AN99231

在 FM0+系列 S6E1C3 器件中使用中断

作者： Mily Wang

相关器件系列：FM0+

相关代码示例：无

相关应用笔记：AN54460、AN90799

要获取本应用笔记的最新版本或相关项目文件，请访问 http://www.cypress.com/go/AN99231。

本应用笔记介绍了 FM0+ MCU 系列中 ARM Cortex-M0+ MCU 的嵌套向量中断控制器（NVIC）功能的基本概念。另外，还说明了新设计的 S6E1C3 系列 NVIC 的操作和优点。

目录

1 简介 2
2 概述 2
2.1 NVIC 架构 2
2.2 NVIC 特性 2
3 S6E1C3 系列 NVIC 设计 3
3.1 通用设计 4
3.2 新设计 4
3.3 中断向量表和中断列表 4
4 示例代码 6
4.1 中断向量表的示例代码 7
4.2 主子程序示例代码 9
4.3 ISR 示例代码 10
5 性能比较 12
5.1 代码空间 12
5.2 执行速度 12
5.3 编码复杂性 13
6 总结 13
7 使用注意事项 13
A 附录 A：异常和中断 14
文档修订记录 18
全球销售和设计支持 19
产品 19
PSoC®解决方案 19
赛普拉斯开发者社区 19
技术支持 19

1 简介

S6E1C3 系列器件高度集成了 32 位微控制器和 ARM Cortex-M0+处理器。它们还包含了新的 NVIC 设计，通过该设计可以提高系统效率。Cortex-M0+处理器支持由 NVIC 汇总和处理的某些中断和系统异常。

本应用笔记介绍了 FM0+系列中 NVIC 功能的基本概念、S6E1C3 系列中的 NVIC 新设计以及新设计 NVIC 的操作。本文档还提供了应用笔记内容。

2 概述

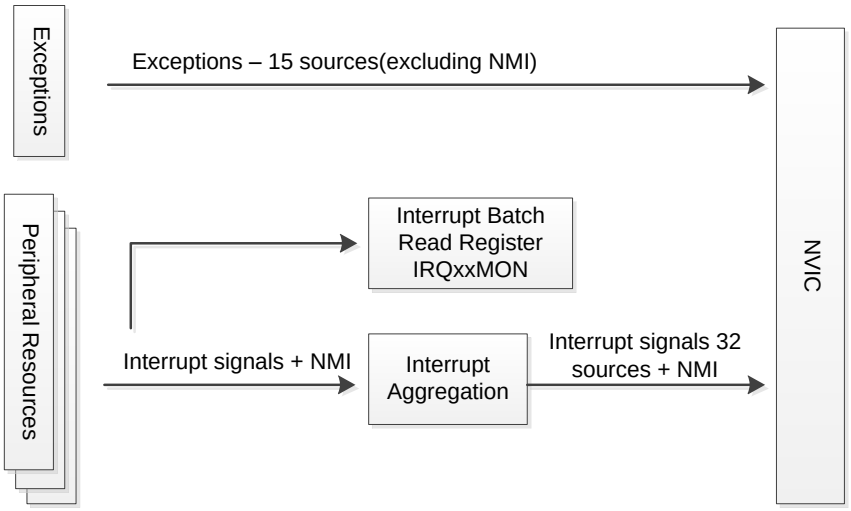
Cortex-M0+的 CPU 内核带有内部 NVIC。NVIC 确定中断请求的优先级，并将请求发送给 CPU。

2.1 NVIC 架构

图 2-1 显示的是 FM0+中断功能的结构。异常是指系统的复位、SysTick 以及硬故障等请求。这些请求被输送给 NVIC 模块，并对应异常编号。更多有关信息，请参考《Cortex-M0+技术参考手册》。

某些外设的中断信号和异常被汇总起来并被输送给相应的中断向量。中断向量的总数为 32 个（不可屏蔽中断（NMI）源除外）。名为“IRQxxMON”的一系列中断批量寄存器指出在系统中发生的中断请求。

