

0 目录

第一章 简介.....	1
1.1 实验仪介绍.....	1
1.2 实验仪配置方案.....	1
1.3 功能特点.....	1
1.3.1 软件.....	1
1.3.2 硬件.....	2
第二章 硬件结构.....	4
2.1 电路外观.....	4
2.2 A1 区: 1602C、12864 液晶显示模块电路.....	5
2.3 A2 区: X5045	6
2.4 A3 区: CPU 总线、I/O 接口 片选区	6
2.5 A4 区: 控制区	6
2.6 B1 区: 语音模块 ISD1110 电路	7
2.7 B2 区: 逻辑笔、单脉冲、频率发生器	7
2.8 B3 区: 8259 电路.....	8
2.9 B4 区 8155、8255 电路	9
2.10 B5、D6 区: 扩展区	9
2.11 C1 区: 电源区	10
2.12 C2 区: 138 译码器.....	10
2.13 C3 区: 继电器	10
2.14 D1 区: 步进电机	10
2.15 D2 区: 光敏电阻、压力测量	11
2.16 D3 区: 8279 键盘/LED 控制器	12
2.17 D4 区: 8250	12
2.18 D5 区: 8253	13
2.19 E1 区: 直流电机转速测量/控制	13
2.20 E2 区: DAC0832 数模转换.....	14
2.21 E3 区: 蜂鸣器	14
2.22 E4 区: 温度测量/控制	14
2.23 E5 区: RS485	15
2.24 E6 区: RS232	15
2.25 F1 区: 红外通讯	15
2.26 F2 区: 0~5V 电压输出.....	15
2.27 F3 区: ADC0809 模数转换.....	16
2.28 F4 区: 键盘&LED	16
2.29 F5 区: 发光管、按键、开关	17
2.30 F6 区: 8*8LED 点阵.....	18

第三章 星研集成环境软件.....	19
3.1 软件安装.....	19
3.1.1 安装星研集成环境软件.....	19
3.1.2 软件卸载.....	19
3.1.3 USB 驱动程序.....	19
3.1.4 软件启动.....	20
3.1.5 编译器.....	21
3.1.6 README 文件.....	21
3.2 如何使用星研集成环境软件.....	22
3.2.1 数据传送程序 (ASM).....	22
3.2.2 数据传送程序 (C).....	35
3.3 实验连线、演示实验、测试实验仪.....	41
3.4 频率计 (EMU598+).....	43
3.5 模拟波形发生器 (EMU598+).....	44
3.6 TDS2、TDS2A (EMU598+) 虚拟示波器.....	44
第四章 软件实验.....	46
实验一 数据传送.....	46
实验二 双字节 BCD 码(十进制数)加法.....	48
实验三 双字节 BCD 码(十进制数)减法.....	51
实验四 四字节十六进制数转十进制数.....	54
实验五 散转.....	57
实验六 冒泡排序.....	59
实验七 二分查找法.....	61
第五章 基础硬件实验.....	63
实验一 8255 控制交通灯实验.....	64
实验二 74HC138 译码器实验.....	67
实验三 8155 输入、输出、SRAM 实验.....	69
实验四 8253 方波实验.....	71
实验五 8259A 中断控制器实验.....	73
实验六 8250 可编程通信实验(与微机).....	76
实验七 8279 键盘显示实验.....	80
实验八 并行 DA 实验.....	83
实验九 并行 AD 实验(数字电压表实验).....	85
实验十 红外通信实验.....	88
实验十一 字符型液晶显示实验(1602C).....	92
实验十二 图形点阵显示实验.....	97
实验十三 8237 DMA 传输实验.....	106
第六章 综合实验.....	109
实验一 简易电子琴实验.....	109
实验二 LED8 * 8 点阵实验.....	119
实验二 LED8 * 8 双色点阵实验(选配).....	124

实验三	数字式温度计实验(18B20).....	130
实验四	步进电机实验.....	136
实验五	直流电机测速实验.....	143
实验六	旋转图形实验.....	149
实验七	ISD1110 语音模块实验	153
实验九	电子钟 (CLOCK).....	163
实验十	光敏电阻测量光照强度实验.....	169

简介

1.1 实验仪介绍

STAR ES598PCIS 实验仪是 STAR ES598PCI 的简化版，提供详尽的 C、汇编例子程序、使用说明，可以满足各大专院校进行单片机、微机原理、32 位微机接口课程的开放式实验教学，通过扩展模块，可以让参加电子竞赛的学生熟悉各种类型的接口芯片，做各种实时控制实验，轻松面对电子竞赛；也可以让刚参加工作的电子工程师迅速成为高手。

实验仪主机就是一个实验平台，通过更换不同的 CPU 模块，可做不同类型 CPU 的实验，极大的提高了性价比。

STAR ES598PCIS 提供实验仪与微机同步演示功能，方便实验室老师的教学、演示。提供一个库文件，如果学生上机时间有限，只需编写最主要的程序，其它调用库文件即可。它布局合理，清晰明了；模块化设计，可以无限升级，让您的选择永不落伍；兼容性强，可以轻松升级，减少设备投资；使用方便，易于维护。

1.2 实验仪配置方案

微机原理二种配置方案：

- 1、实验仪主机、ES8688 模块（CPU：8088）、EMU598 仿真模块（不含虚拟示波器、信号发生器、频率计功能）。
- 2、实验仪主机、EMU598+仿真模块，内含 8086，含虚拟示波器、信号发生器、频率计功能。

1.3 功能特点

1.3.1 软件

1、提供我公司自主知识产权的星研集成环境软件，2004 年它已被认定为上海市高新技术成果转化项目

◇ 集编辑器、项目管理、启动编译、连接、错误定位、下载、调试于一体，多种实验仪、仿真器、多类型 CPU 仿真全部集成在一个环境下，操作方法完全一样。

◇ 完全 VC++ 风格。支持 C、PL/M、宏汇编：同时支持 Keil 公司 C51、Franklin 公司 C51、IAR/Archimedes 公司的 C51、Intel C96、Tasking 的 C196、Borland 公司的 TASM、Turbo C。

◇ 支持 ASM（汇编）、C、PLM 语言，多种语言多模块混合调试，文件长度无限制。

◇ 支持 BIN、HEX、OMF、AUBROF 等文件格式。可以直接转载 ABS、OMF 文件。

◇ 支持所有数据类型观察和修改。自动收集变量于变量窗（自动、局部、模块、全局）。

◇ 无须点击的感应式鼠标提示功能。

◇ 功能强大的项目管理功能，含有调试该项目有关的仿真器、所有相关文件、编译软件、编译连接控制项等所有的硬软件信息，下次打开该项目，无须设置，即可调试

- ◇ 支持 USB、并口、串口通信。
- ◇ 提供模拟调试器。
- ◇ 符合编程语言语法的彩色文本显示,所有窗口的字体、大小、颜色可以随意设置。

3、提供几十个实验的汇编、C 版本的源文件。提供一个库文件,如果学生上机时间有限,只需编写最主要的程序,其它调用库文件即可。

实验仪可提供以下软件实验:十进制数加法,十进制数减法,四字节二进制数转十进制数,数据传送,冒泡排序,二分查找法,散转等。



逻辑分析图

帧号	时间	反汇编	源文件	文件名, 行号
248	000175	290E D8FD DJNZ	R0, 290DH	
249	000176	290D F6 MOV	@R0, A	
250	000178	290E D8FD DJNZ	R0, 290DH	
251	000179	290D F6 MOV	@R0, A	
252	00017b	290E D8FD DJNZ	R0, 290DH	
253	00017c	290D F6 MOV	@R0, A	
254	00017e	290E D8FD DJNZ	R0, 290DH	
255	00017f	290D F6 MOV	@R0, A	
256	000181	290E D8FD DJNZ	R0, 290DH	
257	000183	2910 758121 MOV	SP, #21H	
258	000185	2913 02000E LJMP	000EH	
259	000187	000E 1228FD LCALL	28FDH	
260	000189	28FD 759852 MOV	SCON, #52H	c:\xingyan\examples\keil\whets\whets.c, 41
261	00018b	2900 758920 MOV	TMOD, #20H	c:\xingyan\examples\keil\whets\time.c, 28
262	00018d	2903 758869 MOV	TCON, #69H	c:\xingyan\examples\keil\whets\time.c, 30
263	00018f	2906 758DF3 MOV	TH1, #F3H	c:\xingyan\examples\keil\whets\time.c, 31
264	000191	2909 22 RET		c:\xingyan\examples\keil\whets\time.c, 32

实时跟踪图

1.3.2 硬件

1、传统实验

74HC244、74HC273 扩展简单的 I/O 口;蜂鸣器驱动电路;74HC138 译码;RS232 和 RS485 接口电路;8155、8255 扩展实验;8253 定时、分频实验;8250 串行通讯实验;16*2 液晶显示模块(可以选配多种 128*64 液晶点阵显示模块);8X8 LED 点阵显示模块;键盘 LED 控制器 8279,并配置了 8 位 LED、4 * 4 键盘;32K 数据 RAM 读写,使用 C51 编制较大实验成为可能;并行 AD 实验;并行 DA 实验;直流电机控制;步进电机控制;继电器控制实验;逻辑笔;打印机实验;电子琴实验;74HC4040 分频得到十多种频率;另外提供 8 个拨码盘、8 个发光二极管、8 个独立按键;单脉冲输出。

2、新颖实验

录音、放音模块实验;光敏电阻、压力传感器实验、V/F、F/V 实验(扩展模块)、非接触式 IC 卡读写实验(扩展模块)、NAND FLASH 实验(扩展模块);

串行接口实验

- 1) 一线 DALLAS 公司的 DS18B20 测温实验
- 2) 红外通信实验
- 3) CAN CAN2.0 (扩展模块)
- 4) USB USB1.1、USB2.0 (扩展模块)
- 5) 以太网 10M 以太网模块 (扩展模块)
- 6) GPRS (扩展模块)
- 7) GPS (扩展模块)
- 8) 蓝牙 (扩展模块)

3、闭环控制

- 1) 门禁系统实验
- 2) 光敏电阻、压力传感器实验
- 3) 旋转图形展现实验
- 4) 直流电机转速测量，使用光电开关测量电机转速
- 5) 直流电机转速测量，使用霍尔器件测量电机转速
- 6) 直流电机转速控制，使用霍尔器件、光电开关精确控制电机转速
- 7) 数字式温度控制，通过该实验可较好认识控制在实际中的应用

4、实验扩展区，提供扩展实验接口，用户可自行设计实验

可以提供 USB1.1、USB2.0、USB 主控、10M 以太网接口的 TCP/IP 实验模块、CAN 总线、非接触式 IC 卡、NAND FLASH 模块、FV_VF 模块、触摸屏模块、GPS、GPRS、双通道虚拟示波器、虚拟仪器、读写优盘、CPLD、FPGA、超声波测距、测速模块。其它模块正在陆续推出中，例如：蓝牙。

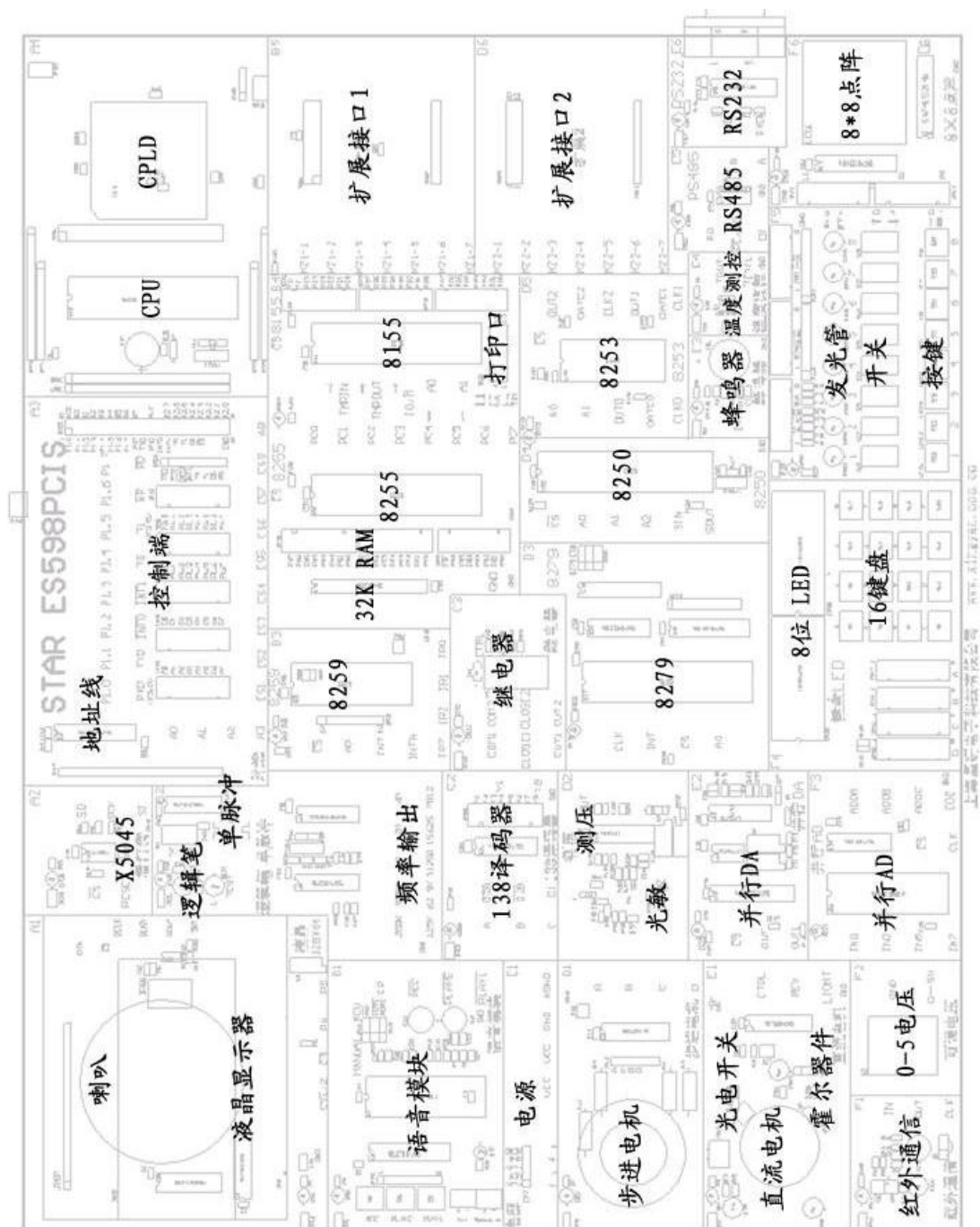
5、EDA —— CPLD、FPGA 可编程逻辑实验

逻辑门电路：与门、或门、非门、异或门、锁存器、触发器、缓冲器等；半加器、全加器、比较器、二、十进制计数器、分频器、移位寄存器、译码器；常用 74 系列芯片、接口芯片实验；8 段数码块显示实验；16x16 点阵式 LED 显示实验；键盘数码块实验（实时钟）、交通灯实验、串行通信收发；I2C 总线实验；SPI 总线（数字电压表）实验等

提供 ABEL、VHDL 语言编写的实验范例

2 硬件结构

2.1 电路外观

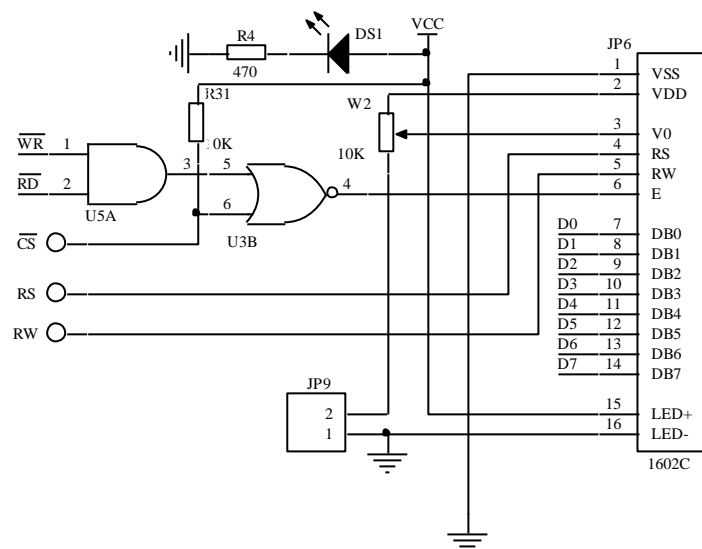


本章将逐一介绍实验仪的各个功能模块、相应的结构，读者在编写程序前，首先熟悉相应的硬件电路。

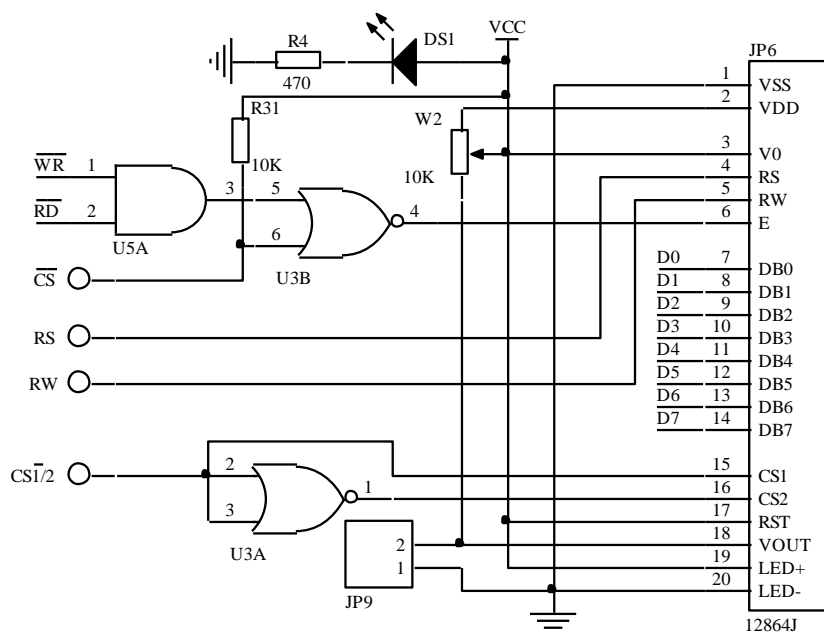
2. 2 A1 区：1602C、12864 液晶显示模块电路

1602C 字符型液晶：CS：片选信号，低电平有效；RS：选择读写的是指令或数据，L：指令，H：为数据。RW：读写控制端，L：写操作，H：读操作。

12864J 图形点阵液晶：CS：片选信号，低电平有效；CS1/2：左右半屏使能选择，H：左半屏，L：右半屏；RS：选择读写的是指令或数据，L：指令，H：为数据。RW：读写控制端，L：写操作，H：读操作。

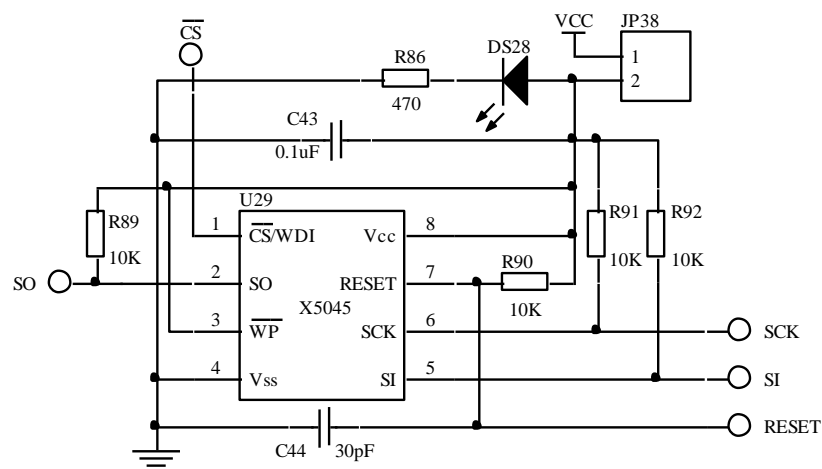


(1602C 字符型液晶)



(12864J 图形点阵液晶)

2. 3 A2 区: X5045



CS: 片选, 低电平有效
SCK: 时钟
SI: 数据输入
SO: 数据输出
RESET: 复位信号输出端, 高电平有效

2. 4 A3 区: CPU 总线、I/O 接口 片选区

JP45: 地址线 A0..A7;

JP48: 低位地址/数据总线

JP51: MCS51 的 P1 口;

JP59: 高位地址线 A8..A15;

JP61: MCS51 的 P3 口, P3.7、P3.6 作读、写信号线用;

JP66: 相当于一个 CPU 座, 使用 40 芯扁线与用户板相连, 可仿真 P0、P2 口作地址/数据使用的 CPU。

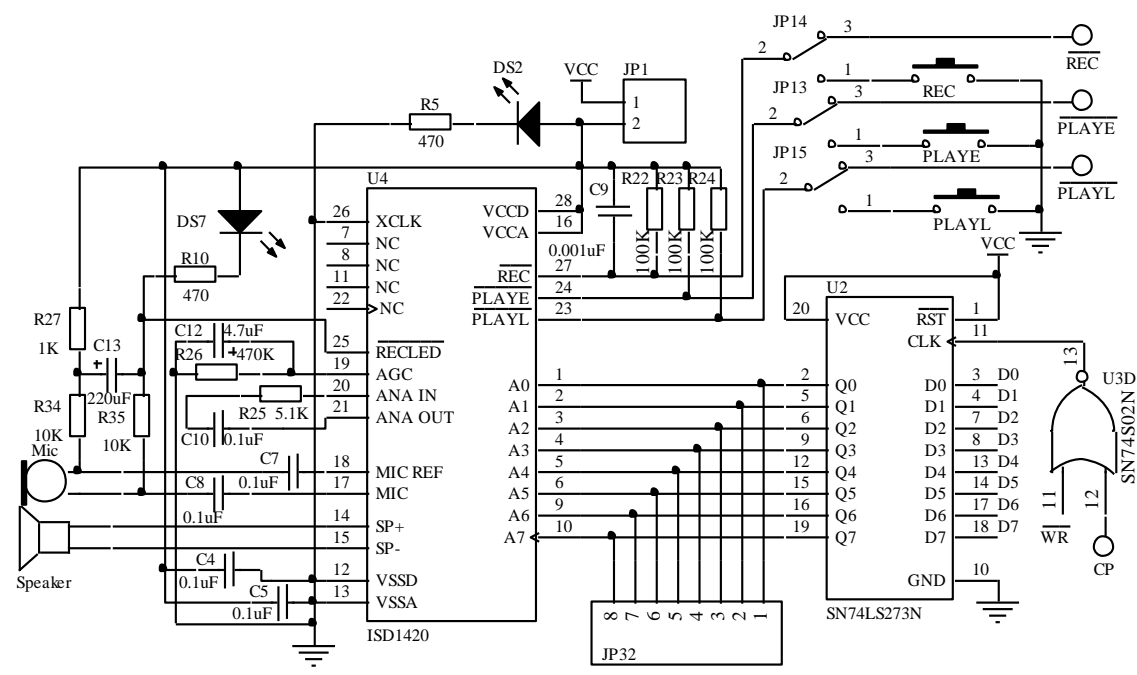
片选区

片选	地址范围	片选	地址范围
CS1	0F000H~0FFFFH	CS5	0B000H~0BFFFH
CS2	0E000H~0EFFFH	CS6	0A000H~0AFFFH
CS3	0D000H~0DFFFH	CS7	09000H~09FFFH
CS4	0C000H~0CFFFH	CS8	08000H~08FFFH

2. 5 A4 区: 控制区

主控部分。

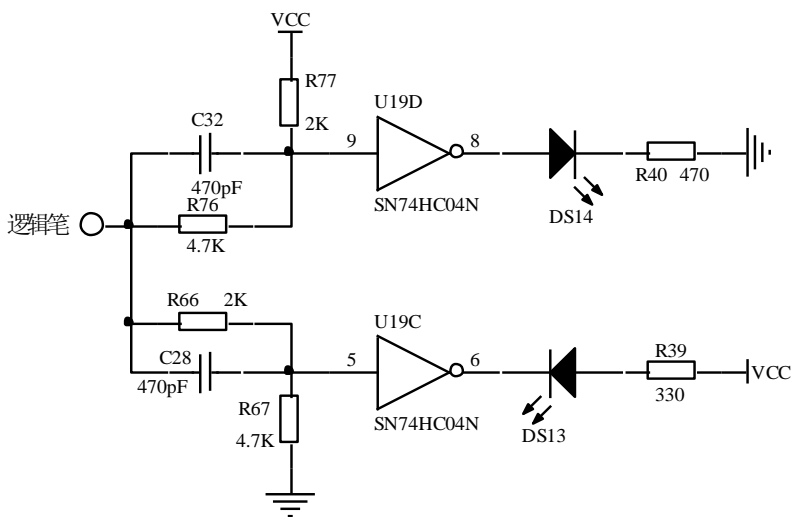
2. 6 B1 区：语音模块 ISD1110 电路



JP13、JP14、JP15：设置操作模式，MCU：CPU 控制方式；MANUAL：手动 (REC、PLAYL、PLAYE) 控制方式。

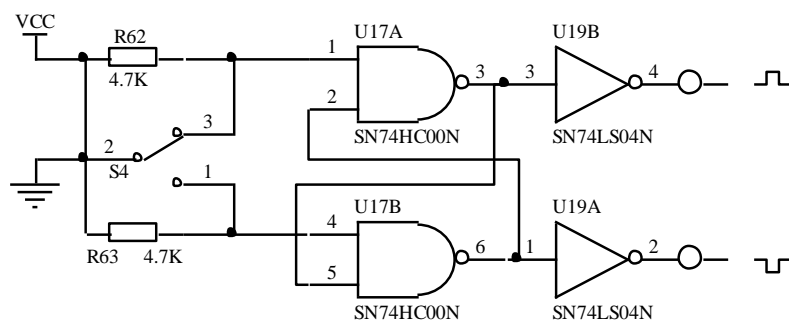
- REC：录音按键，低电平有效；
- PLAYE：电平放音按键，低电平有效，直到放音内容结束停止放音
- PLAYL：边沿放音按键，下降沿有效，并在下一个上升沿停止放音

2. 7 B2 区：逻辑笔、单脉冲、频率发生器



逻辑笔电路原理图

- 逻辑笔：测试接口，输入测量信号
- 绿灯 (DS13)：高电平点亮
- 红灯 (DS14)：低电平点亮
- 两灯同时亮：频率信号

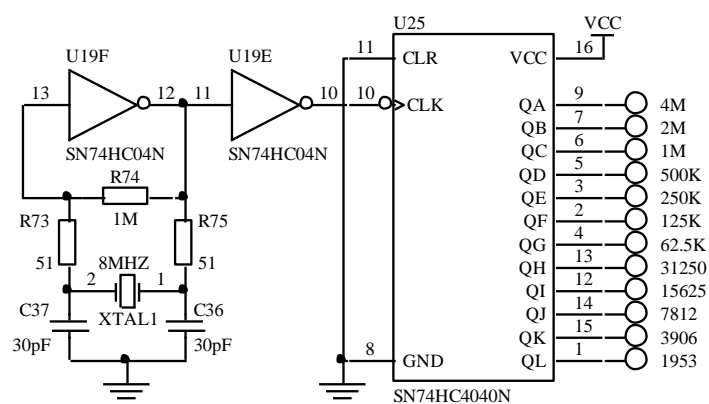


单脉冲电路原理图

S4: 脉冲发生开关

正脉冲: 上凸符号端口输出正脉冲

负脉冲: 下凹符号端口输出负脉冲

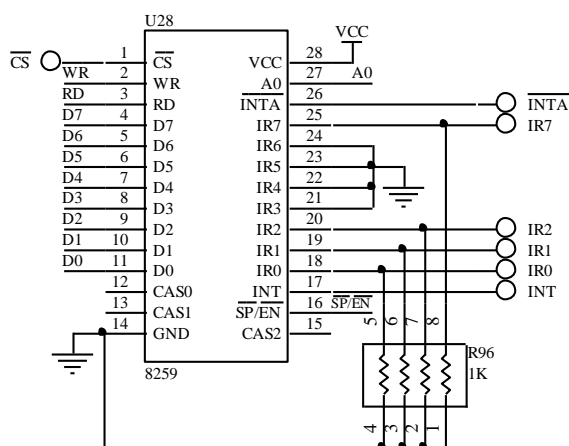


频率发生器电路原理图

4M: 输出 4MHZ 频率信号

其他端口输出的信号频率与端口下标识的数值一致

2. 8 B3 区: 8259 电路



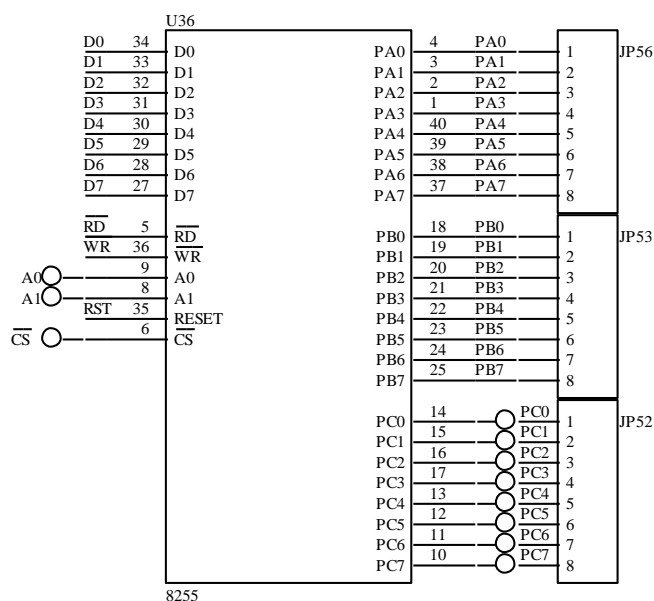
CS: 片选信号, 低电平有效;

A0: 地址信号

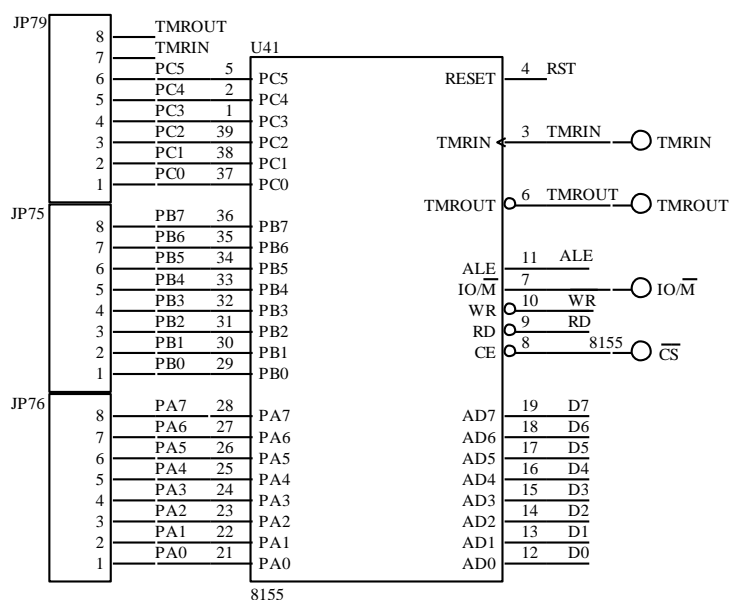
IR0..IN7: 中断输入

INTA: 中断响应

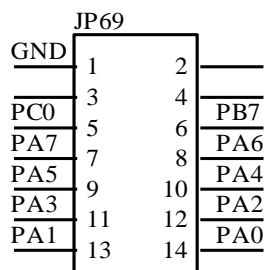
2. 9 B4 区 8155、8255 电路



CS: 片选信号，低电平有效；
A0、A1: 地址信号。
JP52: PC 口；
JP53: PB 口；
JP56: PA 口。



CS: 片选信号，低电平有效。
I0/M: 高电平，选择 I/O 口；
低电平，选择数据 RAM。
JP75: PB 口；
JP76: PA 口；
JP79: PC 口。



打印口

2. 10 B5、D6 区：扩展区

实验仪提供了二个扩展区，用来扩展 USB1.1、USB2.0、USB 主控、以太网、CAN 总线、V/F/V、非接触式 IC 卡、双通道虚拟示波器、CPLD、FPGA、GPS、GPRS、NAND FLASH 等扩展模块，其它

