

微机期末大作业实验报告

1. 项目名称

可调整的多功能电子时钟

2. 项目概述

2.1 设计思路

在“微机原理与接口技术”这门课程的学习中，学到 8253 可编程的定时计数器那一节中有涉及到利用 8253 制作电子时钟的内容，当前网络中的电子时钟微机设计中，有很大的局限性，并不能满足用户的需求，所以我想在此基础上，结合本学期所学内容，利用 8259 中断控制器来进一步扩大电子时钟的功能。

8259 的固定中断优先级为 IR0 到 IR7 由高到低，并且可以设置主片优先级高于从片，操作控制字 OCW1 可以设置屏蔽某片 8259 上 IR0 到 IR7 的中断；所以我的想法是利用主片的中断服务程序，设置从片的 OCW1 的中断屏蔽字，从而达到在计数的时候，从片仅接受来自 8253 的中断完成更新计时，在暂停计数的时候可以对计时进行修改，控制等操作。

此外在我利用两片 6264 来模拟构建奇偶存储体，用于存储即刻计时的时分秒，它对应的 CPU 内存地址范围为 F0000H-F3FFFH。

2.2 项目功能

- (1) 实现循环计时功能即从 00:00:00 开始，到 23:59:59，在下一秒到来时自动更新到 00:00:00。
- (2) 实现计时暂停。
- (3) 在暂停的时候实现选择时，分，秒的置数修改
- (4) 在暂停的时候实现选择时，分，秒的自增修改（自动进位）
- (5) 在暂停的时候实现选择时，分，秒的自减修改（自动降位）
- (6) 加快计时速度
- (7) 减慢计时速度
- (8) 恢复计时初始速度

2.3 按键简述

(1) 控制台端共设置了 9 个按钮，1 个开关，一个两位二进制输入端，一个八位二进制输入端。

(2) 开关控制计数总状态，启动前需为断开状态，断开则强制停止计数，各个按钮的功能已经在电路图中详细标出。

(3) 两位二进制输入端用于控制操作位的选择，00 控制秒位，01 控制分位，10 控制时位，11 为无效控制，这里软件中默认设置不在 00,01,10 中的为控制秒

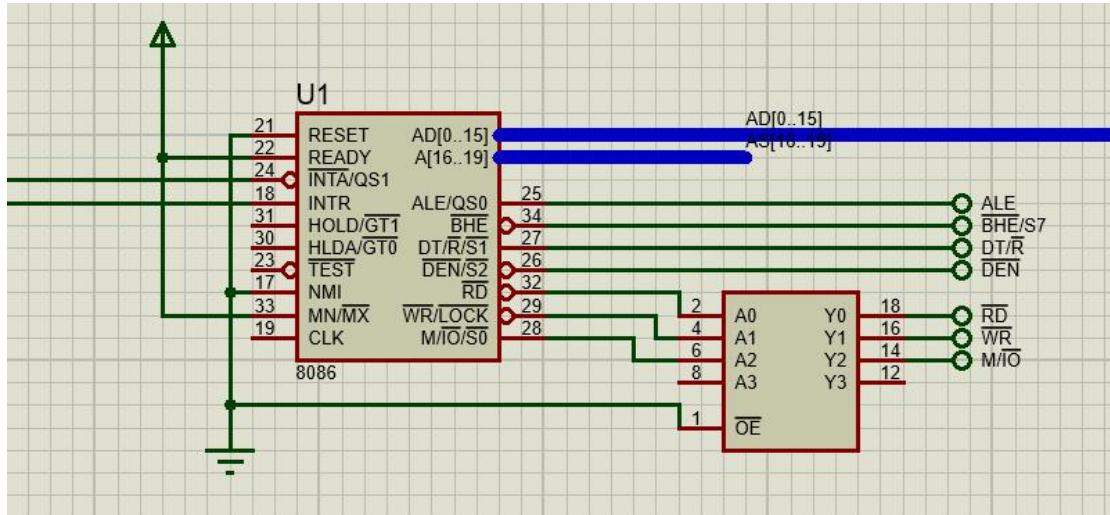
位。

(4) 八位二进制用于置数输入，为了方便显示，用户输入的数按照十进制处理，对于非十进制数，软件中进行十进制调整，对于大于 59 的十进制数，软件中将之强制转换为 59H 处理。

3. 硬件设计

3.1 8086 引脚连接

(1) 该部分电路图

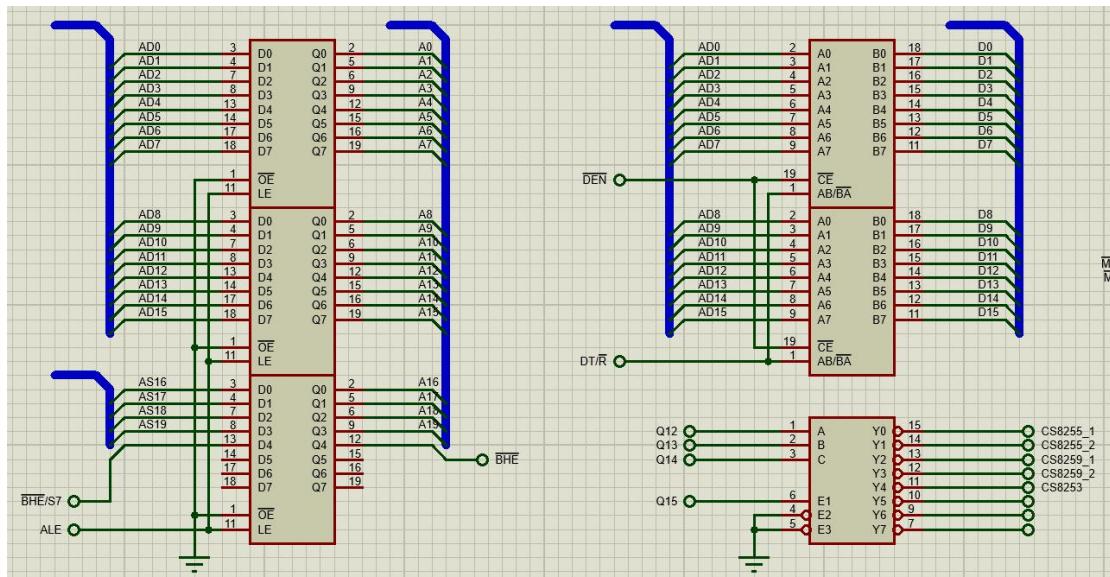


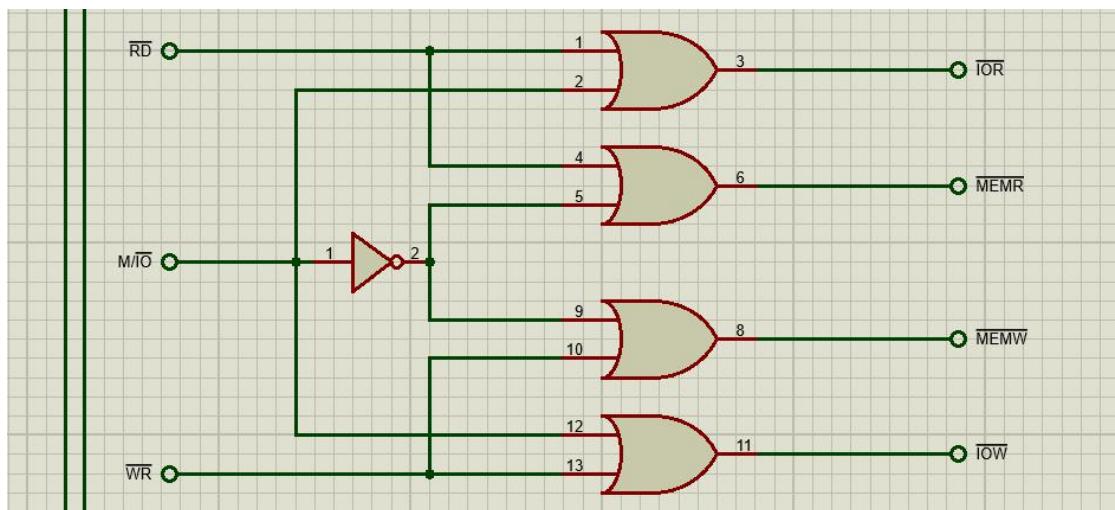
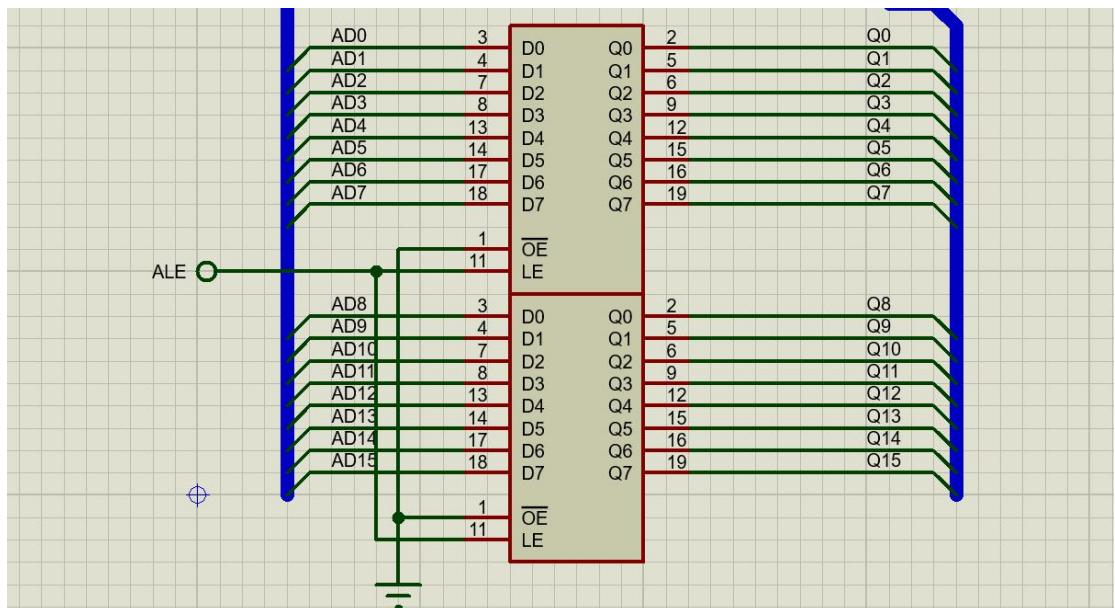
(2) 具体描述

8086CPU 工作在最小模式下，INTR 与非 INTA 信号与 8259 中断控制器对应引脚相连。

3.2 8086 信号锁存与变换

(1) 该部分电路图





(2) 具体描述

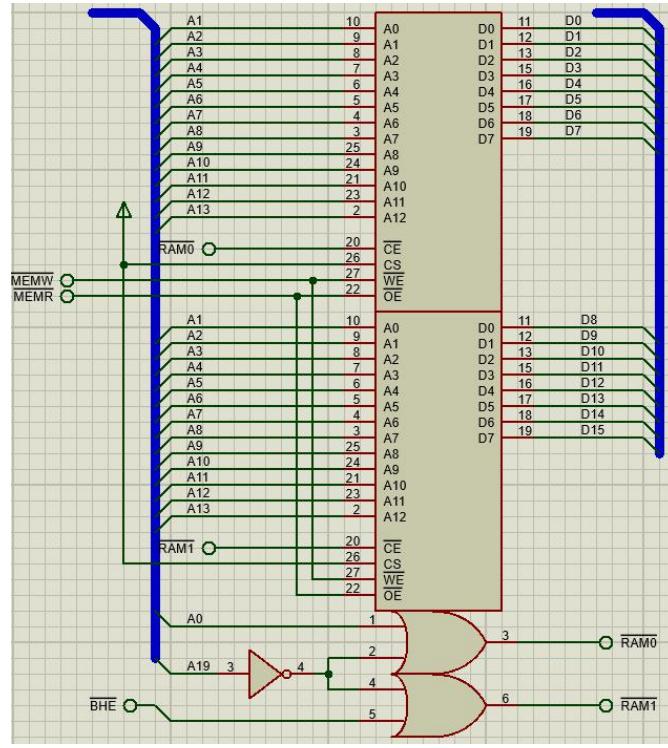
第一张电路图主要用于 6264 存储体构建，在这里实现了分时复用的 20 位地址线与 16 位数据线分别锁存，并且数据端使用了 74LS245 双向的三态总线转换器，根据 CPU 发出的 DT/非 R 信号实现输入输出。同时也包括了一部分根据地址线高四位译码来片选各个端口的 71LS138 译码器。

