第九章 Cortex-M3异常和中断

9.1 Cortex-M3 异常

9.2 NVIC与中断控制

9.1 Cortex-M3 异常

9.1.1 异常类型

- 所有能打断正常执行流的事件都称为异常。CM3支持为数众多的系统异常和外部中断。
- 异常是另一种形式的中断，它是由内部fault引起的，或者内核的SysTick、SVCall等。而中断是由随机的外部事件引发的。
- 编号为1～15的对应系统异常；编号为16～255的对应外部中断。
- 除了个别异常的优先级被定死外，其它异常的优先级都是可编程的。
- 当前运行的异常编号，是由特殊寄存器IPSR或NVIC的中断控制状态寄存器来给出的。

9.1 Cortex-M3 异常

异常表

异常号	异常类型	优先级	描 述
0	N/A	N/A	没有异常在运行
1	复位	-3 (最高)	复位
2	NMI	-2	不可屏蔽中断(外部NMI 输入)
3	硬件fault	-1	各种fault情况
4	内存管理fault	可编程	内存管理fault ； MPU 访问非法地址
5	总线fault	可编程	总线fault，比如预取终止
6	用法fault	可编程	由于程序fault或尝试访问协处理器导致的异常
7-10	保留	N/A	—
11	SVCall	可编程	系统服务调用
12	调试监视器	可编程	调试监视器

9.1 Cortex-M3 异常

(续)

13	保留	N/A	—
14	PendSV	可编程	可挂起系统设备申请
15	SysTick	可编程	系统时钟定时器
16	外部中断#0	可编程	外部中断
17	外部中断#1	可编程	外部中断
…	…	…	…
255	外部中断#239	可编程	外部中断

当一个被使能的异常发生时，如果它不能够被立即执行，它将被挂起(pending)。

9.1 Cortex-M3 异常

9.1.2 优先级定义

- 在CM3中，优先级对于异常来说很关键的，它决定一个异常是否能被屏蔽，以及在未被屏蔽的情况下何时可以响应。
- 优先级的数值越小，则优先级越高。
- CM3支持中断嵌套，使得高优先级异常会抢占(preempt)低优先级异常。
- 3个系统异常：复位、NMI以及硬fault有固定的优先级，并且它们的优先级号是负数，从而高于所有其它异常。
- Cortex-M3支持256级可编程异常，所有其它异常的优先级则都是可编程的。
- 减少优先级数可以通过减少优先级配置寄存器的一些低位来实现。

9.1 Cortex-M3 异常

3bits优先级

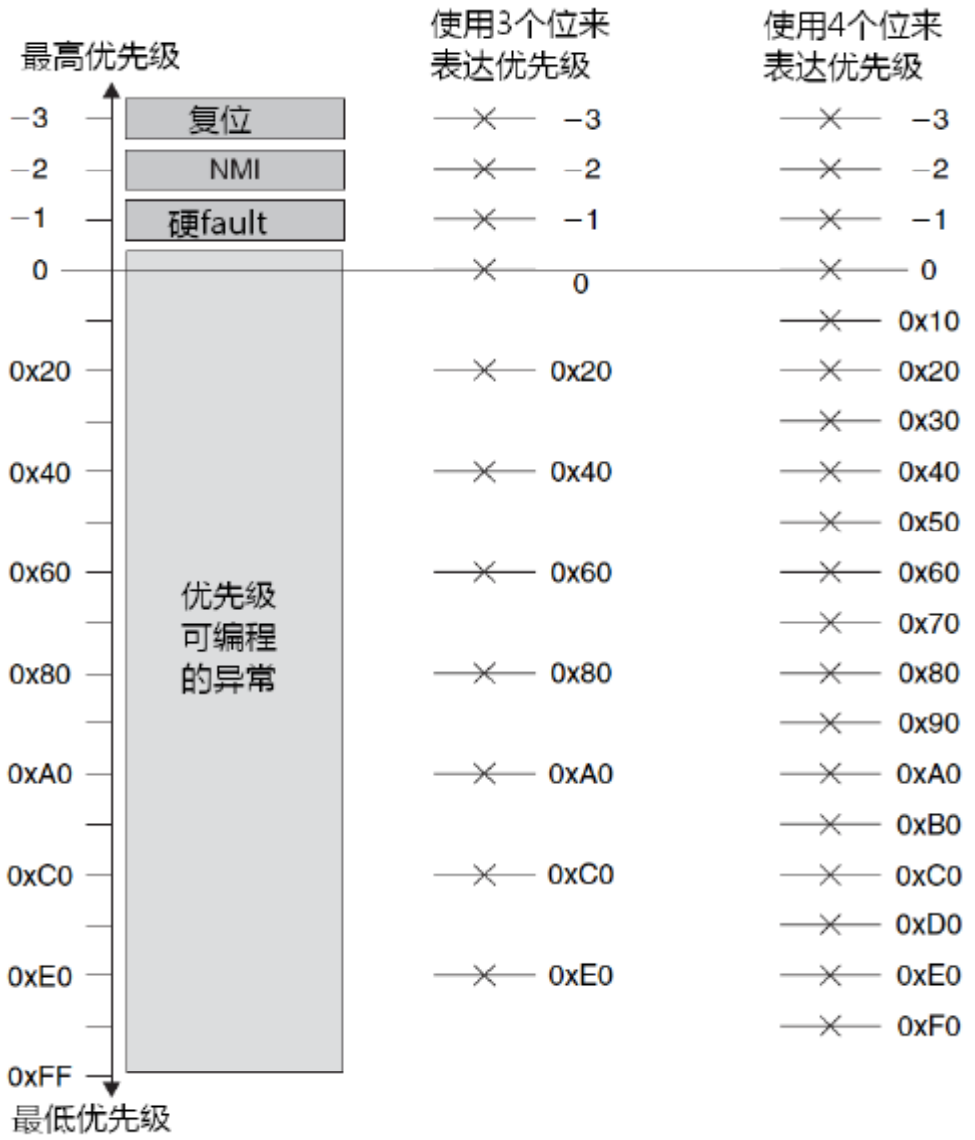
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
使 用			不使用, 读出值为0				

4bits优先级

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
使 用				不使用, 读出值为0			

优先级寄存器的最小宽度值为3比特.

9.1 Cortex-M3 异常



3位表达的优先级VS.4位表达的优先级

9.1 Cortex-M3 异常

使用3-bit, 5-bit, 和8-bit 优先级寄存器

优先级	异常类型	3比特表达	5比特表达	8比特表达
-3 (Highest)	复位	-3	-3	-3
-2	NMI	-2	-2	-2
-1	硬件错误	-1	-1	-1
0 1 … 0xFF	具有优先级并且可编程的异常	0x00 0x20 … 0xE0	0x00 0x08 … 0xF8	0x00, 0x01 0x02, 0x03 … 0xFE, 0xFF

9.1 Cortex-M3 异常

抢占优先级和次优先级

通过NVIC中“应用程序中断及复位控制寄存器的位段“PRIGROUP优先级组”设置。该位段的值对每一个优先级可配置的异常都有影响，把其优先级分为2个位段：MSB所在的位段（左边的）对应抢占优先级，而LSB所在的位段（右边的）对应次优先级。

