西南交通大学

电子信息工程专业

实

验

报

告

册

**课程名称： MCU系统综合实验**

**学生姓名： 王乙斯**

**所在班级： 电子(轨道）2022-01班**

**学生学号： 2022111780**

**实验名称： 实验一——数码管扫描**

**实验目的和要求：**

1. 使用数码管显示“HELLO”并实现位移。
2. 使用数码管循环显示数字“8”。

**实验原理以及程序流程：**

**数码管位选、段选**

**While循环显示**

**实验代码及注释：**

1. **循环显示“HELLO”**

#include <REGX52.H>

//数码管段码表

unsigned char NixieTable[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0xFF,0x7F,0x6F};

unsigned char NixieTable1[]={0x76,0x79,0x38,0x3F};//H E L O

////延时子函数

void Delayms(unsigned int xms)

{

unsigned char i, j;

while(xms--)

{

i = 2;

j = 239;

do

{

while (--j);

} while (--i);

}

}

//数码管显示子函数

void Nixie(unsigned char Location,Number)

{

switch(Location)

{

case 8:P2\_4=1;P2\_3=1;P2\_2=1;break;

case 7:P2\_4=1;P2\_3=1;P2\_2=0;break;

case 4:P2\_4=1;P2\_3=0;P2\_2=1;break;

case 3:P2\_4=1;P2\_3=0;P2\_2=0;break;

case 6:P2\_4=0;P2\_3=1;P2\_2=1;break;

case 5:P2\_4=0;P2\_3=1;P2\_2=0;break;

case 2:P2\_4=0;P2\_3=0;P2\_2=1;break;

case 1:P2\_4=0;P2\_3=0;P2\_2=0;break;

}

P0=NixieTable1[Number];

Delayms(1);

P0=0x00;

}

void main()

{

while(1)

{

Nixie(1,0);

Nixie(2,1);

Nixie(3,2);

Nixie(4,2);

Nixie(5,3);

Delayms(150);

Nixie(2,0);

Nixie(3,1);

Nixie(4,2);

Nixie(5,2);

Nixie(6,3);

Delayms(150);

Nixie(3,0);

Nixie(4,1);

Nixie(5,2);

Nixie(6,2);

Nixie(7,3);

Delayms(150);

Nixie(4,0);

Nixie(5,1);

Nixie(6,2);

Nixie(7,2);

Nixie(8,3);

Delayms(150);

Nixie(5,0);

Nixie(6,1);

Nixie(7,2);

Nixie(8,2);

Nixie(1,3);

Delayms(150);

Nixie(6,0);

Nixie(7,1);

Nixie(8,2);

Nixie(1,2);

Nixie(2,3);

Delayms(150);

Nixie(7,0);

Nixie(8,1);

Nixie(1,2);

Nixie(2,2);

Nixie(3,3);

Delayms(150);

Nixie(8,0);

Nixie(1,1);

Nixie(2,2);

Nixie(3,2);

Nixie(4,3);

Delayms(150);

}

}

**2.循环显示“8”**

#include <REGX52.H>

//数码管段码表

unsigned char NixieTable[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0xFF,0x7F,0x6F};

////延时子函数

void Delayms(unsigned int xms)

{

unsigned char i, j;

while(xms--)

{

i = 2;

j = 239;

do

{

while (--j);

} while (--i);

}

}

//数码管显示子函数

void Nixie(unsigned char Location,Number)

{

switch(Location)

{

case 1:P2\_4=1;P2\_3=1;P2\_2=1;break;

case 2:P2\_4=1;P2\_3=1;P2\_2=0;break;

case 3:P2\_4=1;P2\_3=0;P2\_2=1;break;

case 4:P2\_4=1;P2\_3=0;P2\_2=0;break;

case 5:P2\_4=0;P2\_3=1;P2\_2=1;break;

case 6:P2\_4=0;P2\_3=1;P2\_2=0;break;

case 7:P2\_4=0;P2\_3=0;P2\_2=1;break;

case 8:P2\_4=0;P2\_3=0;P2\_2=0;break;

}

P0=NixieTable[Number];

Delayms(1);

P0=0x00;

}

void main()

{

while(1)

{

Nixie(1,4); //在数码管的第1位置显示1

Delayms(1000);

Nixie(2,4);

Delayms(1000);

Nixie(3,4);

Delayms(1000);

Nixie(4,4);

Delayms(1000);

Nixie(5,4);

Delayms(1000);

Nixie(6,4);

Delayms(1000);

Nixie(7,4);

Delayms(1000);

Nixie(8,4);

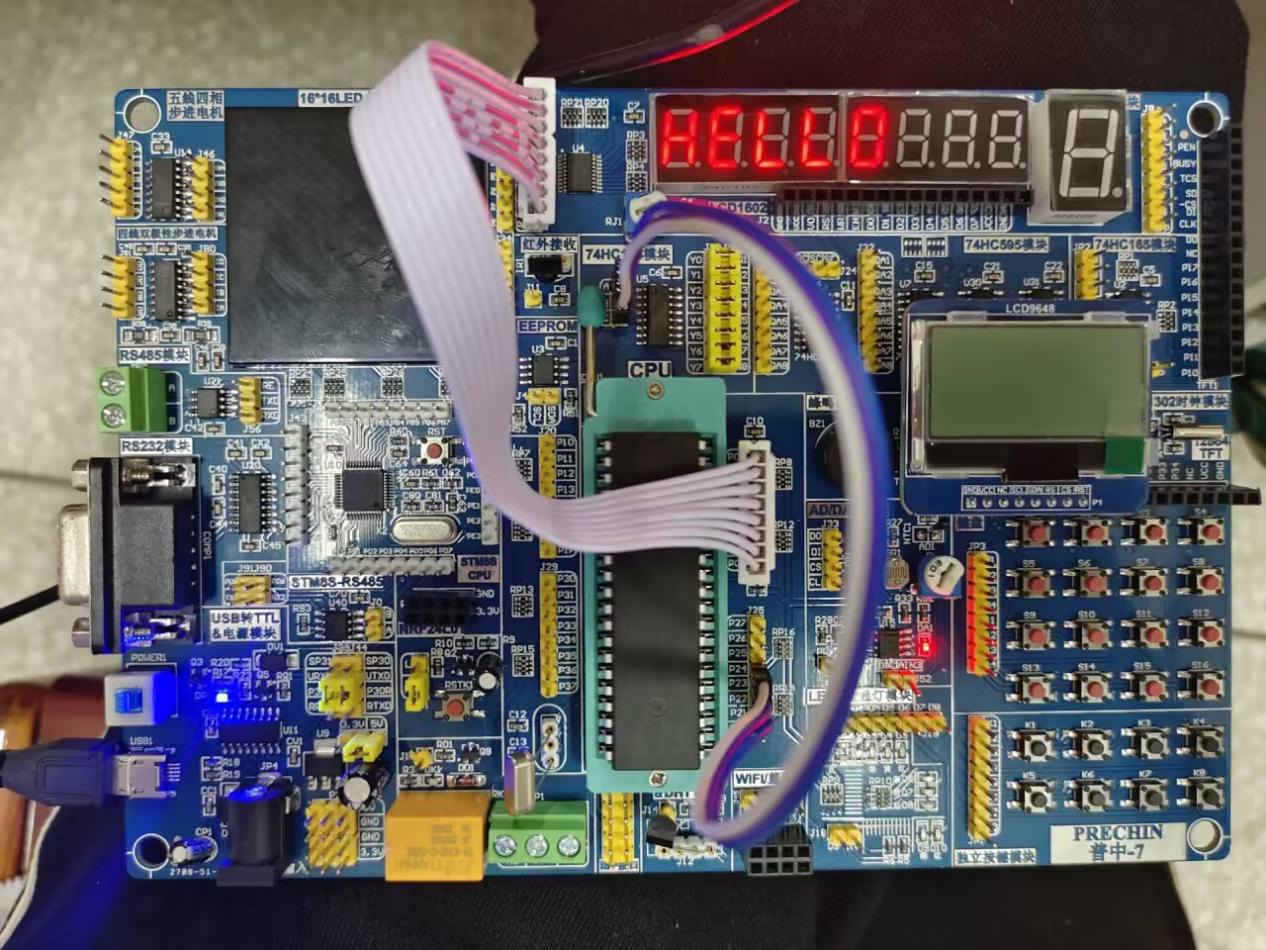
Delayms(1000);