第二张电路图实现了对前 16 位的信号锁存，用于各个接口的 8 位输入端，这样较第一张图更加方便快捷。

第三张图实现了对 CPU 的读写信号以及读写存储体/IO 接口信号的处理，将其分为 IO 读写，存储体读写四种信号，用于之后的存储体读写与各个端口读写信号。

3.3 6264 存储体设计

(1) 该部分电路图



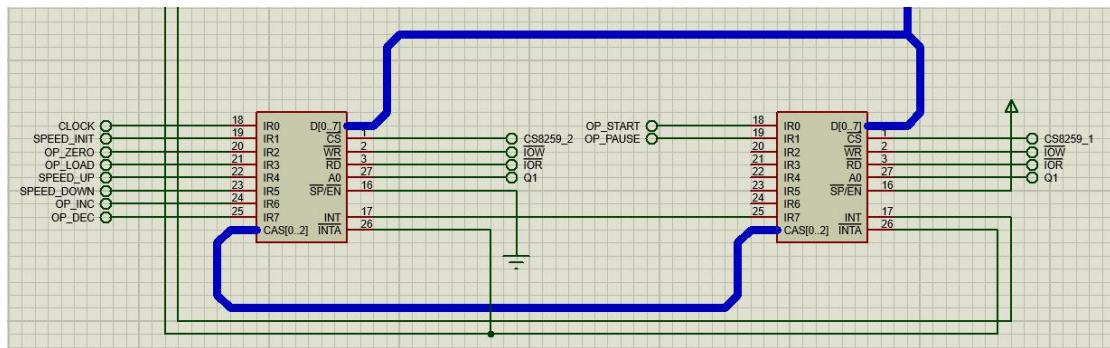
(2) 具体描述

由两片 6264 组成，靠上一片为偶存储体，靠下一片为奇存储体，通过 8286 发出的非 BHE 信号与地址线的 A0 来确定使用哪一个存储体，编码含义如下。

非 BHE	A0	数据总线使用情况
0	0	同时读写高低两个字节
0	1	只读写奇地址库
1	0	只读写偶地址库
1	1	无效

3.4 8259 中断控制器设计

(1) 该部分电路图

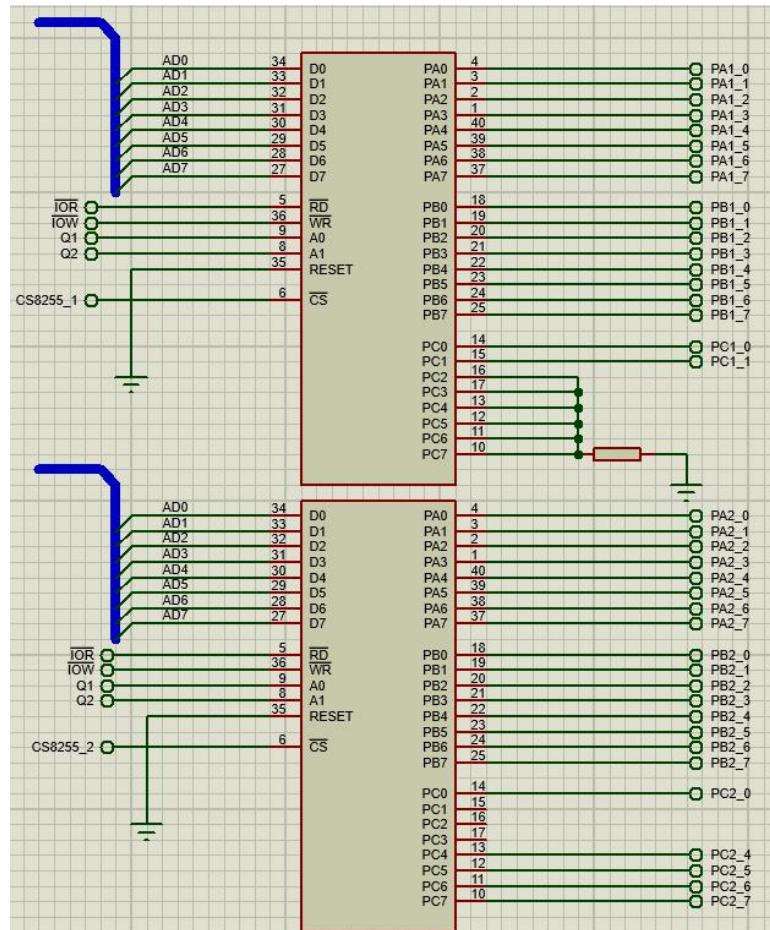


(2) 具体描述

左边 8259 为主片，右边 8259 为从片，从片对应的级联控制字为 111 (IR7)，主片对应着计数开始与暂停的中断请求，计数从片对应着计数中断请求以及多种控制中断请求。

3.5 8255 并行传输接口设计

(1) 该部分电路图

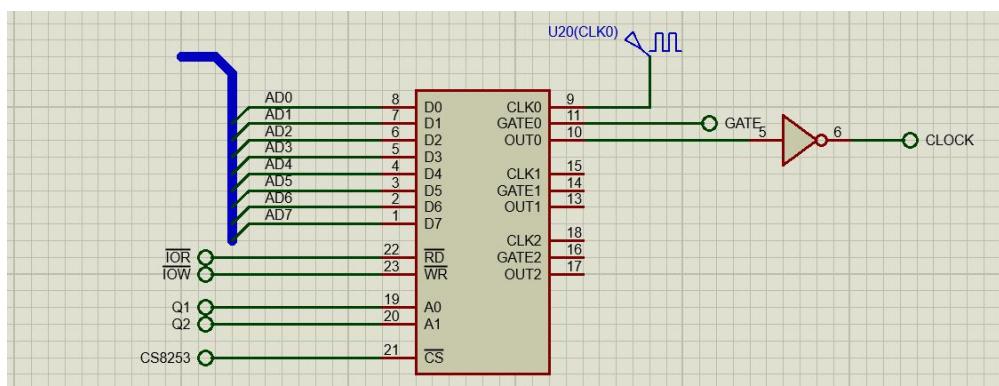


(2) 具体描述

靠上一片 8255 的 AB 端口均为输出端，与时分秒 BCD 显示器相连，C 端口为输入端，与控制台用户控制选择输入端相连；靠下一片 8255 的 AC 端口为输出端，A 端口与时分秒 BCD 显示器相连，C 端口与状态信号 LED 灯，B 端口为输入端，与控制台用户置数输入端相连。

3.6 8253 定时计数器设计

(1) 该部分电路图

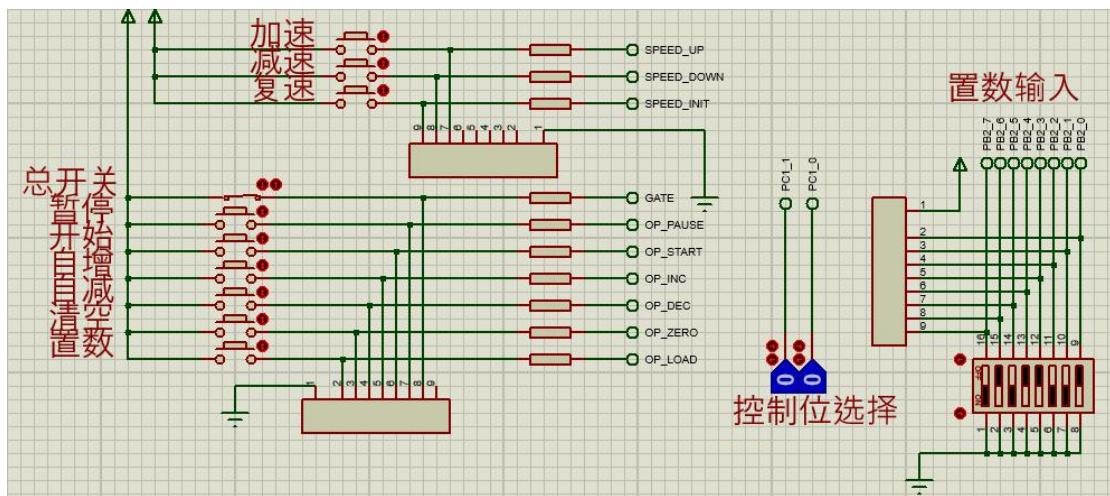


(2) 具体描述

只使用计数器 0，时钟频率为 35Hz，计数 OUT 端输出低电平反向后作为高电平的中断请求，与 8259 从片时钟中断计数请求相连，GATE 端与控制台的开关相连，作为系统开始计数的总开关（物理启动）。

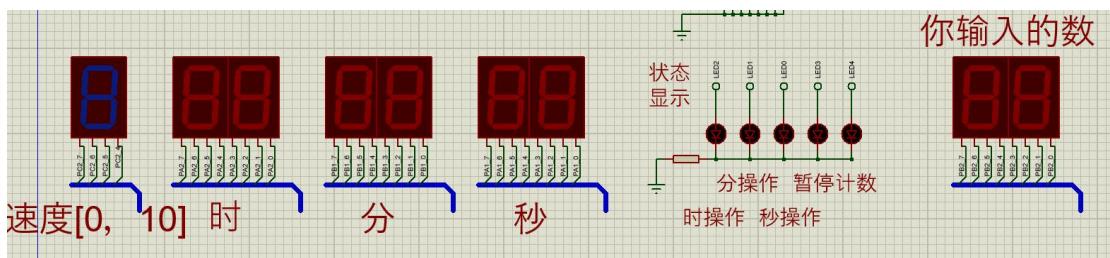
3.7 用户控制台设计

该部分电路图



3.8 状态及数据信息显示

该部分电路图



4. 软件设计

4.1 数据段

程序中定义了 DATA 数据段，其中存放了两字节的内容。

FLAG 为标志位，用于显示当前计数状态，00H 表示当前正在进行计数，01H 表示当前计数暂停，程序中默认状态为暂停计数，即 01H。

CLOCK_SPEED 用于存放当前 8253 的计数初值（也可以理解为计数速度），程序中默认为 0AH，提速，降速，恢复操作均会修改它的值，允许修改的范围为[05H, 0FH]。

4.2 各个接口及数据地址分配

本项目共使用了两片 8255，两片 8259，一片 8253。

8255_1 的地址为 **8000H, 8002H, 8004H, 8006H, 8255_2** 的地址为 **9000H, 9002H, 9004H, 9006H; 8259_1** 的地址为 **A000H, A002H, 8259_2** 的地址为 **B000H, B002H; 8253** 的地址为 **C000H, C002H, C004H, C006H**。

