PSoC™ 6 Forum Discussions

Anonymous Not applicable May 19, 2018 12:19 PM

GPIO + Interrupt = Confused

✔ Jump to solution

Hi,

So I have a simple application with a GPIO and an interrupt.

I want the interrupt to trigger on either edge. So using the GUI I set the pin configuration as an input with a pull up, and the input pin to trigger on either edge. Then I connected the HW connection as shown above and configured the interrupt schematic component using the GUI for auto-select.

So, after building and testing, the interrupt is only triggering on rising edge....

Am I setting this up correctly?

Thanks

Scott

Solved! Go to Solution.

Likes 0 Reply Subscribe

1 Solution ✔

AlanH_86 100 50 25 Employee

May 20, 2018 03:43 AM

Re: GPIO + Interrupt = Confused

☑ There is a bit of a trick here.

If you look at the NVIC you will find that in P6 there are 147 possible sources of interrupts to the M4.

Every I/O port on the chip (not pin... port) has one interrupt signal that is generated with a bunch of logic on an per pin basis and or-ed together to generate that ports interrupt.

In addition each of the UDBs have dedicated interrupt signals.

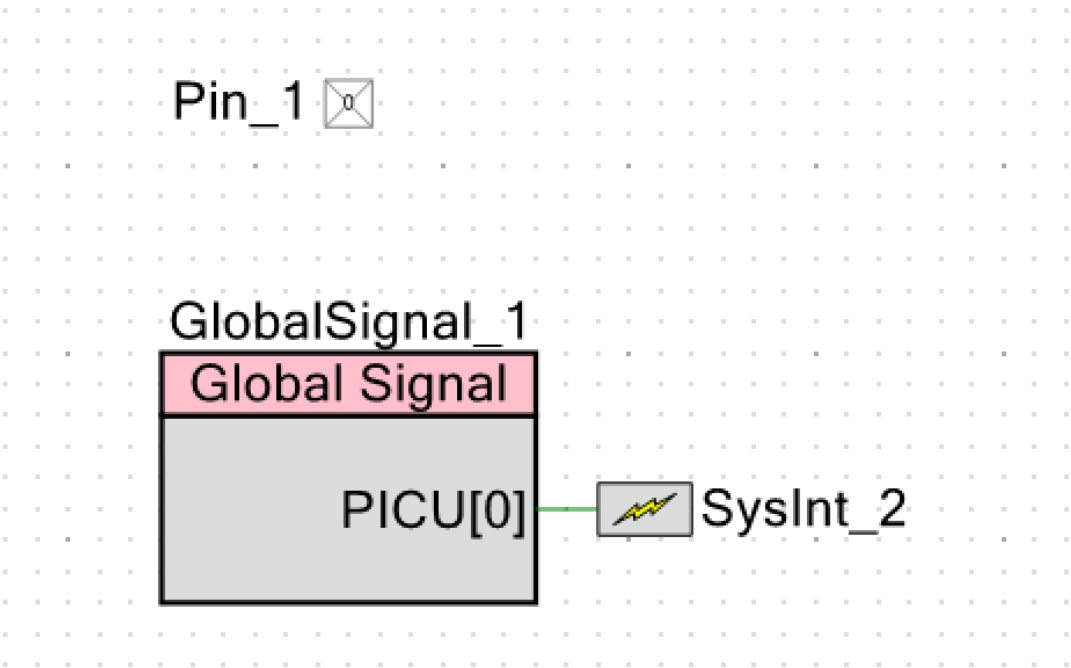
When you connect the pin directly to the interrupt like that you are telling PSoC Creator that you want to put the pin through a UDB and do logic on it to create and interrupt. For some reason (which I dont know) in your configuration you can only

generate falling edge interrupts.

On the GPIO configuration wizard you are configuring the PORT interrupt (not the UDB interrupt). In other words you aren't doing anything.

So. The bottom line is if you need falling and rising edges you need to use the port interrupt.

You can either configure it with PDL... or configure it your schematic like so:



View solution in original post

Is this solution helpful? Yes No

Likes 0 Reply

7 Replies

rola_264706 Level 8

May 19, 2018 12:28 PM

Re: GPIO + Interrupt = Confused

You can select these types of Interrupts

Interrupt Type

This parameter configures which type of waveform the Component will process to trigger the interrupt. There are three possible values for this parameter:

▪ Auto-Select Trigger – This is the default setting. It inspects the driver of the int_signal and infers the interrupt type based on the signal source. For most fixed-function blocks, the interrupt type is LEVEL. For UDB signal sources, the interrupt type is RISING_EDGE.

