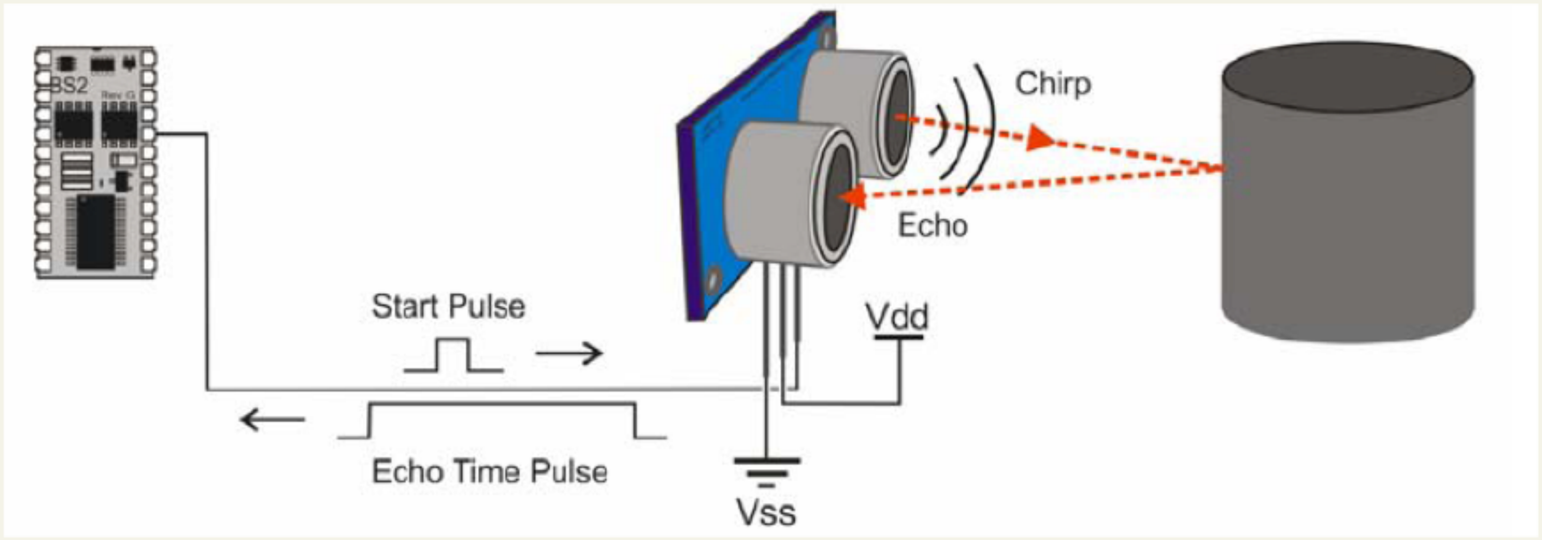
CprE 288 – Timer/Input Capture Interrupt Setup

Instructor:
Dr. Diane Rover
(Slides acknowledgment: Dr. Phillip Jones

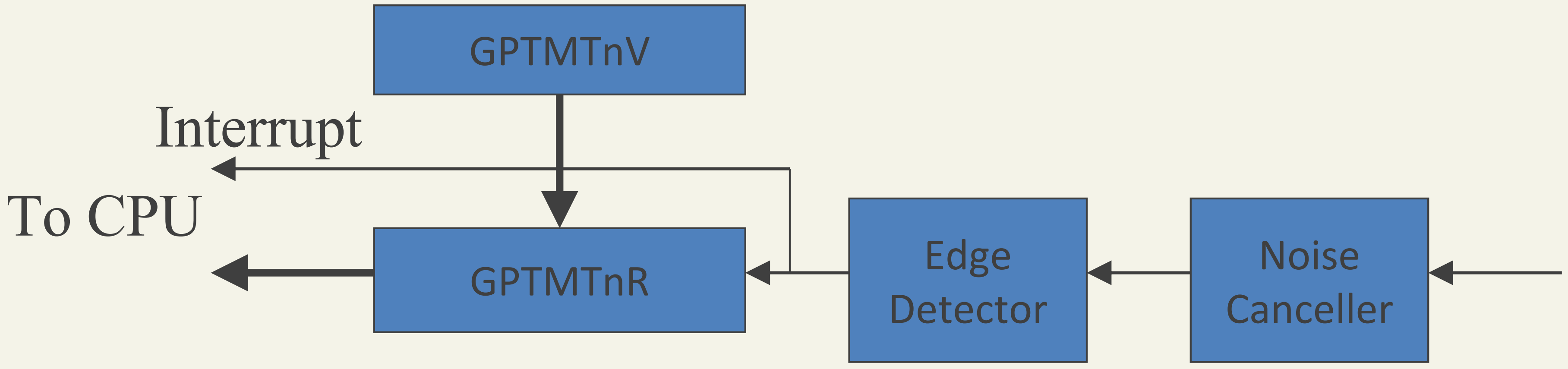
Application: Sonar Device

PING))) sensor: ultrasound distance detection device



Input Capture: Design Principle

Time value (clock count) is captured first, then read by the CPU



GPTMTnV: Timer n Value Register (n is A or B)
GPTMTnR: Timer Register (in Edge-Time mode, this register is loaded with the value in GPTMTnV at the last input edge event)

Input Capture: Design Principle

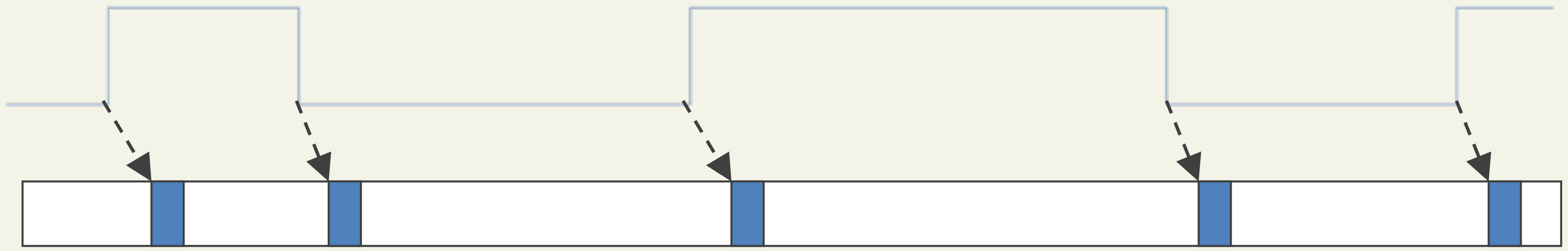
What happens in hardware and software when and after an event occurs:

- The event's time is *captured* in the GPTMTnR (timer register)
- An interrupt is raised to the CPU
- CPU executes the input capture ISR, which reads the timer register and completes the related processing

The captured time is *precise* because it's captured immediately when the event occurs.

