北京交通大学汇编与接口技术实验报告

姓名：程维森

学号：21231264

实验二、利用 8255A 实现 LED 的流水点亮实验

一、 实验目的

该实验的目的在于让学生掌握 8255A 和微机接口的连接方法，了解 8255A 的基本的工作原理和编程方法。

二、 实验内容

PA 口接 8 个拨动开关 K1-K8，PB 口接 8 个 LED。初始由开关 K1-K8 设定 8 位不同的值，当执行程序后 LED 按 K1-K8 初始设定的值点亮，并向右流动（8255A 工作在 0 方式）。选择完成在数码管上显示“8255-A”。

内容分析：

LED 流水灯：

PA 口（端口 280H）连接了 8 个拨动开关（K1-K8），PB 口（端口 281H）连接了 8 个 LED 灯。在程序开始执行时，首先等待拨动开关 K1-K8 设置 8 位不同的值。然后，LED 按照 K1-K8 初始设定的值点亮，并且 LED 灯的状态向右流动，即从 K1-K8 依次亮起。这个实验基于 Intel 8255A 芯片，该芯片提供了 8 位的并行输入/输出端口。

8255-A 显示：

与 LED 流水灯相似，但是显示的为 8255-A 字样

三、 实验接线图

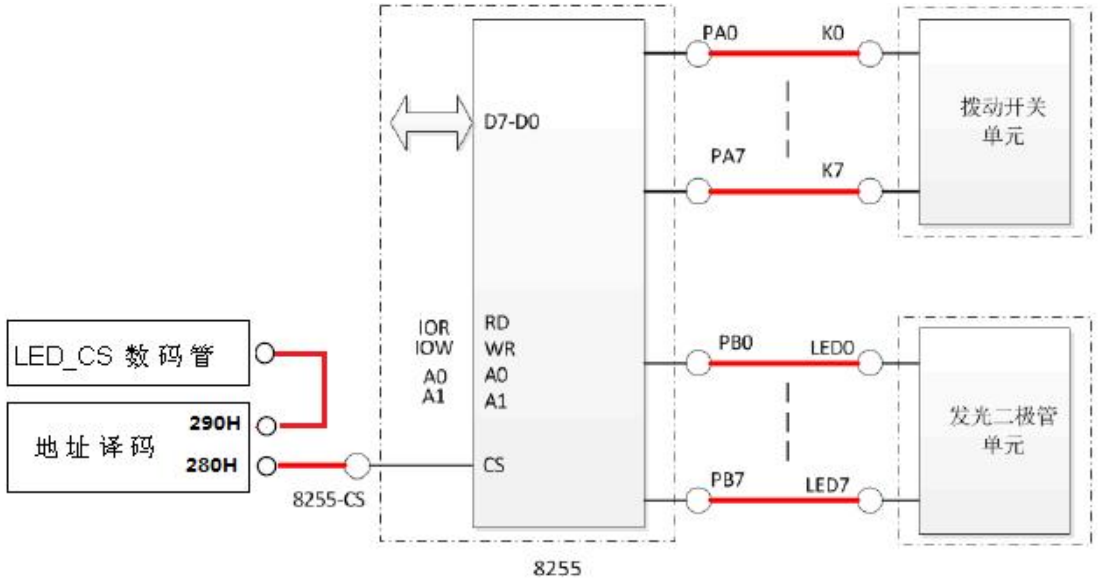
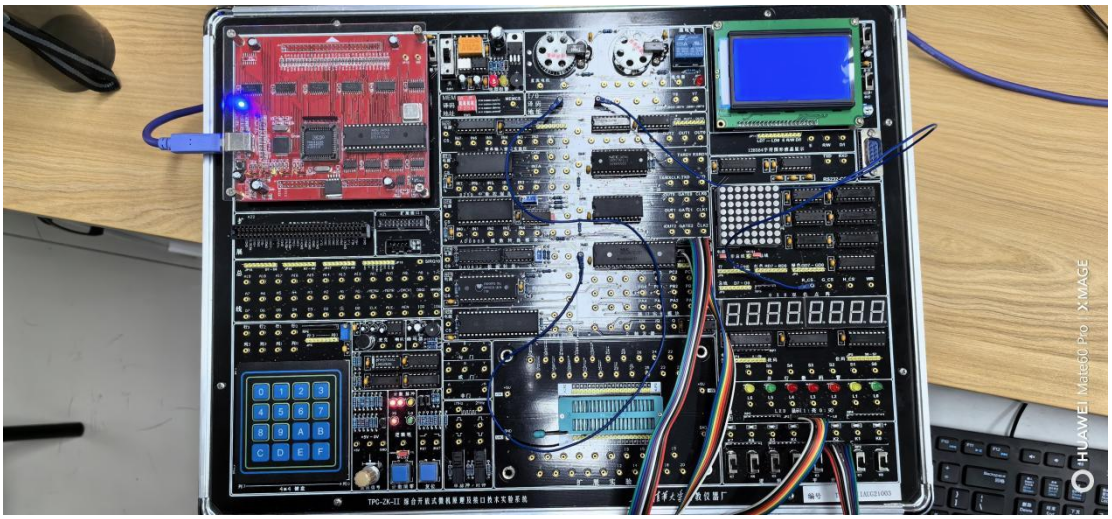
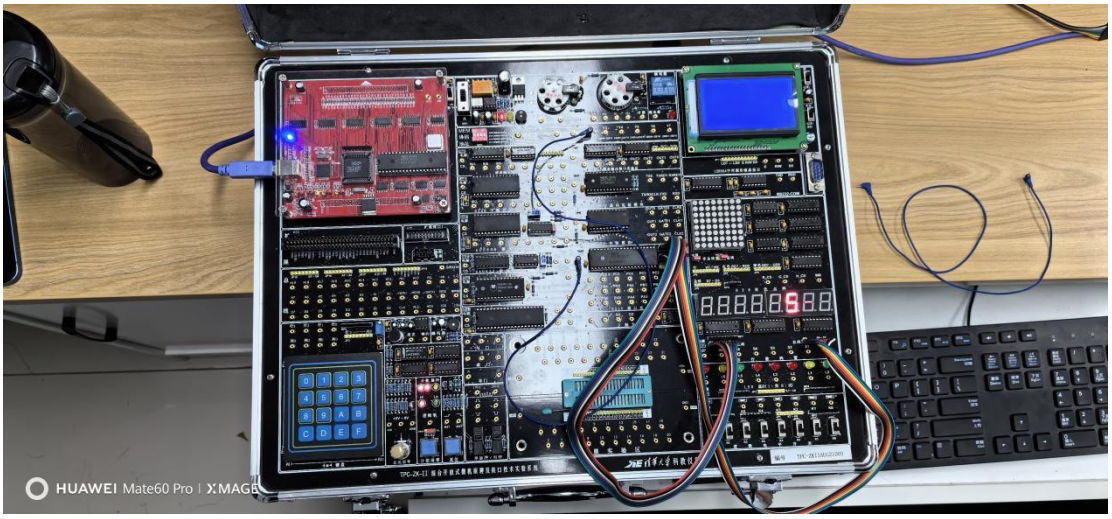


