

## 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

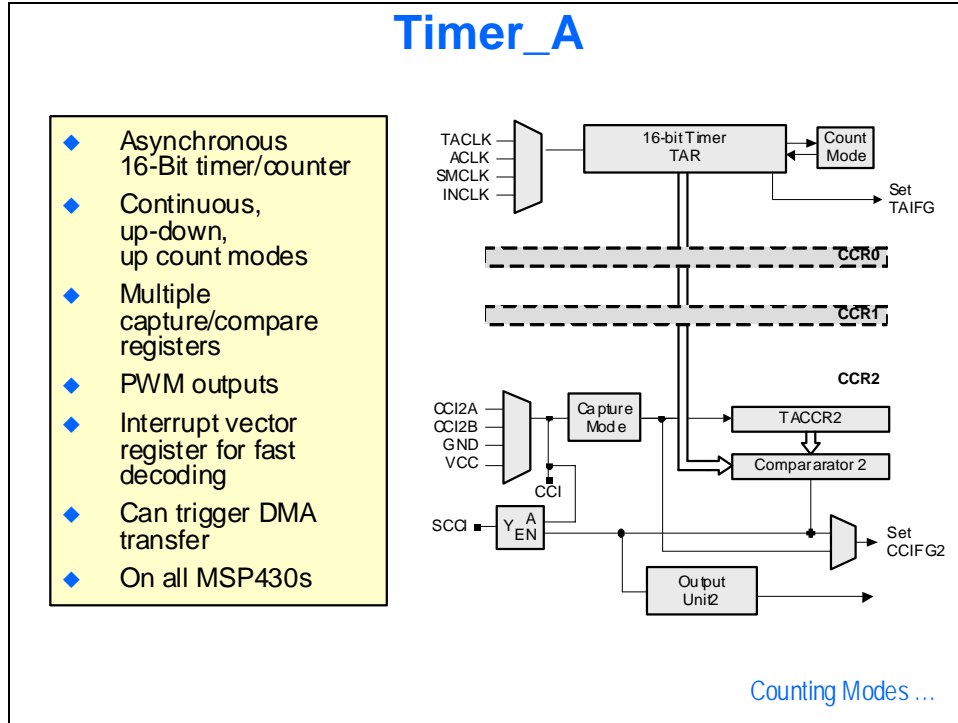
\*\*\* This page left blank with malice aforethought \*\*\*

# Module Topics

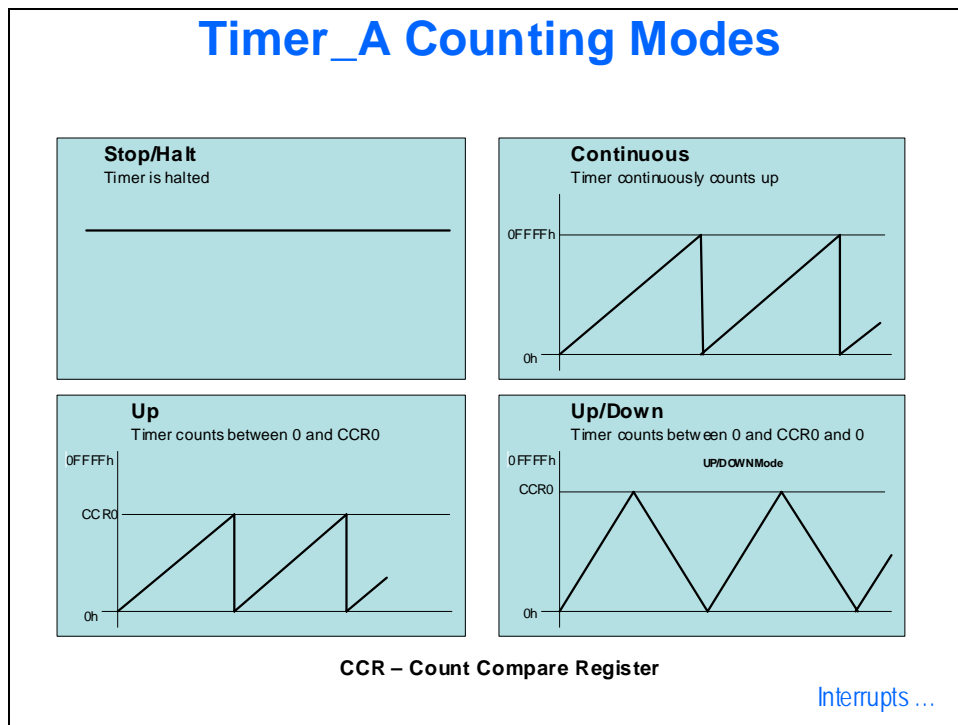
<b>Timers.....</b>	<b>4-1</b>
<i>Module Topics.....</i>	<i>4-3</i>
Timer_A .....	4-5
Counting Modes .....	4-5
Interrupts .....	4-6
TAIV Handler.....	4-6
PWM Example .....	4-7
Direct Hardware Control .....	4-7
UART Implementation.....	4-8
Timer B.....	4-8
<i>Lab 5 – Timer_A .....</i>	<i>4-9</i>
Hardware list: .....	4-10
Software list:.....	4-10
<i>IAR Kickstart Procedure.....</i>	<i>4-11</i>
Start-up .....	4-11
Add Source File.....	4-11
Complete the Code .....	4-11
Shut Down .....	4-14
<i>Code Composer Studio 4.1 Procedure.....</i>	<i>4-15</i>
Start-up .....	4-15
Add Source File.....	4-15
Complete the Code .....	4-15
Shut Down.....	4-18
<i>Review Questions.....</i>	<i>4-19</i>

