Annex 1

Led

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <xc.h>

void main(void) {

TRISB = 0xFF;

TRISD = 0x00;

PORTD = 0x00;

while(1) {

if(PORTBbits.RB0 == 1) {

//LATD = 255;

LATD = 0xFF;

} else {

LATD = 0x00;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PIC18F4550 Configuration Bit Settings.txt

#include <xc.h>

#include <pic18f4550.h>

// PIC18F4550 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1L

#pragma config PLLDIV = 1 // PLL Prescaler Selection bits (No prescale (4 MHz oscillator input drives PLL directly))

#pragma config CPUDIV = OSC1\_PLL2// System Clock Postscaler Selection bits ([Primary Oscillator Src: /1][96 MHz PLL Src: /2])

#pragma config USBDIV = 1 // USB Clock Selection bit (used in Full-Speed USB mode only; UCFG:FSEN = 1) (USB clock source comes directly from the primary oscillator block with no postscale)

// CONFIG1H

#pragma config FOSC = INTOSC\_EC // Oscillator Selection bits (Internal oscillator, CLKO function on RA6, EC used by USB (INTCKO))

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config PWRT = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOR = OFF // Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)

#pragma config BORV = 3 // Brown-out Reset Voltage bits (Minimum setting 2.05V)

#pragma config VREGEN = OFF // USB Voltage Regulator Enable bit (USB voltage regulator disabled)

// CONFIG2H

#pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))

#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H

#pragma config CCP2MX = OFF // CCP2 MUX bit (CCP2 input/output is multiplexed with RB3)

#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)

#pragma config LPT1OSC = OFF // Low-Power Timer 1 Oscillator Enable bit (Timer1 configured for higher power operation)

#pragma config MCLRE = ON // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)

// CONFIG4L

#pragma config STVREN = OFF // Stack Full/Underflow Reset Enable bit (Stack full/underflow will not cause Reset)

#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)

#pragma config ICPRT = OFF // Dedicated In-Circuit Debug/Programming Port (ICPORT) Enable bit (ICPORT disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L

#pragma config CP0 = OFF // Code Protection bit (Block 0 (000800-001FFFh) is not code-protected)

#pragma config CP1 = OFF // Code Protection bit (Block 1 (002000-003FFFh) is not code-protected)

#pragma config CP2 = OFF // Code Protection bit (Block 2 (004000-005FFFh) is not code-protected)

#pragma config CP3 = OFF // Code Protection bit (Block 3 (006000-007FFFh) is not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) is not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM is not code-protected)

// CONFIG6L

#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000800-001FFFh) is not write-protected)

#pragma config WRT1 = OFF // Write Protection bit (Block 1 (002000-003FFFh) is not write-protected)

#pragma config WRT2 = OFF // Write Protection bit (Block 2 (004000-005FFFh) is not write-protected)

#pragma config WRT3 = OFF // Write Protection bit (Block 3 (006000-007FFFh) is not write-protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) are not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot block (000000-0007FFh) is not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM is not write-protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000800-001FFFh) is not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (002000-003FFFh) is not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (004000-005FFFh) is not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (006000-007FFFh) is not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) is not protected from table reads executed in other blocks)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

void main(void) {

PORTD = 0;

TRISDbits.RD7 = 0;

while(1){

LATDbits.LATD7 =1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sample code with 1sec delay loop.txt

#include <xc.h>

#define \_XTAL\_FREQ 8000000

void delay1s (void);

void main(void) {

TRISD=0;

PORTD=255;

while (1){

delay1s();

PORTD=255;

delay1s();

PORTD=0;

}

return;

}

void delay1s (){

for (int i=0; i<25; i++){

\_\_delay\_ms(20);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

configBits\_PIC18F87J11.h

C Scripts:

// PIC18F87J11 Configuration Bit Settings

// 'C' source line config statements

#include <xc.h>

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

// CONFIG1L

#pragma config WDTEN = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on SWDTEN bit))

#pragma config STVREN = OFF // Stack Overflow/Underflow Reset Enable bit (Reset on stack overflow/underflow disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG1H

#pragma config CP0 = OFF // Code Protection bit (Program memory is not code-protected)

// CONFIG2L

#pragma config FOSC = INTOSC // Oscillator Selection bits (Internal oscillator, port function on RA6 and RA7 )

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit (Two-Speed Start-up disabled)

// CONFIG2H

#pragma config WDTPS = 32768 // Watchdog Timer Postscaler Select bits (1:32768)

// CONFIG3L

#pragma config EASHFT = OFF // External Address Bus Shift Enable bit (Address shifting disabled, address on external bus reflects the PC value)

#pragma config MODE = MM // External Memory Bus Configuration bits (Microcontroller mode - External bus disabled)

#pragma config BW = 16 // Data Bus Width Select bit (16-bit external bus mode)

#pragma config WAIT = OFF // External Bus Wait Enable bit (Wait states on the external bus are disabled)

// CONFIG3H

#pragma config CCP2MX = DEFAULT // ECCP2 MUX bit (ECCP2/P2A is multiplexed with RC1)

#pragma config ECCPMX = DEFAULT // ECCPx MUX bit (ECCP1 outputs (P1B/P1C) are multiplexed with RE6 and RE5; ECCP3 outputs (P3B/P3C) are multiplexed with RE4 and RE3)

#pragma config PMPMX = DEFAULT // PMP Pin Multiplex bit (PMP port pins connected to EMB (PORTD and PORTE))

#pragma config MSSPMSK = MSK7 // MSSP Address Masking Mode Select bit (7-Bit Address Masking mode enable)

adc.c

include <xc.h>

#define \_XTAL\_FREQ 8000000

//LCD connections

#define RS PORTDbits.RD2

#define EN PORTDbits.RD3

#define D4 PORTDbits.RD4

#define D5 PORTDbits.RD5

#define D6 PORTDbits.RD6

#define D7 PORTDbits.RD7

#include "lcd.h"

