东南大学电子科学与工程学院

实 验 报 告

课程名称：微机实验

实验名称：**交通灯控制实验 & I2C & 模/数转换器实验**

姓　　名：孙寒石　　学　　号：06219109

实验地点：东南大学无锡国际校区教二 205

实验时间：2021 年 12 月 21 日

评定成绩：_____　　审阅教师：_____

评　　语：_____

目录

交通灯控制实验

I2C 实验

模/数转换器实验

交通灯控制实验

1 实验目的

通过并行接口 8255 实现十字路口交通灯的模拟控制，进一步掌握对并行口的使用。

2 实验内容

如图 29，L7、L6、L5作为南北路口的交通灯与 PC7、PC6、PC5 相连，L2、L1、L0 作为东西路口的交通灯与PC2、PC1、PC0相连。编程使六个灯按交通灯变化规律亮灭。

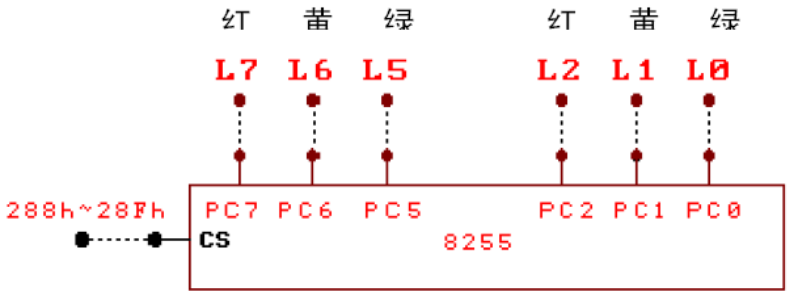


图 29

接线方法：

8255/JP8(PC7~PC0) 接 LED显示/JP2(L7~L0)

8255/CS 接 I/O译码/Y1(288H~28FH)

3 实验原理

3.1 接口

根据定义，8255 有 3 个通道 A、B、C 与外设连接，每个通道又有 8 根线与外设连接，所以

8255 可同时控制 24 路开关。各通道的引脚编号如下：

(1) A口：编号为 PA0~PA7，用于 8255 向外设输入输出 8 位并行数据。

(2) B口：编号为 PB0~PB7，用于 8255 向外设输入输出 8 位并行数据。

(3) C口：编号为 PC0~PC7，用于 8255 向外设输入输出8位并行数据，当 8255 工作于应答I/O方式时，C口用于应答信号的通信。

3.2 工作方式

(1) 方式0----基本输入输出方式

不需任何选通信号，A口、B口、高半C口、低半C口，者可被设定为输入或输出。

作输出口时输出数据存锁；作输入口时输入数据不存锁。

(2) 方式1----选通输入/出方式

A、B、C 三个口分为两组。

A组包括A口及高半C口，A口可编程设定为输入或输出，高半C口作I/O控制及同步信号；B组包括B口及低半C口，B口可编程设定为输入或输出，低半C口作I/O控制及同步信号；A口、B口的输入/输出数据都被存锁。

(3) 方式2----双向选通输入/输出方式

A口（仅A口）作 8 位双向总线，C口的 PC3～PC7 位用作I/O控制及同步信号；B口及C口的 PC0～PC2 可编程设定为方式0或方式1工作。

3.3 引脚定义

• RESET：复位输入线，当该输入端处于高电平时，所有内部寄存器(包括控制寄存器)均被清除，所有I/O口均被置成输入方式。

•CS：芯片选择信号线，当这个输入引脚为低电平时,即/CS=0时,表示芯片被选中，允许8255与CPU进行通讯;/CS=1时,8255无法与CPU做数据传输.

•RD：读信号线，当这个输入引脚为低跳变沿时,即/RD产生一个低脉冲且/CS=0时,允许8255通过数据总线向CPU发送数据或状态信息，即CPU从8255读取信息或数据。

• WR：写入信号，当这个输入引脚为低跳变沿时,即/WR产生一个低脉冲且/CS=0时,允许CPU将数据或控制字写入8255。

• D0~D7：三态双向数据总线，8255与CPU数据传送的通道，当CPU 执行输入输出指令时，通过它实现8位数据的读/写操作，控制字和状态信息也通过数据总线传送。

• PA0~PA7：端口A输入输出线，一个8位的数据输出锁存器/缓冲器，一个8位的数据输入锁存器。工作于三种方式中的任何一种；

• PB0~PB7：端口B输入输出线，一个8位的I/O锁存器，一个8位的输入输出缓冲器。不能工作于方式二；

• PC0~PC7：端口C输入输出线，一个8位的数据输出锁存器/缓冲器，一个8位的数据输入缓冲器。端口C可以通过工作方式设定而分成2个4位的端口，每个4位的端口包含一个4位的锁存器，分别与端口A和端口B配合使用，可作为控制信号输出或状态信号输入端口。不能工作于方式一或二。

• A1，A0：地址选择线，用来选择8255的PA口，PB口，PC口和控制寄存器。

4 实验方案

4.1 十字路口交通灯变化要求

下载程序至实验板后，六盏交通灯按照题目要求的规律循环闪烁，即：

（1）南北路口的绿灯、东西路口的红灯同时亮1秒。

（2）南北路口的黄灯闪烁若干次，同时东西路口的红灯继续亮。

（3）南北路口的红灯、东西路口的绿灯同时亮1秒。

（4）南北路口的红灯继续亮、同时东西路口的黄灯亮闪烁若干次。

（5）转（1）重复。

4.2 流程图

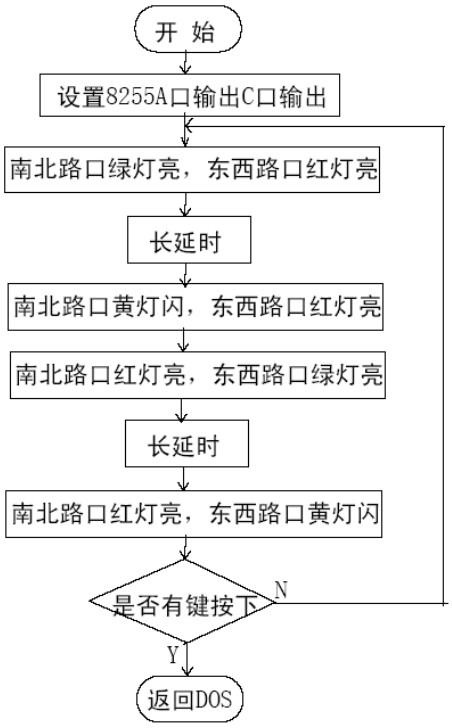


图 30

5 实验结果

5.1 代码编写

```
;*************************************;
;*       十字路口红绿灯模拟演示程序      *;
;* 端口各灯的设置:                        *;
;*     1红  1黄  1绿  0 0 2红  2黄  2绿  *;
;*************************************;
data segment
io8255a             equ 28ah
io8255b             equ 28bh
io8255c         equ 288h
portc1    db    24h,44h,04h,44h,04h,44h,04h              ;六个灯可能
   db    81h,82h,80h,82h,80h,82h,80h                    ;的状态数据
   db    0ffh                                           ;结束标志
portb1    db    4fh,3fh
```

```
data ends
code    segment
        assume    cs:code,ds:data
start:
   mov      ax,data
   mov      ds,ax
   mov      dx,io8255b
   mov      al,90h
   out      dx,al                ;设置8255为C口输出
   mov      dx,io8255a
re_on:  mov      bx,0
on:     mov      al,portc1[bx]
   cmp      al,0ffh
   jz       re_on
   out      dx,al                ;点亮相应的灯
      cmp    al,24h
      jz       aa
      cmp    al,81h
jz       aa
move    al,portb1[1]
move    dx,io8255b
out dx,al
   inc      bx
   mov      cx,200               ;参数赋初值
   test    al,21h               ;是否有绿灯亮
   jz       de1                  ;没有,短延时
   mov      cx,20000         ;有,长延时
de1:    mov      di,20000             ;di赋初值
de0:    dec      di                   ;减1计数
   jnz      de0                  ;di不为0
   loop    de1
   push dx
   mov ah,06h
   mov dl,0ffh
   int 21h
   pop dx
   jz    on                     ;没有,转到on
exit:    mov      ah,4ch               ;返回
   int      21h
code ends
end start
```

