AN99231

# FM0+ファミリの S6E1C3 シリーズにおける割込みの使用について

**著者: Mily Wang**

**関連製品ファミリ: FM0+**

**関連サンプル コード: なし**

**関連アプリケーション ノート: AN54460、AN90799**

**このアプリケーション ノートの最新版または関連プロジェクト ファイルについては、http://www.cypress.com/go/AN99231 にアクセスしてください。**

本アプリケーション ノートは、FM0+ ファミリにおける ARM Cortex-M0+ MCU のネスト型ベクタ割込みコントローラー (NVIC) 機能の基本的な概念についてご説明します。また、新しく設計された S6E1C3 シリーズの NVIC の動作と利点についてご紹介します。

## 目次

1 はじめに ..... 2
2 概要 ..... 2
2.1 NVIC アーキテクチャ ..... 2
2.2 NVIC 機能 ..... 2
3 S6E1C3 シリーズの NVIC デザイン ..... 3
3.1 一般的なデザイン ..... 4
3.2 新しいデザイン ..... 4
3.3 割込みベクタ テーブルと割込みリスト ..... 5
4 サンプル コード ..... 6
4.1 割込みベクタ テーブルのサンプル コード ..... 7
4.2 メイン ルーチンのサンプル コード ..... 9
4.3 ISR サンプル コード ..... 10
5 パフォーマンス比較 ..... 12
5.1 コード空間 ..... 12
5.2 実行速度 ..... 12
5.3 コーディングの複雑さ ..... 13
6 まとめ ..... 13
7 使用上の注意事項 ..... 13
A 付録 A: 例外と割込み要因 ..... 14
改訂履歴 ..... 18
ワールドワイド販売と設計サポート ..... 19
製品 ..... 19
PSoC®ソリューション ..... 19
サイプレス開発者コミュニティ ..... 19
テクニカルサポート ..... 19

# 1 はじめに

S6E1C3 シリーズ デバイスは、ARM Cortex-M0+プロセッサを内蔵する高集積 32 ビット マイクロコントローラーです。S6E1C3 シリーズは、システムをより効率的に利用できる新しい NVIC デザインを搭載しています。Cortex-M0+プロセッサは、いくつかの割込みとシステム例外をサポートします。これらはひとつに集約され、NVIC によって処理されます。

本アプリケーションノートでは、FM0+ファミリの NVIC 機能の基本コンセプト、S6E1C3 シリーズにおける新しい NVIC、およびこの新しく設計された NVIC 機能の動作をご紹介します。本書と共に、アプリケーション ノートのサンプル プロジェクトも提供します。

# 2 概要

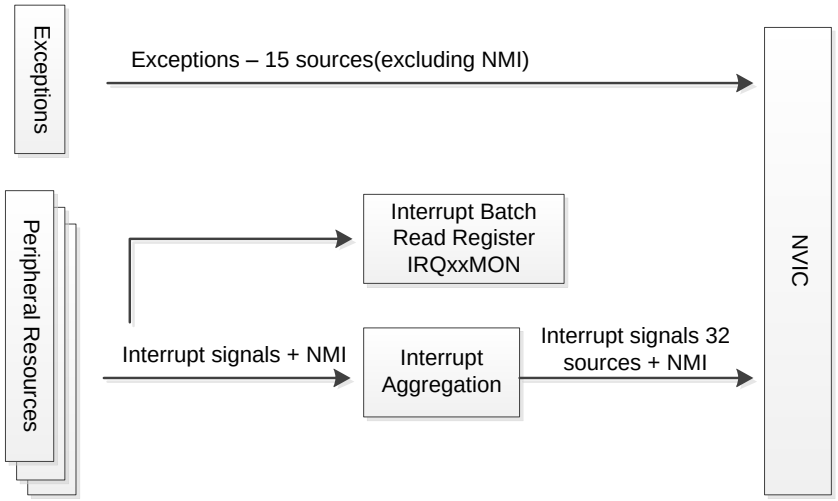
Cortex-M0+ CPU コアは、コア内部に NVIC を搭載しています。NVIC は割込み要求の優先度を判定し、要求を CPU へ送信します。

## 2.1 NVIC アーキテクチャ

図 2-1 に FM0+の割込み機能のアーキテクチャを示します。例外は、リセット、SysTick、および HardFault などのシステム要求です。これらの要求は NVIC モジュールへの入力で、例外番号と一致します。詳細については、Cortex-M0+のテクニカル リファレンス マニュアルをご覧ください。

