

浙江大学 实验报告

课程名称：微机原理及应用

实验名称：实验 3 流水灯与蜂鸣器

指导老师：田翔/马永昌

实验类型：仿真验证实验

专业：生物医学工程

姓名：

学号：

日期：2024 年 5 月 5 日

地点：东 1B-416

目录

1.	实验目的和要求	2
1.1.	实验要求	2
1.2.	实验任务	2
2.	实验内容和原理	2
3.	主要仪器设备	2
4.	操作方法和实验步骤	3
4.1.	需求分析	3
4.2.	系统设计	3
4.3.	硬件设计	3
4.3.1.	输入	3
4.3.2.	输出	3
4.4.	软件设计	3
4.4.1.	硬件资源规划	3
4.4.2.	流程图	6
4.4.3.	代码	7

5.	实验结果和分析	10
5.1.	系统测试	10
5.2.	分析	11
6.	思考和讨论	11
6.1.	DEBUG	11
6.2.	总结	12

1. 实验目的和要求

1.1. 实验要求

- 1.熟悉 MCS-51 教学实验系统硬件结构。
- 2.编写汇编代码。
- 3.对实验任务进行认真分析，写出需求分析报告和系统设计报告。

1.2. 实验任务

1. 用延时方式编写流水灯程序：
实现板上 8 位 LED 自右向左依次以 500ms 间隔循环闪亮。
2. 用中断方式编写蜂鸣器程序：
实现蜂鸣器间隔一段时间响一次或者发出频率不断变化的报警声。

2. 实验内容和原理

内容同 1.2 实验任务

流水灯主要原理为通过控制延时时间来控制点亮间隔

蜂鸣器原理主要为改变单片机引脚输出波形的频率，就可以调整控制蜂鸣器音调，产生各种不同音色、音调的声音。改变输出电平的高低电平占空比，则可以控制蜂鸣器的声音大小。本实验主要通过中断实现蜂鸣器不同频率发声。

3. 主要仪器设备

计算机、仿真软件、开发板

4. 操作方法和实验步骤

4.1. 需求分析

通过控制延时函数层数及循环次数实现控制延时时长，通过控制 P2 端口值实现不同 LED 点亮

通过设置合适计时器 1 初值实现合适频率中断，通过控制延时时长控制蜂鸣器发生频率。

4.2. 系统设计

- 实验装置以单片机为核心
- 输入控制：无外部输入
- 输出控制：流水灯输出为 P2 管脚(控制 LED 灯)
- 控制程序：先计算 $34 \times T$ ，再计算 $3 \times C \times T$ ，再计算 $3 \times C \times T / 200$ ，最后计算和

4.3. 硬件设计

系统核心采用 8051 单片机，该单片机在不进行外扩数据存储器和程序存储器的条件下，共有 32 个 I/O 可供使用。功能强，应用面广，价格低。

Proteus 器件选择: AT89C51

4.3.1. 输入

本实验无外部输入

4.3.2. 输出

本实验无输出

4.4. 软件设计

4.4.1. 硬件资源规划

寄存器分配

A

用途：累加器

描述：在多次计算中使用，用于执行算术运算如加法、减法和乘法。

B

用途：乘法和除法辅助寄存器

描述：在乘法运算中与累加器配合使用，也用于保存除法运算中的余数。

R4

用途：除数存储

描述：存储除法运算的除数。

R7

用途：循环计数器

描述：用于控制除法的迭代次数。

DPTR (DPL, DPH)

