

Introduction

Here are the answers to all those pesky questions in the workshop, along with the lab solutions.

Objectives

- Module review answers
- Lab Question Answers
- Lab Solutions

*** I was somewhat ambivalent about leaving this page blank. ***

Lab Solutions and Review Answers

Addendum.....	7-1
<i>Lab Solutions and Review Answers.....</i>	<i>7-3</i>
Introduction Module Review Answers	7-5
Lab1 IAR Code.....	7-5
Lab1 CCS Code.....	7-6
Flash Programming Exercise.....	7-6
Lab2 IAR Solution.....	7-7
Lab3 Step 11 Answers.....	7-8
Ultra-Low Power Module Review Answers.....	7-9
Lab4 IAR Solution.....	7-10
Analog Peripherals Module Review Answers	7-11
Lab5 IAR Solution.....	7-12
Lab5 Step 9 Answers.....	7-13
Timer Module Review Answers	7-14
Communications Module Review Answers	7-15
Optional Lab6 - USCI/SPI Communications IAR Solution	7-16

*** This page reluctantly left blank ***

Introduction Module Review Answers

Review

- ◆ How many general purpose registers does the MSP430 have?
12
- ◆ What is the purpose of the constant generator?
Reduce code size and cycles by automatically generating commonly used constants
- ◆ Where is the best resource for MSP430 information?
www.ti.com/msp430
- ◆ At reset, all I/O pins are set to ...
Inputs
- ◆ Why should you use standard definitions?
Resulting code is easier to read and debug

Ultra-low Power

Lab1 IAR Code

```
#include "msp430x20x3.h"

                ORG  0F800h                ; Program start

RESET          mov.w #280h,SP              ; Stack
               mov.w #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog
               bis.b #01h,&P1DIR

Mainloop       xor.b #01h,&P1OUT

Delay          dec.w R15
               jnz  Delay
               jmp  Mainloop

                ORG  0FFFEh                ; RESET
vector
```

Lab1 CCS Code

```
.cdecls C,LIST,"msp430x21x1.h" ; Include device header file

RESET      .text                                ; Progam Start
           mov.w #280h,SP                       ; Stack
           mov.w #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog
           bis.b #01h,&P1DIR

Mainloop   xor.b #01h,&P1OUT

Delay      dec.w R15
           jnz Delay
           jmp Mainloop

           .sect ".reset"                       ; MSP430 RESET Vector

.short RESET
.end

          DW RESET
          END
```

Flash Programming Exercise

$f_{FTG} = 476 \text{ kHz}$

$t_{Word} = 30$

Time to program a word or byte = $30/476000 = 63\mu\text{S}$

Time to randomly program 1024 words = $1024 * 63\mu\text{S} = 64.5\text{mS}$

Lab2 IAR Solution

```
#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
    P1IES = BIT0;                // H-L transition
    P1IE = BIT0;                 // Enable interrupt
    _EINT();                      // Enable interrupts
    while (1);
}

// P1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void P1ISR (void)
{
    unsigned volatile int i;
    for (i=10000; i>0; i--);      // Debounce delay
    P1IFG &= ~BIT0;               // Clear P1IFG
    if ((P1IN & 0x01) == 0)
        P2OUT ^= 0x02;           // Toggle P2.1 using exclusive-OR
}
```

Lab3 Step 11 Answers

Why were the I/Os configured as they were?

Unused I/O must be configured as outputs, otherwise, floating gate current will occur. The outputs were then set to values so as not to contend with other on-board circuitry.

Why was LPM3 used?

No clocks are needed. LPM3 leaves on the 32768Hz running and shuts down all other clocks.

Look in the header file to see how LPM3_bits is defined

SCG1+SCG0+CPUOFF

What further low-power improvements could be made?

LPM4 could be used. A timer could be employed for the pushbutton debounce.