The ISR should read the timer register and complete its processing fast enough to avoid loss of events.

Input Capture: Design Principle



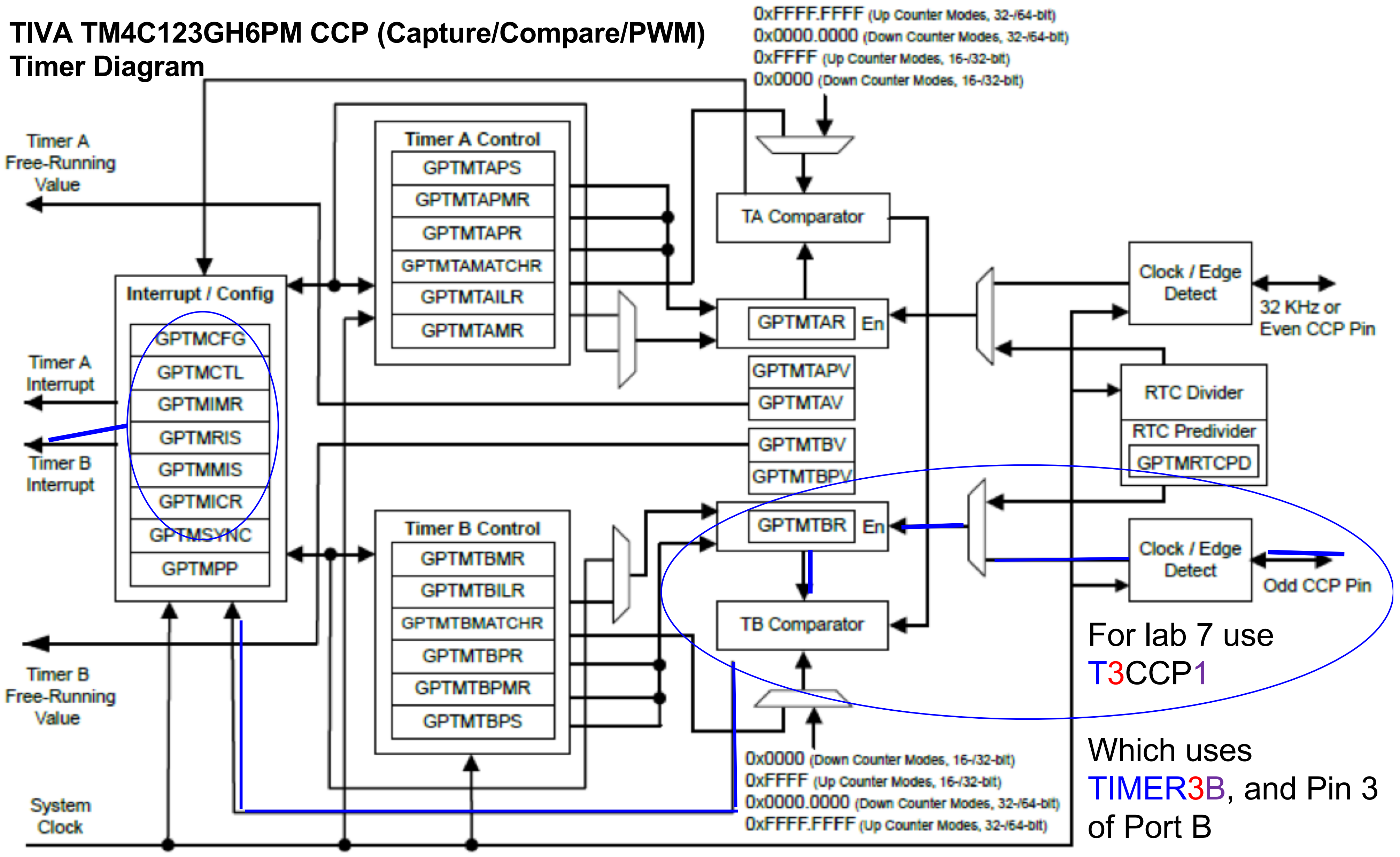
- -▶ Interrupt

🟦 CPU interrupt processing

⬜ CPU foreground computation

Figure 11-1. GPTM Module Block Diagram

TIVA TM4C123GH6PM CCP (Capture/Compare/PWM) Timer Diagram



For lab 7 use T3CCP1

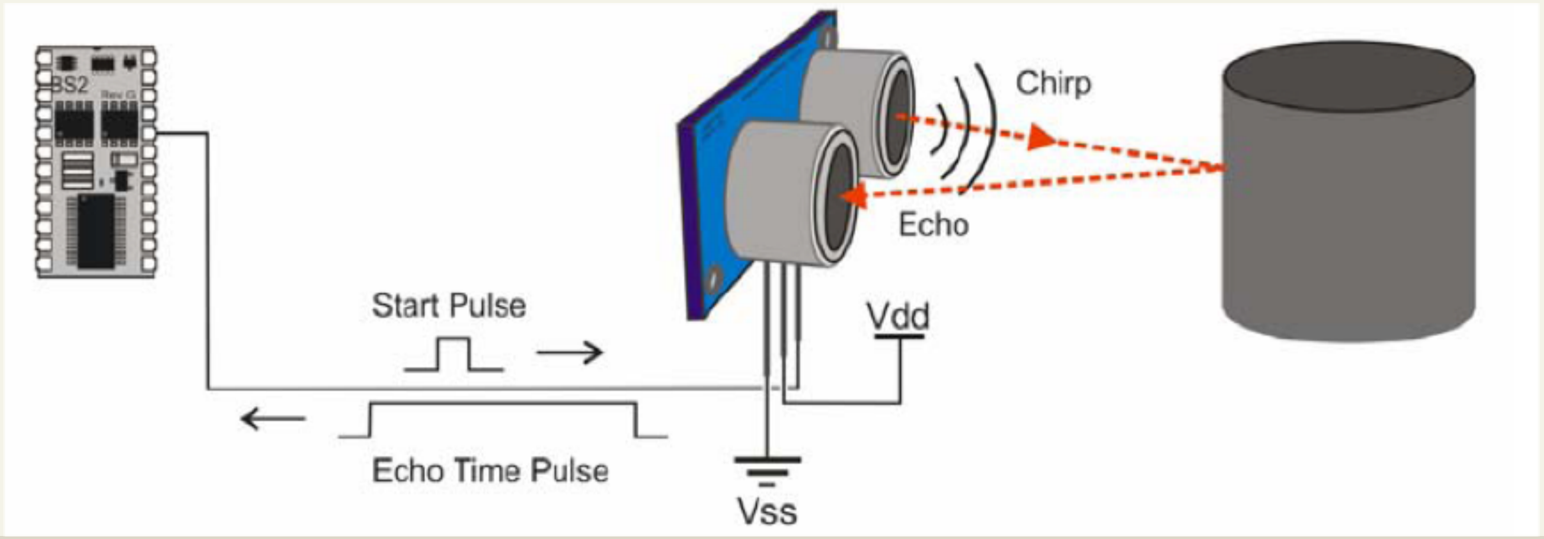
Which uses TIMER3B, and Pin 3 of Port B

(More on pages 704-707 of datasheet)

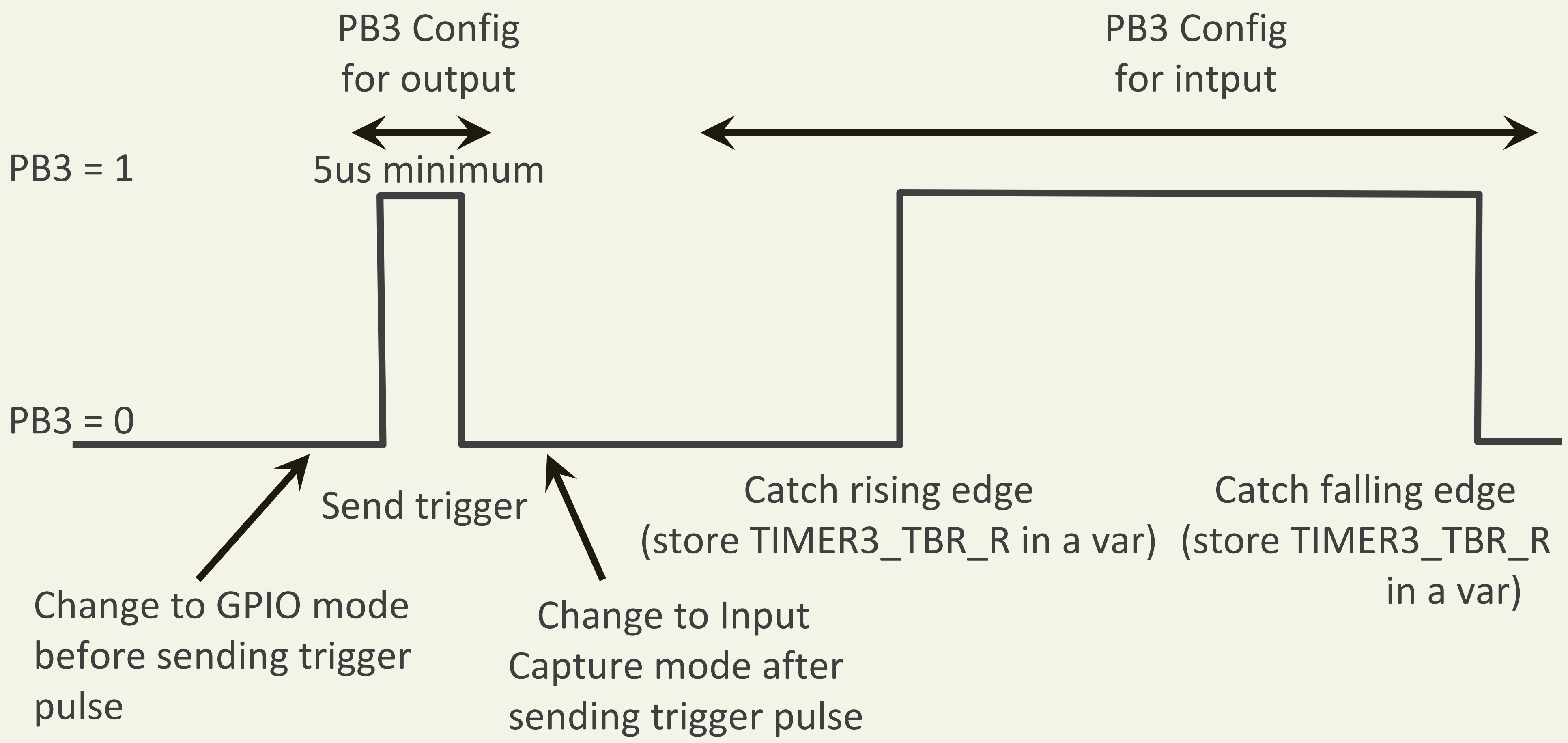
Lab 7 General Idea of Programming

General idea:

- Configure Timer3B for input capture
- Generate a pulse to activate the PING))) sensor
- Capture the time of rising edge event
- Capture the time of falling edge event
- Calculate time difference and then distance to object



Lab 7 General Idea of Programming



Remember only one pin (i.e PB3) used to communicate with the PING))) sensor

Timer Programming Interface - Interrupts

GPTMCTL: GPTM (General Purpose Timer) Control

GPTMCFG: GPTM Configuration

GPTMTnMR: GPTM Timer n Mode (n is A or B)

