东南大学

《微机实验及课程设计》

实验报告

实验五　8255 并行和串口输入输出

姓　　名： 邹滨阳　　**学　　号：** 08022305

专　　业： 自动化　　**实 验 室：** 金智楼 416

实验时间： 2024 年 4 月 23 日　**报告时间：** 2024 年 4 月 23 日

评定成绩：　　**审阅教师：**

实验五 8255 并行和串口输入输出

一. 实验目的与内容

实验目的:

1）掌握 8255 方式 0 的工作原理及使用方法，利用直接输入输出进行控制显示;

2）掌握 8 段数码管的动态刷新显示控制;

3）分析掌握 8255 工作方式 1 时的使用及编程，进一步掌握中断处理程序的编写。

4）了解串行通信的基本原理，认识串行收发过程;

5）掌握串行接口芯片 8251 的工作原理和编程方法。

实验内容:

1）按要求连接电路，使用 8255 方式 0 将 C 口作为输入，接逻辑电平开关 K0～K7，编程 A 口输出 LED 显示电路 L0～L7。将 A 口 L0-L7 控制成流水灯，流水间隔时间由软件产生，流水方向由 K0 键在线控制，随时可切换。流水间隔时间也可由 K4～K7 键编码控制，如 0000 对应停止，0001 对应 1 秒，1111 对应 15 秒，大键盘输入 ESC 键退出。

2）按要求连接电路，将 8255 的 A 口 PA0～PA6（JP6）分别与七段数码管的段码驱动输入端 a～DP 相连（JP3），位码驱动输入端 S1 接+5V（选中），S0 接地（关闭）。编程从键盘输入一位十进制数字（0～9），在七段数码管上显示出来。

3）按要求连接电路，七段数码管段码连接不变，位码驱动输入端 S1、S0 接 8255C 口的 PC1、PC0。编程在两个数码管上显示“56”。字符变换之间应使段位全灭，避免显示“影子”。

4）按照图 3.8-1 连接 8251 串行接口电路，利用 8253 计数器产生 8251 的发送和接收时钟。编写测试程序，初始化 8251，实现连续发送固定字符，用示波器观察发送波形。编程实现从键盘输入一个字符，将其 ASCII 码加 1 后发送出去，再接收回来在屏幕上显示，实现自发自收。

二. 基本实验原理

(1) 8255 方式 0：简单输入输出

实验使用了 8255 方式 0，将 C 口作为输入，连接逻辑电平开关 K0～K7，编程 A 口输出 LED 显示电路 L0～L7。通过这种方式，可以从 C 口输入数据，并从 A 口输出。

(2) 流水灯控制

编程将 A 口 L0-L7 控制成流水灯，流水间隔时间由软件产生，流水方向由 K0 键在线控制，流水间隔时间也可由 K4～K7 键编码控制。

(3) 8 段数码管静态和动态显示

静态显示：使用 8255 的 A 口 PA0～PA6 连接七段数码管的段码驱动输入端 a～DP，位码驱动输入端 S1 接+5V，S0 接地。编程从键盘输入一位十进制数字（0～9），在七段数码管上显示出来。

动态显示：七段数码管段码连接不变，位码驱动输入端 S1、S0 接 8255C 口的 PC1、PC0。编程在两个数码管上显示“56”。字符变换之间应使段位全灭，避免显示“影子”