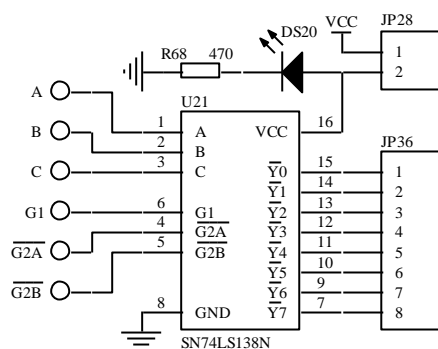
模块正在陆续推出中。

如果扩展模块较大，可以同时使用二个扩展区。

2. 11 C1 区：电源区

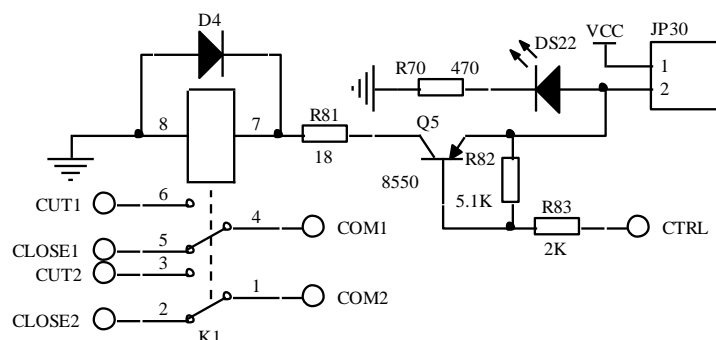
C1 区为用户提供了 5V(2A)、+12V(300mA)、-12V(300mA)等几种电源接口。

2. 12 C2 区：138 译码器



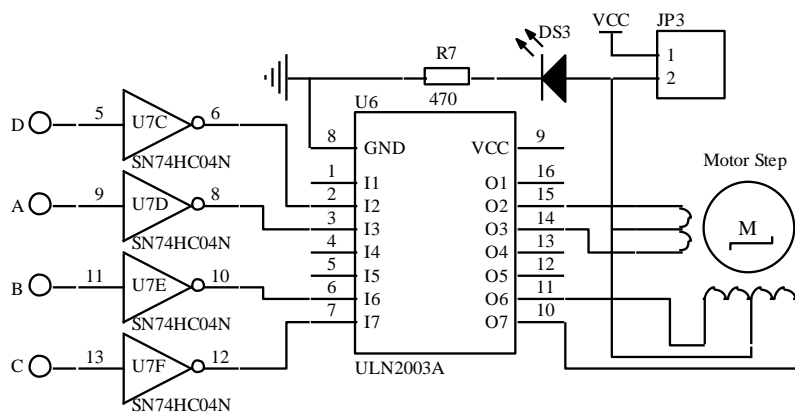
A、B、C： 3 位数据输入口
G1、 $\overline{G2A}$ 、 $\overline{G2B}$ ：译码控制口
Y0~Y7： 8 位译码数据输出口

2. 13 C3 区：继电器



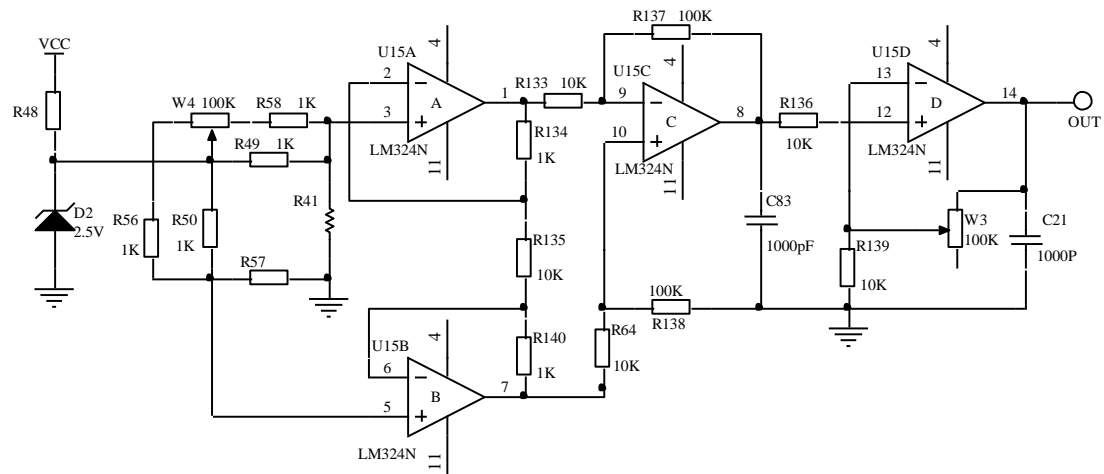
CTRL： 继电器开闭控制端
COM1、COM2：公共端 1、2
CLOSE1、CLOSE2：常闭端 1、2
CUT1、2： 常开端 1、2

2. 14 D1 区：步进电机



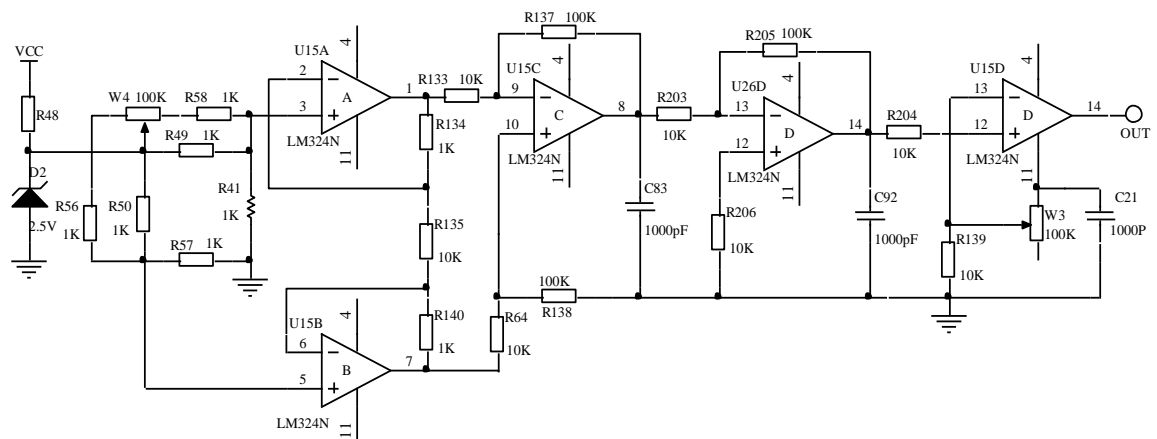
2. 15 D2 区：光敏电阻、压力测量

光敏电路



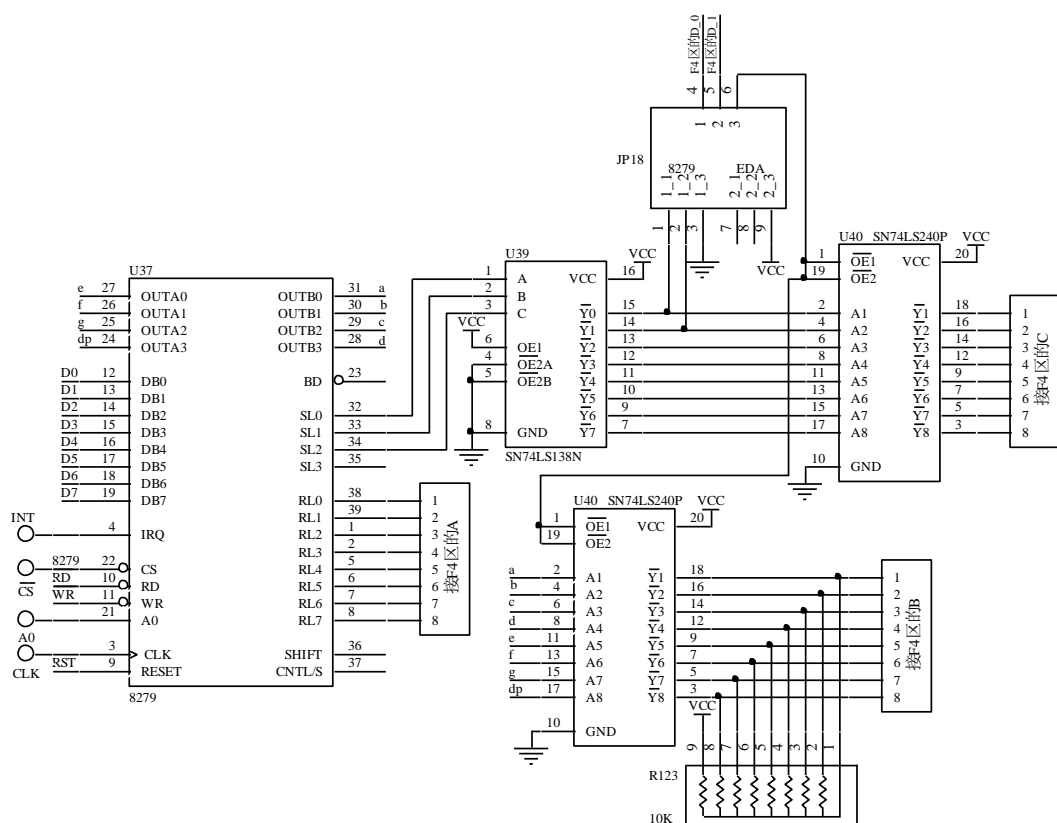
R41、R57 是光敏电阻；OUT：模拟电压信号输出端。

测压电路



压力测量： R41：电阻应变片，阻值 1K ；
OUT：压力模拟电压信号输出端

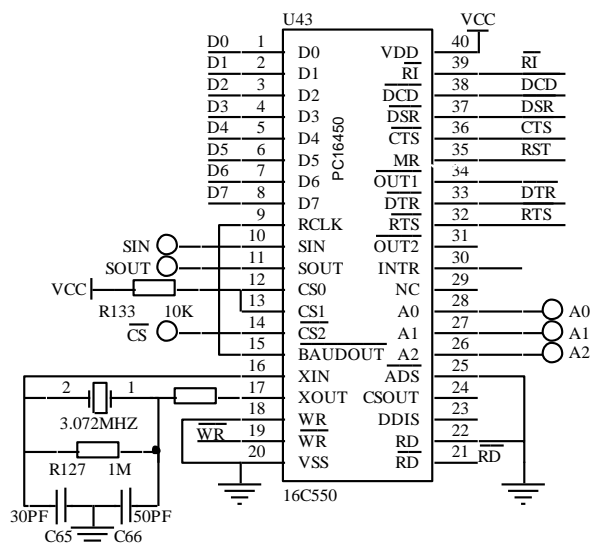
2. 16 D3 区：8279 键盘/LED 控制器



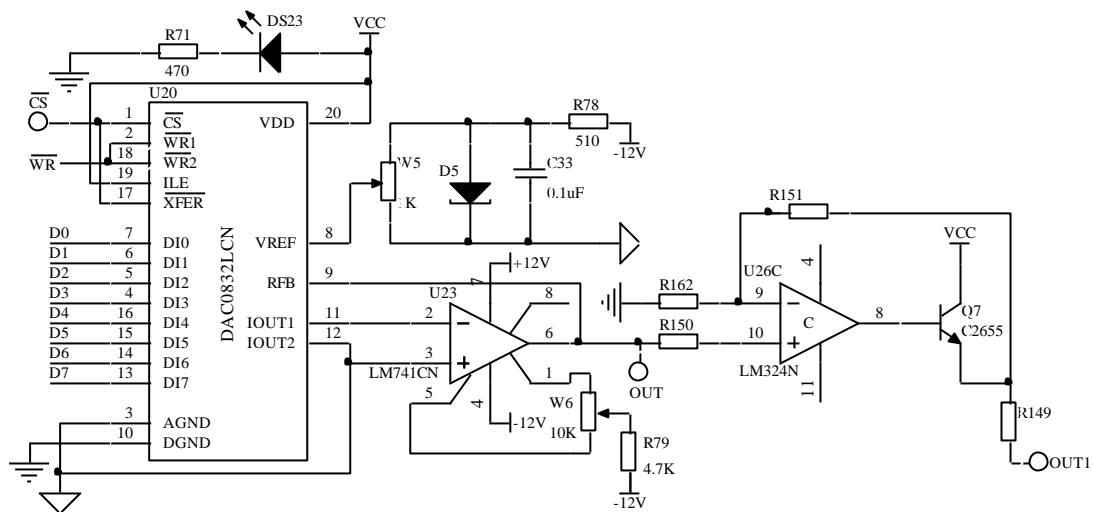
CS:	片选信号，低电平有效	A0:	地址信号
CLK:	时钟		
A:	已连接至按键的列线	B:	已连接至数码管段码
C:	已连接至数码管选择脚	JP18 的 1、2:	已连接至按键的行线

JP18 的短路块连接至 8279 端，F4 区的键盘、数码块由 82C79 扫描；短路块连接至 EDA 端，82C79 与 F4 区的键盘、数码块断开，可由其它芯片接管 F4 区，例如：使用 EDA 模块扫描键盘、数码块。

2. 17 D4 ☒: 8250

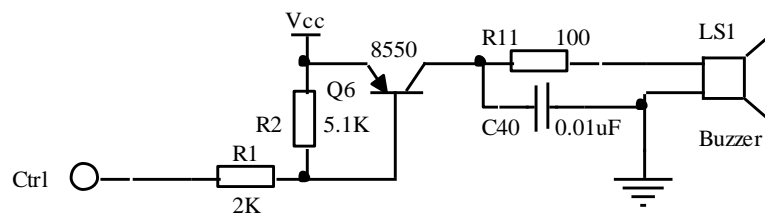


2. 20 E2 区：DAC0832 数模转换



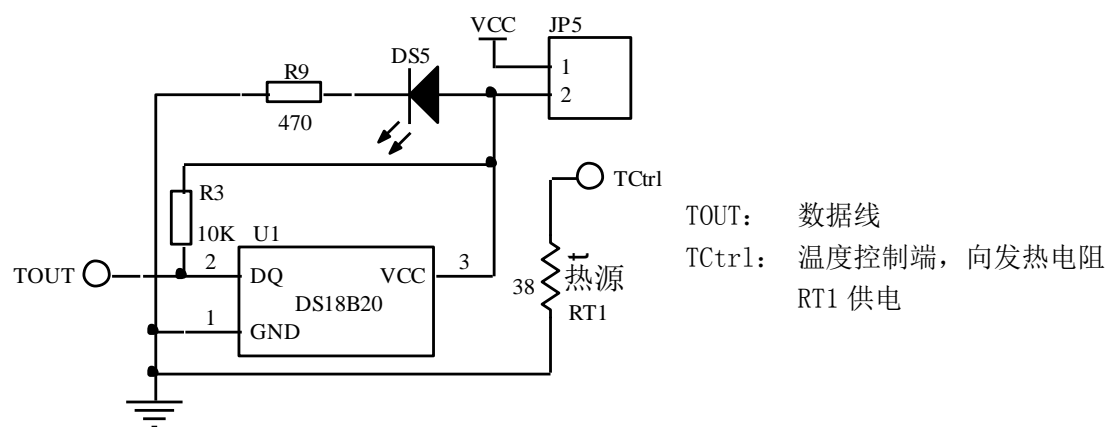
CS: 片选，低有效；OUT: 转换电压输出；OUT1: 经功放电路的电压输出；电位器 W5: 调整基准电压。

2. 21 E3 区：蜂鸣器



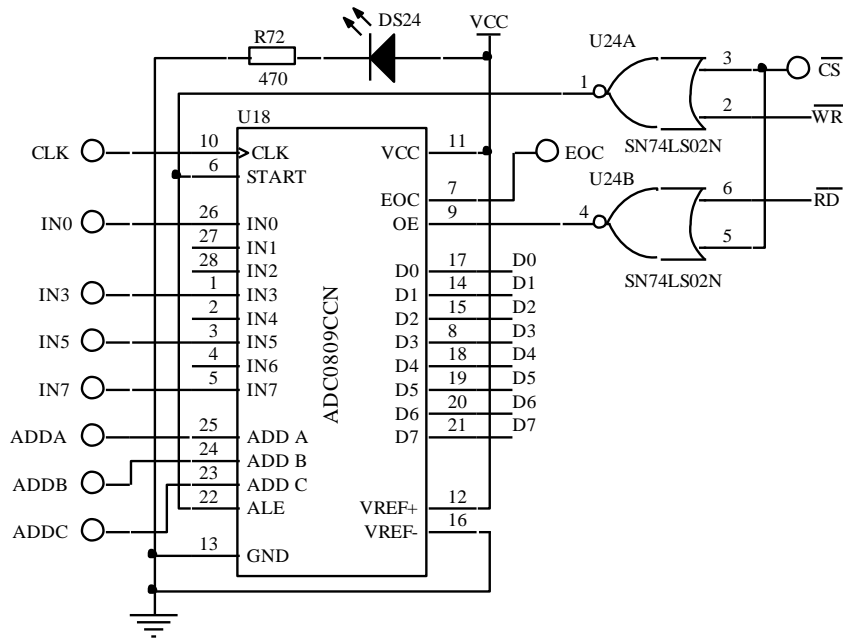
Ctrl: 控制接口，0—蜂鸣

2. 22 E4 区：温度测量/控制



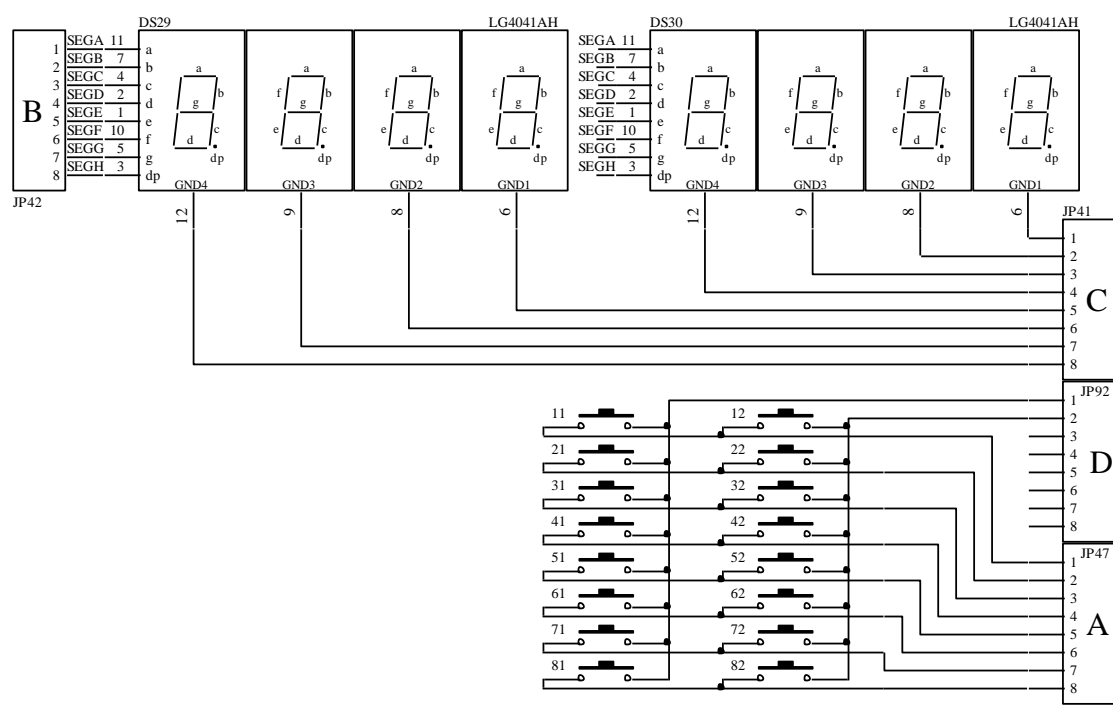
TOUT: 数据线
TCtrl: 温度控制端，向发热电阻 RT1 供电

2. 27 F3 区：ADC0809 模数转换



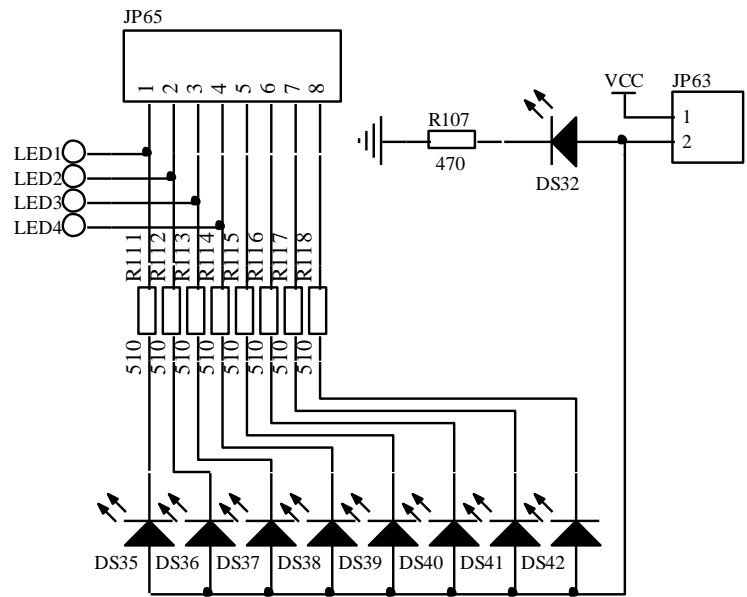
- CS: 片选，低有效；
CLK: 输入时钟 (10k—1280kHz)；
ADD A, ADD B, ADD C: 通道地址输入口；
EOC: 转换结束标志，高有效。
IN0、IN3、IN5、IN7: 模拟量输入

2. 28 F4 区：键盘&LED



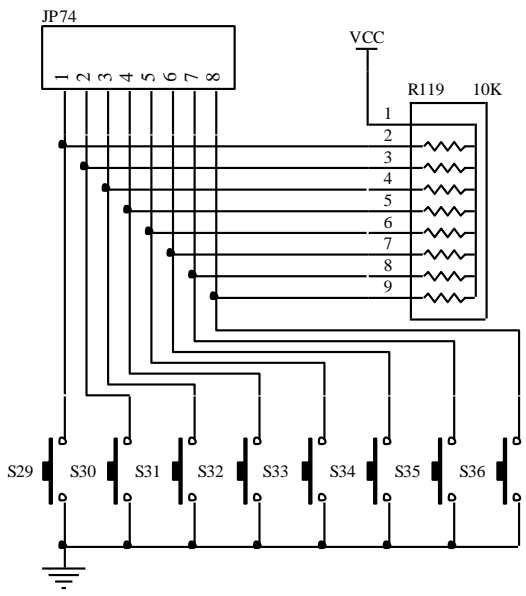
A:	按键的列线	B:	数码管段码
C:	数码管选择脚	D:	按键的行线

2. 29 F5 区：发光管、按键、开关



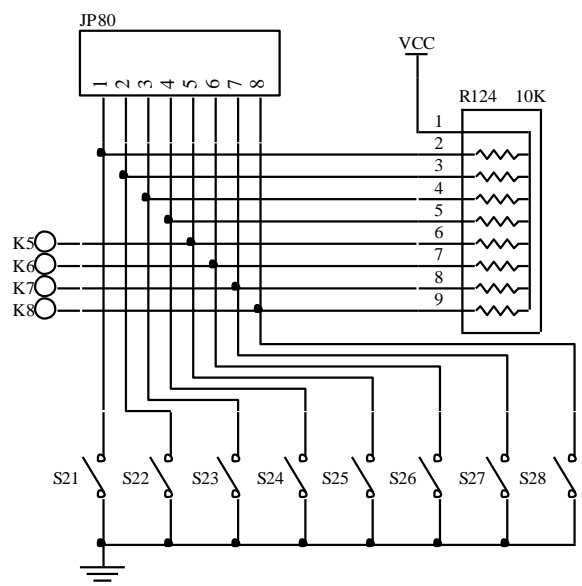
发光管电路原理图

JP65：发光管控制接口，0—灯亮，1—灯灭



按键电路原理图

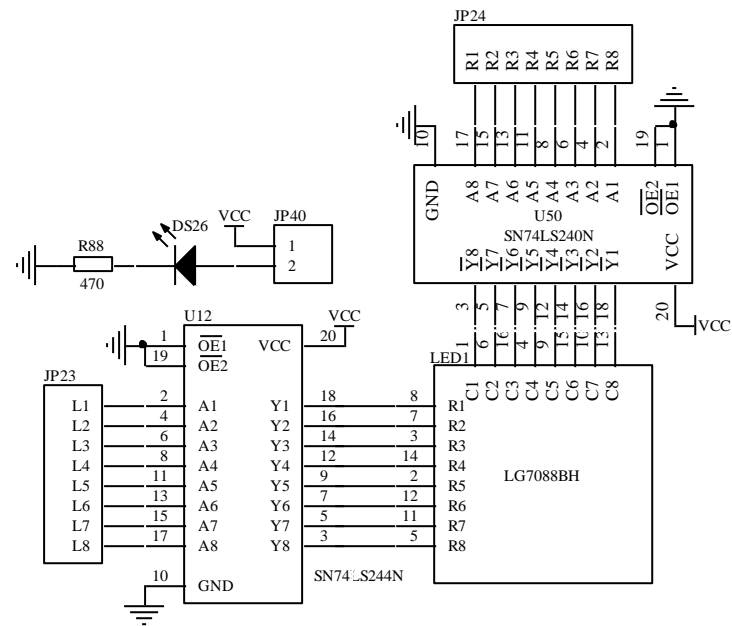
JP74：按键控制接口；按下—0 信号，松开—1 信号



开关电路原理图

JP80：开关控制接口；闭合—0 信号，断开—1 信号

2. 30 F6 区: 8*8LED 点阵



JP23: 8 根行扫描线; JP24: 8 根列扫描线。

星研集成环境软件

USB 接口的仿真器、实验仪客户：USB 设备是即插即用的设备，在第一次安装时，Windows 将调用“添加新设备向导”扫描所有可用的 INF 文件，试图找到合适的驱动程序。为了避免 USB 设备安装可能造成的麻烦，我们强烈的建议您先安装星研集成环境软件，安装程序将自动处理 USB 设备安装所需的 INF 文件和驱动程序。

3. 1 软件安装

3. 1. 1 安装星研集成环境软件

一. 新用户安装步骤

使用光盘安装：

1. 将仿真器、实验仪所配 CD 插入 CD-ROM 驱动器。
2. 在“我的电脑”或“资源管理器”中选择 CD-ROM 驱动器\WIN32\星研，然后运行 SETUP.EXE 文件即可进入安装界面。
3. 中文界面，用户只需按程序提示一步一步进行安装即可。

使用 Internet 下载文件的用户

1. 运行下载文件（XingYan.exe），软件自动执行安装程序。
2. 安装程序为中文显示，用户只需按程序提示一步一步进行安装即可。

二. 已安装过低版本星研集成环境软件的用户安装步骤：

1. 首先将原来的低版本软件进行卸载，具体步骤请参考“软件卸载”部分的内容。
2. 以后按新用户的安装步骤进行安装。

在安装过程中，如果用户没有指定安装目录，安装完成后会在 C: 盘建立一个 C:\XINGYAN 目录(文件夹)，结构如下：

XingYan	可执行文件、DLL 文件、寄存器文件
EXAMPLES	例子程序

3. 1. 2 软件卸载

1. 进入控制面板，运行“添加/删除程序”。
2. 进入“添加/删除程序”窗口，在“安装/卸载”页面上的列表中选择“星研集成环境软件”，按“删除”按钮，之后按自动卸载程序的说明一步一步地操作即可。

3. 1. 3 USB 驱动程序

1、USB 驱动程序的安装

通过 USB（通用串口总线）接口将微机与仿真器、实验仪相连，打开仿真器、实验仪电源。仿真器、实验仪与微机的第一次连接引起驱动程序的安装会变得很简单，您只需等待安装过程

的结束或按驱动程序的安装向导执行完即可。驱动程序的安装会出现如下界面：



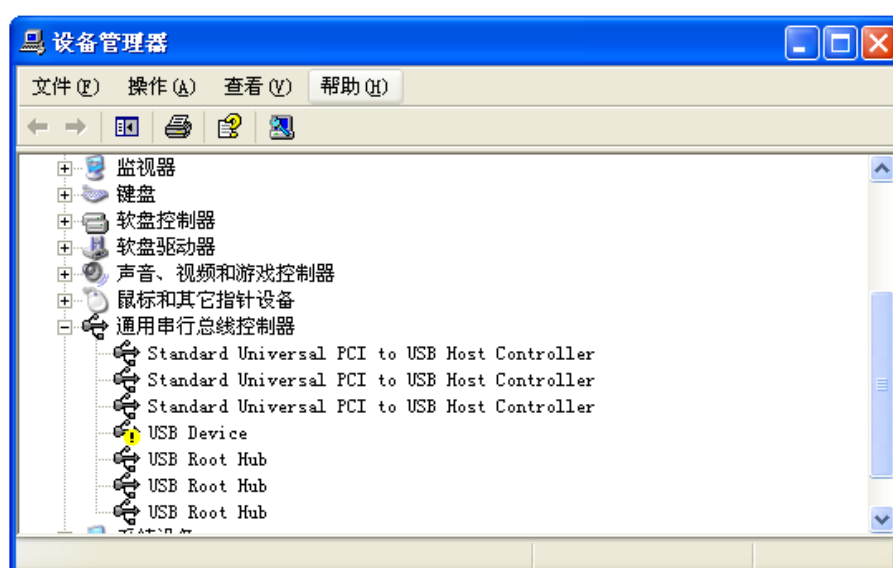
实际的界面可能有些差别，请等待该过程的结束。驱动程序的安装过程中，请勿执行其它应用程序。

2、 如何解决连接不上的情况

如果仿真器、实验仪与微机连接不上是由于未按正确步骤造成的，可根据以下步骤解决：

Window98/Window Me: 重新安装星研集成环境软件，关闭仿真器电源，稍等几秒钟，再打开电源，等待操作系统安装新的驱动程序结束后，运行星研软件即可。

Windows2000/WinXP: 在仿真器电源打开的情况下，使用控制面板中的“设备管理器”，可以看到一个未安装好的 USB 设备：



上图中的“通用串行总线控制器”下有一个打问号的 USB 设备，选中后按鼠标右键，选择菜单中的“卸载”项。重新安装星研集成环境软件，关闭仿真器、实验仪电源，稍等几秒钟，再打开电源，等待操作系统安装新的驱动程序结束后，运行星研软件即可。

注意：必须先安装星研集成环境软件；在 WinXP 中，驱动程序的安装会有选项，按缺省的值选择即可。

3. 1. 4 软件启动

运行 Windows，进入桌面窗口。

鼠标单击“开始”按钮，在“程序”栏中打开“星研集成环境软件”菜单栏，在其中选择“星研（SUPER、STAR 系列仿真器）”，开始启动星研集成环境软件。

注意：当您使用低配置机器时，从星研集成环境软件退出后必须等待足够的时间，让系统完全退出（硬盘停止工作）后，方可再次启动星研集成环境软件。

3. 1. 5 编译器

星研集成环境软件支持的编译器

MCS51	MCS96、MCS196	80X86
Keil A51、C51 Franklin A51、C51 Intel ASM51、PL/M51 Archimedes A8051、C-51	Intel ASM96、PLM96、C96 Tasking ASM196、C196	TC、TASM

编译器请用户自备。

设置工作环境

您的编译器正确安装后，请设置星研集成环境软件的编译器工作环境。

打开[主菜单 » 项目 » 设置工作环境]：



例如：您使用的编译器是 TASM、TC，安装在 C:\xingyan\TASM，C:\xingyan\TC，

TASM 宏汇编路径： C:\xingyan\TASM；

Turbo C 路径： C:\xingyan\TC；

3. 1. 6 README 文件

使用通用的文本编辑器，打开星研集成环境软件安装目录下的 README.DOC 文件，可获得此版本软件新增功能及最新的仿真器、实验仪安装、新增功能和使用信息，这些信息往往未及写入本手册。

