x00000000

2. In the Altera On-chip Flash IP parameter editor, set the **Configuration Mode** to **Dual Compressed Images**.
3. Refer to the following table for the options to program UFM data (HEX file) and settings required in Altera On-chip Flash IP.

Options to program UFM data	Method	Settings in Altera On-Chip Flash IP
Option 1: Initialize the UFM data in the SOF	Quartus II includes the UFM initialization data in the SOF during compilation. SOF recompilation is needed if there are changes in the UFM data.	**1.** Check **Initialize flash content** **2.** If default path is used, add **meminit.qip** generated during "make mem_init_ generate" into Quartus II project. Refer to Figure 27. Make sure the generated HEX naming matches the default naming. **3.** If non-default path is selected, enable the **Enable non-default initialization file** and specify the path of the HEX file. **Note:** For more information about Steps 2 and 3, refer to HEX File Generation section.
Option 2: Combine UFM data with a compiled SOF during programming files (POF) conversion [11]	UFM data is combined with the compiled SOF during the programming files conversion. SOF recompilation is NOT needed even if there are changes in the UFM data.	Uncheck **Initialize flash content**

[11] This is the recommended method for application developer. You are not required to recompile SOF file for application changes during development.

Figure 26: Dual Compressed Images Mode with initialize flash content Turned-on



System: OnChip **Path:** onchip_flash_0

Altera On-Chip Flash
altera_onchip_flash

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 only**	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: 116.0 MHz

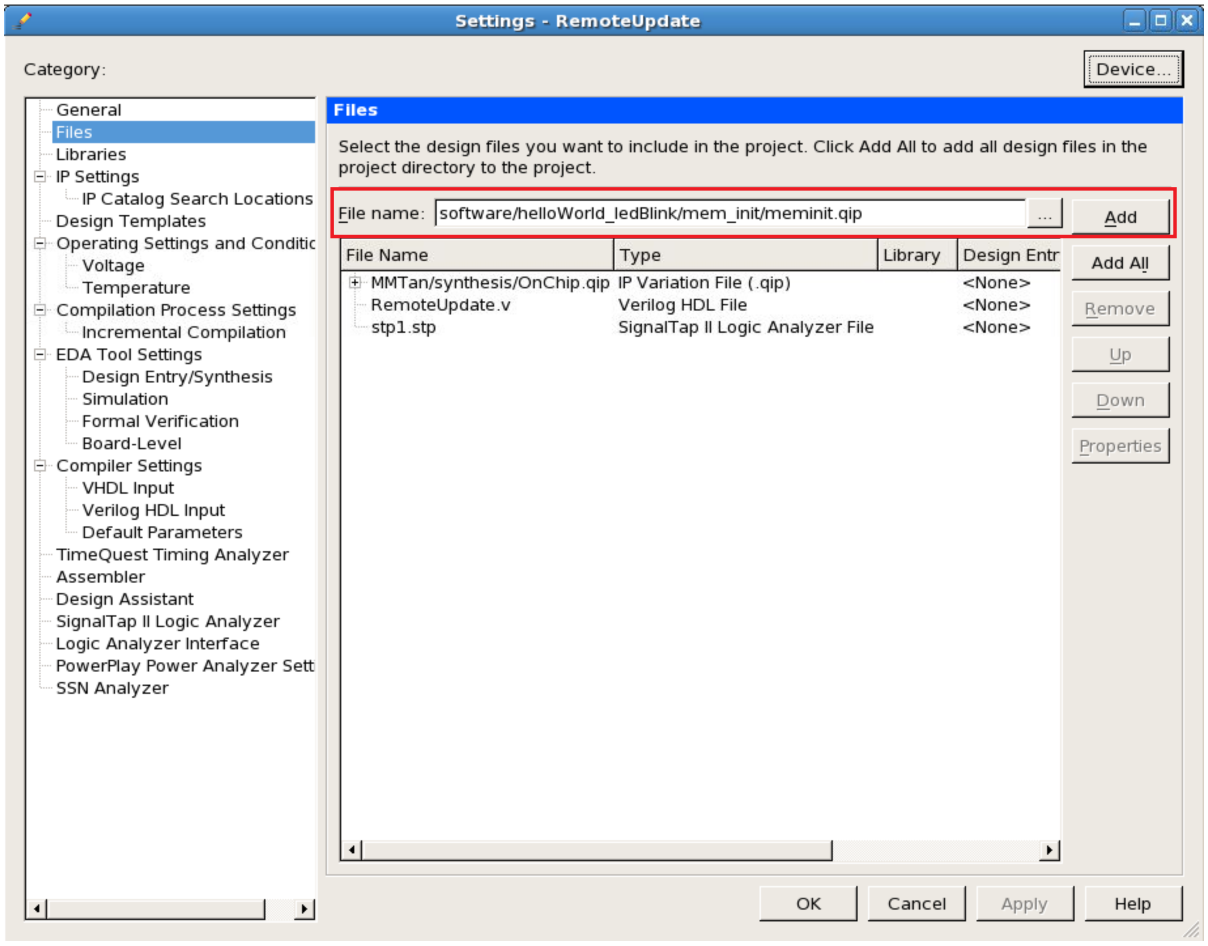
The on-chip flash megafunction will be run with 50000000 Hz clock frequency.

Flash Initialization

☑ Initialize flash content
☐ Enable non-default initialization file
User created hex or mif file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex
User created dat file for simulation: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.dat

The on-chip flash will be initialized from "OnChip_onchip_flash_0.hex"
The on-chip flash will be initialized from "OnChip_onchip_flash_0.dat" for simulation

Figure 27: Adding **meminit.qip** File in Quartus II

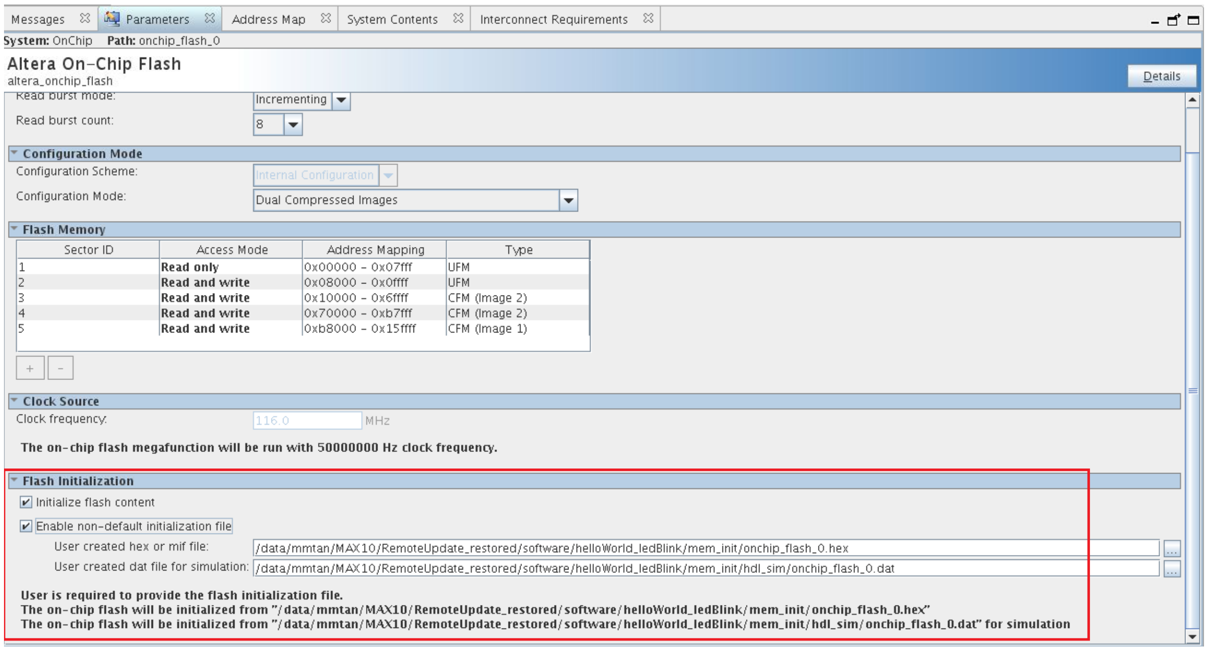


Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name: software/helloWorld_ledBlink/mem_init/meminit.qip

File Name	Type	Library	Design Entr
MMTan/synthesis/OnChip.qip	IP Variation File (.qip)		<None>
RemoteUpdate.v	Verilog HDL File		<None>
stp1.stp	SignalTap II Logic Analyzer File		<None>

Figure 28: Dual compressed Images with Non-default Initialization File Enabled



Messages | Parameters | Address Map | System Contents | Interconnect Requirements

System: OnChip **Path:** onchip_flash_0

Altera On-Chip Flash
altera_onchip_flash

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 only**	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: 116.0 MHz

The on-chip flash megafunction will be run with 50000000 Hz clock frequency.

Flash Initialization

☑ Initialize flash content

☑ Enable non-default initialization file

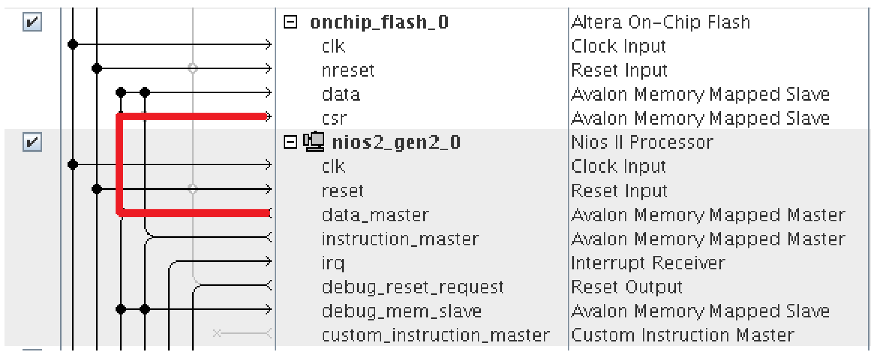
User created hex or mif file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex

User created dat file for simulation: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.dat

User is required to provide the flash initialization file.
The on-chip flash will be initialized from "/data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex"
The on-chip flash will be initialized from "/data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.dat" for simulation

4. Ensure that the Altera On-chip Flash `csr` port is connected to the Nios II processor `data_master` to enable write and erase operations.

Figure 29: CSR Connection to Nios II `data_master`



onchip_flash_0	Altera On-Chip Flash
clk	Clock Input
nreset	Reset Input
data	Avalon Memory Mapped Slave
csr	Avalon Memory Mapped Slave
nios2_gen2_0	Nios II Processor
clk	Clock Input
reset	Reset Input
data_master	Avalon Memory Mapped Master
instruction_master	Avalon Memory Mapped Master
irq	Interrupt Receiver
debug_reset_request	Reset Output
debug_mem_slave	Avalon Memory Mapped Slave
custom_instruction_master	Custom Instruction Master

5. Ensure the Altera Dual Configuration IP is instantiated in Qsys to enable dual images configuration.
6. Click **Generate HDL**, the **Generation** dialog box appears.
7. Specify output file generation options, and then click **Generate**.

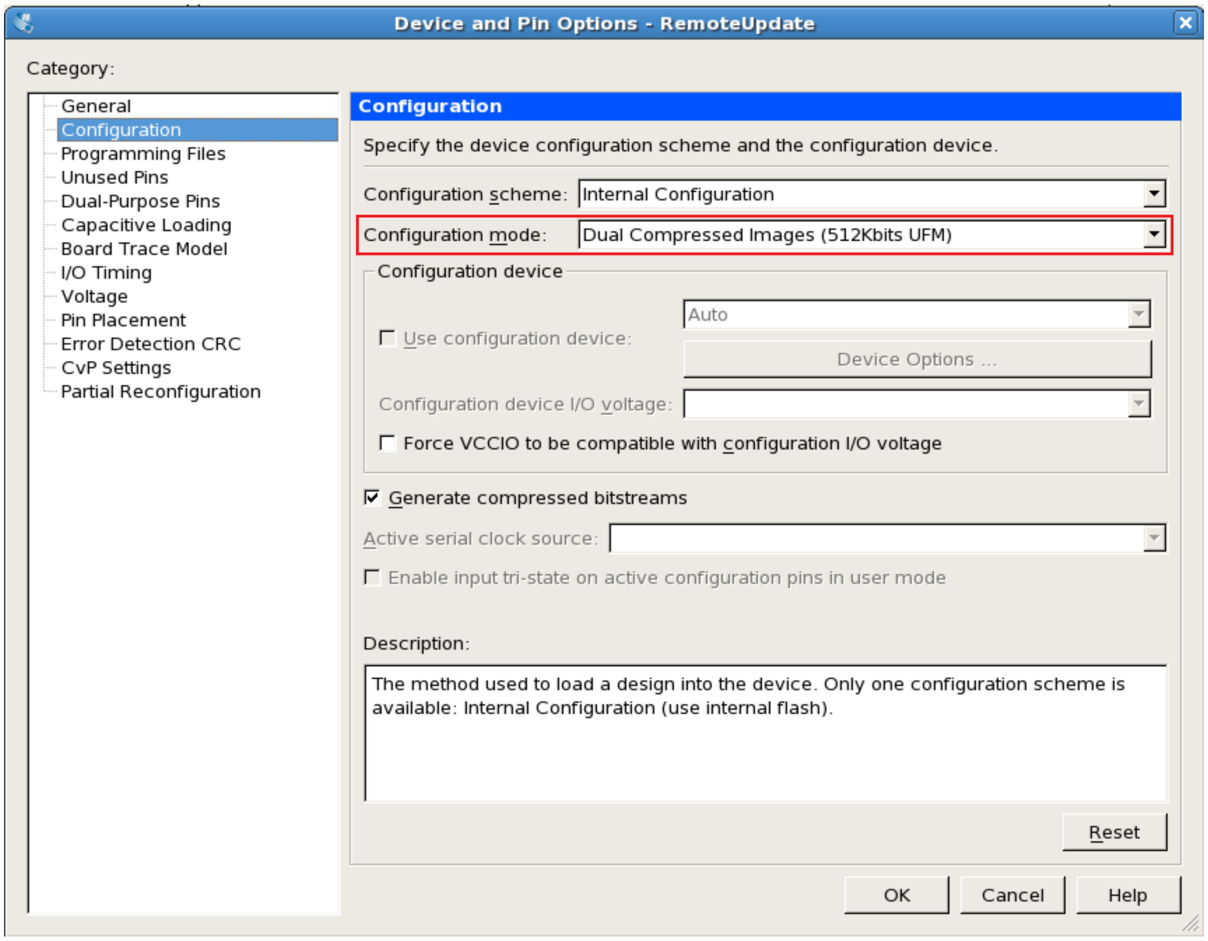
Related Information

HEX File Generation on page 36

Quartus II Software Settings

1. In the Quartus II software, click on **Assignment** -> **Device** -> **Device and Pin Options** -> **Configuration**. Set **Configuration mode** to **Dual Compressed Images**.(12)

Figure 30: Dual Compressed Images Configuration Mode in Quartus II



Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Dual Compressed Images (512Kbits UFM)

Configuration device

☐ Use configuration device: Auto

Configuration device I/O voltage:

☐ Force VCCIO to be compatible with configuration I/O voltage

☑ Generate compressed bitstreams

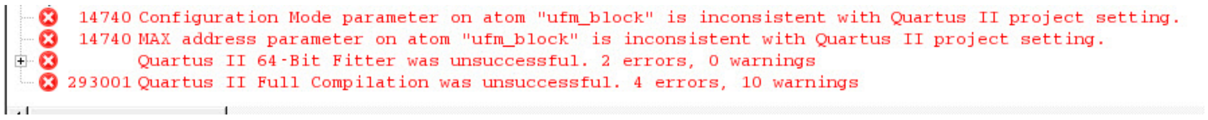
Active serial clock source:

☐ Enable input tri-state on active configuration pins in user mode

Description:

The method used to load a design into the device. Only one configuration scheme is available: Internal Configuration (use internal flash).

Note: 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.



```
14740 Configuration Mode parameter on atom "ufm_block" is inconsistent with Quartus II project setting.
14740 MAX address parameter on atom "ufm_block" is inconsistent with Quartus II project setting.
      Quartus II 64-Bit Fitter was unsuccessful. 2 errors, 0 warnings
293001 Quartus II Full Compilation was unsuccessful. 4 errors, 10 warnings
```

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.

Related Information

HEX File Generation on page 36

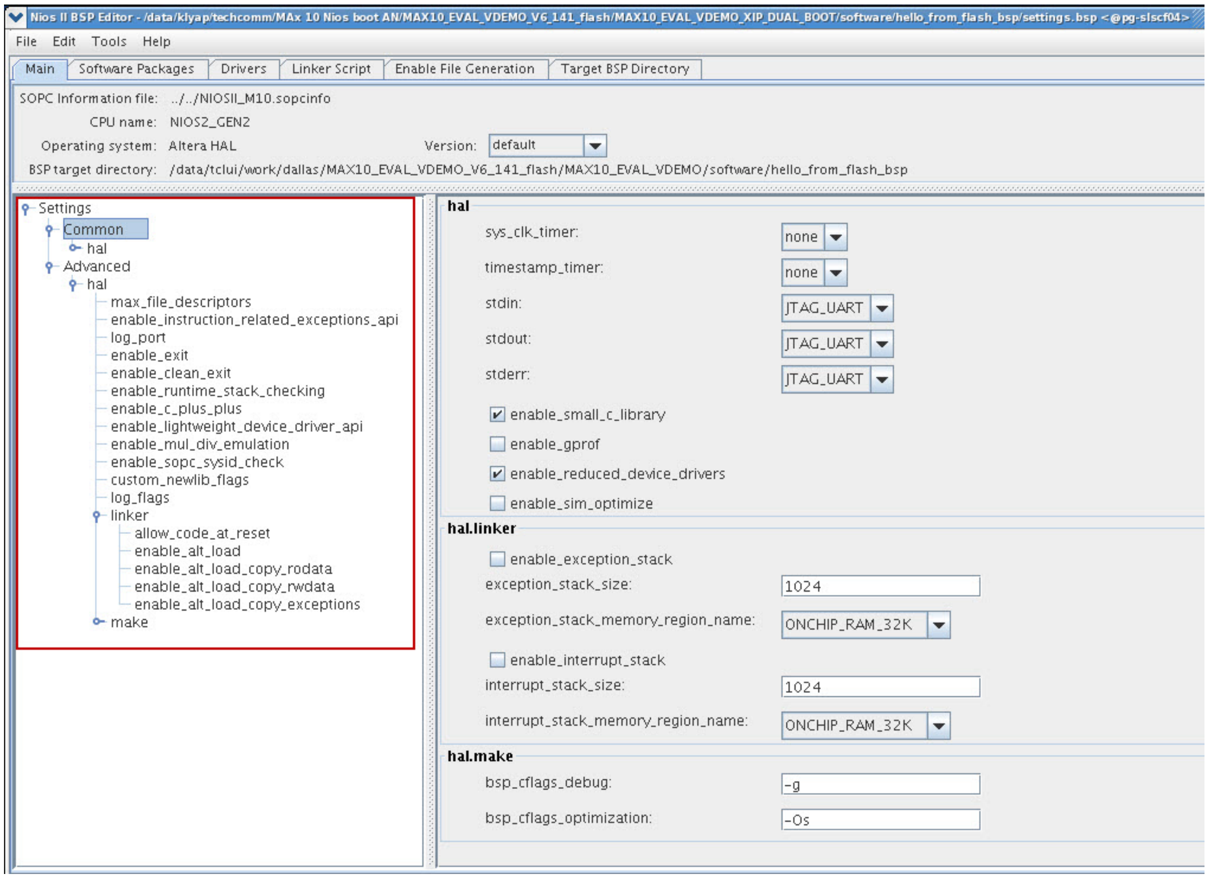
(12) The size of UFM sector will vary according to your device selection.

BSP Editor Settings

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

1. In the Nios II SBT tool, right click on your BSP project in the **Project Explorer** window. Select **Nios II** > **BSP Editor...** to open the **Nios II BSP Editor**.
2. In Nios II BSP Editor, click on **Advanced** tab under **Settings**.
3. Click on **hal** to expand the list.
4. Click on **linker** to expand the list.

Figure 31: BSP Editor Settings


Nios II BSP Editor - /data/klyap/techcomm/MAx 10 Nios boot AN/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO_XIP_DUAL_BOOT/software/hello_from_flash_bsp/settings.bsp <@pg-slscf04>
File
Edit
Tools
Help
Main
Software Packages
Drivers
Linker Script
Enable File Generation
Target BSP Directory

SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL
Version: default
BSP target directory: /data/tclui/work/dallas/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO/software/hello_from_flash_bsp

```
Settings
  Common
    hal
  Advanced
    hal
      max_file_descriptors
      enable_instruction_related_exceptions_api
      log_port
      enable_exit
      enable_clean_exit
      enable_runtime_stack_checking
      enable_c_plus_plus
      enable_lightweight_device_driver_api
      enable_mul_div_emulation
      enable_sopc_sysid_check
      custom_newlib_flags
      log_flags
      linker
        allow_code_at_reset
        enable_alt_load
        enable_alt_load_copy_rodata
        enable_alt_load_copy_rwdata
        enable_alt_load_copy_exceptions
      make
```

hal

sys_clk_timer:	none
timestamp_timer:	none
stdin:	JTAG_UART
stdout:	JTAG_UART
stderr:	JTAG_UART

[✓] enable_small_c_library
[ ] enable_gprof
[✓] enable_reduced_device_drivers
[ ] enable_sim_optimize

hal.linker

[ ] enable_exception_stack

exception_stack_size:	1024
exception_stack_memory_region_name:	ONCHIP_RAM_32K

[ ] enable_interrupt_stack

interrupt_stack_size:	1024
interrupt_stack_memory_region_name:	ONCHIP_RAM_32K

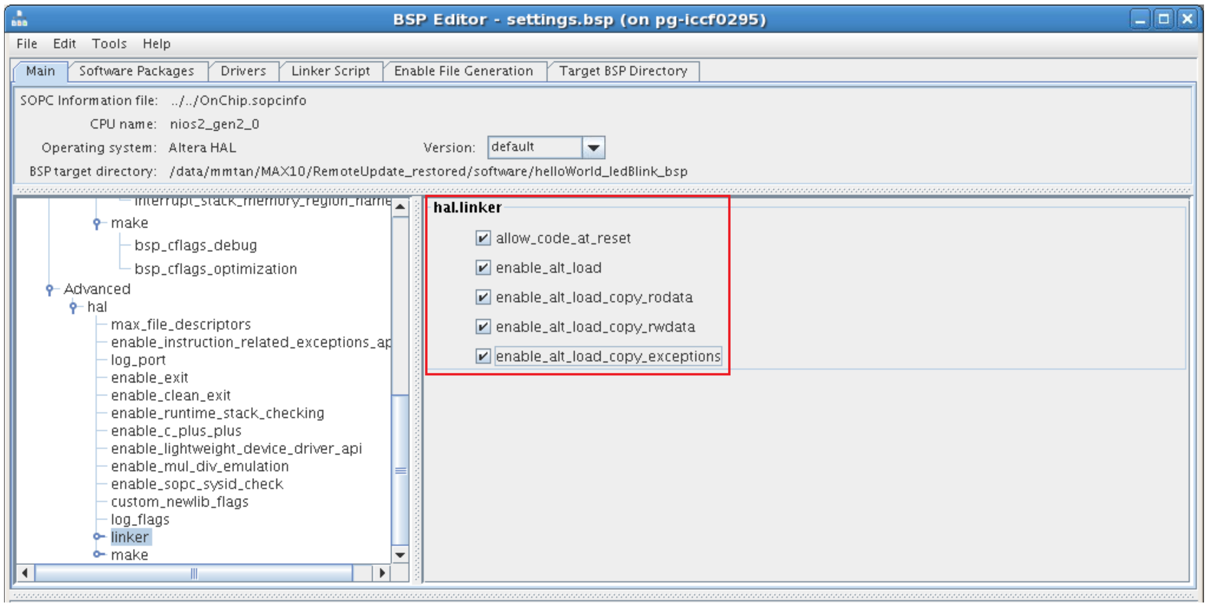
hal.make

bsp_cflags_debug:	-g
bsp_cflags_optimization:	-Os

5. Based on the boot option used, do one of the following:

- For boot option 1a, if exception vector memory is set to OCRAM or External RAM, enable the following:
  - **allow_code_at_reset**
  - **enable_alt_load**
  - **enable_alt_load_copy_rodata**
  - **enable_alt_load_copy_rwdata**
  - **enable_alt_load_copy_exceptions**
- For boot option 1b, if exception vector memory is set to Altera On-chip Flash, enable the following:
  - **allow_code_at_reset**
  - **enable_alt_load**
  - **enable_alt_load_copy_rodata**
  - **enable_alt_load_copy_rwdata**
- For boot option 2, leave all the hal.linker settings unchecked.

Figure 32: Advanced.hal.linker Settings for Boot Option 1a

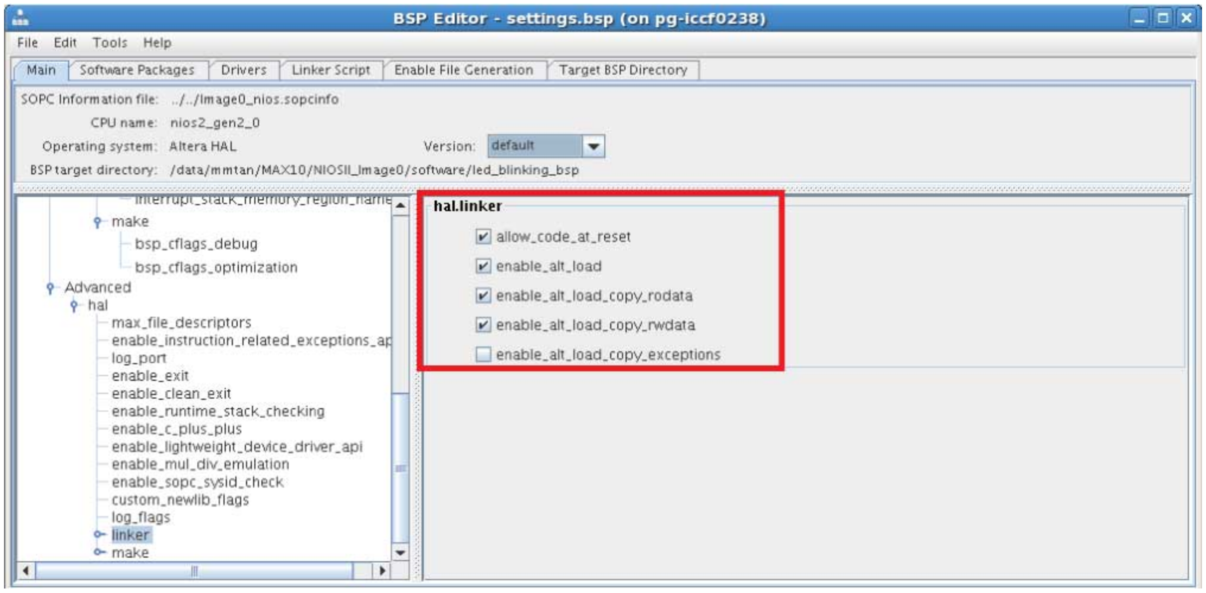


SOPC Information file: ../../OnChip.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL
Version: default
BSP target directory: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink_bsp

hal.linker

- ✓ allow_code_at_reset
- ✓ enable_alt_load
- ✓ enable_alt_load_copy_rodata
- ✓ enable_alt_load_copy_rwdata
- ✓ enable_alt_load_copy_exceptions

Figure 33: Advanced.hal.linker Settings for Boot Option 1b

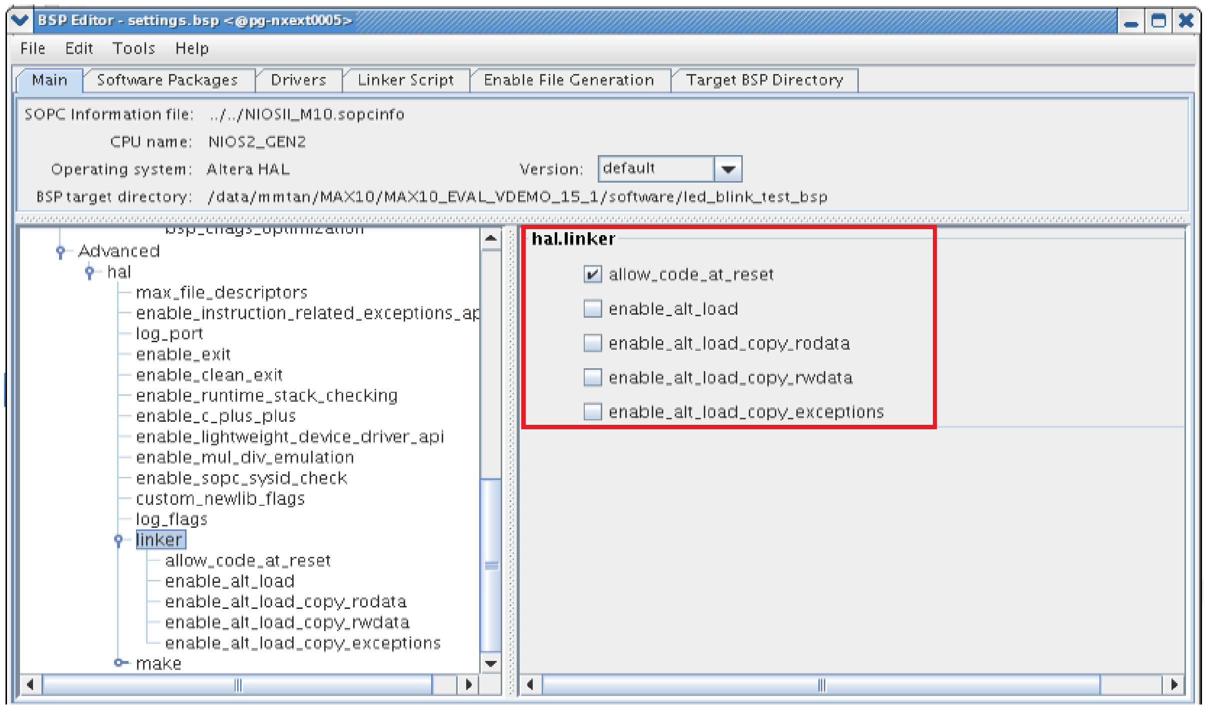

BSP Editor - settings.bsp (on pg-iccf0238)
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory

SOPC Information file: ../../Image0_nios.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL Version: default
BSP target directory: /data/mmtan/MAX10/NIOSII_Image0/software/led_blinking_bsp

```
        interrupt_stack_memory_region_name
      make
        bsp_cflags_debug
        bsp_cflags_optimization
  Advanced
    hal
      max_file_descriptors
      enable_instruction_related_exceptions_ap
      log_port
      enable_exit
      enable_clean_exit
      enable_runtime_stack_checking
      enable_c_plus_plus
      enable_lightweight_device_driver_api
      enable_mul_div_emulation
      enable_sopc_sysid_check
      custom_newlib_flags
      log_flags
      linker
      make
```

hal.linker
- [✓] allow_code_at_reset
- [✓] enable_alt_load
- [✓] enable_alt_load_copy_rodata
- [✓] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

Figure 34: Advanced.hal.linker Settings for Boot Option 2


BSP Editor - settings.bsp <@pg-nxext0005>
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory

SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL Version: default
BSP target directory: /data/mmtan/MAX10/MAX10_EVAL_VDEMO_15_1/software/led_blink_test_bsp

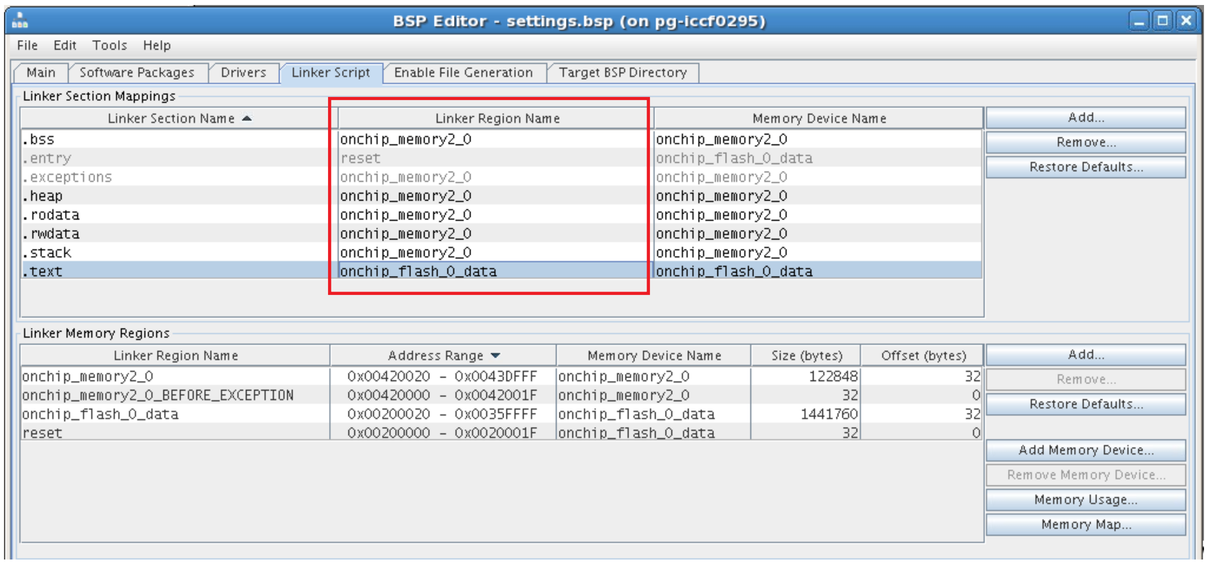
```
        bsp_cflags_optimization
  Advanced
    hal
      max_file_descriptors
      enable_instruction_related_exceptions_ap
      log_port
      enable_exit
      enable_clean_exit
      enable_runtime_stack_checking
      enable_c_plus_plus
      enable_lightweight_device_driver_api
      enable_mul_div_emulation
      enable_sopc_sysid_check
      custom_newlib_flags
      log_flags
      linker
        allow_code_at_reset
        enable_alt_load
        enable_alt_load_copy_rodata
        enable_alt_load_copy_rwdata
        enable_alt_load_copy_exceptions
      make
```

hal.linker
- [✓] allow_code_at_reset
- [ ] enable_alt_load
- [ ] enable_alt_load_copy_rodata
- [ ] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

6. Click on **Linker Script** tab in the Nios II BSP Editor.
7. Based on the boot option used, do one of the following:

- For boot option 1a and 1b, 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 Memory (OCRAM) or external RAM.
- For boot option 2, set all of the items in the **Linker Section Name** list to Altera On-chip Memory (OCRAM) or external RAM.

Figure 35: Linker Region Settings for Boot Option 1a and 1b


BSP Editor - settings.bsp (on pg-iccf0295)
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Add... Remove... Restore Defaults...
Add... Remove... Restore Defaults... Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...