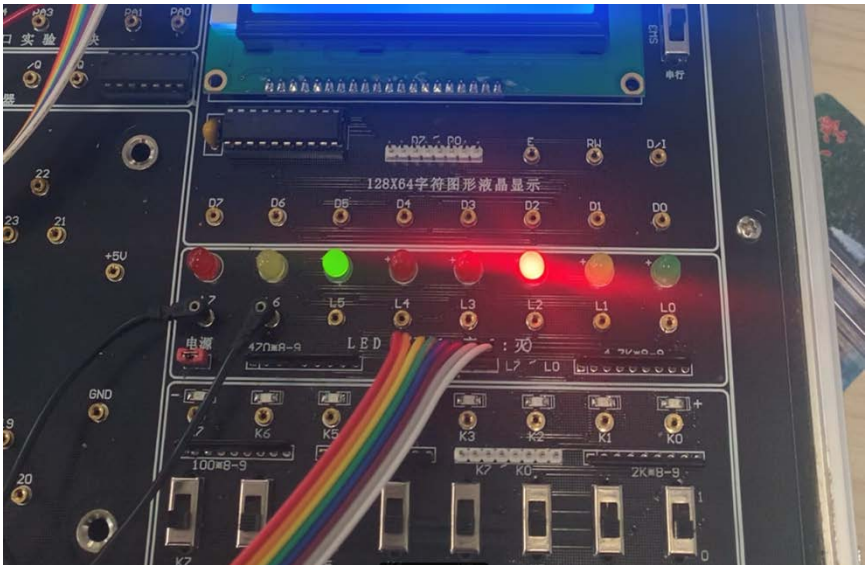
5.2 硬件连接

如图所示：

5.3 实验现象

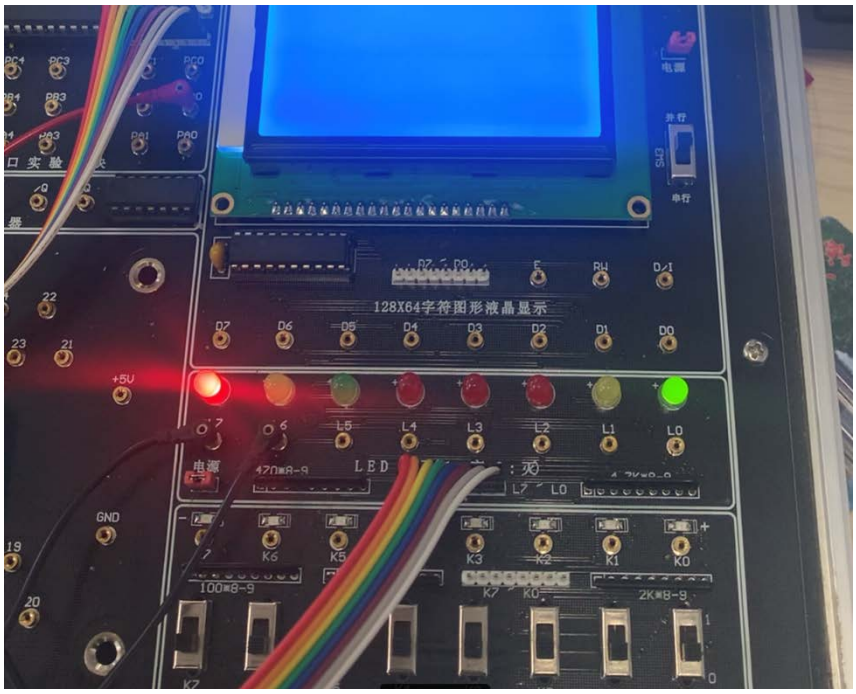
下载程序至实验板后，六盏交通灯按照题目要求的规律循环闪烁，即：

（1）南北路口的绿灯、东西路口的红灯同时亮1秒。

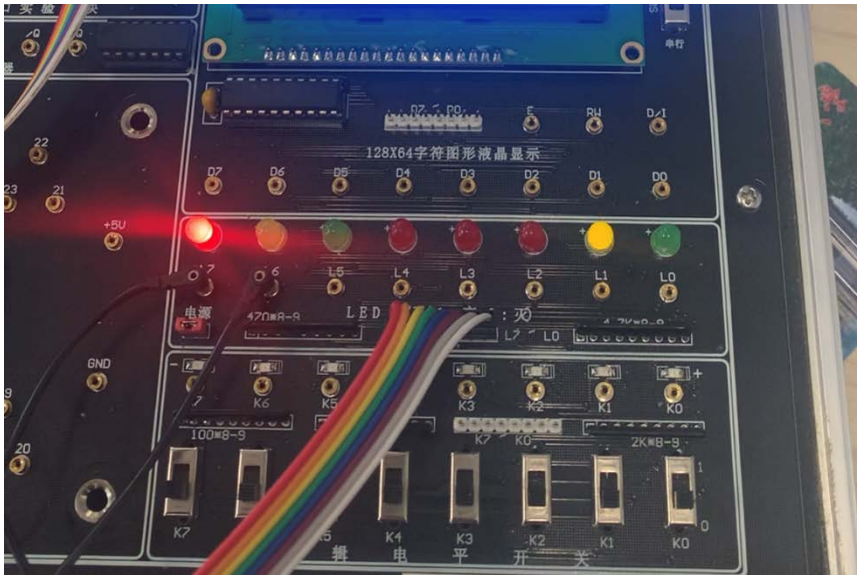


（2）南北路口的黄灯闪烁若干次，同时东西路口的红灯继续亮。

（3）南北路口的红灯、东西路口的绿灯同时亮1秒。



（4）南北路口的红灯继续亮、同时东西路口的黄灯亮闪烁若干次。



（5）转（1）重复。

6 思考题

1、讨论延时时间的长短对显示效果有何影响？

答：代码中对 CX 的赋值决定了长延时和短延时，延时时间长则红绿灯的持续时长以及黄灯闪烁的间隔时长就会变长。

2、在方式 1 时，INTA、INTB 作为中断使能信号，在输入输出时，分别由 C 口的哪二位进行置位和复位的？

答：输入时 INTA 写入 PC4，INTB 写入 PC2，输出时 INTA 写入 PC6，INTB 写入 PC2。

7 实验小结

通过这次微机实验，我学习并掌握了并行接口8255A的内部结构，功能及编程，进一步掌握了对并行口的使用。最后实验结果与理论预期情况相符，但也有一些小问题，延时时间只能在一定范围内，

不能过长，且延时时间由系统主频决定。其实，更加科学合理的办法是用8253计数器控制时间间隔，但这样做略有复杂，在本次试验中不是很必要。

I2C 实验

1 实验目的和要求

为学生提出一个新的实验内容让学生利用已有的实验平台，完成一个新的实验训练。充分发挥学生的 聪明才智，提高知识的运用能力和创新意识。 在现有的实验平台上，通过硬件（即合理的接线）和软件设计完成微机系统对 24C02 串行总线存储器

的读与写的操作。即，先对存储器的一个地址写入数据，再对同一个地址读数据，当写入数据与读出数据一致时表示实验成功。

简单来说，就是使用 I2C 存储器模块对 8086 系统进行拓展，掌握 I2C 时序，并进行相关编程对 I2C 存储器进行读写。

2 实验原理及目的

2.1 关于 I2C 通讯协议的调研

I2C 通讯协议（**Inter-Integrated Circuit**）是由 Phiilps 公司开发的，由于它引脚少，硬件实现简单，可扩展性强，不需要 USART、CAN 等通讯协议的外部收发设备，现在被广泛地使用在系统内多个集成电路(IC)间的通讯。

在计算机科学里，大部分复杂的问题都可以通过分层来简化。对于通讯协议，我们也以分层的方式来理解，最基本的是把它分为物理层和协议层。物理层规定通讯系统中具有机械、电子功能部分的特性，确保原始数据在物理媒体的传输。协议层主要规定通讯逻辑，统一收发双方的数据打包、解包标准。

物理层硬件拓扑

它的物理层有如下特点：

(1) 它是一个支持设备的总线。“总线”指多个设备共用的信号线。在一个 I2C 通讯总线中，可连接多个 I2C 通讯设备，支持多个通讯主机及多个通讯从机。

(2) 一个 I2C 总线只使用两条总线线路，一条双向串行数据线(SDA) ，一条串行时钟线 (SCL)。数据线即用来表示数据，时钟线用于数据收发同步。

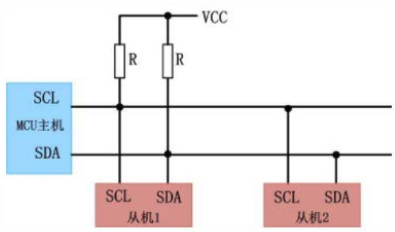
(3) 每个连接到总线的设备都有一个独立的地址，主机可以利用这个地址进行不同设备之间的访问。

