在Cortex–M3处理器上运行ARM7TDMI处理器软件

作 者
Mark Collier,
ARM

纲 要

在考虑用Cortex-M3处理器运行ARM7TDMI处理器软件时，软件开发人员想到的第一个问题是："这样做需要多少时间和工作量？"回答是："基本为零。"在绝大多数情况下，只需进行简单的重定向编译。软件开始运行后，开发人员可能会考虑对其进行提升，以便充分利用ARM7TDMI处理器上没有的Cortex-M3处理器新功能。本文讲述了把ARM7TDMI处理器代码移植到Cortex-M3处理器时是如何快捷方便，以及如何使用随Cortex-M3处理器推出的新功能和v7-M架构，对软件的大小、能力和可维护性进行优化。

关于ARM Cortex–M3

Cortex–M3处理器是基于ARMv7–M架构的第一款ARM处理器内核，专门设计用于在对功耗和成本要求严格的深度嵌入应用（例如微控制器、汽车车身系统、工业控制及无线网络）中实现高系统性能。

与ARM7TDMI处理器相比，Cortex–M3处理器有许多引人入胜的特点：

- ARMv7–M架构，与ARMv4T架构相比有36条新指令。
- 性能更高，业界首屈一指的代码密度，采用Thumb®–2指令集架构（ISA）。
- 中断延迟时间更短，标准化内存映射，内存保护单元，以及带内置SysTick的集成式NVIC。
- 调试与跟踪能力加强。
- 有两种休眠模式，更加节能。
- 采用CoreSight及其他系统功能，为多核系统提供更强支持。

入门

针对现代32位微控制器（如ARM7TDMI处理器）创建的大多数应用程序均以C或C++作为编写语言，以便使代码易于维护，并且便于在来自不同产品系列和供应商的各种装置之间移植。经过简单的重定向编译后，大多数此类应用都可以在Cortex–M3处理器上运行。

此外，在多数情况下，可以采用实时操作系统（RTOS）或由微控制器供应商编写的低级装置驱动程序，从底层硬件中提取出经过妥善编写的应用代码。这样，如果用户拥有可在RTOS（如uC–OS/II、Nucleus 或 ThreadX）上运行的应用代码，在Cortex–M3处理器上运行时只需安装最新版本的RTOS并定向到新装置。如果用户没有运行RTOS，则运行应用程序时不应改动源代码，条件是微控制器供应商已经对应用所需的低级驱动程序进行更新。

不过，从设备供应商的角度出发，为任何新装置提供支持时显然都需要进行一些工作，无论它是现有设备或系列的扩展型号、还是采用了扩展架构（如Cortex–M3处理器）。本白皮书阐述了对用于Cortex–M3设备的、嵌入ARM7TDMI处理器的低级驱动程序进行优化时所需的常见源代码和较小的工具链改动。

当今的开发人员主要使用C语言，但在某些情况下仍会用到传统的汇编代码，例如在时间很紧张的部分。如果汇编程序以16位Thumb代码编写，则它通常会按照"现有状态"在Cortex–M3处理器上运行，因为原始Thumb–2 ISA是Thumb的超集。不过，即便已经采用32位ARM汇编程序，汇编程序仍能把这些指令作为Thumb–2指令对待，而不进行手动干预或只进行很少干预。本白皮书将对此进行深入阐述。

本白皮书假设读者熟悉嵌入式软件开发工作以及如何在目标装置之间移植软件。

ARM7TDMI处理器与Cortex-M3处理器的对比

从软件移植角度考虑，ARM7TDMI处理器与Cortex-M3处理器的主要差别是：

特点	ARM7TDMI处理器	Cortex-M3处理器
执行状态	ARM或Thumb	仅限Thumb-2
内存映射	非结构化	架构指定
中断	IRQ / FIQ 外部中断控制器	NMI + SysTick + 最多240个中断， 集成式NVIC中断控制器
系统状态	PSR（分组寄存器）	xPSR（堆栈寄存器）

▲ 图1 ARM7TDMI处理器与Cortex-M3处理器的主要软件差别

ARM7TDMI处理器可处于ARM（32位指令）或Thumb（16位指令）状态。状态受到软件控制，不过总是在ARM状态下进入中断例程。相反，Cortex-M3处理器仅可在Thumb-2状态（16位及32位指令）下执行。关于Thumb-2技术的详情，请参考第1.2节。

