

ARM® Cortex®-M for Beginners

An overview of the ARM Cortex-M processor family and comparison

Joseph Yiu

September 2016

Abstract

The ARM® Cortex®-M family now has six processors. In this paper, we compare the features of various Cortex-M processors and highlight considerations for selecting the correct processor for your application. The paper includes detailed comparisons of the Cortex-M instruction sets and advanced interrupt capabilities, along with system-level features, debug and trace features, and performance comparisons.

1 Overview

Today, there are six members in the ARM Cortex-M processor family. In addition, there are many other ARM processors in the ARM product portfolio. For many beginners, or even for experienced chip designers who are not familiar with ARM architecture, this can be a bit confusing. Different processors can have different instruction set support, system features and performance. In this article, I am going to capture the key differences between various Cortex-M processors, and how they compare to other ARM processors.

1.1 The ARM Processor family

Over the years, ARM has developed quite a number of different processor products. In the following diagram (fig 1), the ARM processors are divided between the classic ARM processors and the newer Cortex processor product range. In addition, these processors are divided into three groups based on the application spaces:

Application Processors – High-end processors for mobile computing, smart phone, servers, etc. These processors run at higher clock frequency (over 1GHz), and support Memory Management Unit (MMU), which is required for full feature OS such as Linux, Android, MS Windows and mobile OSs. If you are planning to develop a product that requires one of these OSs, you need to use an application processor.

Real-time Processors – These are very high-performance processors for real-time applications such as hard disk controller, automotive power train and base band control in wireless communications. Most of these processors do not have MMU, and usually have Memory Protection Unit (MPU), cache, and other memory features designed for industrial applications. They can run at a fairly high clock frequency (e.g. 200MHz to >1GHz) and have very low response latency. Although these processors cannot run full versions of Linux or Windows, there are plenty of Real Time Operating Systems (RTOS) that can be used with these processors.

Microcontroller Processors – These processors are usually designed to have a much lower silicon area, and much high-energy efficiency. Typically, they have shorter pipeline, and usually lower maximum frequency (although you can find some of these processors running at over 200MHz). At the same time, the newer Cortex-M processor family is designed to be very easy to use; therefore, they are very popular in the microcontroller and deeply embedded systems market.

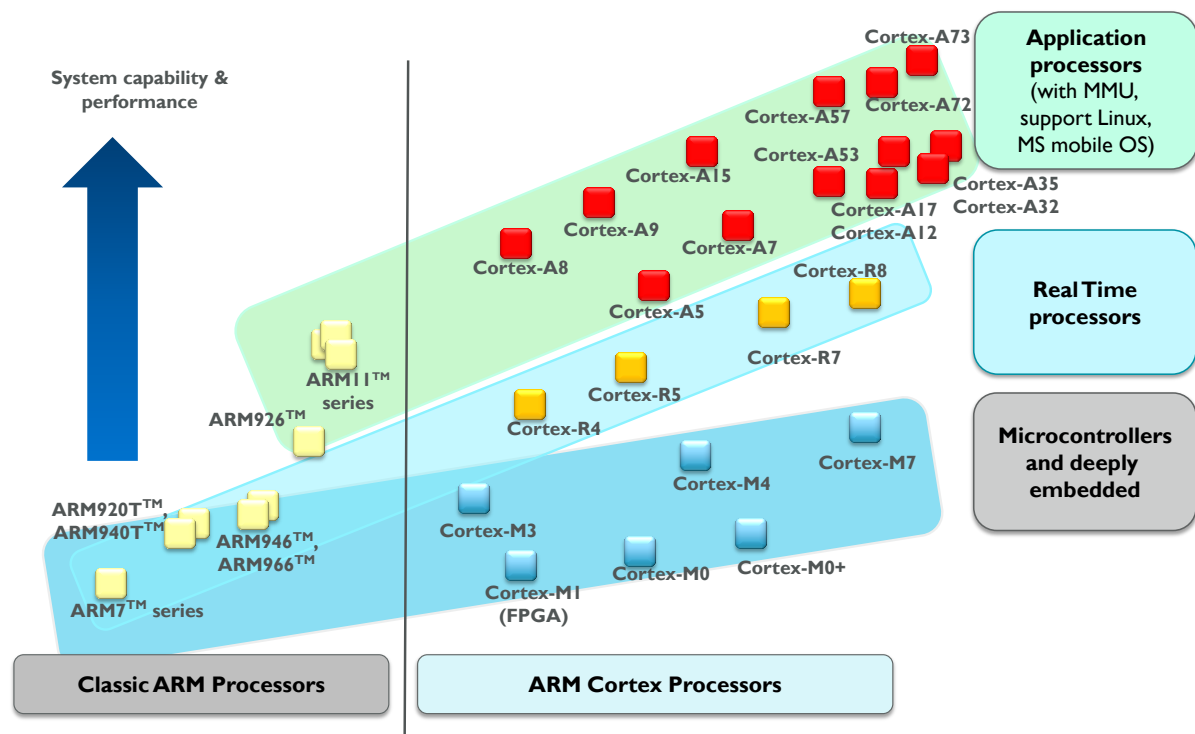


Figure 1: ARM processor family

Table 1 summarized the main characteristics of the three processor ranges.

	Application processors	Real-time processors	Microcontroller processors
Design	High clock frequency, Long pipeline, High performance, Multimedia support (NEON instruction set extension)	High clock frequency, Long to medium pipeline length, Deterministic (low interrupt latency)	Short pipeline, ultra low power, Deterministic (low interrupt latency)
System features	Memory Management Unit (MMU), cache memory, ARM TrustZone® security extension	Memory Protection Unit (MPU), cache memory, Tightly Coupled Memory (TCM)	Memory Protection Unit (MPU), Nested Vectored Interrupt Controller (NVIC), Wakeup Interrupt Controller (WIC)
Targeted markets	Mobile computing, smart phones, energy-efficient servers, high-end microprocessors	Industrial microcontrollers, automotives, Hard disk controllers, Baseband modem	Microcontrollers, Deeply embedded systems (e.g. sensors, MEMS, mixed signal IC), Internet of Things (IoT)

Table 1: Summary of processor characteristics

1.2 The Cortex-M Processor family

The Cortex-M processor family is more focused on the lower end of the performance scale. However, these processors are still quite powerful when compared to other typical processors used in most microcontrollers. For example, the Cortex-M4 and Cortex-M7 processors are being used in many high-performance microcontroller products, with maximum clock frequency going up to over 200MHz.

Of course, performance is not the only factor when selecting a processor. In many applications, low power and cost are the key selection criteria. Therefore, the Cortex-M processor family contains various products to address different needs:

	Descriptions
Cortex-M0	A very small processor (starting from 12K gates) for low cost, ultra low power microcontrollers and deeply embedded applications
Cortex-M0+	The most energy-efficient processor for small embedded system. Similar size and programmer’s model to the Cortex-M0 processor, but with additional features like single cycle I/O interface and vector table relocations
Cortex-M1	A small processor design optimized for FPGA designs and provides Tightly Coupled

Copyright © 2016 ARM Limited. All rights reserved.

The ARM logo is a registered trademark of ARM Ltd.