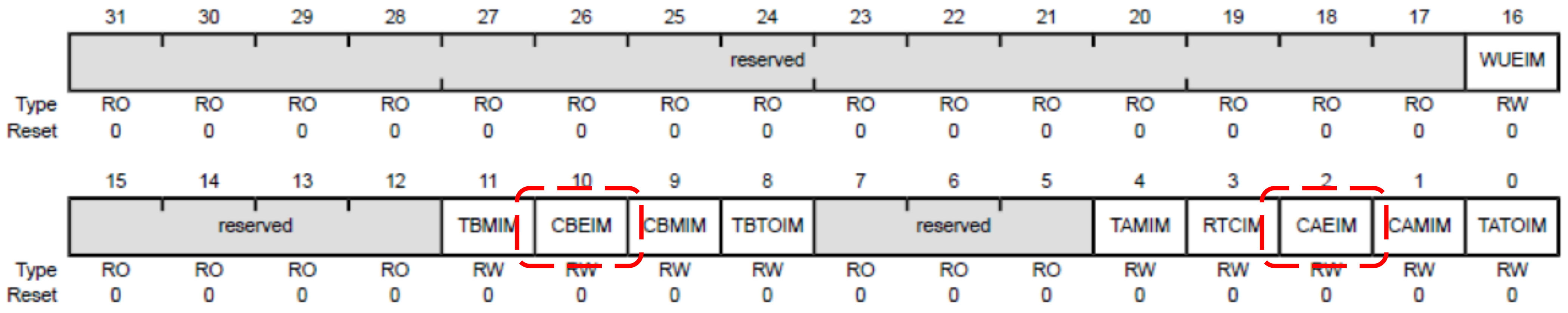
GPTMTnILR: GPTM Timer n Interval Load

GPTMIMR: GPTM Interrupt Mask Register

GPTMMIS: GPTM Masked Interrupt Status

GPTMICR: GPTM Interrupt Clear Register

GPTMIMR (TIMERx_IMR_R)



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								reserved								WUEIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	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			TBMIM	CBEIM	CBMIM	TBTOIM		reserved		TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
Type	RO	RO	RO	RO	RW	RW	RW	RW	RO	RO	RO	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For any interrupt in GPTMIMR write to the corresponding bit:

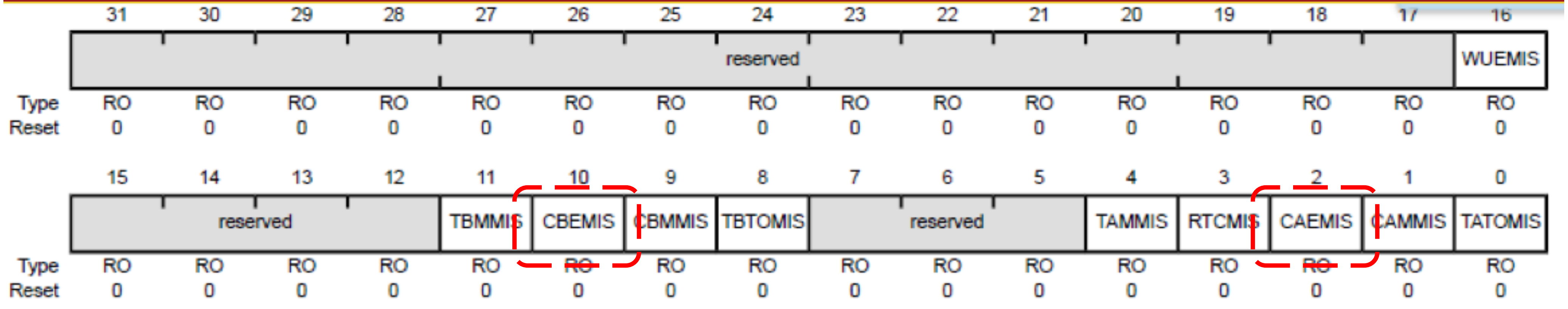
0 to disable the interrupt

1 to enable the interrupt

CnEIM: Timer n Capture Mode Event Interrupt Mask

(p. 745 of datasheet)

GPTMMIS (TIMERx_MIS_R)



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

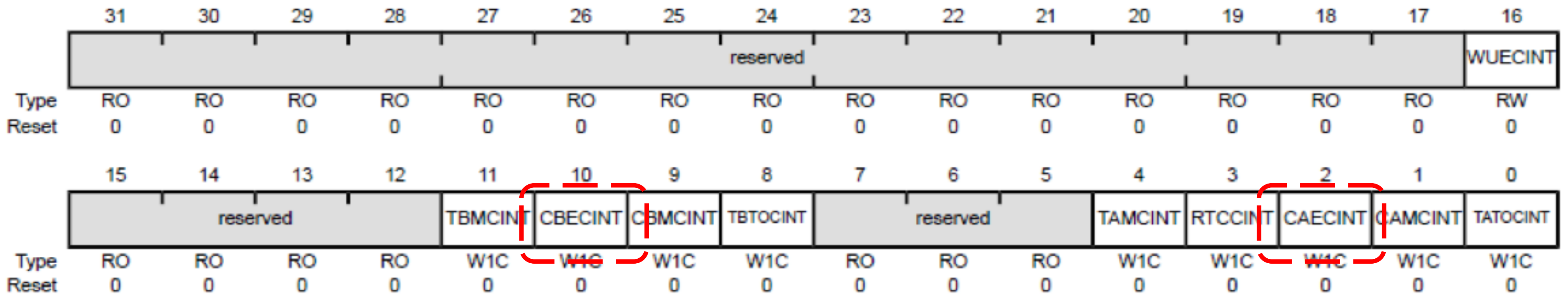
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBMMIS	CBEMIS	CBMMIS	TBTOMIS	reserved			TAMMIS	RTCMIS	CAEMIS	CAMMIS	TATOMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The **GPTMMIS** register shows the status of *unmasked* interrupts. Each timer n has 1 ISR vector that all of its interrupts trigger, so it is necessary to check the GPTMMIS register to see which specific interrupts were set (i.e., polled interrupts).

CnEMIS: Timer n, Capture Mode Event Flag

(p. 751 of datasheet)

GPTMICR (TIMERx_ICR_R)



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															WUECINT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	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				TBMCINT	CBECINT	CBMCINT	TBTOCINT	reserved			TAMCINT	RTCCINT	CAECINT	CAMCINT	TATOCINT
Type	RO	RO	RO	RO	W1C	W1C	W1C	W1C	RO	RO	RO	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

To clear an interrupt flag write a 1 to the corresponding bit in GPTMICR.

