```
//*******************************************************************
//ECGR 5101
//Lab 09
//Naseeruddin Lodge
//Srikar Thakkallapally
//Group 12
//
//
//The objective of this lab is to connect two MSP430G2553 together.
//An ultrasonic sensor is connected to chip 1. It is used to find the
//distance in cm and display it on Quad-Digit 7 segment LED connected
//to chip 2 using UART communication. An accelerometer I also is
//connected, and it displays the x and y axis is a 0-90-degree
//range on the Quad-Digit 7 segment LED using UART connections.
//Two buttons are present. One to switch between ultrasonic sensor
//and accelerometer and the other one is used to select a distance.
//A buzzer is used to buzz when the distance selected by the user is
//displayed or if the accelerometer is placed on a flat surface or
//when the x and y-axis read 0 on the Quad-Digit 7 segment LED.
//All these components were soldered on a perforated board to build
//an embedded system which runs on a battery.
//*******************************************************************

//#include <msp430.h>
#include <msp430g2553.h>



//Global Variables
int button_0 = 0, button_1 = 0;
int milisec, distance, sensorVal, temp=0;
int US_Sensor_Reading[11];
unsigned int count=0;
int presetVal, ADCValx, ADCValy;
unsigned int adc[2];
int step1=0, step2=0;
int step3=0, step4=0;
int flag=2;


//Prototyped Functions
void UART_Setup();
void TimerA_Setup();
void PWM_Setup();
void ADC_Setup();
void US_Sensor_Setup();
void Buzzer_Setup();
void Button0_Setup();
void Button1_Setup();
void ADC_Setup_P10();
void ADC_Setup_P17();
int  uartReceiveData();
void uartTransmitData(int ADCval);
void BuzzerLevel(int PresetValue);
void DisplayLED(int n_SingleDigit);
void Control_Dx(int n);
int Correct_Osciallations_x();
int Correct_Osciallations_y();
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    : 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;
    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    : TimerA_Setup()
//Input   : Void
//Returns : Void
//
//Function to Setup Timer
//*****************************************************************
void TimerA_Setup(){

    //Resolution(Delay per TAR Count) in Seconds =
    //(DIV / Input Clock in HZ)
    //1/1MHZ = 1 * 10^-6 sec


    //CCR0 interrupt enabled
    CCTL0 = CCIE;

    //1ms at 1 MHZ
    CCR0 = 1000;

    //SMCLK, upmode
    TA0CTL = TASSEL_2 + MC_1;
    TA0CCTL1 = OUTMOD_7;


}
```

```
//*****************************************************************
//Name    : PWM_Setup()
//Input   : Void
//Returns : Void
//
//Function to Setup PWM
//*****************************************************************
void PWM_Setup(){

    //PWM period
    TA1CCR0 = 1000;

    //CCR1 PWM Duty Cycle
    TA1CCR1 = 1;

    //CCR1 selection reset-set
    TA1CCTL1 = OUTMOD_7;

    //SMCLK submain clock,upmode
    TA1CTL = TASSEL_2 | MC_1;


}



//*****************************************************************
//Name    : ADC_Setup_P10()
//Input   : void
//Returns : void
//
//Function to Setup ADC
//*****************************************************************
void ADC_Setup_P10(){

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

    ADC10CTL1 = INCH_0;
    ADC10AE0 |= 0x01;

}



//*****************************************************************
//Name    : ADC_Setup_P17()
//Input   : void
//Returns : void
//
//Function to Setup ADC
```

```
//*******************************************************************
void ADC_Setup_P17(){

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

    ADC10CTL1 = INCH_1;
    ADC10AE0 |= 0x02;

}



//*******************************************************************
//Name    : Port_1(void)
//Input   : Void
//Returns : Void
//
//ISR for Echo Pin
//*******************************************************************
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){

    //Check interrupt Status
    if(P1IFG & 0x40){

        //Check rising edge
        if(!(P1IES & 0x40)){

            //Clear timer A
            TACTL|=TACLR;
            milisec = 0;

            //Set to Falling edge
            P1IES |= 0x40;
        }
        else{

            //ECHO length
            sensorVal = milisec*1000 + TAR;
        }

        //Clear flag
        P1IFG &= ~0x40;
    }




    if(P1IFG && BIT3){

        //P1OUT ^= 0x00;
        button_0 = (button_0 + 1) % 2;
        __delay_cycles(100000);
    }

    //P1.3 IFG cleared
    P1IFG &= ~(BIT3);
}



//*****************************************************************
//Name    : Timer_A(void)
//Input   : Void
//Returns : Void