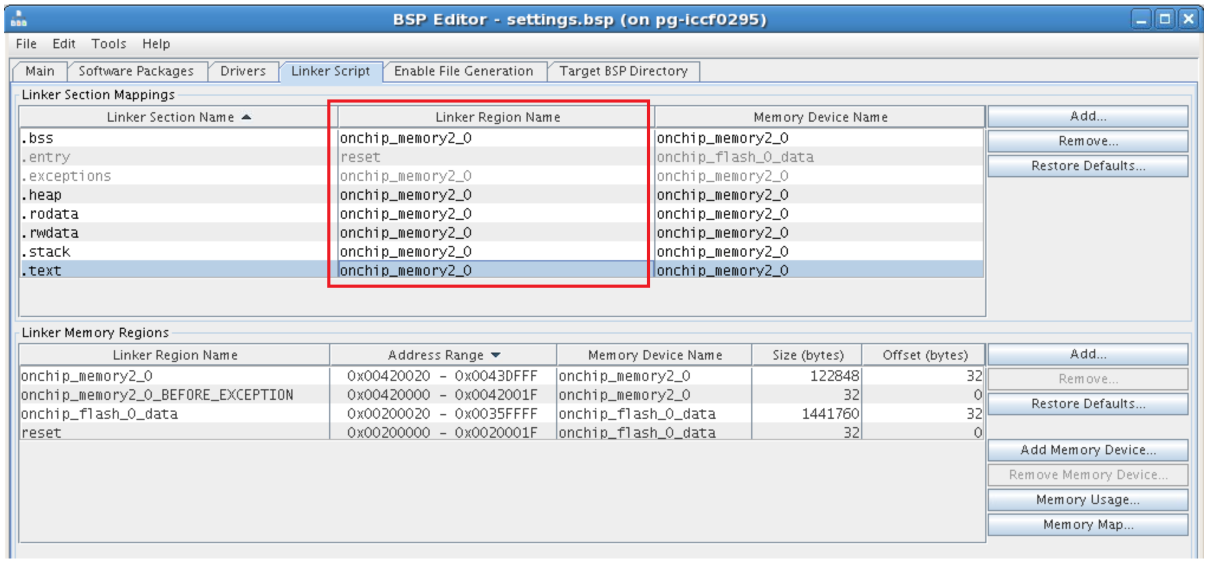
Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	onchip_memory2_0	onchip_memory2_0
.entry	reset	onchip_flash_0_data
.exceptions	onchip_memory2_0	onchip_memory2_0
.heap	onchip_memory2_0	onchip_memory2_0
.rodata	onchip_memory2_0	onchip_memory2_0
.rwdata	onchip_memory2_0	onchip_memory2_0
.stack	onchip_memory2_0	onchip_memory2_0
.text	onchip_flash_0_data	onchip_flash_0_data

Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_memory2_0	0x00420020 - 0x0043DFFF	onchip_memory2_0	122848	32
onchip_memory2_0_BEFORE_EXCEPTION	0x00420000 - 0x0042001F	onchip_memory2_0	32	0
onchip_flash_0_data	0x00200020 - 0x0035FFFF	onchip_flash_0_data	1441760	32
reset	0x00200000 - 0x0020001F	onchip_flash_0_data	32	0

Figure 36: Linker Region Settings for Boot Option 2


BSP Editor - settings.bsp (on pg-iccf0295)
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Add... Remove... Restore Defaults...
Add... Remove... Restore Defaults... Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...

Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	onchip_memory2_0	onchip_memory2_0
.entry	reset	onchip_flash_0_data
.exceptions	onchip_memory2_0	onchip_memory2_0
.heap	onchip_memory2_0	onchip_memory2_0
.rodata	onchip_memory2_0	onchip_memory2_0
.rwdata	onchip_memory2_0	onchip_memory2_0
.stack	onchip_memory2_0	onchip_memory2_0
.text	onchip_memory2_0	onchip_memory2_0

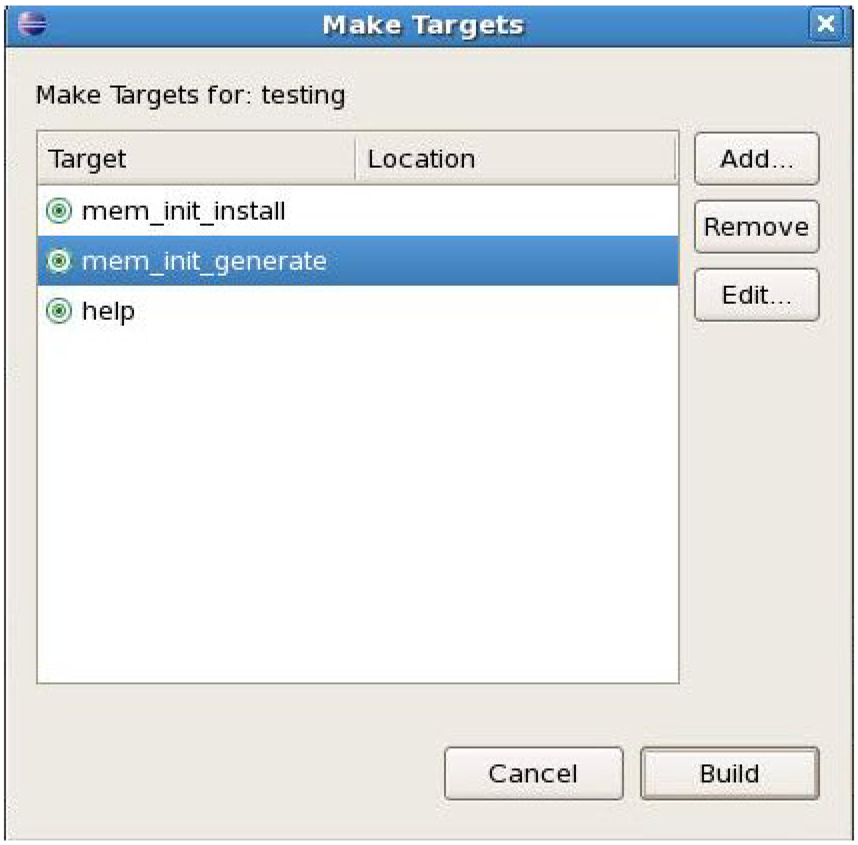
Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_memory2_0	0x00420020 - 0x0043DFFF	onchip_memory2_0	122848	32
onchip_memory2_0_BEFORE_EXCEPTION	0x00420000 - 0x0042001F	onchip_memory2_0	32	0
onchip_flash_0_data	0x00200020 - 0x0035FFFF	onchip_flash_0_data	1441760	32
reset	0x00200000 - 0x0020001F	onchip_flash_0_data	32	0

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.

Figure 37: Selecting mem_init_generate in Make Targets



5. The "mem_init_generate" macro will create two HEX files; **<on_chip_ram.hex>** and **<on_chip_flash.hex>**. The **<on_chip_ram.hex>** will be used for boot option 3 and **<on_chip_flash.hex>** is used for boot option 1 and 2.

   Note:
   - The **mem_init_generate** target also generates a Quartus II IP file (**meminit.qip**). Quartus II software will refer to the **meminit.qip** for the location of the initialization files.
   - All these files can be found under "<project_folder>/software/<application_name>/mem_init" folder.

6. Recompile your project in Quartus II software if you check **Initialize memory content** option in Altera On-Chip Flash IP. This is to include the software data (.HEX) into the SOF file.

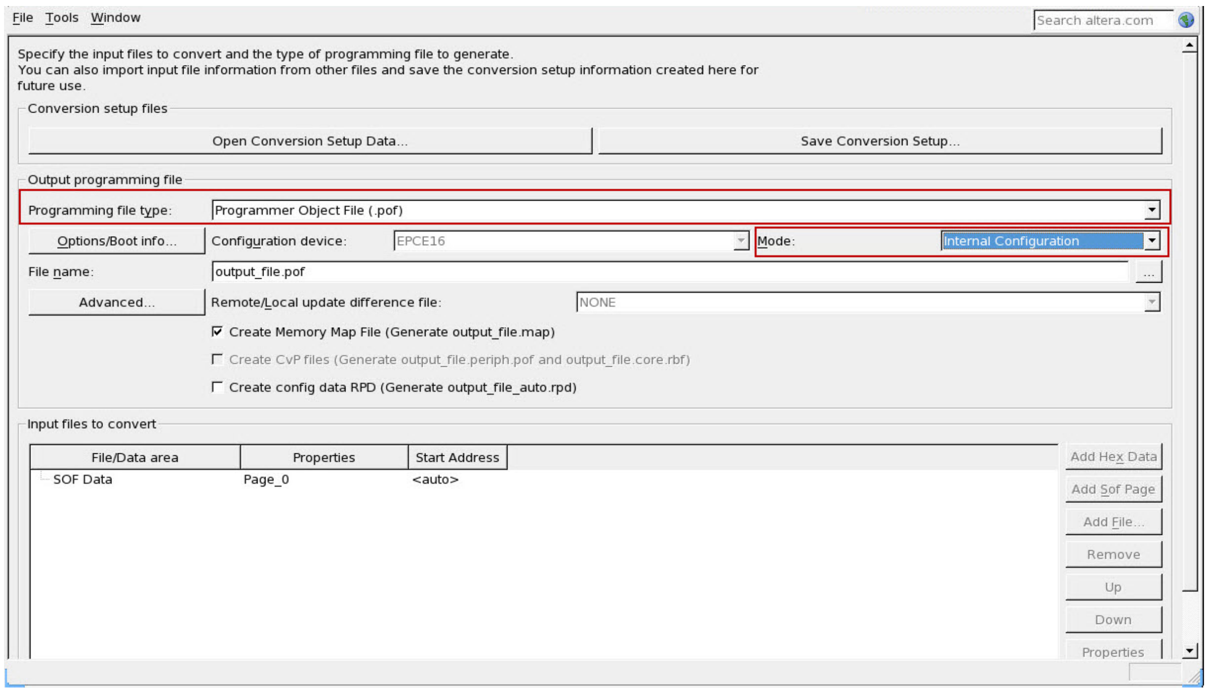
Related Information

- Qsys Settings on page 25
- Quartus II Software Settings on page 30

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**.

Figure 38: Convert Programming File Settings



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_file.pof

Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)

☐ Create CvP files (Generate output_file.periph.pof and output_file.core.rbf)

☐ Create config data RPD (Generate output_file_auto.rpd)

Input files to convert

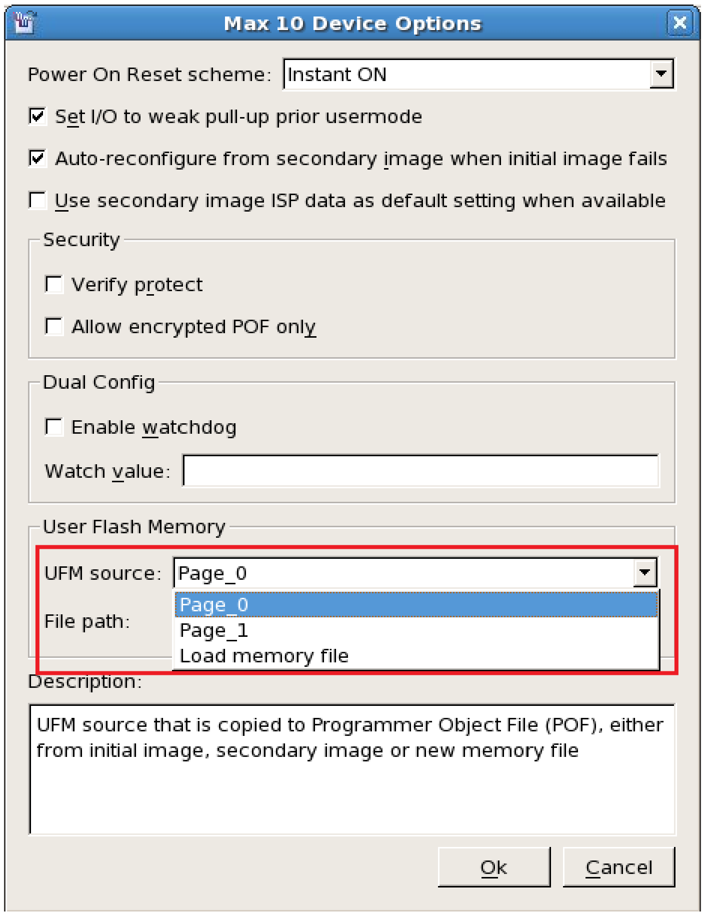
File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>

4. Click on **Options/Boot info...**, the MAX 10 Device Options dialog box appears.
5. Based on the **Initialize flash content** settings, do one of the following:

- If **Initialize flash content** was checked, make sure **Page_0** or **Page_1** is selected for **UFM source:** option. Click OK.

  Note: UFM data (.HEX file) can be included in either Page_0 or Page_1 only. The Altera On-chip flash does not support two .HEX files for Dual Compressed images configuration mode.
- If **Initialize flash content** was not checked, choose **Load memory file** for **UFM source:** option. Browse to the generated Altera On-chip Flash HEX file (**on_chip_flash.hex**) in the **File path:** and click **OK**.

Figure 39: Setting Page_0 or Page_1 for UFM Source If Initialize flash content is Checked



Power On Reset scheme: Instant ON

[✓] Set I/O to weak pull-up prior usermode

[✓] Auto-reconfigure from secondary image when initial image fails

[ ] Use secondary image ISP data as default setting when available

[ ] Verify protect

[ ] Allow encrypted POF only

[ ] Enable watchdog

Watch value:

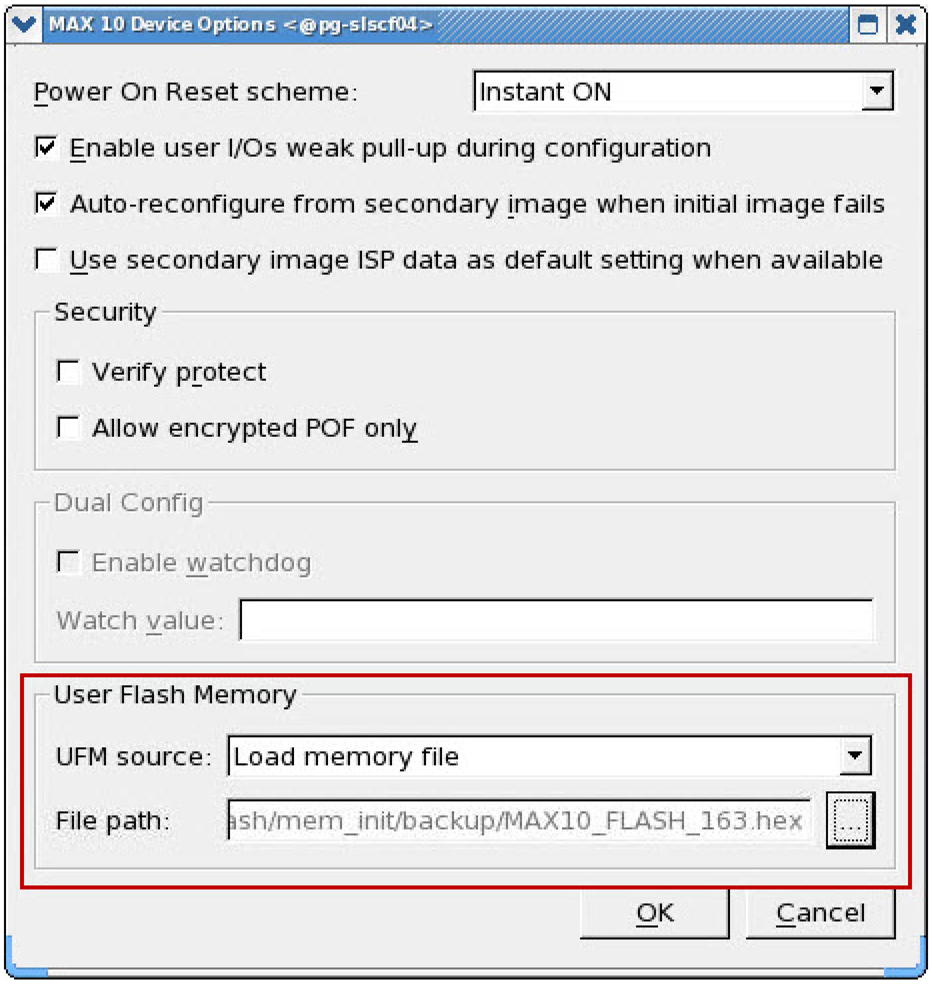
UFM source: Page_0

File path:

Description:

UFM source that is copied to Programmer Object File (POF), either from initial image, secondary image or new memory file

Figure 40: Setting Load Memory File for UFM Source if Initialize flash content is not Checked



Power On Reset scheme: Instant ON

☑ Enable user I/Os weak pull-up during configuration

☑ Auto-reconfigure from secondary image when initial image fails

☐ Use secondary image ISP data as default setting when available

Security

☐ Verify protect

☐ Allow encrypted POF only

Dual Config

☐ Enable watchdog

Watch value:

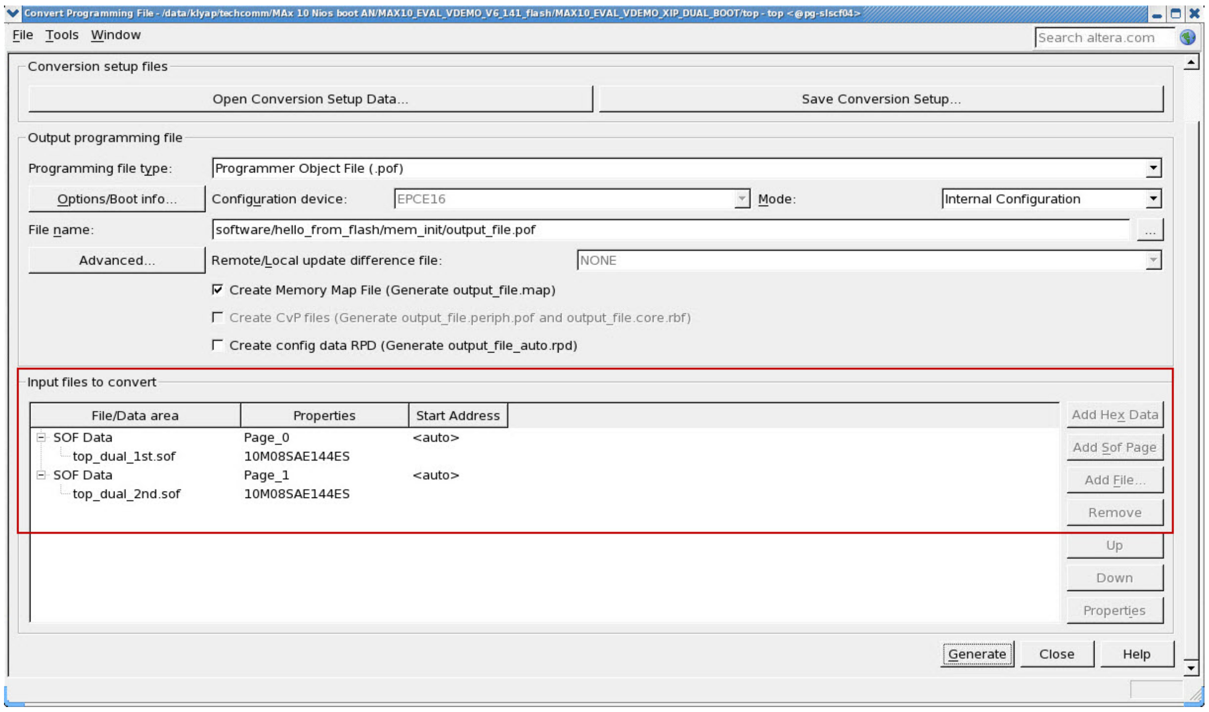
User Flash Memory

UFM source: Load memory file

File path: ash/mem_init/backup/MAX10_FLASH_163.hex

6. 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.
7. 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.

Figure 41: Input Files to Convert in Convert Programming Files for Dual Images Mode


Convert Programming File - /data/klyap/techcomm/MAx 10 Nios boot AN/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO_XIP_DUAL_BOOT/top - top <@pg-slscf04>

Conversion setup files

Output programming file

Programming file type: Programmer Object File (.pof)

Configuration device: EPCE16

Mode: Internal Configuration

File name: software/hello_from_flash/mem_init/output_file.pof

Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)

☐ Create CvP files (Generate output_file.periph.pof and output_file.core.rbf)

☐ Create config data RPD (Generate output_file_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>
top_dual_1st.sof	10M08SAE144ES	
SOF Data	Page_1	<auto>
top_dual_2nd.sof	10M08SAE144ES	

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

Boot Option 3: Nios II Processor Application Executes in-place from OCRAM

The on-chip memory is initialized during FPGA configuration with data from a Nios II application image. This data is built into the FPGA configuration bitstream, the programmer object file. This process eliminates the need for a boot copier, as the Nios II application is already in place at system reset.

This option will not work in any of the following situations:

- When you select a configuration mode that does not support ERAM initialization.
- After a soft reset where the memory contents have been modified by the application and the application code has been corrupted.

Figure 42: Boot Option 3 Block Diagram

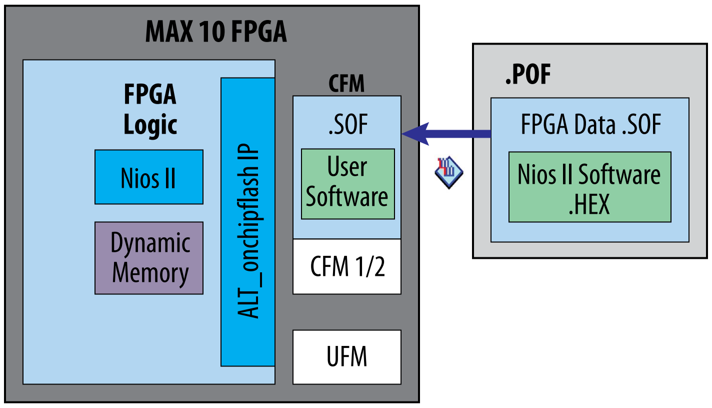
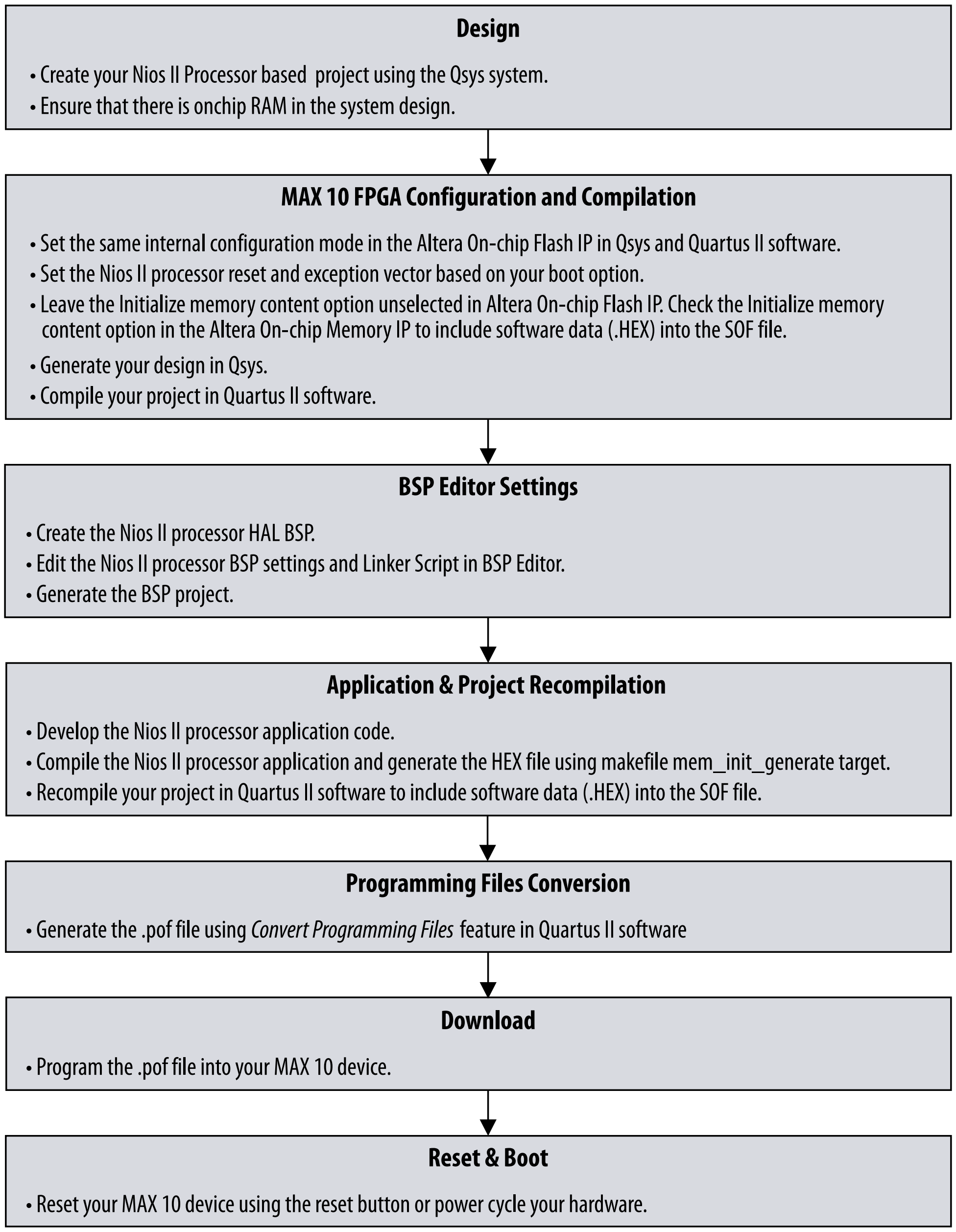


Table 8: RAM and ROM Size Requirement for Boot Option 3

RAM Size Requirement	ROM Size Requirement
Equivalent to the executable code and dynamic memory size required by user program.	Not applicable for this boot option.

Figure 43: Configuration and Booting Flow for Option 3

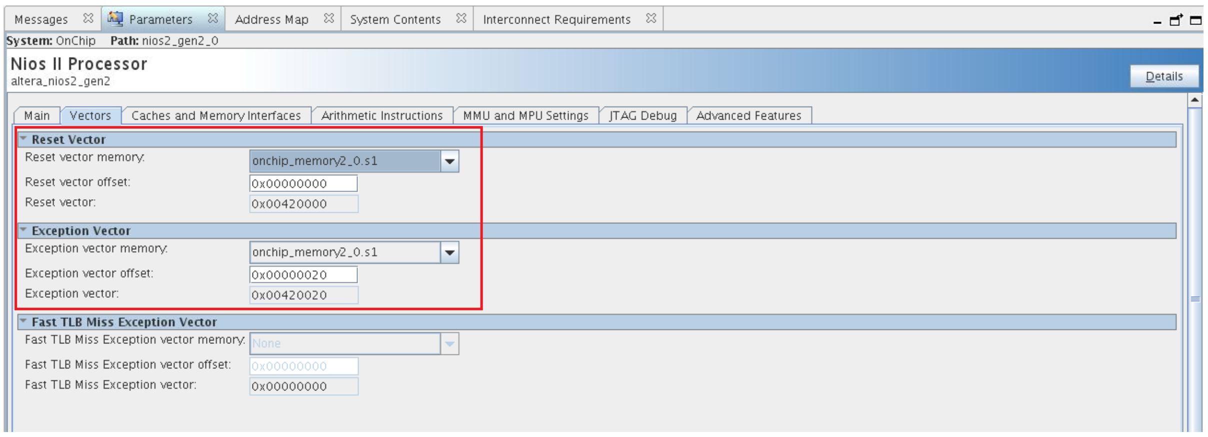

Design
• Create your Nios II Processor based project using the Qsys system.
• Ensure that there is onchip RAM in the system design.
MAX 10 FPGA Configuration and Compilation
• Set the same internal configuration mode in the Altera On-chip Flash IP in Qsys and Quartus II software.
• Set the Nios II processor reset and exception vector based on your boot option.
• Leave the Initialize memory content option unselected in Altera On-chip Flash IP. Check the Initialize memory content option in the Altera On-chip Memory IP to include software data (.HEX) into the SOF file.
• Generate your design in Qsys.
• Compile your project in Quartus II software.
BSP Editor Settings
• Create the Nios II processor HAL BSP.
• Edit the Nios II processor BSP settings and Linker Script in BSP Editor.
• Generate the BSP project.
Application & Project Recompilation
• Develop the Nios II processor application code.
• Compile the Nios II processor application and generate the HEX file using makefile mem_init_generate target.
• Recompile your project in Quartus II software to include software data (.HEX) into the SOF file.
Programming Files Conversion
• Generate the .pof file using Convert Programming Files feature in Quartus II software
Download
• Program the .pof file into your MAX 10 device.
Reset & Boot
• Reset your MAX 10 device using the reset button or power cycle your hardware.

Single Uncompressed/Compressed Image with Memory Initialization Guideline

Qsys Settings

1. In the Nios II Processor parameter editor, set both the **Reset vector memory:** and **Exception vector memory:** to Altera On-Chip Memory (OCRAM).

Figure 44: Nios II Parameter Editor Settings for Boot Option 3


Messages | Parameters | Address Map | System Contents | Interconnect Requirements

System: OnChip Path: nios2_gen2_0

Nios II Processor
altera_nios2_gen2

Main | Vectors | Caches and Memory Interfaces | Arithmetic Instructions | MMU and MPU Settings | JTAG Debug | Advanced Features

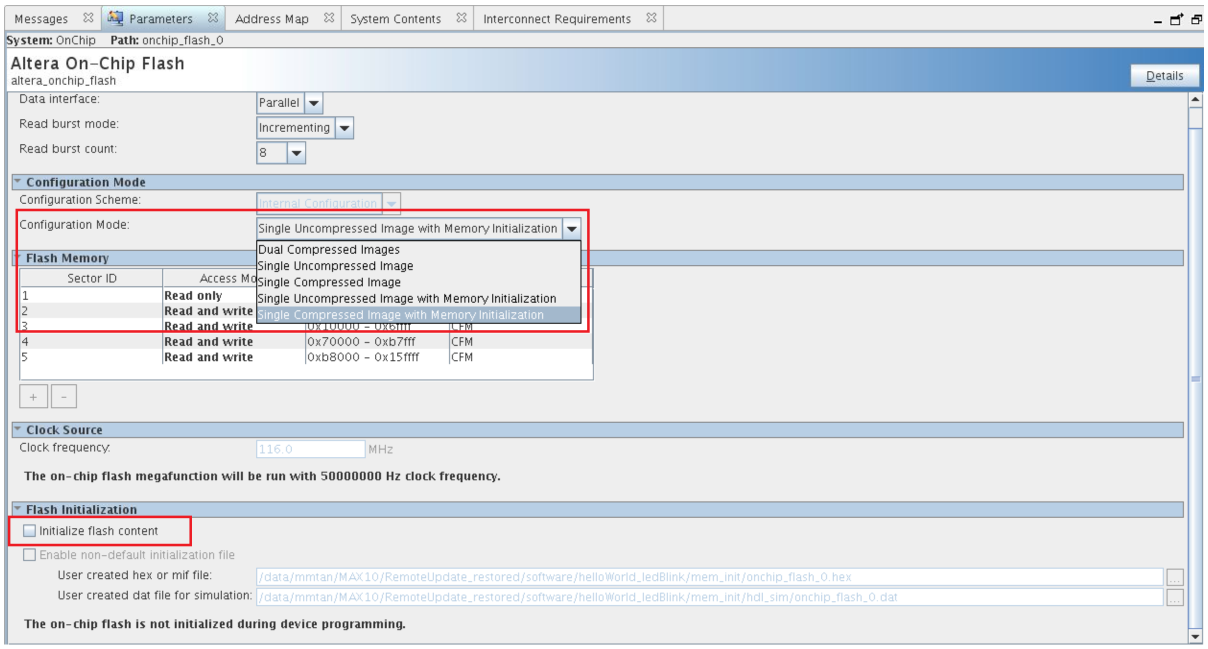
Reset Vector
Reset vector memory: onchip_memory2_0.s1
Reset vector offset: 0x00000000
Reset vector: 0x00420000

Exception Vector
Exception vector memory: onchip_memory2_0.s1
Exception vector offset: 0x00000020
Exception vector: 0x00420020

Fast TLB Miss Exception Vector
Fast TLB Miss Exception vector memory: None
Fast TLB Miss Exception vector offset: 0x00000000
Fast TLB Miss Exception vector: 0x00000000

2. In the Altera On-chip Flash IP parameter editor, set the **Configuration Mode** to **Single Uncompressed Image with Memory Initialization** or **Single Compressed Image with Memory Initialization**. Leave the **Initialize flash content** unchecked. This is because the Altera On-chip flash initialization data will not be enabled.

Figure 45: Configuration Mode with Memory Initialization Selection and Initialize Flash Content Setting


Messages | Parameters | Address Map | System Contents | Interconnect Requirements

System: OnChip Path: onchip_flash_0

Altera On-Chip Flash
altera_onchip_flash

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

Configuration Mode
Configuration Scheme: Internal Configuration
Configuration Mode: Single Uncompressed Image with Memory Initialization

Dual Compressed Images
Single Uncompressed Image
Single Compressed Image
Single Uncompressed Image with Memory Initialization
Single Compressed Image with Memory Initialization

Flash Memory

Sector ID	Access Mo		
1	Read only		
2	Read and write		
3	Read and write	0x10000 – 0x6ffff	CFM
4	Read and write	0x70000 – 0xb7fff	CFM
5	Read and write	0xb8000 – 0x15ffff	CFM

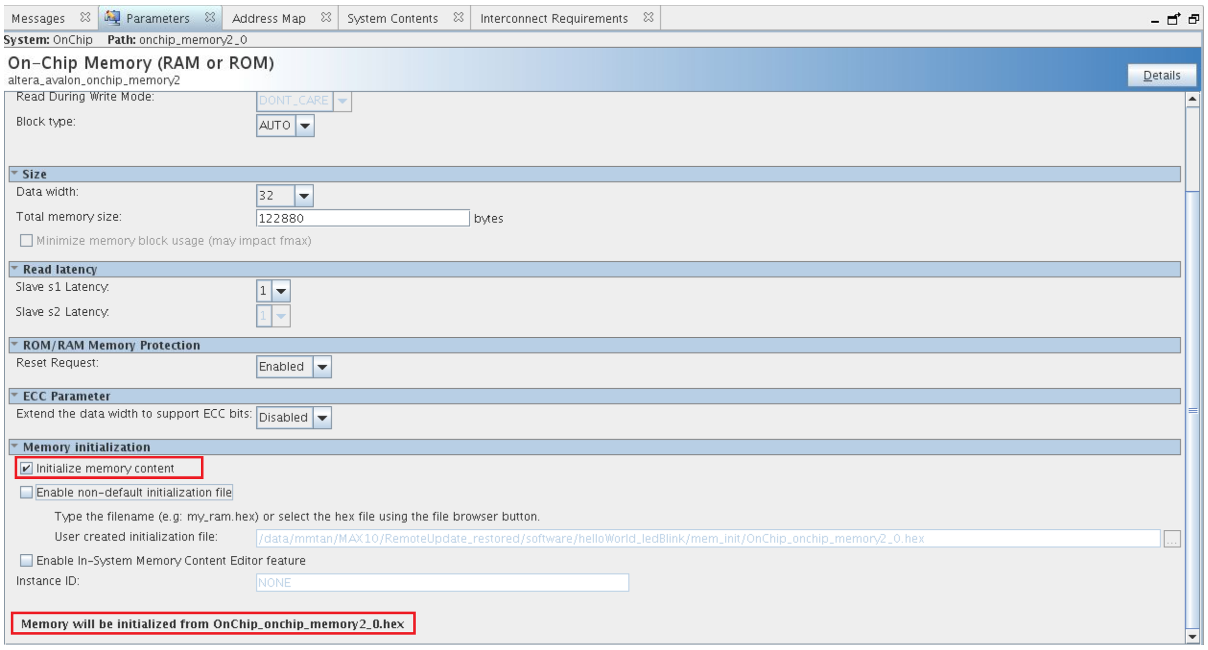
Clock Source
Clock frequency: 116.0 MHz
The on-chip flash megafunction will be run with 50000000 Hz clock frequency.

Flash Initialization
☐ Initialize flash content
☐ Enable non-default initialization file
User created hex or mif file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/onchip_flash_0.hex
User created dat file for simulation: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/hdl_sim/onchip_flash_0.dat
The on-chip flash is not initialized during device programming.

3. In the Altera On-chip Memory (RAM or ROM) IP parameter editor, check **Initialize flash content**. If default path is used, add **meminit.qip** generated during "make mem_init_generate" into Quartus II project. Refer to Figure 47. Make sure the generated HEX naming matches the default naming. If non-default path is selected, check **Enable non-default initialization file** and specify the path of the HEX file (**onchip_memory2_0.hex**).

Note: The **meminit.qip** stores the location information of the initialization files.