图 1-3 实验连线图



LED 流水灯接线图



8255—A 数显接线图

四、 实验编程代码

LED 右流水仅仅取决于第 1 次开关的位置，一旦 LED 流水开始， LED 流水的次序将不再理睬开关位置的重新变化。

提示：为了使流水显示明显，每个位置应加延时程序。

```
DATA SEGMENT
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:
    CALL INIT_PORTS
    CALL WAIT_SWITCH_ON
    CALL PROCESS_SWITCHES
    CALL DISPLAY_PATTERN
    JMP START

INIT_PORTS PROC
    MOV AX, DATA
    MOV DS, AX

    MOV DX, 283H
    MOV AL, 98H
    OUT DX, AL

    MOV DX, 281H
    MOV AL, 00H
    OUT DX, AL
    RET
INIT_PORTS ENDP

WAIT_SWITCH_ON PROC
WAIT_SWITCH_LOOP:
    MOV DX, 280H
    IN AL, DX
    AND AL, 11111111B
    JZ WAIT_SWITCH_LOOP
    RET
WAIT_SWITCH_ON ENDP

PROCESS_SWITCHES PROC
    MOV CX, 08H
    MOV BL, 01H
PROCESS_SWITCHES_LOOP:
    MOV DL, AL
    AND DL, BL
    JNZ PROCESS_SWITCHES_FOUND
    SHL BL, 1H
    LOOP PROCESS_SWITCHES_LOOP
    JMP PROCESS_SWITCHES_LOOP

PROCESS_SWITCHES_FOUND:
    MOV AL, DL
    RET
PROCESS_SWITCHES ENDP

DISPLAY_PATTERN PROC
DISPLAY_LOOP:
    MOV DX, 281H
    OUT DX, AL
    CALL DELAY
    ROR AL, 1H
    JMP DISPLAY_LOOP
DISPLAY_PATTERN ENDP

DELAY PROC
    PUSH CX
    PUSH BX
    MOV BX, 0FFFFH
DELAY_OUTER_LOOP:
    MOV CX, 0FH
DELAY_INNER_LOOP:
    DEC CX
    JNZ DELAY_INNER_LOOP
    DEC BX
    JNZ DELAY_OUTER_LOOP
    POP BX
    POP CX
    RET
DELAY ENDP

CODE ENDS
```

LED 流水灯代码

代码分析：

INIT_PORTS 初始化了端口，将数据加载到端口 283H 和 281H 上，用来配置 8255 芯片的功能。

WAIT_SWITCH_ON 循环等待开关被打开。在端口 280H 读取数据，如果某个位为 1（表示开关打开），则跳出循环。

PROCESS_SWITCHES 处理开关的状态。通过循环检查各个位，找到第一个打开的开

关，然后返回这个开关的状态。
DISPLAY_PATTERN 控制 LED 指示灯的显示。将找到的开关状态输出到端口 281H 上，然后通过 DELAY 过程实现延时，再将数据右移一个位，实现流水灯效果。
DELAY 实现一个简单的延时功能，通过嵌套的循环来进行延时。
代码拓展：
更改流水灯速度：

```
DELAY_OUTER_LOOP:
    MOV CX, 0FH
```

修改 0FH 的值即可
原理：
在这段代码中，`DELAY_OUTER_LOOP` 的内部循环 `MOV CX, 0FH` 控制了外部循环的迭代次数，从而影响了延时的时长。具体地说，`MOV CX, 0FH` 指令将 CX 寄存器的值设置为 15（0FH 的十六进制表示），这就是外部循环的迭代次数。
在这个上下文中，外部循环的目的是实现一个较长的延时，用于控制 LED 指示灯的亮灭速度。修改这个值，即修改 `MOV CX, 0FH` 中的 `0FH`，会直接影响延时的时长。如果你想要减小延时，可以将这个值改小（例如，设置为 `MOV CX, 07H`，对应十进制的 7）。如果想要增加延时，可以将这个值增大（例如，设置为 `MOV CX, 1EH`，对应十进制的 30）。
通过修改这个值，可以控制 LED 流水灯的速度，使其亮灭的间隔时间更短或更长。
更改流水灯方向：

```
DISPLAY_LOOP:
    MOV DX, 281H
    OUT DX, AL
    CALL DELAY
    ROR AL, 1H
    JMP DISPLAY_LOOP
```