图 2-1. FM0+中断架构



2.2 NVIC 特性

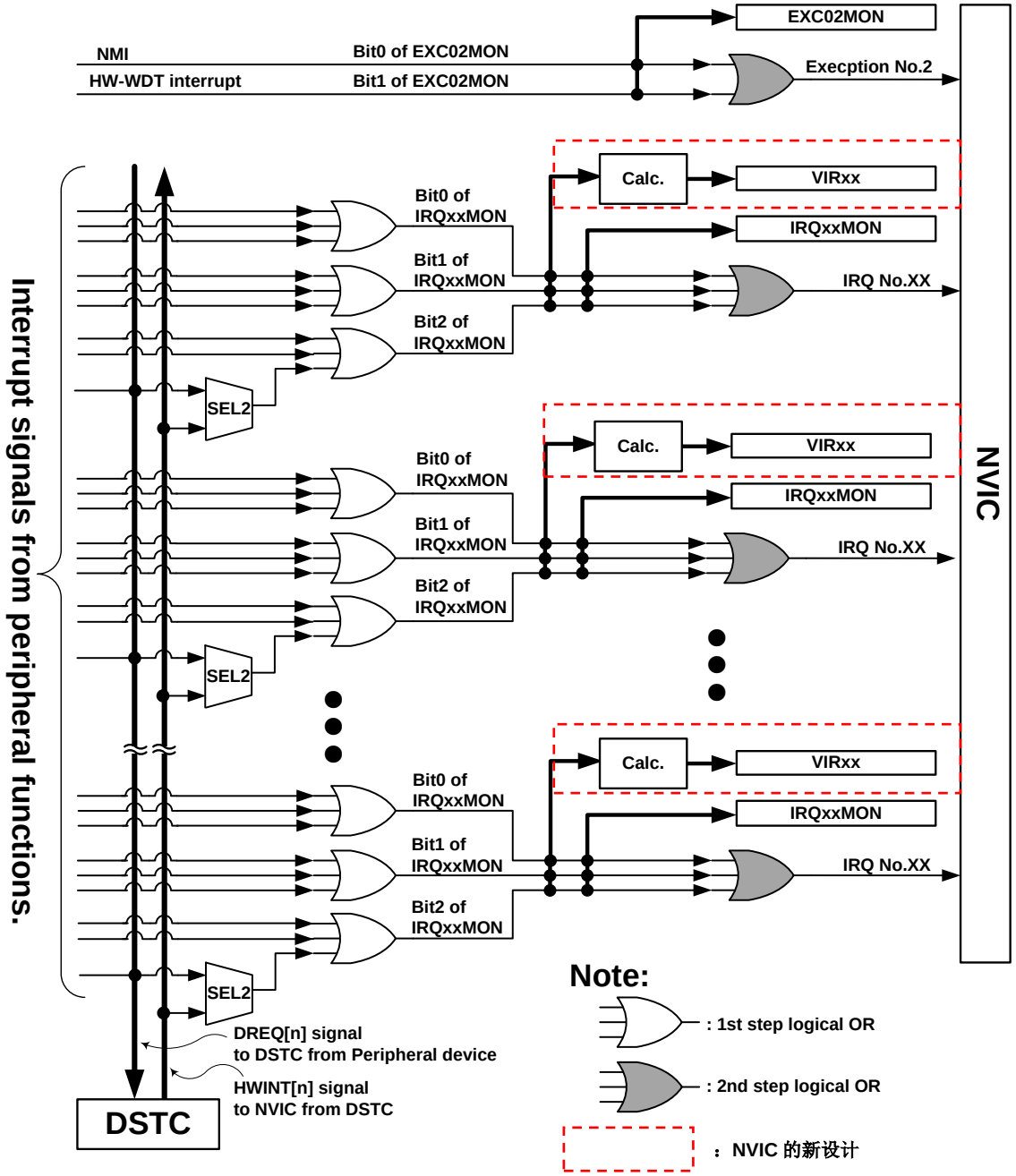
FM0+系列的 NVIC 支持以下特性：

- 支持 32 个可屏蔽的外设中断通道（Cortex-M0+的 16 个异常中断通道除外）
- 支持 4 种可编程中断优先级（使用 2 位来设定中断优先级）
- 支持 NMI 输入
- 支持低延迟处理异常和中断
- 支持系统控制寄存器的执行

3 S6E1C3 系列 NVIC 设计

图 3-1 详细介绍了外设中断信号和 NVIC 模块间的连接。该章节将介绍 S6E1C3 器件中 NVIC 的通用设计和新设计。

图 3-1. 中断信号和 NVIC 间的连接



选中 DSTC 的情况下进行 DMA 传输时，将发生来自 DSTC 的传输完成中断（而不生成来自外设的中断）。关于每个外设功能的 DSTC 传输请求的更多信息，请参考外设手册中外设功能的相关章节。

下面的红色虚线框中显示的是 NVIC 的新设计。图 3-1 中其余部分则显示的是 NVIC 的通用设计。

3.1 通用设计

3.1.1 中断向量表

中断向量表如表 3-1 所示。从 0x00000000 到 0x000000BC 的地址部分用于表示通用设计。

3.1.2 NMI 和硬件看门狗中断

外部中断和 NMI 控制器的 NMI 信号以及硬件看门狗定时器的硬件（HW）看门狗中断通过第二级门逻辑 OR 结合起来，然后被连接到第二个 NVIC 异常的输入端。当发生第二个异常中断时，可以通过读取 EXC02 批读取寄存器（如图 3-1 所示）来识别它们的中断源（是 NMI 或 HW 看门狗中断）。请参考外设手册，了解 EXC02MON 的详细信息。

3.1.3 其他外设中断

外设功能的中断信号通过两级的 OR 电路结合在一起（如图 3-1 所示）。然后，将所汇总的中断信号连接到 NVIC 的 32 个外设中断中的某一个。请参见表 A-1，了解分配到 NVIC 的 IRQ 输入的外设功能中断信号。

生成某个中断时，通过读取中断批量读取寄存器（即为图 3-1 中的 IRQxxMON），可以指出中断源。IRQxxMON 寄存器包含 NVIC 的所有中断输入。NVIC 支持了 32 个寄存器（IRQ00MON 到 IRQ31MON）。请参考外设手册，了解 IRQxxMON 的详细信息。

3.1.4 中断批量读取寄存器

IRQxxMON 寄存器指出所发生的中断。请参见表 A-1，了解 IRQxxMON 寄存器和外设中断之间的关系。

3.2 新设计

新设计以通用设计为基础。它添加了一些特殊设计，旨在提高便利性和效率。

3.2.1 中断向量表

中断向量表如表 3-1 所示。从 0x000000C0 到 0x000001FC 的地址是为新设计而添加的（红色虚线框所示）。

3.2.2 向量指示寄存器

发生中断时，CPU 将通过使用向量指示寄存器（VIRxx）快速启动中断分支操作。VIRxx 寄存器是 32 个寄存器（VIR00 到 VIR31），对应于 NVIC 的 IRQ00–IRQ31 输入。发生中断时，CPU 可以读取 VIRxx 寄存器的子程序地址值。通过将中断处理程序的高位地址放置在该地址区中，可以快速分支中断操作。下面解释了发生 IRQ00 时 VIR00 的行为和使用方式。

