中断
1.70

特性

- 定义硬件触发的中断
- 提供了一种软件 API 来挂起中断

概述

中断组件定义了硬件触发的中断。它是中断设计范围资源系统中不可分割的一部分（请参见 PSoC Creator 帮助中的“设计范围资源”一节）。

中断控制器可处理三种类型的系统中断波形：

- **电平 —** 在固件通过某个操作（例如，读后清除）清除请求源之前，IRQ 资源保持粘滞和活跃状态。大多数固定功能外设具有电平敏感中断，包括 UDB FIFO 和状态寄存器。
- **脉冲 —** 理想情况下，一个脉冲型 IRQ 是一个总线时钟，它记录待处理操作并确保 ISR 操作只执行一次。不需要任何对外设进行的固件操作。
- **边沿 —** 任意同步的波形是边沿检测电路的输入，该波形的上升沿变为同步的单周期脉冲（脉冲模式）。

注意：以上中断波形的类型不同于 **Configure**（配置）对话框中 **InterruptType** 参数的设置。该参数仅配置多路复用器选择线。根据多路复用器选择（电平、边沿），对“IRQ”信号进行处理以发送至中断控制器。

也就是说，无论 **InterruptType** 多路复用器的选择如何，中断控制器都能够处理电平、边沿或者脉冲波形。有关详细信息，请参见相关 TRM 文档。

何时使用中断组件

需要一个硬件触发的中断时，可使用中断组件。中断是不可或缺的，因为与轮询相比较，它们能利用硬件来支持降低延迟和事件检测的开销。

输入/输出接口

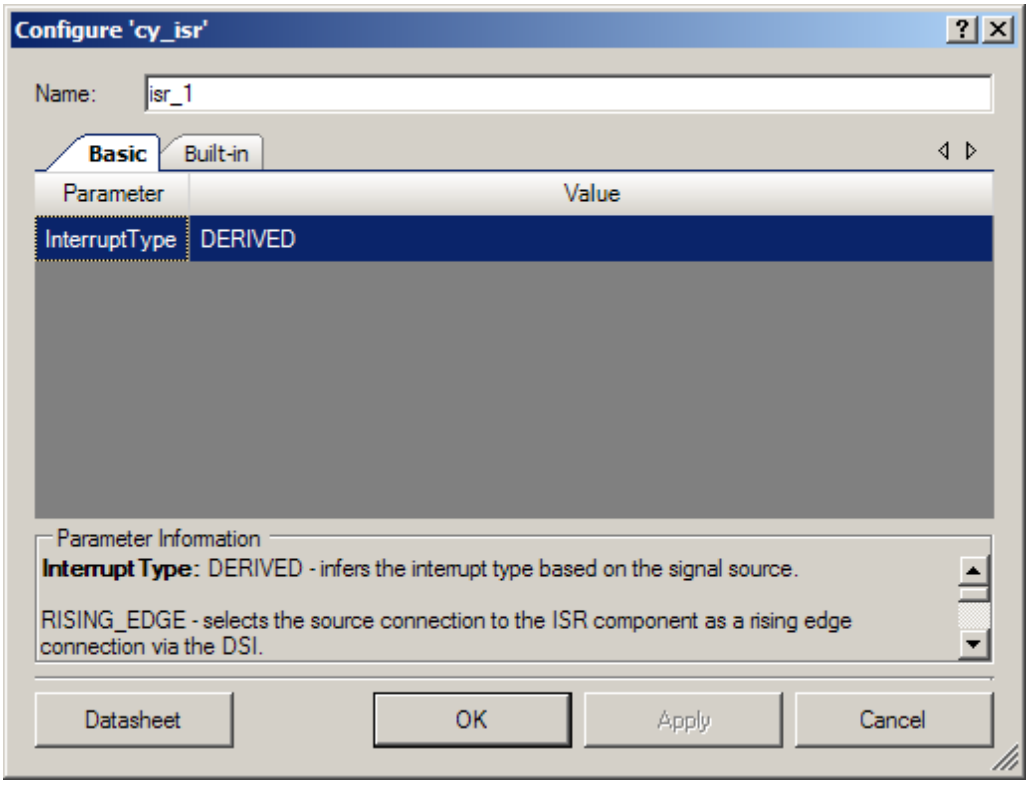
本节介绍中断组件的各种输入和输出接口。

int_signal — 输入

将产生中断的信号连接至该输入。当信号值变为逻辑高时，会触发中断。对于电平类中断，只要信号仍保持为逻辑高电平，则该中断将被持续触发。

组件参数

将一个中断组件拖入设计窗口内，并双击以打开 **Configure**（配置）对话框。



中断组件提供了以下参数：

InterruptType（中断类型）

该参数用于配置组件需要处理以触发中断的波形类型。该参数具有以下三个可能的值：

- **RISING_EDGE**（上升沿） — 在源信号的上升沿上触发中断。如果选择了该选项，“int_signal”输入的上升沿将转换为周期“bus_clk”的脉冲并发送至中断控制器。在 PSoC 5LP 上，如果中断组件连接至唤醒源（用以将器件从睡眠或休眠低功率模式唤醒），则可能无法使用 RISING_EDGE。
- **LEVEL**（电平） — 选择通过 DSI 以电平敏感类型连接至中断的源。如果选择了该选项，“int_signal”输入将直接传递至中断控制器。有关详细信息，请参见概述一节。
- **DERIVED**（派生） — 这是默认设置。中断组件连接到固定功能块（I^2C、USB、CAN 等）时，它会检查“int_signal”的驱动，然后根据所连接的对象派生出相应的中断类型。此自动分配是根据器件的数据手册中的信息进行的。

连接到固定功能中断输出时，类型应设置为 DERIVED。对于其他中断源，通常应选择 RISING_EDGE 来捕捉某个事件（例如周期时钟），并选择 LEVEL 来捕捉某个状态（例如 FIFO 填充电平）。对于 DMA NRQ 信号，任何设置都产生和各 NRQ 事件的单个中断相同的结果。

