**SQUARE WAVE DAC**

#include<REG51.h>

void delay\_time(unsigned int time)

{

unsigned int i,j;

for(i=time;i>0;i--)

{

for(j=0;j<2000;j++);

}

}

void send\_dac(unsigned int dat)

{

P1 = dat;

}

void main(void)

{

unsigned int delay=0xFF;

unsigned char state=0xFF;

while(1)

{

send\_dac(state);

delay\_time(delay);

send\_dac(~state);

delay\_time(delay);

}

}

**TRIANGULAR WAVE DAC**

#include <REG51.h>

void delay\_ramp(unsigned int time)

{

unsigned int i,j;

for(i=time;i>0;i--)

{

for(j=0;j<10;j++);

}

}

void send\_dac(unsigned int dat)

{

P1 = dat;

}

void main(void)

{

unsigned int a,state=0xff;

while(1)

{

// rising ramp edge

for(a=0x0;a<0xFF;a++)

{

send\_dac(a);

delay\_ramp(1);

}

// falling ramp edge

for(a=0xFF;a>0;a--)

{

send\_dac(a);

delay\_ramp(1);

}

}

}

**TRAPEZOIDAL DAC**

#include <REG51.h>

void delay\_time(unsigned int time)

{

unsigned int i,j;

for(i=time;i>0;i--)

{

for(j=0;j<20;j++);

}

}

void delay\_ramp(unsigned int time)

{

unsigned int i;

for(i=time;i>=0;i--);

}

void send\_dac(unsigned int dat)

{

P1 = dat;

}

void main(void)

{

unsigned int a,state=0xFF;

while(1)

{

// rising ramp edge

for(a=0;a<0xFF;a++)

{

send\_dac(a);

// delay\_ramp(0);

}

// high state

send\_dac(0xFF);

delay\_time(100);

// falling ramp edge

for(a=0xFF;a>0;a--)

{

send\_dac(a);

// delay\_ramp(0);

}

// low state

send\_dac(0x00);

delay\_time(100);

}

}

**LED**

#include <reg51.h>

void delay(unsigned int time)

{

unsigned int i,j;

for(i=0;i<time;i++)

for(j=0;j<0xFF;j++);

}

int main()

{

P1 = 0x00;

while(1)

{

P1 = ~P1;

delay(100);

}

}

**Seven Segment Interfacing**

/\*

Seven Segment Interfacing

Connect Seven Segment card to CN2

A -> P0.6

B -> P0.5

C -> P0.0

D -> P0.7

E -> P0.2

F -> P0.3

G -> P0.4

Dp -> P0.1

\*/

#include <reg51.h>

#define ONE 0x21

#define TWO 0xF4

#define THREE 0xF1

#define FOUR 0x39

#define FIVE 0xD9

#define SIX 0xDD

#define SEVEN 0x61

#define EIGHT 0xFD

#define NINE 0xF9

#define ZERO 0xED

unsigned char segData[10] = {ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE};

void delay(unsigned int time)

{

unsigned int i, j;

for (i = 0; i < time; i++)

for (j = 0; j < 100; j++);

}

void main()

{

unsigned int i;

P0 = 0x00; // Initialize P0 as output for 7-segment display (assuming all pins are used)

while (1)

{

for (i = 0; i < 10; i++)

{

P0 = ~segData[i]; // Display each digit on the 7-segment display

delay(500); // Delay between displaying each digit (adjust as needed)

**LCD module**

/\* Connect LCD module to CN2

DATA -> Port0

RS -> P3.2

EN -> P3.3

\*/

#include <reg51.h>

#define DATAPORT P0

#define EN 1<<3 // 0000 1000

#define RS 1<<2 // 0000 0100

void delay(void)

{

unsigned int i;

for(i=0;i<0xFF;i++);

}

void LCDcmd(unsigned char cmd)

{

DATAPORT = cmd;

P3 = 0;

delay();

P3 = EN; // P3 = 0000 1000

delay();

P3 = 0;

delay();

}

void LCDdata(unsigned char dat)

{

DATAPORT = dat;

P3 = RS;

delay();

P3 = RS | EN; //P3 = ( 0000 1000 OR 0000 0100) = 0000 1100

delay();

P3 = RS;

delay();

}

void LCD\_string(int row, int pos, char \*ptr)