如表 3-1 所示，该程序将 IRQ 中断处理程序的高位地址值写入到 0x0000 0040 到 0x0000 00BC 地址区域中。它也将对应于位 0、1、2 的 IRQ 中断处理程序的高位地址值写入到 0x0000 00C0 到 0x0000 01FC 地址区域中。

发生 IRQ00 时，将启动 IRQ00 中断处理程序，并且 CPU 将读取 VIR00 的值。从 VIR00 读取的值显示在表 3-2 中。如果发生 Bit0 中断，从 VIR00 读取的值将为 0x0000 00C0。如果发生 Bit1 中断，VIR00 读取的值将为 0x0000 0140。另外如果发生 Bit2 中断，则从 VIR00 读取的值将为 0x0000 01C0。IRQ00 中断处理程序将执行分支到另一个中断，而该中断函数位于从 VIR00 寄存器读取的地址。通过该方式，可快速执行中断处理分支操作。

3.2.3 VIR 偏移寄存器

通过 VIR 偏移寄存器（VIR_OFFSET），可以定义 VIRxx 的共同偏移值。

3.3 中断向量表和中断列表

表 3-1 显示的是 S6E1C3 中断向量表。如果采用通用设计，中断向量表地址会在 0x00000000 到 0x000000BC 的范围内。如果采用新设计，中断向量表地址将在 0x00000000 到 0x000001FC 的范围内。从 0x000000C0 到 0x000001FC 的地址是为了新设计而添加的（表 3-1 中红色虚线框所示）。

表 3-1. 中断向量表

地址	数据
0x0000 0000	堆栈指针初始值
0x0000 0004	异常 1：复位向量
0x0000 0008	异常 2：NMI/HW-WDT 处理程序的高位地址
0x0000 000C	异常 3：硬故障处理程序的高位地址
0x0000 0010 - 0x0000 0028	保留
0x0000 002C	异常 12：SoCal 处理程序的高位地址
0x0000 0030 - 0x0000 0034	保留
0x0000 0038	异常 14：PendSV 处理程序的高位地址
0x0000 003C	异常 15：SysTick 处理程序的高位地址
0x0000 0040	IRQ00 处理程序的高位地址
0x0000 0044	IRQ01 处理程序的高位地址
....	....
0x0000 00B8	IRQ30 处理程序的高位地址
0x0000 00BC	IRQ31 处理程序的高位地址
0x0000 00C0	IRQ00 – bit0 处理程序的高位地址
0x0000 00C4	IRQ01 – bit0 处理程序的高位地址
....	....
0x0000 0138	IRQ30 – bit0 处理程序的高位地址
0x0000 013C	IRQ31 – bit0 处理程序的高位地址
0x0000 0140	IRQ00 – bit1 处理程序的高位地址
0x0000 0144	IRQ01 – bit1 处理程序的高位地址
....	....
0x0000 01B8	IRQ30 – bit1 处理程序的高位地址
0x0000 01BC	IRQ31 – bit1 处理程序的高位地址
0x0000 01C0	IRQ00 – bit2 处理程序的高位地址
0x0000 01C4	IRQ01 – bit2 处理程序的高位地址
....	....
0x0000 01F8	IRQ14 – bit2 处理程序的高位地址
0x0000 01FC	IRQ15 – bit2 处理程序的高位地址

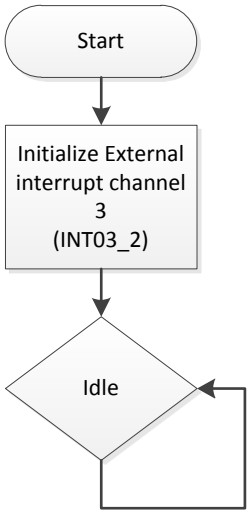
表 3-2. IRQ00 的状态以及从 VIR00 读取的值

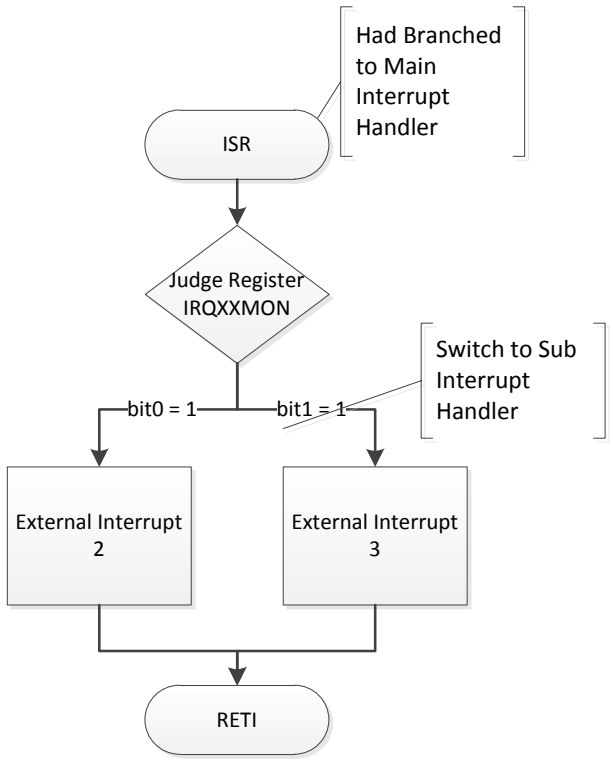
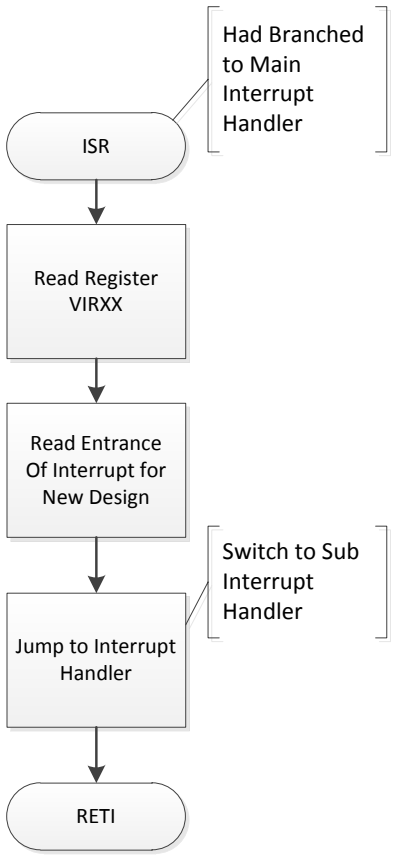
IRQ00 输入信号状态			从 VIR00 读取的值
Bit0 中断状态	Bit1 中断状态	Bit2 中断状态	
1	忽略	忽略	0x0000 00C0 + VIR_OFFSET
0	1	忽略	0x0000 0140 + VIR_OFFSET
0	0	1	0x0000 01C0 + VIR_OFFSET
0	0	0	未定义值