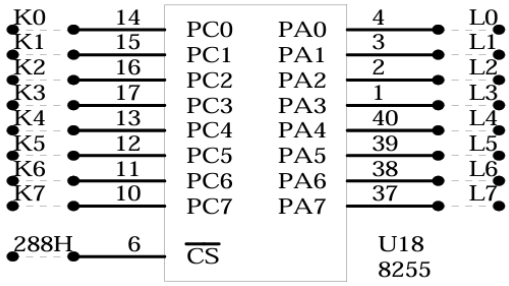
(4) 串行通信与 8251 芯片

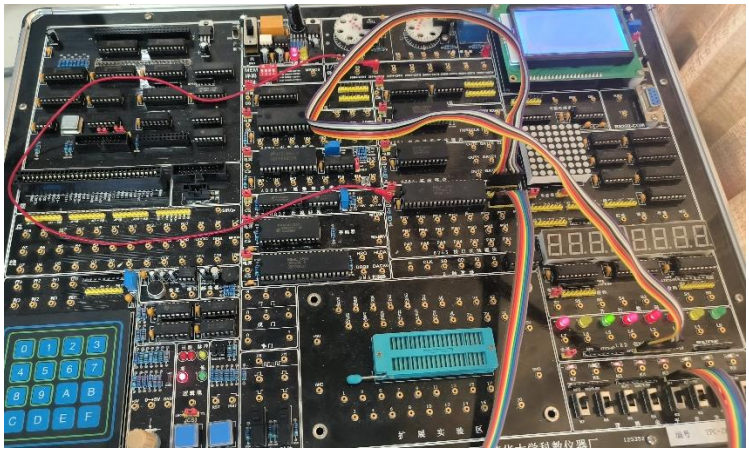
了解串行通信的基本原理，认识串行收发过程。掌握串行接口芯片 8251 的工作原理和编程方法。按照图 3.8-1 连接 8251 串行接口电路，利用 8253 计数器产生 8251 的发送和接收时钟。编写测试程序，初始化 8251，实现连续发送固定字符，用示波器观察发送波形。编程实现从键盘输入一个字符，将其 ASCII 码加 1 后发送出去，再接收回来在屏幕上显示，实现自发自收。

三. 方案实现与测试

1， 8255 方式 0：简单输入输出

按照电路图连接电路：



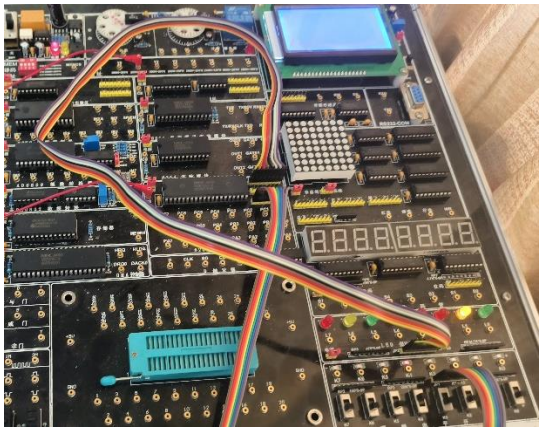


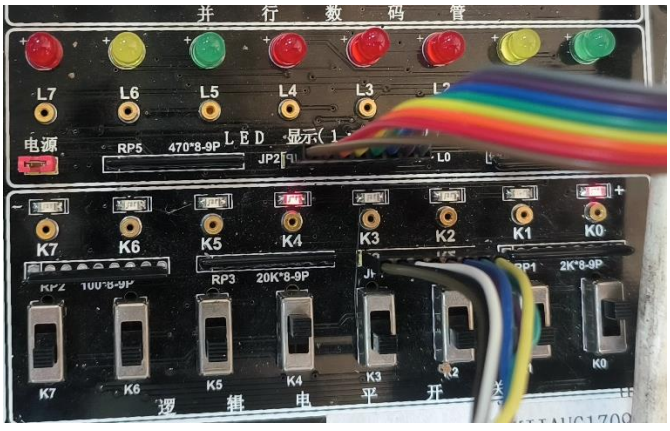
实验电路如图，8255C 口输入接逻辑电平开关 K0～K7，编程 A 口输出接 LED 显示电路 L0～L7；用指令从 C 口输入数据，再从 A 口输出。运行程序后发现验证成功。

```
io8255a    equ    288h
io8255c    equ    28ah
io8255k    equ    28bh
data       ends
code       segment
           assume cs:code,ds:data
start:     mov    ax,data
           mov    ds,ax
           mov    dx,io8255k          ;设8255为C口输入,A口输出
           mov    al,8bh
           out    dx,al
inout:     mov    dx,io8255c          ;从C口输入一数据
           in     al,dx
           mov    dx,io8255a          ;从A口输出刚才自C口
           out    dx,al               ;所输入的数据
           mov    dl,0ffh             ;判断是否有按键
           mov    ah,06h
           int    21h
           jz     inout               ;若无,则继续自C口输入,A口输出
           mov    ax,4c00h            ;否则返回
           int    21h
code       ends
           end    start
```

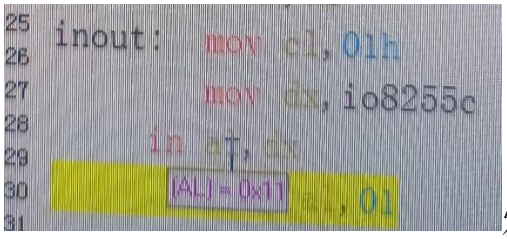
程序比较简单这里就不做调试分析了

（2）编程将 A 口 L0-L7 控制成流水灯，流水间隔时间由软件产生；流水方向由 K0 键在线控制，随时可切换；流水间隔时间也可由 K4～K7 键编码控制，如 0000 对应停止，0001 对应 1 秒，1111 对应 15 秒，大键盘输入 ESC 键退出。电路图如下，和之前的没有区别。