对于不同的优先组设置，抢占优先字段和次优先级字段在优先级寄存器中的定义

优先组	抢占优先级字段	次优先级字段
0	Bit [7:1]	Bit [0]
1	Bit [7:2]	Bit [1:0]
2	Bit [7:3]	Bit [2:0]
3	Bit [7:4]	Bit [3:0]
4	Bit [7:5]	Bit [4:0]
5	Bit [7:6]	Bit [5:0]
6	Bit [7]	Bit [6:0]
7	None	Bit [7:0]

9.1 Cortex-M3 异常

应用程序中断和复位控制寄存器AIRCR(地址0xE000ED0C)

Bits	名　称	类型	复位值	描述
31:16	VECTKEY	R/W	—	存取关键字；为了写入这个寄存器必须向这个地址写人 0x05FA
15	ENDIANNESS	R	—	表示数据的排列顺序： 1 表示大端(BE8) 而0表示小端
10:8	PRIGROUP	R/W	0	优先级分组
2	SYSRESETREQ	W	—	请求芯片控制逻辑产生一个复位信号
1	VECTCLRACTIVE	W	—	清除异常的所有活跃状态信息
0	VECTRESET	W	—	复位Cortex-M3 处理器。 但不会复位处理器以外的电路。

优先组设置的例子

Bit 7	Bit 6	Bit 5	Bit 4 ～ Bit 0
抢占级		次	不使用, 读出值0

Bit 7	Bit 6	Bit 5	Bit 4 ～ Bit 2	Bit 1	Bit 0
抢占级			不使用	次优先级	

配置寄存器的宽度	3	3
优先组	5	1
抢占优先级	4	8
次优先级数	2	—
Bit 7	抢占优先级	抢占优先级[7:5]
Bit 6		
Bit 5	次优先级	
Bit 4		抢占优先级bit[4:2] (未使用，总是0)
Bit 3		
Bit 2		
Bit 1		次优先级(未使用，总是0)
Bit 0		

9.1 Cortex-M3 异常

9.1.3 向量表

响应异常时，CM3需要定位其服务例程的入口地址。这些入口地址被存储在“（异常）向量表”中。

上电后的异常向量表

地址	异常号	值(大小为“字”)
0x00000000	—	MSP 初始值
0x00000004	1	复位向量(程序计数器初始值)
0x00000008	2	NMI 服务例程的入口地址
0x0000000C	3	硬件故障服务例程的入口地址
…	…	其它异常服务例程的入口地址

9.1 Cortex-M3 异常

通过设置NVIC中的*向量表偏移寄存器*，可以将向量表重定位到其它的内存地址。

该地址偏移需要与向量表的大小对齐， 扩展到2的若干次幂

例：

IRQ 输入: 32

总异常数： 32 + 16 (system exceptions) = 48

扩展到2的幂: 64.

乘以4: 256.

向量表地址偏移可以被编程为0x0, 0x100, 0x200等等。

9.1 Cortex-M3 异常

向量表偏移寄存器VTOR(地址0xE000ED08)

Bits	名称	类型	复位值	描述
29	TBLBASE	R/W	0	表基地址在Code (0) 或RAM (1)
28:7	TBLOFF	R/W	0	来自CODE区或RAM区的表偏移范围

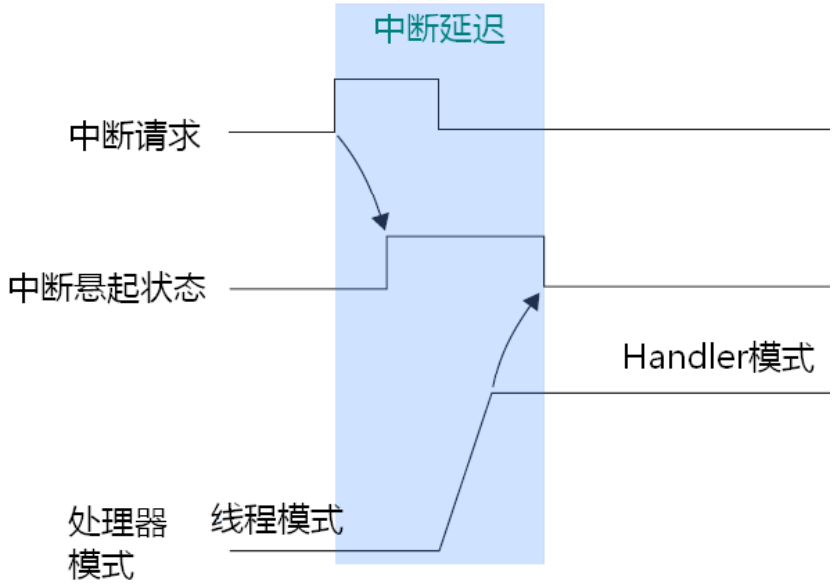
为了允许动态修改向量表， 向量表的起始处包含以下向量：

1. 主堆栈指针的初始值;
2. 复位向量;
3. NMI向量;
4. 硬件fault服务例程。

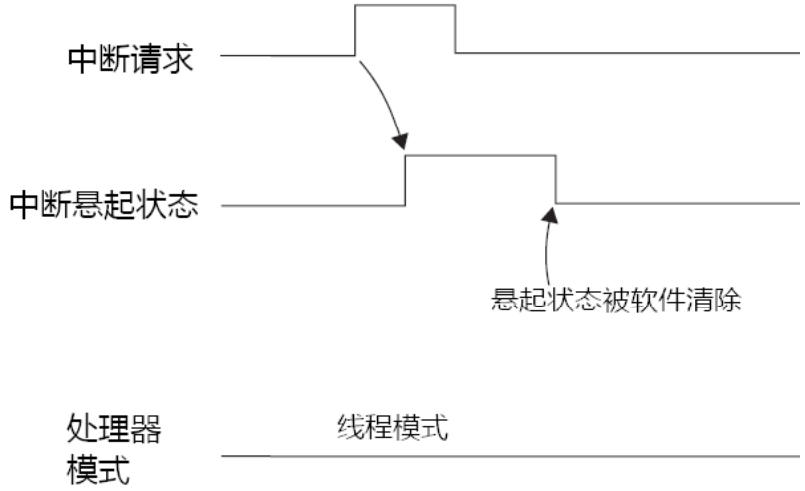
9.1 Cortex-M3 异常

9.1.4 中断输入和悬起行为

- 当中断输入脚被置为有效（assert）后，该中断就被悬起。即使后来中断源撤消了中断请求，已经被标记成悬起的中断也被记录下来。到了系统中它的优先级最高的时候，就会得到响应。
- 如果在某个中断得到响应之前，其悬起状态被清除了（例如，在PRIMASK或FAULTMASK置位的时候软件清除了悬起状态标志），则中断被取消。



