**需要 4\*4矩阵键盘 1602屏**

**(1).main.c**

#include "msp430x26x.h"

#include "math.h"

#include "KEY.h"

#include "LCD1602.h"

#define POINT 128

#define VAL 4095

#define CLK 200000

long unsigned int cnt0 = 0;

long unsigned int cnt2 = 0;

long unsigned int cnt3 = 0;

long unsigned int frq = 1000;

unsigned int sina[512] = {0};

unsigned int val\_old = VAL;

unsigned int point\_old = POINT;

unsigned int val = VAL;

unsigned int point = POINT;

unsigned char sign = 1;

unsigned char flag\_net = 0;

void sin\_net()

{

unsigned int i;

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

{

sina[i] = (int)(val/2\*(sin(2\*3.141592\*i/point) + 1));

}

}

void tr\_net()

{

unsigned int i, j;

j = 0;

for(i = 0; i < point/2; i++)

{

sina[j++] = (int)(2.0\*val/point\*i);

}

for( ; i > 0; i--)

{

sina[j++] = (int)(2.0\*val/point\*i);

}

}

void sq\_net()

{

unsigned int i, j;

j = 0;

for(i = 0; i < point/2; i++)

{

sina[j++] = val;

}

for( ; i > 0; i--)

{

sina[j++] = 0;

}

}

void main(void)

{

unsigned int i;

unsigned int val\_temp;

WDTCTL = WDTPW + WDTHOLD; // Stop Watchdog Timer

BCSCTL1 &= ~XT2OFF;

do

{

IFG1 &= ~OFIFG; // 清除振荡器失效标志

for (i = 0xFF; i > 0; i--); // 延时，等待XT2起振

}

while ((IFG1 & OFIFG) != 0);

BCSCTL2 = SELM\_2 + SELS + DIVM\_0 + DIVS\_0;

ADC12CTL0 = REF2\_5V + REFON; // Internal 2.5V ref on

i = 10000;

while(i--) ;

TACCR0 = 80 - 1;

TACCTL0 |= CCIE; // Compare-mode interrupt.

TACTL = TACLR + MC\_1 + TASSEL\_2; // up mode, SMCLK

DAC12\_0CTL = DAC12IR + DAC12AMP\_7 + DAC12ENC; // Int ref gain 1

P4DIR |= 0xFF;

P4OUT |= 0xAA;

KEY\_init();

LCD1602\_init();

cnt0 = point\*frq;

val\_temp = (int)(2.5\*val/4095\*100);

sin\_net();

\_\_bis\_SR\_register(GIE); // Enter LPM0

write\_cmd(0x01);

frq\_display(0x80, frq);

val\_display(0xC0, val\_temp);

while(1)

{

key\_event();

if(key\_flag == 1)

{

P4OUT &= ~key\_val;

key\_flag = 0;

val\_old = val;

point\_old = point;

switch(key\_val)

{

case 10: frq += 1 ;break;

case 11: frq -= 1 ;break;

case 3: frq += 10 ;break;

case 6: frq -= 10 ;break;

case 2: frq += 100 ;break;

case 5: frq -= 100 ;break;

case 1: frq += 1000;break;

case 4: frq -= 1000;break;

case 7: sign = 1;flag\_net = 1;break;

case 8: sign = 2;flag\_net = 1;break;

case 15: sign = 3;flag\_net = 1;break;

case 0: ;break;

case 12: val += 163 ;break;

case 13: val -= 163 ;break;

case 9: val += 16 ;break;

case 14: val -= 16 ;break;

default: break;

}

if(frq > 6000 && frq <= 7000)

frq = 6000;

else if(frq > 7000 || frq == 0)

frq = 1;

if(val > 4095 && val <= (4095 + 163))

val = 4095;

else if(val > (4095 + 163))

val = 0;

if(frq <= 750)

point = 256;

else if(frq <= 1500 && frq >= 751)

point = 128;

else if(frq <= 3000 && frq >= 1501)

point = 64;

else if(frq <= 6000 && frq >= 3001)

point = 32;

if(val != val\_old || point != point\_old || flag\_net == 1)

{

flag\_net = 0;

val\_old = 0;

point\_old = 0;

val\_temp = (int)(2.5\*val/4095\*100);

switch(sign)

{

case 1: sin\_net();break;

case 2: tr\_net() ;break;

case 3: sq\_net() ;break;

default: break;

}

}

write\_cmd(0x01);

frq\_display(0x80, frq);

val\_display(0xC0, val\_temp);

key\_val = 100;

cnt0 = point\*frq;

}

}

}

#pragma vector = TIMERA0\_VECTOR

\_\_interrupt void TA0\_ISR(void)

{

if(cnt3 >= CLK)

{

cnt3 -= CLK;