(4) 总线通过上拉电阻接到电源。当 I2C 设备空闲时，会输出高阻态，而当所有设备都空闲，都输出高阻态时，由上拉电阻把总线拉成高电平。

(5) 多个主机同时使用总线时，为了防止数据冲突，会利用仲裁方式决定由哪个设备占用总线。

(6) 具有三种传输模式：标准模式传输速率为 100kbit/s ，快速模式为 400kbit/s ，高速模式下可达 3.4Mbit/s，但目前大多 I 2C 设备尚不支持高速模式。

(7) 连接到相同总线的 IC 数量受到总线的最大电容 400pF 限制 。



软件协议层

I2C 基本读写过程

I2C 的协议定义了通讯的起始和停止信号、数据有效性、响应、仲裁、时钟同步和地址广播等环节。

图 24-2 主机写数据到从机

图 24-3 主机由从机中读数据

图 24-4 I2C 通讯复合格式

图例：数据由主机传输至从机 S：传输开始信号

SLAVE_ADDRESS: 从机地址

数据由从机传输至主机 R/W：传输方向选择位，1 为读，0 为写

A/A：应答(ACK)或非应答(NACK)信号

P：停止传输信号

这些图表示的是主机和从机通讯时，SDA 线的数据包序列。

其中 S 表示由主机的 I2C 接口产生的传输起始信号(S)，这时连接到 I2C 总线上的所有从机都会接收到这个信号。

起始信号产生后，所有从机就开始等待主机紧接下来 广播 的从机地址信号(SLAVE_ADDRESS)。在 I2C 总线上，每个设备的地址都是唯一的，当主机广播的地址与某个设备地址相同时，这个设备就被选中了，没被选中的设备将会忽略之后的数据信号。根据 I2C 协议，这个从机地址可以是 7 位或 10 位。

在地址位之后，是传输方向的选择位，该位为 0 时，表示后面的数据传输方向是由主机传输至从机，即主机向从机写数据。该位为 1 时，则相反，即主机由从机读数据。

从机接收到匹配的地址后，主机或从机会返回一个应答(ACK)或非应答(NACK)信号，只有接收

到应答信号后，主机才能继续发送或接收数据。

写数据：若配置的方向传输位为“写数据”方向，即第一幅图的情况，广播完地址，接收到应答信号后，主机开始正式向从机传输数据(DATA)，数据包的大小为 8 位，主机每发送完一个字节数据，都要等待从机的应答信号(ACK)，重复这个过程，可以向从机传输 N 个数据，这个 N 没有大小限制。当数据传输结束时，主机向从机发送一个停止传输信号(P)，表示不再传输数据。

读数据：若配置的方向传输位为“读数据”方向，即第二幅图的情况，广播完地址，接收到应答信号后，从机开始向主机返回数据(DATA)，数据包大小也为 8 位，从机每发送完一个数据，都会等待主机的应答信号(ACK)，重复这个过程，可以返回 N 个数据，这个 N 也没有大小限制。当主机希望停止接收数据时，就向从机返回一个非应答信号(NACK)，则从机自动停止数据传输。

读和写数据：除了基本的读写，I2C 通讯更常用的是复合格式，即第三幅图的情况，该传输过程有两次起始信号(S)。一般在第一次传输中，主机通 SLAVE_ADDRESS 寻找到从设备后，发送一段“数据”，这段数据通常用于表示从设备内部的寄存器或存储器地址(注意区分它与 SLAVE_ADDRESS 的区别)；在第二次的传输中，对该地址的内容进行读或写。也就是说，第一次通讯是告诉从机读写地址，第二次则是读写的实际内容。

以上通讯流程中包含的各个信号分解如下：

通讯的起始和停止信号

前文中提到的起始(S)和停止(P)信号是两种特殊的状态，如图。当 SCL 线是高电平时 SDA 线从高电平向低电平切换，这个情况表示通讯的起始。当 SCL 是高电平时 SDA 线由低电平向高电平切换，表示通讯的停止。起始和停止信号一般由主机产生。

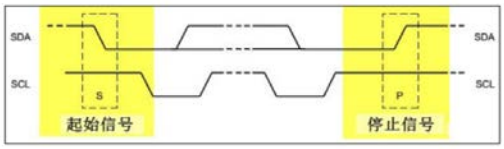


图 24-5 起始和停止信号

数据有效性

I2C 使用 SDA 信号线来传输数据，使用 SCL 信号线进行数据同步。如图，SDA 数据线在 SCL 的每个时钟周期传输一位数据。传输时，SCL 为高电平的时候 SDA 表示的数据有效，即此时的 SDA 为高电平时表示数据“1”，为低电平时表示数据“0”。当 SCL 为低电平时，SDA 的数据无效，一般在这个时候 SDA 进行电平切换，为下一次表示数据做好准备。

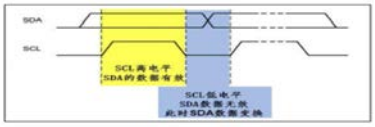


图 24-6 数据有效性

地址及数据方向

I2C 总线上的每个设备都有自己的独立地址，主机发起通讯时，通过 SDA 信号线发送设备地址(SLAVE_ADDRESS)来查找从机。I2C 协议规定设备地址可以是 7 位或 10 位，实际中 7 位的地址应用比较广泛。紧跟设备地址的一个数据位用来表示数据传输方向，它是数据方向位，第 8 位或第 11 位。数据方向位为“1”时表示主机由从机读数据，该位为“0”时表示主机向从机写数据。

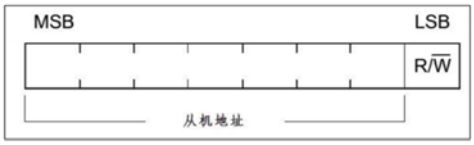
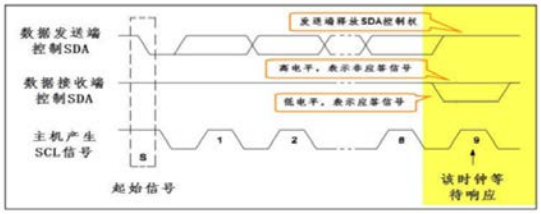


图 24-7 设备地址(7 位)及数据传输方向

响应

I2C 的数据和地址传输都带响应。响应包括“应答(ACK)”和“非应答(NACK)”两种信号。作为数据接收端时，当设备(无论主从机)接收到 I2C 传输的一个字节数据或地址后，若希望对方继续发送数据，则需要向对方发送“应答(ACK)”信号，发送方会继续发送下一个数据；若接收端希望结束数据传输，则向对方发送“非应答(NACK)”信号，发送方接收到该信号后会产生一个停止信号，结束信号传输，如图。



软件 I2C 和硬件 I2C

想要控制产生 I2C 方式的通讯，可以采用软件模拟或硬件 I2C 这两种方式。所谓软件模拟，即直接使用 CPU 内核按照 I2C 协议的要求控制 GPIO 输出高低电平。如控制产生 I2C 的起始信号时，先控制作为 SCL 线的 GPIO 引脚输出高电平，然后控制作为 SDA 线的 GPIO 引脚在此期间完成由高电平至低电平的切换，最后再控制 SCL 线切换为低电平，这样就输出了一个标准的 I2C 起始信号。

而硬件 I2C 是指直接利用芯片中的硬件 I2C 外设，该硬件 I2C 外设跟 USART 串口外设类似，只要配置好对应的寄存器，外设就会产生标准串口协议的时序。使用它的 I2C 外设则可以方便地通过外设寄存器产生 I2C 协议方式的通讯，如初始化好 I2C 外设后，只需要把某寄存器位置 1，那么外设就会控制对应的 SCL 及 SDA 线自动产生 I2C 起始信号，而不需要内核直接控制引脚的电平。

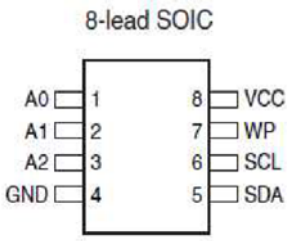
相对来说，硬件 I2C 直接使用外设来控制引脚，可以减轻 CPU 的负担。不过使用硬件 I2C 时

必须使用某些固定的引脚作为 SCL 和 SDA，软件模拟 I2C 则可以使用任意 GPIO 引脚，相对比较灵活。

2.2 关于 24C02 芯片的相关介绍

24C02 芯片是 EEPROM 芯片的一种，它是基于 IIC 总线的存储器件，遵循二线制 I2C 协议，在电路的作 用主要是在掉电的情况下保存数据。实验中使用的 24C02 芯片，总容量是 2K bits（256 bytes）。这里芯片名称里的 02 代表着总容量。