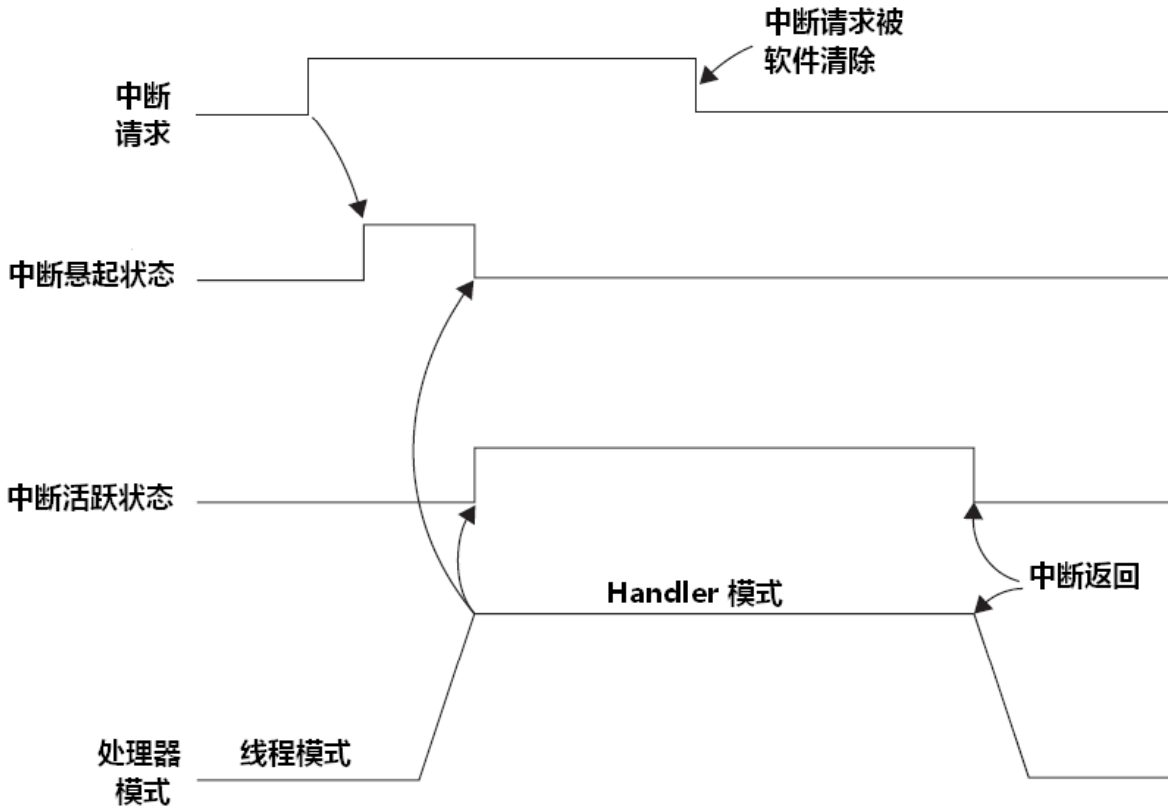
中断挂起



中断在得到处理器响应之前被清除悬起状态

9.1 Cortex-M3 异常

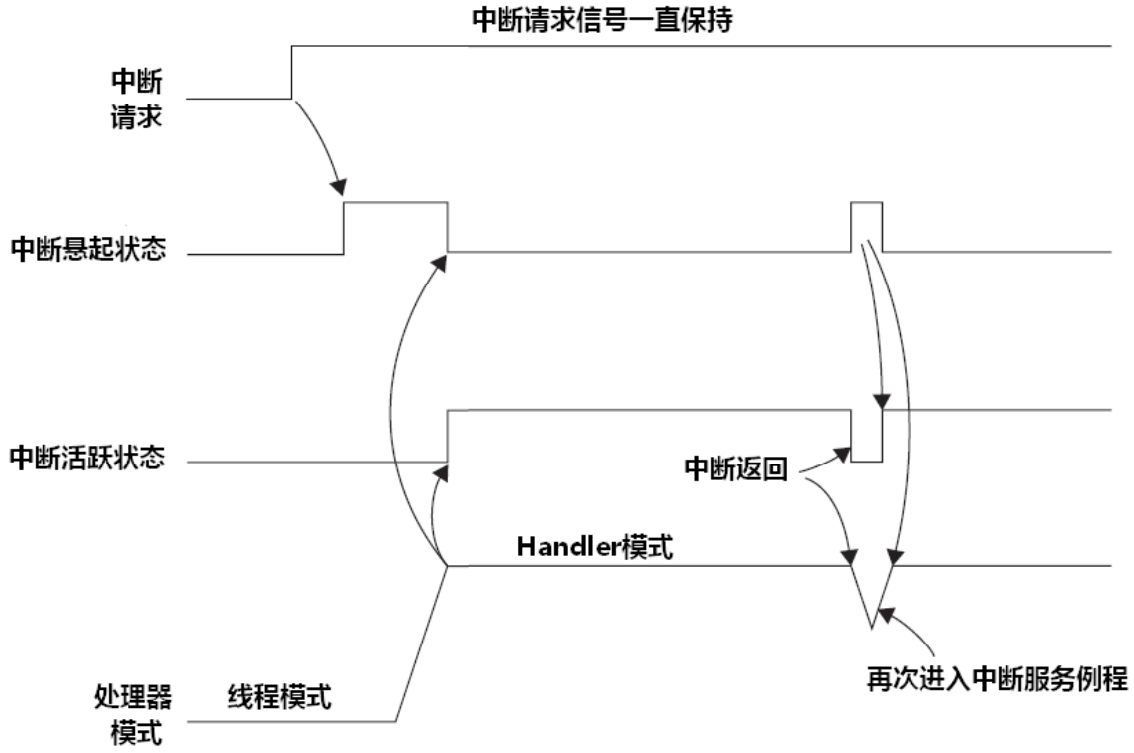
当处理器开始执行一个中断，称其进入“激活”状态，同时挂起位将被自动清除。



处理器进入服务例程后，中断激活状态的设置

9.1 Cortex-M3 异常

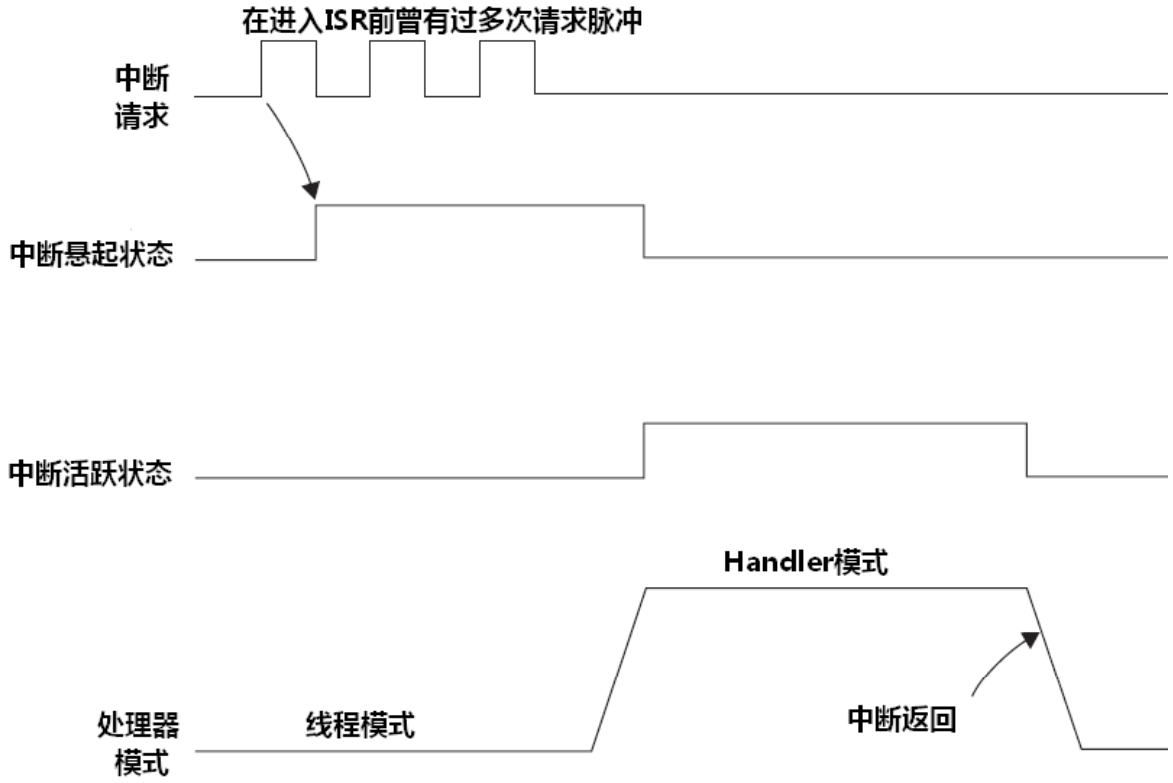
如果一个中断源持续保持中断请求信号活跃，在中断服务程序结束时，中断将被再次挂起。