All other trademarks are the property of their respective owners and are acknowledged

	Memory (TCM) implementation using memory blocks on the FPGAs. Same instruction set as the Cortex-M0
Cortex-M3	A small but powerful embedded processor for low-power microcontrollers that has a rich instruction set to enable it to handle complex tasks quicker. It has a hardware divider and Multiply-Accumulate (MAC) instructions. In addition, it also has comprehensive debug and trace features to enable software developers to develop their applications quicker
Cortex-M4	It provides all the features on the Cortex-M3, with additional instructions target at Digital Signal Processing (DSP) tasks, such as Single Instruction Multiple Data (SIMD) and faster single cycle MAC operations. In addition, it also have an optional single precision floating point unit that support IEEE 754 floating point standard
Cortex-M7	High-performance processor for high-end microcontrollers and processing intensive applications. It has all the ISA features available in Cortex-M4, with additional support for double-precision floating point, as well as additional memory features like cache and Tightly Coupled Memory (TCM)

Table 2: The Cortex-M processor family

Quite different from legacy ARM processors (e.g. ARM7TDMI, ARM9), the Cortex-M processors have a very different architecture. For instance:

- Only support ARM Thumb® instructions, which have been extended to support both 16-bit and 32-bit instructions in Thumb-2.
- Interrupt handling is managed by a built-in interrupt controller called Nested Vector Interrupt Controller (NVIC), which provides automatic prioritization, masking and nesting of interrupts and system exceptions.
- Interrupt handlers can be written as normal C functions and the vectored interrupt handling mechanism avoided the need to use software to determine which interrupt to service. At the same time, interrupt responses are deterministic and have low latency.
- Vector table changed from branch instructions to starting addresses of interrupts and system exception handlers.
- The register bank and some details of the programmer’s model have also been changed.

The changes mean many old assembly code written for classic ARM processors would need modifications, and old projects need to be modified and recompiled when migrated to the Cortex-M world. Detailed information on software porting is documented in an ARM document:

ARM Cortex-M3 Processor Software Development for ARM7TDMI Processor Programmers
http://www.arm.com/files/pdf/Cortex-M3_programming_for_ARM7_developers.pdf

1.3 Common features in Cortex-M processors

There are many similarities between the Cortex-M0, M0+, M3, M4 and M7 processors. For example:

Copyright © 2016 ARM Limited. All rights reserved.

The ARM logo is a registered trademark of ARM Ltd.

All other trademarks are the property of their respective owners and are acknowledged

- Baseline programmer’s model (section 3.1)
- Nested Vectored Interrupt Controller (NVIC) for interrupt management
- Architectural defined sleep modes : sleep and deep sleep (section 4.1)
- OS support features (section 3.3)
- Debug support (section 6)
- Ease of use

For example, the NVIC is an integrated interrupt controller.

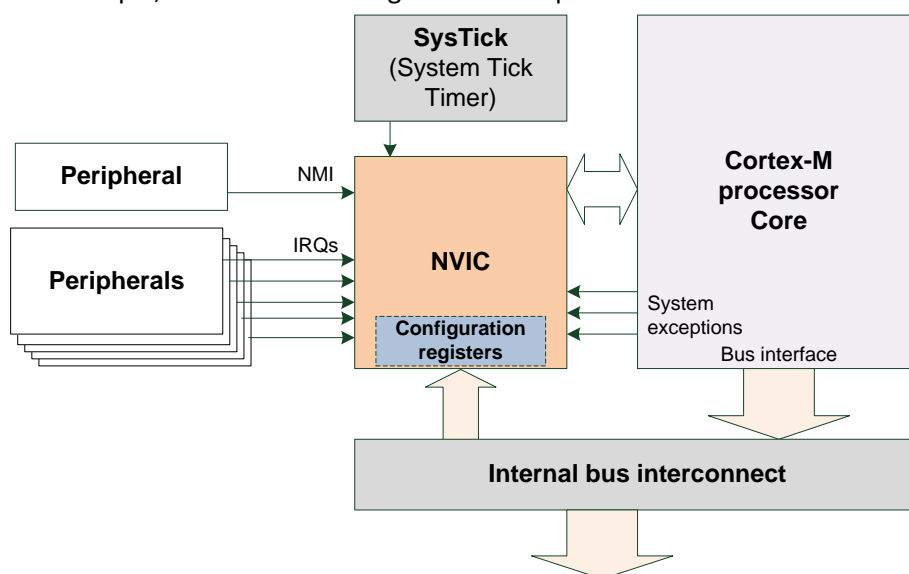


Figure 2: NVIC in Cortex-M processor

The NVIC supports a number of interrupt inputs from peripherals, a Non-Maskable Interrupt request, an interrupt request from a built-in timer called SysTick (see section 3.3) and a number of system exceptions. The NVIC handles the priority management and masking of these interrupt and exceptions.

More information on NVIC and the exception model is covered in section 3.2. Other areas of similarity and difference are covered in the rest of this document.

2 Instruction Set of the Cortex-M processors

2.1 Instruction set overview

In most cases, the application code would be written in C or other high-level languages. However, a basic understanding of the instruction set support in the Cortex-M processor helps to decide which Cortex-M processor is needed for the tasks. The Instruction Set Architecture (ISA) is a part of the processor architecture, and the Cortex-M processors can be grouped in two architecture profiles:

Architecture	Descriptions
ARMv6-M	For Cortex-M0, Cortex-M0+ and Cortex-M1 processors
ARMv7-M	For Cortex-M3, Cortex-M4 and Cortex-M7 processors. The extension of ARMv7-M to support DSP type instructions (e.g. SIMD) is also named as ARMv7E-M.
ARMv8-M	Processor products to be announced. For more information about ARMv8-M architecture, please see ARMv8-M Architecture Technical Overview in https://community.arm.com/docs/DOC-10896

Table 3: ARM Architecture profiles for the Cortex-M processors

All Cortex-M processors support an instruction set called Thumb. The complete Thumb instruction set became fairly large when it was expanded when the Thumb-2 Technology was made available. However, different Cortex-M processors support different subset of the instructions available in the Thumb ISA, as shown in Figure 3.

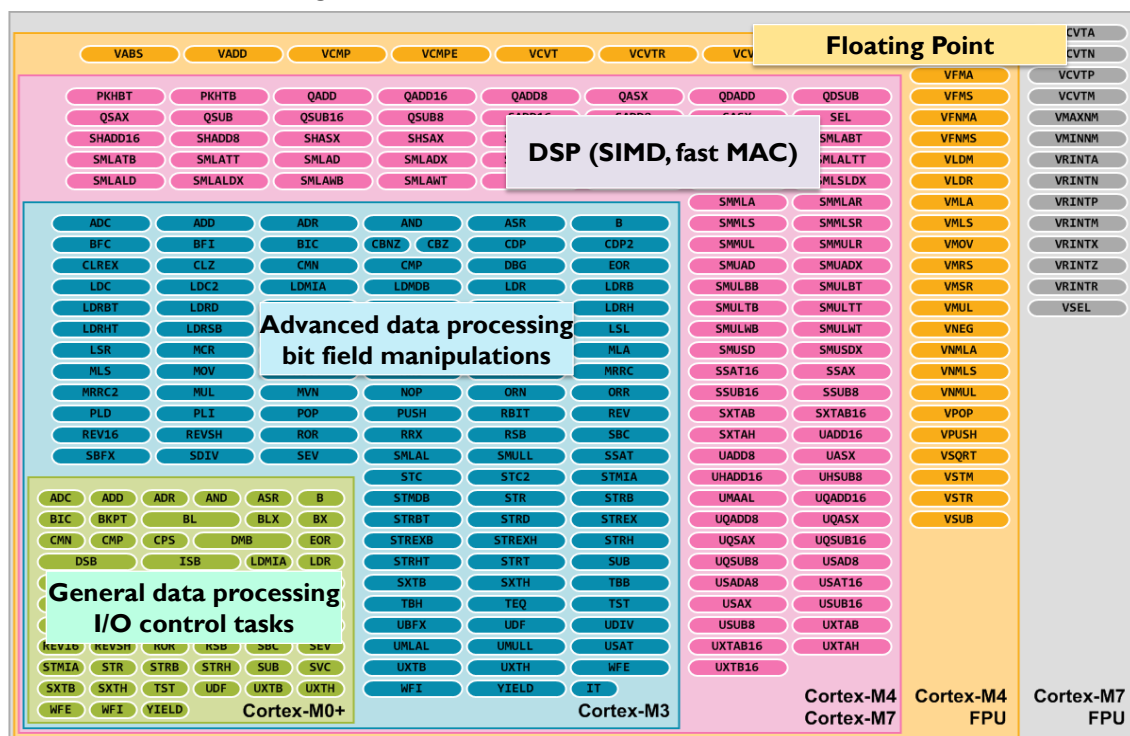


Figure 3: Instruction Set support in the Cortex-M processors

2.2 Instructions support in Cortex-M0/M0+/M1

The Cortex-M0/M0+/M1 processors are based on the ARMv6-M architecture, which has a small instruction set of just 56 instructions and most of them are 16-bit, as shown with the smaller oval shapes in figure 3. However, the registers in the processor and the data being operated on are still 32-bit. For most simple I/O control tasks and general data processing, this small instruction set is sufficient. The small instruction set allows the processor design to be implemented with very small gate count, starting from just 12K gates in the Cortex-M0 and the Cortex-M0+ processors. However, many of these instructions cannot utilize the high registers (R8 to R12), and have limited capability

Copyright © 2016 ARM Limited. All rights reserved.