时分秒的数据存储在 **6264** 存储体重，各为两个字节（避免出现访问奇偶存储体的错误），秒存储地址为 **8000H:0100H**，分存储地址为 **8000H:0200H**，时存储地址为 **8000H:0300H**。

4.3 编程初始化 8259

8259_1 为主片，**ICW1** 设置上升沿触发，多片级联，要写 **ICW4**；**ICW2** 设置中断向量码高五位为 **00100H**；**ICW3** 设置 **IR7** 连接了从片；**ICW4** 设置特殊嵌套，非缓冲方式，非自动 **EOI**；**OCW1** 初始化时不屏蔽 **IR0-IR7**。

8259_2 为从片，**ICW1** 设置上升沿触发，多片级联，要写 **ICW4**；**ICW2** 设置中断向量码高五位为 **00110H**；**ICW3** 设置从片识别码为 **111**（对应 **IR7**）；**ICW4** 设置一般嵌套，非缓冲方式，非自动 **EOI**；**OCW1** 初始化时不屏蔽 **IR0-IR7**；**OCW1** 设置仅屏蔽 **CLOCK** 中断请求——时钟中断服务子程序。

在之后的中断程序操作里会涉及 **OCW1** 的重新设置，**8259_2** 从片会在暂停中断服务子程序中，设置仅屏蔽 **CLOCK** 中断请求，在恢复计数中断服务子程序中，设置仅不屏蔽 **CLOCK** 请求。

编写中断向量表的代码请详见附录。

4.4 编程初始化 8255

8255_1 的 **AB** 端均为方式 **0** 下的输出端，**C** 端为方式 **0** 下的输入端；**8255_2** 的 **AC** 端均为方式 **0** 下的输出端，**B** 端为方式 **0** 下的输入端。

4.5 编程初始化 8253

8253 的控制字为 **00110100B**，启用计数器 **0**，先读写计数器低八位，在读写高八位，工作在方式 **2**，采用二进制计数。并且设置计数初值为 **000AH**。

4.6 中断服务子程序

以下是所有的中断服务子程序，具体代码请见附录，这里展示流程图，其中 **OP_START** 与 **OP_PAUSE** 为主片上的中断服务，其他为从片上的中断服务，其中编号为 **4** 到 **10** 的中断被定义为“时钟调整中断服务程序”。

(1) **OP_START**（恢复计数中断服务子程序）

该中断程序用于结束暂停计数，使得计数开始，并且屏蔽所有的时钟调整中断服务程序，修改状态标志位为 **00H**。其流程图如下：

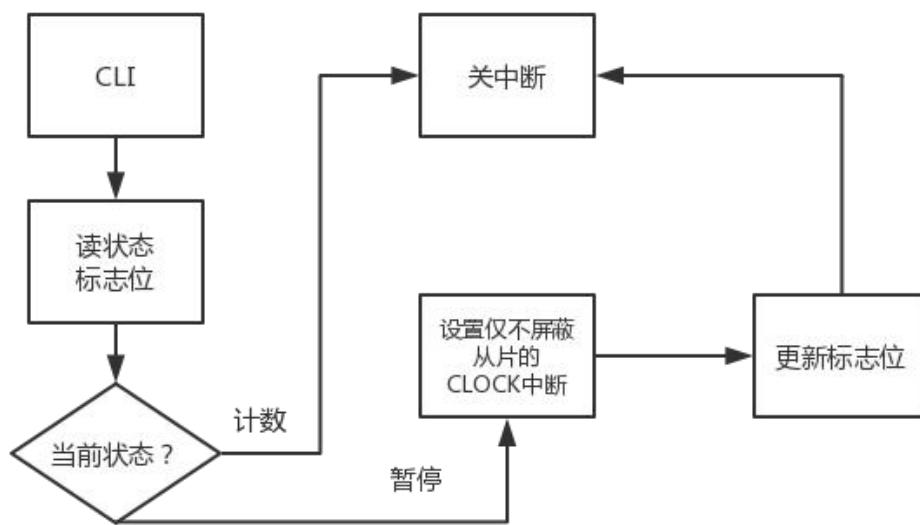


图 4-1 OP_START 中断服务程序流程图

(2) OP_PAUSE (暂停中断服务子程序)

该中断服务程序用于暂停计数，并且设置仅屏蔽 CLOCK 中断请求，修改状态标志位为 01H，其流程图如下：

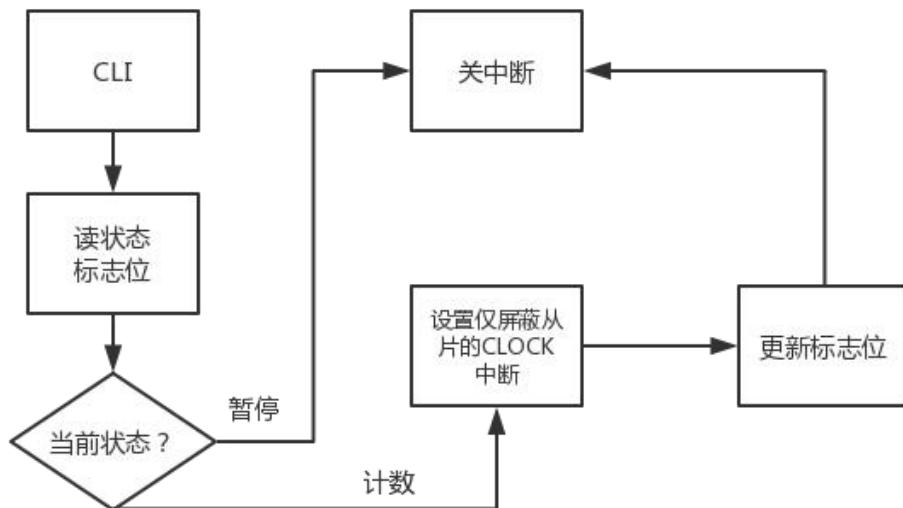


图 4-2 OP_PAUSE 中断服务程序流程图

(3) CLOCK (时钟中断服务子程序)

该中断服务程序用于计时状态，当 8253 的一个计时请求达到时，对存储体内存储的数据加一秒，并且做标准化处理（保证循环计时，保证时分秒各位的规范性）。其流程图如下：

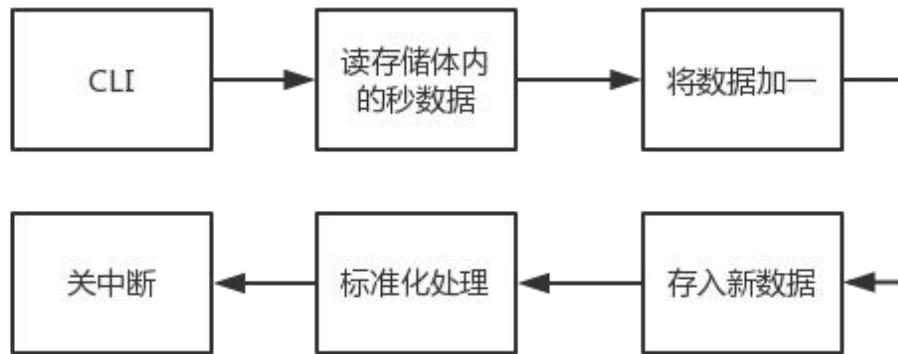


图 4-3 OP_CLOCK 中断服务程序流程图

(4) OP_INC (自增中断子程序)

该中断服务程序用于暂停状态，它会根据用户选择的控制方式，对计时的时，分或秒进行加一操作，并自动做标准化处理。其流程图如下：

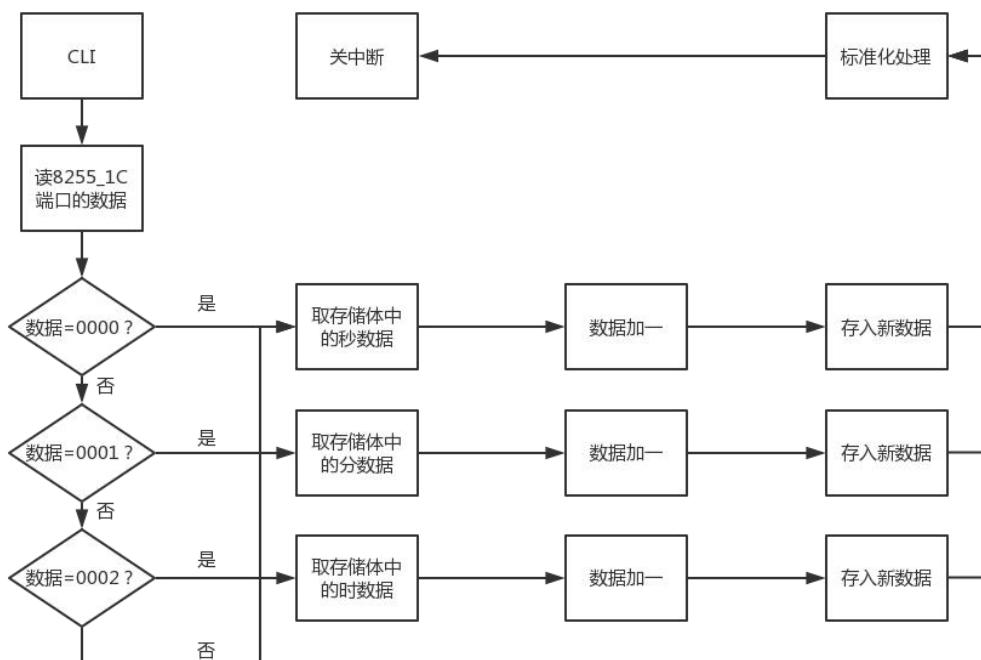


图 4-4 OP_INC 中断服务程序流程图

(5) OP_DEC (自减中断子程序)

该中断服务程序用于暂停状态，它会根据用户选择的控制方式，对计时的时，分或秒进行减一操作，并自动做标准化处理。其流程图如下：

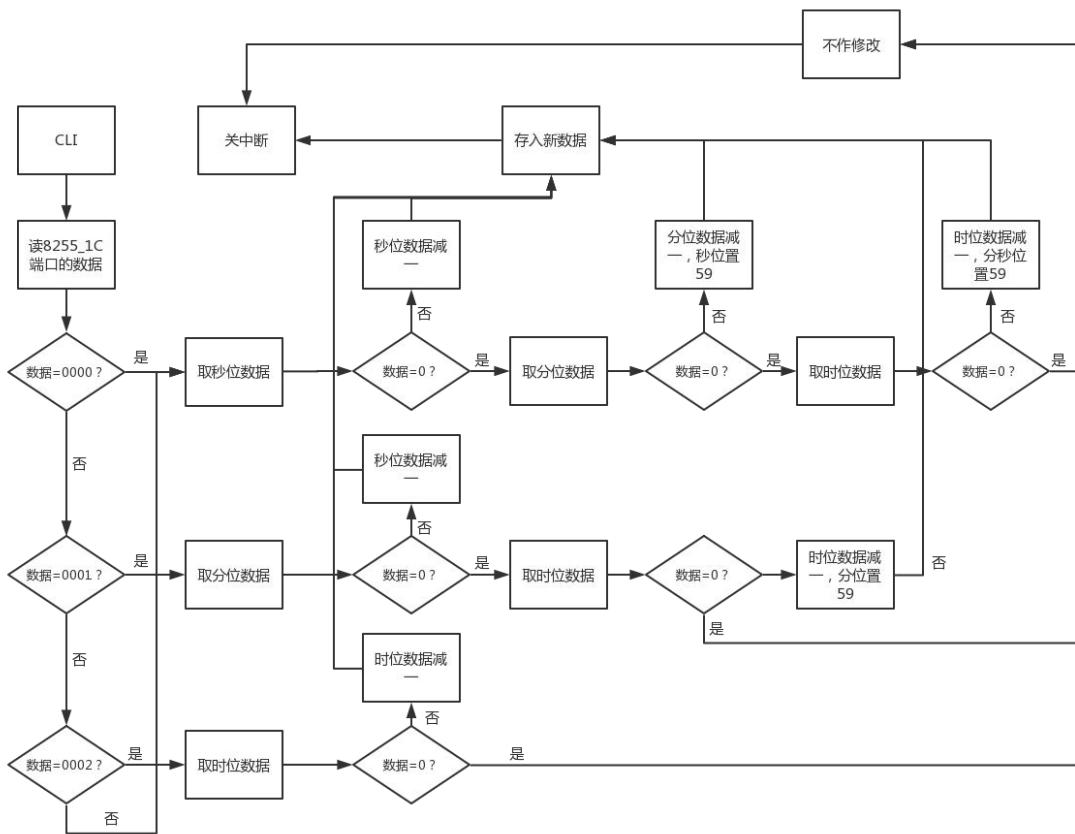


图 4-5 OP_DEC 中断服务程序流程图

(6) SPEED_UP (提高速度中断子程序)