3. 2 如何使用星研集成环境软件

下边几节，介绍如何使用星研集成环境软件：3.2.1 使用汇编语言，将数据段中 3000H~30FFH 单元的内容传送给实验仪 B4 区的 61C256 的 2000H~20FF 中；B4 区的 61C256 在 I/O 设备区，使用 $\overline{\text{IOR}}$ 、 $\overline{\text{IOW}}$ 读写；再将它传送回数据段的 6000H~60FFH 中。3.2.2 使用 Turbo C，重新编写第一个实验。

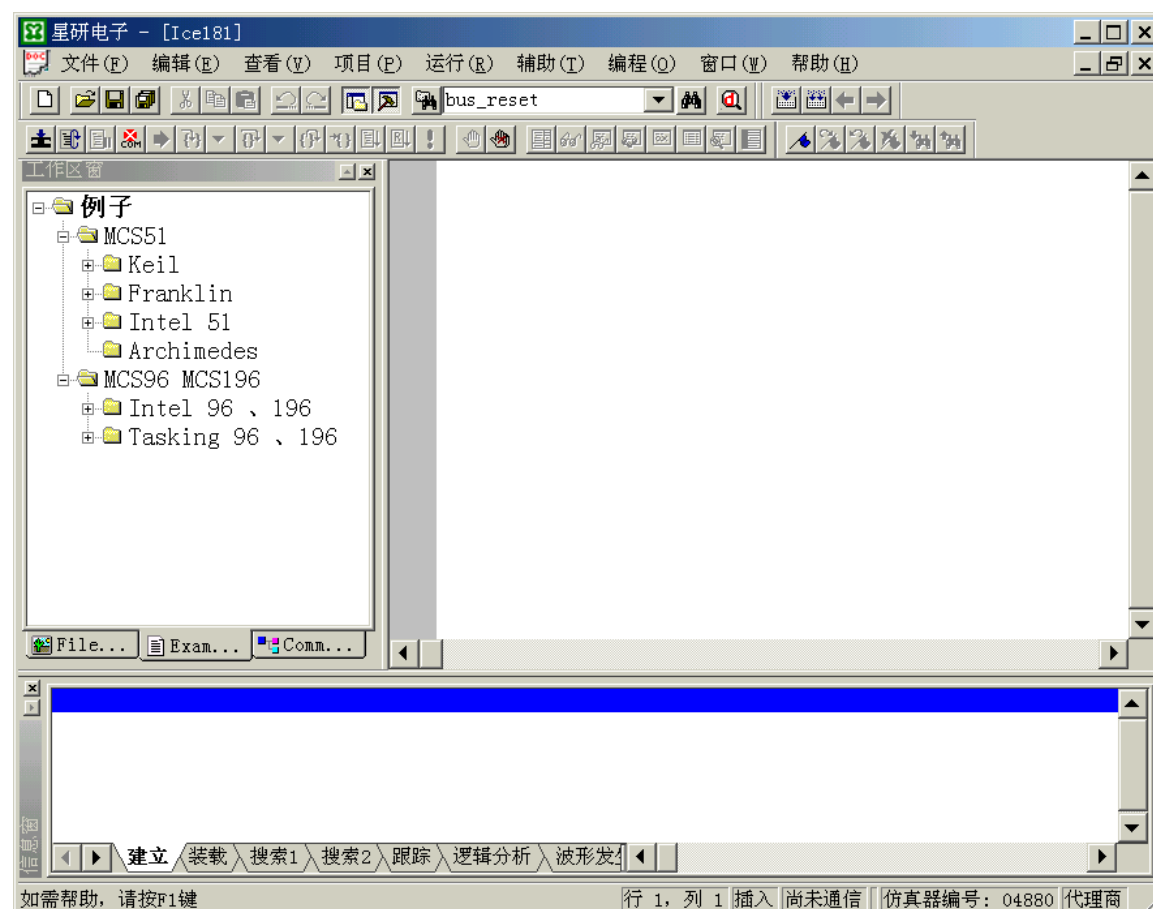
3. 2. 1 数据传送程序（ASM）

星研集成环境软件推荐您使用项目为单位来管理您的程序。如果您做一个简单的实验，或只希望看一个中间结果，您可以不建立项目文件，系统需要的各种设置，来源于“缺省项目”。本节不使用项目文件。

本例子旨在通过建立一个具体的程序来介绍星研集成软件的使用方法以及它的强大的调试功能。使用户很快的上手，体验到我们软件功能的强大和方便。

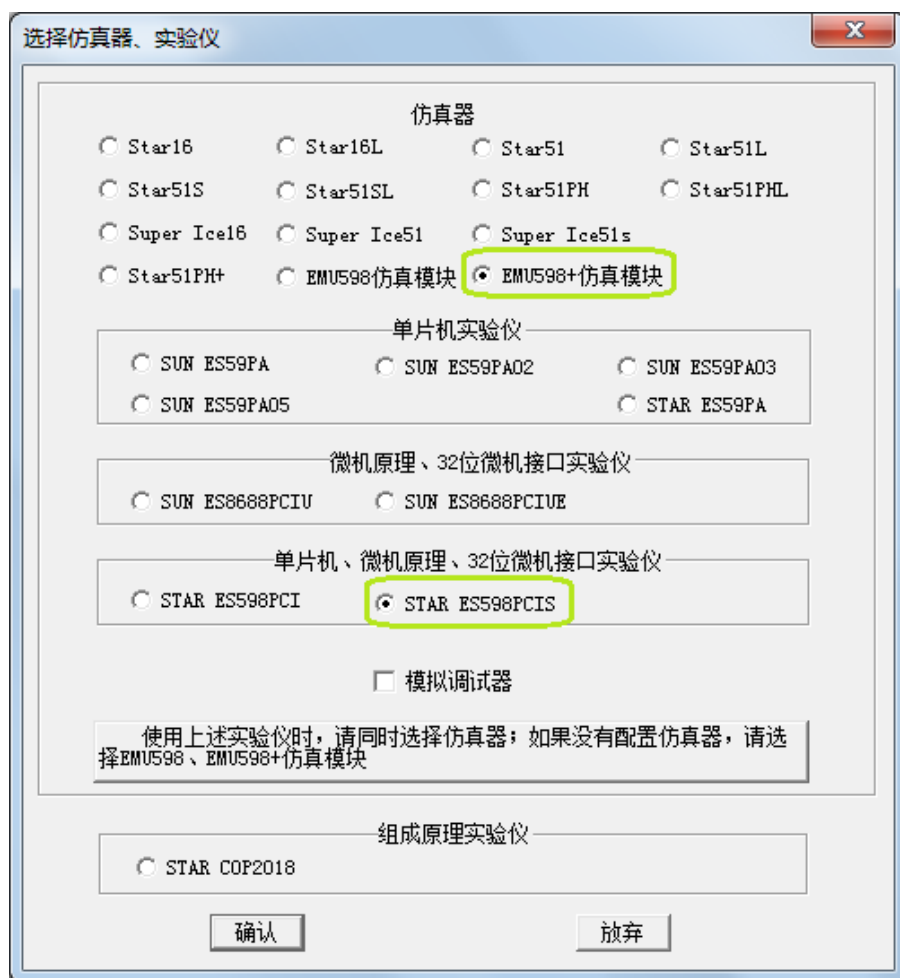
本实例是将数据段中 3000H~30FFH 单元的内容传送给实验仪 B4 区的 61C256 的 2000H~20FFH 中；B4 区的 61C256 在 I/O 设备区，使用 $\overline{\text{IOR}}$ 、 $\overline{\text{IOW}}$ 读写；再将它传送回数据段的 6000H~60FFH 中，程序是用汇编语言来编写。下面介绍相应的操作步骤：

首先运行星研集成软件。启动画面如图：



1、选择仿真器或仿真模块

执行 [主菜单 » 辅助 » 仿真器], 出现一个对话框:



请选择实验仪: STAR ES598PCIS; 仿真器: EMU598+仿真模块; 选择“确认”。

如果选择“模拟调试器”, 实验仪电源不用开启, 使用微机 CPU 模拟执行程序, 可以调用附件中的软中断, 但无法对 I/O 接口操作。

学生在做实验前, 可以选择“模拟调试器”, 在星研集成环境中编写程序, 对它编译、连接, 解决语法错误, 使用模拟调试功能, 初步调试; 可以大幅度减少在实验室中做实验的时间。

2、设置缺省项目

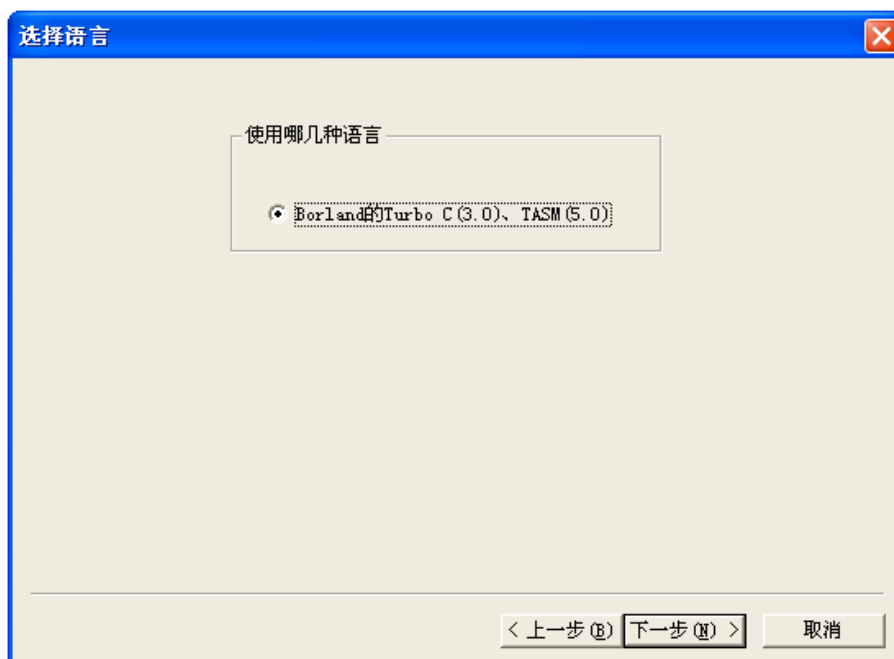
执行 [主菜单 » 辅助 » 缺省项目], 出现一个对话框:



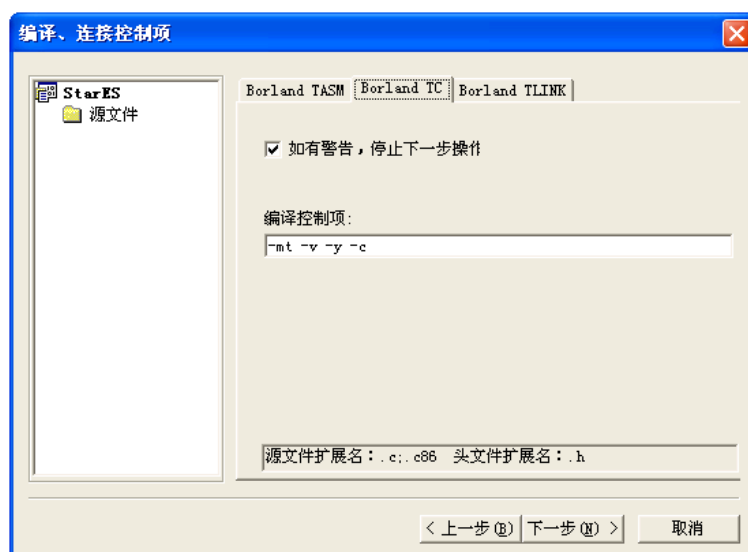
(使用 EMU598 仿真模块, 选择“STAR ES8688 仿真模块”)

(使用 EMU598+仿真模块, 选择“8086 (EMU598+)”)

点击进入下一步：“选择语言”

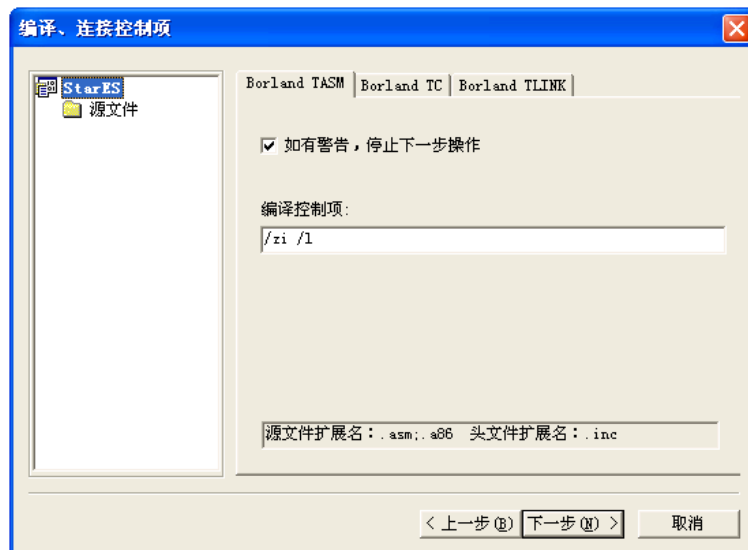


您可以根据自己的需要以及程序的类型作相应的选择，本实例选择 Borland 公司的 Turbo C (3.0)、TASM (5.0) (请确定在选择语言之前已经安装好相应的编译软件)。然后再点击进入下一步：“编译、连接控制项”

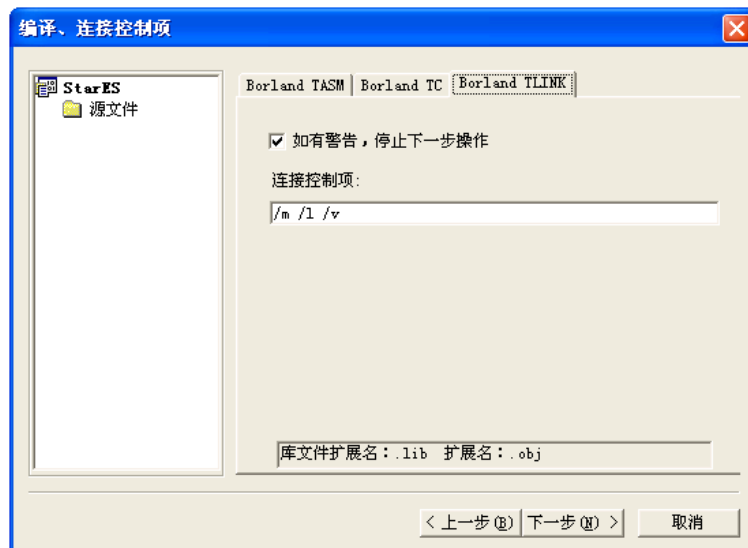


实验仪提供 64K memory 空间，memory model 请选择 tiny，缩写为 mt;如果需要源程序级别调试，必须使用 -v -y 控制项，为了支持多文件编译、连接，必须使用 -c 控制项。

一般不必改变 Turbo C 的编译控制项。

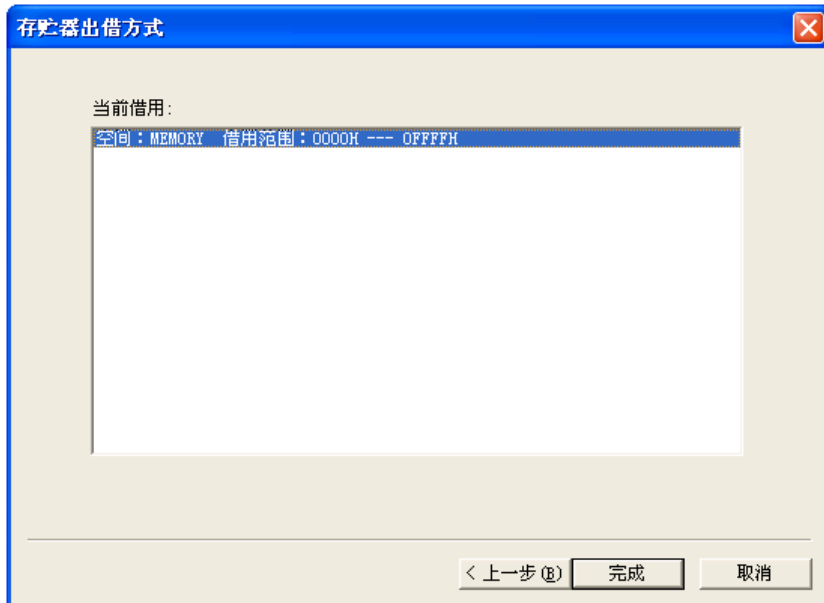


如果需要源程序级别调试，必须使用 /zi /l 控制项。
一般不必改变 Tasm 的编译控制项。




如果需要源程序级别调试，必须使用 /m /l /v 控制项。
一般不必改变 TLINK 的连接控制项。

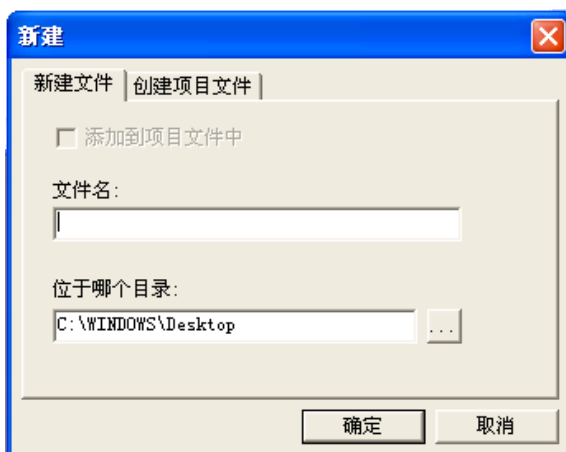
然后再点击进入下一步：“存储器出借方式”



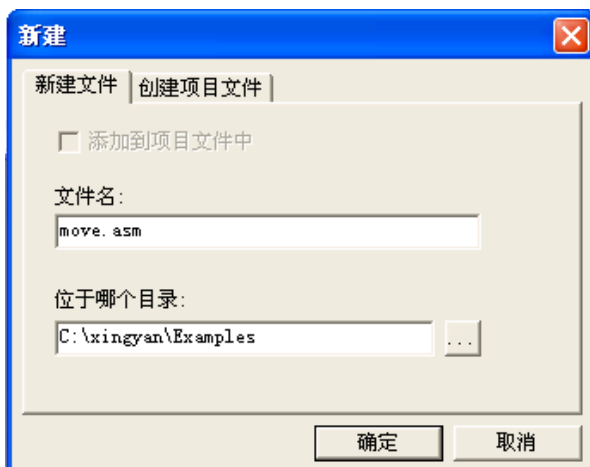
仿真模块 EMU598 提供 64K 仿真 RAM，作程序段（CS）、数据段（DS）、附加段（ES）、堆栈段（SS）使用。

2、建立源文件

下面我们建立源文件，执行 [主菜单 » 文件 » 新建]，（或者点击图标）打开窗口如下：



首先选择存放源文件的目录，输入文件名，注意：一定要输入文件名后缀。对源文件编译、连接、生成代码文件时，系统会根据不同的扩展名启动相应的编译软件。比如：*.asm 文件，使用 TASM 来对它编译。本实例文件名为 move.asm。窗口如下：



按“确定”即可。然后出现文件编辑窗口：

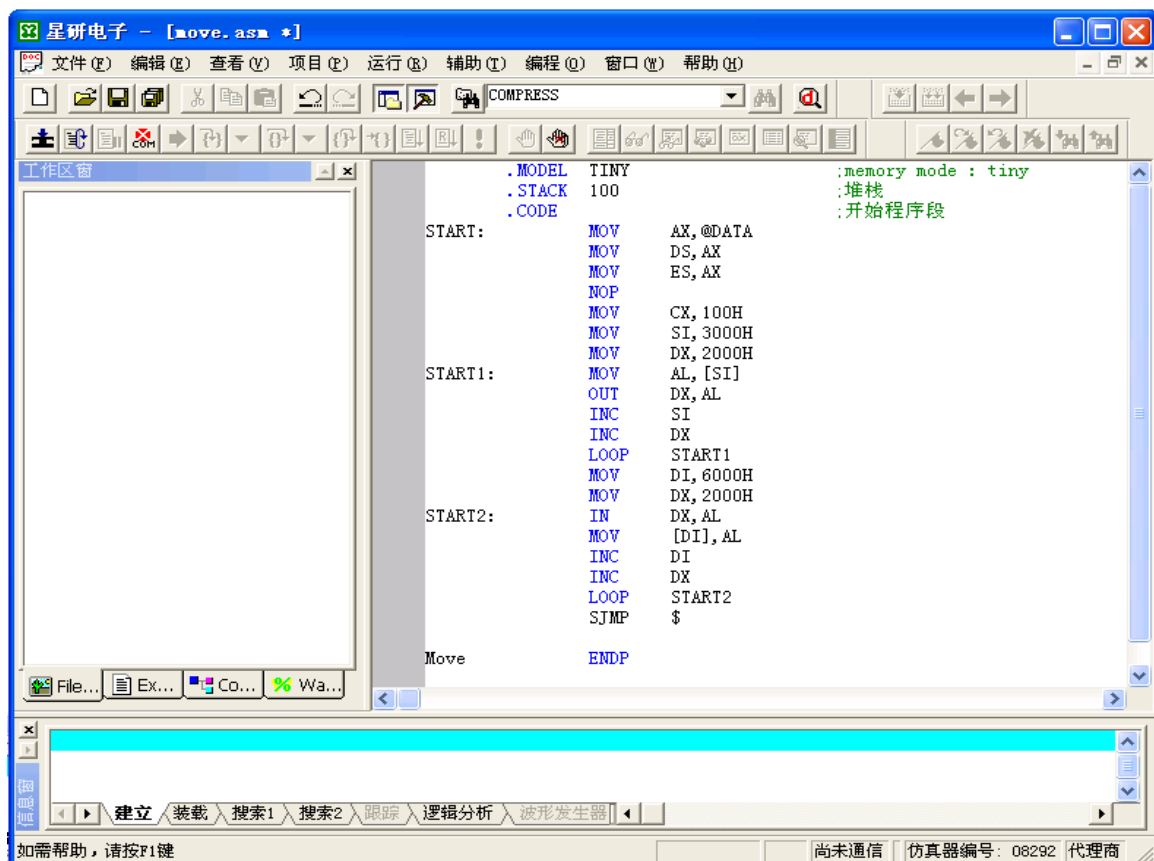


输入源程序，本实例的源程序如下：

```
.MODEL          TINY          ;memory mode : tiny
.STACK          100           ;堆栈
.CODE           ;开始程序段
START:  MOV      AX, @DATA
        MOV      DS, AX
        MOV      ES, AX
        NOP
        MOV      CX, 100H
        MOV      SI, 3000H
        MOV      DX, 2000H
START1:  MOV      AL, [SI]
        OUT      DX, AL
        INC      SI
        INC      DX
        LOOP     START1
        MOV      DI, 6000H
        MOV      DX, 2000H
        MOV      CX, 100H
START2:  IN       DX, AL
        MOV      [DI], AL
        INC      DI
        INC      DX
        LOOP     START2
        SJMP     $



Move     ENDP
        END      START
```

输入源程序，如下图：

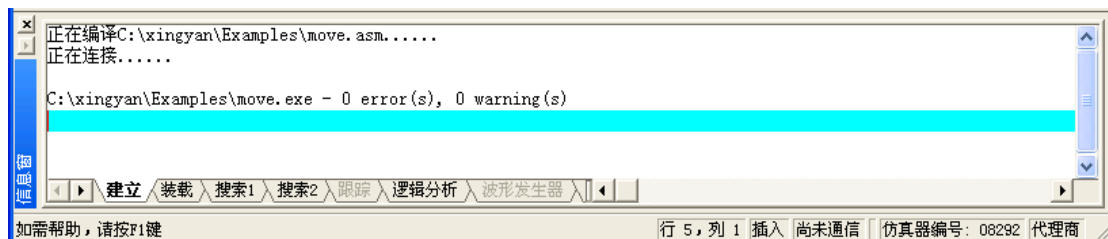


这样一个源文件就建立好了。

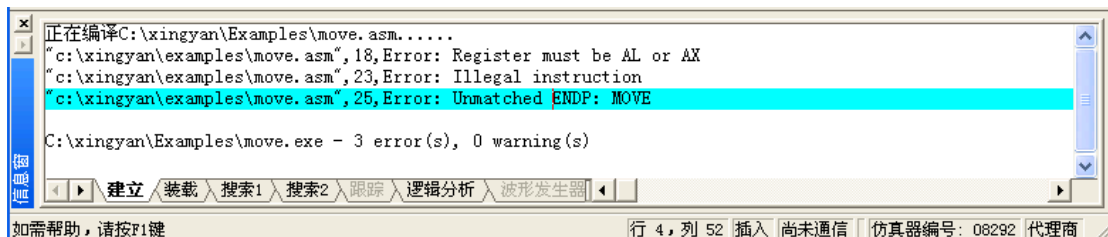
3. 编译、连接文件

首先选择一个源文件，然后可以编译、连接文件了。对文件编译，如果没有错误，再与库文件连接，生成代码文件（DOB、EXE 文件）。编译、连接文件的方法有如下二种：（1）使用[主菜单 » 项目 » 编译、连接]或[主菜单 » 项目 » 重新编译、连接]。（2）点击图标或来“编译、连接”或“重新编译连接”。

“编译连接”与“重新编译、连接”区别：“重新编译、连接”不管源文件是否修改、编译软件是否变化、编译控制项有无修改，对源文件编译，如果没有错误，再与库文件连接，生成代码文件（DOB、EXE 文件）。编译、连接过程中产生的信息显示在信息窗的“建立”视中。编译没有错误的信息如下：



若有错误则出现如下信息框：



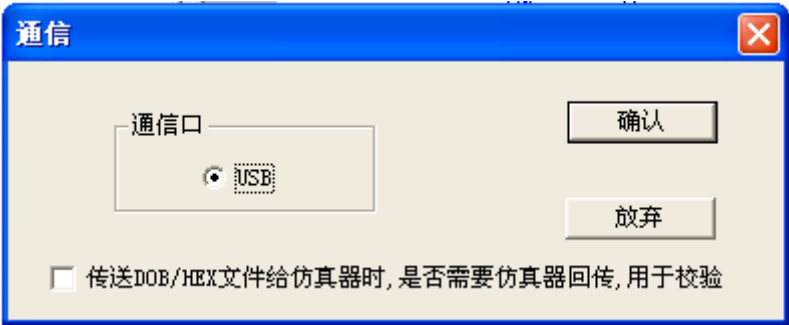
有错误、警告信息，用鼠标左键双击错误、警告信息或将光标移到错误、警告信息上，回车，系统自动打开对应的出错文件，并定位于出错行上。



这时用户可以作相应的修改，直到编译、连接文件通过。


5. 调试

在进入调试状态以前，请正确设置通信口：执行[主菜单 » 辅助 » 通信]，对话框如下：

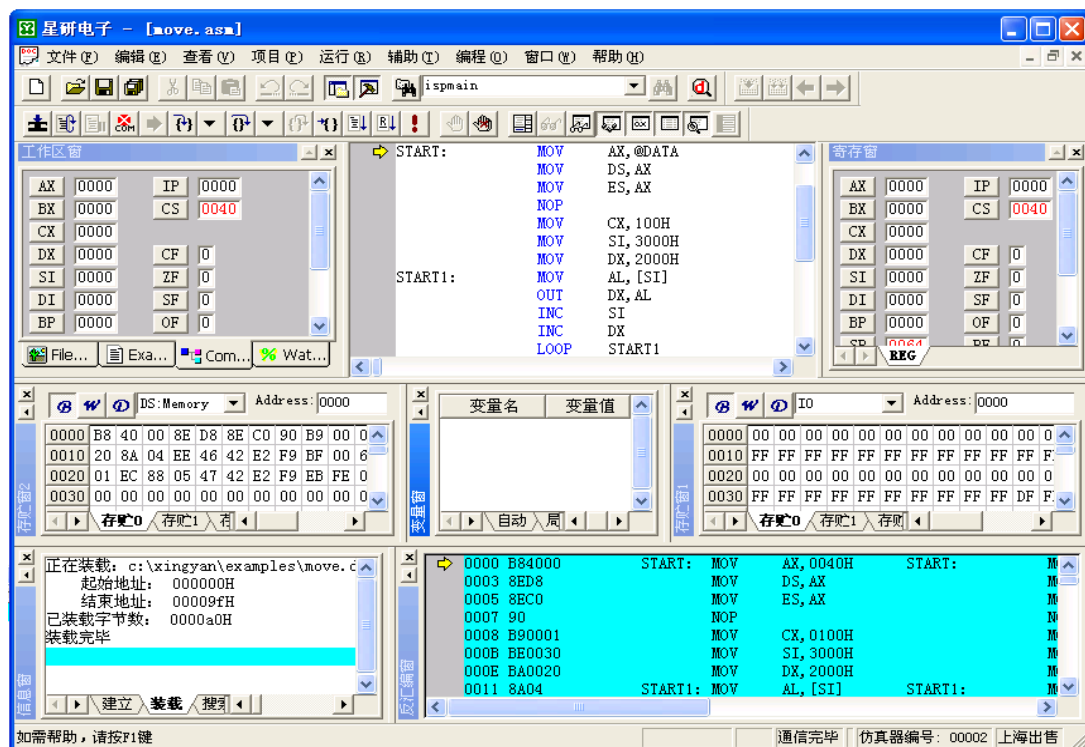


仿真器、实验仪配套的通信线可以与微机 USB 口相连，即为 USB 通信线，请选择 USB。
对于最下面一行的**校验**，通常您不必选中它，可以提高传送 DOB、HEX、BIN 文件时的速度。
在进入调试状态以前，你还必须确定实验仪与微机的正确连接，电源接通，开关打开。

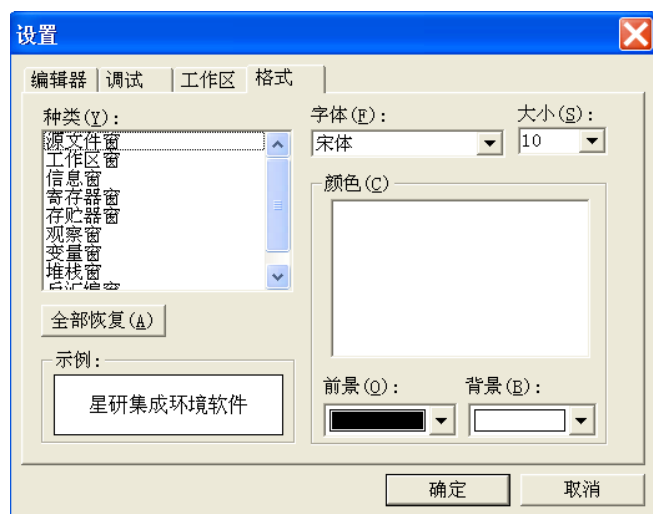
如果编译、连接正确后，可以开始调试程序。进入调试状态方法有：

- a) 执行[主菜单 » 运行 » 进入调试状态]
- b) 点击工具条的
- c) 执行[主菜单 » 运行 » 装载 DOB、HEX、BIN 文件]


进入后的窗口如下：



在整个图片中我们可以看到相对应的窗口信息。在“工作区窗”的“CommonRegister”中我们可以了解通用寄存器的信息。中间的窗口为源程序窗口，用户可在此设置断点，设置光标的运行处，编辑程序等。寄存器窗我们可以看到一些常用的寄存器的数值。存贮窗 1、存贮窗 2 显示相应的程序段（CS）、数据段（DS）、IO 设备区的数据，还有变量窗，自动收集变量显示其中。反汇编窗显示对程序反汇编的信息代码、机器码、对应的源文件。在信息窗的“装载”视图中，显示装载的代码文件，装载的字节数，装载完毕后，显示起始地址，结束地址。这种船坞化的窗口比通常的窗口显示的内容更多，移动非常方便。用鼠标左键点住窗口左边或上方的标题条，移动鼠标，将窗口移到您认为合适的位置；将鼠标移到窗口的边上，鼠标的图标变成可变化窗口时的形状，用鼠标左键点住，移动鼠标，变化一个或一组窗口的大小。在调试过程中，可以根据您的需要，在[主菜单 » 查看]中打开：寄存器窗、存贮器窗 1、2、3、观察窗、变量窗、反汇编窗。您也可以通过[主菜单 » 辅助 » 设置 » 格式]，设置每一种窗口使用的字体、大小、颜色。移动窗口到您喜欢的位置、大小。



