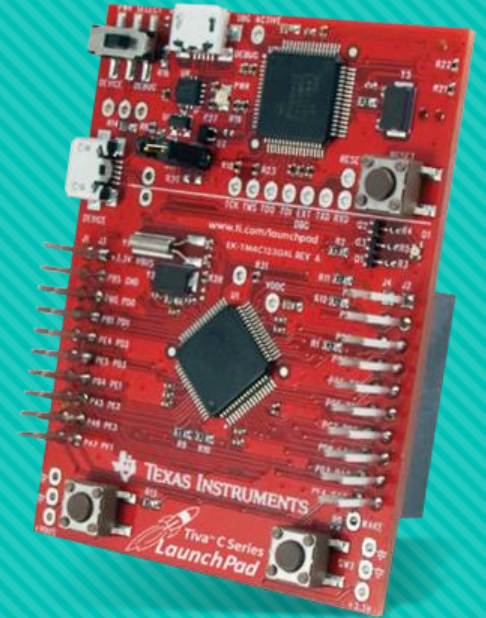


ARM-Based Microcontrollers

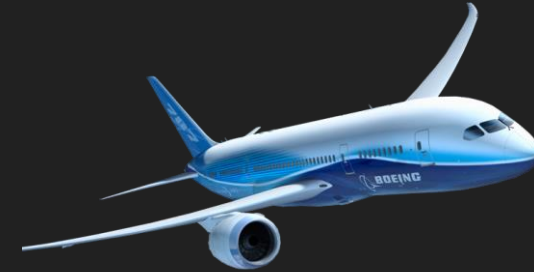
TM4C123GH6PM Interrupts & Timers PERIPHERALS



By : Ahmed Magdy

AGENDA

- Recap
- How to read the datasheet
- Interrupts Controller in ARM
- Interrupts Procedures
- Interrupt Priorities & Latency
- Types of Timers
- Timer PERIPHERAL HW DESIGN
- Timer CONFIGURATION
- Timer PERIPHERAL DRIVER SW DESIGN



How to read the datasheet

○ Interrupts in TM4C123gh6PM Datasheet (Page 104)

Revision History

About This Document

1. Architectural Overview

2. The Cortex-M4F Processor

2.1. Block Diagram

2.2. Overview

2.3. Programming Model

2.4. Memory Model

2.5. Exception Model

2.5.1. Exception States

2.5.2. Exception Types

2.5.3. Exception Handlers

2.5.4. Vector Table

2.5.5. Exception Priorities

2.5.6. Interrupt Priority Grouping

2.5.7. Exception Entry and Return

2.6. Fault Handling

2.7. Power Management

2.8. Instruction Set Summary

3. Cortex-M4 Peripherals

3.1. Functional Description

3.2. Register Map

3.3. System Timer (SysTick) Register Descrip

3.4. NVIC Register Descriptions

3.5. System Control Block (SCB) Register De

c. See SYSPRI1 on page 170.

d. See PRIn registers on page 152.

