Migrating Application Code from ARM Cortex-M4 to Cortex-M7 Processors

Joseph Yiu and Robert Boys **January 2015** **Version 1.1**

The latest version of this document is here: www.keil.com/appnotes/docs/apnt_270.asp

1 Cortex®-M7 Processor Technical Overview

The Cortex-M7 processor is a very high performance 32 bit processor designed for a wide range of embedded applications including microcontrollers, automotive controllers, industrial control systems and wireless communication controllers (e.g. WiFi). For more information visit: www.arm.com/products/processors/cortex-m/cortex-m7-processor.php.

Key characteristics of the Cortex-M7 processor include:

1. High performance 6-stage pipeline with dual-issue (it executes up to two instructions per clock cycle).
2. A 64-bit AXI bus interface as the system bus.
3. Optional instruction cache (4 to 64KB) and data cache (4 to 64KB), with optional ECC (Error Correction Code) support for each of the cache memories.
4. Optional 64-bit Instruction Tightly Coupled Memory (I-TCM), and optional dual 32-bit Data TCM (D-TCM), with support for a custom ECC implementation for each of the TCM interfaces.
5. Optional low-latency AHB peripheral bus interface (referred as AHBP in ARM documents).
6. Integrated Nested Vectored Interrupt Controller (NVIC) with 1 to 240 interrupts with 3 to 8-bit programmable priority level registers.
7. Optional Memory Protection Unit (MPU) with 8 or 16 regions.
8. Optional Floating Point Unit (FPU) with support for single- and double-precision floating point instructions.
9. Support for sleep modes and various low power implementation technologies.
10. Powerful debug features, with optional full instruction and data trace.

Options listed are chosen by the manufacturer of your device. Check the specific device data sheet.

The Cortex-M7 processor is designed based on the ARMv7-M architecture. It supports all the instructions available on the Cortex-M4 processor and uses the same exception model for interrupt handling. In most cases, program code written for a Cortex-M4 processor will run on the Cortex-M7 processors without any problems. However, there are a few cases where some changes could be needed. In this document, we will cover some of the areas that software developers should know about when migrating applications from the Cortex-M3 or a Cortex-M4 to the Cortex-M7 processor.

2 Tool chain updates

Compilation suite

In order to get the best performance out the Cortex-M7 processor, a number of C compilers and their runtime libraries have been optimized and updated.

In addition, a number of changes in the debug system for the Cortex-M7 processor compared to Cortex-M4 means that software developers must update their tool chain to a newer version in order to debug applications on Cortex-M7 based microcontroller products. In a few cases, the firmware on the debug adaptor might also need an update.

As a result, an update to latest development tool chain is strongly recommended.

Keil MDK officially supports the Cortex-M7 starting with version 5.12.

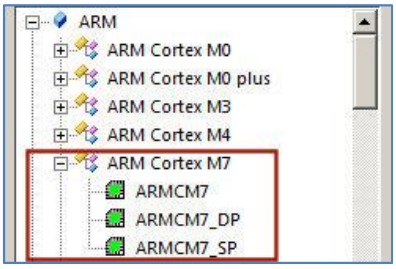
CMSIS-PACK (Software Packs)

Keil MDK 5 uses Software Packs to distribute processor specific components including header, Flash programming, startup and example files, documentation and board support. These Packs are not exclusive to MDK as any tool vendor can choose to use them. The current public Packs are listed at www.keil.com/dd2/packs.

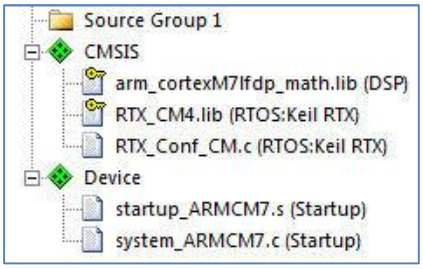
At the time of this writing, Software Packs and other files for Cortex-M7 processors are distributed by the silicon providers. Contact the manufacturer of your processor for these files. This is due to the new introduction of the Cortex-M7.

Until these Packs are provided on the Keil website, µVision® provides three Cortex-M7 targets in the Target Options window that you can use as shown below on the left. A Software Pack by the silicon provider will provide specific part numbers in this window for their Cortex-M7 processors. We recommend you enquire to the availability of such a Pack as they will soon be available.

Provided with MDK are the startup code, CMSIS-DSP and RTX files. This is shown below right. Note RTX uses the compatibility between Cortex-M processors to use the Cortex-M4 library (RTX_CM4.lib) with the Cortex-M7. You can create programs and run them in the Keil Simulator or on a Cortex-M7 target board.



Cortex-M7 Generic Device database



A sample CMSIS project for generic Cortex-M7 with Startup, DSP and RTX source files listed.

RTX is a full feature RTOS for ARM processors. RTX is CMSIS-RTOS compliant and comes with a BSD license. This makes it free. Source code and libraries are provided in all versions of MDK. There are ports provided for various tool chains including Keil and GCC. The RTX manual is located at www.keil.com/pack/doc/cmsis_rtx/index.html.

