Lab Code:

```
//******************************************************************
//ECGR 5101
//Lab 07
//Naseeruddin Lodge
//Srikar Thakkallapally
//Group 12
//
//Two MSP430G2553 are connected to each other. Chip1 is connected
//to a potentiometer and is responsible for the ADC conversion and
//transmits the ADC value using UART. Chip2 receives the ADC value
//and displays it on a quad-digit 7-segment LED.
//******************************************************************
#include <msp430.h>
#include <stdint.h>



//Global Variables
int step1=0, step2=0;



//Prototyped Functions
void ADC_Setup();
void UART_Setup();
int uartReceiveData();
void uartTransmitData(int ADCval);
int Correct_Osciallations();
void DisplayLED(int n_SingleDigit);
void Control_Dx(int n);
void Setup_Chip_one();
void Setup_Chip_two();
void Program_Chip_one();
void Program_Chip_two();



//******************************************************************
//Main Function
//******************************************************************
int main(void){

    //Stop watchdog timer
    WDTCTL = WDTPW | WDTHOLD;


    //Check P1.4 to see if it is connected to GND
    //or VCC. Chip1 is connected to GND and Chip2
    //is connected to VCC. Then setups up the peripherals
    //and pins accordingly.
    if(!(P1IN & BIT4)) Setup_Chip_one();
    else               Setup_Chip_two();
    while(1){

        //Check P1.4 to see if it is connected to GND
        //or VCC. Chip1 is connected to GND and Chip2
        //is connected to VCC. Then uploads the code
        //accordingly.
        if(!(P1IN & BIT4)) Program_Chip_one();
        else               Program_Chip_two();
    }

}



//*****************************************************************
//Name    : ADC_Setup()
//Input   : void
//Returns : void
//
//Function to Setup ADC
//*****************************************************************
void ADC_Setup(){

    //SREF     -> 000b = VR+ = VCC and VR- = VSS
    //ADC10SHT -> 10b = 16 ADC10CLK cycles
    //ADC10ON  -> ADC10 on
    //INCH     -> Input channel select

    ADC10CTL0 = SREF_0 + ADC10SHT_2 + ADC10ON;
    ADC10CTL1 = INCH_3;
    ADC10AE0 |= 0x08;
}



//*****************************************************************
//Name    : UART_Setup()
//Input   : void
//Returns : void
//
//Function to Setup UART
//*****************************************************************
void UART_Setup(){

    //Clear DCO
    DCOCTL = 0;

    //Set to 1MHz
    //MCLK = SMCLK = 1MHZ
    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL = CALDCO_1MHZ;

    //P1.1 = RX = BIT1, P1.2 = TX = BIT2
    P1SEL  |= BIT1 + BIT2;
```

```c
    P1SEL2 |= BIT1 + BIT2;

    //Disable USCI, reset mode
    UCA0CTL1 |= UCSWRST;

    //SMCLK
    UCA0CTL1 |= UCSSEL_2;

    //1MHz
    //Baud Rate -> 9600
    UCA0BR0 = 104;
    UCA0BR1 = 0;

    //Modulation UCBRSx = 1
    UCA0MCTL = UCBRS0;

    //Initialize USCI state machine
    UCA0CTL1 &= ~UCSWRST;

}



//*****************************************************************
//Name    : uartReceiveData()
//Input   : void
//Returns : int
//
//Checks if USCI_A0 RX has been received and returns it
//*****************************************************************
int uartReceiveData(){

    //Check if USCI_A0 RX has been received
    while (!(IFG2 & UCA0RXIFG));

    return UCA0RXBUF;
}



//*****************************************************************
//Name    : uartTransmitData(int ADCval)
//Input   : int
//Returns : void
//
//Checks if USCI_A0 TX buffer is ready and transmits the data
//*****************************************************************
void uartTransmitData(int ADCval){

    //Check if USCI_A0 TX buffer is ready
    while(!(IFG2 & UCA0TXIFG));

    UCA0TXBUF = ADCval;
}
```

```
//*****************************************************************
//Name    : correct_osciallations()
//Input   : void
//Returns : int
//
//This function is responsible for fixing oscillations.
//If n=50 and it oscillates between n=49 and
//n=51, this code check to see if this kind of
//oscillation has occured and sets n=50.
//Also accounts for n=0 and n=1023.
//*****************************************************************
int Correct_Osciallations(){

    int correctedVal;

    //These series of if else if loops check to see if a number is
    //if a number is oscillating between for example 4,5 and 6. These
    //if else if loops fix that oscillation and set the ADC value to 5.
    if(step1 == 0) correctedVal = 0;
    else if(step1 == 1023) correctedVal = 1023;
    else if(step2 == 0) correctedVal = step1;
    else if((step1 == 1) && (step2 == 1)) correctedVal = step1;
    //else if(step1 <= 4) correctedVal = 4;
    //else if(step1 <= 8) correctedVal = 8;

    else if(((step2<=step1+15) && (step2>=step1-15))){

        correctedVal = step2;
        step1 = step2;
    }

    else if(((step2<=step1-15) && (step2>=step1+15))){

        correctedVal = step1;
        step1 = step2;
    }

    step2 = step1;
    return correctedVal;

}



//*****************************************************************
//Name    : DisplayLED(char n_SingleDigit)
//Input   : char
//Returns : void
//
//Used to display a number.