4 示例代码

应用代码包含了中断向量表代码、主子程序代码以及中断服务子程序（ISR）代码。该章节介绍了外部中断频道 3 的应用方法。并选择引脚 INT03_2 用于外部中断功能。新设计和通用设计的源代码包含相同的主子程序代码。它们具有几乎相同的中断向量表，但执行 ISR 代码的流程不相同。更多详细信息，请参考表 4-1。

表 4-1. 通用设计和新设计的应用方法

代码	通用设计	新设计
中断向量表	源代码如代码 1 和代码 2 所示。	中断的入口遵循中断向量表。源代码如代码 1 和代码 2 所示。
主子程序	 源代码显示在代码 4 中。	

代码	通用设计	新设计
ISR	 源代码显示在代码 5 中。	 源代码显示在代码 6 中。

执行 ISR 代码的流程需要向中断处理程序分支两次。第一个中断处理器，也称为“主中断处理程序”（仅适用于本文档），由内核的中断控制器访问。第二个中断处理器，也称为“子中断处理程序”（仅适用于本文档），由软件来访问。ISR 流程的区别在于子中断处理程序的分支方法。请参见表 4-1 中的 “ISR”。

4.1 中断向量表的示例代码

Keil 编译器与 IAR 编译器的中断向量表源代码不太一样。代码 1 显示的是 Keil 编译器的中断向量表源代码的一部分。它包括异常的 16 个向量和外设中断入口的 32 个向量。新设计的中断程序处理入口符合中断向量表中的内容。

代码 1. Keil 编译器的中断向量表的源代码（适用于通用设计和新设计的多种应用）

```
                ;System Exceptions
__Vectors       DCD     __initial_sp              ; Top of Stack
                DCD     Reset_Handler             ; Reset Handler
                DCD     NMI_Handler               ; NMI Handler
                DCD     HardFault_Handler         ; Hard Fault Handler
                DCD     0                         ; Reserved
                DCD     0                         ; Reserved
                DCD     0                         ; Reserved
                DCD     0                         ; Reserved
```

```
                DCD     0                           ; Reserved
                DCD     0                           ; Reserved
                DCD     0                           ; Reserved
                DCD     SVC_Handler                 ; SVCall Handler
                DCD     0                           ; Reserved
                DCD     0                           ; Reserved
                DCD     PendSV_Handler              ; PendSV Handler
                DCD     SysTick_Handler             ; SysTick Handler
                ;Interrupt Handler
                DCD     CSV_SWDT_LVD_IRQHandler     ; 0:
                DCD     MFS0_IRQHandler             ; 1:
                DCD     MFS1_IRQHandler             ; 2:
                DCD     0                           ; 3:
                … …
                DCD     DT_RTC_WC_IRQHandler        ; 15:
                DCD     INT0_1_IRQHandler           ; 16:
                DCD     INT2_3_IRQHandler           ; 17:
                DCD     INT4_5_IRQHandler           ; 18:
                DCD     INT6_7_IRQHandler           ; 19:
                ……
                DCD     ICC1_FLASH_IRQHandler       ; 29:
                DCD     DSTC_IRQHandler             ; 30:
                DCD     0                           ; 31:
                ;Followed the General Design Vector Table
                ;Interrupt Handler of New Design
                SPACE   68
                DCD     EXINT2_IRQHandler           ; At 0x104
                SPACE   124
                DCD     EXINT3_IRQHandler           ; At 0x184
```

代码 2 显示的是 IAR 编译器的中断向量表源代码的一部分。该代码增加了两段，用于配置外部中断处理程序。同时，需要在 IAR 项目中的 ICF 文件中定义额外段的位置。请参见代码 2 和代码 3。

代码 2. IAR 编译器中断向量表的源代码（适用于通用设计和新设计的多种应用）

```
                SECTION .exint2vec:CODE:ROOT(2)
                PUBLIC  __myvector_exitint2_table
                DATA
__myvector_exitint2_table DCD     EXINT2_IRQHandler;

                SECTION .exint3vec:CODE:ROOT(2)
                PUBLIC  __myvector_exitint3_table
                DATA
__myvector_exitint3_table DCD     EXINT3_IRQHandler;

            THUMB

                PUBWEAK EXINT2_IRQHandler
                SECTION .text:CODE:NOROOT:REORDER(1)
EXINT2_IRQHandler
                B       EXINT2_IRQHandler

                PUBWEAK EXINT3_IRQHandler
                SECTION .text:CODE:NOROOT:REORDER(1)
EXINT3_IRQHandler
                B       EXINT3_IRQHandler
```