{

unsigned char temp;

if(row ==1)

temp = (0x80 | pos);

else

temp = (0xC0 | pos);

LCDcmd(temp); // cursor is set to (row, pos)

while(\*ptr)

{

LCDdata(\*ptr);

ptr++;

}

}

int main()

{

LCDcmd(0x38); //8 bit

LCDcmd(0x06); //entry mode

LCDcmd(0x0C); // display on cursor off

LCDcmd(0x80); // cursor on 1st line 1st pos

LCDcmd(0x01); // clear display

LCD\_string(1,1,"MicroEmbedded");

LCD\_string(2,0,"Micro51 S Board");

while(1); //stop here

return 0;

}

**LCD**

#include <p18f4550.h>

#include "lcd.h"

#define LCD\_EN LATCbits.LC1

#define LCD\_RS LATCbits.LC0

#define LCDPORT LATB

void delay(unsigned int time)

{

unsigned int i , j ;

for(i = 0; i < time; i++)

{

for(j=0;j<100;j++);

}

}

void SendInstruction(unsigned char command)

{

//set RS pin low

LCD\_RS = 0; // RS low : Instruction

delay(10);

//Set EN pin HIgh

LCD\_EN = 1; // EN High

delay(10);

//Set Data on pins

LCDPORT = command; // DATA

delay(10);

//set EN pin Low

LCD\_EN = 0; // EN Low

delay(10);

LCD\_RS = 0; // RS Low

}

void SendData(unsigned char lcddata)

{

//set RS pin low

LCD\_RS = 1; // RS HIGH : DATA

delay(10);

//Set EN pin HIgh

LCD\_EN = 1; // EN High

delay(10);

//Set Data on pins

LCDPORT = lcddata; // DATA

delay(10);

//set EN pin Low

LCD\_EN = 0; // EN Low

delay(10);

LCD\_RS = 0; // RS Low

}

void SetLineNumber(unsigned char linenum)

{

if(linenum == 1)

{

SendInstruction( SETDDRAMADDR | 0x00 ); // set ddram address as 0x00

}

else

{

SendInstruction( SETDDRAMADDR | 0x40 ); // set ddram address as 0x40

}

}

void SetAddress(unsigned char memory, unsigned char address )

{

address &= 0x7F;

if(memory == DDRAM)

{

SendInstruction( SETDDRAMADDR | address );

}

else

{

SendInstruction( SETCGRAMADDR | address );

}

}

void BlinkDisplay(void)

{

SendInstruction( DISPONOFFCTRL );

delay(5000);

SendInstruction( DISPONOFFCTRL | DISPLAY\_ON | CURSOR | BLINKCHAR\_ON );

delay(9000);

}

void ShiftDisplay(unsigned char direction, unsigned char clicks)

{

unsigned int i;

if(direction == RIGHTSHIFT)

{

for(i = 0; i < clicks ; i++)

{

SendInstruction( DISPCURSHIFT | SHIFTRIGHT );

}

}

else

{

for(i = 0; i < clicks ; i++)

{

SendInstruction( DISPCURSHIFT );

}

}

}

void ClearDisplay(void)

{

SendInstruction(CLEARDISPLAY);

}

void InitLCD(void)

{

TRISB = 0x00; //set data port as output

TRISCbits.RC0 = 0; //EN pin

TRISCbits.RC1 = 0; // RS pin

//initialise the lcd : function set command : 8 bit interface : 5ms wait

SendInstruction( FUNCTIONSET | FS\_DATALENGTH );

delay(5);

//initialise the lcd : function set command : 8 bit interface : 100us wait

SendInstruction( FUNCTIONSET | FS\_DATALENGTH );

delay(1);

//initialise the lcd : function set command : 8 bit interface : set interface , line numbers and font size

SendInstruction( FUNCTIONSET | FS\_DATALENGTH | FS\_LINES2 ); // set interface 8 bit, no of lines =2 and character font 5x7

//display off

SendInstruction(DISPONOFFCTRL);

//clear display

SendInstruction(CLEARDISPLAY);

//entry mode set : increment / decrement

SendInstruction(ENTRYMODESET | ENTRYMODESET\_INC); //set as increment of address

//display on : set cursor and blinking

SendInstruction(DISPONOFFCTRL | DISPLAY\_ON | CURSOR ); // display on and cursor blinking on

