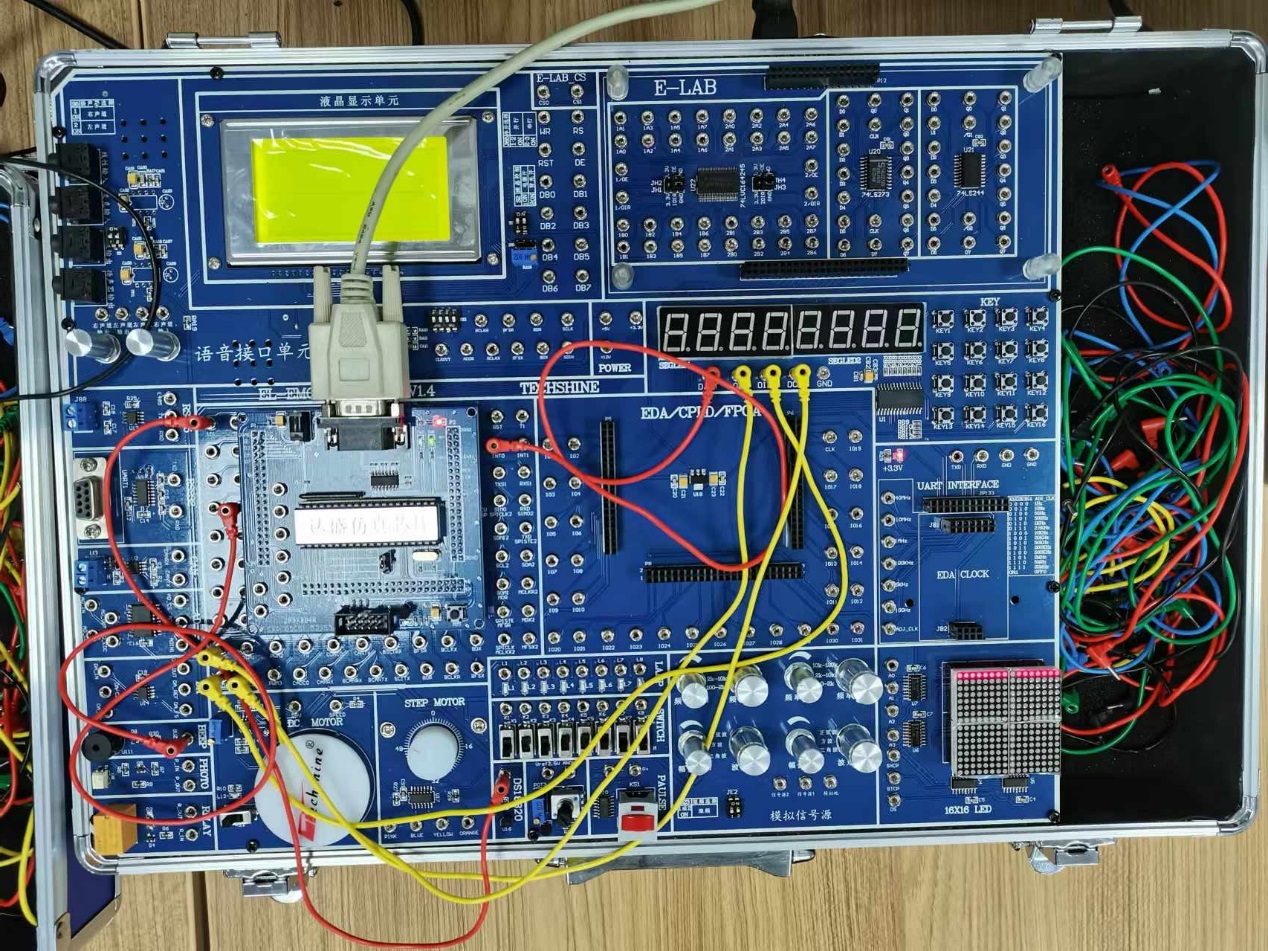
**实验报告五 键盘与温度测量**

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 组内分工 | 工作量占百分比 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