//
//ISR for Timer
//*****************************************************************
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A(void){

    milisec++;
}



//*************************************************************
//Name    : __interrupt void Port_2(void)
//Input   : void
//Returns : void
//
//Port 1 interrupt service routine
//*************************************************************
#pragma vector=PORT2_VECTOR
__interrupt void PORT2_ISR(void){


    if(P2IFG && BIT1){

        //P1OUT ^= 0x00;
        button_0 = 0;
        button_1 = (button_1 + 1) % 2;
    }

    __delay_cycles(100000);

    //P1.3 IFG cleared
    P2IFG &= ~(BIT1);
}



//*****************************************************************
//Name    : Button0_Setup()
//Input   : void
//Returns : void
//
```

```
//Sets up all the ports and pins used by Push Button 0(For
//UltraSonic Sensor)
//*****************************************************************
void Button0_Setup(){

    //P1.3 set as Output
    P1DIR &= ~BIT3;

    //Enable Pull Up resistor for SW2
    P1REN |= BIT3;

    //Pull Up mode for Button 0
    P1OUT |= BIT3;

    //P1.3 interrupt enabled
    P1IE |= BIT3;

    //P1.3 Hi/lo edge
    P1IES |= BIT3;

    //P1.3 IFG cleared
    P1IFG &= ~(BIT3);

}



//*****************************************************************
//Name    : Button1_Setup()
//Input   : void
//Returns : void
//
//Sets up all the ports and pins used by Push Button 0(For
//UltraSonic Sensor)
//*****************************************************************
void Button1_Setup(){

    //P1.3 set as Output
    P2DIR &= ~BIT1;

    //Enable Pull Up resistor for SW2
    P2REN |= BIT1;

    //Pull Up mode for Button 0
    P2OUT |= BIT1;

    //P1.3 interrupt enabled
    P2IE |= BIT1;

    //P1.3 Hi/lo edge
    P2IES |= BIT1;

    //P1.3 IFG cleared
    P2IFG &= ~(BIT1);
```

```
}


//*****************************************************************
//Name    : US_Sensor_Setup()
//Input   : void
//Returns : void
//
//Function to Setup all pins related to UltraSonic Sensor
//*****************************************************************
void US_Sensor_Setup(){

    //Disable interupt
    P1IE &= ~0x01;

    //Trigger to P1.5
    P1DIR |= 0x20;

    //Generate pulse from Trigger
    P1OUT |= 0x20;

    //Generate pulse from Trigger for 10us
    __delay_cycles(10);

    //Stop pulse from Trigger
    P1OUT &= ~0x20;

    //Echo to P1.6
    P1DIR &= ~0x40;

    //Clear Flag
    P1IFG = 0x00;

    //Enable interrupt for ECHO pin
    P1IE |= 0x40;

    //Set ECHO PIN to rising edge
    P1IES &= ~0x40;
}


//*****************************************************************
//Name    : Buzzer_Setup()
//Input   : void
//Returns : void
//
//Function to Setup all pins related to UltraSonic Sensor
//*****************************************************************
void Buzzer_Setup(){

    //Set Direction of buzzer as output
    P2DIR &= BIT2;
    //Set select Pin for P2.1
    P2SEL &= BIT2;
}



//*****************************************************************
//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;
}



void BuzzerLevel(int PresetValue){


    //Five levels of sound for UltraSonic Sensor
    if(PresetValue == 6){

        P2DIR |= BIT2;
        P2SEL |= BIT2;

        //PWM period
        TA1CCR0 = 1000;

        //CCR1 selection reset-set
        TA1CCTL1 = OUTMOD_7;
        //SMCLK submain clock,upmode
        TA1CTL = TASSEL_2|MC_1;

        //CCR1 PWM Duty Cycle as 900
        TA1CCR1 = 900;
    }
    else if(PresetValue == 30){

        P2DIR |= BIT2;
        P2SEL |= BIT2;

        TA1CCR0 = 3000;
        TA1CCTL1 = OUTMOD_7;
        TA1CTL = TASSEL_2|MC_1;
        TA1CCR1 = 1000;
    }
    else if(PresetValue == 60){

        P2DIR |= BIT2;
        P2SEL |= BIT2;

        TA1CCR0 = 10000;
        TA1CCTL1 = OUTMOD_7;
        TA1CTL = TASSEL_2|MC_1;
        TA1CCR1 = 5000;
    }
    else if(PresetValue == 120){

        P2DIR |= BIT2;
        P2SEL |= BIT2;

        TA1CCR0 = 10000;
        TA1CCTL1 = OUTMOD_7;
        TA1CTL = TASSEL_2|MC_1;
        TA1CCR1 = 1000;
    }
    else if(PresetValue == 210){

        P2DIR |= BIT2;
        P2SEL |= BIT2;

        TA1CCR0 = 10000;
        TA1CCTL1 = OUTMOD_7;
        TA1CTL = TASSEL_2|MC_1;
        TA1CCR1 = 1000;
    }

    else if(PresetValue == 77){

        //Set Direction of buzzer as output
        P2DIR &= ~BIT2;

        //Set select Pin for P2.1
        P2SEL &= ~BIT2;
        //TA1CCR0 = 10000;
        //TA1CCTL1 = OUTMOD_7;
        //TA1CTL = TASSEL_2|MC_1;
        //TA1CCR1 = 1;
    }
}