}

//

unsigned char DISP\_STR[]={'2','x','1','6','C','h','a','r','a','c','t','e','r','L','C','D'};

unsigned char \*String =" Micro-PIC Board";

void main(void)

{

int i;

InitLCD();

ClearDisplay();

SetLineNumber(1);

for(i=0;i<16;i++)

{

SendData(DISP\_STR[i]);

}

SetLineNumber(2);

while(\*String)

{

SendData(\*String);

String++;

}

while(1)

BlinkDisplay();

}

**LCD HEADER**

#define CLEARDISPLAY 0x01 // clear display command mask

#define RETURNHOME 0X02 // return home command mask

#define ENTRYMODESET 0X04 // entry mode set command mask

#define DISPONOFFCTRL 0X08 // display on off control mask

#define DISPCURSHIFT 0X10 // display cursor shift command mask

#define FUNCTIONSET 0X20 // function set command mask

#define SETCGRAMADDR 0x40 // set cgram address mask

#define SETDDRAMADDR 0X80 // set ddram address mask

#define ENTRYMODESET\_INC 0X02 // entry mode increment mask

#define ENTRYMODESET\_DEC 0X04 // entry mode decrement mask

#define ENTRYMODESET\_SHIFT 0X01 // entry mode shift mask

#define DISPLAY\_ON 0x04 // display control entire display onoff mask

#define CURSOR 0x02 // display control curson onoff mask

#define BLINKCHAR\_ON 0x01 // display control character blink mask

#define SHIFTCURSOR 0x08 // cursor shift mask

#define SHIFTRIGHT 0x04 // cursor shift direction mask

#define FS\_DATALENGTH 0x10 // function set data length for 8 bit

#define FS\_LINES2 0x08 // function set line number for 2 line display

#define FS\_FONT 0x04 // function set font info mask for 5x10 dots

#define DDRAM 1

#define CGRAM 0

#define RIGHTSHIFT 1

#define LEFTSHIFT 0

void SendInstruction(unsigned char);

void SendData(unsigned char);

void InitLCD(void);

void SetLineNumber(unsigned char);

void SetAddress(unsigned char, unsigned char );

void BlinkDisplay(void);

void ShiftDisplay(unsigned char, unsigned char);

void RollDisplay(void);

**LED**

//#include"18F4550ConfigSettings.h"

#include<P18F4550.h>

void delay(void);

void delay()

{

unsigned int i;

for(i=0;i<30000;i++);

}

void main()

{

unsigned char i,key;

TRISB = 0x00; //LED pins as output

LATB = 0x00;

// PORTE = 0x80;

// SPPCON = 0x00;

// CCP1CON = 0x00;

// PORTE = 0x80;

TRISDbits.TRISD0 = 1; //set RD0 as input

TRISDbits.TRISD1 = 1; //set RD1 as input

TRISDbits.TRISD2 = 0; //set buzzer pin RD2 as output

TRISAbits.TRISA4 = 0; //set relay pin RA4 as output

while(1)

{

LATDbits.LD0 = 1;

LATDbits.LD1 = 1;

if(PORTDbits.RD0 == 0) key =0;

if(PORTDbits.RD1 == 0) key =1;

if(key == 0)

{

LATAbits.LATA4 = 1;

LATDbits.LATD2 = 0;

for(i=0;i<8;i++)

{

LATB = 1<<i;

delay();

LATB = 0x00;

delay();

}

}

if(key == 1)

{

LATAbits.LATA4 = 0;

LATDbits.LATD2 = 1;

for(i=7;i> 0;i--)

{

LATB = 1<<i;

delay();

LATB = 0x00;

delay();

}

}

}

}

**Square wave**

#include"18F4550ConfigSettings.h"

#include<p18f4550.h>

volatile bit timer\_set = 0;

void timerInit(void)

{

// Timer0 configuration

T0CONbits.TMR0ON = 0; // Stop the timer

T0CONbits.T08BIT = 0; // Run in 16-bit mode

T0CONbits.T0CS = 0; // Use system clock to increment timer

T0CONbits.PSA = 0; // A prescaler is assigned for Timer0

T0CONbits.T0PS2 = 1; // Use a 1:256 prescaler

T0CONbits.T0PS1 = 1;

T0CONbits.T0PS0 = 1;

TMR0H = 0xA4;