▪ Rising-Edge Triggered – Triggers the interrupt on the rising edge of the source signal. This type of connection uses UDB resources.

▪ Level Triggered – Selects the source connected to the interrupt as a level-sensitive connection. All fixed function peripherals are level triggered.

As a guideline, use RISING_EDGE when capturing a signal change (for example, periodic clock), and use LEVEL when capturing a state change of a peripheral (for example, FIFO fill Levels.

Likes 0 Reply

rola_264706 Level 8

May 19, 2018 12:37 PM

In response to **rola_264706**

Re: GPIO + Interrupt = Confused

The GPIO pin

Interrupt settings

This parameter selects whether the pin can generate an interrupt and, if selected, the interrupt type. All pins on a port logically OR their interrupts together and generate a single interrupt signal via a dedicated Port Interrupt. A device level Combined Port Interrupt (AllPortInt) signal can also be used. Both types are available through the Global Signal Reference Component.

After an interrupt occurs, the interrupt source must be cleared in software to clear the latched pin events to enable detection of future events. This is accomplished by calling the Cy_GPIO_ClearInterrupt() function.

The following options are supported:

None is Default.

Rising Edge

Falling Edge

Both Edges

Likes 0 Reply

AlanH_86 100 50 25 Employee

May 20, 2018 03:43 AM

Re: GPIO + Interrupt = Confused

There is a bit of a trick here.

If you look at the NVIC you will find that in P6 there are 147 possible sources of interrupts to the M4.

Every I/O port on the chip (not pin... port) has one interrupt signal that is generated with a bunch of logic on an per pin basis and or-ed together to generate that ports interrupt.

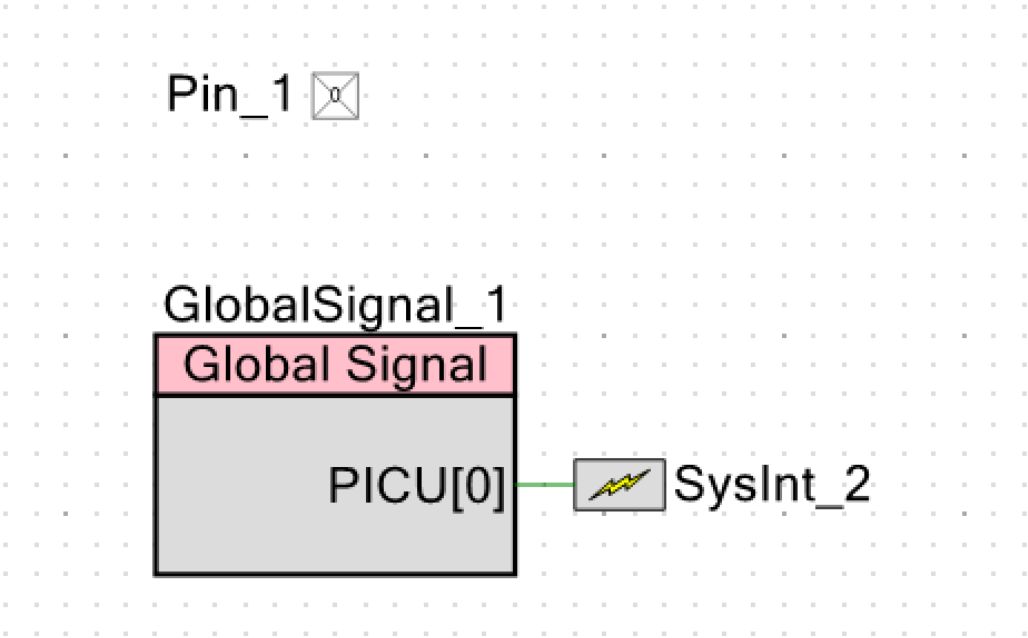
In addition each of the UDBs have dedicated interrupt signals.

When you connect the pin directly to the interrupt like that you are telling PSoC Creator that you want to put the pin through a UDB and do logic on it to create and interrupt. For some reason (which I dont know) in your configuration you can only generate falling edge interrupts.

