

Application Note 214

Flash programming in the ARM Cortex-M1 FPGA Development Kit Altera Edition

Document number: ARM DAI 0214A

Issued: 2nd September, 2008

Copyright ARM Limited 2008

Application Note 214

Flash programming in the ARM Cortex-M1 FPGA Development Kit

Altera Edition

Copyright © 2008 ARM Limited. All rights reserved.

Release information

Change history

Date	Issue	Change
September 2008	A	First release

Proprietary notice

Words and logos marked with © and ™ are registered trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

Altera is a trademark and service mark of Altera Corporation in the United States and other countries. Altera products contain intellectual property of Altera Corporation and are protected by copyright laws and one or more U.S. and foreign patents and patent applications.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Confidentiality status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Unrestricted Access is an ARM internal classification.

Feedback on this Application Note

If you have any comments on this Application Note, please send email to errata@arm.com giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

ARM web address

<http://www.arm.com>

Table of Contents

1. Introduction	1-1
2. Programming designs to flash	2-1
2.1 Converting SOFs into POFs to program configuration flash	2-1
2.2 Programming the configuration flash using the POF file	2-3
3. Loading software into flash memory	3-1
3.1 Requirements for loading software into flash memory	3-1
3.2 Creating Intel Hexadecimal files for software flash initialization	3-1
3.3 Combining SOF and Hex data in the POF file	3-3
3.4 Programming the POF file	3-4
4. Appendix 1: Converting hex files using hex2hex.....	4-1
4.1 Flash memory initialization file requirements	4-1
4.2 Running hex2hex.....	4-1
4.3 Running hex2hex in RealView MDK	4-2
5. References	5-1

1. Introduction

The ARM Cortex-M1 FPGA Development Kit allows systems based around the ARM Cortex-M1 processor to be created easily using the Altera Quartus II SOPC Builder environment. The development kit also includes the Keil RealView MDK software development environment which incorporates the ARM RealView Compiler Toolchain.

This Application Note describes how synthesized SOPC Builder designs can be loaded onto your board's configuration flash so that the design is loaded onto the FPGA automatically at power-on time. It also describes how software images from RealView MDK can be included in the flash configuration data.

This Application Note assumes that you are familiar with the Altera Quartus II toolchain and Keil RealView MDK software. It applies to the following software versions:

- ARM Cortex-M1 FPGA Development Kit version 1.1
- Keil RealView MDK version 3.22a
- Altera Quartus II version 8.0

You should refer to the documentation for each tool for information about how to use the software. You may also refer to the *ARM Cortex-M1 FPGA Development Kit Cortex-M1 User Guide* for more information about the ARM Cortex-M1 processor in the ARM Cortex-M1 FPGA Development Kit.

This document uses the Altera Cyclone III Starter Board in the examples. You will need to refer to the documentation for your own development board for details about what flash memory is available and how this is used to configure the FPGA at power-on.

2. Programming designs to flash

The Altera Quartus II software creates an SRAM Object File (SOF) and a Programmer Object File (POF) when you compile a hardware design. You can use the Quartus II programmer to configure the FPGA with the SOF file via a download cable, but the design does not persist after the device is powered off.

The POF file that the Quartus II Assembler creates is used to program CPLD devices. You can generate an alternative POF file from the SOF file and use this to program a configuration flash device. The FPGA can be configured from the flash at power-on, avoiding the need to program the device manually each time it is powered on.

Note

- Refer to your development board's documentation for information about what flash memory is available and how this is used for power-on configuration.
- You can not convert SOF files that contain time-limited evaluation IP into POF files for flash configuration.

2.1 Converting SOFs into POFs to program configuration flash

Before programming your design into the configuration flash, you must convert your Quartus II project's SOF file into a POF file. To do this, after synthesizing your design successfully, select the **File** menu in the main Quartus II window and choose **Convert Programming Files**. This opens a window like the one shown in Figure 1 on page 2-2.

Note

You can also use the standalone Quartus II Programmer, available from Altera, if you already have a SOF file for your design. You do not need to install the complete Quartus II development environment if you only want to program and convert existing SOF files. This Application Note uses the programmer module from the complete Quartus II environment, but you can substitute this for the standalone Quartus II Programmer if necessary.

