Sheridan

Course: **ENGI30000 Embedded Systems Applications**

Project #: **3**

Project Title: **UART and ADC interface for "RUN LED" project**

Professor: **MD – Nazrul Khan**

Marking Scheme	Marks
Demonstration	3
Coding	3
Report Content	4

Submitted by:

1): Rishab Singh

2): -NA-

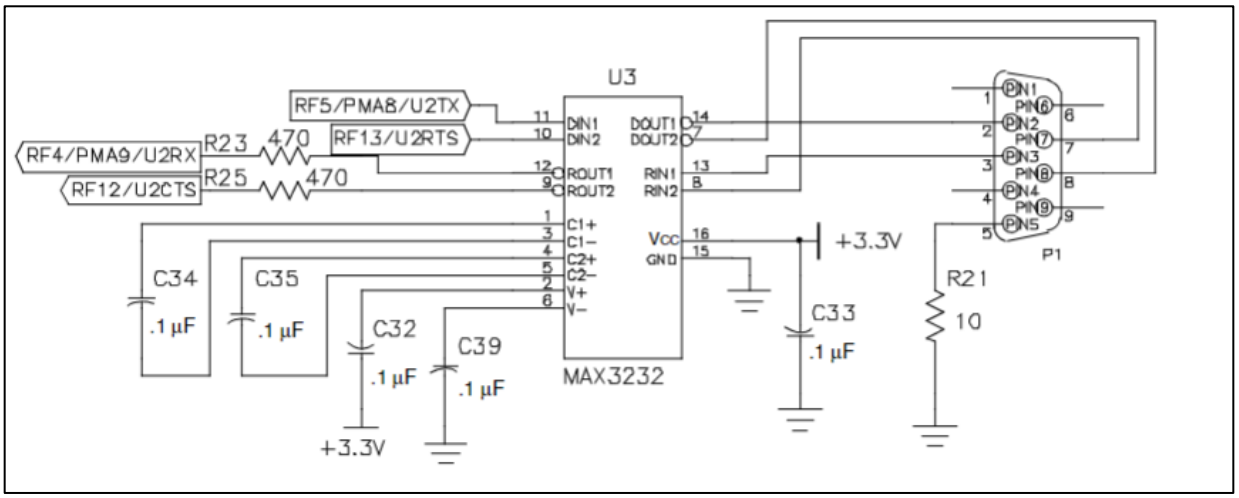
Date of Submission: November 14, 2022

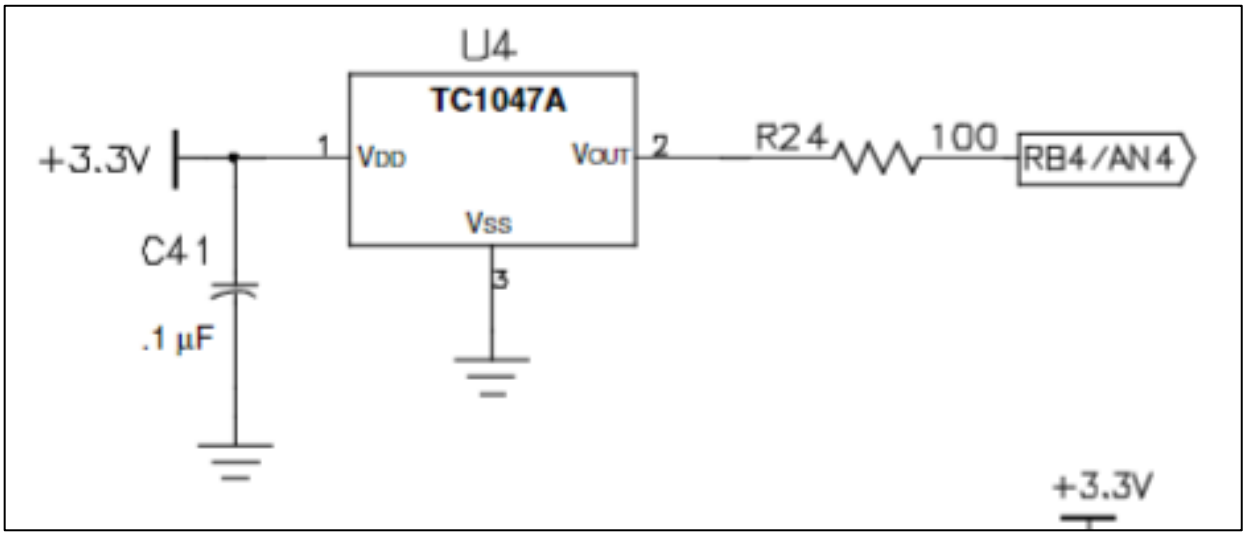
Notes:

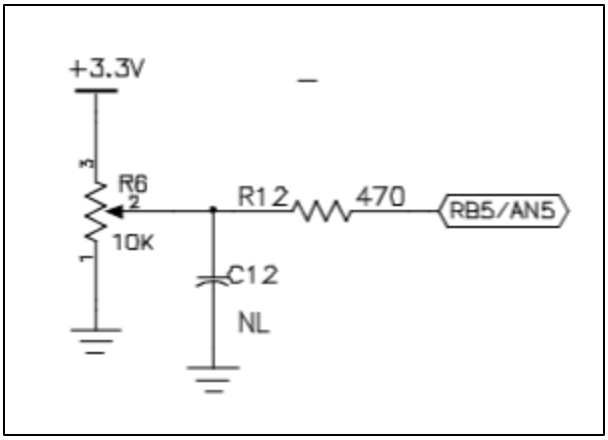
Name and Objective (in short):

The name if the project is “UART and ADC interface for RUN LED project and sensing temperature and pot voltage”, and the objective of this project is to measure the current temperature and potentiometer voltage and use UART and A/D convertor to establish communication between the Explorer 15 board and the personal device.

Schematic:







Configuration

To perform an A/D conversion:

1. Configure the A/D module:
   a) Select the port pins as analog inputs (AD1PCFG<15:0>).
   b) Select a voltage reference source to match the expected range on the analog inputs (AD1CON2<15:13>).
   c) Select the analog conversion clock to match the desired data rate with the processor clock (AD1CON3<7:0>).
   d) Select the appropriate sample/conversion sequence (AD1CON1<7:0> and AD1CON3<12:8>).
   e) Select how conversion results are presented in the buffer (AD1CON1<9:8>).
   f) Select the interrupt rate (AD1CON2<5:2>).
   g) Turn on the A/D module (AD1CON1<15>).
2. Configure the A/D interrupt (if required):
   a) Clear the AD1IF bit.
   b) Select the A/D interrupt priority.

17.2 Transmitting in 8-Bit Data Mode

1. Set up the UARTx:
   a) Write appropriate values for data, parity and Stop bits.
   b) Write appropriate baud rate value to the UBRGx register.
   c) Set up transmit and receive interrupt enable and priority bits.
2. Enable the UARTx.
3. Set the UTXEN bit (causes a transmit interrupt).
4. Write data byte to lower byte of UTXxREG word. The value will be immediately transferred to the Transmit Shift Register (TSR) and the serial bit stream will start shifting out with the next rising edge of the baud clock.
5. Alternately, the data byte may be transferred while UTXEN = 0 and then the user may set UTXEN. This will cause the serial bit stream to begin immediately because the baud clock will start from a cleared state.
6. A transmit interrupt will be generated as per interrupt control bit, UTXISELx.

We are following the above figures from the PIC24 datasheet to configure the UART2 and ADC properties.

For UART2:

- We set the value of U2Mode register to enable the UART (bit15) and set the parity bit (bit1 and bit2) as well as bit13 has been set to 0 so BRG=0 hence.

- Then we set the U2BRG register to get the desired Baud Rate of 9600.

```
void UART2_Initialize (void)
{
    /**
     * Enable the UART (bit 15) and
     * set parity bit to "even parity"
     * (bit 2=0, bit 1=1)
     */
    U2MODE = 0x8002;

    U2STA = 0x0000; //resetting U2STA register
    /**
     * Clearing the U2 transmit register (<- lower half of the register)
     * to prepare for transmission
     */
    U2TXREG = 0x0000;

    /**
     * Baud Rate = 9600 = 0x0019;
     * Frequency = 4 MHz;
     */
    U2BRG = 0x0019;

    /**
     * Enabling Rx Interrupt
     * to detect receiving information
     */
    IEC1bits.U2RXIE = 1;

    /**
     * Enabling transmission by
     * setting the enable transmit bit = 1
     */
    U2STAbits.UTXEN = 1;

    /**
     * We don't need Tx Interrupts so
     * we disable it
     */
    IEC1bits.U2TXIE = 0; //Tx interrupt disabled
```

```
    /**
     * Setting the interrupt flag bits for
     * Rx and Tx interrupts
     */
    IFS1bits.U2RXIF = 0;
    IFS1bits.U2TXIF = 0;

    /**
     * Setting the priority of
     * the Rx interrupts to 5
     */
    IPC7bits.U2RXIP2 = 1;
    IPC7bits.U2RXIP1 = 0;
    IPC7bits.U2RXIP0 = 1;
}
```