ARM7TDMI 处理器需要从地址0开始放置异常矢量，不过对于在哪里放置内存映射装置则不加限制。相反，Cortex-M3处理器规定了某些顺序，来改善在不同装置之间的可移植性，同时不牺牲差异性。不过，内存映射区经过仔细挑选，以便与现有的大多数基于ARM7TDMI处理器的微控制器相匹配。

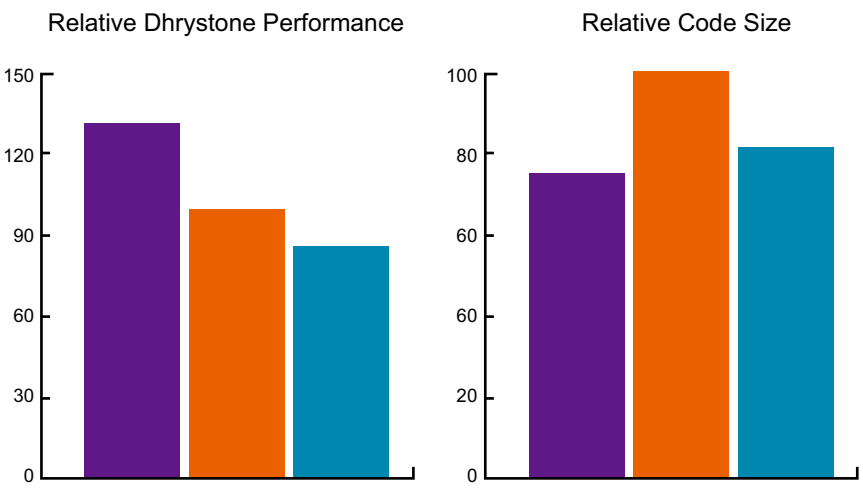
ARM7TDMI处理器有两条外中断线：IRQ（中断请求）与FIQ（快速中断请求）。处理两个以上中断时需要外中断控制器，如VIC（矢量中断控制器）或GIC（通用中断控制器），把信号复用到IRQ（或FIQ）线上。相反，Cortex-M3处理器假设需要许多外部中断，因此包括一个集成式NVIC（嵌套矢量中断控制器），后者具有NMI（不可屏蔽中断）。

ARM7TDMI处理器具有复杂的程序员模型，有6种模式和分布于各模式的20个分组寄存器。相反，Cortex-M3处理器要简单得多，只有2种模式和1个分组寄存器。这意味着发生异常时寄存器会自动压栈。

为什么Cortex-M3处理器只采用Thumb-2?

在某些情况下，支持Thumb模式的处理器仍然需要ARM指令。Thumb-2的设计目的是取消这一限制，从而可以仅仅用Thumb-2指令来建造整个嵌入式软件系统。所以，Cortex-M3处理器只采用Thumb-2，因为不再需要ARM解码器。

更为重要的是，借助Thumb-2设计，可以用Thumb代码密度实现ARM性能水平。为了实现这一看似不可能的目标，它在Thumb ISA中加入新的16位和32位指令，从而可以更为简洁地对操作进行编码，同时无需进行状态切换。



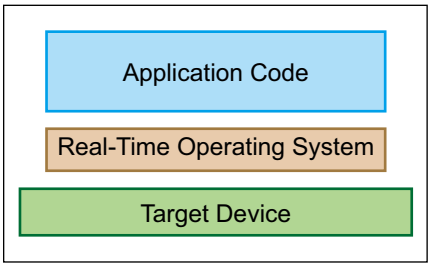
▲ 图2 Cortex-M3处理器和ARM7TDMI处理器（ARM及Thumb状态）的相对Dhrystone性能及代码大小性能及代码大小

Thumb-2与Thumb完全后向兼容，因此Thumb二进制代码可以不加修改而直接运行。Thumb-2是所有Cortex系列内核的标配。

嵌入式软件系统的组成部件

以下章节假设已经进行了相应工作，以保证软件在访问硬件时具有正确的地址和寄存器定义。在任意两个目标设备（与内核无关）之间移动时需要这一条件，并且这取决于两个设备在时钟、中断、地址映射和所用的设备编程接口等方面的相似程度。