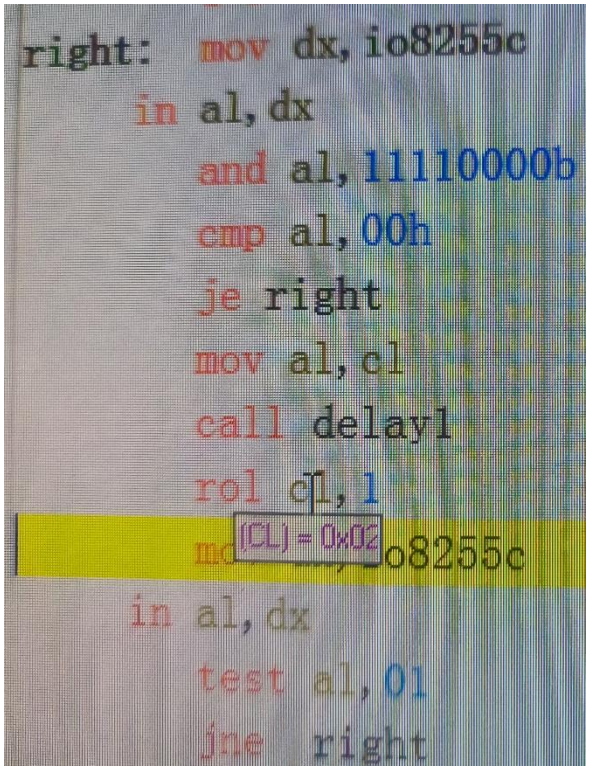




调试原代码：

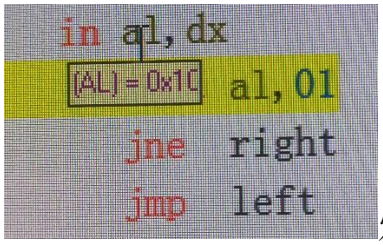


发现读取开关的读书为 00010001B=11H 符合图片

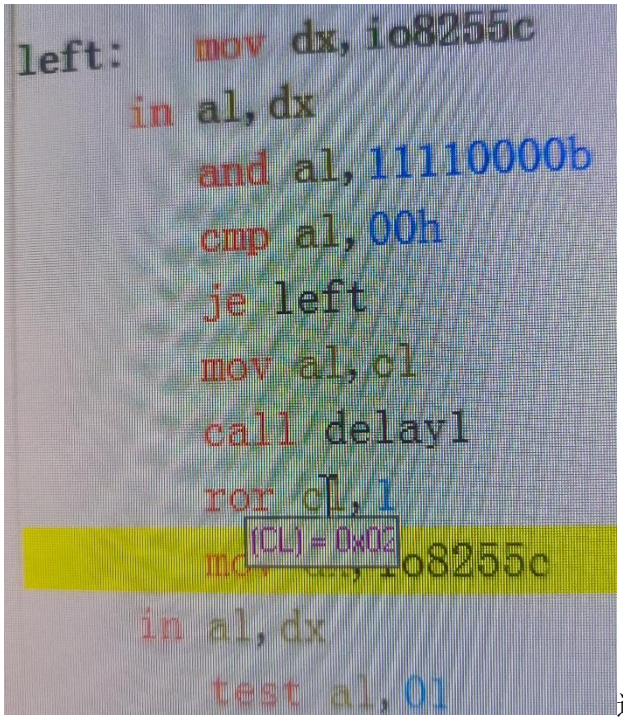


由于是最后 K0=1 所以是向右移动用 rol cl 实现

修改开关为 0001000B



发现 al 出现对应的更改



进入左移部分，用 ror cl 来处理流水灯的左移

整体解读：

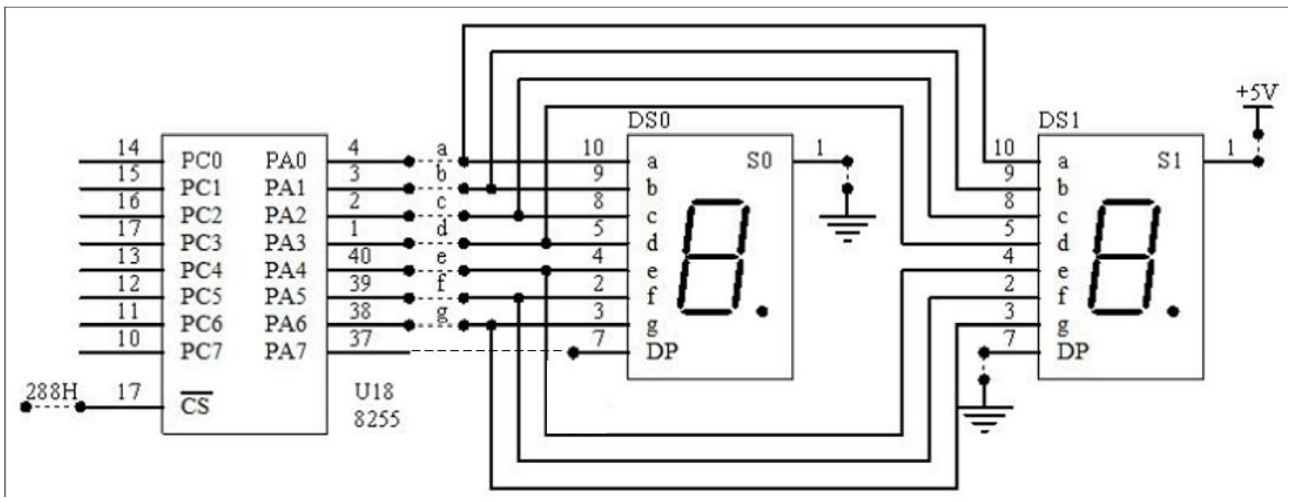