在中断结束后，连续中断请求再次挂起

9.1 Cortex-M3 异常

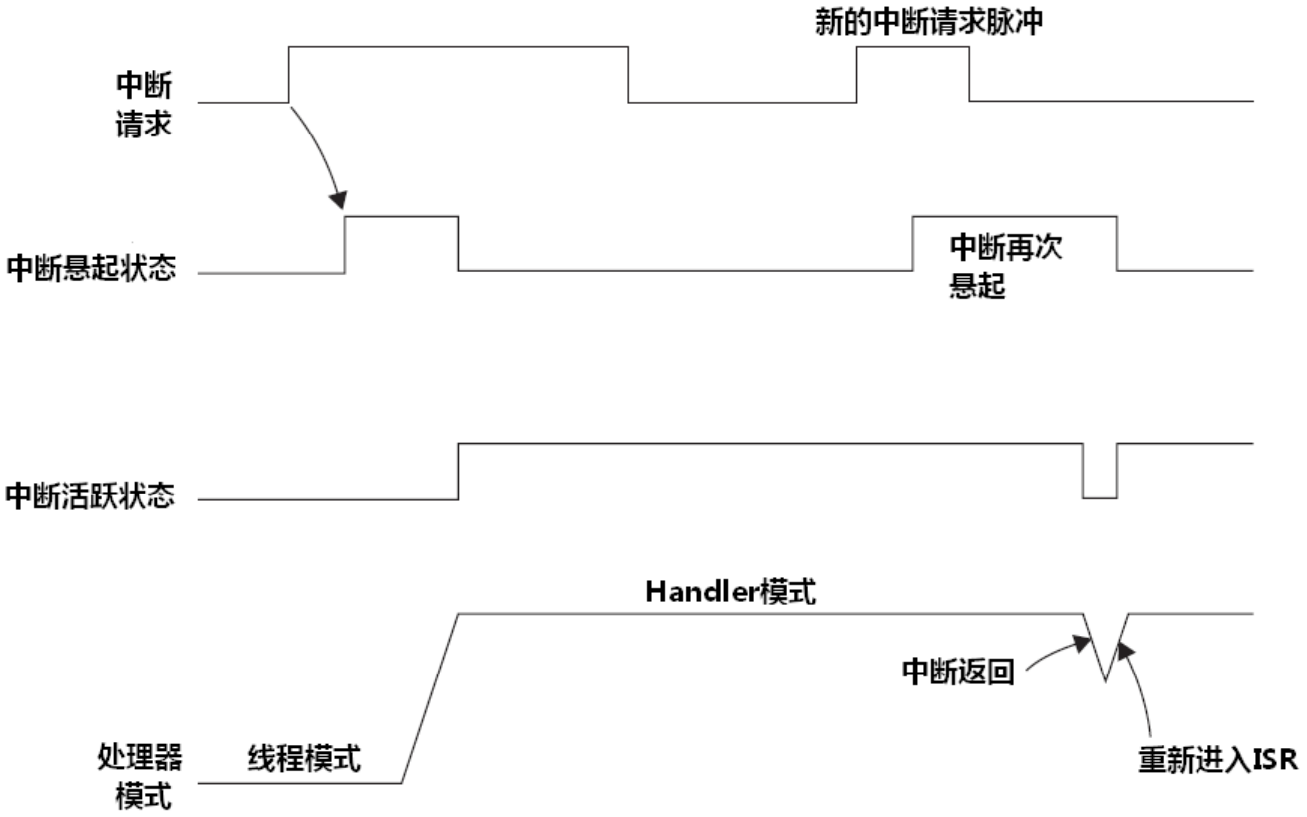
如果一个中断在处理器执行前，其请求信号多次以脉冲的方式出现，则将被视为一次中断请求，而不是多次。



只有一次中断挂起，即使在服务例程执行前有多个脉冲

9.1 Cortex-M3 异常

如果一个中断请求释放，但在服务例程结束前再次被置为有效，那么它将仍然会被再次挂起。



执行服务例程期间再次挂起中断

9.1 Cortex-M3 异常

9.1.5 Fault异常

有若干个系统异常专用于fault处理。CM3中的Faults可分为以下几类:

- 总线faults
- 存储器管理faults
- 用法faults
- 硬fault

9.1.5.1 总线错误

当在AHB接口数据传输过程中一个fault响应被接收时，总线fault将会产生。

总线fault可能发生的情况:

1. 取指令;
2. 数据读/写;
3. 在中断处理开始阶段的压栈;
4. 中断处理结束阶段的出栈;
5. 当处理器启动中断服务序列后，读取一个中断向量地址。

9.1 Cortex-M3 异常

当这些类型的总线fault发生时:

If

1. 该总线fault的服务例程被使能.
2. 没有其它同级的异常在运行.

Then

该总线服务例程将被执行.

Else if

同时处理器接收另一个更高优先级的异常.

Then

总线fault的异常将被挂起.

在一些特殊的情况下, 硬件fault服务例程将被执行，内核将进入锁定状态。

9.1 Cortex-M3 异常

NVIC 有一些fault状态寄存器. 其中一个就是***总线fault状态寄存器*** (BFSR).

总线fault状态寄存器BFSR(0xE000ED29)

Bits	名	类型	复位值	描述
7	BFARVALID	—	0	=1时表示BFAR 无效
6:5	—	—	—	—
4	STKERR	R/Wc	0	入堆栈时发生错误
3	UNSTKERR	R/Wc	0	出堆栈时发生错误
2	IMPREISERR	R/Wc	0	不精确的数据访问违例
1	PRECISERR	R/Wc	0	精确的数据访问违例
0	IBUSERR	R/Wc	0	取指的访问违例

9.1 Cortex-M3 异常

9.1.5.2 存储器管理Fault

常见的存储器管理fault包括：

1. 在MPU启动后，访问了MPU区域之外的地址；
2. 访问了没有存储器与之对应的空地址；
3. 从不可执行的存储器区域试图取指；
4. 向只读区域写数据；
5. 在用户级下访问了只允许特权级下访问的特权地址。

