实 验 报 告
EXPERIMENT REPORT

姓名：高星杰

学号：2021307220712

专业：计算机科学与技术

教师：高俊祥

科目：接口技术

信 息 学 院
COLLEGE OF INFORMATIC

一、 实验目的

1、熟悉 8086 的最小模式，掌握基本的硬件编程步骤。
2、熟练掌握使用可编程并行接口 8255A 芯片，体会并行通信。
3、熟练掌握使用 8253 定时计数器的方法，体会分频器的用途。
4、熟练掌握中断控制器接口 8259 进行中断的方法，体会中断系统的威力。
5、熟练掌握 A/D 数模转换器，对计算机数据捕获有一定理解
5、培养硬件思维，提高设计实验方案的能力和虚拟仿真的能力。
6、培养解决问题能力，能够读借助现有工具和资源解决问题。
7、锻炼硬件编程代码能力，能够编写逻辑清楚，风格规范的代码

二、 实验介绍

实验名称：基于 8086 微处理器以及 8255A、8253、8259、A\D0808 等接口芯片的多功能时钟

实验背景：根据接口技术实验课的课程安排，要实现一个功能合理的使用芯片丰富的项目，我们力求功能丰富的同时，又能够有一定的实际意义，我们选择了多功能时钟。同时由于项目的复杂性要求我们的有更好的工具去编写项目，所以我们在尝试仅使用汇编后，选择了使用 c 语言和汇编混合编写。

实验项目功能：

1.时间显示功能：能够实时的显示小时/分钟/秒
2.闹钟设置功能：能够简单的设置二十四小时内的闹钟
3.闹钟到时响铃功能：时间到闹钟时能够发出声音提醒
4.开启/关闭闹钟功能：在闹钟提醒时或其他平常工作时能够关闭闹钟提醒功能或者开启闹钟提醒功能
5.显示周围温度功能：能够实时的显示从周围环境获取的温度信息

实验前置依赖：

1. Proteus Professional 8.9
2. 8086 最小模式的知识
3. 8255A、8253、8259、A/D0809 接口芯片的基础知识
4. 基本汇编语法知识
5. 基本 C 语言语法知识

三、 实验设计与实现

1、功能需求分析：

（1） 显示时间：

选择显示的器件：7 段数码管

为什么选择 7 段数码管？想要显示时间就需要使用显示器，而显示器种类繁多，功能有简易的有复杂的，在这里针对我们显示时间的需要来说，使用 7 段数码管最为合适，原因有二：一是显示时间的根本是显示数字，而 7 段数码管的数字显示以一致的方式进行，每个数字都用相同的 7 个段表示。这一致性有助于提高数字的可读性，使人们容易识别和理解时间。二是数字时钟通常需要显示小时和分钟，有时甚至包括秒。使用 7 段数码管，每个数字可以通过点亮或熄灭相应的 LED 段来表示，使得数字时钟的控制电路相对简单。

时间显示的范围：24 小时/60 分钟/60 秒

为什么要使用这个时间范围？原本考虑要加年/日/天，但是年/日/天和小时/分钟/秒实际上在实现上比较重复，仅仅是在小时/分钟/秒的基础上添加几个变量，增添几个进位逻辑，考虑到重复实现的意义不是很大，所以我们就没有再去重复实现。

（2） **闹钟设置：**

设置闹钟的方式：我们使用的是类似滑动选择器一样，用户可以使用按钮来选择小时和分钟。这种界面设计可以更直观地让用户增加到他们想要的确切时间。

闹钟响铃方式：我们使用的扬声器发声响铃。

关闭闹钟的方式：使用按钮关闭闹钟

（3） **温度收集：**

温度收集的方式：需要使用温度传感器来采集周围温度数据

2、模块化设计：

为什么使用模块化设计？一是如果不使用模块化设计，不能实现代码的复用，要写很多重复的代码，花费更多的时间，二是使用模块化设计的代码能够的更加明确功能，然后用清晰的逻辑编写代码，三是使用模块化设计更容易定位 bug，解决 bug（这点最重要的优点，在我们项目后期帮了我们很大的忙），其次使用模块化设计还有很多优点不知不觉已经帮我们解决了一堆问题，帮了很大的忙。所以我们强烈推荐用模块化设计的方式来设计，来写代码。