应用编程接口（API）

通过应用编程接口（API）子程序，您可以使用软件对组件进行配置。下面的表格列出并说明了每个函数的接口。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“isr_1”分配给指定设计中组件的第一个实例。您可以将其重新命名为遵循标识符语法规则的任何唯一值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。出于可读性考虑，下表中使用的实例名称为“ISR.”

函数	说明
ISR_Start()	设置函数的中断。
ISR_StartEx()	设置函数的中断并将地址设为中断的ISR向量。
ISR_Stop()	禁用并删除中断。
ISR_Interrupt()	ISR的默认中断处理程序。
ISR_SetVector()	将地址设为中断的新ISR向量。
ISR_GetVector()	获得中断当前ISR向量的地址。
ISR_SetPriority()	设置中断的优先级。
ISR_GetPriority()	获得中断的优先级。
ISR_Enable()	使能中断控制器的中断。

函数	说明
ISR_GetState()	获得中断的状态（使能、禁用）。
ISR_Disable()	禁用中断。
ISR_SetPending()	使中断进入待处理状态，这是一种生成中断的软件方法。
ISR_ClearPending()	清除待处理的中断。

void ISR_Start(void)

说明： 设置并使能中断。该函数会禁用中断，设置默认中断向量，根据设计范围资源中断编辑器中的值设置优先级，然后使能中断控制器的中断。

参数： void

返回值： void

其他影响： 无

void ISR_StartEx(cyisraddress address)

说明： 设置并使能中断。该函数会禁用中断，根据传入的地址设置中断向量，根据设计范围资源中断编辑器中的值来设置优先级，然后使能中断控制器的中断。

定义ISR函数时，可通过使用CY_ISR和CY_ISR_PROTO宏来提供跨编译器的一致性定义：

函数定义示例：

```
CY_ISR(MyISR)
{
        /* ISR Code here
}
```

函数原型示例：

```
CY_ISR_PROTO(MyISR);
```

参数： address：中断向量表中设置的ISR地址

返回值： void

其他影响： 无

void ISR_Stop(void)

说明： 禁用并删除中断。

参数： void

返回值： void

其他影响： 无

void ISR_Interrupt(void)

说明： 组件的默认ISR。将该函数置于对应的C文件中，并在START和END注释间加入代码。

参数： void

返回值： void

其他影响： 无

void ISR_SetVector(cyisraddress address)

说明： 更改中断的ISR向量。使用该函数将ISR向量更改为不同的中断服务子程序的地址。请注意，调用ISR_Start()会覆盖该API具备的所有效果。要在组件启动之前设置向量，请使用ISR_StartEx()。