The ARM logo is a registered trademark of ARM Ltd.

All other trademarks are the property of their respective owners and are acknowledged

of generating immediate data on the fly. This is a compromise between requirements in ultra low-power processor design and the performance available.

2.3 Instructions support in Cortex-M3

The Cortex-M3 processor is based on the ARMv7-M architecture, and supports a much richer instruction set, including many 32-bit instructions that allow the high registers to be utilized efficiently. In addition, it also supports

- Table branch instructions and conditional execution (using IT instruction),
- hardware divide instructions,
- multiply-accumulate (MAC), and
- various bit field operations

The richer instruction enhanced the performance in a number of ways; for example, the 32-bit Thumb instructions provide larger range of immediate data values, branch offset and immediate offset for data memory accesses. It also has basic support for DSP operation (e.g., a few MAC instructions are available, which take multiple clock cycles, and saturation adjustment instructions are also available). Finally, the 32-bit instructions allow the use of the barrel shifter together with many data operations in a single instruction.

The richer instruction set comes at a cost of larger silicon area and higher power. In typical microcontrollers, the gate count of the Cortex-M3 can be more than double of the Cortex-M0 or Cortex-M0+ designs. However, given the silicon of the processor is only a small part in most modern microcontrollers, the larger silicon area and power is often insignificant.

2.4 Instructions support in Cortex-M4

The Cortex-M4 processor is very similar to the Cortex-M3 in many ways: pipeline, programmer's model. It supports all the features in the Cortex-M3, and in additional support various instructions target for DSP applications like SIMD, saturation arithmetic instructions, a wide range of MAC instructions which can execute in single cycles (compared to multiple-cycles and limited selections in the Cortex-M3), and an optional floating unit that support single precision floating point operations.

The SIMD operations in the Cortex-M4 handle two 16-bit data or four 8-bit data in parallel. For example, Figure 4 shows the QADD8 and QADD16 operations:

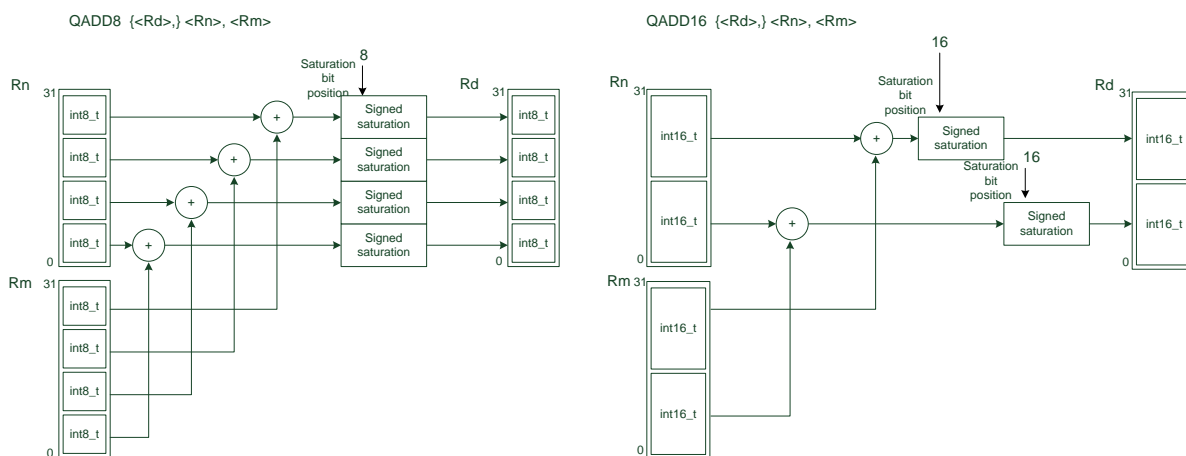


Figure 4: Example of SIMD instructions: QADD8 and QADD16

The uses of SIMD enable much faster computation of 16-bit and 8-bit data in certain DSP operations as the calculation can be parallelized. However, in general programming, C compilers are unlikely to utilize the SIMD capability. That is why the typical benchmark results of the Cortex-M3 and Cortex-M4. However, the internal data path of the Cortex-M4 is different from Cortex-M3, which enable faster operations in a few cases (e.g. single cycle MAC, and allow write back of two registers in a single cycle).

2.5 ISA feature comparison summary

There are many ISA features in the ARMv6-M and ARMv7-M architecture so it is difficult to cover all of these in details. However, the following table (table. 4) summarized the key differences.

	Cortex-M0/M0+	Cortex-M1	Cortex-M3	Cortex-M4	Cortex-M7
Architecture	ARMv6-M	ARMv6-M	ARMv7-M	ARMv7E-M	ARMv7E-M
v4T, v5T, v6-M	Y	Y	Y	Y	Y
Thumb ISA					
v7-M Thumb ISA	-	-	Y	Y	Y
Low power / Sleep mode : WFE, WFI, SEV	Y	Execute as NOP	Y	Y	Y
Single cycle Multiply (32-bit result)	Y	Y	Y	Y	Y

Bit field processing	-	-	Y	Y	Y
Hardware divide (integer)	-	-	Y	Y	Y
Unaligned data access	-	-	Y	Y	Y
Table branch	-	-	Y	Y	Y
Conditional execution (IT)	-	-	Y	Y	Y
Compare & branch (CBZ, CBNZ)	-	-	Y	Y	Y
Floating point	-	-	-	Single precision (optional)	Single precision / Single + double precision (optional)
MAC	-	-	Y (multi-cycle, limited)	Y (single cycle)	Y (single cycle)
SIMD	-	-	-	Y	Y
Saturation	-	-	USAT, SSAT only	Y	Y
Exclusive access	-	-	Y	Y	Y
Memory barrier	Y	Y	Y	Y	Y
SVC	Y	Optional	Y	Y	Y

Table 4: comparison of ISA features

One of the key characteristics of the ISA in the Cortex-M processors is the upward compatibility: Instruction supported in the Cortex-M3 processor is a superset of Cortex-M0/M0+/M1. So theoretically if the memory map is identical, a binary image for Cortex-M0/M0+/M1 can run directly on a Cortex-M3. The same applies to the relationship between Cortex-M4/M7 and other Cortex-M processors; instructions available on Cortex-M0/M0+/M1/M3 can run on a Cortex-M4/M7.

Although the Cortex-M0/M0+/M1/M3 processors do not have floating point unit option, floating-point calculations can be done using software. This also applies to products based on Cortex-M4/M7 without floating point unit. When floating-point data is used in a program, the compiler suite inserts the required runtime library functions during linking stage. Floating-point calculation using software

can take longer and could increase code size slightly. However, if the floating-point calculations are not frequent, it might be suitable for your applications.