In the **Output programming file** pane of the **Convert Programming Files** window, ensure that the **Programming file type** is set to **Programmer Object File (.pof)** and choose an output filename in the **File name** field. You should set the other options in this pane to suitable values for your development board's flash memory and configuration mode. For the Altera Cyclone III Starter Board, you can use the following values:

- **Configuration device:** CFI_128MB
- **Mode:** Active Parallel

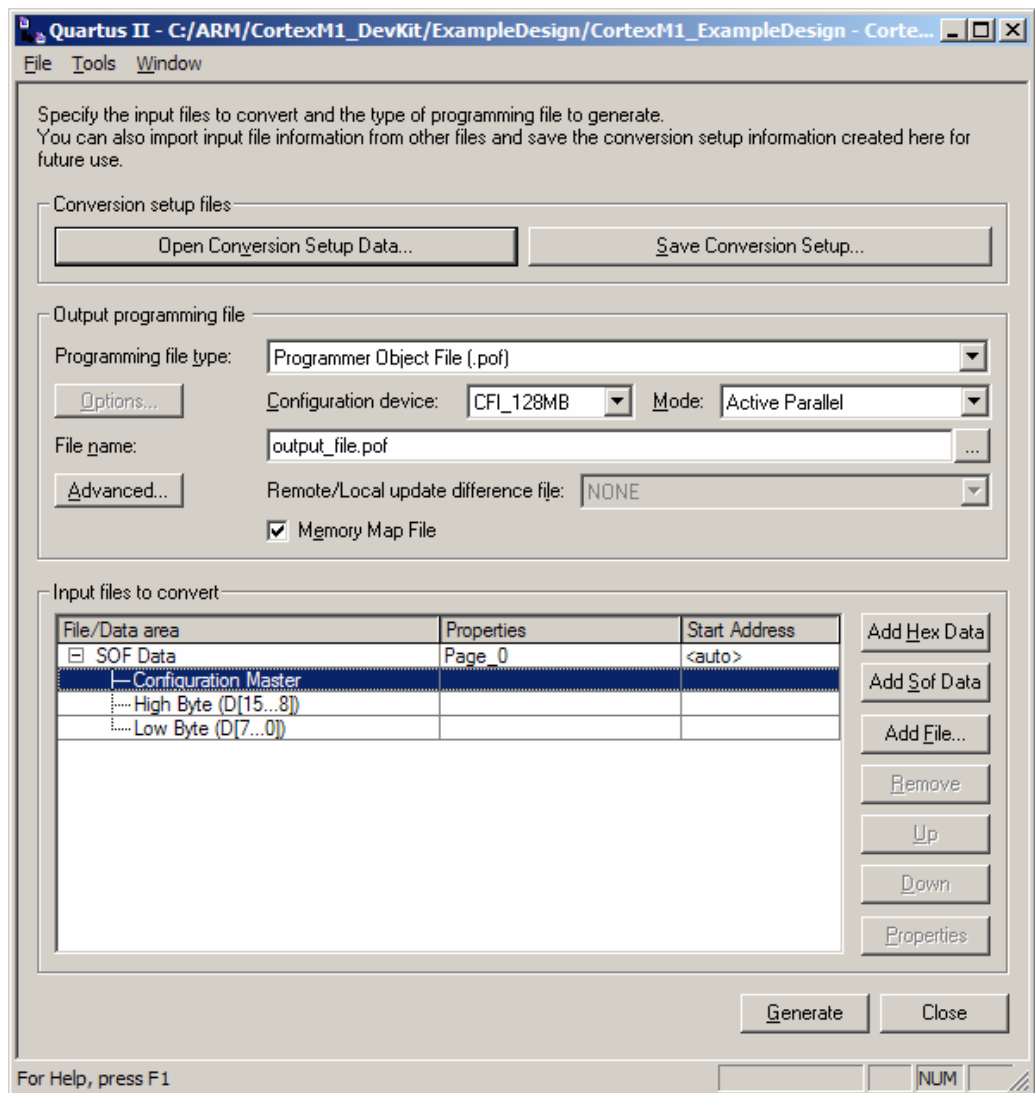


Figure 1 – Quartus II Convert Programming Files window

In the **Input files to convert** pane of the programming file conversion window, click on the **Configuration Master** row of the table to select it. Then:

1. Press the **Add File** button and browse to your design's SOF file.
2. Press the **Open** button in the file browser dialog when you have selected the SOF file.
3. Click on the **SOF Data** row of the table to select it, then click on the **Properties** button to open the **SOF Data Properties** window as shown in Figure 2 on page 2-3.

Note

The **Configuration Master** is the first device in the configuration chain. If you have multiple devices in your configuration chain, you can add more than one SOF file. Refer to the Altera Quartus II help for further information.

