

LCD Challenge 1

As before, lots of ways to do this. This is not very fancy, but we tried to develop the program so it was very straightforward to read.

```
#include <msp430.h>
                      // Required for the LCD
#include <driverlib.h>
                                // Required for the LCD
#include "myGpio.h"
#include "myClocks.h"
                                // Required for the LCD
#include "myLcd.h"
                                // Required for the LCD
#define ENABLE PINS
                      0xFFFE
void
        ADC_SETUP(void);
                               // Used to setup ADC12 peripheral
main()
{
   unsigned long i;
                                // Use for delay
                               // Stop WDT
   WDTCTL = WDTPW | WDTHOLD;
   PM5CTL0 = ENABLE_PINS;
                                  // Enable inputs and outputs
                                // Initializes Inputs and Outputs for LCD
    initGPIO();
                                // Initialize clocks for LCD
    initClocks();
                                  // Prepares LCD to receive commands
    myLCD init();
   myLCD_showSymbol(LCD_UPDATE , LCD_BRACKETS , 0); // Brackets on
   ADC_SETUP();
                                           // Sets up ADC peripheral
   ADC12IER0 = ADC12IE0;
                                           // Enable ADC interrupt
                                            // Activate interrupts
   _BIS_SR(GIE);
      ADC12CTL0 = ADC12CTL0 | ADC12ENC; // Enable conversion
      ADC12CTL0 = ADC12CTL0 | ADC12SC;
                                            // Start conversion
      while(1);
}
```



```
//* ADC12 Interrupt Service Routine**************************
#pragma vector = ADC12_VECTOR
 interrupt void ADC12 ISR(void)
   // *** If slide approximately more than 10% up ******************
   if (ADC12MEM0 > 0x800)
                                              //*** Turn on bar 1
     myLCD_showSymbol(LCD_UPDATE , LCD_B1 , 0);
   }
   else
   {
     myLCD_showSymbol(LCD_CLEAR , LCD_B1 , 0);
   // *** If slide approximately more than 20% up *****************
   if (ADC12MEM0 > 0xA00)
                                              //*** Turn on bar 2
     myLCD_showSymbol(LCD_UPDATE , LCD_B2 , 0);
   }
   else
   {
     myLCD_showSymbol(LCD_CLEAR , LCD_B2 , 0);
   }
   // *** If slide approximately more than 30% up *****************
                                              //*** Turn on bar 3
   if (ADC12MEM0 > 0xC00)
     myLCD_showSymbol(LCD_UPDATE , LCD_B3 , 0);
   }
   else
     myLCD_showSymbol(LCD_CLEAR , LCD_B3 , 0);
   }
   // *** If slide approximately more than 40% up ****************
   if (ADC12MEM0 > 0xD00)
                                              //*** Turn on bar 4
     myLCD_showSymbol(LCD_UPDATE , LCD_B4 , 0);
   }
   else
   {
     myLCD_showSymbol(LCD_CLEAR , LCD_B4 , 0);
   }
```



```
// *** If slide approximately more than 50% up **************
   if (ADC12MEM0 > 0xE00)
                                          //*** Turn on bar 5
   {
     myLCD_showSymbol(LCD_UPDATE , LCD_B5 , 0);
   }
   else
   {
     myLCD_showSymbol(LCD_CLEAR , LCD_B5 , 0);
   }
   // *** If slide approximately more than 70% up ****************
   if (ADC12MEM0 > 0xF00)
                                          //*** Turn on bar 6
   {
     myLCD_showSymbol(LCD_UPDATE , LCD_B6 , 0);
   }
   else
   {
     myLCD_showSymbol(LCD_CLEAR , LCD_B6 , 0);
   }
   ADC12CTL0 = ADC12CTL0 | ADC12SC;
                               // Start next conversion
}
//* Configure Analog-to-Digital Converter peripheral**************
void ADC_SETUP(void)
                                      // 16 clock cycles for sample and hold
   #define ADC12 SHT 16
                          0x0200
                                      // Used to turn ADC12 peripheral on
   #define ADC12_ON
                          0x0010
   #define ADC12_SHT_SRC_SEL 0x0200
                                      // Selects source for sample & hold
                                      // Selects 12-bits of resolution
   #define ADC12 12BIT
                          0x0020
                          0x000A
   #define ADC12 P92
                                      // Use input P9.2 for analog input
   ADC12CTL0 = ADC12_SHT_16 | ADC12_ON; // Turn on, set sample & hold time
                                      // Specify sample & hold clock source
   ADC12CTL1 = ADC12 SHT SRC SEL;
   ADC12CTL2 = ADC12_12BIT;
                                      // 12-bit conversion results
   ADC12MCTL0 = ADC12 P92;
                                       // P9.2 is analog input
}
```



All tutorials and software examples included herewith are intended solely for educational purposes. The material is provided in an "as is" condition. Any express or implied warranties, including, but not limited to the implied warranties of merchantability and fitness for particular purposes are disclaimed.

The software examples are self-contained low-level programs that typically demonstrate a single peripheral function or device feature in a highly concise manner. Therefore, the code may rely on the device's power-on default register values and settings such as the clock configuration and care must be taken when combining code from several examples to avoid potential side effects.

Additionally, the tutorials and software examples should not be considered for use in life support devices or systems or mission critical devices or systems.

In no event shall the owner or contributors to the tutorials and software be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.