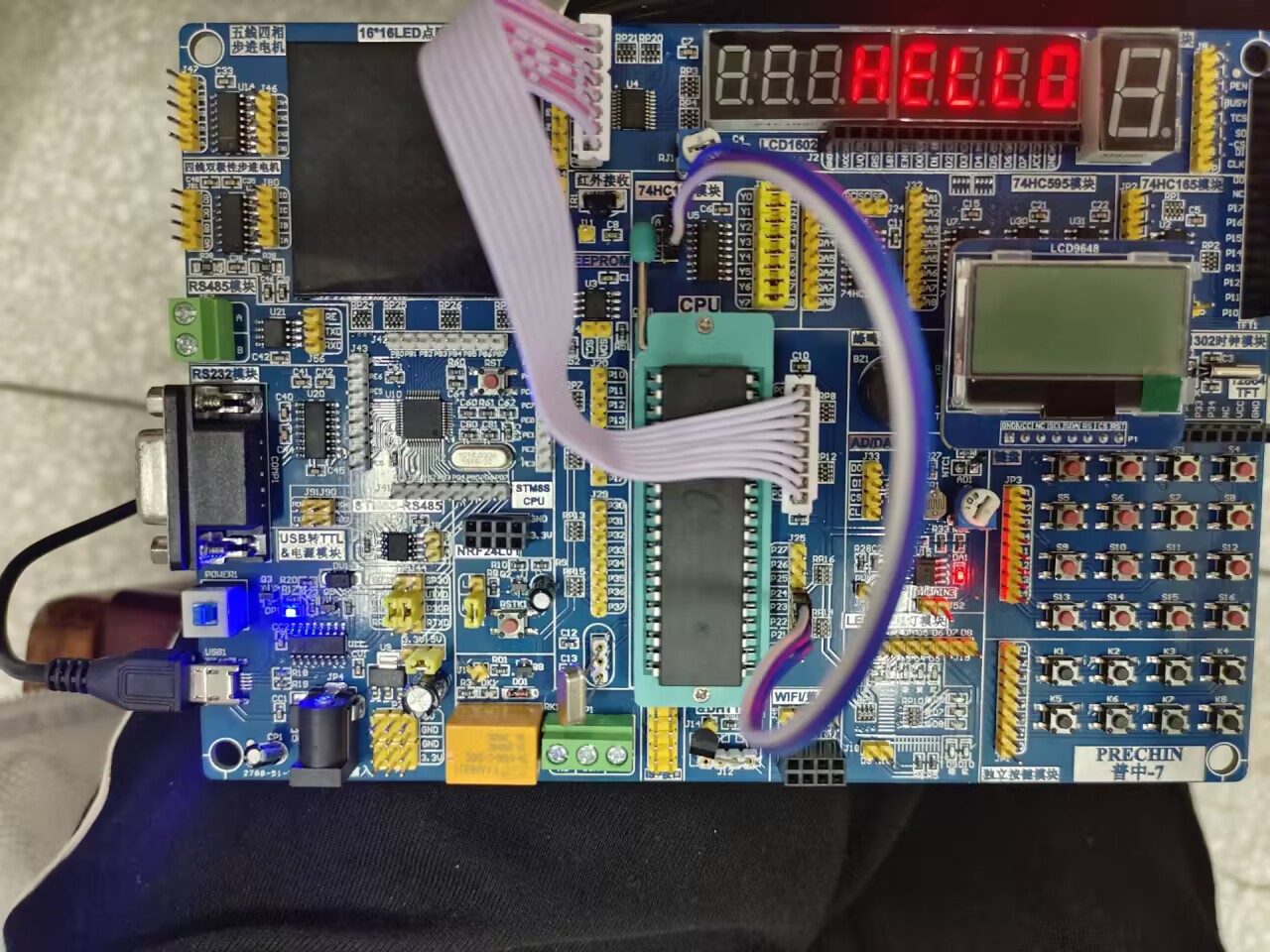
}

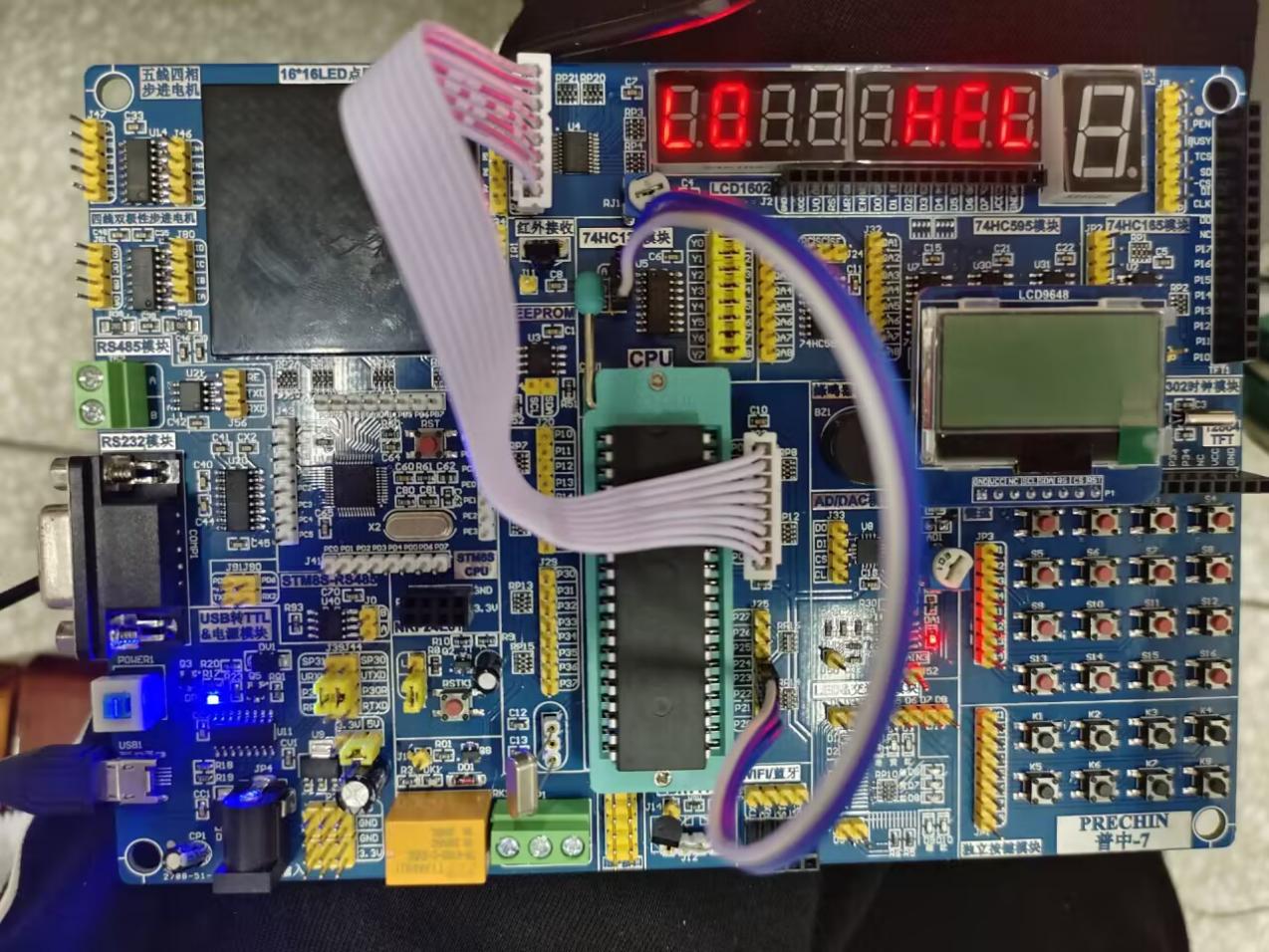
}

**实验现象描述：**

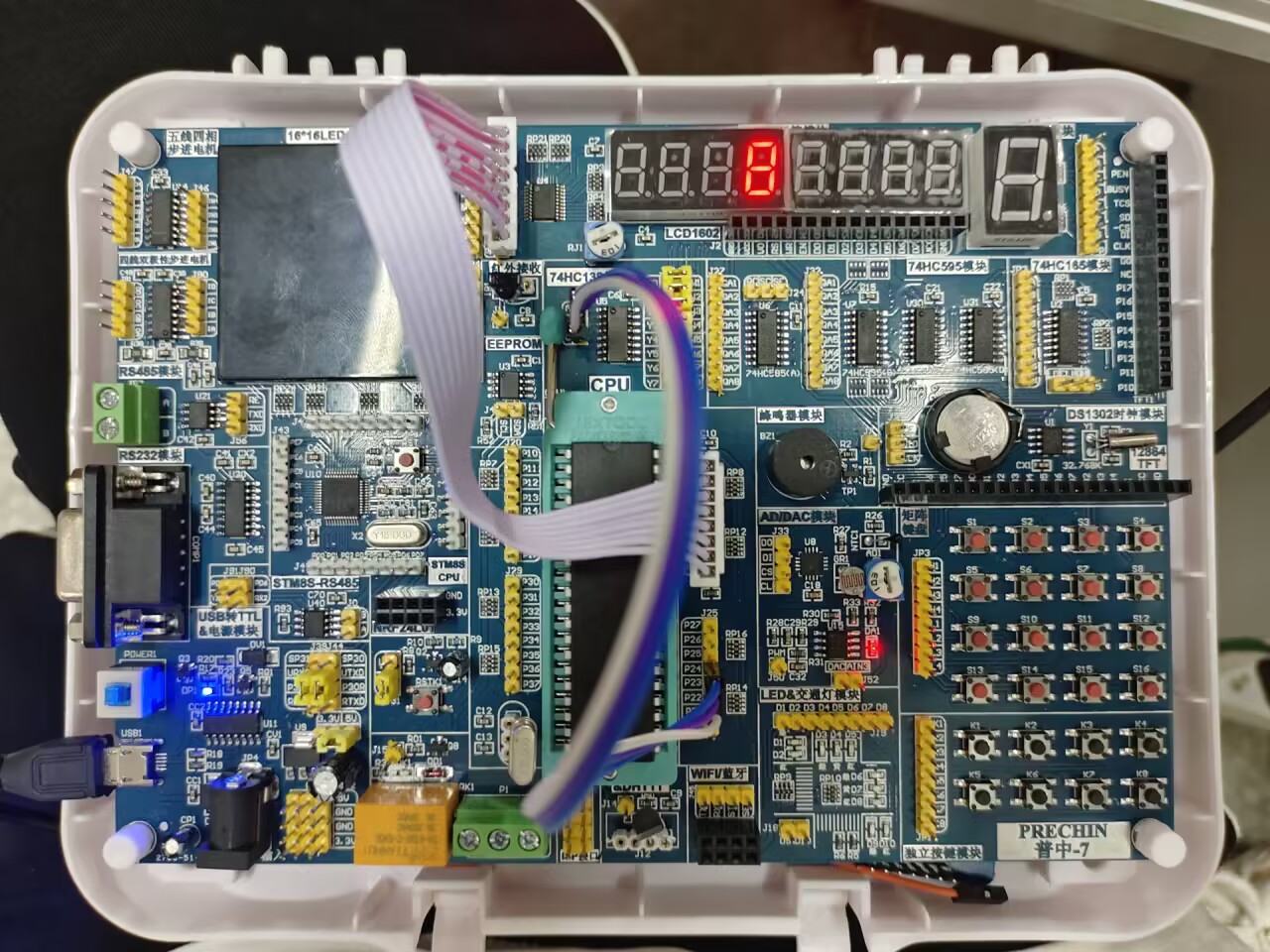
1. **循环移位HELLO**

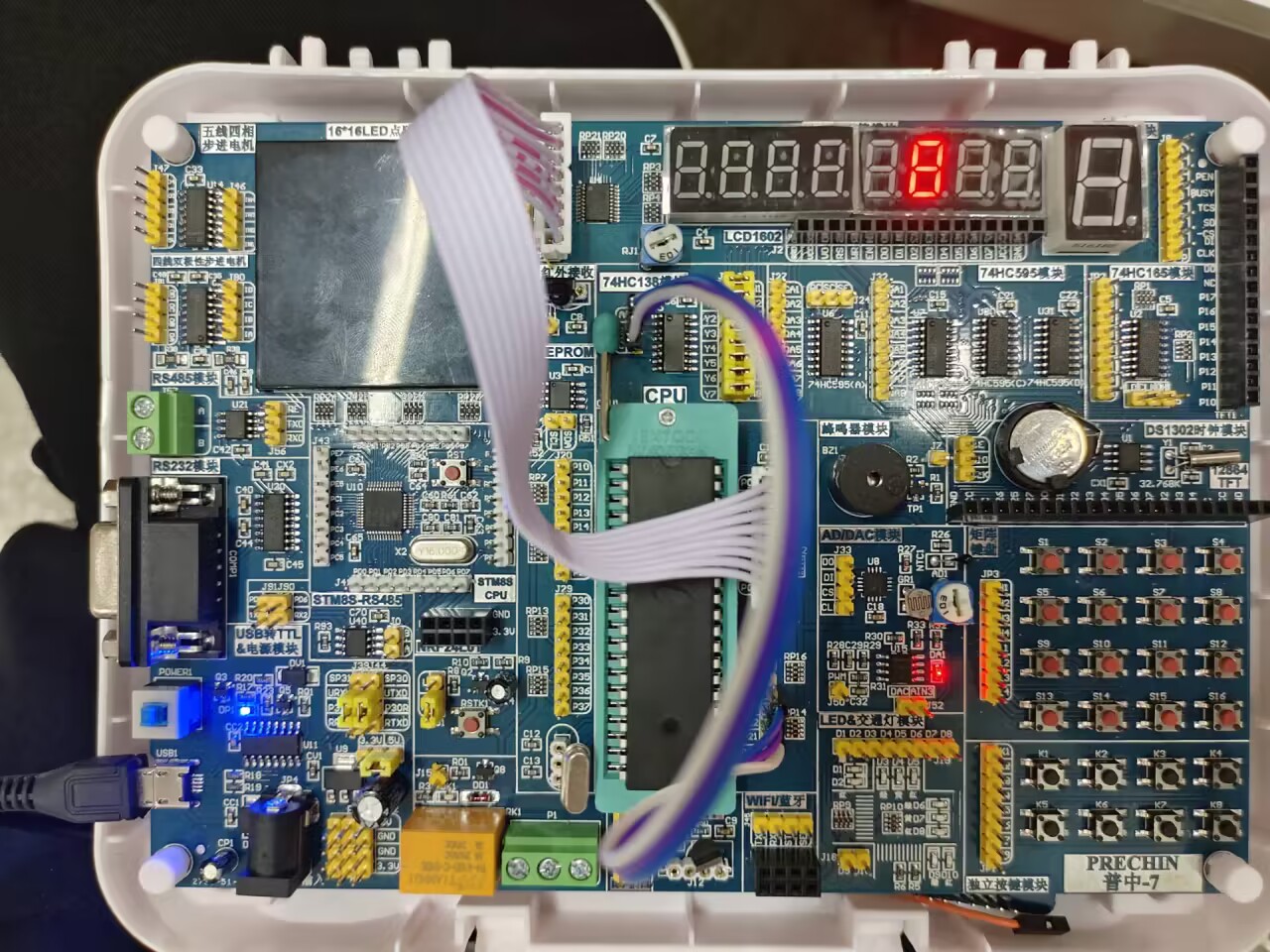


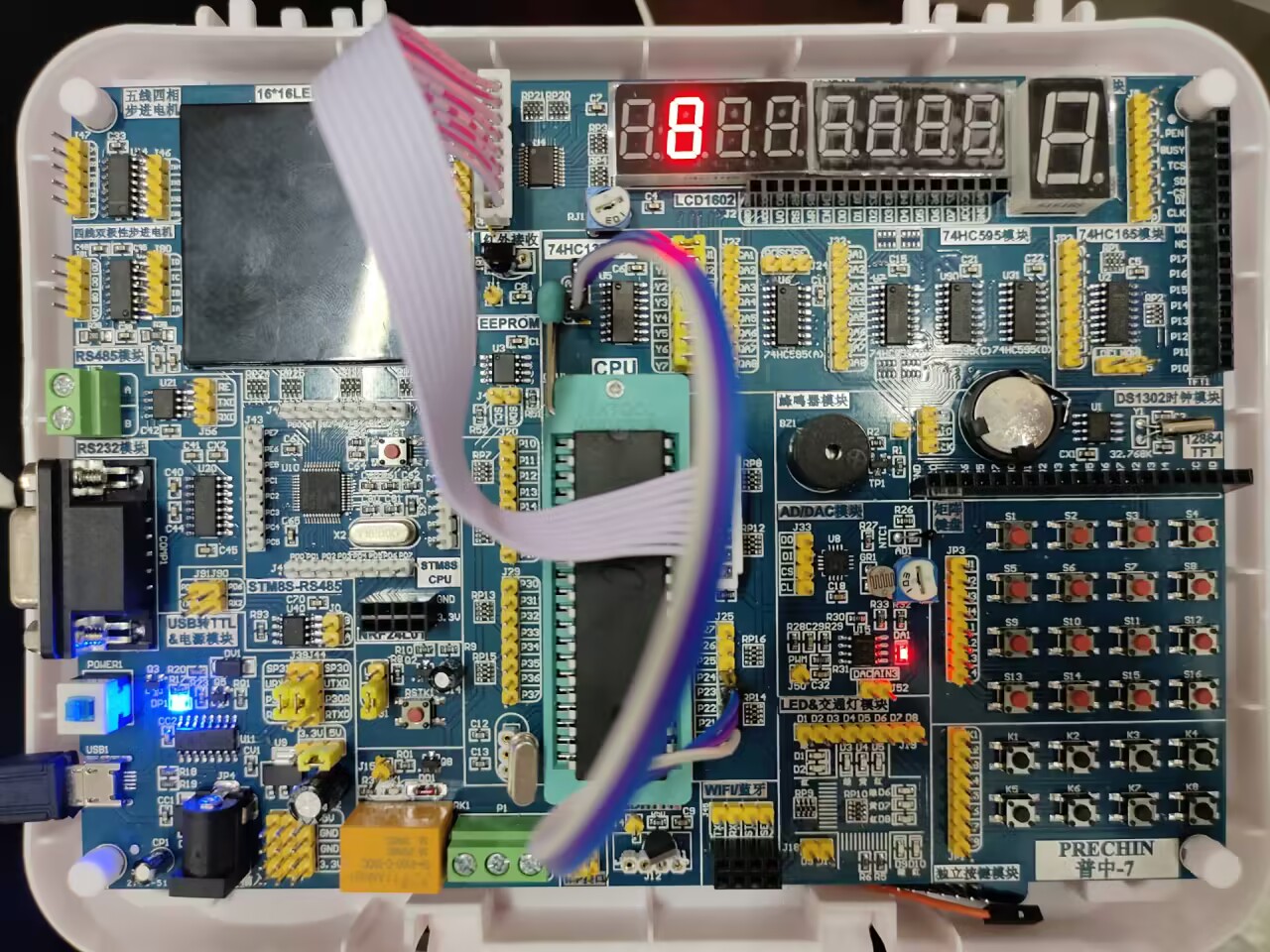




1. **循环移位8**







**问题思考及实验收获感想：**

**1.问题及解决方法：**

在第一次接触51单片机并尝试实现数码管循环显示“hello”（虽然数码管通常用于显示数字或少量字符，这里可能是通过编码或特定硬件支持来模拟显示）和循环显示数字“8”的实验中，我遇到了几个主要问题：

1. 编程逻辑理解困难：最初对如何控制单片机的I/O口以及如何通过编程实现时序控制感到困惑。解决方法是反复阅读教材，观看教学视频，并尝试编写简单的LED闪烁程序作为起点，逐步理解控制逻辑。
2. 数码管编码问题：由于数码管需要通过特定的编码来显示不同的字符或数字，如何正确设置这些编码成为了一个挑战。我通过查阅数据手册，了解每个字符对应的段码，并编写了一个函数来根据输入字符返回相应的编码。
3. 显示切换的流畅性：在循环显示“hello”和“8”时，如何控制显示切换的速度和稳定性也是一个问题。通过调整延时函数中的时间参数，我逐渐找到了最佳的显示效果。

**实验名称： 实验二——任意多路矩形波发生器**

**实验目的和要求：**

1.掌握定时器的多个寄存器功能，作用，设置方法。

2.掌握精确调整矩形波占空比和周期的方法。

3.分析不同定时模式的特性与适用场合。

**实验原理以及程序流程：**

以实现占空比为10%为例

**设置定时器0的定时时间**

**T0Count%10==0为高电平，其余时间为低电平。**

**实验代码及注释：**

#include <reg51.h>

sbit buzzer = P1^0;

void Timer0Init(void) //400微秒@11.0592MHz

{

TMOD &= 0xF0; //设置定时器模式

TMOD |= 0x01; //设置定时器模式

TL0 = 0x8F; //设置定时初值

TH0 = 0xFE; //设置定时初值

TF0 = 0; //清除TF0标志

TR0 = 1; //定时器0开始计时

ET0=1;

EA=1;

PT0=0;

}

void main() {

Timer0Init();

while (1) {

}

}

int T0Count=0;

void Timer0\_ISR() interrupt 1 {

T0Count++;

TL0 = 0x8F; //设置定时初值

TH0 = 0xFE; //设置定时初值

if(T0Count%10==0){

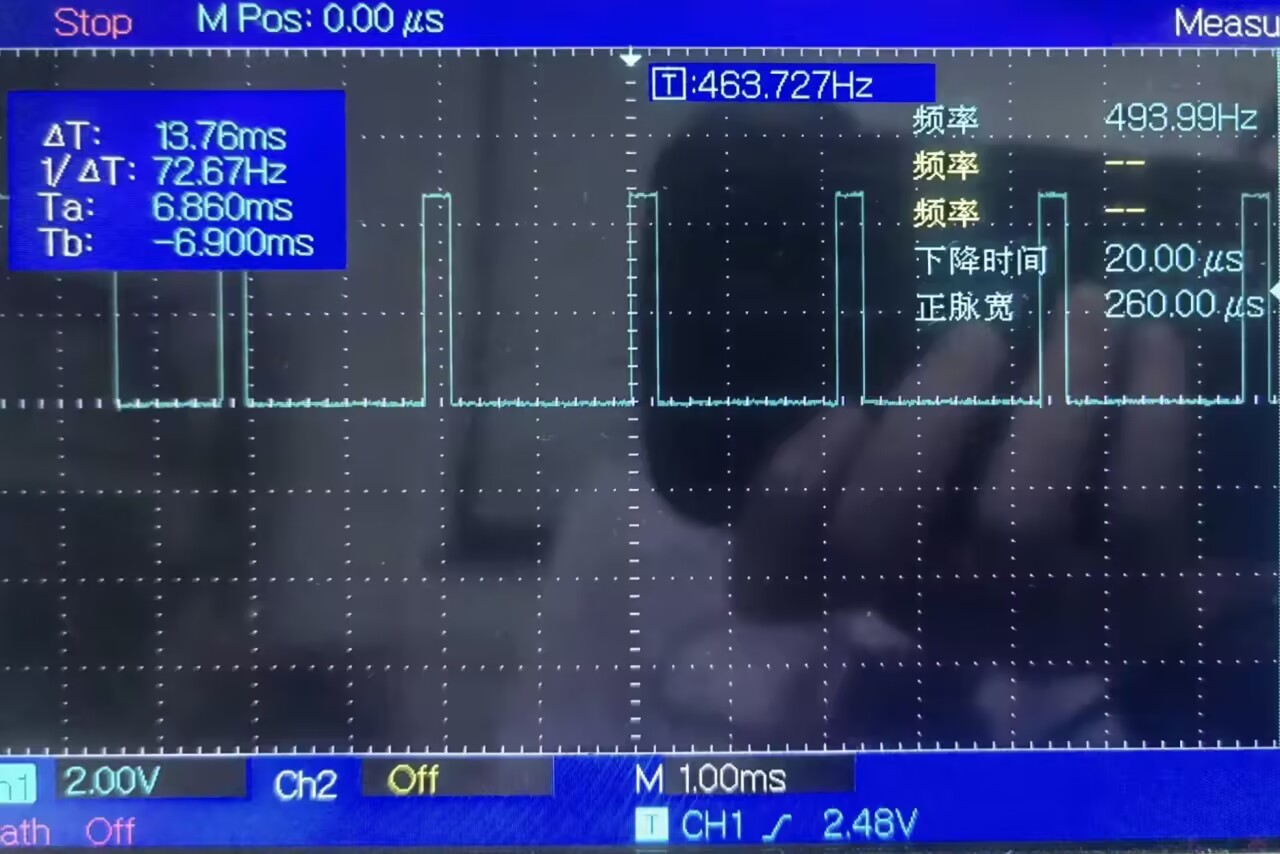
buzzer = 1;