9.1 Cortex-M3 异常

当存储器fault发生时，

if

1. 该存储器管理fault服务例程被使能。
2. 没有其它相同或更高优先级异常在运行。

Then

存储器管理fault服务例程将被执行。

Else if

同时，处理器接受另一个具有更高有优先级的异常。

Then

存储器管理fault异常将被挂起。

在一些特殊的情况下，硬件fault服务例程将被执行，或者内核将进入锁定状态。

9.1 Cortex-M3 异常

NVIC包括一个存储器管理fault寄存器(MFSR) 来映射存储器管理fault的情况。

存储器管理fault寄存器MFSR(0xE000ED28)

Bits	名称	类型	复位值	描述
7	MMARVALID	—	0	=1表示MMAR有效
6:5	—	—	—	—
4	MSTKERR	R/Wc	0	入堆栈时发生错误
3	MUNSTKERR	R/Wc	0	出堆栈时发生错误
2	—	—	—	—
1	DACCVIOL	R/Wc	0	数据访问违例
0	IACCVIOL	R/Wc	0	取指访问违例

9.1 Cortex-M3 异常

9.1.5.3 用法Fault

可以引起用法fault的有：

1. 执行了协处理器指令;
2. 执行了未定义指令;
3. 尝试转换到ARM状态;
4. 无效中断返回(链接寄存器包括无效/不正确的数值);
5. 使用多重加载或存储指令时，地址没有对齐;

通过设置NVIC中的一些位， 产生用法fault:

1. 除以0;
2. 任何非连续内存访问.

9.1 Cortex-M3 异常

当一个用法fault产生时，

if

1. 用法fault服务例程使能。
2. 没有相同或更高优先级的异常运行。

Then

用法fault异常将被执行。

Else if

同时处理器接收另一个更高优先级的异常。

Then

用法fault异常将被挂起。

在一些特殊的情况下，硬件fault服务例程将被执行，或者内核将进入锁定状态。

9.1 Cortex-M3 异常

NVIC提供一个*用法fault寄存器* (UFSR) 使得用法fault异常程序能够确定fault的原因。

用法fault寄存器UFSR(0xE000ED2A)

Bits	名称	类型	复位值	描述
9	DIVBYZERO	R/Wc	0	表示出现了除以0的fault (仅当DIV_0_TRP 置位有效)
8	UNALIGNED	R/Wc	0	访问一个地址不连续区域的fault
7:4	—	—	—	—
3	NOCP	R/Wc	0	试图执行一个协处理器指令
2	INVPC	R/Wc	0	试图去处理一个有fault返回值的异常
1	INVSTATE	R/Wc	0	试图切换到无效状态(e.g., ARM)
0	UNDEFINSTR	R/Wc	0	试图去执行没有定义的指令

9.1 Cortex-M3 异常

9.1.5.4 硬件Fault

可以引起硬件fault的有：

1. 用法fault， 总线fault， 内存管理fault;
2. 总线fault期间的获取向量.

硬件fault寄存器HFSR(0xE000ED2C)

Bits	名称	类型	复位值	描述
31	DEBUGEVT	R/Wc	0	表示硬件fault是由调试时间触发的的
30	FORCED	R/Wc	0	表示硬件fault是由总线fault， 内存管理fault或用法fault引起的
29:2	—	—	—	—
1	VECTBL	R/Wc	0	表明硬件fault是由总线fault期间的获取向量引起的
0	—	—	—	—

9.1 Cortex-M3 异常

9.1.5.5 Fault处理

在一个真实运行的系统中，在检查到faults的原因后，软件需要决定下一步要做什么。

一些faults处理办法：

1. 复位：

使用在NVIC中的”应用程序中断和复位控制寄存器”里的VECTRESET位(只复位处理器内核，不复位其它片上设备)。

2. 恢复：

尽可能解决产生fault异常的问题的。

3. 任务终止：

如果系统运行了一个RTOS，则相关的任务可以被终止或者重新开始。

9.1 Cortex-M3 异常

9.1.6 SVC 和PendSV

SVC (系统服务调用) 和***PendSV*** (挂起的系统调用) 是两个目标为软件和操作系统的异常。

9.1.6.1 SVC

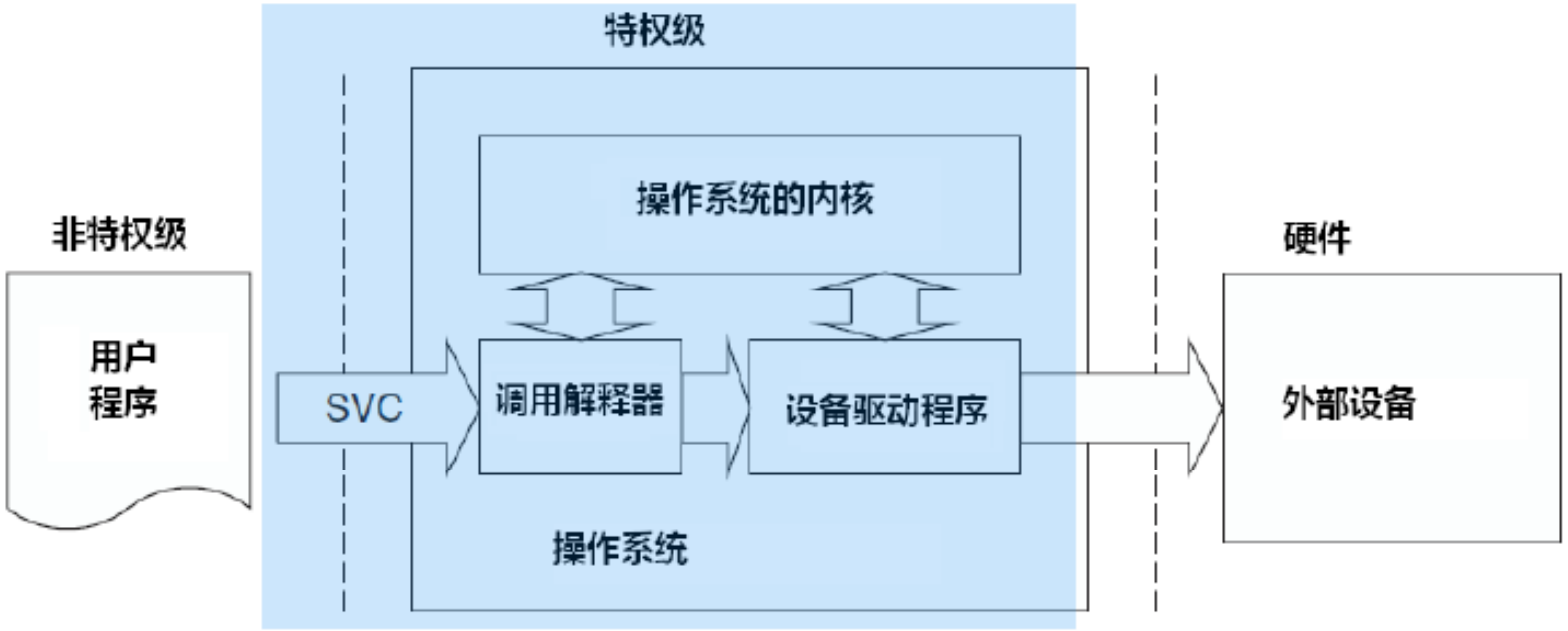
SVC是用来产生系统函数的调用请求。

例:

操作系统不允许用户程序直接访问硬件，而是通过提供一些系统服务函数，让用户程序使用SVC发出对系统服务函数的呼叫请求，以此调用它们来间接访问硬件。

SVC可以提高软件的可移植性，因为用户程序无需知道硬件的编程细节。

9.1 Cortex-M3 异常



SVC 作为操作系统函数Gateway

使用SVC 指令产生SVC

例:

```
SVC   0x3   ; Call SVC function 3
```

9.1 Cortex-M3 异常

6.2 PendSV

PendSV (挂起系统调用)和SVC协同使用。

对于SVC异常，必须在执行SVC指令后立即得到响应，同时，应用程序执行SVC时希望所提出的请求立即得到响应。

对于PendSV，其可以像普通的中断一样被挂起。另外，OS通过挂起一个异常来保证一个动作在另一个很重要的任务完成后再执行。这样的挂起很有用。

PendSV挂起的方法：

手动向NVIC的PendSV挂起寄存器中写入1。

挂起后，如果优先级不够高，则将等待执行。

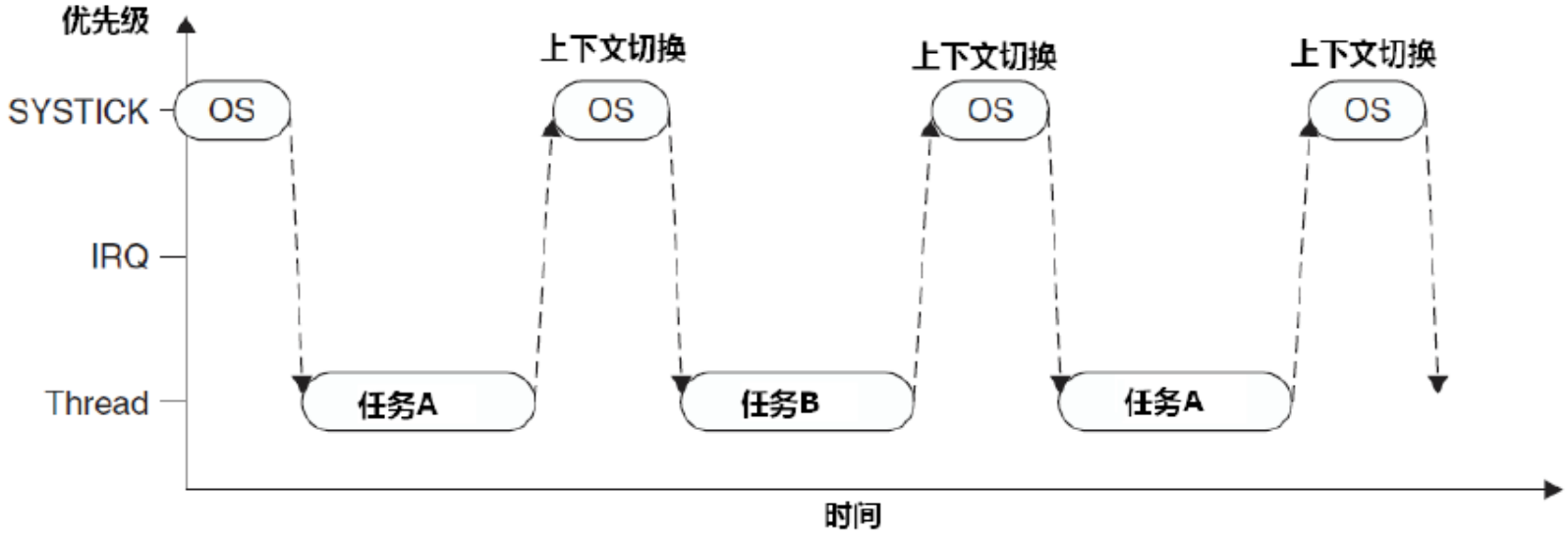
9.1 Cortex-M3 异常

PendSV一个典型的用途是上下文切换(不同任务的切换)

例:

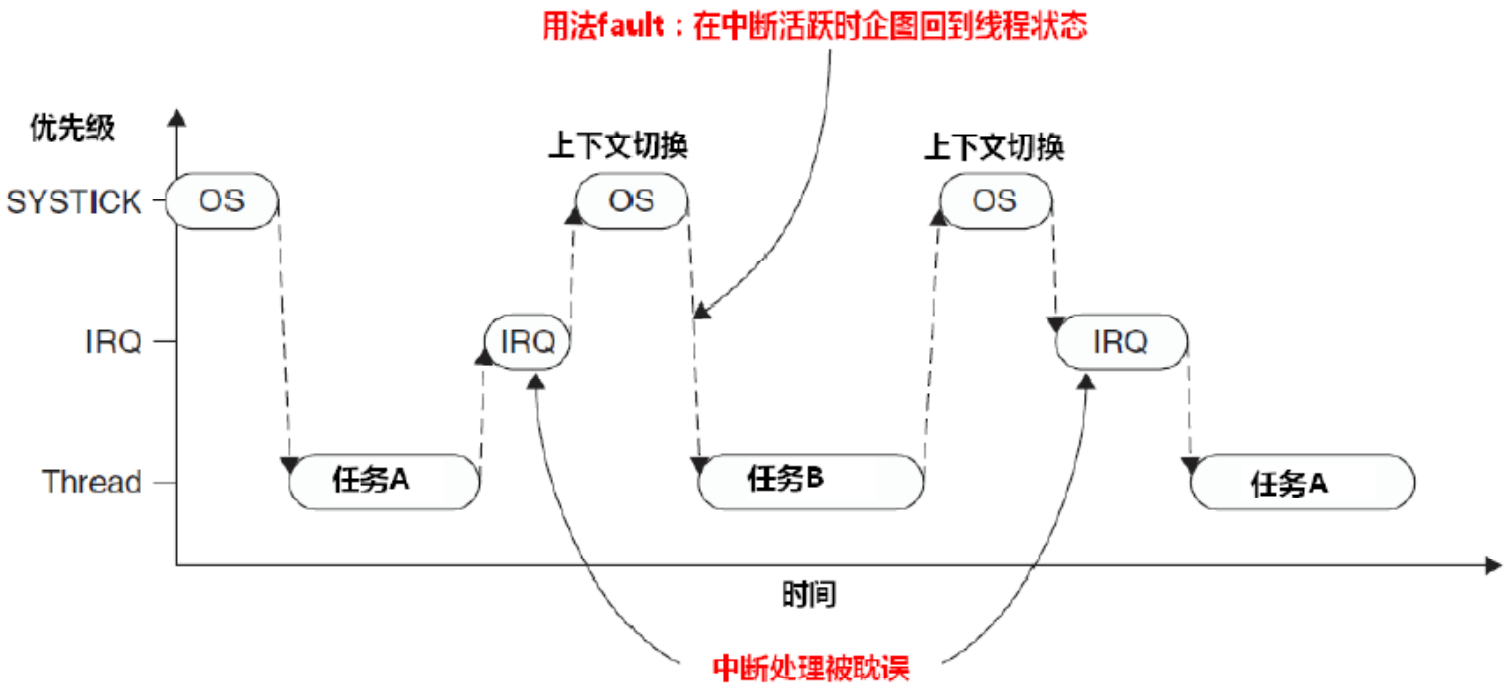
一个只有两个就绪任务的系统， 并且通过SYSTICK异常来启动上下文切换的。

如果在SYSTICK异常之前有一个中断请求，SYSTICK异常将会抢占IRQ句柄。



两个任务间通过SysTick进行轮转调度的简单模式

9.1 Cortex-M3 异常



发生IRQ时上下文切换的问题

9.2 NVIC与中断控制

9.2.1 NVIC 概述

- NVIC（Nested Vectored Interrupt Controller，嵌套向量中断控制器）是Cortex-M3 处理器的一个重要组成部分。
- NVIC 支持1 到240 外部中断输入和一个 不可屏蔽中断(NMI) 输入。
- NVIC 可以通过内存地址0xE000E000访问。
- 大多数中断控制/状态寄存器只能够在特权级下访问，只有软件触发中断寄存器可以在用户级下访问以产生软件中断。
- 所有的中断控制／状态寄存器均可按字／半字／字节的方式访问。

9.2 NVIC与中断控制

9.2.2 基本中断配置

每个外部中断有一些相应的寄存器，包括：

1. 使能和清除使能寄存器;
2. 设置挂起和清除挂起寄存器;
3. 优先级寄存器;
4. 激活状态寄存器.

