Application Note 213

Cortex-M1 TCM initialization in the ARM Cortex-M1 FPGA Development Kit Altera Edition

Document number: ARM DAI 0213A
Issued: 2nd September, 2008
Copyright ARM Limited 2008
Application Note 213
Cortex-M1 TCM initialization in the ARM Cortex-M1 FPGA Development Kit Altera Edition

Copyright © 2008 ARM Limited. All rights reserved.

Release information

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2008</td>
<td>A</td>
<td>First release</td>
</tr>
</tbody>
</table>

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. Memory initialization requirements .............................................................................. 2-1
   2.1 Instruction memory and the vector table .............................................................. 2-1
   2.2 Data memory ........................................................................................................... 2-2

3. Creating memory initialization files ............................................................................ 3-1
   3.1 Using the hex2hex conversion utility to create TCM initialization files ............ 3-1
   3.2 Calling hex2hex automatically in RealView MDK .............................................. 3-1

4. Using Hex files to initialize TCMs .............................................................................. 4-1
   4.1 Configuring the ARM Cortex-M1 SOPC Builder component .............................. 4-1
   4.2 Copying memory initialization files to the project directory .............................. 4-2
   4.3 Synthesize the project ......................................................................................... 4-2

5. References .................................................................................................................... 5-1
1. Introduction

The ARM Cortex-M1 processor has optional, configurable Tightly Coupled Memories (TCMs) for instructions and data. Highest performance is achieved when code is run from the TCMs, and you can also pre-initialize the TCMs when the processor is synthesized so that they contain known data when the device is powered on.

This Application Note describes how you can initialize the processor TCMs in the Cortex-M1 FPGA Development Kit, Altera Cyclone III Edition. It also shows how to convert the memory images that are produced by RealView MDK into a format that is suitable for Altera’s Quartus II toolchain. The same technique also applies to other on-chip memories that you may have in your ARM Cortex-M1 based system.

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.
2. Memory initialization requirements

The ARM Cortex-M1 processor loads the exception vector table from memory address 0x00000000 at reset. The vector table must be initialized with pointers to the exception handlers in your software before the processor comes out of reset.

When it is enabled, the Instruction Tightly Coupled Memory (ITCM) starts at 0x00000000 in the processor’s memory map. The vector table will therefore reside in the ITCM when the ITCM is present.

2.1 Instruction memory and the vector table

You should initialize the ITCM so that it contains at least an initialized vector table. You may also decide to initialize the ITCM with your entire software image because the ARM Cortex-M1 processor can achieve better performance when it is executing code from the TCMs. If you only initialize the vector table, then you must ensure that other memory in your system is initialized to contain startup code.

If you configured the processor to have no ITCM then you must ensure that some external memory is available at address 0x00000000 and that this is initialized to contain a vector table.

The following sections summarize some of the instruction memory initialization techniques that you can use with your software.

2.1.1 Initializing the ITCM with the entire software image

If your code fits inside the ITCM then you can initialize the ITCM with the vector table and the entire software image. When the processor is reset the ITCM will already be initialized and the processor will start to execute the software immediately. This method has several advantages:

- no boot loaders are required;
- highest performance is achieved when executing from TCMs.

However, you would need to re-synthesize the system to change the contents of the ITCM.

Note
On-chip memories, such as those used for the ARM Cortex-M1 TCMs, are initialized during the FPGA’s configuration phase. This is typically when the device is powered on or when a new image is programmed using a download cable. Resetting the processor does not reset the on-chip memories, so you must ensure that software running on the processor does not accidentally overwrite the instruction code or else the software might not work correctly when the processor is reset. You can configure the processor to have a read-only ITCM to protect against this.

2.1.2 Initializing the ITCM with vectors and using boot code

You can initialize the ITCM with initial vectors pointing to code in external memory that copies the software image into the ITCM before executing it. Once the software image has been copied to the ITCM, the bootloader can either branch to the ITCM code or, after updating the vectors, request a reset.

The biggest advantage of this technique is that the software can be changed without re-synthesizing the processor.

If you use the boot loader method, you should ensure that:

- the processor has been configured to have a read/write ITCM so that the boot loader can modify the contents of the ITCM;
Memory initialization requirements

- your boot code includes appropriate handlers for exceptions that may occur while executing the boot code, for example Non-Maskable Interrupt, HardFault and any exceptions that your boot code enables;
- your boot code updates the ITCM vectors to their application run-time values;
- your external memory has been initialized with the software image and boot code.

2.2 Data memory

You can also initialize the Data Tightly Coupled Memory (DTCM) if your startup code requires initialized data.
3. Creating memory initialization files

The Keil RealView MDK software uses the ARM RealView Compiler Tools to generate images in AXF format, but there is also an option to produce a Hex image in the Intel Hexadecimal format. The Intel Hexadecimal format for memory images can be read by many memory programmers.

RealView MDK uses the fromelf utility in the ARM RealView Compiler Tools to generate the Intel Hexadecimal image. You can also run fromelf manually to generate a variety of other memory initialization image formats, and information about how to use this utility is provided in the RealView MDK’s online help system.

The Altera Quartus II software uses a variation of the Intel Hexadecimal image format which has a number of differences. You cannot use the Intel Hexadecimal files that RealView MDK produces to initialize on-chip memories, such as the ARM Cortex-M1 TCMs, using Altera Quartus II. However, the ARM Cortex-M1 FPGA Development Kit includes a hex2hex utility that you can use to convert the Intel Hexadecimal files into the Altera Quartus II format.

The rest of this section describes how to use the hex2hex memory image conversion utility and how you can configure RealView MDK to call this utility automatically each time your software is compiled.