Table 2-9. Interrupts

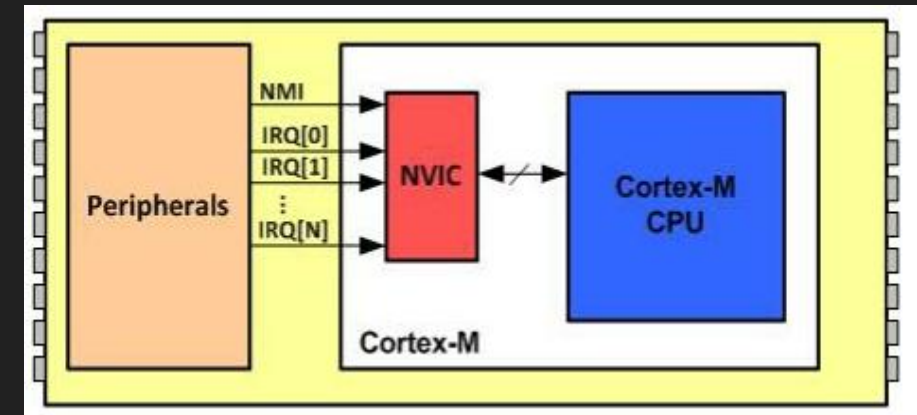
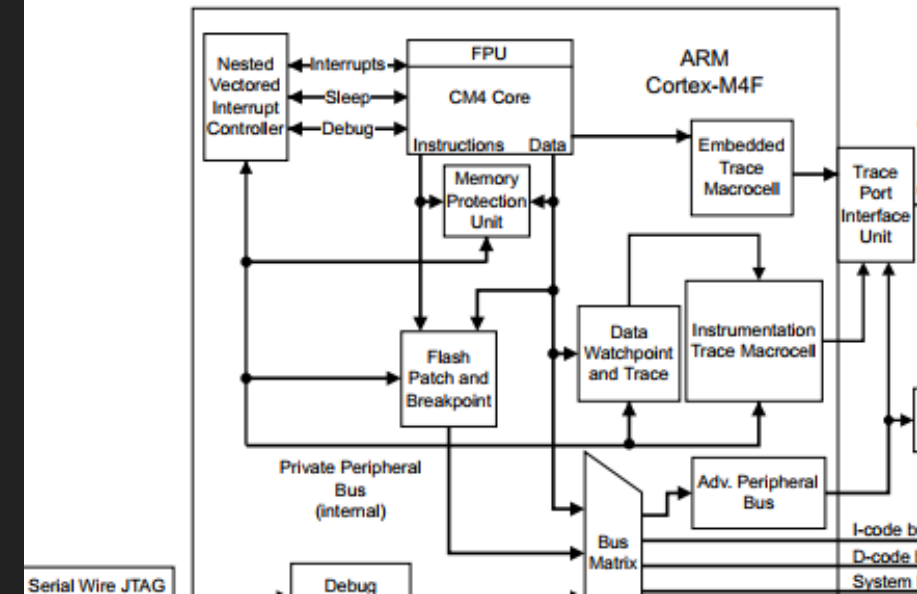
Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
0-15	-	0x0000.0000 - 0x0000.003C	Processor exceptions
16	0	0x0000.0040	GPIO Port A
17	1	0x0000.0044	GPIO Port B
18	2	0x0000.0048	GPIO Port C
19	3	0x0000.004C	GPIO Port D
20	4	0x0000.0050	GPIO Port E
21	5	0x0000.0054	UART0
22	6	0x0000.0058	UART1
23	7	0x0000.005C	SSI0
24	8	0x0000.0060	I ² C0
25	9	0x0000.0064	PWM0 Fault
26	10	0x0000.0068	PWM0 Generator 0
27	11	0x0000.006C	PWM0 Generator 1
28	12	0x0000.0070	PWM0 Generator 2
29	13	0x0000.0074	QEI0
30	14	0x0000.0078	ADC0 Sequence 0
31	15	0x0000.007C	ADC0 Sequence 1
32	16	0x0000.0080	ADC0 Sequence 2
33	17	0x0000.0084	ADC0 Sequence 3
34	18	0x0000.0088	Watchdog Timers 0 and 1
35	19	0x0000.008C	16/32-Bit Timer 0A
36	20	0x0000.0090	16/32-Bit Timer 0B
37	21	0x0000.0094	16/32-Bit Timer 1A