在这里，我们使用 CL 寄存器作为计数器。程序从 C 口读取数据以检测外部开关 K0 的状态。如果 K0 是高电平，流水灯将向右移动；如果是低电平，流水灯将向左移动。通过使用 ROL 和 ROR 指令，我们可以改变流水灯的移动方向。此外，我们使用 AND 和 CMP 指令来确定是否需要延时。

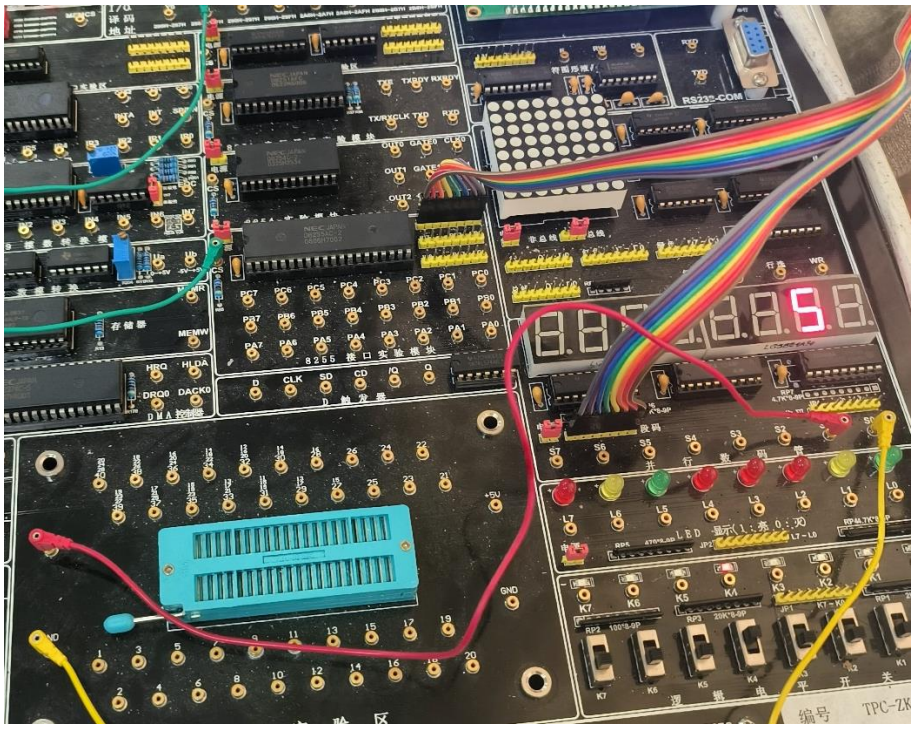
在延时子程序 delay1 中，我们将从 C 口读取的数据输出到 A 口，并调用 delay 子程序进行延时。之后，我们清空 A 口的数据，并再次调用 delay 子程序以进行额外的延时。