实时操作系统



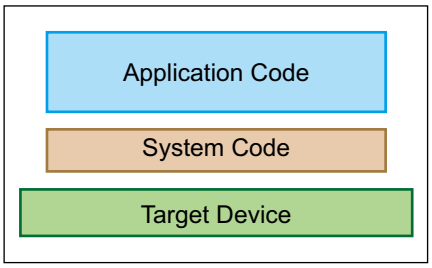
▲ 图3 RTOS嵌入式软件系统的组成部分

如果应用程序使用实时操作系统（RTOS），则它与目标设备的诸多细节隔离。假设Cortex-M3处理器可使用RTOS，则需要对应用进行简单的重定向编译。开发人员可能希望重建所有Thumb对象，以便获得Thumb-2的性

能优点，不过，这对简单移植来说并非必要。

由于Thumb-2对ARM功能性的覆盖度非常高，汇编程序应该可以毫无困难地把ARM应用汇编程序汇编到Thumb-2指令中，它只需知道目标是Cortex-M3处理器。由于Cortex-M3处理器的效率更高，为了提高性能，开发人员可能希望用更容易维护的C语言重新编写所含的内嵌或嵌入汇编程序。

无操作系统



▲ 图4　嵌入式软件系统的组成部分

如果应用不使用RTOS，软件系统中的代码通常会被划分为应用代码和系统代码。

应用代码

应用代码占据软件的大部分，并且需要简单的重定向编译。开发人员可能希望重建所有Thumb对象，以便获得Thumb-2的性能优点，不过，这对简单移植来说并非必要。

由于Thumb-2对ARM功能性的覆盖度非常高，汇编程序应该可以毫无困难地把ARM应用汇编程序汇编到Thumb-2指令中，它只需知道目标是Cortex-M3处理器。由于Cortex-M3处理器的效率更高，为了提高性能，开发人员可能希望用更容易维护的C语言重新编写所含的内嵌或嵌入汇编程序。

系统代码

系统代码在软件中占据较小部分，它可访问底层硬件，通常由设备供应商作为示例库与/或示例集提供。

在极少见的情况下，没有给装置配备库，则系统代码通常主要以C语言编写，并且只需根据应用代码进行重新编译。剩下的一小部分是ARM汇编程序代码，用于处理：

- 启动与矢量表。
- 异常。
- 访问受指令或内核寄存器支配的系统功能。

通常，启动代码和异常矢量表都包含在一个汇编程序源文件中。对于Cortex-M3处理器，该文件要比在ARM7TDMI处理器中更为简单，因为矢量表由地址（而非ARM指令）组成，而且不再要求代码在所有6种模式中初始化分组寄存器。的确，借助RVDS，可以完全除去汇编程序源文件，并用代码连接器设定重设矢量，同时设置C栈和堆（之后可以由C应用程序执行进一步的初始化）。

Cortex-M3处理器的异常处置工作更加简单，因为寄存器会被自动压栈/出栈，因此异常处置程序（包括重新进入的）可以完全用C编写，而不需与外部中断处理程序相互作用，从而节省代码。

如果用很短的函数实现对系统功能的访问（例如启用或停用中断），可能需要进行移植。通常这类函数只有很少的几种。

注：对于采用建议停用的‘mov pc’指令从函数返回的ARM汇编程序，需要用‘bx’指令来代替这些指令。

充分利用Cortex-M3处理器

之前的章节探讨了如何把为ARM7TDMI处理器目标设备编写的现有应用程序移植到Cortex-M3处理器目标设备中。移植工作的范围取决于系统设计的相似程度，以及那些在Cortex-M3处理器中没有、或其处理方式不同的ARM7TDMI功能的使用水平。本节旨在就如何利用Cortex-M3处理器具有、而ARM7TDMI处理器没有的特点，增强新移植的应用程序提出建议。RVCT 3.0将在36条新指令中自动选出更适合用于表达所需行为的指令。

位操作

Cortex-M3处理器有6条新指令可在寄存器内的位域中运行，以协助外设编程和包格式化，对前导零进行计数，以及在寄存器中进行位反转。

字节与半字操作

Cortex-M3处理器有3条新指令，可使字中的字节和半字反转以及添加/取消符号，并把寄存器中的最低有效字节和最低有效半字扩展为32位值。

算术

Cortex-M3处理器有11条新的有符号与无符号硬件乘除法指令，用于更好地支持饱和及64位整数算术。

程序流控制

Cortex-M3处理器有3条用于加强程序流控制的新指令：

“If Then”（如果…就），用于条件执行；“Compare and branch if zero/no-zero”（比较，如为零/非零，则进行分支跳转），用于空指针检查；以及“Table Branch”（表式分支跳转），用于分支表。