AT24C02 工作电压是低压的存储芯片，串行电可擦除只读存储器有 2K 的容量，数学计算下来就是 256 个字节，其中每个字节是 8bit，这种存储器在各种领域的运用非常的广泛。该芯片工作电压是 1.8V～5.5V，其输入输出的引脚电压兼容 5V 电压，硬件连接电路是二线串行接口，对 I2C 需要很深刻的了解，内部结构为 256*8(2K)，每个输入引脚都有经施密特触发器，通过施密特触发器来抑制噪声，I2C 接口是双向的传输数据接口，工作频率兼容 400KHz，有写保护功能，而且效率高，可靠性高，擦写次数多，保存时间长的优点，其擦写次数大概是 1000000 次，保存时间大概是 100 年之久。



A0-A2 为地址输入线，悬空时默认为 0，当 A2、A1、A0 均为 0 时，24C02 的写地址（主机向 24C02 写）为 A0H，读地址为 A1H。WP 为写保护，这里不需要用到，接地或者悬空。GND、VCC 提供地线和电源。SCL 为时钟线，SDA 为数据线，这两个引脚分别与 I2C 总线的 SCL 和 SDA 连接，并且都需要外接上拉电阻提供空闲时的高电平。

对 24C02 芯片进行写字节操作的时候，步骤如下：

• 开始位，后面紧跟从器件地址位（0xA0），等待应答，这是为了在 IIC 总线上确定 24C02 的从地址位置；

• 确定操作 24C02 的地址，等待应答，也就是将字节写入到 24C02 中 256 个字节中的位置；确定需要写入 24C02 芯片的字节，等待应答，停止位。

对 24C02 芯片进行读字节操作的时候，步骤如下：

• 开始位，后面紧跟从器件地址位（0xA0），等待应答，这是为了在 IIC 总线上确定 24C02 的从地 址位置；

• 确定操作 24C02 的地址，等待应答，也就是从 24C02 中 256 个字节中读取字节的位置；

• 再次开始位，后面紧跟从器件地址位（0xA1），等待应答；

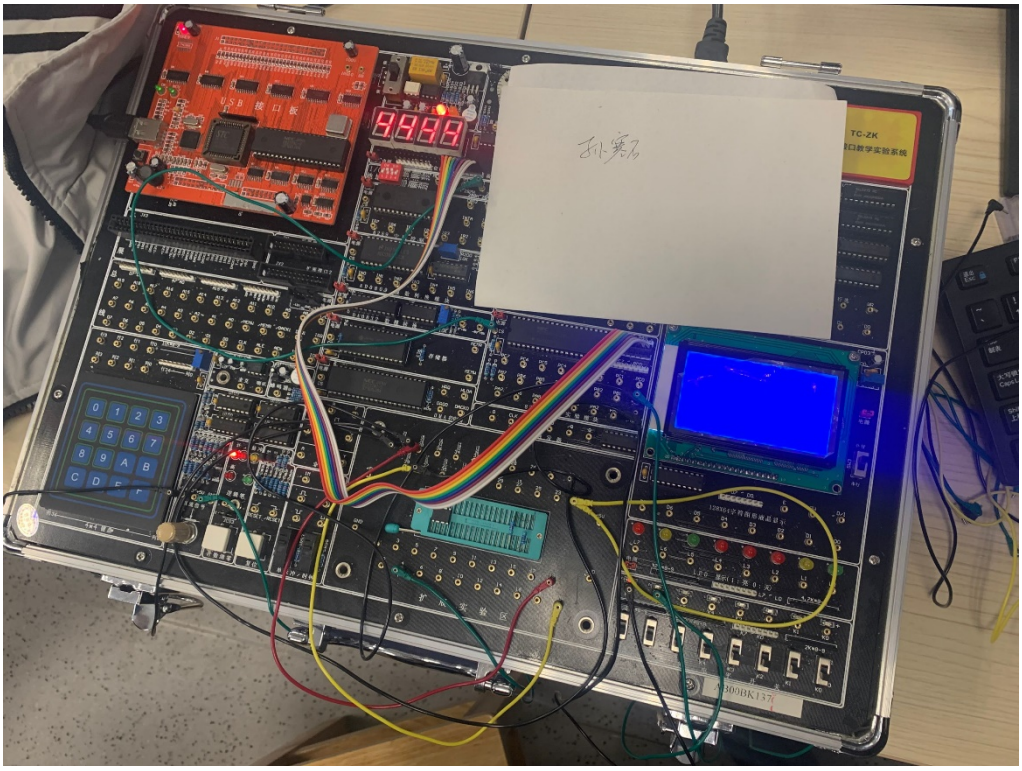
• 获取从 24C02 芯片中读取的字节，发出非应答信号，停止位。

3 实验方案

3.1 硬件连线

由于在本实验中只有微机一个主机和 24C02 一个从机，因此 SCL 固定为主机输出、SDA 则为双向。于是本方案中使用 8255 来分别对 SCL 和 SDA 单独进行控制：设置 B 口为输出后让 PB0 作 SCL，A 口的 PA0 作 SDA，模式在输入和输出之间切换。另外，我们还接入了两个电阻，为了防止电流过大烧毁芯片。

具体连线如下图所示：



3.2 软件编程设计

```
 SDA     EQU      288H;8255A 口地址,CS-Y1
LEDEQU      289H;8255B 口地址
SCL EQU 28AH;8255C 口地址
I8255B EQU 28BH;8255 控制寄存器地址
DATA SEGMENT

BUFFERSEND DB      00H;存放要写的数据
```

```
COUNTSEND DB 50
BUFFERREC DB  20 DUP(0);存放读出的数据
COUNTREC DB   50
COUNT1DW 1
LED1     DB   3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH,77H,7CH,39H,5EH,79H,71H;段码

NUM      DB 00H  ;存储器的地址
ADDROM     DB 00H  ;存储器的地址
DATA ENDS
STACKS SEGMENT STACK
DW 256 DUP (?)
STACKS ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA,SS:STACKS

MAIN PROC FAR
START1: MOV AX,STACKS
MOV SS,AX
MOV AX,DATA
MOV DS,AX
;初始化各段基址
LOOPP:
MOV DX,I8255B
MOV AL,80H
MOV DX,LED
MOV AL,00h
OUT DX,AL
 ;设置 8255 为 A 端口、B 端口和 C 端口输出

;BUFFERREC[DI],B 口初始化
MOV DI,0;
MOV NUM,0
CALL READ
;从存储器中读数
CALL DISPLAY1

PUSH CX
MOV CX,20
HONG:
INC NUM
;数加 1 在写到存储器中
```

```
CMP NUM ,10
JNC DONE
JMP NEXT2
DONE:    MOV NUM,0
NEXT2:CALL WRITE
PUSH CX
MOV CX,50

ting1:
CALL DELAY
LOOP ting1
POP CX
CALL READ ;读出并显示到数码管
CALL DISPLAY1
PUSH CX
MOV CX,50
HONG1:
CALL DELAY
LOOP HONG1
POP CX
LOOP HONG
POP CX
MOV AH,4CH
INT 21H
MAIN ENDP

;延时：
DELAY   PROC
PUSH BX
PUSH CX
MOV BX,2000
ZZZ:       MOV CX,1
ZZ:  LOOP ZZ
DEC BX
JNZ ZZZ
POP CX
POP BX
RET
DELAY ENDP
DISPLAY PROC
PUSH DX
MOV DX,I8255B
```

```
MOV AL,80H ;设置 8255 为 A 端口、B 端口和 C 端口输出
CALL DELAY
MOV DX,LED
MOV AL,BUFFERREC[DI]
OUT DX,AL
PUSH CX
MOV CX,200
;BUFFERREC[DI]是入参
zuod1:

CALL DELAY
LOOP zuod1
POP CX
POP DX
RET
DISPLAY ENDP
DISPLAY1 PROC ;数码管显示
PUSH BX
PUSH SI
PUSH CX
PUSH AX
lll:   MOV BL,NUM
MOV BH,0;
MOV SI,OFFSET LED1;
ADD SI,BX
MOV AL,BYTE PTR[SI];
MOV DX,LED
OUT DX,AL
MOV AL,02H
MOV DX,SCL
OUT DX,AL
MOV CX,8000

YS:  CALL DELAY;
LOOP YS
HONG2: LOOP HONG2
POP AX
POP CX
POP SI
POP BX
RET
```

```
DISPLAY1 ENDP
;启动信号
START PROC
CALL DELAY
MOV DX,I8255B
MOV AL,80H ;设置 8255 为 A 端口和 C 端口输出
OUT DX,AL
MOV AL,01H
PUSH DX
MOV DX,SC
OUT DX,Al
POP DX
;CALL DELAY;

MOV AL,01H
PUSH DX
MOV DX,SDA
OUT DX,Al
POP DX
CALL DELAY;
CALL DELAY;
PUSH DX
MOV AL,0
MOV DX,SDA
OUT DX,AL
POP DX
CALL DELAY;
CALL DELAY;
PUSH DX
MOV DX,SCL
MOV AL,0
OUT DX,AL
POP DX
CALL DELAY;
RET
START ENDP

;停止信号
STOP PROC
MOV DX,I8255B
MOV AL,80H ;设置 8255 为 A 端口和 C 端口输出
OUT DX,AL
```

```
MOV AL,00H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
MOV AL,01H
PUSH DX
MOV DX,SCL
OUT DX,AL
POP DX
CALL DELAY
CALL DELAY
MOV AL,01H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
CALL DELAY
CALL DELAY
RET
STOP ENDP

;应答：sda 低电平时间大于高电平时间
ACK PROC
MOV DX,I8255B
MOV AL,80H ;设置 8255 为 A 端口和 C 端口输出
OUT DX,AL
CALL DELAY
MOV AL,00H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
CALL DELAY
MOV AL,01H
PUSH DX
MOV DX,SCL
OUT DX,AL
POP DX
CALL DELAY
MOV AL,00H
PUSH DX
```

```
MOV DX,SCL
OUT DX,AL
POP DX
CALL DELAY
MOV AL,01H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
RET
ACK ENDP

;非应答：
NOACK PROC
MOV DX,I8255B
MOV AL,80H ;设置 8255 为 A 端口和 C 端口输出
OUT DX,AL
CALL DELAY
MOV AL,01H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
CALL DELAY
MOV AL,01H
PUSH DX
MOV DX,SCL
OUT DX,AL
POP DX
CALL DELAY
MOV AL ,00H
PUSH DX
MOV DX,SCL
OUT DX,AL
POP DX
CALL DELAY
MOV AL,00H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
RET
```

```
NOACK ENDP

;按位发送数据,一个字节，入参 BL
SEND PROC
PUSH CX
MOV CX,8
LOOP1:
CALL DELAY

SHL BL,1      ;Bl 中保存要发送的数据
JC SDA1
MOV AL,00H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
JMP NEXT
SDA1:    MOV AL,01H
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
NEXT: CALL DELAY
MOV AL,01H
PUSH DX
MOV DX,SCL
OUT DX,AL
POP DX
CALL DELAY
MOV AL,00H
PUSH DX
MOV DX,SCL
OUT DX,AL
POP DX
SUB CX,1
CMP CX,0
JNZ ZUO2
JMP ZUO3
ZUO2:
JMP LOOP1
;写完一个字节之后将 SDA,SCL 拉高
ZUO3:   MOV AL,01H
```

```
PUSH DX
MOV DX,SDA
OUT DX,AL
POP DX
CALL DELAY
POP CX
RET
SEND ENDP

;按位接收数据,，入参 DI，保存至 BUFFERREC[DI]
RECE PROC
PUSH BX
PUSH CX
XOR BX,BX
MOV CX,8
MOV AL,01H
PUSH DX
MOV DX,SDA
CALL DELAY
OUT DX,AL

;sda=1
POP DX
MOV DX,I8255B
MOV AL,090H
OUT DX,AL

;8255 初始化，A 端口输入，C 端口输出
MOV AL,1
LOOP2:SHL BH,1
PUSH DX;
MOV DX,SCL;
OUT DX,AL

;BH 保存读入的数据
;scl=1
POP DX;
CALL DELAY
PUSH DX
xor al,al
MOV DX,SDA
```