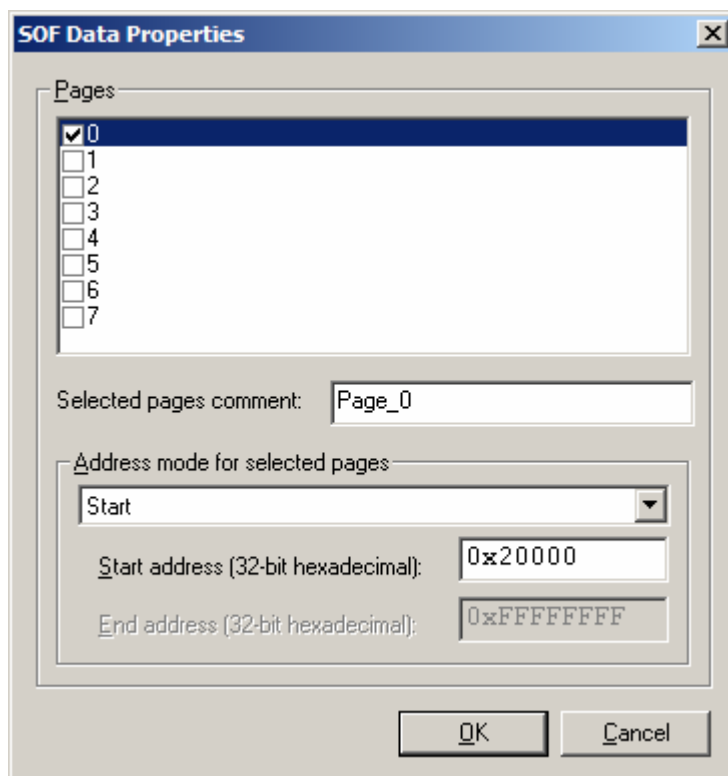


Figure 2 – SOF Data Properties window

You should enter the appropriate configuration options for your target development board's configuration flash device in the **SOF Data Properties** window. For the Altera Cyclone III Starter Board, you can use the following values:

- **Pages:** 0
- **Address mode for selected pages:** Start
- **Start address (32-bit hexadecimal):** 0x20000

These configuration options specify that the start address for the configuration data in flash memory is 0x20000. This is the default address from which the Cyclone III Active Parallel configuration loads configuration data.

Click on the **OK** button in the **SOF Data Properties** window, then click on **Generate** in the main programming files conversion window to write the POF file.

2.2 Programming the configuration flash using the POF file

After creating the POF file, you can program it to the configuration flash device using the Quartus II programmer. This process is similar to configuring the FPGA using a SOF file.

To program the POF to the configuration flash device, open the Quartus II programmer by clicking the **Tools** menu in the main Quartus II window and selecting **Programmer**. The Quartus II programmer window appears as shown in Figure 3 on page 2-4.

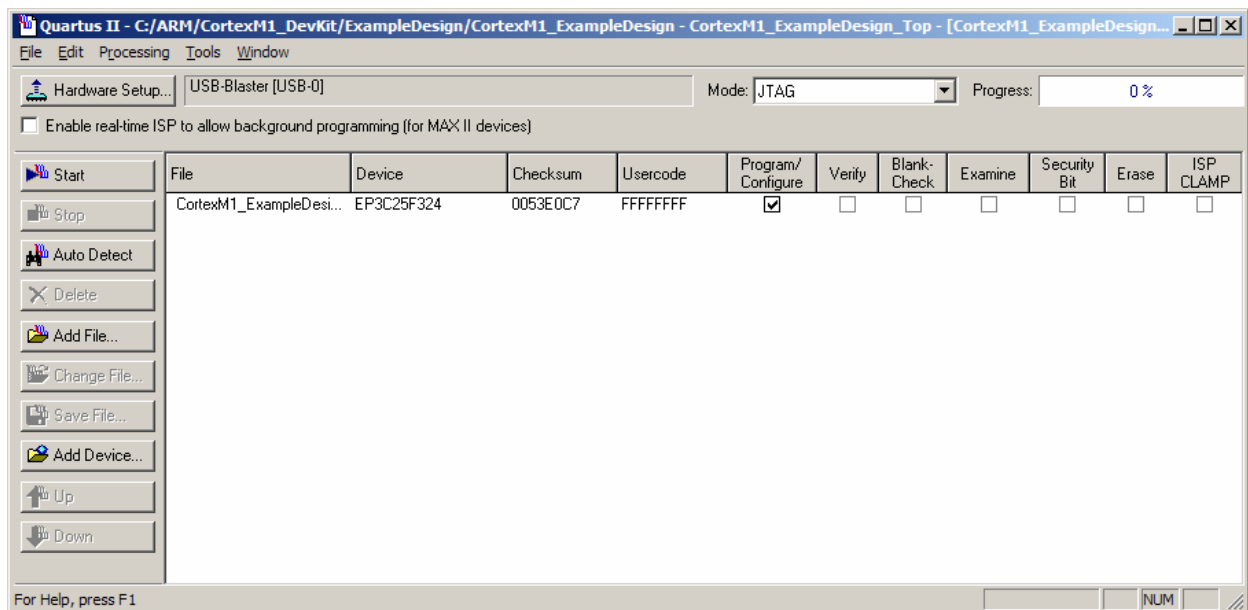


Figure 3 – Quartus II Programmer window

Note

If you have a project open and this has been fully compiled, the SOF file will automatically appear in the programming list as shown in the example in Figure 3.

