

Lab 2

Microprocessor Systems Lab

Timers and Periodic Interrupts

October 11th, 2024

California State University of Northridge

Edgar Gutierrez

Introduction.

This lab demonstrated the use of the TM4C123G's General-Purpose Timer Module (GPTM) and Nested Vectored Interrupt Controller (NVIC). A Timer module was configured to generate periodic interrupts, producing a square wave signal verified with an oscilloscope. Additionally, a counter was incremented in a periodic task and displayed on a Seven-Segment Display.

1. How many 16/32-bit timer modules are provided by the General-Purpose Timer Module (GPTM) peripheral? List the names of all the timers. Refer to the General-Purpose Timers section of the Tiva TM4C123GH6PM Microcontroller Datasheet.

General-Purpose Timer Module (GPTM) contains six 16/32-bit GPTM blocks.

16/32-bit Timer 0, 16/32-bit Timer 1, 16/32-bit Timer 2, 16/32-bit Timer 3,16/32-bit Timer 4, 16/32-bit Timer 5

2.

Review the Nested Vectored Interrupt Controller (NVIC) section of the Tiva TM4C123GH6PM Microcontroller Datasheet and provide a description of the NVIC. How many interrupts does it support and what is the range of the priority level that can be assigned for an interrupt? Which level corresponds to the highest interrupt priority?

The NVIC manages exceptions in Handler Mode by prioritizing interrupt requests (IRQs) from peripherals or software. The NVIC allows eight priority levels, with 0 as the highest and 7 as the lowest, for 7 system handlers and 78 interrupts.

3.

List the address offsets in hexadecimal format and the descriptions of the following General-Purpose Timer Module (GPTM) registers. Provide the register access type (read-write, write-only, or read-only). Refer to the General-Purpose Timers section (Table 11-12, Timers Register Map) of the Tiva TM4C123GH6PM Microcontroller Datasheet.

|  |  |  |  |
| --- | --- | --- | --- |
| Register | Address Offset | Access Type | Description |
| GPTMCFG | 0x000 | RW | GPTM Configuration |
| GPTMTAMR | 0x004 | RW | GPTM Timer A Mode |
| GPTMCTL | 0x00C | RW | GPTM Control |
| GPTMIMR | 0x018 | RW | GPTM Interrupt Mask |
| GPTMMIS | 0x020 | RO | GPTM Masked Interrupt Status |
| GPTMICR | 0x024 | W1C | GPTM Interrupt Clear |
| GPTMTAILR | 0x028 | RW | GPTM Timer A Interval Load |
| GPTMTAPR | 0x038 | RW | GPTM Timer A Prescale |
| GPTMTAR | 0x048 | RO | GPTM Timer A |
| GPTMTAV | 0x050 | RW | GPTM Timer A Value |

4.

Write a void function named PD\_Init that takes no arguments and initializes the following pins as GPIO pins with the specified direction.

Inputs: PD0, PD2, PD4, and PD6

Outputs: PD1, PD3, PD5, and PD7

**void PD\_Init(void)**

**{**

SYSCTL-> RCGCGPIO =| 0x08;

GPIOD-> DIR &= ~(0x55);

GPIOD-> AFSEL &= ~(0x55);

GPIOD-> DEN |= 0x55;

GPIOD-> DIR |= 0xAA;

GPIOD-> AFSEL &= ~(0xAA);

GPIOD-> DEN |= 0xAA;

**}**

Tools and Materials

Tiva C Series TM4C123G LaunchPad made by Texas Instruments

USB-A to Micro-USB Cable

EduBase Board

Windows Computer

Oscilloscope and Probes

Methods

After downloading the project from the GitHub repository and opening it in Keil, we started making additions to the Timer 0A driver file. We reviewed the implementation of the the *Timer\_0A\_Interrupt\_Init* function and the TIMER0A\_Handler interrupt service routine (ISR) located in the Timer\_0A\_Interrupt.c.

We learned how to use the Timer 0A ISR by blinking an LED on the EduBase board. In the main.c file we declared the Timer\_0A\_Periodic\_Task function prototype and initialized two uint32\_t global variables, Timer\_0A\_ms\_elapsed and toggle\_rate\_ms, to zero. In main, we called the EduBase\_LEDs\_Init and Timer\_0A\_Interrupt\_Init functions before the while-loop, passing the address of Timer\_0A\_Periodic\_Task using the & operator to ensure it was executed periodically during timer interrupts.

We implemented the Timer\_0A\_Periodic\_Task function, which increments the Timer\_0A\_ms\_elapsed global variable by 1 each time Timer 0A generates an interrupt. Once 500 milliseconds have passed, the variable is reset to zero, and LED0 on the EduBase board is toggled. This function was placed below the main function.