3 Architectural features

3.1 Programmer's model

The programmer's model of the Cortex-M processor family is highly consistent. For example, R0 to R15, PSR, CONTROL and PRIMASK are available to all Cortex-M processors. Two special registers - FAULTMASK and BASEPRI - are available only on the Cortex-M3, Cortex-M4 and Cortex-M7, and the floating-point register bank and FPSCR (Floating Point Status and Control Register) is available on the Cortex-M4/M7 within the optional floating-point unit.

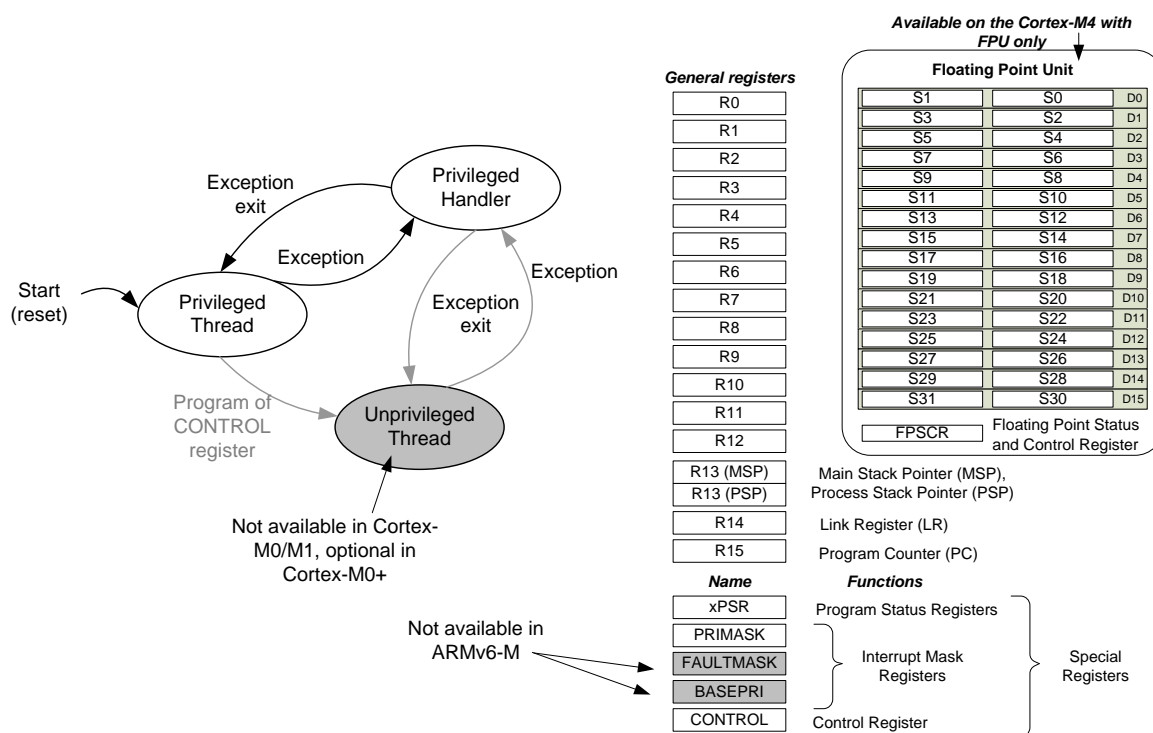


Figure 5: Programmer's model

The BASEPRI register allows you to block exceptions and interrupts of certain priority level or lower priorities. This can be important in ARMv7-M as you can have a large number of priority levels on Cortex-M3, Cortex-M4 and Cortex-M7, whereas it is limited to four program levels on the ARMv6-M. FAULTMASK is typically used in complex fault handlers (see section 3.4).

Unprivileged execution level is optional in ARMv6-M and is always available in ARMv7-M. On the Cortex-M0+ this is optional, and is not available on Cortex-M0 and Cortex-M1. This difference means that the CONTROL register can have minor different between different Cortex-M processors. The FPU option also affects the CONTROL register, as shown in Figure 6.

		31:3	2	1	0
ARMv6-M	CONTROL			SPSEL	nPRIV
Cortex-M3	CONTROL			SPSEL	nPRIV
Cortex-M4/M7 with FPU	CONTROL		FPCA	SPSEL	nPRIV

Figure 6: The CONTROL register

Another difference between the programmer’s model is the details of the PSR (Program Status Register). In all Cortex-M processor, the PSR can be sub-divided into Application PSR, Execution PSR and Interrupt PSR. The Q bit in APSR and ICI/IT bits in EPSR are not available in the ARMv6-M, and the GE bits are available on the ARMv7E-M only (e.g. Cortex-M4). In addition, the width of the interrupt number in IPSR has a smaller range in ARMv6-M. This is shown in Figure 7.

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
ARMv6-M (Cortex-M0/M0+)	N	Z	C	V			T				Exception Number					
ARMv7-M (Cortex-M3)	N	Z	C	V	Q	ICI/IT	T			ICI/IT	Exception Number					
ARMv7E-M (Cortex-M4/M7)	N	Z	C	V	Q	ICI/IT	T		GE[3:0]	ICI/IT	Exception Number					

Figure 7: PSR differences

Please note that the programmer’s model of Cortex-M is different from classic ARM processors such as the ARM7TDMI. Beside from the register bank, the “mode” and “state” definitions of classic ARM processor is different in the Cortex-M. There are only two modes in Cortex-M: Thread and Handler, and Cortex-M processors always operate in Thumb state (ARM instructions are not supported).

3.2 Exception model and NVIC

All the Cortex-M processors include Nested Vectored Interrupt Controller (NVIC) and share the same exception model. When an exception occurred and is higher priority than the current level, and not blocked by any masking registers, the processor accept the interrupt / exception and stack 8 registers to the current stack. This stacking mechanism enable interrupt handlers to be written as

Copyright © 2016 ARM Limited. All rights reserved.

The ARM logo is a registered trademark of ARM Ltd.

All other trademarks are the property of their respective owners and are acknowledged

normal C functions, and enable many small interrupt functions to start actual work immediately as no further stacking is required.

Some interrupts and systems exceptions available on ARMv7-M are not available on ARMv6-M, see figure 8. For example, in the Cortex-M0, M0+ and M1 the number of interrupts is limited to 32, there is no Debug Monitor exception and fault exception is limited to HardFault (see section 3.4 for fault handling details). In contrast, the Cortex-M3, Cortex-M4 and Cortex-M7 processors support up to 240 peripheral interrupts.

Another difference is the number of priority levels available. In ARMv6-M architecture, the interrupt/exception priority level contains 2 fixed levels (for NMI and HardFault) and four programmable levels (2-bits per priority level register). This is sufficient for most microcontroller applications.

In ARMv7-M, the number of programmable priority level range from 8 levels (3-bits) to 256 (8-bits). In practice, most devices only implement just eight (3-bits) to 16 levels (4-bits) due to silicon area. ARMv7-M also has a feature called priority grouping, which enable you to sub-divide priority level registers into group priority and sub-priority, so that you can define preemption behavior in details.