//*****************************************************************
//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(10000);

    }

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

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

        DisplayLED(10);

        P1OUT = 0x01;
        SingleDigit = n % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);

        P1OUT = 0x20;
        SingleDigit = n / 10 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);
    }

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

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

        P1OUT = 0x01;
        SingleDigit = n % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);

        P1OUT = 0x20;
        SingleDigit = n / 10 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);

        P1OUT = 0x40;
        SingleDigit = n / 100 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);
    }
    else if(n >= 1000){

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

        P1OUT = 0x01;
        SingleDigit = n % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);

        P1OUT = 0x20;
        SingleDigit = n / 10 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);

        P1OUT = 0x40;
        SingleDigit = n / 100 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);

        P1OUT = 0x80;
        SingleDigit = n / 1000 % 10;
        DisplayLED(SingleDigit);
        __delay_cycles(10000);

        DisplayLED(10);
    }


}



//*******************************************************************
//Name    : correct_osciallations_x()
//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_x(){
    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(((step2<=step1+40) && (step2>=step1-40))){

        correctedVal = step2;
        step1 = step2;
    }

    else if(((step2<=step1-40) && (step2>=step1+40))){

        correctedVal = step1;
        step1 = step2;
    }

    step2 = step1;
    return correctedVal;

}



//*******************************************************************
//Name    : correct_osciallations_y()
//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_y(){

    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(step3 == 0) correctedVal = 0;
    else if(step3 == 1023) correctedVal = 1023;
    else if(step4 == 0) correctedVal = step3;
    else if((step3 == 1) && (step4 == 1)) correctedVal = step3;

    else if(((step4<=step3+20) && (step4>=step3-20))){

        correctedVal = step4;
        step3 = step4;
    }

    else if(((step4<=step3-20) && (step4>=step3+20))){

        correctedVal = step3;
        step3 = step4;
    }

    step4 = step3;
    return correctedVal;

}



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

    //Function to Setup UART
    UART_Setup();

    //P1IFG  = 0x00;

    //Buzzer Setup
    Buzzer_Setup();

    //Timer Setup
    TimerA_Setup();

    //Accelerometer Setup
    P1DIR |= 0x81;

    //Set up Button 0 for UltraSonic Sensor
    Button0_Setup();

    //Set up Button 1 for Accelerometer
    Button1_Setup();

    //ADC Setup
    //ADC_Setup();
    P1DIR &= ~BIT0;
    P1DIR &= ~BIT7;
    ADC10CTL0 = SREF_0 + ADC10SHT_2 + ADC10ON;

    //Enable Interrupts
    _enable_interrupts();

}
```

```c
//*****************************************************************
//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(){

    volatile unsigned int i;
    unsigned int j,k;
    //int ADCVal;

    //Sets up UltraSonic Sensor to read distance
    US_Sensor_Setup();

    //Delay for 30ms
    //If no object it found, ECHO times out
    __delay_cycles(30000);

    //Converting ECHO value to CM
    distance = sensorVal/58;


    //Get 10 Ultrasonic Readings in an array
```

```
if(count <= 11){


    if(distance <= 3)
        US_Sensor_Reading[count] = 4;

    else if(distance >= 4 && distance <= 400)

        if(distance == 111)
            US_Sensor_Reading[count] = 112;
        else if(distance == 99)
            US_Sensor_Reading[count] = 100;
        else if(distance == 177)
            US_Sensor_Reading[count] = 176;
        else
            US_Sensor_Reading[count] = distance;

    else if(distance > 400)
        US_Sensor_Reading[count] = 400;

    count++;
}

//Sort the array with UltraSonic Sensor readings in
//ascending order and their median will be the
//distance in CM
else if(count == 12){

    for(k=0; k<11; k++){

        for (j = 0; j+1<11-k; j++){


            if (US_Sensor_Reading[j] > US_Sensor_Reading[j + 1]){

                temp = US_Sensor_Reading[j];
                US_Sensor_Reading[j] = US_Sensor_Reading[j + 1];
