HW #9

ECEN 621

Mrinmoy Sarkar

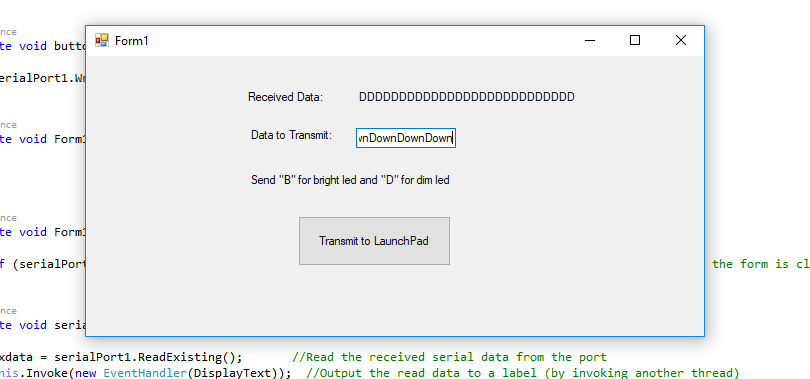
Date: 11/13/2019**LaunchPad Implementation:**

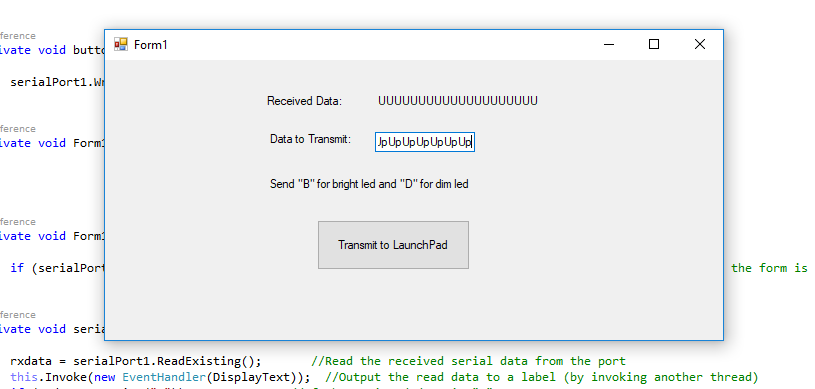
|  |
| --- |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#include** "msp.h"  **volatile** uint16\_t JOYresult; // store the ADC value of the joystick  // Configure system clock to 12Mz  // Alternatively you may simply define \_SYSTEM\_CLOCK as 12000000 in the system\_msp432p401r.c file  **void** **clock\_config\_12MHz**(){  CS->KEY = CS\_KEY\_VAL; // Unlock CS module for register access  CS->CTL0 = 0; // Reset tuning parameters  CS->CTL0 = CS\_CTL0\_DCORSEL\_3; // Set DCO to 12MHz (nominal, center of 8-16MHz range)  CS->CTL1 = CS\_CTL1\_SELA\_2 | // Select ACLK = REFO  CS\_CTL1\_SELS\_3 | // SMCLK = DCO  CS\_CTL1\_SELM\_3; // MCLK = DCO  CS->KEY = 0; // Lock CS module from unintended accesses  }  //Congigure neeed I/O ports  **void** **port\_config**(){  P1->SEL0 |= BIT2 | BIT3; // select the eUSCI funtions of P1.2 and P1.3  P1->DIR &= ~BIT2; // P1.2 is the receive data (RXD) input pin  P1->DIR |= BIT3; // P1.3 is the transmit data (TXD) input pin  // P1->DIR |= BIT0; // The Red Led will be used for visual output  // P1->DIR &= ~(BIT1|BIT4); // Configure pins P1.1 and P1.4 as inputs for S1 and S2 switches  // P1->REN |= (BIT1|BIT4); // Activate pull resistors for S1 and S2  // P1->OUT |= (BIT1|BIT4); // Connect Pull resistors to Vcc to make them pull-up resistors  //configure pwm for MKII Red led  P2->DIR |= BIT6;  P2->OUT |= BIT6;  P2->SEL0 |= BIT6;  TIMER\_A0->CCR[0]=999; //period  TIMER\_A0->CCR[3]=500; // duty cycle  TIMER\_A0->CCTL[3]=TIMER\_A\_CCTLN\_OUTMOD\_7;  TIMER\_A0->CTL=TIMER\_A\_CTL\_TASSEL\_2 | TIMER\_A\_CTL\_MC\_1 | TIMER\_A\_CTL\_CLR;  // configure ADC for JoyStick  // Configure GPIO for ADC  P4->SEL1 |= BIT4; // Enable A/D channel A9  P4->SEL0 |= BIT4;  // Turn on ADC14, extend sampling time to avoid overflow of results  ADC14->CTL0 = ADC14\_CTL0\_ON | ADC14\_CTL0\_MSC | ADC14\_CTL0\_SHT0\_\_192 | ADC14\_CTL0\_SHP | ADC14\_CTL0\_CONSEQ\_3;  ADC14->MCTL[0] = ADC14\_MCTLN\_INCH\_9; // ref+=AVcc, channel = A9  ADC14->MCTL[1] = ADC14\_MCTLN\_EOS;  ADC14->IER0 = ADC14\_IER0\_IE1; // ADC interrupt enable for ADC14->MEM[1]  SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // sleep on exit  // Start conversion-software trigger  