|  |
| --- |
|  |
| **51单片机拓展集** |
| --为《51单片机代码集》的补充部分 |

|  |
| --- |
| Author：王卫东  2023-4-17  Email：1598593280@qq.com |

目录

[**一、一些提醒** 2](#_Toc132990630)

[有关定时器设置： 2](#_Toc132990631)

[开发板AD转换芯片检测通道： 3](#_Toc132990632)

[**二、CS10A超声波测距** 4](#_Toc132990633)

[CS-100A.h 4](#_Toc132990634)

[CS100A.h 4](#_Toc132990635)

[CS100A.c 4](#_Toc132990636)

[**三、DS1302时钟芯片** 6](#_Toc132990637)

[DS1302.h 6](#_Toc132990638)

[DS1302.c 6](#_Toc132990639)

[**四、TM1838八位LED** 10](#_Toc132990640)

[TM1838.h 10](#_Toc132990641)

[main.c 12](#_Toc132990642)

[**五、开发板原理图** 14](#_Toc132990643)

[**六、模块原理图** 15](#_Toc132990644)

[1.TMPI1638原理图 15](#_Toc132990645)

**一、一些提醒**

有关定时器设置：

设置定时器0：中断号1

TMOD |= 0X01（方式1,16位定时器）/0X02（方式2,8位自动装填初值）

ET0 = 1;

TR0 = 1;

EA = 1

设置定时器1：中断号3

TMOD |= 0X10（方式1,16位定时器）/0X20（方式2，八位自动装填）

ET1 = 1;

TR1 = 1;

EA = 1;

设置计数器0 中断号1

TMOD |=0X05（方式1,16位计数器）/0X06（方式2，八位自动装填）

设置计数器1 中断号3

TMOD |= 0X50（方式1,16位计数器）/0X60（方式2，八位自动装填）

设置外部中断 外部中断0中断号为0，外部中断1中断号为2

EX0 = 1 / EX1 = 1;（外部中断0，外部中断1）

IT0 = 1 / IT1 = 1;（同上）

EA = 1;

开发板AD转换芯片检测通道：

检测电位器阻值：0X94

检测光敏电阻：0XA4

检测热敏电阻：0XD4

检测外部IN3模拟信号：0XE4

**二、CS10A超声波测距**

CS-100A.h

#include<reg52.h>

void CS100A\_init(void);

void measure\_distance(void);

void launch\_wave(void);

sbit input = P1^0;

CS100A.h

#include<reg52.h>

#include "main.h"

void CS100A\_init(void);

void measure\_distance(void);

void launch\_wave(void);

void data\_dealing(void);

sbit input = P1^0;

sbit output = P1^1;

sbit LED = P2^0;

extern u16 count;

extern u16 a;

extern u16 b;

extern u16 c;

//extern u16 d;

extern u16 flag;

CS100A.c

#include<reg52.h>

#include "CS100A.h"

#include "main.h"

#include "delay.h"

#include "lcd12864.h"

u16 flag = 0;

u16 count = 0;

u16 a, b, c;

long int distance = 0;

#define VELOCITY\_30C 3495 //30摄氏度时的声速，声速V= 331.5 + 0.6\*温度；

#define VELOCITY\_23C 3453 //23摄氏度时的声速，声速V= 331.5 + 0.6\*温度；

void CS100A\_init()

{

input = 0;

output = 0;

count = 0;

distance = 0;

}

void launch\_wave()

{

output = 1;

delay\_us(50);

output = 0;

}

void measure\_distance()

{

u16 l;

u16 h, y;

TR0 = 1;

while (input);

TR0 = 0;

l = TL0;

h = TH0;

y = (h << 8) | l;

y = y - 64536;//us部分

distance = y + 1000 \* count;//计算总时间

TH0 = (65535 - 921) / 256;

TL0 = (65535 - 921) % 256;

delay\_ms(32);

distance = VELOCITY\_30C \* distance / 200000;

}

void data\_dealing()

{

a = distance / 100;

b = (distance % 100) / 10;

c = distance % 10;

//d = distance % 10;

}

**三、DS1302时钟芯片**

DS1302.h

#include<reg52.h>

#include "main.h"

#include "intrins.h"

sbit DS1302\_CE = P3^5; //:DS1302使能端口

sbit DS1302\_IO = P3^4; //：DS1302数据端口

sbit DS1302\_SCLK = P3^6; //：DS1302时钟信号端口

void DS1302\_init();

extern u8 year[7];

DS1302.c

#include<reg52.h>

#include "DS1302.h"

#include "main.h"