Interrupts Controller in ARM

Interrupts in TM4C123gh6PM Datasheet

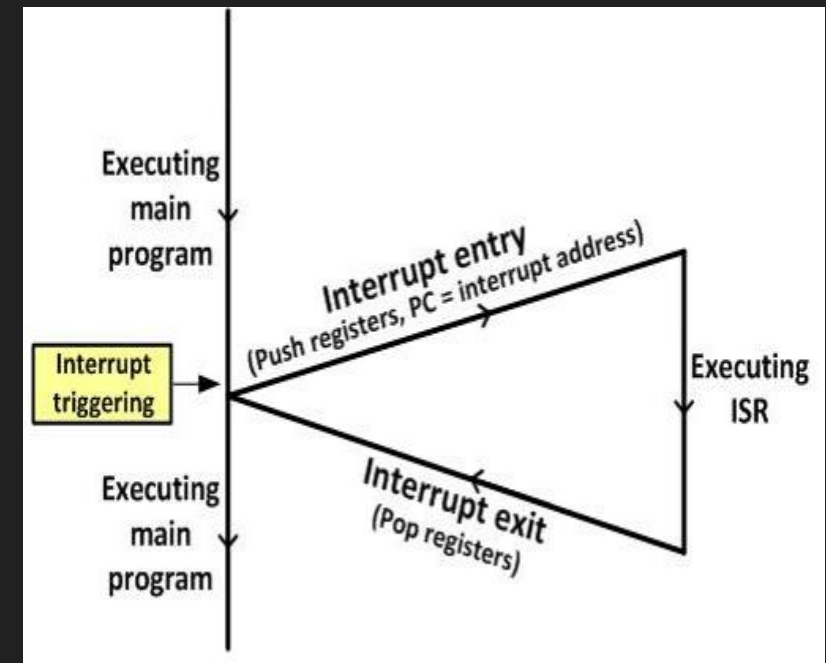
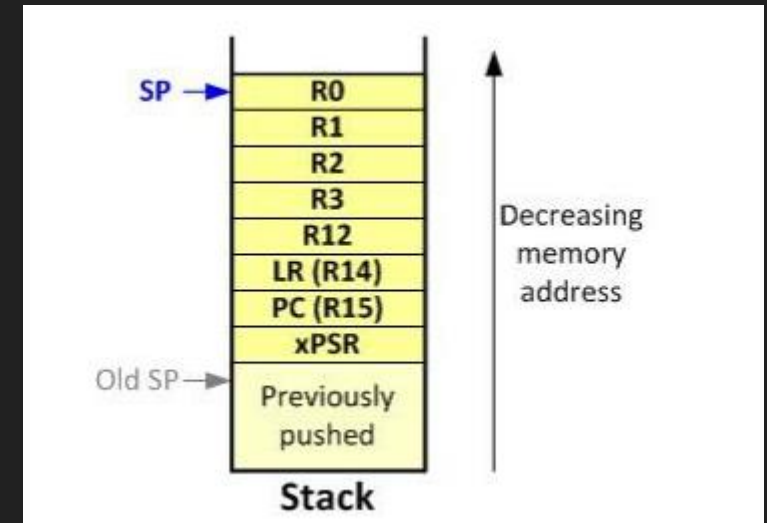
- The NVIC of the ARM Cortex-M has room for the total of 255 interrupts and exceptions.
- The predefined Interrupts (INT 0 to INT 15) like
 - **Stack Pointer** initial value
 - **Reset Vector** which has PC).
 - **Exceptions (Faults)** (Hard Fault for clock or registers, Memory Management from MPU unit, Memory Management like divide by zero).
 - **Non-Maskable Interrupts NMI**, A non-maskable Interrupt (NMI) can be signaled using the NMI signal or triggered by software using the Interrupt Control and State (INTCTRL) register. This exception has the highest priority other than reset.
- IRQ Peripheral interrupts (ADC, UART, Timer, Watchdog ...) are programmable by the Developer.