ペリフェラルからのいくつかの割込み信号および例外は集約され、対応する割込みベクタに入力されます。割込みベクタの総数は、ノンマスカブル割込み (NMI) を除いて 32 個です。「IRQxxMON」と呼ばれる IRQ 一括読出しレジスタは、システム内でどの割込みが発生したかを示します。

図 2-1. FM0+ 割込みアーキテクチャ




## 2.2 NVIC 機能

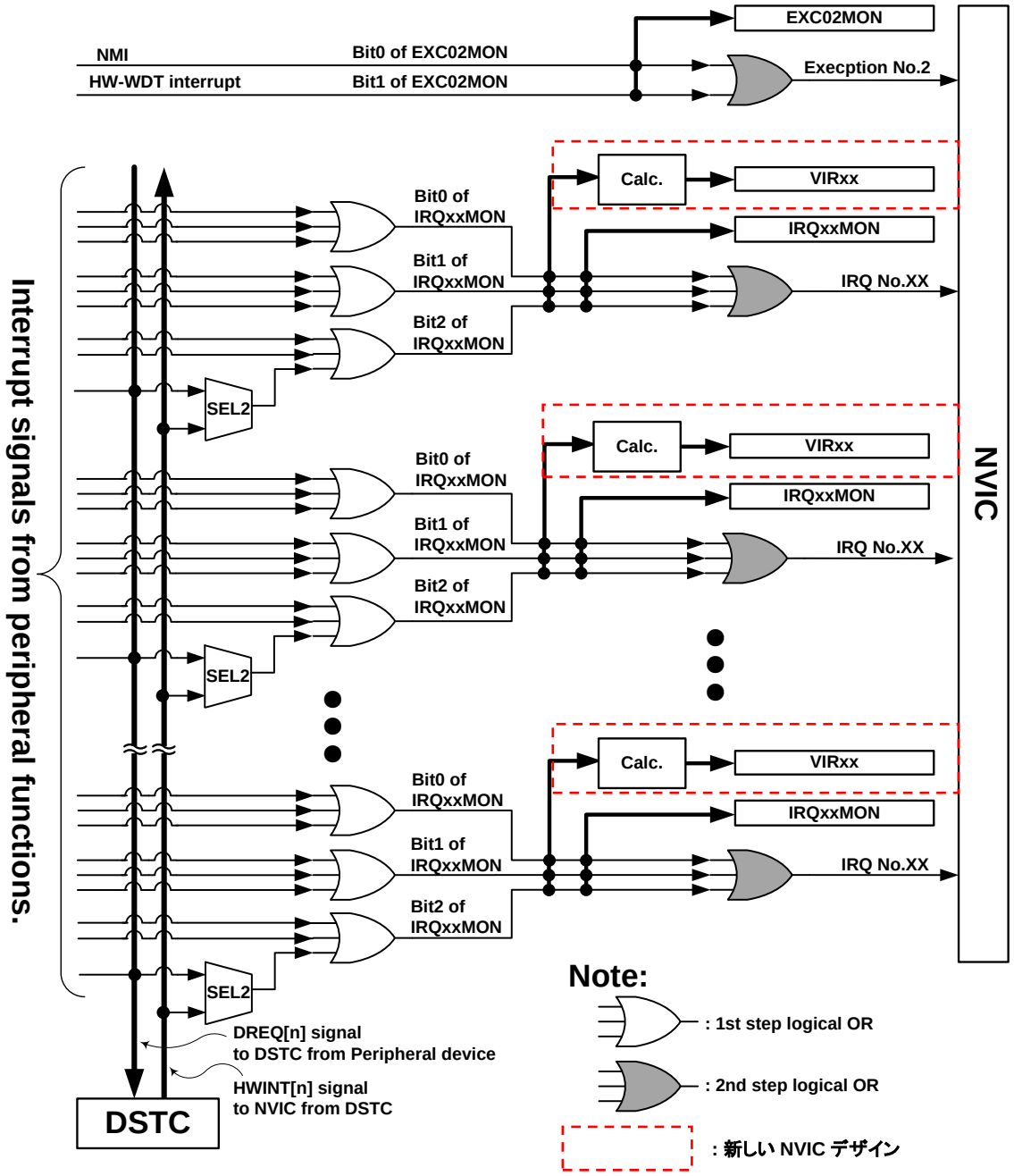
FM0+ファミリの NVIC には次の機能があります。

- 32 個のマスク可能なペリフェラル割込みチャネル (Cortex-M0+の 16 個の例外割込みは含みません)
- プログラム可能な 4 つの割込み優先レベル (2 ビットの優先割込みを使用)
- NMI 入力対応
- レイテンシの短い例外と割込み処理に対応
- システム制御レジスタの実装