用途：数据指针

描述：在除法运算中用作被除数的地址指针。

内存分配

30H, 31H, 32H

用途：温度和时间参数

描述：存储环境温度和时间的高位和低位。

33H-3FH

用途：中间结果和最终结果存储

描述：

33H：存储 $3 \times C$ 的结果。

34H-36H：存储 $34 \times T$ 的结果。

37H-3AH：用于除法运算的中间变量和最终的商的部分。

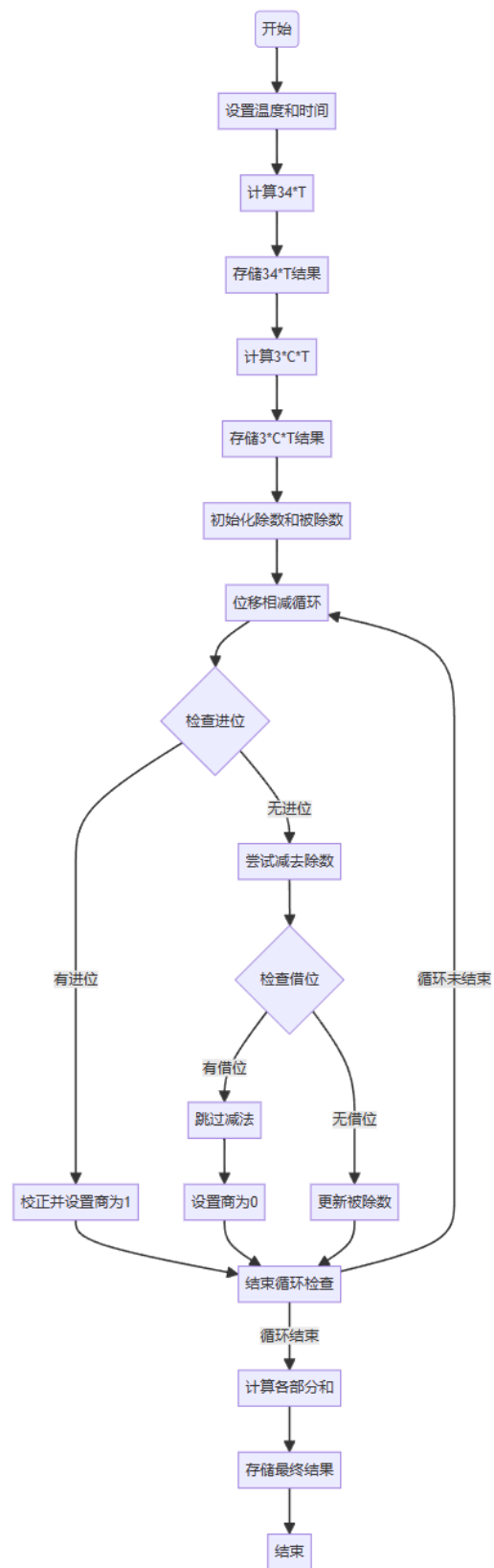
3BH-3DH：最终的商值。

52H, 51H, 50H

用途：最终计算结果的存储

描述：存储 $3 \cdot C \cdot T / 200$ 和 $34 \cdot T$ 的总和的低中高位。

4.4.2. 流程图



4.4.3. 代码

```
;设定初始参数
ORG 0000H ;程序起点
;设置环境温度和时间 T
MOV 30H,#14H ;C=20
MOV 31H,#01H ;T 高位
MOV 32H,#90H ;T 低位

;计算 34*T
MOV A, #34
MOV B, 32H
MUL AB ; A = 34, B = 90H (144)
MOV 36H, A ; 储存结果低位 (低位结果)
MOV 35H, B ; 储存结果中位 (高位结果)
MOV A, 31H ; 加载 T 的高位
MOV B, #34 ; 重新加载常量 34
MUL AB ; 再次乘法运算 (34 * 1)
MOV 3EH, B
ADD A, 35H ; 加上之前的中位结果
MOV 35H, A ; 更新中位
MOV A, #00
MOV B, 3EH
ADDC A, B ; 加上高位, 注意进位
MOV 34H, A ; 存储结果的高位

;计算 3*C*T
MOV A, 30H
ADD A, 30H
ADD A, 30H ; 得到 3*C = 3 * 20 = 60
MOV 33H, A
MOV B, 32H ; 加载 T 的低位 (90H, 十进制 144)
MUL AB ; 60 * 144 = 8640
MOV 38H, A ; 保存低位 (60 * 144 的低位)
MOV 39H, B ; 保存中位 (60 * 144 的高位)
MOV A, 31H ; 加载 T 的高位 (01H, 十进制 1)
MOV B, 33H ; 加载 3*C 的结果 (60)
MUL AB ; 60 * 1 = 60
MOV 3FH, B ; 保存最高位
MOV B, 39H ; 加载先前的中位结果
ADD A, B ; 将当前低位与先前中位相加
MOV 39H, A ; 更新中位结果
MOV A, #00 ; 加载先前最高位 (如果有必要初始化为 0)
MOV B, 3FH
ADDC A, B ; 加上更高位, 并考虑进位
MOV 3AH, A ; 保存最高位
;计算 3*C*T/200
```

```

;初始化除数和被除数
;计算 3*C*T/200
; 初始化除数和被除数
MOV A, #0C8H      ; 使用累加器 A 载入立即数 200
MOV R4, A          ; 设置除数为 200
MOV A, #00H
MOV 37H, A
MOV R7, #24        ; 设定除法迭代次数为 24 次
MOV DPL, 38H       ; 被除数低位
MOV DPH, 39H       ; 被除数中位
MOV B, 3AH         ; 被除数高位
CLR 3BH            ; 清除商低位储存位置
CLR 3CH            ; 清除商中位
CLR 3DH            ; 清除商高位

; 位移相减循环
DIV_LOOP:
    CLR C          ; 清除进位, 为左移准备
    ; 左移商, 准备接受新的最低位
    MOV A, 3BH
    RLC A
    MOV 3BH, A
    MOV A, 3CH
    RLC A
    MOV 3CH, A
    MOV A, 3DH
    RLC A
    MOV 3DH, A
    ; 左移被除数
    CLR C
    MOV A, DPL
    RLC A
    MOV DPL, A
    MOV A, DPH
    RLC A
    MOV DPH, A
    MOV A, B
    RLC A
    MOV B, A
    MOV A, 37H
    RLC A
    MOV 37H, A

    ; 如果产生进位, 则说明是 9 位数, 必定大于 200, 跳转 DIV_0, 如果没有产生进位, 则跳转
    DIV_1
    JNC DIV_1
DIV_0:
    MOV A, #56      ; 第九位为 1, 故 256-200=56, 所以直接给 37H 加上 56

