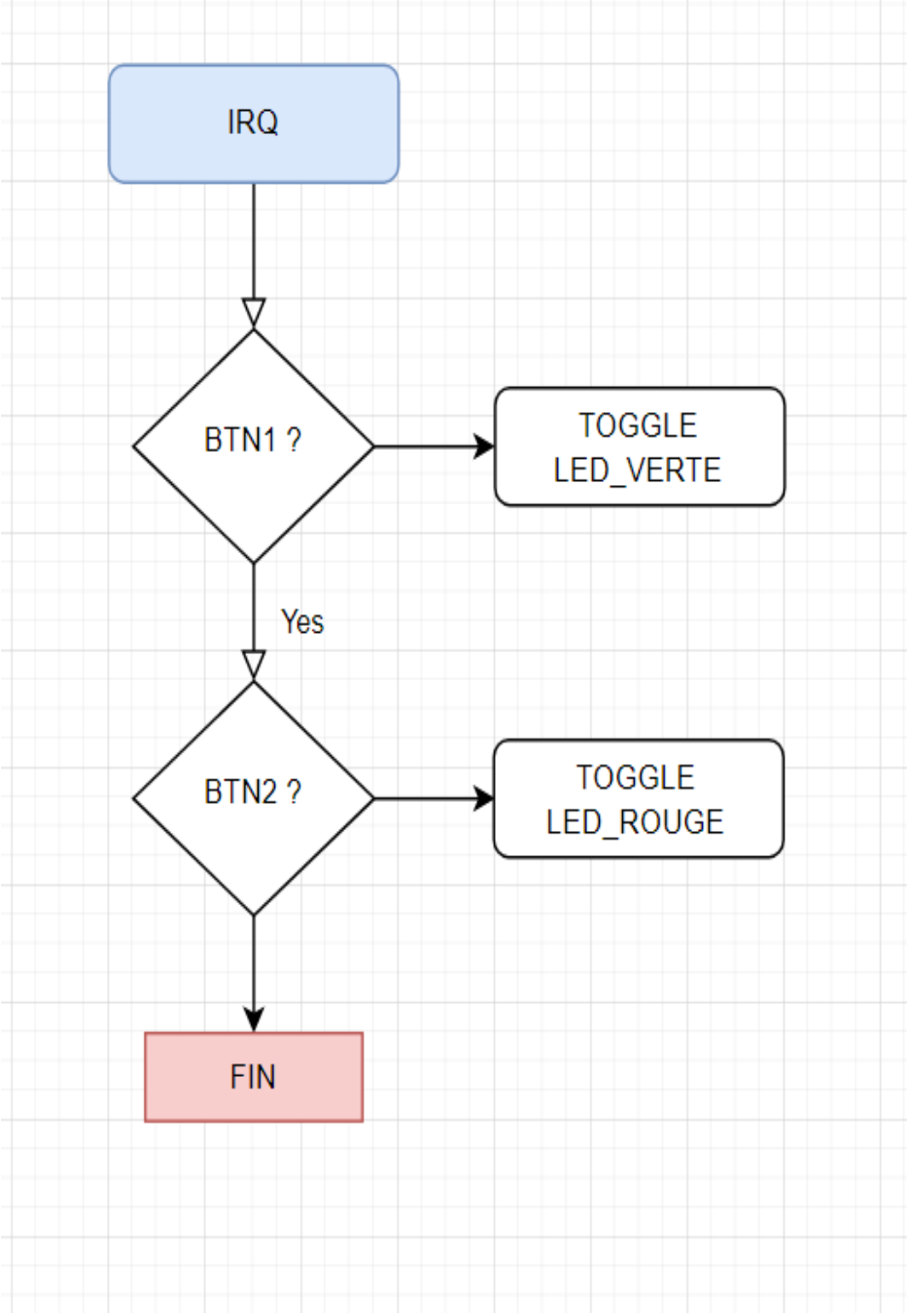
Exemples d'Interruption Hardware (IRQ)

** Une interruption hardware est généralement, un évènement générer par un élément extérieur (généralement aléatoire)
** Une interruption hardware possède la **priorité la plus haute** du système.
** Un IRQ ne **peut pas être préempté**.

Exercice :
- Création de la fonction d'interruption + fonction de traitement
- Liaison de la fonction avec le noyau temps réel