To program the configuration flash, ensure that the board is connected to the host workstation with a download cable, then:

1. Press the **Auto Detect** button. This will clear any existing files from the programming list and automatically detect the target board.
2. Select the detected device in the list. This will be an **EP3C25** device if you are using the Altera Cyclone III Starter Board.
3. From the **Edit** menu, select **Attach Flash Device** to open the **Select Flash Device** window as shown in Figure 4 on page 2-5.
4. In the **Select Flash Device** window, choose the flash device that is on your development board. The Altera Cyclone III Starter Board has a CFI_128MB device. You can select this by clicking **Flash Memory** in the **Device family** pane and then **CFI_128MB** in the **Device name** pane. Press **OK** to return to the main Quartus II programmer window.

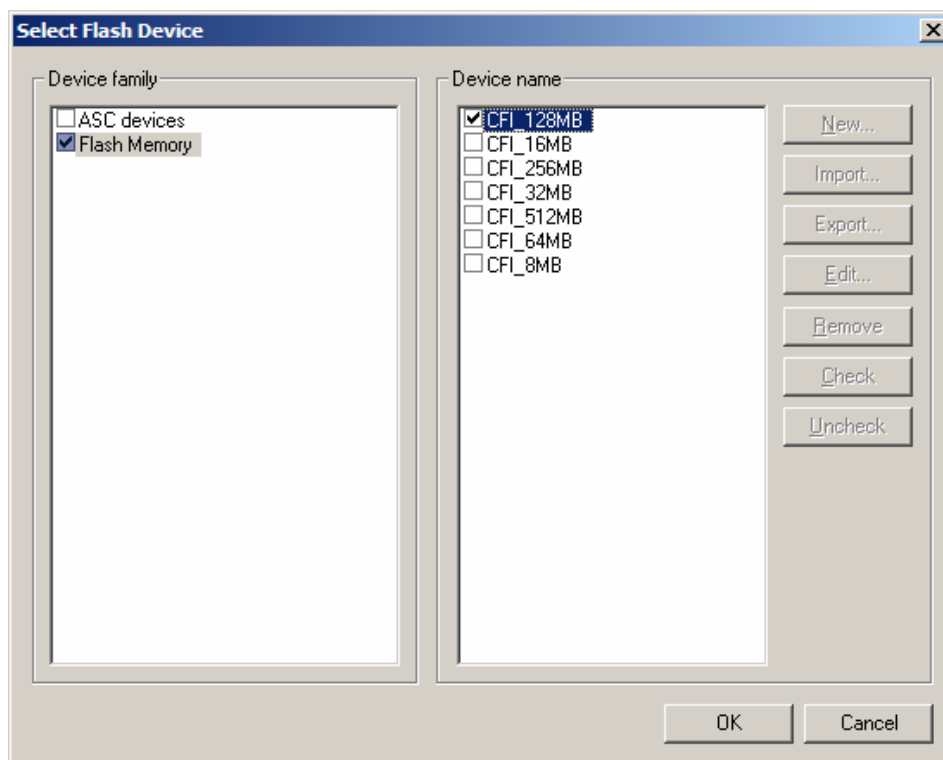


Figure 4 – Select Flash Device window

5. In the Quartus II Programmer window, select the CFI_128MB device that has appeared in the list.
6. Press the **Change File** button and browse to the POF file that was created in Section 2.1. Press **Open** in the file dialog after selecting the POF file.
7. The POF file appears in the Quartus II programmer list. Turn on the **Program/Configure** option for all devices in the list by clicking on the check-box under the **Program/Configure** column for the row of the table corresponding to the POF file. Enabling the option for the POF file automatically enables it for the other devices.
8. Click on the **Start** button to program the flash.
9. Turn off the board and then turn it on again. The FPGA will now be configured with the new image from the configuration flash device.

3. Loading software into flash memory

If the Cortex-M1 processor in your system contains initialized TCMs, or other initialized on-chip memories, the initialization data for these memories will be embedded in the SOF file. When you create a POF file for the configuration flash, the memory initialization data will also be embedded in the configuration flash data. This means that initialized TCMs and on-chip memories will be re-loaded with their initial data when the FPGA is configured.

You can also initialize other parts of the flash memory with separate data, such as Cortex-M1 instructions or program data. This might be useful if your hardware system contains a flash memory interface that the processor can access. You can add extra data to the flash memory by using Hex files when creating the POF file.

Note

The Cortex-M1 processor achieves the best performance when it is executing from its TCMs. Executing code directly from flash memory will give lower performance. However, for some applications you may wish to program software into the flash memory and use a boot loader to copy it into the TCMs or volatile memory.