该中断服务程序用于暂停状态，它会将计时速度加快一个计量单位，已经达到最大值，则自动恢复到初始速度。

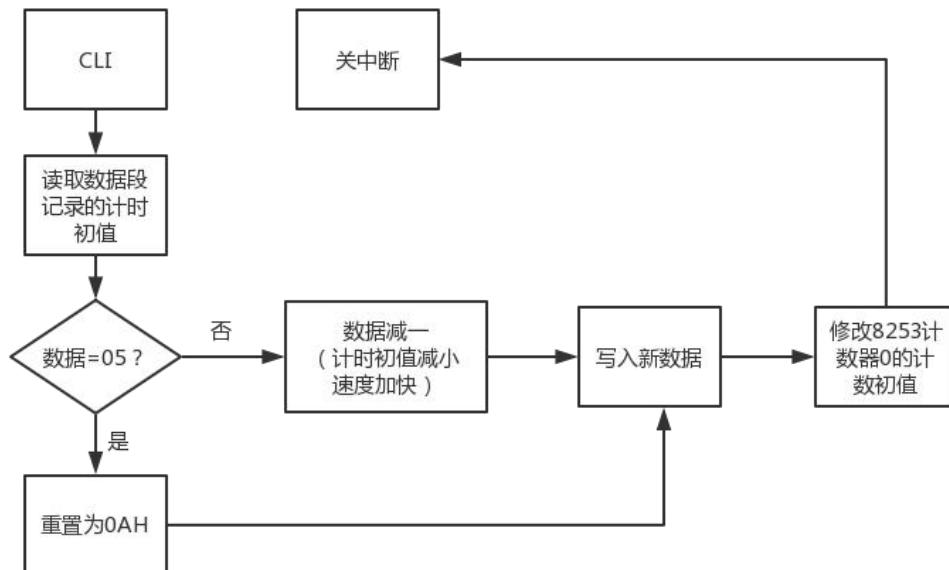


图 4-6 SPEED_UP 中断服务程序流程图

(7) SPEED_DOWN (降低速度中断子程序)

该中断服务程序用于暂停状态，它会将计时速度减小一个计量单位，已经达到最小值，则自动恢复到初始速度。

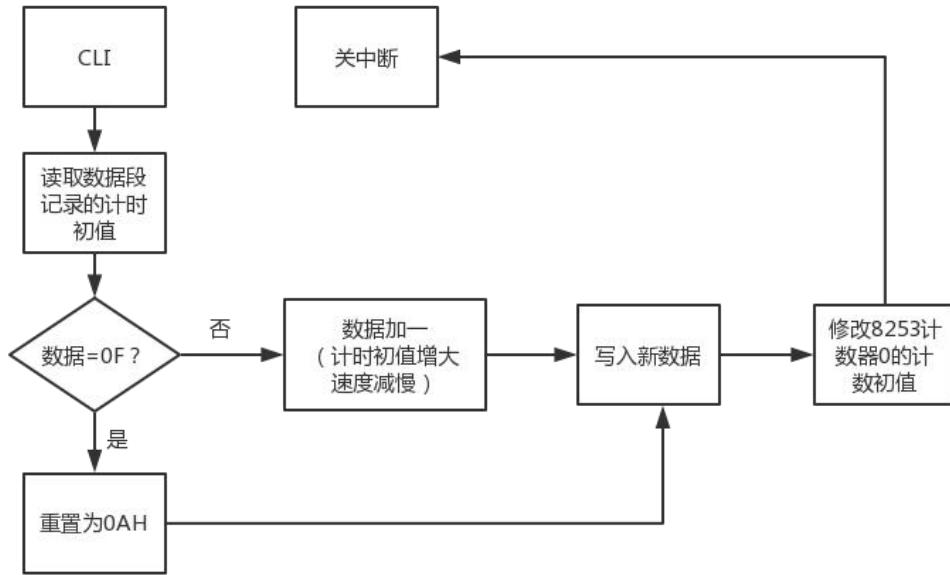


图 4-7 SPEED_DOWN 中断服务程序流程图

(8) SPEED_INIT (复位速度中断子程序)

该中断服务程序用于暂停状态，它会将计时速度恢复到初始速度。

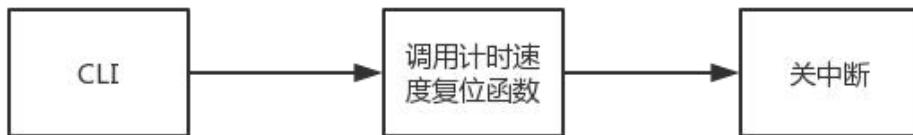


图 4-8 SPEED_INIT 中断服务程序流程图

(9) OP_ZERO (复位中断子程序)

该中断服务程序用于暂停状态，它会将当前计时值恢复到 00:00:00。

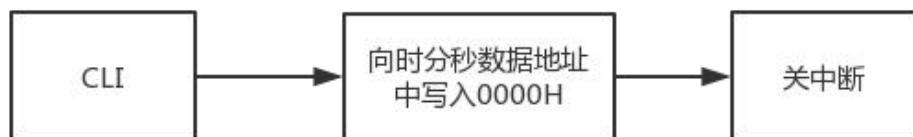


图 4-9 OP_ZERO 中断服务程序流程图

(10) OP_LOAD (置数中断子程序)

该中断服务程序用于暂停状态，它会将控制端输入的数据存入指定的时、分或秒位，若大于某位的最大值，则调整为 59H (时位为 23H)。

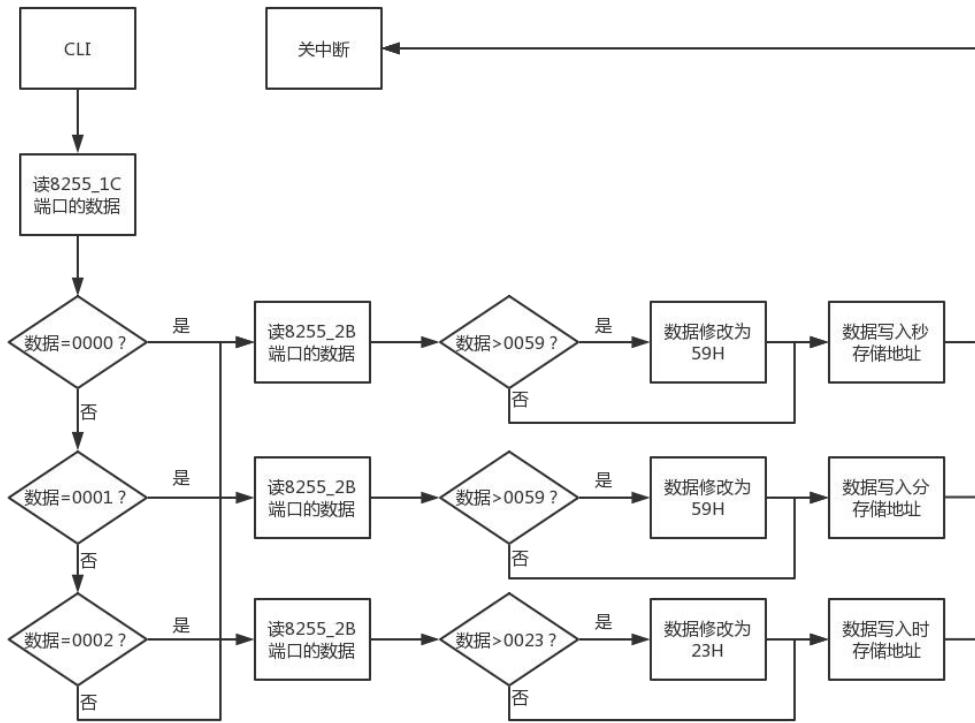


图 4-10 OP_LOAD 中断服务程序流程图

4.7 其他函数程序

(1) INITCLOCK (计时速度复位函数)

该函数用于将数据段的 SPEED_CLOCK 值修改为 0AH，其汇编代码如下：

INITCLOCK:

```

MOV BX,OFFSET CLOCK_SPEED
MOV AX,CONDATA
MOV AH,00H
MOV DS:[BX],AL;更新后的计时速度写入 data 段
MOV DX,TCON0;修改计数初值
OUT DX,AL
MOV AL,AH
OUT DX,AL
RETF

```

(2) SAMPLE (时钟标准化函数)