Using an oscilloscope and probes, we measured the signal’s frequency to verify the proper operation of the Timer 0A driver. Using the parameters provided in the lab we compared the values we obtained to those of the lab.

We incremented a global counter and displayed its value on the Seven-Segment Display on the EduBase board using the Timer 0A periodic task and the main while loop. The driver for the Seven-Segment Display was provided, and we reviewed functions in the Seven\_Segment\_Display.c source file and its corresponding documentation in the header file. In main.c, we declared and initialized an 8-bit global variable called count.

We added the SysTick\_Delay.h and Seven\_Segment\_Display.h. header files and updated the *Timer\_0A\_Periodic\_Task* function by adding if-else statements to increment the count by 1 for every 500 ms.

We modified the main program by calling SysTick\_Delay\_Init and Seven\_Segment\_Display\_Init before the while-loop to set up the SysTick timer and display pins. In the loop, we called Seven\_Segment\_Display with the count variable.

For the Tasks portion of the lab we implemented the Change\_Counter\_Speed function in GPIO.c, which takes a uint8\_t parameter button\_status and returns a uint32\_t variable toggle\_rate\_ms. This function updates the toggle rate for Timer\_0A\_Periodic\_Task based on the EduBase board's push buttons, following the structure of the Note\_Controller function, with a provided mapping for reference.

Results

To analyze and verify that we were generating the correct signal frequency the code we were running on the Tiva board, we used an oscilloscope and our probes to visually see the signal we were outputting. As we can see from the image below, we were able to generate a square wave which we then used to display a counter on the seven segment display on the EduBase board.

A screen with a blue screen

Description automatically generated

We verified the frequency of the signal by placing two cursors on the display, one in the beginning and the end of one cycle.

A screen with a blue screen

Description automatically generated

We were able to expand on the project by implementing a global counter and displaying the value on the seven segment display of the EduBase board. We used the periodic task for Timer0A and the main loop. After being able to display the counter we took It one step further and programmed the buttons to change the speed of the counter with each one of the buttons (SW2 – SW5) to have a different toggle rate.

A close up of a circuit board

Description automatically generated

Discussion

This lab felt somewhat smoother coding-wise than the previous labs, even though we implemented a seven-segment display that needed its own driver. What took the most time with this lab was the oscilloscope portion, getting the signal to appear as the example in the lab was a little bit tedious.

One thing that stood out from this lab, was the fact that the driver for the LEDs can be repurposed to drive a display and other peripherals. It is satisfying and rewarding when you are able to learn a new tool and finding different ways to use them.

**Questions:**

Provide at least three use cases for the General-Purpose Timer Module (GPTM).

PWM generation, generate interrupts and determine the elapsed time between the interrupt and the entry into the ISR.

2.

If the Timer\_0A\_Periodic\_Task had been updated to have a toggle rate of 200 ms, what would be the frequency of the square wave generated by the PB0 pin?

**void Timer\_0A\_Periodic\_Task(void)**

**{**

Timer\_0A\_ms\_elapsed++;

**if (Timer\_0A\_ms\_elapsed >= 200)**

{

Timer\_0A\_ms\_elapsed = 0;

GPIOB->DATA ^= 0x01;

}

f = 1/T = 1/.2 = 5Hz

3.

What are the key features of the General-Purpose Timer Module (GPTM)? Which modes does the GPTM support? List the available modes. Refer to the Timer Modes section (11.3.2) of the Tiva TM4C123GH6PM Microcontroller Datasheet.

The GPTM has 16-bit and 32-bit timer configurations, interrupt handling, PWM generation, and can run 2 timers at once. Modes include, One-Shot/Periodic Timer Mode, Real-Time Clock Timer Mode, Input Edge-Count Mode, Input Edge-Time Mode, PWM Mode, Wait-for-Trigger Mode,

4.

Which bit from the GPTM Timer A Mode register (GPTMTAMR) can be set to change the timer’s count direction to up? Refer to pages 729 – 732 of the Tiva TM4C123GH6PM Microcontroller Datasheet.

Bit 4, 0 for down, 1 for up.

5.

Explain how this lab demonstrated the concept of concurrency and discuss the advantages of using periodic interrupts over the polling method (i.e. using the while-loop in the *main* function).

This lab demonstrated concurrency by allowing us to run multiple tasks at the same time. We were able to modify the output in real-time simply by implementing an interrupts when a button was pressed. Polling is less efficient then interrupts and it is more taxing on the CPU. It is more responsive and it reduces power consumption.