代码 3. ICF 文件中的位置定义

```
define symbol __NewDesign_extint2_start__ = 0x104;
define symbol __NewDesign_extint3_start__ = 0x184;

place at address mem:__NewDesign_extint2_start__ { readonly section .exint2vec };
place at address mem:__NewDesign_extint3_start__ { readonly section .exint3vec };
```

4.2 主子程序示例代码

代码 4 显示的是主子程序的源代码。

代码 4. 主子程序的源代码（适用于通用设计和新设计的多种应用）

```
int32_t main(void)
{
    /* Initialize INT03_2 interrupt*/
    FM0P_GPIO->PFR3_f.P0 = 1;          // Use a pin as peripheral function
    FM0P_GPIO->EPFR06_f.EINT03S = 3u; // Chose peripheral pin INT03_2
    FM0P_EXTI->ELVR_f.LA3 = 1u;
    FM0P_EXTI->ELVR_f.LB3 = 1u;        // Falling edge
```

```
    FM0P_EXTI->EICL_f.ECL3 = 0;        // clear interrupt factor
    FM0P_EXTI->ENIR_f.EN3 = 1u;        // Enable INT03
    /* Enable NVIC */
    NVIC_ClearPendingIRQ(EXINT2_3_IRQn);
    NVIC_SetPriority(EXINT2_3_IRQn, 0);
    NVIC_EnableIRQ(EXINT2_3_IRQn);
    /* Initialize GPIO P0C for test*/
    FM0P_GPIO->PFR0_f.PC = 0;          // Enable P30 for GPIO function
    FM0P_GPIO->PDOR0_f.PC = 1;         // Set output high level
    FM0P_GPIO->DDR0_f.PC = 1;          // Set GPIO output

    while(1)
    {
        // write code here
    }
}
```

4.3 ISR 示例代码

4.3.1 通用设计

代码 5 显示的是通用设计的 ISR 源代码。在源代码中，INT2_3_IRQHandler 函数是主中断处理程序。EXINT2_IRQHandler 和 EXINT3_IRQHandler 函数是子中断处理程序。系统通过 MCU 内核分支到主中断处理程序，然后通过读取 IRQ17MON 寄存器的值来分支到子中断处理程序。

代码 5. 用于通用设计的 ISR 源代码

```
void EXINT3_IRQHandler(void)
{
    FM0P_GPIO->PDOR0_f.PC = 0;   //Test IO output low
    FM0P_EXTI->EICL_f.ECL3 = 0;  //Clear interrupt factor
}

void EXINT2_IRQHandler(void)
{
    FM0P_EXTI->EICL_f.ECL2 = 0;   //Clear interrupt factor
}

void INT2_3_IRQHandler(void)
{
    uint32_t u32IrqMon = FM0P_INTREQ->PDL_IRQMON_INT2_3; //Get register IRQ17MON
    /*External interrupt 2*/
    if (0x00000001u == (u32IrqMon & 0x00000001u))
```

```
    {
        EXINT2_IRQHandler();
    }
    /*External interrupt 3*/
    if (0x00000002u == (u32IrqMon & 0x00000002u))
    {
        EXINT3_IRQHandler();
    }
}
```

4.3.2 新设计

代码 6 显示的是用于新设计的 ISR 源代码。类似于通用代码，INT2_3_IRQHandler 函数是主中断处理程序，并且 EXINT2_IRQHandler 和 EXINT3_IRQHandler 函数是子中断处理程序。系统通过 MCU 内核分支到主中断处理程序，然后通过读取 VIR17 寄存器的值来分支到子中断处理程序。

代码 6. 用于新设计的 ISR 源代码

```
void EXINT3_IRQHandler(void)
{
    FM0P_GPIO->PDOR0_f.PC = 0;    //Test IO output low
    FM0P_EXTI->EICL_f.ECL3 = 0;   //Clear interrupt factor
}

void EXINT2_IRQHandler(void)
{
    FM0P_EXTI->EICL_f.ECL2 = 0;   //Clear interrupt factor
}

void INT2_3_IRQHandler(void)
{
    typedef void (*func_ptr_parg32_t)(void);
    func_ptr_parg32_t Irq_Entry;
    uint32_t u32IrqEntrance;
    u32IrqEntrance = *((uint32_t *)FM0P_IRQ_VIR->VIR17); //Get IRQ entrance in New
Design
    Irq_Entry = (func_ptr_parg32_t)u32IrqEntrance;
    Irq_Entry();              //Run to IRQ handler
}
```

5 性能比较

该章节对通用设计和新设计的应用性能进行了比较。该比较使用了 Keil 编译器。

5.1 代码空间

如表 4-1 所示，通用设计源代码和新设计源代码间在 ISR 代码上存在区别。从 Keil 编译器生成的映射文件选出了代码 7 和代码 8。在该映射文件中，INT2_3_IRQHandle 函数占用了通用设计应用的 42 个字节和新设计应用的 8 个字节。在当前示例中，这种区别表明新设计的 ISR 代码占用的空间更小。

但是，因为新设计应用的中断向量表需要更大的代码空间，而实际的代码空间则由具体情况决定，因此在代码空间中不能确定哪种应用的优势更大。

代码 7. 通用设计源代码的代码空间

Symbol Name	Value	Ov Type	Size	Object(Section)
EXINT3_IRQHand	0x00000375	Thumb Code	22	interrupts_fm0p.o(.text)
EXINT2_IRQHand	0x0000038b	Thumb Code	12	interrupts_fm0p.o(.text)
INT2_3_IRQHandle	0x00000397	Thumb Code	42	interrupts_fm0p.o(.text)