\*\*\* Blank! Blank! My kingdom for a blank! \*\*\*

# Timer\_A



## Counting Modes





## PWM Example

### Timer\_A PWM Example

MSP430F11x1	
TEST	TA2/P1.7
Vcc	P1.6
P2.5	P1.5
Vss	P1.4
XOUT	P1.3
XIN	TA1/P1.2
RST	P1.1
P2.0	P1.0
P2.1	P2.4
P2.2	P2.3

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

ADC12 Control ...

## Direct Hardware Control

### Direct Hardware Control With Timer\_A

Example: ADC12

**TAIFG:**  
Reference & ADC on

**TACCR1:**  
Ref delay / ADC trigger

**ADC12IFG:**  
Process ADC result  
Ref/ADC Off

CPU Active Mode

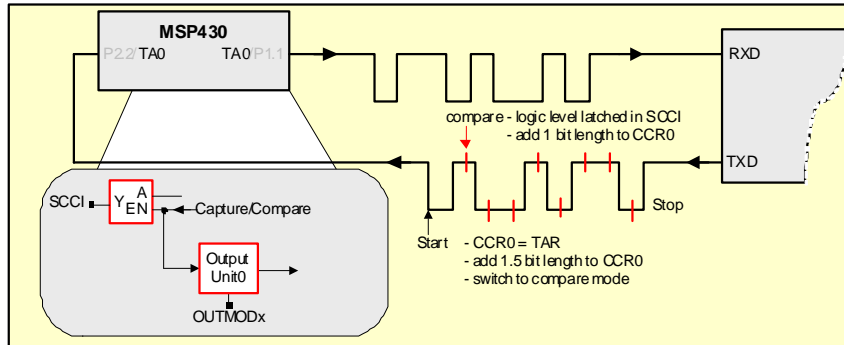
2s

17ms

UART ...

## UART Implementation

### Low-Overhead UART Implementation



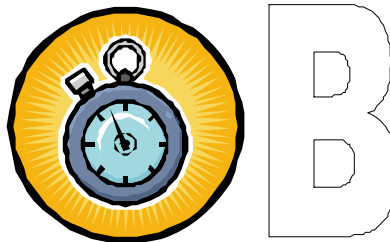
- ◆ 100% hardware bit latching and output
- ◆ Full speed from LPM3 and LPM4
- ◆ Low CPU Overhead
- ◆ App Note SLAA078 on web

Timer\_B ...

## Timer B

### 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
- ◆ SSCI latch not implemented (no UART function)
- ◆ Tri-state function from external pin
- ◆ Default Function is identical to Timer\_A



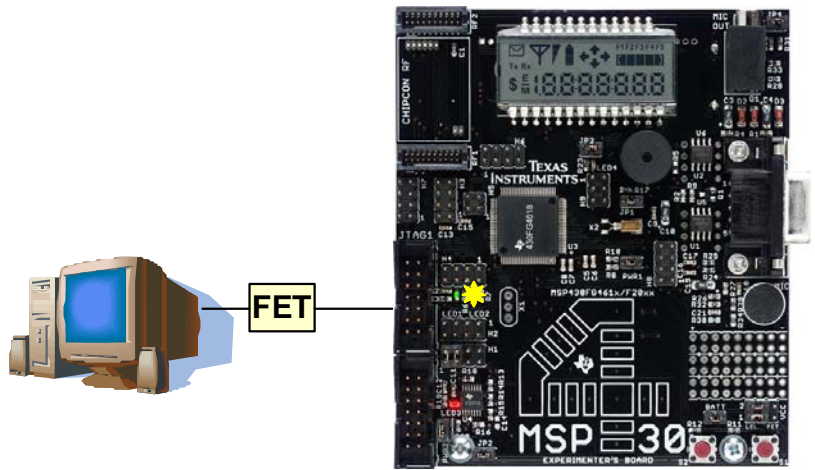
Lab5 ...

## 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.

### Lab5: Timer\_A

Configure Timer\_A on the MSP430FG4618/9 to wake up the CPU and toggle an LED



The diagram illustrates the experimental setup. On the left, a computer system (monitor, tower, and speakers) is connected to an MSP430 microcontroller board. A box labeled 'FET' is connected to the board, indicating the connection point for the Field-Effect Transistor. The board features a Texas Instruments MSP430FG4618/9 microcontroller, a small LCD display showing 'SE 10000000', and an LED labeled 'LED\_A02'. The board is labeled 'MSP30 EXPERIMENTOR'S BOARD'.

[Review ...](#)

## 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 through 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. **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** *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 through 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 + WDTCTL; // 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=TIMER_A0_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 TAIV 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.

\*\*\* !KCOR s034PSM \*\*\*