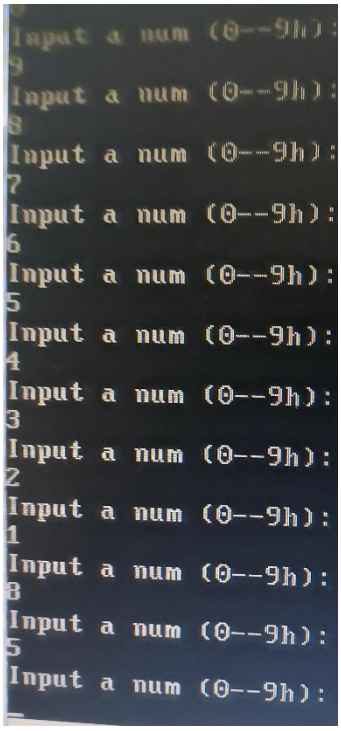
最后，延时子程序 DELAY 实现了一个简单的延时函数，用于流水灯的移动效果。

综上所述，这个程序通过读取 8255 的 C 口来检测外部开关 K0 的状态，并据此控制 LED 流水灯的方向和速度。流水灯的移动速度和方向通过程序中的延时和移位操作来控制。

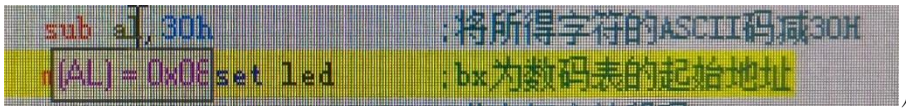
（3）8 段数码管静态显示:按图连接好电路，将 8255 的 A 口 PA0～PA6（JP6）分别与七段数码 管的段码驱动输入端 a～DP 相连（JP3），位码驱动输入端 S1 接+5V（选中），S0 接地（关闭）。编程从键盘输入一位十进制数字（0～9，在七段数码管上显示出来。