该函数用于将当前存储的时分秒数据标准化，因为整个计时过程中只会出现加一进位的情况，所以这里只考虑这一种标准化。

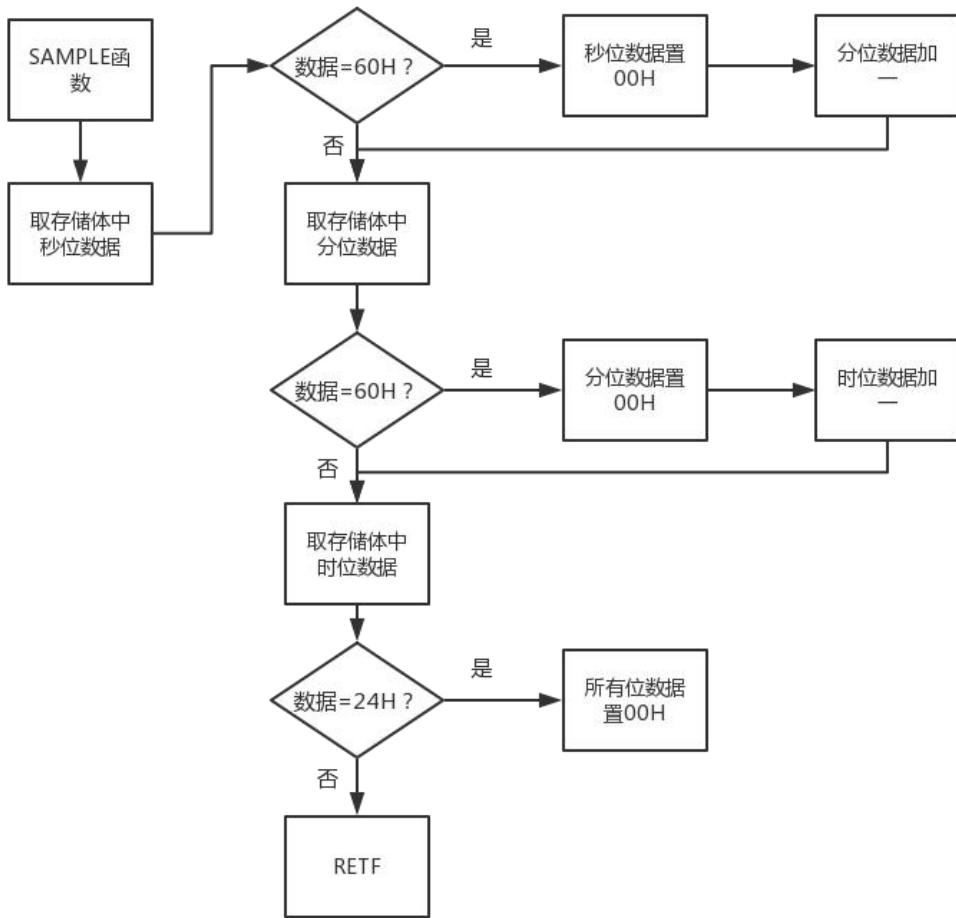


图 4-11 SAMPLE 函数流程图

4.8 系统流程图

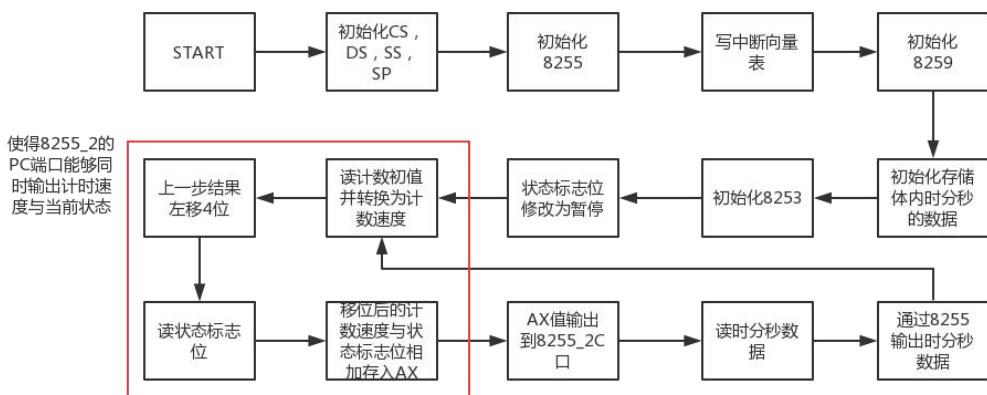
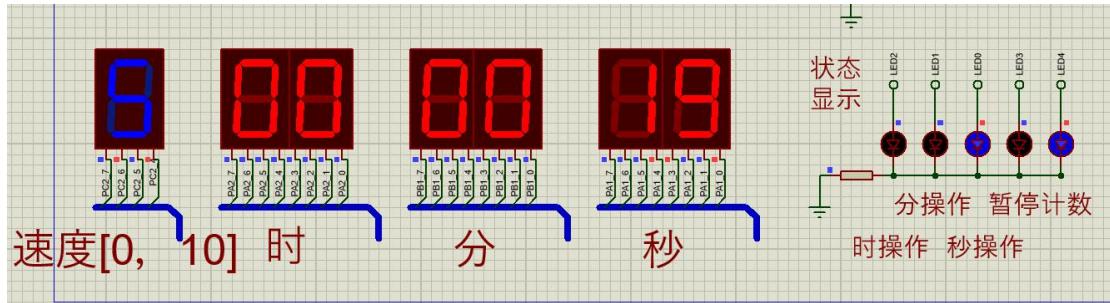


图 4-12 系统流程图

5. 系统调试

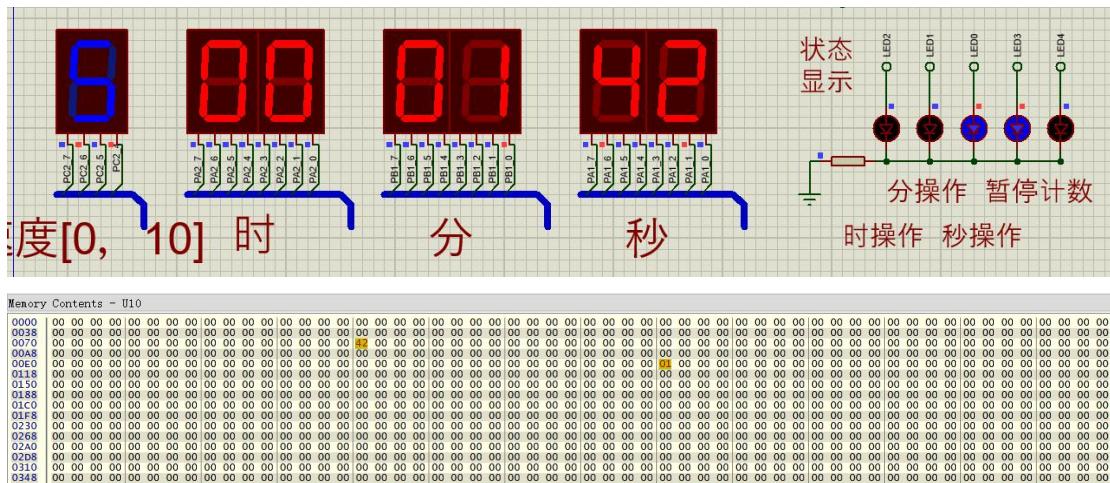
5.1 开始计时

将开关闭合后，点击下“开始”按钮，即可发现计时开始，数据显示端实时显示当前计时值。这是可以点击“自增”等时钟调整按钮，会发现并无反应，因为这里将其中断请求屏蔽了。