DAC12\_0DAT = sina[cnt2++];

if(cnt2 >= point)

cnt2 = 0;

}

cnt3 += cnt0;

}

**(2). Key.h**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*全局变量\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char key\_pressed = 0; //按键是否被按下:1--是，0--否

unsigned char key\_val = 0; //存放键值

unsigned char key\_flag = 1; //按键是否已放开：1--是，0--否

unsigned char key\_old = 0;

unsigned char key\_new = 0;

unsigned char time = 100;

unsigned char key\_map[] = {15,0,14,13,7,8,9,12,4,5,6,11,1,2,3,10};

unsigned char flag\_B = 1;

unsigned int tb\_cnt = 0;

void TB\_init()

{

TBCTL = TBCLR + TBSSEL\_1 + MC\_1;

TBCCTL0 = CCIE;

TBCCR0 = 500 - 1;

}

#pragma vector = TIMERB0\_VECTOR

\_\_interrupt void TB0\_ISR()

{

tb\_cnt++;

}

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

函数名称：Init\_Keypad

功 能：初始化扫描键盘的IO端口

参 数：无

返回值 ：无

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

void KEY\_init(void)

{

TB\_init();

P1DIR = BIT1 + BIT3 + BIT5 + BIT7; //P1.0~P1.3设置为输入状态, P1.4~P1.7设置为输出状态

P2DIR &= ~(BIT1 + BIT3 + BIT5 + BIT7);

P2REN |= BIT1 + BIT3 + BIT5 + BIT7;

P2OUT |= BIT1 + BIT3 + BIT5 + BIT7;

P1OUT |= BIT1 + BIT3 + BIT5 + BIT7; // P1.4~P1.7输出高电平

}

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

\* check\_key(),检查按键，确认键值

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

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

函数名称：check\_key

功 能：扫描键盘的IO端口，获得键值

参 数：无

返回值 ：无

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

void check\_key(void)

{

unsigned char row ,col,tmp1,tmp2;

tmp1 = 0x80;

for(row = 0;row < 4;row++) //行扫描

{

P1OUT = BIT1 + BIT3 + BIT5 + BIT7; //P1.4~P1.7输出全1

P1OUT -= tmp1; //P1.4~p1.7输出四位中有一个为0

tmp1 >>= 2;

if ((P2IN & 0xAA) < 0xAA) //是否P1IN的P1.0~P1.3中有一位为0

{

tmp2 = 0x02; // tmp2用于检测出那一位为0

for(col = 0;col < 4;col++) // 列检测

{

if((P2IN & tmp2) == 0x00) // 是否是该列,等于0为是

{

key\_val = key\_map[row \* 4 + col]; // 获取键值

return; // 退出循环

}

tmp2 <<= 2; // tmp2右移1位

}

}

}

}

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

函数名称：key\_event

功 能：检测按键，并获取键值

参 数：无

返回值 ：无

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

void key\_event(void)

{

unsigned char tmp;

P1OUT &= 0x00; // 设置P1OUT全为0，等待按键输入

tmp = P2IN; // 获取 p1IN

if ((key\_pressed == 0x00)&&((tmp & 0x0f) < 0x0f)) //如果有键按下

{

key\_pressed = 1; // 如果有按键按下，设置key\_pressed标识

}

else if (key\_pressed == 1)

{

if((tmp & 0x0f) == 0x0f) //如果按键已经释放

{

key\_pressed = 0; // 清除key\_pressed标识

if(tb\_cnt > 10)

key\_flag = 1; // 设置key\_flag标识

tb\_cnt = 0;

time = 100;

}

else if(tb\_cnt >= time)

{

tb\_cnt = 0;

time = 10;

if(flag\_B == 1)

{

check\_key();

key\_old = key\_val;

flag\_B = 0;

}

else if(flag\_B == 0)

{

check\_key();

key\_new = key\_val;

flag\_B = 1;

if(key\_old == key\_new && key\_val != 100)

{

key\_flag = 1;

key\_pressed = 0; // 清除key\_pressed标识

}

}

}

}

else

{

\_NOP();

}

}

**(3) LCD1602.h**

#define WIDTH 5 //显示的数据宽度

void lcd\_delay(unsigned int t)

{

while(t) t--;

}

void lcd1602\_io() //端口初始化

{

P1DIR |= BIT0 + BIT2 + BIT4 + BIT6; //8位数据口

P2DIR |= BIT0 + BIT2 + BIT4 + BIT6;

P1OUT &= ~(BIT0 + BIT2 + BIT4 + BIT6);

P2OUT &= ~(BIT0 + BIT2 + BIT4 + BIT6);

P3DIR |= BIT2+BIT3+BIT4; //RS RW EN

P3OUT &= ~(BIT2 + BIT3 + BIT4);