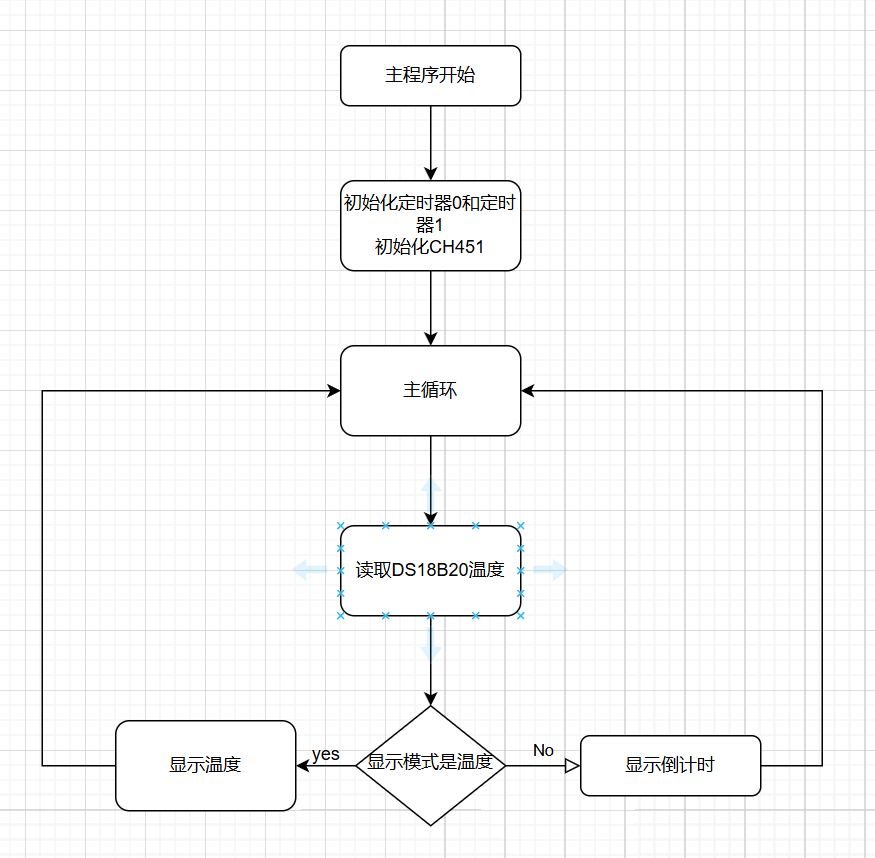
一、电路图

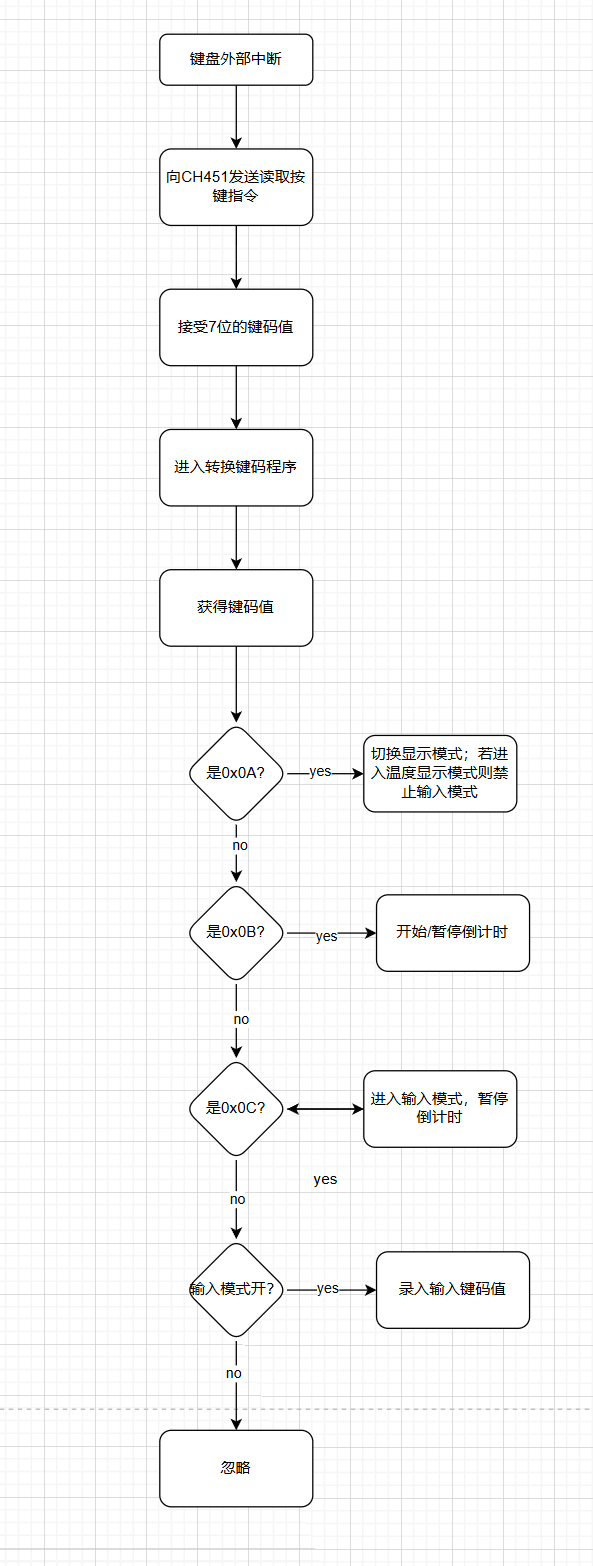
按照要求设计的电路图（截图），及简要介绍。



二、程序分析

1、以流程图的形式给出程序设计的思路，





2、代码及必要注释。

main.c

#include <reg51.h>

#include "main.h"

#include "delay.h"

#include "ch451.h"

#include "ds18b20.h"

char flashDigit = 0;

sbit BUZZER1=P2^0;

sbit P20=P2^0;

unsigned int buz\_cnt = 0;

bit flashFlag = 0;

bit buz\_flag=0;

bit cnt\_or\_tmp = 0;

unsigned char dig4, dig3, dig2, dig1;

// 显示倒计时

void printNum()

{

// 0-3位用作显示倒计时

ch451\_write(CH451\_DIG0 | dig1);

ch451\_write(CH451\_DIG1 | dig2);

ch451\_write(CH451\_DIG2 | (dig3+128)); //点号

ch451\_write(CH451\_DIG3 | dig4);

if(flashDigit > 0 && flashFlag)

// DIG3 通过加上位数\*0x0100来选择显示的数码管是哪个，或27表示不显示

// flashFlag 翻转，实现闪烁

ch451\_write((0x0700 + flashDigit \* 0x0100) | 27);

flashFlag = !flashFlag;

// 高位清空不显示

ch451\_write(CH451\_DIG4 | 27);

ch451\_write(CH451\_DIG5 | 27);

ch451\_write(CH451\_DIG6 | 27);

ch451\_write(CH451\_DIG7 | 27);

}

void printTemperature(unsigned int temp)

{

// 0-3位用作显示温度

unsigned char digit;

digit = temp % 10;

ch451\_write(CH451\_DIG0 | digit);

digit = (temp/=10) % 10;

ch451\_write(CH451\_DIG1 | (digit + 128)); // 点号

digit = (temp/=10) % 10;

ch451\_write(CH451\_DIG2 | digit);

// 高位清空不显示

ch451\_write(CH451\_DIG3 | 27);

ch451\_write(CH451\_DIG4 | 27);