# 3 S6E1C3 シリーズの NVIC デザイン

図 3-1 にペリフェラル割込み信号と NVIC モジュール間の接続を説明します。本節では、NVIC の一般的なデザインと S6E1C3 シリーズの新しいデザインを紹介します。

図 3-1. 割込み信号と NVIC の接続




DSTC を使用した DMA 転送が選択された場合、ペリフェラルからの割込みに代わって DSTC から転送完了割込みが発生します。各ペリフェラル機能からの DSTC 転送要求に関する詳細については、ペリフェラル マニュアルの各ペリフェラル機能の章を参照して下さい。

新しい NVIC デザインは赤い点線で囲まれた部分です。図 3-1 の残りの部分は NVIC の一般的なデザインを示します。

## 3.1 一般的なデザイン

### 3.1.1 割込みベクタ テーブル

割込みベクタ テーブルを表 3-1 に示します。アドレス 0x00000000～0x000000BC が一般的なデザインのために割当てられています。

### 3.1.2 NMI とハードウェア ウォッチドッグ割込み

外部割込みからの NMI 信号と、NMI コントローラおよびハードウェア (HW) ウォッチドッグ タイマからのハードウェア ウォッチドッグ割込みは、2 段目の論理 OR により集約され、NVIC の例外 No.2 に接続されています。例外 No.2 の割込みが発生した場合、割込み要因が NMI か HW ウォッチドッグのどちらかは、EXC02 一括読出しレジスタ (図 3-1 の EXC02MON) を読出すことで特定できます。EXC02MON の詳細についてはペリフェラル マニュアルを参照してください。

### 3.1.3 その他のペリフェラル割込み

ペリフェラル機能からの割込み信号は 2 段の OR 回路によって集約されます (図 3-1)。集約された割込み信号は NVIC の 32 個のペリフェラル割込みの 1 つに接続されます。それぞれのペリフェラル機能の割込み信号が NVIC のどの IRQ 入力に割当てられているかは、表 A-1 を参照してください。

割込みが発生した場合、一括読出しレジスタ (図 3-1 の IRQxxMON) を読出すことで、割込要因を特定できます。IRQxxMON レジスタは、すべての NVIC の割込み入力を網羅しています。32 個のレジスタ (IRQ00MON～IRQ31MON) をサポートしています。IRQxxMON の詳細については、ペリフェラル マニュアルを参照してください。

### 3.1.4 IRQ 一括読出しレジスタ

IRQxxMON レジスタはどの割込みが発生しているかを示します。IRQxxMON レジスタとペリフェラル割込みの関係については表 A-1 を参照してください。

## 3.2 新しいデザイン

新しいデザインは一般的なデザインに基づいて実装されています。この手法により、幾つかの特別なデザインを、便利で効率的に追加しています。

### 3.2.1 割込みベクタ テーブル

割込みベクタ テーブルを表 3-1 に示します。アドレス 0x000000C0～0x000001FC は新しいデザインのために追加され、赤色の枠で囲まれています。

### 3.2.2 ベクタ指示レジスタ

割込み発生時、CPU はベクタ指示レジスタ (VIRxx) を利用して割込み分岐動作を高速に開始できます。VIRxx レジスタは NVIC の IRQ00～IRQ31 入力に対応し、32 個のレジスタ (VIR00～VIR31) です。割込みが発生すると、CPU は決められたアドレスを VIRxx から読出します。割込みハンドラの先頭アドレスをこのアドレス領域に配置することで、割込み動作を高速に分岐できます。以下に IRQ00 発生時の VIR00 の動作と使用方法について説明します。

表 3-1 に示すように、プログラムは IRQ 割込みハンドラの先頭アドレスを 0x0000 0040～0x0000 00BC のアドレス領域に配置します。同様に、ビット 0、1、2 に対応した IRQ 割込みハンドラの先頭アドレスをアドレス領域 0x0000 00C0～0x0000 01FC に配置します。