#include "intrins.h"

u8 code DS1302\_write\_date[7] = {0x80,0x82,0x84,0x86,0x88,0x8a,0x8c};

u8 code DS1302\_read\_date[7] = {0x81,0x83,0x85,0x87,0x89,0x8b,0x8d};

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DS1302初始化时间为2023年1月22日星期日17时0分0秒\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u8 year[7] = {0x00,0x41,0x21,0x22,0x01,0x07,0x23};

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DS1302写入数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DS1302\_write(u8 addr,u8 dat)

{

u8 n;

DS1302\_CE = 0;

\_nop\_();

DS1302\_SCLK = 0;

\_nop\_();

DS1302\_CE = 1;

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

{

DS1302\_IO = addr & 0x01;

addr >>= 1;

DS1302\_SCLK = 1;

\_nop\_();

DS1302\_SCLK = 0;

\_nop\_();

}

\_nop\_();

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

{

DS1302\_IO = dat & 0x01;

dat >>= 1;

DS1302\_SCLK = 1;

\_nop\_();

DS1302\_SCLK = 0;

\_nop\_();

}

DS1302\_CE = 0;

\_nop\_();

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DS1302读数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

u8 DS1302\_read(u8 addr)

{

u8 n,dat,dat1;

DS1302\_CE = 0;

\_nop\_();

DS1302\_SCLK = 0;

\_nop\_();

DS1302\_CE = 1;

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

{

DS1302\_IO = addr & 0x01;

addr >>= 1;

DS1302\_SCLK = 1;

\_nop\_();

DS1302\_SCLK = 0;

\_nop\_();

}

\_nop\_();

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

{

dat1 = DS1302\_IO;

dat = (dat>>1) | (dat1<<7);

DS1302\_SCLK = 1;

\_nop\_();

DS1302\_SCLK = 0;

\_nop\_();

}

DS1302\_CE = 0;

\_nop\_();

DS1302\_SCLK = 1;

\_nop\_();

DS1302\_IO = 0;

\_nop\_();

DS1302\_IO = 1;

\_nop\_();

return dat;

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DS1302初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DS1302\_init()

{

u8 n = 0;

DS1302\_write(0x8e,0x00); //:关闭写保护：WP为0时关闭写保护，为1时开启写保护

for (n = 0;n < 7;n++)

{

DS1302\_write(DS1302\_write\_date[n],year[n]);

}

DS1302\_write(0x8e,0x80);

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*读取时钟信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DS1302\_read\_time()

{

u8 n = 0;

for (n = 0;n < 7;n++)

{

year[n] = DS1302\_read(DS1302\_read\_date[n]);

}

}

**四、TM1838八位LED**

TM1838.h

#ifndef \_TM1638\_H

#define \_TM1638\_H

#include <REG52.H>

#define DATA\_COMMAND 0X40

#define DISP\_COMMAND 0x80

#define ADDR\_COMMAND 0XC0

//TM1638模块引脚定义

sbit DIO = P1 ^ 0;

sbit CLK = P1 ^ 1;

sbit STB = P1 ^ 2;

//共阴数码管显示代码

unsigned char code tab[] = { 0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,

0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71 };

void TM1638\_Write(unsigned char DATA) //写数据函数

{

unsigned char i;

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

{

CLK = 0;

if (DATA & 0X01)

DIO = 1;

else

DIO = 0;

DATA >>= 1;

CLK = 1;

}

}

unsigned char TM1638\_Read(void) //读数据函数

{

unsigned char i;

unsigned char temp = 0;

DIO = 1; //设置为输入

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

{

temp >>= 1;

CLK = 0;

if (DIO)

temp |= 0x80;

CLK = 1;

}

return temp;

}

void Write\_COM(unsigned char cmd) //发送命令字

{

STB = 0;

TM1638\_Write(cmd);

STB = 1;

}

unsigned char Read\_key(void)

{

unsigned char c[4], i, key\_value = 0;

STB = 0;

TM1638\_Write(0x42); //读键扫数据 命令

for (i = 0; i < 4; i++)

c[i] = TM1638\_Read();

STB = 1; //4个字节数据合成一个字节

for (i = 0; i < 4; i++)

key\_value |= c[i] << i;

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

if ((0x01 << i) == key\_value)

break;

return i;

}

void Write\_DATA(unsigned char add, unsigned char DATA) //指定地址写入数据

{

Write\_COM(0x44);

STB = 0;

TM1638\_Write(0xc0 | add);

TM1638\_Write(DATA);

STB = 1;