TMR0L = 0x6F;

T0CONbits.TMR0ON = 1; // Start the timer

}

void Interrupt\_Init(void)

{

RCONbits.IPEN = 1;

INTCONbits.GIEH = 1;

INTCONbits.GIEL = 1;

INTCONbits.TMR0IE = 1;

INTCONbits.TMR0IF = 0;

INTCON2bits.TMR0IP = 0;

}

void interrupt low\_priority timerinterrupt(void)

{

if(INTCONbits.TMR0IF == 1)

{

T0CONbits.TMR0ON = 0; // Stop the timer

INTCONbits.TMR0IF = 0;

TMR0H = 0xA4;

TMR0L = 0x6F;

timer\_set = 1;

T0CONbits.TMR0ON = 1; // Start the timer

}

}

void main(void)

{

TRISB = 0x00;

LATB = 0xFF;

Interrupt\_Init();

timerInit();

while(1)

{

if(timer\_set )

{

LATB =~LATB;

timer\_set = 0;

}

}

}

**ADC**

#include <p18f4550.h>

#include "LCD.h"

//#include"18F4550ConfigSettings.h"

#include<stdio.h>

void delay(void);

void delay()

{

unsigned int i;

for(i=0;i<5000;i++);

}

void ADCInit(void)

{

TRISEbits.RE1 = 1;

TRISEbits.RE2 = 1;

ADCON1bits.VCFG0 = 0;

ADCON1bits.VCFG1 = 0;

ADCON1bits.PCFG0 =1;

ADCON1bits.PCFG1 =1;

ADCON1bits.PCFG2 =1;

ADCON1bits.PCFG3 =0;

ADCON2bits.ADFM = 1;

ADCON2bits.ACQT = 0x05;

ADCON2bits.ADCS = 0x06;

}

unsigned short Read\_ADC(unsigned char Ch)

{

ADCON0bits.ADON = 1;

ADCON0bits.CHS = Ch;

ADCON0bits.GODONE = 1;

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

ADCON0bits.ADON = 0;

return ADRES;

}

unsigned char hextoascii(unsigned char hexval)

{

if (hexval < 10)

{

return hexval + '0';

}

else

{

return (hexval - 0xA) + 'A';

}

}

void DisplayResult(unsigned short ADCVal)

{

unsigned char i,text[16];

unsigned short tempv;

tempv = ADCVal;

ADCVal = (5500/1024)\*tempv;

sprintf(text,"%04dmv",ADCVal);

for(i=0;i<6;i++)

{

SendData(text[i]);

}

SetLineNumber(2);

for(i=0;i<8;i++)

{

if(tempv & 0x80)

{

SendData('1');

}

else

{

SendData('0');

}

tempv<<=1;

}

}

void main()

{

unsigned short Ch\_result;

TRISB = 0x00;

SPPCON = 0x00;

ADCInit();

InitLCD();

ClearDisplay();

while(1)

{

SetLineNumber(1);

LCDDisplayStr("ADC Value=");

Ch\_result = Read\_ADC(7);

DisplayResult(Ch\_result);

delay();

delay();

}

}

**UART**

//#include"18F4550ConfigSettings.h"

#include<p18F4550.h>

#include"UART1.h"

#include<stdio.h>

void InitUART(unsigned char baudrate)

{

TRISCbits.RC6 = 0; //TX pin set as output

TRISCbits.RC7 = 1; //RX pin set as input

SPBRG = baudrate;

BAUDCONbits.BRG16 = 0;

BAUDCONbits.RXDTP = 0;

BAUDCONbits.TXCKP = 0;

TXSTAbits.SYNC = 0;

TXSTAbits.TX9 = 0;

TXSTAbits.BRGH = 0;

RCSTAbits.CREN = 1;

TXSTAbits.TXEN = 1;

RCSTAbits.SPEN = 1;

}

void SendChar(unsigned char data)

{

while(TXSTAbits.TRMT == 0);

TXREG = data;

}

void putch(unsigned char data)

{

SendChar(data);

}

unsigned char GetChar(void)

{

while(!PIR1bits.RCIF);

return RCREG;

}

void main(void)

{

InitUART(77);

printf("\r\nHello MicroPIC-18F: Enter any Key from Keyboard\r\n");

while(1)

{

printf("%c",GetChar());

}

while(1);

}