For more information about Keil MDK, please refer to: Getting Started Keil MDK 5: www.keil.com/mdk5/

3 General software changes

Typically the following changes should be done when migrating software from a Cortex-M3 or a Cortex-M4 to a Cortex-M7 processor:

1) Update the CMSIS-CORE header to use the Cortex-M7 header files. The CMSIS-CORE header files for the Cortex-M7 processor are available starting with CMSIS version 4.2. This version includes a number of cache maintenance functions. Additional cache maintenance functions are to be included in version 4.3. You can get the most updated CMSIS package from www.keil.com/dd2/pack/ under the ARM heading or www.arm.com/cmsis. CMSIS manuals are found here and in any MDK installation: www.keil.com/pack/doc/CMSIS/General/html/index.html

2) Update the CMSIS-DSP library to the Cortex-M7 specific version. The Cortex-M7 specific version is optimized for the pipeline behavior of the Cortex-M7 processor and therefore can offer higher performance.

3) New APIs are included in the CMSIS-CORE headers for cache configuration. If the Cortex-M7 device you are using executes a program from a slow memory (e.g. flash memory) via the AXI interface, you should enable both the I-cache and D-cache for better performance.

4) In addition, for best performance, all program code and software code should be recompiled in order to allow the compiler to optimize the instruction sequencing better for the Cortex-M7 processor pipeline.

5) Adjustments for floating point calculations (if any). See the Floating Point entry on the next page.

4 Cache specific changes

In some cases, additional cache maintenance operation might be needed during runtime. For example, this can be needed if a cacheable memory location is shared between the processor and a separate bus master (e.g. a DMA controller):

A. If the memory location updated by the Cortex-M7 processor has to be accessed by another bus master, a cache clean is needed to ensure the other bus master can see the new data.

B. If the memory location has been updated by a different bus master, the Cortex-M7 processor has to do a cache invalidate so that next time it reads the memory location, it will fetch the information from the main memory system.

In case you need to power down the processor to preserve power, you might also want to clean the cache before the power down process if the microcontroller you use does not support cache data retention. Please check with the microcontroller vendor for details.

Cache functions are found in the CMSIS-Core file core_cm7.h. This file is found in CMSIS version 4.2 and later.

5 Floating Point Unit

The Cortex-M7 processor supports several floating point support options:

1. No floating point unit (FPU)
2. Single precision floating point unit
3. Singe and double precision floating point unit

If your application can benefit from double precision floating point unit support, you should update and recompile your application to make use of the double precision FPU. Depending on the tool chain used (e.g. GCC), some of the compilation options might need to be changed.

- If migrating from a Cortex-M3 to a Cortex-M7 processor, you need to enable floating point option (or add additional command line options if using gcc in the command line) to enable the use of the floating point unit.
- Depending on the floating point calculation required and the floating point unit available on the Cortex-M7 you are using, you need to adjust the floating point options accordingly.
- Double precision floating point calculations on Cortex-M3 and Cortex-M4 processors are often compiled with a soft floating point ABI. On the Cortex-M7 processor you might want to switch to a hard floating point ABI for best performance if the floating point unit supports double precision.

Even if your application uses only single precision floating point operations and it is already compiled for the Cortex-M4 processor with floating point, recompiling your code for the Cortex-M7 processor can also be benefical because the FPU in the Cortex-M7 is based on FPv5, whereas the FPU in the Cortex-M4 processor is FPv4 based. The FPv5 has additional floating point processing instructions which might help speed up the floating point data processing in your application.

6 Other potential software changes

There are a number of potential areas in the program code that might need changes:

Program execution timing related – due to the higher performance of the processor, it is possible that some program code might need adjustment due to much faster execution. This is most common for applications that use hard coded timing delay loops.

Memory address related – Often system memory maps change when migrating from one microcontroller device to another. Also, in the Cortex-M7 processor, the initial vector table does not necessarily start in address 0x00000000. If your application code assumes the initial vector table is at address 0, you might need to update the code so that it can determine the initial vector table location from the Vector Table Offset Register.

Memory barrier – Due to the multiple bus interface and more capable write buffers in the Cortex-M7 processor, you might find that you need to insert additional memory barrier instructions in your program code. The guide for memory barrier usage is documented in ARM application note AN321 – ARM Cortex-M Programming Guide to Memory Barrier Instructions. You can also search for *memory barrier* on www.arm.com to find this document. In the Cortex-M4 processor, due to the simpler nature of the pipeline, omitting the memory barrier usually does not cause any issues. In the Cortex-M7 processor, the memory barrier requirements are more strict.

