中断

1.50

特性



- 定义硬件触发的中断
- 提供一种软件方法来挂起中断

概述

Interrupt 组件定义硬件触发的中断。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，设计范围资源一节）。

中断控制器可以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 资源保持占用和活跃状态。大多数固定功能外设具有电平敏感型中断，包括 UDB FIFO 和状态寄存器。
- **脉冲** — 理想情况下，一个脉冲 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。无需对外设进行固件操作。
- **沿** — 任意同步波形是边沿检测电路的输入，并且该波形的正沿变成一个同步单周期脉冲（脉冲模式）。

注：以上中断波形的类型不同于 **InterruptType** 的 **Configure（配置）**对话框中的设置。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，无论选择何种 **InterruptType**
多路复用器，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，请参见适用的 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，它们能利用硬件支持降低延迟和事件检测的开销。

输入/输出连接

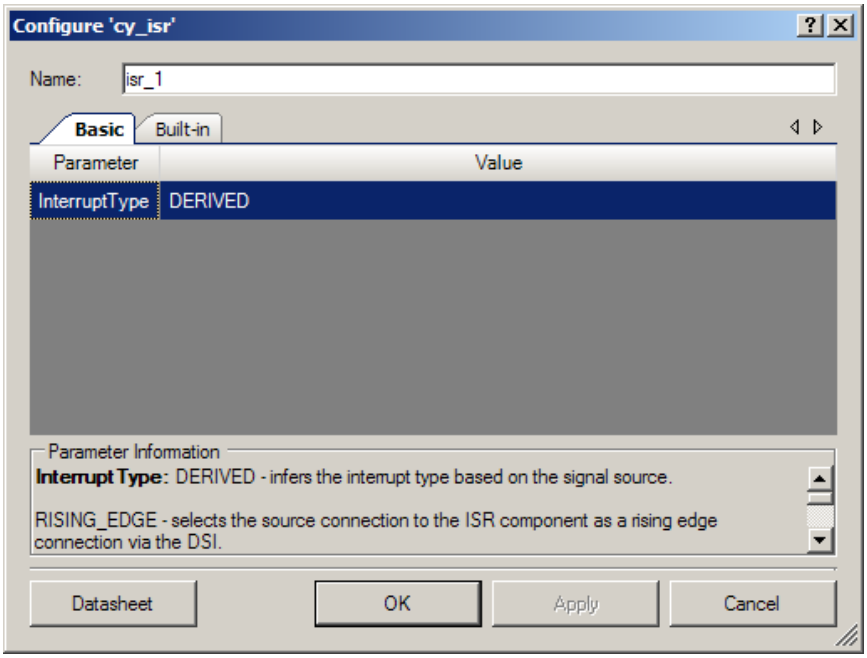
本节介绍中断组件的各种输入和输出连接。

int_signal – Input（输出）

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。

元件参数

将一个中断组件拖放到您的设计上，并双击以打开 **Configure**（配置）对话框。



中断组件提供了以下参数：

InterruptType

该参数配置用于进行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** —
  在源信号的上升沿触发中断。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。
- **LEVEL** — 选择源连接至中断，作为通过 DSI
  进行的电平敏感型连接。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** — 这是默认设置。连接到固定功能块（I^2C、USB、CAN
  等）时，它检查“int_signal”的驱动，派生出它连接时所依据的中断类型。自动分配根据器件数据手册中的信息进行。

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 来捕获事件（例如周期时钟），选择 LEVEL 捕获状态（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。

放置

每个中断组件占用器件中断矢量存储器中的一个入口。

资源

中断组件使用以下器件资源：

- 中断服务子程序 (ISR) 代码的闪存空间
- 器件中断矢量存储器中的一个入口

模拟模块	数字模块					API 存储器（字节）		引脚（每个外部 I/O）
	数据路径	宏单元	状态寄存器	控制寄存器	计数器 7	闪存	随机访问内存	
不可用	不可用	不可用	不可用	不可用	不可用	189	0	不可用

应用程序编程接口

应用程序编程接口 (API)

子程序允许您使用软件配置组件。下表列出了每个函数的接口，并进行了说明。以下各节将详细介绍每个函数。

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 矢量。
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 矢量。
ISR_GetVector()	获得中断当前 ISR 矢量的地址。
ISR_SetPriority()	设置中断的优先级。
ISR_GetPriority()	获得中断的优先级。
ISR_Enable()	启用中断控制器的中断。
ISR_GetState()	获得中断的状态（启用、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： 设置中断并启用。该函数会禁用中断，设置默认中断矢量，根据设计范围资源中断编辑器中的值设置优先级，然后启用中断控制器的中断。

参数： void

Return Value（返回值）： void

Side Effects（副作用）： 无

void ISR_StartEx(cyisraddress address)

说明：	设置中断并启用。该函数会禁用中断，根据传入的地址设置中断矢量，根据设计范围资源中断编辑器中的值设置优先级，然后启用中断控制器的中断。
参数：	address：中断矢量表中设置的 ISR 地址
Return Value（返回值）：	void
Side Effects（副作用）：	无

void ISR_Stop(void)

说明：	禁用并删除中断。
参数：	void
Return Value（返回值）：	void
Side Effects（副作用）：	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
Return Value（返回值）：	void
Side Effects（副作用）：	无

void ISR_SetVector(cyisraddress address)