代码 8. 新设计源代码的代码空间

Symbol Name	Value	Ov Type	Size	Object(Section)
EXINT3_IRQHandle	0x00000375	Thumb Code	22	interrupts_fm0p.o(.text)
EXINT2_IRQHandle	0x0000038b	Thumb Code	12	interrupts_fm0p.o(.text)
INT2_3_IRQHandler	0x00000397	Thumb Code	8	interrupts_fm0p.o(.text)

5.2 执行速度

为了确定哪个应用具有更高的执行速度，需要将 GPIO 配置为测试流程。 在该测试流程中，相同初始状态的 GPIO 输出高电平。INT03_2 引脚上的下降沿输入信号触发了外部中断。进入外部中断 3 处理程序（即为子中断处理程序）后，GPIO 输出将处于低电平状态。然后在 INT03_2 上的下降沿和 GPIO 上的下降沿之间的这段时间内进行记录。

系统时钟为 40 MHz 时，通用设计应用的记录时间为 1.750 µs，如图 5-1 和图 5-2 所示。 GPIO 输出为低电平的时间约为 0.325 µs。除去 GPIO 输出为低电平的时间，通用设计应用的最终中断分支时间为 1.425 µs。同样，新设计应用的最终中断分支时间为 1.259 µs。

请参见表 5-1 中的“总结”部分。该部分中的信息确认，在执行速度方面上，新设计的应用占有绝对优势。

图 5-1. 通用设计应用的分支时间

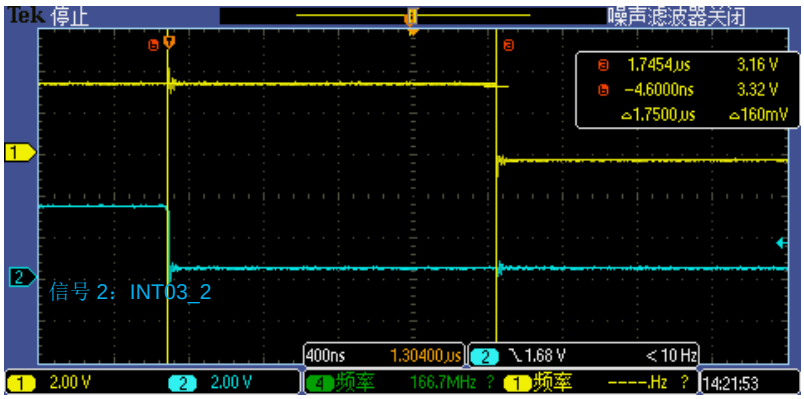


图 5-2. 新设计应用的分支时间

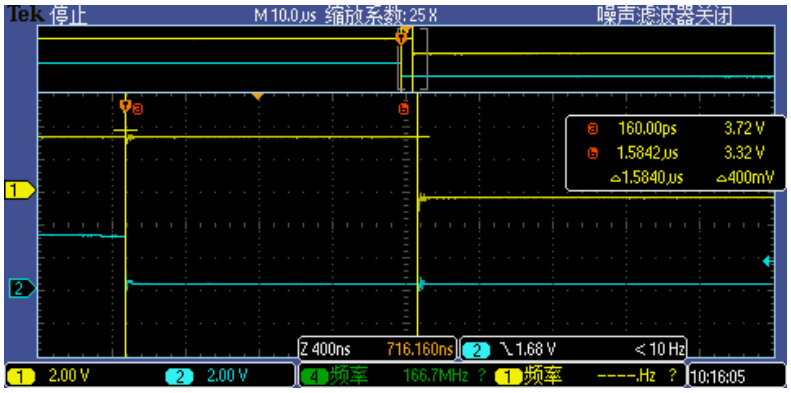


表 5-1. 执行速度总结

应用	记录时间	输出为低电平的时间	中断分支时间	总结
通用设计	1.750 µs	0.325 µs	1.425 µs	新设计应用能节省的时间为 0.166 µs，为 12%。
新设计	1.584 µs	0.325 µs	1.259 µs	

5.3 编码复杂性

如通用设计的代码 5 所示，中断处理程序的源代码将读取 IRQ17MON，并切换到相应的子函数。因此，您需要对每个中断处理程序编写相应的函数。

如新设计的代码 6 所示，所有中断处理程序的源代码都有相同的流程图。

由于具有完善的中断向量表，新设计应用在编码复杂性方面也占有绝对优势。

6 总结

本应用笔记对基于 NVIC 的通用设计和新设计中多个应用的性能进行了比较。同时也得出了结论，如果代码空间对应用程序不敏感，而基于新 NVIC 设计的中断模块便是一个好的选择。