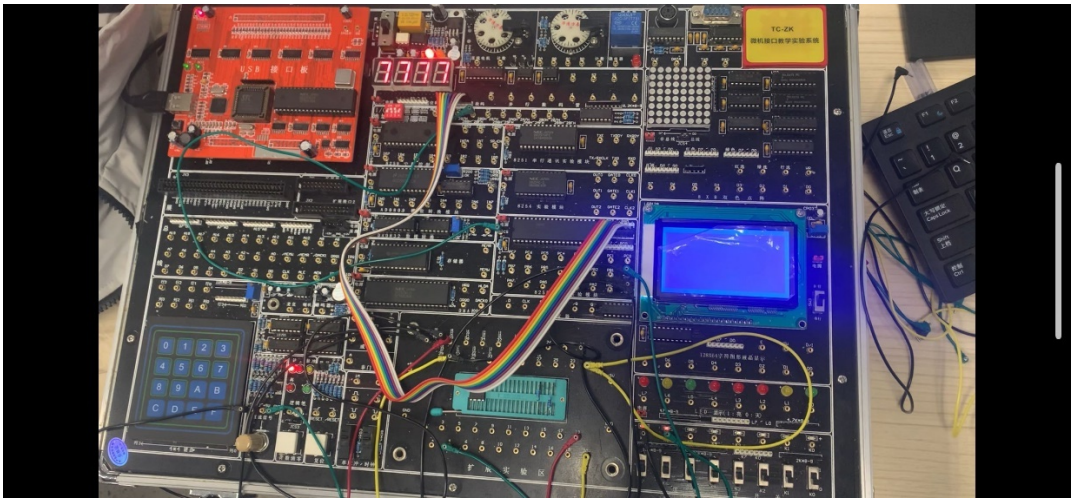
```
;读入 SDA,加到 BH 上
IN AL,DX
POP DX
AND AL,01H
ADD BH,AL
CALL DELAY
MOV AL,00h
PUSH DX
MOV DX,SCL
OUT DX,AL
POP DX
;scl=0，sda=1

MOV AL,01h
PUSH DX
MOV DX,SDA
OUT DX,AL
CALL DELAY

POP DX
LOOP LOOP2
MOV BUFFERREC[DI],BH ;将读入的数据保存至 BUFFERREC[DI]
POP CX
POP BX
RET
RECE ENDP
;对存储器字节一个写  ;入参 ADDROM，NUM
WRITE   PROC
CALL START
MOV   BL,0A0H  ;BL 中保存的是发送命令+芯片编号+P0+W(器件地址)
CALL SEND
CALL ACK
MOV BL,ADDROM    ;BL 保存存储器内部地址
CALL SEND
CALL ACK
MOV   BL,NUM  ;BL 中保存要发送的数据
CALL SEND
CALL ACK
CALL STOP
CALL DELAY
RET
WRITE ENDP
```

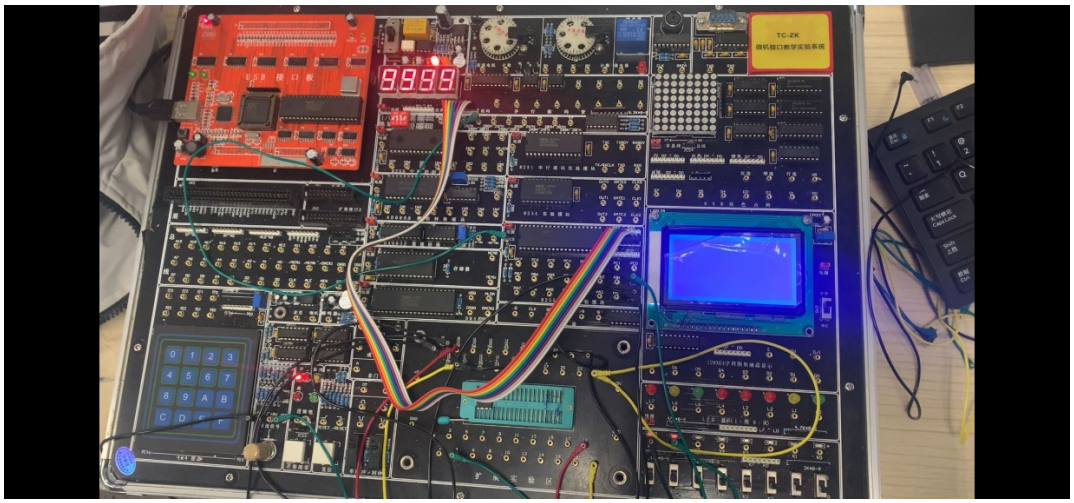
```
;对存储器页写（本次实验不要求）

YEWRITE PROC
push cx
MOV CX,COUNT1 ;数据个数
MOV SI,0
CALL START
MOV     BL,0A0H;BL 中保存的是发送命令+芯片编号+P0+W
CALL SEND
CALL ACK
MOV     BL,ADDROM
CALL SEND
CALL ACK
LOOP5:MOV BL,BUFFERSEND[SI]
CALL SEND
CALL ACK
CALL STOP
CALL DELAY
pop cx
RET
YEWRITE ENDP  ;对存储器字节读  入参 ADDROM，DI,出参保存至 BUFFERREC[DI]
READ PROC
CALL START
MOV BL,0A0H    ;BL 是 SEND 的入参,BL 中保存的是发送命令+芯片编号+P0+W
CALL SEND
CALL ACK
MOV BL,ADDROM    ;BL 保存寄存器地址
CALL SEND
CALL ACK
CALL START
MOV BL,0A1H
 ;BL 中保存的是发送命令+芯片编号+P0+R
CALL SEND
CALL ACK
MOV DI,0
CALL RECE
;入参 DI，保存至 BUFFERREC[DI]
CALL NOACK
CALL STOP
PUSH CX
MOV CX,500
ting2:
```