IRQ00 の割込み発生時、IRQ00 の割込みハンドラが開始され、CPU は VIR00 から値を読出します。VIR00 からの読出し値を表 3-2 に示します。ビット 0 割込みが発生すると、VIR00 の読出し値は 0x0000 00C0 です。ビット 1 割込みが発生すると、VIR00 の読出し値は 0x0000 0140 です。また、ビット 2 割込みが発生すると、VIR00 の読出し値は 0x0000 01C0 になります。IRQ00 割込みハンドラの分岐動作は、割込みアドレスを VIR00 から読出したアドレスに配置します。この方法により、割込みハンドラの分岐動作を高速に実行します。

### 3.2.3 VIR オフセット レジスタ

VIR オフセット レジスタ (VIR_OFFSET) は VIRxx の共通オフセット値の設定に使用します。

## 3.3 割込みベクタ テーブルと割込みリスト

S6E1C3 の割込みベクタ テーブルを表 3-1 に示します。一般的なデザインを選択した場合、割込みベクタ テーブルのアドレスは 0x00000000～0x000000BC の範囲になります。新しいデザインを選択した場合、割込みベクタ テーブルのアドレスは 0x00000000～0x000001FC の範囲になります。0x000000C0～0x000001FC のベクタ アドレスは新しいデザインのために追加され、表 3-1 に示すように赤色の枠で囲まれています。

表 3-1. 割込みベクタ テーブル

| アドレス | データ |
|---|---|
| 0x0000 0000 | スタック ポインタの初期値 |
| 0x0000 0004 | 例外 1: リセット ベクタ |
| 0x0000 0008 | 例外 2: NMI/HW-WDT ハンドラのトップ アドレス |
| 0x0000 000C | 例外 3: Hard fault ハンドラのトップ アドレス |
| 0x0000 0010 ～ 0x0000 0028 | 予約 |
| 0x0000 002C | 例外 12: SoCal ハンドラのトップ アドレス |
| 0x0000 0030 ～ 0x0000 0034 | 予約 |
| 0x0000 0038 | 例外 14: PendSV ハンドラのトップ アドレス |
| 0x0000 003C | 例外 15: SysTick ハンドラのトップ アドレス |
| 0x0000 0040 | IRQ00 ハンドラのトップ アドレス |
| 0x0000 0044 | IRQ01 ハンドラのトップ アドレス |
| .... | .... |
| 0x0000 00B8 | IRQ30 ハンドラのトップ アドレス |
| 0x0000 00BC | IRQ31 ハンドラのトップ アドレス |
| 0x0000 00C0 | IRQ00 – ビット 0 ハンドラのトップ アドレス |
| 0x0000 00C4 | IRQ01 –ビット 0 ハンドラのトップ アドレス |
| .... | .... |
| 0x0000 0138 | IRQ30 –ビット 0 ハンドラのトップ アドレス |
| 0x0000 013C | IRQ31 –ビット 0 ハンドラのトップ アドレス |
| 0x0000 0140 | IRQ00 –ビット 1 ハンドラのトップ アドレス |
| 0x0000 0144 | IRQ01 –ビット 1 ハンドラのトップ アドレス |
| .... | .... |
| 0x0000 01B8 | IRQ30 –ビット 1 ハンドラのトップ アドレス |
| 0x0000 01BC | IRQ31 –ビット 1 ハンドラのトップ アドレス |
| 0x0000 01C0 | IRQ00 –ビット 2 ハンドラのトップ アドレス |
| 0x0000 01C4 | IRQ01 –ビット 2 ハンドラのトップ アドレス |
| .... | .... |
| 0x0000 01F8 | IRQ14 – ビット 2 ハンドラのトップ アドレス |
| 0x0000 01FC | IRQ15 – ビット 2 ハンドラのトップ アドレス |