3.1 Requirements for loading software into flash memory

To create a POF file which contains software code or data in addition to the FPGA configuration data, you can use the Quartus II programming file conversion utility. You will need:

- the SOF file for your compiled Quartus II project;
- software initialization data in the Intel Hexadecimal format.

You can use the Keil RealView MDK software to create Intel Hexadecimal format files from your software project. Then you can combine the SOF and Hex data into an output POF file for flash memory programming.

3.2 Creating Intel Hexadecimal files for software flash initialization

The RealView MDK software can produce Hex images in the Intel Hexadecimal format, and these files can be read by the Quartus II programming file conversion utility.

You must describe the size and location of the flash memory in RealView MDK and write your software to make use of this region. You must also enable Hex file output so that the software image can be imported into the Quartus II programming file conversion utility.

When writing software that uses flash memory, you must consider:

- the base address of the flash memory in the processor's memory map;
- the addresses within the flash memory that are used for FPGA configuration data.

To configure the flash memory region in RealView MDK and enable Hex file output:

1. Open the MDK project options and select the **Target** tab.
2. Enter the details of your flash memory as it appears in the processor's address map. You must specify the base address and size in the memory layout pane. Figure 5 shows an example where a 16MB block of flash memory is accessible at a base address of 0x60000000 in the processor's address space.

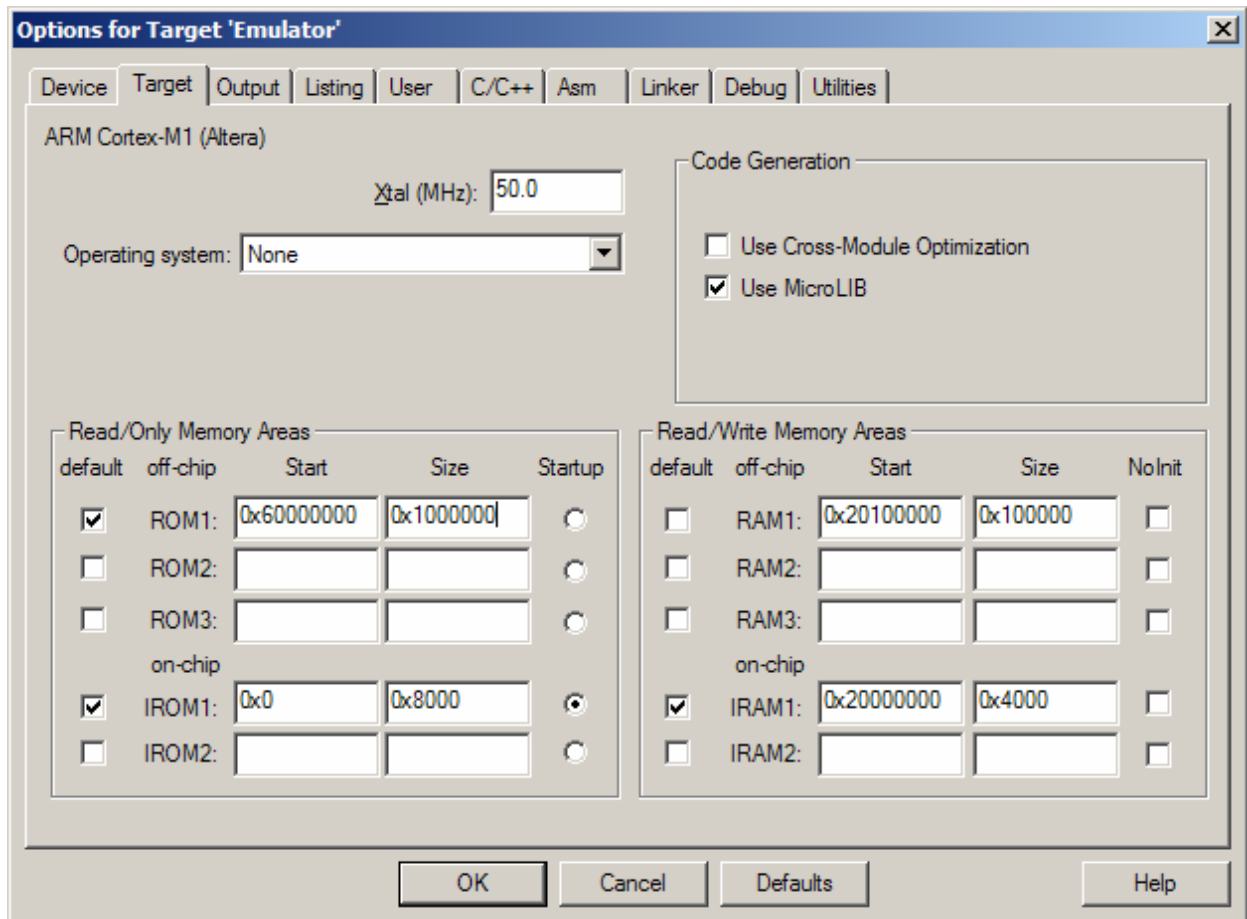


Figure 5 – RealView MDK Target options

3. Select the **Output** tab of the MDK project options window.
4. Enable Hex file generation by checking the **Create HEX File** option.
5. Press **OK** in the MDK project options window.