Figure 2-1. CPU Block Diagram



Interrupts Procedures

1. The Current processor status register (CPSR) is pushed onto the stack and SP is decremented by 4, since CPSR is a 4-byte register.
2. The current PC, LR (R14) and rest of CPU registers are pushed onto the stack.
3. Save Floating point coprocessor registers or move SP if lazy stacking is enabled.
4. LR is loaded with a number with bit 31-5 all 1s.
5. The INT number (type) is multiplied by 4 to get the address of the location within the vector table to fetch the program counter of the interrupt service routine (interrupt handler).
6. From the memory locations pointed to by this new PC, the CPU starts to fetch and execute instructions belonging to the ISR program.
7. When one of the return instructions is executed in the interrupt service routine, the CPU recognizes that it is in the Handler Mode from the value of the LR. It then restores the registers saved when entering ISR including the program counter from the stack



Interrupt Priorities & Latency

1. What happens if two interrupts want the attention of the CPU at the same time? Which has priority ?

Priority levels can be set by the programmer.

Programmable priority levels are values between 0 and 7 with **7 has the lowest priority**. (Check top of page 104)

Table 2-8. Exception Types

Exception Type	Vector Number	Priority ^a	Vector Address or Offset ^b	Activation
-	0	-	0x0000.0000	Stack top is loaded from the first entry of the vector table on reset
Reset	1	-3 (highest)	0x0000.0004	Asynchronous
Non-Maskable Interrupt (NMI)	2	-2	0x0000.0008	Asynchronous
Hard Fault	3	-1	0x0000.000C	-
Memory Management	4	programmable ^c	0x0000.0010	Synchronous
Bus Fault	5	programmable ^c	0x0000.0014	Synchronous when precise and asynchronous when imprecise
Usage Fault	6	programmable ^c	0x0000.0018	Synchronous
-	7-10	-	-	Reserved
SVCall	11	programmable ^c	0x0000.002C	Synchronous
Debug Monitor	12	programmable ^c	0x0000.0030	Synchronous
-	13	-	-	Reserved

Interrupt Priorities & Latency

○ Interrupt latency

The time from the moment the event that triggers an interrupt signal to the moment the CPU starts to execute the ISR code is called the interrupt latency.

Sources that Affects Latency:

- 1- whether the source of the interrupt is an internal (e.g., exceptions) or external hardware (e.g., peripheral hardware IRQ) interrupt.
- 2- Can also be affected by the type of the instruction which the CPU was executing when the interrupt occurs.
- 3- Another source of the interrupt latency is the interrupt priority .

Types of Timers

○ Uses of Timers :

1. Counting events
2. Making delays (Using Counter as a Timer)
3. Measuring the time between 2 events

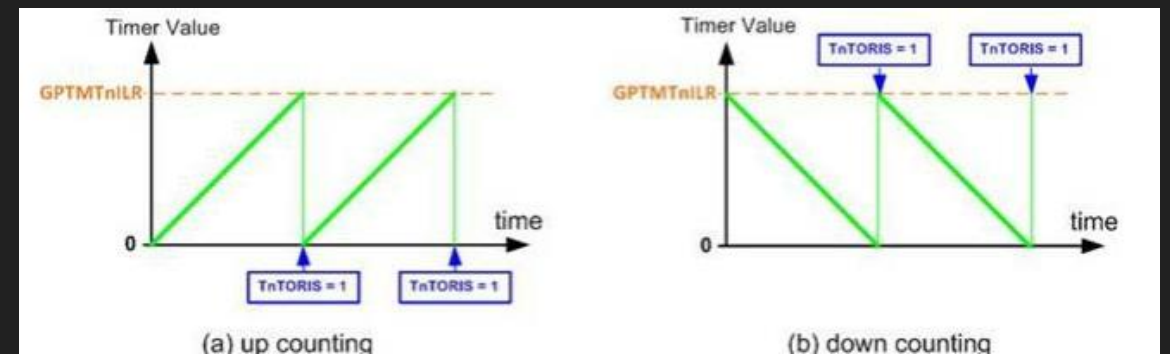
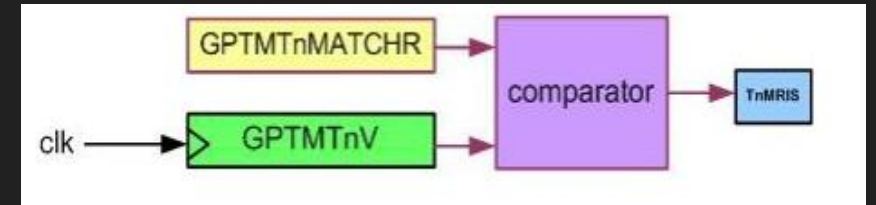
○ Types:

○ 1- Compare Match (One Shot mode, periodic Mode)

Timer keeps counting it is compared with the contents of this register. Whenever the contents of free-running TimerA counter and Timer A Match register are equal, the TAMRIS Flag goes up indicating there is a match.

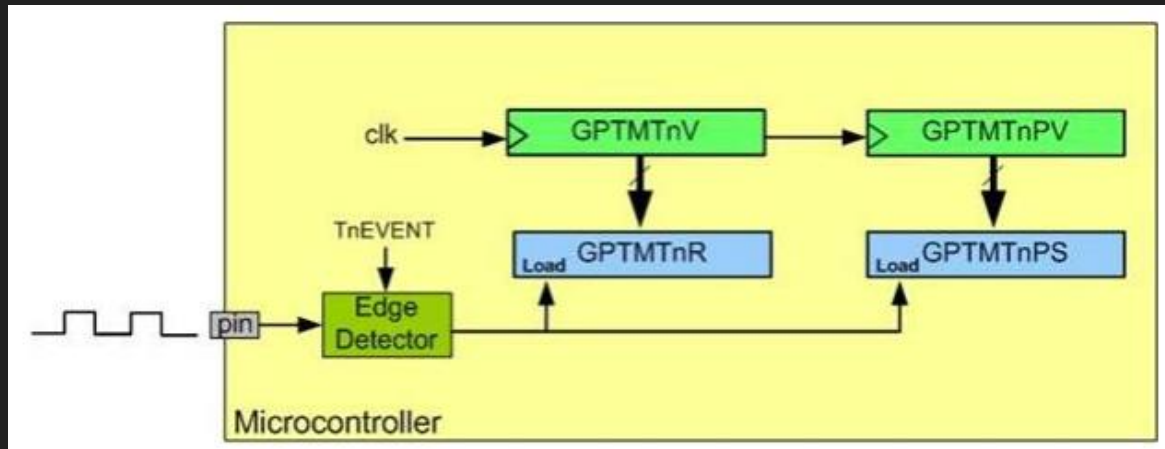
○ 2- Pulse Width Modulation PWM

○ 3- Input Capture

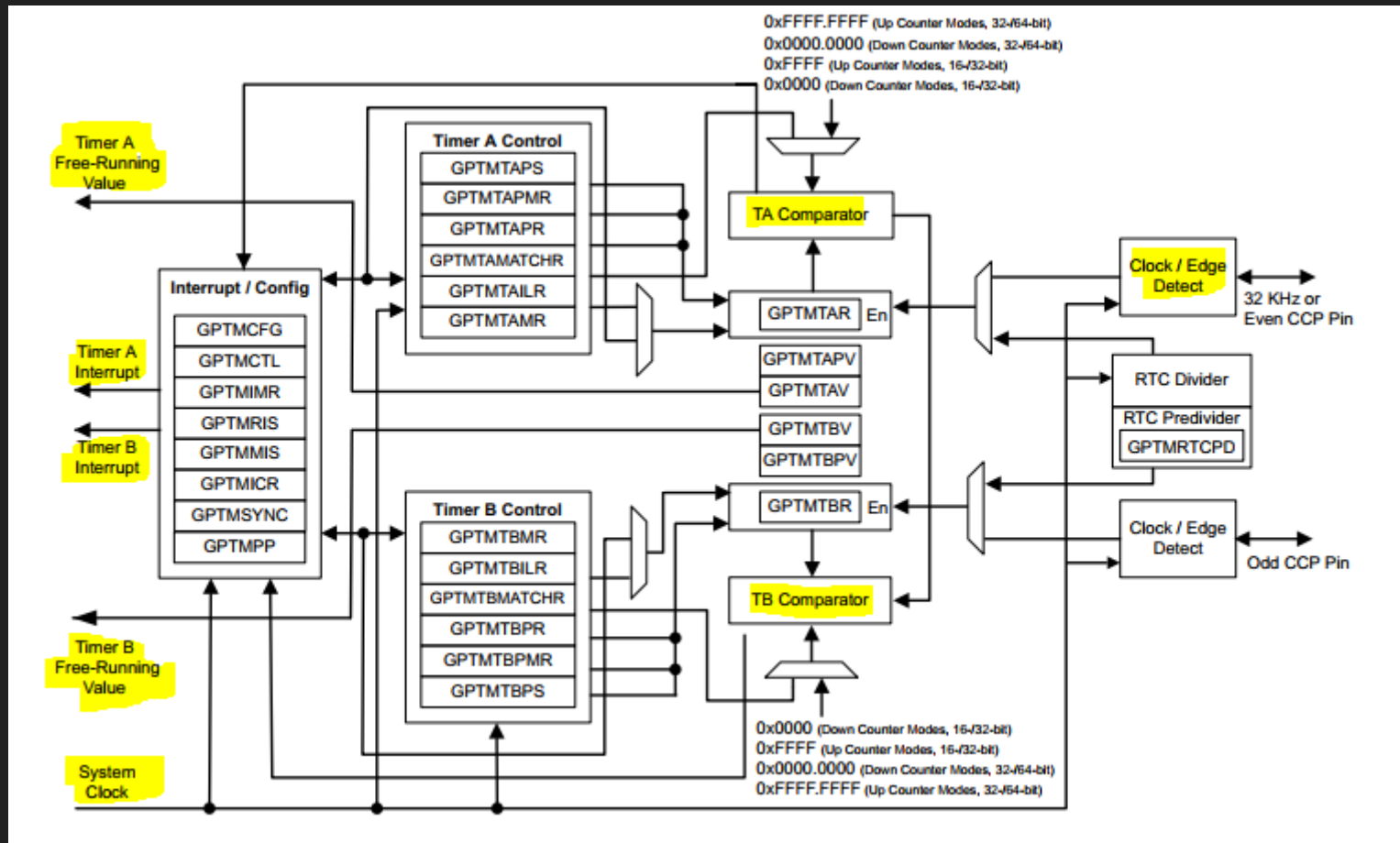


Types of Timers

- 3- Input Capture:
- In input edge-time mode, an I/O pin is used to capture the signal transition events. When an event occurs, the content of the timer counter is captured in another register while the counter keeps counting. The program can then read the counter value when the event occurs at a slightly later time.
To configure the timer to input edge time mode the TAMR and TnCMR bits of GPTMTAMR should be set to capture and Time edge mode (TnMR = 3 and TnCMR = 1).



Timer PERIPHERAL HW DESIGN



Timer CONFIGURATION

11.4.1 One-Shot/Periodic Timer Mode

The GPTM is configured for One-Shot and Periodic modes by the following sequence:

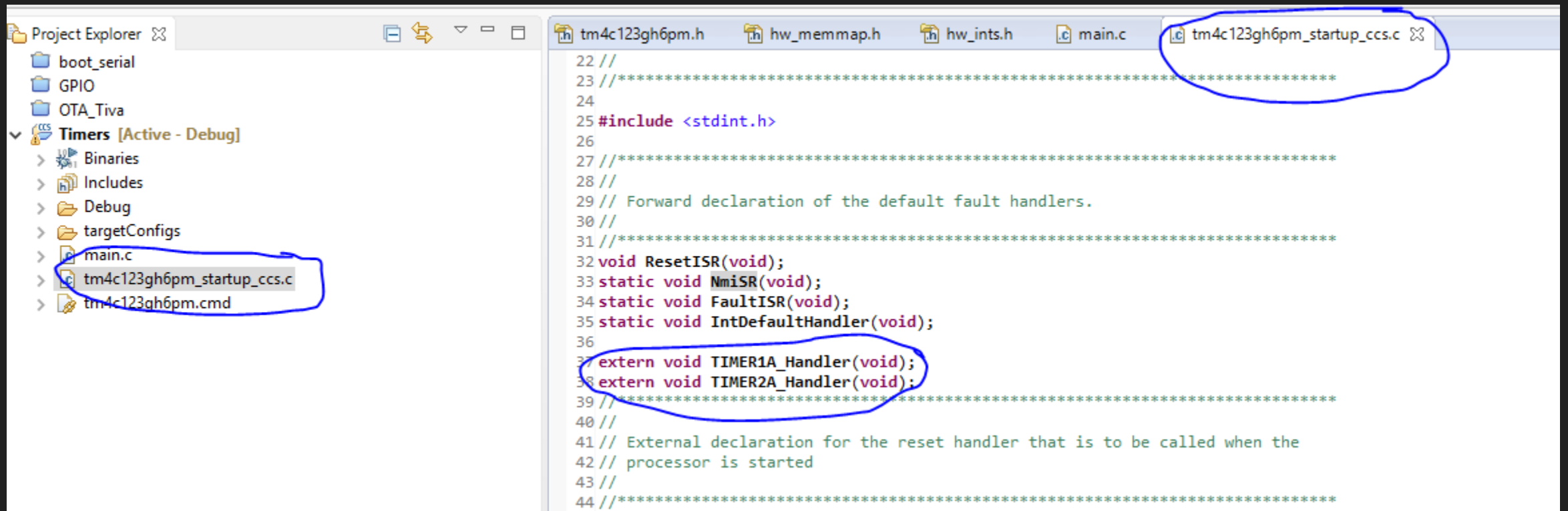
1. Ensure the timer is disabled (the **TnEN** bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0000.
3. Configure the **TnMR** field in the **GPTM Timer n Mode Register (GPTMTnMR)**:
 - a. Write a value of 0x1 for One-Shot mode.
 - b. Write a value of 0x2 for Periodic mode.
4. Optionally configure the **TnSNAPS**, **TnWOT**, **TnMTE**, and **TnCDIR** bits in the **GPTMTnMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. Load the start value into the **GPTM Timer n Interval Load Register (GPTMTnILR)**.

Timer CONFIGURATION

6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the **TnEN** bit in the **GPTMCTL** register to enable the timer and start counting.
8. Poll the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