首先在“种类”中选择一个窗口，然后选择“字体”、“大小”，在“颜色”中选择某一类，在“前景”、“背景”中选择您喜欢的颜色。

对于高级语言，在您的程序前有一段库文件提供的初始化代码，（当前可执行标志）不会出现在您的文件行上，如果您使用 C 语言，可将光标移到 main 函数上，按 F4 功能键，让 CPU 全速运行到 main 行上后停下；如果您使用 PL/M 语言，按 F7 功能键，让 CPU “单步进入”，运行到您的任何一个可执行行后停下。

您可以使用以下命令调试您的程序：



设置或清除断点（功能键为 F2）

在当前光标行上设置或清除一个断点




单步进入（功能键 F7）

单步执行当前行或当前指令，可进入函数或子程序。



连续单步进入（功能键 Ctrl + F7）

连续执行“单步进入”，用鼠标点击或按任意键后，停止运行。




单步（功能键 F8）

单步执行当前行或当前指令，**将函数或子程序作为一条指令来执行**。如果当前行中含有函数、子程序或发生中断，CPU 将执行完整个函数、子程序或中断，停止于当前行或当前指令的下一有代码的行上。



连续单步（功能键 Ctrl + F8）

连续执行“单步”，用鼠标点击或按任意键后，停止运行。



运行到光标行（功能键 F4）

从当前地址开始全速运行用户程序，碰到光标行、断点或用鼠标点击，停止运行。




全速断点（功能键 F9）

从当前地址开始全速运行用户程序，碰到断点或用鼠标点击，停止运行。



全速运行（功能键 Ctrl + F10）

从当前地址开始全速运行用户程序，此时，按用户系统的复位键，CPU 从头开始执行用户程序，用鼠标点击，停止运行。全速运行时，屏蔽了所有断点，即不会响应任何断点。



停止运行



终止微机与仿真器之间通信（功能键 ESC）。

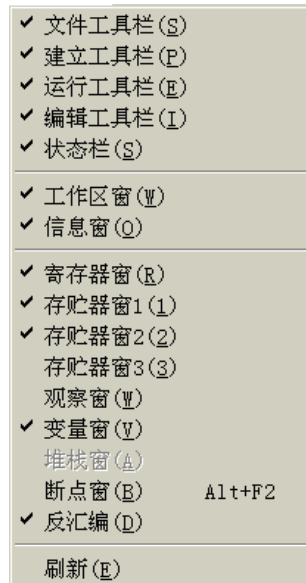
注意：欲终止微机与仿真器之间通信，功能键 **ESC** 是一个很方便的键，它的效果比点击相应的图标的效果要好。建议用户多用 **ESC** 键。在系统运行“连续单步”或者“连续单步进入”时 ESC 键被禁止，这时用户可以按键盘的其他任意键停止其运行。

5. 调试的方法及技巧

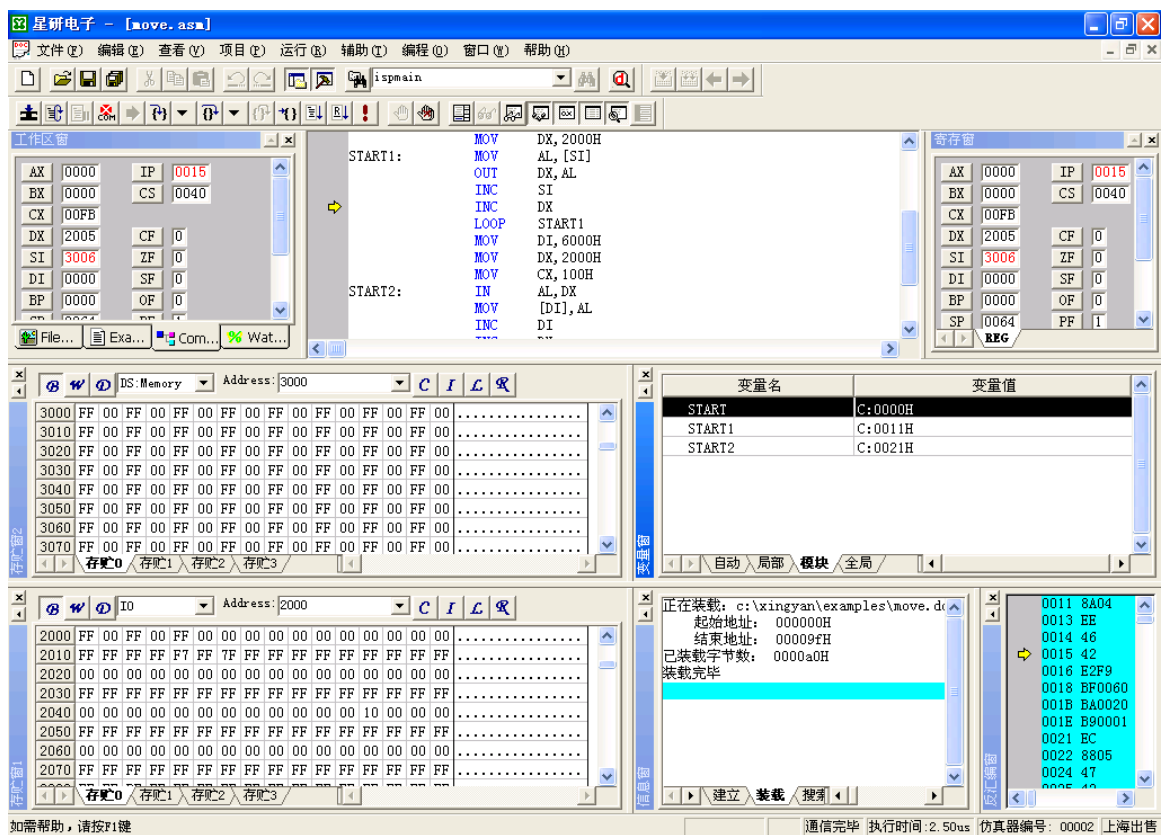
一般来说，用户的程序或多或少的会有一些逻辑错误，我们的仿真器、实验仪和星研集成软件可以帮助用户很快的定位，很快的查出相应的错误。

在调试状态的窗口中我们可以看到很多的窗口，用户只要熟练地应用这些窗口来观察、分析数据就会很快的调试好程序，达到事半功倍的效果。

进入调试界面后，由于我们本次操作需要观察三个数据块：数据段 3000H~30FFH，数据段 6000H~60FFH，I/O 区 2000H~20FFH，可以打开二到三个存储器窗口，具体操作是：[主菜单]>>查看]

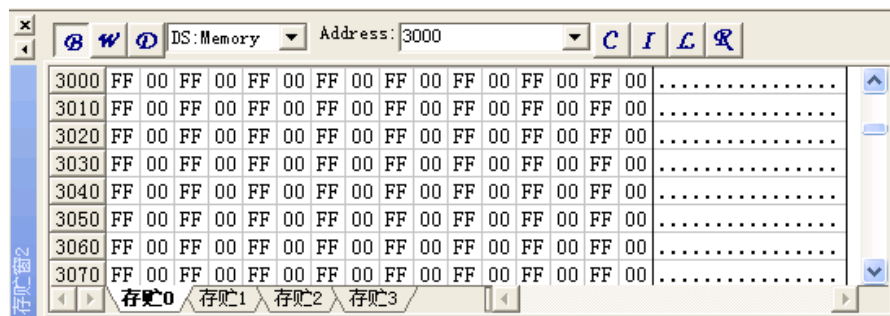


然后根据你的需要打开不同的窗口。调整后的调试界面为：

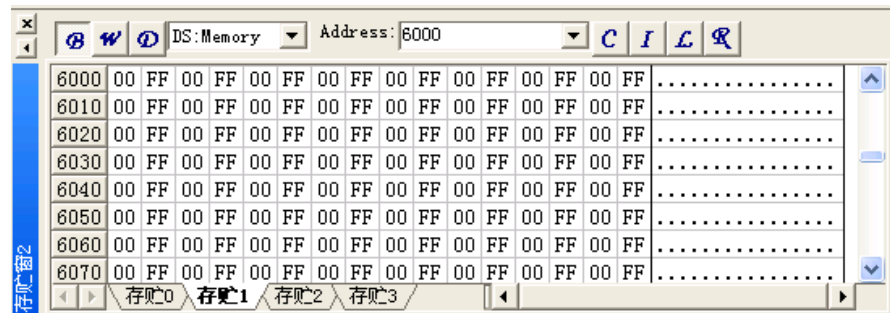


由于我们本次操作主要是观察存储器窗口，所以我们拉大了这两个存储器窗口的大小。每个窗口设置了 4 个分页项：**存储0** / **存储1** / **存储2** / **存储3**，我们可以在不同的分页项设置不同的观察数据空间以及地址范围。在 **DS: Memory** 中可以选择 CS: Memory, DS: Memory, I/O，根据需要可以做不同的选择。在 **Address: 0000** 中可以直接输入地址，然后按回车，就可以直接转到我们输入的地址的窗口上面观察数据。由于我们在此程序中的写入数据的 RAM 空间分别为 DS: 3000H~30FFH、DS: 6000H~60FFH、I/O 区 2000H~20FFH，故我们建立的分页项如下：

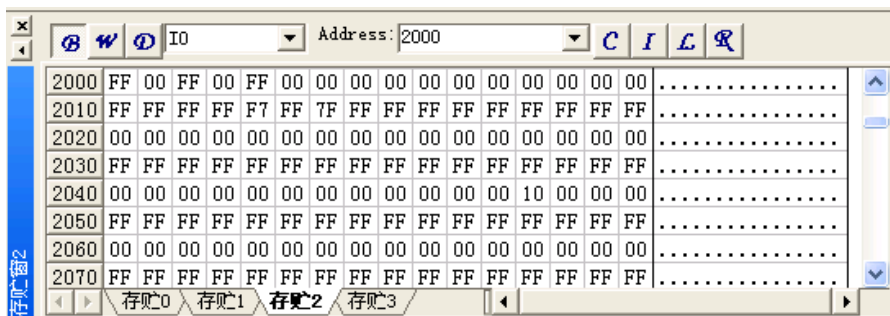
存储 0 分页项：



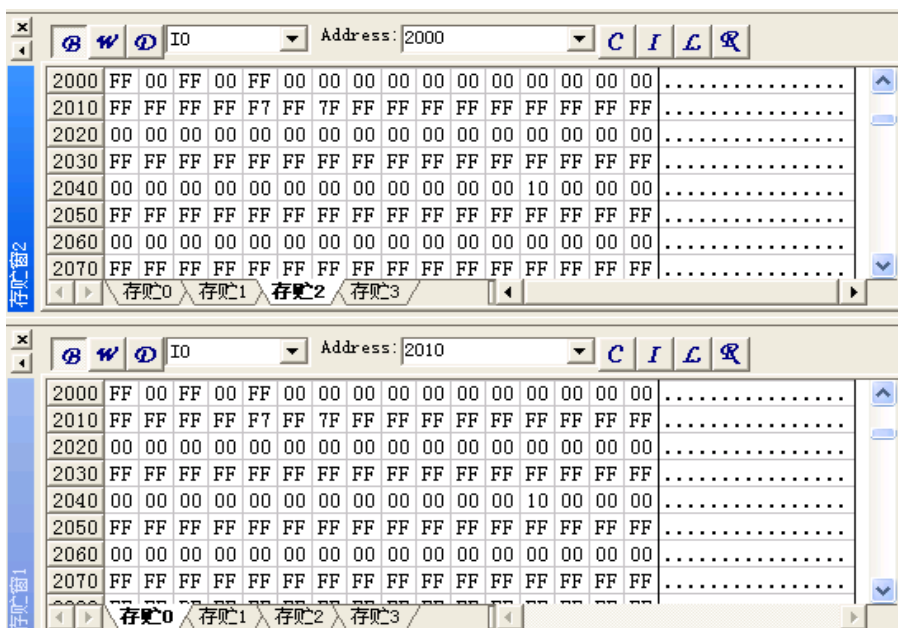
存贮 1 分页项:



存贮 2 分页项:

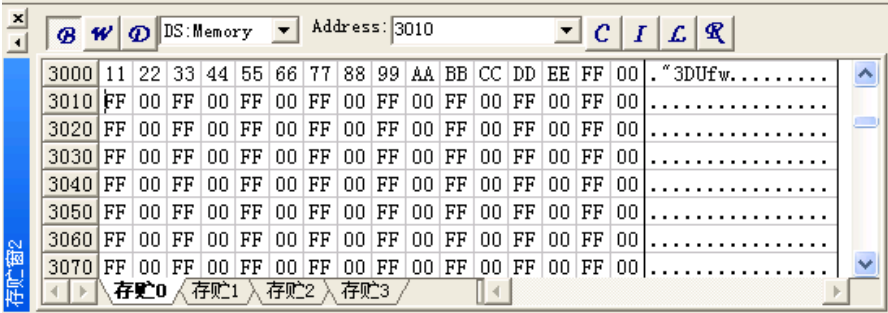


我们这样设置界面的目的就是当用户要观察不同地址段的数据时，只要切换一下分页项就行了。由于本次程序需要同时观察 DS: 3000~30FFH、I/O: 2000~20FFH 和 I/O: 2000H~20FFH、DS: 6000H~60FFH，所以打开二个存贮器窗。如图：



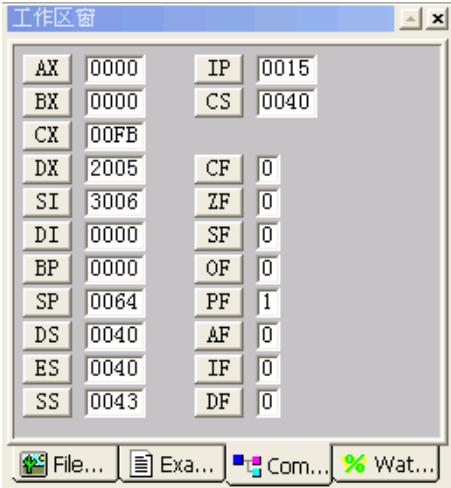
软件中总共存在 3 个存储器窗。可以同时观察三个不同的地址。

存储器窗口支持数据的直接修改功能。本软件的所有窗口中的数据都支持直接修改功能。用户可以根据自己的需要在窗口中直接修改数据。比如：执行程序前，将 DS: 3000H~300FH 中的数据改为 11、22、33、44、55、66、77、88、99、AA、BB、CC、DD、EE、FF、00，在相对应的地址中直接输入数据即可。如图：



一般刚刚写好的程序，在进入调试状态后，执行“单步”或者“单步进入”，我们推荐您能记住这些操作的相对应的功能键，这样您就在调试程序的过程中很方便。

在刚才的调试程序中我们多次执行“单步 (F8)”命令，在工作区窗口的 CommonRegister 视中查看通用的寄存器：



我们可以观察到在本程序中所使用的一些寄存器的变化，比如 AX、CX、DX、SI 的数值的变化，每一次循环，CX 减一，DX、SI 加一，AX 寄存器的低字节 AL 暂存从 DS: [SI]取出的数值：11、22、33、44、55、66……。



我们可以看到存储器窗口中的相对应的 RAM 的数据的变化。比如

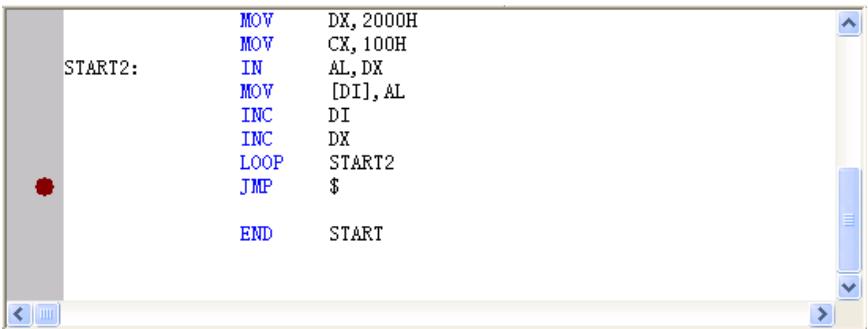




其中右边为相应数据的 ASCII 码。切换分页项我们可以观察到其它地址的数据。

把光标移动到 MOV DI, 6000H 行上，点击图标 (功能键 F4)，全速运行到光标行，检查 IO: 2000H~20FFH 内容，是否与 DS: 3000H~30FFH 相同，如果完全一样，说明以上程序没有任何问题。

切换分页项，存储器窗显示 DS: 6000H 开始的单元内容，将光标移到 JMP \$ 行的左边，

鼠标变为，点击鼠标，在该行上设置了一个断点，也可以用鼠标点击该行，将光标移到鼠标处，点击图标（功能键为 F2），设置断点，重复操作，清除断点。




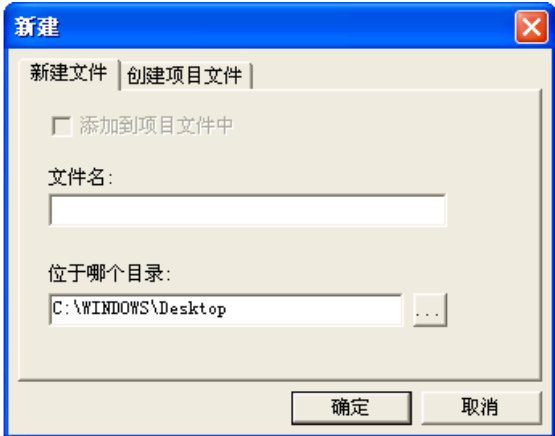
点击图标（功能键 F9），CPU 全速断点运行到光标处停下，检查 DS: 6000H~60FFH 内容，与 I/O: 2000H~20FFH 内容是否一样，相同表示程序没问题；如果不相同，将光标移到 MOV DI, 6000H 行上（具体操作是：用鼠标点击该行，然后再点击图标），使用单步进入命令 F7 或连续单步进入命令 Ctrl + F7，检查结果，判断程序出错原因。

3. 2. 2 数据传送程序 (C)

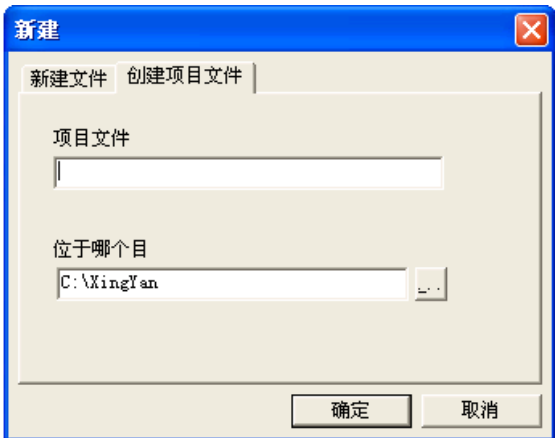
本例子使用项目文件来管理，旨在通过建立一个具体的项目来介绍星研集成软件的使用方法。如果您的系统有几个文件组成，就必须使用项目文件。

1、建立项目文件

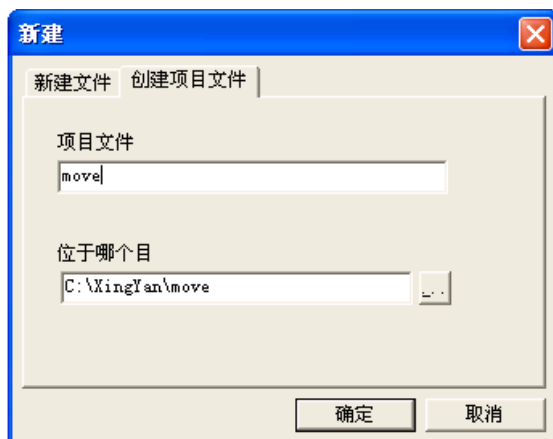
执行 [主菜单 » 文件 » 新建]，（或者点击图标）打开窗口如下：



由于星研集成软件是以项目为单位来管理程序的。所以我们在建立文件之前先要建立项目文件。点击“创建项目文件”分页项，如图示：



我们可以输入项目文件名，以及选择目录，星研集成软件在您输入一个项目文件名时，就建立了以项目文件名为名的一个文件夹，以后您在编译、调试过程中生成的所有文件都在此文件夹里。这体现了星研集成软件的人性化设计。键入项目文件名“move”，如下：



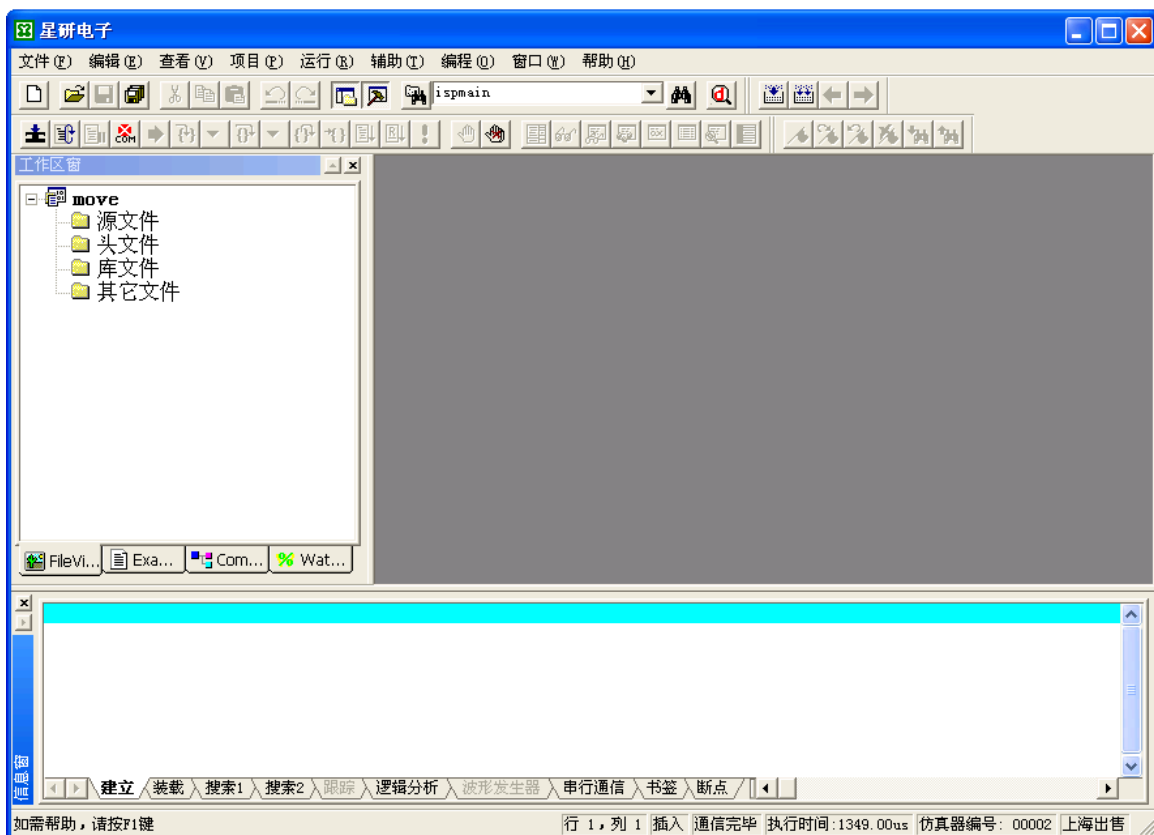
然后按确定，进入“设置项目文件”部分。

2、设置项目文件


设置项目文件与设置缺省项目操作完全一样，请参阅上一节。

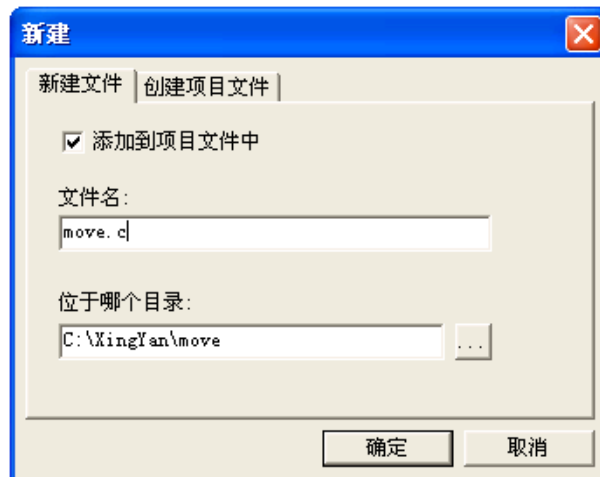
3、建立源文件

建立好项目文件的窗口如下图所示：



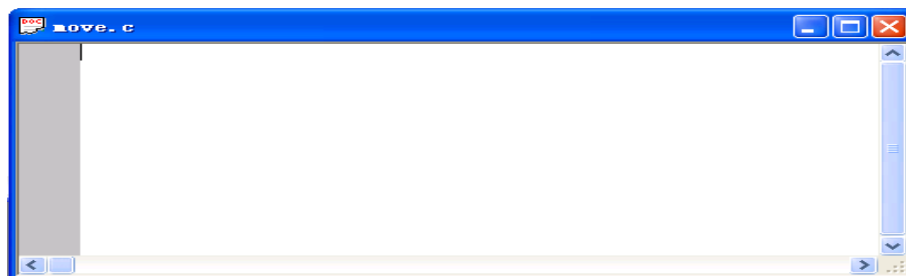
工作区窗的项目视图中，包含“源文件”、“头文件”、“库文件”、“其它文件”，“其它文件”中通常包含对该项目用途作一些说明的文件。“库文件”通常包含编译软件自带的 OBJ 文件、LIB 等库文件。

下面我们建立源文件，执行 [主菜单 » 文件 » 新建]，（或者点击图标）打开窗口如下：



选定刚才建立的项目文件的文件夹，输入文件名，注意：一定要输入文件名后缀。系统会根据不同的后缀名给文件归类。比如：*.asm 文件系统会自动归类为源文件。选中“添加到项目文件中”，系统自动将该模块文件加入到项目中。本实例文件名为move.c。

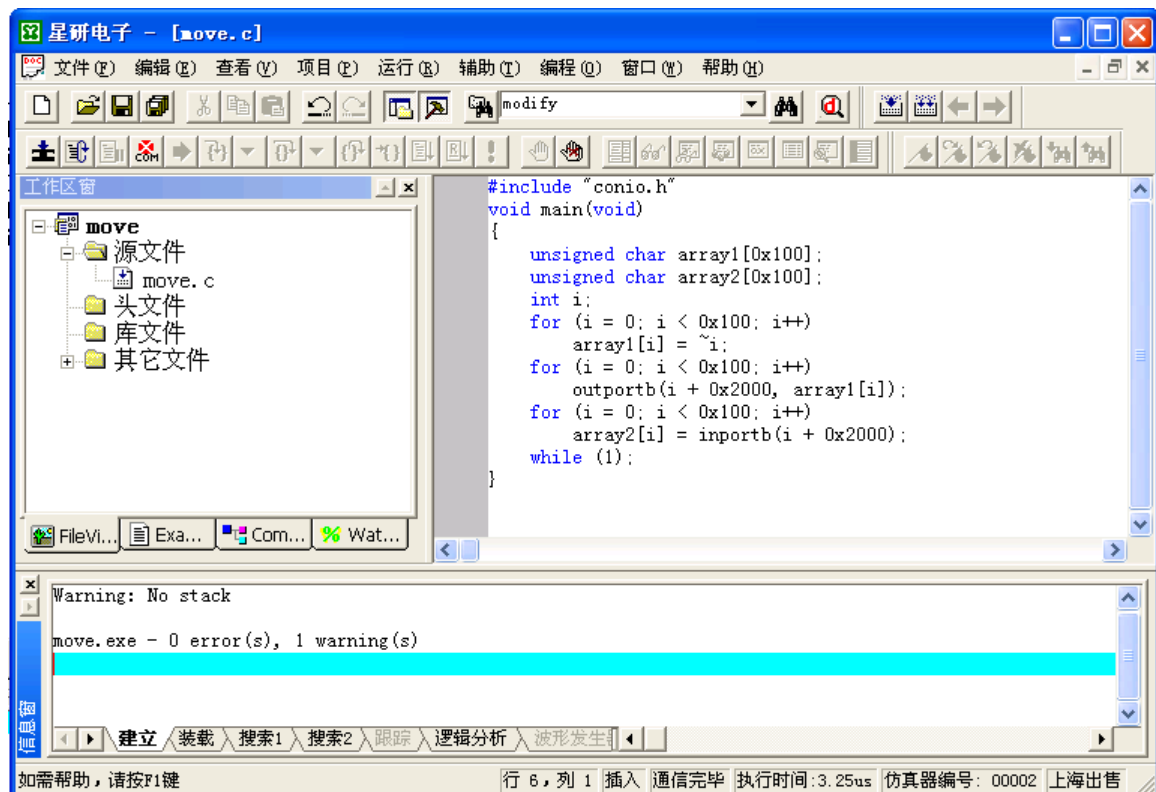
按“确定”即可。然后即出现文件编辑窗口：



程序清单：

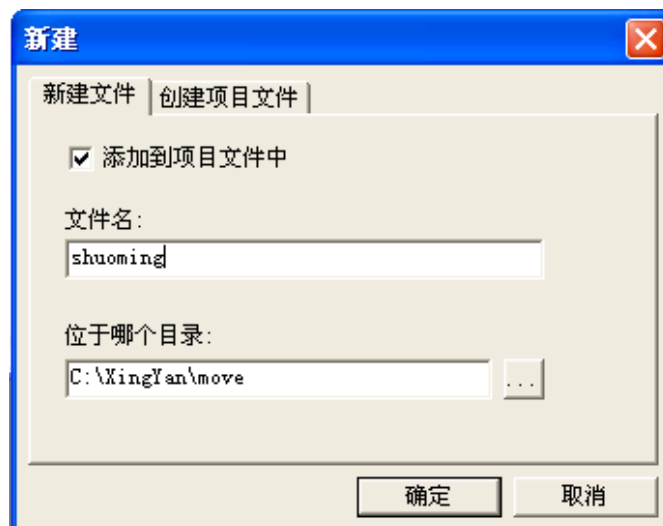
```
#include "conio.h"
void main(void)
{
    unsigned char array1[0x100];
    unsigned char array2[0x100];
    int i;
    for (i = 0; i < 0x100; i++)
        array1[i] = ~i;
    for (i = 0; i < 0x100; i++)
        outportb(i + 0x2000, array1[i]);
    for (i = 0; i < 0x100; i++)
        array2[i] = inportb(i + 0x2000);
    while (1);
}
```

建立好文件的窗口如下：

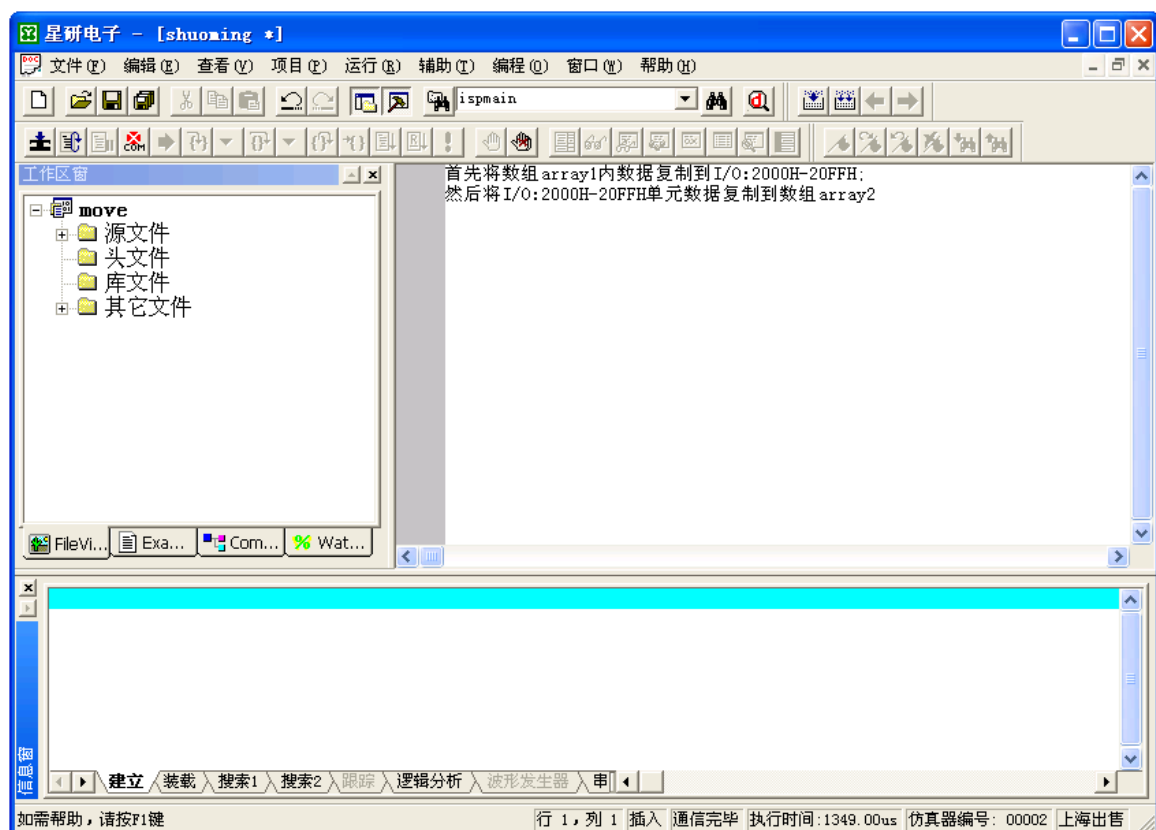


附：

注意：若在新建文件时不输入文件后缀，则其文件不会保存在源文件那一项，而是保存在其他文件的文件夹中。一般我们建立对项目说明的文件即可用此方法。如图建立一个本程序的说明文档“shuoming”