```



```

    ADD A, 37H
    MOV 37H, A
    ORL 3BH, #01H    ; 将当前商最低位设置为 1
    SJMP END_LOOP    ; 跳转到循环末尾

DIV_1:
    MOV A, 37H        ; 加载最高位
    CLR C
    SUBB A, R4        ; 减去除数
    JC NO_SUB        ; 如果有借位, 跳过减法
    ORL 3BH, #01H    ; 将当前商最低位设置为 1
    ; 执行减法并更新被除数
    MOV 37H, A        ; 更新最高位
    MOV A, B
    SUBB A, #0
    MOV B, A
    MOV A, DPH
    SUBB A, #0
    MOV DPH, A
    MOV A, DPL
    SUBB A, #0
    MOV DPL, A
    DJNZ R7, DIV_LOOP ; 递减计数器并根据需要重复循环

NO_SUB:
    ANL 3BH, #0FEH    ; 将当前商最低位设置为 0

END_LOOP:
    DJNZ R7, DIV_LOOP ; 递减计数器并根据需要重复循环

; 计算和
CLR C
MOV A, 36H
ADD A, 3BH
MOV 52H, A ; 储存低位
MOV A, 35H
ADDC A, 3CH
MOV 51H, A ; 储存中位
MOV A, 34H
ADDC A, 3DH
MOV 50H, A ; 储存高位
END

```

5. 实验结果和分析

5.1. 系统测试

Keil5 调试截图

默认参数下

31H-3FH 值

Address: D:31H
D:0x31: 01 90 3C 00 35 20 00 C0 5D 00 78 00 00 00
D:0x98: 00 00 00 00 00 00 00 00 FF 00 00 00 00 00
D:0xFF: 00 00 00 00 00 00 C8 00 00 00 00 00 00 00

50H-52H 值

The screenshot displays the Keil5 IDE interface during a simulation. The 'Registers' window shows the state of various registers, with r4 highlighted at 0x00. The 'exp2.asm' window shows the following assembly code:

```
1 ;设定初始参数
2 ORG 0000H ;程序起点
3 ;设置环境温度和时间T
4 MOV 30H,#14H ;C=20
5 MOV 31H,#01H ;T高位
6 MOV 32H,#90H ;T低位
7
8 ;计算34*T
9 MOV A, #34
10 MOV B, 32H
11 MUL AB ; A = 34, B = 90H (144)
12 MOV 36H, A ; 储存结果低位 (低位结果)
```

The 'Memory 1' window shows a memory dump starting at address D:50H. The dump includes data for addresses D:0x50 through D:0xA6, showing hexadecimal values and their corresponding decimal representations. The simulation status at the bottom indicates 'Simulation' and 't1: 0.00045650 sec'.