7 Unsupported Cortex-M4 features

Most of the features in the Cortex-M4 processor are supported in the Cortex-M7 processor with four exceptions:

1. **Bit band** – The bit band feature is not available internally inside the Cortex-M7 processor. Potentially, a chip designer can add a bus level bit band component for the peripheral region, but there is no bit band support for the SRAM region. Application code that uses the bit band feature might need to change the code to use a software read-modify-write sequence:
   A. If the code is for peripheral control and a bus level bit band wrapper is available, you might not need to change the program code.
   B. If the code is for peripheral control and the peripheral used has equivalent bit set/clear features, you can modify the code to use the peripheral specific feature.
   C. If the code is for peripheral control and there is no bit band or bit level access feature in the peripheral, you need to change the code to carry out software read-modify write sequences.
   D. If the code is for bit band in SRAM region, you need to change the code to carry out software read-modify-write sequences, or use other features (e.g. semaphore operations that can be carried out using exclusve accesses).
2. **Flash patch** – There is no flash patch feature in the Cortex-M7 processor. Most of these microcontroller devices are flash memory based so the flash patch feature is rarely needed. In case you are using OTP (One Time Programmable) devices or mask ROM devices, please contact your device supplier to see if a device specific patching mechanism has been included.
3. **Auxilary Control Register** changes – In the Cortex-M3 and Cortex-M4 processors, the Auxiliary Control Register allows the default write buffer in the processor to be disabled (this can be overridden by the MPU settings), and allows the interruption of multi-cycle instructions (e.g. LDM, STM, 64-bit multiply and divide instructions) to be disabled. These settings are not available in the Cortex-M7 processor.
4. **Configurable double word stack alignment** – In the Cortex-M3 and Cortex-M4 processors, the Configuration and Control Register (CCR) has a bit (`STKALIGN`) to enable/disable double word stack alignment on exception entry. Due to the use of 64-bit memory buses in the Cortex-M7 processor, this bit is fixed to 1 and therefore always enforces double word stack alignment on exception entries. The side effect is a potential small stack size increase for exception stack frames if your application previously disabled this `STKALIGN` bit.

8 Additional information

Technical details of the Cortex-M7 processor are now available on the ARM web site:

- ARM Cortex-M7 Processor Technical Reference Manual (rev r0p2) : http://infocenter.arm.com/help/topic/com.arm.doc.ddi0489b/index.html
- ARMv7-M Architecture Reference Manual (Issue E.b) : http://infocenter.arm.com/help/topic/com.arm.doc.ddi0403e.b/index.html

9 Document Resources:

Books:

1. ***NEW!*** **Getting Started Keil MDK 5:** Obtain this free book here: www.keil.com/mdk5/.
2. There is a good selection of books available on ARM processors. A good list of books on ARM processors is found at www.arm.com/support/resources/arm-books/index.php.
3. µVision contains a window titled Books. Many documents including data sheets are located there.
4. **A list of resources is located at:** www.arm.com/products/processors/cortex-m/index.php
   Click on the Resources tab. Or select the Cortex-M processor you want in the Processor panel on the left.
5. Or search for "Cortex-M7" on www.arm.com and click on the Resources tab.
6. The Definitive Guide to the ARM Cortex-M0/M0+ by Joseph Yiu. Search the web for retailers.
7. The Definitive Guide to the ARM Cortex-M3/M4 by Joseph Yiu. Search the web for retailers.
8. Embedded Systems: Introduction to Arm Cortex-M Microcontrollers (3 volumes) by Jonathan Valvano.

Application Notes: www.keil.com/appnotes

1. ***NEW!*** ARM Compiler Qualification Kit: Compiler Safety Certification: www.keil.com/safety
2. Using Cortex-M3 and Cortex-M4 Fault Exceptions www.keil.com/appnotes/files/apnt209.pdf
3. Segger emWin GUIBuilder with µVision™ www.keil.com/appnotes/files/apnt_234.pdf
4. Porting mbed Project to Keil MDK™ www.keil.com/appnotes/docs/apnt_207.asp
5. MDK-ARM™ Compiler Optimizations www.keil.com/appnotes/docs/apnt_202.asp
6. Using µVision with CodeSourcery GNU www.keil.com/appnotes/docs/apnt_199.asp
7. RTX CMSIS-RTX Ports in MDK 5 Eval Version: C:\Keil_v5\ARM\Pack\ARM\CMSIS\
8. Barrier Instructions http://infocenter.arm.com/help/topic/com.arm.doc.dai0321a/index.html
9. Lazy Stacking on the Cortex-M4: www.arm.com and search for DAI0298A
10. Cortex Debug Connectors: www.arm.com and search for cortex_debug_connectors.pdf or www.keil.com/coresight/coresight-connectors/
11. Sending ITM printf to external Windows applications: www.keil.com/appnotes/docs/apnt_240.asp

ARM Community Forums: www.keil.com/forum and http://community.arm.com/groups/tools/content

ARM University program: www.arm.com/university. Email: university@arm.com

ARM Accredited Engineer Program: www.arm.com/aae

mbed™: http://mbed.org

For more information on the ARM CMSIS standard: www.arm.com/cmsis.

Keil Technical Support in USA: support.us@keil.com or 800-348-8051. Outside the US: support.intl@keil.com

For comments, corrections or additions, please email bob.boys@arm.com