Exception Type	ARMv6-M	ARMv7-M	Vector Table	Vector address (initial)
255			Interrupt#239 vector	0x000003FC
47	Device Specific Interrupts	Device Specific Interrupts	Interrupt#31 vector	0x000000BC
17			Interrupt#1 vector	0x00000044
16			Interrupt#0 vector	0x00000040
15			SysTick	SysTick
14	PendSV	PendSV	PendSV vector	0x00000038
13	Not used	Not used	Not used	0x00000034
12	Not used	Debug Monitor	Debug Monitor vector	0x00000030
11	SVC	SVC	SVC vector	0x0000002C
10	Not used	Not used	Not used	0x00000028
9			Not used	0x00000024
8			Not used	0x00000020
7			SecureFault (ARMv8-M Mainline)	SecureFault (ARMv8-M Mainline)
6		Usage Fault	Usage Fault vector	0x00000018
5		Bus Fault	Bus Fault vector	0x00000014
4		MemManage (fault)	MemManage vector	0x00000010
3	HardFault	HardFault	HardFault vector	0x0000000C
2	NMI	NMI	NMI vector	0x00000008
1			Reset vector	0x00000004
0			MSP initial value	0x00000000

Figure 8: Exception type in Cortex-M processors

All Cortex-M processor rely on vector table during exception handling. The vector table stores the starting address of exception handler and by default it is located in the starting of the memory map (address 0x0, as shown in figure 8). The vector table starting address can be changed using a feature called Vector Table Offset Register (VTOR). This is a useful feature for:

- relocating vector table to SRAM to allow dynamically changing exception handler entrance points
- relocating vector table to SRAM for faster vector fetch (if flash memory is slow)
- relocating vector table to different one in ROM (or flash), so that different stage of program execution can have different exception handlers.

The VTOR is available on Cortex-M0+/M3/M4/M7 and ARMv8-M processors (optional on the Cortex-M0+ and ARMv8-M Baseline).

The NVIC programmer’s models also have some additional differences amongst different Cortex-M processors. The differences are summarized in Table 5:

	Cortex-M0	Cortex-M0+	Cortex-M1	Cortex-M3/M4/M7
Number of Interrupts	Up to 32	Up to 32	Up to 32	Up to 240
NMI	Y	Y	Y	Y
SysTick	Y (optional)	Y (optional)	Y (optional)	Y
Fault handlers	1 (HardFault)	1 (HardFault)	1 (HardFault)	4
VTOR	-	Y (optional)	-	Y
Dbg Monitor	-	-	-	Y
Programmable priority levels	4	4	4	8 to 256
Software trigger interrupt register	-	-	-	Y
Interrupt Active status	-	-	-	Y
Register accesses	32-bit	32-bit	32-bit	8/16/32-bit
Dynamic priority change	-	-	-	Y

Table 5: NVIC programmer’s model and feature differences

In most cases, access to interrupt control feature in the NVIC is handled by APIs provided in CMSIS-Core, which is included inside device driver library from the microcontroller supplier. However, in the Cortex-M3/M4/M7 you can change the priority of an interrupt even it is enabled. This dynamic

priority level change is not supported in ARMv6-M, so you need to disable an interrupt temporarily when changing its priority level.

3.3 OS support features

The architectures for the Cortex-M processors are designed with OS in mind. The features to support OS included:

- Shadowed stack pointer
- SVC and PendSV exceptions
- SysTick timer – a 24-bit down counter for generating periodic OS exception for time keeping and task management
- Unprivileged execution level and Memory Protection Unit (MPU) in Cortex-M0+/M3/M4 and M7

The SVC exception is triggered by the SVC instruction, which enable an application tasks running in unprivileged state to trigger privileged OS services. The PendSV exception is useful for OS to schedule non-critical operations like context switching.

In order to get the Cortex-M1 to fit in some very small FPGA devices, we made all these OS support features optional in Cortex-M1. In Cortex-M0 and Cortex-M0+, the SysTick timer is optional.

In general, OS support is available for all Cortex-M processors. In Cortex-M0+, Cortex-M3, Cortex-M4, Cortex-M7 and ARMv8-M processors, application tasks can run at unprivileged execution level and this can work together with an optional Memory Protection Unit (MPU) to prevent memory access violations. This can enhance the system robustness.

3.4 Fault handling

One of the differences between ARM processors and some other microcontroller architectures is the fault-handling capability. When a fault is detected, a fault exception is triggered to that software can carry out appropriate actions. Faults can be:

- Undefined instructions (e.g. flash memory corrupted)
- Access to illegal address space (e.g. stack pointer corruption) or MPU access violations
- Illegal operations (e.g. trying to trigger SVC exception when the processor is already at an interrupt priority higher than the SVC)

Fault handling enables an embedded system to react to various issues much quicker. Otherwise, if the system hanged, it could take a long time for a watchdog timer to reset the system.

In the ARMv6-M architecture, all the fault events trigger the HardFault handler, which has a priority level of -1 (higher priority than all programmable exceptions, but just below the Non-Maskable

Interrupt NMI). All faults are considered unrecoverable and normally we just carry out error reporting and potentially generate a self-reset inside the HardFault Handler.

In ARMv7-M, there are three configurable fault handlers in addition to the HardFault:

- Memmanage (Memory Management Fault)
- Bus Fault (Bus returns an error response)
- Usage Fault (undefined instructions or other illegal operations)

These exceptions have programmable priority levels and can be individually enabled / disabled. They can also make use of the FAULTMASK register to escalate their priority to be same level as HardFault, if needed. There are also numerous fault status registers that provide hints about what might have triggered the fault exception, and fault address register, which could be used to pin point the transfer address that triggered the fault to make debugging easier.

The extra fault handlers in ARMv7-M provide more flexible fault-handling capability, and the fault status registers made it easier to locate and debug the fault event. A number of debuggers in commercial development suites have included features to diagnose fault events using the fault status registers. Moreover, potentially during runtime, the fault handlers can carry some of the remedy actions out.

	ARMv6-M (Cortex-M0, M0+, M1) and ARMv8-M Baseline	ARMv7-M (Cortex-M3/M4/M7) and ARMv8-M Mainline
HardFault	Y	Y
MemManage	-	Y
Usage Fault	-	Y
Bus Fault	-	Y
SecureFault	-	ARMv8-M Mainline only
Fault Status Registers	- (one Debug FSR for debug only)	Y
Fault Address Register	-	Y

Table 6: Summary of Fault handling feature comparison

4 System features

4.1 Low power support

Low power is one key advantage of the Cortex-M processors. The low power support is built into the architecture:

- WFI and WFE instructions
- Architectural sleep mode definitions

In addition, there are a number of other low power features that are available on Cortex-M processors:

- Sleep and deep sleep modes: Supported by the architecture, and can be further expanded using device specific power control registers.
- Sleep-on-exit: Enable lower power in interrupt-driven applications. When enabled, the processor enters sleep automatically when finishing an exception handler and if no other exception is pending. This reduces power by avoiding extra active cycles of executing code in Thread mode, and reduces unnecessary stack operations.
- Wake-up Interrupt Controller (WIC): An optional feature that allows interrupt-detection to be carried out by a small block outside the processor during certain low power state, E.g. when the processor is powered down in State Retention Power Gating (SRPG) designs.
- Clock gating and architectural clock gating: Allows clocks to registers or sub-modules of the processor to be turned off to reduce power.

All these features are supported in Cortex-M0, Cortex-M0+, Cortex-M3, Cortex-M4 and Cortex-M7 processors. In addition, various low power design techniques are used to reduce power consumptions.

Due to the low gate counts, the Cortex-M0 and Cortex-M0+ processors have lower power than the Cortex-M3, Cortex-M4 and Cortex-M7 processors. In addition, the Cortex-M0+ has additional optimizations to reduce the amount of program accesses (e.g. branch shadows) to keep the system-level power consumption lower.

4.2 Bit-band feature

The Cortex-M3 and Cortex-M4 processors have an optional feature called bit band that allows two 1MB address ranges (one in SRAM, from 0x20000000, the other in Peripheral, from 0x40000000) to be bit addressable via bit band alias addresses. The Cortex-M0, M0+ and Cortex-M1 processors do not have this feature, but this can be added on the system level using a bus level component from the ARM Cortex-M System Design Kit (CMSDK). The Cortex-M7 processor does not support bit band because its cache support feature cannot be used with bit band (the cache controller does not know the aliasing of memory spaces).