Figure 46: Enable Initialize Memory Content with Default Initialization File in On-Chip Memory Parameter Editor Settings



System: OnChip **Path:** onchip_memory2_0

On-Chip Memory (RAM or ROM)
altera_avalon_onchip_memory2

Read During Write Mode: DONT_CARE
Block type: AUTO

Size
Data width: 32
Total memory size: 122880 bytes
☐ Minimize memory block usage (may impact fmax)

Read latency
Slave s1 Latency: 1
Slave s2 Latency: 1

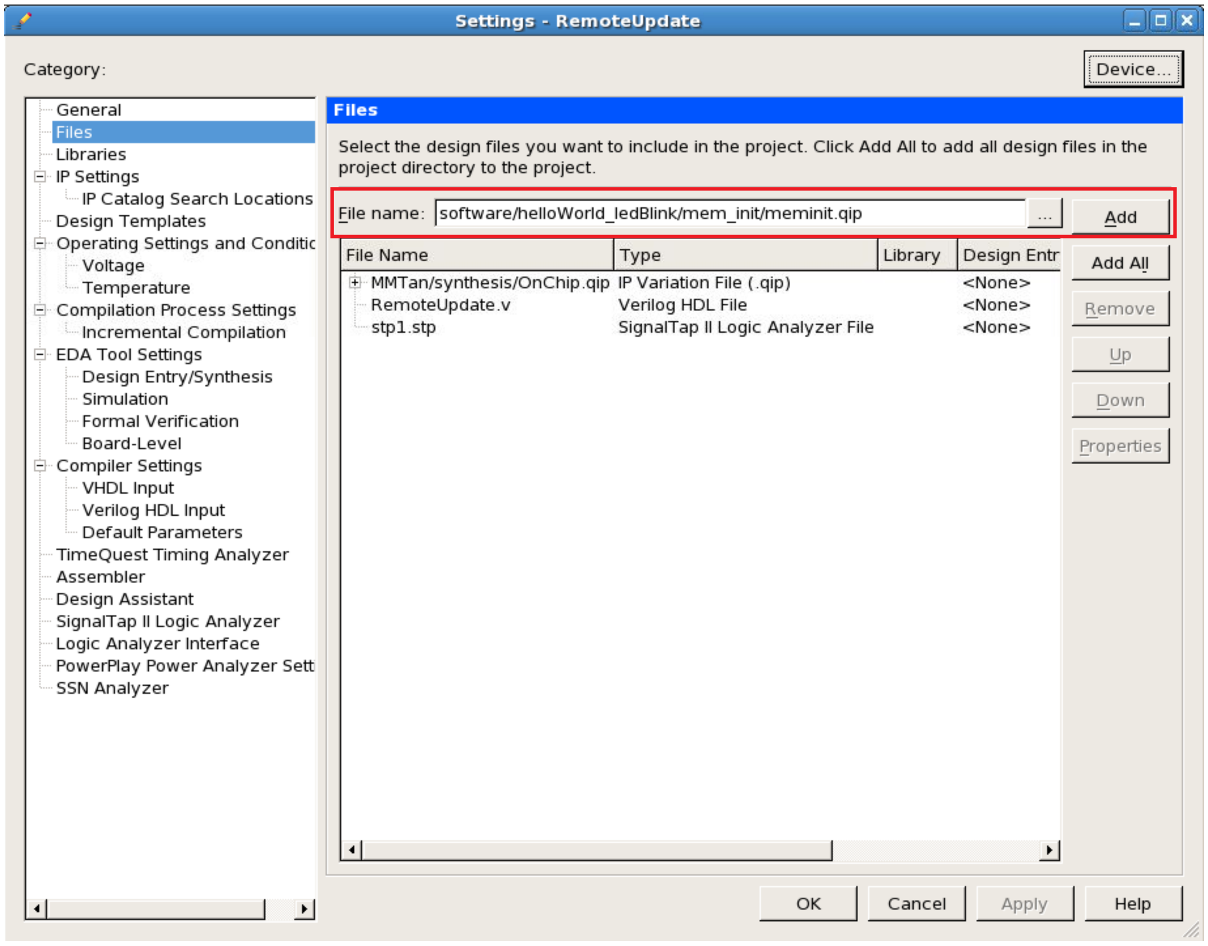
ROM/RAM Memory Protection
Reset Request: Enabled

ECC Parameter
Extend the data width to support ECC bits: Disabled

Memory initialization
☑ Initialize memory content
☐ Enable non-default initialization file
Type the filename (e.g: my_ram.hex) or select the hex file using the file browser button.
User created initialization file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/OnChip_onchip_memory2_0.hex
☐ Enable In-System Memory Content Editor feature
Instance ID: NONE

Memory will be initialized from OnChip_onchip_memory2_0.hex

Figure 47: Adding **meminit.qip** File into Quartus II

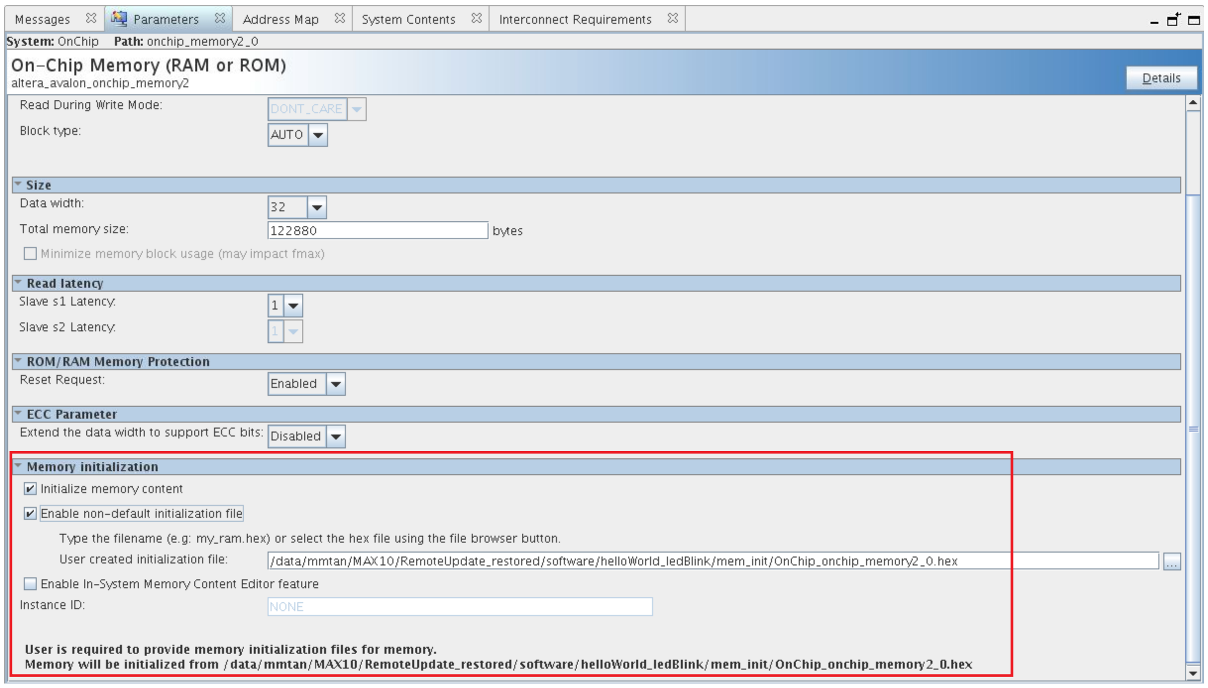


Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name: software/helloWorld_ledBlink/mem_init/meminit.qip

File Name	Type	Library	Design Entr
MMTan/synthesis/OnChip.qip	IP Variation File (.qip)		<None>
RemoteUpdate.v	Verilog HDL File		<None>
stp1.stp	SignalTap II Logic Analyzer File		<None>

Figure 48: Enable Initialize Memory Content with Non-default Initialization File in On-Chip Memory Parameter Editor Settings



System: OnChip **Path:** onchip_memory2_0

On-Chip Memory (RAM or ROM)
altera_avalon_onchip_memory2

Read During Write Mode: DONT_CARE
Block type: AUTO

Size
Data width: 32
Total memory size: 122880 bytes
☐ Minimize memory block usage (may impact fmax)

Read latency
Slave s1 Latency: 1
Slave s2 Latency: 1

ROM/RAM Memory Protection
Reset Request: Enabled

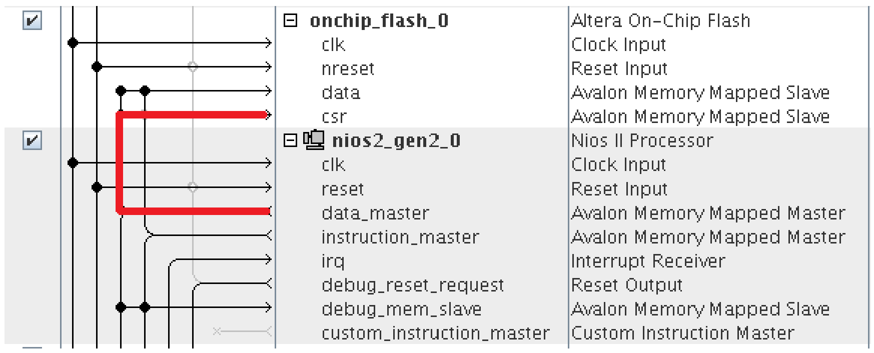
ECC Parameter
Extend the data width to support ECC bits: Disabled

Memory initialization
☑ Initialize memory content
☑ Enable non-default initialization file
Type the filename (e.g: my_ram.hex) or select the hex file using the file browser button.
User created initialization file: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/OnChip_onchip_memory2_0.hex
☐ Enable In-System Memory Content Editor feature
Instance ID: NONE

User is required to provide memory initialization files for memory.
Memory will be initialized from /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink/mem_init/OnChip_onchip_memory2_0.hex

4. Ensure that Altera On-chip Flash `csr` port is connected to the Nios II processor `data_master` to enable write and erase operations.

Figure 49: CSR Connection to Nios II `data_master`



☑ **onchip_flash_0**	Altera On-Chip Flash
clk	Clock Input
nreset	Reset Input
data	Avalon Memory Mapped Slave
csr	Avalon Memory Mapped Slave
☑ **nios2_gen2_0**	Nios II Processor
clk	Clock Input
reset	Reset Input
data_master	Avalon Memory Mapped Master
instruction_master	Avalon Memory Mapped Master
irq	Interrupt Receiver
debug_reset_request	Reset Output
debug_mem_slave	Avalon Memory Mapped Slave
custom_instruction_master	Custom Instruction Master

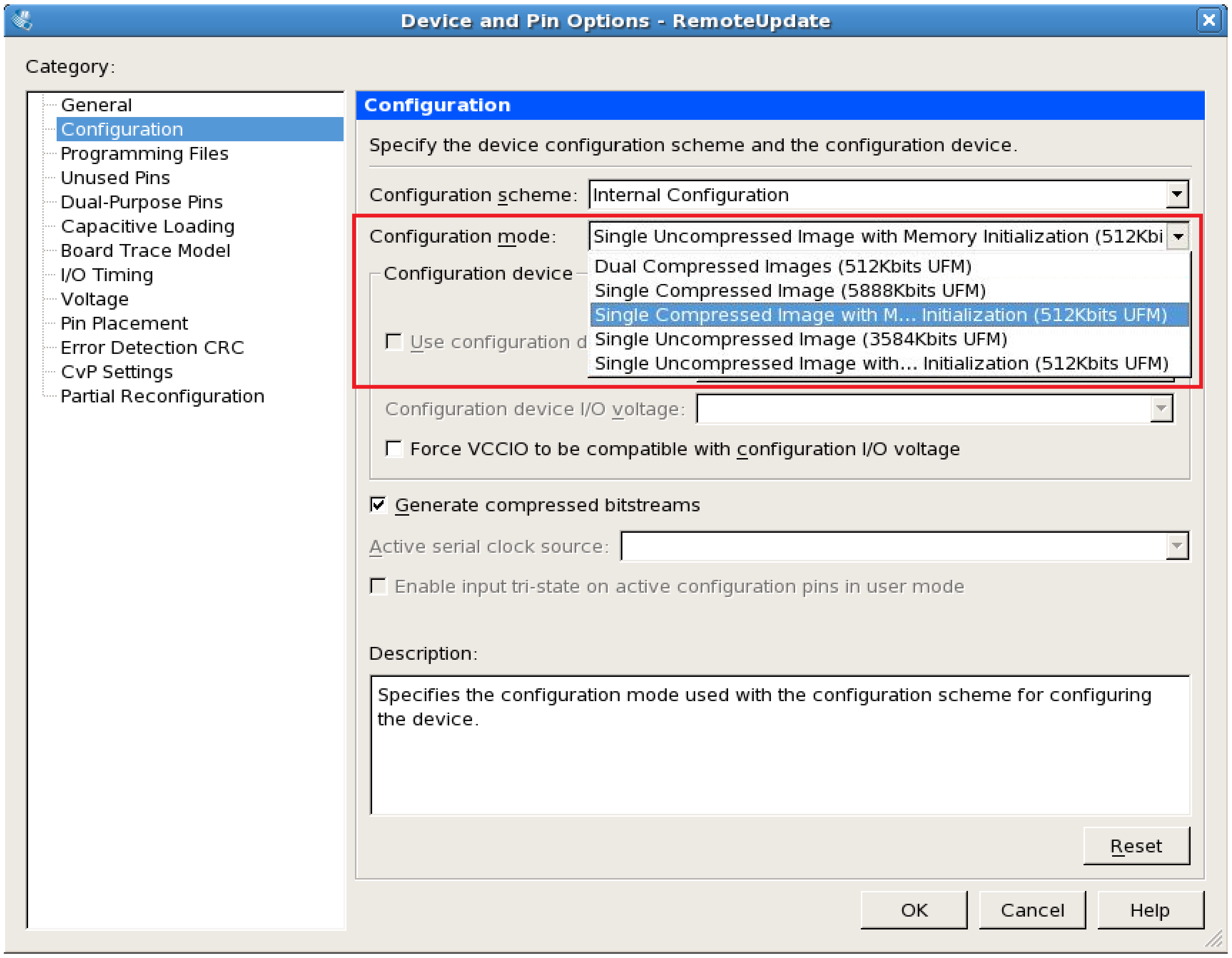
5. Click **Generate HDL**, the **Generation** dialog box appears.
6. Specify output file generation options, and then click **Generate**.

Quartus II Software Settings

1. In the Quartus II software, click on **Assignment** -> **Device** -> **Device and Pin Options** -> **Configuration**. Set **Configuration mode** to **Single Uncompressed Image with Memory Initialization** or **Single Compressed Image with Memory Initialization**.[13]

[13] The size of UFM sector will vary according to your device selection.

Figure 50: Configuration Modes with Memory Initialization Selection in Quartus II



Device and Pin Options - RemoteUpdate

Category:

- General
- Configuration
- Programming Files
- Unused Pins
- Dual-Purpose Pins
- Capacitive Loading
- Board Trace Model
- I/O Timing
- Voltage
- Pin Placement
- Error Detection CRC
- CvP Settings
- Partial Reconfiguration

Configuration

Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Single Uncompressed Image with Memory Initialization (512Kbi

- Dual Compressed Images (512Kbits UFM)
- Single Compressed Image (5888Kbits UFM)
- Single Compressed Image with M... Initialization (512Kbits UFM)
- Single Uncompressed Image (3584Kbits UFM)
- Single Uncompressed Image with... Initialization (512Kbits UFM)

Configuration device

☐ Use configuration d

Configuration device I/O voltage:

☐ Force VCCIO to be compatible with configuration I/O voltage

☑ Generate compressed bitstreams

Active serial clock source:

☐ Enable input tri-state on active configuration pins in user mode

Description:

Specifies the configuration mode used with the configuration scheme for configuring the device.

Reset

OK Cancel Help

Note:
- 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.

```
14740 Configuration Mode parameter on atom "ufm_block" is inconsistent with Quartus II project setting.
14740 MAX address parameter on atom "ufm_block" is inconsistent with Quartus II project setting.
      Quartus II 64-Bit Fitter was unsuccessful. 2 errors, 0 warnings
293001 Quartus II Full Compilation was unsuccessful. 4 errors, 10 warnings
```

- If configuration mode without memory initialization is selected with **Initialize flash content** checked in Altera On-chip Memory IP, the Quartus II project compilation will fail with the following error message:

```
16031 Current Internal Configuration mode does not support memory initialization or ROM. Select Internal Configuration mode with ERAM.
16031 Current Internal Configuration mode does not support memory initialization or ROM. Select Internal Configuration mode with ERAM.
16031 Current Internal Configuration mode does not support memory initialization or ROM. Select Internal Configuration mode with ERAM.
16031 Current Internal Configuration mode does not support memory initialization or ROM. Select Internal Configuration mode with ERAM.
21057 Implemented 7144 device resources after synthesis - the final resource count might be different
      Quartus II 64-Bit Analysis & Synthesis was unsuccessful. 32 errors, 17 warnings
293001 Quartus II Full Compilation was unsuccessful. 34 errors, 17 warnings
```

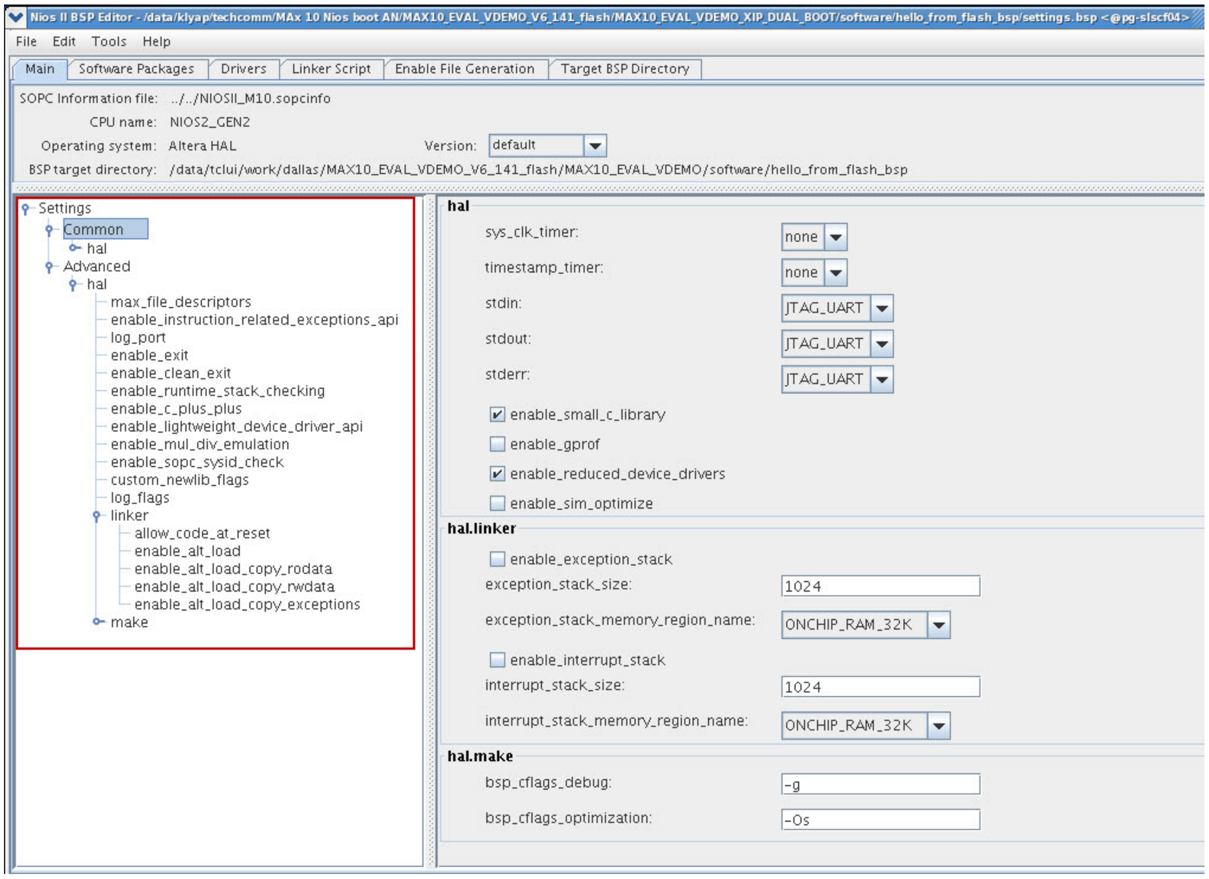
2. Click **OK** to exit the **Device and Pin Options window**.
3. Click **OK** to exit the **Device** window.
4. Click **Start Compilation** to compile your project and generate the **.sof** file.

BSP Editor Settings

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

1. In the Nios II SBT tool, right click on your BSP project in the **Project Explorer** window. Select **Nios II** > **BSP Editor...** to open the **Nios II BSP Editor**.
2. In Nios II BSP Editor, click on **Advanced** tab under **Settings**.
3. Click on **hal** to expand the list.
4. Click on **linker** to expand the list.

Figure 51: BSP Editor Settings


Nios II BSP Editor - /data/klyap/techcomm/MAx 10 Nios boot AN/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO_XIP_DUAL_BOOT/software/hello_from_flash_bsp/settings.bsp <@pg-slscf04>

File Edit Tools Help

Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory

SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL   Version: default
BSP target directory: /data/tclui/work/dallas/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO/software/hello_from_flash_bsp

- Settings
  - Common
    - hal
  - Advanced
    - hal
      - max_file_descriptors
      - enable_instruction_related_exceptions_api
      - log_port
      - enable_exit
      - enable_clean_exit
      - enable_runtime_stack_checking
      - enable_c_plus_plus
      - enable_lightweight_device_driver_api
      - enable_mul_div_emulation
      - enable_sopc_sysid_check
      - custom_newlib_flags
      - log_flags
      - linker
        - allow_code_at_reset
        - enable_alt_load
        - enable_alt_load_copy_rodata
        - enable_alt_load_copy_rwdata
        - enable_alt_load_copy_exceptions
      - make

hal

sys_clk_timer:	none
timestamp_timer:	none
stdin:	JTAG_UART
stdout:	JTAG_UART
stderr:	JTAG_UART

[✓] enable_small_c_library
[ ] enable_gprof
[✓] enable_reduced_device_drivers
[ ] enable_sim_optimize

hal.linker

[ ] enable_exception_stack

exception_stack_size:	1024
exception_stack_memory_region_name:	ONCHIP_RAM_32K

[ ] enable_interrupt_stack

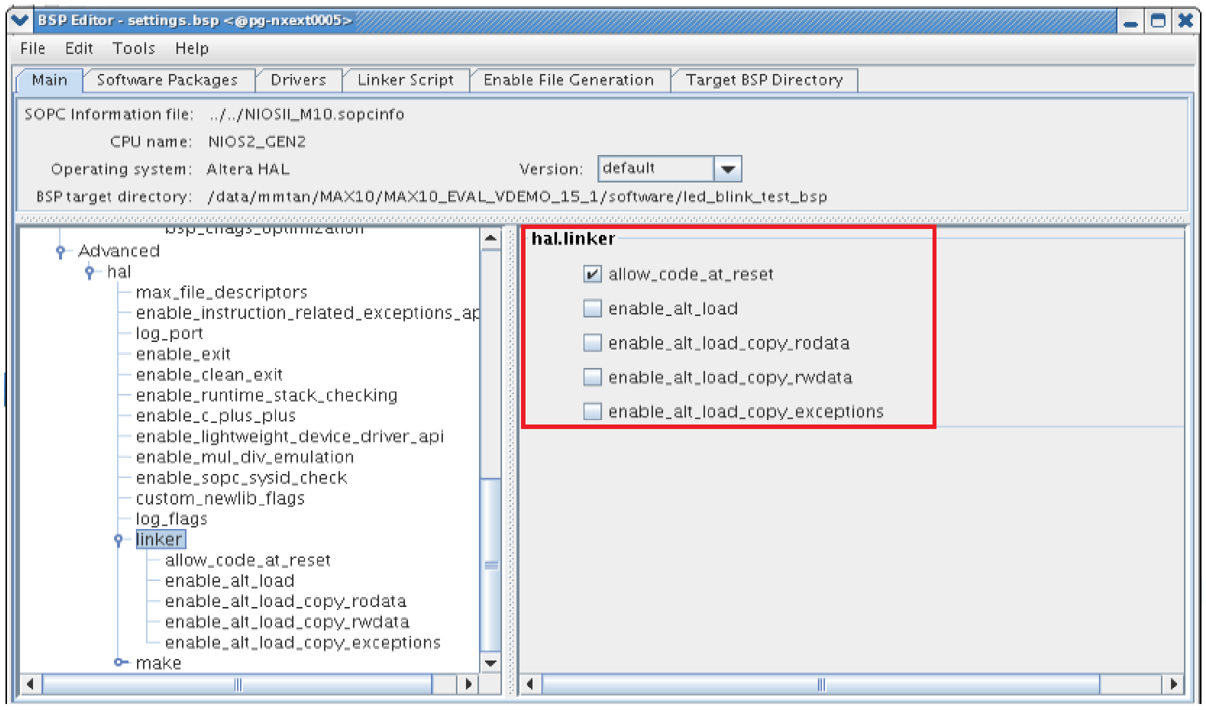
interrupt_stack_size:	1024
interrupt_stack_memory_region_name:	ONCHIP_RAM_32K

hal.make

bsp_cflags_debug:	-g
bsp_cflags_optimization:	-Os

5. Enable **allow_code_at_reset** and leave others unchecked to make sure the application starts at address `0x00`.

Figure 52: Advanced.hal.linker Default Settings


BSP Editor - settings.bsp <@pg-nxext0005>

File Edit Tools Help

Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory

SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL   Version: default
BSP target directory: /data/mmtan/MAX10/MAX10_EVAL_VDEMO_15_1/software/led_blink_test_bsp

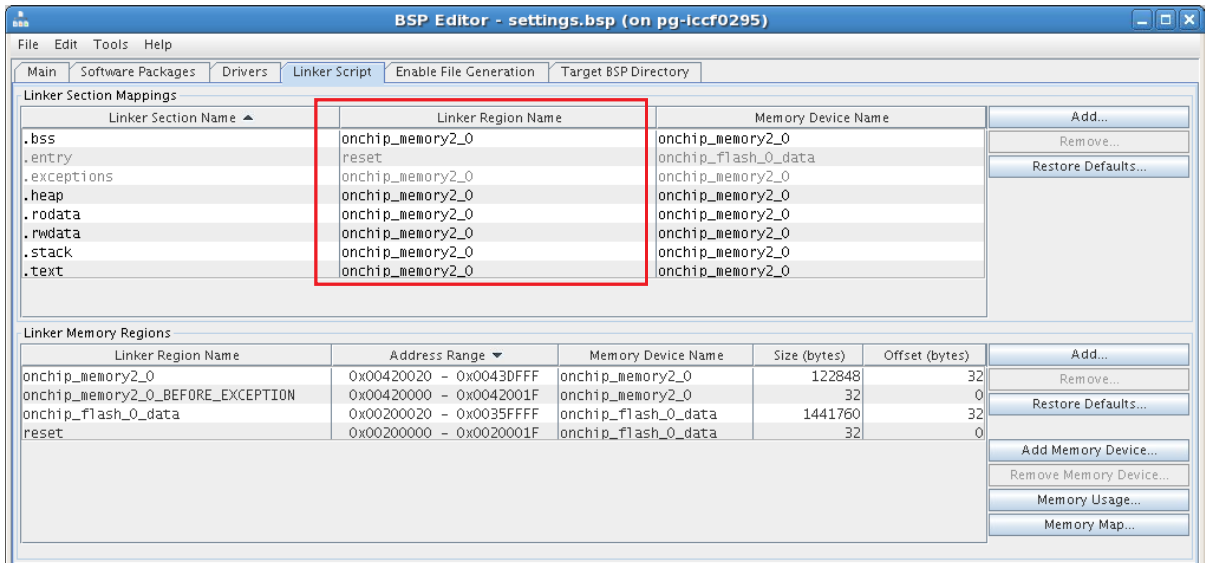
```
bsp_cflags_optimization
Advanced
  hal
    max_file_descriptors
    enable_instruction_related_exceptions_ap
    log_port
    enable_exit
    enable_clean_exit
    enable_runtime_stack_checking
    enable_c_plus_plus
    enable_lightweight_device_driver_api
    enable_mul_div_emulation
    enable_sopc_sysid_check
    custom_newlib_flags
    log_flags
    linker
      allow_code_at_reset
      enable_alt_load
      enable_alt_load_copy_rodata
      enable_alt_load_copy_rwdata
      enable_alt_load_copy_exceptions
    make
```

hal.linker

- [✓] allow_code_at_reset
- [ ] enable_alt_load
- [ ] enable_alt_load_copy_rodata
- [ ] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

6. Click on **Linker Script** tab in the **Nios II BSP Editor**.
7. Set all the **Linker Section Regions** to **Altera On-chip Memory (OCRAM)**.

Figure 53: Linker Region Default Settings


BSP Editor - settings.bsp (on pg-iccf0295)

File Edit Tools Help

Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory

Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	onchip_memory2_0	onchip_memory2_0
.entry	reset	onchip_flash_0_data
.exceptions	onchip_memory2_0	onchip_memory2_0
.heap	onchip_memory2_0	onchip_memory2_0
.rodata	onchip_memory2_0	onchip_memory2_0
.rwdata	onchip_memory2_0	onchip_memory2_0
.stack	onchip_memory2_0	onchip_memory2_0
.text	onchip_memory2_0	onchip_memory2_0

Add... Remove... Restore Defaults...

Linker Memory Regions

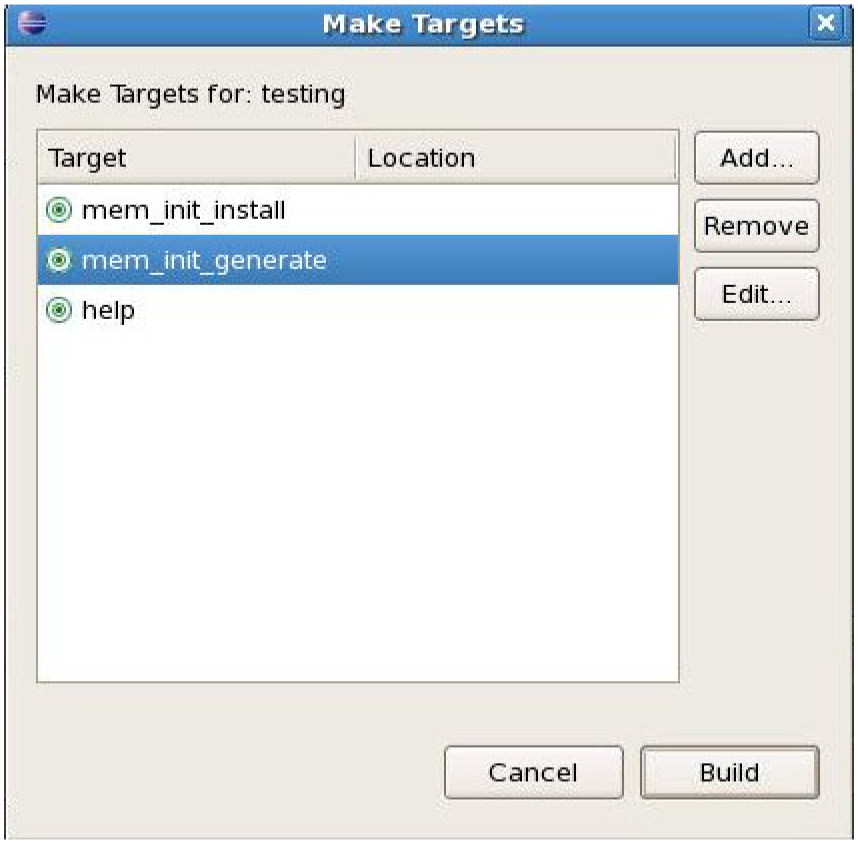
Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_memory2_0	0x00420020 - 0x0043DFFF	onchip_memory2_0	122848	32
onchip_memory2_0_BEFORE_EXCEPTION	0x00420000 - 0x0042001F	onchip_memory2_0	32	0
onchip_flash_0_data	0x00200020 - 0x0035FFFF	onchip_flash_0_data	1441760	32
reset	0x00200000 - 0x0020001F	onchip_flash_0_data	32	0

Add... Remove... Restore Defaults... Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...

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.

Figure 54: Selecting mem_init_generate in Make Targets



5. The “mem_init_generate” macro will create two HEX files; **<on_chip_ram.hex>** and **<on_chip_flash.hex>**. The **<on_chip_ram.hex>** will be used for boot option 3 and **<on_chip_flash.hex>** is used for boot option 1 and 2.

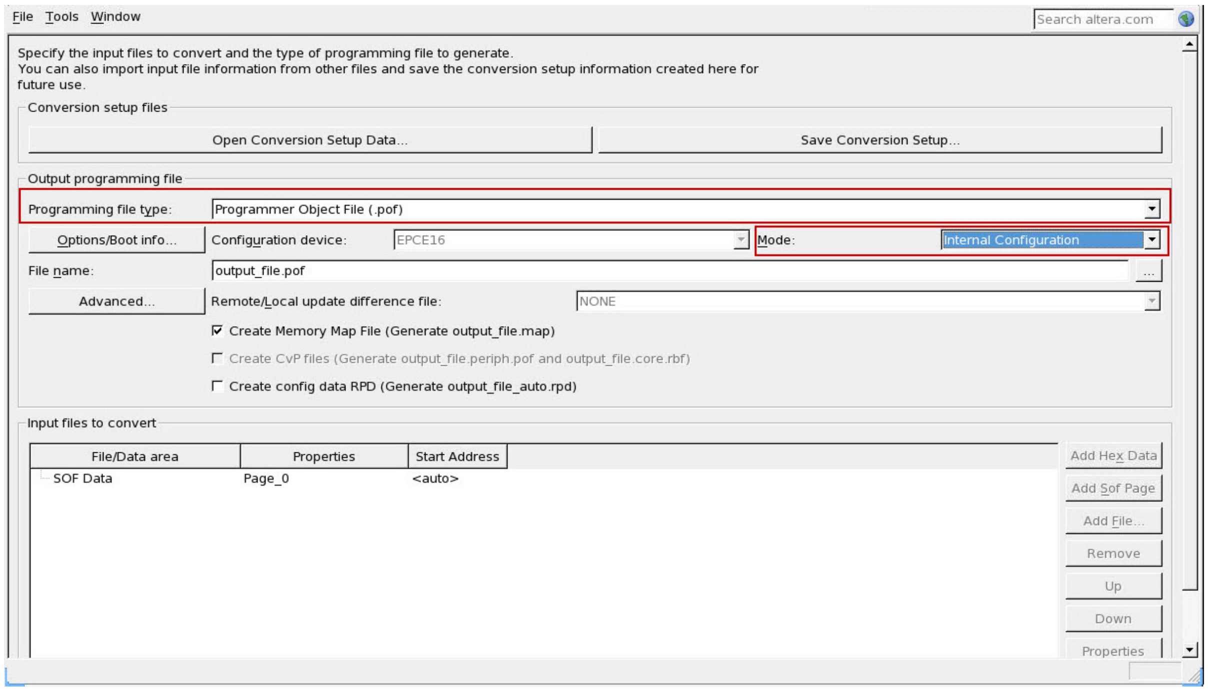
   Note:
   - The **mem_init_generate** target also generates a Quartus II IP file (**meminit.qip**). Quartus II software will refer to the **meminit.qip** for the location of the initialization files.
   - All these files can be found under "<project_folder>/software/<application_name>/mem_init" folder.
6. Recompile your project in Quartus II software to include the software data (.HEX) into the SOF file.

Programmer Object File (.pof) Generation

Note: If you are using Quartus II software version 15.0 and above, you can skip all the steps in this section. From Quartus II 15.0 onwards, the Quartus II software will generate both SOF and POF files. Since boot option 3 comprises single image, you may use the generated POF file to program into the MAX 10 FPGA device directly.

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**.

Figure 55: Convert Programming File Settings



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_file.pof
Remote/Local update difference file: NONE

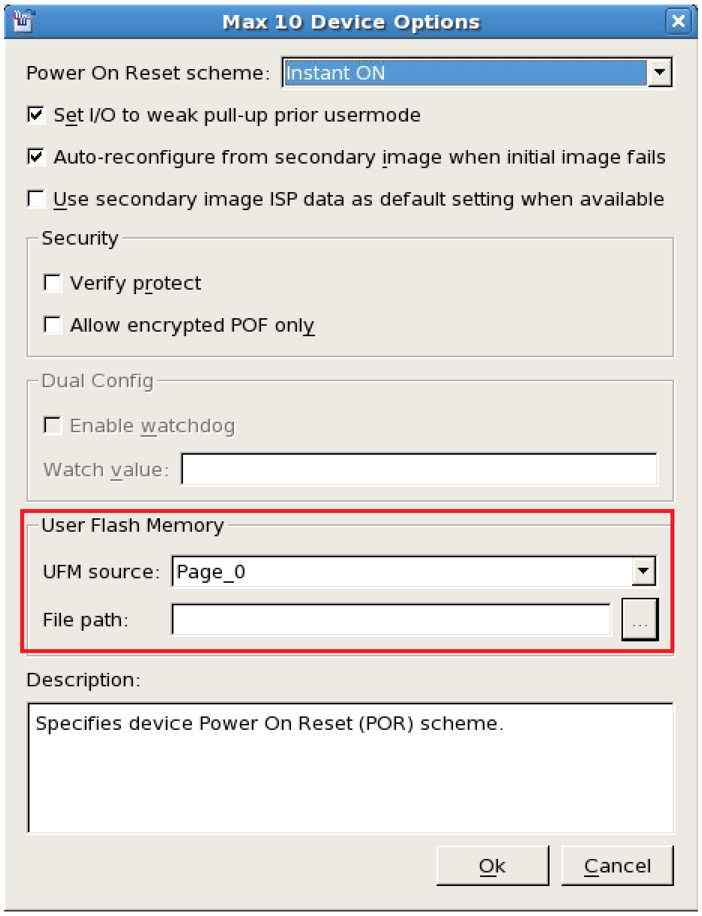
☑ Create Memory Map File (Generate output_file.map)
☐ Create CvP files (Generate output_file.periph.pof and output_file.core.rbf)
☐ Create config data RPD (Generate output_file_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>

4. Click on **Options/Boot info...**, the MAX 10 Device Options dialog box appears.
5. Make sure Page_0 is set as the **UFM source:** option. Click **OK**.