}else buzzer = 0;

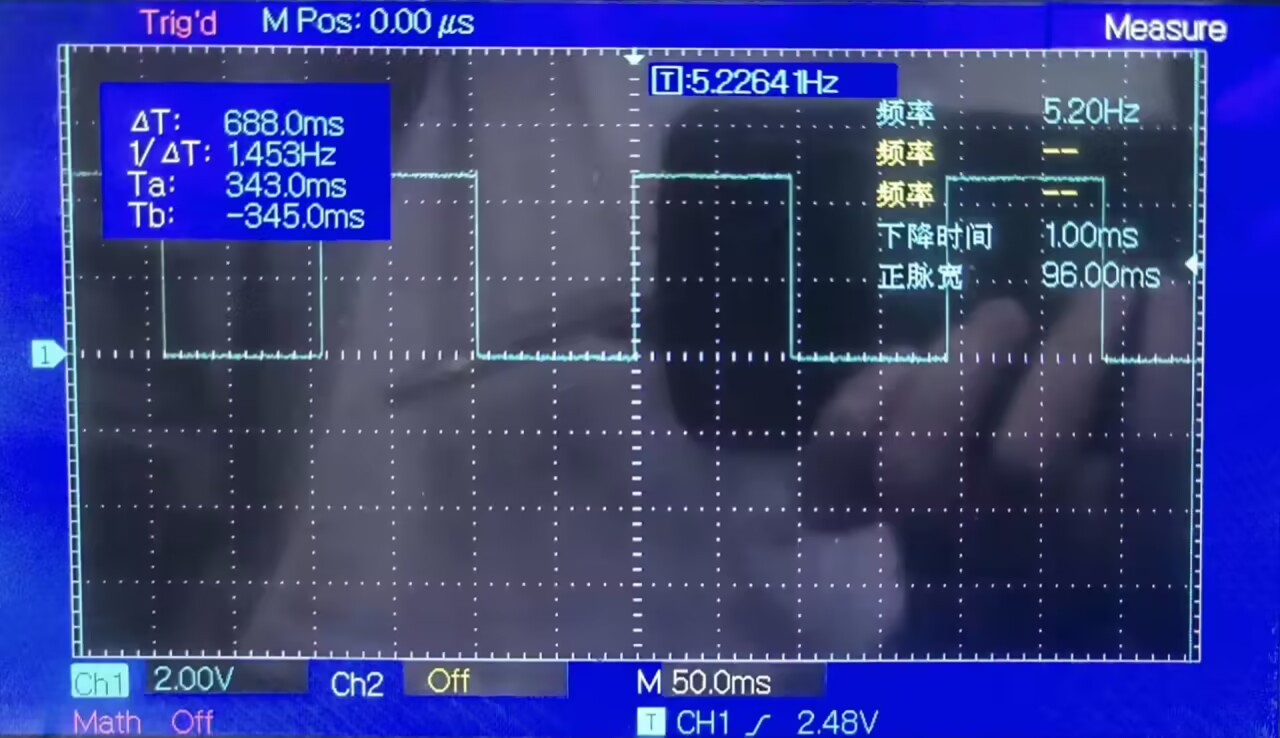
}

**实验现象描述：**

1. **占空比为10%**



1. **5Hz方波**



1. **50Hz方波**



**问题思考及实验收获感想：**

**1.问题及解决方案：**

定时器寄存器配置复杂：初始时，对定时器T0和T1的多个寄存器（如TH0、TL0、TMOD等）的功能和设置方法感到困惑。为了解决这个问题，我首先仔细研读了实验手册，通过对比不同寄存器的位定义和功能描述，逐步理解了它们的作用。随后，通过编写简单的测试程序，逐步调试和验证每个寄存器的配置效果，直至完全掌握其使用方法。

精确调整矩形波占空比和周期：在尝试生成特定占空比和周期的矩形波时，发现实际输出与预期存在偏差。经过分析，发现是定时器的重载值计算不够精确导致的。通过进一步学习定时器的工作原理和定时周期的计算公式，我重新计算了合适的重载值，并调整了程序中的相关参数，最终实现了精确控制矩形波的占空比和周期。

1. **实验结果分析及结论：**

通过本次实验，我成功掌握了51单片机定时器的多个寄存器功能、作用及设置方法，能够灵活运用这些寄存器来实现定时控制任务。同时，我也学会了如何精确调整矩形波的占空比和周期，这对于后续的信号处理、电机控制等应用具有重要意义。

**3. 现代工具和手段**

我还利用了示波器等电子测量仪器来观察和分析实验现象，为实验结果的分析提供了有力的支持。此外，我还通过互联网搜索相关资料和教程，学习他人的经验和技巧，不断拓宽自己的知识面和视野。

**实验名称： 实验三——方波产生及频率的测量、数码管显示**

**实验目的和要求：**

用到两个定时器，一个用来产生方波，产生方波的输出端口接入单片机的外部中断引脚，控制启动计数器计数，方波的下降沿再次触发外部中断时，停止计数，通过计算，算出频率，并用数码管动态输出频率。

**实验原理以及程序流程：**

**初始化外部中断0，设置为下降沿触发，启用外部中断0**

**初始化定时器0，实现1ms定时，启用定时器0中断**

**P1\_0电平翻转，输出波形**

**接收下降沿并计数**

**在定时器中，每经过1s清零**

**数码管显示**

**实验代码及注释：**

#include <REGX51.H>

unsigned int count = 0;

unsigned int number = 0;

unsigned char NixieTable[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

void Timer0\_Init(void) //1毫秒@12.000MHz

{

TMOD &= 0xF0; //设置定时器模式

TMOD |= 0x01; //设置定时器模式

TL0 = 0x18; //设置定时初始值

TH0 = 0xFC; //设置定时初始值

TF0 = 0; //清除TF0标志

TR0 = 1; //定时器0开始计时

ET0 = 1; //使能定时器0中断

EA = 1;

}

void Int0\_Init(){//外部中断P3\_2

IT0 = 1; // 边沿触发方式（下降沿)

EX0 = 1;

EA = 1;

}

void Delay(unsigned int xms)

{

unsigned char i, j;

while(xms--)

{

i = 2;

j = 239;

do

{

while (--j);

} while (--i);

}

}

//数码管显示子函数

void Nixie(unsigned char Location,Number)

{

switch(Location){

case 1:P2\_4=0;P2\_3=0;P2\_2=0;break;

case 2:P2\_4=0;P2\_3=0;P2\_2=1;break;

case 3:P2\_4=0;P2\_3=1;P2\_2=0;break;

case 4:P2\_4=0;P2\_3=1;P2\_2=1;break;

case 5:P2\_4=1;P2\_3=0;P2\_2=0;break;

case 6:P2\_4=1;P2\_3=0;P2\_2=1;break;

case 7:P2\_4=1;P2\_3=1;P2\_2=0;break;

case 8:P2\_4=1;P2\_3=1;P2\_2=1;break;

}

P0=NixieTable[Number]; //段码输出

Delay(1); //显示一段时间

P0=0x00; //段码清0，消影

}

void main(){

Timer0\_Init();

Int0\_Init();

while(1){//显示测得频率

Nixie(1,number /100);

Nixie(2,number %100/10);

Nixie(3,number %10);

}

}

void Timer0\_Isr(void) interrupt 1

{

static unsigned int count\_1 = 0;

TL0 = 0x18; //设置定时初始值

TH0 = 0xFC; //设置定时初始值

P1\_0=~P1\_0;

count\_1++;

if(count\_1>=1000){

number = count;

count = 0;

count\_1=0;

}

}

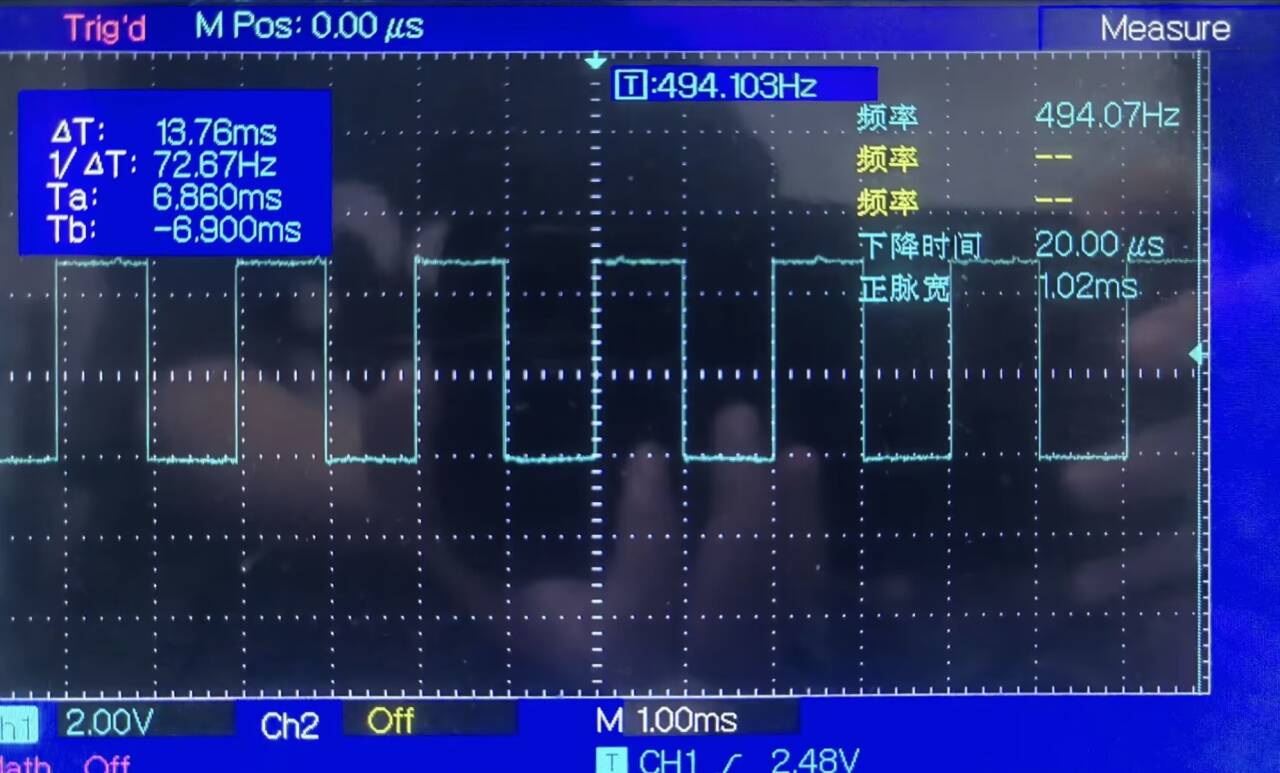
void INTCounts() interrupt 0{//接收到下降沿count++

count++;

