Timers

Introduction

In many microprocessors, timers are used for determining simple intervals. The MSP430 timers are significantly more capable. They can be used to generate multiple PWM frequencies, control ADC hardware or even implement a UART port. Let’s learn a bit more about them now.

Objectives

• Timer_A Architecture
• Count modes
• Interrupts
• TAIV
• Timer_B differences
• Timer lab
*** This page left blank with malice aforethought ***
Module Topics

Timers........................................................................................................................................4-1

Timer_A.......................................................................................................................................4-3
Counting Modes..........................................................................................................................4-5
Interrupts...................................................................................................................................4-5
TAIV Handler..............................................................................................................................4-6
PWM Example............................................................................................................................4-6
Direct Hardware Control............................................................................................................4-7
UART Implementation...............................................................................................................4-8
Timer B....................................................................................................................................4-8

Lab 5 – Timer_A..........................................................................................................................4-9
Hardware list:............................................................................................................................4-10
Software list:.............................................................................................................................4-10

IAR Kickstart Procedure.............................................................................................................4-11
Start-up.....................................................................................................................................4-11
Add Source File........................................................................................................................4-11
Complete the Code..................................................................................................................4-11
Shut Down.................................................................................................................................4-14

Code Composer Studio 4.1 Procedure......................................................................................4-15
Start-up.....................................................................................................................................4-15
Add Source File........................................................................................................................4-15
Complete the Code..................................................................................................................4-15
Shut Down.................................................................................................................................4-18

Review Questions......................................................................................................................4-19
*** Blank! Blank! My kingdom for a blank! ***
Timer_A

- Asynchronous 16-Bit timer/counter
- Continuous, up-down, up count modes
- Multiple capture/compare registers
- PWM outputs
- Interrupt vector register for fast decoding
- Can trigger DMA transfer
- On all MSP430s

Counting Modes

Timer_A Counting Modes

- Stop/Halt: Timer is halted
- Continuous: Timer continuously counts up
- Up: Timer counts between 0 and CCR0
- Up/Down: Timer counts between 0 and CCR0 and 0

Interrupts ...
Interrupts

Timer_A Interrupts

The Timer_A Capture/Comparison Register 0 Interrupt Flag (TACCR0) generates a single interrupt vector:

TACCR0 CCIFG → TIMERA0_VECTOR

No handler required

TACCR1, 2 and TA interrupt flags are prioritized and combined using the Timer_A Interrupt Vector Register (TAIV) into another interrupt vector:

TACCR1 CCIFG → TAIV → TIMERA1_VECTOR

Your code must contain a handler to determine which Timer_A1 interrupt triggered

TAIV Handler

TAIV Handler Example

<table>
<thead>
<tr>
<th>Source</th>
<th>TAIV Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>No interrupt pending</td>
<td>0</td>
</tr>
<tr>
<td>TACCR1 CCIFG</td>
<td>02h</td>
</tr>
<tr>
<td>TACCR2 CCIFG</td>
<td>04h</td>
</tr>
<tr>
<td>Reserved</td>
<td>06h</td>
</tr>
<tr>
<td>Reserved</td>
<td>08h</td>
</tr>
<tr>
<td>TAIFG</td>
<td>0Ah</td>
</tr>
<tr>
<td>Reserved</td>
<td>0Ch</td>
</tr>
<tr>
<td>Reserved</td>
<td>0Dh</td>
</tr>
</tbody>
</table>

IAR C code

```
#pragma vector = TIMERA1_VECTOR
__interrupt void TIMERA1_ISR(void)
{
    switch(__even_in_range(TAIV,10))
    {
        case 2 :  // TACCR1 CCIFG
            P1OUT ^= 0x04; break;
        case 4 :  // TACCR2 CCIFG
            P1OUT ^= 0x02; break;
        case 10 :  // TAIFG
            P1OUT ^= 0x01; break;
    }
}
```

Assembly code

```
0xF814  add.w &TAIV,PC
0xF818  reti
0xF81A  jmp 0xF824
0xF81C  jmp 0xF82A
0xF81E  reti
0xF820  reti
0xF822  xor.b #0x4,&P1OUT
0xF824  reti
0xF826  xor.b #0x2,&P1OUT
0xF828  reti
0xF82A  xor.b #0x1,&P1OUT
0xF82C  reti
0xF830  xor.b #0x1,&P1OUT
0xF834  reti
```