Figure 56: Setting Page_0 for UFM Source


Max 10 Device Options

Power On Reset scheme: Instant ON

[✓] Set I/O to weak pull-up prior usermode

[✓] Auto-reconfigure from secondary image when initial image fails

[ ] Use secondary image ISP data as default setting when available

Security

[ ] Verify protect

[ ] Allow encrypted POF only

Dual Config

[ ] Enable watchdog

Watch value:

User Flash Memory

UFM source: Page_0

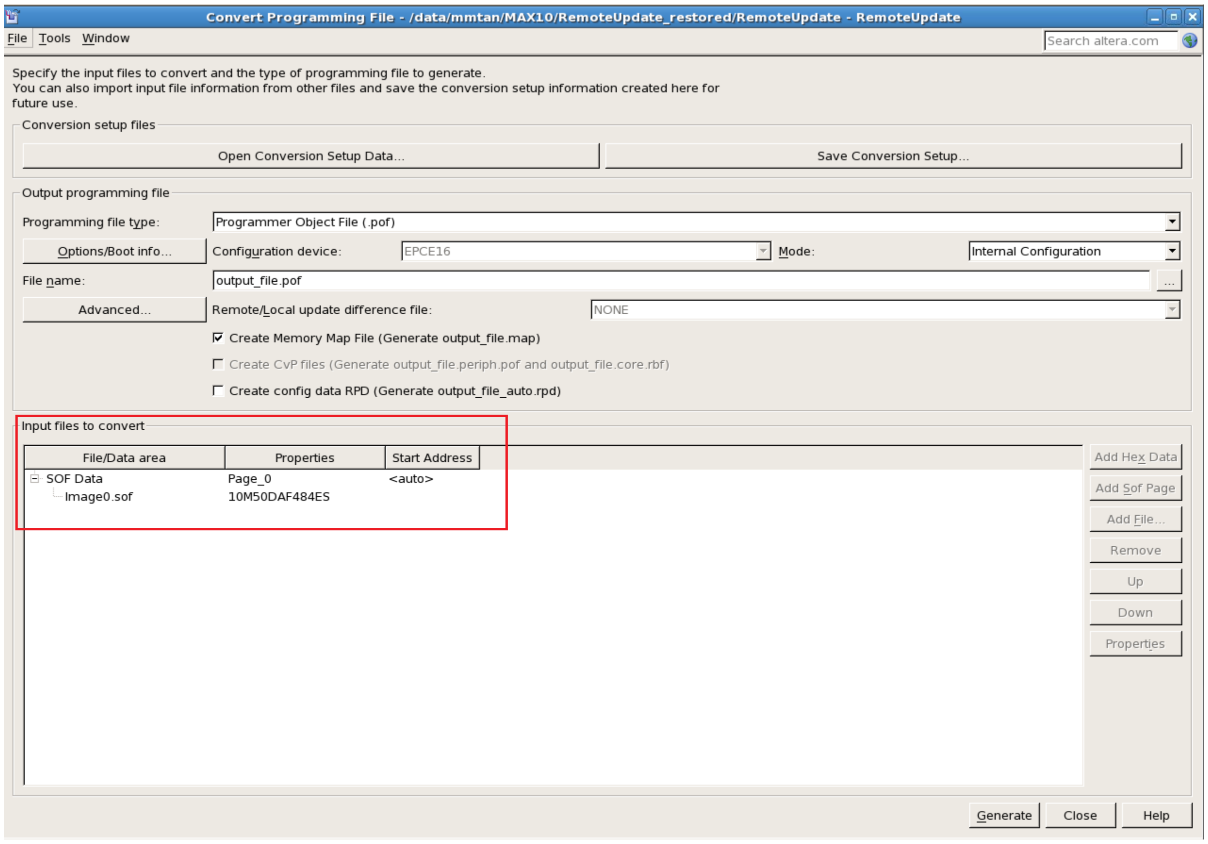
File path:

Description:

Specifies device Power On Reset (POR) scheme.

6. 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 to add the **.sof** file at **page_0**.

Figure 57: Input Files to Convert in Convert Programming Files


Convert Programming File - /data/mmtan/MAX10/RemoteUpdate_restored/RemoteUpdate - RemoteUpdate

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_file.pof
Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)
☐ Create CvP files (Generate output_file.periph.pof and output_file.core.rbf)
☐ Create config data RPD (Generate output_file_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
SOF Data	Page_0	<auto>
Image0.sof	10M50DAF484ES	

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

Boot Option 4 and Option 5

Boot Option 4: Nios II Processor Application Executes in-place from QSPI Flash

This option is suitable for Nios II processor applications which require code space that is larger than the UFM and/or limited on-chip memory usage. It has a similar concept to boot option 1 where the Nios II processor application execute-in-place from UFM but with more memory space due to use of QSPI flash (depending on QSPI chip selection) instead of the UFM. This is an advantage for supporting larger or multiple software applications.

The alt_load() function operates as a mini boot copier that initializes and copies the writable memory sections only to OCRAM or external RAM. The code section (**.text**), which is a read-only section, remains in the QSPI flash memory region. Retaining the read-only section in QSPI minimizes RAM usage but may limit the code execution performance. The Nios II processor application is programmed into the QSPI flash.

The Nios II processor reset vector points to the QSPI 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 break-point because the QSPI cannot support random memory access.

Figure 58: Boot Option 4 Block Diagram

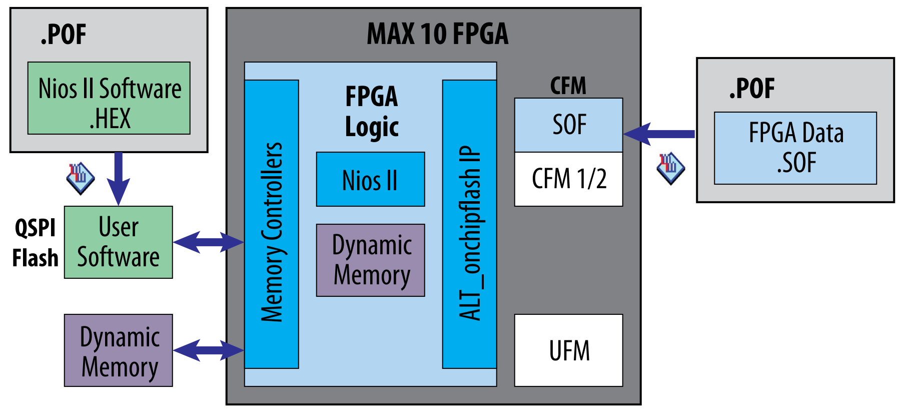


Table 9: RAM and ROM Size Requirement for Boot Option 4

RAM Size Requirement	ROM Size Requirement
Equivalent to the dynamic memory space usage during run time which is the sum of the maximum heap and stack size.	Executable code must not exceed the size of the QSPI flash.

Option 5: Nios II Processor Application Copied from QSPI Flash to RAM Using Boot Copier

Using a boot copier to copy the Nios II application from QSPI flash to RAM is suitable when multiple iterations of application software development and high system performance are required. It has a similar concept to boot option 2 where Nios II processor application copied from QSPI flash to RAM using boot copier but with larger memory space due to QSPI flash (depending on QSPI chip selection). This is an advantage for supporting software applications that require larger program space and high software performance (compared to XIP).

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 that copies the application from the QSPI to the on-chip memory or external RAM. Once this process is complete, the Nios II processor transfers the program control over to the application.

Figure 59: Boot Option 5 Block Diagram

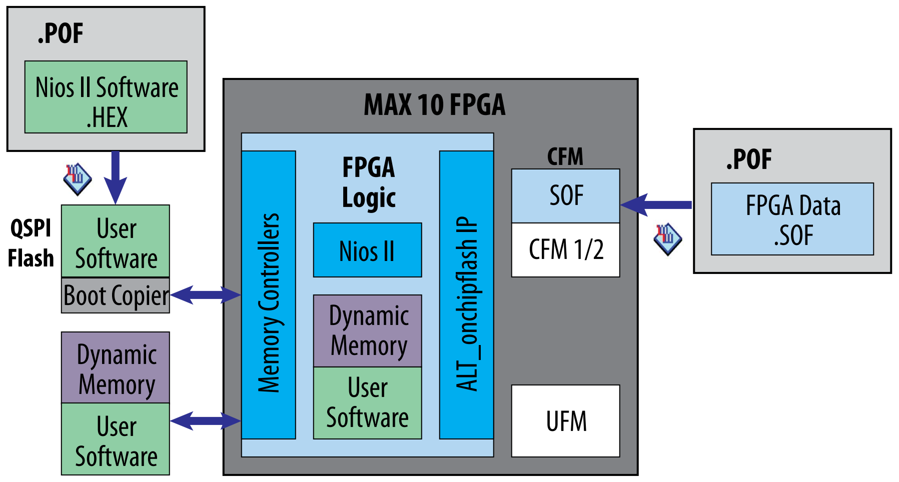
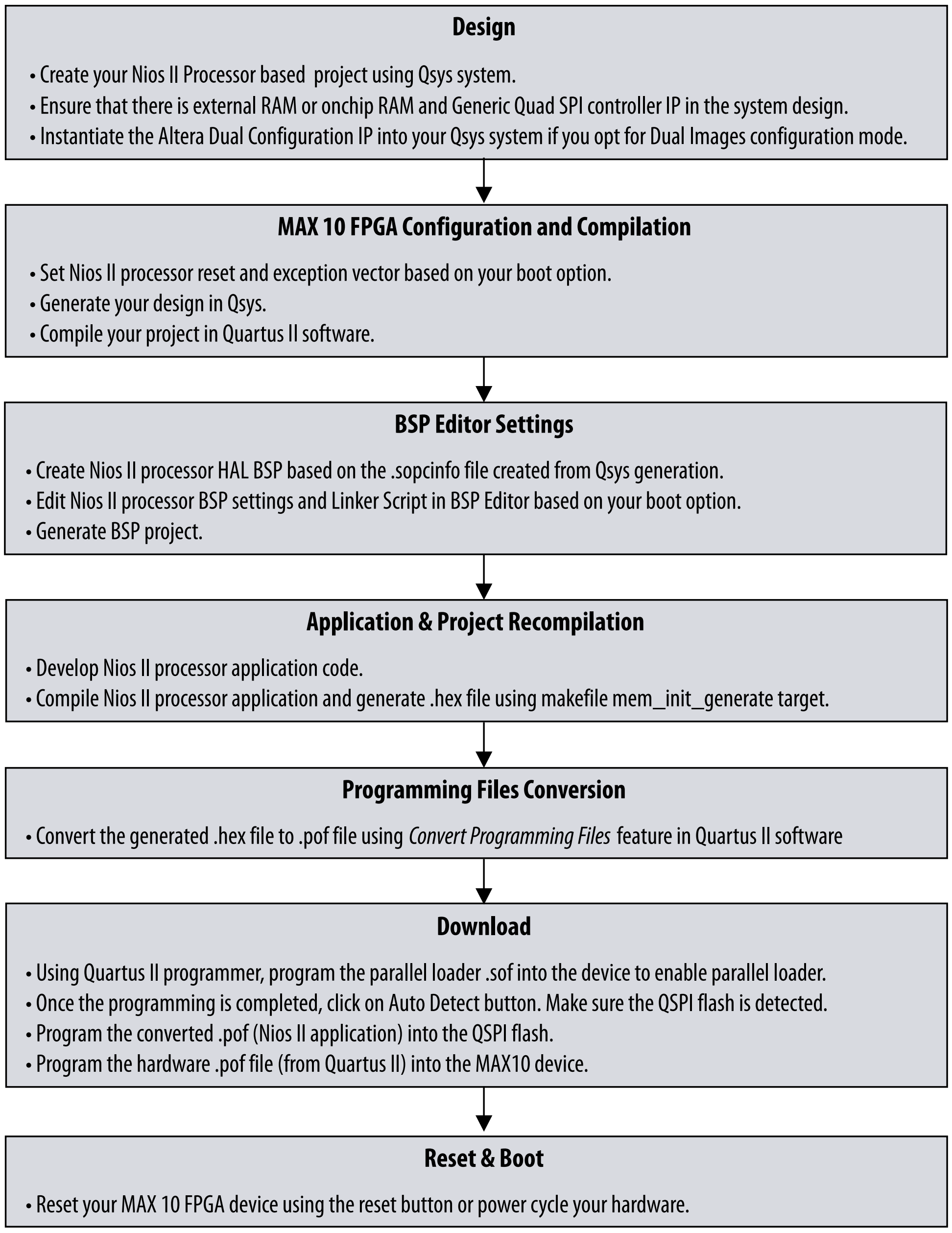


Table 10: RAM and ROM Size Requirement for Boot Option 5

RAM Size Requirement	ROM Size Requirement
Equivalent to the executable code and dynamic memory size required by user program.	Executable code and boot copier must not exceed the size of the QSPI flash.

Figure 60: Configuration and Booting Flow for Option 4 and Option 5


Design
• Create your Nios II Processor based project using Qsys system.
• Ensure that there is external RAM or onchip RAM and Generic Quad SPI controller IP in the system design.
• Instantiate the Altera Dual Configuration IP into your Qsys system if you opt for Dual Images configuration mode.
MAX 10 FPGA Configuration and Compilation
• Set Nios II processor reset and exception vector based on your boot option.
• Generate your design in Qsys.
• Compile your project in Quartus II software.
BSP Editor Settings
• Create Nios II processor HAL BSP based on the .sopcinfo file created from Qsys generation.
• Edit Nios II processor BSP settings and Linker Script in BSP Editor based on your boot option.
• Generate BSP project.
Application & Project Recompilation
• Develop Nios II processor application code.
• Compile Nios II processor application and generate .hex file using makefile mem_init_generate target.
Programming Files Conversion
• Convert the generated .hex file to .pof file using Convert Programming Files feature in Quartus II software
Download
• Using Quartus II programmer, program the parallel loader .sof into the device to enable parallel loader.
• Once the programming is completed, click on Auto Detect button. Make sure the QSPI flash is detected.
• Program the converted .pof (Nios II application) into the QSPI flash.
• Program the hardware .pof file (from Quartus II) into the MAX10 device.
Reset & Boot
• Reset your MAX 10 FPGA device using the reset button or power cycle your hardware.

Related Information

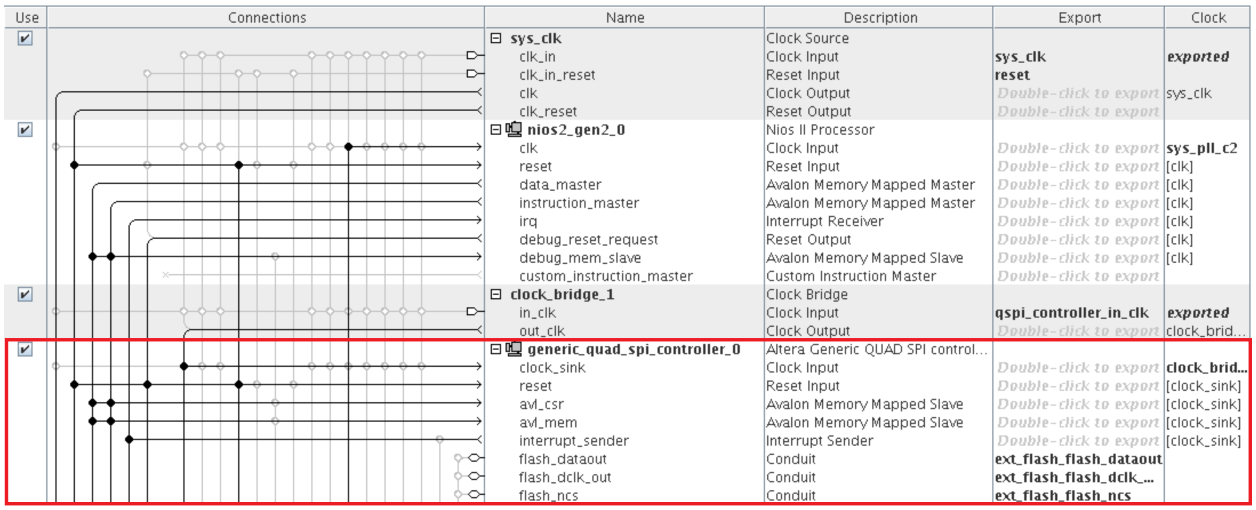
MAX 10 FPGA Configuration User Guide, section Remote System Upgrade in Dual Compressed Images.

Single Uncompressed/Compressed Image Bootable System Guideline

Design Creation

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

Figure 61: Connection for Generic Quad SPI Controller IP



Use	Connections	Name	Description	Export	Clock
✓		**sys_clk**	Clock Source		
		clk_in	Clock Input	**sys_clk**	*exported*
		clk_in_reset	Reset Input	**reset**	
		clk	Clock Output	Double-click to export	sys_clk
		clk_reset	Reset Output	Double-click to export	
✓		**nios2_gen2_0**	Nios II Processor		
		clk	Clock Input	Double-click to export	**sys_pll_c2**
		reset	Reset Input	Double-click to export	[clk]
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]
		irq	Interrupt Receiver	Double-click to export	[clk]
		debug_reset_request	Reset Output	Double-click to export	[clk]
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]
		custom_instruction_master	Custom Instruction Master	Double-click to export	
✓		**clock_bridge_1**	Clock Bridge		
		in_clk	Clock Input	**qspi_controller_in_clk**	*exported*
		out_clk	Clock Output	Double-click to export	clock_brid...
✓		**generic_quad_spi_controller_0**	Altera Generic QUAD SPI control...		
		clock_sink	Clock Input	Double-click to export	**clock_brid...**
		reset	Reset Input	Double-click to export	[clock_sink]
		avl_csr	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
		avl_mem	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
		interrupt_sender	Interrupt Sender	Double-click to export	[clock_sink]
		flash_dataout	Conduit	**ext_flash_flash_dataout**	
		flash_dclk_out	Conduit	**ext_flash_flash_dclk_...**	
		flash_ncs	Conduit	**ext_flash_flash_ncs**	

Note: The maximum input clock for Generic Quad SPI Controller IP is 25 MHz. The input clock must not exceed this maximum value.

Qsys Settings

1. In the Nios II Processor parameter editor, set the reset vector memory and exception vector memory based on the boot options below:

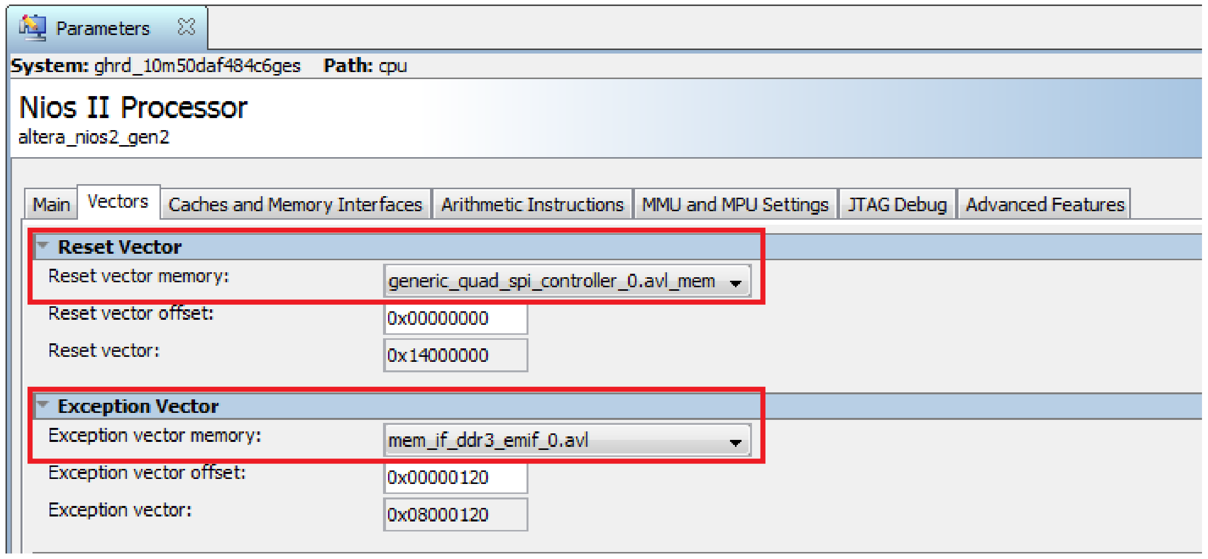
Boot Option	Reset vector memory	Exception vector memory
Option 4a [14][15]	QSPI flash	OCRAM/ External RAM
Option 4b[14]	QSPI flash	QSPI flash
Option 5	QSPI flash	OCRAM/ External RAM

[14] You can set the exception vector for Boot Option 4 to OCRAM/ External RAM (Option 4a) or QSPI Flash (Option 4b) according to your design preference.

[15] Boot option 4a which sets exception vector memory to OCRAM/External RAM is recommended to make the interrupt processing faster.

Figure 62: Nios II Parameter Editor Settings Boot Option 4a and 5

Note: **mem_if_ddr3_emif_0.avl** is the external memory (DDR3) used in this example.



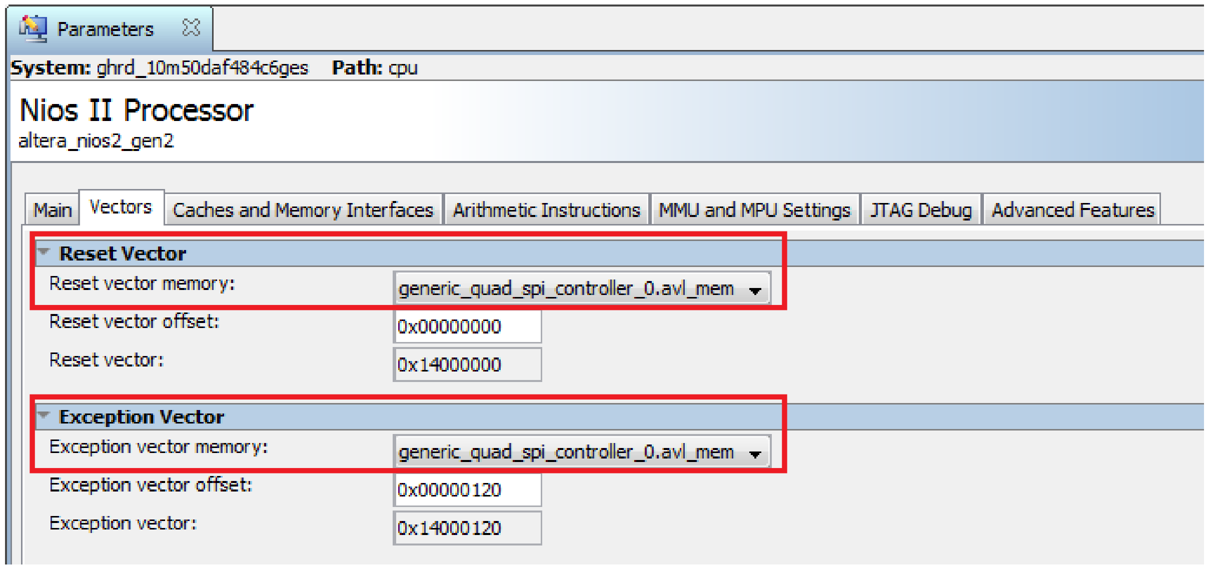
System: ghrd_10m50daf484c6ges Path: cpu

Nios II Processor
altera_nios2_gen2

Reset Vector
Reset vector memory: generic_quad_spi_controller_0.avl_mem
Reset vector offset: 0x00000000
Reset vector: 0x14000000

Exception Vector
Exception vector memory: mem_if_ddr3_emif_0.avl
Exception vector offset: 0x00000120
Exception vector: 0x08000120

Figure 63: Nios II Parameter Editor Settings Boot Option 4b



System: ghrd_10m50daf484c6ges Path: cpu

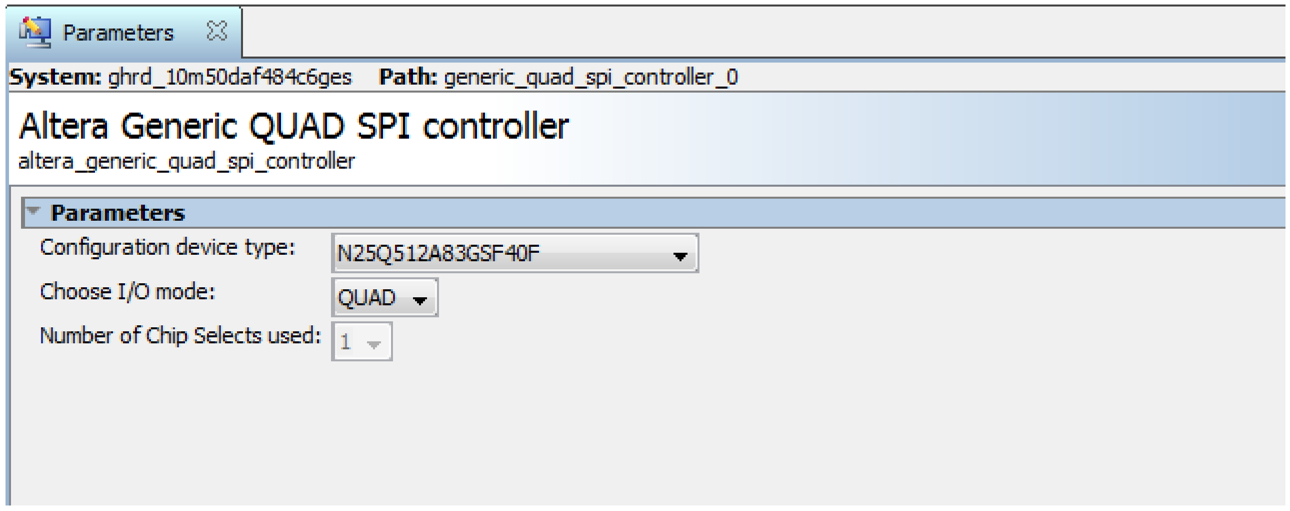
Nios II Processor
altera_nios2_gen2

Reset Vector
Reset vector memory: generic_quad_spi_controller_0.avl_mem
Reset vector offset: 0x00000000
Reset vector: 0x14000000

Exception Vector
Exception vector memory: generic_quad_spi_controller_0.avl_mem
Exception vector offset: 0x00000120
Exception vector: 0x14000120

2. Open **Altera Generic Quad SPI** controller parameter editor. Change the **Configuration device type** to the QSPI flash selection and make sure the **I/O mode** is set to **QUAD**.

Figure 64: Altera Generic Quad SPI Controller Parameter Settings



System: ghrd_10m50daf484c6ges **Path:** generic_quad_spi_controller_0

Altera Generic QUAD SPI controller
altera_generic_quad_spi_controller

Parameters

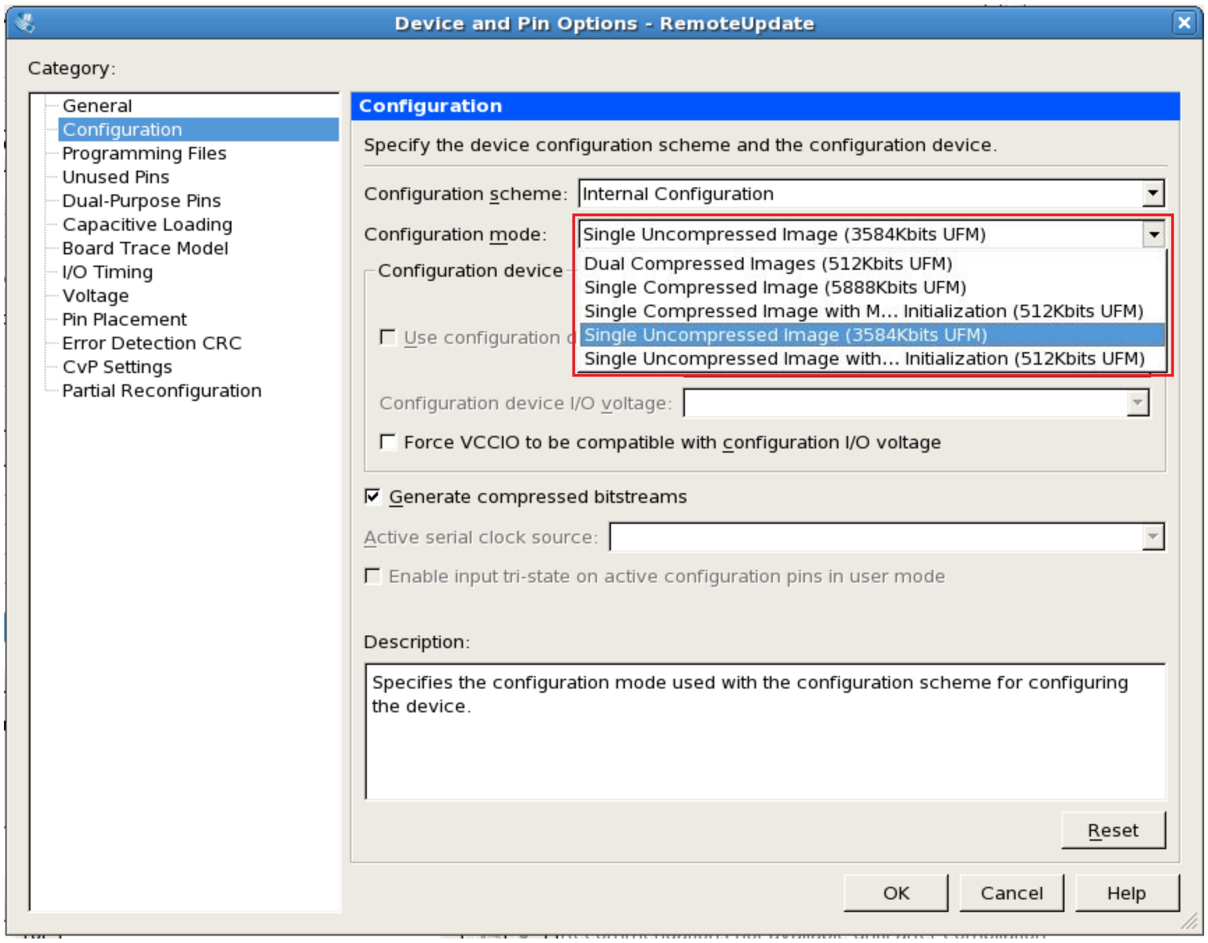
Configuration device type: N25Q512A83GSF40F
Choose I/O mode: QUAD
Number of Chip Selects used: 1

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

Quartus II Software Settings

1. In the Quartus II software, click on **Assignment** -> **Device** -> **Device and Pin Options** -> **Configuration**. Set **Configuration mode** to **Single Uncompressed Image** or **Single Compressed Image**.

Figure 65: Configuration Mode Selection in Quartus II Software



Device and Pin Options - RemoteUpdate

Category:
- General
- Configuration
- Programming Files
- Unused Pins
- Dual-Purpose Pins
- Capacitive Loading
- Board Trace Model
- I/O Timing
- Voltage
- Pin Placement
- Error Detection CRC
- CvP Settings
- Partial Reconfiguration

Configuration

Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Single Uncompressed Image (3584Kbits UFM)
- Dual Compressed Images (512Kbits UFM)
- Single Compressed Image (5888Kbits UFM)
- Single Compressed Image with M... Initialization (512Kbits UFM)
- Single Uncompressed Image (3584Kbits UFM)
- Single Uncompressed Image with... Initialization (512Kbits UFM)

Configuration device

☐ Use configuration d

Configuration device I/O voltage:

☐ Force VCCIO to be compatible with configuration I/O voltage

☑ Generate compressed bitstreams

Active serial clock source:

☐ Enable input tri-state on active configuration pins in user mode

Description:

Specifies the configuration mode used with the configuration scheme for configuring the device.

Reset

OK Cancel Help

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.

Note: SOF to POF file conversion is not required because there is only single hardware image and Nios II application data will be loaded into the QSPI flash separately. You can use the POF generated during Quartus II project compilation to program into the MAX 10 FPGA.

Related Information

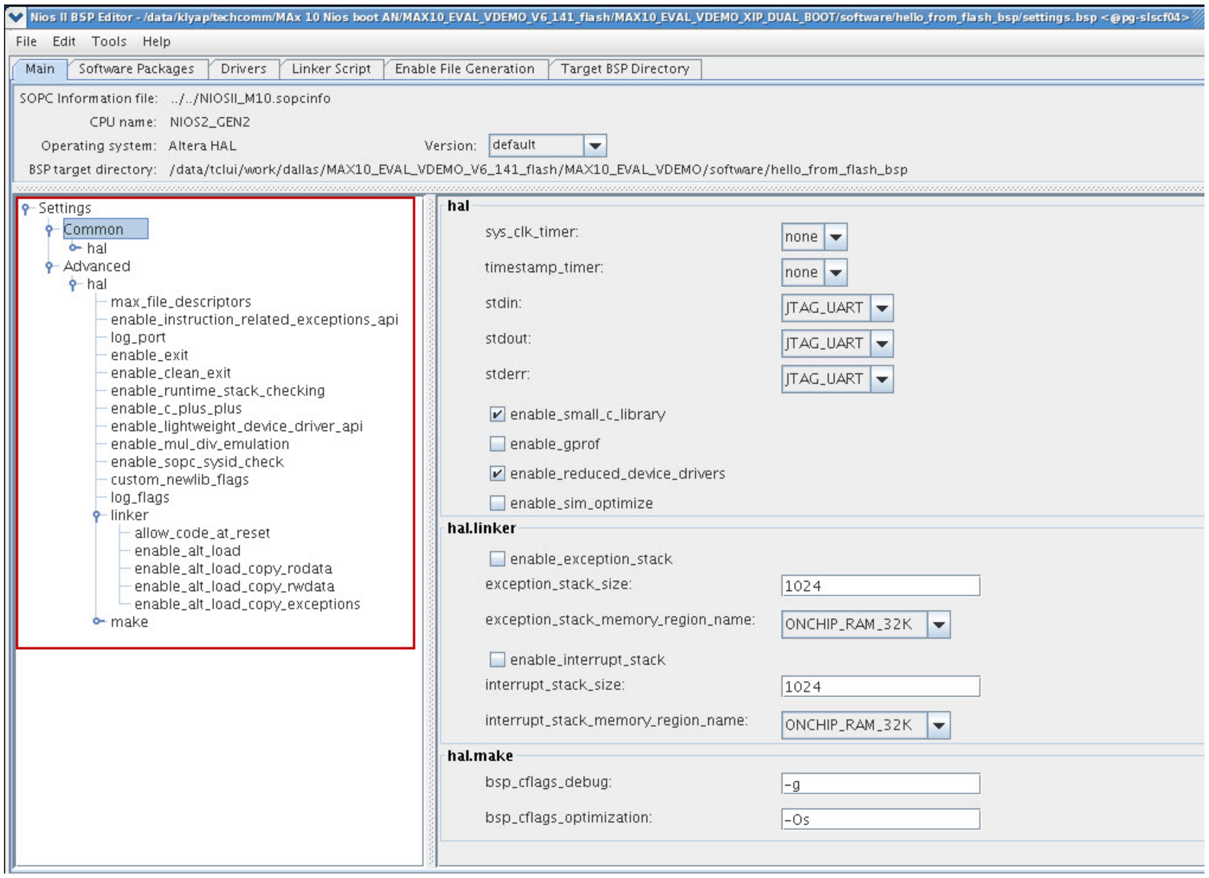
Programming Hardware Design POF File into the MAX10 FPGA on page 68

BSP Editor Settings

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

1. In the Nios II SBT tool, right click on your BSP project in the **Project Explorer** window. Select **Nios II** > **BSP Editor...** to open the **Nios II BSP Editor**.
2. In Nios II BSP Editor, click on **Advanced** tab under **Settings**.
3. Click on **hal** to expand the list.
4. Click on **linker** to expand the list.