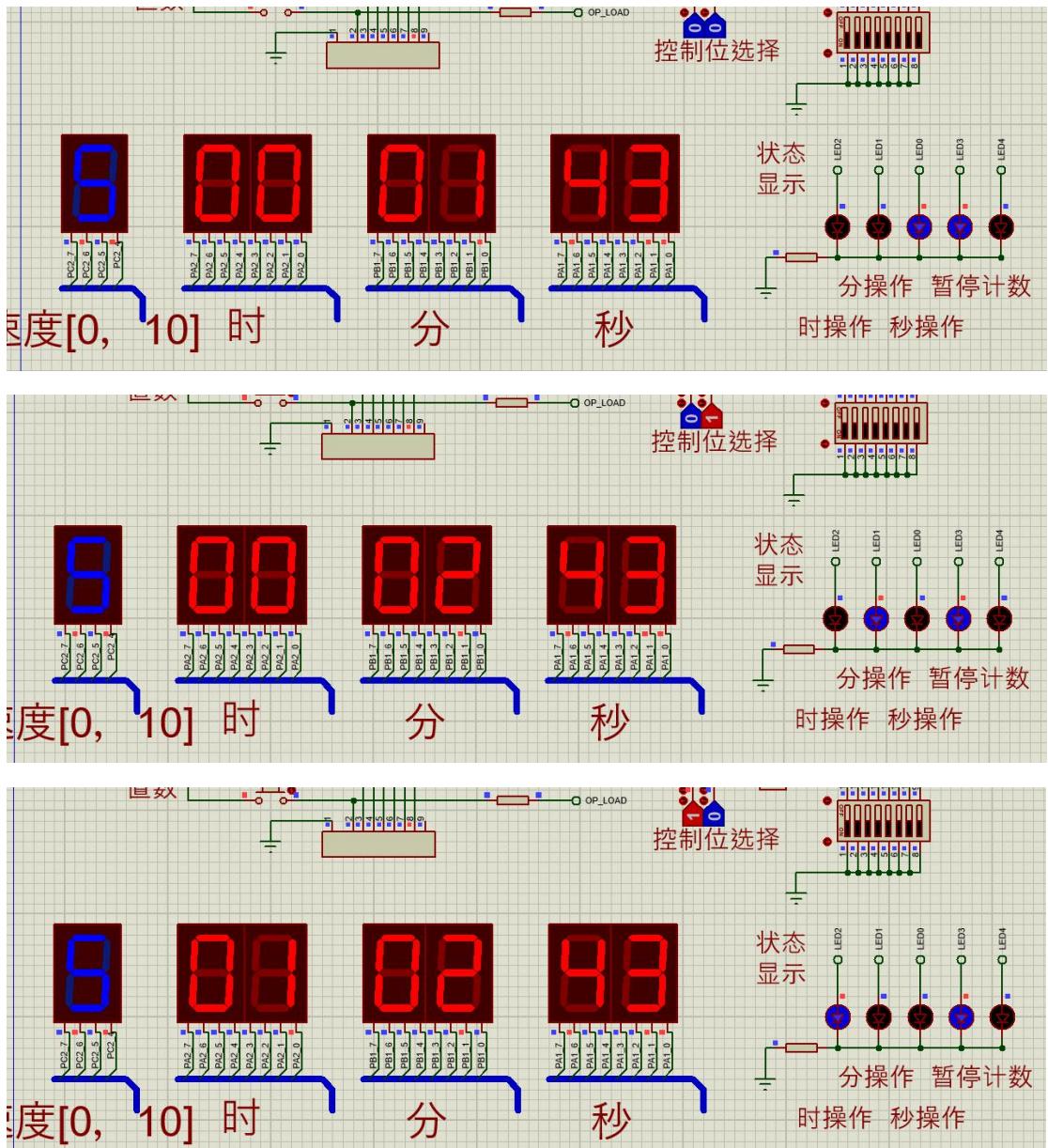
5.2 暂停计时

点击下“暂停”按钮，即可发现计时暂停，暂停仿真观察 6264 存储体内的值。



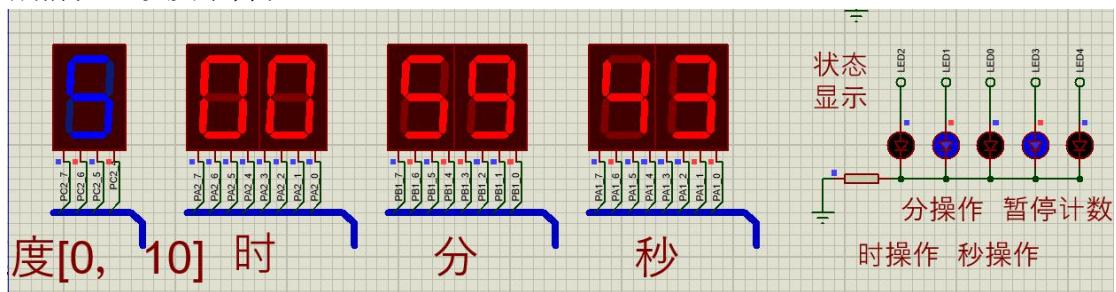
5.3 自增

在“暂停”状态下，分别设置控制位为“00”，“01”，“10”，点击下“自增”按钮，观察自增情况。



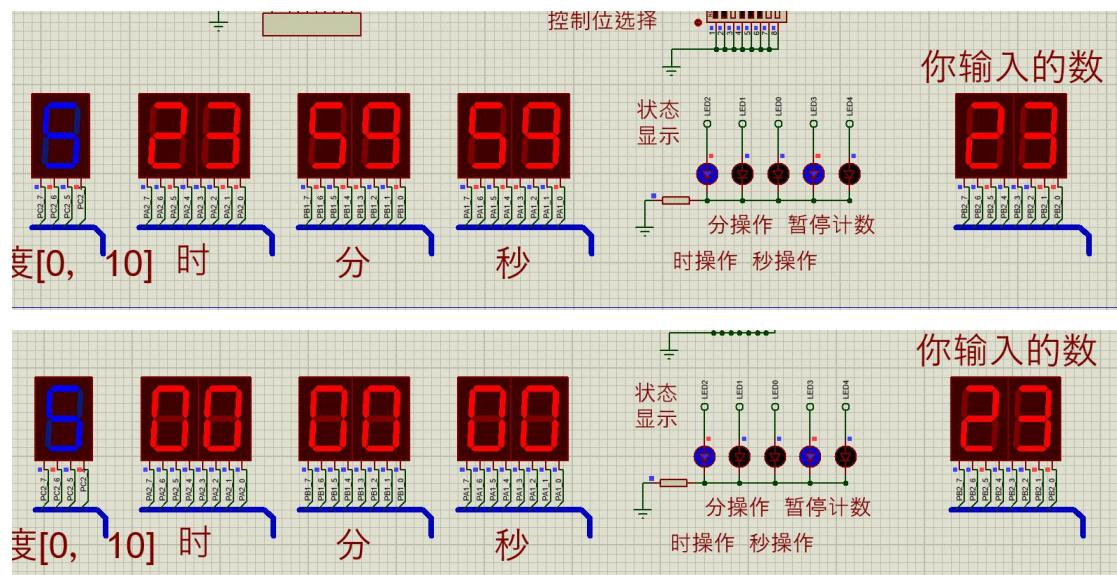
5.4 自减

在“暂停”状态下，设置控制位为“01”，点击三下“自减”按钮，观察自减情况（涉及了降位）。

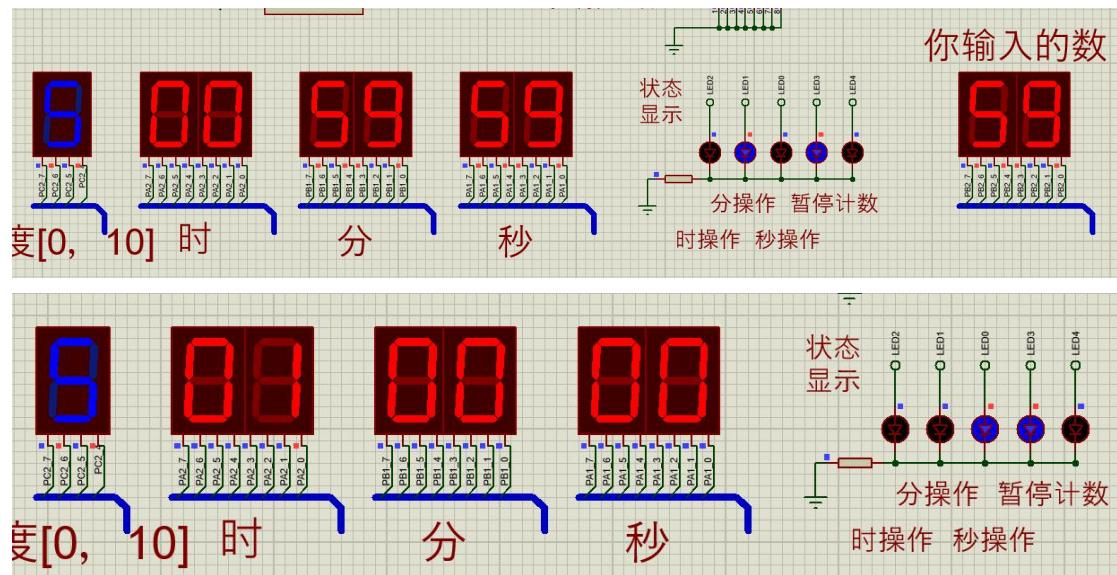


5.5 置数

在“暂停”状态下，通过控制位，设置为 23:59:59 秒后，点击自增，观察循环情况。

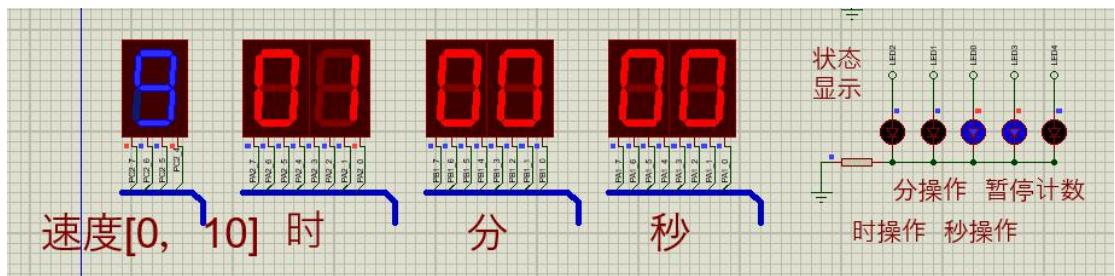


在“暂停”状态下，通过控制位，设置为 00:59:59 秒后，选择“秒”控制位，点击自增，观察进位情况。



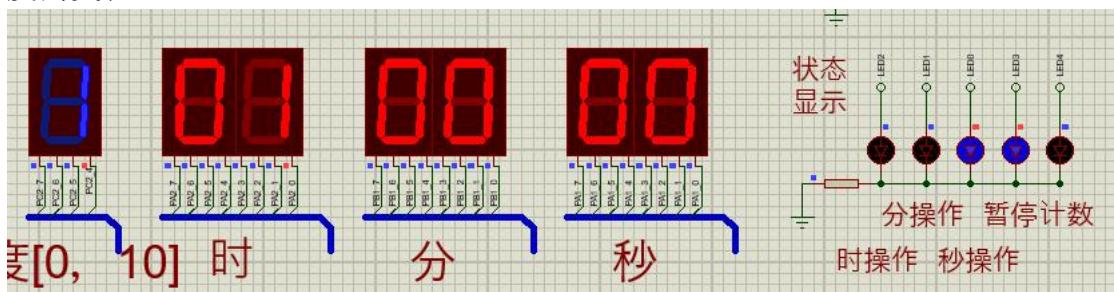
5.6 加快计时速度

在“暂停”状态下，点数次击“加速”按钮，再点击“开始”按钮，观察速度的变化。



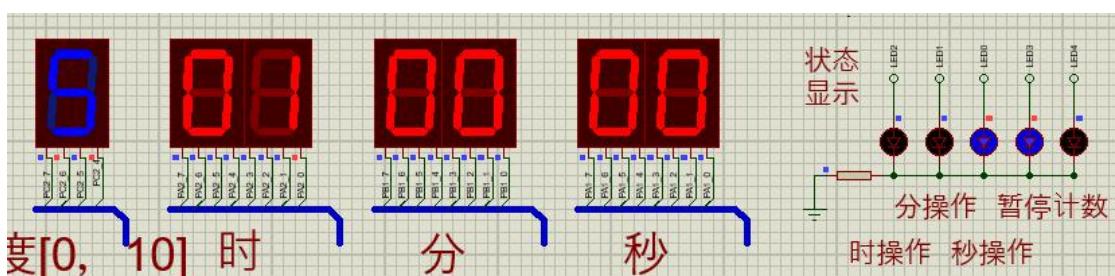
5.7 减慢计时速度

在“暂停”状态下，点数次击“减速”按钮，再点击“开始”按钮，观察速度的变化。



5.8 恢复计时速度

在“暂停”状态下，点数次击“复速”按钮，再点击“开始”按钮，观察速度的变化。



5.9 恢复计时

在“暂停”状态下，若要恢复计时，则点击一下“开始”按钮即可。

6. 心得体会

在本次大作业中，我的收获还是很多的。其中最大的收获应该是熟悉了本学期所学的各种接口芯片的功能及其使用方法。上学期在汇编学习中，我也学习了 8086 的一部分内容，但是具体对于他各种寻址方式是如何达到的，奇偶存储体是如何实现的，CPU 内部各种时序是怎样变换的还不是很了解，这学期通过微机的几次试验，特别是奇偶存储体构造的试验，我进一步加深了这方面的认识。

在设计本实验的过程中，我想尽可能多的使用各个芯片的不同的功能，所以添加了很多不同的中断请求，在一步步的设计中，我也尽力去根据功能尝试使用各个接口芯片的不同工作方式，在经过多次修改测试，尝试下，我才决定了本项目的最终设计。我尝试使用 8255 的方式 1 与方式 2 的工作方式，但是连接后

的效果不是很好，由于涉及中断信号，所以它的电路比较复杂，所以我最终抛弃了这个方案，而改换为简单的方式 0，这种输入输出比较稳定；对于 8253 的使用，我发现方式 2 是最适合本实验设计的，后续调整中我发现方式 3 也是可以的，但是这在我想要保持 4 位二进制输出计时速度时，其转换不如方式 2 方便（后来发现其实也可以通过调整时钟频率来解决），所以最终采用了方式 2；在 8259 的使用中，我认为我好的设计是灵活的运用了 OCW1 来设置中断屏蔽字，因为在本实验中，中断优先级并没有那么明显，而且用户在使用中很少去同时按下两个按钮，所以 OCW2 控制字并没有很好的用到。

关于计时速度我还有一个设计思路就是在 8253 上连接 3 种不同的时钟频率，然后 OUT 端通过锁存器控制，使得只有一个有效信号输出到对应的 CLOCK 中断，也可以设计一种时钟频率，但是工作在 3 中不同的计时方式下，也可以达到调整计时速度的目的。

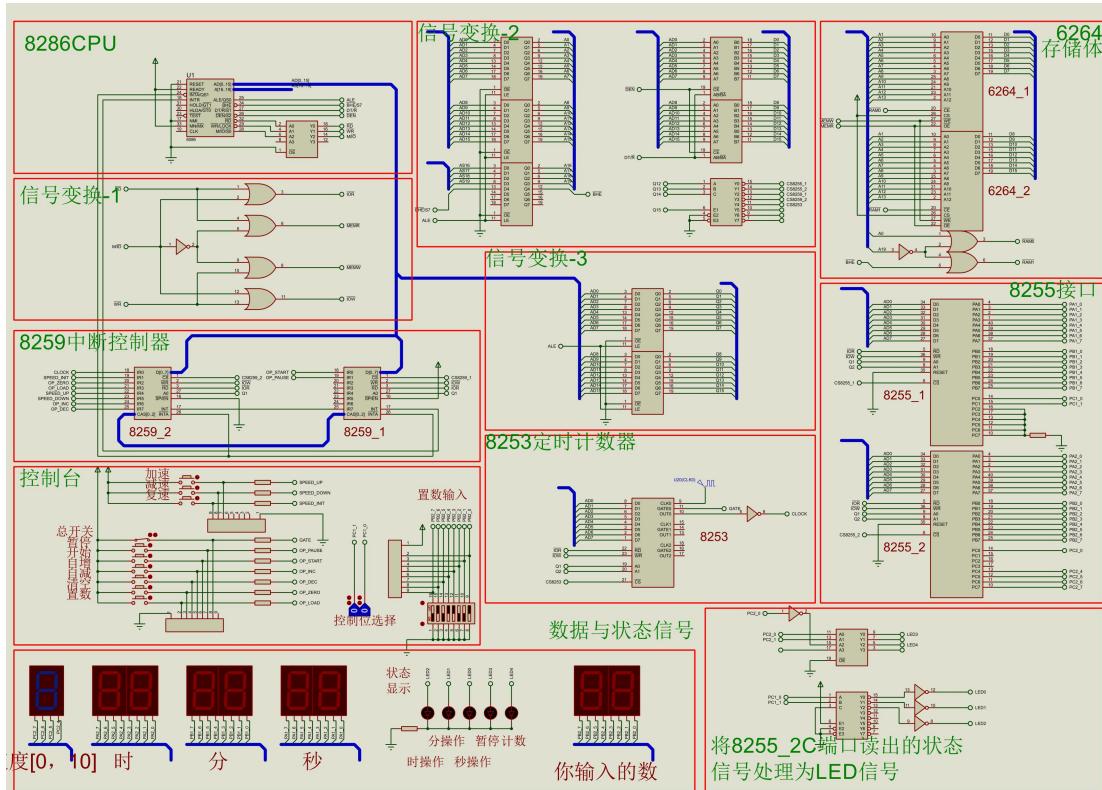
在软件设计部分，由于我在硬件设计中设计保护电路比较少，所以我在软件设计中对于用户数据输入的合法性，保证状态位只有 00H 与 01H 两种状态，保证数据存储始终为压缩 BCD 码（每次置数，自增自减都要用到 DAA 与 DAS 指令）等数据保护方面思考比较多。