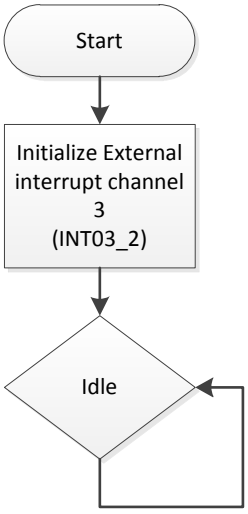
表 3-2. IRQ00 のステータスと VIR00 の読出し値

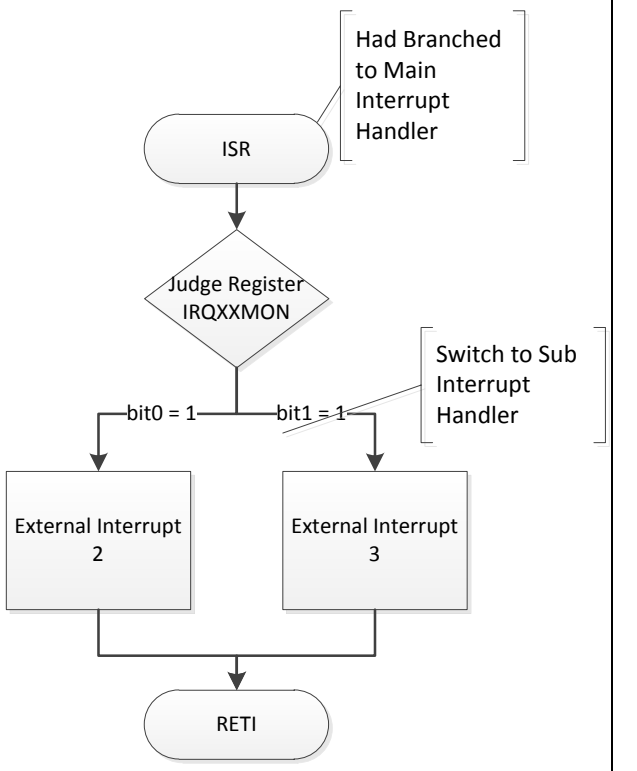
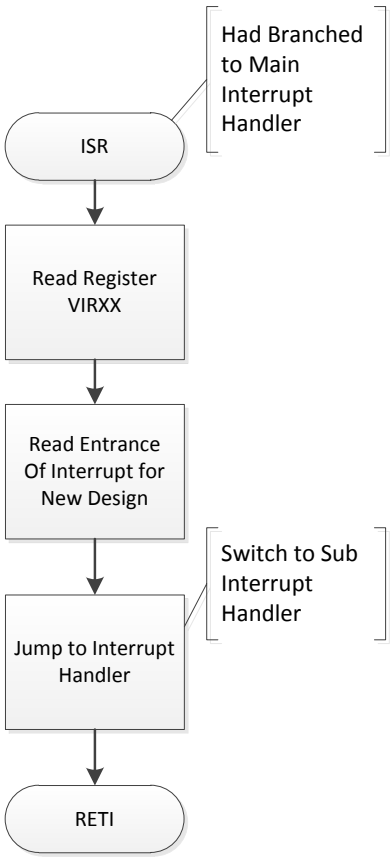
| IRQ00 入力信号ステータス | | | VIR00 読出し値 |
|---|---|---|---|
| ビット 0 割込みステータス | ビット 1 割込みステータス | ビット 2 割込みステータス | |
| 1 | 影響なし | 影響なし | 0x0000 00C0 + VIR_OFFSET |
| 0 | 1 | 影響なし | 0x0000 0140 + VIR_OFFSET |
| 0 | 0 | 1 | 0x0000 01C0 + VIR_OFFSET |
| 0 | 0 | 0 | 未定義値 |

# 4 サンプル コード

サンプル コードには、割込みベクタ テーブル コード、メイン ルーチン コード、および割込みサービス ルーチン (ISR) コードがあります。本節では、外部割込みチャネル 3 を使用した適用方法を説明します。端子 INT03_2 は外部割込み機能が選択されています。新しいデザインと一般的なデザインのソース コードは同じメイン ルーチンのコードを含みます。両方の割込みベクタ テーブルはほぼ同じですが、ISR コード用のフローが異なります。詳細は、表 4-1 を参照してください。

表 4-1. 一般的なデザインと新しいデザインの適用方法

| コード | 一般的なデザイン | 新しいデザイン |
|---|---|---|
| 割込みベクタ テーブル | ソース コードはコード 1 とコード 2 に表示 | 割込みの入り口は割込みベクタ テーブルに従う。ソース コードはコード 1 とコード 2 に表示 |
| メイン ルーチン |  ソース コードはコード 4 に示す。 | |

| コード | 一般的なデザイン | 新しいデザイン |
|---|---|---|
| ISR |  ソース コードはコード 5 に示す。 |  ソース コードはコード 6 に示す。 |

