Experiment 7

Interrupts and ISR Programming

Interrupt Control in Cortex M4

TM4C123GH6PM 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 effcient 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 interrupt occurs and it is accepted by processor core, the corresponding interrupt service routine is executed. Starting address of each interrupt 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 or interrupt handler. The vector table is located at address 0x0000.0000 after reset.
Every exception/interrupt handler is located at the address obtained by multiplying the corresponding exception/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 $2 \times 4 =$ 0x00000008. Consult table 2-8 and 2-9 for the entries of the vector table from article **2.5.2 The Cortex-M4F Processor - Exception Types** of the datasheet.

Processor Interrupt Configuration

Global level configurations of interrupts are handled at processor level. These are handled by PRIMASK, FAULTMASK and BASEPRI registers. The default values of these registers are such that all interrupts are enabled. However, in case of critical code execution, that should not be interrupted, global interrupt enabling and disabling can be performed. For that purpose, either the PRIMASK or FAULTMASK register can be used. The PRIMASK register prevents activation of all exceptions with configurable priority. The FAULTMASK register prevents activation of all exceptions except for Non-Maskable Interrupt (NMI).

Base Priority Register

The flexibility in masking the interrupts is introduced by the BASEPRI register. The interrupt masking by the BASEPRI is performed depending on the current priority level configuration. Since the number of priority levels in TM4C123GH6PM is 8, it requires 3 bits to be used by the BASEPRI register (Figure 7.1).

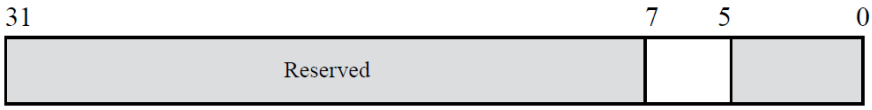


Figure 7.1: 3-bit implementation for the BASEPRI register

When BASEPRI is set to a non-zero value, it blocks all the interrupts of the same and lower priority, while it allows the processor to accept the interrupts of higher priority for processing. When BASEPRI is set to 0, it is disabled. These interrupt masking registers are accessed through special register access instructions. Move to special register from general-purpose register (MSR) and move special register to general-purpose register (MRS) assembly programming instructions are used for this purpose. **Add the following lines of code before the Reset_Handler in your startup file.**

```
        EXPORT  DisableInterrupts
        EXPORT  EnableInterrupts
        EXPORT  EnablePriorityInterrupts
        EXPORT  WaitForInterrupt

DisableInterrupts
        CPSID   I ; set PRIMASK to disable interrupts
        BX      LR

EnableInterrupts
        CPSIE   I ; clear PRIMASK to enable interrupts
        BX      LR

EnablePriorityInterrupts
        MOV     R0, #0x40 ; set base priority 2
        MSR     BASEPRI, R0
        BX      LR

WaitForInterrupt
        WFI
        BX      LR
```

NVIC Configuration

Note: *The register map for NVIC is to be consulted from article* ***3.2 - Cortex-M4 Peripherals - Register Map*** *of the controller datasheet.*

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

1. Enable the source from the corresponding NVIC enable register.
2. Set the priority for that interrupt.

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) corresponding to GPIO_Port_F (Figure 7.2).

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
45	29	0x0000.00B4	Flash Memory Control and EEPROM Control
46	30	0x0000.00B8	GPIO Port F
47-48	31-32	-	Reserved
49	33	0x0000.00C4	UART2
50	34	0x0000.00C8	SSI1

Interrupt Number

2. Find the interrupt register that is needed to enable IRQ30 from Table 3-8. 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 (Figure 7.3).