ADC14->CTL0 |= ADC14\_CTL0\_ENC | ADC14\_CTL0\_SC;  NVIC->ISER[0] = 1 << ((*ADC14\_IRQn*) & 31); // Enable ADC interrupt in NVIC module  }  // Configure the eUSCI for 9600 Baud UART communication  **void** **uart\_config**(){  EUSCI\_A0->CTLW0 |= EUSCI\_A\_CTLW0\_SWRST; // Put eUSCI in reset  EUSCI\_A0->CTLW0 = EUSCI\_A\_CTLW0\_SWRST | // Remain eUSCI in reset  EUSCI\_B\_CTLW0\_SSEL\_\_SMCLK | // Configure eUSCI clock source for SMCLK  EUSCI\_A\_CTLW0\_PEN | // enable parity  EUSCI\_A\_CTLW0\_PAR; //even parity  // Baud Rate 19200  EUSCI\_A0->BRW = 39; // 12000000/16/9600  EUSCI\_A0->MCTLW = (0x00 << EUSCI\_A\_MCTLW\_BRS\_OFS) | (1 << EUSCI\_A\_MCTLW\_BRF\_OFS) |  EUSCI\_A\_MCTLW\_OS16;  EUSCI\_A0->CTLW0 &= ~EUSCI\_A\_CTLW0\_SWRST; // Initialize eUSCI  EUSCI\_A0->IFG &= ~EUSCI\_A\_IFG\_RXIFG; // Clear eUSCI RX interrupt flag  EUSCI\_A0->IE |= EUSCI\_A\_IE\_RXIE; // Enable USCI\_A0 RX interrupt  }  // Main function  **int** **main**(**void**)  {  clock\_config\_12MHz(); //Or simply define \_SYSTEM\_CLOCK as 12000000 in the system\_msp432p401r.c file  port\_config();  uart\_config();  NVIC->ISER[0] = 1 << ((*EUSCIA0\_IRQn*) & 31);// Enable eUSCIA0 interrupt in NVIC module  \_\_enable\_irq();// Enable global interrupt  // Enter LPM0  \_\_sleep();  }  // UART interrupt service routine for Bright and dim of Red Led OF Booster Pack  **void** **EUSCIA0\_IRQHandler**(**void**)  {  **if** (EUSCI\_A0->IFG & EUSCI\_A\_IFG\_RXIFG) // Whenever a character is received  {  **while**(!(EUSCI\_A0->IFG & EUSCI\_A\_IFG\_TXIFG)); // Check if the TX buffer is empty first  //EUSCI\_A0->TXBUF = EUSCI\_A0->RXBUF; // Echo received from the PC character back to the PC  **if**(EUSCI\_A0->RXBUF =='B') //If a "B" is received from the PC Keypboard  {  TIMER\_A0->CCR[3]=900; // duty cycle  }  **else** **if**(EUSCI\_A0->RXBUF =='D') //If a "D" is received from the PC Keypboard  {  TIMER\_A0->CCR[3]=100; // duty cycle  }  }  }  // ADC14 interrupt service routine  **void** **ADC14\_IRQHandler**(**void**)  {  **if** (ADC14->IFGR0 & ADC14\_IFGR0\_IFG1)  {  JOYresult = ADC14->MEM[0]; // Move A9 results, IFG is cleared. vertical joystick  **if**(JOYresult < 3000)  {  EUSCI\_A0->TXBUF = 'D'; // send 'D' to PC  }  **else** **if**(JOYresult > 10000)  {  EUSCI\_A0->TXBUF = 'U'; // send 'U' to PC  }  \_\_delay\_cycles(2000); //small delay  }  } |

