Experiment 11

External/Internal Interrupts and ISR Programming

Lab Objective

In this lab we will learn to configure the interrupts in Cortex-M4 based Stellaris microcontroller. We will configure an interrupt for push button connected to pin 4 of GPIO port F.

Interrupt Control in Cortex-M4

LM4F120H5QR implements Nested Vectored Interrupt Controller to handle the interrupts. All the exceptions and interrupts are processed in handler mode. The processor returns to the thread mode when it is finished with all exception processing. The processor automatically saves its current state to the stack when it accepts an interrupt to be serviced. The state is automatically restored from the stack upon the exit from the interrupt service routine(ISR). When an interrupt occurs, state saving and vector fetch are performed in parallel reducing interrupt latency and enabling efficient interrupt entry.

Software can set eight priority levels(0 to 7; a higher level corresponds to a lower priority, i.e., level 0 is the highest interrupt priority) on seven exceptions(such as, reset, software interrupt, hardware interrupt, bus fault, etc.) and 65 interrupts.

When an exception occurs and it is accepted by processor core, the corresponding interrupt service routine is executed. Starting address of each exception is loaded from the vector table which is an array of word-sized data. Each entry in the vector table points to the starting address of one exception. The vector table is located at address 0x0000.0000 after reset.

Every exception handler is located at the address obtained by multiplying the corresponding interrupt number with 4. For example, if the reset is exception type 1, the address of the reset vector is 1 times 4 (each word is 4 bytes), which equals 0x00000004, and NMI vector (type 2) is located in 2 4 = 0x00000008. The entries of this vector table are given in table 2-8 and 2-9 of datasheet.

Enabling an Interrupt

To activate an interrupt source, following two steps must be followed:

1. Enable the source from the corresponding NVIC enable register [pg. 139].
2. Set the priority for that interrupt. [pg. 149]

For better understanding, we discuss the example of enabling the interrupt for Port F. Follow the following steps to activate the interrupt for port F.

1. Find the interrupt number (i.e., bit position in interrupt register) from Table 2-9 (2nd column) on pg.101 corresponding to GPIO Port F.
2. Find the interrupt register that is needed to enable IRQ30 from Table 3-8 (pg.131). It is NVIC_EN0_R. So, it tells you that you need to set bit 30 of NVIC_EN0_R to 1 to enable interrupt on Port F.
3. From Table 3-8, find the register needed to set the priority of IRQ 30. It is NVIC_PRI7_R. To set a priority value, say 5, you may use the following statement in C:

```
NVIC_PRI7_R = (NVIC_PRI7_R & 0xFF00FFFF) | 0x00A00000;
```

Configuring GPIO as Interrupt

To configure GPIO pin as interrupt and select the source of the interrupt, its polarity and edge properties following steps must be followed:

1. Disable the interrupts before writing to the control registers.
2. Select whether the source of interrupt is edge-sensitive or level sensitive using GPIO Interrupt Sense register (GPIOIS).[pg.623]
3. To enable interrupts for both edges write the appropriate value in the GPIO Interrupt Both Edges register(GPIOIBE). [pg.624]
4. Write the appropriate value in GPIO Interrupt Event register (GPIOIEV) to configure the corresponding pin to detect rising or falling edges depending on the corresponding bit value in the GPIO Interrupt Sense (GPIOIS) register. [pg. 625]
5. Clear the interrupt flag for the corresponding pin by asserting the appropriate bit in the GPIO Interrupt Clear Register (GPIOICR). [pg. 629]
6. Enable the interrupts by asserting the corresponding bits in GPIO Interrupt Mask register (GPIOIM). [pg. 626]