For ADC:

- In the ADC_Initialize() function, we set the **AD1PCFG** to 0xFFCF to set the pin4 and pin5 as Analog inputs (1 for temperature and 1 for pot voltage).

- **AD1CHS** value has to be changed to get the value of pin4 and pin5 as analog input so we set the value of the 0x0004 and 0x0005 as needed.

- **AD1CON1bits.ADON** has to be set to 1 to turn on the Analog to Digital Converter.

- AD1CON3 has been set to 0x1100 so ADCS is set to 0 and the SAMC has been set to 0x0001.

Calculations (Current/voltage/resistance/Time/Frequency/ etc.,):

Converting physical quantities

$$(Physical - quantity \times Resolution) - Offset$$

In Code:-

```
temperature = ADC_Operate(4); //using ANI4
temperature = (((temperature *0.00322) -0.5)/0.01);

potVoltage = ADC_Operate(5);  //using ANI5
potVoltage = potVoltage * 0.00322;
```

Resolution of the Device

The resolution of this device is

Resolution (R) = $\frac{V_{ref}}{2^n - 1}$

For 10-bit ADC,

Resolution (R) = $\frac{3.3V}{2^{10} - 1}$ = 3.22 mV

UBRGX

The BRGH value is set to 0.

$$Baud\ Rate = 9600 \frac{bits}{sec}$$

$$U2BRG = \left(\frac{F_{cy}}{16\ x\ Baud\ Rate}\right) - 1$$

TAD, Auto sampling time and Conversion Time

$$T_{cy} = \frac{1}{F_{cy}} = \frac{1}{(\frac{Fosc}{2})} = 0.25\ us$$

As ADCS has been set to zero,

$$T_{AD} = ADCS \times T_{cy} = 1 \times T_{cy} = 0.25\ us$$

As SAMC (bit8-bit12) has been set to 0001,

$$Auto\ sampling\ time\ (T_s) = SAMC \times T_{AD} = 1 \times T_{AD} = 0.25us$$

$$Conversion\ Time\ (t) = 12 \times T_{AD} = 12 \times 0.25us = 3us$$

Critical Observation/Thinking (One paragraph):

To make this project functional, we have to setup ADC and UART in the previous project (RUN LED) so that we can measure the value of the temperature and the potentiometer and use UART to establish communication between the Explorer 16 board and our laptop and print the measured values on our laptop.

We configure the required ADC and UART in the ADC_Initialize() and UART_Initalize() functions that configure the respective UART and ADC modules.

We have modified the ADC_Operate({var}) value and passed a {var} parameter to the function to set the value of **AD1CHS** as required so for temperature, we call ADC_Operate(4) and for potentiometer value, we call ADC_Operate(5). We these functions in the while loop and with every loop we collect the value of temperature and potentiometer. Then we use the **sprintf** function to generate a combined string and then pass that as a parameter to U2_TxString() function.