ch451\_write(CH451\_DIG5 | 27);

ch451\_write(CH451\_DIG6 | 27);

ch451\_write(CH451\_DIG7 | 27);

}

// 重置DS18B20

void startTemperature(void){

reset\_ds18b20();

delay\_ms(1);

writeByte\_ds18b20(0xCC);

writeByte\_ds18b20(0x44);

}

unsigned int readTemperature(void){

unsigned int tempInt;

float tempFloat;

unsigned char a, b;

reset\_ds18b20();

delay\_ms(1);

writeByte\_ds18b20(0xCC);

writeByte\_ds18b20(0xBE);

a = readByte\_ds18b20();

b = readByte\_ds18b20();

tempInt = b;

tempInt <<= 8;

tempInt = tempInt | a;

tempFloat = tempInt \* 0.0625;

tempInt = tempFloat \* 10 + 0.5;

return tempInt;

}

unsigned int seconds;

unsigned int displayNum;

unsigned char count;

bit paused = 0;

unsigned int BEEP1;

#define TIMER\_PRESET\_VALUE 50000 // 1/1000000 s = 1us, 50000us = 50ms

#define TIMER\_1\_INT 10000 // 10ms

void timer0\_ISR(void) interrupt 1

{

count += 1;

// 每秒进入一次，更新秒数

if(count % 20 == 0)

{

if(!paused)

{

if(seconds > 0)

{

seconds -= 1;

// 倒计时到0，置buz\_flag为1

if (seconds == 0)

buz\_flag = 1;

}

dig4 = seconds / 600 % 10;

dig3 = seconds / 60 % 10;

dig2 = seconds % 60 / 10;

dig1 = seconds % 10;

}

}

// 通过cnt\_or\_tmp判断显示温度还是倒计时，用于显示

if(count % 8 == 0)

{

if (cnt\_or\_tmp)

// 显示倒计时

printNum();

else

// 显示温度

printTemperature(readTemperature());