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

参数： address：中断矢量表中设置的 ISR 地址

Return Value（返回值）： void

Side Effects（副作用）： 调用该函数前禁用中断，然后再重新启用。

cyisraddress ISR_GetVector(void)

说明： 获得中断当前 ISR 矢量的地址。

参数： void

Return Value（返回值）： cyisraddress：当前 ISR 的地址

Side Effects（副作用）： 无

void ISR_SetPriority(uint8 priority)

说明： 设置中断的优先级。

注：调用 ISR_Start() 或 ISR_StartEx() 会覆盖该方法具备的所有效果。该方法应仅在已调用 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（优先级）：中断的优先级。0 到 7，0 为最高。

Return Value（返回值）： void

Side Effects（副作用）： 无

uint8 ISR_GetPriority(void)

说明： 获得中断的优先级。

参数： void

Return Value（返回值）： 中断的优先级。0 到 7，0 为最高。

Side Effects（副作用）： 无

void ISR_Enable(void)

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

参数： void

Return Value（返回值）： void

Side Effects（副作用）： 无

uint8 ISR_GetState(void)

说明： 获得中断的状态（启用、禁用）。

参数： void

Return Value（返回值）： 如果已启用，则为 1；如果已禁用，则为 0。

Side Effects（副作用）： 无

void ISR_Disable(void)

说明： 禁用中断控制器的中断。

参数： void

Return Value（返回值）： void

Side Effects（副作用）： 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方法。
参数：	void
Return Value（返回值）：	void
Side Effects（副作用）：	如果中断已启用且设置正确，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
Return Value（返回值）：	void
Side Effects（副作用）：	某些中断源同样将需要使用适当的模块 API（GPIO、UART 等）进行清除，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。

固件源代码示例

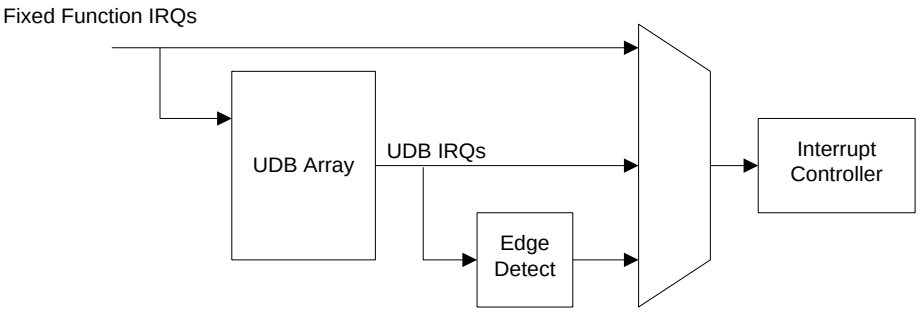
PSoC Creator 在“Find Example Project（查找示例项目）”对话框中提供了很多包括原理图和代码示例的示例项目。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要获取通用的示例，请打开开始页或**File**（文件）菜单中的对话框。根据需要，使用对话框中的**Filter Options**（滤波器选项）可缩小可选项目的列表。

有关更多信息，请参考 PSoC Creator 帮助中的“查找示例项目”主题。

功能描述

PSoC 3 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) 路由的高级视图如图1所示。IDMUX 从中断请求的可用源中进行选择。

图1. IDMUX 路由选择



设计范围资源

在设计中采用一个中断组件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	ES2 Patch	Vector
isr_1	Default <7>	☐	1
isr_2	Default <7>	☐	3

- **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。
- **ES2 Patch**（ES2 修补）— 这是解决 PSoC 3 中断问题的临时解决方法。高优先级中断到达的同时,优先级较低的中断刚好返回，优先级较高的中断会丢失。如果应用中使用了多个中断，请检查该选项。请注意，在芯片更新之前，该选项是临时的。
- **Vector**（矢量）— 表示中断矢量。

组件更改

本节介绍组件与以前版本相比的主要更改。

版本	更改说明	更改/影响原因
1.50.c	改进数据手册中对 Derived（派生）选项的解释。	
1.50.b	数据手册纠正	
1.50.a	对数据表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确定的功能。
	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。该变更仅影响 PSoC 5。
	使用 CY_ISR 确定 ISR。	这将促使编译器生成能确保 PSoC 5 上堆栈正确对齐的代码。
	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“将包含的内容、定义和代码放置于此”一节中加入*cydevice.h*。
	添加了 ISR_StartEx	允许在中断开始前在中断矢量表中设定 ISR 地址设置，使其可用作默认值，而不是 ISR_Interrupt。

版本	更改说明	更改/影响原因
	将 `=ReentrantKeil($INSTANCE_NAME . "_...")` 添加到下列函数中： 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。	

©
赛普拉斯半导体公司，2010年2012。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品的内嵌电路之外，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，否则赛普拉斯产品不保证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。

PSoC® 是赛普拉斯半导体公司的注册商标，PSoC Creator™ 和 Programmable System-on-Chip™ 是赛普拉斯半导体公司的商标。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。

所有源代码（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。除上述指定的用途之外，未经赛普拉斯的明确书面许可，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（但不仅限于）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。

产品使用可能受适用的赛普拉斯软件许可协议限制。