第17章

# 开始Cortex-M3开发

* 选择一款Cortex-M3产品
* Cortex-M3修订版0与修订版1的区别
* Cortex-M3修订版1与修订版2的区别
* 开发工具

## 17.1 选择一款Cortex-M3产品

在根据自己的应用选择具体的CM3芯片时，除了要考虑存储器、外设配置以及最高主频之外，其它一些因素也会使一款CM3芯片与众不同，CM3的设计允许下列参数是可以配置的，它们是：

* + 外中断的数目
  + 表达优先级的位数（优先级寄存器的有效宽度）
  + 是否配备了MPU
  + 是否配备了ETM
  + 对调试接口的选择（SW, JTAG或两者兼有）

对于大多数项目而言，单片机的功能和规格我们在选择时的首要考虑因素，例如：

* 1. 外设。对于大多数的项目，片载的外设是最重要的选择依据。外设也并非多多益善，因为它会影响到功耗和价格。
  2. 存储器：CM3单片机的闪存可以少到几KB，多至几MB。此外，片内RAM的容量也是很重要的。这些参数往往对价格有重大的冲击。
  3. 时钟速度：CM3的设计可以在0.18um的粗线条工艺上，也轻松上到100MHz。然而，因为存储器访问速度的限制，芯片厂商会降低最大主频。
  4. 脚印：CM3单片机的封装也多种多样。很多CM3单片机的脚数都比较少，以使之更适合于低成本的应用中。

## 17.2 Cortex-M3修订版0与修订版1的区别

早期的Cortex-M3产品是基于Cortex-M3处理器修订版0的。在2006年第3季度之后的CM3产品可以使用修订版1。在本书出版之时，所有的新CM3器件应该都是基于修订版1的。了解自己使用的芯片基于哪个修订版是很重要的，因为在修订版1中作出了许多重要的改变和改进。在本书前面章节中，都是按新的修订版1来叙述的。

在翻译本书时，有两个CM3芯片生产商，分别是Luminary和ST。译者查看了它们的资料，判定它们都是使用修订版1的处理器的。后续会有更多的CM3芯片生产商，但它们肯定不会使用老的修订版0了。

在编程模型中可以看见的改变包括如下内容：

* 从修订版1开始，响应异常时的寄存器操作可以被配置成强制对齐到双字边界，这可以通过置位NVIC\_CCR.STKALIGN来启用。
* 因为刚才的理由，NVIC\_CCR中加入了STKALIGN位
* 修订版1的修订版中引入了新的AUXFAULT（辅助fault）状态寄存器（可选）
* DWT中添加了诸如数值匹配的新功能
* ID寄存器的值因修订版号位段而改变

在编程模式中看不见的改变更多，它们是：

代码存储空间的存储器属性被硬线连接到可缓存，已分配（allocated），不可缓冲，不可共享。这会影响I-Code AHB和D-Code AHB，但是不会影响系统总线接口。

支持在I-Code AHB和D-Code AHB间的总线复用操作。在此操作模式下，可以使用一个简单的总线复用器来把I-Code和D-Code归并(merge)，这可以降低总门数，旧修订版的则必须使用ADK总线矩阵组件。

新添加了用于连接AHB跟踪单元(HTM)的输出端口。AHB是一个CoreSight中定义的调试组件，服务于复杂的数据跟踪操作。

调试组件或调试寄存器可以在系统复位期间访问，只有在上电复位时才无法访问。

在修订版１中，NVIC\_ICSR.VECTPENDING位段可以受NVIC\_DHCSR.C\_MASKINTS位的影响：当C\_MASKINTS置位时，如果掩蔽了一个悬起的中断，会使VECTPENDING的值为零。

JTAG-DP调试接口被SWJ-DP模块取代。但是仍然允许芯片厂商使用JTAG-DP，因为它也是CoreSight家庭中的成员。

因为修订版0的CM3在响应异常时没有双字对齐堆栈的功能，有些编译器，如ARM的RVDS和Keil的RVMDK ，都提供了特殊的编译选项以决定是否允许软件调整入栈，以使开发出来的产品是EABI兼容的，当软件需要与其它EABI-兼容开发工具时，这还是相当重要的。