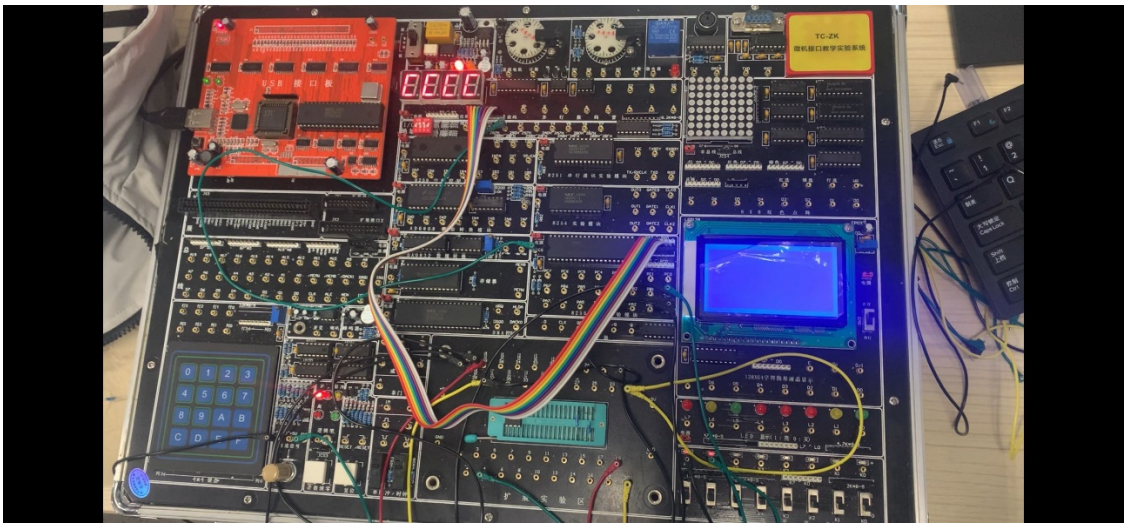
```
CALL DELAY
LOOP ting2
POP CX
CALL DELAY
RET
READ ENDP
CODE ENDS

END START1
```

4 实验数据及功能实现

现象：先从 0 写到 F，再读出，写（读）到哪个数就在数码管上显示哪个数。经过两个周期，就结束。所以数码管上是 4 个相同的数字（因为是静态显示），从 0~F，循环两次。







5 实验中出现的问题及解决方法

- 实验中，由于缺少电阻，所以使用了模电实验剩下的电阻。
- 采购芯片时没有注意到封装的形式，刚开始买成了贴片电阻，应该买直插 DIP 8 的。
- 实验箱上面的拓展接口有问题，换了同学的另外一个试验箱才成功。
- 代码方面调试 DEBUG

6 实验结果

通过点亮数码管可以知道目前读写的数字是多少，可以确定程序没有问题，可以正常运行，理论上已经成功读写，因为读写以及显示的地址就是存储器地址，使用其它地址会出错。

本次实验借鉴了上一届给出的 I2C 实验，在此基础上成功做了出来。为了不烧毁芯片，我们更是加上了电阻，限制了电流。上一届宝贵的资料给了我们充分的灵感和经验，在此表示感谢！没有前人的经验，我们做这个实验会更加困难。

7 从学长报告中汲取的灵感

我之前接触过 51 单片机，但是 51 是用 C 语言写的，和汇编还是有一定的区别的。而且，51 单片机可以直接用 GPIO 接口实现模拟 I2C 通信协议，这一点是比实验箱好很多的。前期，在网上也是看到了很多关于 51 单片机模拟 I2C 通信协议的实验参考，但是关于实验箱平台的方案却是十分之少了。由于互联网上没有使用 8088 微机实验箱的 I2C 方案，缺乏 TPC-PCI 实验箱电路连接和内部原理图的参考资料。所以只能结合两个芯片的数据手册以及微机理论来构建方案。在一段时间中，我十分

灰心，认为这个实验着实是有些超出能力范围了。

在刚开始，我是想要做从键盘中输入一个数，然后经过 DOS 中断扫描输入一个字符（0~F 之间的），之后再将通过 I2C 总线协议其写入了其 I2C 协议的芯片中。最后，再从芯片中读出所写入的数据，显示在数码管上。但是，在中断的时候，不知道为什么出现了一些奇怪的问题，于是我开始阅读学长学姐们的报告，希望可以汲取一些成功的经验。

在第一份报告中，作者提出了一个很重要的事情：进入读取循环之前先拉低 SCL，在每读一个 bit 时，先拉高 SCL 后 24C02 将数据发到 SDA 上，等待一个 WAITNOP，主机读取此时的 SDA，然后时钟拉低，然后读取下一个 bit。这个操作成功纠正了这样一个错误：存储器的写入和读出不一致。读出值发生了一些错误，错误规律以及发生读写的现象，认为错误大概率发生在读取字节的子程序 RDBYTE，因为此段程序中，读入 SDA 发生在 SCL=0 期间，而不是时序规定的 SCL=1 的数据有效期间。而错误发生在发送字节子程序的概率较小，因为发送器件的读写地址而非数据时也使用了同一个发送字节子程序，而发生“在将读取字节地址修改为与写入字节地址不一样时，读取数据不随写入数据值的变化而变化”这一现象需要能够正确地发送地址。

在实验中，我也遇到了类似的问题，于是，参考了这个方法，纠正了一些，但是还是有一些其他奇怪的 BUG。

在第二份报告中，我获得了指导性的意见！第二位学长化繁为简，他不再像我一样想要从键盘键入某个数字，然后经过写入再读出这类过程。而是从一开始，就将所要输入和读出的信息写好，那么便是 0~9 的七段数码管号。当然，我也吸取了他在实验过程中出现的错误——实验箱上是 A1A0 连接 8255 的 CS，转换时忽略了这一点，导致片选出现问题，无法使数码管正确点亮。还有就是需要注意实验箱上数码管为共阴极，而仿真时使用的是共阳极数码管，段码需要修改。

但是，第二个学长用的器件和我们用的出现了一些差距，他用了一个硬件底座，并且用类似于杜邦线的东西（看不清）在拓展接口上接出去，由于我并没有提前购置相应的器件，所以无法复现他的实验。于是我转向第一个实验的接线，利用了拓展接口，配合大二模电实验中的剩余电阻来限制电流大小，从而成功地实现了这一实验。

需要注意的是，第一位学长并未成功实现功能，我结合了第二位学长的思路和代码参考，成功达到了实验的目的。

8 实验小结

本次实验，从前期对于 I2C 总线协议的调研，到之后的芯片采购，再到程序的 DEBUG 和硬件的实现，都是一次勇敢的尝试。刚开始，网络上大多数只有 51 单片机相关实现 I2C 的一些实验参考，使得我灰心丧气，但是在看了学长的报告之后，我有了灵感，如同醍醐灌顶！同时，要感谢王老师的辛勤教导，不仅微机理论有课时，实验课还要看着我们，辛苦了！