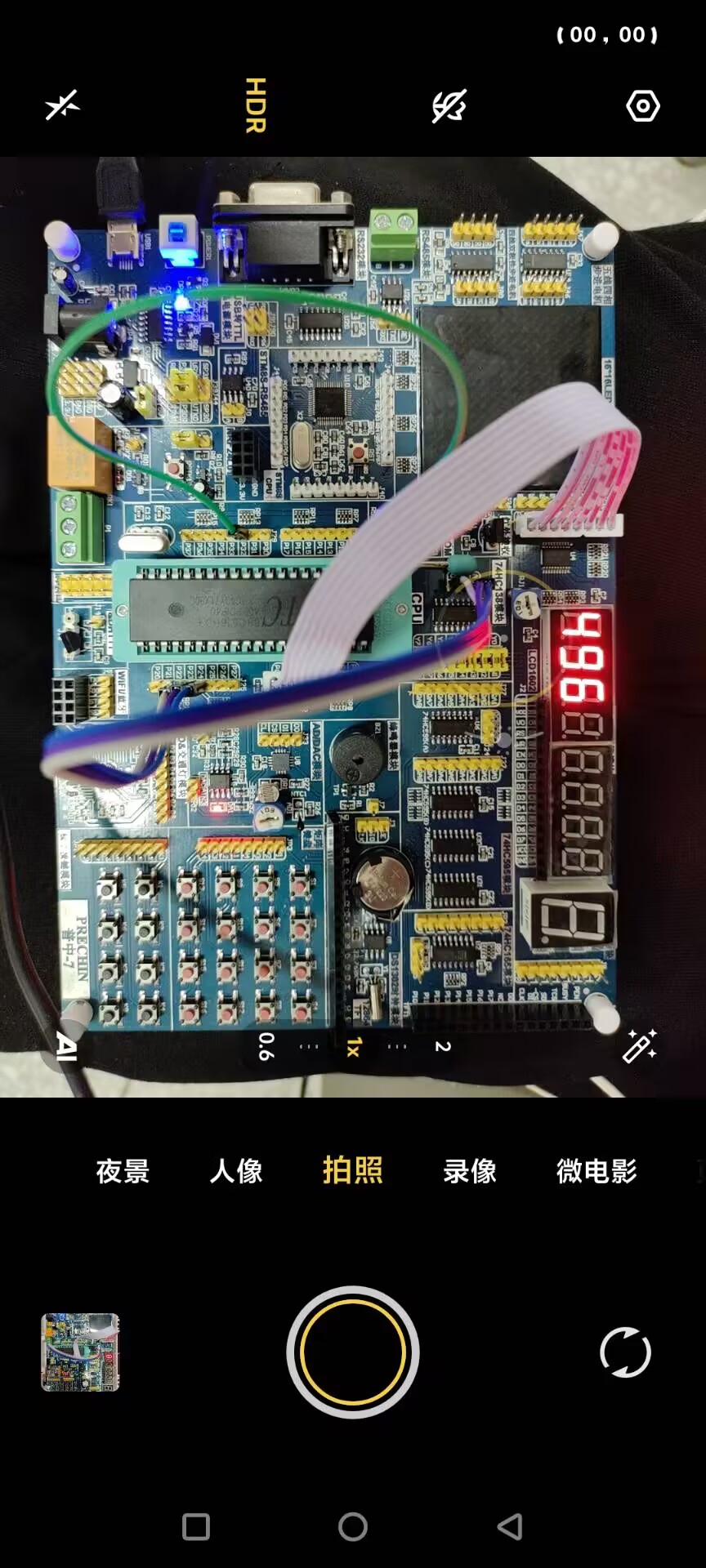
}

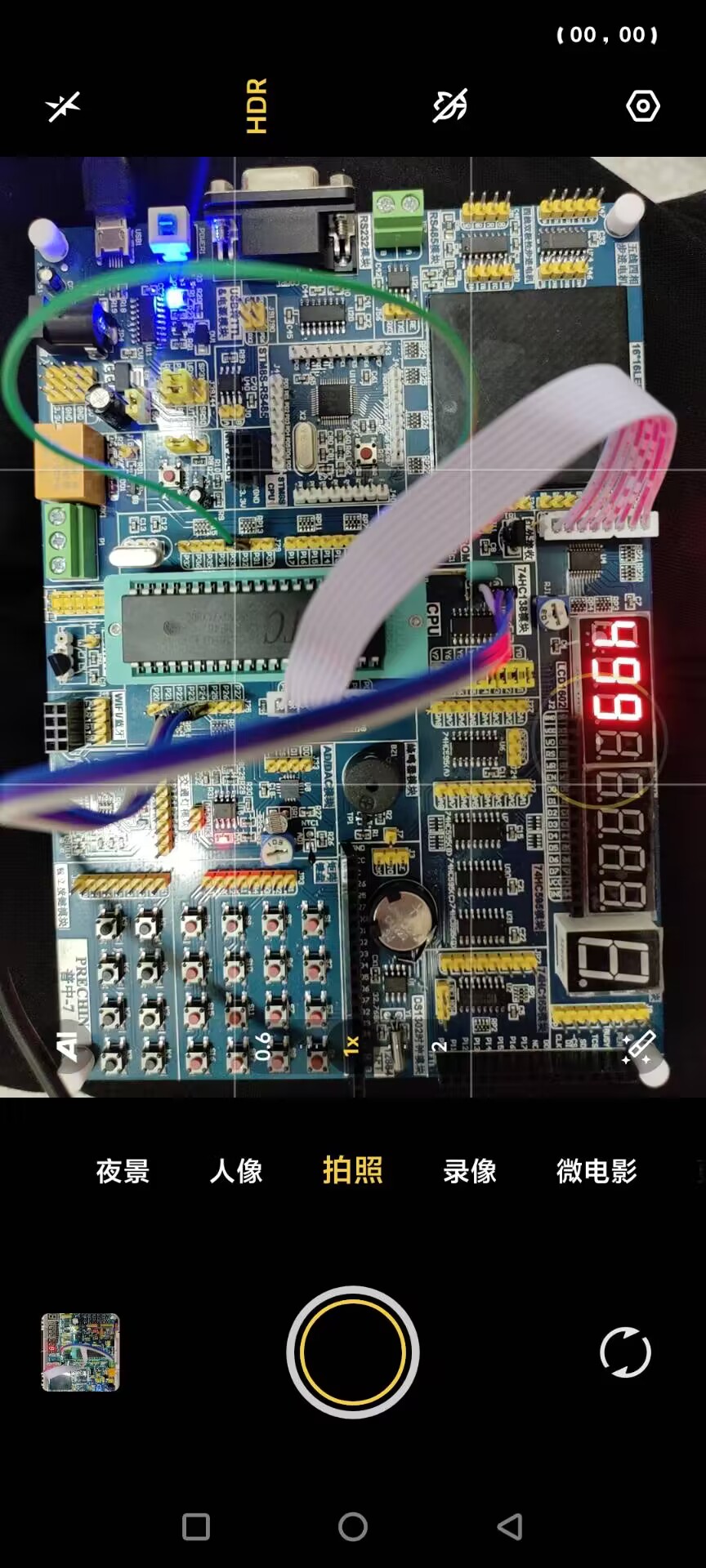
**实验现象描述：**

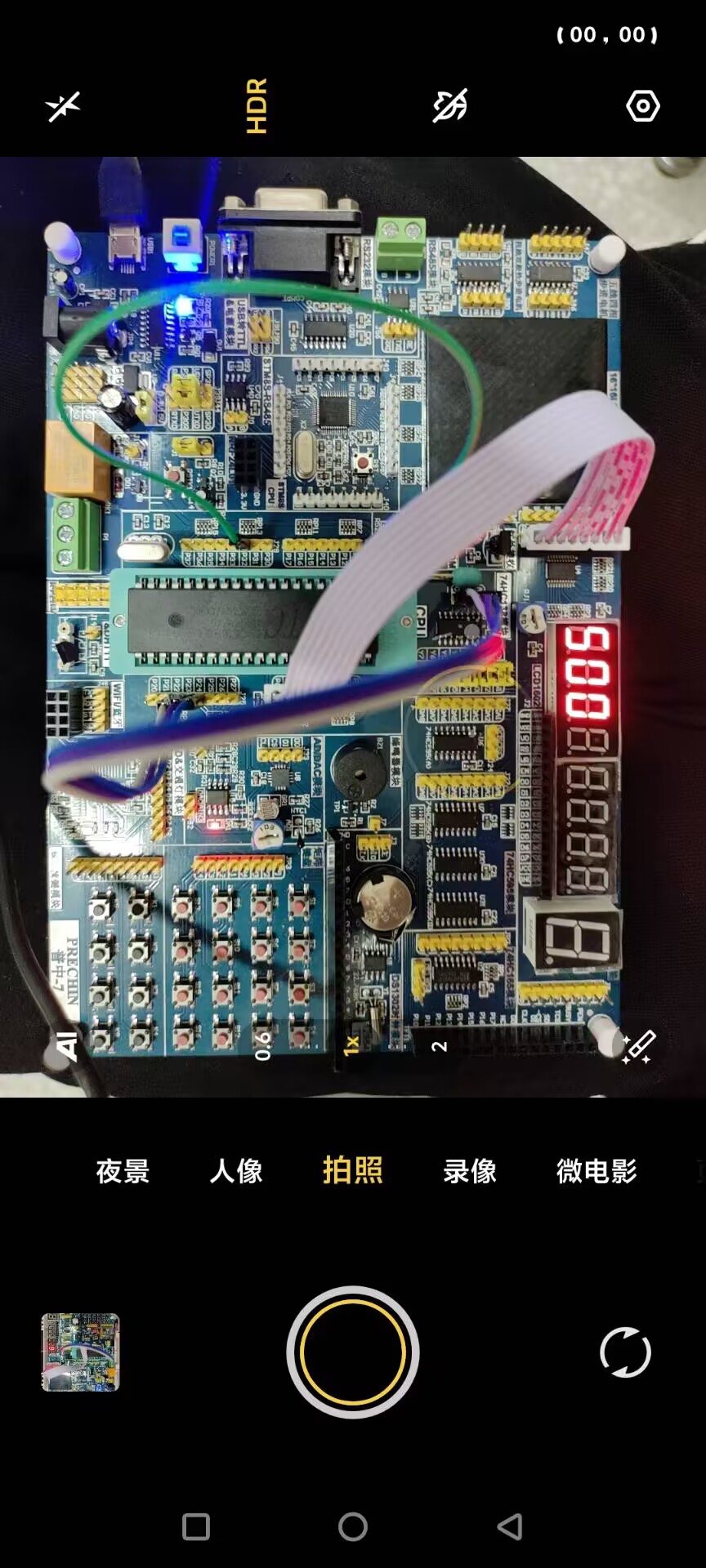
1. **示波器测量**



1. **数码管显示**







**问题思考及实验收获感想：**

1. **遇到的问题及解决方案**

外部中断的触发与计数器的控制：将方波输出接入外部中断引脚后，如何准确地在方波的下降沿触发中断，并在再次下降沿时停止计数是一个挑战。我通过精确配置中断控制位（IT0选择下降沿触发），并在中断服务程序中合理控制计数器的启停，成功实现了对方波周期的精确测量。

1. **实验结果的分析与解释：**

实验结果表明，该系统能够稳定地产生指定频率的方波信号，并通过外部中断和计数器精确测量方波的周期，进而计算出频率。数码管显示的数据准确且更新迅速，没有出现明显的闪烁或延迟现象。这充分证明了定时器配置的正确性、中断服务的可靠性以及频率计算与显示算法的有效性。

1. **现代工具和手段的应用：**

我还利用了示波器等电子测量仪器来观察和分析方波信号的特性，为频率测量提供了有力的支持。

**实验名称： 实验四——串行通信**

**实验目的和要求：**

1. 操作配置寄存器，设置波特率4800，数据位8位，一位停止位，无校验位。

2. 电脑使用串口调试助手，接收单片机通过串口发送的数据。

3.通过编写简单函数y=ax+b进行信息简单加密发送，通过x=(y-b)/a进行解密，a,b 为正整数系数，实现开发板与单片机之间的加密通信。

**实验原理以及程序流程：**

**初始化串口1，波特率设置为4800**

**串口中断中，接收满两个字节数据后才进行解密为1个字节数据**

**将该解密后的1字节数据发送给上位机**

**实验代码及注释：**

#include <REGX52.H>

#include "intrins.h"

#include "stdio.h"

unsigned int count = 0;

#define BUF\_SIZE 2 // 缓冲区大小，用于存储接收到的数据

unsigned char rx\_buffer[BUF\_SIZE]; // 接收缓冲区

unsigned char rx\_index = 0; // 接收缓冲区索引

void Uart1\_Init(void) //4800bps@12.000MHz

{

PCON |= 0x80; //使能波特率倍速位SMOD

SCON = 0x50; //8位数据,可变波特率

TMOD &= 0x0F; //设置定时器模式

TMOD |= 0x20; //设置定时器模式

TL1 = 0xF3; //设置定时初始值

TH1 = 0xF3; //设置定时重载值

ET1 = 0; //禁止定时器中断

TR1 = 1; //定时器1开始计时

EA = 1;

ES = 1;

}

void UART\_SendByte(unsigned char Byte){

SBUF=Byte;

while(TI==0);

TI=0;

}

void Uart\_Isr() interrupt 4 using 1{

if (RI) { // 如果接收到数据

RI = 0; // 清除接收中断标志位

rx\_buffer[rx\_index++] = SBUF; // 将接收到的数据存储到缓冲区中

if (rx\_index == BUF\_SIZE) { // 如果接收到了足够的字节

count=(rx\_buffer[0]<<8)+rx\_buffer[1];//

count = (count-2)/3;

UART\_SendByte(count);

rx\_index = 0; // 重置缓冲区索引，准备接收下一组数据

}

}

}

void main()

{

Uart1\_Init();

while(1)

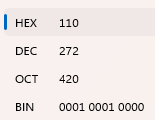
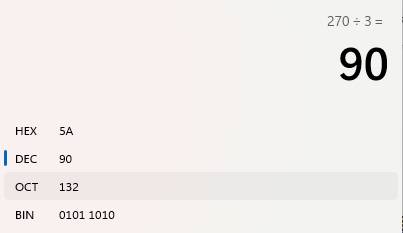
{

}

}

**实验现象描述：**

**注：包含能反映实验现象和结果的贴图，并对结果做合理解释。**



解释：电脑发送两个字节数据0110，经过16进制转10进制后为272。在下位机进行解密：X=（272-2）/3=90。90再经过10进制转16进制，得到0x5a，单片机再将该数据通过串口发送给上位机。

**问题思考及实验收获感想：**

1. **遇到的问题及解决方案：**

在实现数据加密发送和解密接收时，初步代码运行时存在计算结果错误的情况。经过仔细检查，发现是数据类型溢出和除法运算精度问题。让单片机在接收完两个字节之后在及进行解算。扩大了输入范围。

1. **实验结果分析与结论**

实验结果充分验证了通过配置51单片机的串行通信寄存器，可以实现与PC之间稳定的数据交换。通过自定义的加密解密函数（y=3x+2 和 x=(y-2)/3），成功实现了数据的简单加密传输和解密接收，证明了这种简单的数学方法能够用于提升通信过程中的数据安全性。PC端输入两个字节数据0110，换算成10进制数据是272，经过公式x=(y-2)/3解密得到x=90，16进制表示为5A，与实验结果相符。

1. **现代工具和手段的应用**

在解决实验过程中遇到的问题时，我充分利用了串口调试助手这一现代工具，通过实时查看数据发送和接收情况。

**实验名称： 实验五——多路模拟信号的AD转换**

**实验目的和要求：**

多路模拟信号的AD转换，并用四位数码管在多路之间切换显示，含两位小数。

**实验原理以及程序流程：**

**初始化SPI接口**

**初始化定时器0，实现每**

**1s切换对象读取AD值**

**每1s内，单片机通过SPI通信，发送不同地址，切换读取对象**

**读取的AD值，通过数码管刷新显示**

**实验代码及注释：**

1. **XPT2046.c**

#include <REGX52.H>

#include <INTRINS.H>

#include "XPT2046.h"

//引脚定义

sbit XPY2046\_DIN=P3^4;

sbit XPY2046\_CS=P3^5;

sbit XPY2046\_DCLK=P3^6;

sbit XPY2046\_DOUT=P3^7;

/\*\*

\* @brief ZPT2046读取AD值

\* @param Command 命令字，范围：头文件内定义的宏，结尾的数字表示转换的位数

\* @retval AD转换后的数字量，范围：8位为0~255，12位为0~4095

\*/

unsigned int XPT2046\_ReadAD(unsigned char Command)

{

unsigned char i;

unsigned int Data=0;

XPY2046\_DCLK=0;

XPY2046\_CS=0;

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

{

XPY2046\_DIN=Command&(0x80>>i);

XPY2046\_DCLK=1;

XPY2046\_DCLK=0;

}

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

{

XPY2046\_DCLK=1;

XPY2046\_DCLK=0;

if(XPY2046\_DOUT){Data|=(0x8000>>i);}

}

XPY2046\_CS=1;

return Data>>4;

}

1. **Main.c**

#include "delay.h"

#include "seg.h"

#include "XPT2046.h"

#include "Nixie.h"

unsigned int ADValue;

bit flag1=0;

bit flag2=0;

bit flag3=0;

void Timer0\_Init(void) //1毫秒@12.000MHz

{

TMOD &= 0xF0; //设置定时器模式

TMOD |= 0x01; //设置定时器模式

TL0 = 0x18; //设置定时初始值

TH0 = 0xFC; //设置定时初始值

TF0 = 0; //清除TF0标志

TR0 = 1; //定时器0开始计时

ET0 = 1; //使能定时器0中断

EA = 1;

}

void main(void)

{

Timer0\_Init();

while(1)

{

if(flag1 ==1){

ADValue=XPT2046\_ReadAD(XPT2046\_XP); //读取AIN0，可调电阻

Nixie\_SetBuf(4,ADValue/1000%10);

Nixie\_SetBuf(3,(ADValue%1000/100)+12);

Nixie\_SetBuf(2,ADValue/10%10);

Nixie\_SetBuf(1,ADValue%10);

Nixie\_SetBuf(5,10);

Nixie\_SetBuf(6,10);

Nixie\_SetBuf(7,10);

Nixie\_SetBuf(8,1);

flag1 =0;

}

if(flag2 ==1){

ADValue=XPT2046\_ReadAD(XPT2046\_YP); //读取AIN1，温度

Nixie\_SetBuf(4,ADValue/1000);

Nixie\_SetBuf(3,(ADValue%1000/100)+12);

Nixie\_SetBuf(2,ADValue%100/10);

Nixie\_SetBuf(1,ADValue%10);

Nixie\_SetBuf(5,10);

Nixie\_SetBuf(6,10);

Nixie\_SetBuf(7,10);

Nixie\_SetBuf(8,2);

flag2 =0;

}

if(flag3 ==1){

ADValue=XPT2046\_ReadAD(XPT2046\_VBAT); //读取AIN2，光敏电阻

Nixie\_SetBuf(4,ADValue/1000);

Nixie\_SetBuf(3,(ADValue%1000/100)+12);

Nixie\_SetBuf(2,ADValue%100/10);

Nixie\_SetBuf(1,ADValue%10);

Nixie\_SetBuf(5,10);

Nixie\_SetBuf(6,10);

Nixie\_SetBuf(7,10);

Nixie\_SetBuf(8,3);

flag3 =0;

}

Delay(100);

}

}

void Timer0\_Isr(void) interrupt 1

{

static unsigned int T0Count1,T0Count2,T0Count3;

TL0 = 0x18; //设置定时初始值

TH0 = 0xFC; //设置定时初始值

T0Count3++;

if(T0Count3>=1000){

T0Count1++;

T0Count3 = 0;

}

if(T0Count1<=10)

{

flag1 = 1;

}

else if(T0Count1>10&&T0Count1<=20){

flag2 = 1;

}

else if(T0Count1>20&&T0Count1<=30){

flag3 = 1;

}

else if(T0Count1>30){

T0Count1=0;

}

T0Count2++;

if(T0Count2>=2)

{

T0Count2=0;