}

// 倒计时到0，且flag为1

// 此时开启蜂鸣器

if (seconds == 0 && buz\_flag == 1)

{

P20=1; // 打开蜂鸣器，P20高电平，计数到三秒后重设为低电平

buz\_flag = 0; // 进入后马上清除flag，否则会重复进入该if语句

buz\_cnt = 1;

}

// 蜂鸣器计数，计数到3000/50=60次，即3秒后关闭蜂鸣器，重置

if (buz\_cnt > 0)

{

buz\_cnt++;

if (buz\_cnt==60)

{

P20=0; // 关闭蜂鸣器

buz\_cnt=0; // 重置计数

}

}

// 重设

TH0 = (65536 - TIMER\_PRESET\_VALUE) / 256; // 设置定时器的高字节

TL0 = (65536 - TIMER\_PRESET\_VALUE) % 256; // 设置定时器的低字节

}

// 按下键盘中断

bit setTimeMode = 0;

void keydown(void) interrupt 0

{

unsigned char key;

unsigned char num;

EA = 0;

key = ch451\_read();

EA = 1;

// 开始设置 / 重置输入

if(key == KEY\_16)

{

setTimeMode = 1;

paused = 1;

flashDigit = 4;

}

// 设置完毕

else if(key == KEY\_15)

{

seconds = dig4 \* 600 + dig3 \* 60 + dig2 \* 10 + dig1;

setTimeMode = 0; // 结束设置时间

paused = 0; // 开始计时

flashDigit = 0; // 闪烁位清零，不再闪烁

buz\_flag=0; // 重置flag

}

// PAUSE/RUN

else if(key == KEY\_13 && !setTimeMode) // 按下KEY\_13且在 非设置时间模式

{

paused = !paused;

}

else if (key == KEY\_12) // 切换cnt\_or\_tmp，显示温度或者倒计时

{

cnt\_or\_tmp = !cnt\_or\_tmp;

}

else // 数字键, KEY1/2/3/5/6/7/9/10/11/14

{

if(setTimeMode)

{

if(key == KEY\_1)num=7;

else if(key == KEY\_2)num=8;

else if(key == KEY\_3)num=9;

else if(key == KEY\_5)num=4;

else if(key == KEY\_6)num=5;

else if(key == KEY\_7)num=6;

else if(key == KEY\_9)num=1;

else if(key == KEY\_10)num=2;

else if(key == KEY\_11)num=3;

else if(key == KEY\_14)num=0;

else return;

if(flashDigit == 4)

dig4 = num;

if(flashDigit == 3)

dig3 = num;

if(flashDigit == 2)

{

if(num >= 6) num = 5; // 限制秒的高位，最大为5

dig2 = num;

}

if(flashDigit == 1) // 秒的低位

dig1 = num;

// 闪烁位递减

flashDigit = flashDigit > 1 ? flashDigit-1 : 4;

}

}

}

void main(void)

{

// 初始化buz\_flag, buz\_cnt, P20

buz\_cnt = 0;

buz\_flag = 0;

P20=0;

ch451\_write(0x201);

ch451\_init();

// 系统参数设定：显示驱动使能，键盘扫描使能，其余默认

ch451\_write(0x403);

// 显示参数设定：选择 BCD 译码方式，其余默认

ch451\_write(0x580);

// 设置定时器的工作模式和预设值

TMOD = 0x11;

TH0 = (65536 - TIMER\_PRESET\_VALUE) / 256; // 设置定时器的高字节

TL0 = (65536 - TIMER\_PRESET\_VALUE) % 256; // 设置定时器的低字节

// 开启定时器并使能定时器中断

TR0 = 1; // 开启定时器

ET0 = 1; // enable timer0 int

IT0 = 0; // 低电平触发

EX0 = 1; // enable timer0 int

EA = 1; // 开总中断

while(1)

{

startTemperature();

delay\_ms(1000);

//printTemperature(readTemperature());

}

}

Ch451.c

#include <reg51.h>

#include "ch451.h"

sbit CH451\_DOUT = P3^2; // DOUT INT0

sbit CH451\_DIN = P1^1; // DIN SDA

sbit CH451\_CLK = P1^2; // DCLK SCL

sbit CH451\_LOAD = P1^3; // LOAD PWM0

void ch451\_init(void)

{

CH451\_DIN = 0; /\*先低后高，选择4线输入\*/

CH451\_DIN = 1;

CH451\_LOAD = 1;

CH451\_CLK = 1;

}

void ch451\_write(unsigned int command)