然后编辑文档，如下图：



然后保存，就可以了。

4、编译、连接文件

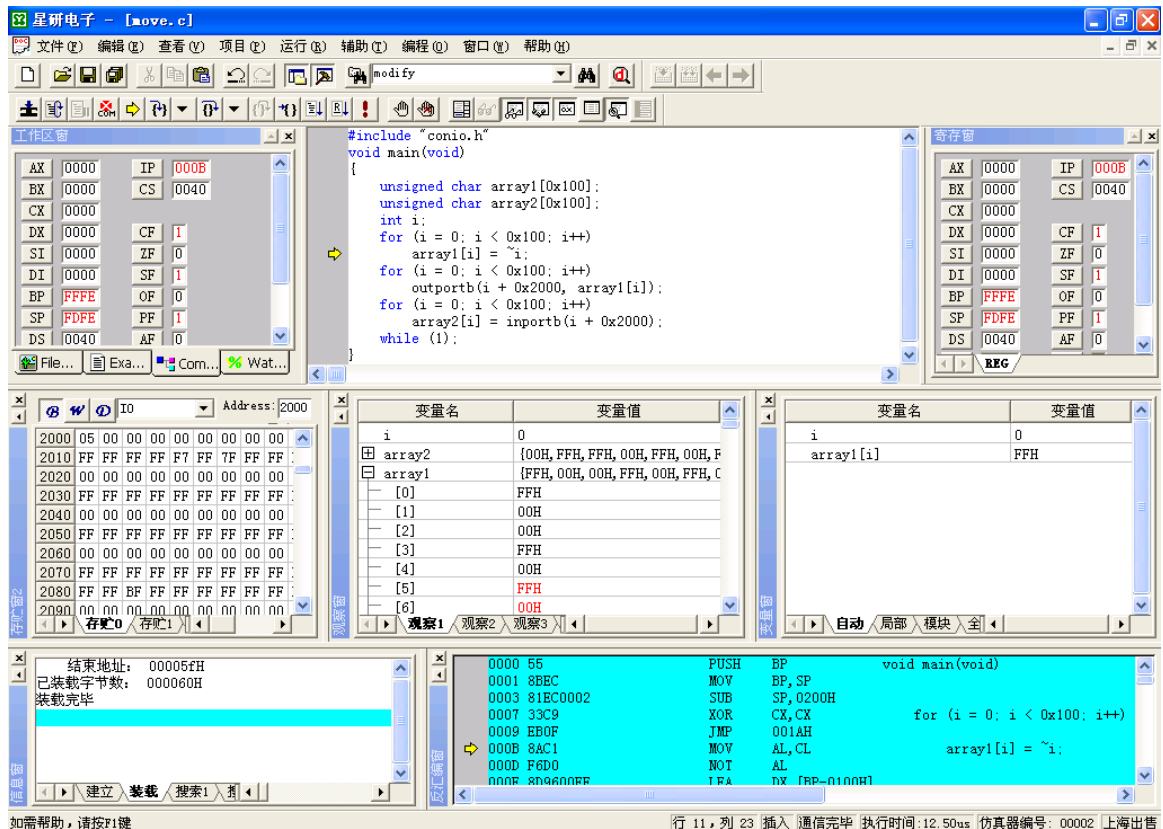
在建立好项目文件、源文件后，就可以编译、连接文件了。对工作区窗项目视的“源文件”中所有模块文件编译，如果没有错误，再与“库文件”中所有库文件连接，生成代码文件（DOB、EXE 文件）。编译、连接文件的方法有如下三种：（1）在工作区窗的项目视中按鼠标右键，系统弹出快捷菜单，选择“编译、连接”或“重新编译连接”。（2）使用[主菜单 » 项目 » 编译、连接]或[主菜单 » 项目 » 重新编译、连接]。（3）点击图标或来“编译、连接”或“重新编译连接”。

“编译连接”与“重新编译、连接”区别：“重新编译、连接”不管项目中有无添加、删除模块文件、编译软件是否变化、编译控制项有无修改、模块文件有无修改，对“源文件”中所有模块文件编译，如果没有错误，再与“库文件”中所有库文件连接，生成代码文件（DOB、EXE 文件）。编译、连接过程中产生的信息显示在信息窗的“建立”视中。

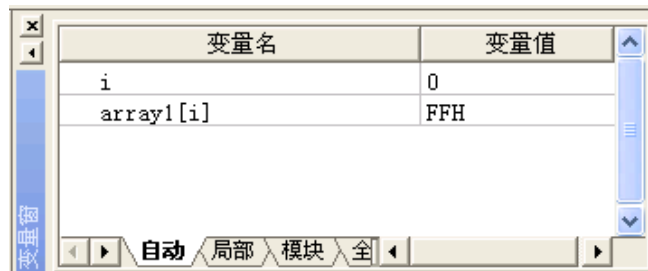
详细请参阅上一节。

5、调试项目文件

下面我们进入调试状态，调试 C 语言程序时，我们观察的比较多的是寄存器窗、观察窗和变量窗。所以我们把这些窗口放在前台，并调整至适当的大小。如图所示：



我们执行“单步”命令时就会在变量窗口中看到相应的变量的变化：



变量窗包含“自动”、“局部”、“模块”、“全局”四个标签视。

自动： 星研自动搜集当前行（PC 指针对应的文件行）及前二行上的变量。通常这三行有您最关心的变量，也是星研集成环境的一大特色。

局部： 显示当前函数或当前过程中的所有变量。

模块： 显示当前模块文件中所有模块级变量。

全局： 显示所有全局变量。

经常查看的变量分别放入观察窗的 4 个标签视中，您会感觉非常方便、快捷。在观察窗口中我们可以随意的添加我们想要观察的变量，具体方法是：（1）在文件窗中，用鼠标左键双击变量名，按住鼠标左键，将该变量名拖至观察窗中，释放鼠标左键，星研自动将该变量添至观察窗中。（推荐方法）（2）用鼠标左键双击观察窗中的虚线框，出现一个编辑框，在编辑框中输入一个变量名，输入回车即可。观察窗中的四个页面项的作用完全一样，当您要观察的变量很多时，用户可以在不同的页面项输入观察变量，这样观察时只要点击一下页面项就可以了。这也是星研软件的人性化设计的一个方面。比如；我们在观察窗中添加变量 i, array1, array2 观察，如图：



首先，请正确选择串口，星研软件通过串口，向实验仪发出各种测试命令；然后，选择使用的液晶，12864J：12864 图形点阵液晶；12864M：带汉字库的 12864 图形点阵液晶；12864C：12864 图形点阵液晶，通常同时配置触摸屏。

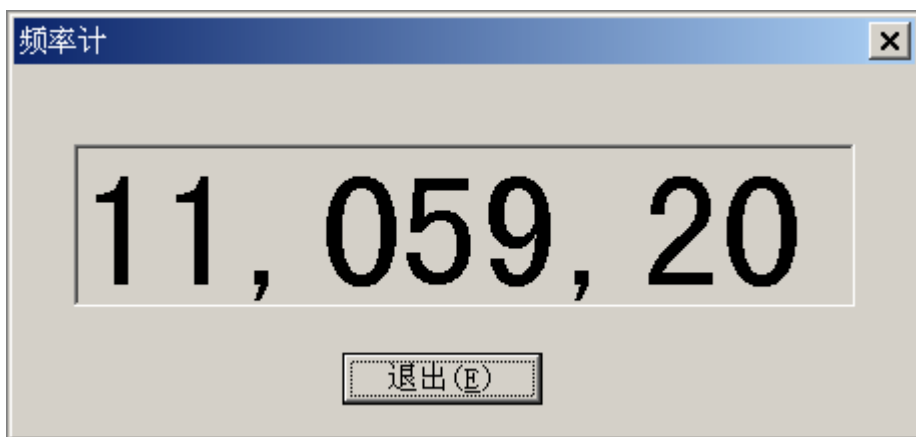
如果需要测试 RS485，必须选择当前实验仪是主机还是从机；如果还需测试模块，请选择模块。

如果只需测试实验仪的一部分，请自行选择。

点击“开始测试”，根据对话框的下半部分提示，完成对实验仪的完整测试。（整个测试过程，只需连接扁线、一、二根连接线，非常适合新学期开始，对所有作一检测，了解实验仪的实际状况。）

3. 4 频率计 (EMU598+)

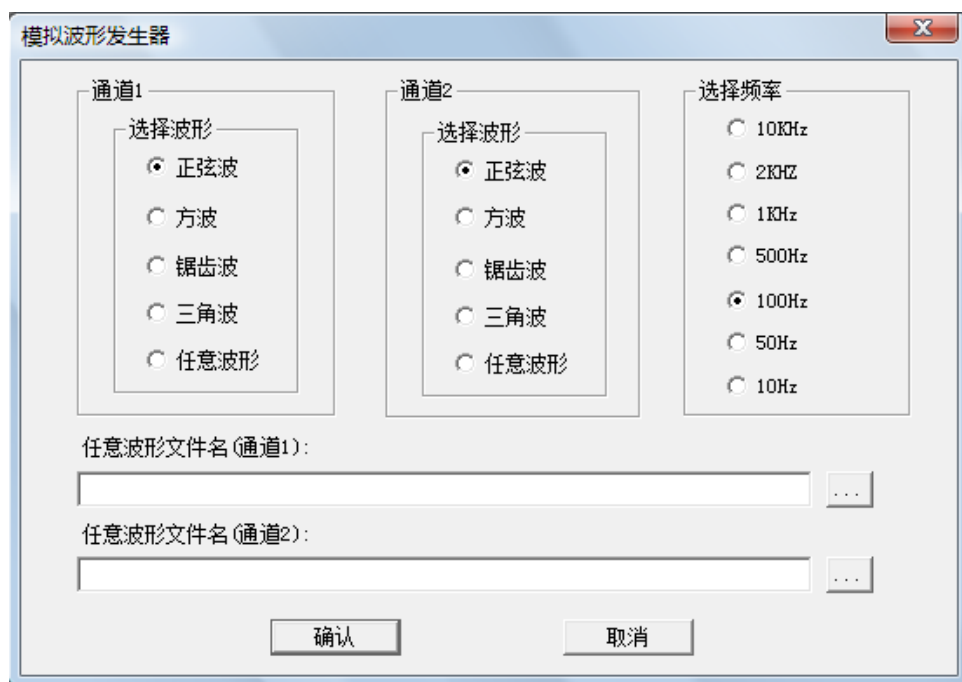
如果您需要测试 CPU 的振荡频率、您电路中其它信号的频率，您可以选择频率计功能：[主菜单 » 分析手段 » 频率计]。它可以测试 100M 以内的信号。



EMU598+仿真模块的 FREQ 与被测信号相连。
频率计、仿真部分可以并行运行。

3. 5 模拟波形发生器 (EMU598+)

EMU598+仿真模块可以提供 2 路模拟波形：正弦波、方波、锯齿波、三角波或自定义波形。
您可以选择模拟波形发生器功能：[主菜单 » 分析手段 » 模拟波形发生器]。



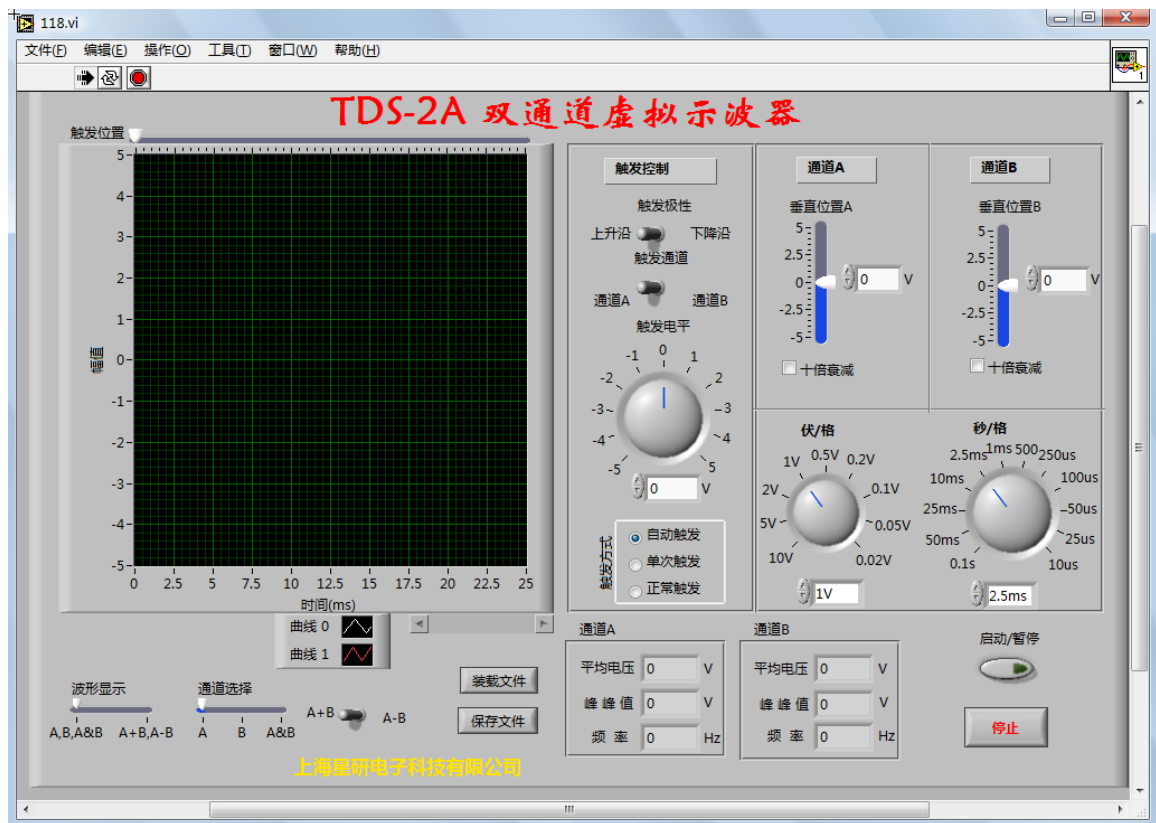
EMU598+仿真模块的 W1、W2 对应于通道 1、通道 2。

任意波形：首先创建一个 BIN 文件，包含 100 个字数据；每个字数据包含 12bit 二进制数，即每个字数据的有效范围为 0000H-0FFFH；根据您希望产生的波形，换算出 100 个字数据。

模拟波形发生器、仿真部分可以并行运行。

3. 6 TDS2、TDS2A (EMU598+) 虚拟示波器

TDS2 虚拟示波器模块的安装软件在“实验仪\TDS2”文件夹；TDS2A 虚拟示波器的安装软件在“实验仪\TDS2A (EMU598+)”文件夹，运行 SETUP.EXE 文件即可进入安装界面，您只需按程序提示一步一步进行安装即可。



点击“启动/暂停”按钮，可以启动或暂停虚拟示波器功能。操作与一般示波器类似。

4 软件实验

软件实验部分共有七个实验组成,通过对这些实验程序的编写、调试,使学生熟悉 8086/8088 的指令系统等,了解程序设计过程,掌握汇编程序、C 程序设计方法以及如何使用实验系统提供的各种调试、分析手段来排除程序错误。

实验一 数据传送

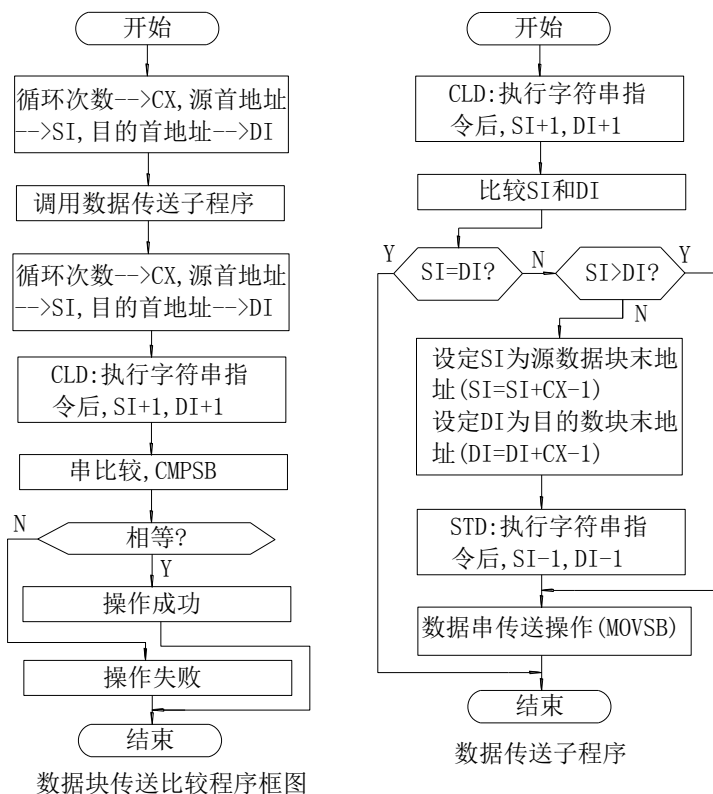
一、实验目的

熟悉星研集成环境软件的使用方法。熟悉 Borland 公司的 TASM 编译器熟悉 8086 汇编指令,能自己编写简单的程序,掌握数据传输的方法。

二、实验内容

- 1、熟悉星研集成环境软件。
- 2、编写程序,实现数据段的传送、校验。

三、程序框图



四、实验步骤

在 DS 段内 3000H~30FFH 中输入数据;使用单步、断点方式调试程序,检测 DS 段内 6000H ~ 60FFH 中的内容。熟悉查看特殊功能寄存器、CS 段、DS 段的各种方法。

五、程序清单

```
.MODEL                TINY

.STACK                100
.DATA

.CODE
START:  MOV            AX, @DATA
        MOV            DS, AX
        MOV            ES, AX
        NOP
        MOV            CX, 100H
        MOV            SI, 3000H
        MOV            DI, 6000H
        CALL           Move
        MOV            CX, 100H
        MOV            SI, 3000H
        MOV            DI, 6000H
        CLD
        REPE           CMPSB
        JNE            ERROR
TRUE:   JMP            $
ERROR:  JMP            $
Move    PROC          NEAR
        CLD
        CMP            SI, DI
        JZ             Return
        JNB            Move1
        ADD            SI, CX
        DEC            SI
        ADD            DI, CX
        DEC            DI
        STD
Move1:  REP            MOVSB
Return: RET
Move    ENDP
END                START
```

六、思考题

1、子程序 Move 中为什么比较 SI、DI？

源数据块与目标范围有可能部分重叠，需要考虑从第一个字节开始复制（顺序复制），还是从最后一个字节开始复制（倒序复制）。

2、编写一个程序，将 DS 段中的数据传送到实验仪 B4 区的 61C256 中。

说明：B4 区的 61C256 在 I/O 设备区，使用 $\overline{\text{IOR}}$ 、 $\overline{\text{IOW}}$ 读写。

实验二 双字节 BCD 码(十进制数)加法

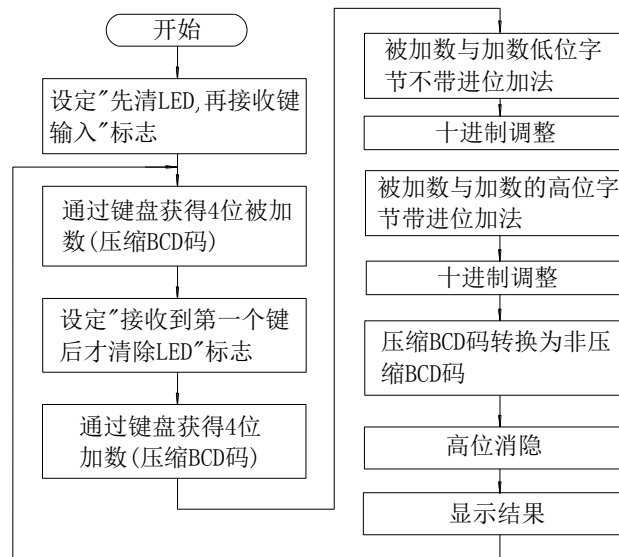
一、实验目的

熟悉 8086 汇编指令，学会使用星研集成环境软件，能自己编写简单的程序，熟悉 BCD 码，了解如何调用系统提供的子程序。

二、实验内容

从键盘上输入 4 位被加数、加数，实现双字节 BCD 码(四位数)的加法，结果显示在数码管上；熟悉使用断点、单步进入、单步、运行到光标处、修改 PC 指针、全速运行等各种调试手段；熟悉查看特殊功能寄存器、CS 段、DS 段存贮器的各种方法。

三、实验框图



双字节BCD码加法程序框图

四、实验步骤

1、连线说明：

D3 区 : CLK	——	B2 区: 2M
D3 区 : CS	——	A3 区: CS5
D3 区 : A0	——	A3 区: A0

2、在 F4 区的键盘上输入 4 位被加数、加数

3、结果显示在 F4 区的数码管上

五、程序清单

```
.MODEL            TINY
EXTRN             Display8:NEAR,GetBCDKey:NEAR
EXTRN             F1:BYTE
                  .STACK    100
                  .DATA
BUFFER            DB            8 DUP(?)
```

augend	DB	2 DUP (?)	;被加数
addend	DB	2 DUP (?)	;加数
	. CODE		
START:	MOV	AX, @DATA	
	MOV	DS, AX	
	MOV	ES, AX	
	NOP		
	MOV	F1, 0	;先清除显示，再接收键输入
START1:	LEA	DI, augend	
	MOV	CX, 4	;按键次数
	CALL	GetBCDKey	;得到双字节十进制数(被加数)
	MOV	F1, 1	;接收到第一个键，才清除显示
	LEA	DI, addend	
	MOV	CX, 4	;按键次数
	CALL	GetBCDKey	;得到双字节十进制数(加数)
	MOV	AL, augend	
	ADD	AL, addend	;低位
	DAA		
	XCHG	AL, AH	
	MOV	AL, augend + 1	
	ADC	AL, addend + 1	;高位
	DAA		
	XCHG	AL, AH	
	MOV	BL, 0	
	ADC	BL, 0	;进位
	CLD		
	LEA	DI, BUFFER	;存放显示结果
	CALL	B1toB2	;低位
	MOV	AL, AH	
	CALL	B1toB2	;高位
	MOV	AL, BL	
	STOSB		
	MOV	AL, 10H	;最高三位消隐
	STOSB		
	STOSB		
	STOSB		
	LEA	SI, BUFFER+4	
	MOV	CX, 4	
	CALL	BlackDisplay	;将高位0消隐
	LEA	SI, BUFFER	
	CALL	Display8	
	JMP	START1	
;将一个字节压缩BCD码转换成二个字节非压缩BCD码			
B1toB2	PROC	NEAR	

	PUSH	AX	
	AND	AL, 0FH	
	STOSB		
	POP	AX	
	AND	AL, 0F0H	
	ROR	AL, 4	
	STOSB		
	RET		
B1toB2	ENDP		
BlackDisplay	PROC	NEAR	
	STD		
	MOV	DI, SI	
BlackDisplay1:	LODSB		;将高位0消隐
	CMP	AL, 0	
	JNZ	Exit	
	MOV	AL, 10H	
	STOSB		
	LOOP	BlackDisplay1	
Exit:	CLD		
	RET		
BlackDisplay	ENDP		
	END	START	

实验三 双字节 BCD 码(十进制数)减法

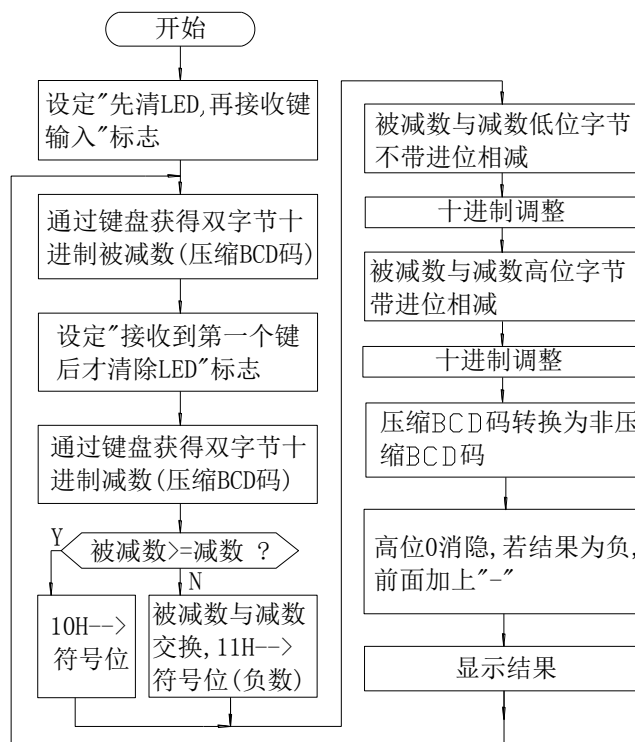
一、实验目的

熟悉 8086 汇编指令，学会使用星研集成环境软件，掌握汇编语言的设计和调试方法。能看懂程序流程框图，能自己设计程序，熟悉 BCD 码、补码，了解如何调用系统提供的子程序。

二、实验内容

从键盘上输入 4 位被减数、减数，实现双字节 BCD 码(四位数)的减法，结果显示在数码管上；进一步熟悉使用断点、单步进入、单步、运行到光标处、修改 PC 指针、全速运行等各种调试手段；熟悉查看特殊功能寄存器、CS 段、DS 段存取器的各种方法。

三、程序框图



双字节十进制减法程序框图

四、实验步骤

1、连线说明：

D3 区 : CLK	——	B2 区: 2M
D3 区 : CS	——	A3 区: CS5
D3 区 : A0	——	A3 区: A0

2、在 F4 区的键盘上输入 4 位被减数、减数

3、结果显示在 F4 区的数码管上

五、程序清单

```
.MODEL            TINY
EXTRN             Display8:NEAR, GetBCDKey:NEAR
EXTRN             F1:BYTE
```

	. STACK	100	
	. DATA		
BUFFER	DB	8 DUP(?)	
minuend	DW	1 DUP(?)	;被减数
subtrahend	DW	1 DUP(?)	;减数
	. CODE		
START:	MOV	AX, @DATA	
	MOV	DS, AX	
	MOV	ES, AX	
	NOP		
	MOV	F1, 0	;先清除显示，再接收键输入
START1:	LEA	DI, minuend	
	MOV	CX, 4	;按键次数
	CALL	GetBCDKey	;得到双字节十进制数(被减数)
	MOV	F1, 1	;接收到第一个键，才清除显示
	LEA	DI, subtrahend	
	MOV	CX, 4	;按键次数
	CALL	GetBCDKey	;得到双字节十进制数(减数)
	MOV	AX, minuend	
	MOV	BX, subtrahend	
	MOV	DL, 10H	
	CMP	AX, BX	
	JNB	START2	
	XCHG	AX, BX	
	MOV	DL, 11H	;负数
START2:	SUB	AL, BL	;低位
	DAS		
	XCHG	AL, AH	
	SBB	AL, BH	;高位
	DAS		
	XCHG	AL, AH	
	CLD		
	LEA	DI, BUFFER	;存放显示结果
	CALL	B1toB2	;低位
	MOV	AL, AH	
	CALL	B1toB2	;高位
	MOV	AL, 10H	;最高三位消隐
	STOSB		
	STOSB		
	STOSB		
	STOSB		
	LEA	SI, BUFFER+3	
	MOV	CX, 3	
	CALL	BlackDisplay	;将高位0消隐


```

                                LEA            SI, BUFFER
                                CALL           Display8
                                JMP            START1
;将一个字节压缩BCD码转换成二个字节非压缩BCD码
B1toB2      PROC                NEAR
                                PUSH          AX
                                AND           AL, 0FH
                                STOSB
                                POP           AX
                                AND           AL, 0F0H
                                ROR           AL, 4
                                STOSB
                                RET
B1toB2      ENDP
BlackDisplay PROC                NEAR                ;将高位0消隐
BlackDisplay1: MOV             AL, [SI]
                                CMP           AL, 0
                                JNZ           Exit
                                MOV           AL, 10H
                                MOV           [SI], AL
                                DEC           SI
                                LOOP          BlackDisplay1
Exit:        CLD
                                MOV           [SI+1], DL
                                RET
BlackDisplay ENDP

                                END            START

```

六、思考题

带符号的十进制数加法程序如何编写？

实验四 四字节十六进制数转十进制数

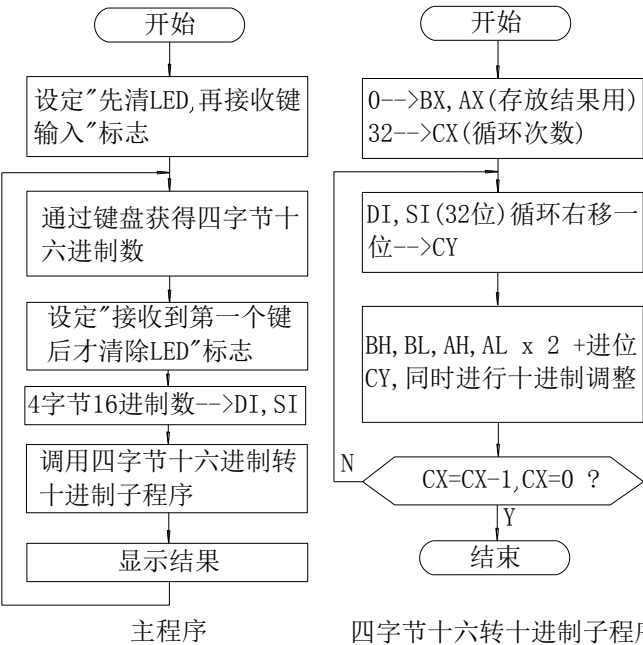
一、实验目的

进一步熟悉 8086 汇编指令，了解十六进制数转十进制数的方法。

二、实验内容

从键盘上输入 8 位十六进制数，实现四字节十六进制数转 8 位十进制数，并在数码管上显示。注意输入数据必须在 00000000H~05F00000H 范围，否则，结果超出 8 位十进制数，无法正确显示。