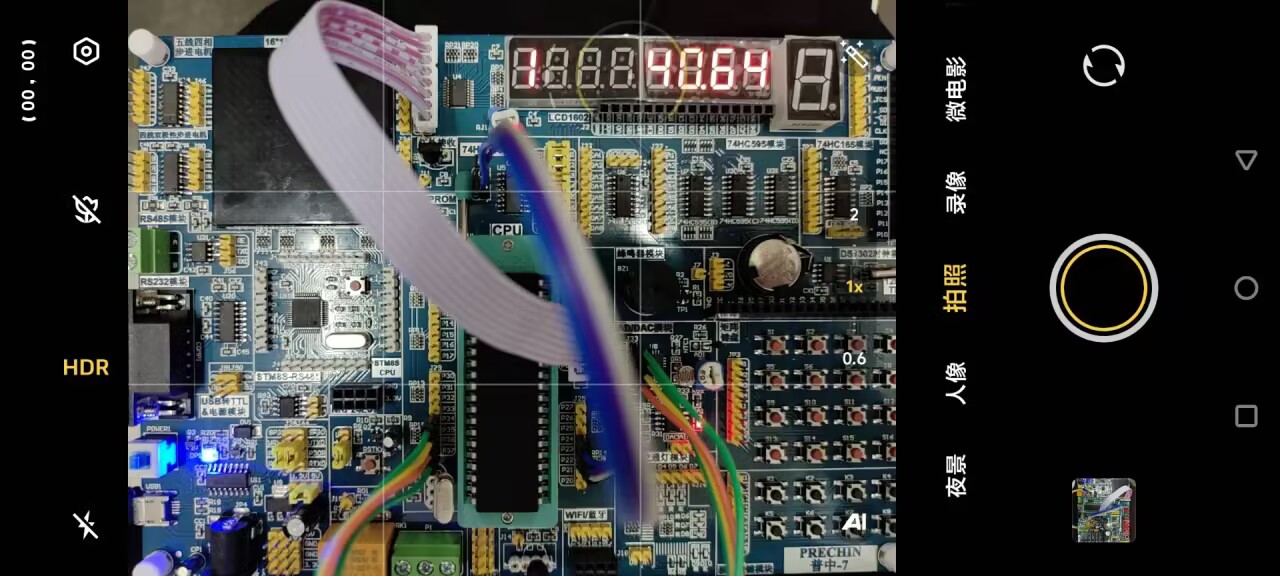
Nixie\_Loop();//2ms调用一次数码管驱动函数

}

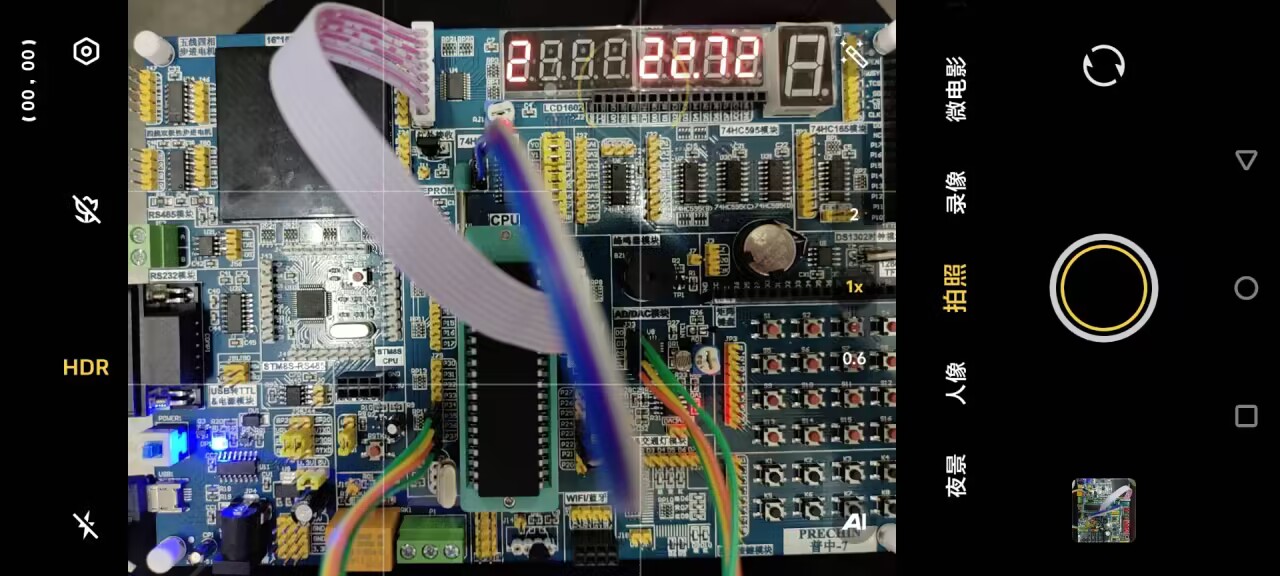
}

**实验现象描述：**

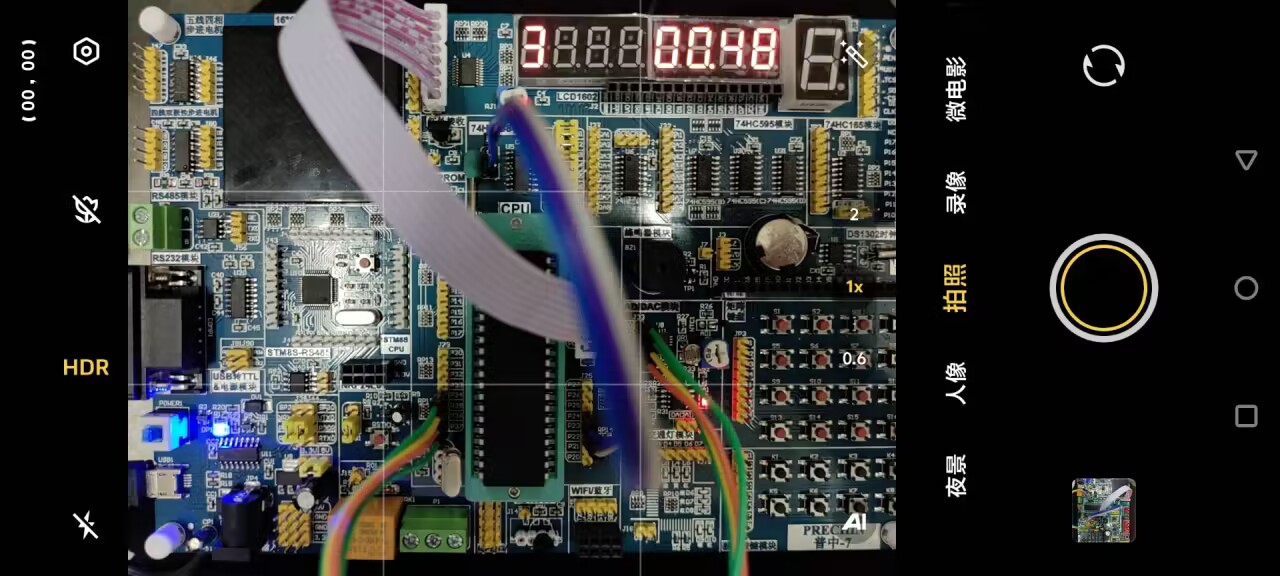
1.读取可调电阻



2.读取温度



3.读取光敏电阻



**问题思考及实验收获感想：**

1. **实验结果分析与结论：**

通过对实验结果的分析，我能够清晰地看到各模拟信号经过AD转换后的数字量，并通过四位数码管准确地切换显示这些值，包括两位小数的部分。这证明了软件编程（如AD转换程序、数码管驱动程序）都是有效的。

**2. 实验收获：**  
在该实验中，我学习并掌握了SPI通信的原理与时序逻辑，学习了ADC的基本概念、工作原理及其在现代电子系统中不可或缺的角色，特别是在数据采集与信号处理领域的广泛应用。学会了如何借助SPI通信实现读取AD值。不仅锻炼了我的硬件接口编程能力，也提升了我对数字信号与模拟信号转换过程中误差控制的理解。同时加深了对定时器的理解，熟练地将定时器运用于各种设计之中。

**实验名称： 实验六——PWM实现DA转换**

**实验目的和要求：**

通过PWM实现DA转换，转换的结果送给AD 采样输出数码管显示

**实验原理以及程序流程：**

**初始化SPI接口**

**初始化定时器1，P1\_0产生PWM波形**

**单片机通过SPI通信，发送地址，读取对象**

**数码管刷新显示**

**实验代码及注释：**

1. **XPT2046.c与实验五相同**
2. **Main.c**

#include <REGX52.H>

#include "XPT2046.h"

#include "Nixie.h"

unsigned int ADValue;

unsigned char count = 0;

unsigned char compared = 2;

void Timer1\_Isr(void) interrupt 3

{

// TL1 = 0xFC; //设置定时初始值

// TH1 = 0xFC; //设置定时重载值

count++;

if(count >= 5){//产生占空比20%

count = 0;

}

if(count < compared){

P2\_1 = 1;

}

else{

P2\_1 = 0;

}

}

void Timer1\_Init(void)

{

TMOD &= 0x0F; //设置定时器模式

TMOD |= 0x20; //设置定时器模式

TL1 = 0xFF; //设置定时初始值

TH1 = 0xFF; //设置定时重载值

TF1 = 0; //清除TF1标志

TR1 = 1; //定时器1开始计时

ET1 = 1; //使能定时器1中断

EA = 1;

}

void main(){

Timer1\_Init();

while(1){

ADValue=XPT2046\_ReadAD(XPT2046\_AUX);

// Nixie\_SetBuf(4,ADValue/1000%10);

// Nixie\_SetBuf(3,ADValue/100%10);

// Nixie\_SetBuf(2,ADValue/10%10);

// Nixie\_SetBuf(1,ADValue%10);

// Nixie\_SetBuf(5,10);

// Nixie\_SetBuf(6,10);

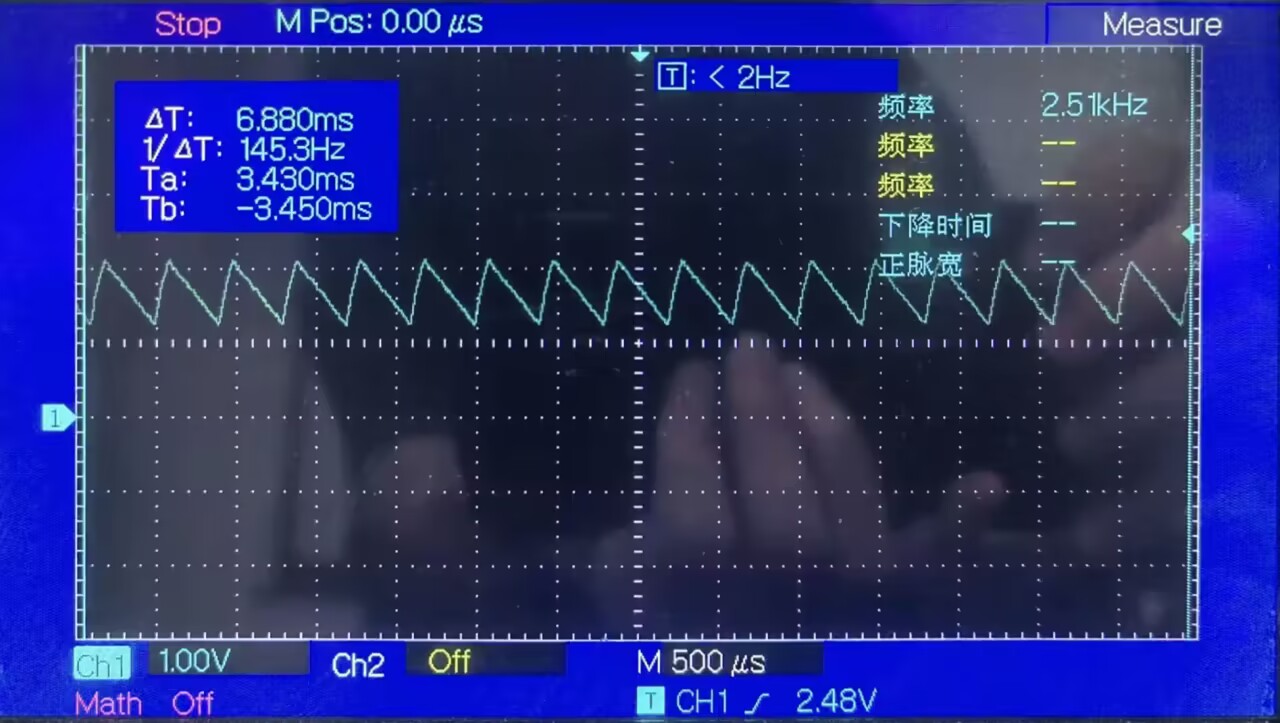
// Nixie\_SetBuf(7,10);

// Nixie\_SetBuf(8,10);

}

}

**实验现象描述：**



已经经过硬件电路滤波的波形。

**问题思考及实验收获感想：**

1. **遇到的问题及解决办法：**
2. PWM波形调节与稳定性：最初，我生成的PWM波形频率过低，导致DA转换的精度受限。通过提高PWM的频率与精度，我成功地提高了波形的稳定性和精度。
3. 数码管显示对程序速度的影响：由于PWM波形的产生与数码管显示的更新位于同一个定时器中断函数中，导致电平翻转的速度变慢，而DA转换需要高频率的被波形，这样的程序设计导致波形失真。最终舍弃了数码管显示部分，程序服务于产生高频PWM波形。
4. **实验结果分析与结论：**

在本次实验中，我不仅在技术实现上进行了探索，还尝试了一些创新性的设计。例如，在PWM波形调节中，我采用了动态调整算法，根据实时反馈调整PWM参数，以达到更优的转换效果。此外，在AD采样数据处理上，我自主设计并实现了滑动平均滤波算法，有效提高了数据的抗干扰能力。这些创新尝试不仅加深了我对知识点的理解，也锻炼了我的创新思维和问题解决能力。

1. **现代工具和手段的应用：**

我使用了示波器等电子测量仪器来观察和分析PWM信号的幅频特性，为频率测量提供了有力的支持。同时测量了DA转换后输出的波形特征，加深了对DA转换功能的理解。

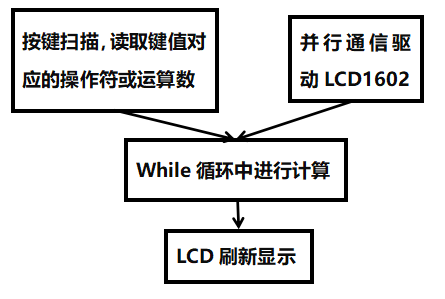
**实验名称： 实验七——矩阵键盘及LCD1602显示**

**实验目的和要求：**

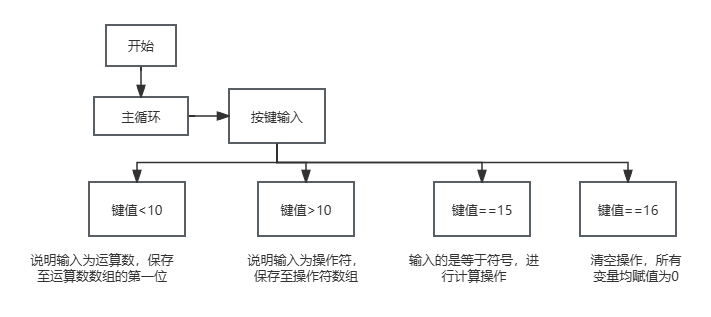
用4\*4矩阵键盘实现100以内任意两个数字的加减乘除运算，在1602上显示运算过程与结果。

**实验原理以及程序流程：**

1.程序设计原理:



2.计算流程:

****

**实验代码及注释：**

1. **并行通信驱动LCD1602：**

//引脚配置：

sbit LCD\_RS=P2^6;

sbit LCD\_RW=P2^5;

sbit LCD\_EN=P2^7;

#define LCD\_DataPort P0

1. **MatricKey.c：**

#include <REGX52.H>

#include "Delay.h"

/\*\*

\* @brief 矩阵键盘读取按键键码

\* @param 无

\* @retval KeyNumber 按下按键的键码值

如果按键按下不放，程序会停留在此函数，松手的一瞬间，返回按键键码，没有按键按下时，返回0

\*/

unsigned char MatrixKey()

{

unsigned char KeyNumber=0;

P1=0xFF;

P1\_3=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=1;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=5;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=9;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=13;}

P1=0xFF;

P1\_2=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=2;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=6;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=10;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=14;}

P1=0xFF;

P1\_1=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=3;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=7;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=11;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=15;}

P1=0xFF;

P1\_0=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=4;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=8;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=12;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=16;}

return KeyNumber;

}

1. **Main.c：**

#include <REGX51.H>

#include "LCD1602.h"

#include "MatrixKey.h"