（1） **显示模块**

功能：负责接收要显示的数据，然后输出数据给 7 段数码管，让 7 段数码管正常显示。

输入：要显示的数据

输出：使用对应的片选信号，让 7 段数码管正常显示

（2）**闹钟模块**

功能 1：响铃

输入：无

输出：使用 8255A 对应的片选信号，输出对应的数据，让扬声器播放声音

功能 2：停止响铃

输入：无

输出：使用 8255A 对应的片选信号，输出对应的数据，让扬声器停止播声音

功能 3：屏蔽响铃（关闭闹钟功能）

输入：当前屏蔽响铃的状态

输出：更新当前屏蔽响铃状态（更新内存中变量的值）。

（2） **温度模块**

功能 1：获取当前温度

输入：无

输出：通过输出响应的触发信号，接收来自 0808 的转换完成后的数据

（3） **调度模块（程序入口）**

功能 1：循环调用显示模块

输入：无

输出：无

功能 2：维护全局变量（如温度、时间、闹钟时间）

输入：要修改的变量的值

输出：无

功能 3：提供封装好的输入输出方法

输入：要让 CPU 输入输出的地址和数据
输出：无
功能 4：初始化接口芯片
输入：无
输出：无

（4） **中断处理模块**

功能 1：初始化 8259 芯片
输入：无
输出：无
功能 2：初始化中断向量表
输入：无
输出：无
功能 3：中断服务程序，在有中断产生后进行处理
输入：中断的信号
输出：无

每一个模块，都可以在我们代码中很清晰的找到。

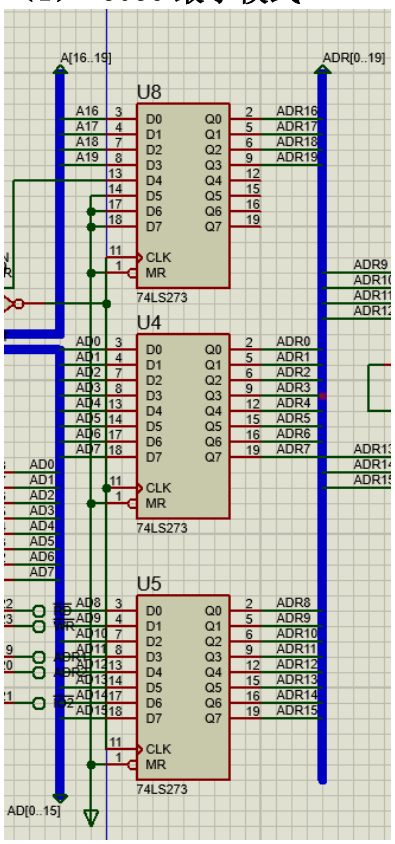
3、电路设计与代码编写

我们在进行项目的实现过程中，不自觉的就将电路设计和代码编写放在了一起进行，而且非常行之有效，并且我们总结出了设计实现的一般步骤：

1. 需求分析，功能结构设计
2. 逻辑步骤分析
3. 使用电路图实现功能
4. 编写程序代码驱动硬件

基于此设计实现的一般步骤我们对每个模块进行了实现

（1） **8086 最小模式**



最小模式介绍：

高老师在 ppt 上讲的 8086 最小模式是 3 片地址锁存器 8282 或 74LS273、 2 片双向数据缓冲器 8286 或 74LS245，但是这里我们仅使用了 3 片地址锁存器，而没有使用数据锁存器，因为我们当时使用 74LS245 时，总时获取不到数据，也就时 74LS245 不工作，这里我们就调了几乎一上午，发现仍然不工作，可能是软件 bug 也有可能是该芯片的引脚，没有分配正确，然后询问了其他组成员，发现都没有使用数据缓存器，并且发现在我们的项目中数据缓存器是可有可无的，我们可以直接使用数据线上的数据而不进行缓存。

为什么使用地址锁存器？

由于的 8086 的总线是数据线和地址线复用的（总线既是数据线也是地址线），我们要使片选和数据同时所存就要使用到锁存器来锁存其中一个，我们锁存的是地址线。工作实例解释：8086 在向外输出数据时，是先向地址总线输出地址，然后我们使用的 74LS273 芯片锁存地址，再当 8086 输出数据时，我们就可以使用锁存的地址来片选到想要输出的地方，然后该芯片再获取的数据总线上的数据。同时我们又可以发现一个问题既然 8086 是地址线和数据线复用的那我们该如何区别是数据还是地址呢？可以带着这个问题往下看。

为什么使用三片 74LS245?