Figure 66: BSP Editor Settings

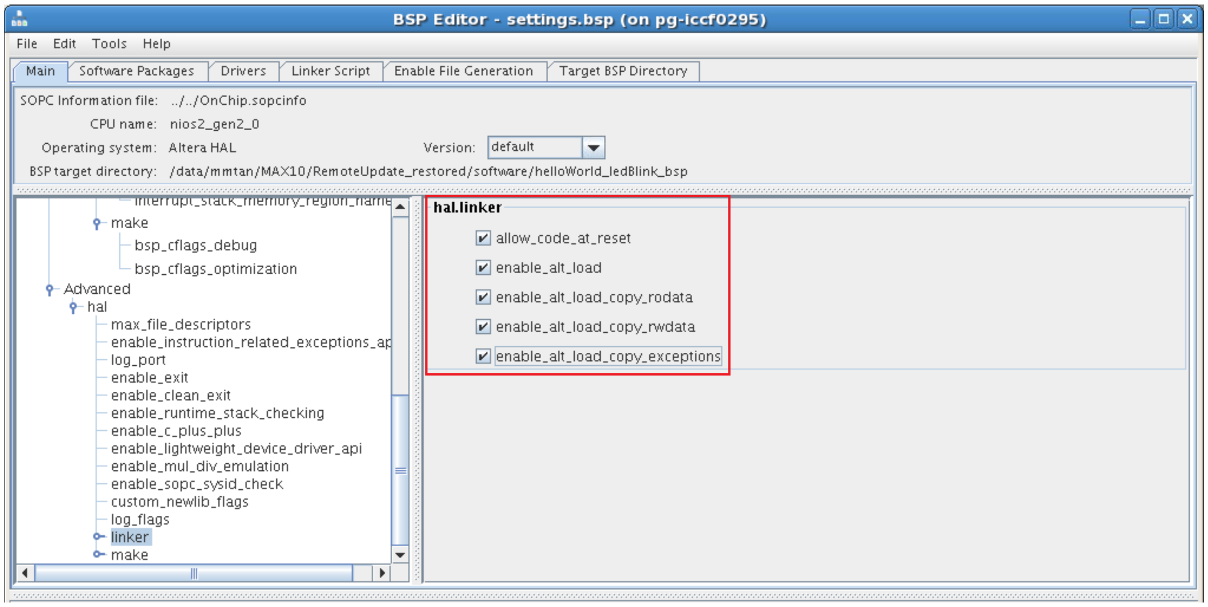


```
Nios II BSP Editor - /data/klyap/techcomm/MAx 10 Nios boot AN/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO_XIP_DUAL_BOOT/software/hello_from_flash_bsp/settings.bsp <@pg-slscf04>

SOPC Information file:  ../../NIOSII_M10.sopcinfo
         CPU name:  NIOS2_GEN2
 Operating system:  Altera HAL                    Version:  default
BSP target directory:  /data/tclui/work/dallas/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO/software/hello_from_flash_bsp

Settings
  Common
    hal
  Advanced
    hal
      max_file_descriptors
      enable_instruction_related_exceptions_api
      log_port
      enable_exit
      enable_clean_exit
      enable_runtime_stack_checking
      enable_c_plus_plus
      enable_lightweight_device_driver_api
      enable_mul_div_emulation
      enable_sopc_sysid_check
      custom_newlib_flags
      log_flags
      linker
        allow_code_at_reset
        enable_alt_load
        enable_alt_load_copy_rodata
        enable_alt_load_copy_rwdata
        enable_alt_load_copy_exceptions
      make

hal
  sys_clk_timer:                         none
  timestamp_timer:                       none
  stdin:                                 JTAG_UART
  stdout:                                JTAG_UART
  stderr:                                JTAG_UART
  [✓] enable_small_c_library
  [ ] enable_gprof
  [✓] enable_reduced_device_drivers
  [ ] enable_sim_optimize
hal.linker
  [ ] enable_exception_stack
  exception_stack_size:                  1024
  exception_stack_memory_region_name:    ONCHIP_RAM_32K
  [ ] enable_interrupt_stack
  interrupt_stack_size:                  1024
  interrupt_stack_memory_region_name:    ONCHIP_RAM_32K
hal.make
  bsp_cflags_debug:                      -g
  bsp_cflags_optimization:               -Os
```

5. Based on the boot option used, do one of the following:

   - For boot option 4a, if exception vector memory is set to OCRAM or External RAM, enable the following:
     - **allow_code_at_reset**
     - **enable_alt_load**
     - **enable_alt_load_copy_rodata**
     - **enable_alt_load_copy_rwdata**
     - **enable_alt_load_copy_exceptions**
   - For boot option 4b, if exception vector memory is set to QSPI flash, enable the following:
     - **allow_code_at_reset**
     - **enable_alt_load**
     - **enable_alt_load_copy_rodata**
     - **enable_alt_load_copy_rwdata**
   - For boot option 5, leave all the hal.linker settings unchecked.

Figure 67: Advanced.hal.linker Settings for Boot Option 4a

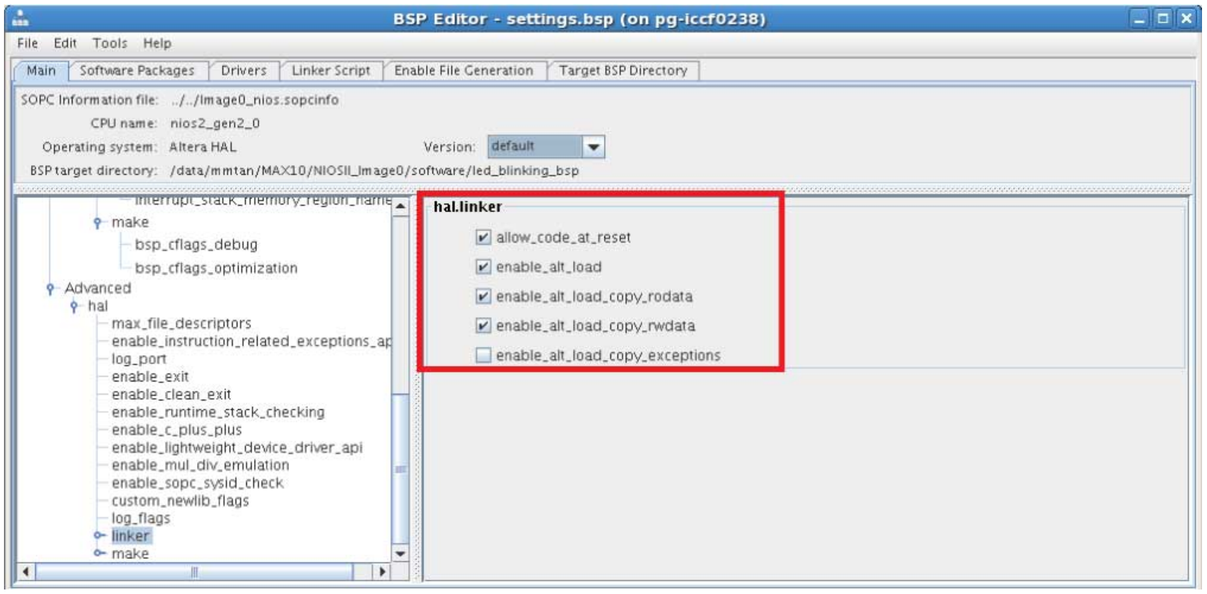

BSP Editor - settings.bsp (on pg-iccf0295)
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory

SOPC Information file: ../../OnChip.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL Version: default
BSP target directory: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink_bsp

interrupt_stack_memory_region_name
make
bsp_cflags_debug
bsp_cflags_optimization
Advanced
hal
max_file_descriptors
enable_instruction_related_exceptions_ap
log_port
enable_exit
enable_clean_exit
enable_runtime_stack_checking
enable_c_plus_plus
enable_lightweight_device_driver_api
enable_mul_div_emulation
enable_sopc_sysid_check
custom_newlib_flags
log_flags
linker
make

hal.linker
- [✓] allow_code_at_reset
- [✓] enable_alt_load
- [✓] enable_alt_load_copy_rodata
- [✓] enable_alt_load_copy_rwdata
- [✓] enable_alt_load_copy_exceptions

Figure 68: Advanced.hal.linker Settings for Boot Option 4b

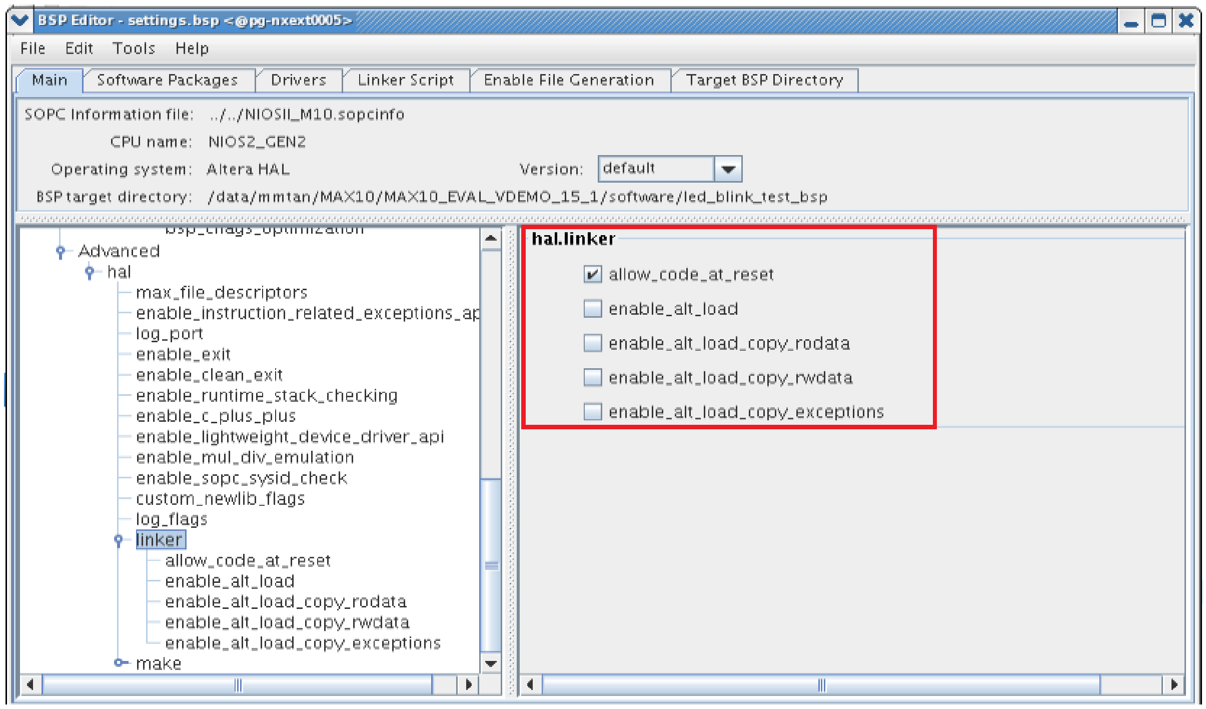

BSP Editor - settings.bsp (on pg-iccf0238)
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory

SOPC Information file: ../../image0_nios.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL Version: default
BSP target directory: /data/mmtan/MAX10/NIOSII_Image0/software/led_blinking_bsp

interrupt_stack_memory_region_name
make
bsp_cflags_debug
bsp_cflags_optimization
Advanced
hal
max_file_descriptors
enable_instruction_related_exceptions_ap
log_port
enable_exit
enable_clean_exit
enable_runtime_stack_checking
enable_c_plus_plus
enable_lightweight_device_driver_api
enable_mul_div_emulation
enable_sopc_sysid_check
custom_newlib_flags
log_flags
linker
make

hal.linker
- [✓] allow_code_at_reset
- [✓] enable_alt_load
- [✓] enable_alt_load_copy_rodata
- [✓] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

Figure 69: Advanced.hal.linker Settings for Boot Option 5



SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL
Version: default
BSP target directory: /data/mmtan/MAX10/MAX10_EVAL_VDEMO_15_1/software/led_blink_test_bsp

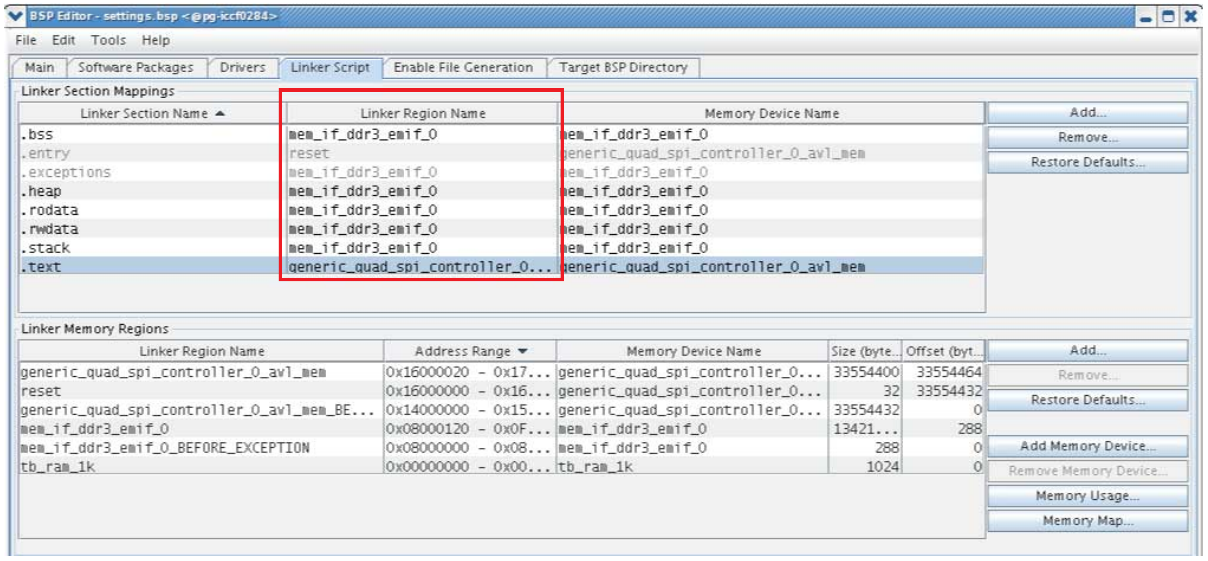
- Advanced
  - hal
    - max_file_descriptors
    - enable_instruction_related_exceptions_ap
    - log_port
    - enable_exit
    - enable_clean_exit
    - enable_runtime_stack_checking
    - enable_c_plus_plus
    - enable_lightweight_device_driver_api
    - enable_mul_div_emulation
    - enable_sopc_sysid_check
    - custom_newlib_flags
    - log_flags
    - linker
      - allow_code_at_reset
      - enable_alt_load
      - enable_alt_load_copy_rodata
      - enable_alt_load_copy_rwdata
      - enable_alt_load_copy_exceptions
    - make

hal.linker

- [✓] allow_code_at_reset
- [ ] enable_alt_load
- [ ] enable_alt_load_copy_rodata
- [ ] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

6. Click on **Linker Script** tab in the Nios II BSP Editor.
7. Based on the boot option used, do one of the following:
   - For boot option 4a and 4b, set the **.text** item in the **Linker Section Name** to the QSPI flash in the **Linker Region Name**. Set the rest of the items in the **Linker Section Name** list to the Altera On-chip Memory (OCRAM) or external RAM.
   - For boot option 5, set all of the items in the **Linker Section Name** list to Altera On-chip Memory (OCRAM) or external RAM.

Figure 70: Linker Region Settings for Boot Option 4a and 4b



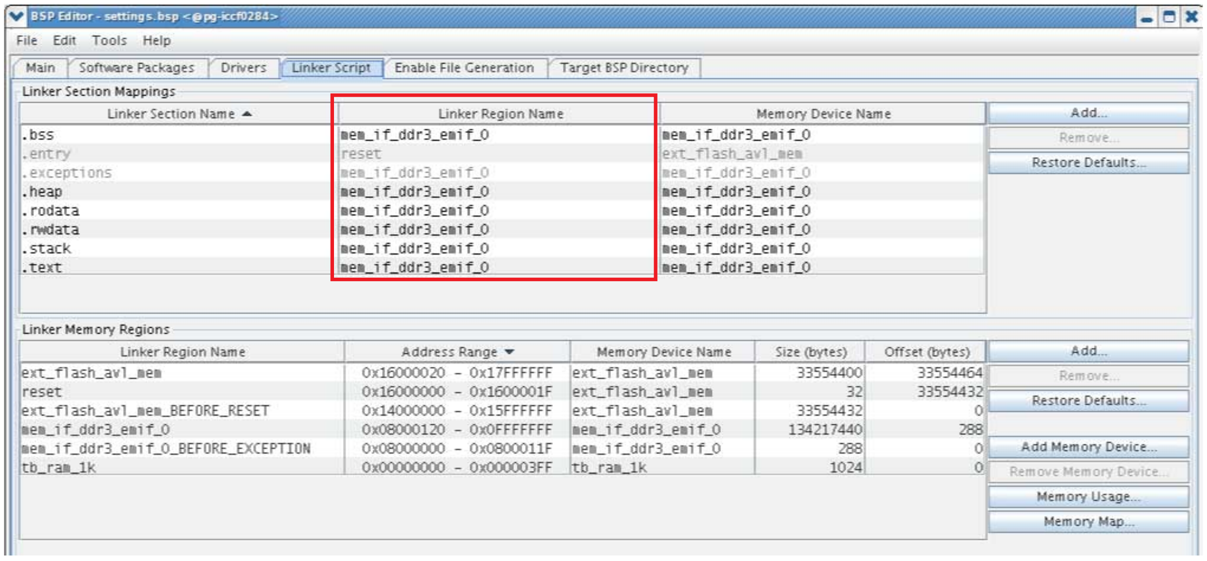
Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.entry	reset	generic_quad_spi_controller_0_avl_mem
.exceptions	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.heap	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rodata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rwdata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.stack	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.text	generic_quad_spi_controller_0...	generic_quad_spi_controller_0_avl_mem

Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (byte...	Offset (byt...
generic_quad_spi_controller_0_avl_mem	0x16000020 - 0x17...	generic_quad_spi_controller_0...	33554400	33554464
reset	0x16000000 - 0x16...	generic_quad_spi_controller_0...	32	33554432
generic_quad_spi_controller_0_avl_mem_BE...	0x14000000 - 0x15...	generic_quad_spi_controller_0...	33554432	0
mem_if_ddr3_emif_0	0x08000120 - 0x0F...	mem_if_ddr3_emif_0	13421...	288
mem_if_ddr3_emif_0_BEFORE_EXCEPTION	0x08000000 - 0x08...	mem_if_ddr3_emif_0	288	0
tb_ram_1k	0x00000000 - 0x00...	tb_ram_1k	1024	0

Figure 71: Linker Region Settings for Boot Option 5



Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.entry	reset	ext_flash_avl_mem
.exceptions	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.heap	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rodata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rwdata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.stack	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.text	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0

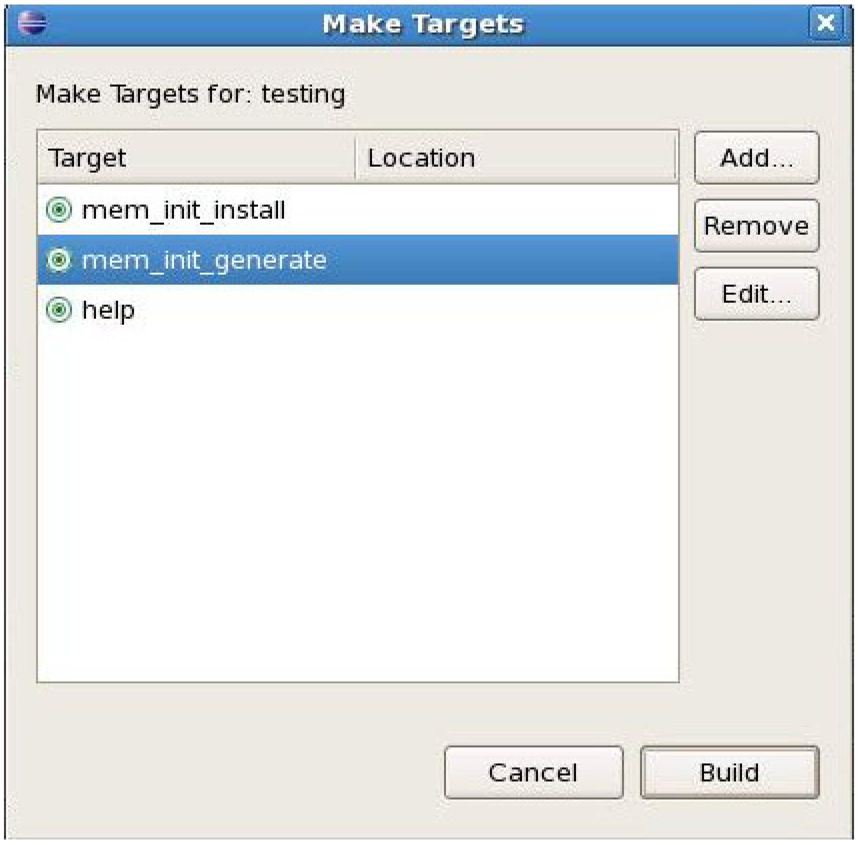
Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
ext_flash_avl_mem	0x16000020 - 0x17FFFFFF	ext_flash_avl_mem	33554400	33554464
reset	0x16000000 - 0x1600001F	ext_flash_avl_mem	32	33554432
ext_flash_avl_mem_BEFORE_RESET	0x14000000 - 0x15FFFFFF	ext_flash_avl_mem	33554432	0
mem_if_ddr3_emif_0	0x08000120 - 0x0FFFFFFF	mem_if_ddr3_emif_0	134217440	288
mem_if_ddr3_emif_0_BEFORE_EXCEPTION	0x08000000 - 0x0800011F	mem_if_ddr3_emif_0	288	0
tb_ram_1k	0x00000000 - 0x000003FF	tb_ram_1k	1024	0

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.

Figure 72: Selecting mem_init_generate in Make Targets



5. The “mem_init_generate” macro will create two HEX files; **<OCRAM_name>.hex** and **<QSPIFlash_name>.hex**. Use **<QSPIFlash_name>.hex** for boot option 4 and 5 as you are booting from QSPI flash, not from OCRAM.

Related Information

Software Programmer Object File (.pof) Generation on page 65

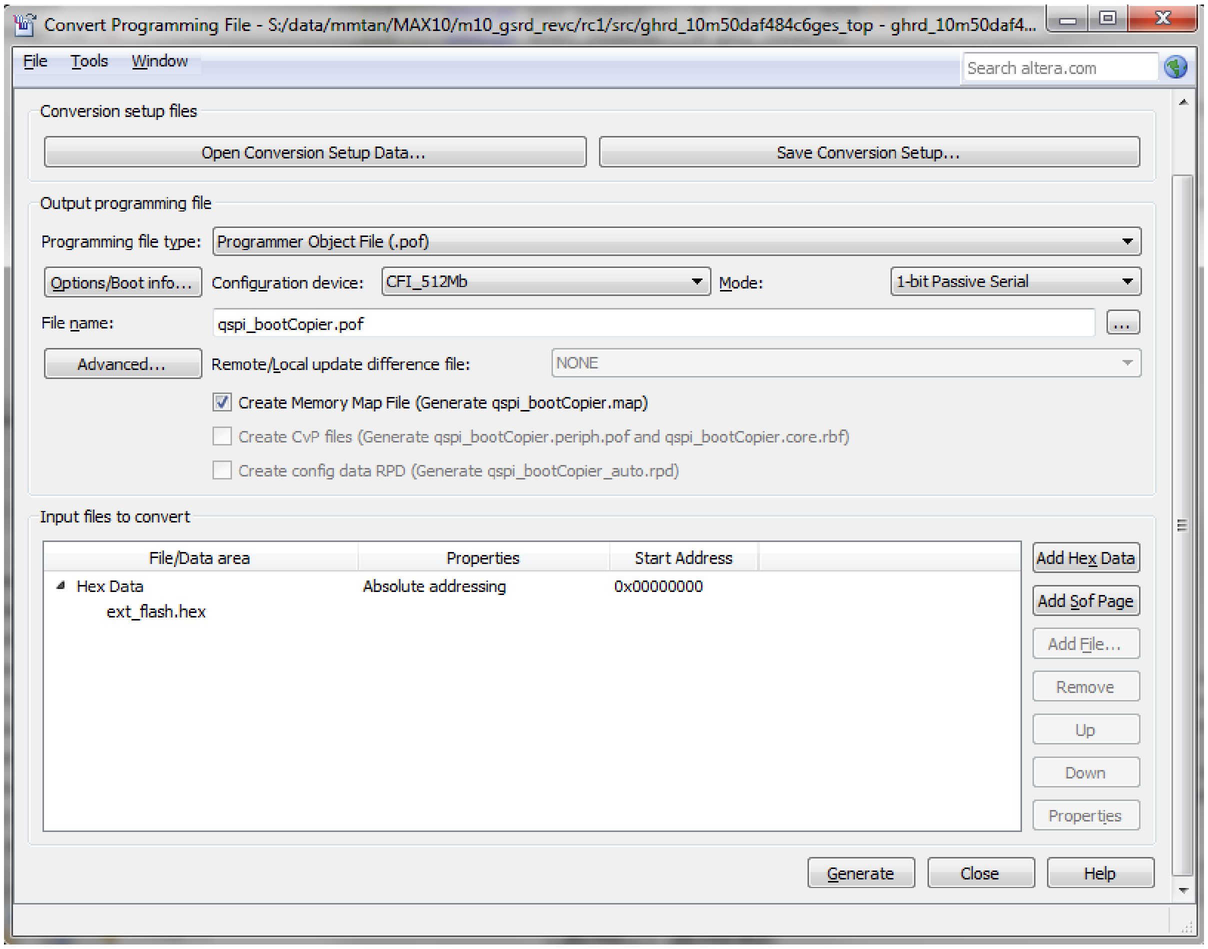
Software Programmer Object File (.pof) Generation

Important: The **quartus.ini** file with`PGMIO_SWAP_HEX_BYTE_DATA=ON` content is required to byteswap the programming file during the POF generation. Please create the **quartus.ini** file or use the **quartus.ini** available in the related information and place it under Quartus II tool directory or project directory before you proceed.

1. In Quartus II, click on **Convert Programming Files (.pof)** from the **File** tab.
2. Choose **Programmer Object File** as **Programming file type**.
3. Set **Mode** to **1-bit Passive Serial**.
4. Set **Configuration device** to **CFI_512Mb**.
5. Change the **File name** to the desired path and name.

6. Remove the SOF **Page_0**.
7. Click on **Add HEX Data**, choose the HEX file generated in **HEX Generation** section. Select **Absolute Addressing** and click **OK**.
8. Click **Generate** to create the .pof file.

Figure 73: HEX to POF file Conversion Settings


Convert Programming File - S:/data/mmtan/MAX10/m10_gsrd_revc/rc1/src/ghrd_10m50daf484c6ges_top - ghrd_10m50daf4...

Conversion setup files

Output programming file
Programming file type: Programmer Object File (.pof)
Configuration device: CFI_512Mb
Mode: 1-bit Passive Serial
File name: qspi_bootCopier.pof
Remote/Local update difference file: NONE
☑ Create Memory Map File (Generate qspi_bootCopier.map)
☐ Create CvP files (Generate qspi_bootCopier.periph.pof and qspi_bootCopier.core.rbf)
☐ Create config data RPD (Generate qspi_bootCopier_auto.rpd)

Input files to convert

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

Related Information

- **HEX File Generation** on page 64
- **POF file Programming into QSPI flash** on page 66
- **Quartus.ini file**

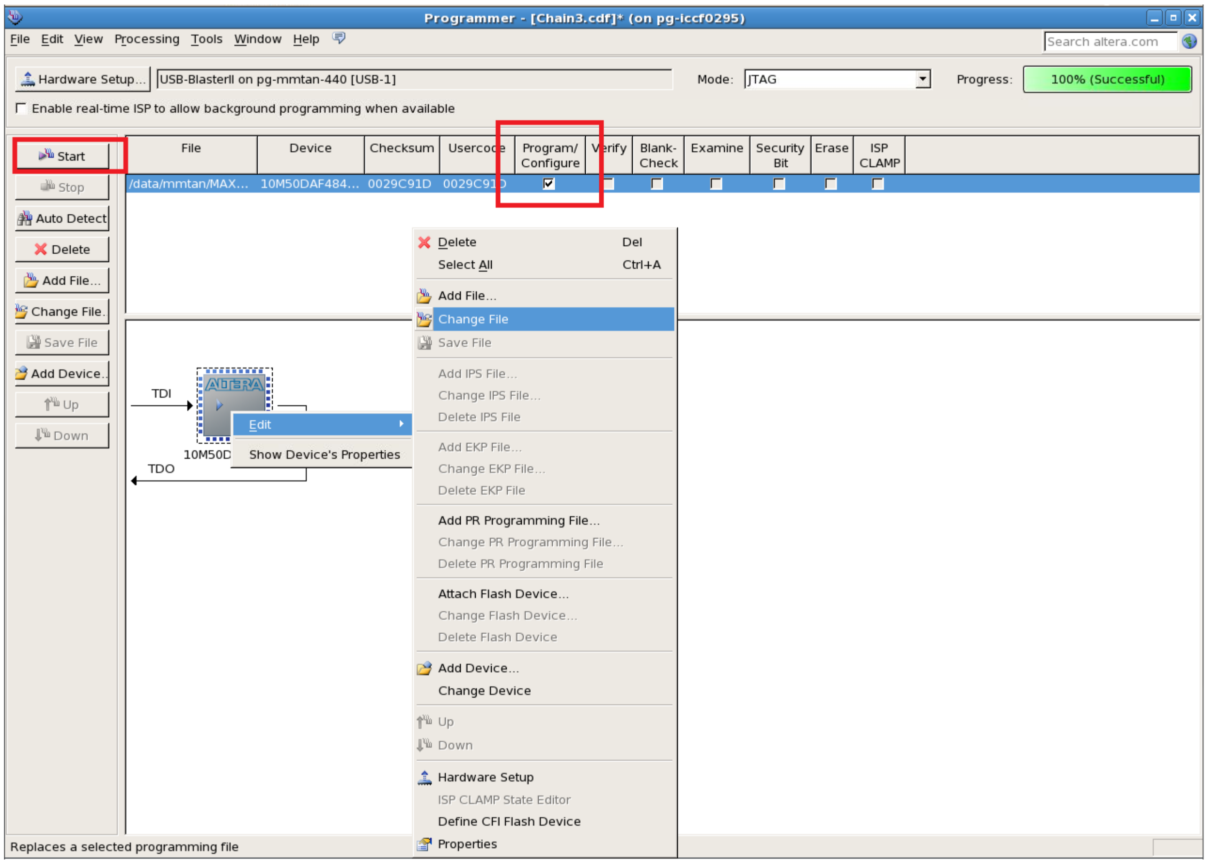
POF file Programming into QSPI flash

1. Programming Parallel Flash Loader into MAX10 Device

   Note: You need to program the parallel flash loader into the MAX 10 device before programming the QSPI flash.

a. Create Parallel Flash Loader for MAX 10 FPGA in the Quartus II. Assign QSPI pins based on your design. Compile the project to obtain **max10_qpfl.sof** file.
b. Open Quartus II programmer from the Quartus II tool (**Tools** -> **Programmer**).
c. Make sure the **Hardware Setup** is set to your **USB blaster**.
d. Click on **Auto Detect**, select your MAX10 FPGA and select **OK**.
e. Right click on the **MAX 10 FPGA** and select **Edit** -> **Change File**. Choose the **max_qpfl.sof** file.
f. Check MAX 10 device under **Program/Configure** and click **Start** to start programming.
g. Click on **Auto Detect** after **max10_qpfl.sof** is successfully programmed. Click **Yes** if you are asked to overwrite the existing settings. A new QSPI flash device will be shown on the screen.

Figure 74: Programming Parallel Flash Loader Programmer Settings


Programmer - [Chain3.cdf]* (on pg-iccf0295)

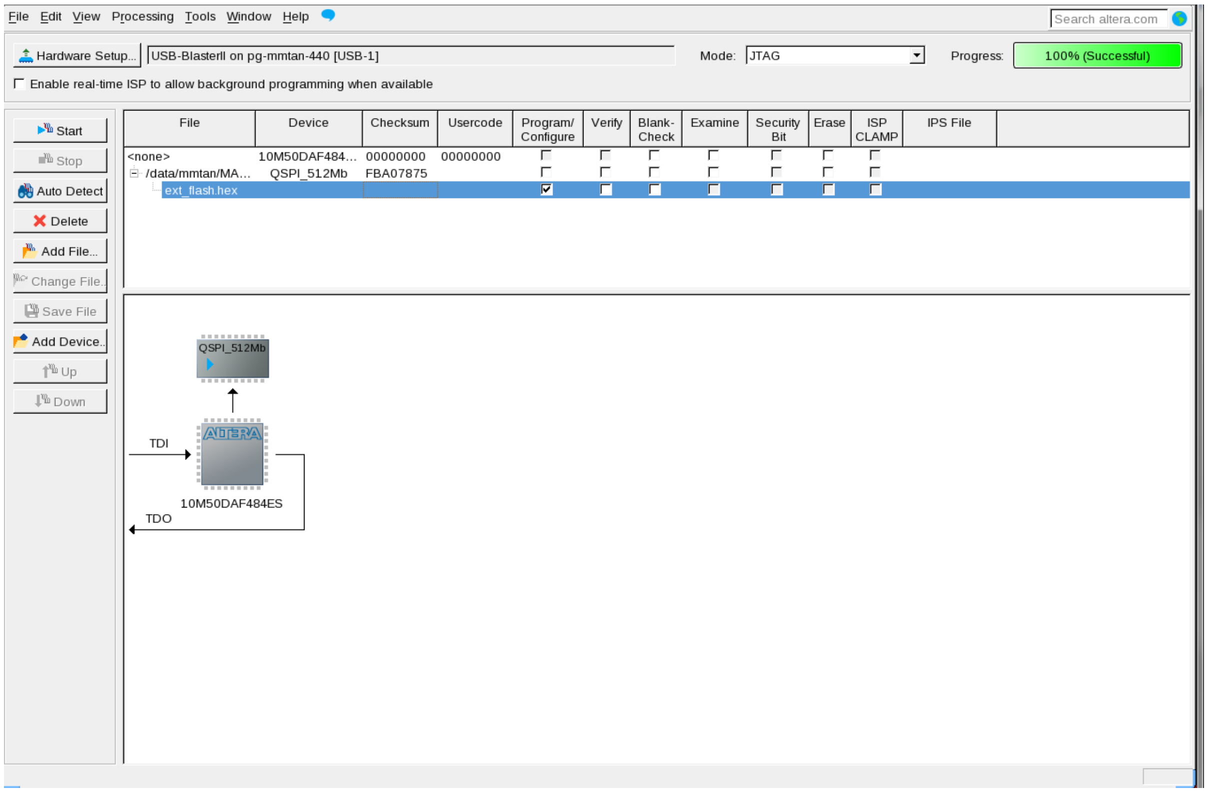
Hardware Setup... USB-BlasterII on pg-mmtan-440 [USB-1]
Mode: JTAG
Progress: 100% (Successful)
☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
/data/mmtan/MAX...	10M50DAF484...	0029C91D	0029C91D	☑	☐	☐	☐	☐	☐	☐

Replaces a selected programming file

2. Programming HEX image into QSPI Flash
   a. Right click on the **QSPI device** and select **Edit** -> **Change File**. Choose the generated POF file from Software POF Generation section.
   b. Check the HEX file under **Program/Configure** column and click **Start** to start programming.