另外，下列寄存器也可以影响中断处理

1. 异常屏蔽寄存器(PRIMASK, FAULTMASK以及BASEPRI)
2. 中断向量偏移寄存器;
3. 软件触发中断寄存器;
4. 优先级分组.

9.2 NVIC与中断控制

9.2.3 中断使能和清除使能

各个中断的使能和清除使能分别使用各自的寄存器来控制。

如允许一个中断，则需向对应的SETENA的位写1；

如屏蔽一个中断，则需向对应的CLRENA的位写1；

如写入0，则无效.

外部中断：上限到240，则SETENA位和CLRENA位可达240对，对应的32位寄存器需有8对，采用数字后缀来区分该寄存器，分别为SETENA0/CLRENA0, SETENA1/CLRENA1 ... SETENA7/CLRENA7。对于具体的芯片，只有该芯片实现的中断，其对应的位才有意义。

SETENA: 0xE000E100－0xE000E11C

CLRENA: 0xE000E180－0xE000E19C

9.2 NVIC与中断控制

中断使能寄存器和中断清除使能寄存器

名称	类型	地址	复位值	描述
SETENA0	R/W	0xE000_E100	0	中断 0-31 的使能寄存器，共 32 个使能位 位[n]，中断#n 使能（异常号 16+n）
SETENA1	R/W	0xE000_E104	0	中断 32-63 的使能寄存器，共 32 个使能位
…	…	…	…	…
SETENA7	R/W	0xE000_E11C	0	中断 224-239 的使能寄存器，共 16 个使能位
CLRENA0	R/W	0xE000_E180	0	中断 0-31 的除能寄存器，共 32 个除能位 位[n]，中断#n 除能（异常号 16+n）
CLRENA1	R/W	0xE000_E184	0	中断 32-63 的除能寄存器，共 32 个除能位
…	…	…	…	…
CLRENA7	R/W	0xE000_E19C	0	中断 224-239 的除能寄存器，共 16 个除能位

9.2 NVIC与中断控制

9.2.4 中断挂起和清除挂起

如果一个中断发生，但是不能够立刻执行，它将被挂起。

中断挂起状态可以通过中断挂起设置(SETPEND)和中断挂起清除(CLRPEND)寄存器来控制。

与中断使能/中断屏蔽寄存器用法类似，可以有多达8对的SETPEND和CLRPEND寄存器。

SETPEND：0xE000E200-0xE000E21C

CLRPEND：0xE000E280-0xE000E29C

9.2 NVIC与中断控制

中断挂起设置寄存器和中断挂起清除寄存器

名称	类型	地址	复位值	描述
SETPEND0	R/W	0xE000_E200	0	中断 0-31 的悬起寄存器，共 32 个悬起位 位[n]，中断#n 悬起（异常号 16+n）
SETPEND1	R/W	0xE000_E204	0	中断 32-63 的悬起寄存器，共 32 个悬起位
…	…	…	…	…
SETPEND7	R/W	0xE000_E21C	0	中断 224-239 的悬起寄存器，共 16 个悬起位
CLRPEND0	R/W	0xE000_E280	0	中断 0-31 的解悬寄存器，共 32 个解悬位 位[n]，中断#n 解悬（异常号 16+n）
CLRPEND1	R/W	0xE000_E284	0	中断 32-63 的解悬寄存器，共 32 个解悬位
…	…	…	…	…
CLRPEND7	R/W	0xE000_E29C	0	中断 224-239 的解悬寄存器，共 16 个解悬位

9.2 NVIC与中断控制

9.2.4.1 优先级

每一个外部中断都有一个相关联的优先级控制寄存器(3-8 bit位宽).

中断优先级控制寄存器(0xE000E400-0xE000E4EF)

名称	类型	地址	复位值	描述
PRI_0	R/W	0xE000E400	0 (8-bit)	外中断#0的优先级
PRI_1	R/W	0xE000E401	0 (8-bit)	外中断#1的优先级
…	…	…	…	..
PRI_239	R/W	0xE000E4EF	0 (8-bit)	外中断#239的优先级

9.2 NVIC与中断控制

9.2.4.2 激活状态

每一个外部中断都有一个激活状态位。当处理器开始执行中断处理程序时，该位设置为1；在中断返回时清零。

中断激活状态寄存器(0xE000E300-0xE000E31C)

名称	类型	地址	复位值	描述
ACTIVE0	R	0xE000E300	0	中断#0-31的活动状态寄存器
ACTIVE1	R	0xE000E304	0	中断#32-63的活动状态寄存器
…	…	…	…	…
ACTIVE7	R	0xE000E31C	0	中断#224-239的活动状态寄存器

9.2 NVIC与中断控制

9.2.4.3 PRIMASK和FAULTMASK特殊功能寄存器

PRIMASK寄存器通过将当前的优先级改为0来屏蔽所有异常。

该寄存器可以通过使用MRS和MSR命令来编程控制。

Example:

```
MOV   R0,   #1
MSR   PRIMASK,   R0    ; Write 1 to PRIMASK to disable all interrupts
```

And:

```
MOV   R0,   #0
MSR   PRIMASK,   R0    ; Write 0 to PRIMASK to allow interrupts
```

FAULTMASK寄存器通过将当前的优先级改为-1来屏蔽所有异常。FAULTMASK会在异常退出时自动清零。

9.2.4.4 BASEPRI特殊功能寄存器

BASEPRI寄存器能够屏蔽优先级小于特定值的中断。

Example:

屏蔽不高于0x60的中断:

```
MOV   R0,   #0x60
MSR   BASEPRI,   R0      ; Disable interrupts with priority 0x60-0xFF
```

取消对中断的屏蔽:

```
MOV   R0,   #0x0
MSR   BASEPRI,   R0      ; Turn off BASEPRI masking
```

BASEPRI寄存器同样能够通过BASEPRI_MAX名称进行访问

9.2 NVIC与中断控制

使用BASEPRI_MAX寄存器名访问时，只能够改变到一个更高的优先级。

Example:

```
MOV   R0,   #0x60
MSR   BASEPRI_MAX,   R0 ; Disable interrupts with priority 0x60,
                         ; 0x61,..., etc
MOV   R0,   #0xF0
MSR   BASEPRI_MAX,   R0 ; This write will be ignored because
                          ; it is lower level than 0x60
MOV   R0,   #0x40
MSR   BASEPRI  MAX,   R0 ; This write is allowed and change the
                         ; masking level to 0x40
```