8086 的地址总线一共 20 位，一片芯片可以锁存 8 位所以使用了 3 片 74LS245

74LS273 芯片引脚介绍：

CLK：时钟信号，用来控制芯片工作的，上升沿触发，即当 CP 从低到高电平时，D0~D7 的数据通过芯片；为 0 时将数据锁存，D0~D7 的数据不变。

MR: MR 引脚用于将 74HC273 的所有触发器置于复位状态。当 MR 引脚为低电平（或使能）时，所有的触发器将被复位，即其输出将被强制为低电平。这意味着，无论之前的输入数据是什么，触发器的输出都将被清零。当 MR 引脚为高电平时，触发器将根据输入信号进行正常的工作，即根据时钟信号和 D 输入信号更新其输出状。（74HC273 的 MR 引脚我们没有用到，如果有用到可以不让它接高电平）

为什么 74LS273 芯片 CLK 要连接 8086 的 ALE?

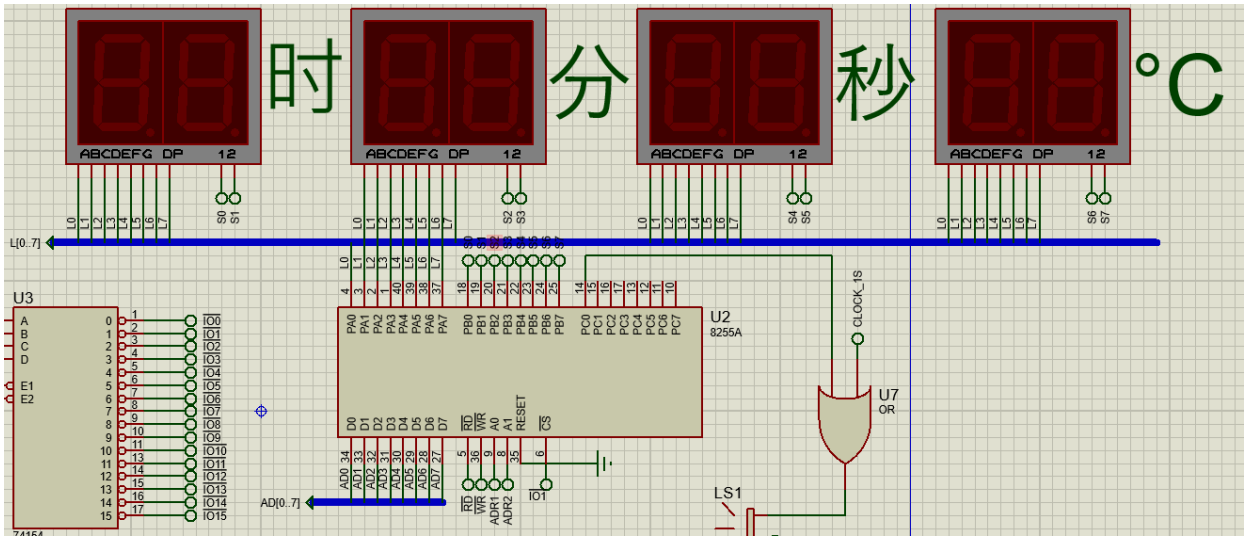
首先要清楚 8086ALE 引脚的作用：当 8086 处理器需要将地址传输到外部总线上时，它会通过 ALE 引脚发出一个脉冲信号。这个脉冲信号的上升沿会触发外部设备对地址进行锁存，以便外部设备可以读取或响应该地址。简单来说就是 8086 在输出地址之前会通过 ALE 发出一个脉冲信号。也就是通过这个信号来判断总线上是地址而不是数据（还有其他引脚来判断总线上数据而不是地址）。