为了判定使用的单片机使用了哪个修订版的CM3内核，可以使用NVIC中的CPUID寄存器，revison和变种位段指出了具体使用的CM3修订版。如表17.1所示：

表17.1 CPUID基寄存器

表17.1 CPUID基寄存器（地址：0xE000\_ED00）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 实现者  [31:24] | 变种  [23:20] | 常数  [19:16] | PartNo  [15:4] | Revision  [3:0] |
| 修订版0(r0p0) | 0x41 | 0x0 | 0xF | 0xC23 | 0 |
| 修订版1(r1p0) | 0x41 | 0x0 | 0xF | 0xC23 | 1 |
| 修订版1 (r1p1) | 0x41 | 0x1 | 0xF | 0xC23 | 1 |
| 修订版2(r2p0) | 0x41 | 0x2 | 0xF | 0xC32 | 0 |

译者查看了ST的STM32系列使用的内核，得到的结果是r1p1。

### 17.2.1 修订版1：从JTAG-DP到SWJ-DP

串行线JTAG调试端口（SWJ-DP）把SW-DP和JTAG-DP的功能合二为一，并且支持自动协议检测。使用这个组件，CM3设备可以支持both SW和JTAG接口。（目前可以使用的，由LM和ST所提供的芯片都是使用了SWJ-DP——译者注）。

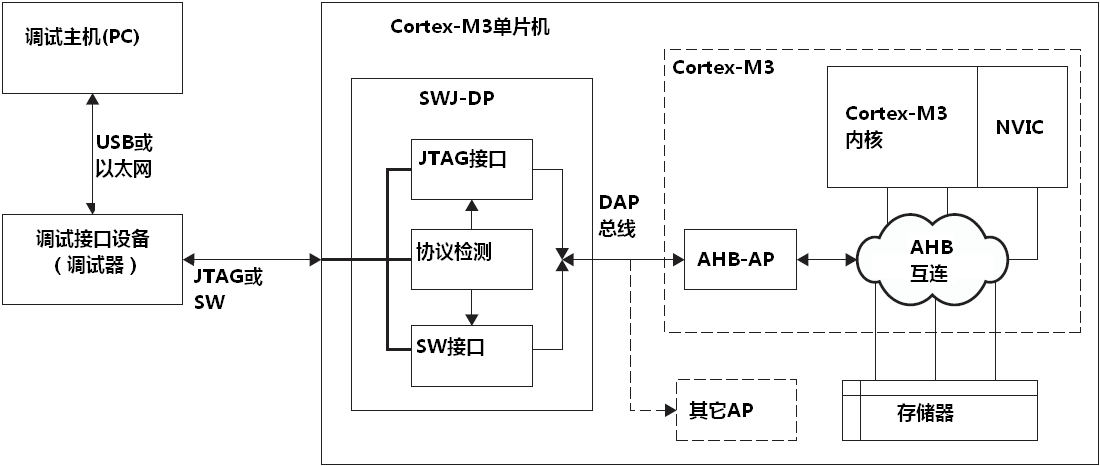


图17.1 SWJ-DP：合并了JTAG-DP和 SW-DP的功能

## 17.2 Cortex-M3修订版1与修订版2的区别

在2008年中期，Cortex-M3的修订版2发布了。估计到2008年底，在市场上就能见到基于修订版2的芯片了。修订版2新增了很多特性，它们大多数都致力于降低功耗以及提高调试的灵活性。

在修订版2中，程序员模式也跟着有以下的更新。

### 17.2.1 双字堆栈对齐方式成为缺省值

影响异常入栈顺序和内存使用的双字堆栈对齐方式，在修订版2中成为缺省使用的方式（注意：芯片厂商可能会选择使用修订版1的方式）。使用此方式，会给大多数C程序减少启动代码的额外开销（无需再在NVIC配置控制寄存器中置位STKALIGN比特）。

### 17.2.2 新增辅助控制寄存器（Auxiliary Control Register）

为了更细腻地调校处理器的行为方式，新增了辅助控制寄存器。比如，为了调试方便，通过设置此寄存器，可以关闭Cortex-M3的写缓冲，从而使总线faults总是能与存储器访问指令同步——也就是说使总线faults总是精确的。这样，就可以每次都能从入栈的返回地址中精确地揪出肇事指令了。

辅助控制寄存器的细节如下表所示：

辅助控制寄存器（0xE000\_E008）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特号 | 字段名 | 类型 | 初值 | 功能描述 |
| 2 | DISFOLD | R/W | 0 | 除能IT折叠(folding)，使IT指令与下一条指令（在流水线中）的执行级(execution phase)不会交迭 |
| 1 | DISDEFWBUF | R/W | 0 | 在缺省的存储器映射中除能写缓冲（对由MPU映射的regions不起作用） |
| 0 | DISMCYCINT | R/W | 0 | 除能“指令可中断”功能。也就是不再打断LDM, STM, 64位乘法，以及除法指令。 |