                US_Sensor_Reading[j + 1] = temp;
            }
        }
    }

    temp = US_Sensor_Reading[6];

    count = 0;
}



if(button_0 == 0 && button_1 == 0){

    uartTransmitData(177);
    for(i=1000; i>0; i--);

    BuzzerLevel(77);
```

```
if(button_1 == 0){

    for(i=1000; i>0; i--){

        if(button_1 == 0){
            presetVal = 6;
            uartTransmitData(presetVal/2);
        }
        else
            return;
    }

    for(i=1000; i>0; i--){

        if(button_1 == 0){
            presetVal = 30;
            uartTransmitData(presetVal/2);
        }
        else
            return;
    }

    for(i=1000; i>0; i--){

        if(button_1 == 0){
            presetVal = 60;
            uartTransmitData(presetVal/2);
        }
        else
            return;
    }

    for(i=1000; i>0; i--){

        if(button_1 == 0){
            presetVal = 120;
            uartTransmitData(presetVal/2);
        }
        else
            return;
    }

    for(i=1000; i>0; i--){

        if(button_1 == 0){
            presetVal = 210;
            uartTransmitData(presetVal/2);
        }
        else
            return;
    }
}
```

```c
}

else if(button_0 == 0 && button_1 == 1){

    uartTransmitData(177);
    for(i=1000; i>0; i--);

    uartTransmitData(temp/2);
    for(i=1000; i>0; i--);

    if(temp==presetVal)BuzzerLevel(presetVal);
    else BuzzerLevel(77);
}


else if(button_0 == 1){

    button_1 = 0;

    uartTransmitData(111);
    for(i=1000; i>0; i--)

    //( old value - x) * (180) / (y - x)
    // -90

    //x-axis
    //380 - 580
    ADC_Setup_P10();
    ADC10CTL0 |= ENC + ADC10SC;
    step1 = ADC10MEM;
    ADCValx = Correct_Osciallations_x();

    ADCValx = (((float)(ADCValx-380)*180)/(float)200)-90;
    ADCValx = abs(ADCValx);
    if(ADCValx > 90) ADCValx = 90;
    if(ADCValx >= 0 && ADCValx <= 11){
        ADCValx = 0;
        BuzzerLevel(6);
    }
    else if(ADCValx >= 12) BuzzerLevel(77);

    uartTransmitData(ADCValx);
    for(i=1000; i>0; i--);


    uartTransmitData(99);
    for(i=1000; i>0; i--);

    //y-axis
    //470 - 520
    ADC_Setup_P17();
    ADC10CTL0 |= ENC + ADC10SC;
    step3 = ADC10MEM;
    ADCValy = Correct_Osciallations_y();
```

```
        ADCValy = (((float)(ADCValy-470)*180)/(float)50)-90;
        ADCValy = abs(ADCValy);
        if(ADCValy > 90) ADCValy = 90;
        if(ADCValy >= 0 && ADCValy <= 5) ADCValy = 0;


        uartTransmitData(ADCValy);
        for(i=1000; i>0; i--);

    }


}



//*****************************************************************
//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(){

    int recieve = uartReceiveData();
    int SingleDigit;

    if(recieve == 111)      flag = 0;
    else if(recieve == 99) flag = 1;
    else if(recieve == 177) flag = 2;

    else{

        if(flag == 0){

            if(recieve <= 9){

                //D4 - 0x00100000

                DisplayLED(10);

                P1OUT = 0x40;
                SingleDigit = recieve;
                DisplayLED(SingleDigit);
                __delay_cycles(10000);

                DisplayLED(10);

            }

            else if((recieve>=10) && (recieve<=99)){
                //D4 - 0x00100000
                //D3 - 0x00010000

                DisplayLED(10);

                P1OUT = 0x40;
                SingleDigit = recieve % 10;
                DisplayLED(SingleDigit);
                __delay_cycles(10000);

                DisplayLED(10);

                P1OUT = 0x80;
                SingleDigit = recieve / 10 % 10;
                DisplayLED(SingleDigit);
                __delay_cycles(10000);

                DisplayLED(10);
            }
        }

        else if(flag == 1){

            if(recieve <= 9){

                //D4 - 0x00100000

                DisplayLED(10);

                P1OUT = 0x01;
                SingleDigit = recieve;
                DisplayLED(SingleDigit);
                __delay_cycles(10000);

                DisplayLED(10);

            }

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

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

                DisplayLED(10);

                P1OUT = 0x01;
                SingleDigit = recieve % 10;
                DisplayLED(SingleDigit);
                __delay_cycles(10000);

                DisplayLED(10);

                P1OUT = 0x20;
                SingleDigit = recieve / 10 % 10;
                DisplayLED(SingleDigit);
                __delay_cycles(10000);

                DisplayLED(10);
            }
        }

        else if(flag == 2)
            Control_Dx(recieve*2);
    }

}
```