Embedded Systems
LA4 (C5-C6)

Name: Ahmed ElShaarany

Date: 6/14/2015

Lab 6a Worksheet

Goal: Write a function setting up Timer_A to generate an interrupt every two seconds.

1. How many clock cycles does it take for a 16-bit counter to ‘rollover’? (Hint: 16-bits)

   65536

2. Our goal is to generate a two second interrupt rate based on the timer clock input diagramed above.

   Clock input (circle one): **ACLK**

3. Calculate the Timer settings for the clocks & divider values needed to create a timer interrupt every 2 seconds. (That is, how can we get a timer period rate of 2 seconds.)

   Input Clock: **ACLK** running at the frequency = 32KHz

   Timer Clock: 32768 / 1 = 32768

   Timer Rate: (1 / 32768) * 65536 = 2

4. Which Timer do you need to use for this lab exercise?

   TIMER_A0_BASE

5. Write the TIMER_A_initContinuousMode() function.

```
Timer_A_initContinuousModeParam initContParam = { 0 };
initContParam.clockSource = TIMER_A_CLOCKSOURCE_ACLK;
initContParam.clockSourceDivider = TIMER_A_CLOCKSOURCE_DIVIDER_1;
initContParam.timerInterruptEnable_TAIE = TIMER_A_TAIE_INTERRUPT_ENABLE;
initContParam.timerClear = TIMER_A_DO_CLEAR;
initContParam.startTimer = false;
Timer_A_initContinuousMode(TIMER_A0_BASE, &initContParam );
```

6. Skipped

7. Complete the code to for the 3rd part of the “Timer Setup Code”.

```
// Clear the timer interrupt flag
Timer_A_clearTimerInterruptFlag (TIMER_A0_BASE); // Clear TA0IFG
// Start the timer
Timer_A_startCounter ( // Function to start timer
TIMER_A0_BASE, // Which timer?
TIMER_A_CONTINUOUS_MODE); // Run in Continuous mode
```

8. Change the following interrupt code to toggle LED2 when Timer_A rolls-over.

   a. Port/Pin number for LED2: Port 4, Pin 7

   Timer Interrupt Vector: #pragma vector = TIMER_A0_BASE

   Timer Interrupt Vector register: TA0IV

   b. Here is the interrupt code that exists from a previous exercise, change it as needed

```
#pragma vector= TIMER0_A0_VECTOR
__interrupt void timer1_ISR (void)
{
// 4. Timer ISR and vector
switch( __even_in_range(TA0IV, 14 )) { // Read timer IV register
case 0: break; // None
case 2: break; // CCR1 IFG
case 4: break; // CCR2 IFG
case 6: break; // CCR3 IFG
case 8: break; // CCR4 IFG
case 10: break; // CCR5 IFG
case 12: break; // CCR6 IFG
case 14: // TAR overflow
// Toggle LED2 (Green) on/off
GPIO_toggleOutputOnPin( GPIO_PORT_P4, GPIO_PIN7 );
break;
default: _never_executed();
}
}
```

Lab 6b Worksheet

1. Calculate the timer period (for CCR0) to create a 1 second interrupt rate.

   Timer Rate = (1/32768) * 32768 = 1 second

2. Complete the TIMER_A_initUpMode() function?

```
Timer_A_initUpModeParam initUpParam = { 0 };
initUpParam.clockSource = TIMER_A_CLOCKSOURCE_ACLK;
initUpParam.clockSourceDivider = TIMER_A_CLOCKSOURCE_DIVIDER_1;
initUpParam.timerPeriod =     32768         ; // (calculated in previous question)
initUpParam.timerInterruptEnable_TAIE = TIMER_A_TAIE_INTERRUPT_ENABLE;
initUpParam.captureCompareInterruptEnable_CCR0_CCIE =
TIMER_A_CAPTURECOMPARE_INTERRUPT_ENABLE;  // Enable CCR0 compare interrupt
initUpParam.timerClear = TIMER_A_DO_CLEAR;
initUpParam.startTimer = false;
Timer_A_initUpMode(TIMER_A0_BASE, &initUpParam );
```

3. Modifying our previous code, we need to clear both interrupts and start the timer.

```
// Clear the timer flag and start the timer
Timer_A_clearTimerInterruptFlag( TIMER_A0_BASE ); // Clear TA0IFG
Timer_A_clearCaptureCompareInterruptFlag ( // Clear CCR0IFG
TIMER_A0_BASE,
TIMER_A_CAPTURECOMPARE_REGISTER_0 );
Timer_A_startCounter( TIMER_A0_BASE, // Start timer in
TIMER_A_UP_MODE ); // up mode
```

4. Add a second ISR to toggle the LED1 whenever the CCR0 interrupt fires.

   On your Launchpad, what Port/Pin number does the LED1 use? P1.0

```
#pragma vector= TIMER0_A1_VECTOR
__interrupt void ccr0_ISR (void)
{
// Toggle the LED1 on/off


GPIO_toggleOutputOnPin( GPIO_PORT_P1, GPIO_PIN0 );


}
```

Step 15

Which ISR was reached first? The **ccr0_ISR** was reached first

Why? Because The CC0IFG occurs when there is a compare match, while the TA0IFG interrupt occurs once the counter goes back to zero. So that is why the CC ISR is reached first and hence, both interrupts are one cycle apart.