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

| 1 | | 2 | | 3 | | 4 |

| 5 | | 6 | | 7 | | 8 |

| 9 | | 0 | | + | | - |

| \* | | / | | = | |clear|

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

int shuju[5]; //存放数据的数组

char yunsuan[5]; //存放运算符的数组

int keynum; //返回矩阵键盘的

int dat;

int j=0;

int bb=0;

int i=0;

int jj;

int ii;

int j3=0;

int i3=0;

int yshuju[5]; //存储\*与/对应的数据

char youxian[5]; //存储 \*与/运算符的数组

void result() //结果显示函数

{

LCD\_ShowString(1,16," ");

if(shuju[jj]<=9){LCD\_ShowNum(2,8,shuju[jj],1);}

if(shuju[jj]>9&&shuju[jj]<=99){LCD\_ShowNum(2,8,shuju[jj],2);}

if(shuju[jj]>99&&shuju[jj]<=999){LCD\_ShowNum(2,8,shuju[jj],3);}

}

void showdata() //显示输入的数据

{

if(dat<=9){LCD\_ShowNum(1,1,dat,1);}

if(dat>9&&dat<=99){LCD\_ShowNum(1,1,dat,2);}

if(dat>99&&dat<=999){LCD\_ShowNum(1,1,dat,3);}

}

void showdata1() //显示输入的数据

{

if(dat<=9){LCD\_ShowNum(1,4,dat,1);}

if(dat>9&&dat<=99){LCD\_ShowNum(1,4,dat,2);}

if(dat>99&&dat<=999){LCD\_ShowNum(1,4,dat,3);}

}

int flag=0;

void main()

{

LCD\_Init();

LCD\_ShowString(2,1,"OUTPUT:");

while(1)

{

keynum=MatrixKey(); //返回矩阵键盘的键值

if(keynum)

{

if(keynum>10)

{

flag=1;

shuju[j]=dat;

j++;

dat=0;

LCD\_ShowString(1,7," ");

if(keynum==11){yunsuan[i]='+';LCD\_ShowString(1,3,"+");i++;}

if(keynum==12){yunsuan[i]='-';LCD\_ShowString(1,3,"-");i++;}

if(keynum==13){yunsuan[i]='\*';LCD\_ShowString(1,3,"\*");i++;}

if(keynum==14){yunsuan[i]='/';LCD\_ShowString(1,3,"/");i++;}

if(keynum==15)//计算结果

{

for(ii=0,jj=1;jj<j;jj++) //优先遍历，找出\*与/对应的数组元素，并优先计算

{

if(yunsuan[ii]=='\*'|yunsuan[ii]=='/')

{

youxian[i3]=yunsuan[ii];

yshuju[j3]=shuju[jj];

if(youxian[i3]=='\*')

{

yshuju[j3]=shuju[jj]\*shuju[jj-1];

shuju[jj]=yshuju[j3];

shuju[jj-1]=0;

}

if(youxian[i3]=='/')

{

if(shuju[jj]==0){LCD\_ShowString(2,8,"ERROR");}

else

yshuju[j3]=shuju[jj-1]/shuju[jj];

shuju[jj]=yshuju[j3];

shuju[jj-1]=0;

}

yunsuan[ii]='+';

}

ii++;

}

for(ii=0,jj=1;jj<j;jj++) //第二次遍历，计算+ -法

{

if(yunsuan[ii]=='+')

{

shuju[jj]=shuju[jj]+shuju[jj-1];

result();

}

if(yunsuan[ii]=='-')

{

shuju[jj]=shuju[jj-1]-shuju[jj];

result();

}

if(yunsuan[ii]=='\*')

{

shuju[jj]=shuju[jj]\*shuju[jj-1];

result();

}

if(yunsuan[ii]=='/')

{

shuju[jj]=shuju[jj-1]/shuju[jj];

result();

}

ii++;

}

}

if(keynum==16){//清空，所有变量置0

youxian[5]=0;yshuju[5]=0;

j=0;i=0; jj=0;ii=0; j3=0;i3=0;

flag=0;yunsuan[5]=0;LCD\_ShowString(1,1," ");

shuju[5]=0;dat=0;LCD\_ShowString(2,8," ");}

}

if(keynum<=10) //输入的数据处理，若超过两位则左移

{

dat\*=10;

dat+=keynum%10;