模/数转换器实验

1 实验目的

了解模/数转换器的基本原理，掌握ADC0809芯片的使用方法。

2 实验内容

通过实验台左下角电位器RW1 输出0～5Ｖ直流电压送入ADC0809

通道0(IN0),利用debug 的输出命令启动Ａ/Ｄ转换器，输入命令读取转换结果，验证输入电压与转换后数字的关系。

启动IN0 开始转换: Out 0298 0

读取转换结果: In 0298

1、显示数据

编程采集IN0 输入的电压,在屏幕上显示出转换后的数据（用16 进制数）。

2、显示波形

将JP3 的1、2 短接，使IN2 处于双极性工作方式，并给IN1 输入一个低频交流信号（幅度为±５Ｖ），编程采集这个信号数据并在屏幕上显示波形。

3 实验原理与接线

3.1 实验原理

（1）ADC0809的IN0口地址为298H，IN1口地址为299H。

（2）IN0单极性输入电压与转换后的数字的关系为：

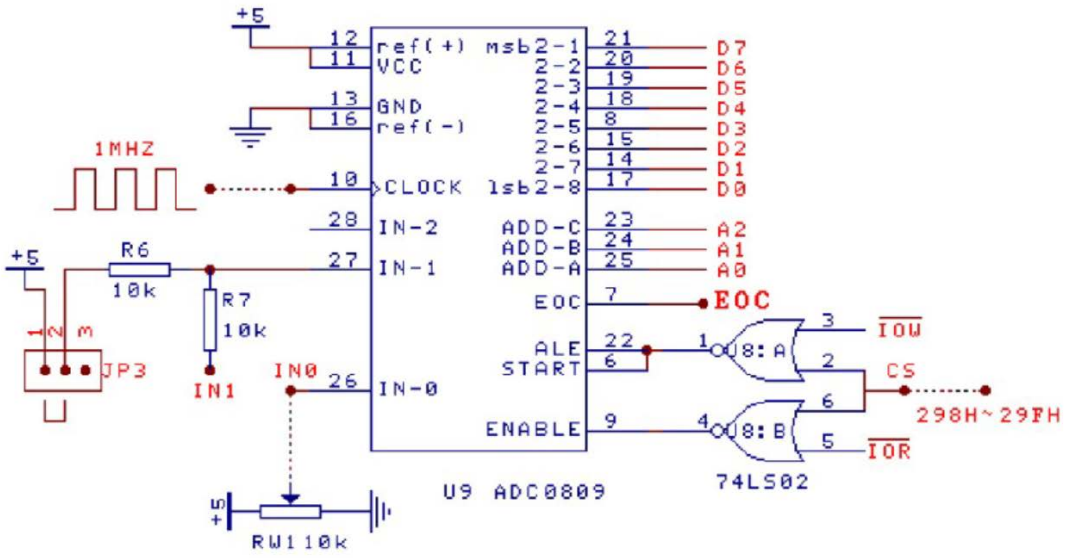
$$N = \frac{U_i}{U_{REF}/256}$$

（3）一次A/D转换的程序可以为：

```
MOV DX,口地址
OUT DX,AL          ;启动转换
;延时
IN AL,DX           ;读取转换结果放在AL中
```

3.2 实验接线图

（1）A/D转换显示数据



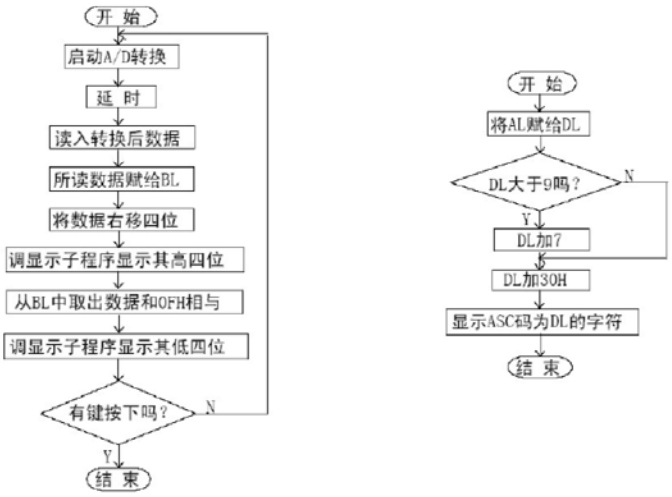
（2）A/D转换显示波形

将JP3 的1、2 短接，使IN2 处于双极性工作方式，并给IN1 输入一个低频交流信号（幅度为±5 V）。

4 实验方案

4.1A/D 转换显示数据

流程图：



(A) 主程序 图 39 (B) 显示子程序

4.2A/D 转换显示波形

流程图：

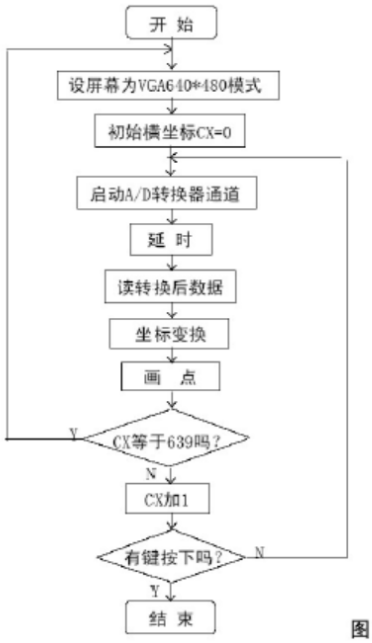


图 40

5 实验结果

5.1 A/D 转换显示数据

5.1.1 代码编写

```
io0809a     equ 298h
code  segment
    assume cs:code
start:mov  dx,io0809a        ;启动 A/D 转换器
    out  dx,al
    mov  cx,0ffh        ;延时
delay:loop delay
    in   al,dx          ;从 A/D 转换器输入数据
    mov  bl,al          ;将 AL 保存到 BL
    mov  cl,4
    shr  al,cl          ;将 AL 右移四位
    call disp           ;调显示子程序显示其高四位
    mov  al,bl
    and  al,0fh
    call disp           ;调显示子程序显示其低四位
    mov  ah,02
    mov  dl,20h         ;加回车符
    int  21h
    mov  dl,20h
    int  21h
    push dx
    mov  ah,06h           ;判断是否有键按下
    mov  dl,0ffh
    int 21h
    pop dx
    je   start          ;若没有转 START
    mov  ah,4ch         ;退出
    int  21h
disp  proc near           ;显示子程序
    mov  dl,al
    cmp  dl,9           ;比较 DL 是否>9
    jle  ddd            ;若不大于则为'0'-'9',加 30h 为其 ASCII 码
    add  dl,7           ;否则为'A'-'F',再加 7
```

```
ddd:    add  dl,30h            ;显示
    mov  ah,02
    int  21h
    ret
disp endp
code ends
end start
```

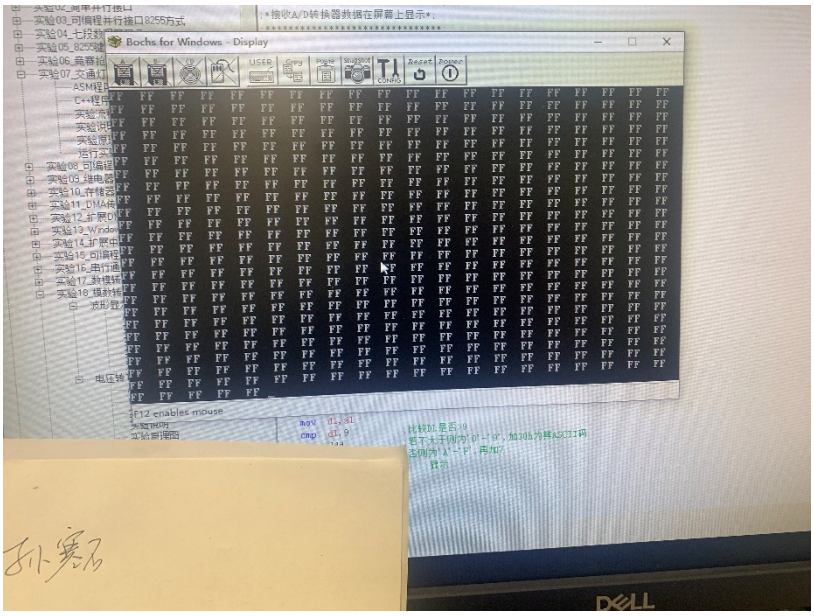
5.1.2 实验现象

下载程序至实验板后，顺时针转动实验板上电压输出旋钮，可输出0-5V的直流电压，旋钮如下：

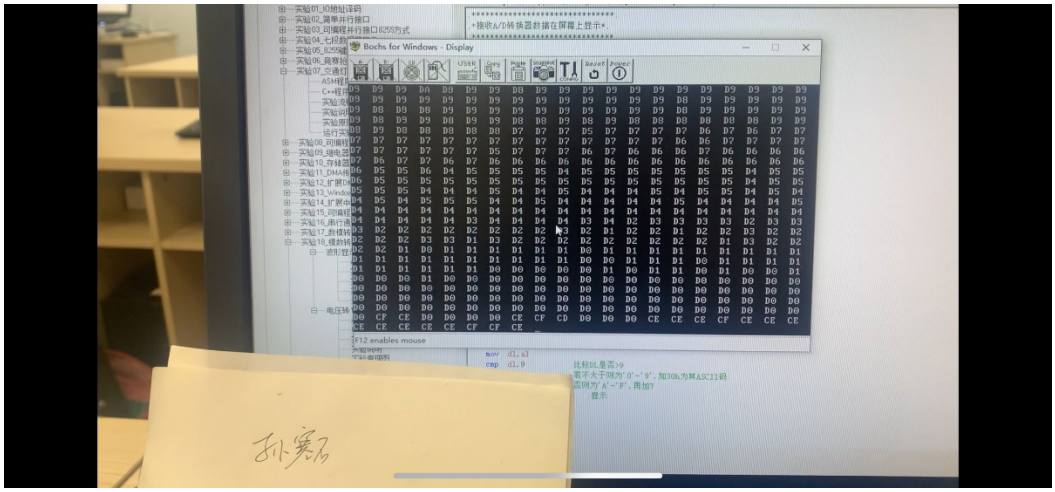
在显示屏上可看到数值在不断刷新，随着旋钮顺时针旋转，输出电压升高，屏幕上显示数值变大，且满足关系：