On the GPIO configuration wizard you are configuring the PORT interrupt (not the UDB interrupt). In other words you aren't doing anything.

So. The bottom line is if you need falling and rising edges you need to use the port interrupt.

You can either configure it with PDL... or configure it your schematic like so:



Likes 0 Reply

AlanH_86 100 50 25 Employee

May 20, 2018 03:44 AM

In response to **AlanH_86**

Re: GPIO + Interrupt = Confused

And you should look at the application note for interrupts

http://www.cypress.com/documentation/application-notes/an217666-psoc-6-mcu-interrupts

Likes 0 Reply

Anonymous Not applicable May 20, 2018 04:42 AM

In response to **AlanH_86**

Re: GPIO + Interrupt = Confused

Hi,

Again, thanks for all of the details....

So in the additional image you attached, shown with the global signal icon connected to the interrupt icon, that uses the native m4 interrupt and no udb logic?

To reiterate my understanding : configure the pin in the gui as needed for both edges, then drag in the icon as you show, and set it up as shown in the example, yes?

When you use the pin GUI to set up both edges, is this equivalent to these function calls

Cy_GPIO_SetInterruptEdge(SW2_P0_4_PORT, SW2_P0_4_NUM, CY_GPIO_INTR_BOTH);

Cy_GPIO_SetInterruptMask(SW2_P0_4_PORT, SW2_P0_4_NUM, CY_GPIO_INTR_EN_MASK);

so therefore the above calls are no longer needed in the code?

And what is the difference between the calls

Cy_GPIO_ClearInterrupt(PIN_X_LIMIT_PORT, PIN_X_LIMIT_NUM);

and

NVIC_ClearPendingIRQ(IRQ_X_LIMIT_cfg.intrSrc);

What is the second function used for if the latched signal is cleared with the first call?

--Scott

Likes 0 Reply

Anonymous Not applicable

May 20, 2018 05:37 AM

In response to **Anonymous**

Re: GPIO + Interrupt = Confused

Thank you, my code is now working as expected....

Likes 0 Reply

AlanH_86 100 50 25 Employee

May 21, 2018 05:19 AM

In response to **Anonymous**

Re: GPIO + Interrupt = Confused

>So in the additional image you attached, shown with the global signal icon

>connected to the interrupt icon, that uses the native m4 interrupt and no

>udb logic?

YEs that is correct. On the global signal pick the GPIO port you are talking about.

All the interrupt component does is hook your code up to the right place in the NVIC interrupt table.. and give you a few APIs

>To reiterate my understanding : configure the pin in the gui as needed for

>both edges, then drag in the icon as you show, and set it up as shown in

>the example, yes?

Correct

>When you use the pin GUI to set up both edges, is this equivalent to these

>function calls

>Cy_GPIO_SetInterruptEdge(SW2_P0_4_PORT, SW2_P0_4_NUM, CY_GPIO_INTR_BOTH);

>Cy_GPIO_SetInterruptMask(SW2_P0_4_PORT, SW2_P0_4_NUM, CY_GPIO_INTR_EN_MASK);

It is almost equivalent. The only difference is those settings get turned into binary which are then coped into the right place when the chip boots. (I think)

>so therefore the above calls are no longer needed in the code?

Correct

> And what is the difference between the calls

>Cy_GPIO_ClearInterrupt(PIN_X_LIMIT_PORT, PIN_X_LIMIT_NUM);

If you look in the TRM you will find a picture of the GPIO. In the GPIO PORT you will find a REGISTER that holds the OR of a bunch of signals in the port. That register is then connected to the NVIC.

So when you run the above function you are CLEARING that register... in other words you are stopping the port from keeping the interrupt active. If you dont clear the triggering interrupt, when you finish the ISR it will jump right back into it.

>and

>NVIC_ClearPendingIRQ(IRQ_X_LIMIT_cfg.intrSrc);

This is the CMSIS API to clear the NVIC interrupt. It is effectively automatic in an ISR.

>What is the second function used for if the latched signal is cleared with

>the first call?

You might do this for instance if you have interrupts OFF and before you turn the interrupts back on you want to clear possible pending interrupts... in other words you want to put the system into a known state before you turn the interrupts back on.

--Scott

Like 1 Reply