了解了这个几个问题我们大概也就清楚了 8086 的最小模式了，然后我们就可以继续基于 8086 的最小模式往下开发了。

（2） 显示模块

使用芯片：7SEG-MPX2-CC(7 段数码管),74154(4-16 译码器)， 8255A

连接电路：



在这里地址线怎么用，片选信号怎么设置，都可以自己定义了，但是要主要这里定义了之后写的程序就需要使用这个地址。

我们使用 A 端口输出数据就是让 7 段数码管显示的数据，B 端口进行片选哪个 7 段数码管（在这里我们使用数据来进行了片选）

为什么使用数据线进行片选 7 段数码管？

因为我们使用该 8255A 专门用来显示，只要用地址线片选到该 8255A 就可以了，然后用数据线来显示就可以，没有必要使用地址再来片选哪一个数码管的了。

8255A 的引脚连接说明：

RD/WR:是出入口，8086 告诉 8255A 当前是要读数据，还是要写数据

CS：是片选信号，低电平有效，我们讲 4-16 译码器的译出来的码连接到这里进行片选。

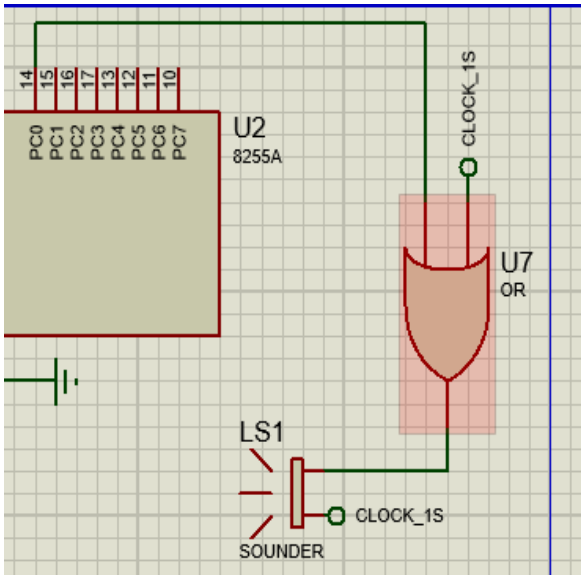
对应的代码：可以看我们项目文件中的 show.c 文件

代码逻辑：将传入的数据转化为数码管的显示信号，并输出，使数码管能够显示

（3） 闹钟模块

使用芯片：SOUNDER

连接电路：



对应代码：

```
        if(hour==hour_alarm&&minute==minute_alarm&&flag2==0)
        {
                output(0x204,1);
        }
        else if(minute!=minute_alarm)
        {
                output(0x204,0);
        }
```

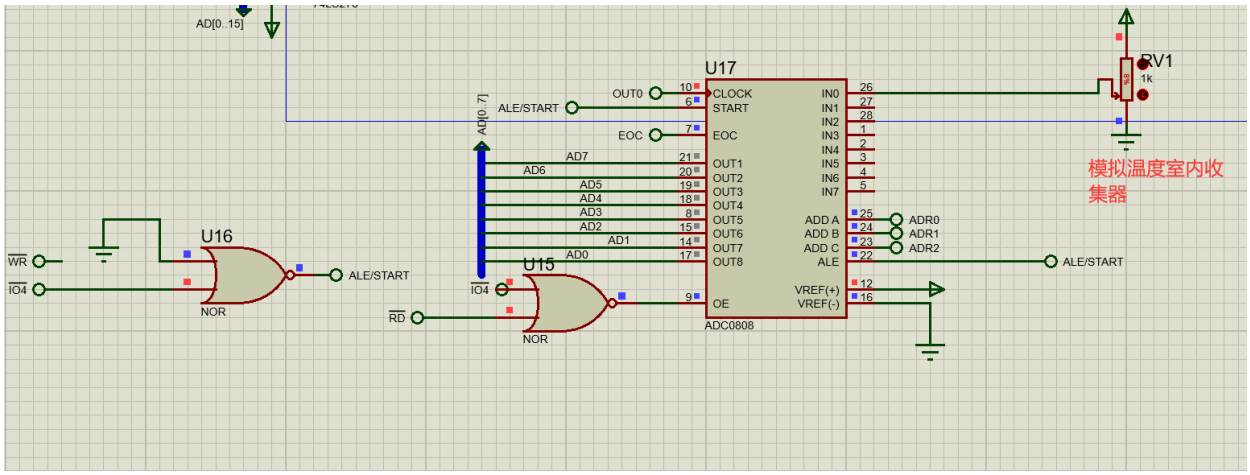
代码逻辑：判断是否是闹钟的 1 分钟之内，如果是则响铃，如果已经不是了就停止响铃。

停止闹钟：这个功能是嵌入在中断处理模块的。

（4） 温度模块

使用芯片：POT-HG，ADC0808

连接电路：



对应代码：获取温度也是在中断系统中实现的。

（5） 调度模块

对应代码（main.c、main.h）：

```
void main(void)
{
    init();//初始化
    while(1)//并行
    {
        if (flag==0)
        {
            //如果是显示设置闹钟模式
            show(hour_alarm,minute_alarm,second_alarm,heat);
         }
         else
        {
            show(hour,minute,second,heat);
        }
        if(hour==hour_alarm&&minute==minute_alarm&&flag2==0)
        {
                output(0x204,1);
        }
        else if(minute!=minute_alarm)
        {
                output(0x204,0);
        }
        //heat=readHeat();
        //delay(200);
    }

}
```

