Interrupts with PSoC 5

Interrupts (PSoC 3 & 5LP)

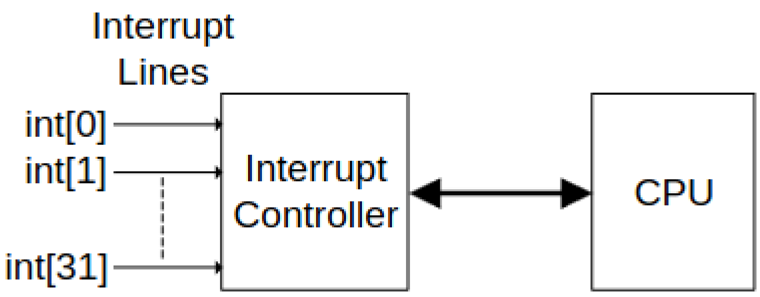

Fig. 1: PSoC 3, PSoC 5LP Interrupt Architecture

In PSoC Creator

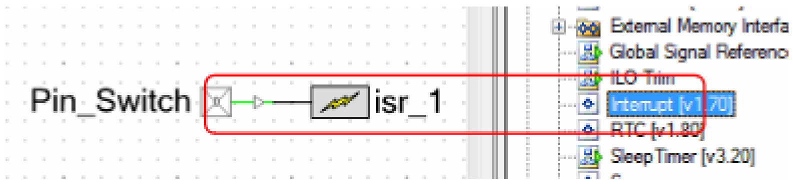

Fig. 2: Editor

Configuration

InterruptType can be either

- DERIVED
- RISING_EDGE
- LEVEL

Skriptum Kupfner

Interrupts: S11
Supports 32 Interrupt signals with 8 Priorities (0 (highest) to 7 (lowest)) which can be changed dynamically.
When a higher priority interrupt occurs, the currently running interrupt is interrupted (as if it was a normal program).

Traps vs Interrupts

- Interrupt: Asynchron

- Trap: Synchron

Level vs Pulse Interrupts

Level has to be cleared by firmware, Pulse is executed on rising edge

Beispiel

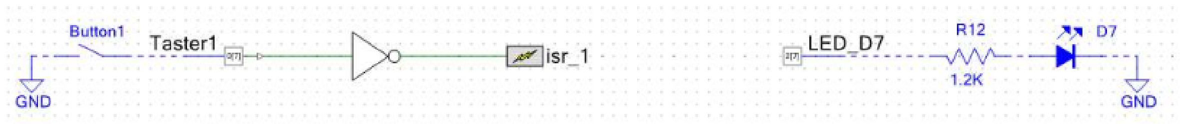

Fig. 3: 89aeb397399df729ba02eacd49034a59.png

example code

```
#include "project.h"

CY_ISR(Taster1_ISR) {
  LED_D7_Write(1);
  CyDelay(500);
  LED_D7_Write(0);
  CyDelay(100);
}

int main(void) {
  isr_1_StartEx(Taster1_ISR);
  isr_1_ClearPending();
  CyGlobalIntEnable; /* Enable global interrupts. */

  /* Place your initialization/startup code here (e.g. MyInst_Start()) */

  for(;;)
  {
  /* Place your application code here. */
  }
}
```

in this case, `CY_ISR` is a macro for interrupt-definitions. The name (in this case **Taster1_ISR**) can be chosen freely, but must be tha same in the macro and the `StartEx` function.

`ClearPending();` resets clears all interrupt flags, `CyGlobalIntEnble;` enables all global interrupts.

Using Falling_Edge

When interrupting on falling_edge instead of risint_edge, the flag needs to be reset manually. If it is not reset, the interrupt function will becall over and over until it is reset.
To reset the Flag, `Taster1_ClearInterrupt();` needs to be called from within the interrupt function.

Interrupt priority

To change the interrupt priority in the **GUI**, go to *Design Wide Resources* $\longrightarrow$ *Interrupts*.
To change the interrupt priority in **code**, call `<interrupt name>_SetPriority(<priority>);`.

The currently set priority can be read with `<interrupt_name>_GetPriority()`.

Enable / Disable

Interrupts can be enabled / disabled using

- `<interrupt_name>_Enable();`
- `<interrupt_name>_Disable();`

Interrupt functions for c

Name	Description
CY_ISR(<interrupt_name>)	Macro for interrupt-function-definition
<interrupt_name>_StartEx(<interrupt_name>)	starts the interrupt
<interrupt_name>_ClearPending()	clears set interrupt flags
…	

Beispiel

Not-aus Industrieanlage (LED als Motor)

```
/*
main.c

18.10.2023

Author:
Nilusink
*/
#include "project.h"
#include "stdbool.h"

bool e_stop = false;


void all_off()
{
    LED1_Write(0);
    LED2_Write(0);
    LED3_Write(0);
    LED5_Write(0);
    LED6_Write(0);
    LED7_Write(0);
}
```

```
CY_ISR(EStop) {
    isr_1_ClearPending();

    e_stop = true;

    all_off();

    ESTOP_LED_Write(1);
}

int main(void) {
    isr_1_StartEx(EStop);
    isr_1_ClearPending();
    CyGlobalIntEnable; /* Enable global interrupts. */

    ESTOP_LED_Write(0);  // make sure that the E-Stop light is off

    int i = 0;
    for(;;)
    {
        // motor is doing something continually
        // (in this case, just blink)
        all_off();

        if (!e_stop)  // check for emergency stop
        {
            // turn on leds in a circle-animation
            switch (i % 6)
            {
                case 0:
                    LED1_Write(1);
                    break;
                case 1:
                    LED2_Write(1);
                    break;
                case 2:
                    LED3_Write(1);
                    break;
                case 3:
                    LED5_Write(1);
                    break;
                case 4:
                    LED6_Write(1);
                    break;
                case 5:
                    LED7_Write(1);
                    break;
            }
        }
        else
        {
            // if emergency-stop is pressed, blink the center led
            if (i % 5 == 0)
            {
                ESTOP_LED_Write(1);
            }
            else
            {
                ESTOP_LED_Write(0);
            }
        }
        CyDelay(50);
        i++;
    }
}
```

Quellen

PSoC Interrupts