}

void Write\_oneLED(unsigned char num, unsigned char flag) //单独控制一个LED函数，num为需要控制的led序号，flag为0时熄灭，不为0时点亮

{

if (flag)

Write\_DATA(2 \* num + 1, 1);

else

Write\_DATA(2 \* num + 1, 0);

}

void Write\_allLED(unsigned char LED\_flag) //控制全部LED函数，LED\_flag表示各个LED状态，其中1位点亮LED1，128点亮LED8

{

unsigned char i;

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

{

if (LED\_flag & (1 << i))

//Write\_DATA(2\*i+1,3);

Write\_DATA(2 \* i + 1, 1);

else

Write\_DATA(2 \* i + 1, 0);

}

}

//TM1638初始化函数

void init\_TM1638(void)

{

unsigned char i;

Write\_COM(0x8b); //亮度 (0x88-0x8f)8级亮度可调

Write\_COM(0x40); //采用地址自动加1

STB = 0; //

TM1638\_Write(0xc0); //设置起始地址

for (i = 0; i < 16; i++) //传送16个字节的数据

TM1638\_Write(0x00);

STB = 1;

}

#endif

main.c

/\*该演示示例说明了该如何使用这个集成模块，下面对这个程序中实现功能函数进行详细说明

1.Read\_key函数为取键值函数，执行一次该函数可取得一个键值，与开发板相同

2.Write\_DATA函数为数码管显示函数，其调用非常方便，其中第一个传入参数为需要其显示的数码管位数乘以2，

第二个参数为需要显示的数字，其为tab[]数组里的元素。例如，如果要在第三个位置显示数字7，则两个传入参

数分别应为2\*2 或 4，以及tab[7]。

3.Write\_oneLED函数为控制一个LED的函数，若需要使第4个LED亮，则两个传入参数应为3和1，灭则为3和0

4.Write\_allLED函数为控制所有LED的函数，用二进制表示各个led的亮灭，如00000001位点亮LED1，10000000为

点亮LED8，这也是该函数的传入参数

5.其初始化过程如实例所示\*/

#include <REG52.H>

#include <tm1638.h>

unsigned char num[8]; //各个数码管显示的值

int main(void)

{

unsigned char i;

init\_TM1638(); //初始化TM1638

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

Write\_DATA(i<<1,tab[0]); //初始化寄存器

while(1)

{

// i=Read\_key(); //读按键值

// if(i<8)

// {

// num[i] = i;

// while(Read\_key()==i); //等待按键释放

// if(num[i]>15)

// num[i]=0;

// Write\_DATA(i\*2,tab[num[i]]);

// Write\_allLED(1<<i);