4.3 Memory Protection Unit (MPU)

The Cortex-M0+, Cortex-M3, Cortex-M4 and Cortex-M7 processors have an optional MPU, which can be used to define Memory access permissions and Memory attributes or memory regions. In systems with RealTime Operating System (RTOS), the OS can define the memory access permission and memory configuration for each task to ensure that each task cannot corrupt memory ranges used by other tasks or the OS kernel. The MPU in Cortex-M0+, Cortex-M3 and Cortex-M4 all have eight programmable regions and have very similar programmer's model. The main different is that the MPU in Cortex-M3/M4 allows two levels of memory attributes (e.g. system level cache types) while the one in the Cortex-M0+ processor only support one level. The optional MPU in Cortex-M7

can be configured to support 8 or 16 regions, and allows two levels of memory attributes. The Cortex-M0 and Cortex-M1 do not support MPU.

4.4 Single cycle I/O interface

The single cycle I/O interface is a unique feature on the Cortex-M0+ processor, which enables this processor to be very fast in I/O control tasks. In other Cortex-M processors, the bus interfaces are based on the AHB Lite protocol, which is a pipelined bus protocol and allows high clock frequency operation. However, it means it needs two clock cycles per transfer. The single cycle I/O interface add an extra simple bus interface that is not pipelined, for connection to a small set of device specific peripherals like GPIO (General Purpose Inputs/Outputs). When combining this feature with the low branch penalty nature of the Cortex-M0+ processor (since its pipeline is only two stages), many I/O control operations can be carried out faster than most other microcontroller architectures.

5 Performance considerations

5.1 General data processing performance

In general microcontroller market, benchmark figures are often used to measure the performance of microcontrollers. Table 7 shows the performance of the Cortex-M processors running commonly used benchmarks:

	Dhrystone DMIPS/MHz (v2.1) – official	Dhrystone DMIPS/MHz (v2.1) – full optimization	Coremark/MHz (v1.0)
Cortex-M0	0.84	1.21	2.33
Cortex-M0+	0.94	1.31	2.42
Cortex-M3	1.25	1.89	3.32
Cortex-M4	1.25	1.95	3.40
Cortex-M7	2.14	2.55	5.01

Table 7: Performance of the Cortex-M processor using commonly used benchmarks

(Source : CoreMark.org website and ARM website)

One thing we need to be careful about Dhrystone is that official the code should be compiled without inline and without multi-file compilation, and for consistency, the original K&R version of source code r2.1 should be used. However, many microcontroller vendors quotes Dhrystone figures with full optimizations enabled.

However, results from benchmark suite might not provide an accurate projection of what you can get in your applications. For example, effect of the single cycle I/O interface and acceleration using

SIMD in DSP applications, or using of FPU in the Cortex-M4/M7 does not necessary shows in these figures.

In general, Cortex-M3 and Cortex-M4 provides higher data processing performance because of:

- richer instruction set features
- Harvard bus architecture
- write buffer (single cycle write operation)
- speculative fetch of branch targets

The Cortex-M7 processor provides an even higher performance because of its six-stage dual issue pipeline (allows execution of up to two instructions in the same clock cycle) and branch prediction support. Also, it enables higher system-level performance by offering instruction and data caches, as well as tightly couple memories to avoid performance penalty even when having slow main memories (e.g. embedded flash).

However, some I/O intensive tasks can be faster on the Cortex-M0+ processor because of:

- Shorter pipeline (branches only take two cycles)
- Single cycle I/O interface

There are also device specific factors. For example, system level designs, memory speed can also affect the performance of the system.

Very often, your own application code can be the best benchmark you need. The processor with double the CoreMark score than another does not necessary it can perform twice as fast your own applications. In applications with I/O intensive tasks, the system-level architecture can have a great impact of the performance, which is device specific.

5.2 Interrupt Latency

Another aspect of performance is interrupt latency. This is typically measured from the number of clock cycles from the assertion of an interrupt request to the execution of first instruction in the interrupt service routine. Table 8 listed the interrupt latency for scenarios with zero wait state memory system:

	Interrupt latency (number of clock cycles)
Cortex-M0	16
Cortex-M0+	15
Cortex-M3	12
Cortex-M4	12
Cortex-M7	Typically 12, worst case 14

Table 8: Interrupt latency with zero wait state in memory system

In practice, the actual interrupt latencies are affected by the wait states of the memory system. For example, many microcontrollers running at over 100MHz have a slower flash memory (e.g. 30 to 50MHz). Although flash access acceleration hardware is used to improve performance, the interrupt latency can still be affected by the flash memory wait states. So it is well possible that a Cortex-M0/M0+ system running with zero wait state memory system got shorter interrupt latency than a Cortex-M3/M4/M7 system.

Do not forget to take the execution time for your interrupt handler into account when investigating the performance. Some 8-bit or 16-bit processor architectures might have very short interrupt latency, but can take several times the number of clock cycles to finish the interrupt request processing. Effectively have a much slow interrupt response time and lower interrupt processing bandwidth.

6 Debug and Trace features

6.1 Debug and trace features overview

There are a number of differences between different Cortex-M processor products. These are summarized in Table 9.

	Cortex-M0/M1	Cortex-M0+	Cortex-M3/M4	Cortex-M7
Protocol of debug connection	JTAG / Serial Wire	JTAG / Serial Wire	JTAG / Serial Wire / both	JTAG / Serial Wire
Protocol of trace connection	-	-	Trace port (4 pin data + clock) / Serial Wire Viewer (SWV, single pin)	Trace port (4 pin data + clock) / Serial Wire Viewer (SWV, single pin)
Hardware Breakpoint comparators	Up to 4	Up to 4	Up to 8 (6 for instruction addresses, 2 literal addresses)	Up to 8 (all for instruction addresses)
Software breakpoint (BKPT instruction)	Y	Y	Y	Y
Data Watchpoint comparators	Up to 2	Up to 2	Up to 4	Up to 4
Instruction Trace	-	Limited history using MTB	ETM trace with unlimited history	ETM trace with unlimited history

Data Trace	-	-	Selective data trace using DWT	Selective data trace using DWT, Optional full data trace via ETM
Event & profiling Trace	-	-	Using DWT	Using DWT
Instrumentation (Software) trace	-	-	Instrumentation Trace Macrocell (ITM)	Instrumentation Trace Macrocell (ITM)
Trace timestamp	-	-	Y	Y
On the fly memory accesses	Y	Y	Y	Y
Debug using software agent (Debug monitor)	-	-	Y	Y
Multi-core debug synchronization	Y	Y	Y	Y
PC sampling	Y (via debug connection)	Y (via debug connection)	Y (via debug connection or by trace)	Y (via debug connection or by trace)

Table 9: Debug and Trace feature comparison

The debug architecture of the Cortex-M processors is based on ARM CoreSight debug architecture. This is a very scalable architecture and support multi-processor systems.

The information shown in table 9 is for typical designs. Under CoreSight architecture, the debug interface and trace interface modules are decoupled from the processor. Therefore potentially the debug and trace connection of the device you are using can be different from what is shown in table 9. It is also possible to add some of the debug features by adding additional CoreSight debug components.

6.2 Debug connections

The debug connection allows the debugger to

- access registers that control the debug and trace features.
- access the memory map. For the Cortex-M processors, that can be done even when the processors are running. This is called on-the-fly memory accesses.
- access core registers. This can only be done when the processor is halted.

- access trace history generated from the Micro Trace Buffer (MTB) in the Cortex-M0+ processor.

In addition, the debug connection can also be used for:

- flash programming

In the Cortex-M processors, there is a choice of the traditional JTAG connection, which need 4 to 5 pins (TDI, TDO, TCK, TMS and optional nTRST), or a new Serial Debug protocol which need only two pins, which is ideal for devices with limited pin counts.