ISR コード用のフローは割込みハンドラへの分岐が 2 回必要です。第 1 の割込みハンドラ (本書ではメイン割込みハンドラと呼ばれる) はコアの割込みコントローラーによってハンドラに入ります。第 2 の割込みハンドラ (本書ではサブ割込みハンドラと呼ばれる) はソフトウェアによってハンドラに入ります。ISR フローの違いは、サブ割込みハンドラの分岐処理方法にあります。表 4-1 の「ISR」を参照してください。

## 4.1 割込みベクタ テーブルのサンプル コード

割込みベクタ テーブルのソース コードは Keil コンパイラと IAR コンパイラで異なります。Keil コンパイラ用の割込みベクタ テーブルのソース コードはコード 1 に示されます。ソース コードは、例外用の 16 個のベクタと、ペリフェラル割込み用の 32 個のベクタを含んでいます。新しいデザイン用の割込みハンドラは、割込みベクタ テーブルに続きます。

コード 1. Keil コンパイラ用の割込みベクタ テーブルのソース コード (一般的なデザインと新しいデザインのアプリケーションに適用可能)

```
                ;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
```

IAR コンパイラ用の割込みベクタ テーブルのソース コードはコード 2 に示されます。2 つの新しいセグメントが外部割込みハンドラの位置決めのために作成されます。同時に、ICF ファイルでは、IAR プロジェクトに使用する追加セグメントの配置を定義することも必要です。詳しくはコード 2 とコード 3 を参照してください。

コード 2. IAR コンパイラ用の割込みベクタ テーブルのソース コード (一般的なデザインと新しいデザインのアプリケーションに適用可能)

```
                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
```

コード 3. ICF ファイル内の配置定義

```
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 メイン ルーチンのサンプル コード

コード 4 にメイン ルーチンのソース コードを示します。

コード 4. メイン ルーチンのソース コード (一般的なデザインと新しいデザインのアプリケーションに適用可能)

```
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 サンプル コード

### 4.3.1 一般的なデザイン

一般的なデザインの ISR ソース コードはコード 5 に示します。ソース コード内の INT2_3_IRQHandler 関数はメイン割込みハンドラです。EXINT2_IRQHandler と EXINT3_IRQHandler 関数はサブ割込みハンドラです。システムは MCU コアによってメインの割込みハンドラに分岐した後、IRQ17MON レジスタの値を読出すことでサブ割込みハンドラに分岐します。

コード 5. 一般的なデザイン向けの ISR ソース コード

```
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 新しいデザイン

新しいデザインの ISR ソース コードはコード 6 に示しています。一般的なデザインと同様に、INT2_3_IRQHandler 関数はメイン割込みハンドラで、EXINT2_IRQHandler と EXINT3_IRQHandler 関数はサブ割込みハンドラです。システムは MCU コアによってメイン割込みハンドラに分岐した後、VIR17 レジスタの値を読出すことでサブ割込みハンドラに分岐します。

コード 6. 新しいデザイン向けの ISR ソース コード

```
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 パフォーマンス比較

本節では、一般的なデザインと新しいデザインのアプリケーションのパフォーマンスを比較します。Keil コンパイラがこの比較のために使用されます。

## 5.1 コード空間

表 4-1 に示すように、一般的なデザインと新しいデザイン間のソース コードの違いは ISR コードにあります。コード 7 とコード 8 は、Keil コンパイラで生成したマップファイルから選択されます。マップ ファイルには、INT2_3_IRQHandle 関数は一般的なデザイン アプリケーションのために 42 バイトを、新しいデザイン アプリケーションのために 8 バイトを要します。この違いから、この例で新しいデザインの ISR コードがより小さなコード空間を占めることが分かります。

ただし、新しいデザイン アプリケーションの割込みベクタ テーブルが多くのコード空間を必要とし、さらに実際のコード空間は特定の状況に左右されるため、コード空間の面ではどのアプリケーションが絶対的優位性を有するかを判定できません。

コード 7. 一般的なデザインのソース コードのコード空間

```
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)
```