Nested Vectored Interrupt Controller (NVIC) Registers				
0x100	EN0	RW	0x0000.0000	Interrupt 0-31 Set Enable
0x104	EN1	RW	0x0000.0000	Interrupt 32-63 Set Enable
0x108	EN2	RW	0x0000.0000	Interrupt 64-95 Set Enable
0x10C	EN3	RW	0x0000.0000	Interrupt 96-127 Set Enable
0x110	EN4	RW	0x0000.0000	Interrupt 128-138 Set Enable

Interrupt Enable

3. From Table 3-8, find the register needed to set the priority of IRQ 30. It is NVIC_PRI7_R (Figure 7.4).

0x418	PRI6	RW	0x0000.0000	Interrupt 24-27 Priority
0x41C	PRI7	RW	0x0000.0000	Interrupt 28-31 Priority
0x420	PRI8	RW	0x0000.0000	Interrupt 32-35 Priority
0x424	PRI9	RW	0x0000.0000	Interrupt 36-39 Priority

Interrupt Priority

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 (Device) as Interrupt

Note: *The register map for GPIO is to be consulted from article* ***10.4 - General-Purpose Input/Outputs (GPIOs) - Register Map*** *of the controller datasheet.*

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 from GPIO Interrupt Mask register (GPIOIM) Figure 7.5.

GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x410
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								IME							
RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	RW	0x00	GPIO Interrupt Mask Enable

Value	Description
0	The interrupt from the corresponding pin is masked.
1	The interrupt from the corresponding pin is sent to the interrupt controller.

GPIO Interrupt Mask Register

2. Select whether the source of interrupt is edge-sensitive or level sensitive using GPIO Interrupt Sense register (GPIOIS) Figure 7.6.

GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x404
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								IS							
RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	RW	0x00	GPIO Interrupt Sense

Value	Description
0	The edge on the corresponding pin is detected (edge-sensitive).
1	The level on the corresponding pin is detected (level-sensitive).

GPIO Interrupt Sense Register

3. To enable interrupts for both edges write the appropriate value in the GPIO Interrupt

Both Edges register(GPIOIBE) Figure 7.7.

GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x408
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								IBE							
RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	RW	0x00	GPIO Interrupt Both Edges

Value	Description
0	Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 666).
1	Both edges on the corresponding pin trigger an interrupt.

GPIO Interrupt Both Edges Register

4. Write the appropriate value in GPIO Interrupt Event register (GPIOIEV) to configure the corresponding pin to detect rising (high level) or falling (low level) edges depending on the corresponding bit value in the GPIO Interrupt Sense (GPIOIS) register Figure 7.8.

GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x40C
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								IEV							
RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	RW	0x00	GPIO Interrupt Event

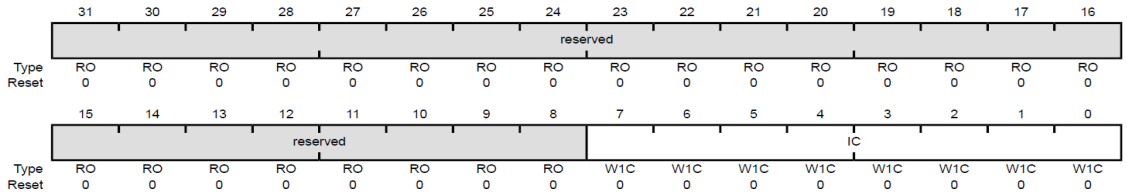
Value	Description
0	A falling edge or a Low level on the corresponding pin triggers an interrupt.
1	A rising edge or a High level on the corresponding pin triggers an interrupt.

GPIO Interrupt Event Register

5. Clear the interrupt flag for the corresponding pin by asserting the appropriate bit in the GPIO Interrupt Clear Register (GPIOICR) Figure 7.9.

6. Enable the interrupts by asserting the corresponding bits in GPIO Interrupt Mask register (GPIOIM) Figure 7.5.

GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x41C
Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IC	W1C	0x00	GPIO Interrupt Clear Value Description 0 The corresponding interrupt is unaffected. 1 The corresponding interrupt is cleared.

GPIO Interrupt Clear Register

Exercise

Configure an interrupt on SW1 (PF4) that is falling edge triggered and has a priority 5. Write a code that blinks LEDs of different colours on switch1 press. Template is provided on piazza. Complete the missing portions. Consult the datasheet where needed.