

Valparaiso University

College of Engineering

ECE-422: Embedded Microcontrollers II

Assignment 3 – MSP430 – Large Numbers on the LCD

Due: Before the Start of FRIDAY’s class (January 31, 2020)

Honor Code: \_\_\_\_\_\_\_\_I have neither given or received, nor have I tolerated others’ use of unauthorized aid\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name: \_\_\_\_\_Joe Leveille\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_Joseph Leveille\_\_\_\_\_

After you instructor finishes the lecture portion, open the LCD Bigger Numbers document (on Blackboard) and follow the instructions. Get answer questions, and do the coding as described in the document and below.

8. What happens when the number increments past 999,999 and reaches 1,000,000?

The LCD displays “ERROR”

1. Ok, what happened when we started with a negative number?

“ERROR”

PBW ECE-422 Assignment 3 - MSP430 Large Numbers on the LCD - Spring 2020 v2.docx 1 of 2



Valparaiso University

College of Engineering

ECE-422: Embedded Microcontrollers II

1. Ok. Challenge Time!

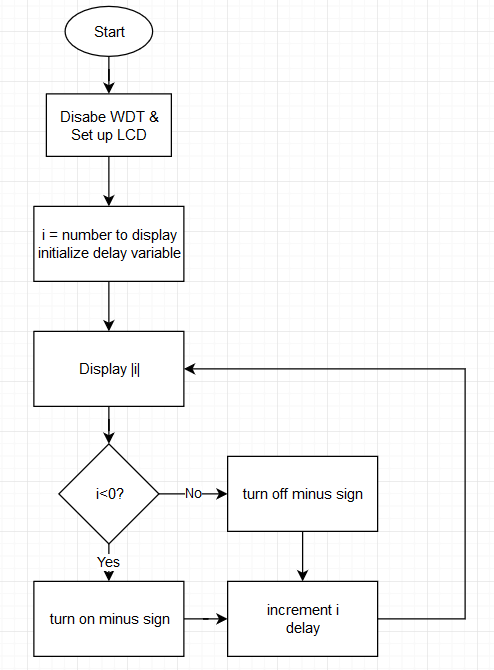
Design a program (make a design flowchart) that can display any number between -999,999 and +999,999.

Recall from the LCD Symbols lab manual that we have a negative sign symbol which can be turned on and off with the following commands:

myLCD\_showSymbol(LCD\_UPDATE , LCD\_NEG , 0); // Turn on negative sign

myLCD\_showSymbol(LCD\_CLEAR , LCD\_NEG , 0); // Turn off negative sign

Turn in your final Design Flowchart attached to this sheet.



1. Implement your flowchart design in C in CCS (no assembly!). Revise and edit your design (and your design flowchart) as necessary to get it working. Print out and turn in your final working code and attach.

/\* ECE 422 - Joe Leveille

\* Assignment 3 - Displaying numbers on LCD

\* Part A

\*/

**#include** <msp430.h>

**#include** <stdlib.h>

**#include** <string.h>

**#include** <driverlib.h> // Required for the LCD

**#include** "myGpio.h" // Required for the LCD

**#include** "myClocks.h" // Required for the LCD

**#include** "myLcd.h" // Required for the LCD

**#include** <time.h>

**main**() {

WDTCTL = 0x5A80; // Disable WDT (poor doggy!)

initGPIO(); // Initializes Inputs and Outputs for LCD

initClocks(); // Initialize clocks for LCD

myLCD\_init(); // Prepares LCD to receive commands

**signed** **long** i = -50; // Number to be displayed

**unsigned** **long** j = 0; // For delay

**while** (1) {

myLCD\_displayNumber(**abs**(i)); // Display the number

**if**(i<0){

myLCD\_showSymbol(LCD\_UPDATE , LCD\_NEG , 0); // Turn on negative sign

} **else**{

myLCD\_showSymbol(LCD\_CLEAR , LCD\_NEG , 0); // Turn off negative sign

}

i = i + 1; // Increment the number

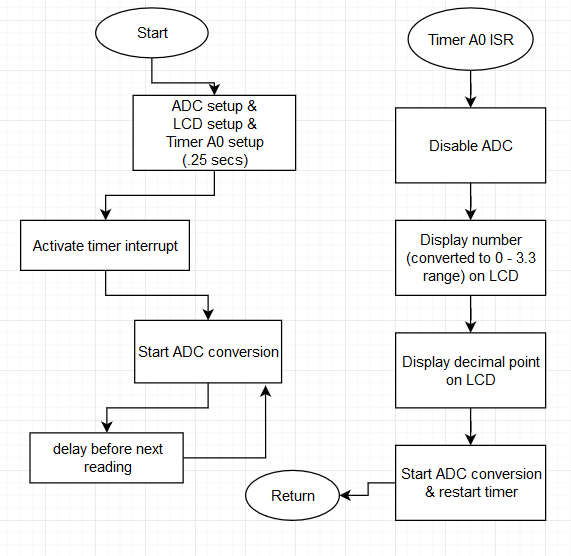
**for** (j = 0; j < 123456; j++); // Delay

}

}

1. Last Challenge. Let’s use the output of the ADC to display the input voltage on the LCD. Connect a 1k or 5k Ohm potentiometer to your LaunchPad; the Pot wiper is connected to Pin P9.2 on your LaunchPad with one side of your Pot connected to a 3.3V pin and the other side of the Pot connected to a GND pin.