if(flag==0){showdata();}

if(flag==1){showdata1();}

}

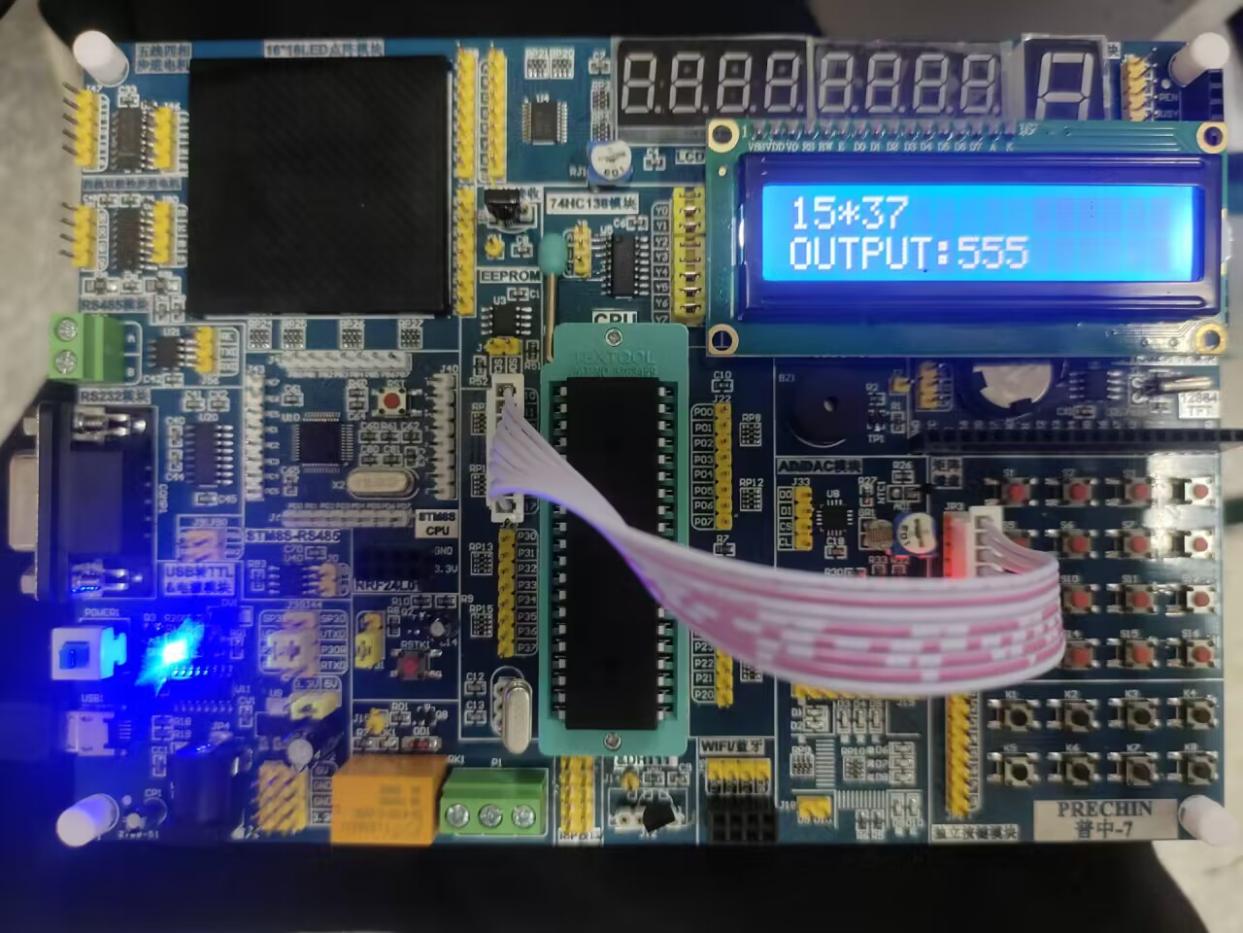
}

}

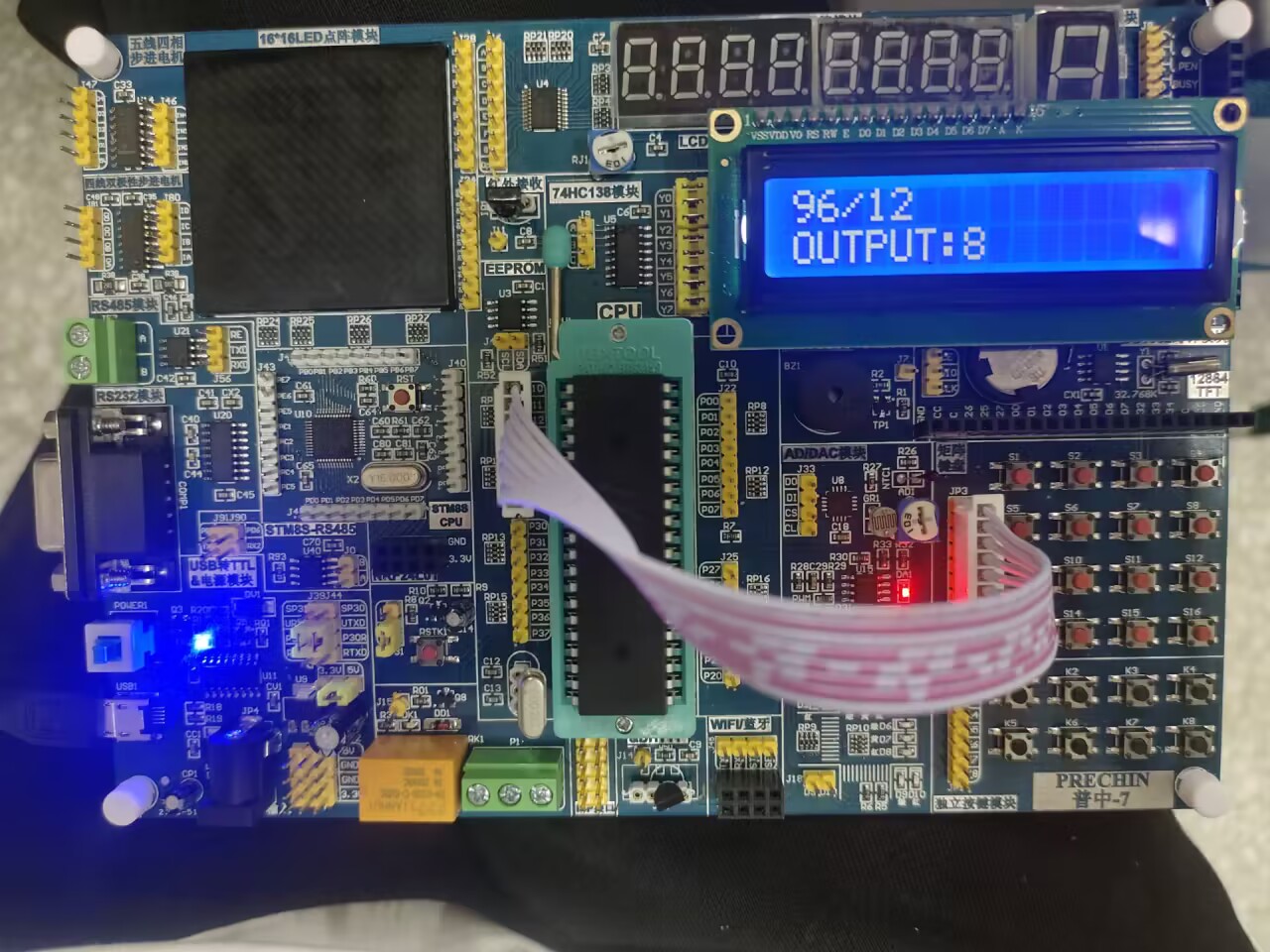
}

**实验现象描述：**

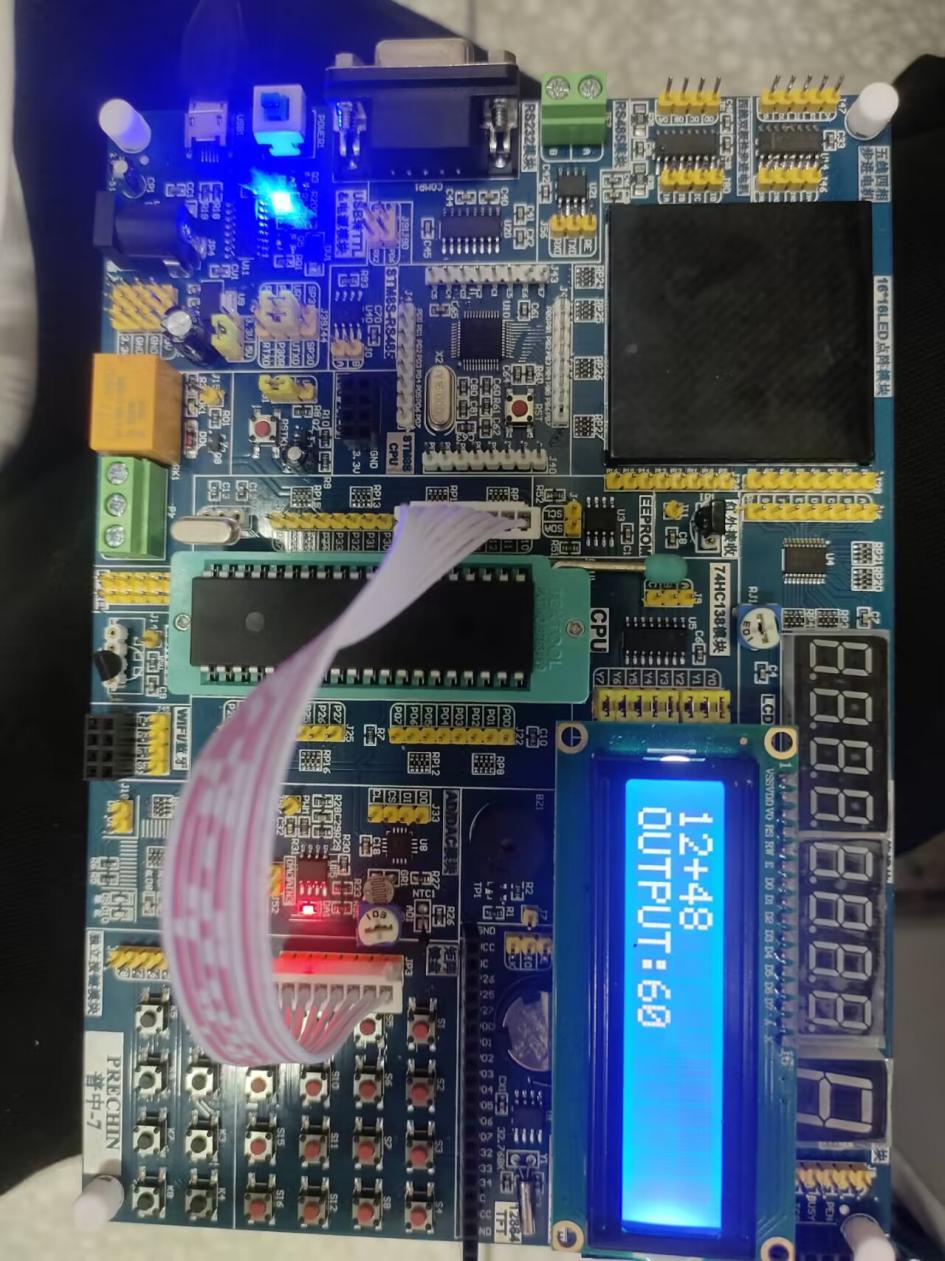
1.乘法运算



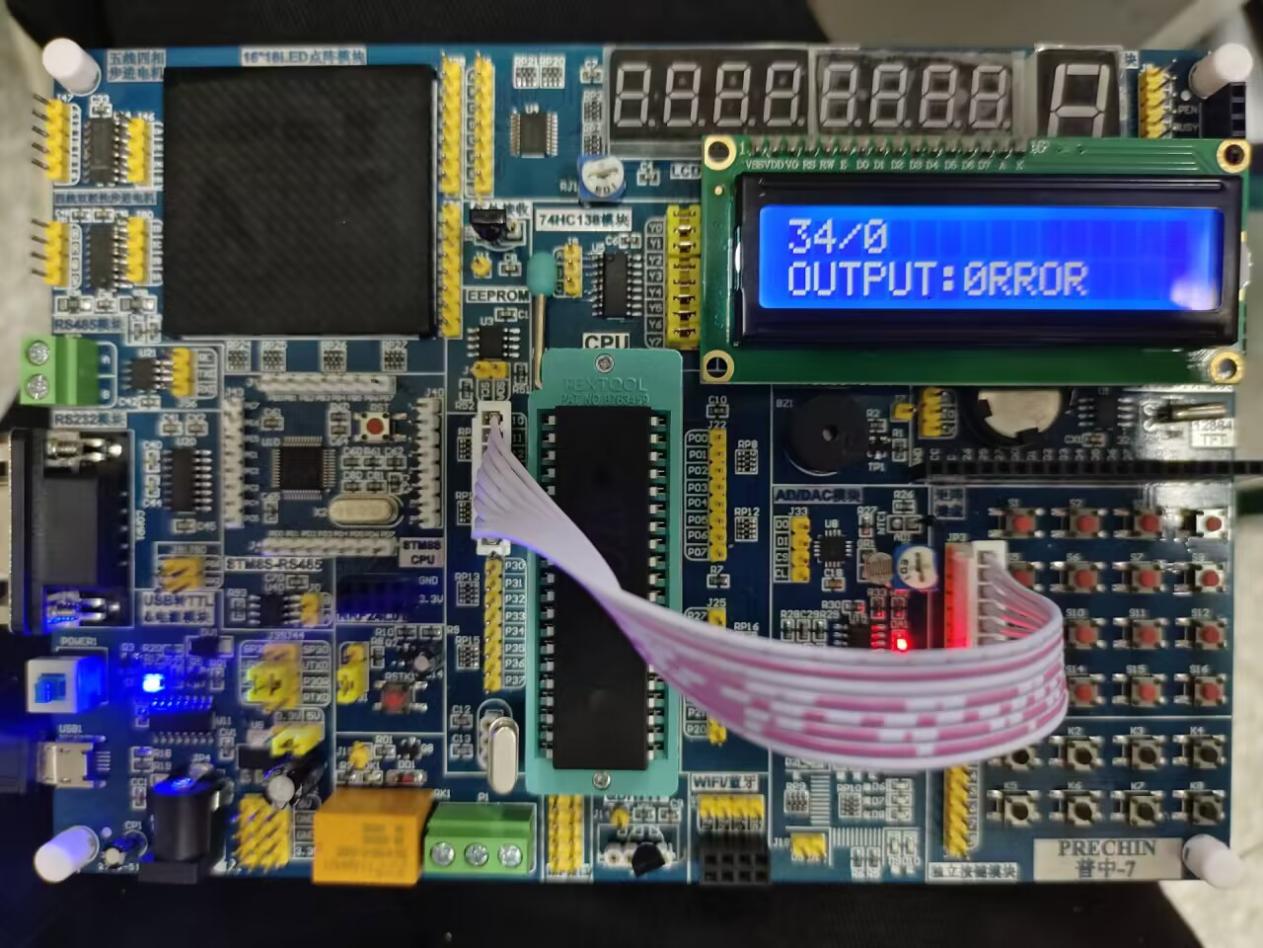
2.除法运算



3.减法运算



4.错误提示



**问题思考及实验收获感想：**

**1.遇到的问题及解决办法**：

1.按键扫描与去抖：初始时，键盘扫描的响应不够稳定，容易出现误触。通过引入软件去抖机制，即在检测到按键按下后，延迟一段时间（20ms）再次检测.

2.运算逻辑处理：该实验的设计中，软件编程能力相较于前几次的实验更加复杂，在该实验中我遇到的问题主要是在清零后重新运算时显示结果不正常，解决方法是通过将所有变量置零。

**2.实验结果分析与解释**：

LCD1602能够清晰地显示用户通过4\*4矩阵键盘输入的两个数字、选择的运算符以及最终的运算结果。通过对实验结果的反复验证，确认系统能够准确执行100以内任意两个数字的加减乘除运算，且界面友好，用户操作便捷。

**3.创新意识的体现**：

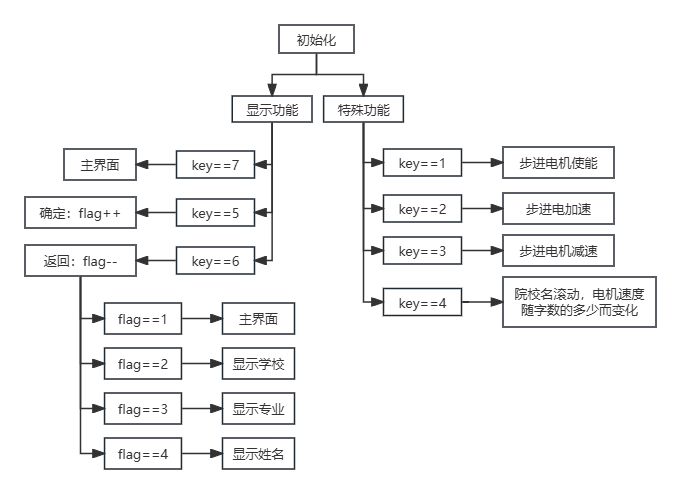
在实验过程中，我尝试了一些创新性的想法。例如，为了提升用户体验，我设计了一个简单的错误提示机制，当用户在除法运算中输入被除数为0时，在程序中加入了相应的判断和处理逻辑，LCD会显示“Error”提示信息，避免了程序崩溃。

**实验名称： 实验八——OLED菜单功能**

**实验目的和要求：**

给出三个菜单 用四个按键控制菜单的上翻，下翻，执行，返回，被选中的菜单黑白反转（黑字白底转成白字黑底），按下执行，执行自定义的动作，按下返回，返回主菜单。

**实验原理以及程序流程：**



**实验代码及注释：**

#include <REGX51.H>

#include "Delay.h"

#include "LCD9648.h"

#include "ST7565.h"

#include<intrins.h> //因为要用到左右移函数，所以加入这个头文件

typedef unsigned int u16; //对数据类型进行声明定义

typedef unsigned char u8;

////修改此值可改变电机旋转速度，不能过大或过小 2-10

sbit KEY1=P3^0;

sbit KEY2=P3^1;

sbit KEY3=P3^2;

sbit KEY4=P3^3;

int stepmove=0;//步进电机旋转标志位，0为停止，1为旋转

//typedef unsigned int u16; //对系统默认数据类型进行重定义

//typedef unsigned char u8;

//typedef unsigned long u32;

//使用宏定义独立按键按下的键值

#define KEY1\_PRESS 1

#define KEY2\_PRESS 2

#define KEY3\_PRESS 3

#define KEY4\_PRESS 4

#define KEY\_UNPRESS 0

unsigned char key\_scan()

{

static u8 key=1;

// if(mode)key=1;//连续扫描按键

if(key==1&&(KEY1==0||KEY2==0||KEY3==0||KEY4==0))//任意按键按下

{

Delay(20);//消抖

key=0;

if(KEY1==0)

return KEY1\_PRESS;

else if(KEY2==0)

return KEY2\_PRESS;

else if(KEY3==0)

return KEY3\_PRESS;

else if(KEY4==0)

return KEY4\_PRESS;

}

else if(KEY1==1&&KEY2==1&&KEY3==1&&KEY4==1) //无按键按下

{

key=1;

}

return KEY\_UNPRESS;

}

unsigned char Counter,Compare; //计数值和比较值，用于输出PWM

unsigned char KeyNum,Speed;

void Timer0\_Init(void)

{

TMOD &= 0xF0; //设置定时器模式

TMOD |= 0x01; //设置定时器模式

TL0 = 0x9C; //设置定时初值

TH0 = 0xFF; //设置定时初值

TF0 = 0; //清除TF0标志

TR0 = 1; //定时器0开始计时

ET0=1;

EA=1;

PT0=0;

}

#define SPEEDMAX 20

void Timer0Init(void) //1毫秒@12.000MHz

{

// AUXR &= 0x7F; //定时器时钟12T模式

TMOD &= 0xF0; //设置定时器模式

TMOD |= 0x01; //设置定时器模式

TL0 = 0x18; //设置定时初值

TH0 = 0xFC; //设置定时初值

TF0 = 0; //清除TF0标志

TR0 = 1; //定时器0开始计时

}

void main()

{

u8 key=0;

u8 dir=0;

u8 step=0;

u8 speed=SPEEDMAX;

uchar i = 0;

uchar j = 0;

int flag=0;

int fg=0;

LCD\_Init();

Clear();

Timer0\_Init();

while(1)

{

if(key==KEY1\_PRESS)//按键一按下令步进电机旋转或停止

{

stepmove=!stepmove;

}

if(stepmove==1) //判断是否要让步进电机旋转

{

// step\_28byj48\_control(step++,dir);

if(step==8) step=0;

Delay(speed);

}

if(key==KEY4\_PRESS)//按下按键四令步进电机反转

{

dir=!dir;

}

//\*\*\*\*\*\*\*\*\*步进电机速度控制环节

else if (key==KEY2\_PRESS)//按下按键二令步进电机加速se if (key==KEY3\_PRESS)

{

if(speed>SPEEDMAX)

speed-=1;

}

{

// if(speed<SPEEDMIN) //按下按键三令步进电机减速

speed+=1;

}Delay(SPEEDMAX);//通过每一步的延时来调节步进电机的速度

if(P3\_4==0){Delay(20);while(P3\_3==0);Delay(20);

//flag=0;

// Speed++;

// Speed%=6;

// if(Speed==0){Compare=0;LCD12864\_Write16CnCHAR(0,0,"院空空空空空");} //设置比较值，改变PWM占空比

// if(Speed==1){Compare=20;LCD12864\_Write16CnCHAR(0,0,"术院空空空空");}

// if(Speed==2){Compare=40;LCD12864\_Write16CnCHAR(0,0,"技术院空空空");}

// if(Speed==3){Compare=60;LCD12864\_Write16CnCHAR(0,0,"村技术院空空");}

// if(Speed==4){Compare=80;LCD12864\_Write16CnCHAR(0,0,"湾村技术院空");} //设置比较值，改变PWM占空比

// if(Speed==5){Compare=100;LCD12864\_Write16CnCHAR(0,0,"林湾村技术院");}

}

if(P3\_1==0) //如果K2按键按下

{

flag=0;

Delay(20);

while(P3\_1==0);

Delay(20);

Clear();

while(1){

LCD12864\_Write16CnCHAR(0,0,"院空空空空空");Compare=10;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"术院空空空空");Compare=30;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"技术院空空空");Compare=40;

Delay(100);

LCD12864\_Write16CnCHAR(0,0,"村技术院空空");Compare=60;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"湾村技术院空");Compare=80;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"林湾村技术院");Compare=100;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"空林湾村技术");Compare=80;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"空空林湾村技");Compare=60;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"空空空林湾村");Compare=40;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"空空空空林湾");Compare=30;

Delay(100);

if(P3\_7==0) break;

LCD12864\_Write16CnCHAR(0,0,"空空空空空林"); Compare=20;

Delay(100);

if(P3\_7==0) break;}

}

if(P3\_5==0) //如果K5按键按下

{

Delay(20);

while(P3\_1==0);

Delay(20);

flag++;

}

if(P3\_6==0) //如果K6按键按下

{

Delay(20);

while(P3\_1==0);

Delay(20);

fg++;

flag--;

}

if(P3\_7==0||flag==1) //如果K7按键按下

{

Delay(20);

while(P3\_1==0);

Delay(20);

//compare=0;

LCD12864\_Write16CnCHAR(0,0,"设置空空菜单");

LCD12864\_Write16CnCHAR(0,4,"显示学校？空");

flag=1;

}

if(flag==2){

LCD12864\_Write16CnCHAR(0,0,"林湾村技术院");

LCD12864\_Write16CnCHAR(0,4,"显示专业？空");

}

if(flag==3){

LCD12864\_Write16CnCHAR(0,0,"电子信息空空");

LCD12864\_Write16CnCHAR(0,4,"显示姓名？空");

}

if(flag==4){

LCD12864\_Write16CnCHAR(0,0,"王乙斯空空空");

LCD12864\_Write16CnCHAR(0,4,"菜单空空空");

}

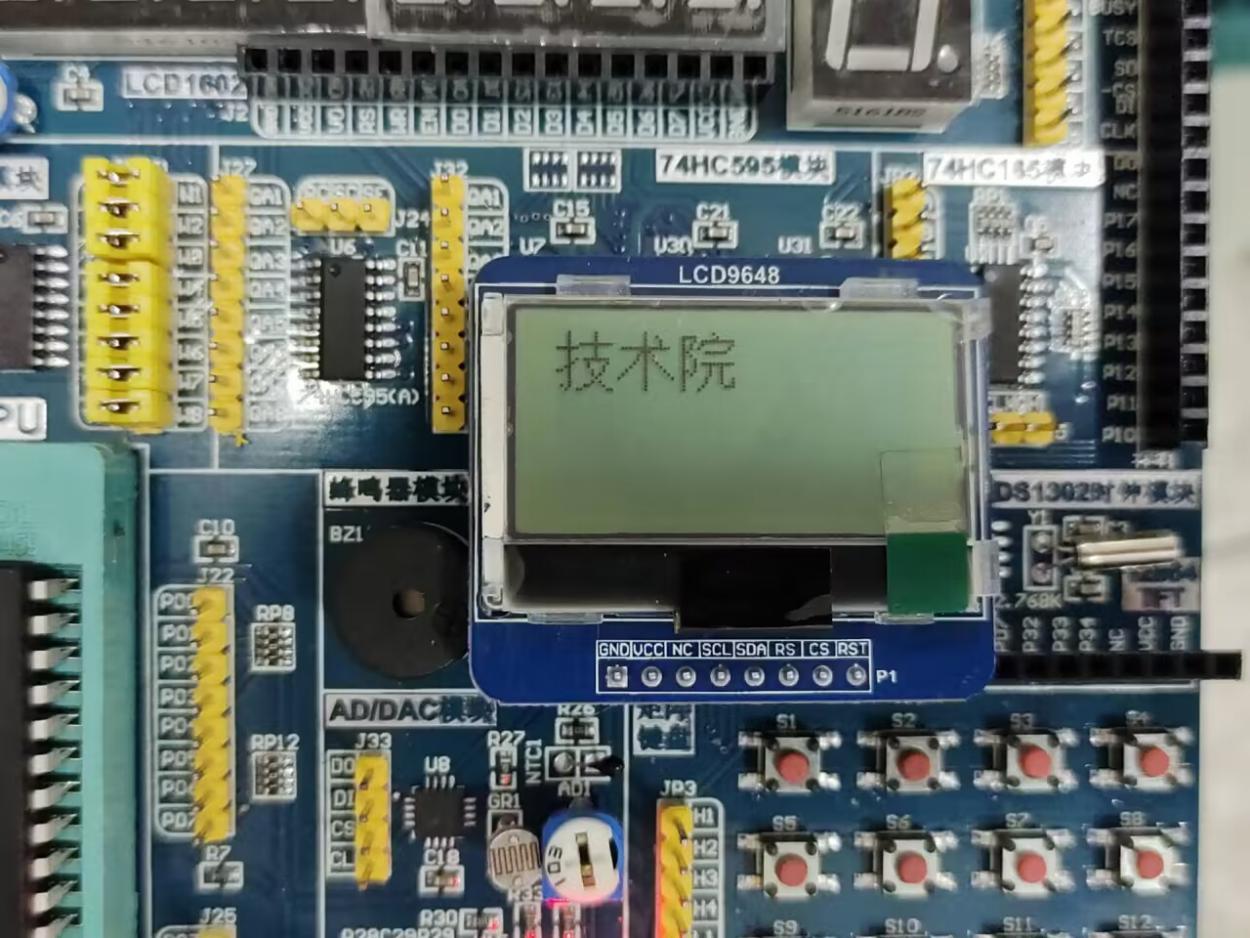
if(flag>=5)flag=1;