Building the project will now create a Hex image in the Intel Hexadecimal format in the `obj` subdirectory of your MDK project directory.

Note

This Hex image is not in the same format as the Hex images used to initialize Altera on-chip memories such as those used for the Cortex-M1 TCMs. To find out how to initialize on-chip memories in the ARM Cortex-M1 FPGA Development Kit, see *Application Note 213: Cortex-M1 TCM initialization in the ARM Cortex-M1 FPGA Development Kit Altera Edition*.

Each memory region that contains initialized data at power-on is known as a *load region*. The MDK project Hex file contains the combined data for every load region. If you have multiple load regions and do not want to program all of these to the flash memory, then you will need to extract the appropriate regions from the Hex file. Read Appendix 1 for details about how to extract data from and convert Hex files using the **hex2hex** utility.

3.3 Combining SOF and Hex data in the POF file

To create the final POF file containing the SOF file and Hex data, follow the procedure in Section 2.1 to add the SOF file. Instead of pressing **OK** in the Quartus II programming file conversion utility, follow these additional steps to add the Hex data:

1. In the **Input files to convert** pane, click on the **Add Hex Data** button to open the **Add Hex Data** window as shown in Figure 6.



Figure 6 – Add Hex Data window

2. In the **Add Hex Data** window, click on the ... button in the **Hex file** pane. Select the Hex file that you want to program into flash memory in the file selection dialog and press **Open**.
3. In the **Addressing mode** pane of the **Add Hex Data** window, select either **Absolute addressing** or **Relative addressing**:
 - Choose **Absolute addressing** if you want the addresses in the Hex file to be interpreted as absolute addresses. This means that data will be loaded into the exact addresses that are specified in the Hex file, relative to the base address of the flash memory.
 - Choose **Relative addressing** if you want the addresses in the Hex file to be treated as relative addresses. The addresses will be treated as relative to the end of the previous Hex file, unless the **Set start address** option is enabled. In that case, the addresses in the Hex file will be treated as relative to the address that you enter into the **Set start address** field, which is also relative to the base address of the flash memory.

You must ensure that all software code that is loaded into flash memory is loaded to the correct addresses that your software expects. The Quartus II programming file conversion utility interprets all addresses as relative to the flash memory base address, so depending on where your flash memory appears in the processor memory map, you may need to re-base the addresses in the Hex file to a starting address of 0x00000000.

ARM recommends that you use **hex2hex** to re-base your flash initialization data to a starting address of 0x00000000, as shown in Appendix 1, and use:

- **Absolute addressing** if your flash data starts at the flash memory base address; or
 - **Relative addressing** with an appropriate offset in the **Set start address** field if your flash data does not start at the flash memory base address.
4. Repeat steps 1 to 3 for any other Hex files that you want to add.
 5. Press **Generate** in the Quartus II programming file conversion utility to generate the POF file.

3.4 Programming the POF file

You can program the POF file as described in Section 2.2

When you program a POF file that contains Hex data in addition to SOF data, you can control whether the Hex data is programmed by enabling or disabling the **Program/Configure** option for your Hex file in the Quartus II Programmer window.

4. Appendix 1: Converting hex files using hex2hex

The Keil RealView MDK software produces a single Intel Hexadecimal file containing all of the software code and data. If you want to initialize different memories using different parts of the software image, you will need to extract the appropriate parts of the Hex file for each memory. For example, if part of your software is initialized in the Cortex-M1 processor's Instruction Tightly Coupled Memory (ITCM) and another part is initialized into flash memory you will need to generate an ITCM initialization file and a flash memory initialization file.

Note

See *Application Note 213: Cortex-M1 TCM initialization in the ARM Cortex-M1 FPGA Development Kit Altera Edition* for information about how to initialize the Cortex-M1 Tightly Coupled Memories.

4.1 Flash memory initialization file requirements

When you initialize flash memory with a Hex file using the Quartus II programmer, you must consider the following requirements:

- The Hex file must be in the Intel Hexadecimal format.
- The addresses in the original Hex file might need to be re-based to a starting address of 0x00000000 because the Quartus II programmer loads flash relative to the flash memory base address. Figure 7 shows how the addresses differ between the processor's memory map and the Quartus II programmer for a section of data in flash memory.
- Some parts of the flash memory might be used to store FPGA power-on configuration information, so you must not load software into these regions.