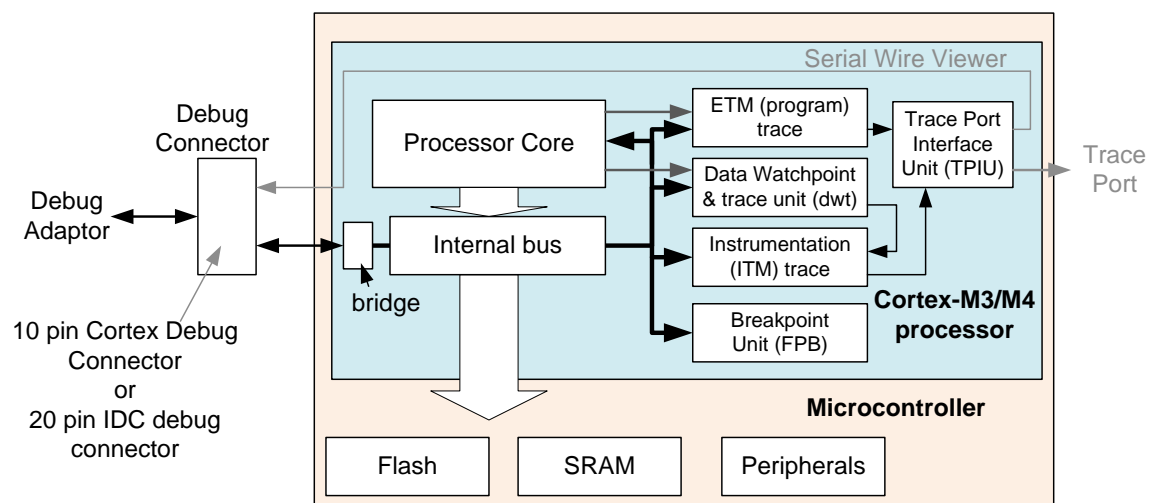


Figure 9: Serial Wire or JTAG Debug connection allows access to processor's debug features and memory space including peripherals

The Serial Wire debug protocol can handle all features available in JTAG and can provide parity checking. The Serial Debug protocol is widely adopted by ARM tools vendors, many debug adaptors support both protocols and the Serial Wire signals share the TCK and TMS pins on debug connectors.

6.3 Trace connection

The trace connection allows debugger to collect information about program execution in real time (with a small delay) during program execution. The information collected could be program flow information (instruction trace) generated from Embedded Trace Macrocell (ETM), available on the Cortex-M3/M4/M7, data/event/profiling trace generated from Data WatchPoint and Trace (DWT) unit, or information generated from the Instrumentation Trace Macrocell (ITM) by software control.

There are two type of trace connection available:

- Trace port – multiple data pin plus clock signal. This provides much higher trace bandwidth than SWV and can support all trace type in SWV plus instruction trace. On Cortex-M3/M4 or Cortex-M7 devices, the trace port usually has four data pins and one clock pin. (Figure 10)

- Serial Wire Viewer (SWV) – single pin trace connection that can support selective data trace, event trace, profiling and instrumentation trace. (Figure 11)

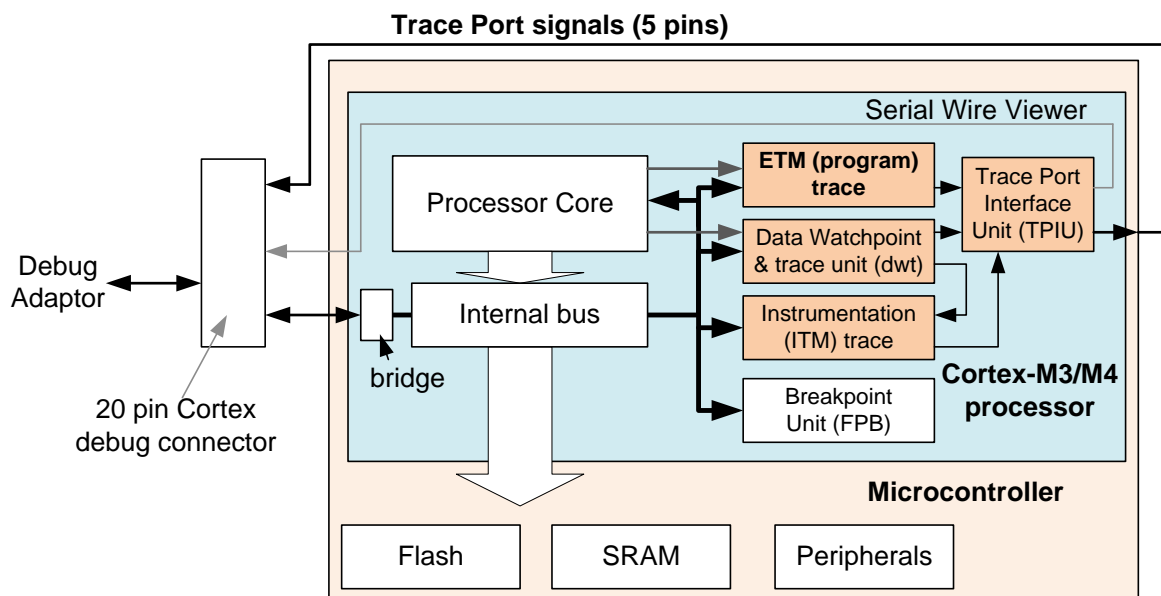


Figure 10: Trace port provides required trace bandwidth for instruction trace plus other trace

The trace connection allows plenty of useful information to be captured while the processor is running. For example, the Embedded Trace Macrocell (ETM) enables instruction execution history (instruction trace) to be captured, and the Instrumentation Trace Macrocell (ITM) enables software to generate message (e.g. via printf) and capture via the trace connection. In addition, the Data Watchpoint and Trace (DWT) module in Cortex-M3/M4/M7 enables:

- Selective data trace: Information about a memory location (e.g. combination of address, data value & timestamp) can be captured each time the location is accessed by the processor
- Profiling trace: Number of clock cycles the CPU used in different operations (e.g. memory accesses, sleep)
- Event trace: Provides history and duration of interrupts/exceptions being served by the processor

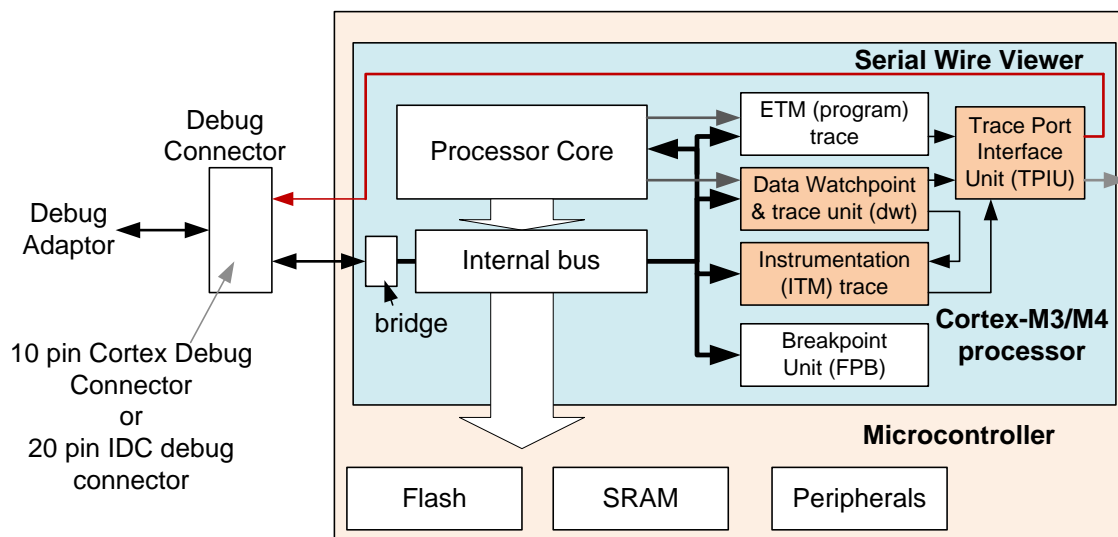


Figure 11: Serial wire viewer provides low cost, low pin count trace

These trace features are widely supported by various tool vendors and the information can be visualized in a number of ways. For example, the data trace from DWT can be visualized as waveform in Keil μ Vision debugger (part of the Keil Microcontroller Development Kit), as shown in Figure 12.