三、程序框图



四、实验步骤

1、连线说明：

D3 区 : CLK	——	B2 区: 2M
D3 区 : CS	——	A3 区: CS5
D3 区 : A0	——	A3 区: A0

2、在 F4 区的键盘上输入 8 位十六进制数

3、结果显示在 F4 区的数码管上

五、程序清单

```
.MODEL            TINY
EXTRN             Display8:NEAR, GetKey:NEAR
EXTRN             F1:BYTE
                  .STACK            100
                  .DATA
BUFFER            DB                8 DUP(?)
```

```

        . CODE
START:   MOV     AX, @DATA
        MOV     DS, AX
        MOV     ES, AX
        NOP
        MOV     F1, 0           ;先清除显示，再接收键输入
START1:  LEA     DI, BUFFER
        MOV     CX, 8           ;按键次数
        CALL    GetKey          ;得到4字节十六进制数
        MOV     F1, 1           ;接收到第一个键，才清除显示
        MOV     SI, WORD PTR BUFFER
        MOV     DI, WORD PTR BUFFER + 2
        CALL    B4toD4          ;转换成十进制数
        LEA     DI, BUFFER      ;存放显示结果
        CALL    B1toB2          ;低位
        MOV     AL, AH
        CALL    B1toB2
        MOV     AL, BL
        CALL    B1toB2
        MOV     AL, BH
        CALL    B1toB2
        LEA     SI, BUFFER+7
        MOV     CX, 7
        CALL    BlackDisplay    ;将高位0消隐
        LEA     SI, BUFFER
        CALL    Display8
        JMP     START1
;将一个字节压缩BCD码转换成二个字节非压缩BCD码
B1toB2  PROC     NEAR
        PUSH    AX
        AND     AL, 0FH
        STOSB
        POP     AX
        AND     AL, 0F0H
        ROR     AL, 4
        STOSB
        RET
B1toB2  ENDP
BlackDisplay  PROC     NEAR
        STD
        MOV     DI, SI
BlackDisplay1: LODSB            ;将高位0消隐
        CMP     AL, 0
        JNZ     Exit

```

```

                                MOV            AL, 10H
                                STOSB
                                LOOP          BlackDisplay1
Exit:                          CLD
                                RET
BlackDisplay                  ENDP
;四字节十六进制数转十进制数: DISI为十六进制, BXAX为压缩BCD码
B4toD4                        PROC            NEAR
                                XOR            AX, AX
                                XOR            BX, BX
                                MOV            CX, 32
B4toD4_1:                     RCL            SI, 1
                                RCL            DI, 1
                                ADC            AL, AL
                                DAA
                                XCHG           AL, AH
                                ADC            AL, AL
                                DAA
                                XCHG           AL, BL
                                ADC            AL, AL
                                DAA
                                XCHG           AL, BH
                                ADC            AL, AL
                                DAA
                                XCHG           AL, BH
                                XCHG           AL, BL
                                XCHG           AL, AH
                                LOOP          B4toD4_1
                                RET
B4toD4                        ENDP

                                END            START

```

六、思考题

如果不考虑在数码管上显示, 不限制数据范围, 程序应如何编写。

实验五 散转

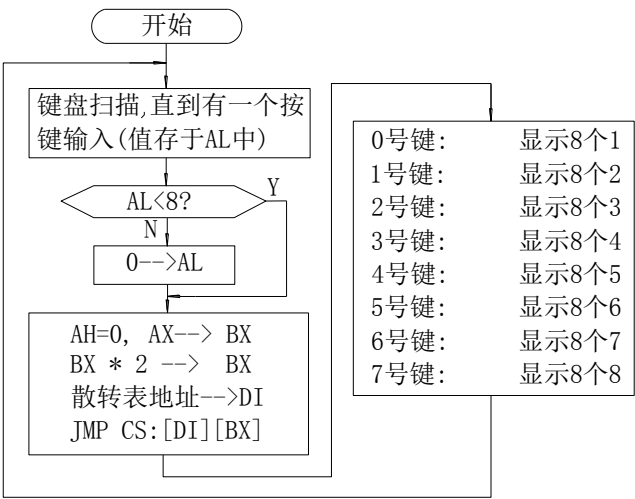
一、实验目的

熟悉使用 8086 指令，掌握汇编语言的设计和调试方法；理解并能运用散转指令。

二、实验内容

编写程序，根据接收到的键值，作不同的处理。

三、程序框图



散转程序流程框图

四、实验步骤

1、连线说明：

D3 区：CLK	——	B2 区：2M
D3 区：CS	——	A3 区：CS5
D3 区：A0	——	A3 区：A0

2、在 F4 区的键盘上输入 1 位数

3、使用各种手段调试程序

3、结果显示在 F4 区的数码管上

五、程序清单

;调用 GetKeyB 返回键值, 根据键值执行相应的程序

```
.MODEL TINY
EXTRN Display8:NEAR, GetKeyB:NEAR
.STACK 100
.DATA
BUFFER DB 8 DUP(?)
.CODE
START: MOV AX, @DATA
MOV DS, AX
MOV ES, AX
```

```

                                NOP
START1:                       CALL     GetKeyB           ;键值存放在AL中
                                CMP      AL, 8
                                JB       START2
                                XOR      AL, AL           ;大于7, 作0处理
START2:                       XOR      AH, AH
                                MOV      BX, AX
                                SHL      BX, 1
                                LEA      DI, Table_1
                                JMP      CS:[DI][BX]
Table_1:                      DW       Key0, Key1, Key2, Key3, Key4, Key5, Key6, Key7
Key0:                         MOV      AL, 1
                                JMP      Key
Key1:                         MOV      AL, 2
                                JMP      Key
Key2:                         MOV      AL, 3
                                JMP      Key
Key3:                         MOV      AL, 4
                                JMP      Key
Key4:                         MOV     AL, 5
                                JMP      Key
Key5:                         MOV      AL, 6
                                JMP      Key
Key6:                         MOV      AL, 7
                                JMP      Key
Key7:                         MOV      AL, 8
                                JMP      Key
Key:                          MOV      CX, 8
                                LEA      DI, BUFFER
                                REP      STOSB
                                LEA      SI, BUFFER
                                CALL     Display8
                                MOV      CX, 60000
                                LOOP     $               ;延时
                                JMP      START1

                                END      START

```

六、思考题

程序中为什么要把输入的值作乘以 2 处理？

实验六 冒泡排序

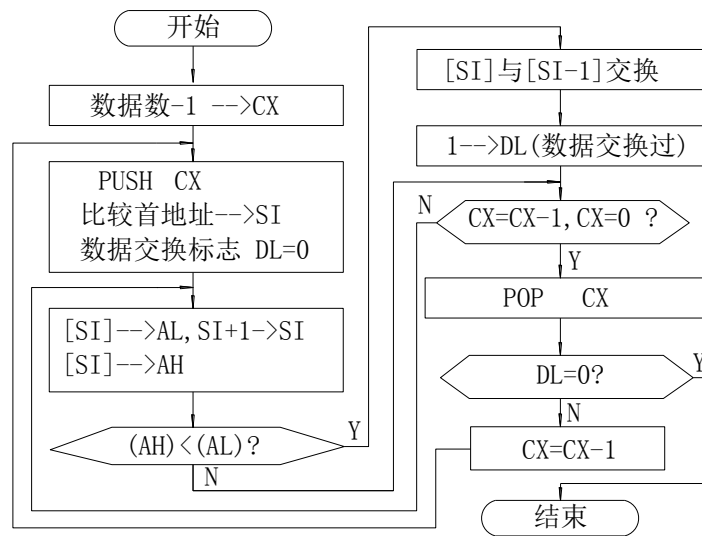
一、实验目的

熟悉使用 8086 指令,掌握汇编语言的设计和调试方法,了解如何使用高效方法对数据排序。

二、实验内容

编写并调试一个排序程序,要求使用冒泡法将一组数据从小到大重新排列。

三、程序框图



冒泡排序程序框图

四、实验步骤

使用断点方式调试程序,检查数据段中数据是否按从小到大的顺序排列。

五、程序清单

```
.MODEL            TINY
.STACK            100
.DATA
TAB_1:            DB  0H, 5H, 6H, 3H, 8H, 92H, 04H, 57H, 46H, 01H, 0FFH, 0A0H, 45H, 99H, 55H, 66H
.CODE
START:            MOV     AX, @DATA
                  MOV     DS, AX
                  NOP
                  MOV     CX, 16 - 1           ;存放比较次数 = 数据个数 - 1
STAR2:            PUSH    CX
                  LEA     SI, TAB_1
                  MOV     DL, 0                ;0->交换过数据标志
STAR3:            LODSB
                  MOV     AH, [SI]
                  CMP     AH, AL
```

	JNB	STAR5	
	MOV	[SI], AL	
	MOV	[SI - 1], AH	
	MOV	DL, 1	;1->交换过数据标志
STAR5:	LOOP	STAR3	
	POP	CX	
	CMP	DL, 0	
	JZ	Exit	
	LOOP	STAR2	
Exit:	JMP	\$	
	END	START	

六、思考题

你还知道哪些排序方法？另外编写一个排序子程序。（数据结构方面的教材上有十几种排序方法）。

实验七 二分查找法

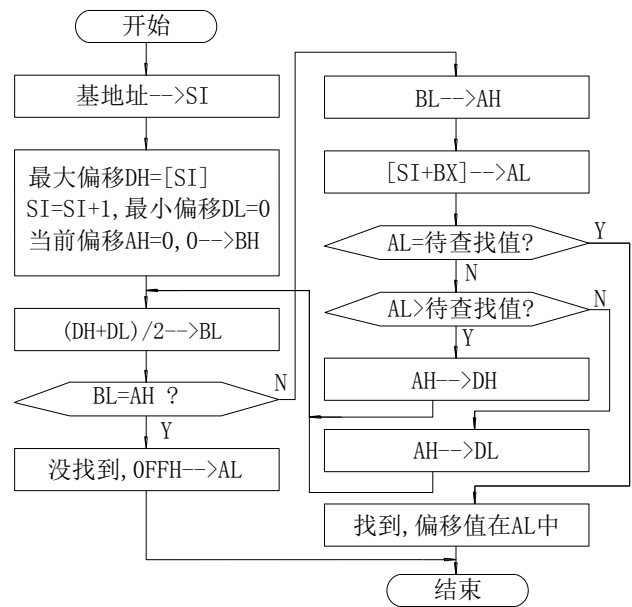
一、实验目的

熟悉使用 8086 指令，掌握汇编语言的设计和调试方法。

二、实验内容

编写并调试一个二分查找法程序，要求在一组从小到大排列的数据中查找一个数。

三、程序框图



二分查找法流程框图

四、实验步骤

在 Search_Data 中定义一个需要查找的数据，运行程序，是否能找到指定的数据，结果是否正确。

五、程序清单

```
.MODEL            TINY
Search_Data      EQU      60                ;需要查找的数据
                .STACK    100
                .DATA
TAB_1:          DB      32                ;共有32个数
                DB      01, 03, 05, 06, 07, 10, 11, 12, 13, 14, 15, 16, 20, 25, 26, 29
                DB      37, 38, 39, 42, 43, 44, 45, 50, 52, 53, 56, 59, 60, 62, 66, 68
                .CODE
START:          MOV      AX, @DATA
                MOV      DS, AX
                NOP
                LEA      SI, TAB_1
```

```

                                LODSB
                                MOV     DH, AL           ;最大位置
                                MOV     DL, 0           ;最小位置
                                MOV     AH, 0           ;当前位置
                                XOR     BH, BH
STAR1:                        MOV     BL, DH
                                ADD     BL, DL
                                CLC
                                SHR     BL, 1
                                CMP     BL, AH
                                JNE     STAR2
                                MOV     AL, 0FFH        ;没有找到
                                JMP     NoFind
STAR2:                        MOV     AH, BL
                                MOV     AL, [SI + BX]
                                CMP     AL, Search_Data
                                JNZ     STAR3
                                MOV     AL, AH
                                JMP     Find
STAR3:                        JB      STAR4
                                MOV     DH, AH
                                JMP     STAR1
STAR4:                        MOV     DL, AH
                                JMP     STAR1
Find:                         JMP     $
NoFind:                       JMP     $

                                END         START

```

六、思考题

1、程序只能实现 256 字节范围内的查找，请读者考虑，若查找范围大于 256 字节，程序该怎么编写？

5 基础硬件实验

基础硬件实验

本章和下一章将结合实验仪的所有单元电路（包括标准配置和可选各种模块）向读者逐一介绍各个实验，由浅入深，从最基础的实验开始，直到读者学会使用当今流行的各种外围电路，开发有一定深度的项目。硬件实验分为基础实验和综合实验两部分，本章主要介绍常用外围电路；综合实验介绍一些新颖外围电路，将各个单元电路灵巧组合、深入挖掘，生成一些具有实际意义的工程。读者也可以根据自己的理解、需要，将各个单元电路自行组合而成具有实际意义的复杂控制电路，在设计电路板前，在实验仪上作一认证。可见，STAR ES598PCI 适合于不同层次的学者、工程师以及电子爱好者进行学习、实践，STAR ES598PCI 提供了完整的源代码，可以直接复制到用户系统中，为你节省大量宝贵时间。

实验一 8255 控制交通灯实验

一、实验目的与要求

- 1、了解 8255 芯片的工作原理，熟悉其初始化编程方法以及输入、输出程序设计技巧。学会使用 8255 并行接口芯片实现各种控制功能，如本实验（控制交通灯）等。
- 2、熟悉 8255 内部结构和与 8088 的接口逻辑，熟悉 8255 芯片的 3 种工作方式以及控制字格式。
- 3、认真预习本节实验内容，尝试自行编写程序，填写实验报告。

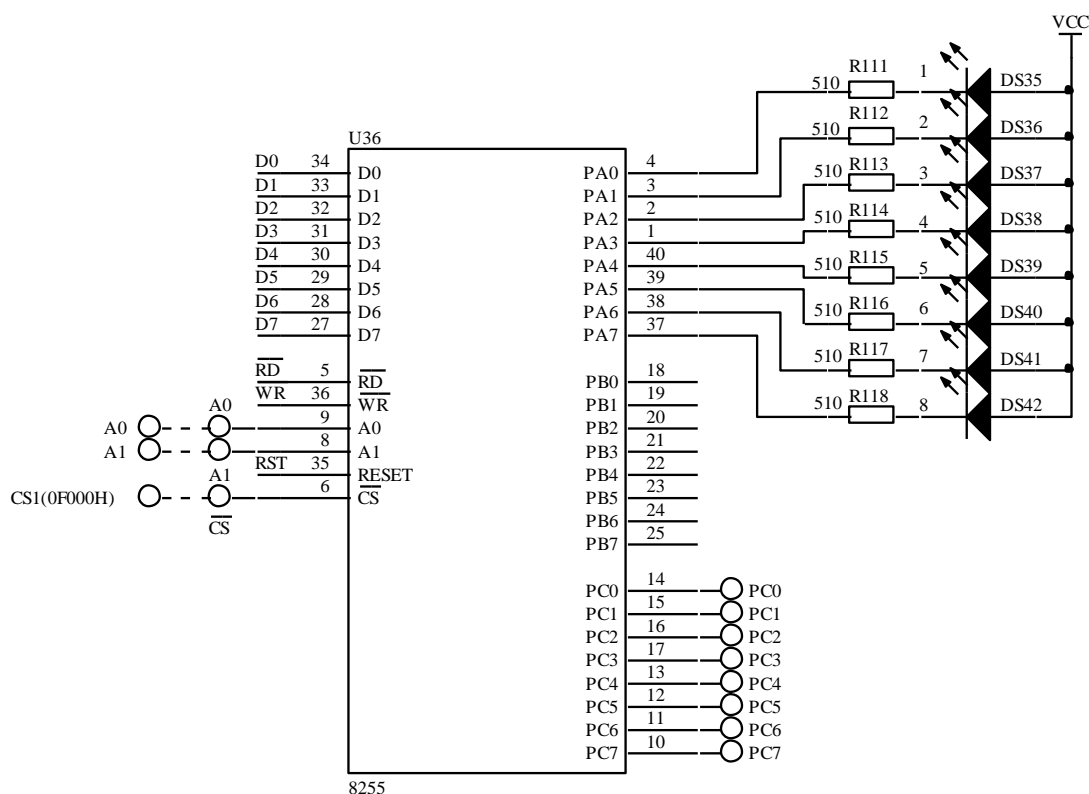
二、实验设备

STAR 系列实验仪一套、PC 机一台

三、实验内容

- 1、编写程序：使用 8255 的 PA0..2、PA4..6 控制 LED 指示灯，实现交通灯功能。
- 2、连接线路验证 8255 的功能，熟悉它的使用方法。

四、实验原理图



五、实验步骤

- 1、连线说明：

B4 区：CS(8255)、A0、A1	——	A3 区：CS1、A0、A1
B4 区：JP56 (PA 口)	——	F5 区：JP65

- 2、观察实验结果，是否能看到模拟的交通灯控制过程。

六、演示程序

```

.MODEL TINY
COM_ADD EQU 0F003H
PA_ADD EQU 0F000H
PB_ADD EQU 0F001H
PC_ADD EQU 0F002H
.STACK 100
.DATA
LED_Data DB 01111101B ;东西绿灯，南北红灯
          DB 11111101B ;东西绿灯闪烁，南北红灯
          DB 10111101B ;东西黄灯亮，南北红灯
          DB 11010111B ;东西红灯，南北绿灯
          DB 11011111B ;东西红灯，南北绿灯闪烁
          DB 11011011B ;东西红灯，南北黄灯亮
.CODE
START: MOV AX, @DATA
        MOV DS, AX
        NOP
        MOV DX, COM_ADD
        MOV AL, 80H ;PA、PB、PC为基本输出模式
        OUT DX, AL
        MOV DX, PA_ADD ;灯全熄灭
        MOV AL, 0FFH
        OUT DX, AL
        LEA BX, LED_Data
START1: MOV AL, 0
        XLAT
        OUT DX, AL ;东西绿灯，南北红灯
        CALL DL5S
        MOV CX, 6
START2: MOV AL, 1
        XLAT
        OUT DX, AL ;东西绿灯闪烁，南北红灯
        CALL DL500ms
        MOV AL, 0
        XLAT
        OUT DX, AL
        CALL DL500ms
        LOOP START2
        MOV AL, 2 ;东西黄灯亮，南北红灯
        XLAT
        OUT DX, AL
        CALL DL3S
        MOV AL, 3 ;东西红灯，南北绿灯
        XLAT

```

	OUT	DX, AL	
	CALL	DL5S	
	MOV	CX, 6	
START3:	MOV	AL, 4	;东西红灯，南北绿灯闪烁
	XLAT		
	OUT	DX, AL	
	CALL	DL500ms	
	MOV	AL, 3	
	XLAT		
	OUT	DX, AL	
	CALL	DL500ms	
	LOOP	START3	
	MOV	AL, 5	;东西红灯，南北黄灯亮
	XLAT		
	OUT	DX, AL	
	CALL	DL3S	
	JMP	START1	
DL500ms	PROC	NEAR	
	PUSH	CX	
	MOV	CX, 60000	
DL500ms1:	LOOP	DL500ms1	
	POP	CX	
	RET		
DL500ms	ENDP		
DL3S	PROC	NEAR	
	PUSH	CX	
	MOV	CX, 6	
DL3S1:	CALL	DL500ms	
	LOOP	DL3S1	
	POP	CX	
	RET		
	ENDP		
DL5S	PROC	NEAR	
	PUSH	CX	
	MOV	CX, 10	
DL5S1:	CALL	DL500ms	
	LOOP	DL5S1	
	POP	CX	
	RET		
	ENDP		
	END	START	

七、实验扩展及思考

1、如何对 8255 的 PC 口进行位操作？

实验二 74HC138 译码器实验

一、实验目的与要求

- 1、掌握 74HC138 译码器的工作原理，熟悉 74HC138 译码器的具体运用连接方法，了解 74HC138 是如何译码的。
- 2、认真预习本节实验内容，尝试自行编写程序，填写实验报告

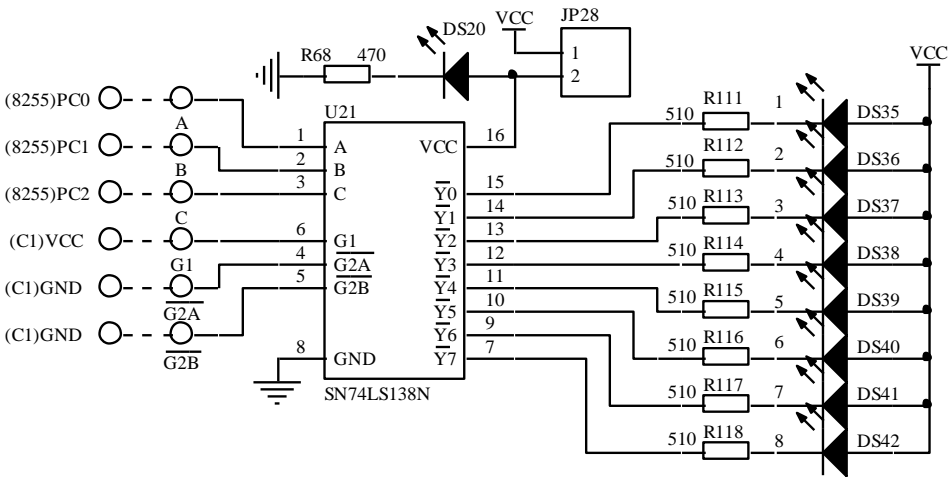
二、实验设备

STAR 系列实验仪一套、PC 机一台

三、实验内容

- 1、编写程序：使用 82C55 的 PC0、PC1、PC2 控制 74HC138 的数据输入端，通过译码产生 8 选 1 个选通信号，轮流点亮 8 个 LED 指示灯。
- 2、运行程序，验证译码的正确性。

四、实验原理图



五、实验步骤

- 1、连线说明：

C2 区：A、B、C	——	B4 区：PC0、PC1、PC2
C2 区：G1、G2A、G2B	——	C1 区：VCC、GND、GND
C2 区：JP36	——	F5 区：JP65 (LED 指示灯)
B4 区：CS(8255)、A0、A1	——	A3 区：CS1、A0、A1

- 2、调试程序，查看运行结果是否正确。

六、演示程序

```
.MODEL                TINY
Con_8255               EQU    0F003H           ;8255控制口
PC_8255                EQU    0F002H           ;8255 PC口
.STACK                100
.CODE
START:                MOV     DX, Con_8255
```

```

                                MOV        AL, 80H
                                OUT        DX, AL           ;8255初始化, PC口作输出用
                                MOV        DX, PC_8255
                                MOV        AL, 0
START1:                        OUT        DX, AL
                                CALL       Delay
                                INC        AL
                                JMP        START1
Delay                          PROC        NEAR           ;延时
Delay1:                        XOR        CX, CX
                                LOOP       $
                                RET
Delay                          ENDP

                                END        START

```

七、实验扩展及思考

在应用系统中，74HC138 通常用来产生片选信号，请读者考虑一下，应如何处理？

实验三 8155 输入、输出、SRAM 实验

一、实验目的与要求

了解 8155 的内部资源与结构；了解 8155 与 8088 的接口逻辑；熟悉对 8155 的初始化编程、输入和输出程序的设计方法、8155 定时器/计数器的使用方法。

认真预习，做好实验前的准备工作，填写实验报告

二、实验设备

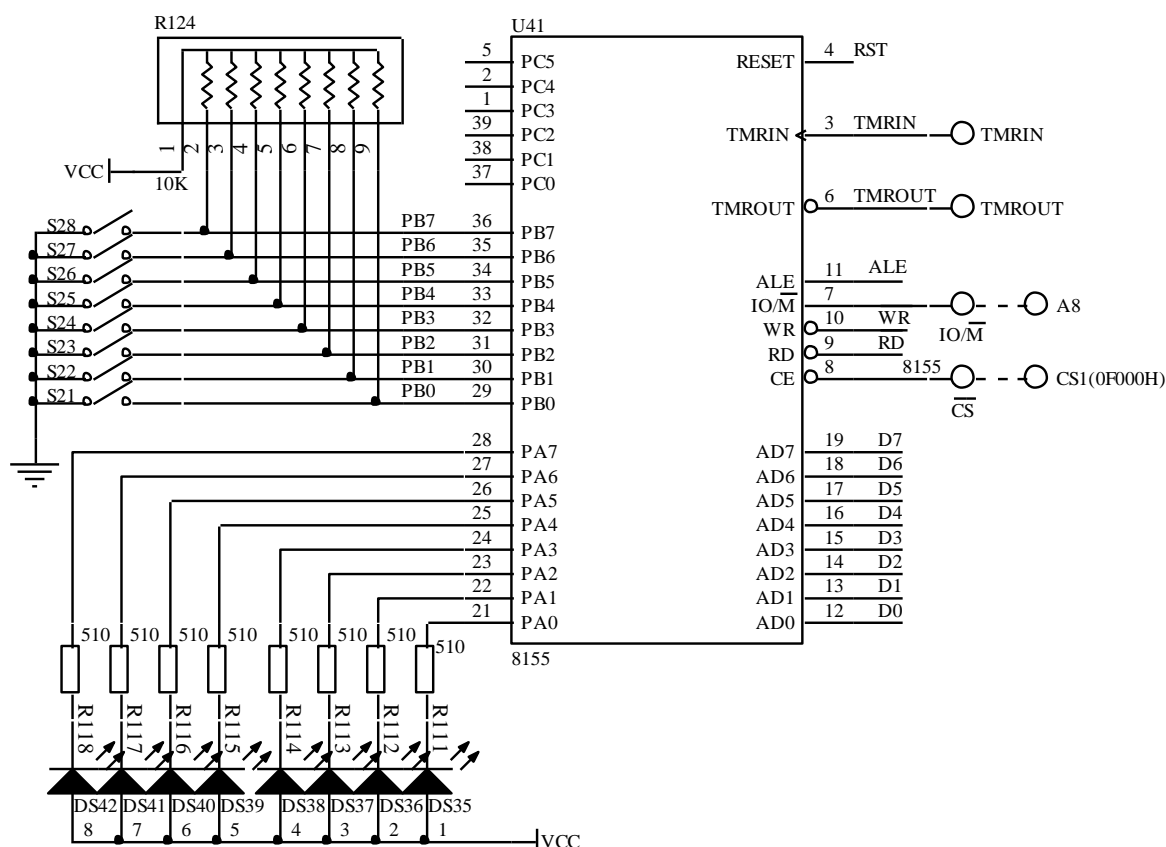
STAR 系列实验仪一套、PC 机一台

三、实验内容

1、编写程序：从 8155 的 PB 口将 F5 区的 8 位开关读入，写入 8155 的内部 RAM，再读出后，写入 PA 口，显示于 LED 指示灯上。

2、连接线路，验证 8155 的功能，熟悉它的使用方法。

四、实验原理图



五、实验步骤

1、连线说明：

B4 区：CS(8155)、IO/M	——	A3 区：CS1、A8
B4 区：JP76 (PA 口)	——	F5 区：JP65
B4 区：JP75 (PB 口)	——	F5 区：JP83

2、测试实验结果：F5 区的开关状态反应在 F5 区的 LED 指示灯上。

六、演示程序

```
.MODEL TINY
COM_8155 EQU 0F100H ;命令子/状态器寄存器
PA_8155 EQU 0F101H ;PA口地址
PB_8155 EQU 0F102H ;PB口地址
RAM_8155 EQU 0F000H ;8155内部RAM 00单元地址
.STACK 100
.CODE
STAR: MOV DX, COM_8155 ;控制口地址
      MOV AL, 1 ;PA为基本输出，PB为基本输入
      OUT DX, AL
STAR1: MOV DX, PB_8155 ;从PB口获得输入值（拨码盘输入）
      IN AL, DX
      MOV DX, RAM_8155 ;存入8155内部RAM里
      OUT DX, AL
      MOV AL, 0FFH ;消除输入结果
      IN AL, DX ;重新从8155相同地址取数
      MOV DX, PA_8155
      OUT DX, AL ;输出送显示（八个发光二极管）
      JMP STAR1
      END STAR
```

七、实验扩展及思考

- 1、例子程序中只展示了 8155 的输入输出和读写数据 RAM 的功能，8155 还有定时器/计数器的功能，有兴趣读者可以自己编写程序，通过 8155 来实现定时，当作定时器用时，如何接线？
- 2、若是要对 PC 口位操作，应该如何编写程序？
- 3、如何使用 8155 实现键盘扫描和 LED 显示？

实验四 8253 方波实验

一、实验目的与要求

了解 8253 的内部结构、工作原理；了解 8253 与 8088 的接口逻辑；熟悉 8253 的控制寄存器和初始化编程方法，熟悉 8253 的 6 种工作模式。

二、实验设备

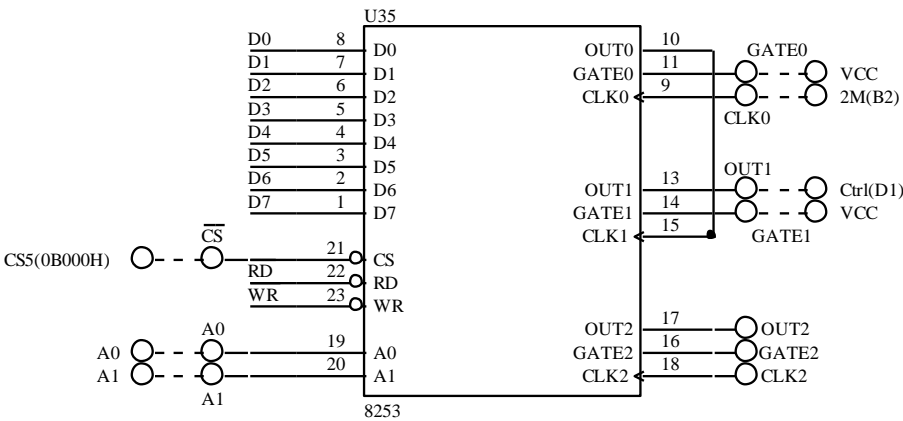
STAR 系列实验仪一套、PC 机一台

三、实验内容

1、编写程序：使用 8253 的计数器 0 和计数器 1 实现对输入时钟频率的两级分频，得到一个周期为 1 秒的方波，用此方波控制蜂鸣器，发出报警信号，也可以将输入脚接到逻辑笔上来检验程序是否正确。

2、连接线路，验证 8253 的功能，熟悉它的使用方法。

四、实验原理图



五、实验步骤

1、连线说明：

D5 区：CS、A0、A1	——	A3 区：CS5、A0、A1
D5 区：CLK0	——	B2 区：2M
D5 区：OUT0	——	D5 区：CLK1
D5 区：OUT1	——	E3 区：Ctrl(蜂鸣器)
D5 区：GATE0、GATE1	——	C1 区的 VCC

2、测试实验结果：蜂鸣器发出时有时无的声音；用逻辑笔测试蜂鸣器的输入端口，红绿灯交替点亮。

六、演示程序

```
.MODEL TINY
COM_ADDR EQU 0B003H
T0_ADDR EQU 0B000H
T1_ADDR EQU 0B001H
.STACK 100
```

```

        . CODE
START:   MOV     DX, COM_ADDR
        MOV     AL, 35H
        OUT     DX, AL           ;计数器T0设置在模式2状态, BCD码计数
        MOV     DX, TO_ADDR
        MOV     AL, 00H
        OUT     DX, AL
        MOV     AL, 10H
        OUT     DX, AL           ;CLK0/1000
        MOV     DX, COM_ADDR
        MOV     AL, 77H
        OUT     DX, AL           ;计数器T1为模式3状态, 输出方波, BCD码计数
        MOV     DX, T1_ADDR
        MOV     AL, 00H
        OUT     DX, AL
        MOV     AL, 10H
        OUT     DX, AL           ;CLK1/1000
        JMP     $                 ;OUT1输出1S的方波

        END     START

```

七、实验扩展及思考

- 1、8253 还有其它五种工作方式，其它工作模式下，硬件如何设计？程序如何编写？
- 2、使用 8253，编写一个实时钟程序。

实验五 8259A 中断控制器实验

一、实验目的与要求

了解 8259A 的内部结构、工作原理；了解 8259A 与 8088 的接口逻辑；掌握对 8259A 的初始化编程方法，了解 8088 是如何响应中断、退出中断的。