总体来说，我认为本项目电子时钟的功能还是相对全面的，但是我有一点遗憾或者不足就是没有设计好一个倒计时功能而且也没有用到最后学到的 D/A 转换器与 A/D 转换器。我也尝试过增加这样一个倒计时功能，但是在之后的设计中我发现仅靠这两片 8259 与中断屏蔽字去实现倒计时与计时的同步进行还是不足，如何处理计时开始，计时暂停，倒计时开始，倒计时暂停他们不冲突，我认为还应设置优先级以及中断结束方式，而且在软件设计中还要再增加一位标志位，修改 OP_START 与 OP_PAUSE 服务程序来实现，由于最终的成果有一个部分做的不是很满意，所以最终删除了这样的一个功能。

恳请老师指正。

7. 附录

7.1 总电路图



7.2 汇编程序

;可调整的多功能电子时钟

ASSUME CS:CODE,DS:DATA,SS:STACK

;各个常量的定义

;秒分时在存储体中的段地址

A_SMH EQU 8000h

;秒分时在存储体中的偏移地址

A_SECOND EQU 0100H

A_MINUTE EQU 0200H

A_HOUR EQU 0300H

;8255_1 工作方式

MODE8255_1 EQU 10001001B

;8255_1 各个口地址

PA8255_1 EQU 8000H

PB8255_1 EQU 8002H

PC8255_1 EQU 8004H

```
CON8255_1 EQU 8006H
;8255_2 工作方式
MODE8255_2 EQU 10000010B
;8255_2 各个口地址
PA8255_2 EQU 9000H
PB8255_2 EQU 9002H
PC8255_2 EQU 9004H
CON8255_2 EQU 9006H

;8259_1 主片偶端口
CS8259_1A EQU 0A000H
;8259_1 主片奇端口
CS8259_1B EQU 0A002H
;8259_1 的各个操作字
AICW1 EQU 00010001B
AICW2 EQU 00100000B
AICW3 EQU 10000000B
AICW4 EQU 00010001B
AOCW1 EQU 00000000B
;8259_2 从片偶端口
CS8259_2A EQU 0B000H
;8259_2 从片奇端口
CS8259_2B EQU 0B002H
;8259_2 的各个操作字
BICW1 EQU 00010001B
BICW2 EQU 00110000B
BICW3 EQU 00000111B
BICW4 EQU 00000001B
BOCW1 EQU 00000001B;初始默认为暂停计数，屏蔽 CLOCK 中断

;8253 的端口地址
TCON0 EQU 0C000H
TCON1 EQU 0C002H
TCON2 EQU 0C004H
CONTR EQU 0C006H
;8253 控制字
CON8253 EQU 00110100B
;计数初值
CONDATA EQU 000AH
;*****
;*****
;数据段
DATA SEGMENT
FLAG DB 01H
```

```
;标志位，00H 表示计数不暂停，01H 表示计数暂停
CLOCK_SPEED DB 0AH
;用于显示当前计数速度

DATA ENDS
;*****
;*****
;栈段
STACK SEGMENT
    STA DB 100 DUP(?)
    TOP EQU LENGTH STA
STACK ENDS
;*****
;*****
;代码段
CODE SEGMENT

START:
    ORG 800H
    MOV AX,DATA
    MOV DS,AX
    MOV AX,STACK
    MOV SS,AX
    MOV AX,TOP
    MOV SP,AX
    ;初始化 8255

INIT8255:
    ;8255_1
    MOV AL,MODE8255_1
    MOV DX,CON8255_1
    OUT DX,AL
    ;8255_2
    MOV AL,MODE8255_2
    MOV DX,CON8255_2
    OUT DX,AL
    ;中断向量表
    CLI
    MOV AX,0000H
    MOV ES,AX
    MOV BX,CS
    MOV AX,OFFSET OP_START
    MOV ES:[128],AX
    MOV ES:[128+2],BX
    MOV BX,CS
    MOV AX,OFFSET OP_PAUSE
```

```
MOV ES:[132],AX
MOV ES:[132+2],BX
MOV BX,CS
MOV AX,OFFSET CLOCK
MOV ES:[192],AX
MOV ES:[192+2],BX
MOV BX,CS
MOV AX,OFFSET OP_INC
MOV ES:[216],AX
MOV ES:[216+2],BX
MOV BX,CS
MOV AX,OFFSET OP_DEC
MOV ES:[220],AX
MOV ES:[220+2],BX
MOV BX,CS
MOV AX,OFFSET SPEED_UP
MOV ES:[208],AX
MOV ES:[208+2],BX
MOV BX,CS
MOV AX,OFFSET SPEED_DOWN
MOV ES:[212],AX
MOV ES:[212+2],BX
MOV BX,CS
MOV AX,OFFSET SPEED_INIT
MOV ES:[196],AX
MOV ES:[196+2],BX
MOV BX,CS
MOV AX,OFFSET OP_ZERO
MOV ES:[200],AX
MOV ES:[200+2],BX
MOV BX,CS
MOV AX,OFFSET OP_LOAD
MOV ES:[204],AX
MOV ES:[204+2],BX
STI
;初始化 8259
INIT8259:
;8259_1
MOV DX,CS8259_1A
MOV AL,AICW1
OUT DX,AL
MOV DX,CS8259_1B
MOV AL,AICW2
OUT DX,AL
```

```
MOV AL,AICW3
OUT DX,AL
MOV AL,AICW4
OUT DX,AL
MOV AL,AOCW1
OUT DX,AL
;8259_2
MOV DX,CS8259_2A
MOV AL,BICW1
OUT DX,AL
MOV DX,CS8259_2B
MOV AL,BICW2
OUT DX,AL
MOV AL,BICW3
OUT DX,AL
MOV AL,BICW4
OUT DX,AL
MOV AL,BOCW1
OUT DX,AL
;初始化时钟存储
INIT_CLOCK:
;初始化时分秒
MOV AX,A_SMH
MOV ES,AX
MOV AX,0000H
MOV BX,A_SECOND
MOV ES:[BX],AX
MOV BX,A_MINUTE
MOV ES:[BX],AX
MOV BX,A_HOUR
MOV ES:[BX],AX
;初始化 8253
INIT8253:
MOV AL,CON8253
MOV DX,CONTR
OUT DX,AL
MOV AX,CONDATA
MOV DX,TCON0
OUT DX,AL
MOV AL,AH
OUT DX,AL
;更新标志位
MOV AL,01H
MOV BX,OFFSET DATA
```

```
MOV DS:[BX],AL  
;循环显示  
LP:  
    ;读 DATA 段， 显示当前状态  
    MOV BX,OFFSET CLOCK_SPEED  
    MOV AL,DS:[BX];读当前计数初值  
    MOV AH,00H  
    MOV BX,AX  
    MOV AX,000FH  
    SUB AX,BX;转换为计数速度=0FH-计数初值  
    SHL AX,1  
    SHL AX,1  
    SHL AX,1  
    SHL AX,1;在 AX 中左移 4 位， 使得能在 8255_2C 端口高四位输出  
    MOV BX,OFFSET DATA  
    MOV DL,DS:[BX];读当前状态  
    ADD AL,DL;AL=计数速度(四位二进制)0000B+00000000 当前状态(一位二进制)B  
    ;从而实现 8255_2C 端口能同时读出计数速度与当前状态  
    MOV DX,PC8255_2  
    OUT DX,AL;同时读出计数速度与当前状态  
    ;读存储体，并显示计时  
    MOV AX,A_SMH  
    MOV ES,AX  
    MOV BX,A_SECOND  
    MOV AX,ES:[BX]  
    MOV DX,PA8255_1  
    OUT DX,AL  
    MOV BX,A_MINUTE  
    MOV AX,ES:[BX]  
    MOV DX,PB8255_1  
    OUT DX,AL  
    MOV BX,A_HOUR  
    MOV AX,ES:[BX]  
    MOV DX,PA8255_2  
    OUT DX,AL  
    JMP LP  
  
;-----恢复计数中断服务子程序  
OP_START:  
    CLI  
    MOV BX,OFFSET DATA  
    MOV AL,DS:[BX];读当前状态  
    MOV AH,00H  
    CMP AX,0000H
```

```
JZ START-END;不是暂停则关中断
JMP _START;结束暂停并修改标志位
_START:
;屏蔽各个时钟调整程序的中断，取消屏蔽 CLOCK 中断
MOV DX,CS8259_2B
MOV AL,BICW2
OUT DX,AL
MOV AL,BICW3
OUT DX,AL
MOV AL,BICW4
OUT DX,AL
MOV AL,11111110B;仅不屏蔽 CLOCK 中断
OUT DX,AL
;更新标志位
MOV AL,00H
MOV BX,OFFSET DATA
MOV DS:[BX],AL
JMP START-END
START-END:;关中断
MOV DX,CS8259_1A
MOV AL,20H
OUT DX,AL
STI
IRET

;-----暂停中断服务子程序
OP_PAUSE:
CLI
MOV BX,OFFSET DATA
MOV AL,DS:[BX];读当前状态
MOV AH,00H
CMP AX,0001H
JZ PAUSE-END;是暂停则关中断
JMP _PAUSE;暂停计数并修改标志位
_PAUSE:
;取消屏蔽各个时钟调整程序的中断，屏蔽 CLOCK 中断
MOV DX,CS8259_2B
MOV AL,BICW2
OUT DX,AL
MOV AL,BICW3
OUT DX,AL
MOV AL,BICW4
OUT DX,AL
MOV AL,00000001B;仅屏蔽 CLOCK 中断
```

```
OUT DX,AL
;更新标志位
MOV AL,01H
MOV BX,OFFSET DATA
MOV DS:[BX],AL
JMP PAUSE_END
PAUSE_END:;关中断
MOV DX,CS8259_1A
MOV AL,20H
OUT DX,AL
STI
IRET
OP_END:
CLI
MOV AX,0000H
MOV BX,0020H
ADD AL,BL
DAA
MOV DX,PA8255_1
OUT DX,AL

MOV DX,CS8259_1A
MOV AL,20H
OUT DX,AL
STI
IRET
```

```
;-----时钟中断服务子程序
CLOCK:
CLI
;秒级计数器加一
MOV AX,A_SMH
MOV ES,AX
MOV BX,A_SECOND
MOV AX,ES:[BX]
ADD AX,0001H
DAA
MOV ES:[BX],AX
;调用时钟数规则化函数，进行进位处理
CALL FAR PTR SAMPLE
;关中断
MOV DX,CS8259_2A
MOV AL,20H
OUT DX,AL
```

```
MOV DX,CS8259_1A
MOV AL,20H
OUT DX,AL
STI
IRET

;-----自增中断子程序
OP_INC:
    CLI
    ;读 8255_1PC 端口
    MOV DX,PC8255_1
    IN AL,DX
    MOV AH,00H
    ;根据 8255_1PC 口的输入来判断是给哪一位自增
    CMP AX,0000H
    JZ SECOND_INC
    CMP AX,0001H
    JZ MINUTE_INC
    CMP AX,0002H
    JZ HOUR_INC
    JMP SECOND_INC

END_INC:
    ;规则化处理
    CALL FAR PTR SAMPLE
    MOV DX,CS8259_2A
    MOV AL,20H
    OUT DX,AL
    MOV DX,CS8259_1A
    MOV AL,20H
    OUT DX,AL
    STI
    IRET

SECOND_INC:
    MOV AX,A_SMH
    MOV ES,AX
    MOV BX,A_SECOND
    MOV AX,ES:[BX]
    ADD AX,1
    DAA
    MOV ES:[BX],AX
    JMP END_INC

MINUTE_INC:
    MOV AX,A_SMH
    MOV ES,AX
```

```
MOV BX,A_MINUTE
MOV AX,ES:[BX]
ADD AX,1
DAA
MOV ES:[BX],AX
JMP END_INC

HOUR_INC:
    MOV AX,A_SMH
    MOV ES,AX
    MOV BX,A_HOUR
    MOV AX,ES:[BX]
    ADD AX,1
    DAA
    MOV ES:[BX],AX
    JMP END_INC

;-----自减中断子程序

OP_DEC:
    CLI
    ;读 8255_1PC 端口
    MOV DX,PC8255_1
    IN AL,DX
    MOV AH,00H
    ;根据 8255_1PC 口的输入来判断是给哪一位自增
    CMP AX,0000H
    JZ SECOND_DEC
    CMP AX,0001H
    JZ MINUTE_DEC
    CMP AX,0002H
    JZ HOUR_DEC
    JMP SECOND_DEC

END_DEC:
    MOV DX,CS8259_2A
    MOV AL,20H
    OUT DX,AL
    MOV DX,CS8259_1A
    MOV AL,20H
    OUT DX,AL
    STI
    IRET

SECOND_DEC:;秒位自减
    MOV AX,A_SMH
    MOV ES,AX
    MOV BX,A_SECOND
```

```
MOV AX,ES:[BX]
CMP AX,0;先判断秒是否为零
JZ SECOND_ZERO_1
SUB AX,1;秒不为零则减一
DAS
MOV ES:[BX],AX
JMP END_DEC
SECOND_ZERO_1:;秒位为零时
    MOV BX,A_MINUTE
    MOV AX,ES:[BX]
    CMP AX,0;先判断分是否为零
    JZ MINUTE_ZERO_1
    SUB AX,1;分不为零则减一
    DAS
    MOV ES:[BX],AX
    MOV AX,0059H;然后将秒置 59
    MOV BX,A_SECOND
    MOV ES:[BX],AX
    JMP END_DEC
MINUTE_ZERO_1:;分秒都为零时
    MOV BX,A_HOUR
    MOV AX,ES:[BX]
    CMP AX,0;先判断时是否为零
    JZ HOUR_ZERO_1
    SUB AX,1;时不为零则减一
    DAS
    MOV ES:[BX],AX
    MOV AX,0059H;然后将秒与分置 59
    MOV BX,A_SECOND
    MOV ES:[BX],AX
    MOV BX,A_MINUTE
    MOV ES:[BX],AX
    JMP END_DEC
HOUR_ZERO_1:;时分秒都为零时
    ;不做修改
    JMP END_DEC
MINUTE_DEC:;分位自减
    MOV AX,A_SMH
    MOV ES,AX
    MOV BX,A_MINUTE
    MOV AX,ES:[BX]
    CMP AX,0;先判断分是否为零
    JZ MINUTE_ZERO_2
    SUB AX,1;不为零则减一
```

```
DAS
MOV ES:[BX],AX
JMP END_DEC
MINUTE_ZERO_2:
    MOV BX,A_HOUR
    MOV AX,ES:[BX]
    CMP AX,0;先判断时是否为零
    JZ HOUR_ZERO_2
    SUB AX,1
    DAS
    MOV ES:[BX],AX
    MOV BX,A_MINUTE
    MOV AX,0059H;然后将分置为 59
    MOV ES:[BX],AX
    JMP END_DEC
HOUR_ZERO_2;;时和分都为零时
    ;不做修改
    JMP END_DEC
HOUR_DEC;;时位自减
    MOV AX,A_SMH
    MOV ES,AX
    MOV BX,A_HOUR
    MOV AX,ES:[BX]
    CMP AX,0;先判断时是否为零
    JZ HOUR_ZERO_3
    SUB AX,1;不为零则减一
    DAS
    MOV ES:[BX],AX
    JMP END_DEC
HOUR_ZERO_3;;时都为零时
    ;不做修改
    JMP END_DEC

;-----复位中断子程序
OP_ZERO:
    CLI
    ;将存储体内的对应时分秒复位到 00:00:00
    MOV AX,A_SMH
    MOV ES,AX
    MOV AX,0000H
    MOV BX,A_SECOND
    MOV ES:[BX],AX
    MOV BX,A_MINUTE
    MOV ES:[BX],AX
```

```
MOV BX,A_HOUR
MOV ES:[BX],AX
;关中断
MOV DX,CS8259_2A
MOV AL,20H
OUT DX,AL
MOV DX,CS8259_1A
MOV AL,20H
OUT DX,AL
STI
IRET

;-----置数中断子程序
OP_LOAD:
CLI
;读 8255_1PC 端口
MOV DX,PC8255_1
IN AL,DX
MOV AH,00H
;根据 8255_1PC 口的输入来判断是给哪一位置数
CMP AX,0000H
JZ LOAD_SECOND
CMP AX,0001H
JZ LOAD_MINUTE
CMP AX,0002H
JZ LOAD_HOUR
JMP LOAD_SECOND

NEXT_LOAD:
;关中断
MOV DX,CS8259_2A
MOV AL,20H
OUT DX,AL
MOV DX,CS8259_1A
MOV AL,20H
OUT DX,AL
STI
IRET

LOAD_SECOND:
MOV DX,PB8255_2
IN AL,DX
MOV AH,00H
CMP AX,60H
JAE SECOND_1
SECOND_2:
```

```
ADD AX,0
DAA
MOV DX,AX
MOV AX,A_SMH
MOV ES,AX
MOV BX,A_SECOND
MOV ES:[BX],DX
JMP NEXT_LOAD

SECOND_1:
MOV AX,59H
JMP SECOND_2

LOAD_MINUTE:
MOV DX,PB8255_2
IN AL,DX
MOV AH,00H
CMP AX,60H
JAE MINUTE_1

MINUTE_2:
ADD AX,0
DAA
MOV DX,AX
MOV AX,A_SMH
MOV ES,AX
MOV BX,A_MINUTE
MOV ES:[BX],DX
JMP NEXT_LOAD

MINUTE_1:
MOV AX,59H
JMP MINUTE_2

LOAD_HOUR:
MOV DX,PB8255_2
IN AL,DX
MOV AH,00H
CMP AX,24H
JAE HOUR_1

HOUR_2:
ADD AX,0
DAA
MOV DX,AX
MOV AX,A_SMH
MOV ES,AX
MOV BX,A_HOUR
MOV ES:[BX],DX
JMP NEXT_LOAD
```