CnECINT: Clears the Timer n Capture Mode Event Flag

(p. 754 of datasheet)

Timer3B IC Interrupt Setup

```
// Disable Timer3B while setting it up…

// Assign various configuration settings...

// Clear capture interrupt flag

TIMER3_ICR_R |= 0x400; // CBECINT = bit 10, 0b100_0000_0000

// Enable event capture interrupts

TIMER3_IMR_R |= 0x400; // CBEIM = bit 10, 0b100_0000_0000

// Set up NVIC for Timer3B IC interrupts (next slide)…

//Bind Timer3B interrupt requests to your interrupt handler
IntRegister(INT_TIMER3B, TIMER3B_Handler);

// Enable Timer3B after setting it up…

// Enable global interrupts…
```

Timer3B IC NVIC Setup

- Set up NVIC for Timer3B IC interrupts
  - Set up the priority for T3CCP1 interrupts using the appropriate NVIC_PRIx_R register
  - Enable the interrupt at the NVIC level for T3CCP1 interrupts
- **See the following slides from the textbook, Chapter 5**

5.2.3.3 Definitions of the Priority Levels

- In TM4C123GH6PM MCU system, only the last **4** groups, from group priority **4** to group priority **7**, are available since only **3** bits (bits **7** ~ **5**) are used for the priority levels definitions in this system.

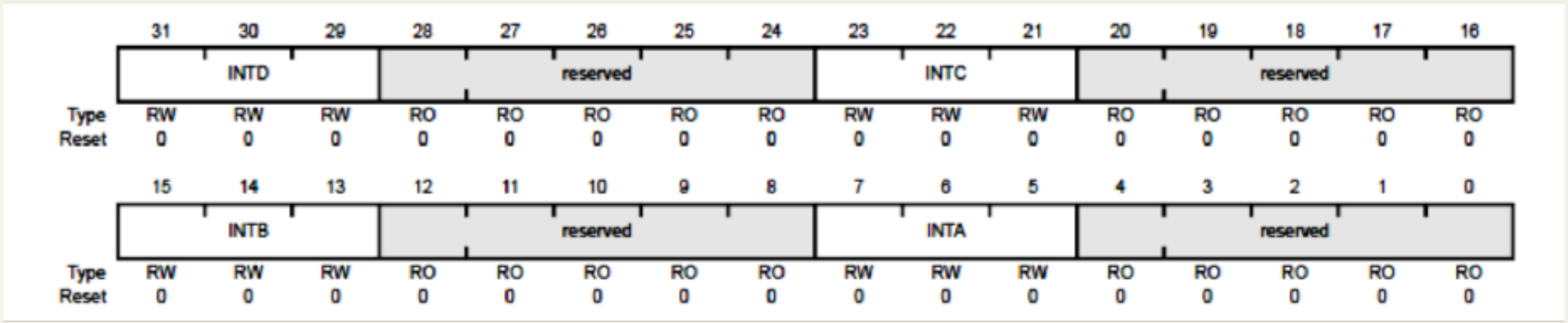
- The reason for using the **group priority** level is to determine whether an interrupt that has the same priority level as one that is currently being processed by the processor can be accepted.

Table 5.7 Definitions of group priority and sub-priority fields.

Priority Group	Group Priority Field	Sub-priority Field
0 (default)	Bits 7 ~ 1	Bit 0
1	Bits 7 ~ 2	Bits 1 ~ 0
2	Bits 7 ~ 3	Bits 2 ~ 0
3	Bits 7 ~ 4	Bits 3 ~ 0
4	Bits 7 ~ 5	Bits 4 ~ 0
5	Bits 7 ~ 6	Bits 5 ~ 0
6	Bit 7	Bits 6 ~ 0
7	None	Bits 7 ~ 0

5.3.2.1 The NVIC Interrupt Priority-Level Registers

- It can be found from Figure 5.6 that **PRI0** is a group **0** priority register with four segments:
  - **Segment 1** - Bits 7 ~ 5: GPIO Port A interrupt priority level control (INTA).
  - **Segment 2** - Bits 15 ~ 13: GPIO Port B interrupts priority level control (INTB).
  - **Segment 3** - Bits 23 ~ 21: GPIO Port C interrupts priority level control (INTC).
  - **Segment 4** - Bits 31 ~ 29: GPIO Port D interrupts priority level control (INTD).



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		INTD				reserved				INTC				reserved		
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	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
		INTB				reserved				INTA				reserved		
Type	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.6 An example of the NVIC Priority Level Register PRI0.

5.3.2.1 The NVIC Interrupt Priority-Level Registers

- Table 5.10 shows relationships between interrupt, its handler and its priority bits.

Table 5.10 The relationship between vectors and NVIC definitions.

Vector Address	Exception Number	IRQ Number	ISR Name in Startup_TM4C123.s	NVIC Macros for Priority Register	Priority Bits
0x00000038	14	-2	PendSV_Handler	NVIC_SYS_PRI3_R	23 - 21
0x0000003C	15	-1	SysTick_Handler	NVIC_SYS_PRI3_R	31 - 29
0x00000040	16	0	GPIOA_Handler	NVIC_PRI0_R	7 - 5
0x00000044	17	1	GPIOB_Handler	NVIC_PRI0_R	15 - 13
0x00000048	18	2	GPIOC_Handler	NVIC_PRI0_R	23 - 21
0x0000004C	19	3	GPIOD_Handler	NVIC_PRI0_R	31 - 29
0x00000050	20	4	GPIOE_Handler	NVIC_PRI1_R	7 – 5
0x00000054	21	5	UART0_Handler	NVIC_PRI1_R	15 – 13
0x00000058	22	6	UART1_Handler	NVIC_PRI1_R	23 – 21
0x0000005C	23	7	SSI0_Handler	NVIC_PRI1_R	31 – 29
0x00000060	24	8	I2C0_Handler	NVIC_PRI2_R	7 – 5
0x00000064	25	9	PWM0_Fault_Handler	NVIC_PRI2_R	15 – 13
0x00000068	26	10	PWM0_0_Handler	NVIC_PRI2_R	23 – 21
0x0000006C	27	11	PWM0_1_Handler	NVIC_PRI2_R	31 – 29
0x00000070	28	12	PWM0_2_Handler	NVIC_PRI3_R	7 – 5
0x00000074	29	13	QEI0_Handler	NVIC_PRI3_R	15 – 13
0x00000078	30	14	ADC0SS0_Handler	NVIC_PRI3_R	23 – 21
0x0000007C	31	15	ADC0SS1_Handler	NVIC_PRI3_R	31 – 29
0x00000080	32	16	ADC0SS2_Handler	NVIC_PRI4_R	7 – 5
0x00000084	33	17	ADC0SS3_Handler	NVIC_PRI4_R	15 – 13
0x00000088	34	18	WDT0_Handler	NVIC_PRI4_R	23 – 21
0x0000008C	35	19	TIMER0A_Handler	NVIC_PRI4_R	31 – 29
0x00000090	36	20	TIMER0B_Handler	NVIC_PRI5_R	7 – 5
0x00000094	37	21	TIMER1A_Handler	NVIC_PRI5_R	15 – 13
0x00000098	38	22	TIMER1B_Handler	NVIC_PRI5_R	23 – 21
0x0000009C	39	23	TIMER2A_Handler	NVIC_PRI5_R	31 – 29

5.3.2.1 The NVIC Interrupt Priority-Level Registers

- Table 5.10 shows relationships between interrupt, its handler and its priority bits.