### 17.2.3 ID寄存器的更新

在NVIC以及调试组件中的很多ID寄存器都更新了。例如，在NVIC中的CPUID寄存器变成了

CPUID寄存器（0xE000\_ED00）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 实现者  [31:24] | 变种  [23:20] | 常数  [19:16] | PartNo  [15:4] | Revision  [3:0] |
| 修订版 2 (r2p0) | 0x41 | 0x2 | 0xF | 0xC23 | 0x0 |

### 17.2.4 调试功能

修订版2对调试功能有了好几处改进

* + DWT中的观察点数据跟踪现在支持两种新的跟踪方式：仅跟踪读传送，以及仅跟踪写传送。这样就可以仅在数据被改变或被读时才产生跟踪数据流，于是降低了数据跟踪所需的带宽。
  + 在实现调试特性时提供了更高的灵活性。比如，允许裁减可用的断点和观察点数，这样就降低了所设计产品的尺寸，这对于超低功耗的设计非常有帮助。
  + 对多核系统的调试，支持力度更大。为了实现多核同时重启和单步，新增了一个接口（注意，在程序员眼中看不到这种改变）。

### 17.2.5 睡眠特性

在系统级设计层上，现有的睡眠特性也得到了改进。在r2p0中，对处理器的唤醒可以延迟，从而使得在芯片中可以更大面积地“停电”，并且在系统中所有其它部件都就绪后才继续执行程序。这个改进主要是为了照顾下面一些应用：在它们里面，有一些硬件在低功耗模式下需要关闭，但是重新打开这些硬件需要的时间比较久。

在睡眠功能的扩展之外，为降低功耗还有新招。在旧版的CM3中，为了让内核能醒来，在睡眠期间，依然不能停止送往内核的“自由运行时钟”。尽管该时钟消耗的能量很低，但总归还是关了更省电。

为解决这个问题，可以在处理器外面布设一个简单的中断控制器。这个控制器，取名为“唤醒中断控制器（WIC）”。在深度睡眠期间，它要提供在NVIC中的，“中断掩蔽功能”的镜像，并且负责告知电源管理系统何时需要唤醒。这样，就可以在深度睡眠期间关断所有送往CM3处理器的时钟了。

除了可以停止时钟外，修订版2还可以使处理器的大多数部分都掉电，把它们的状态存储在若干特殊的逻辑小室中。在中断到达时，WIC往电源管理单元（PMU）发送一个唤醒请求。在处理器重新上电后，先前的状态从特殊的逻辑小室中恢复，然后就可以响应这个中断了。

可见，有了修订版2中这个新的掉电能力，CM3可以在深度睡眠期间进一步降低功耗。不过，这个特性还需要内核外的单元配合，因此不一定在所有修订版2的产品中都支持。

### 17.2.6 使用修订版2带来的好处和注意事项

那么，上文所讲的这些新特性，又对嵌入式产品开发带来了什么呢？

首先，是更低的功耗和更久的电池寿命。在进入了有WIC支持的深度睡眠后，整个电路就只有很小一部分还在活动中。此外，在要求极低功耗的应用中（如体内植入式医疗设备），芯片厂商可以通过减少断点和观察点的数量，来裁减芯片的尺寸。

第二，在调试和解决疑难问题的过程中，它提供了更好的灵活性。不仅体现在使用调试器的数据跟踪特性上，还新增了一个辅助控制寄存器。通过它我们可以给写缓冲做个旁路手术，从而使总线faults总是精确的。我们还可以使需要较多周期才能执行的指令不被打断，如LDM/STM指令。这样一来，在分析存储器的内容时就可以放心了。最后，对于使用多个Cortex-M3内核的系统，修订版2带来的“同时重启”和“多核单步执行”的功能正好雪中送炭。除此之外，在修订版2中还有若干个内部优化，以提高性能和改善接口特性。这样，芯片供应商就可以设计出更快的CM3产品。

然而，在享受温柔的同时，也请嵌入式程序员们留意下面的问题：

#### 双字堆栈对齐方式与异常堆栈帧

在缺省情况下，异常堆栈帧会自动对齐到双字存储器位置。早期为修订版0/1写的汇编程序，如果要通过堆栈来把数据传送给异常服务程序，可能会受到影响。为了准确判定堆栈帧的起始位置是否往下挪移了一个字，异常服务例程要先读取入栈PSR的比特9。如果不想动旧的程序，也可以手工把STKALIGN比特清除，这样就与以前的一样了。与EABI标准兼容的应用程序不会受影响。这些程序通常是C程序，并且使用与EABI兼容的编译器编译。