3.1 Using the hex2hex conversion utility to create TCM initialization files

The RealView MDK project Hex file must be converted into the Altera Quartus II format before being used for TCM initialization. If your RealView MDK project Hex file contains data for more than one memory, you also need to extract the appropriate section for each memory. The hex2hex utility can do this.

You should refer to Application Note 215: Converting memory initialization files in the ARM Cortex-M1 FPGA Development Kit Altera Edition to learn how to use the hex2hex utility. The following example shows a typical command line for creating an ITCM initialization file. You can use a similar command line to create a DTCM initialization file, if required.

hex2hex.bat --infile=input.hex --outfile=itcm.hex --saddr=0x00000000 --osize=32K --oformat=hex --padding

This command:
- reads input data from input.hex;
- writes the output file to itcm.hex;
- converts a 32KB data section starting at the ITCM address 0x00000000 in input.hex;
- pads any empty address space in the output file with zeroes.

Note
You must modify the example command line to use the correct input Hex file and ITCM configuration. Refer to the Application Note 215: Converting memory initialization files in the ARM Cortex-M1 FPGA Development Kit Altera Edition for information about the hex2hex command-line options.

3.2 Calling hex2hex automatically in RealView MDK

You can configure your RealView MDK software projects so that the hex2hex utility is automatically executed every time you compile your project. This will ensure that your initialization files are always up-to-date with your RealView MDK project, and removes the need to run the command manually each time the project is compiled. To set up this process, follow the steps listed in the headings below.
3.2.1 Enable Hex file output

Open the options window for your RealView MDK project and select the Output tab. Ensure that the Create HEX File option is selected, as shown in Figure 1. This option causes RealView MDK to generate an image in the Intel Hexadecimal format.

![Options for Target 'Emulator'](image)

Figure 1 – MDK Output options window

The Intel Hexadecimal file is saved in the `obj` subdirectory of your RealView MDK project when the project is compiled. The filename matches the name supplied in the Name of Executable field in the Output options tab, which is also shown in Figure 1, with the addition of a `.hex` file extension.
3.2.2 Set up user program execution

In the User tab of the project options window, enable the Run #1 option under the Run User Program After Build/Rebuild section. Enter the following command in the adjacent text box, all on the same line, as shown in Figure 2. You can adjust the arguments to hex2hex.bat to suit your project.

C:\ARM\CortexM1_DevKit\Utilities\hex2hex.bat --infile="#H" --outfile="C:\MyProject\%H" --saddr=0x00000000 --osize=32K --oformat=hex --padding

You might need to change the path to hex2hex.bat depending on where you installed the ARM Cortex-M1 FPGA Development Kit.

RealView MDK automatically translates the #H argument to the --infile option to the name of the project’s Intel Hexadecimal file, including the complete path. The %H argument to the --outfile option is translated into the name of the Intel Hexadecimal file without the path. Therefore the example command above will save the output Hex file to the given directory C:\MyProject using the same filename as the original Intel Hexadecimal file.

The #H and %H tokens are two examples of Key Sequences which have special meanings in RealView MDK commands. For a complete list of available Key Sequences, refer to the RealView MDK online help by pressing the Help button in the project options window.

If a second Hex file is required, for example for DTCM initialization, you can enable the Run #2 option and enter a similar command using hex2hex.

Press the OK button in the options window to confirm the changes.
3.2.3 Compiling the project

When you build your RealView MDK project in the normal way, the commands that you entered in the user programs section of the project options window will be executed after the project’s Intel Hexadecimal file has been generated.
4. Using Hex files to initialize TCMs

Altera Quartus II uses Hex files to initialize on-chip memories such as the ARM Cortex-M1 TCMs. These Hex files are in the Altera Hexadecimal format and you need one Hex file for each memory that you want to initialize.

You must ensure that each Hex file:

- uses relative addresses, relative to the base address of the memory;
- uses word addressing;
- has data records matching the word size of the memory.

Section 3 describes how Hex files meeting these requirements can be generated using the ARM Cortex-M1 FPGA Development Kit hex2hex utility.

4.1 Configuring the ARM Cortex-M1 SOPC Builder component

In the ARM Cortex-M1 FPGA Development Kit, you configure the processor TCM initialization options in the SOPC Builder environment in the Altera Quartus II design software. Figure 3 shows the configuration dialog that appears when you instantiate the ARM Cortex-M1 processor in SOPC Builder. You can also change the configuration options by double-clicking on a processor that has already been instantiated in the SOPC Builder design pane.
To enable ITCM initialization:

- tick the Initialize ITCM contents check-box; and
- enter the name of the memory initialization file in the From file box.

**Note**
You can also use the ITCM Read-only option to prevent software running on the processor from writing to the ITCM. This can be useful if you are initializing the ITCM to include your entire software image and want to prevent accidental corruption and be sure that the processor will use the intended code after a reset. This only affects writes from the processor core; a debugger attached to a debug version of the processor can always write to the ITCM.

Press Finish in the processor configuration dialog to apply the changes. After configuring your entire SOPC Builder system, you must generate it by pressing the Generate button in the main SOPC Builder window. This will apply your configuration options and generate the system RTL.

### 4.2 Copying memory initialization files to the project directory

During synthesis the Quartus II software will search your Quartus II project directory for memory initialization files. You must copy all memory initialization files to the Quartus II project directory, otherwise you will see synthesis warnings relating to missing files and your initialization settings will not take effect.

If you want to simulate your design then you will also need to copy the memory initialization files into your simulation project directory.

### 4.3 Synthesize the project

You can now synthesize the Quartus II project. The TCM initialization data will be included in the FPGA programming file so the memories will be initialized when the FPGA is configured.
5. References

This Application Note refers to the following ARM documentation:


- *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](http://infocenter.arm.com) for access to ARM documentation

- [http://www.altera.com](http://www.altera.com) for access to Altera documentation