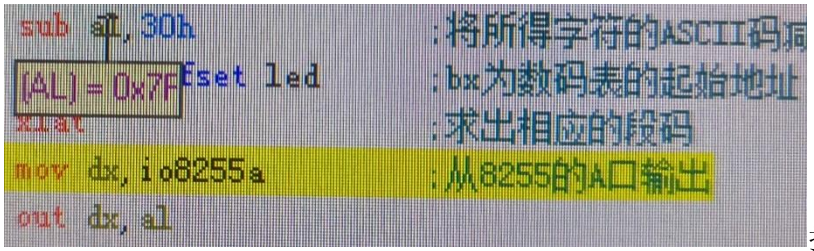




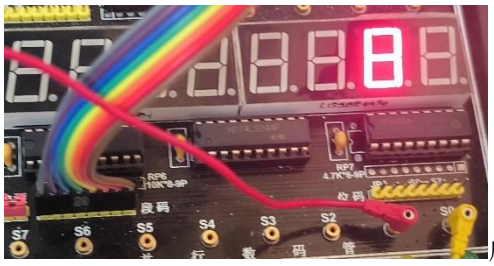
代码调试：



从键盘中得到需要输出的字符



把 al 转化成数码管的格式并输出

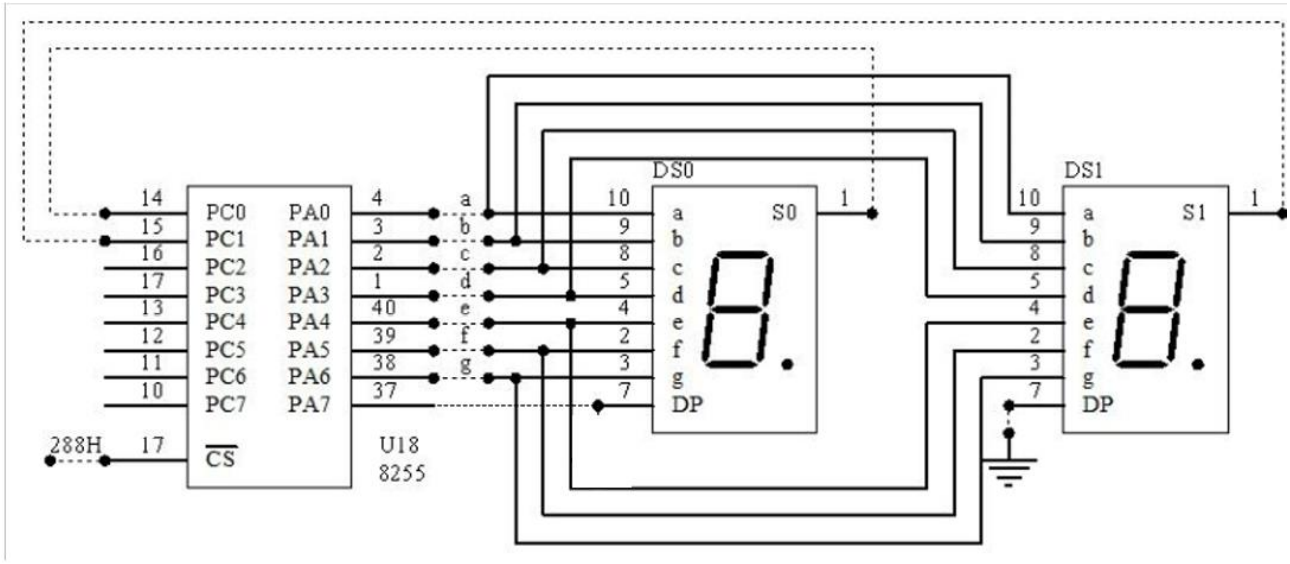


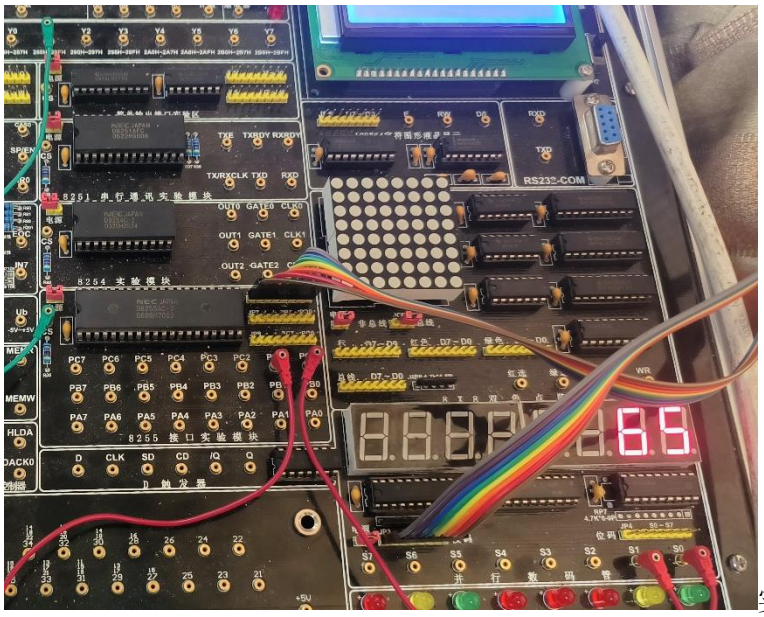
成功输出

代码整体解读：

这段代码用于从用户那里获取一个数字（0 到 9），并将该数字转换为七段 LED 的显示。首先，在数据段中定义了与 8255 接口芯片通信所需的端口地址、LED 的段码数据和提示信息。然后，在代码段中，程序配置了 8255 的 A 端口为输出模式，并进入一个循环，提示用户输入一个数字。用户输入的数字将转换为相应的七段 LED 段码，并通过 8255 的 A 端口显示。最后，程序结束并返回 DOS。

(4)8 段数码管动态显示：按图连接好电路，七段数码管段码连接不变，位码驱动输入端 S1、74 微机实验及课程设计教程 S0 接 8255C 口的 PC1、PC0。编程在两个数码管上显示“56”。（注意字符变换之间应使段位全灭，避免显示“影子”）。




实验验证成功

调试部分：

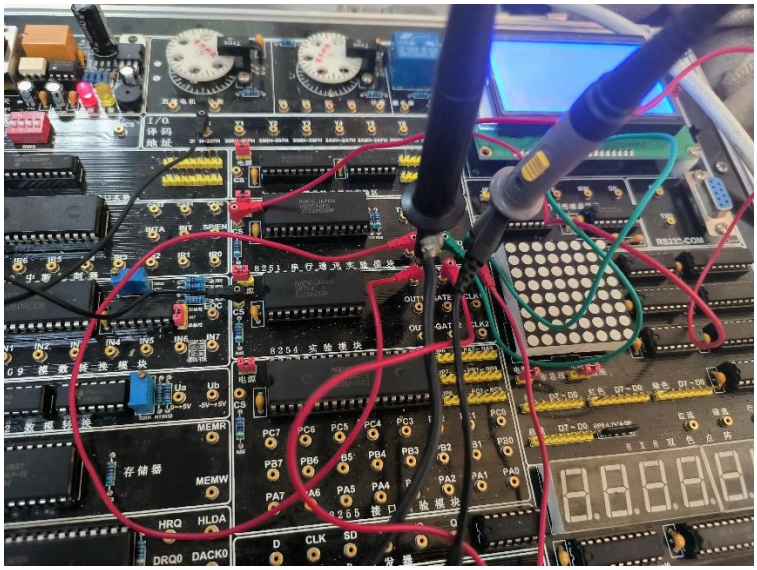
通过调试我们发现这个代码通过不断的调整 bh，bz 来让输出的数在 6 和 5 连续快速显示，从而表现出同时显示这两个数的效果。效果如下：