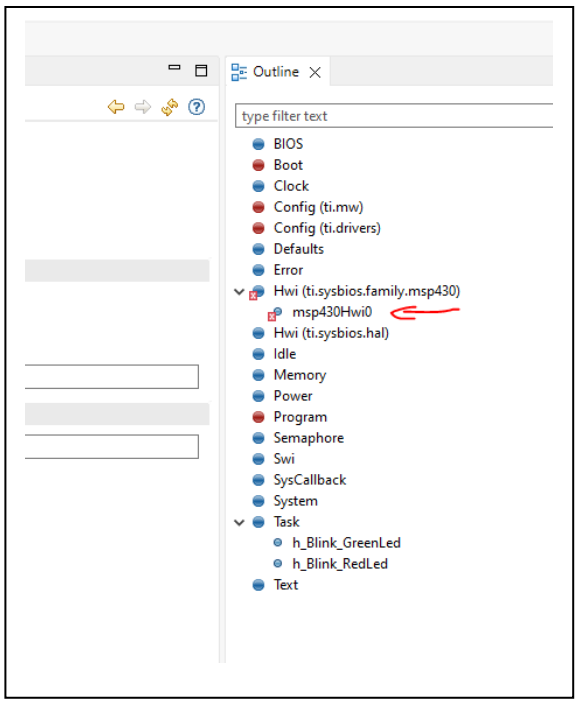
```c
/*
 * main.h
 *
 *  Created on: 14 oct. 2022
 *      Author: souns
 */

#ifndef MAIN_H_
#define MAIN_H_

//PORT1
#define Board_RED                GPIO_PIN0
#define BUTTON0                  GPIO_PIN1
#define BUTTON1                  GPIO_PIN2
//PORT9
#define Board_GREEN              GPIO_PIN7

#endif /* MAIN_H_ */
```

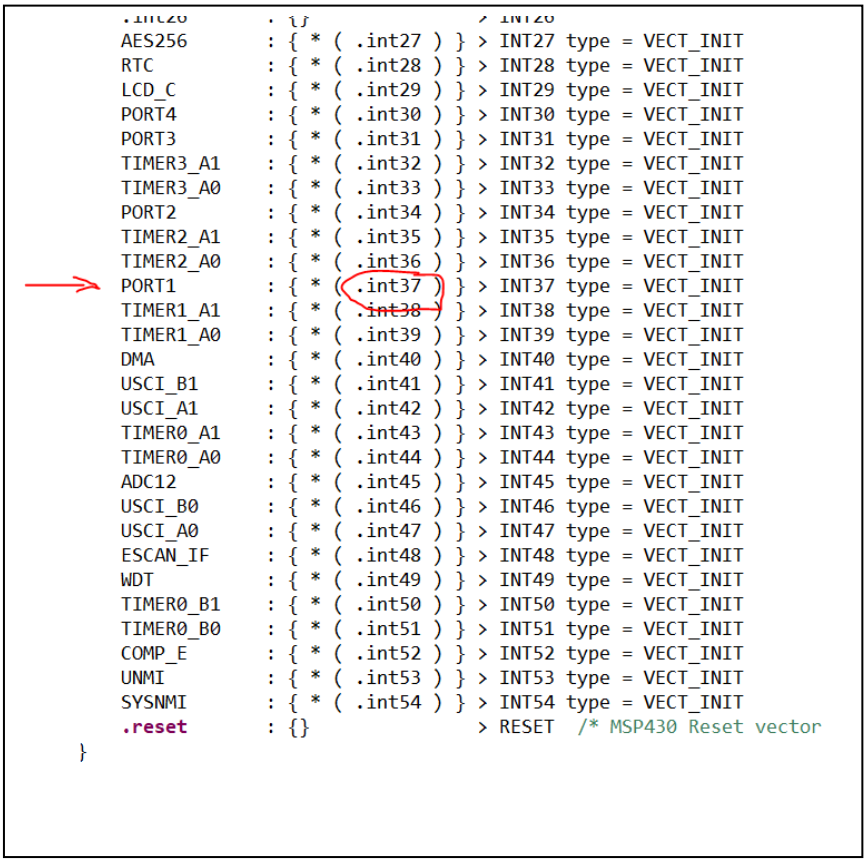
```c
//*****************************************************************************
// main.c
//*****************************************************************************
/* XDCtools Header files */
#include <xdc/std.h>
#include <xdc/runtime/System.h>
#include <xdc/cfg/global.h>

/* BIOS Header files */
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>

/* TI-RTOS Header files */
#include <driverlib.h>

/* Board Header file */
#include "main.h"


//*****************************************************************************
// Prototype
//*****************************************************************************

void Init_GPIO(void);
void IRQ_Bouton(unsigned int index);

//*****************************************************************************
// GPIO Init
//*****************************************************************************
void Init_GPIO(void)
{
    //PORT1
    GPIO_setAsOutputPin(GPIO_PORT_P1, Board_RED);
    GPIO_setOutputLowOnPin(GPIO_PORT_P1, Board_RED);
    GPIO_setAsInputPin(GPIO_PORT_P1, BUTTON0 + BUTTON1);
    GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P1, BUTTON0 + BUTTON1);
    GPIO_selectInterruptEdge(GPIO_PORT_P1, BUTTON0 + BUTTON1,
GPIO_HIGH_TO_LOW_TRANSITION);
```