定义ISR函数时，可通过使用CY_ISR和CY_ISR_PROTO宏来提供跨编译器的一致性定义：

函数定义示例：

```
CY_ISR(MyISR)
{
        /* ISR Code here
}
```

函数原型示例：

```
CY_ISR_PROTO(MyISR);
```

参数： address：中断向量表中设置的ISR地址

返回值： void

其他影响： 调用该函数前禁用中断，然后再重新使能它。

cyisraddress ISR_GetVector(void)

说明： 获得中断当前ISR向量的地址。

参数： void

返回值： cyisraddress：当前ISR的地址

其他影响： 无

void ISR_SetPriority(uint8 priority)

说明： 设置中断的优先级。

注意：调用ISR_Start()或ISR_StartEx()会覆盖该API具备的所有效果。该API应仅在已调用ISR_Start()或ISR_StartEx()后调用。要设置组件的初始优先级，请使用设计范围资源中断编辑器。

参数： priority：中断的优先级。

在PSoC 3和PSoC 5LP上，优先级的范围为0至7。

在PSoC 4上，优先级的范围为0至3。

在所有情况中，0都是最高优先级。

返回值： void

其他影响： 无

uint8 ISR_GetPriority(void)

说明： 获得中断的优先级。

参数： void

返回值： 中断的优先级。

在PSoC 3和PSoC 5LP上，优先级的范围为0至7。

在PSoC 4上，优先级的范围为0至3。

在所有情况中，0都是最高优先级。

其他影响： 无

void ISR_Enable(void)

说明： 使能中断控制器的中断。请勿调用该函数，除非已调用ISR_Start()或者已调用设置向量和优先级的ISR_Start()函数的功能。

参数： void

返回值： void

其他影响： 无

uint8 ISR_GetState(void)

说明： 获得中断的状态（使能、禁用）。

参数： void

返回值： 如果已启用，则为1；如果已禁用，则为0。

其他影响： 无

void ISR_Disable(void)

说明： 禁用中断控制器的中断。

参数： void

返回值： void

其他影响： 无

void ISR_SetPending (void)

说明： 使中断进入待处理状态，这是一种生成中断的软件API。

参数： void

返回值： void

其他影响： 如果已使能中断且正确设置，则将进入ISR（取决于该中断及其他待处理中断的优先级）。

void ISR_ClearPending (void)

说明： 清除中断控制器的一个待处理中断。

参数： void

返回值： void

其他影响： 需要通过适当的模块API（GPIO、UART 等）才能清除某些中断源，否则它们将使中断再次进入待处理状态。进入ISR将清除某些中断源的待处理位。

MISRA 合规性

本节介绍了 MISRA-C:2004 合规性和本组件的偏差情况。定义了两种类型的偏差：项目偏差 — 适用于所有 PSoC Creator 组件的偏差；特定偏差 — 仅适用于该组件的偏差。本节提供了有关组件特定偏差的信息。*《系统参考指南》*的 MISRA 合规性章节中介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

此中断组件具有如下特定偏差。

MISRA-C:2004 规则	规则类别（必须/建议）	规则说明	偏差说明
8.8	R	外部目标或函数只能在一个文件中声明。	使用外部链接声明IntDefaultHandler（PSoC 5LP/PSoC 4和CyRamVectors（PSoC 4），但是这些声明并不位于头文件中。
10.5	R	如果按位运算符“~”和“<<”被采用于底层类型无符号“char”或无符号“short”的操作数，得到的结果被立即转换为底层类型操作数。	只适用于PSoC 4。将“~”使用于SetPriority()函数前，掩码不被转换为“uint32”。此特殊情况没有其他影响。
17.4	R	数组索引是唯一允许的指针运算形式。	只适用于PSoC 5LP。指针运算形式用于与Ram向量表一起操作。

样例固件源代码

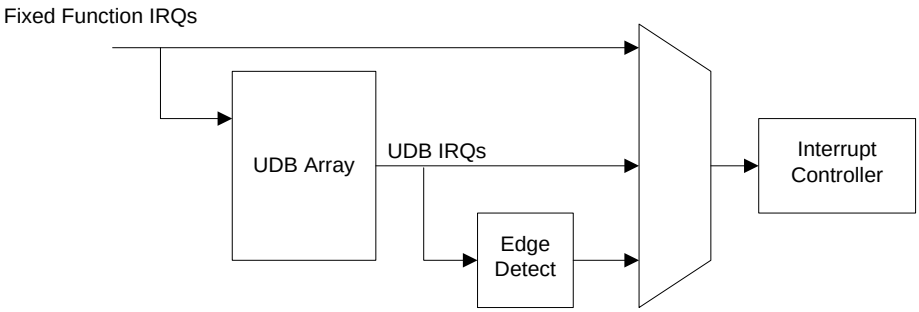
PSoC Creator 在“Find Example Project”（查找示例项目）对话框中提供了很多包括原理图和代码示例的示例项目。要查看特定组件实例，请打开“Component Catalog”中的对话框或原理图中的组件样例。要查看通用示例，请打开“Start Page”或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的项目列表。