HOUR_1:

```
MOV AX,23H  
JMP HOUR_2
```

;-----计时速度复位函数

INITCLOCK:

```
MOV BX,OFFSET CLOCK_SPEED  
MOV AX,CONDATA  
MOV AH,00H  
MOV DS:[BX],AL;更新后的计时速度写入 data 段  
MOV DX,TCON0;修改计数初值  
OUT DX,AL  
MOV AL,AH  
OUT DX,AL  
RETF
```

;-----提高速度中断子程序

SPEED_UP:

```
CLI  
MOV BX,OFFSET CLOCK_SPEED  
MOV AH,00H  
MOV AL,DS:[BX]  
CMP AX,0005H  
JZ INITCLOCK_UP  
SUB AX,1;计数速度增加 1  
MOV DS:[BX],AL;修改后的速度存入 data 段  
MOV DX,TCON0;修改计数初值  
OUT DX,AL  
MOV AL,AH  
OUT DX,AL  
JMP END_UP
```

INITCLOCK_UP:

```
CALL FAR PTR INITCLOCK  
;关中断
```

END_UP:

```
MOV DX,CS8259_2A  
MOV AL,20H  
OUT DX,AL  
MOV DX,CS8259_1A  
MOV AL,20H  
OUT DX,AL  
STI  
IRET
```

;-----降低速度中断子程序

```
SPEED_DOWN:  
    CLI  
    MOV BX,OFFSET CLOCK_SPEED  
    MOV AH,00H  
    MOV AL,DS:[BX]  
    CMP AX,000FH  
    JZ INITCLOCK_DOWN  
    ADD AX,1;计数速度减 1  
    MOV DS:[BX],AL;修改后的速度存入 data 段  
    MOV DX,TCON0;修改计数初值  
    OUT DX,AL  
    MOV AL,AH  
    OUT DX,AL  
    JMP END_DOWN  
INITCLOCK_DOWN:  
    CALL FAR PTR INITCLOCK  
    ;关中断  
END_DOWN:  
    MOV DX,CS8259_2A  
    MOV AL,20H  
    OUT DX,AL  
    MOV DX,CS8259_1A  
    MOV AL,20H  
    OUT DX,AL  
    STI  
    IRET
```

```
;-----复位速度中断子程序  
SPEED_INIT:  
    CLI  
    CALL FAR PTR INITCLOCK  
    ;关中断  
END_INIT:  
    MOV DX,CS8259_2A  
    MOV AL,20H  
    OUT DX,AL  
    MOV DX,CS8259_1A  
    MOV AL,20H  
    OUT DX,AL  
    STI  
    IRET
```

```
;-----时钟记录标准化函数  
SAMPLE:
```

```
MOV AX,A_SMH
MOV ES,AX
MOV BX,A_SECOND
MOV AX,ES:[BX]
CMP AX,0060H
JAE SAMPLE_SECOND
SAMPLE_NEXT_1:
    MOV BX,A_MINUTE
    MOV AX,ES:[BX]
    CMP AX,0060H
    JAE SAMPLE_MINUTE
SAMPLE_NEXT_2:
    MOV BX,A_HOUR
    MOV AX,ES:[BX]
    CMP AX,0024H
    JAE SAMPLE_HOUR
SAMPLE_NEXT_3:
    RETF
SAMPLE_SECOND:
    MOV AX,0
    MOV ES:[BX],AX
    MOV BX,A_MINUTE
    MOV AX,ES:[BX]
    ADD AX,0001H
    DAA
    MOV ES:[BX],AX
    JMP SAMPLE_NEXT_1
SAMPLE_MINUTE:
    MOV AX,0
    MOV ES:[BX],AX
    MOV BX,A_HOUR
    MOV AX,ES:[BX]
    ADD AX,0001H
    DAA
    MOV ES:[BX],AX
    JMP SAMPLE_NEXT_2
SAMPLE_HOUR:
    MOV AX,0
    MOV BX,A_SECOND
    MOV ES:[BX],AX
    MOV BX,A_MINUTE
    MOV ES:[BX],AX
    MOV BX,A_HOUR
    MOV ES:[BX],AX
```

JMP SAMPLE_NEXT_3

CODE ENDS

;*****

END START