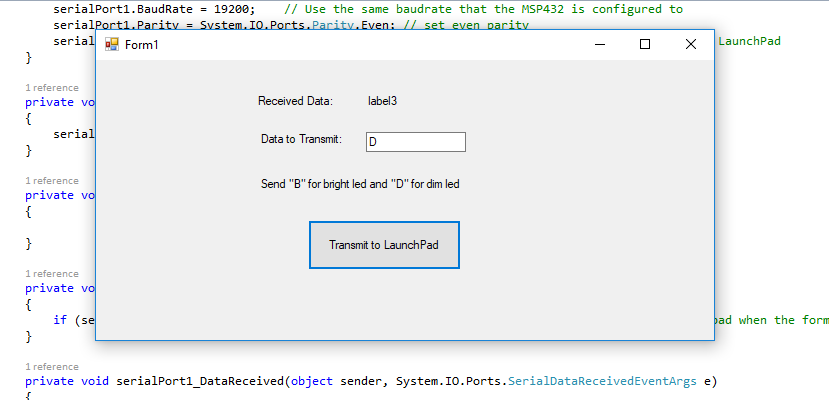
**Windows Form App:**

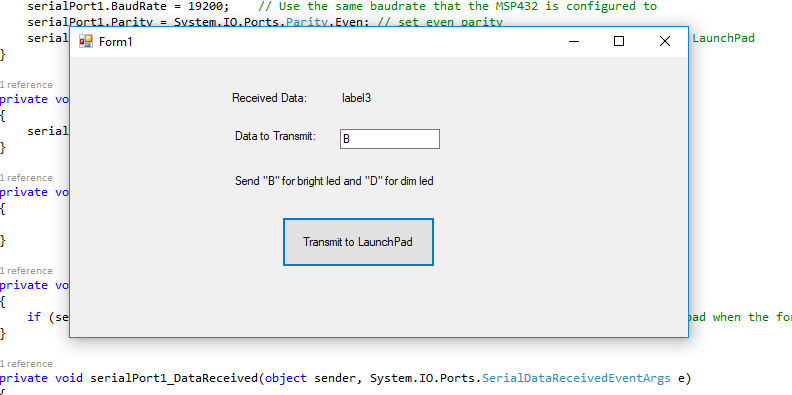
|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace WindowsFormsApplication2  {  public partial class Form1 : Form  {  String rxdata; // serial data received  public Form1()  {  InitializeComponent();  }  private void Form1\_Load(object sender, EventArgs e)  {  serialPort1.PortName = "COM8"; // Choose the UART serial port associated with the MSP432 LaunchPad (See "Device Manager")  serialPort1.BaudRate = 19200; // Use the same baudrate that the MSP432 is configured to  serialPort1.Parity = System.IO.Ports.Parity.Even; // set even parity  serialPort1.Open(); // Open the UART serial port for communication witht he MSP432 LaunchPad  }  private void button1\_Click(object sender, EventArgs e)  {  serialPort1.Write(textBox1.Text); // Send the data from the textbox to the serial port  }  private void Form1\_FormClosed(object sender, FormClosedEventArgs e)  {  }  private void Form1\_FormClosing(object sender, FormClosingEventArgs e)  {  if (serialPort1.IsOpen) serialPort1.Close(); // Close the connection to the MSP432 Launchpad when the form is closed (odApp exited)  }  private void serialPort1\_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)  {  rxdata = serialPort1.ReadExisting(); //Read the received serial data from the port  this.Invoke(new EventHandler(DisplayText)); //Output the read data to a label (by invoking another thread)  if (rxdata.Contains("D")) //If the recieved data is "D"  {  SendKeys.SendWait("Down"); //so send the letters "D", "o","w", and "n" to the current Windows App in the forground (whatever the App may be)  }  else if (rxdata.Contains("U")) //If the recieved data is "U"  {  SendKeys.SendWait("Up"); //so send the letters "U", and "p" to the current Windows App in the forground  }  }  private void DisplayText(object sender, EventArgs e)  {  label3.Text = rxdata; //Output the received data to the label  }  }  } |

**Screenshot of Windows Form App:**

****

****

****

****