Figure 75: HEX Image Programming into QSPI Flash



USB-BlasterII on pg-mmtan-440 [USB-1] Mode: JTAG Progress: 100% (Successful)

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

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP	IPS File
<none>	10M50DAF484...	00000000	00000000	☐	☐	☐	☐	☐	☐	☐	
/data/mmtan/MA...	QSPI_512Mb	FBA07875		☐	☐	☐	☐	☐	☐	☐	
ext_flash.hex				✓	☐	☐	☐	☐	☐	☐	

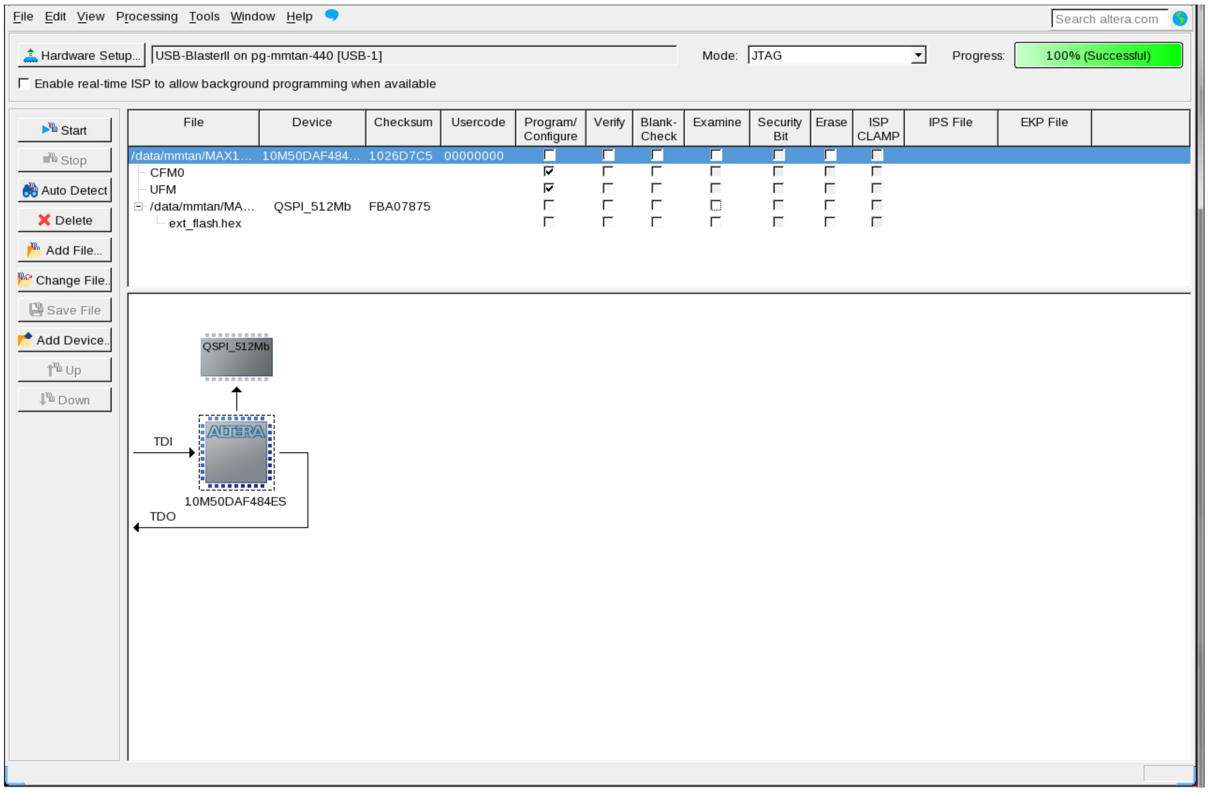
Related Information

Software Programmer Object File (.pof) Generation on page 65

Programming Hardware Design POF File into the MAX10 FPGA

1. After you successfully programmed HEX data into Quad SPI flash, right click on the **MAX 10 FPGA** and select **Edit** -> **Change File**. Choose the downloaded POF file generated from Quartus II project compilation.
2. Check the MAX10's **CFM0** and **UFM** under **Program/Configure** column and click **Start** to start programming.

Figure 76: POF Image Programming into QSPI Flash


Hardware Setup... USB-BlasterII on pg-mmtan-440 [USB-1] Mode: JTAG Progress: 100% (Successful)

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

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP	IPS File	EKP File
/data/mmtan/MAX1…	10M50DAF484…	1026D7C5	00000000	☐	☐	☐	☐	☐	☐	☐		
CFM0				✓	☐	☐	☐	☐	☐	☐		
UFM				✓	☐	☐	☐	☐	☐	☐		
/data/mmtan/MA…	QSPI_512Mb	FBA07875		☐	☐	☐	☐	☐	☐	☐		
ext_flash.hex				☐	☐	☐	☐	☐	☐	☐		

Related Information

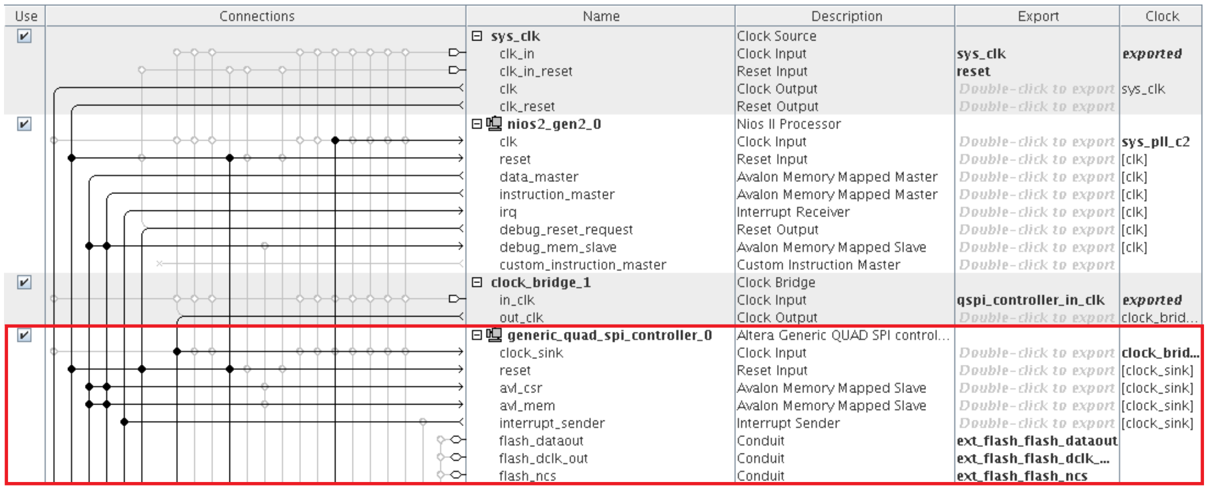
[Quartus II Software Settings](#) on page 59

Dual Compressed Images Bootable System Guideline

Design Creation

1. Create your Nios II processor project using Quartus II and Qsys.
2. Ensure the Generic Quad SPI Controller IP is added into your Qsys system. Refer to the diagram below for IP connection in Qsys.

Figure 77: Connection for Generic Quad SPI Controller IP



Use	Connections	Name	Description	Export	Clock
✓		⊟ sys_clk	Clock Source		
		clk_in	Clock Input	sys_clk	exported
		clk_in_reset	Reset Input	reset	
		clk	Clock Output	Double-click to export	sys_clk
		clk_reset	Reset Output	Double-click to export	
✓		⊟ nios2_gen2_0	Nios II Processor		
		clk	Clock Input	Double-click to export	sys_pll_c2
		reset	Reset Input	Double-click to export	[clk]
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]
		irq	Interrupt Receiver	Double-click to export	[clk]
		debug_reset_request	Reset Output	Double-click to export	[clk]
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]
		custom_instruction_master	Custom Instruction Master	Double-click to export	
✓		⊟ clock_bridge_1	Clock Bridge		
		in_clk	Clock Input	qspi_controller_in_clk	exported
		out_clk	Clock Output	Double-click to export	clock_brid...
✓		⊟ generic_quad_spi_controller_0	Altera Generic QUAD SPI control...		
		clock_sink	Clock Input	Double-click to export	clock_brid...
		reset	Reset Input	Double-click to export	[clock_sink]
		avl_csr	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
		avl_mem	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
		interrupt_sender	Interrupt Sender	Double-click to export	[clock_sink]
		flash_dataout	Conduit	ext_flash_flash_dataout	
		flash_dclk_out	Conduit	ext_flash_flash_dclk_...	
		flash_ncs	Conduit	ext_flash_flash_ncs	

3. Ensure Altera Dual Configuration IP is instantiated in your Qsys system to enable dual images configuration.

Note: The maximum input clock for Generic Quad SPI Controller IP is 25 MHz. The input clock must not exceed this maximum value.

Qsys Settings

1. In the Nios II Processor parameter editor, set the reset vector memory and exception vector memory based on the boot options below:

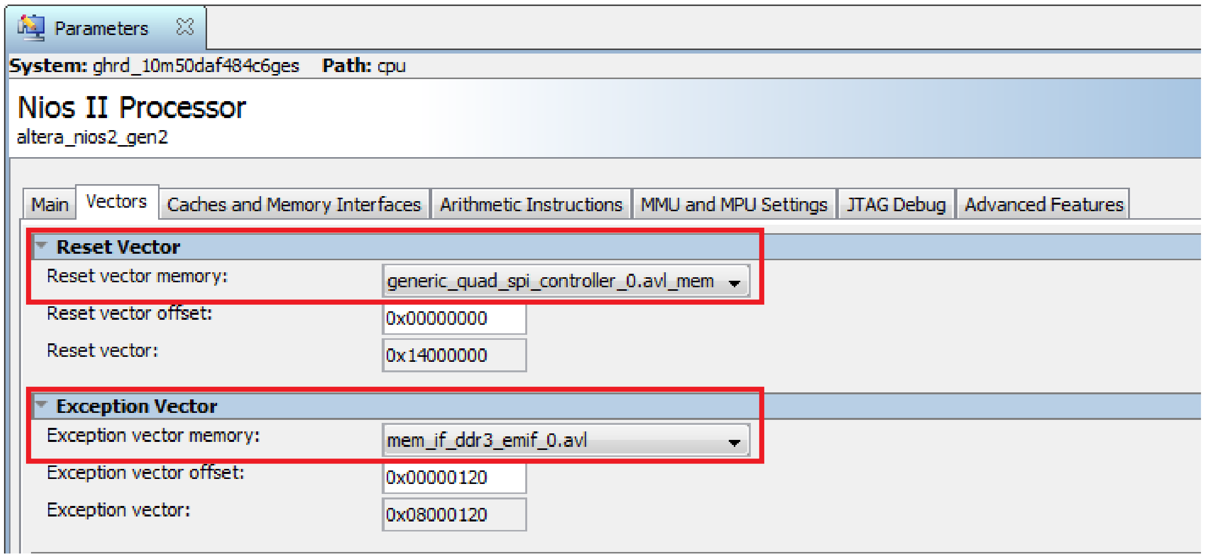
Boot Option	Reset vector memory	Exception vector memory
Option 4a [16][17]	QSPI flash	OCRAM/ External RAM
Option 4b[16]	QSPI flash	QSPI flash
Option 5	QSPI flash	OCRAM/ External RAM

[16]) You can set the exception vector for Boot Option 4 to OCRAM/ External RAM (Option 4a) or QSPI Flash (Option 4b) according to your design preference.

[17] Boot option 4a which sets exception vector memory to OCRAM/External RAM is recommended to make the interrupt processing faster.

Figure 78: Nios II Parameter Editor Settings Boot Option 4a and 5

Note: **mem_if_ddr3_emif_0.avl** is the external memory (DDR3) used in this example.



Parameters

System: ghrd_10m50daf484c6ges **Path:** cpu

Nios II Processor
altera_nios2_gen2

Main | Vectors | Caches and Memory Interfaces | Arithmetic Instructions | MMU and MPU Settings | JTAG Debug | Advanced Features

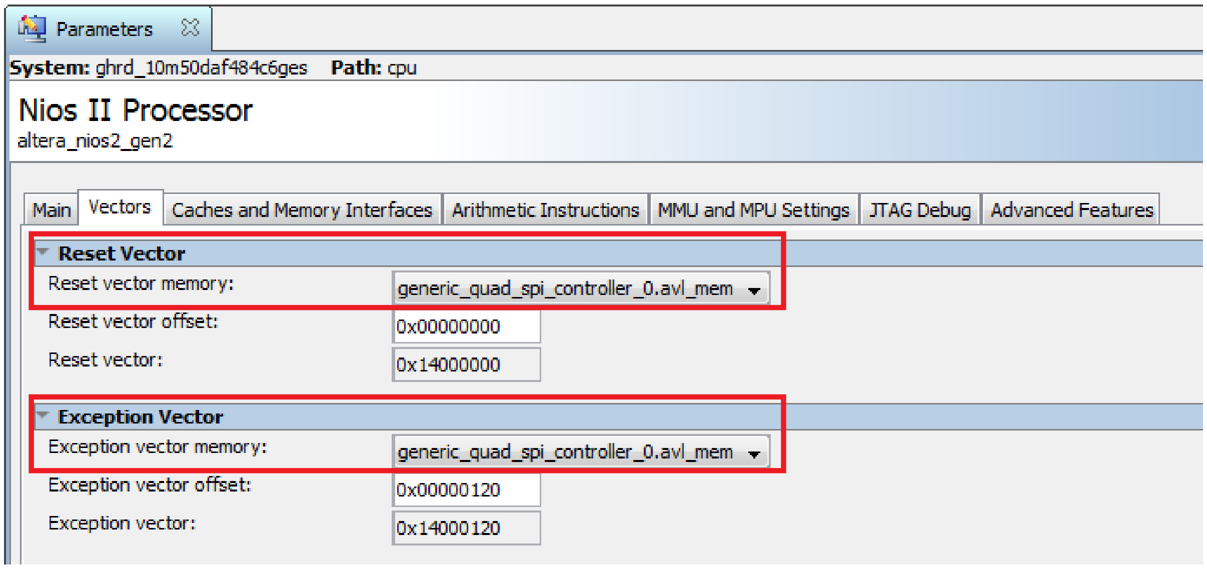
Reset Vector

Reset vector memory:	generic_quad_spi_controller_0.avl_mem
Reset vector offset:	0x00000000
Reset vector:	0x14000000

Exception Vector

Exception vector memory:	mem_if_ddr3_emif_0.avl
Exception vector offset:	0x00000120
Exception vector:	0x08000120

Figure 79: Nios II Parameter Editor Settings Boot Option 4b



Parameters

System: ghrd_10m50daf484c6ges **Path:** cpu

Nios II Processor
altera_nios2_gen2

Main | Vectors | Caches and Memory Interfaces | Arithmetic Instructions | MMU and MPU Settings | JTAG Debug | Advanced Features

Reset Vector

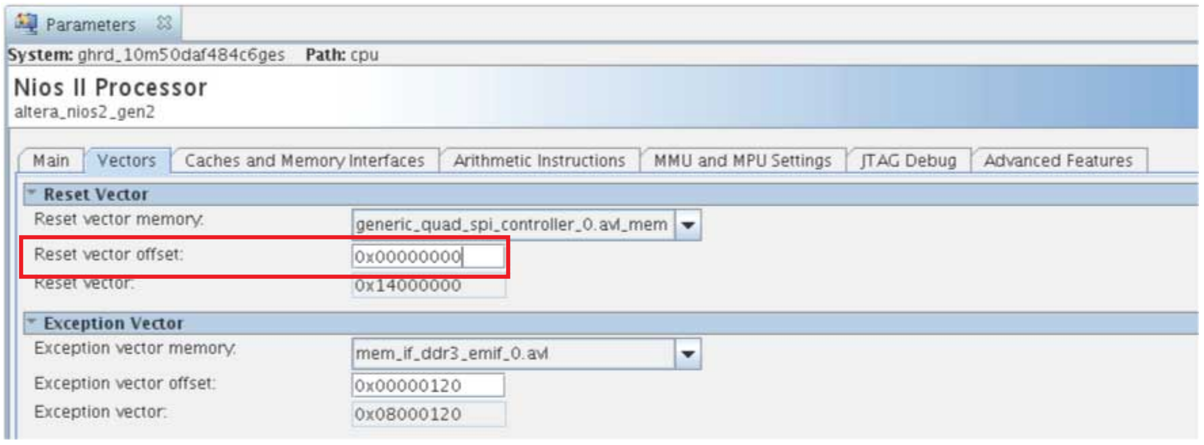
Reset vector memory:	generic_quad_spi_controller_0.avl_mem
Reset vector offset:	0x00000000
Reset vector:	0x14000000

Exception Vector

Exception vector memory:	generic_quad_spi_controller_0.avl_mem
Exception vector offset:	0x00000120
Exception vector:	0x14000120

2. There are two Nios II application data (HEX file) stored into the QSPI for supporting dual configuration images. Reset vector memory offset has to set correctly for the configuration images to call up the correct HEX data.
3. Set Reset vector memory offset of the Nios II Processor in first Qsys design to address `0x00000000`.
4. Set Reset vector memory offset of the Nios II Processor in second Qsys design to another address to avoid overlapping. For example: Address `0x02000000` which is half of the QSPI memory size (512Mb).

Figure 80: Reset Vector Offset Setting for First Qsys design



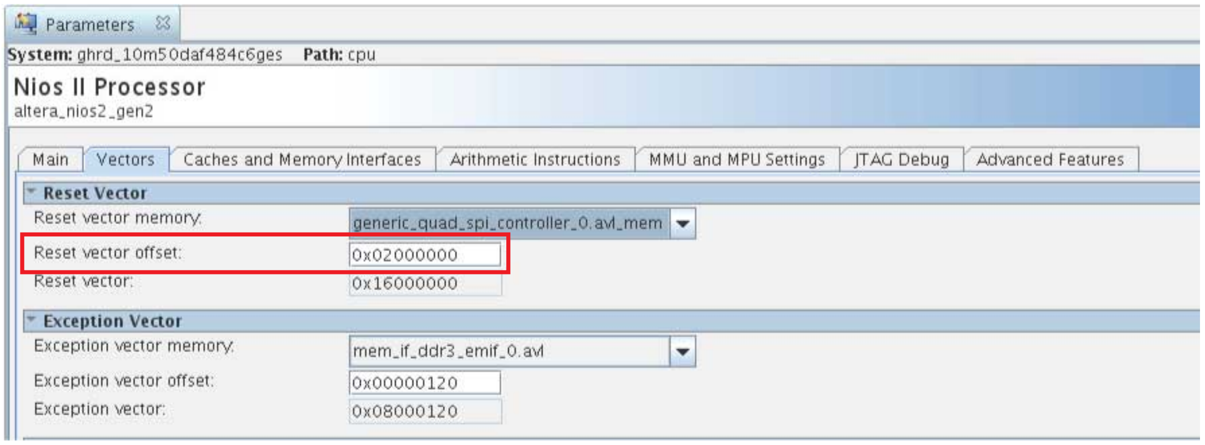
System: ghrd_10m50daf484c6ges Path: cpu

Nios II Processor
altera_nios2_gen2

Reset Vector
Reset vector memory: generic_quad_spi_controller_0.avl_mem
Reset vector offset: 0x00000000
Reset vector: 0x14000000

Exception Vector
Exception vector memory: mem_if_ddr3_emif_0.avl
Exception vector offset: 0x00000120
Exception vector: 0x08000120

Figure 81: Reset Vector Offset Setting for Second Qsys design



System: ghrd_10m50daf484c6ges Path: cpu

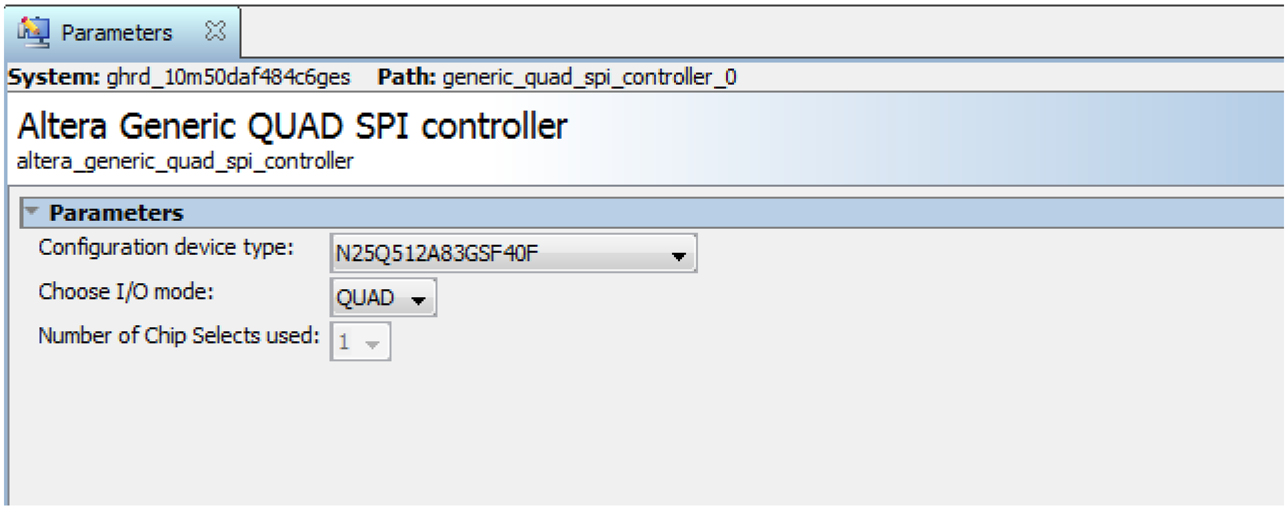
Nios II Processor
altera_nios2_gen2

Reset Vector
Reset vector memory: generic_quad_spi_controller_0.avl_mem
Reset vector offset: 0x02000000
Reset vector: 0x16000000

Exception Vector
Exception vector memory: mem_if_ddr3_emif_0.avl
Exception vector offset: 0x00000120
Exception vector: 0x08000120

5. Open **Altera Generic Quad SPI** controller parameter editor. Change the **Configuration device type** to the QSPI flash selection and make sure the **I/O mode** is set to **QUAD**.

Figure 82: Altera Generic Quad SPI Controller Parameter Settings



System: ghrd_10m50daf484c6ges **Path:** generic_quad_spi_controller_0

Altera Generic QUAD SPI controller
altera_generic_quad_spi_controller

Parameters

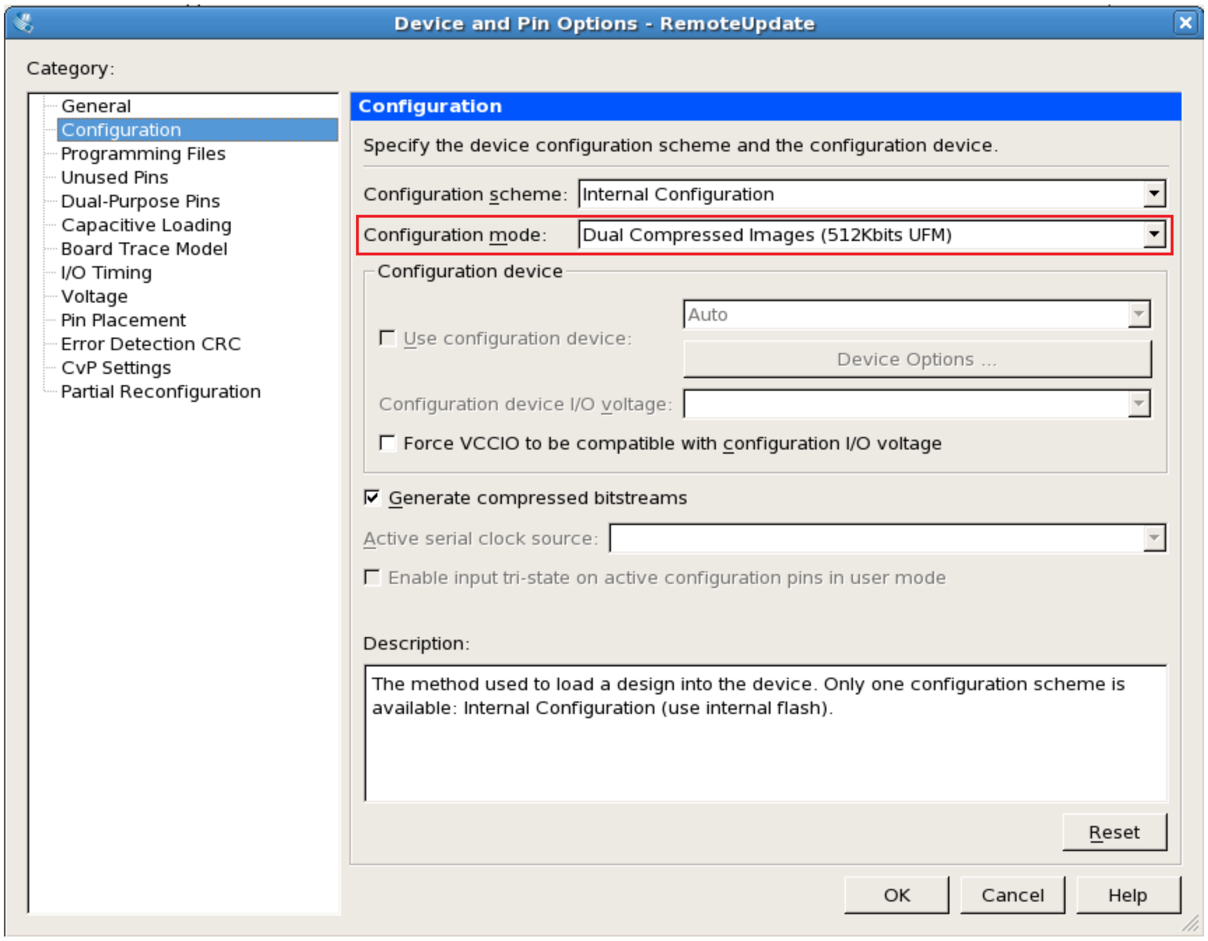
Configuration device type: N25Q512A83GSF40F
Choose I/O mode: QUAD
Number of Chip Selects used: 1

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

Quartus II Software Settings

1. In the Quartus II software, click on **Assignment** -> **Device** -> **Device and Pin Options** -> **Configuration**. Set **Configuration mode** to **Dual Compressed Images**.

Figure 83: Configuration Mode Selection in Quartus II Software


Device and Pin Options - RemoteUpdate

Category:

- General
- Configuration
- Programming Files
- Unused Pins
- Dual-Purpose Pins
- Capacitive Loading
- Board Trace Model
- I/O Timing
- Voltage
- Pin Placement
- Error Detection CRC
- CvP Settings
- Partial Reconfiguration

Configuration

Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Dual Compressed Images (512Kbits UFM)

Configuration device

☐ Use configuration device: Auto

Device Options ...

Configuration device I/O voltage:

☐ Force VCCIO to be compatible with configuration I/O voltage

☑ Generate compressed bitstreams

Active serial clock source:

☐ Enable input tri-state on active configuration pins in user mode

Description:

The method used to load a design into the device. Only one configuration scheme is available: Internal Configuration (use internal flash).

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.

Related Information

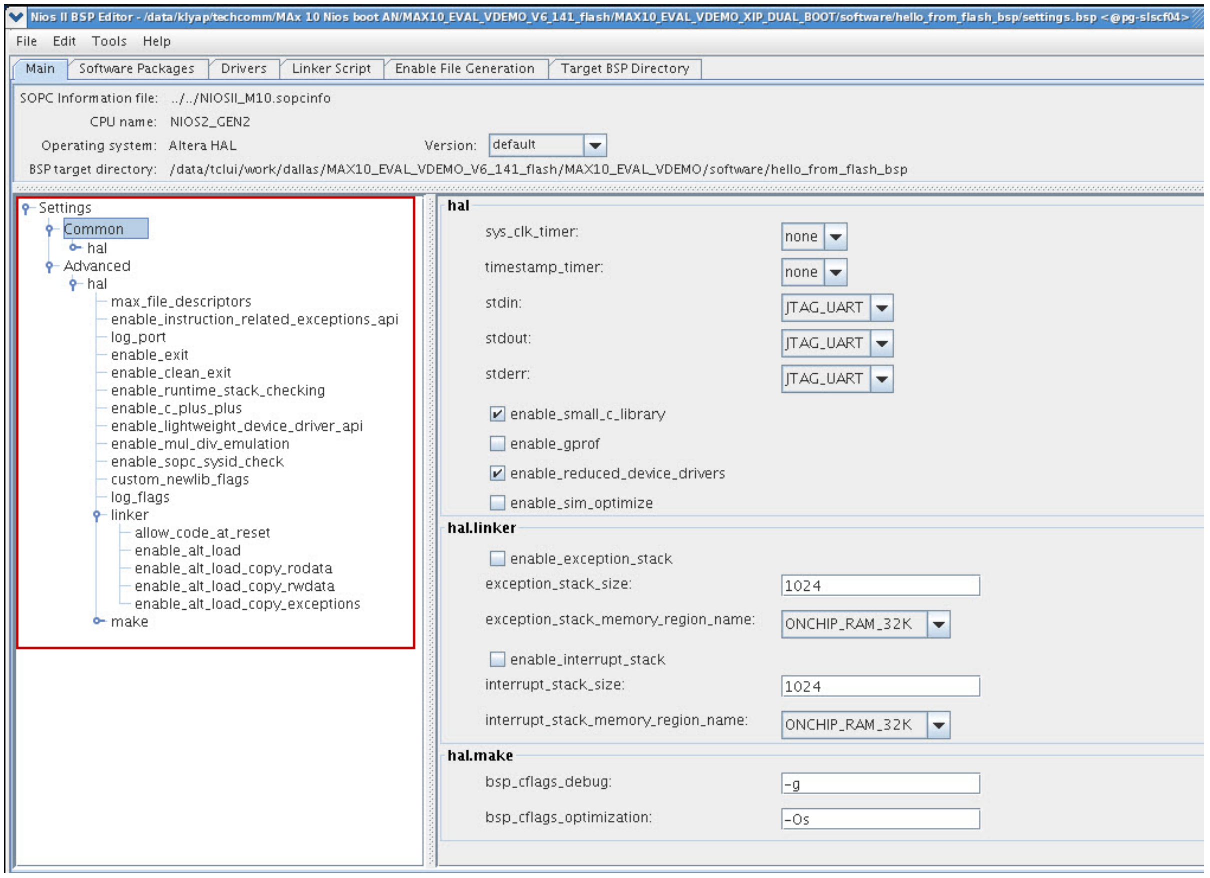
Programming Hardware Design POF File into the MAX10 FPGA on page 83

BSP Editor Settings

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

1. In the Nios II SBT tool, right click on your BSP project in the **Project Explorer** window. Select **Nios II** > **BSP Editor...** to open the **Nios II BSP Editor**.
2. In Nios II BSP Editor, click on **Advanced** tab under **Settings**.
3. Click on **hal** to expand the list.
4. Click on **linker** to expand the list.

Figure 84: BSP Editor Settings


Nios II BSP Editor - /data/klyap/techcomm/MAx 10 Nios boot AN/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO_XIP_DUAL_BOOT/software/hello_from_flash_bsp/settings.bsp <@pg-slscf04>

SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL Version: default
BSP target directory: /data/tclui/work/dallas/MAX10_EVAL_VDEMO_V6_141_flash/MAX10_EVAL_VDEMO/software/hello_from_flash_bsp

```
Settings
  Common
    hal
  Advanced
    hal
      max_file_descriptors
      enable_instruction_related_exceptions_api
      log_port
      enable_exit
      enable_clean_exit
      enable_runtime_stack_checking
      enable_c_plus_plus
      enable_lightweight_device_driver_api
      enable_mul_div_emulation
      enable_sopc_sysid_check
      custom_newlib_flags
      log_flags
      linker
        allow_code_at_reset
        enable_alt_load
        enable_alt_load_copy_rodata
        enable_alt_load_copy_rwdata
        enable_alt_load_copy_exceptions
      make
```

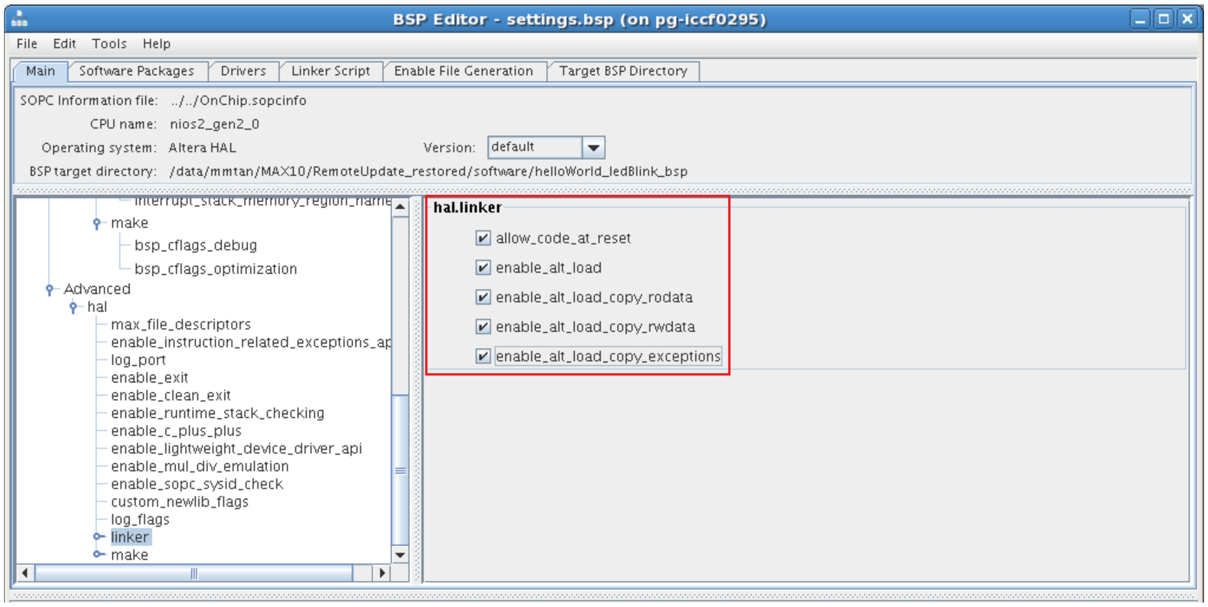
hal
- sys_clk_timer: none
- timestamp_timer: none
- stdin: JTAG_UART
- stdout: JTAG_UART
- stderr: JTAG_UART
- [✓] enable_small_c_library
- [ ] enable_gprof
- [✓] enable_reduced_device_drivers
- [ ] enable_sim_optimize

hal.linker
- [ ] enable_exception_stack
- exception_stack_size: 1024
- exception_stack_memory_region_name: ONCHIP_RAM_32K
- [ ] enable_interrupt_stack
- interrupt_stack_size: 1024
- interrupt_stack_memory_region_name: ONCHIP_RAM_32K

hal.make
- bsp_cflags_debug: -g
- bsp_cflags_optimization: -Os

5. Based on the boot option used, do one of the following:
    - For boot option 4a, if exception vector memory is set to OCRAM or External RAM, enable the following:
        - **allow_code_at_reset**
        - **enable_alt_load**
        - **enable_alt_load_copy_rodata**
        - **enable_alt_load_copy_rwdata**
        - **enable_alt_load_copy_exceptions**
    - For boot option 4b, if exception vector memory is set to QSPI flash, enable the following:
        - **allow_code_at_reset**
        - **enable_alt_load**
        - **enable_alt_load_copy_rodata**
        - **enable_alt_load_copy_rwdata**
    - For boot option 5, leave all the hal.linker settings unchecked.