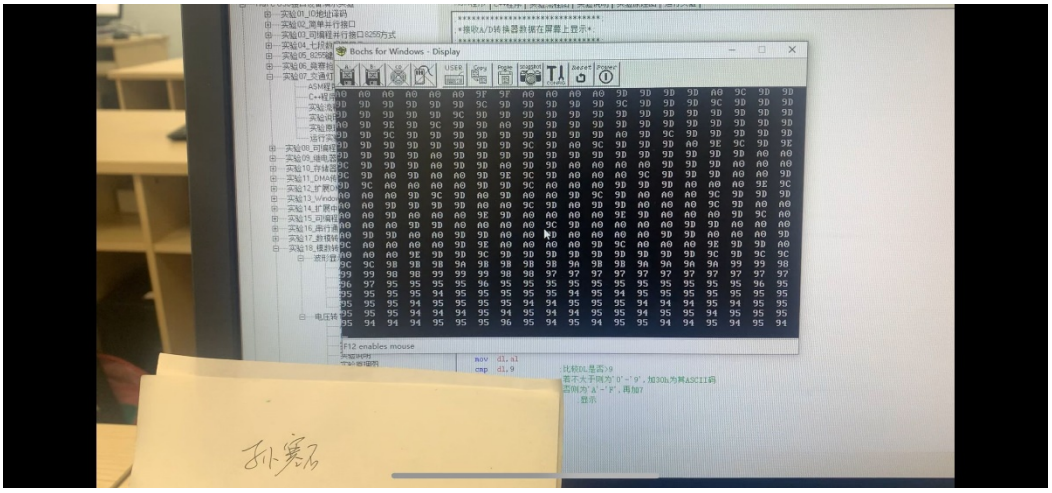
$$N = \frac{U_i}{5/256}$$

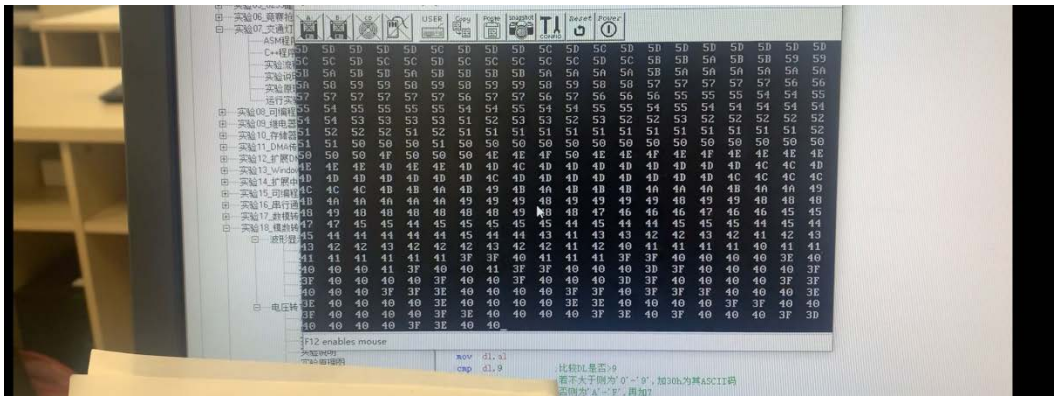
当旋钮所转角度最大时，显示数据为十六进制ff，即十进制256：



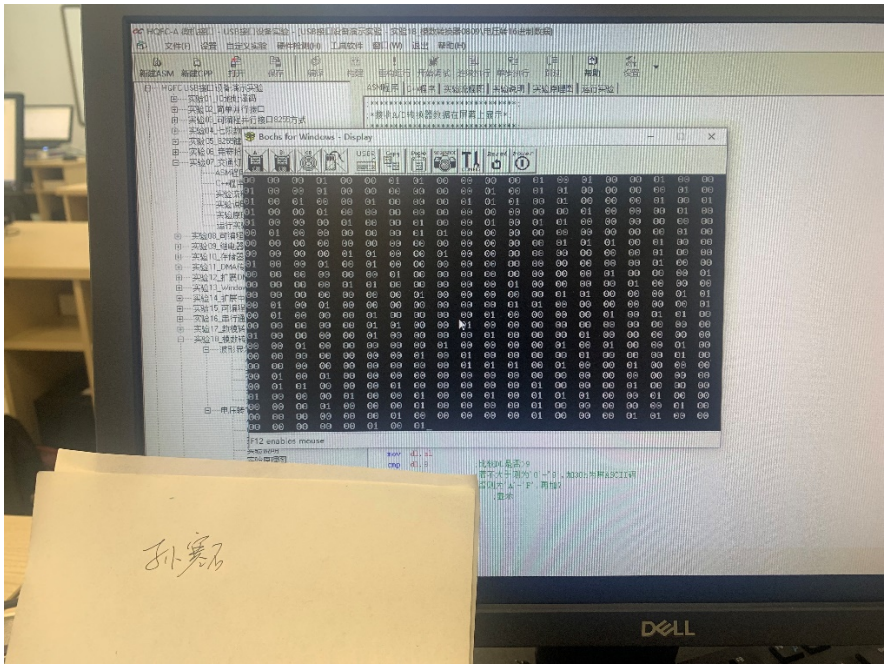
随着旋钮旋转，数值变化如图：







当旋钮所转角度最小时，显示数据为 00：



5.2 A/D 转换显示波形

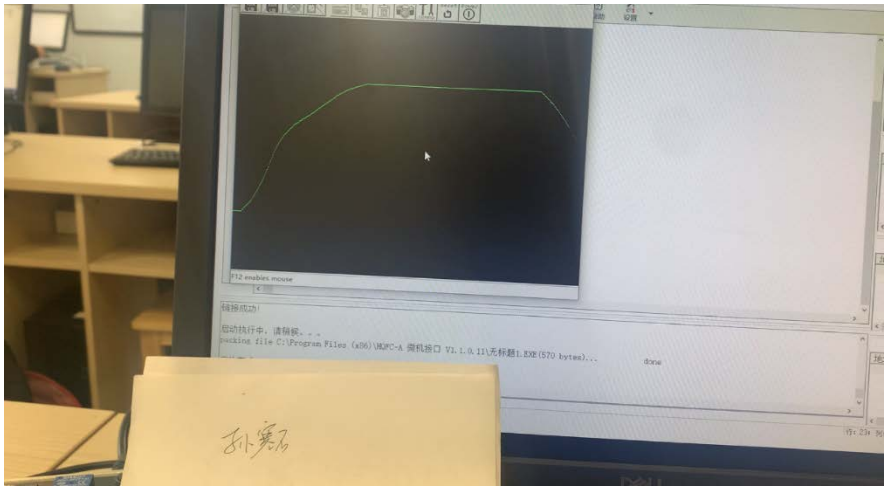
5.2.1 代码编写

```
io0809b     equ 299h
code segment
    assume    cs:code
start:  mov ax,0012h            ;设屏幕显示方式为 VGA 640X480 模示
    int 10h
    and cx,0                ;cx 为横坐标
draw:   mov dx,io0809b          ;启动 A/D 转换器通道 1
    out dx,al
    mov bx,500              ;延时
delay:  dec bx
    jnz delay
    in al,dx                ;读入数据
    mov ah,0
    mov dx,368              ;dx 为纵坐标
    sub dx,ax
    mov al,0ah              ;设置颜色
    mov ah,0ch              ;画点
```

```
    int 10h
    cmp cx,639              ;一行是否满
    jz start                ;是则转 start
    inc cx                  ;继续画点
    push dx
    mov ah,06h              ;是否有键按下
    mov dl,0ffh
    int 21h
    pop dx
    je draw                 ;无,则继续画点
    mov ax,0003             ;有恢复屏幕为字符方式
    int 10h
    mov ah,4ch              ;返回 DOS
    int 21h
code ends
end start
```

5.2.2 实验现象

IN1 为输入信号，下载程序至实验板后，将 IN1 连接，可以旋转旋钮，改变波形，如图：



6 实验小结

做完I2C实验之后，我又做了模数转换器的实验，学习并掌握了并行接口ADC0809的内部结构，功能及编程，进一步掌握了对模数转换芯片的使用。

参考文献

[1]姚静.基于 Proteus 的微机原理实验教学研究[J].信息通信,2020(10):99-101.

[2]杨帆. 片上 I2C 总线设计[D].贵州大学,2015.