Table 5.10 The relationship between vectors and NVIC definitions.

Vector Address	Exception Number	IRQ Number	ISR Name in Startup_TM4C123.s	NVIC Macros for Priority Register	Priority Bits
0x000000A0	40	24	TIMER2B_Handler	NVIC_PRI6_R	7 – 5
0x000000A4	41	25	COMP0_Handler	NVIC_PRI6_R	15 – 13
0x000000A8	42	26	COMP1_Handler	NVIC_PRI6_R	23 – 21
0x000000AC	43	27	COMP2_Handler	NVIC_PRI6_R	31 – 29
0x000000B0	44	28	SYSCTL_Handler	NVIC_PRI7_R	7 – 5
0x000000B4	45	29	FLASH_Handler	NVIC_PRI7_R	15 – 13
0x000000B8	46	30	GPIOF_Handler	NVIC_PRI7_R	23 – 21
0x000000BC	47	31	GPIOG_Handler	NVIC_PRI7_R	31 – 29
0x000000C0	48	32	GPIOH_Handler	NVIC_PRI8_R	7 – 5
0x000000C4	49	33	UART2_Handler	NVIC_PRI8_R	15 – 13
0x000000C8	50	34	SSI1_Handler	NVIC_PRI8_R	23 – 21
0x000000CC	51	35	TIMER3A_Handler	NVIC_PRI8_R	31 – 29
0x000000D0	52	36	TIMER3B_Handler	NVIC_PRI9_R	7 – 5
0x000000D4	53	37	I2C1_Handler	NVIC_PRI9_R	15 – 13
0x000000D8	54	38	QEI1_Handler	NVIC_PRI9_R	23 – 21
0x000000DC	55	39	CAN0_Handler	NVIC_PRI9_R	31 – 29
0x000000E0	56	40	CAN1_Handler	NVIC_PRI10_R	7 – 5
0x000000E4	57	41	CAN2_Handler	NVIC_PRI10_R	15 – 13
0x000000E8	58	42	0 (Reserved)	NVIC_PRI10_R	23 – 21
0x000000EC	59	43	HIB_Handler	NVIC_PRI10_R	31 – 29
0x000000F0	60	44	USB0_Handler	NVIC_PRI11_R	7 – 5
0x000000F4	61	45	PWM0_3_Handler	NVIC_PRI11_R	15 – 13
0x000000F8	62	46	UDMA_Handler	NVIC_PRI11_R	23 – 21
0x000000FC	63	47	UDMAERR_Handler	NVIC_PRI11_R	31 – 29

5.3.2.1 The NVIC Interrupt Priority-Level Registers

- When using Table 5.10 to build the user’s program to access **NVIC** registers directly to configure and setup related exceptions and interrupts, the following points should be noted:

  1. The **third column** in **Table 5.10** lists the interrupt numbers or **IRQ numbers** for all interrupts.

     Note that the interrupts that have **IRQ numbers 0** ~ **3** (GPIO Ports A ~ D) are controlled by the same priority level register (group **0**) **NVIC_PRI0**, and this means that each group **n** register **NVIC_PRIn** can control up to **4** peripherals (priority levels) with the interrupt numbers from **4n** to (**4n + 3**).

     This relationship is shown in **Table 5.11**. The related priority bits for each **PRIn** are shown in **column 6** on **Table 5.10**.

     These **IRQ numbers** are closely related to the associated peripherals to be enabled by using five NVIC Interrupt Enable Registers, **NVIC_EN0_R** ~ **NVIC_EN4_R**. **Table 5.13** shows the relationship between each bit on these five registers and each related peripheral to be enabled.

5.3.2.1 The NVIC Interrupt Priority-Level Registers

Table 5.11 The bit filed of priority levels and related interrupt priority group.

PRIn Register Bit Field	Interrupt Source	Priority Register Macros
Bits 31:29	Interrupt[IRQ] = Interrupt[4n + 3]	NVIC_PRI**n**_R NVIC→IP[4**n**] ~ NVIC→IP[4**n** + 3]
Bits 23:21	Interrupt[IRQ] = Interrupt[4n + 2]	
Bits 15:13	Interrupt[IRQ] = Interrupt[4n + 1]	
Bits 7:5	Interrupt[IRQ] = Interrupt[4n]	

Table 5.12 Most popular Priority Registers used in the TM4C123GH6PM NVIC.

Priority Register	Priority Bits				Address
	31 - 29	23 - 21	15 - 13	7 - 5	
NVIC_PRI0_R	GPIO Port D	GPIO Port C	GPIO Port B	GPIO Port A	0xE000E400
NVIC_PRI1_R	SSI0, Rx Tx	UART1, Rx Tx	UART0, Rx Tx	GPIO Port E	0xE000E404
NVIC_PRI2_R	PWM Gen 1	PWM Gen 0	PWM Fault	I2C0	0xE000E408
NVIC_PRI3_R	ADC Seq 1	ADC Seq 0	Quad Encoder	PWM Gen 2	0xE000E40C
NVIC_PRI4_R	Timer 0A	Watchdog	ADC Seq 3	ADC Seq 2	0xE000E410
NVIC_PRI5_R	Timer 2A	Timer 1B	Timer 1A	Timer 0B	0xE000E414
NVIC_PRI6_R	Comp 2	Comp 1	Comp 0	Timer 2B	0xE000E418
NVIC_PRI7_R	GPIO Port G	GPIO Port F	Flash Control	System Control	0xE000E41C
NVIC_PRI8_R	Timer 3A	SSI1, Rx Tx	UART2, Rx Tx	GPIO Port H	0xE000E420
NVIC_PRI9_R	CAN0	Quad Encoder 1	I2C1	Timer 3B	0xE000E424
NVIC_PRI10_R	Hibernate	Ethernet	CAN2	CAN1	0xE000E428
NVIC_PRI11_R	uDMA Error	uDMA Soft Tfr	PWM Gen 3	USB0	0xE000E42C
NVIC_SYS_PRI3_R	SysTick	PendSV	--	Debug	0xE000ED20