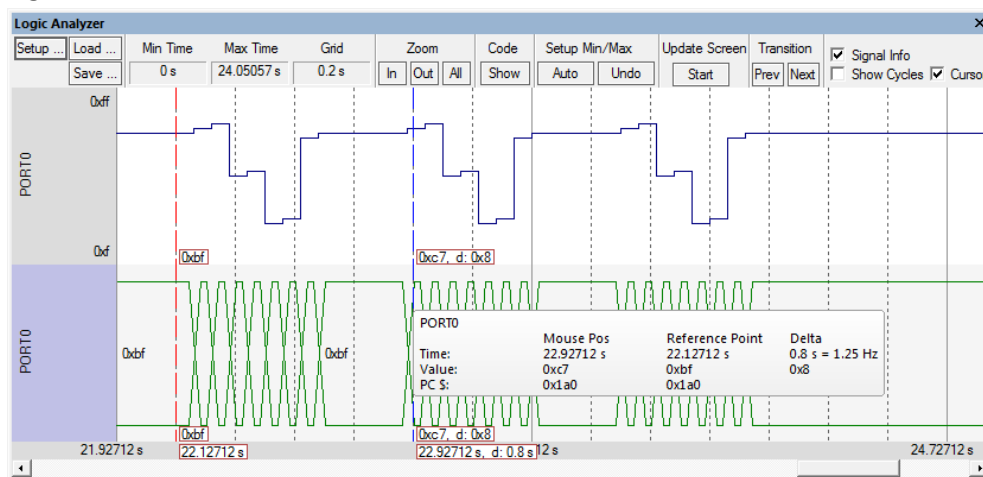


Figure 12: Logic Analyzer in Keil μ Vision debugger

While the Cortex-M0 and Cortex-M0+ processors do not have trace interface, the Cortex-M0+ process has a feature called Micro Trace Buffer (MTB, Figure 13). The MTB enable users to allocate a small part of the system SRAM as a buffer for storing instruction, usually in a circular buffer arrangement so that the recent instruction execution history can be captured and shown on the debugger.

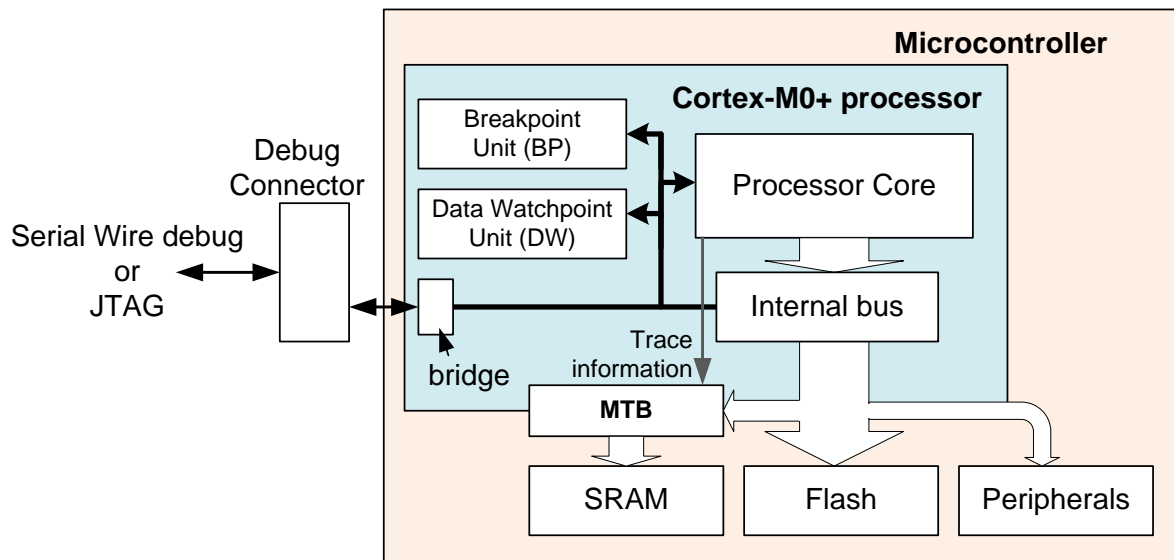


Figure 13: MTB feature in Cortex-M0+ processor provides low cost instruction trace solution

7 Product development with the ARM Cortex-M processor

7.1 Why Cortex-M processors are easy to use

Although the Cortex-M processors are packed with features, they are also very easy to use. For instance, almost everything can be programmed in high-level language like C. Although there is a big variety of different Cortex-M processor-based products (e.g. with different memory size, peripherals, performance, packages, etc), the consistency of the architecture make it easy to start using a new Cortex-M processor once you have experience with one of them.

To make software development easier, and to enable better software reusability and portability, ARM developed the CMSIS-CORE, where CMSIS stands for Cortex Microcontroller Software Interface Standard. The CMSIS-CORE provides a standardized Hardware Abstraction Layer (HAL) for various features in the processors such as interrupt management control, using a set of APIs. The CMSIS-CORE is integrated in the device driver libraries from various microcontroller vendors, and is supported by various compilation suites.

Beside from CMSIS-CORE, CMSIS also have a DSP software library (CMSIS-DSP). It provides various DSP functions and is optimized for Cortex-M4, and also supports other Cortex-M processors. Both CMSIS-Core and CMSIS-DSP are free and can be downloaded from the ARM website (www.arm.com/cmsis), and are supported by multiple tool vendors.

7.2 Selecting a processor

For most microcontroller users, the criteria of selecting a microcontroller device are mostly heavily dependent on the cost and peripherals available. However, many of you could be silicon designers

Copyright © 2016 ARM Limited. All rights reserved.

The ARM logo is a registered trademark of ARM Ltd.

All other trademarks are the property of their respective owners and are acknowledged

choosing the processor core for your next silicon products. In that case, the processor itself would be the focus.

It is obvious that performance, silicon size, power, and costs would be vital factors in such scenarios. At the same time, there can be various other factors to be considered. For example, if you are developing an internet-connected product, you might want to opt for a processor with the Memory Protection Unit (MPU) so that you can run some tasks in unprivileged state and protect the memory space using the MPU. On the other hand, if you need to certify your product in some way, the instruction trace generated from the ETM on Cortex-M3, Cortex-M4 and Cortex-M7 can be very useful for code coverage prove.

On the other spectrum of silicon design, if you are designing a small sensor running on harvested energy, then the Cortex-M0+ processor could be the best choice as it is very small and have state-of-the-art power optimization.

7.3 Ecosystem

One of the key advantages of using ARM Cortex-M processors is the wide support of devices, tools, and middleware. Currently there are:

- More than 15 microcontroller vendors shipping microcontroller products based on ARM Cortex-M processors
- More than 10 development suites supporting the Cortex-M processors
- More than 40 OS vendors support OS for Cortex-M

This gives you plenty of choices, and allows you to get the best combination of devices, tools and middleware for your target applications.

8 Conclusions

There is always a tradeoff between performance, features, against silicon size and power. As a result, ARM developed various Cortex-M processors for different level of instruction set features, performance, system and debug features. This document covered various similarities and differences in the Cortex-M processor family.

Although there are differences, the consistency in the architecture and the availability of standardize APIs in CMSIS-CORE made software more portable and reusable. At the same time, the Cortex-M processors are very easy to use. Therefore, the Cortex-M is quickly becoming the most popular 32-bit processor architecture in microcontroller segment.