#include "config.h"

#include <pic18f87j11.h>

void showVoltageOnDisplay (float adc2);

void configureADC (void);

void main()

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*set up LCD \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

TRISD=0x00;

PORTD=0;

PORTB=0x00;

TRISB=0x00;

Lcd\_Init();

Lcd\_Clear();

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String(" Voltage");

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of set up LCD \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

configureADC();

/\*

\* Next part below starts the AD conversion process on channel AN0,

\* waits for conversion to finish

\* calculates the actual analog voltage,

\* displays it on a LCD for verification

\*/

while(1) {

\_\_delay\_ms(20);

float adc=0.0;

ADCON0bits.GO\_DONE = 1; // Start Conversion

while (ADCON0bits.GO\_DONE == 1){

// wait

}

adc=ADRESH\*256+ADRESL; //ADRESL = lower bits //ADRESH = higher bits

adc = adc\*0.00488;

if (adc > 3){

LATB = 0xFF;

} else {

LATB = 0x00;

}

showVoltageOnDisplay(adc);

}

}

void configureADC (void){

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*configure port A for ADC

TRISA=0xFF; //port A as input

ADCON0bits.VCFG1 =0; // use internal voltage VSS and VDD as reference

ADCON0bits.VCFG0 =0;

ADCON0bits.CHS0=0; //select channel AN0

ADCON0bits.CHS1=0;

ADCON0bits.CHS2=0;

ADCON0bits.CHS3=0;

ADCON1bits.ADFM=1; //right justification of result

ADCON1bits.ACQT0=0; //set AD acquisition time of 16TAD

ADCON1bits.ACQT1=1;

ADCON1bits.ACQT2=1;

ADCON1bits.ADCS0=0; //select AD clock, FOSC/4 // this is needed for PIC18F87J11

ADCON1bits.ADCS1=0;

ADCON1bits.ADCS2=1;

ADCON0bits.ADON =1; //turn on ADC

}

/\*

\* This code belows only converts analog value to characters so that it

\* can be displayed on a LCD.

\* number 0 and character 0 are not the same thing. Character 0 is obtained

\* by adding 48 to 0. So character 1 will be created by adding 48 to number 1,

\*/

void showVoltageOnDisplay (float adc2) {

int te =0;

char lcdc[] = "00.00";

te = adc2/10;

if (te>0) {

lcdc[0] = te + 48; // convert it to a character

adc2 = adc2 - (te)\*10;

} else {

lcdc[0] = 48;

}

te = adc2;

if (te>0) {

lcdc[1] = te + 48;

adc2 = adc2 - (te);

} else {

lcdc[1] = 48;

}

te = adc2\*10;

if (te>0) {

lcdc[3] = te + 48;

adc2 = adc2 - (te);

} else {

lcdc[3] = 48;

}

te = adc2\*100;

if (te>0) {

lcdc[4] = te + 48;

//adc2 = adc2 - (te)/10;

} else {

lcdc[4] = 48;

}

Lcd\_Set\_Cursor(2,5);

Lcd\_Write\_String(lcdc);

Lcd\_Write\_Char(' '); //ASCII code for degree sign

Lcd\_Write\_Char('V');

return;

}

lcd.h

//LCD Functions Developed by electroSome

void Lcd\_Port(char a)

{

if(a & 1)

D4 = 1;

else

D4 = 0;

if(a & 2)

D5 = 1;

else

D5 = 0;

if(a & 4)

D6 = 1;

else

D6 = 0;

if(a & 8)

D7 = 1;

else

D7 = 0;

}

void Lcd\_Cmd(char a)

{

RS = 0; // => RS = 0

Lcd\_Port(a);

EN = 1; // => E = 1

\_\_delay\_ms(4);

EN = 0; // => E = 0

}

Lcd\_Clear()

{

Lcd\_Cmd(0);

Lcd\_Cmd(1);

}

void Lcd\_Set\_Cursor(char a, char b)

{

char temp,z,y;

if(a == 1)

{

temp = 0x80 + b - 1;

z = temp>>4;

y = temp & 0x0F;

Lcd\_Cmd(z);

Lcd\_Cmd(y);

}

else if(a == 2)

{

temp = 0xC0 + b - 1;

z = temp>>4;

y = temp & 0x0F;

Lcd\_Cmd(z);

Lcd\_Cmd(y);

}

}

void Lcd\_Init()

{

Lcd\_Port(0x00);

\_\_delay\_ms(20);

Lcd\_Cmd(0x03);

\_\_delay\_ms(5);

Lcd\_Cmd(0x03);

\_\_delay\_ms(11);

Lcd\_Cmd(0x03);

/////////////////////////////////////////////////////

Lcd\_Cmd(0x02);

Lcd\_Cmd(0x02);

Lcd\_Cmd(0x08);

Lcd\_Cmd(0x00);

Lcd\_Cmd(0x0C);

Lcd\_Cmd(0x00);

Lcd\_Cmd(0x06);

}

void Lcd\_Write\_Char(char a)

{

char temp,y;

temp = a&0x0F;

y = a&0xF0;

RS = 1; // => RS = 1

Lcd\_Port(y>>4); //Data transfer

EN = 1;

\_\_delay\_us(40);

EN = 0;

Lcd\_Port(temp);

EN = 1;

\_\_delay\_us(40);

EN = 0;

}

void Lcd\_Write\_String(char \*a)

{

int i;

for(i=0;a[i]!='\0';i++)

Lcd\_Write\_Char(a[i]);

}