更多有关信息，请参考《PSoC Creator 帮助》中主题为“查找示例项目”中的内容。

功能说明

PSoC 架构中的中断路由具有灵活性。除了固定功能外设之外，UDB 阵列路由中的所有数据信号都可用于生成中断。中断多路复用器（IDMUX）路由的高层视图如图 1 所示。IDMUX 从中断请求的可用源中进行选择。

图 1. IDMUX 路由



设计范围资源

在设计中采用一个中断组件时会在设计范围资源编辑器中生成一个入口。**Interrupts**（中断）选项卡包含下列参数：

Instance Name	Priority	Vector
isr_1	Default <7>	0
isr_2	Default <7>	1

- **Instance Name**（实例名称） — 显示设计中的组件实例名称。
- **Priority**（优先级） — 显示并允许设置实例的优先级。
- **Vector**（向量）— 表示中断向量。

资源

每个中断组件占用器件中断向量存储器中的一个入口。

API 存储器大小

根据编译器、器件、所使用的 API 数量以及组件的配置情况不同，组件的存储器大小也不一样。下表提供了组件配置中所有 API 占用的存储器大小。

通过使用“释放”模式中的相应编译器，可以进行测量操作。在该模式下，存储器的大小得到优化。有关特定的设计，可分析编译器生成的映射文件以确定存储器的大小。

配置	PSoC 3 （Keil_PK51）		PSoC 4（GCC）		PSoC 5LP（GCC）	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
默认值	141	0	214	0	196	0

组件更改

本节列出了各版本的主要组件更改内容。

版本	更改说明	更改原因/影响
1.70.c	向SetVector和StartEx API的说明内容中增加了有关CY_ISR/CY_ISR_PROTO宏的注意事项	说明了CY_ISR/CY_ISR_PROTO宏的使用情况。
	进行了较小程度的编辑和更新	
1.70.b	更新了MISRA-C规则表。	
1.70.a	添加了针对不同器件中断优先级的注意事项。	PSoC 4支持。
1.70	已添加了MISRA合规性章节。	此组件具有一项特定偏差。
1.60	对数据手册进行了少量编辑和更新	
1.50.c	改进数据表中对Derived（派生）选项的解释。	
1.50.b	数据手册纠正	
1.50.a	对数据手册进行了少量编辑和更新	
1.50	添加了InterruptType参数。	原有功能（相当于选择新版本中的“DERIVED”功能）无法在所有情况下确定所需的中断类型，因此新版本中添加了手动指定的功能。
	如果CYINT_VECTORS和CYINT_IRQ_BASE已经存在，请不要重新定义。	这些宏已经在*CyLib.h*中进行了定义。重新定义会引起某些版本的cy_boot警告。该变更仅对PSoC 5产生影响。
	使用CY_ISR来确定ISR。	这样将促使编译器生成能够确保PSoC 5上堆栈准确对齐的代码。

版本	更改说明	更改原因/影响
	使用*cydevice_trm.h*，而不是*cydevice.h*。	*cydevice.h*已过期，只有在需要与原组件和固件进行兼容时才可以使用。如果中断API函数中的代码需要使用*cydevice.h*，则请在“Place your includes, defines, and code here”部分中加入*cydevice.h*。
	添加了ISR_StartEx	允许在中断开始前在中断向量表中设定ISR地址设置，使其可用作默认值，而不是ISR_Interrupt。
	向以下函数添加了`=ReentrantKeil($INSTANCE_NAME . "_...")`： void ISR_Stop() void ISR_SetVector() cyisraddress ISR_GetVector() void ISR_SetPriority() uint8 ISR_GetPriority() void ISR_Enable() uint8 ISR_GetState() void ISR_Disable() void ISR_SetPending() void ISR_ClearPending()	如需重入，请允许用户将这些API设置为重新进入。
1.20	ES2 ISR修补。	