7 使用注意事项

对于新设计应用，要确保中断向量表的正确性。否则，在系统执行过程中会引起严重错误。

A 附录 A：异常和中断

表 A-1 列出了被输入到 NVIC 的异常和中断。下面内容介绍的是表中各列的详细情况。

异常编号

指的是 NVIC 的异常编号。

IRQ 编号

指的是 NVIC 的外设中断编号（IRQ 编号 = 异常编号 16）。

向量偏移

该列显示的是发生中断时被参照的中断向量存储地址，是指 NVIC 中所描述的值 + 向量表偏移寄存器（VTOR）。

批量读取寄存器 – 名称

指的是批量读取寄存器的名称。该列中的“–”符号表示相应异常或中断没有对应的批量读取寄存器。

批量读取寄存器 – 位

该列指出分配给外设函数的每个异常和中断的批量读取寄存器位。该列中的“–”符号表示相应异常或中断没有对应的批量读取寄存器。

VIR 值

指的是发生中断时从 VIRxx 读取的值。读取值为所描述的值 + VIR_OFFSET。该列中的“–”符号表示没有 VIRxx。

异常或中断名称

该列显示了外设功能的异常和中断名称。

DSTC

指的是支持由 DSTC 控制的 DMA 传输。在该文档中未提到。

表 A-1. 异常和中断源（第 1/3 页）

Exception No.	IRQ No.	Vector Offset	Batch read register		VIR value	Exception or Interrupt name	DSTC
			Name	bit			
0	-	0x000	-	-	-	(Stack pointer initial value)	-
1	-	0x004	-	-	-	Reset	-
2	-	0x008	EXC02MON	0	-	Non-maskable interrupt (NMI)	-
				1	-	Hardware watchdog timer interrupt	-
3	-	0x00C	-	-	-	Reserved	-
4	-	0x010	-	-	-	Reserved	-
5	-	0x014	-	-	-	Reserved	-
6	-	0x018	-	-	-	Reserved	-
7	-	0x01C	-	-	-	Reserved	-
8	-	0x020	-	-	-	Reserved	-
9	-	0x024	-	-	-	Reserved	-
10	-	0x028	-	-	-	Reserved	-
11	-	0x02C	-	-	-	SVCall(supervisor call)	-
12	-	0x030	-	-	-	Reserved	-
13	-	0x034	-	-	-	Reserved	-
14	-	0x038	-	-	-	PendSV	-
15	-	0x03C	-	-	-	SysTick	-
16	0	0x040	IRQ00MON	0	0x0C0	Anomalous frequency detection interrupt by CSV	-
				1	0x140	Software watchdog timer interrupt	-
				2	0x1C0	Low-voltage detection (LVD) interrupt	-
17	1	0x044	IRQ01MON	0	0x0C4	MFS ch.0 reception interrupt	0
				1	0x144	MFS ch.0 transmission interrupt	1
				2	0x1C4	MFS ch.0 status interrupt	-
18	2	0x048	IRQ02MON	0	0x0C8	MFS ch.1 reception interrupt	2
				1	0x148	MFS ch.1 transmission interrupt	3
				2	0x1C8	MFS ch.1 status interrupt	-
19	3	0x04C	IRQ03MON	0	0x0CC	Reserved	-
				1	0x14C	Reserved	-
				2	0x1CC	Reserved	-
20	4	0x050	IRQ04MON	0	0x0D0	MFS ch.3 reception interrupt	6
				1	0x150	MFS ch.3 transmission interrupt	7
				2	0x1D0	MFS ch.3 status interrupt	-
21	5	0x054	IRQ05MON	0	0x0D4	MFS ch.4 reception interrupt	8
				1	0x154	MFS ch.4 transmission interrupt	9
				2	0x1D4	MFS ch.4 status interrupt	-
22	6	0x058	IRQ06MON	0	0x0D8	Reserved	-
				1	0x158	Reserved	-
				2	0x1D8	Reserved	-
23	7	0x05C	IRQ07MON	0	0x0DC	MFS ch.6 reception interrupt	12
						I2CSLAVE reception interrupt	48
				1	0x15C	MFS ch.6 transmission interrupt	13
						I2CSLAVE transmission interrupt	49
				2	0x1DC	MFS ch.6 status interrupt	-
						I2CSLAVE status interrupt	-

表 A-1. 异常和中断源（第 2/3 页）

Exception No.	IRQ No.	Vector Offset	Batch read register		VIR value	Exception or Interrupt name	DSTC
			Name	bit			
24	8	0x060	IRQ08MON	0	0x0E0	MFS ch.7 reception interrupt	14
				1	0x160	MFS ch.7 transmission interrupt	15
				2	0x1E0	MFS ch.7 status interrupt	-
25	9	0x064	IRQ09MON	0	0x0E4	A/D converter unit0 priority conversion interrupt	50
				1	0x164	A/D converter unit0 scan conversion interrupt	51
				2	0x1E4	A/D converter unit0 FIFO overrun interrupt	-
						A/D converter unit0 conversion result comparison int.	-
						A/D converter unit0 range comparison result int.	-
26	10	0x068	IRQ10MON	0	0x0E8	USB ch.0 device endpoint1 DRQ interrupt	52
				1	0x168	USB ch.0 device endpoint2 DRQ interrupt	53
				2	0x1E8	USB ch.0 device endpoint3 DRQ interrupt	54
27	11	0x06C	IRQ11MON	0	0x0EC	USB ch.0 device endpoint4 DRQ interrupt	55
				1	0x16C	USB ch.0 device endpoint5 DRQ interrupt	56
				2	0x1EC	USB ch.0 device endpoint0 DRQI interrupt	-
28	12	0x070	IRQ12MON	0	0x0F0	USB ch.0 device endpoint0 DRQO interrupt	-
				1	0x170	USB ch.0 device SUSP interrupt	-
						USB ch.0 device SOF interrupt	-
						USB ch.0 device BRST interrupt	-
						USB ch.0 device CONF interrupt	-
						USB ch.0 device WKUP interrupt	-
				2	0x1F0	USB ch.0 device SPK interrupt	-
29	13	0x074	IRQ13MON	0	0x0F4	USB ch.0 host DIRQ interrupt	-
						USB ch.0 host URIRQ interrupt	-
						USB ch.0 host RWKIRQ interrupt	-
						USB ch.0 host CNNIRQ interrupt	-
				1	0x174	USB ch.0 host SOFIRQ interrupt	-
						USB ch.0 host CMPIRQ interrupt	-
				2	0x1F4	Reserved	-
30	14	0x078	IRQ14MON	0	0x0F8	Main PLL oscillation stabilization wait completion int.	-
						Main clock oscillation stabilization wait completion int.	-
						Sub clock oscillation stabilization wait completion int.	-
				1	0x178	Reserved	-
				2	0x1F8	Reserved	-
31	15	0x07C	IRQ15MON	0	0x0FC	Watch counter interrupt	57
				1	0x17C	Real timer counter (RTC) interrupt	-
						Dual timer ch.1 interrupt	-
						Dual timer ch.2 interrupt	-
				2	0x1FC	Reserved	-