Ultra-Low Power Module Review Answers

Review

- ◆ **To minimize power consumption, you should maximize your time in what LPM mode?**
LPM3
- ◆ **Why are unused pins set as outputs?**
To avoid floating gate currents
- ◆ **You should control program flow with ...**
Interrupts
- ◆ **Most MSP430 designs utilize a _____ crystal.**
32,768 Hz

Analog Peripherals

Lab4 IAR Solution

```

#include "msp430xG46x.h"

volatile unsigned int i;
volatile unsigned int ADCresult;
volatile unsigned long int DegC, DegF;

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop watchdog timer
    ADC12CTL0 = ADC12ON + REFON + REF2_5V + SHT0_7;
                                         // Turn ADC on, ref on. Ref = 2.5V,
                                         // Set sampling time
    ADC12CTL1 = SHP;                    // Use sampling timer
    ADC12MCTL0 = INCH_10 + SREF_1;      // Select channel A10, Vref+
    ADC12IE = 0x01;                    // Enable ADC12IFG.0
    for (i = 0; i < 0x3600; i++);       // Delay for reference start-up
    ADC12CTL0 |= ENC;                   // Enable conversions
    __enable_interrupt();               // Enable interrupts

    while(1)
    {
        ADC12CTL0 |= ADC12SC;           // Start conversion
        __bis_SR_register(LPM0_bits);   // Enter LPM0

        // DegC = (Vsensor - 986mV)/3.55mV
        // Vsensor = (Vref)(ADCresult)/4095
        // DegC -> ((ADCresult - 1615)*704)/4095
        DegC = (((long)ADCresult-1615)*704)/4095);
        DegF = ((DegC * 9/5)+32);        // Calculate DegF
        __no_operation();               // SET BREAKPOINT HERE
    }
}

#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR(void)
{
    ADCresult = ADC12MEM0;              // Move results, IFG is cleared
    __bic_SR_register_on_exit(LPM0_bits); // Exit LPM0
}

```

Analog Peripherals Module Review Answers

Review

- ◆ **What is your lowest power option for triggering an ADC?**
Trigger conversion with a timer.
- ◆ **Name the four ADC conversion modes:**
Single,
Sequence
Repeat-single,
Repeat-sequence
- ◆ **What is the purpose of the DTC?**
The Direct Transfer Controller moves the conversion result of the ADC10 into any MSP430 memory
- ◆ **ADC10 and ADC12 can sample at what speed?**
200ksps

[Timer Section ...](#)

Lab5 IAR Solution

```
#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 = TASSEL_1 + TACLK;            // Clock = ACLK (32768), clear
    TACCTL0 = CCIE;                      // CCR0 interrupt enabled
    TACCR0 = 32768-1;                    // #counts for 1s
    TACTL |= MC_1;                       // 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
}
```

Lab5 Step 9 Answers

Why was TAIE not set in TACTL?

Actually, we didn't use the interrupt generated when you enable TAIE. Timer_A has several interrupts. TAIE enables an interrupt to occur on overflow. We could have used it, but instead we used the TACCR0 interrupt, which fires when TAR hits the CCR0 value.

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

Setting the MCx bits starts the timer running, so you want all setup to be completed beforehand.

Timer Module Review Answers

Review

- ◆ Name the counting modes.

Up, Up/Down and Continuous

- ◆ What is the TAIV register's purpose?

To combine three interrupts into a single interrupt to the CPU. Also acts as an indicator for handler code to determine which interrupt triggered.

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

ADC12 hardware control, PWM, UART

[Communication section ...](#)

Communications Module Review Answers

Review

- ◆ The new, standard MSP430 serial communication module is:
the USCI module
- ◆ Implementing SPI on the USI or USCI provides a _____
and _____ solution.
low-power, low-cycle count
- ◆ The best place to look for code examples is:
the MSP430 website
- ◆ The best place to find technical documentation is:
the MSP430 website

Wrap Up ...

Optional Lab6 - USCI/SPI Communications IAR Solution

```
P3SEL |= 0x06; // Assign I2C pins to USCI_B0
UCB0CTL1 |= UCSWRST; // Enable SW reset (why?)
UCB0CTL0 = UCMST + UCMODE_3 + UCSYNC; // I2C Master, synchronous mode
UCB0CTL1 = UCSSEL_2 + UCSWRST; // Use SMCLK, keep UCSWRST set
UCB0BR0 = 11; // fSCL = SMCLK/11 = 95.3kHz
UCB0BR1 = 0;
UCB0I2CSA = 0x48; // Set slave address
UCB0CTL1 &= ~UCSWRST; // Clear SW reset, resume operation
UCB0I2CIE |= UCNACKIE; // Interrupt on slave Nack
IE2 |= UCB0RXIE; // Enable RX interrupt
```