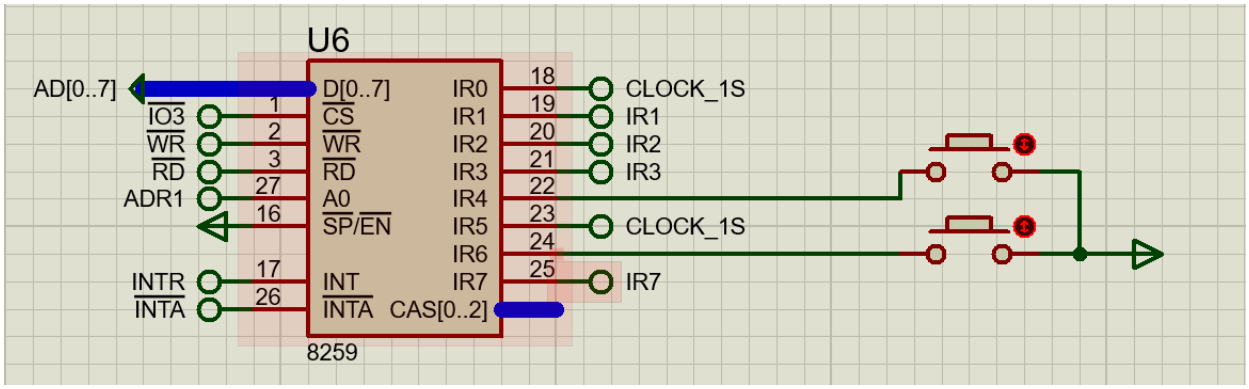
全局变量：

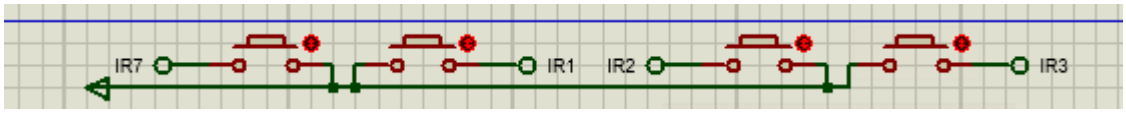
```
int hour =8, minute = 59, second = 55;//时钟显示时间
int hour_alarm = 9,minute_alarm = 0, second_alarm = 5;//闹钟时间
int heat=0;
//闹钟和时钟切换 true 为时钟模式
int flag = 1;
//表示是屏蔽闹钟
//true是屏蔽
int flag2=0;
```

这里实现的是并发的逻辑，首先初始化，然后进入一个循环程序，我们在代码中看到的是串行的但是实际在运行的过程中，我们看起来更像是并行，所以称其为并发的。

（6） 中断处理模块（interrupt.c）

连接电路：





8259 引脚连接说明

INT：是向 8086 申请中断的引脚，连接 8086 的 INTR

INTA：是 8086 向 8259 返回的响应中断的引脚，连接 8086 的 INTA

对应代码：在 interrupt.c 文件中这里由于篇幅问题就不在详细介绍了。

代码逻辑：就是提供 8259 初始化，中断向量表的初始化

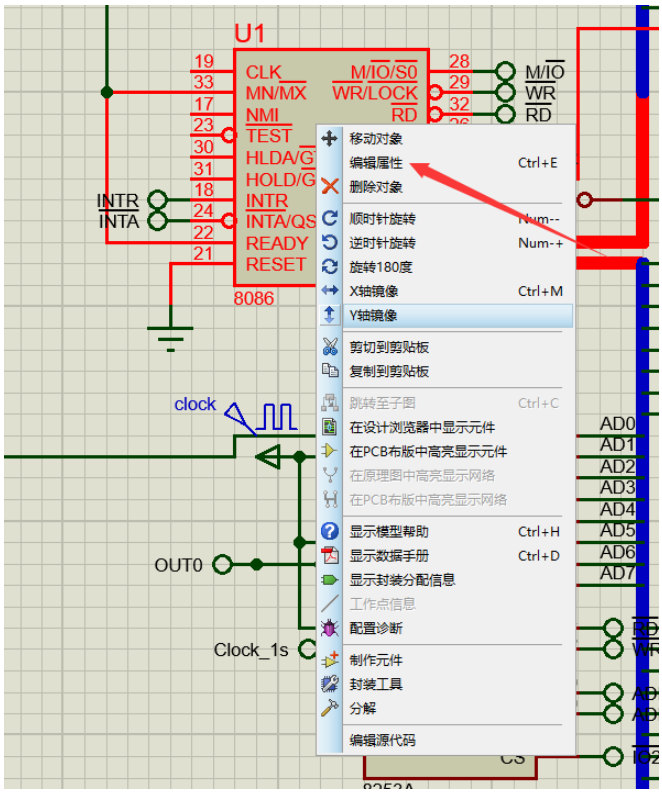
四、 实验中出现的问题与解决

相信你看完前面已经了解的我们的项目了，你一定会想，这么简单的逻辑，这么简单的电路谁不行啊？我们一开始在实现项目之前也是这样想的，但是实际实现的过程花费了大量的时间，在“发现问题——解决问题——出现新问题”的周期中无线循环着，似乎永远见不到光明一样，不知道其他的组是否和我们一样举步维艰。但是最后的最后我们终于走出来了，现在把我们的探索出的武功秘籍赠与你。