Design a program (make a design flowchart) that every 0.25 seconds will display the voltage being input to the ADC. Turn in your final Design Flowchart attached to this sheet.



1. Implement your flowchart design in C in CCS (no assembly!). Revise and edit your design (and your design flowchart) as necessary to get it working. Print out and turn in your final working code and attach.

/\* ECE 422 - Joe Leveille

\* Assignment 3 - Displaying numbers on LCD

\* Part B

\*/

**#include** <msp430.h>

**#include** <stdlib.h>

**#include** <string.h>

**#include** <driverlib.h> // Required for the LCD

**#include** "myGpio.h" // Required for the LCD

**#include** "myClocks.h" // Required for the LCD

**#include** "myLcd.h" // Required for the LCD

**#include** <time.h>

**#define** ACLK 0x0100 // Timer ACLK source

**#define** UP 0x0010 // Timer Up mode

**void** **ADC\_SETUP**(**void**){

**#define** ADC12\_SHT\_16 0x0200 // 16 clock cycles for sample and hold

**#define** ADC12\_ON 0x0010 // Used to turn ADC12 peripheral on

**#define** ADC12\_SHT\_SRC\_SEL 0x0200 // Selects source for sample & hold

**#define** ADC12\_12BIT 0x0020 // Selects 12-bits of resolution

**#define** ADC12\_P84 0x0007 // Use input P8.4 for analog input

**#define** ADC12\_P85 0x0006 // Use input P8.5 for analog input

**#define** ADC12\_P86 0x0005 // Use input P8.6 for analog input

**#define** ADC12\_P87 0x0004 // Use input P8.7 for analog input

**#define** ADC12\_P90 0x0008 // Use input P9.0 for analog input

**#define** ADC12\_P91 0x0009 // Use input P9.1 for analog input

**#define** ADC12\_P92 0x000A // Use input P9.2 for analog input

**#define** ADC12\_P93 0x000B // Use input P9.3 for analog input

**#define** ADC12\_P95 0x000D // Use input P9.5 for analog input

**#define** ADC12\_P96 0x000E // Use input P9.6 for analog input

ADC12CTL0 = ADC12\_SHT\_16 | ADC12\_ON ; // Turn on, set sample & hold time

ADC12CTL1 = ADC12\_SHT\_SRC\_SEL; // Specify sample & hold clock source

ADC12CTL2 = ADC12\_12BIT; // 12-bit conversion results

ADC12MCTL0 = ADC12\_P92; // P9.2 is analog input

}

**main**() {

ADC\_SETUP(); // Sets up ADC peripheral

WDTCTL = 0x5A80; // Disable WDT (poor doggy!)

initGPIO(); // Initializes Inputs and Outputs for LCD

initClocks(); // Initialize clocks for LCD

myLCD\_init(); // Prepares LCD to receive commands

TA0CCR0 = 9600; //set timer for quarter second

TA0CTL = ACLK + UP + TACLR;

TA0CCTL0 = CCIE; // Enable interrupt for Timer\_0

**int** delay;

\_BIS\_SR(GIE); // Activate interrupts

**while**(1){

ADC12CTL0 = ADC12CTL0 | ADC12ENC; // Enable conversion

ADC12CTL0 = ADC12CTL0 | ADC12SC; // Start next conversion

**for**(delay=0 ; delay<1234 ; delay=delay+1); // Delay between shifts)

}

}

**#pragma** vector=TIMER0\_A0\_VECTOR

**\_\_interrupt** **void** **timerInterrupt**(**void**) {

ADC12CTL0 = ADC12CTL0 & (~ADC12ENC); // Need to disable peripheral

myLCD\_displayNumber((ADC12MEM0\*330000)/4096); // Display the number

myLCD\_showSymbol(LCD\_UPDATE, LCD\_A1DP, 0); // Turn on decimal point

ADC12CTL0 = ADC12CTL0 | ADC12ENC; // Re-enable conversion

ADC12CTL0 = ADC12CTL0 | ADC12SC; // Start next conversion

TA0CTL |= TACLR;

TA0CTL &= ~TAIFG;

}

PBW ECE-422 Assignment 3 - MSP430 Large Numbers on the LCD - Spring 2020 v2.docx 2 of 2