Timer CONFIGURATION

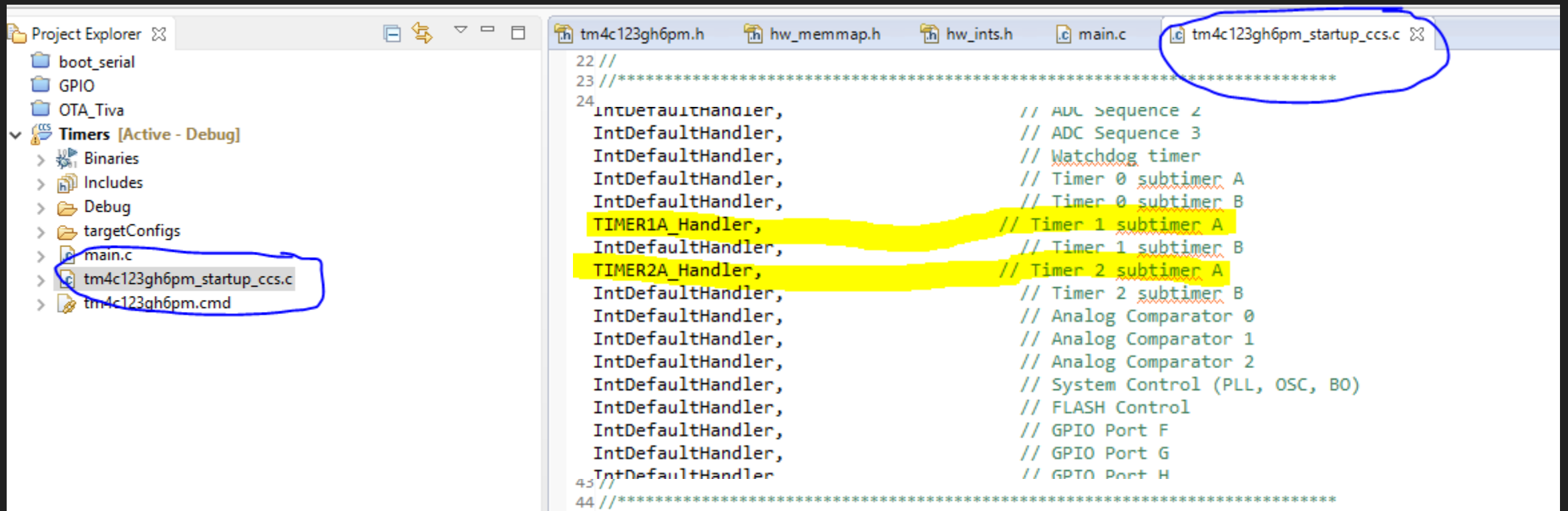
Add Timer Interrupt Handler functions to the Vector table in the Start-Up code:



```
22 //
23 //*****
24
25 #include <stdint.h>
26
27 //*****
28 //
29 // Forward declaration of the default fault handlers.
30 //
31 //*****
32 void ResetISR(void);
33 static void NmiISR(void);
34 static void FaultISR(void);
35 static void IntDefaultHandler(void);
36
37 extern void TIMER1A_Handler(void);
38 extern void TIMER2A_Handler(void);
39 //*****
40 //
41 // External declaration for the reset handler that is to be called when the
42 // processor is started
43 //
44 //*****
```

Timer CONFIGURATION

Add Timer Interrupt Handler functions to the Vector table in the Start-Up code:



The screenshot shows an IDE with the Project Explorer on the left and the Code Editor on the right. In the Project Explorer, the 'Timers' folder is expanded, and the file 'tm4c123gh6pm_startup_ccs.c' is selected and circled in blue. In the Code Editor, the file 'tm4c123gh6pm_startup_ccs.c' is open, and the Vector table is visible. The Vector table entries are listed with their corresponding interrupt handlers. The entries for 'TIMER1A_Handler' and 'TIMER2A_Handler' are highlighted in yellow. The entry for 'TIMER1A_Handler' is at line 24, and the entry for 'TIMER2A_Handler' is at line 25. The entry for 'TIMER1A_Handler' is circled in blue. The entry for 'TIMER2A_Handler' is also circled in blue. The entry for 'TIMER1A_Handler' is circled in blue.

```
22 //
23 //*****
24 IntDefaultHandler, // ADC Sequence 2
IntDefaultHandler, // ADC Sequence 3
IntDefaultHandler, // Watchdog timer
IntDefaultHandler, // Timer 0 subtimer A
IntDefaultHandler, // Timer 0 subtimer B
TIMER1A_Handler, // Timer 1 subtimer A
IntDefaultHandler, // Timer 1 subtimer B
TIMER2A_Handler, // Timer 2 subtimer A
IntDefaultHandler, // Timer 2 subtimer B
IntDefaultHandler, // Analog Comparator 0
IntDefaultHandler, // Analog Comparator 1
IntDefaultHandler, // Analog Comparator 2
IntDefaultHandler, // System Control (PLL, OSC, BO)
IntDefaultHandler, // FLASH Control
IntDefaultHandler, // GPIO Port F
IntDefaultHandler, // GPIO Port G
IntDefaultHandler, // GPIO Port H
43 //
44 //*****
```