PWM ...
PWM Example

- Completely automatic
- Independent frequencies with different duty cycles can be generated for each CCR
- Code examples on the MSP430 website

Direct Hardware Control

Example: ADC12

TAIFG:
Reference & ADC on

TACCR1:
Ref delay / ADC trigger

ADC12IFG:
Process ADC result
Ref/ADC Off

CPU Active Mode
**Module Topics**

**UART Implementation**

**Low-Overhead UART Implementation**

- 100% hardware bit latching and output
- Full speed from LPM3 and LPM4
- Low CPU Overhead
- App Note SLAA078 on web

**Timer B**

**Timer_B Differences**

- 8,10,12 or 16-bit timer or counter
- Up to 7 CCRx units available
- Outputs double-buffered for simultaneous loading
- CCRx registers can be grouped for simultaneous updates
- SCCI latch not implemented (no UART function)
- Tri-state function from external pin
- Default Function is identical to Timer_A
Lab 5 – Timer_A

Let’s configure a timer to wake the MSP430 from a low power mode and blink an LED. Granted, that’s a pretty simple task, but the idea here is to learn how to program the timer.

Configure Timer_A on the MSP430FG4618/9 to wake up the CPU and toggle an LED
Module Topics

**Hardware list:**
- WinXP PC
- MSP-FET430UIF
- USB cable
- JTAG ribbon cable
- MSP430FG461x/F28xx Experimenter’s Board
- Jumpers

**Software list:**
- IAR Kickstart for MSP430 version 4.21B
- Code Composer Studio 4.1
- Labs
- Additional pdf documentation
- Adobe™ Reader
IAR Kickstart Procedure

Configure a timer to wake up the CPU from a low power mode and blink an LED … pretty straight-forward.

Start-up

1. Hardware

Assure that the debug interface is correctly connected to the PC and the FG4618/9 debug port.

2. Start IAR

Start IAR Kickstart. Create a new workspace and a new project in the Lab5 folder. Configure the project options as shown earlier.

Add Source File

3. Add the source file to the project

Add Lab5_exercise.c from the C:\MSP430\IAR Labs\Lab5 folder to the project. Double-click on the file in the Project pane to open it for editing.

Complete the Code

Like the previous lab, the next steps will lead you though the process of filling in the four blanks in the code. You should already have the process down, though, so we won’t give you nearly the level of detail as you had in the previous lab.

You’ll probably want to open slau056g.pdf and the msp430xG46x.h header file.

Again, if you’re lazy and want to skip to the solution, you can either look in the Addendum at the back of this workbook or open up the solution file.
#include <msp430xG46x.h>

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;  // Stop WDT
    FLL_CTL0 |= XCAP14PF;       // Configure load caps
    P2DIR |= BIT1;              // Set P2.1 to output direction
    TACTL = ________________;   // Clock = ACLK (32768), clear
    TACCTL0 = ____;             // CCR0 interrupt enabled
    TACCR0 = _______;           // #counts for 1s
    TACTL |= ____;              // Setting mode bits starts timer

    _BIS_SR(LPM3_bits + GIE);   // Enter LPM3 w/ interrupt
}

// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
    P2OUT ^= 0x02;              // Toggle P2.1 using exclusive-OR
}


4. \( \text{TACTL} = \_\_\_\_\_\_\_\_\_\_\_; \)

Clock = ACLK (32768Hz)

To set the clock source select to ACLK you’re going to need to know how the TASSELx field is configured. That’s enough of a hint …

Clear

Finding the counter clear is pretty easy.

5. \( \text{TACCTL0} = \_\_\_\_\_\_\_\_\_\_\; \)

Enable CCR0 interrupt

Enable the capture/compare interrupt. If you’ve ever been geo-caching, this process is analogous. The GPS will get you close, but then you’ve got to hunt around on your hands and knees for the prize.

6. \( \text{TACCR0} = \_\_\_\_\_\_\_\_\_\_\; \)

Number of counts for one second

This one takes just a little bit of thought. What’s the clock frequency we’re using to drive the timer? (Hint: We selected it in step 4). How many clock cycles would equal one second? Bear in mind that when the timer rolls over to zero, that is also counted as a tick, so to get \( n \) ticks, you put \( n-1 \) in the CCR0 register.