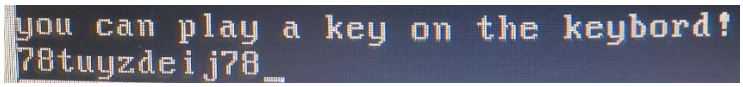
对这个代码的整体解读：

这段代码的主要功能是通过 8255 接口芯片控制七段 LED 显示器显示数字 5 和 6。代码首先在数据段定义了 8255 接口的端口地址、LED 的段码数据以及存放要显示的数字的缓冲区。在代码段开始，程序首先配置 8255 的 A 端口为输出模式，并设置显示缓冲区的初始地址。接着，程序进入一个循环，其中 bh 寄存器被设置为 2，代表要显示的数位是个位。在内部循环中，程序根据 di 寄存器的位置读取要显示的数字，并找到对应的 LED 段码。然后，程序通过 8255 的 A 端口将该段码输出到七段 LED 上。接下来，程序进行一个简单的延时操作，并将显示的数字切换到 6。此后，程序再次进行延时操作并检查是否有按键按下，如果有则退出循环，否则继续显示 5 和 6。最后，程序关闭 LED 显示并返回 DOS。

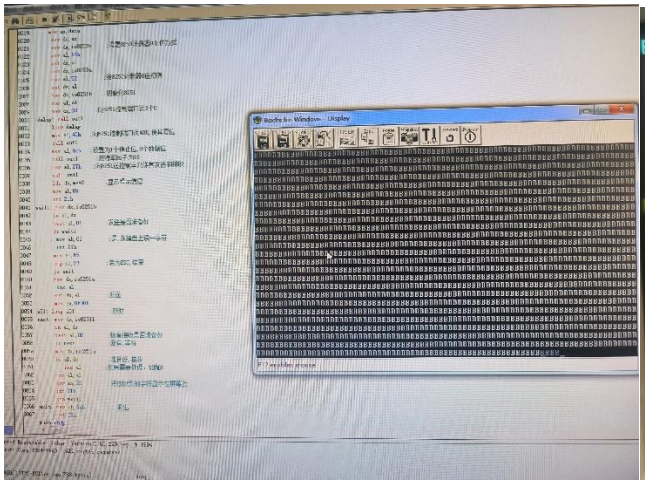
（5）按连接好电路 ，其中 8254 计数器用于产生 8251 的发送和接收时钟，TXD 和 RXD 直接相连（注意：实验系统未采用 MC1488/1489 或 ICL232 等长线收发器，故有关 RS232/422 等电平标准及转 换内容可自行参考相关资料)。编写测试程序，初始化 8251，实现连续发送固定字符，用示波器观察发送波形，试用多种不同 字符格式和字符。编程实现从键盘输入一个字符，将其 ASCII 码加 1 后发送出去,再接收回来在屏幕上显示，实现自发自收。