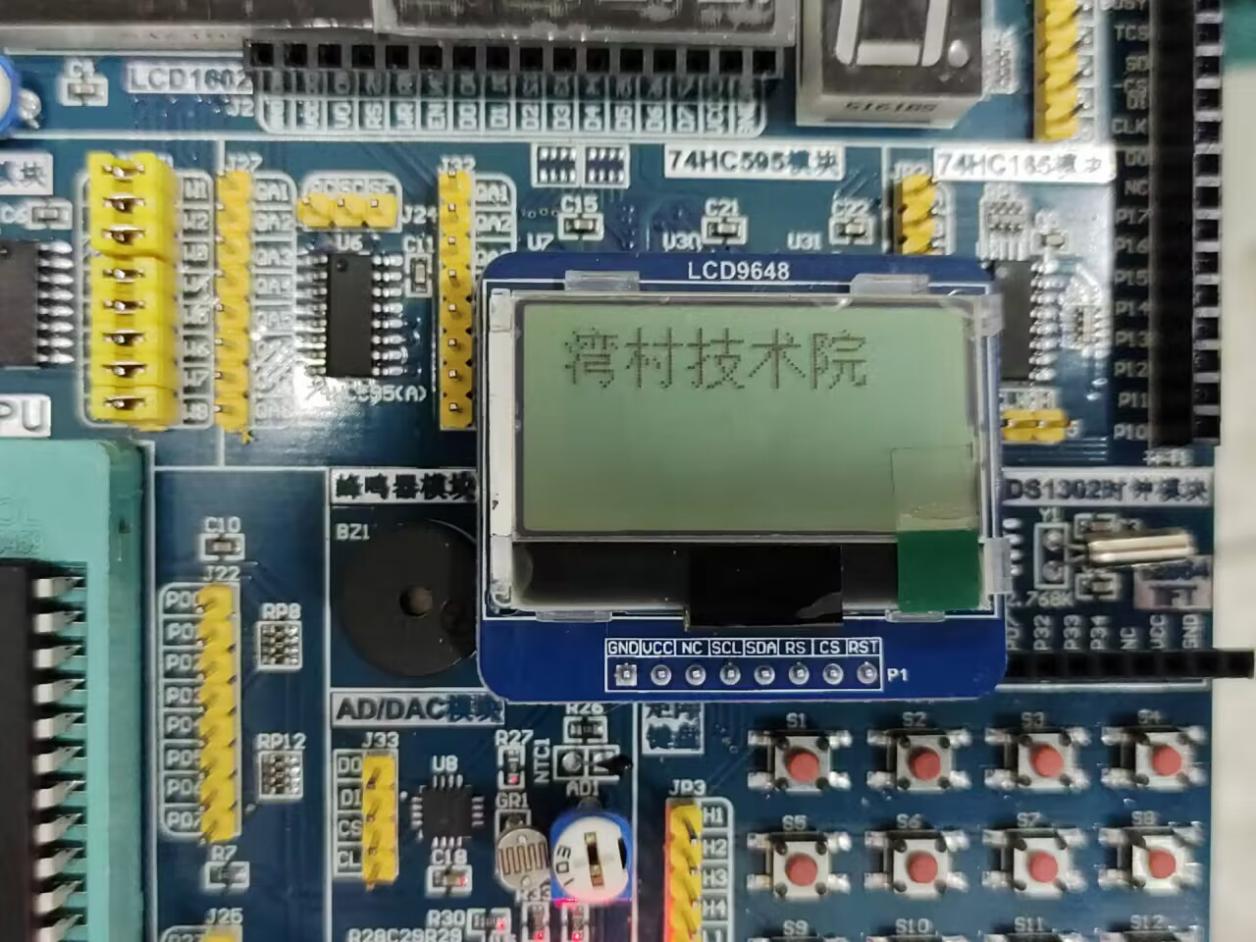
}

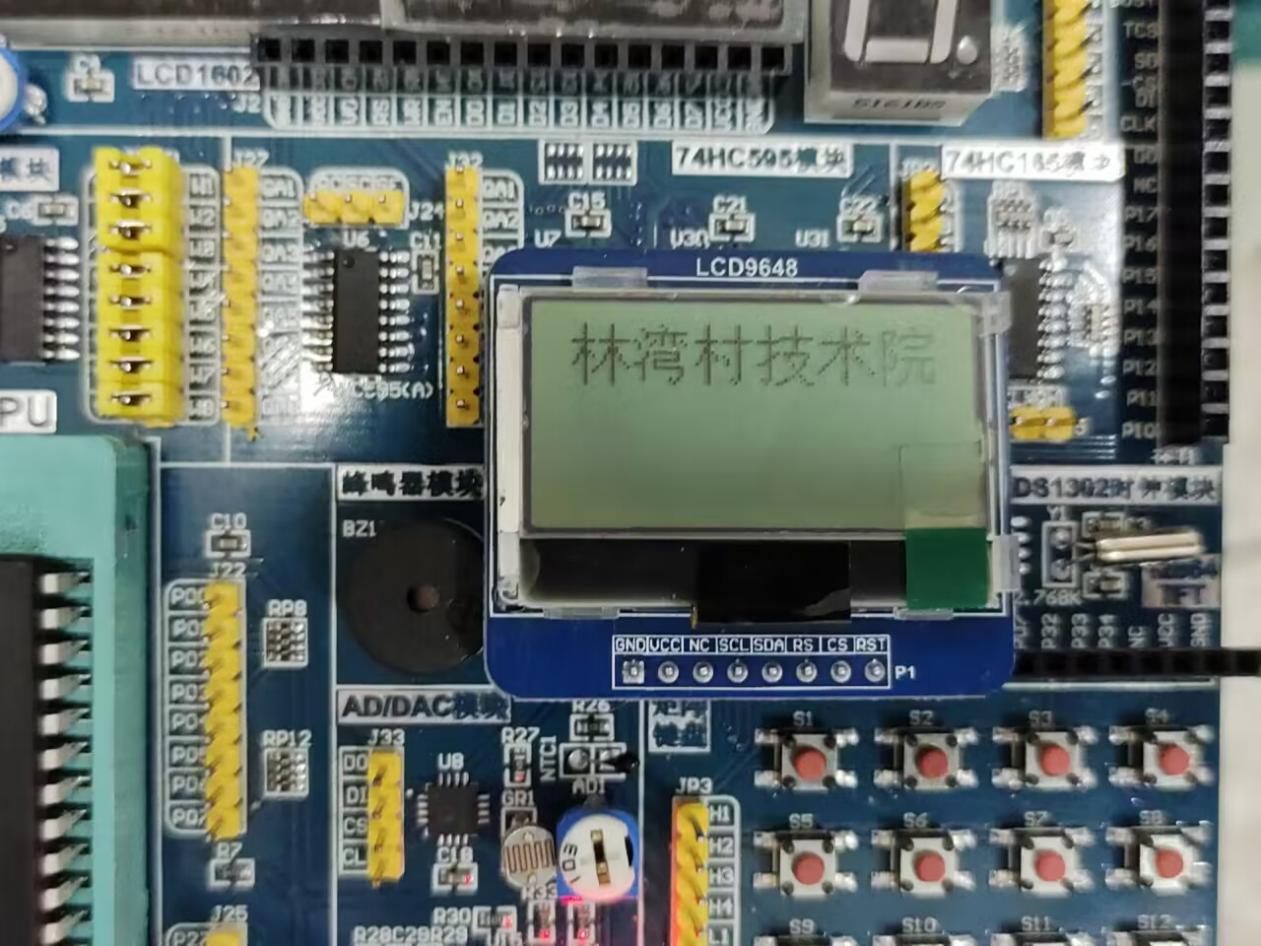
}

**实验现象描述：**

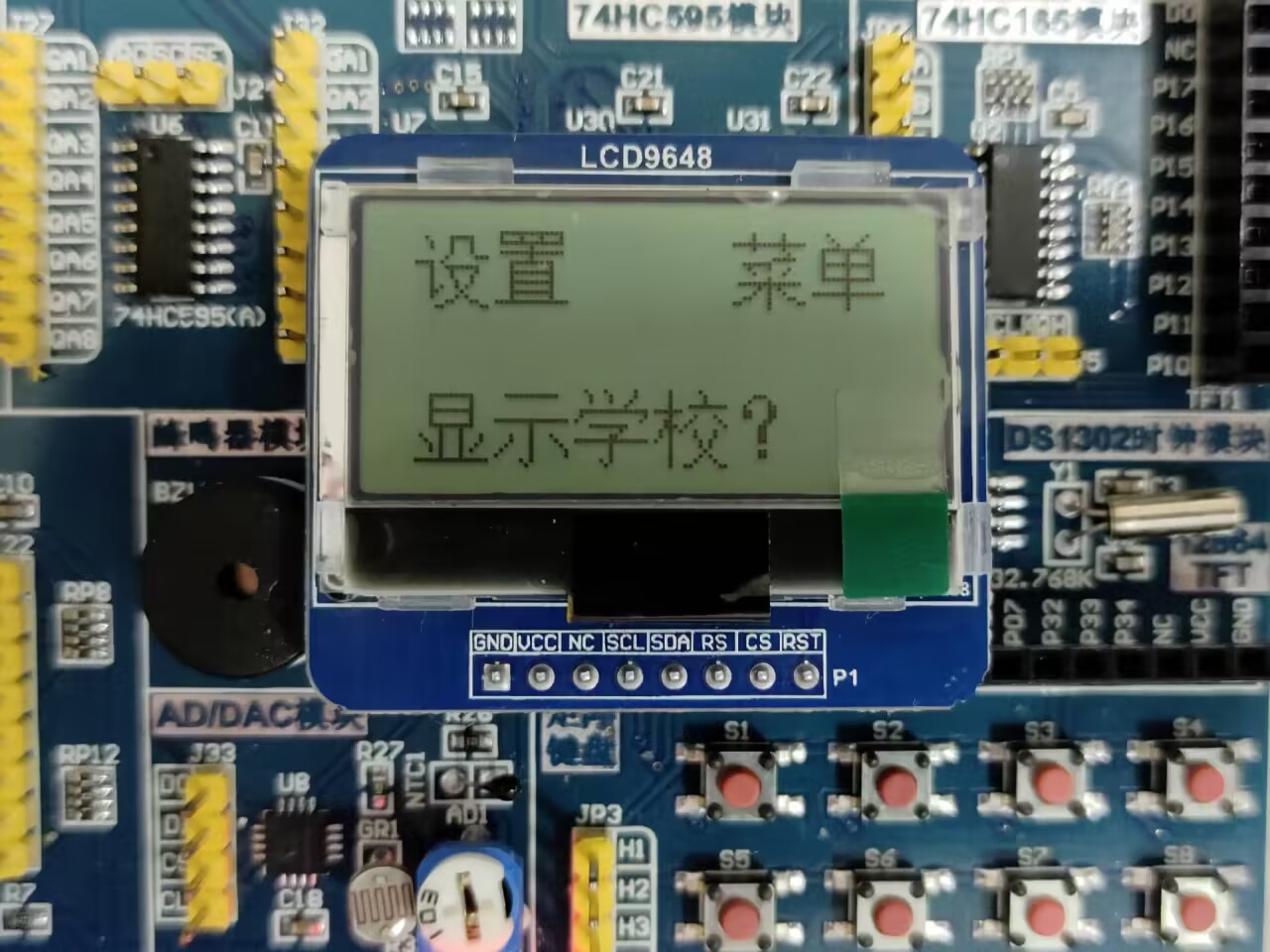
1. 滚动字幕



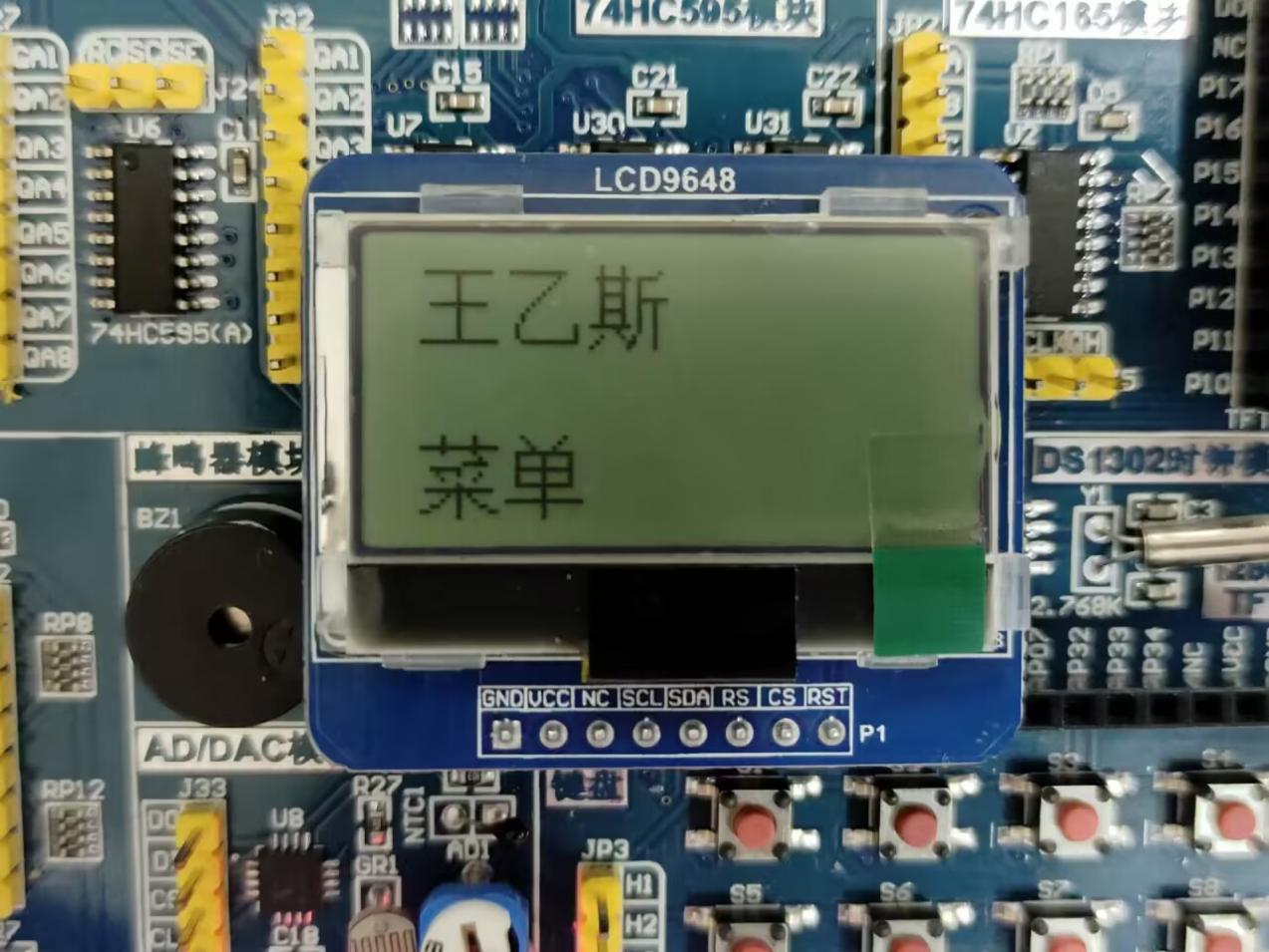




1. 页面







**问题思考及实验收获感想：**

**1.遇到的问题及解决办法**：

1.菜单功能逻辑混乱：在编写菜单切换逻辑时，由于按键去抖动处理不当，导致菜单跳转不稳定。我采用了软件延时去抖动的方法，即在检测到按键状态变化后，延迟一段时间再次检测，确保按键状态稳定后再进行下一步操作，从而解决了这一问题。

**2.实验结果分析与解释**：

通过OLED显示屏，我们成功实现了一个直观的用户界面，用户可以方便地在菜单中点击执行和返回操作。电机调速和步进电机调速实验的成功，验证了PWM信号和步进控制的有效性。

**3.实验收获：**

驱动了OLED并实现了菜单功能，同时完成了对电机及步进电机的调速控制。过程中遇到初始化、信号不稳定等挑战，但通过细致调试和资料查阅一一克服。此次实验不仅提升了我的编程与硬件调试能力，还加深了对控制算法的理解。收获颇丰，为未来的嵌入式系统开发打下了坚实基础。