7. \( \text{TACTL} |= \_\_\_\_\_\_\_\_\_\_\; \)

Check the User’s Guide and make sure which mode you want the timer to operate in, then find the correct symbol in the header file.

8. Build, Download and Run

Try out the code and make sure it works properly. Correct any errors you may have. Observe the LED and verify that it blinks at the proper interval. Feel free to play around with the interval period in the code.
9. **A Few More Questions**

Here’s a great opportunity to show off your ability to search the User’s Guide. The answers are in the Addendum at the back of this workbook.

**Why was TAIE not set in TACTL?**

**Why were the MCx bits not set initially when TACTL was configured?**

---

**Shut Down**

10. **Shut down**

**Halt** the debugger and **shut down IAR Kickstart.**

---

IAR Users … you’re done. Proceed to the review questions on page 4-19.
Code Composer Studio 4.1 Procedure

Configure a timer to wake up the CPU from a low power mode and blink an LED … pretty straight-forward.

Start-up

1. Hardware

Assure that the debug interface is correctly connected to the PC and the FG4618/9 debug port.

2. Start CCS

Start CCS. Create a new workspace in C:\MSP430ODW\CCS Labs\Lab5\workspace and a new project in the folder called Lab5. Configure the project settings as shown earlier.

Add Source File

3. Add the source file to the project

Add Lab5_exercise.c from the C:\MSP430\CCS Labs\Lab5 folder to the project. Double-click on the file in the Project pane to open it for editing.

Complete the Code

Like the previous lab, the next steps will lead you though the process of filling in the four blanks in the code. You should already have the process down, though, so we won’t give you nearly the level of detail as you had in the previous lab.

You’ll probably want to open slau056g.pdf and the msp430xG46x.h header file.

Again, if you’re lazy and want to skip to the solution, you can either look in the Addendum at the back of this workbook or open up the solution file.
```c
#include <msp430xG46x.h>

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;    // Stop WDT
    FLL_CTL0 |= XCAP14PF;                        // Configure load caps
    P2DIR |= BIT1;                              // Set P2.1 to output direction
    TACTL = ________________;                  // Clock = ACLK (32768), clear
    TACCTL0 = ____;                            // CCR0 interrupt enabled
    TACCR0 = _______;                           // #counts for 1s
    TACTL |= ____;                             // Setting mode bits starts timer
    _BIS_SR(LPM3_bits + GIE);                 // Enter LPM3 w/ interrupt
}

// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
    P2OUT ^= 0x02;                            // Toggle P2.1 using exclusive-OR
}
```
4. **TACTL = _______________;**

   **Clock = ACLK (32768Hz)**

   To set the clock source select to **ACLK** you’re going to need to know how the **TASSELx** field is configured. That’s enough of a hint …

   **Clear**

   Finding the **counter clear** is pretty easy.

5. **TACCTL0 = ______________;**

   **Enable CCR0 interrupt**

   Enable the capture/compare interrupt. If you’ve ever been geo-caching, this process is analogous. The GPS will get you close, but then you’ve got to hunt around on your hands and knees for the prize.

6. **TACCR0 = ______________;**

   **Number of counts for one second**

   This one takes just a little bit of thought. What’s the clock frequency we’re using to drive the timer? (Hint: We selected it in step 4). How many clock cycles would equal one second? Bear in mind that when the timer rolls over to zero, that is also counted as a tick, so to get n ticks, you put n-1 in the CCR0 register.

7. **TACTL |= ______________;**

   Check the User’s Guide and make sure which mode you want the timer to operate in, then find the correct symbol in the header file.

8. **Build, Download and Run**

   Try out the code and make sure it works properly. Correct any errors you may have. Observe the LED and verify that it blinks at the proper interval. Feel free to play around with the interval period in the code.
9. A Few More Questions

Here’s a great opportunity to show off your ability to search the User’s Guide. The answers are in the Addendum at the back of this workbook.

Why was TAIE not set in TACTL?

Why were the MCx bits not set initially when TACTL was configured?

Shut Down

10. Shut down

Halt the debugger and shut down Code Composer Studio.

CCS Users … you’re done.
Review Questions

Review

- Name the counting modes.

- What is the TAILV register's purpose?

- In addition to normal timer functions, name some other functions the timer can perform.

You can find the answers to these questions in the Addendum section at the end of this workbook.