CODE:

```
//Smit Chawda//11132022//991561339

#include <xc.h>
#include <stdint.h>
#include <stdbool.h>

#define _XTAL_FREQ  8000000UL

// CONFIG2
#pragma config POSCMOD = XT      // Primary Oscillator Select->XT Oscillator mode selected
#pragma config OSCIOFNC = OFF    // Primary Oscillator Output Function->OSC2/CLKO/RC15 functions as CLKO (FOSC/2)
#pragma config FCKSM = CSDCMD    // Clock Switching and Monitor->Clock switching and Fail-Safe Clock Monitor are disabled
#pragma config FNOSC = PRI       // Oscillator Select->Primary Oscillator (XT, HS, EC)
#pragma config IESO = ON         // Internal External Switch Over Mode->IESO mode (Two-Speed Start-up) enabled
// CONFIG1
#pragma config WDTPS = PS32768   // Watchdog Timer Postscaler->1:32768
#pragma config FWPSA = PR128     // WDT Prescaler->Prescaler ratio of 1:128
#pragma config WINDIS = ON       // Watchdog Timer Window->Standard Watchdog Timer enabled,(Windowed-mode is disabled)
#pragma config FWDTEN = OFF      // Watchdog Timer Enable->Watchdog Timer is disabled
#pragma config ICS = PGx2        // Comm Channel Select->Emulator/debugger uses EMUC2/EMUD2
//#pragma config COE = OFF       // Set Clip On Emulation Mode->Reset Into Operational Mode
#pragma config BKBUG = OFF       // Background Debug->Device resets into Operational mode
#pragma config GWRP = OFF        // General Code Segment Write Protect->Writes to program memory are allowed
#pragma config GCP = OFF         // General Code Segment Code Protect->Code protection is disabled
#pragma config JTAGEN = OFF      // JTAG Port Enable->JTAG port is disabled

////////////////////////////////////////////////////////////////////////////
//Function Proto type
```

```
void SYSTEM_Initialize(void);
void OSCILLATOR_Initialize(void);
void PIN_MANAGER_Initialize(void);
void INTERRUPT_Initialize (void);
void TIMER_Initialize(void);

void delayFunc(void);
void projKingRiderMainFunction(void);
void projReverseKingRiderMainFunction(void);
void resetPattern(void);
void startTimerAndInterrupt(void);
void __attribute__ ((__interrupt__,no_auto_psv)) _T1Interrupt(void);

////////////////////////////////////////////////////////////////////////////////
//Global variables
unsigned int count1 = 200;
unsigned int count2 = 200;
unsigned int buffer_variable = 0;
char rcvByte;
float temperature =0.0f;
float potVoltage = 0.0f;

////////////////////////////////////////////////////////////////////////////////
//Program Entry Point
int main(void)
{
 // initialize the device
 SYSTEM_Initialize();
 TRISDbits.TRISD6 = 1;
 printf("#------ PROJECT 3 - Smit Chawda - 991561339 ------#");
 while (1)
 {
    char TxString[100] = "";

    temperature = ADC_Operate(4); //using ANI4
    temperature = (((temperature *0.00322) -0.5)/0.01);
```

```
    potVoltage = ADC_Operate(5);  //using ANI5
    potVoltage = (potVoltage * 0.00322);

    sprintf(TxString,"\n\n#> Voltage: %4.2f       Temperature: %4.2f", potVoltage,
temperature);
    U2_TxString(TxString);

    if(PORTDbits.RD6 == 1)
    {
        projReverseKingRiderMainFunction();
    }
    else
    {
       projKingRiderMainFunction();
    }
 }
   return -1; //Never reach here!
}

void __attribute__ ((__interrupt__,no_auto_psv)) _T1Interrupt(void)
{
   if(buffer_variable < 100)
   {
      buffer_variable++;
   }
   if(buffer_variable >= 100)
   {
      buffer_variable = 0;
//       PORTA ^= 0x80; //toggling bit 7 of register A - RA7 - LED10
      if(PORTAbits.RA7 == 0)
      {
         PORTAbits.RA7 = 1;
      }
      else {
         PORTAbits.RA7 = 0;
      }
   }
   IFS0bits.T1IF = 0;
```

```
}

void __attribute__ ( ( interrupt, no_auto_psv ) ) _U2RXInterrupt( void )
{
    if(IEC1bits.U2RXIE == 1)
    {
        if(IFS1bits.U2RXIF == true) //check the status flag