// }

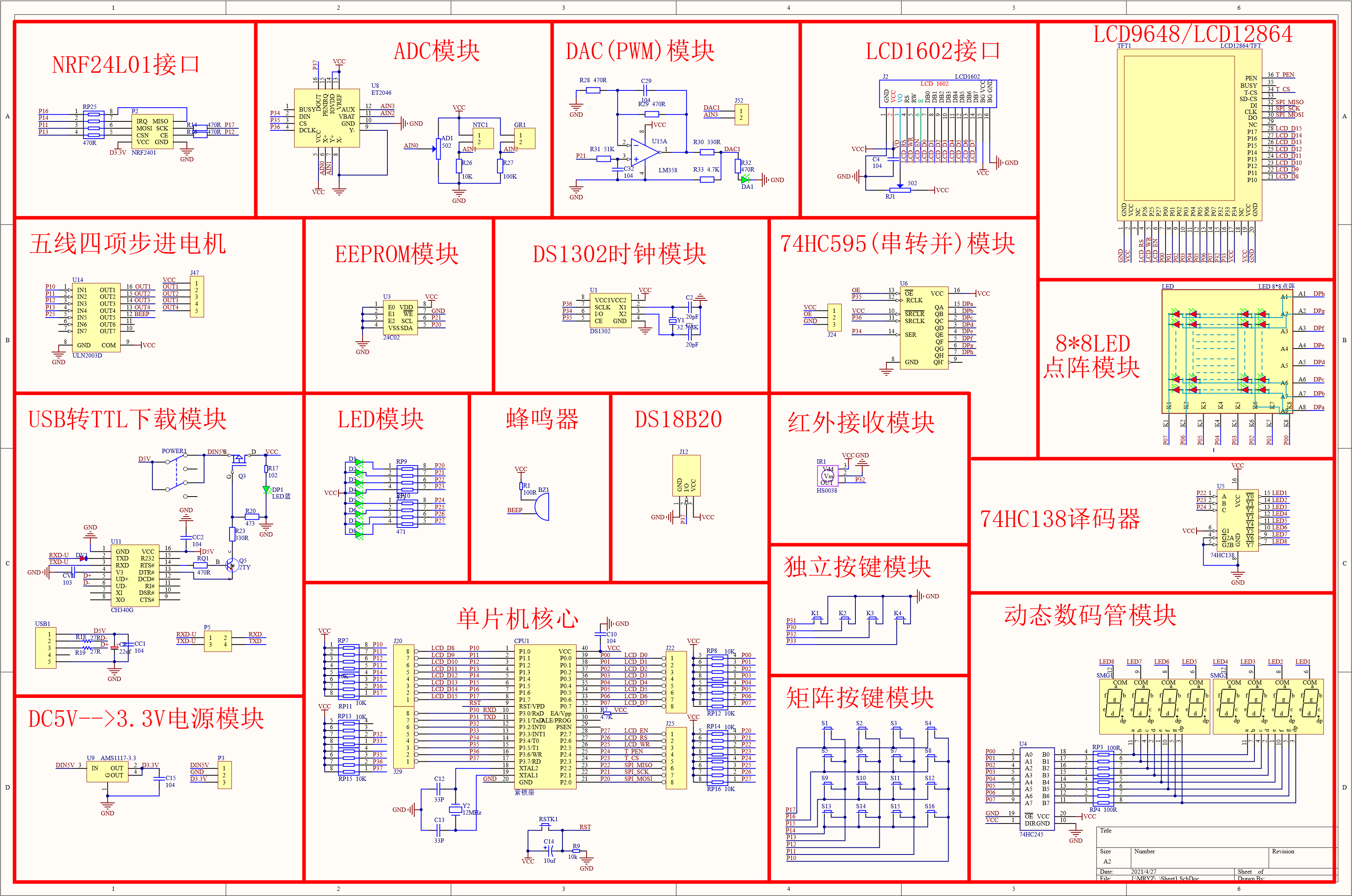
Write\_DATA(8,tab[4]);

Write\_allLED(1);

}

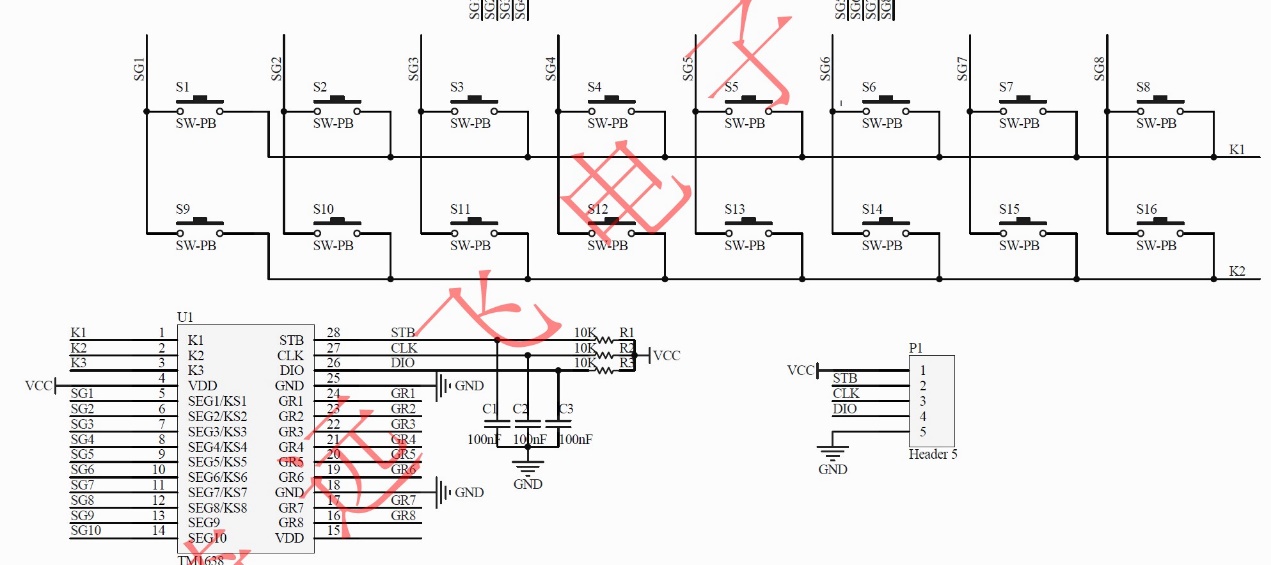
}

**五、开发板原理图**



**六、模块原理图**

1.TMPI1638原理图



说明：