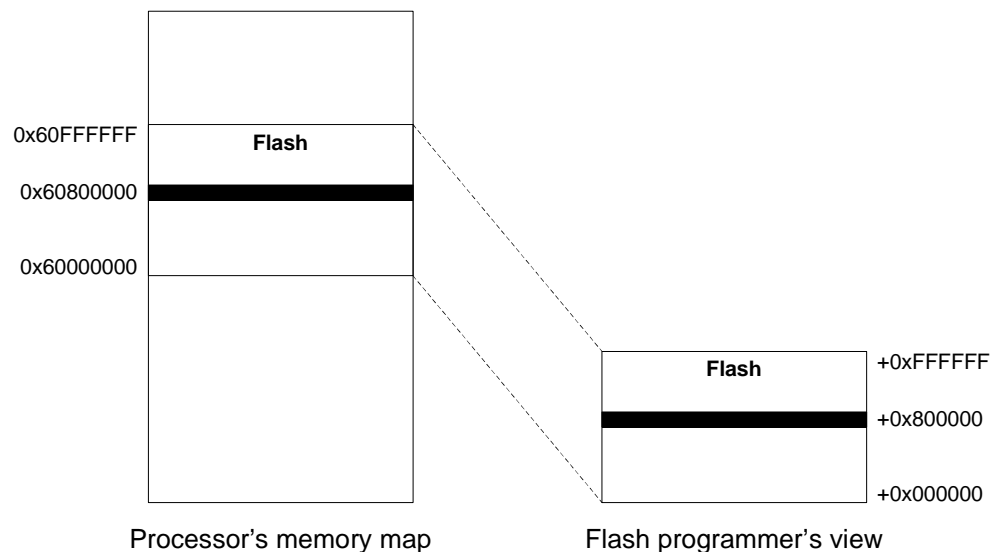


Figure 7 – Re-based addresses for flash memory initialization

4.2 Running hex2hex

The **hex2hex** utility is installed in the `Utilities` subdirectory of the ARM Cortex-M1 FPGA Development Kit installation. You can run it from a Windows command prompt using the **hex2hex.bat** execution wrapper.

Note

hex2hex requires a valid installation of the Altera Quartus II software.

The **hex2hex** utility takes several arguments and options. The complete set of options are documented in *Application Note 215: Converting memory initialization files in the ARM Cortex-M1 FPGA Development Kit Altera Edition*, which you should read to become familiar with the utility. For most flash memory initialization tasks, you can use the following arguments:

- **--infile=***file*, where *file* is the filename of the RealView MDK Hex file. This tells the **hex2hex** utility the file from which it should read data.
- **--outfile=***file*, where *file* is the output filename that will contain data for flash memory initialization. The **hex2hex** utility will create this file.
- **--oformat=***ihex*, which configures **hex2hex** to produce an Intel Hexadecimal formatted output file.
- **--saddr=***address*, where *address* is the start address of the flash data in the input Hex file. The address is the byte address of the data as it appears in the processor's address map.
- **--eaddr=***address*, where *address* is the address of the final data for flash memory in the input Hex file. The address is the byte address of the data as it appears in the processor's address map.

The output file will be re-based to an address of 0x00000000 by default.

For example, if your input Hex file is named `project.hex` and contains flash memory initialization data from 0x60000000 to 0x6000FFFF, you can use the following command line to create a flash memory initialization Hex file named `flash.hex`:

```
hex2hex.bat --infile=project.hex --outfile=flash.hex
--oformat=ihex --saddr=0x60000000 --eaddr=0x6000FFFF
```

Note

Instead of using the `saddr` and `eaddr` arguments, you can use `saddr` and `osize` to specify a base address and region size. Refer to *Application Note 215: Converting memory initialization files in the ARM Cortex-M1 FPGA Development Kit Altera Edition* for complete information about the **hex2hex** options.

4.3 Running hex2hex in RealView MDK

You can configure RealView MDK to run the **hex2hex** utility automatically each time your software project is compiled. This means that your flash memory initialization files, as well as other initialization files such as those for the Cortex-M1 TCMS, will be kept up-to-date.

To configure RealView MDK to run **hex2hex** automatically, follow these steps:

1. Open the RealView MDK project options for your current project.
2. In the project options window, select the **User** tab as shown in Figure 8 on page 4-3.
3. In the **Run User Programs After Build/Rebuild** section, place a tick in the box beside **Run #1** and enter your **hex2hex** command-line in the adjacent text field. You may use `#H` as the value for the `infile` argument, which MDK will automatically replace with the name of the project Hex file. For example:

```
C:\ARM\CortexM1_DevKit\Utilities\hex2hex.bat
--infile="#H" --outfile=C:\MyProject\flash.hex
--oformat=ihex --saddr=0x60000000 --eaddr=0x6000FFFF
```

You should replace the output file name and the other options with your own preferences.