        {
            rcvByte = U2RXREG; //Read the received byte
        }
    }
    IFS1bits.U2RXIF = false; //reset the status flag
}

void __attribute__ ((__interrupt__,__no_auto_psv__)) _ADC1Interrupt (void)
{
    if (IFS0bits.AD1IF == 1) {
//        v = voltageValue = ADC1BUF0;
    }
    IFS0bits.AD1IF = 0;
}

void projKingRiderMainFunction(void)
{
    PORTAbits.RA0 = 1;
    delayFunc();
    PORTAbits.RA1 = 1;
    delayFunc();
    PORTAbits.RA2 = 1;
    delayFunc();
    PORTAbits.RA3 = 1;
    delayFunc();
    PORTAbits.RA4 = 1;
    delayFunc();
    PORTAbits.RA5 = 1;
    delayFunc();
    PORTAbits.RA6 = 1;
```

```
    delayFunc();
//    PORTAbits.RA7 = 1;
    delayFunc();
    resetPattern();
}

void projReverseKingRiderMainFunction(void)
{
//    PORTAbits.RA7 = 1;
    delayFunc();
    PORTAbits.RA6 = 1;
    delayFunc();
    PORTAbits.RA5 = 1;
    delayFunc();
    PORTAbits.RA4 = 1;
    delayFunc();
    PORTAbits.RA3 = 1;
    delayFunc();
    PORTAbits.RA2 = 1;
    delayFunc();
    PORTAbits.RA1 = 1;
    delayFunc();
    PORTAbits.RA0 = 1;
    delayFunc();
    resetPattern();
//    projKingRiderMainFunction();
}

void resetPattern(void)
{
    PORTA &= (0x8000);
}

void UART2_Initialize (void)
{
    /**
     * Enable the UART (bit 15) and
```

```
 * set parity bit to "even parity"
 * (bit 2=0, bit 1=1)
 */
U2MODE = 0x8002;

U2STA = 0x0000; //resetting U2STA register
/**
 * Clearing the U2 transmit register (<- lower half of the register)
 * to prepare for transmission
 */
U2TXREG = 0x0000;

/**
 * BaudRate = 9600 = 0x0019;
 * Frequency = 4 MHz;
 */
U2BRG = 0x0019;

/**
 * Enabling Rx Interrupt
 * to detect receving information
 */
IEC1bits.U2RXIE = 1;

/**
 * Enabling transmission by
 * setting the enable transmit bit = 1
 */
U2STAbits.UTXEN = 1;

/**
 * We don't need Tx Interrupts so
 * we disable it
 */
IEC1bits.U2TXIE = 0; //Tx interrupt disabled

/**
```

```
    * Setting the interrupt flag bits for
    * Rx and Tx interrupts
    */
   IFS1bits.U2RXIF = 0;
   IFS1bits.U2TXIF = 0;

   /**
    * Setting the priority of
    * the Rx interrupts to 5
    */
   IPC7bits.U2RXIP2 = 1;
   IPC7bits.U2RXIP1 = 0;
   IPC7bits.U2RXIP0 = 1;
}

void U2_TxByte(char value)
{
   while (!U2STAbits.TRMT);
   U2TXREG = value;

   //delay
   Nop();
   Nop();
   Nop();
}

void U2_TxString(char *str)
{
   char nextChar;
   while(*str != '\0')
   {
      nextChar = *str++;
      Nop();
      Nop();

      U2_TxByte(nextChar);
   }
```

```
    U2_TxByte(0x0d); //CR - print \r
    U2_TxByte(0x0a); //LF - print \n
}

void ADC_Initialize ( void )
{

    /**
     * Setting pin4 and pin5 as a analog input
     */
    AD1PCFG = 0xFFCF;

    /**
     * Start the sample enable bit
     * from bit2 and set the SSRC<2:0>
     * to 111 - (Internal counter ends sampling
     * and starts conversion (auto-convert))
     */
    AD1CON1 = 0x00E2;


    AD1CON2 = 0x0000;

    /**
     * AD1CON3 value bit7 to bit0 has been set to 0 so
     * that ADCS is set to zero and TAD = Tcy
     */
    AD1CON3 = 0x1100;
    AD1CSSL = 0x0000;
    /**
     * Setting the initial value for AD1CHS to 0x0004
     */
    AD1CHS = 0x0004;
    AD1CON1bits.ADON = 1;
}

int ADC_Operate (int value)
```

```
{
    while (1) {
        AD1CHS = value;
        AD1CON1bits.SAMP = 1;
        while (!AD1CON1bits.DONE);
        return ADC1BUF0;
    }
}

////////////////////////////////////////////////////////////////////////////
void SYSTEM_Initialize(void)
{
    PIN_MANAGER_Initialize();
    OSCILLATOR_Initialize();
    TIMER_Initialize();
    ADC_Initialize();
    UART2_Initialize();
    INTERRUPT_Initialize();
}
////////////////////////////////////////////////////////////////////////////
void OSCILLATOR_Initialize(void)