实验验证成功：

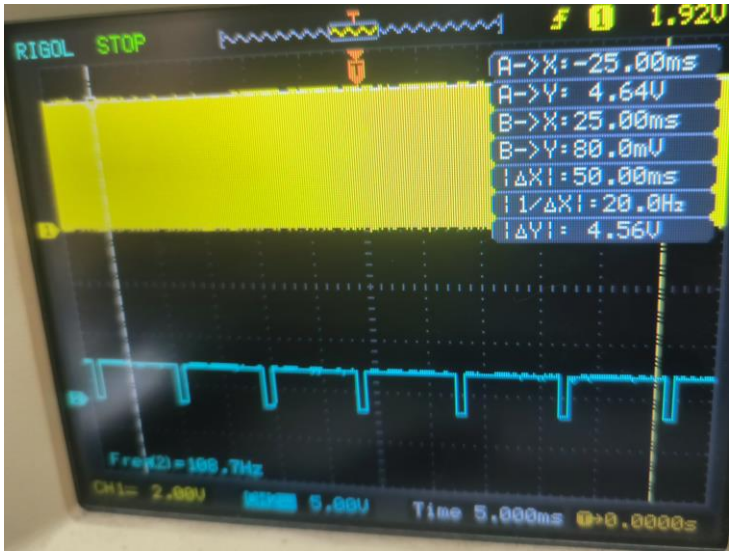




接下来用示波器验证：

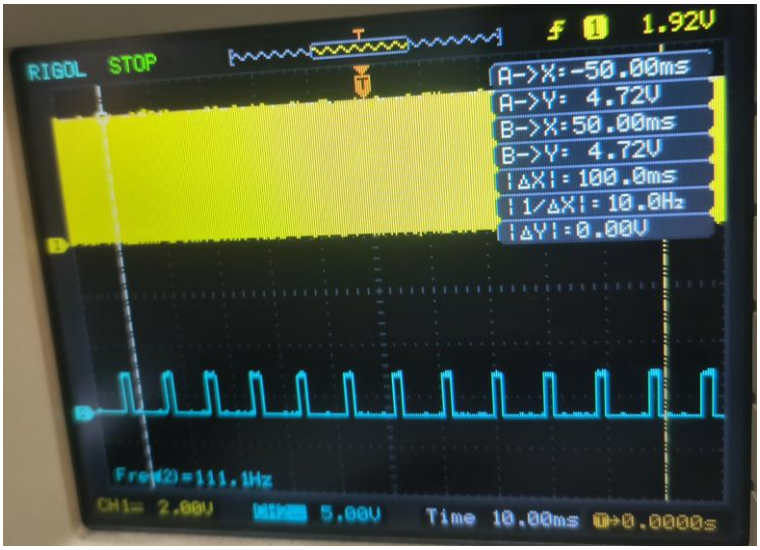


验证成功，接下来验证起始位和空闲位




```
mov al,0FFH
```

发现出现了明显的起始位




```
mov al,00H
```

发现出现了明显的空闲位

对代码的解读：

这段代码是一个基于 8251 串行通信芯片的程序，用于处理串行数据的发送和接收。程序首先设置了与 8253 和 8251 相关的 IO 端口地址。接着，它初始化了 8253 计数器 0 和 8251 串行通信芯片。在主程序的循环中，它检查是否可以发送数据，如果可以，它会读取键盘输入的字符并通过 8251 发送出去。同时，它也会检查是否有数据接收到，并将接收到的数据显示在屏幕上。程序使用了简单的延时方法来等待数据的准备和接收。当接收到的字符为 ESC 时，程序会退出。整体来说，这是一个基础的串行通信程序，用于键盘输入的发送和显示接收到的数据。

四．提高与创新研究

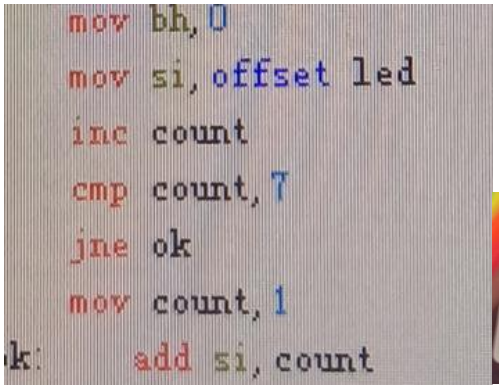
在图 3.7-3 的基础上，扩展到 8 个 8 段数码管，编程实现显示 123456

```
buffer1  db  1,2,3,4,5,6
```

修改暂存的数为 123456

修改 bh，让数码管能同时显示 6 位

最开始直接运行程序，我们发现实验上只显示 000421，这是因为原来是 add si,bz，而 bz 是成两倍变化的，所以我们引入新的变量并改写程序，从而实现了对 654321 的同时显示



五．分析与总结

本次实验主要围绕 8255 并行输入输出和 8251 串口通信进行了深入的探究和实践。实验的各个部分均成功实现了预期的功能，从简单的流水灯控制到复杂的串行通信，都得到了令人满意的结果。

在 8255 并行输入输出的实验中，我们学习了如何使用 8255 方式 0 进行基础的输入输出控制。通过配置 C 口为输入，我们成功地读取了逻辑电平开关的状态，并将这些数据通过 A 口输出到 LED 显示电路。流水灯的控制实现了从硬件到软件的完整控制链路，展示了如何通过编程实现实时的数据处理和显示。

对于七段数码管的静态和动态显示，我们进一步熟悉了 8255 的输出模式，并学习了如何在数码管上显示数字。通过对数字的转换和适当的延时，我们实现了预期的显示效果，这为我们后续的数字显示应用打下了坚实的基础。

在串行通信方面，我们深入理解了 8251 串口通信芯片的工作原理和编程方法。通过配置 8253 计数器产生 8251 的发送和接收时钟，我们实现了从键盘输入的字符的发送和接收，实现了基本的自发自收功能。通过示波器的观察，我们也验证了发送和接收的波形符合预期，这进一步加深了对串行通信的理解。

总的来说，本次实验不仅加深了我们对微机原理和接口芯片的理解，也提高了我们的实践能力和问题解决能力。在实验过程中，我们遇到了一些挑战，但通过不断的调试和优化，最终都得到了满意的解决。这次实验为我们今后的学习和研究提供了宝贵的经验和启示，使我们更加熟练地运用微机技术进行应用开发和创新研究。