}

unsigned char chk\_busy() //检忙

{

unsigned char busy;

//P4DIR=0x00;

P1DIR &= ~(BIT0 + BIT2 + BIT4 + BIT6);

P2DIR &= ~(BIT0 + BIT2 + BIT4 + BIT6);

P3OUT&=~BIT2;

P3OUT|=BIT3;

P3OUT|=BIT4;

busy=P2IN&0x55;

busy=busy&0x40;

P1DIR |= BIT0 + BIT2 + BIT4 + BIT6; //8位数据口

P2DIR |= BIT0 + BIT2 + BIT4 + BIT6;

lcd\_delay(1);

P3OUT&=~BIT4;

return busy;

}

void write\_cmd(unsigned char cmd) //向1602写指令

{

unsigned char tmp1,tmp0;

tmp0 = ((cmd & 0x08) << 3);

tmp0 += ((cmd & 0x04) << 2);

tmp0 += ((cmd & 0x02) << 1);

tmp0 += ((cmd & 0x01) << 0);

tmp1 = ((cmd & 0x80) >> 1);

tmp1 += ((cmd & 0x40) >> 2);

tmp1 += ((cmd & 0x20) >> 3);

tmp1 += ((cmd & 0x10) >> 4);

while(chk\_busy());

P3OUT|=BIT4;

P3OUT&=~BIT2;

P3OUT&=~BIT3;

//P4OUT=cmd;

P1OUT = tmp0 + 0xAA;

P2OUT = tmp1 + 0xAA;

lcd\_delay(1);

P3OUT&=~BIT4;

}

void write\_data(unsigned char dat) //向1602写数据

{

unsigned char tmp1,tmp0;

tmp0 = ((dat & 0x08) << 3);

tmp0 += ((dat & 0x04) << 2);

tmp0 += ((dat & 0x02) << 1);

tmp0 += ((dat & 0x01) << 0);

tmp1 = ((dat & 0x80) >> 1);

tmp1 += ((dat & 0x40) >> 2);

tmp1 += ((dat & 0x20) >> 3);

tmp1 += ((dat & 0x10) >> 4);

while(chk\_busy());

P3OUT|=BIT4;

P3OUT|=BIT2;

P3OUT&=~BIT3;

P1OUT = tmp0 + 0xAA;

P2OUT = tmp1 + 0xAA;

lcd\_delay(1);

P3OUT&=~BIT4;

}

void LCD1602\_init()//屏幕初始化

{

lcd1602\_io();

write\_cmd(0x38); //置功能 4位总线?? 双行显示

write\_cmd(0x08); //显示开，无光标，也无闪烁

write\_cmd(0x0c); //置输入模式 光标返回

write\_cmd(0x01); //清屏

}

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

//////////////////十进制格式显示数据/////////////////////////////////////////////////////

void frq\_display(unsigned char add,unsigned int data) //显示地址 要显示的数据

{

unsigned char i, rang;

unsigned int temp0,temp1;

rang = 0;

temp0 = data;

/\*\*\*\*计算要显示数据的位数\*\*\*\*/

do

{

rang++;

temp0 /= 10;

}

while(temp0 != 0);

write\_cmd(add);

write\_data('F');

write\_data('r');

write\_data('q');

write\_data(':');

temp0 = data;

i = WIDTH - 1;

/\*\*\*\*\*\*　显示数据　\*\*\*\*\*\*/

for( ;rang > 0; rang--)

{

temp1 = temp0%10;

temp0 /= 10;

write\_cmd(add + i + 4);

i--;

write\_data(temp1 + '0');

}

/\*\*\*\*\*\*显示单位\*\*\*\*\*\*\*\*\*/

/\*\*/

write\_cmd(add + WIDTH + 4);

write\_data(' ');

write\_data('H');

write\_data('Z');

}

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

//////////////////十进制格式显示数据/////////////////////////////////////////////////////

void val\_display(unsigned char add,unsigned int data) //显示地址 要显示的数据

{

unsigned char i, rang;

unsigned int temp0,temp1;

rang = 0;

temp0 = data;

rang = 4;

temp0 = data;

write\_cmd(add);

write\_data('V');

write\_data('p');

write\_data('p');

write\_data(':');

i = WIDTH - 1 + 4;

/\*\*\*\*\*\*　显示数据　\*\*\*\*\*\*/

for( ;rang > 0; rang--)

{

write\_cmd(add+i);

if(i == (WIDTH - 3 + 4))

{

write\_data('.');

i--;

continue;

}

temp1=temp0%10;

temp0/=10;

i--;

write\_data(temp1 + '0');

}

write\_cmd(add + WIDTH + 4);

write\_data(' ');

write\_data('V');

}

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