#### SysTick定时器也许会在深度睡眠期间停止。

如果使用的CM3单片机确实包含了掉电功能，或者是其它原因使得送往内核的时钟全体都在深度睡眠中停止，则SysTick定时器在深度睡眠期间就无法再运行。这样一来，使用了RTOS的嵌入式应用程序就需要一个外部时钟，用它来在唤醒时提供调度所需的滴答信号。

#### **当处理器连接到一个调试器时，会自动除能新的掉电功能**。

这是因为调试器需要访问处理器的调试相关寄存器。在调试会话中，能够控制内核停机或进入睡眠模式，但哪怕使能了掉电功能，也不会触发掉电序列。为了准确地测试掉电操作时的功耗，必须解除被测设备与调试器的连接。

## 17.4 开发工具

在开始使用Cortex-M3之前，需要准备好一些开发工具，典型的如：

* + 编译器/汇编器：把C和汇编源程序转换成目标文件。几乎所有的C编译器套件都包含了对应的汇编器。
  + 指令系统模拟器：模拟指令的执行，用于在软件开发早期的调试。
  + 在线仿真器（ICE）或者调试探测器（probe）：连接到电脑和目标板上的调试硬件，与目标板的接口通常是JTAG或SW。
  + 一块开发板。
  + 跟踪捕捉仪：可选的硬件设备和周边软件，可以用它来捕捉来自DWT以及ITM的输出，并且以可读的形式显示出来。
  + 嵌入式操作系统：在单片机上运行的操作系统。这也是一个可选件，许多简单的应用程序不需要操作系统。但是在开发复杂度较高或者有高性能指标的系统时，常常需要使用。

### 17.4.1 C编译器

截止到目前，已经有若干个C编译器套件可以使用了，如表17.3所列。

表17.3 支持Cortex-M3的开发工具

|  |  |
| --- | --- |
| 公司 | 产品 |
| ARM  [www.arm.com](http://www.arm.com) | Cortex-M3在RealView开发套件3.0(RVDS)中得到支持。在RealView-ICE 1.5可以用于连接调试硬件和调试环境。更早的ADS1.2和SDT不支持Cortex-M3 |
| KEIL(an ARM company)  [www.keil.com](http://www.keil.com) | 大名鼎鼎的KEIL，一度在8051的开发中享有盛誉。在其最新的Realview MDK开发工具中，支持了Cortrex-M3，其配套的仿真器是ULINK和ULINK2。 |
| CodeSourcery  [www.codesourcery.com](http://www.codesourcery.com) | 支持Cortex-M3的GNU工具链现在已经可用了，下载地址是[www.codesourcery.com/gnu\_toolchains/arm](http://www.codesourcery.com/gnu_toolchains/arm)。  它基于GNU 4.0版本 |
| Rowley Associates  [www.rowley.co.uk](http://www.rowley.co.uk) | 这个工具也源自GNU C编译器  [www.rowley.co.uk/arm/index.htm](http://www.rowley.co.uk/arm/index.htm) |
| IAR Systems  [www.iar.com](http://www.iar.com) | IAR Embedded Workbench for ARM and Cortex，它提供了C/C++编译器和调试环境（从4.40版本开始）。IAR在早在AVR单片机的开发中就是出类拔萃的。与IAR配套的仿真器是JLINK |
| Lauterbach  www.lauterbach.com | 提供了JTAG访真器和跟踪设备 |

### 17.4.2 嵌入式操作系统支持

上档次应用程序常常需要OS，尤其是RTOS。许多OS已经被开发出来用于嵌入式产品，目前，支持Cortex-M3的OS如表17.4所列：

表17.4 支持Cortex-M3的嵌入式操作系统

|  |  |
| --- | --- |
| 公司 | 产品 |
| FreeRTOS  www.freertos.org | FreeRTOS |
| Express Logic  www.expresslogic.com | ThreadX™ RTOS |
| Micrium  www.micrium.com | uC/OS-II |
| Accelerated Technology  www.acceleratedtechnology.com | Nucleus |
| Pumpkin Inc.  www.pumpkininc.com | Salvo RTOS |
| CMX Systems  www.cmx.com | CMX-RTX |
| KEIL  www.keil.com | ARTX-ARM |
| Segger  www.segger.com | embOS |
| IAR Systems  ww.iar.com | IAR PowerPac for ARM |
| T-Engine论坛  www.t-engine.org | uT-Kernel |
|  |  |