发现的问题和解决：

1.8086 编写汇编后不能运行

解决：改了 8086 内存大小，load_segement 大小



2. 地址不能缓存，debug 乱跳（不是一步一步执行的）

解决：使用了样例的汇编模板，和我们课上学习一般的汇编程序有所不同

```
.MODEL   SMALL

.8086
.stack
.code
extern _main:near
.startup
    ;cli            ; interrupt disable
    call near ptr _main
endless:
    jmp  endless
.data
public    __acrtused          ; trick to force in startup

__acrtused = 9876h      ; funny value not easily matched ; ; in SYMDEB

END
```

这里写你的代码

3. 数据不能用 74 芯片缓冲缓冲

解决：直接用数据线，不缓冲

4. 7 段数码管不亮或者只亮一个

解决：没有搞清楚他的工作方式，是并发的，不是并行的，要分时复用，要写一个循环程序。

```
    output(B, 0b11111110  );
    output(A, hour_1_LED);
    delay(1);
    output(B, 0b11111101  );
    output(A, hour_2_LED);
    delay(1);
    output(B, 0b11111011  );
    output(A, minute_1_LED);
    delay(1);
    output(B, 0b11110111  );
    output(A, minute_2_LED);
    delay(1);
    output(B, 0b11101111  );
    output(A, second_1_LED);
    delay(1);
    output(B, 0b11011111  );
    output(A, second_2_LED);
    delay(1);
    output(B, 0b10111111  );
    output(A, heat_1_LED);
    delay(1);
    output(B, 0b01111111  );
    output(A, heat_2_LED);
    delay(1);
```

这段代码是让每一个数码管亮一次，然后再外面循环调用这个程序才能看起来是同时亮的。

然后重中之重就是要 delay

```
/*
函数类型：延时函数
功能：延时
*/
void delay(int t)
{
    int i = 0;
    for (i = 0; i < t * 120; i++);
}
```