多处理器系统

Cortex-M3处理器有8条新指令，用于在多处理器系统中改善运行情况。有3条专用的访问指令，用于管理在处理

器之间分享的信号量；3条新的内存屏障指令，在与另一内核分享SRAM或外部RAM时使用；有2条新指令用于在处理器之间发送事件。

电源管理

Cortex-M3处理器有2条新指令，向处理器指示何时可以进入并保持节电状态，直至发生中断或事件。这些指令通常用在空闲线程中。也可对Cortex-M3处理器进行编程，使其在完成对中断的处理后进入休眠。最后，目标设备支持的话，Cortex-M3处理器还可以一种深度休眠模式。

系统时钟

Cortex-M3处理器有一个集成在NVIC控制器中的新型24位系统计时器。这意味着RTOS不必依赖于外部计时器，从而显著增强了它们的可移植性。

内存保护单元

Cortex-M3处理器有一个选配的集成式MPU，可支持8个大小不同的内存区。可以把属性应用于这些内存区的物理地址，以控制允许用户使用的访问等级和特权模式应用，并用于描述内存区（内存类型、缓存策略等）。等于或大于256K的区可被分割为8个子区，从而在区中留出没有应用属性的空隙。内存区可以重叠；指定区号最高的属性优先。

调试

Cortex-M3处理器具有完善的内置调试功能。断点、观察点、指定异常和外部（芯片外/其他内核）调试请求可以使内核暂停，或产生“Hard Fault”（硬故障）或“Debug Monitor”（调试监视）异常。内核暂停可使其进行单步执行。发生“Debug Monitor”异常时，可在每个点调用软件监视程序/状态输出。也可通过可从外部访问的寄存器读写内核寄存器。

“Flash Patch”单元提供了2个文字比较器，可用于以SRAM或外部存储器中的值来代替代码区中的常量；以及6个指令比较器，可用作断点指令，或以SRAM或外部存储器中的值代替代码区中的指令。

“Data Watchpoint and Trace”（数据观察点与跟踪）单元提供了4个比较器，可用作硬件观察点、嵌入式跟踪模块（ETM）触发器、PC或数据地址采样事件触发器。观察点是在PC地址或数据存取类型与数据地址匹配时激活的事件，它与断点的相似之处在于，二者都可使内核暂停或产生异常，但在调试故障状态寄存器中设置的标记有所不同。它还提供了6个计数器，用于获取性能概貌，可用于在溢出时发出ETM事件。“Instrumentation Trace Macrocell”（仪器跟踪宏单元）为应用程序提供了一种机制，以便使用跟踪端口来输出调试消息，同时尽可能减少开销。它把这些包与来自DWT的包合并，还可把Δ时间戳记包注入流中。

总的说来，应用程序开发人员不必了解上述所有细节，因为会通过所选的调试工具来提供各项功能。

位段

Cortex-M3处理器在地址映射中定义了两个位段区：

- SRAM位段区，用于变量。
- 外设位段区，用于控制和状态寄存器。

通过对位段别名地址进行字读写，可以在位段区中的相应字内访问各个单独的位。

个案研究

为了实际考察上述各项，为LuminaryMicro LM3S811 Cortex-M3评估工具包编写了一个独立的C软件应用程序，并移植到Keil MCB2130 ARM7TDMI 开发板中。通过该应用程序，可以用按钮或串行端口上的文本指令打开/关闭一个发光二极管或令其闪烁。大量使用中断，并且需要进行乘除，以便打印出开关次数，因为它以十进制方式重设。在两种情况下都使用了示范性汇编程序启动文件。

代码与数据大小

作为ARM、ARM/Thumb以及Thumb-2编译的应用程序（-O2）代码和数据的大小（单位为字节）如图5所示：

所编译的代码和数据段	ARM7TDMI ARM	ARM7TDMI ARM / Thumb	Cortex-M3 Thumb-2
应用代码	1596	1164	1000
软件除法代码	364	364	0（硬件除法）
应用数据	156	156	156
启动代码	272	272	200
合计	**2388**	**1956**	**1356**