{

unsigned char i;

CH451\_LOAD=0; /\*命令开始\*/

for(i=0;i<12;i++)

{

CH451\_DIN=command&1;

CH451\_CLK=0;

CH451\_CLK=1;

command>>=1; /\*上升沿有效\*/

}

CH451\_LOAD=1; /\*加载数据\*/

}

unsigned int ch451\_read()

{

unsigned char key;

unsigned char i;

ch451\_write(0x700);

key = 0;

for(i=0;i<7;i++) //从CH451读出按键值

{

key <<= 1;

key |= CH451\_DOUT;

CH451\_CLK = 0;

CH451\_CLK = 1;

}

return key;

}

三、思考题（见PPT）

1. 说明单线通信的优点和缺点。

优点:

1. 结构简单:只需要一根信号线和一个公共地线即可实现通信,硬件连接简单。
2. 成本低:由于只需要很少的硬件,因此系统成本较低。
3. 抗干扰能力强:单线传输可以采用差分信号,有利于抑制共模干扰。
4. 节省I/O端口:只占用一个I/O口,可以节省微控制器的I/O资源。
5. 适合远距离传输:单线传输可以实现较远距离的通信。
6. 易于实现多设备连接:可以方便地实现总线式连接,连接多个设备。

缺点:

1. 传输速率低:由于只有一根信号线,传输速度受到限制。
2. 通信效率较低:需要额外的时序控制来区分数据位,降低了有效数据传输率。
3. 同步困难:单线通信往往采用异步通信方式,同步较困难。
4. 容易受噪声影响:单根线路容易受到外部噪声干扰。
5. 通信距离有限:虽然可以实现远距离传输,但距离增加会导致信号衰减。
6. 协议复杂:为了实现可靠通信,需要设计相对复杂的通信协议。
7. 双向通信困难:在同一根线上实现双向通信需要复杂的时序控制。
8. 故障诊断困难:由于只有一根信号线,故障定位和诊断相对困难。

2. 影响单片机精确定时的因素有哪些？如何 尽量避免这些因素的影响？

影响因素:

1. 晶振精度:  
   晶振的频率误差直接影响定时精度。
2. 温度变化:  
   温度变化会影响晶振频率和单片机内部电路的工作特性。
3. 电源波动:  
   电源电压的波动会影响单片机的工作频率。
4. 中断延迟:  
   中断响应时间会影响定时精度。
5. 程序执行时间:  
   不同指令的执行时间不同,影响定时精度。
6. 时钟分频:  
   时钟分频可能引入额外的误差。
7. 定时器分辨率:  
   定时器的位数限制了定时的精度。
8. 外部干扰:  
   电磁干扰可能影响单片机的正常工作。
9. 老化效应:  
   长期使用后,晶振和其他元件的性能可能发生变化。
10. 软件设计:  
    不合理的软件设计可能导致定时不准确。

避免措施:

1. 使用高精度晶振:  
   选用温度补偿晶振(TCXO)或恒温晶振(OCXO)提高精度。
2. 温度补偿:  
   使用温度传感器进行实时温度补偿。
3. 稳压措施:  
   采用高质量的稳压电源,必要时使用电池供电。
4. 优化中断处理:  
   减少中断处理时间,使用高优先级中断。
5. 使用汇编语言:  
   关键定时部分使用汇编语言编程,精确控制指令执行时间。
6. 谨慎使用分频:  
   尽量避免使用分频,或选择合适的分频比。
7. 选择合适的定时器:  
   使用高位数定时器提高分辨率。
8. EMI防护:  
   做好电路板EMI设计,必要时使用屏蔽。
9. 定期校准:  
   对系统进行定期校准,补偿老化效应。
10. 软件优化:  
    使用定时器中断而非延时函数,避免长时间关中断。

3. 如果要求检测到按键抬起时才执行对应功能，如何修改程序？

在进入按键中断程序后，检测当前按键码第7位状态位，若按键为按下状态则不做操作。利用定时器中断定时发送读键码请求，直到按键状态转换为抬起再执行相关操作。

四、问题分析

1、实验过程中遇到的问题及解决方法。

1）程序始终进不去主循环，发现是在某一个初始化中使用了循环，循环的递减变量使用了Unsigned类型，减到0后回到正无穷，循环始终出不去

2）键盘按键没有反应，debug后确定原因是外部中断函数始终进不去，上网查找发现是EX0外部中断0允许位没有打开

2、实验的收获或感想。（可选）

循环变量不要用unsigned类型；多利用网上的51相关资源