Source Code

Following code blinks LEDs of different colours on switch press. Understand the code and complete the missing portions. Consult the datasheet for complete understanding of the code.

```
1  // User button connected to PF4
2  //(turn on different LEDs on falling edge of button press)
3
4
5  #define SYSCTL_RCGCGPIO_R       (*((volatile unsigned long *)0x400FE608))
6
7  // IRQ 0 to 31 Set Enable Register
8  #define NVIC_EN0_R              (*((volatile unsigned long *)0xE000E100))
9  // IRQ 28 to 31 Priority Register
10 #define NVIC_PRI7_R             (*((volatile unsigned long *)0xE000E41C))
11
12 #define GPIO_PORTF_DATA_R       (*((volatile unsigned long *)0x400253FC))
13 #define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
14 #define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))
15 #define GPIO_PORTF_PUR_R        (*((volatile unsigned long *)0x40025510))
16
17 #define GPIO_PORTF_IS_R         (*((volatile unsigned long *)0x40025404))
18 #define GPIO_PORTF_IBE_R        (*((volatile unsigned long *)0x40025408))
19 #define GPIO_PORTF_IEV_R        (*((volatile unsigned long *)0x4002540C))
20 #define GPIO_PORTF_IM_R         (*((volatile unsigned long *)0x40025410))
21 #define GPIO_PORTF_ICR_R        (*((volatile unsigned long *)0x4002541C))
22
23 #define NVIC_EN0_INT30          ----------  //Interrupt 30 enable
24 #define PORTF_CLK_EN            ----------  //Clock enable for Port F
25 #define LEDs                    ----------  //Enable LEDs
26 #define SW1                     ----------  //Enable user switch SW1
27 #define INT_PF4                 ----------  //Interrupt at PF4
28
29
30 void EnableInterrupts(void);      //Disable interrupts
31 void DisableInterrupts(void);     //Enable interrupts
32 void Init_INT_GPIO(void);         //Initialize GPIO and Interrupts
33 void Delay(unsigned long value); //Implements delay
34 void WaitForInterrupt(void);
35
36 volatile unsigned long i = 0;
37
38 void Init_INT_GPIO(void){
39   volatile unsigned delay_clk;
40
41   SYSCTL_RCGCGPIO_R |= PORTF_CLK_EN;      //Enable clock for PORTF
42   delay_clk = SYSCTL_RCGCGPIO_R; //dummy read to stable the clock
43
```

```
44    //GPIO
45    GPIO_PORTF_DEN_R |= (SW1|LEDs); //Enable digital I/O on PF4, PF3 - PF1
46    GPIO_PORTF_DIR_R = ___________; //Make PF4 input and PF3 - PF1 output
47    GPIO_PORTF_PUR_R |= SW1;        //Enable weak pull up on PF4
48
49    //INTERRUPT
50    DisableInterrupts();
51    GPIO_PORTF_IS_R &= ~INT_PF4;      //PF4 is edge sensitive
52    GPIO_PORTF_IBE_R &= ~INT_PF4;     //PF4 is not both edges
53    GPIO_PORTF_IEV_R &= ~INT_PF4;     //PF4 is falling edge
54    GPIO_PORTF_ICR_R |= INT_PF4;      //Clear interrupt flag for PF4
55    GPIO_PORTF_IM_R |= INT_PF4;       //Enable interrupt on PF4
56
57    NVIC_PRI7_R = ________________;  //Set PF4 priority 5
58    NVIC_EN0_R = NVIC_EN0_INT30;      //Enable interrupt 30 in NVIC
59    EnableInterrupts();
60
61 }
62
63 void Delay(unsigned long value){
64    unsigned long i = 0;
65
66    for(i=0; i< value; i++);
67 }
68
69 void GPIOPortF_Handler(void){
70    int j;
71    GPIO_PORTF_ICR_R = INT_PF4;
72
73    if(i==3)
74      i = 1;
75    else
76      i++;
77    for(j = 0; j < 2; j++)
78    {
79      GPIO_PORTF_DATA_R ^= 1<<i;
80      Delay (1000000);
81    }
82 }
83
84 int main(){
85    Init_INT_GPIO();
86    while(1)
87      {
88        WaitForInterrupt();
89      }
90 }
```