{
    //NOSC PRI; SOSCEN disabled; OSWEN Switch is Complete;
    __builtin_write_OSCCONL((uint8_t) (0x0200 & 0x00FF));
    // RCDIV FRC/2; DOZE 1:8; DOZEN disabled; ROI disabled;
    CLKDIV = 0x3100;
    // TUN Center frequency;
    OSCTUN = 0x0000;
    // WDTO disabled; TRAPR disabled; SWDTEN disabled; EXTR disabled; POR disabled;
SLEEP disabled; BOR disabled; IDLE disabled; IOPUWR disabled; VREGS disabled; CM
disabled; SWR disabled;
    RCON = 0x0000;
}
////////////////////////////////////////////////////////////////////////////
void PIN_MANAGER_Initialize(void)
{
    /**************************************************************************
     * Setting the Output Latch SFR(s)
```

```
 ******************************************************************************/
LATA = 0x0000;
LATB = 0x0000;
LATC = 0x0000;
LATD = 0x0000;
LATE = 0x0000;
LATF = 0x0000;
LATG = 0x0000;

/******************************************************************************
 * Setting the GPIO Direction SFR(s)
 ******************************************************************************/
TRISA = 0xC600;
TRISB = 0xFFFF;
TRISC = 0xF01E;
TRISD = 0xFFFF;
TRISE = 0x03FF;
TRISF = 0x31FF;
TRISG = 0xF3CF;

/******************************************************************************
 * Setting the Weak Pull Up and Weak Pull Down SFR(s)
 ******************************************************************************/
CNPU1 = 0x0000;
CNPU2 = 0x0000;

/******************************************************************************
 * Setting the Open Drain SFR(s)
 ******************************************************************************/
ODCA = 0x0000;
ODCB = 0x0000;
ODCC = 0x0000;
ODCD = 0x0000;
ODCE = 0x0000;
ODCF = 0x0000;
ODCG = 0x0000;
```

```
    /*****************************************************************************
     * Setting the Analog/Digital Configuration SFR(s)
     *****************************************************************************/
    AD1PCFG = 0x00C0;
}
////////////////////////////////////////////////////////////////////////////////
void INTERRUPT_Initialize (void)
{

}

void TIMER_Initialize(void)
{
    TMR1 = 0x0000;
    //Instructions from Manual
//  1. Set the TON bit (= 1).
    /**
     * Timer setting done at the last
     */
//  2. Select the timer pre scaler ratio using the
    T1CONbits.TCKPS = 0b00;  //1:1 ratio selected
    /**
     * We are using the lowest possible pre scaler to get
     * a more precis time interval facilitating more control
     * over the time
     */
//  3. Set the Clock and Gating modes using the TCS
    T1CONbits.TCS = 0; //using internal clock so TCS=0
    T1CONbits.TGATE = 0; //TGATE is set to 0 as we want the output from the PreScaler to
generate out interrupts

//  4. Set or clear the TSYNC bit to configure synchronous or asynchronous operation.
    T1CONbits.T1SYNC = 1; //turning on synchronization of the clock

//  5. Load the timer period value into the PR1
    PR1 = (uint16_t) 0xE28F; //The value of the pre scaler goes here register.
    //58000 = E290
```

```
    IFS0bits.T1IF = 0;

    // Interrupt priority setting
    // 7 = 0b0111
    IPC0bits.T1IP2 = 1;
    IPC0bits.T1IP1 = 1;
    IPC0bits.T1IP0 = 1;

    IEC0bits.T1IE = 1;
    T1CONbits.TON = 1;
//  Ans: Check the Interrupt_Initialize for more details
}

////////////////////////////////////////////////////////////////////////////////
void delayFunc(void)
{
  int j,k;
  int a;

  for(j = 0; j < count1; j++)
  {
     for(k=0; k < count2; k++)
     {
        a = 0; //How many times (frequency) this instruction getting executed?
     }
  }
}
////////////////////////////////////////////////////////////////////////////////
/**
 End of File
*/
```