▲ 图5　功能相当的嵌入式软件系统的代码及数据大小

源代码的差异

把ARM7TDMI处理器和Cortex-M3处理器的C源文件放在一起对比，可以发现下列区别（空白部分和注释除外）：

内核系统源之所以不同，是因为Cortex-M3处理器应用中采用了嵌入式汇编程序，以便启用中断，并利用新的“Wait For Interrupt”（等待中断）指令（参见第3.6节）。在ARM7TDMI处理

源	ARM7TDMI 处理器与Cortex-M3处理器源文件的区别
应用源	相同
内核系统源	13行不同（3%）
装置系统源	122行不同（29%）

图6　两种嵌入式软件系统的行数差异

器应用中还有一个“#pragma thumb”指令，用于编译Thumb。

设备系统源之所以不同，是因为UART外设不同；在ARM7TDMI处理器中由外部VIC处理中断，而在Cortex-M3处理器中则由集成式NVIC处理；在ARM7TDMI处理器中，由外部计时器产生周期计时中断，而在Cortex-M3处理器中则由集成式系统时钟产生。

此处所用的应用相当简单；在真实的嵌入式系统中，系统代码的百分比会小得多。

开发工具

选择适当的开发工具对于成功开发嵌入式软件项目具有举足轻重的意义。本节说明了Cortex-M3处理器可用的各种开发工具。

ARM

ARM提供范围广泛的多种开发工具：

- RealView®开发套件（RVDS）：可为ARM架构开发C、C++及汇编程序软件。
- RealView ICE及RealView Trace：连接RVDS与目标装置，进行调试和跟踪。
- AMBA® Designer、RealView SoC Designer及RealView模型库：用于硬件开发。
- 指令集仿真模型和实时仿真模型：用于软件开发。
- Emulation Baseboard：用于硬件FPGA原型制作。
- RealView微控制器开发套件(Keil)：用于在目标设备和模拟器上开发嵌入式软件，使用RVDS提供的RealView编译工具。
- ULINK：连接RealView MDK与目标装置，进行调试和跟踪。与目标装置，进行调试和跟踪。

如需详情，请访问www.arm.com/products/DevTools。

生态系统

围绕Cortex-M3处理器有一个丰富而不断成长的生态系统，因此，尽管ARM提供了完整的端到端RealView工具解决方案，开发人员仍越来越青睐这种首选工具链。目前支持Cortex-M3处理器的有下列开发工具：

结论

我们看到，在Cortex-M3处理器装置上运行ARM7TDMI处理器应用软件时只需简单地进行重定向编译，而运行ARM7TDMI系统软件时则可能需要对源代码进行少量易于确认的修改。重定向编译时将自动利用Cortex-M3处理器中的新v7-M指令，对于RVDS，则选择针对Cortex-M3处理器予以优化的Thumb-2库。

开发人员可以选择进行用C语言实现汇编程序代码编写的操作，或扩展来利用新型Cortex-M3处理器的功能。除ARM提供的全面支持外，开发人员还拥有范围广泛且种类不断增多的工具和RTOS，可以促进开发工作。

除了在设备之间完成正常移植外，把系统处理器内核从ARM7TDMI处理器变成Cortex-M3处理器是一项较为简单的任务，只需进行很少的工作。一旦完成这一工作，就可充分利用Cortex-M3处理器的优异性能、较低能耗和更高的代码密度，获得可观的竞争优势。

工具	供应商	供应商网站
CrossWorks	Rowley Associates	www.rowley.co.uk
EDGE	Accelerated Technology	www.mentor.com
EWARM	IAR Systems	www.iar.com
GNU	CodeSourcery	www.codesourcery.com
TRACE32	Lauterbach	www.lauterbach.com

▲ 图7　Cortex-M3处理器的开发工具供应商

RTOS	供应商	供应商网站
FreeRTOS	FreeRTOS	www.freertos.org
Nucleus Plus	Accelerated Technology	www.mentor.com
ThreadX	Express Logic	www.rtos.com
RTX	ARM	www.arm.com
Salvo	Pumpkin	www.pumpkininc.com
uCOS-II	Micrium	www.micrium.com
uVelOSity	Green Hills Software	www.ghs.com

▲ 图8　Cortex-M3处理器的RTOS供应商