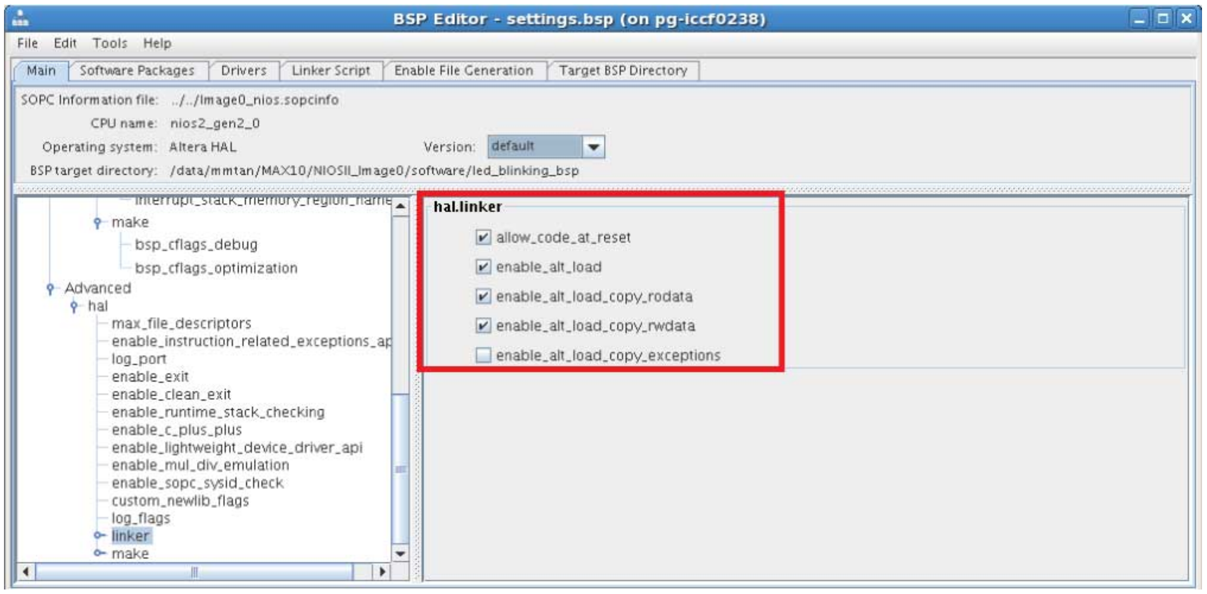
Figure 85: Advanced.hal.linker Settings for Boot Option 4a



SOPC Information file: ../../OnChip.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL
Version: default
BSP target directory: /data/mmtan/MAX10/RemoteUpdate_restored/software/helloWorld_ledBlink_bsp

hal.linker
- [✓] allow_code_at_reset
- [✓] enable_alt_load
- [✓] enable_alt_load_copy_rodata
- [✓] enable_alt_load_copy_rwdata
- [✓] enable_alt_load_copy_exceptions

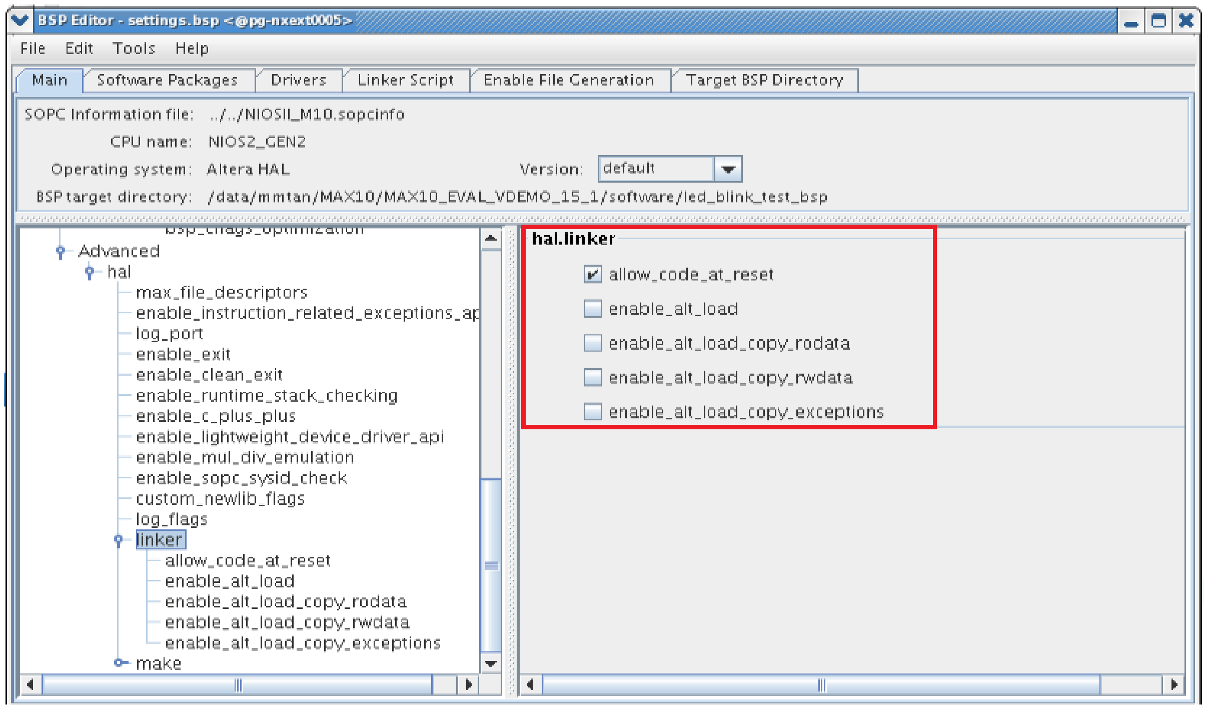
Figure 86: Advanced.hal.linker Settings for Boot Option 4b



SOPC Information file: ../../Image0_nios.sopcinfo
CPU name: nios2_gen2_0
Operating system: Altera HAL
Version: default
BSP target directory: /data/mmtan/MAX10/NIOSII_Image0/software/led_blinking_bsp

hal.linker
- [✓] allow_code_at_reset
- [✓] enable_alt_load
- [✓] enable_alt_load_copy_rodata
- [✓] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

Figure 87: Advanced.hal.linker Settings for Boot Option 5


BSP Editor - settings.bsp <@pg-nxext0005>

File Edit Tools Help

Main | Software Packages | Drivers | Linker Script | Enable File Generation | Target BSP Directory

SOPC Information file: ../../NIOSII_M10.sopcinfo
CPU name: NIOS2_GEN2
Operating system: Altera HAL Version: default
BSP target directory: /data/mmtan/MAX10/MAX10_EVAL_VDEMO_15_1/software/led_blink_test_bsp

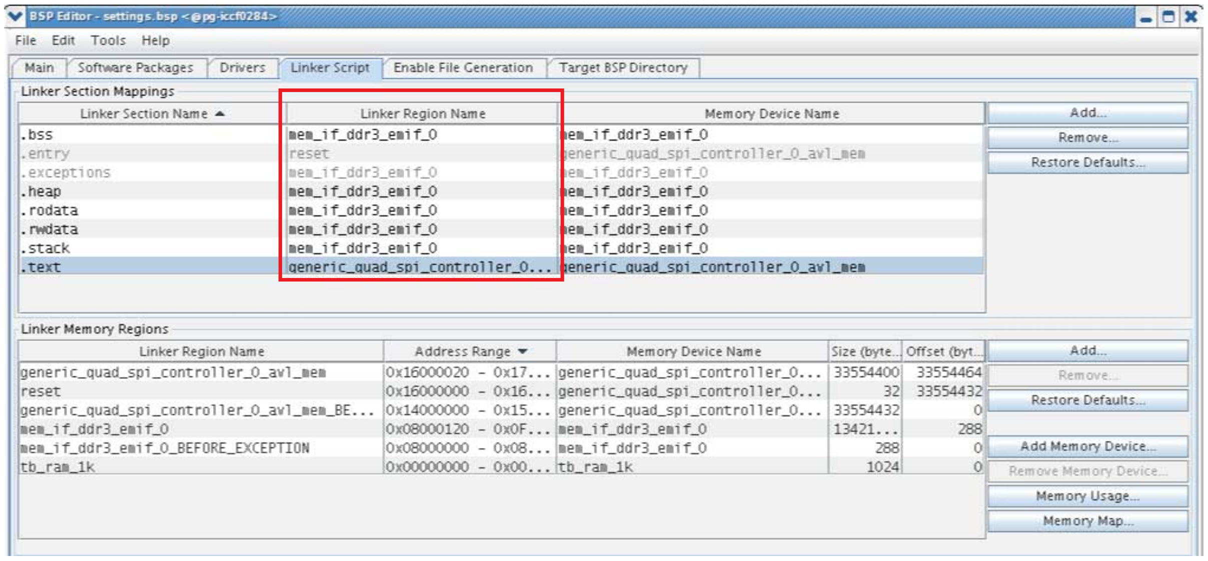
```
bsp_cflags_optimization
Advanced
  hal
    max_file_descriptors
    enable_instruction_related_exceptions_ap
    log_port
    enable_exit
    enable_clean_exit
    enable_runtime_stack_checking
    enable_c_plus_plus
    enable_lightweight_device_driver_api
    enable_mul_div_emulation
    enable_sopc_sysid_check
    custom_newlib_flags
    log_flags
    linker
      allow_code_at_reset
      enable_alt_load
      enable_alt_load_copy_rodata
      enable_alt_load_copy_rwdata
      enable_alt_load_copy_exceptions
    make
```

hal.linker

- [✓] allow_code_at_reset
- [ ] enable_alt_load
- [ ] enable_alt_load_copy_rodata
- [ ] enable_alt_load_copy_rwdata
- [ ] enable_alt_load_copy_exceptions

6. Click on **Linker Script** tab in the Nios II BSP Editor.
7. Based on the boot option used, do one of the following:
   - For boot option 4a and 4b, set the **.text** item in the **Linker Section Name** to the QSPI flash in the **Linker Region Name**. Set the rest of the items in the **Linker Section Name** list to the Altera On-chip Memory (OCRAM) or external RAM.
   - For boot option 5, set all of the items in the **Linker Section Name** list to Altera On-chip Memory (OCRAM) or external RAM.

Figure 88: Linker Region Settings for Boot Option 4a and 4b


BSP Editor - settings.bsp <@pg-iccf0284>
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Add... Remove... Restore Defaults...
Add... Remove... Restore Defaults... Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...

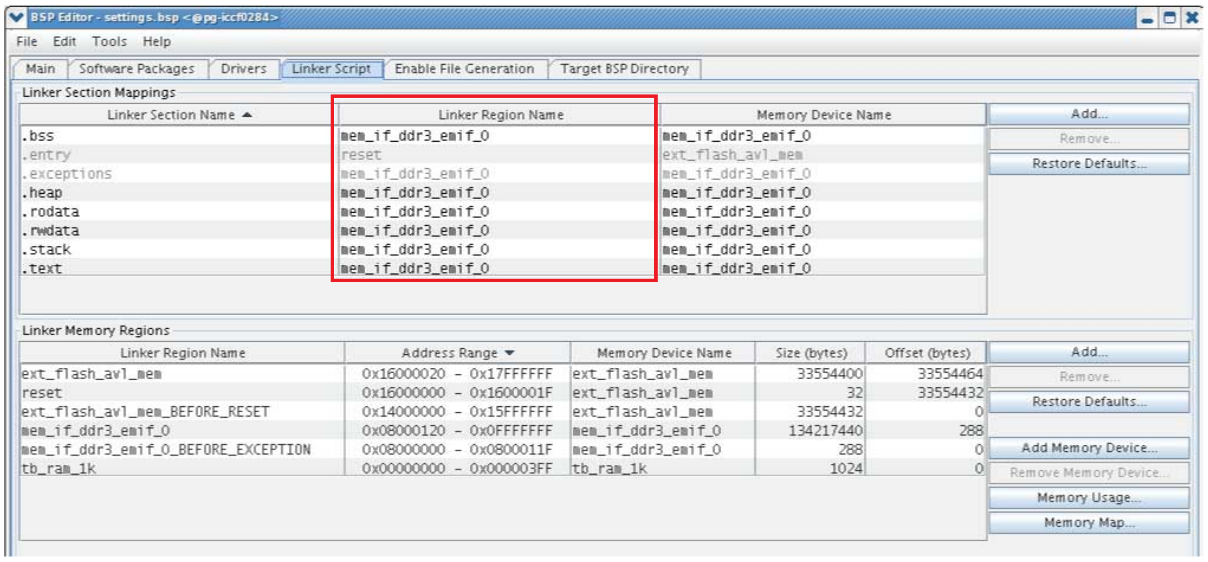
Linker Section Mappings

Linker Section Name ▲	Linker Region Name	Memory Device Name
.bss	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.entry	reset	generic_quad_spi_controller_0_avl_mem
.exceptions	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.heap	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rodata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rwdata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.stack	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.text	generic_quad_spi_controller_0...	generic_quad_spi_controller_0_avl_mem

Linker Memory Regions

Linker Region Name	Address Range ▼	Memory Device Name	Size (byte...	Offset (byt...
generic_quad_spi_controller_0_avl_mem	0x16000020 - 0x17...	generic_quad_spi_controller_0...	33554400	33554464
reset	0x16000000 - 0x16...	generic_quad_spi_controller_0...	32	33554432
generic_quad_spi_controller_0_avl_mem_BE...	0x14000000 - 0x15...	generic_quad_spi_controller_0...	33554432	0
mem_if_ddr3_emif_0	0x08000120 - 0x0F...	mem_if_ddr3_emif_0	13421...	288
mem_if_ddr3_emif_0_BEFORE_EXCEPTION	0x08000000 - 0x08...	mem_if_ddr3_emif_0	288	0
tb_ram_1k	0x00000000 - 0x00...	tb_ram_1k	1024	0

Figure 89: Linker Region Settings for Boot Option 5


BSP Editor - settings.bsp <@pg-iccf0284>
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Add... Remove... Restore Defaults...
Add... Remove... Restore Defaults... Add Memory Device... Remove Memory Device... Memory Usage... Memory Map...

Linker Section Mappings

Linker Section Name ▲	Linker Region Name	Memory Device Name
.bss	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.entry	reset	ext_flash_avl_mem
.exceptions	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.heap	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rodata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.rwdata	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.stack	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0
.text	mem_if_ddr3_emif_0	mem_if_ddr3_emif_0

Linker Memory Regions

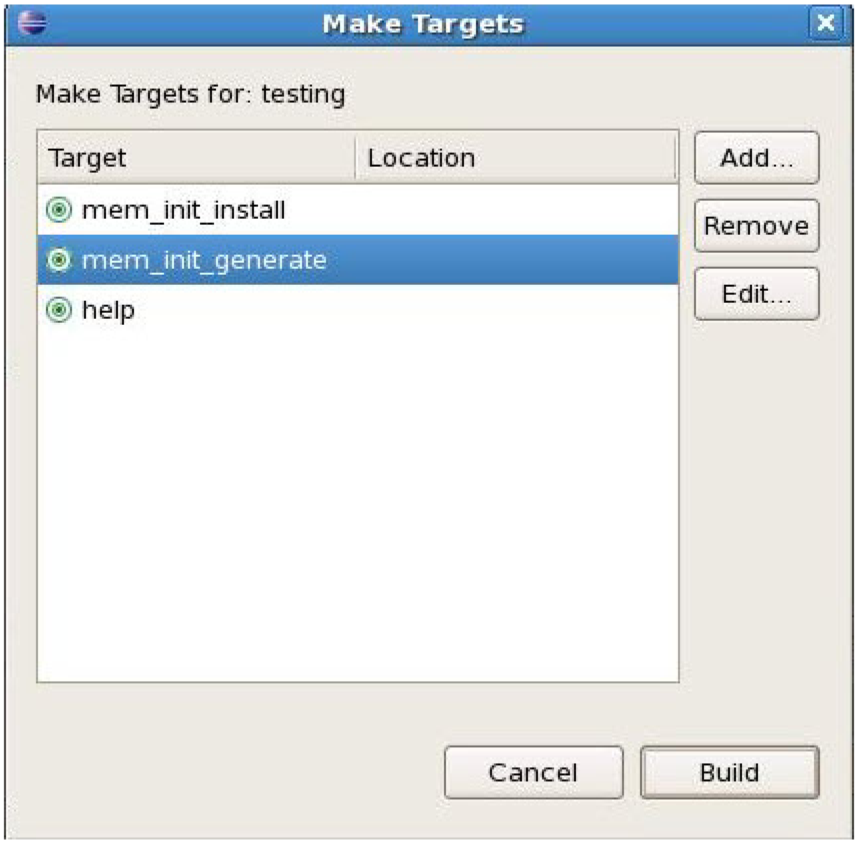
Linker Region Name	Address Range ▼	Memory Device Name	Size (bytes)	Offset (bytes)
ext_flash_avl_mem	0x16000020 - 0x17FFFFFF	ext_flash_avl_mem	33554400	33554464
reset	0x16000000 - 0x1600001F	ext_flash_avl_mem	32	33554432
ext_flash_avl_mem_BEFORE_RESET	0x14000000 - 0x15FFFFFF	ext_flash_avl_mem	33554432	0
mem_if_ddr3_emif_0	0x08000120 - 0x0FFFFFFF	mem_if_ddr3_emif_0	134217440	288
mem_if_ddr3_emif_0_BEFORE_EXCEPTION	0x08000000 - 0x0800011F	mem_if_ddr3_emif_0	288	0
tb_ram_1k	0x00000000 - 0x000003FF	tb_ram_1k	1024	0

HEX File Generation

Note: The following steps apply to both first and second Nios II applications.

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.

Figure 90: Selecting mem_init_generate in Make Targets

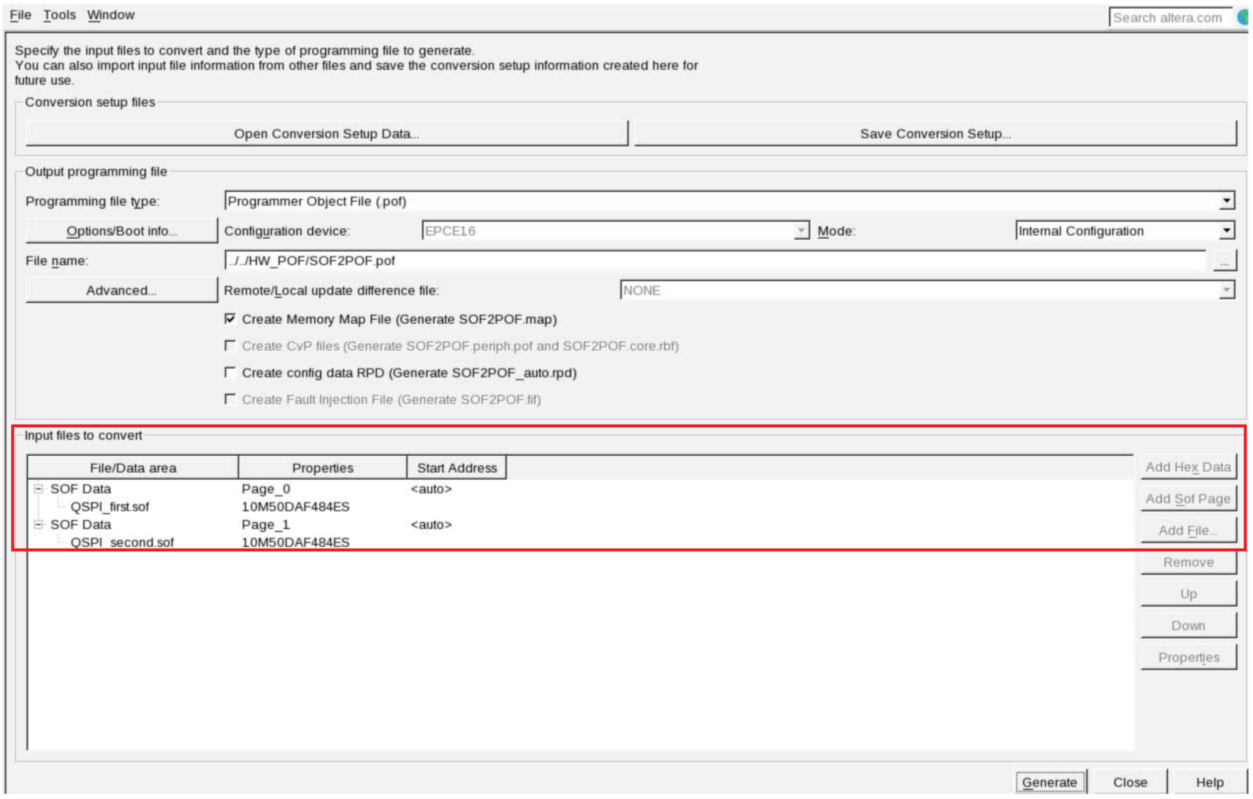


5. The “mem_init_generate” macro will create two HEX files; **<OCRAM_name>.hex** and **<QSPIFlash_name>.hex**. Use **<QSPIFlash_name>.hex** for boot option 4 and 5 as you are booting from QSPI flash, not from OCRAM.

Hardware Programmer Object File (.pof) Generation

1. In Quartus II, click on **Convert Programming Files (.pof)** from the **File** tab.
2. Choose **Programmer Object File** as **Programming file type**.
3. Set **Mode** to **Internal Configuration**.
4. Change the **File name** to the desired path and name.
5. Under **Input files to convert**, select **Page_0** and click on **Add File** button on the right.
6. Browse to the first **.sof** file and click **OK**.
7. Click on **Add Sof Page** to create additional page for **.sof** file. This creates SOF data **Page_1** automatically.
8. Click **Add File...** and point to the second **.sof** file to add into **Page_1**.
9. Click **Generate** to create the **.pof** file.

Figure 91: SOF to POF Files Conversion Settings



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.

Programming file type: Programmer Object File (.pof)
Configuration device: EPCE16
Mode: Internal Configuration
File name: ../../HW_POF/SOF2POF.pof
Remote/Local update difference file: NONE

- [x] Create Memory Map File (Generate SOF2POF.map)
- [ ] Create CvP files (Generate SOF2POF.periph.pof and SOF2POF.core.rbf)
- [ ] Create config data RPD (Generate SOF2POF_auto.rpd)
- [ ] Create Fault Injection File (Generate SOF2POF.fif)

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

Related Information

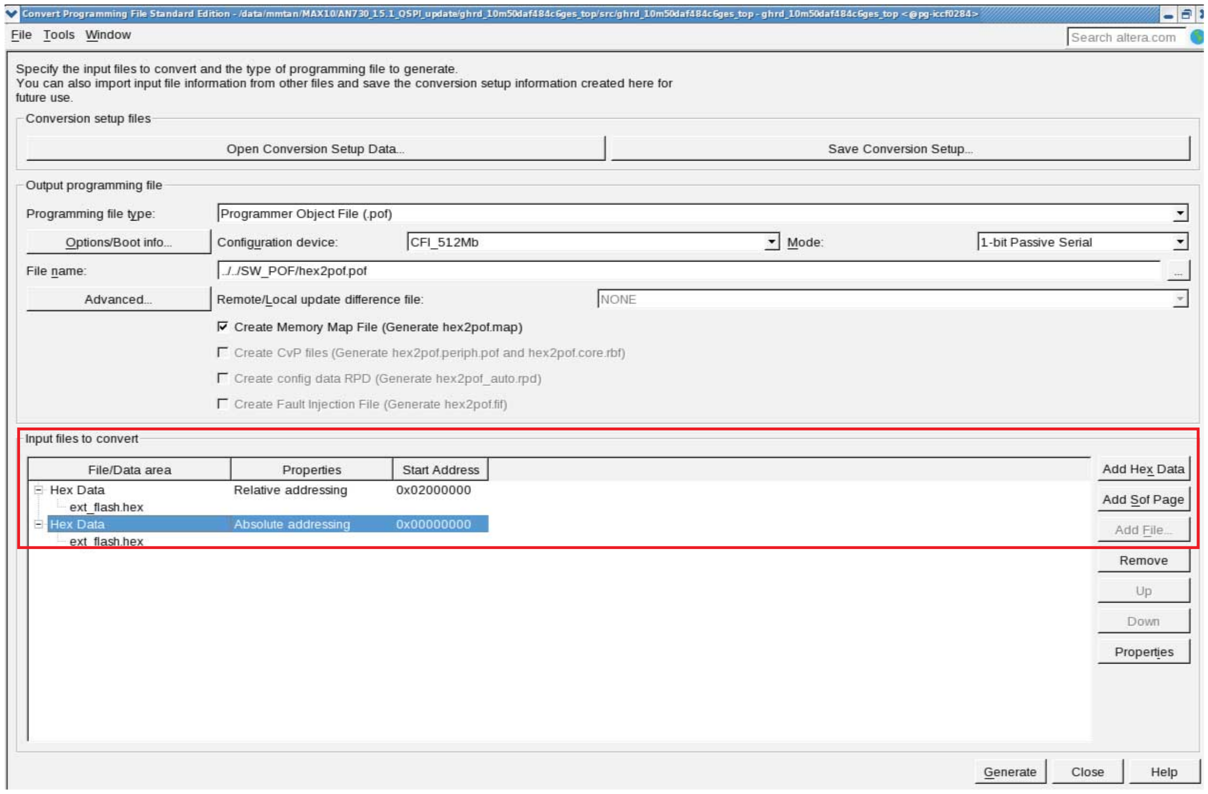
Programming Hardware Design POF File into the MAX10 FPGA on page 83

Software Programmer Object File (.pof) Generation

Important: The **quartus.ini** file with `PGMIO_SWAP_HEX_BYTE_DATA=ON` content is required to byteswap the programming file during the POF generation. Please create the **quartus.ini** file or use the **quartus.ini** available in the related information and place it under Quartus II tool directory or project directory before you proceed.

1. In Quartus II, click on **Convert Programming Files (.pof)** from the **File** tab.
2. Choose **Programmer Object File** as **Programming file type**.
3. Set **Mode** to **1-bit Passive Serial**.
4. Set **Configuration device** to **CFI_512Mb**.
5. Change the **File name** to the desired path and name.
6. Remove the SOF **Page_0**.
7. Click on **Add HEX Data**, choose the HEX file generated in **HEX Generation** section. Select **Absolute Addressing** and click **OK**.
8. Repeat Step 7 to add second HEX file.
9. Click **Generate** to create the .pof file.

Figure 92: HEX to POF file Conversion Settings


Convert Programming File Standard Edition - /data/mmtan/MAX10/AN730_15.1_QSPI_update/ghrd_10m50daf484c6ges_top/src/ghrd_10m50daf484c6ges_top - ghrd_10m50daf484c6ges_top <@pg-iccf0284>

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: ../../SW_POF/hex2pof.pof

Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate hex2pof.map)

☐ Create CvP files (Generate hex2pof.periph.pof and hex2pof.core.rbf)

☐ Create config data RPD (Generate hex2pof_auto.rpd)

☐ Create Fault Injection File (Generate hex2pof.fif)

Input files to convert

File/Data area	Properties	Start Address
Hex Data	Relative addressing	0x02000000
ext_flash.hex		
Hex Data	Absolute addressing	0x00000000
ext_flash.hex		

Related Information

Quartus.ini file

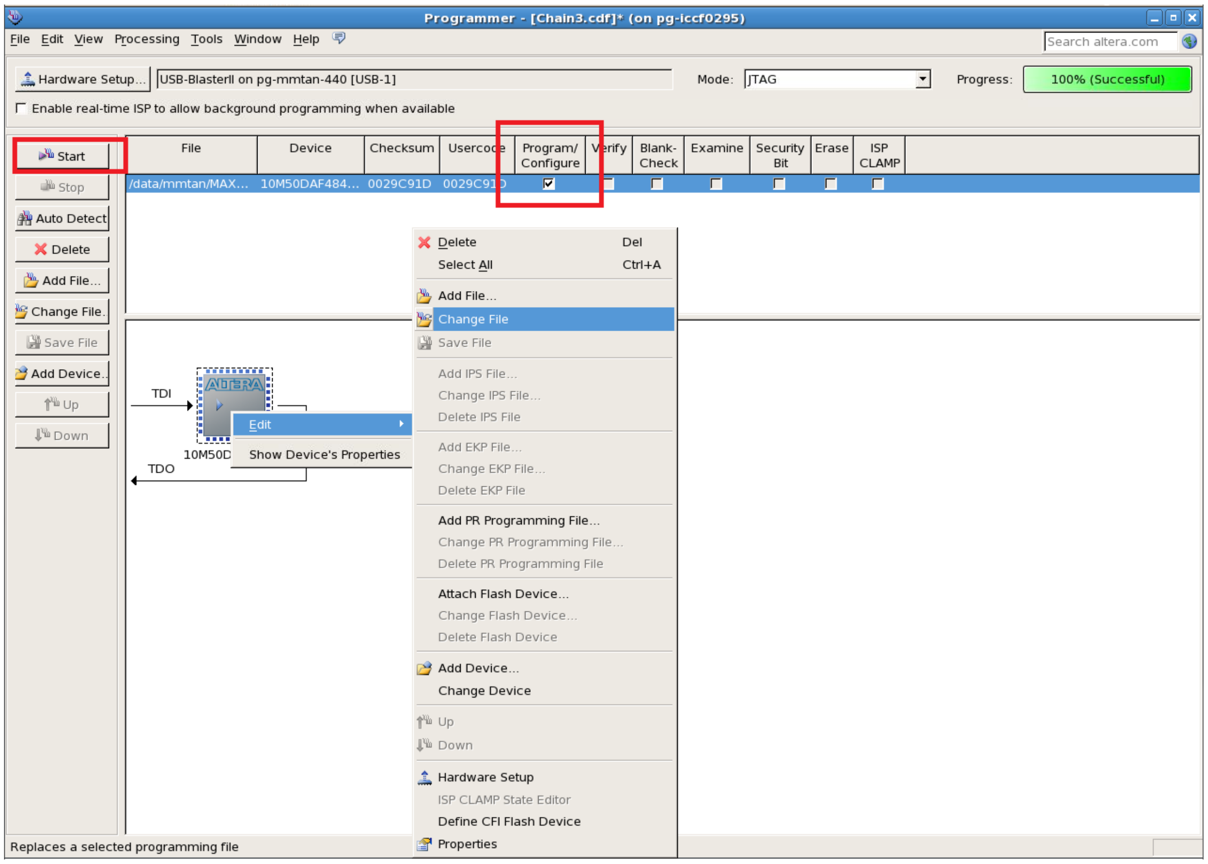
POF file Programming into QSPI flash

1. Programming Parallel Flash Loader into MAX10 Device

   Note: You need to program the parallel flash loader into the MAX 10 device before programming the QSPI flash.

   a. Create Parallel Flash Loader for MAX 10 FPGA in the Quartus II. Assign QSPI pins based on your design. Compile the project to obtain **max10_qpfl.sof** file.
   b. Open Quartus II programmer from the Quartus II tool (**Tools** -> **Programmer**).
   c. Make sure the **Hardware Setup** is set to **USB blaster**.
   d. Click on **Auto Detect**, select your MAX10 FPGA and select **OK**.
   e. Right click on the **MAX 10 FPGA** and select **Edit** -> **Change File**. Choose the **max_qpfl.sof** file.
   f. Check MAX 10 device under **Program/Configure** and click **Start** to start programming.
   g. Click on **Auto Detect** after **max10_qpfl.sof** is successfully programmed. Click **Yes** if you are asked to overwrite the existing settings. A new QSPI flash device will be shown on the screen.

Figure 93: Programming Parallel Flash Loader Programmer Settings



Programmer - [Chain3.cdf]* (on pg-iccf0295)

File Edit View Processing Tools Window Help

Hardware Setup... USB-BlasterII on pg-mmtan-440 [USB-1]

Mode: JTAG

Progress: 100% (Successful)

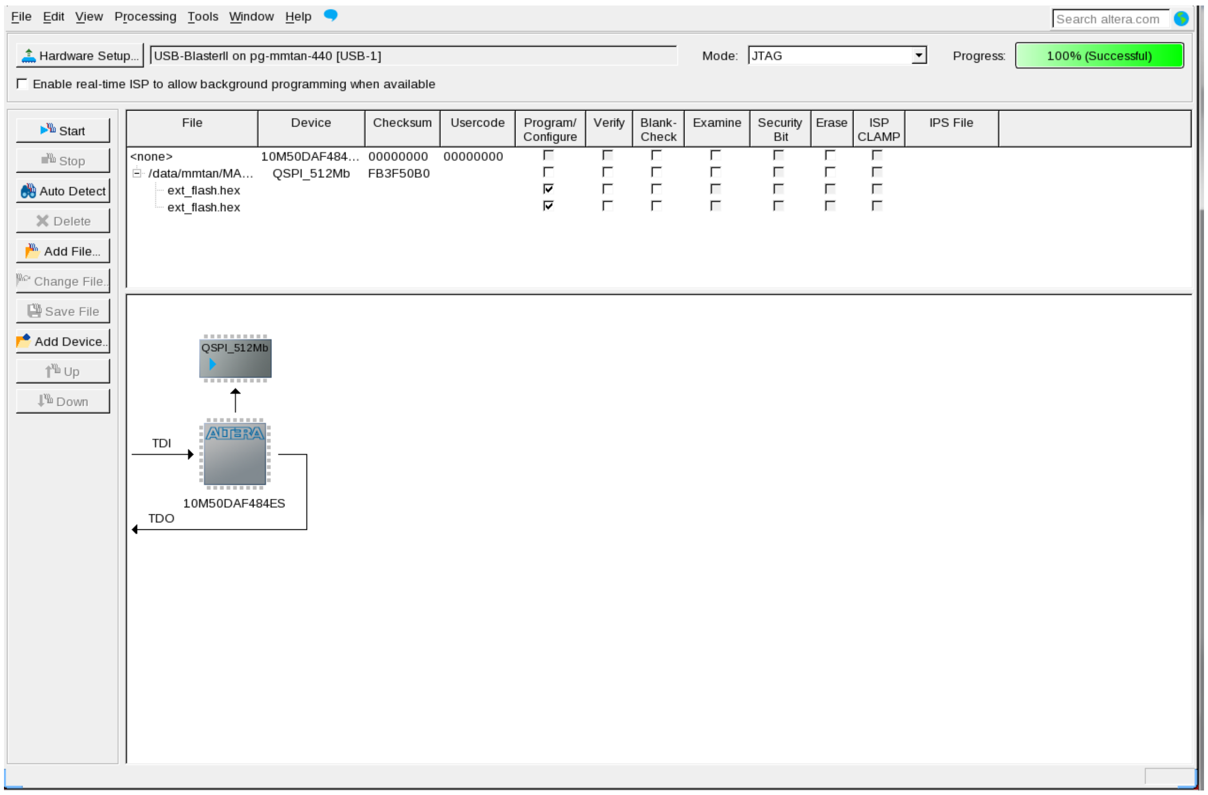
Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
/data/mmtan/MAX...	10M50DAF484...	0029C91D	0029C91D	☑	☐	☐	☐	☐	☐	☐

Replaces a selected programming file

2. Programming HEX image into QSPI Flash
   a. Right click on the **QSPI device** and select **Edit** -> **Change File**. Choose the generated POF file from *Software POF Generation* section.
   b. Check the HEX file under **Program/Configure** column and click **Start** to start programming.

Figure 94: HEX Image Programming into QSPI Flash



Hardware Setup... USB-BlasterII on pg-mmtan-440 [USB-1]   Mode: JTAG   Progress: 100% (Successful)

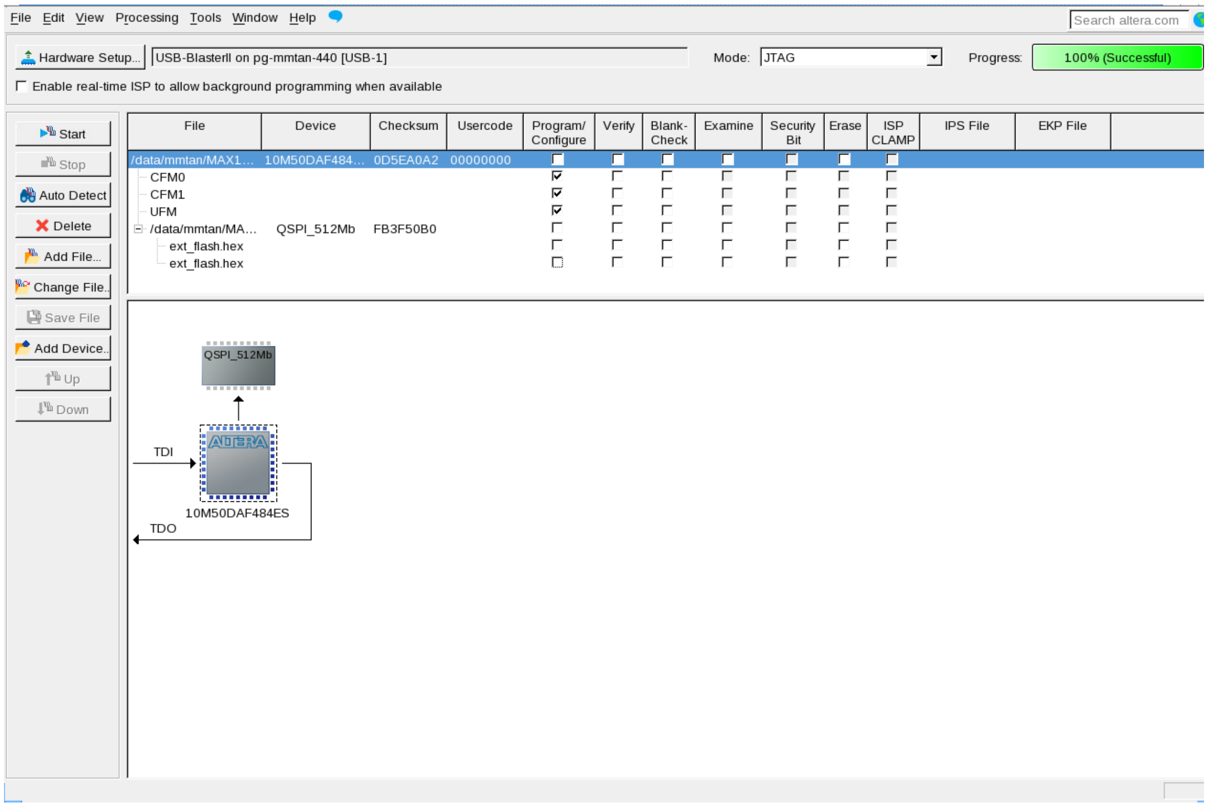
☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP	IPS File
<none>	10M50DAF484...	00000000	00000000	☐	☐	☐	☐	☐	☐	☐	
/data/mmtan/MA...	QSPI_512Mb	FB3F50B0		☐	☐	☐	☐	☐	☐	☐	
ext_flash.hex				✓	☐	☐	☐	☐	☐	☐	
ext_flash.hex				✓	☐	☐	☐	☐	☐	☐	

Programming Hardware Design POF File into the MAX10 FPGA

1. After you successfully programmed HEX data into Quad SPI flash, right click on the **MAX 10 FPGA** and select **Edit** -> **Change File**. Choose the hardware POF file generated from Hardware POF Generation section.
2. Check the MAX10's **CFM0** and **UFM** under **Program/Configure** column and click **Start** to start programming.