5.3.2.1 The NVIC Interrupt Priority-Level Registers

- Tables **5.10**, **5.11** and **5.12** can be used together to get a clear picture about the peripheral devices, their interrupt numbers, their interrupt vectors, their interrupt handlers and their related priority registers and bit fields.
- For example, the **ADC2_Handler** that is corresponding to the Analog-to-Digital Converter 2 peripheral in Table 5.10, and the **IRQ number** for this peripheral is **16**.
- From Table 5.11, it can be found that in order to make the equation Interrupt[4**n]** = 16 hold, the group number of the priority register, **n** = **4**. Therefore the corresponded Priority Level Register should be NVIC_PRI**4**_R and it can configure priority levels for 4**n** ~ 4**n** + 3 peripherals whose IRQ numbers are ranged 4 × 4 ~ 4 × 4 + 3 = 16 ~ 19.
- It can be found from Table 5.10 that the peripherals whose IRQ numbers are **16** ~ **19** are: **ADC2_Handler, ADC3_Handler, WDT0_Handler** and **TIMER0A_Handler**, respectively.
- Take a look at Table 5.12, the Priority Level Register **NVIC_PRI4_R** that is located at row **5** can exactly handle the priority levels for these four peripherals in 4 segments, bits 7 ~ 5, bits 15 ~ 13, bits 23 ~ 21 and bits 31 ~ 29.

5.3.2.2 The NVIC Interrupt Set Enable Registers

- The NVIC provides five (5) Interrupt Set Enable Registers, **EN0** ~ **EN4**. The symbolic definitions for these registers are: **NVIC_EN0_R** ~ **NVIC_EN4_R**.

- All these five Set Enable Registers are 32-bit registers. Except **EN4**, all other Set Enable Registers use the full length, 32-bit, to enable related peripherals. These registers, **EN0** ~ **EN4**, use single bit, or bit by bit, to enable related peripheral with the following functions:
  - **NVIC_EN0_R**: Provides 32 bit-by-bit enable control ability to 32 peripherals whose IRQ numbers are 0 ~ 31 (see Table 5.10 for IRQ numbers). This means that bit-0 controls the peripheral whose IRQ number is 0 (GPIO Port A), bit-1 controls the peripheral whose IRQ number is 1 (GPIO Port B), and bit-31 controls the peripheral whose IRQ number is 31 (GPIO Port G).
  - **NVIC_EN1_R**: Provides 32 bit-by-bit enable control ability to 32 peripherals whose IRQ numbers are 32 ~ 63.
  - **NVIC_EN2_R**: Provides 32 bit-by-bit enable control ability to 32 peripherals whose IRQ numbers are 64 ~ 95.
  - **NVIC_EN3_R**: Provides 32 bit-by-bit enable control ability to 32 peripherals whose IRQ numbers are 96 ~ 127.
  - **NVIC_EN4_R**: Provides 11 bit-by-bit enable control ability to 10 peripherals whose IRQ numbers are 128 ~ 138.

5.3.2.2 The NVIC Interrupt Set Enable Registers

- These registers are only used to set enables to related peripherals by writing 1 to the related bits. Writing zeros to any bits has no effects. To disable any peripherals, one needs to use the corresponding bits in the NVIC Clear Enable Registers, **NVIC_DIS0_R** through **NVIC_DIS4_R**.
- Each bit on the **Set Enable Register** is to enable one peripheral, and the bit number is equal to the IRQ number of the peripheral to be enabled.
- For example, to enable the GPIO **Port F** whose IRQ number is **30** (Table 5.10), the bit-**30** on the **NVIC_EN0_R** should be set to 1. To enable **UART2** whose IRQ number is **33**, the bit-1 on the **NVIC_EN1_R** should be set to 1. Since each **NVIC_ENn_R** register can only handle 32 peripherals (0 ~ 31), if the IRQ number is greater than 31, the target bit number should be calculated as:
  - Target bit number = **IRQ number** – 32 × (**n** – 1).
    - **n** = 1 if IRQ < 31,
    - **n** = 2 if (64 > IRQ > 31),
    - **n** = 3 if (96 > IRQ > 63), **n** = 4 if (128 > IRQ > 95).

5.3.2.2 The NVIC Interrupt Set Enable Registers

- Table 5.13 lists most popular used peripherals and their bit numbers in the **NVIC Set Enable Registers**. Each bit is associated with a peripheral and a setting to a bit enable the selected peripheral. Each 32-bit Set Enable Register can be used enable 32 peripherals.

- During the programming process, users can directly use various macros defined for all Set Enable Registers, **NVIC_EN0_R** ~ **NVCI_EN4_R**, in their program to access them to perform the enable configurations for selected peripherals.

Table 5.13 Relationship between each bit on interrupt enable register and related peripheral.

Enable Register	32 Enable Bits									Address
	0	1	2	3	4	5	6 - 29	30	31	
NVIC_EN0_R	PORTA	PORTB	PORTC	PORTD	PORTE	UART0	......	PORTF	PORTG	0xE000E100
NVIC_EN1_R	PORTH	UART2	SSI1	Timer3A	Timer3B	I2C1	......	UART6	UART7	0xE000E104
NVIC_EN2_R	...	...	...	...	I2C2	I2C3	......	WTimer0A	WTimer0B	0xE000E108
NVIC_EN3_R	WT1A	WT1B	WT2A	WT2B	WT3A	WT3B	......	GPIOQ2	GPIOQ3	0xE000E10C

Timer3B IC NVIC Setup

- Set up the priority for T3CCP1 interrupts using the appropriate NVIC_PRIx_R register (use priority = 1)

```
NVIC_PRI9_R |= 0x20; // set bits 7:5 to 0b001
```