表 A-1. 异常和中断源（第 3/3 页）

Exception No.	IRQ No.	Vector Offset	Batch read register		VIR value	Exception or Interrupt name	DSTC
			Name	bit			
32	16	0x080	IRQ16MON	0	0x100	External pin interrupt ch.0	16
				1	0x180	External pin interrupt ch.1	17
33	17	0x084	IRQ17MON	0	0x104	External pin interrupt ch.2	18
				1	0x184	External pin interrupt ch.3	19
34	18	0x088	IRQ18MON	0	0x108	External pin interrupt ch.4	20
				1	0x188	External pin interrupt ch.5	21
35	19	0x08C	IRQ19MON	0	0x10C	External pin interrupt ch.6	22
				1	0x18C	External pin interrupt ch.7	23
36	20	0x090	IRQ20MON	0	0x110	External pin interrupt ch.8	24
				1	0x190	Reserved	-
37	21	0x094	IRQ21MON	0	0x114	Reserved	-
				1	0x194	Reserved	-
38	22	0x098	IRQ22MON	0	0x118	External pin interrupt ch.12	28
				1	0x198	External pin interrupt ch.13	29
39	23	0x09C	IRQ23MON	0	0x11C	Reserved	-
				1	0x19C	External pin interrupt ch.15	31
40	24	0x0A0	IRQ24MON	0	0x120	Base timer ch.0 source0 (IRQ0) interrupt	32
						Base timer ch.0 source1 (IRQ1) interrupt	33
				1	0x1A0	Base timer ch.4 source0 (IRQ0) interrupt	34
						Base timer ch.4 source1 (IRQ1) interrupt	35
41	25	0x0A4	IRQ25MON	0	0x124	Base timer ch.1 source0 (IRQ0) interrupt	36
						Base timer ch.1 source1 (IRQ1) interrupt	37
				1	0x1A4	Base timer ch.5 source0 (IRQ0) interrupt	38
						Base timer ch.5 source1 (IRQ1) interrupt	39
42	26	0x0A8	IRQ26MON	0	0x128	Base timer ch.2 source0 (IRQ0) interrupt	40
						Base timer ch.2 source1 (IRQ1) interrupt	41
				1	0x1A8	Base timer ch.6 source0 (IRQ0) interrupt	42
						Base timer ch.6 source1 (IRQ1) interrupt	43
43	27	0x0AC	IRQ27MON	0	0x12C	Base timer ch.3 source0 (IRQ0) interrupt	44
						Base timer ch.3 source1 (IRQ1) interrupt	45
				1	0x1AC	Base timer ch.7 source0 (IRQ0) interrupt	46
						Base timer ch.7 source1 (IRQ1) interrupt	47
44	28	0x0B0	IRQ28MON	0	0x130	CEC Reception/Remote reception ch.0 interrupt	-
						CEC Transmission ch.0 interrupt	-
				1	0x1B0	CEC Reception/Remote reception ch.1 interrupt	-
						CEC Transmission ch.1 interrupt	-
45	29	0x0B4	IRQ29MON	0	0x134	Smart Card ch.1 interrupt	-
				1	0x1B4	FLASH memory RDY/HANG interrupt	-
46	30	0x0B8	IRQ30MON	0	0x138	DSTC SW transfer complete interrupt	-
				1	0x1B8	DSTC error interrupt	-
47	31	0x0BC	IRQ31MON	0	0x13C	Reserved	-
				1	0x1BC	Reserved	-

文档修订记录

文档标题： AN99231 — 在 FM0+系列的 S6E1C3 器件中使用了中断

文档编号： 002-09819

版本	ECN	变更者	提交日期	变更说明
**	5026161	JCUI	11/30/2015	本文档版本号为 Rev**，译自英文版 001-99231 Rev**。

全球销售和设计支持

赛普拉斯公司具有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。要想查找离您最近的办事处，请访问赛普拉斯办公所在地。

产品

汽车级产品	cypress.com/go/automotive
时钟与缓冲区	cypress.com/go/clocks
接口	cypress.com/go/interface
照明与电源控制	cypress.com/go/powerpsoc
存储器	cypress.com/go/memory
PSoC	cypress.com/go/psoc
触摸感应	cypress.com/go/touch
USB 控制器	cypress.com/go/usb
无线/射频	cypress.com/go/wireless

PSoC®解决方案

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

赛普拉斯开发者社区

社区 | 论坛 | 博客 | 视频 | 培训

技术支持

cypress.com/go/support

PSoC 是赛普拉斯半导体公司的注册商标。此处引用的所有其他商标或注册商标归其各自所有者所有。

赛普拉斯半导体公司
198 Champion Court
San Jose, CA 95134-1709

电话 ：408-943-2600
传真 ：408-943-4730
网址 ：www.cypress.com

©赛普拉斯半导体公司，2015。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品内嵌的电路外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不会根据专利或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯不保证产品能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。
该源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途外，未经赛普拉斯明确的书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。
免责声明：赛普拉斯不针对此材料提供任何类型的明示或暗示保证，包括（但不限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于可能发生运转异常和故障，并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品使用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。
产品使用可能受限于赛普拉斯软件许可协议。