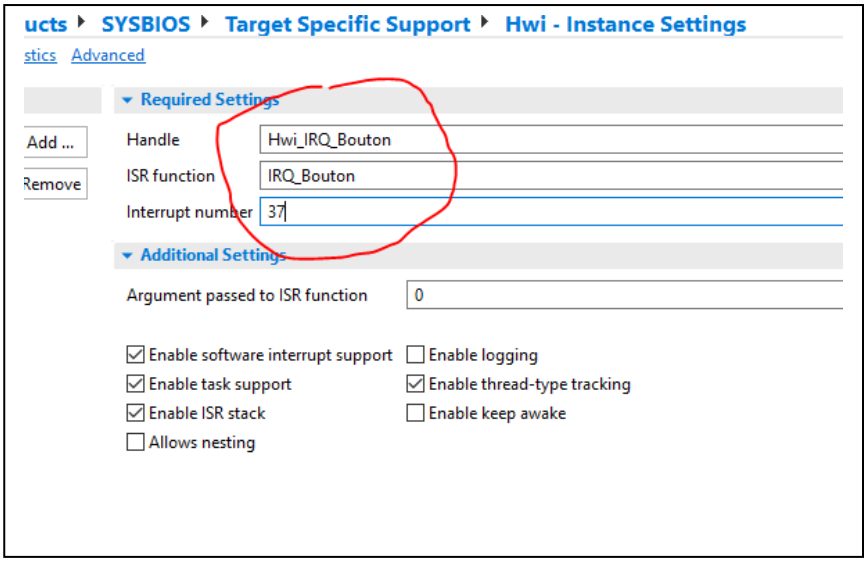
```
    GPIO_enableInterrupt(GPIO_PORT_P1, BUTTON0 + BUTTON1);
    //PORT9
    GPIO_setAsOutputPin(GPIO_PORT_P9, Board_GREEN);
    GPIO_setOutputLowOnPin(GPIO_PORT_P9, Board_GREEN);
}


//*********************************************************************
// IRQ sur les boutons
//*********************************************************************
void IRQ_Bouton(unsigned int index)
{
    if(!GPIO_getInputPinValue(GPIO_PORT_P8, BOUTTON1))
        {
        GPIO_toggleOutputOnPin(GPIO_PORT_P1, Board_RED);
        GPIO_clearInterrupt(GPIO_PORT_P8, BOUTTON1);
        }

    if(!GPIO_getInputPinValue(GPIO_PORT_P8, BOUTTON2))
        {
        GPIO_toggleOutputOnPin(GPIO_PORT_P9, Board_GREEN);
        GPIO_clearInterrupt(GPIO_PORT_P8, BOUTTON2);
        }
}
```

Liaison avec le noyau temps réel :

Ouvrir le fichier *.cfg et le fichier MSP_EXP430FR6989.cmd
Clic droit sur la rubrique HWI et création d'une nouvelle entrée
Remplir le tableau avec le nom de la fonction d'interruption ainsi que le numéro du vecteur d'interruption (dans le fichier MSP_EXP430FR6989.cmd).




.int26 : {} > INT26
AES256 : { * (.int27) } > INT27 type = VECT_INIT
RTC : { * (.int28) } > INT28 type = VECT_INIT
LCD_C : { * (.int29) } > INT29 type = VECT_INIT
PORT4 : { * (.int30) } > INT30 type = VECT_INIT
PORT3 : { * (.int31) } > INT31 type = VECT_INIT
TIMER3_A1 : { * (.int32) } > INT32 type = VECT_INIT
TIMER3_A0 : { * (.int33) } > INT33 type = VECT_INIT
PORT2 : { * (.int34) } > INT34 type = VECT_INIT
TIMER2_A1 : { * (.int35) } > INT35 type = VECT_INIT
TIMER2_A0 : { * (.int36) } > INT36 type = VECT_INIT
PORT1 : { * (.int37) } > INT37 type = VECT_INIT
TIMER1_A1 : { * (.int38) } > INT38 type = VECT_INIT
TIMER1_A0 : { * (.int39) } > INT39 type = VECT_INIT
DMA : { * (.int40) } > INT40 type = VECT_INIT
USCI_B1 : { * (.int41) } > INT41 type = VECT_INIT
USCI_A1 : { * (.int42) } > INT42 type = VECT_INIT
TIMER0_A1 : { * (.int43) } > INT43 type = VECT_INIT
TIMER0_A0 : { * (.int44) } > INT44 type = VECT_INIT
ADC12 : { * (.int45) } > INT45 type = VECT_INIT
USCI_B0 : { * (.int46) } > INT46 type = VECT_INIT
USCI_A0 : { * (.int47) } > INT47 type = VECT_INIT
ESCAN_IF : { * (.int48) } > INT48 type = VECT_INIT
WDT : { * (.int49) } > INT49 type = VECT_INIT
TIMER0_B1 : { * (.int50) } > INT50 type = VECT_INIT
TIMER0_B0 : { * (.int51) } > INT51 type = VECT_INIT
COMP_E : { * (.int52) } > INT52 type = VECT_INIT
UNMI : { * (.int53) } > INT53 type = VECT_INIT
SYSNMI : { * (.int54) } > INT54 type = VECT_INIT
.reset : {} > RESET /* MSP430 Reset vector
}


ucts ▸ SYSBIOS ▸ Target Specific Support ▸ Hwi - Instance Settings
stics Advanced
Required Settings
Add ...
Remove
Handle Hwi_IRQ_Bouton
ISR function IRQ_Bouton
Interrupt number 37
Additional Settings
Argument passed to ISR function 0
Enable software interrupt support
Enable logging
Enable task support
Enable thread-type tracking
Enable ISR stack
Enable keep awake
Allows nesting