//char 0-9 is given as an input and the corresponding
//light turns on using a switch statement
//Case 10 is used to turn off LED and turn off All the LED digits
```

```
//*****************************************************************
void DisplayLED(int n_SingleDigit){

    switch(n_SingleDigit){

        //P2OUT = 0xhgfedcba
        //0x00111111
        case 0:
            P2OUT = 0xC0;
            break;

        //0x00000110
        case 1:
            P2OUT = 0xF9;
            break;

        //0x01011011
        case 2:
            P2OUT = 0xA4;
            break;

        //0x01001111
        case 3:
            P2OUT = 0xB0;
            break;

        //0x01100110
        case 4:
            P2OUT = 0x99;
            break;

        //0x01101101
        case 5:
            P2OUT = 0x92;
            break;

        //0x01111101
        case 6:
            P2OUT = 0x82;
            break;

        //0x00000111
        case 7:
            P2OUT = 0xF8;
            break;

        //0x01111111
        case 8:
            P2OUT = 0x80;
            break;

        //0x01110111
        case 9:
            P2OUT = 0x90;
            break;
        case 10:
            P1OUT = 0x00;
            P2OUT = 0xFF;
            break;


    }

}



//*****************************************************************
//Name    : Control_Dx(int n)
//Input   : int
//Returns : void
//
//This function is responsible for displaying the numbers.
//It gets an input of the reading of the potentiometer.
//Depending on the number of digits required, the digits are turned on.
//Then the Display LED function is used to turn on the numbers, one
//number at a time.
//If the ADC number is 357. 7 is displayed on D1, 5 on D2 and 3 on D3.
//DisplayLED(10) is used to turn off everything
//*****************************************************************
void Control_Dx(int n){

    int SingleDigit;

    //if n = 271, 2 is displayed on D1
    //then D1 is turned off and D2 is turned on
    //and 7 is displayed. Then D3 is turned on
    //and 1 is displayed

    if(n <= 9){

        //D4 - 0x00100000

        P1OUT = 0x01;
        SingleDigit = n;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

    }

    else if((n>=10) && (n<=99)){

        //D4 - 0x00100000
        //D3 - 0x00010000

        DisplayLED(10);

        P1OUT = 0x01;
        SingleDigit = n % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);

        P1OUT = 0x20;
        SingleDigit = n / 10 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);
    }

    else if((n>=100) && (n<=999)){

        //D4 - 0x00100000
        //D3 - 0x00010000
        //D2 - 0x00000100

        P1OUT = 0x01;
        SingleDigit = n % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);

        P1OUT = 0x20;
        SingleDigit = n / 10 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);

        P1OUT = 0x40;
        SingleDigit = n / 100 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);
    }

    else if(n >= 1000){

        //D4 - 0x00100000
        //D3 - 0x00010000
        //D2 - 0x00000100
        //D1 - 0x00000010

        P1OUT = 0x01;
        SingleDigit = n % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);
        P1OUT = 0x20;
        SingleDigit = n / 10 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);

        P1OUT = 0x40;
        SingleDigit = n / 100 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);

        P1OUT = 0x80;
        SingleDigit = n / 1000 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(6000);

        DisplayLED(10);
    }


}



//*******************************************************************
//Name    : Setup_Chip_one()
//Input   : void
//Returns : void
//
//Sets up all the ports and pins used by Chip 1
//*******************************************************************
void Setup_Chip_one(){

    //Set P1.4 input (potentiometer)
    P1DIR = 0x00;

    //Function to setup ADC
    ADC_Setup();

    //Function to setup UART
    UART_Setup();


}



//*******************************************************************
//Name    : void Setup_Chip_two()
//Input   : void
//Returns : void
//
```

```
//Sets up all the ports and pins used by Chip 2
//*****************************************************************
void Setup_Chip_two(){

    //Set XIN and XOUT to GPIO
    P2SEL = 0;
    P2SEL2 = 0;

    //P1.0,P1.5,P1.6,P1.7 -> D1,D2,D3,D4
    P1DIR = 0xE1;

    //Set P2.0 - P2.5 to output
    //P2.0 - P2.7 -> abcdefgh
    P2DIR = 0xFF;

    //Function to setup UART
    UART_Setup();

}



//*****************************************************************
//Name    : Program_Chip_one()
//Input   : void
//Returns : void
//
//Program to be uploaded to Chip 2 This program is placed into an
//infinite while loop so it keeps occurring forever
//*****************************************************************
void Program_Chip_one(){

    int correctedVal;

    //Gets ready for ADC Conversion
    ADC10CTL0 |= ENC + ADC10SC;

    //Stores the digital value of the potentiometer
    //ADC10MEM is a number between 0 to ((2^10) - 1)
    step1 = ADC10MEM;

    //Function used to fix oscillations
    correctedVal = Correct_Osciallations();
    correctedVal = correctedVal/4;

    //Send ADC Value to Chip 2
    uartTransmitData(correctedVal);
}



//*****************************************************************
//Name    : Program_Chip_two()
//Input   : void
//Returns : void
```

```
//
//Program to be uploaded to Chip 2 This program is placed into an
//infinite while loop so it keeps occurring forever
//*****************************************************************
void Program_Chip_two(){

    //Receives the ADC Value from Chip 1
    //int correctVal = uartReceiveData()*4;
    //if(correctVal>=1020) correctVal=1023;

    step1 = uartReceiveData()*4;
    int correctVal = Correct_Osciallations();
    if(correctVal>=1010) correctVal=1023;

    //Display ADC Value from Chip 1 on the LED
    Control_Dx(correctVal);

}
```