复习本节实验内容，可尝试自行编写程序，做好实验准备工作，填写实验报告。

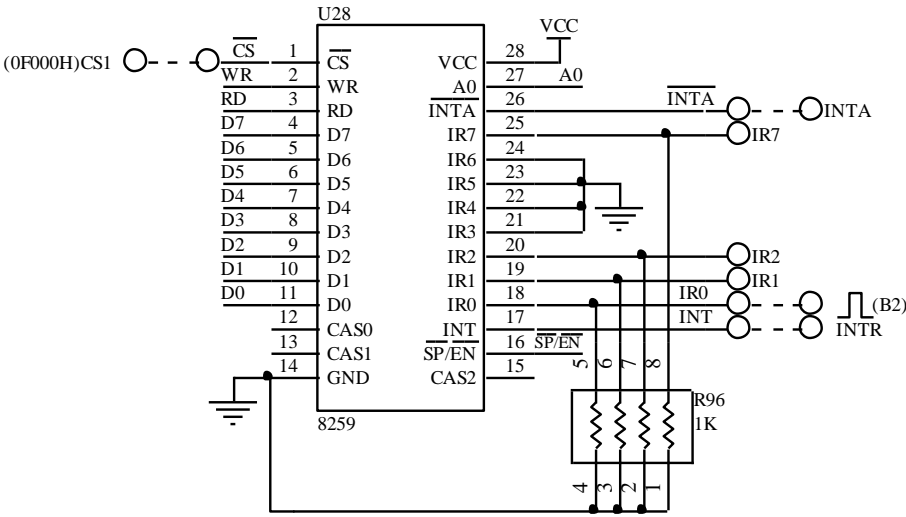
二、实验设备

STAR 系列实验仪一套、PC 机一台

三、实验内容

1、编制程序：拨动单脉冲开关，“ \square ”送给 8259A 的 IR0，触发中断，8088 计数中断次数，显示于 F4 区的数码管上

四、实验原理图



五、实验步骤

1、连线说明：

B3 区：CS、A0	——	A3 区：CS1、A0
B3 区：INT、INTA	——	EMU598+：INTR、INTA
B3 区：IR0	——	B2 区：单脉冲 \square
D3 区：CS、A0	——	A3 区：CS5、A0
D3 区：CLK	——	B2 区：2M

2、运行程序

3、上下拨动单脉冲开关，拨动二次，产生一个“ \square ”，观察结果，数码管上显示的次数与拨动开关次数是否对应。

六、演示程序

```
.MODEL            TINY
EXTRN             Display8:NEAR
I08259_0           EQU             0F000H
I08259_1           EQU             0F001H
. STACK           100
. DATA
```

BUFFER	DB	8 DUP(?)	
Counter	DB	?	
ReDisplayFlag	DB	0	
	. CODE		
START:	MOV	AX, @DATA	
	MOV	DS, AX	
	MOV	ES, AX	
	NOP		
	CALL	Init8259	
	CALL	WriIntver	
	MOV	Counter, 0	; 中断次数
	MOV	ReDisplayFlag, 1	; 需要显示
	STI		; 开中断
START1:	CMP	ReDisplayFlag, 0	
	JZ	START1	
	CALL	LedDisplay	
	MOV	ReDisplayFlag, 0	
	JMP	START1	
Init8259	PROC	NEAR	
	MOV	DX, I08259_0	
	MOV	AL, 13H	
	OUT	DX, AL	
	MOV	DX, I08259_1	
	MOV	AL, 08H	
	OUT	DX, AL	
	MOV	AL, 09H	
	OUT	DX, AL	
	MOV	AL, 0FEH	
	OUT	DX, AL	
	RET		
Init8259	ENDP		
WriIntver	PROC	NEAR	
	PUSH	ES	
	MOV	AX, 0	
	MOV	ES, AX	
	MOV	DI, 20H	
	LEA	AX, INT_0	
	STOSW		
	MOV	AX, CS	
	STOSW		
	POP	ES	
	RET		
WriIntver	ENDP		
LedDisplay	PROC	NEAR	

```

MOV        AL, Counter
MOV        AH, AL
AND        AL, 0FH
MOV        Buffer, AL
AND        AH, 0F0H
ROR        AH, 4
MOV        Buffer + 1, AH
MOV        Buffer + 2, 10H      ;高六位不需要显示
MOV        Buffer + 3, 10H
MOV        Buffer + 4, 10H
MOV        Buffer + 5, 10H
MOV        Buffer + 6, 10H
MOV        Buffer + 7, 10H
LEA        SI, Buffer
CALL       Display8
RET
LedDisplay ENDP
INT_0:     PUSH        DX
           PUSH        AX
           MOV         AL, Counter
           ADD         AL, 1
           DAA
           MOV         Counter, AL
           MOV         ReDisplayFlag, 1
           MOV         DX, I08259_0
           MOV         AL, 20H
           OUT         DX, AL
           POP         AX
           POP         DX
           IRET

           END         START

```

七、实验扩展及思考

- 1、从 8259A 收到上升沿，到 8088 响应中断，试画这个过程的时序图。

实验六 8250 可编程通信实验(与微机)

一、实验目的与要求

了解 8250 的内部结构、工作原理；了解 8250 与 8088 的接口逻辑；掌握对 8250 的初始化编程方法，学会使用 8250 实现设备之间的串行通信。

认真预习，做好实验前的准备工作，填写实验报告

二、实验设备

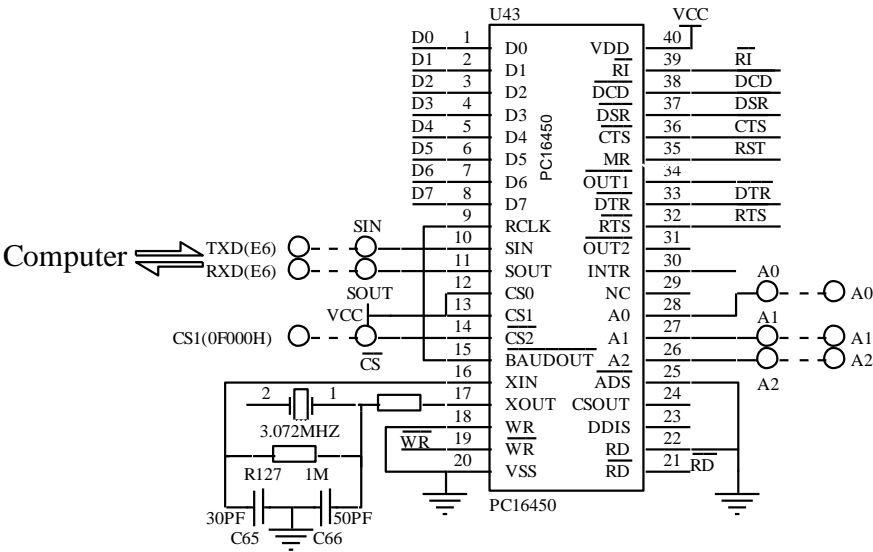
STAR 系列实验仪一套、PC 机一台

三、实验内容

1、编写程序：通过初始化 8250，设置波特率为 4800bps(或其它，但与微机部分一致)，数据格式为 8 数据位，1 停止位，偶校验；然后打开 PC 机的串行通信测试软件，向 8250 发送一批数据，8250 接收完数据之后，再将数据依次发送回去。

2、按图连线，运行程序，观察实验结果，掌握 8250 的各项功能及编程方法。

四、实验原理图



五、实验步骤

1、连线说明：

D4 区：CS、A0、A1、A2	——	A3 区：CS1、A0、A1、A2
D4 区：SIN、SOUT	——	E6 区：Rx D、Tx D

2、运行程序

3、运行“串口助手(ComPort.EXE)”，设置串口(波特率 4800, 8 个数据位，一个停止位，偶校验)，打开串口，选择“HEX 发送”、“HEX 显示”，向 8250 发送 10 个字节数据(输入数据之间用空格分隔)，是否能接收到 10 个字节数据，接收到的数据是否与发送数据一致。

4、改变传输数据的数目，重复实验，观察结果。

六、演示程序

.MODEL TINY

;8250和PC机通信，需要在PC上运行一个串口软件，并设置与8250相同的波特率

```

NS8250_Base_Address    EQU        0FF00H
RHR                     EQU        NS8250_Base_Address    ;接收数据缓冲区
THR                     EQU        NS8250_Base_Address    ;发送数据缓冲区
IER                     EQU        NS8250_Base_Address+1  ;中断控制寄存器
FCR                     EQU        NS8250_Base_Address+2  ;FIFO控制寄存器
ISR                     EQU        NS8250_Base_Address+2  ;中断状态寄存器
LCR                     EQU        NS8250_Base_Address+3  ;串行口控制寄存器
MCR                     EQU        NS8250_Base_Address+4  ;MODEM控制寄存器
LSR                     EQU        NS8250_Base_Address+5  ;串行口状态寄存器
MSR                     EQU        NS8250_Base_Address+6  ;MODEM状态寄存器
DLL                     EQU        NS8250_Base_Address    ;波特率除数锁存器低位
DLM                     EQU        NS8250_Base_Address+1  ;波特率除数锁存器高位

        . STACK      100
        . DATA
Receive_Buffer          DB          10 DUP(0)              ;接受缓冲器
Send_Buffer             EQU        Receive_Buffer          ;发送缓冲器

        . CODE
START:                  MOV         AX, @DATA
                        MOV         DS, AX
                        MOV         ES, AX
                        NOP
                        CALL        INIT8250
START2:                 MOV         CX, 10                  ;接收数据(接收完设定的数据个数)
                        CALL        RECEIVE_GROUP
                        MOV         CX, 10                  ;发送数据(发完设定的数据个数)
                        CALL        SEND_GROUP
                        JNC         START2
WARNING1:               JMP         $

;*****发送一组字符子程序，个数在CX中*****
Send_Group              PROC        NEAR
                        LEA         SI, Send_Buffer
Send_Group1:            LODSB
                        CALL        Send_Byte
                        JC          Send_Group2
                        LOOP        Send_Group1
                        CLC
Send_Group2:            RET
Send_Group              ENDP

;*****接收一组字符子程序，存放首地址在DPTR中，个数在R6R7中*****
Receive_Group           PROC        NEAR
                        LEA         DI, Receive_Buffer
Receive_Group1:         CALL        Receive_Byte
                        STOSB

```

```

                                LOOP                Receive_Group1
                                CLC
                                RET
Receive_Group1 ENDP
INIT8250 PROC                NEAR
                                MOV                DX, ISR
                                MOV                AL, 06H
                                OUT                DX, AL
                                MOV                DX, LCR
                                MOV                AL, 83H                ;允许访问波特率因子寄存器
                                OUT                DX, AL
                                MOV                DX, DLL
                                MOV                AL, 40
                                ;除数低位寄存器, 波特率设为4800=(3.072*1000000/16)/DLMDLL
                                OUT                DX, AL
                                MOV                DX, DLM                ;00H送高字节寄存器
                                MOV                AL, 00H
                                OUT                DX, AL
                                MOV                DX, LCR                ;不允许访问波特率因子寄存器
                                MOV                AL, 1BH                ;数据格式为8数据位, 1停止位, 偶校验
                                OUT                DX, AL
                                RET
INIT8250 ENDP
;*****发送一个字节子程序, 发送A中的数, 失败置CY*****
Send_Byte PROC                NEAR
                                PUSH                CX
                                PUSH                AX
                                MOV                CX, 1000
                                MOV                DX, LSR
REP11:                        IN                AL, DX
                                TEST                AL, 20H
                                JNZ                OUTPORT1
                                LOOP                REP11
                                POP                AX
                                STC
                                JMP                EXIT8250
OUTPORT1:                    POP                AX
                                MOV                DX, RHR
                                OUT                DX, AL
                                CLC
EXIT8250:                    POP                CX
                                RET
Send_Byte ENDP
;*****接收一个字节子程序, 接收字节在A中, 接收失败置1CY*****

```

Receive_Byte	PROC	NEAR
	MOV	DX, LSR
Receive1:	IN	AL, DX
	TEST	AL, 1
	JZ	Receive1
Receive2:	MOV	DX, RHR
	IN	AL, DX
Receive3:	RET	
Receive_Byte	ENDP	
	END	START

七、实验扩展及思考

- 1、思考 8250 与 8251 有何异同之处？
- 2、8250 也可以做自发自收的实验，该如何连线及修改程序？
- 3、如何通过中断处理方式实现 8250 串行接收, 需要更改哪些线路？

实验七 8279 键盘显示实验

一、实验目的与要求

了解 8279 的内部结构、工作原理；了解 8279 与 8088 的接口逻辑；掌握对 8279 的编程方法，掌握使用 8279 扩展键盘、显示器的方法。

认真预习，做好实验前的准备工作，自行编写程序，填写实验报告

二、实验设备

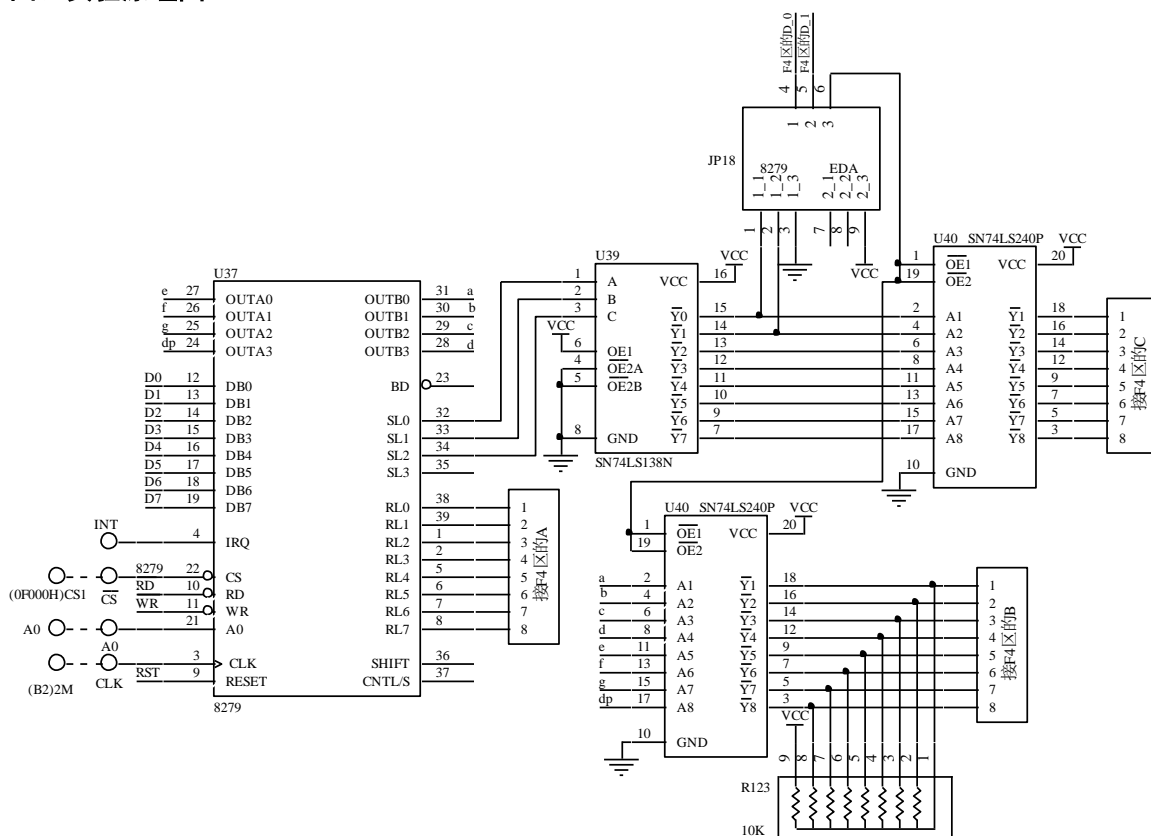
STAR 系列实验仪一套、PC 机一台

三、实验内容

1、编写程序：利用 8279 实现对 F4 区的键盘扫描，将键号显示于 8 位数码管上

2、按图连线，运行程序，观察实验结果，能熟练运用 8279 扩展显示器和键盘。

四、实验原理图



五、实验步骤

1、连线说明：

D3 [X]: CS、A0	——	A3 [X]: CS5、A0
D3 [X]: CLK	——	B2 [X]: 2M

2、运行程序，观察实验结果（任意按下 F4 区 4X4 键盘几个键，它上面的 8 个 LED 显示器会将按键的编码从左至右依次显示出来），可依此验证对 8279 芯片操作的正确性。

六、演示程序

```

.MODEL TINY
CMD_8279 EQU 0BF01H ;8279命令字、状态字地址
DATA_8279 EQU 0BF00H ;8279读写数据口的地址

.STACK 100
.DATA

KEYCOUNT DB ?
LED_TAB DB 0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0F8H
DB 080H, 90H, 88H, 83H, 0C6H, 0A1H, 86H, 8EH

.CODE

START: MOV AX, @DATA
MOV DS, AX
NOP
CALL INIT8279 ;初始化子程序
MOV KEYCOUNT, 0

START1: CALL SCAN_KEY ;键扫描
JNC START1 ;没有按键
XCHG AL, KEYCOUNT
INC AL
CMP AL, 9
JNZ START2
MOV KEYCOUNT, 0
CALL INIT8279_1; 8个数码块全有字符显示后, 再按键, 清除显示
JMP START1

START2: XCHG AL, KEYCOUNT
CALL KEY_NUM ;键值转换为键号
LEA BX, LED_TAB ;字型码表
XLAT
CALL WRITE_DATA
JMP START1

START_EXIT: JMP $
;8279初始化

INIT8279 PROC NEAR
MOV DX, CMD_8279 ;CMD_8279为写命令地址、读状地址
MOV AL, 34H ;可编程时钟设置, 设置分频系数(20分频)
OUT DX, AL
MOV AL, 0 ;8*8字符显示, 左边输入, 外部译码键扫描方式
OUT DX, AL
; MOV AL, 0A0H
; OUT DX, AL
CALL INIT8279_1
RET
INIT8279 ENDP
INIT8279_1 PROC NEAR

```

```

CALL CLEAR ;清显示
MOV AL, 90H ;从第一个数码管开始移位显示
OUT DX, AL
RET
INIT8279_1 ENDP
CLEAR PROC NEAR
MOV DX, CMD_8279
MOV AL, 0DEH ; 清除命令
OUT DX, AL
WAIT1: IN AL, DX
TEST AL, 80H
JNZ WAIT1 ; 显示RAM清除完毕吗?
RET
CLEAR ENDP
SCAN_KEY PROC NEAR
MOV DX, CMD_8279
IN AL, DX ;读状态
READ_FIFO: AND AL, 7
JZ NO_KEY ;是否有键按下
READ: MOV AL, 40H
OUT DX, AL ;读FIFO RAM
MOV DX, DATA_8279
IN AL, DX
STC ;有键
SCAN_KEY1: RET
NO_KEY: CLC ;无键按下，清CY
JMP SCAN_KEY1
SCAN_KEY ENDP
KEY_NUM PROC NEAR
AND AL, 3FH
RET
KEY_NUM ENDP
WRITE_DATA PROC NEAR
MOV DX, DATA_8279
OUT DX, AL
RET
WRITE_DATA ENDP

END START

```

七、实验扩展及思考

重新编写软件实验二，自己编写键扫描、显示程序

实验八 并行 DA 实验

一、实验目的

了解数模转换的原理；了解 0832 与 8088 的接口逻辑，掌握使用 DAC0832 进行数模转换。

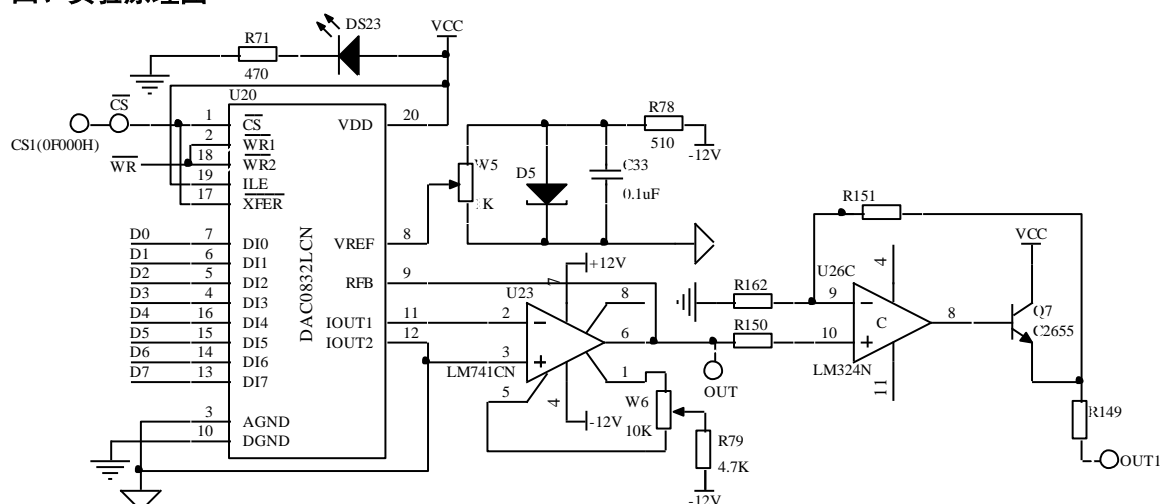
二、实验设备

STAR 系列实验仪一套、PC 机一台、示波器一台。

三、实验内容

- 1、编写程序：用 0832 输出正弦波
- 2、按图连线，运行程序，使用示波器观察实验结果。

四、实验原理图



五、实验步骤

- 1、连线说明：

E2 区：CS	——	A3 区：CS1
---------	----	----------

- 2、运行程序，示波器的探头接 E2 区的 OUT，观察实验结果，是否产生正弦波。

六、演示程序

```

.MODEL          TINY
ADDR_0832      EQU          0FF00H          ;0832输出口地址
.STACK         100
.DATA
TAB_1          DB          7FH, 8BH, 96H, 0A1H, 0ABH, 0B6H, 0C0H, 0C9H, 0D2H
               DB          0DAH, 0E2H, 0E8H, 0EEH, 0F4H, 0F8H, 0FBH, 0FEH, 0FFH, 0FFH
               DB          0FFH, 0FEH, 0FBH, 0F8H, 0F4H, 0EEH, 0E8H, 0E2H, 0DAH, 0D2H
               DB          0C9H, 0C0H, 0B6H, 0ABH, 0A1H, 096H, 08BH, 07FH
               DB          74H, 69H, 5EH, 54H, 49H, 40H, 36H, 2DH, 25H, 1DH, 17H, 11H, 0BH, 7, 4, 2, 0, 0
               DB          0, 2, 4, 7, 0BH, 11H, 17H, 1DH, 25H, 2DH, 36H, 40H, 49H, 54H, 5EH, 69H, 74H
.CODE
START:         MOV          AX, @DATA
    
```

	MOV	DS, AX
	NOP	
	MOV	DX, ADDR_0832
START1:	LEA	SI, TAB_1
	MOV	CX, 72
START2:	LODSB	
	OUT	DX, AL
	CALL	DELAY
	LOOP	START2
	JMP	START1
DELAY	PROC	NEAR
	PUSH	CX
	MOV	CX, 30
	LOOP	\$
	POP	CX
	RET	
DELAY	ENDP	
	END	START

实验九 并行 AD 实验(数字电压表实验)

一、实验目的与要求

- 1、了解几种类型 AD 转换的原理；掌握使用 ADC0809 进行模数转换
- 2、认真预习实验内容，做好准备工作，完成实验报告。

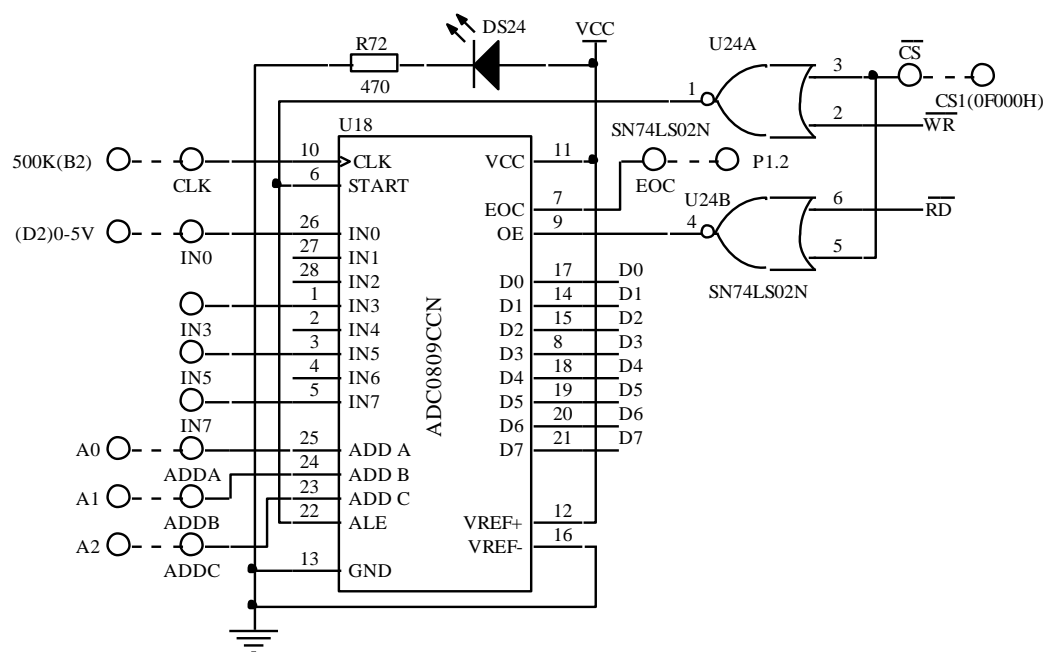
二、实验设备

STAR 系列实验仪一套、PC 机一台、万用表一个。

三、实验内容

- 1、ADC0809 (F3 区)
 - (1) 模数转换器，8 位精度，8 路转换通道，1 路并行输出
 - (2) 转换时间 100us，转换电压范围 0~5V
- 2、编写程序：制作一个电压表，测量 0~5V，结果显示于数码管上。

四、实验原理图



五、实验步骤

- 1、连线说明：

F3 区：CS、ADDA、ADDB、ADDC	——	A3 区：CS1、A0、A1、A2（选择通道）
F3 区：CLK	——	B2 区：500K
F3 区：IN0	——	F2 区：0~5V
D3 区：CLK	——	B2 区：2M
D3 区：CS	——	A3 区：CS5
D3 区：A0	——	A3 区：A0

- 2、调节 0~5V 电位器（F2 区）输出电压，显示在 LED(最右边 2 位)上的电压数字量会随之改变。用万用表验证 AD 转换的结果。

六、演示程序

```

.MODEL TINY
ADDR_0809 EQU 0F000H
EXTRN Display8:NEAR
.STACK 100
.DATA
BUFFER DB 8 DUP(?)
LastAD DB 0 ;上一次AD转换值
.CODE
START: MOV AX, @DATA
MOV DS, AX
NOP
XOR AL, AL
JMP START6
START1: MOV CX, 50 ;采样五十次
MOV BX, 0 ;累计五十次的采样值
START2: CALL AD0809
XOR AH, AH
ADD BX, AX
LOOP START2
MOV AX, 50
XCHG AX, BX
DIV BL ;五十次的平均值
CMP AL, LastAD
JZ START3
START6: MOV LastAD, AL
CALL Display_Data
LEA SI, BUFFER
CALL Display8
START3: CALL DLTime
JMP START1
AD0809 PROC NEAR
PUSH CX
MOV AL, 0
MOV DX, ADDR_0809
OUT DX, AL
MOV CX, 200
LOOP $ ;延时, 等待AD转换完成
MOV DX, ADDR_0809
IN AL, DX
POP CX
RET
AD0809 ENDP
DISPLAY_DATA PROC NEAR
MOV AH, AL

```

```

AND        AL, 0FH
MOV        BUFFER + 4, AL
MOV        AL, AH
AND        AL, 0F0H
ROR        AL, 4
MOV        BUFFER + 5, AL
MOV        AL, AH
XOR        AH, AH
MOV        BL, 51                ;255/51 (16进制的1 = 1/51V)
DIV        BL
OR         AL, 80H                ;加上小数点
MOV        BUFFER + 2, AL
MOV        AL, 10
MUL        AH
DIV        BL
MOV        BUFFER + 1, AL        ;第一位小数
MOV        AL, 10
MUL        AH
DIV        BL
MOV        BUFFER, AL            ;第二位小数
MOV        buffer+3, 10H
MOV        buffer+6, 10H
MOV        buffer+7, 10H        ;消隐
RET
DISPLAY_DATA ENDP
DLTime      PROC NEAR
MOV        CX, 30000
LOOP       $
RET
DLTime      ENDP

END        START

```

七、实验扩展及思考

如何实现多路模拟量的数据采集、显示？

实验十 红外通信实验

一、实验目的

1、理解红外通讯原理；2、掌握红外通讯；3、熟练使用 8250

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

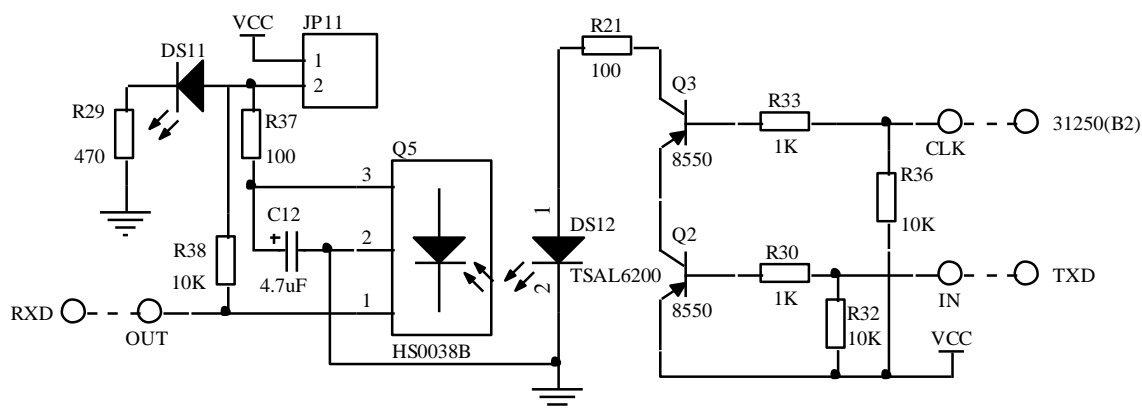
1、红外通讯原理

当红外接收器收到 38kHz 频率的信号，输出电平会由 1→0，一旦没有此频率信号，输出电平会由 0→1。因此，红外发射头控制通断发射 38kHz 信号，就可以将数据发送出来

2、实验过程

- (1) 8250 使用红外发送管和接收器进行数据自发自收
- (2) 根据接收到的数据，通过 8255 的 PA 口点亮 8 个发光管，会看到发光管不断变化

四、实验原理图



五、实验步骤

1、连线说明：

F1 区：IN、OUT	——	D4 区：SOUT、SIN
F1 区：CLK	——	B2 区：31250
D4 区：CS、A0、A1、A2	——	A3 区：CS1、A0、A1、A2
B4 区：CS(8255)、A0、A1	——	A3 区：CS2、A0、A1
B4 区：JP56 (PA)	——	F5 区：JP65

2、调试该程序时，使用较厚的白纸挡住红外发射管红外信号，使它反射到接收头。

说明：一般红外接收模块的解调频率为 38KHz，当它接收到 38KHz 左右的红外信号时将输出低电平，但连续输出低电平的时间是有限制的（如：100ms），也就是说输出低电平的宽度是有限制的。

3、发送数据，并接收，根据接收到的数据点亮 8 个发光管，程序运行之后，会看到 8 个发光管(F5 区)在闪烁，从第 8 个(最右边)向第 1 个逐一点亮过去。本实验通过红外通讯发送、接收数据，发送的数据从 00H 开始+1，接收到该数据后用来点亮 8 个发光管。亮-1，熄-0。