就是让 cpu 空跑一下，如果不延时一下会导致所有芯片彻底停止工作（具体原因还不知道为啥）

但是 delay 多久可能取决于电脑的性能。不能 delay 太久那样会导致显示会一个一个闪，看起来就不是同时显示数字了。

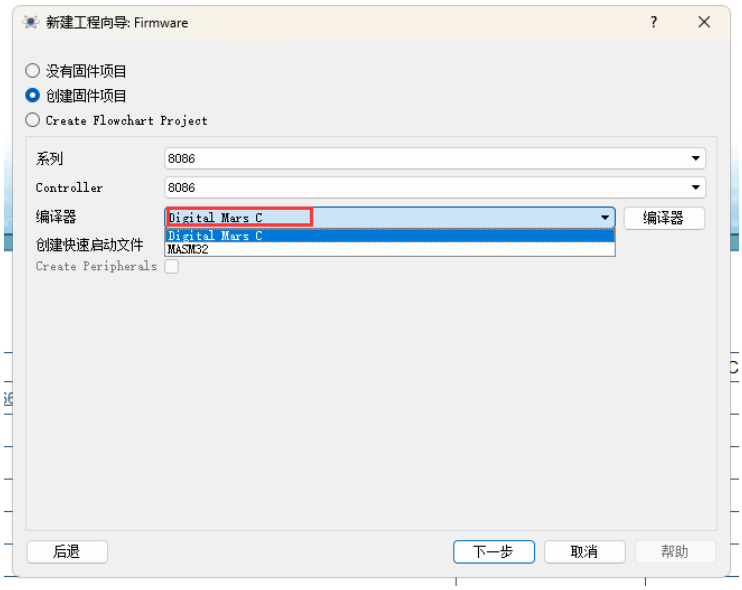
5.7 段数码管的片选可以通过数据片选

6.汇编编译立即数无法用 16 进制表示出错

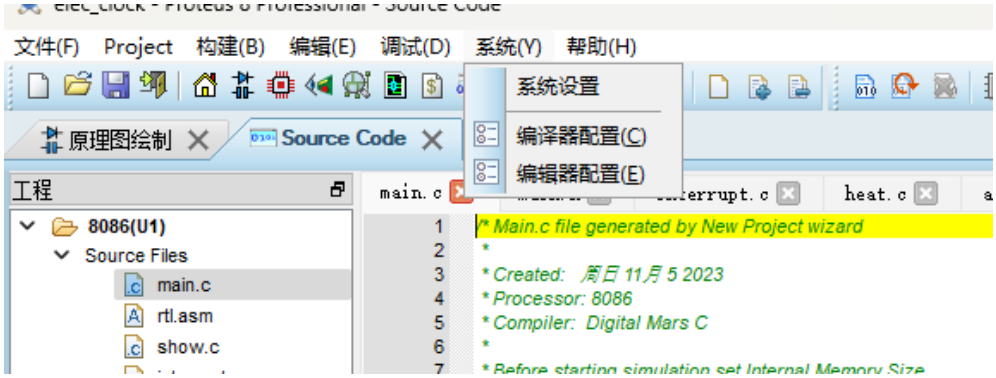
解决：因为 16 进制数如果是字母开头的，编译器会认为他是个变量，要添加一个 0 才能被识别为立即数