コード 8. 新しいデザインのソース コードのコード空間

```
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 実行速度

どのアプリケーションがより速い実行速度を確定するには、1 本の GPIO をテスト フローのために設定します。テスト フローでは、システム初期化時にはその GPIO が出力 HIGH です。INT03_2 ピンの立下がりエッジの入力で、外部割込みが発生します。外部割込み 3 ハンドラ (サブ割込みハンドラ) に移行した後、GPIO は出力 LOW になります。INT03_2 の立下がりエッジと GPIO の立下がりエッジ間が記録される時間です。

図 5-1 と図 5-2 に示すように、システム クロックが 40MHz の場合、一般的なデザインの記録時間は 1.750µs です。GPIO 上の出力 LOW の実行時間は約 0.325µs です。GPIO 上の出力 LOW の実行時間を除くと、一般的なデザインのアプリケーションの最終的な割り込み分岐時間は 1.425µs になります。同様に、新しいデザインのアプリケーションの最終的な割込み分岐時間は 1.259µs です。

表 5-1 の「まとめ」をご覧ください。実行速度の面では新しいデザインのアプリケーションが絶対的優位性を有することを確認できます。

図 5-1. 一般的なデザイン アプリケーションの分岐時間

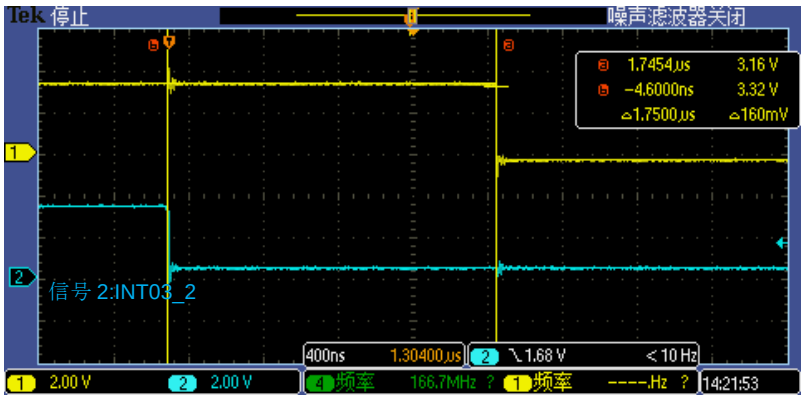



図 5-2. 新しいデザイン アプリケーションの分岐時間

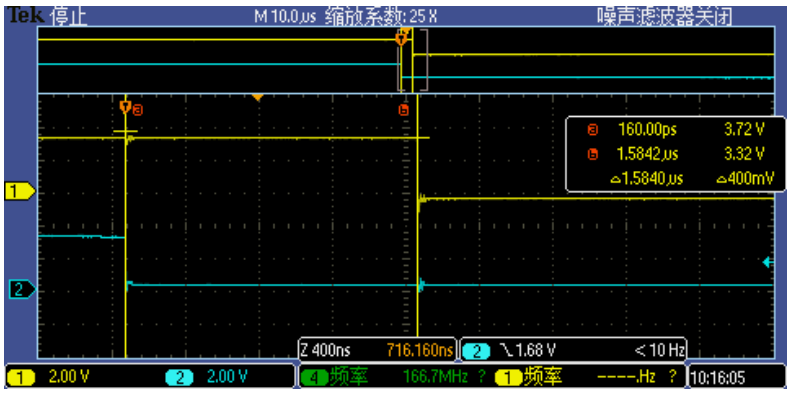



表 5-1. 実行速度のまとめ

| アプリケーション | 記録時間 | LOW 出力のための時間 | 割込み分岐時間 | まとめ |
|---|---|---|---|---|
| 一般的なデザイン | 1.750µs | 0.325µs | 1.425µs | 新しいデザイン アプリケーションで節約できる時間は約 12%で 0.166µs である |
| 新しいデザイン | 1.584µs | 0.325µs | 1.259µs | |

### 5.3 コーディングの複雑さ

一般的なデザインであるコード 5 に示すように、割込みハンドラのソース コードは IRQ17MON から読出し、対応するサブ関数に切替えます。よって、各割込みハンドラ毎に異なる関数をコーディングする必要があります。

一方、新しいデザインであるコード 6 に示すように、あらゆる割込みハンドラのソース コードは同じフローチャートを利用します。

割込みベクタ テーブルの実装を含め、コーディングの複雑さの面においては新しいデザインのアプリケーションが絶対的優位性を有します。

## 6 まとめ

本アプリケーション ノートは、NVIC の一般的なデザインと新しいデザインに基づいてアプリケーションの性能を比較しました。結論として、アプリケーションに対してコード サイズが問題でない場合は、新しい NVIC デザインに基づいた割込みモジュールの方が優れていることがわかります。

## 7 使用上の注意事項

新規設計のアプリケーションに対して、割込みベクタ テーブルが正しいことを確認してください。さもなければ、システム上の重大な実行エラーの原因となります。

# A 付録 A: 例外と割込み要因

表 A-1 には、NVIC に入力する例外と割込み一覧を示します。表内の各項目について、以下に説明します。

**例外番号**

NVIC の例外番号です。

**IRQ 番号**

NVIC のペリフェラル割込み番号です (IRQ 番号 = 例外番号 - 16)。

**ベクタ オフセット**

割込み発生時に参照される割込みベクタの格納アドレスを示します。参照対象は説明される値 + NVIC のベクタ テーブル オフセット レジスタ (VTOR) です。

**一括読出しレジスター名前**

一括読出しレジスタの名前を示します。この列の「–」は例外や割り込みに対応するバッチ読み出しレジスタがないことを意味します。

**一括読出しレジスタービット**

一括読出しレジスタのどのビットが、ペリフェラル機能からの割込み要因や例外要因に割当てられているかを示します。この列の「–」は例外や割込みに対応する一括読出しレジスタがないことを意味します。

**VIR 値**

割込みが発生する時に VIRxx から読出す値を示します。表記値 + VIR_OFFSET が読出されます。この列の「–」は VIRxx がないことを意味します。

**例外または割り込みの名前**

例外やペリフェラル機能からの割込み要因名を示します。

**DSTC**

DSTC による DMA 転送がサポートされることを示します。本書では詳細についてはスキップします。

表 A-1. 例外および割込み要因 (1／3)

| 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. 例外および割込み要因 (2／3)

| 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. 例外および割込み要因 (3／3)