修改 ROR 为 ROL 即可
原理：
这段代码的主要目的是控制 LED 的流水灯效果。其中，`DISPLAY_PATTERN` 过程通过 `DISPLAY_LOOP` 循环来控制 LED 的亮灭，而 `ROR AL, 1H` 指令是将 `AL` 寄存器的内容向右循环右移 1 位。在这个特定的场景下，将 `ROR` 指令改为 `ROL` 指令，即将 `AL` 寄存器的内容向左循环左移 1 位，将改变 LED 流水灯的方向。
原来的 `ROR AL, 1H` 将 `AL` 寄存器的最低位（最右边的位）移到最高位（最左边的位），导致 LED 流水灯向右移动。而如果将 `ROR` 改为 `ROL`，则将 `AL` 寄存器的最高位（最左边的位）移到最低位（最右边的位），导致 LED 流水灯向左移动。所以，修改 `ROR` 为 `ROL` 会改变流水灯的方向。

```
CODE SEGMENT
ASSUME CS:CODE
START:
MOV DX,293H
MOV AL,10001001B
OUT DX,AL

XOR AX,AX
MOV BL,0
LOP:
MOV DX,292H
IN AL,DX
CMP AL,AH
JZ NEXT
MOV AH,AL
MOV BL,AH
ROL BL,1
NEXT:
ROR BL,1
MOV AL,BL
MOV DX,291H
OUT DX,AL
MOV CX,3FFFH

DOIT:
MOV DX,291H
MOV AL,00000000B
OUT DX,AL
MOV DX,290H
MOV AL,7FH
OUT DX,AL
MOV DX,291H
MOV AL,00100000B
OUT DX,AL
CALL DELAY
MOV DX,291H
MOV AL,00000000B
OUT DX,AL
MOV DX,290H
MOV AL,5BH
OUT DX,AL
MOV DX,291H
MOV AL,00010000B
OUT DX,AL
CALL DELAY

MOV DX,291H
MOV AL,00000000B
OUT DX,AL
MOV DX,290H
MOV AL,6DH
OUT DX,AL
MOV DX,291H
MOV AL,00001000B
OUT DX,AL
CALL DELAY

MOV DX,291H
MOV AL,00000000B
OUT DX,AL
MOV DX,290H
MOV AL,6DH
OUT DX,AL
MOV DX,291H
MOV AL,00000100B
OUT DX,AL
CALL DELAY
MOV DX,291H
MOV AL,00000000B
OUT DX,AL
MOV DX,290H
MOV AL,40H
OUT DX,AL
MOV DX,291H
MOV AL,00000010B
OUT DX,AL
CALL DELAY

MOV DX,291H
MOV AL,00000000B
OUT DX,AL
MOV DX,290H
MOV AL,77H
OUT DX,AL
MOV DX,291H
MOV AL,00000001B
OUT DX,AL
CALL DELAY
LOOP DOIT
JMP LOP

DELAY PROC
PUSH BX
PUSH CX
MOV BX,0FFFFH
DL1:MOV CX,06FH
DL2:DEC CX
JNZ DL2
DEC BX
JNZ DL1
POP CX
POP BX
RET
DELAY ENDP

CODE ENDS
END START
```

8255-A 显示代码

原理几乎同 LED 流水灯，不予阐述

五、 实验步骤

1、 根据原理图正确连接实验线路（需要连接红线）。

2、 正确理解实验原理。

3、 编写实验程序，并上机调试，观察实验结果。

请见上实验报告

六、 思考题

在本实验的硬件电路中，能使用 C 口对 LED 指示灯控制码？如果可以写出 2 种控制方法（编程方法）。

在 8255 芯片中，A、B、C 三个端口分别对应 8255 的端口 A、端口 B、端口 C。对于控制 LED 指示灯，我们可以将数据发送到 C 端口。

以下是两种不同的编程方法，可以通过 C 口控制 LED 指示灯的控制码：

方法一：

在原始代码中，将数据发送到 C 端口（端口 C 的地址是 282H）：

```
MOV DX, 282H
MOV AL, 01H
OUT DX, AL
```

在这种方法中，将控制码数据直接发送到 C 端口。

方法二：

如果需要更多的控制，可以将控制码数据合并到 C 端口的原有数据中。在这种方法中，我们首先从 C 端口读取数据，然后修改其中的特定位，最后将修改后的数据发送回 C 端口。

```
MOV DX, 282H
IN AL, DX
OR AL, 01H
OUT DX, AL
```

在这种方法中，我们通过 OR 操作将特定位设置为 1，保持其他位不变。这种方式可以实现更加灵活的控制，可以修改多个位的状态。

注意 0：要结合学生实验指导书 pdf 文档来理解本实验。

注意 1：8 个数码管分别对应 S0-S7 等 8 个位码插孔。每个位码插孔接 5V 高电平时，则其对应的数码管被选中；接 0V 低电平时，则其对应的数码管未被选中。 当某个位码插孔接 5V 高电平时，则 8bit 的段码(a-dp)上的内容就会被输出到该位码插孔对应的数码管上显示出来。

提示：利用视觉驻留，进行多位的数码管显示。