修改 T 为 500ms, 十六进制下为 01F4H

C 不变

理论上结果为 17150(42FEH)

31H-3FH 值

Address: D:31H
D:0x31: 01 F4 3C 00 42 68 00 30 75 00 96 00 00 00 00 00 00
D:0x98: 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00
D:0xFF: 00 00 00 00 00 00 C8 00 00 00 00 00 00 00 00 00 00
D01:0x66: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0xCD: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

50H-52H 值

Address:	D:50H
D:0x50:	00 42 FE 00 00 00 00 00 00 00
D:0xB7:	00 00 00 00 00 00 00 00 00 00
D01:0x1E:	00 00 00 00 00 00 00 00 00 00

再次修改 C 为 40，此时理论计算结果为 17300 (4394H)

30H-3FH 值

Address:	D:30H
D:0x30:	28 01 F4 78 00 42 68 00 60 EA 00 2C 01 00 00 00
D:0x97:	00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00
D:0xFE:	00 00 00 00 00 00 C8 00 00 00 00 00 00 00 00

50H-52H 值

Address:	D:50H
D:0x50:	00 43 94 00 00 00 00 00 00 00
D:0xB7:	00 00 00 00 00 00 00 00 00 00

5.2. 分析

系统运行结果表明，本实验设计的基于 51 单片机的硬件电路和软件组成的系统，采用数码管, IO 硬件和循环、条件程序，基于 $S = 34 * T + 3C * T / 200$ 实现了对于 S 值的正确计算及储存

完成了实验任务。

6. 思考和讨论

6.1. DEBUG

1.位移相减循环中一开始在将被除数高位移到拓展位 37H 时并没有考虑到特殊的情况（及 37H 产生进位，此时实际上已经是 9 位，而 200 只是 8 位，此时必定能够减去 200，故须设置商为 1）所以计算得商小于实际值

修正后

```

; 如果产生进位，则说明是 9 位数，必定大于 200，跳转 DIV_0, 如果没有产生进位，则跳转
DIV_1
    JNC DIV_1
DIV_0:
    MOV A, #56      ;第九位为 1，故 256-200=56，所以直接给 37H 加上 56
    ADD A, 37H
    MOV 37H, A
    ORL 3BH, #01H   ; 将当前商最低位设置为 1
    SJMP END_LOOP   ; 跳转到循环末尾

```

2.忘记在减法更新末尾直接跳转循环（116行），导致设置商为1后又置为0，导致最终商结果为70H（1110000），预期为78H（1111000）

```

99  DIV_1:
100  MOV A, 37H      ; 加载最高位
101  CLR C
102  SUBB A, R4      ; 减去除数
103  JC NO_SUB      ; 如果有借位，跳过减法
104  ORL 3BH, #01H   ; 将当前商最低位设置为1
105  ; 执行减法并更新被除数
106  MOV 37H, A      ; 更新最高位
107  MOV A, B
108  SUBB A, #0
109  MOV B, A
110  MOV A, DPH
111  SUBB A, #0
112  MOV DPH, A
113  MOV A, DPL
114  SUBB A, #0
115  MOV DPL, A
116  ; DJNZ R7, DIV_LOOP ; 递减计数器并根据需要重复循环
117
118  NO_SUB:
119  ANL 3BH, #0FEH   ; 将当前商最低位设置为0
120
121  END_LOOP:
122  DJNZ R7, DIV_LOOP ; 递减计数器并根据需要重复循环

```

修正前

Address:	D:3BH
D:0x3B:	70 00 00 00 00 00 00 00 00 00 00 00
D:0xA2:	00 00 00 00 00 00 00 00 00 00 00 00
D01:0x09:	00 00 00 00 00 00 00 00 00 00 00 00

修正后

Address:	D:3BH
D:0x3B:	78 00 00 00 00 00 00 00 00 00 00 00
D:0xA2:	00 00 00 00 00 00 00 00 00 00 00 00
D01:0x09:	00 00 00 00 00 00 00 00 00 00 00 00

6.2. 总结

作为第二次实验，本次实验较第一次实验复杂许多，在编写代码的过程中不可避免写出了一些 BUG，在 DEBUG 过程中深入了解了 DEBUG 的流程，在实践中也加深了对于微机这门课程的理解。