| 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 | - |

# 改訂履歴

文書名: AN99231– FM0+ファミリの S6E1C3 シリーズにおける割込みの使用について

文書番号: 002-09820

| 版 | ECN | 変更者 | 発行日 | 変更内容 |
|---|---|---|---|---|
| ** | 5026163 | EIHA | 11/25/2015 | これは英語版 001-99231 Rev. **を翻訳した日本語版 002-09820 Rev. **です。 |

## ワールドワイド販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店および販売代理店の世界的なネットワークを持っています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

### 製品

| | |
|---|---|
| 車載用 | cypress.com/go/automotive |
| クロック & バッファ | cypress.com/go/clocks |
| インターフェース | cypress.com/go/interface |
| 照明 & 電源管理 | cypress.com/go/powerpsoc |
| メモリ | cypress.com/go/memory |
| PSoC | cypress.com/go/psoc |
| タッチ センシング | cypress.com/go/touch |
| USB コントローラー | cypress.com/go/usb |
| ワイヤレス／RF | cypress.com/go/wireless |

### PSoC®ソリューション

psoc.cypress.com/solutions

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

### サイプレス開発者コミュニティ

コミュニティ | フォーラム | ブログ | ビデオ | トレーニング

### テクニカル サポート

cypress.com/go/support

PSoC はサイプレス セミコンダクタ社の登録商標です。本書で言及するその他すべての商標または登録商標は、それぞれの所有者に帰属します。

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2015. 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporation (サイプレス セミコンダクタ社) は、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対して一切の責任を負いません。サイプレス セミコンダクタ社は、特許またはその他の権利に基づくライセンスを譲渡することも、または含意することもありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、または安全の用途のために使用することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。
このソース コード (ソフトウェアおよび／またはファームウェア) はサイプレス セミコンダクタ社 (以下「サイプレス」) が所有し、全世界の特許権保護 (米国およびその他の国)、米国の著作権法ならびに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によりライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであり、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタム ソフトウェアおよび／またはカスタム ファームウェアを作成する目的に限って、サイプレスのソース コードの派生著作物をコピー、使用、変更そして作成するためのライセンス、ならびにサイプレスのソース コードおよび派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソース コードを複製、変更、変換、コンパイル、または表示することはすべて禁止します。
免責条項: サイプレスは、明示的または黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性または特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品または回路を適用または使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤作動や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。
ソフトウェアの使用は、適用されるサイプレス ソフトウェア ライセンス契約によって制限され、かつ制約される場合があります。