Note

The `#H` string is an example of a RealView MDK Key Sequence, which have special meanings when entered into user commands. For a complete list of MDK Key Sequences, refer to the RealView MDK help.

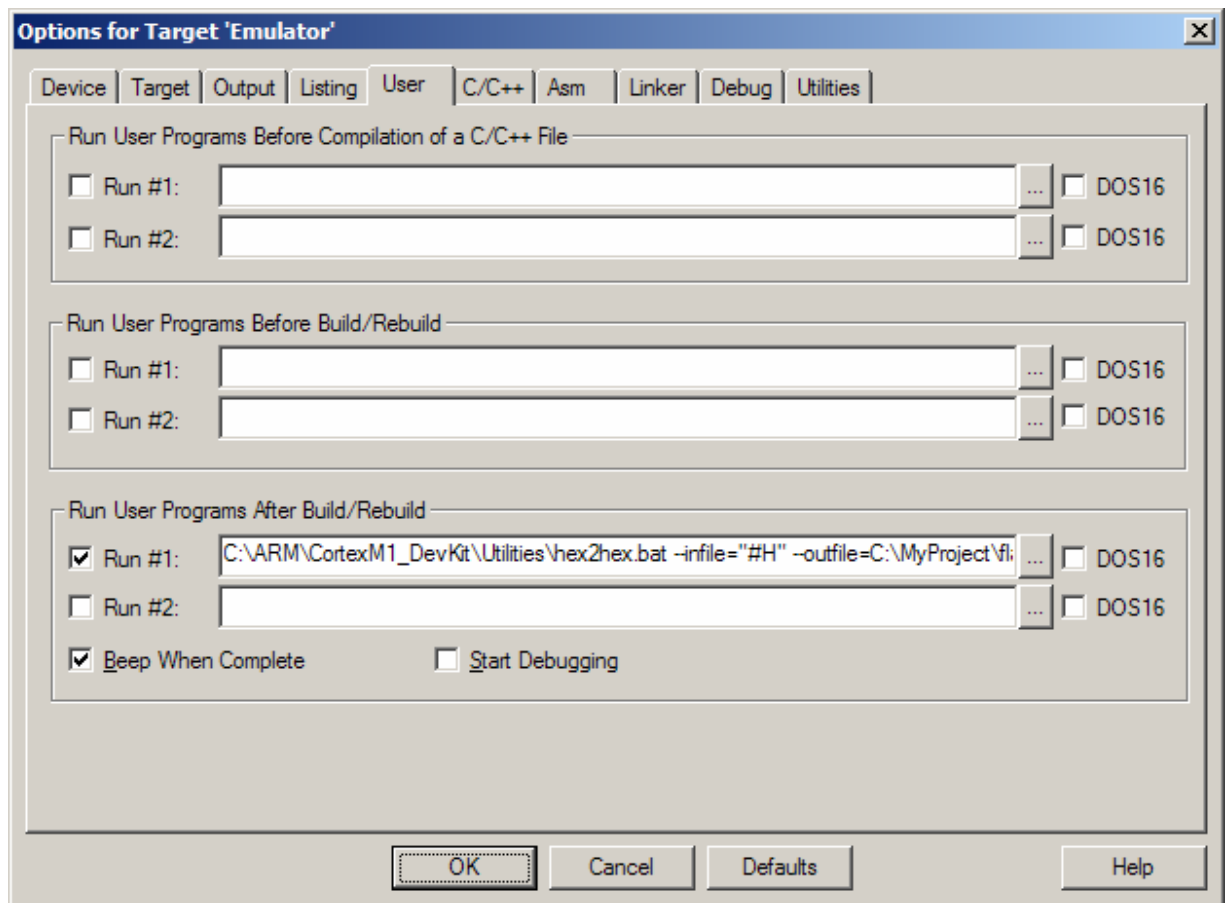


Figure 8 – RealView MDK User options

When you build your project files, RealView MDK will run the user commands to create the additional Hex files. If there are any errors or warnings from the user commands, these will be printed in the RealView MDK output window.

5. References

This Application Note refers to the following ARM documentation:

- *ARM Cortex-M1 FPGA Development Kit Altera Cyclone III Edition Cortex-M1 User Guide* (ARM DUI 0395)
- *Application Note 213: Cortex-M1 TCM initialization in the ARM Cortex-M1 FPGA Development Kit Altera Edition* (ARM DAI 0213)
- *Application Note 215: Converting memory initialization files in the ARM Cortex-M1 FPGA Development Kit Altera Edition* (ARM DAI 0215)

You can also refer to the following online resources for further related documentation:

- <http://infocenter.arm.com> for access to ARM documentation
- <http://www.altera.com> for access to Altera documentation