Figure 95: POF Image Programming into QSPI Flash



Hardware Setup... USB-BlasterII on pg-mmtan-440 [USB-1] Mode: JTAG Progress: 100% (Successful)

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

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP	IPS File	EKP File
/data/mmtan/MAX1...	10M50DAF484...	0D5EA0A2	00000000	☐	☐	☐	☐	☐	☐	☐		
CFM0				☑	☐	☐	☐	☐	☐	☐		
CFM1				☑	☐	☐	☐	☐	☐	☐		
UFM				☑	☐	☐	☐	☐	☐	☐		
/data/mmtan/MA...	QSPI_512Mb	FB3F50B0		☐	☐	☐	☐	☐	☐	☐		
ext_flash.hex				☐	☐	☐	☐	☐	☐	☐		
ext_flash.hex				☐	☐	☐	☐	☐	☐	☐		

Related Information

- Quartus II Software Settings on page 73
- Hardware Programmer Object File (.pof) Generation on page 79

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 11: Summary of Nios II Processor Vector Configurations and BSP Settings

Boot Option	Reset Vector Configuration	Exception Vector Configuration	BSP Editor Setting: Settings.Advanced.hal.linker	BSP Editor Setting: Linker Script
Option 1: Nios II processor application execute-in-place from Altera On-chip Flash (UFM)	Altera On-chip Flash	1. OCRAM/ External RAM, **OR** 2. Altera On-chip Flash	If the exception vector memory is set to OCRAM/ External RAM, 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 memory is set to Altera On-chip Flash, 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	• Set **.text** Linker Section to Altera On-chip Flash • Set other Linker Sections (**.heap, .rwdata, .rodata, .bss, .stack**) to OCRAM/ External RAM
Option 2: Nios II processor application copied from UFM to RAM using boot copier	Altera On-chip Flash	OCRAM/ External RAM	Make sure all settings in **Settings.Advanced.hal.linker** are left unchecked	Make sure all Linker Sections are set to OCRAM/ External RAM
Option 3: Nios II processor application execute-in-place from Altera On-chip Memory (OCRAM)	OCRAM	OCRAM	Enable **allow_code_at_reset** in **Settings.Advanced.hal.linker** and other settings are left unchecked.	Make sure all Linker Sections are set to OCRAM

Boot Option	Reset Vector Configuration	Exception Vector Configuration	BSP Editor Setting: Settings.Advanced.hal.linker	BSP Editor Setting: Linker Script
Option 4: Nios II processor application executes in-place from QSPI flash	QSPI flash	1. OCRAM/ External RAM, or 2. QSPI Flash	If the exception vector memory is set to OCRAM/ External RAM, 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 memory is set to QSPI Flash, 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	• Set **.text** Linker Section to QSPI Flash • Set other Linker Sections(**.heap, .rwdata, .rodata, .bss, .stack**) to OCRAM/ External RAM
Option 5: Nios II processor application copied from QSPI flash to RAM using boot copier	QSPI flash	OCRAM/ External RAM	Make sure all settings in **Settings.Advanced.hal.linker** are left unchecked	Make sure all Linker Sections are set to OCRAM/ External RAM

Appendix A: Booting Elements

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

- The memcpy-based boot copier
- The alt_load function
- The Nios II SBT "make mem_init_generate" target
- The Convert Programming Files feature

Nios II Processor Memcpy-based Boot Copier

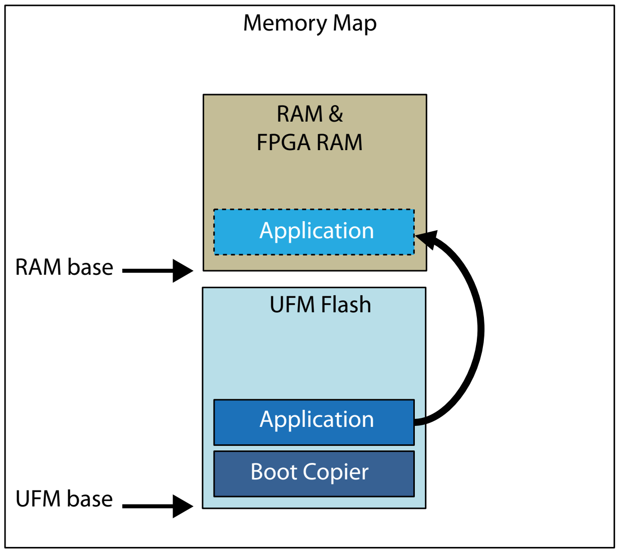
The Nios II processor memcpy-based boot copier has the following features:

- Supports EPCQ, CFI, QSPI and Altera On-chip Flash (UFM) flash memories
- Locates software application in the memory
- Unpacks and copies software application image to RAM
- Automatically switches to application code in RAM after copy completes

The memcpy-based boot copier is used to support boot option 2 and 5. The memcpy-based boot copier is automatically appended into the HEX file during memory initialization file generation ("mem_init_generate" target). When you download your **.pof** file into the FPGA, the boot copier will be placed at the beginning of the UFM or QSPI sector and the software application will be placed at the end of the boot copier.

The function of the memcpy-based boot copier is to copy the software application image to the RAM which is the entry point indicated by the software application (ELF file). Depending on the system design, RAM can be either OCRAM or external RAM . Once the copying is done, the boot copier will pass the system control to the application in RAM.

Figure 96: Memory Map of An Example System Using Default Boot Copier



The alt_load() function

The alt_load() function is a mini-bootcopier included in the HAL code and provides the following capabilities:

- Optionally copies sections from the boot memory to RAM based on BSP settings.
- Able to copy data sections (**.rodata**, **.rwdata**, **.exceptions**) to RAM but not the code sections (**.text**).

The alt_load() function can be enabled in the BSP Settings as shown in the following table:

Table 12: BSP Settings and the alt_load() Functions

BSP Settings	Functions
hal.linker.enable_alt_load	enable alt_load() function

BSP Settings	Functions
hal.linker.enable_alt_load_copy_rodata	alt_load() copies .rodata section to RAM
hal.linker.enable_alt_load_copy_rwdata	alt_load() copies .rwdata section to RAM
hal.linker.enable_alt_load_copy_exceptions	alt_load() copies .exceptions section to RAM

Nios II SBT Makefile “mem_init_generate” Target

The Nios II SBT application Makefile “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 tool automatically adds the Nios II processor memcpy-based 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” target.

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

- For boot option 1, 3 and 4 the generated HEX file contains ELF loadable section.
- For boot option 2 and 5, the generated HEX file contains the boot copier and the ELF payload.

The **mem_init_generate** target also generates a Quartus II IP file (**meminit.qip**). Quartus II software will refer to the **meminit.qip** for the location of the initialization files.

The Convert Programing Files Option

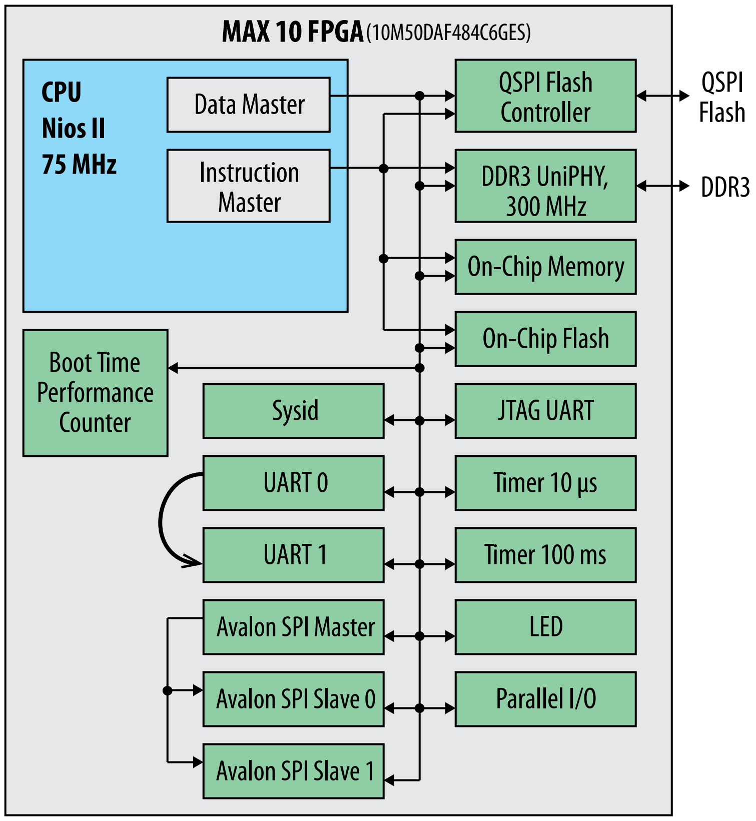
You can use the Convert Programming Files option in Quartus II software to convert programming files from one file format to another. This tool is used for combining a **.sof** and a HEX file into a single **.pof** file for programming into the Altera On-chip Flash.

Appendix B: Boot Time Performance Analysis

Altera performed a boot time performance analysis for MAX 10 FPGA based on several use cases. This section provides guidance on how to reduce Nios II boot time in MAX10 design and boot time estimation based on performance analysis. You can use the information in this section as a guideline when designing your custom design.

Figure 97: Boot Time Performance Analysis Design Block Diagram

Diagram shows the design was being used to run the MAX 10 FPGA boot time performance analysis. The Nios II processor was configured with special settings for some of the use cases.



Regardless of the boot device, there are only 2 types of boot methods:

1. Nios II processor application execute-in-place.
2. Nios II processor application copied from boot device to RAM using boot copier.

For boot performance analysis, the Nios II processor application sizes varies between 10kB to 64kB.

Boot Time Performance Analysis Design Example

The design example was tested on MAX10 10M50 Development Kit (Rev B) on ACDS 15.0 Build 145.

The design example boots from the Altera On-chip Flash (UFM). If you want to boot from other boot memory device, you have to change the reset vector and BSP settings according to Table 11.

Related Information

Boot Time Performance Analysis Design Example

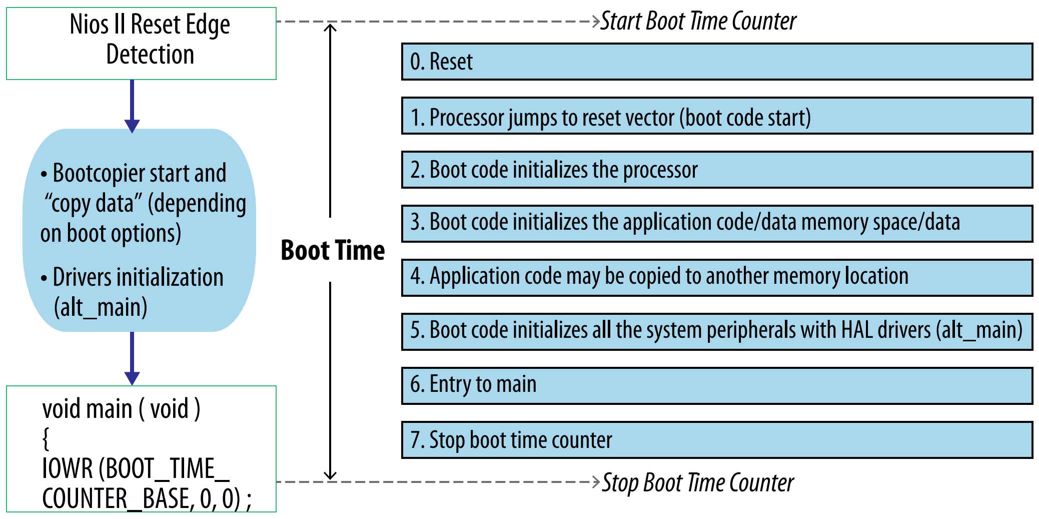
Boot Time Measurement Strategy

The boot time measurement was done using a boot time performance counter. The boot time counter starts when Nios II comes out of reset and the system will start the boot sequence according to the selected boot options:

1. Nios II processor application execute-in-place:
   - The Alt_load() function copies data sections (**.rodata**, **.rwdata** , **.exceptions**) from boot memory to RAM.
   - The code section (**.text**) remains in the boot memory.
   - Automatically jump to the user generated application code in the boot memory after copy completed.
   - Run system/ driver initialization (alt_main).
2. Nios II processor application copied from boot device to RAM using boot copier:
   - Bootcopier locates the software application in the boot memory and copies the software application to RAM.
   - Automatically jump to the user generated application code in the RAM after copy completed.
   - Run system/ driver initialization (alt_main).

The boot time counter is controlled through software and it will stop once the driver initialization completes.

Figure 98: Boot Time Measurement


Nios II Reset Edge Detection
• Bootcopier start and "copy data" (depending on boot options)
• Drivers initialization (alt_main)
void main (void)
{
IOWR (BOOT_TIME_COUNTER_BASE, 0, 0) ;
Boot Time
Start Boot Time Counter
0. Reset
1. Processor jumps to reset vector (boot code start)
2. Boot code initializes the processor
3. Boot code initializes the application code/data memory space/data
4. Application code may be copied to another memory location
5. Boot code initializes all the system peripherals with HAL drivers (alt_main)
6. Entry to main
7. Stop boot time counter
Stop Boot Time Counter

Reducing Nios II Boot Time in MAX 10 FPGA Design

There are a few elements that you can consider to improve the Nios II boot time in MAX10 designs. Based on the Nios II Boot Time analysis that Altera has performed, the following subsections can be referred to for general guidance.

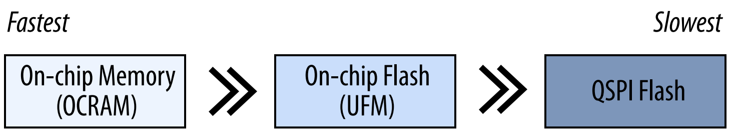
Boot Methods

1. Nios II application boot time for execute-in-place is faster than using a boot copy/run from RAM based boot (due to the time taken to copy the code to RAM being much longer than the time gained by the higher performance when running from RAM instead of ROM).
2. The boot copier method is recommended for systems that require higher performance. Although this configuration takes longer to boot, it delivers higher application performance compared to an execute-in-place configuration.

Boot Device Performance

1. For Nios II application execute-in-place, different boot devices will have different boot times based on their individual memory performance.
2. The following shows the performance for the supported boot device in MAX 10 FPGAs, when Nios II applications execute-in-place:

Figure 99: Supported Boot Device Performance



Peripheral Initialization

Nios II systems initialize all HAL peripherals before main() by default, therefore the boot time has a dependency on the peripherals selected. Peripherals that are slow to initialize or have external dependencies, will increase the boot time and potentially make it less deterministic. If this occurs, you need to calibrate the external memory such as DDR3 for it to work properly.

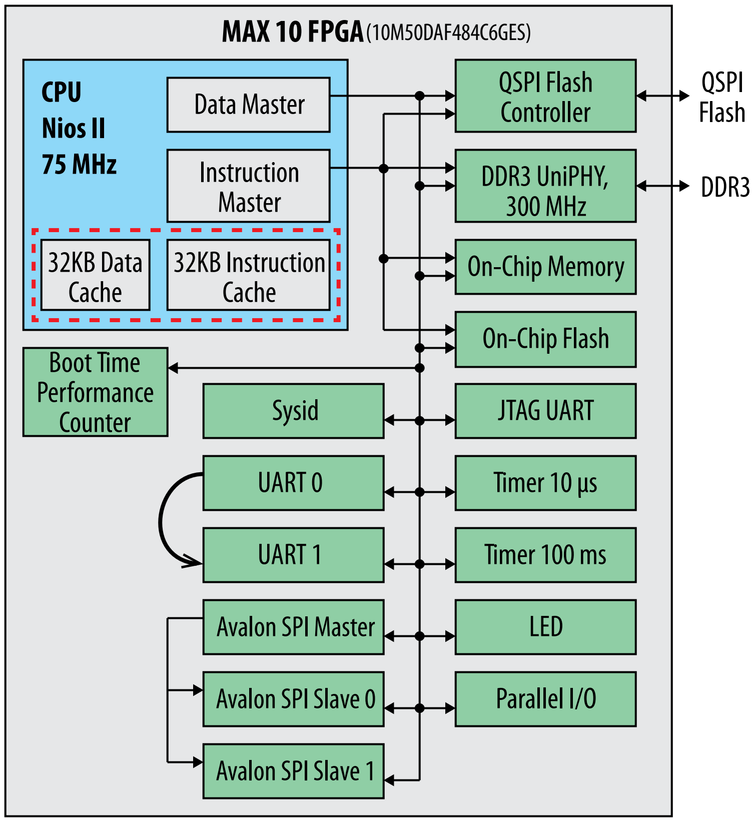
DDR3 is an example of a peripheral where the initialization time is significant in comparison to the boot time. The calibration time is long particularly when compared to boot times for execute-in-place boot configurations. The calibration time will significantly impact Nios II application that execute-in-place.

To avoid this, in execute-in-place boot configurations, remove the external memory from the Nios II linker region if it is not in use. If size is not an issue, you can choose to use OCRAM. If you are confident working with Nios II software, another option is to remove the DDR3 initialization routine from the boot code and initialize the memory later–once the application code has started running.

Nios II Processor Caches

1. Enable Nios II processor caches. Caches improve the boot time because the data and instruction caches reduce memory bandwidth limitations during the boot sequence.
2. You can add caches to your Nios II hardware configuration for both execute-in-place or bootcopier boot methods to improve boot time.

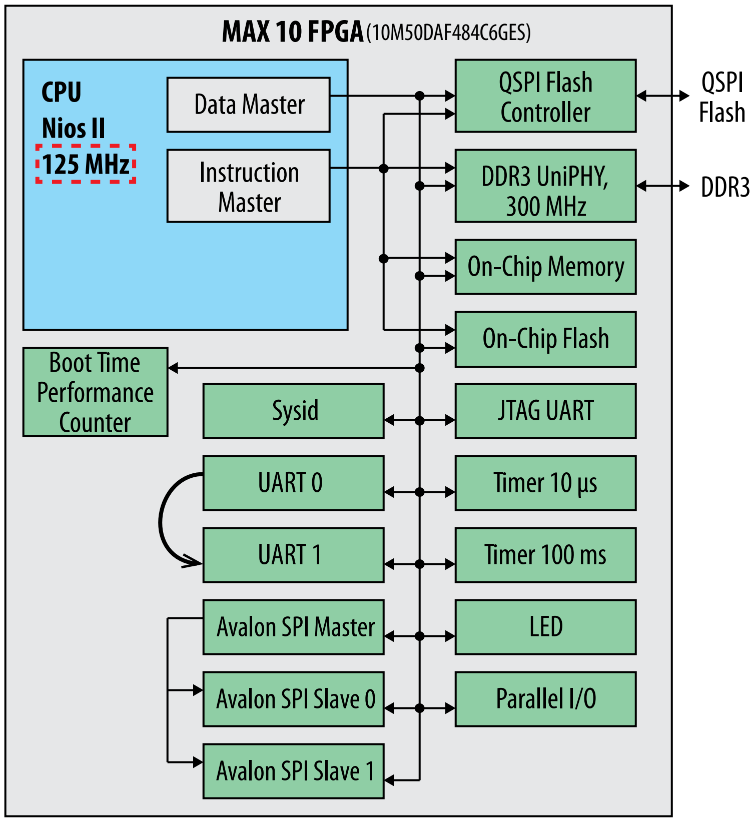
Figure 100: Nios II Design Block Diagram with Processor Caches Enabled



System Speed

Using Nios II processor with higher clock speed improves the boot time. For example, you can increase the Nios II processor speed from 75 Mhz to 125 MHz.

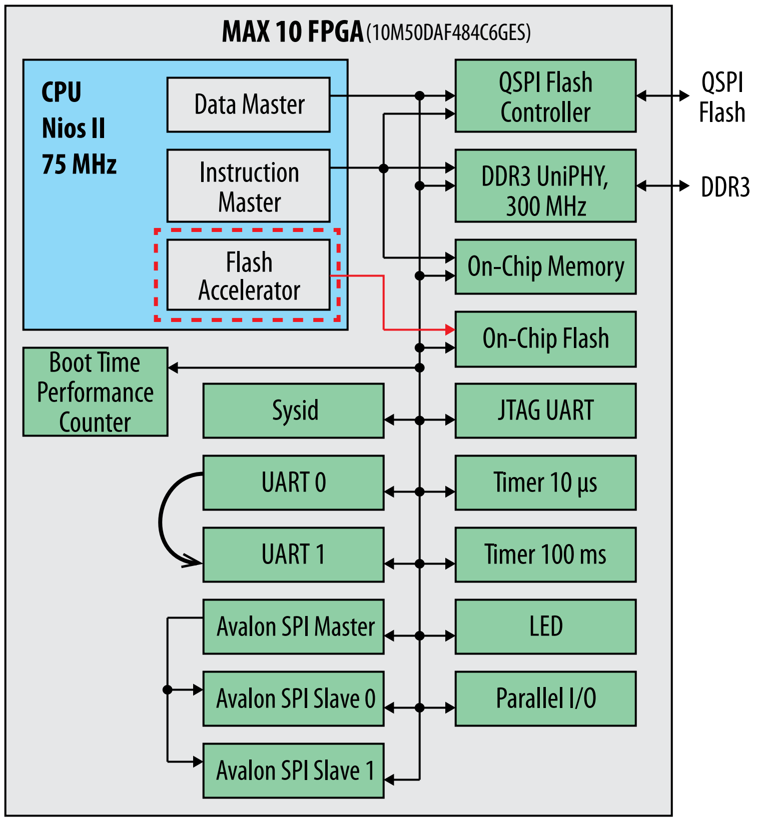
Figure 101: Nios II Design Block Diagram with Higher System Speed



MAX 10 FPGA Nios II Flash Accelerator

Enable the Nios II flash accelerator when system resets from the MAX 10 FPGA on-chip flash to improve the boot time.

Figure 102: Nios II Design Block Diagram with Flash Accelerator Enabled



Related Information

AN-740: Nios II Flash Accelerator Using MAX 10

Boot Time Performance and Estimation

Boot Time Performance for Design that Boots from On Chip Flash with Flash Accelerator

The following bar graphs show the boot time performance for designs that boot from on-chip flash (application code stored in on-chip flash) with and without the flash accelerator (FA) unit. Three scenarios have been evaluated:

1. Design with EMIF that runs from external memory DDR3 (using boot copier).
2. Design without EMIF that runs from on-chip memory (using boot copier)
3. Design without EMIF that runs from on-chip flash (execute in-place)

Note: None of the designs include an instruction or data cache.

Figure 103: Summary of Boot Times for Designs that Boot from On-Chip Flash with or without Flash Accelerator

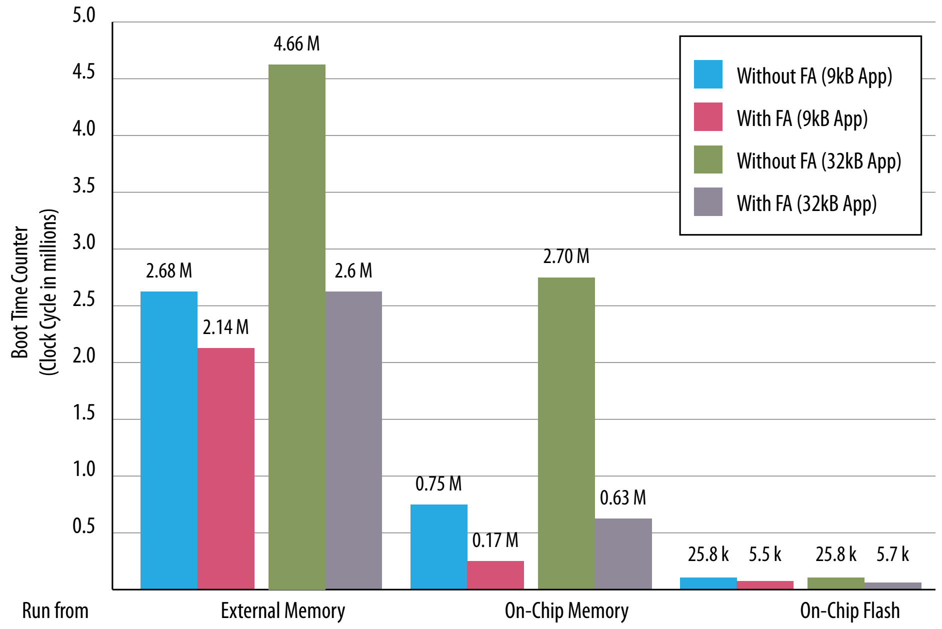
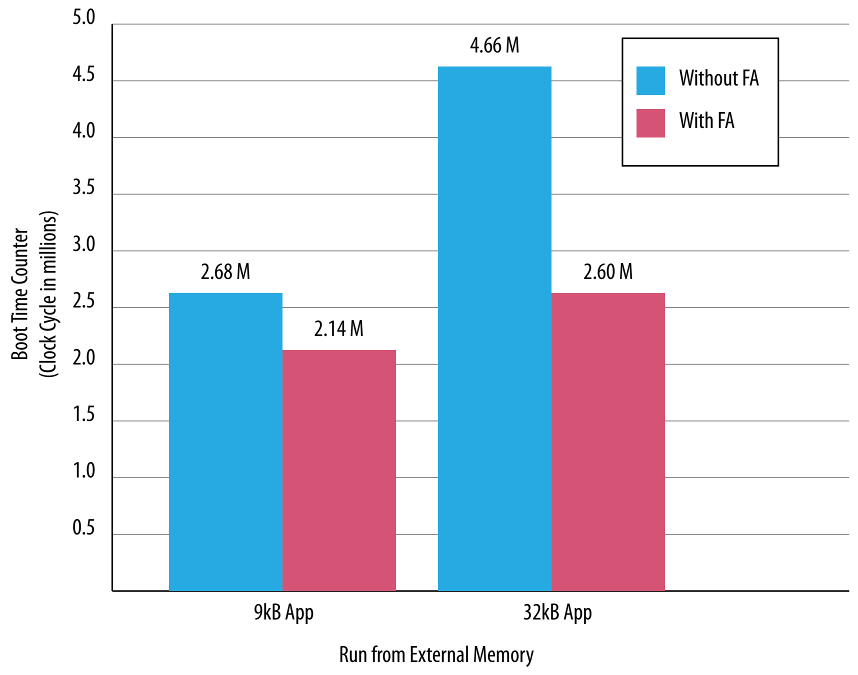
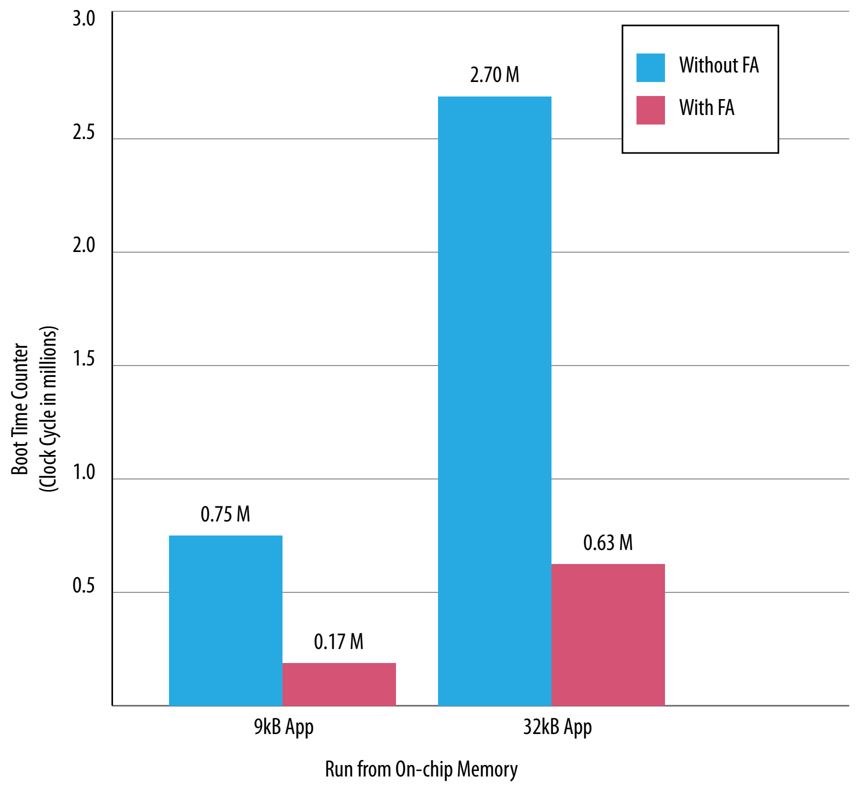


Figure 104: Boot Time for Design with EMIF that runs from External Memory (DDR3)



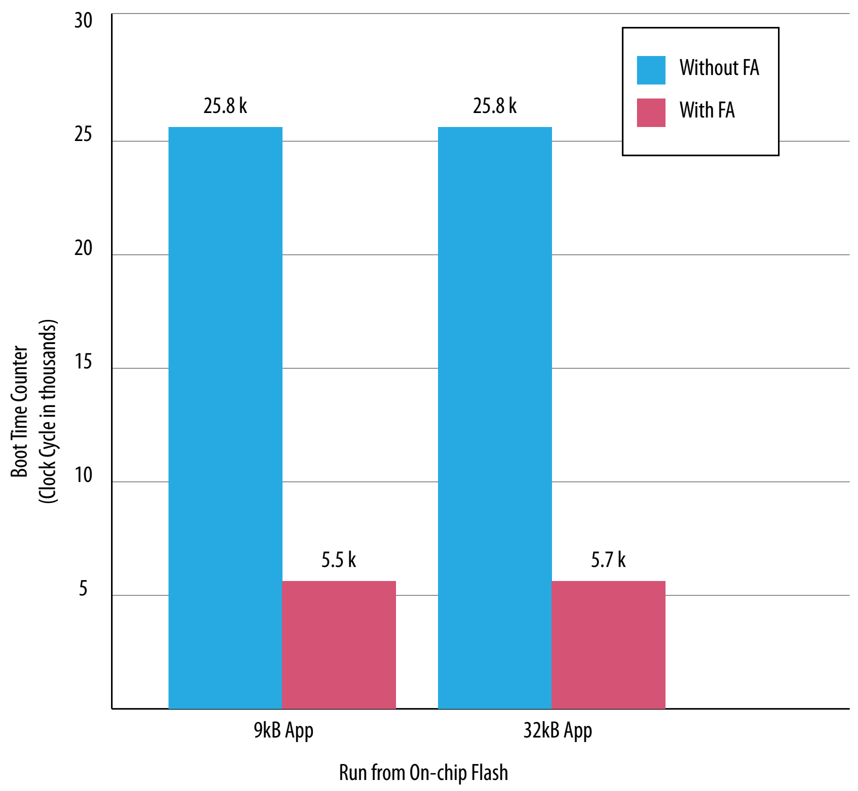
The boot time was reduced by ~20% and ~44% for 9 kB and 32 kB application sizes respectively when the Nios II FA is enabled. Larger application sizes will result in longer boot times because the code has to be copied from the on-chip flash to the external memory during the boot process. Overall, the boot time for the scenario where the boot code is running from the external memory is the slowest. This is due to the long time taken by external memory calibration during system boot up and the time taken to copy the application code to external memory.

Figure 105: Boot Time for Design without EMIF that runs from On-Chip Memory



The boot time was reduced by ~77% and ~76% for 9 kB and 32 kB application sizes respectively when the Nios II FA is enabled. Running the boot code from the on-chip memory is faster than running from external memory because the on-chip memory does not need to go through memory calibration during system boot up. Additionally, access time to the on-chip memory are faster than those for external DDR3 memory.

Figure 106: Boot Time for Design without EMIF that runs from On-Chip Flash (Execute In-Place)



The boot time was reduced by ~78% and ~77% for 9 kB and 32 kB application sizes respectively when the Nios II FA is enabled. Running the boot code from the on-chip flash (excute in-place) is faster than running from external memory and on-chip memory. This is because the code executes in-place directly from the on-chip flash and does not need to be copied into the external memory or on-chip memory for execution which saves a lot of time.

Boot Time Estimation

The tables below show estimates of boot time for different MAX 10 FPGA boot configurations. The boot time shown will help you to gauge the boot configuration required for your system design.

Note:
- Instruction or data cache and flash accelerator are not enabled in the design.
- Boot time values are based on design with 75 MHz Nios II processor speed.

Table 13: Boot Time Estimation on Execute-in-place test cases

Test Case	Boot From	Boot Time Counter (Approximate Nios II Clock cycles)[18]	Boot Time (millisecond)
Design without External Memory	On Chip Flash	41,000	0.55
	On Chip Memory	18,000	0.24
	QSPI Flash	410,000	5.5
Design with External Memory	On Chip Flash	2,000,000	27
	On Chip Memory	2,000,000	27
	QSPI Flash	2,300,000	31

Table 14: Boot Time Estimation on Boot Copier test cases

Test Case	Boot from	Run From	Boot Time Counter (Clock cycle)[19]	Boot Time (millisecond)
Design without External Memory	On Chip Flash	On Chip Memory	2,800,000	37
	Quad SPI Flash	On Chip Memory	28,000,000	370
Design with External Memory	On Chip Flash	External Memory	4,700,000	63
		On Chip Memory	2,800,000	37
	Quad SPI Flash	External Memory	30,000,000	400
		On Chip Memory	30,000,000	400

(18) Boot time counter is applicable to all software application (.elf) sizes.
(19) Boot time counter is for every 32kB .elf size.

Document Revision History

Table 15: Document Revision History

Date	Version	Changes
May 2016	2016.05.24	• Updated *Configuration and Booting Flow* figures for all boot options. • Updated *BSP Editor Settings* for Boot option 3. • Updated *Summary of Nios II Processor Vector Configurations and BSP Settings* table. • Updated *Programming Hardware Design POF File into the MAX10 FPGA* step for Boot option 4 and 5 dual compressed images. • Added note in *Programmer Object File (.pof) Generation* for Boot option 3.
November 2015	2015.11.19	• Added *Boot Time Performance for Design that Boots from On Chip Flash with Flash Accelerator* subsection.
September 2015	2015.09.15	• Replaced *MAX 10 FPGA Nios II Design Boot Time Estimation and Guidance* with *Boot Time Performance Analysis and Estimation* section. • Restructure boot options and flow by combining boot options and guidelines. • Added Boot option 4 and boot option 5 supporting QSPI. • Added Boot Time Performance Analysis design example information and link.
June 2015	2015.06.15	• Added *MAX 10 FPGA Nios II Design Boot Time Estimation and Guidance* section containing boot time performance analysis. • Added *The alt_load() function* table. • Added ROM Size Requirement to *RAM and ROM Size Requirement For Each Boot Option* table. • Added block diagrams for all boot options in *Nios II Processor Booting Options Using On-chip Flash*. • Added OCRAM size in *UFM and CFM Array Size* table. • Added Boot Option: 3 Nios II processor application execute in-place from Altera On-chip Memory. • Updated 'Configuration and Booting Flow' to include Boot Option: 3. • Updated Steps to Build a Bootable System to Guidelines to Build a Bootable System. • Updated Single Uncompressed/Compressed Image Bootable System Guideline to support Single Compressed Image Mode. • Added the third guideline; 'Single Uncompressed/Compressed Image with Memory Initialization Bootable System Guideline'. • Editorial changes.

Date	Version	Changes
January 2015	2015.01.23	Initial release.