7．汇编的函数调用太过于麻烦

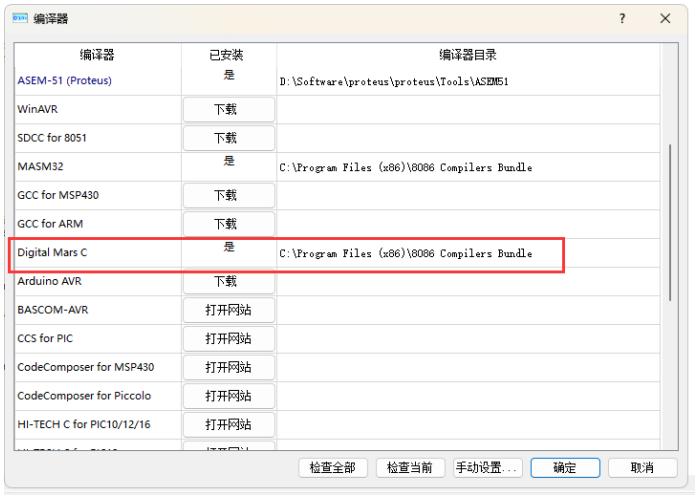
解决：用 c 语言建项目



在新建项目时选择 DigitalMarsC 编译器



选择编译器配置



下载安装这个编译器

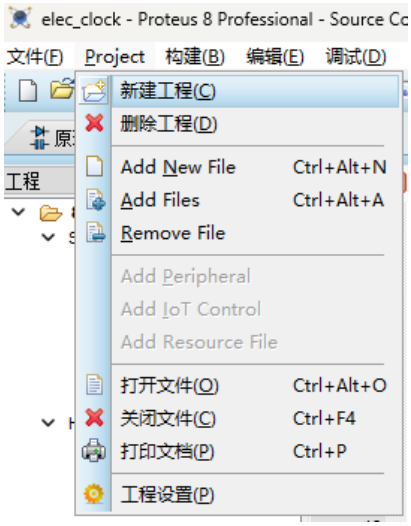
8.C 语言无法编译语法问题

解决：镶嵌在 c 语言的汇编的立即数语法和之前汇编立即数不一样

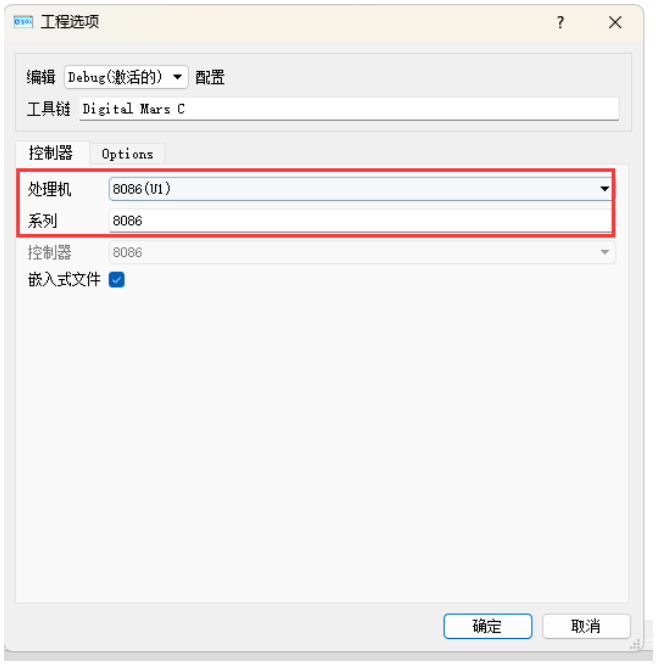
C 语言 16 进制使用 0x 开头的，且末尾不加 H

9.C 语言无法实时模拟（fatal simulation error s encountered）

解决：项目设置问题



点工程设置



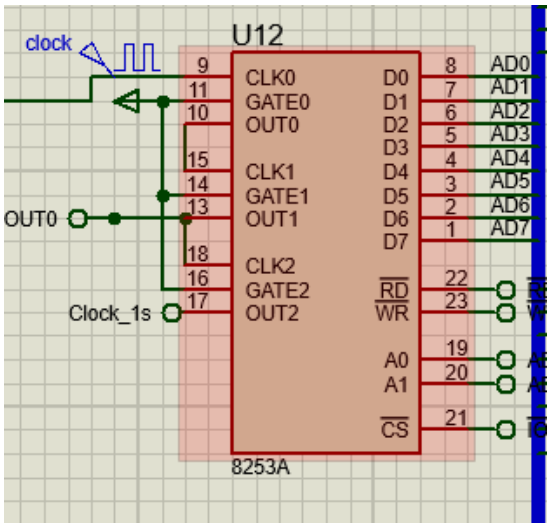
然后这两个要选对

10.8255A 不工作

解决：8255A 控制字只能在一开始设置一次，不能没回 show 一次就设置一次，不然会导致 8255A 无法正常工作

11.8253 仅使用一个无法实现完整足够的分频

使用通道级联的方式



12.8253A 不工作

解决：老师上课讲的 8253 控制字其中 x 不能用 0 得用 1 不然无法识别，无法正常工作

13.如何判断分频后是不是 1 秒

解决：使用示波器，但是示波器一直显示不是 1s，但是最后直接用频率查看后才是 1s，虽然显示出现的不是 1s 但是看两点只差的数据是 1s

14.Cpu 一直不响应中断（卡了好久好久）

解决：原来项目生成的时候自动给生成了关中断的代码注释掉就好了

```
.8086
.stack
.code
extern _main:near
.startup
    ;cli            ; interrupt disable
    call near ptr _main
endless:
    jmp  endless
.data
public    __acrtused        ; trick to force in startup

__acrtused = 9876h     ; funny value not easily matched ; ; in SYMDEB

END
```

15.7 段数码管显示不正常：：

解决：每个显示中要 delay 一下

16. proteus 出现 No model specified for D1 和 Simulation FAILED due to partition analysis error(s)错误

解决：右键编辑器件，点附加层次模块（但是这个没有解决问题，会导致芯片不工作）最终解决是更换正确的芯片

总结出的规律：

17.C 语言分模块写，调用其他文件的函数，不用提前声明函数

18.proteus8086 这个芯片中内部已经整合了内存不用再用内存芯片了（问的老师）

19.我们写的指令就相当于再写一个非常非常小的功能不健全的操作系统，放在 cpu 上一个一个指令执行

20.总嵌入汇编，直接用_asm{}指出然后汇编里可以直接使用 c 语言中的变量

21.用 C 语言编写的实质，就是封装了汇编的指令，实际上还是基于汇编的。

22.out 指令的流程：先向总线发送地址然后发送数据，实现了总线的复用

23.in 指令的流程：先向总线获取地址数据，然后获取数据，并不会阻塞

24.使用.h 文件代码更加简洁高效

五、 总结与感悟

“真正的勇士，敢于直面惨淡的人生，敢于正视淋漓的鲜血”，鲁迅先生如是说，当我们在解决无数问题的时候，我的脑中，突然蹦出了小时候学到的鲁迅先生的教诲，原来先贤的声音早已刻在了我们记忆深处，只不过有时我们并未察觉。虽然我们没有像刘和珍先辈一样抛头颅洒热血，但是我们普通人也能在她事迹中受到鼓舞，并且受到鲁迅先辈的支持和勉励。很难想象如果一个人的命运像我们的项目一样坎坷的的话，她该如何面对，虽然我们的项目的坎坷只是九牛一毛，但是我们已经痛不欲生了。

我们逐渐在解决问题中成长了，收获了，我们已经培养了一定的硬件思维和硬件编程能力了，最重要的是我们培养了解决问题的自信，相信这个项目对我们的影响会十分深远。

六、 待完善和发展的地方

如果你看到了我们的实验项目，你应该可以很轻松的走过我们走过的路，但是也不用担心，剩下的路还有很长可以给你来探索。下面我们来给出一些可以基于我们项目来发展的地方。

1. 闹钟铃声的更改，改为用 8253 实现的音乐播放
2. 时间维度的扩展，增加年、月、日（甚至可以增加农历）
3. 增加闹钟的数量（你应该也发现了，谁的闹钟只能定一个闹钟的啊）
4. 增加闹钟的功能（比如可以倒计时、正向计时等）
5. 其他（比如增加设置自定义闹铃等等）