- Enable the interrupt at the NVIC level for T3CCP1 interrupts

```
NVIC_EN1_R |= 0x10; // set bit 4
```

IC Programming Example

```
volatile enum {LOW, HIGH, DONE} state;
volatile unsigned int rising_time;     // start time of the return pulse
volatile unsigned int falling_time;    // end time of the return pulse

// module initialization and configuration not shown

/* start and read the ping sensor once, return distance in cm */
unsigned ping_read()
{
  send_pulse();                 // send the starting pulse to PING

  // TODO get time of the rising edge of the pulse

  // TODO get time of the falling edge of the pulse

  // Calculate the width of the pulse; convert to centimeters

}
```

IC Programming Example

```
/* send a start pulse on PB3 */
void send_pulse()
{
   GPIO_PORTB_DIR_R |= 0x08;                    // set PB3 as output
   GPIO_PORTB_DATA_R |= 0x08;                   // set PB3 to high
   // wait at least 5 microseconds based on data sheet
   GPIO_PORTB_DATA_R &= 0xF7;                   // set PB3 to low
   GPIO_PORTB_DIR_R &= 0xF7;                    // set PB3 as input
}

/* convert time in clock counts to single-trip distance in cm */
unsigned time2dist(unsigned time)
{
   …
}
```

IC Programming Example

```
// ISR: Record the current event time
void TIMER3B_Handler(void)
{

// check for interrupt event and clear capture interrupt flag if triggered

// depending on current status … rising_time = TIMER3_TBR_R;

// depending on current status … falling_time = TIMER3_TBR_R;

// update status

}
```

5.3.4 The Vector Table and Vectors Used in the TM4C123GH6PM MCU

```
__Vectors       DCD     __initial_sp            ; Top of Stack
                DCD     Reset_Handler           ; Reset Handler
                DCD     NMI_Handler             ; NMI Handler
                DCD     HardFault_Handler       ; Hard Fault Handler
                DCD     MemManage_Handler       ; MPU Fault Handler
                DCD     BusFault_Handler        ; Bus Fault Handler
                DCD     UsageFault_Handler      ; Usage Fault Handler
                DCD     0                       ; Reserved
                DCD     0                       ; Reserved
                DCD     0                       ; Reserved
                DCD     0                       ; Reserved
                DCD     SVC_Handler             ; SVCall Handler
                DCD     DebugMon_Handler        ; Debug Monitor Handler
                DCD     0                       ; Reserved
                DCD     PendSV_Handler          ; PendSV Handler
                DCD     SysTick_Handler         ; SysTick Handler

; External Interrupts   Vector or Handler       ; IRQ#   Peripheral  ----------------

                DCD     GPIOA_Handler           ;  0:    GPIO Port A
                DCD     GPIOB_Handler           ;  1:    GPIO Port B
                DCD     GPIOC_Handler           ;  2:    GPIO Port C
                DCD     GPIOD_Handler           ;  3:    GPIO Port D
                DCD     GPIOE_Handler           ;  4:    GPIO Port E
                DCD     UART0_Handler           ;  5:    UART0 Rx and Tx
                DCD     UART1_Handler           ;  6:    UART1 Rx and Tx
                DCD     SSI0_Handler            ;  7:    SSI0 Rx and Tx
                DCD     I2C0_Handler            ;  8:    I2C0 Master and Slave
                DCD     PMW0_FAULT_Handler      ;  9:    PWM Fault
                DCD     PWM0_0_Handler          ; 10:    PWM Generator 0
                DCD     PWM0_1_Handler          ; 11:    PWM Generator 1
                DCD     PWM0_2_Handler          ; 12:    PWM Generator 2
                DCD     QEI0_Handler            ; 13:    Quadrature Encoder 0
                DCD     ADC0SS0_Handler         ; 14:    ADC Sequence 0
```

Figure 5.7 The vector definitions in the Cortex-M4 microcontroller

5.3.4 The Vector Table and Vectors Used in the TM4C123GH6PM MCU

```
; External Interrupts   Vector or Handler        ; IRQ#  Peripheral
               DCD     ADC0SS1_Handler           ; 15:   ADC Sequence 1
               DCD     ADC0SS2_Handler           ; 16:   ADC Sequence 2
               DCD     ADC0SS3_Handler           ; 17:   ADC Sequence 3
               DCD     WDT0_Handler              ; 18:   Watchdog timer
               DCD     TIMER0A_Handler           ; 19:   Timer 0 subtimer A
               DCD     TIMER0B_Handler           ; 20:   Timer 0 subtimer B
               DCD     TIMER1A_Handler           ; 21:   Timer 1 subtimer A
               DCD     TIMER1B_Handler           ; 22:   Timer 1 subtimer B
               DCD     TIMER2A_Handler           ; 23:   Timer 2 subtimer A
               DCD     TIMER2B_Handler           ; 24:   Timer 2 subtimer B
               DCD     COMP0_Handler             ; 25:   Analog Comparator 0
               DCD     COMP1_Handler             ; 26:   Analog Comparator 1
               DCD     COMP2_Handler             ; 27:   Analog Comparator 2
               DCD     SYSCTL_Handler            ; 28:   System Control (PLL, OSC, BO)
               DCD     FLASH_Handler             ; 29:   FLASH Control
               DCD     GPIOF_Handler             ; 30:   GPIO Port F
               DCD     GPIOG_Handler             ; 31:   GPIO Port G
               DCD     GPIOH_Handler             ; 32:   GPIO Port H
               DCD     UART2_Handler             ; 33:   UART2 Rx and Tx
               DCD     SSI1_Handler              ; 34:   SSI1 Rx and Tx
               DCD     TIMER3A_Handler           ; 35:   Timer 3 subtimer A
               DCD     TIMER3B_Handler           ; 36:   Timer 3 subtimer B
               DCD     I2C1_Handler              ; 37:   I2C1 Master and Slave
               DCD     QEI1_Handler              ; 38:   Quadrature Encoder 1
               DCD     CAN0_Handler              ; 39:   CAN0
               DCD     CAN1_Handler              ; 40:   CAN1
               DCD     CAN2_Handler              ; 41:   CAN2
               DCD     0                         ; 42:   Reserved
               DCD     HIB_Handler               ; 43:   Hibernate
               DCD     USB0_Handler              ; 44:   USB0
```

Figure 5.7 The vector definitions in the Cortex-M4 microcontroller