9.2 NVIC与中断控制

9.2.4.5 其它异常的配置寄存器

用法异常、内存管理异常和总线异常通过系统处理控制和状态寄存器使能。

系统处理控制和状态寄存器 (0xE000ED24)

位	名称	类型	复位值	描述
18	USGFAULTENA	R/W	0	用法fault服务例程使能位
17	BUSFAULTENA	R/W	0	总线fault服务例程使能位
16	MEMFAULTENA	R/W	0	存储器管理fault服务例程使能位
15	SVCALLPENDED	R/W	0	SVC悬起中。本来已经要SVC服务例程，但是却被更高优先级异常取代
14	BUSFAULTPENDED	R/W	0	总线fault悬起中，细节同上。

9.2 NVIC与中断控制

(Continued)

位	名称	类型	复位值	描述
13	MEMFAULTPENDED	R/W	0	存储器管理fault悬起中，细节同上
12	USGFAULTPENDED	R/W	0	用法fault悬起中，细节同上
11	SYSTICKACT	R/W	0	SysTick异常活动中
10	PENDSVACT	R/W	0	PendSV异常活动中
8	MONITORACT	R/W	0	Monitor异常活动中
7	SVCALLACT	R/W	0	SVC异常活动中
3	USGFAULTACT	R/W	0	用法fault异常活动中
1	BUSFAULTACT	R/W	0	总线fault异常活动中
0	MEMFAULTACT	R/W	0	存储器管理fault异常活动中

9.2 NVIC与中断控制

中断控制和状态寄存器(0xE000ED04)

位	名称	类型	复位值	描述
31	NMIPENDSET	R/W	0	写1以悬起NMI
28	PENDSVSET	R/W	0	写1以悬起PendSV。读取它则返回PendSV的状态
27	PENDSVCLR	W	0	写1以清除PendSV悬起状态
26	PENDSTSET	R/W	0	写1以悬起SysTick。读取它则返回PendSV的状态
25	PENDSTCLR	W	0	写1以清除SysTick悬起状态
23	ISRPREEMPT	R	0	=1时，则表示一个悬起的中断将在下一步时进入活动状态（用于单步执行时的调试目的）
22	ISRPENDING	R	0	1=当前正有外部中断被悬起（不包括NMI）
21:12	VECTPENDING	R	0	悬起的ISR的编号
11	RETTOBASE	R	0	如果异常返回后将回到基级(base level)，并且没有其它异常悬起时，此位为1
9:0	VECTACTIVE	R	0	当前活动的ISR编号

9.2 NVIC与中断控制

9.2.5 中断系统设置全过程示例

1. 设置优先级组寄存器;
2. 将硬件故障和NMI处理例程复制到一个新的矢量表位置;
3. 设置矢量表偏移寄存器;
4. 读取矢量表偏移寄存器，计算准确的中断处理程序内存地址，设置中断向量;
5. 设置中断优先级;
6. 使能中断。

The program in assembly:

```
LDR R0, =0xE000ED0C      ; 应用程序中断及复位控制寄存器
LDR R1, =0x05FA0500      ; 使用优先级组5 (2/6)
STR R1, [R0]             ;设置优先级组
```

9.2 NVIC与中断控制

```
…
MOV R4, #8                    ; ROM中的向量表
LDR R5, =(NEW_VECT_TABLE+8)
LDMIA R4!, {R0-R1}            ;读取NMI和硬fault的向量
STMIA R5!, {R0-R1}            ;拷贝它们的向量到新表中
…
LDR R0, =0xE000ED08           ;向量表偏移量寄存器的地址
LDR R1, =NEW_VECT_TABLE
STR R1, [R0]                  ;把向量表重定位
…
LDR R0, =IRQ7_Handler  ;取得IRQ #7服务例程的入口地址
LDR R1, =0xE000ED08    ;向量表偏移量寄存器的地址
LDR R1, [R1]
ADD R1, R1, #(4*(7+16))  ;计算IRQ #7服务例程的入口地址

STR R0, [R1]                  ;在向量表中写入IRQ #7服务例程的入口地址
…
LDR R0, =0xE000E400     ;外部中断优先级寄存器组的基地址
```

9.2 NVIC与中断控制

```
MOV R1, #0xC0
STRB R1, [R0,#7]          ;把IRQ #7的优先级设置为0xC0
…
LDR R0, =0xE000E100       ; SETEN寄存器的地址

MOV R1, #(1<<7)           ;置位IRQ #7的使能位
STR R1, [R0]              ;使能IRQ #7
```

中断控制类型寄存器给出了所支持的输入中断的数量，粒度为32

中断控制类型寄存器**ICTR(0xE000E004)**

Bits	Name	Type	Reset Value	Description
4:0	INTLINESNUM	R	–	Number of interrupt inputs in step of 32 0 = 1 to 32 1 = 33 to 64 …

9.2 NVIC与中断控制

9.2.6 软件中断

软件中断可由以下方式触发:

1. 使用相应的SETPEND寄存器
2. 软件触发中断寄存器(STIR)

软件触发中断寄存器STIR(0xE000EF00)

位段	名称	类型	复位值	描述
8:0	INTID	W	-	影响编号为INTID的外部中断，其悬起位被置位。例如，写入8，则悬起IRQ #8

系统异常(NMI, faults, PendSV等等)不能被该寄存器挂起

9.2 NVIC与中断控制

9.2.7 系统定时器

系统定时器(SYSTICK)与NVIC集成在一起，能够用于产生一个SYSTICK异常。

系统定时器由四个寄存器控制。

SYSTICK 控制和状态寄存器(0xE000E010)

位段	名称	类型	复位值	描述
16	COUNTFLAG	R	0	如果在上次读取本寄存器后，SysTick已经计到了0，则该位为1。如果读取该位，该位将自动清零
2	CLKSOURCE	R/W	0	0=外部时钟源(STCLK) 1=内核时钟(FCLK)
1	TICKINT	R/W	0	1=SysTick倒数计数到0时产生SysTick异常请求 0=数到0时无动作
0	ENABLE	R/W	0	SysTick定时器的使能位

9.2 NVIC与中断控制

SYSTICK重载寄存器 (0xE000E014)

位段	名称	类型	复位值	描述
23:0	RELOAD	R/W	0	当倒数计数至零时，将被重装载的值

SYSTICK当前值寄存器 (0xE000E018)

位段	名称	类型	复位值	描述
23:0	CURRENT	R/Wc	0	读取时返回当前倒计数的值，写它则使之清零，同时还会清除在SysTick控制及状态寄存器中的COUNTFLAG标志

9.2 NVIC与中断控制

SYSTICK 校准数值寄存器 (0xE000E01C)

位段	名称	类型	复位值	描述
31	NOREF	R	-	1=没有外部参考时钟（STCLK不可用） 0=外部参考时钟可用
30	SKEW	R	-	1=校准值不是准确的10ms 0=校准值是准确的10ms
23:0	TENMS	R/W	0	在10ms的间隔中倒计数的格数。芯片设计者应该通过Cortex-M3的输入信号提供该数值。若该值读回零，则表示无法使用校准功能

系统定时器同时能够被用于:

1. 作为报警定时器
2. 测量时间

END