六、演示程序

```

.MODEL TINY
;8250和PC机通信，需要在PC上运行一个串口软件，并设置与8250相同的波特率
NS8250_Base_Address EQU 0FF00H
RHR EQU NS8250_Base_Address ;接收数据缓冲区
THR EQU NS8250_Base_Address ;发送数据缓冲区
IER EQU NS8250_Base_Address+1 ;中断控制寄存器
FCR EQU NS8250_Base_Address+2 ;FIFO控制寄存器
ISR EQU NS8250_Base_Address+2 ;中断状态寄存器
LCR EQU NS8250_Base_Address+3 ;串行口控制寄存器
MCR EQU NS8250_Base_Address+4 ;MODEM控制寄存器
LSR EQU NS8250_Base_Address+5 ;串行口状态寄存器
MSR EQU NS8250_Base_Address+6 ;MODEM状态寄存器
DLL EQU NS8250_Base_Address ;波特率除数锁存器低位
DLM EQU NS8250_Base_Address+1 ;波特率除数锁存器高位
COM_8255 EQU 0E003H ;8255控制口
PA_8255 EQU 0E000H ;8255PA口
.STACK 100
.DATA
Receive_Buffer DB 10 DUP(0) ;接受缓冲器
Send_Buffer EQU Receive_Buffer ;发送缓冲器
.CODE
START: MOV AX, @DATA
MOV DS, AX
MOV ES, AX
NOP
MOV DX, COM_8255
MOV AL, 80H
OUT DX, AL ;PA、PB、PC为基本输出模式
INC DX
MOV AL, 0FFH
OUT DX, AL ;熄灭指示灯
CALL INIT8250
MOV BX, 0
START1: CALL Infrared_Test ;调用自收自发红外通讯子程序
CALL DelayTime ;延时
JMP START1 ;循环进行红外通讯
INIT8250 PROC NEAR
MOV DX, ISR
MOV AL, 06H
OUT DX, AL
MOV DX, LCR
MOV AL, 83H ;允许访问波特率因子寄存器
OUT DX, AL

```

```

MOV        DX, DLL
MOV        AL, 80        ;波特率为2400=(3.072*1000000/16)/DLMDLL
OUT        DX, AL
MOV        DX, DLM        ;00H送高字节寄存器
MOV        AL, 00H
OUT        DX, AL
MOV        DX, LCR        ;不允许访问波特率因子寄存器
MOV        AL, 1BH        ;数据格式为8数据位, 1停止位, 偶校验
OUT        DX, AL
RET
INIT8250    ENDP
;*****发送一个字节子程序, 发送A中的数, 失败置1CY*****
Send_Byte    PROC        NEAR
PUSH        AX
MOV        CX, 1000
MOV        DX, LSR
REP11:      IN            AL, DX
TEST       AL, 20H
JNZ        OUTPORT1
LOOP       REP11
POP        AX
STC
JMP        EXIT8250
OUTPORT1:   POP        AX
MOV        DX, RHR
OUT        DX, AL
CLC
EXIT8250:   RET
Send_Byte    ENDP
;*****接收一个字节子程序, 接收字节在A中, 接收失败置1CY*****
Receive_Byte    PROC        NEAR
MOV        CX, 1000
MOV        DX, LSR
Receive1:    IN            AL, DX
TEST       AL, 1
JNZ        Receive2
LOOP       Receive1
STC
JMP        Receive3
Receive2:    MOV        DX, RHR
IN            AL, DX
CLC
Receive3:    RET
Receive_Byte    ENDP

```

```

;红外通讯
Infrared_Test    PROC        NEAR
                  MOV        AL, BL
                  CALL        Send_Byte
                  JNB         Infrared_Test1
                  MOV        AL, 0FFH          ;无法发送
                  JMP         Infrared_Test2
Infrared_Test1:  CALL        Receive_Byte
                  JNB         Infrared_Test2
                  MOV        AL, 0            ;无法接收

Infrared_Test2:  CALL        Light            ;根据接收到的数据点亮8个红色发光管
                  INC        BX              ;发送数据逐步递增
                  RET
Infrared_Test    ENDP
;点亮8个发光管
Light            PROC        NEAR
                  NOT        AL              ;0-亮, 1-灭
                  MOV        DX, PA_8255
                  OUT        DX, AL
                  RET
Light            ENDP
;延时程序
DelayTime        PROC        NEAR
                  MOV        CX, 60000      ;延时0.5s
                  LOOP       $
                  RET
DelayTime        ENDP

                  END        START

```

七、实验扩展及思考

实验名称：红外遥控器实验

实验目的：了解日常所用的家电红外遥控器是如何工作的

实验内容：结合按键模拟 4 路红外遥控器，遥控发光管或电机转动快慢

实验十一 字符型液晶显示实验(1602C)

一、实验目的与要求

了解字符型液晶模块的控制方法；了解它与 8088 的接口逻辑。

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

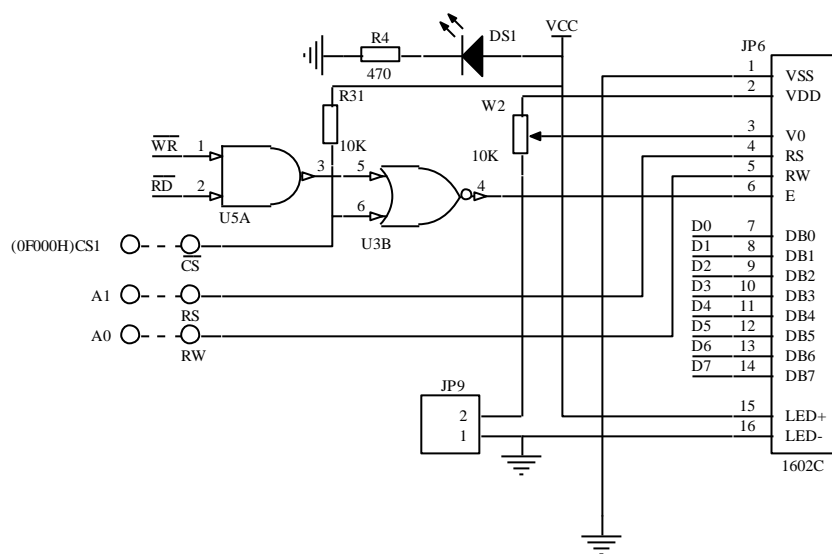
1、1602C 液晶显示器

- (1) 字符型液晶显示器，可以显示二行，每行最多 16 个字符
- (2) 采用 8 位数据总线并行输入输出和 3 条控制线。
- (3) 指令简单，7 种指令

2、实验过程

在 1602C 液晶上，第一行显示“STAR ES598PCIS”，第二行滚动显示“Shanghai Xingyan Electronics Co.,LTD.”。

四、实验原理图



五、实验步骤

1、主机连线说明：

A1 区：CS、RW、RS、CS1/2	——	A3 区：CS1、A0、A1、A2
---------------------	----	-------------------

2、运行程序，验证显示结果。

六、演示程序

```
.MODEL TINY
LCD_Line_Length EQU 16
Y12864_W_CON EQU 0F000H ;写指令地址
Y12864_R_CON EQU 0F001H ;读取忙状态地址
Y12864_W_Data EQU 0F002H ;写数据地址
```



```

Y12864_R_Data    EQU        0F003H                ;读数据地址
                  .STACK    100
                  .DATA
Tab1:             DB ' STAR ES598PCIS ', 0
; 上海星研电子科技有限公司
Tab2:             DB ' Shanghai Xingyan Electronics Co.,LTD. '
                  DB ' ', 0                        ;以 00H 结束
                  .CODE
START:            MOV        AX, @DATA
                  MOV        DS, AX
                  NOP
                  CALL       InitLCD                ;液晶显示初始化
                  MOV        AL, 0
                  CALL       Clear_Line             ;清第一行
                  MOV        AL, 80H
                  LEA        SI, Tab1
                  CALL       Disp_LineDP            ;第一行：显示实验仪型号
MAIN1:            CALL       DIS_2                  ;第二行：显示公司名称（动态显示）
                  JMP        MAIN1
DIS_2             PROC       NEAR
                  MOV        AL, 1
                  CALL       Clear_Line
                  MOV        BX, 0                  ;从哪一个字符开始显示
                  MOV        DI, 0C0H+LCD_Line_Length - 1 ;从最右边第一位开始显示
DIS_21:           MOV        AX, DI                  ;光标定位
                  CALL       WR_Con
                  MOV        CX, 0C0H+LCD_Line_Length
                  CLC
                  SUB        CX, DI                  ;每次输出字符计数
                  LEA        SI, Tab2
                  ADD        SI, BX
DIS_22:           LODSB
                  OR         AL, AL
                  JZ         DIS_24                  ;0 表示结束
                  CALL       WR_Data                ;输出一个字符
                  LOOP        DIS_22
                  CALL       DelayTime_Move
                  MOV        AX, DI
                  CMP        AX, 0C0H
                  JNZ        DIS_23                  ;第一个字符是否已经移位到最右边
                  INC        BX                      ;是，修改偏移量
                  JMP        DIS_21
DIS_23:           DEC        DI
                  JMP        DIS_21

```

```

DIS_24:      RET
DIS_2        ENDP
DelayTime_Move  PROC      NEAR      ;延时子程序
MOV          CX, 2
DelayTime_Move1:  PUSH     CX
MOV          CX, 0FFFFH
LOOP         $
POP          CX
LOOP         DelayTime_Move1
RET
DelayTime_Move  ENDP
InitLCD        PROC      NEAR      ;1602 液晶初始化
CALL         DELAY1      ;延时至少 15ms
MOV          AL, 38H
CALL         WrconNoBusy  ;写指令，查询结果
CALL         DELAY2      ;延时至少 5ms
MOV          AL, 38H
CALL         WrconNoBusy
CALL         DELAY2
MOV          AL, 38H      ;5*7 点阵，8 位 CPU 接口
CALL         WR_Con      ;写指令，并等待写指令被成功接受才返回
MOV          AL, 38H
CALL         WR_Con
MOV          AL, 09H
CALL         WR_Con
MOV          AL, 01H      ;清屏，置 AC 为 0
CALL         WR_Con
MOV          AL, 06H      ;设定光标正向移动，且整显示不移动
CALL         WR_Con
MOV          AL, 0CH      ;设定为整体显示，光标不显示
CALL         WR_Con
RET
InitLCD        ENDP
DELAY1        PROC      NEAR
MOV          CX, 3000H
LOOP         $
RET
DELAY1        ENDP
DELAY2        PROC      NEAR
MOV          CX, 1000H
LOOP         $
RET
DELAY2        ENDP
WrconNoBusy    PROC      NEAR      ;写指令子程序，不 查询忙标志

```

```

MOV      DX, Y12864_W_CON      ;写控制命令
OUT      DX, AL
RET
WrconNoBusy ENDP
WR_Con PROC NEAR              ;写指令子程序
MOV      DX, Y12864_W_CON      ;写控制命令
OUT      DX, AL
NOP
MOV      DX, Y12864_R_CON      ;读忙状态
WR_Con1: IN      AL, DX
TEST     AL, 80H
JNZ      WR_Con1              ;检查液晶显示是否处于忙状态
RET
WR_Con ENDP
WR_Data PROC NEAR              ;写数据子程序
MOV      DX, Y12864_W_Data
OUT      DX, AL
MOV      DX, Y12864_R_CON
WR_Data1: IN      AL, DX
TEST     AL, 80H
JNZ      WR_Data1            ;检查液晶显示是否处于忙状态
RET
WR_Data ENDP
Clear_LCD PROC NEAR            ;清屏
MOV      AL, 01H
CALL     WR_Con
RET
Clear_LCD ENDP
Close_Cursor PROC NEAR        ;关光标
MOV      AL, 0CH
CALL     WR_Con
RET
Close_Cursor ENDP
Clear_Line PROC NEAR          ;清行  AL:哪一行
CMP      AL, 0                ;入口参数在 AL 里, AL=0, 清零第一行
JZ       Clear_Line1          ;清除某一行的显示内容
CMP      AL, 1
JNZ      Clear_Line2
MOV      AL, 0C0H              ;AL = 1, 清零第二行
JMP      Clear_Line3
Clear_Line1: MOV      AL, 80H
JMP      Clear_Line3
Clear_Line2: CMP      AL, 2
JNZ      Clear_Line4

```

```

MOV      AL, 94H                ;AL=2, 清零第三行
JMP      Clear_Line3
Clear_Line4: MOV      AL, 0D4H    ;AL=3, 清零第四行
Clear_Line3: CALL     WR_Con
MOV      CX, LCD_Line_Length
Clear_Line5: MOV      AL, 20H
CALL     WR_Data
LOOP     Clear_Line5
RET
Clear_Line ENDP
Set_CgramCursor PROC      NEAR    ;设置光标  AL--光标位置
AND      AL, 7FH                ;此为对于用户自行设计的图形而言
OR       AL, 40H
CALL     WR_Con
RET
Set_CgramCursor ENDP
;设置光标 AL--光标位置: (A)=00H~13H, 光标在第一行; (A)=40H~53H, 光标在第二行;
(A)=14H~27H, 光标在第三行; (A)=54H~67H, 光标在第四行
Set_DdramCursor PROC      NEAR
OR       AL, 80H
CALL     WR_Con
RET
Set_DdramCursor ENDP
;从位置(AL)开始显示 SI 指向的一行, 0 为结束标志
Disp_LineDP PROC      NEAR
CALL     WR_Con                ;定位, 第一个数据显示的位置
Disp_LineDP1: LODSB
OR       AL, AL
JZ       Disp_LineDP2          ;判断是否到了显示结束标志
CALL     WR_Data
JMP      Disp_LineDP1
Disp_LineDP2: RET
Disp_LineDP ENDP

END      START

```

七、实验扩展及思考

实验内容：显示一幅图画，进一步熟练液晶显示的操作。

实验十二 图形点阵显示实验

一、实验目的与要求

了解图形液晶模块的控制方法；了解它与 8088 的接口逻辑；掌握使用图形点阵液晶显示字体和图形。

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

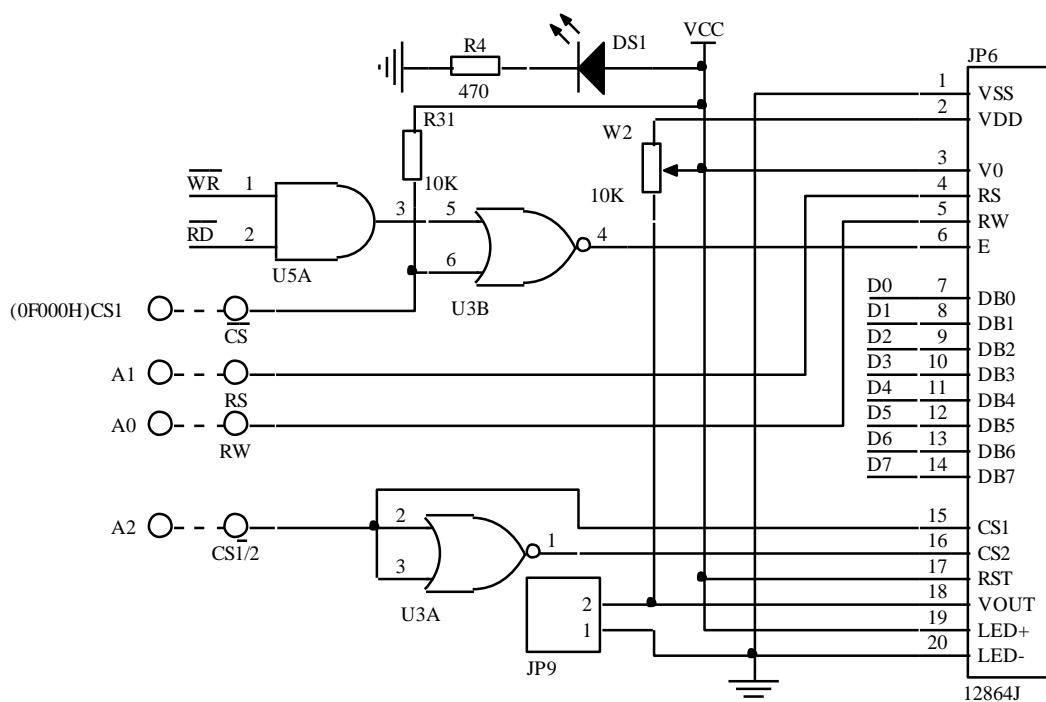
1、12864J 液晶显示器

- (1) 图形点阵液晶显示器，分辨率为 128X64。可显示图形和 8×4 个(16×16 点阵)汉字。
- (2) 采用 8 位数据总线并行输入输出和 4 条控制线。
- (3) 指令简单，7 种指令

2、实验过程

在 12864J 液晶上显示一段字，包括汉字和英文：“星研电子”、“STAR ES51PRO”、“欢迎使用”，三行字。

四、实验原理图



五、实验步骤

1、主机连线说明：

A1 区：CS、RW、RS、CS1/2	——	A3 区：CS1、A0、A1、A2
---------------------	----	-------------------

2、运行程序，验证显示结果。

六、演示程序

```

.MODEL TINY
WR_COM_AD_L EQU 0F004H ;写左半屏指令地址
WR_COM_AD_R EQU 0F000H ;写右半屏指令地址
WR_DATA_AD_L EQU 0F006H ;写左半屏数据地址
WR_DATA_AD_R EQU 0F002H ;写右半屏数据地址
RD_BUSY_AD EQU 0F001H ;查忙地址
RD_DATA_AD EQU 0F003H ;读数据地址
X EQU 0B8H ;起始显示行基址
Y EQU 040H ;起始显示列基址
FirstLine EQU 0C0H ;起始显示行

.STACK 100
.DATA
;-- 文字: 星 --
Line1_1 DB 00H, 00H, 0FCH, 82H, 82H, 0AAH, 2AH, 0AAH, 0AAH, 0AAH, 02AH, 02H, 02H, 0FCH, 00H, 00H
          DB 00H, 0EEH, 9BH, 90H, 98H, 94H, 95H, 80H, 80H, 80H, 95H, 95H, 95H, 95H, 0FFH, 00H
;-- 文字: 研 --
Line1_2 DB 9EH, 62H, 02H, 02H, 02H, 32H, 0FEH, 62H, 02H, 02H, 32H, 02H, 02H, 02H, 62H, 0DCH
          DB 03H, 3CH, 40H, 40H, 46H, 40H, 0F1H, 8EH, 80H, 40H, 7CH, 80H, 80H, 80H, 0FEH, 03H
;-- 文字: 电 --
Line1_3 DB 00H, 0F8H, 04H, 04H, 44H, 44H, 06H, 02H, 02H, 46H, 44H, 04H, 04H, 0F8H, 00H, 00H
          DB 00H, 0FH, 10H, 10H, 11H, 11H, 0F0H, 80H, 90H, 91H, 91H, 8CH, 84H, 87H, 0C8H, 78H
;-- 文字: 子 --
Line1_4 DB 80H, 40H, 5EH, 52H, 52H, 52H, 32H, 72H, 82H, 82H, 42H, 62H, 52H, 4CH, 0C0H, 00H
          DB 07H, 04H, 04H, 04H, 0FCH, 8CH, 8CH, 80H, 80H, 7CH, 04H, 04H, 04H, 04H, 07H, 00H
;" STARES51PRO"
Line2_1 DB 00H, 70H, 88H, 08H, 08H, 08H, 38H, 00H, 00H, 38H, 20H, 21H, 21H, 22H, 1CH, 00H
Line2_2 DB 18H, 08H, 08H, 0F8H, 08H, 08H, 18H, 00H, 00H, 00H, 20H, 3FH, 20H, 00H, 00H, 00H
Line2_3 DB 00H, 00H, 0C0H, 38H, 0E0H, 00H, 00H, 00H, 20H, 3CH, 23H, 02H, 02H, 27H, 38H, 20H
Line2_4 DB 08H, 0F8H, 88H, 88H, 88H, 88H, 70H, 00H, 20H, 3FH, 20H, 00H, 03H, 0CH, 30H, 20H
Line2_5 DB 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H
Line2_6 DB 08H, 0F8H, 88H, 88H, 0E8H, 08H, 10H, 00H, 20H, 3FH, 20H, 20H, 23H, 20H, 18H, 00H
Line2_7 DB 00H, 70H, 88H, 08H, 08H, 08H, 38H, 00H, 00H, 38H, 20H, 21H, 21H, 22H, 1CH, 00H
Line2_8 DB 00H, 0F8H, 08H, 88H, 88H, 08H, 08H, 00H, 00H, 19H, 21H, 20H, 20H, 11H, 0EH, 00H
Line2_9 DB 00H, 10H, 10H, 0F8H, 00H, 00H, 00H, 00H, 00H, 20H, 20H, 3FH, 20H, 20H, 00H, 00H
Line2_10 DB 08H, 0F8H, 08H, 08H, 08H, 08H, 0F0H, 00H, 20H, 3FH, 21H, 01H, 01H, 01H, 00H, 00H
Line2_11 DB 08H, 0F8H, 88H, 88H, 88H, 88H, 70H, 00H, 20H, 3FH, 20H, 00H, 03H, 0CH, 30H, 20H
Line2_12 DB 0E0H, 10H, 08H, 08H, 08H, 10H, 0E0H, 00H, 0FH, 10H, 20H, 20H, 20H, 10H, 0FH, 00H
;-- 文字: 欢 --
Line3_1 DB 14H, 24H, 44H, 84H, 64H, 1CH, 20H, 18H, 0FH, 0E8H, 08H, 08H, 28H, 18H, 08H, 00H
          DB 20H, 10H, 4CH, 43H, 43H, 2CH, 20H, 10H, 0CH, 03H, 06H, 18H, 30H, 60H, 20H, 00H
;-- 文字: 迎 --
Line3_2 DB 40H, 41H, 0CEH, 04H, 00H, 0FCH, 04H, 02H, 02H, 0FCH, 04H, 04H, 04H, 0FCH, 00H, 00H
          DB 40H, 20H, 1FH, 20H, 40H, 47H, 42H, 41H, 40H, 5FH, 40H, 42H, 44H, 43H, 40H, 00H
;-- 文字: 使 --

```

```

Line3_3 DB 40H, 20H, 0F0H, 1CH, 07H, 0F2H, 94H, 94H, 94H, 0FFH, 94H, 94H, 94H, 0F4H, 04H, 00H
          DB 00H, 00H, 7FH, 00H, 40H, 41H, 22H, 14H, 0CH, 13H, 10H, 30H, 20H, 61H, 20H, 00H
;-- 文字: 用 --
Line3_4 DB 00H, 00H, 00H, 0FEH, 22H, 22H, 22H, 22H, 0FEH, 22H, 22H, 22H, 22H, 0FEH, 00H, 00H
          DB 80H, 40H, 30H, 0FH, 02H, 02H, 02H, 02H, 0FFH, 02H, 02H, 42H, 82H, 7FH, 00H, 00H

          . CODE
START:    MOV     AX, @DATA
          MOV     DS, AX
          NOP
START1:   CALL    LCD_INIT      ;液晶初始化
          CALL    DelayTime
          CALL    DisLine1      ;第2行显示" 星研电子"
          CALL    DelayTime
          CALL    DisLine2      ;第3行显示" STAR ES51PRO"
          CALL    DelayTime
          CALL    DisLine3      ;第4行显示" 欢迎使用"
          CALL    DelayTime
          JMP     START1

;延时程序
DelayTime PROC     NEAR
          MOV     CX, 0
          LOOP    $
          LOOP    $
          RET
DelayTime ENDP

;第2行显示" 星研电子"
DisLine1 PROC     NEAR
          LEA     SI, Line1_1
          MOV     AL, 2          ;A-起始显示行地址, 第2行
          MOV     AH, 32        ;B-起始显示列地址, 第32列, 以下同
          CALL    WordDISL      ;左半屏, 显示一个字子程序
          LEA     SI, Line1_2
          MOV     AL, 2
          MOV     AH, 48
          CALL    WordDISL
          LEA     SI, Line1_3
          MOV     AL, 2
          MOV     AH, 0
          CALL    WordDISR      ;右半屏, 显示一个字子程序
          LEA     SI, Line1_4
          MOV     AL, 2
          MOV     AH, 16
          CALL    WordDISR
          RET

```

```

DisLine1      ENDP
;第3行显示” STAR ES51PRO”
DisLine2      PROC      NEAR
                LEA      SI,Line2_1
                MOV      AL,4                ;A-起始显示行地址，第4行
                MOV      AH,16              ;B-起始显示列地址，第16列，以下同
                CALL     ByteDISL          ;左半屏，显示一个字节子程序
                LEA      SI,Line2_2
                MOV      AL,4
                MOV      AH,24
                CALL     ByteDISL
                LEA      SI,Line2_3
                MOV      AL,4
                MOV      AH,32
                CALL     ByteDISL
                LEA      SI,Line2_4
                MOV      AL,4
                MOV      AH,40
                CALL     ByteDISL
                LEA      SI,Line2_5
                MOV      AL,4
                MOV      AH,48
                CALL     ByteDISL
                LEA      SI,Line2_6
                MOV      AL,4
                MOV      AH,56
                CALL     ByteDISL
                LEA      SI,Line2_7
                MOV      AL,4
                MOV      AH,0
                CALL     ByteDISR          ;右半屏字节显示数据
                LEA      SI,Line2_8
                MOV      AL,4
                MOV      AH,8
                CALL     ByteDISR
                LEA      SI,Line2_9
                MOV      AL,4
                MOV      AH,16
                CALL     ByteDISR
                LEA      SI,Line2_10
                MOV      AL,4
                MOV      AH,24
                CALL     ByteDISR
                LEA      SI,Line2_11

```



```

MOV        AL, 4
MOV        AH, 32
CALL       ByteDISR
LEA        SI, Line2_12
MOV        AL, 4
MOV        AH, 40
CALL       ByteDISR
RET
DisLine2   ENDP
;第4行显示” 欢迎使用”
DisLine3   PROC        NEAR
LEA        SI, Line3_1
MOV        AL, 6        ;A-起始显示行地址， 第6行
MOV        AH, 32       ;B-起始显示列地址， 第32列， 以下同
CALL       WordDISL     ;左半屏， 显示一个字子程序
LEA        SI, Line3_2
MOV        AL, 6
MOV        AH, 48
CALL       WordDISL
LEA        SI, Line3_3
MOV        AL, 6
MOV        AH, 0
CALL       WordDISR     ;右半屏， 显示一个字子程序
LEA        SI, Line3_4
MOV        AL, 6
MOV        AH, 16
CALL       WordDISR
RET
DisLine3   ENDP
;液晶初始化
LCD_INIT   PROC        NEAR
MOV        AL, 3EH      ;初始化左半屏， 关显示
CALL       WRComL       ;写指令子程序
MOV        AL, FirstLine ;设置起始显示行， 第0行
CALL       WRComL
MOV        AL, 3EH      ;初始化右半屏， 关显示
CALL       WRComR       ;写指令子程序
MOV        AL, FirstLine ;设置起始显示行， 第0行
CALL       WRComR
CALL       LCDClear     ;清屏
MOV        AL, 3FH      ;开显示
CALL       WRComL
MOV        AL, 3FH      ;开显示
CALL       WRComR

```

```

                                RET
LCD_INIT                        ENDP
;清屏
LCDClear                        PROC            NEAR
                                ;清左半屏
                                MOV             AL, 0             ;起始行，第0行
                                MOV             AH, 0             ;起始列，第0列
LCDClearL1:                    PUSH            AX
                                MOV             CX, 64
                                CALL             SETXYL           ;设置起始显示行列地址
LCDClearL2:                    MOV             AL, 0
                                CALL             WRDATAL
                                LOOP            LCDClearL2
                                POP             AX
                                INC             AX
                                CMP             AL, 8             ;共8行
                                JNZ            LCDClearL1
                                ;清右半屏
                                MOV             AL, 0             ;起始行，第0行
                                MOV             AH, 0             ;起始列，第0列
LCDClearR1:                    PUSH            AX
                                MOV             CX, 64
                                CALL             SETXYR           ;设置起始显示行列地址
LCDClearR2:                    XOR             AL, AL
                                CALL             WRDATAR
                                LOOP            LCDClearR2
                                POP             AX
                                INC             AL
                                CMP             AL, 8             ;共8行
                                JNZ            LCDClearR1
                                RET
LCDClear                        ENDP
;显示字体，显示一个数据要占用X行两行位置
;左半屏显示一个字节/字：AL-起始显示行序数X(0-7)；AH-起始显示列序数Y(0-63)；SI-显示字
数据首地址
ByteDisL                        PROC            NEAR
                                MOV             CX, 8             ;显示8个字节数据，用于显示一个英文/符号
                                CALL             DispL
                                RET
ByteDisL                        ENDP
WordDisL                        PROC            NEAR
                                MOV             CX, 16            ;显示16字节数据，用于显示一个汉字
                                CALL             DispL
                                RET

```

```

WordDisL      ENDP
DispL          PROC          NEAR
                PUSH        AX
                PUSH        CX
                CALL         SETXYL      ;设置起始显示行列地址
                CALL         DisplayL    ;显示上半行数据
                POP          CX
                POP          AX
                INC          AL
                CALL         SETXYL      ;设置起始显示行列地址
                CALL         DisplayL    ;显示下半行数据
                RET
DispL          ENDP
;右半屏显示一个字节/字: AL-起始显示行序数X(0-7); AH-起始显示列序数Y(0-63); SI-显示字
;数据首地址
ByteDisR       PROC          ENAR
                MOV          CX, 8      ;显示8个字节数据, 用于显示一个英文/符号
                CALL         DispR
                RET
ByteDisR       ENDP
WordDisR       PROC          NEAR
                MOV          CX, 16     ;显示16字节数据, 用于显示一个汉字
                CALL         DispR
                RET
WordDisR       ENDP
DispR          PROC          NEAR
                PUSH        AX
                PUSH        CX
                CALL         SETXYR      ;设置起始显示行列地址
                CALL         DisplayR    ;显示上半行数据
                POP          CX
                POP          AX
                INC          AL
                CALL         SETXYR      ;设置起始显示行列地址
                CALL         DisplayR    ;显示下半行数据
                RET
DispR          ENDP
;显示图形
;显示左半屏一行图形, AL-X起始行序数(0-7), AH-Y起始列地址序数(0-63)
LineDisL       PROC          NEAR
                MOV          CX, 64
                CALL         SETXYL      ;设置起始显示行列
                CALL         DisplayL    ;显示数据
                RET

```

```

LineDisL      ENDP
;显示右半屏一行图形, AL-X起始行地址序数(0-7), AH-Y起始列地址序数(0-63)
LineDisR      PROC          NEAR
                MOV          CX, 64
                CALL          SETXYR          ;设置起始显示行列
                CALL          DisplayR        ;显示数据
                RET
LineDisR      ENDP
;基本控制
;显示左半屏数据, R7-显示数据个数
DisplayL      PROC          NEAR
                LODSB
                CALL          WRDataL          ;写左半屏数据
                LOOP          DisplayL
                RET
DisplayL      ENDP
;显示右半屏数据, R7-显示数据个数
DisplayR      PROC          NEAR
                LODSB
                CALL          WRDataR          ;写左半屏数据
                LOOP          DisplayR
                RET
DisplayR      ENDP
;设置左半屏起始显示行列地址, AL-X起始行序数(0-7), AH-Y起始列序数(0-63)
SETXYL      PROC          NEAR
                OR            AL, X            ;行地址=行序数+行基址
                CALL          WRComL
                MOV          AL, AH
                OR            AL, Y            ;列地址=列序数+列基址
                CALL          WRComL
                RET
SETXYL      ENDP
;设置右半屏起始显示行列地址, AL-X起始行序数(0-7), AH-Y起始列序数(0-63)
SETXYR      PROC          NEAR
                OR            AL, X            ;行地址=行序数+行基址
                CALL          WRComR
                MOV          AL, AH
                OR            AL, Y            ;列地址=列序数+列基址
                CALL          WRComR
                RET
SETXYR      ENDP
;写左半屏控制指令, A-写入指令
WRComL      PROC          NEAR
                MOV          DX, WR_COM_AD_L

```

```

                                OUT            DX, AL
WRComL1:                       MOV            DX, RD_BUSY_AD
                                IN             AL, DX
                                TEST           AL, 80H           ;检查液晶显示是否处于忙状态
                                JNZ            WRComL1
                                RET
WRComL                         ENDP
;写右半屏控制指令, A-写入指令
WRComR                         PROC          NEAR
                                MOV            DX, WR_COM_AD_R
                                OUT            DX, AL
WRComR1:                       MOV            DX, RD_BUSY_AD
                                IN             AL, DX
                                TEST           AL, 80H           ;检查液晶显示是否处于忙状态
                                JNZ            WRComR1
                                RET
WRComR                         ENDP
;写左半屏数据, A-写入数据
WRDataL                       PROC          NEAR
                                MOV            DX, WR_DATA_AD_L
                                OUT            DX, AL
WRDataL1:                      MOV            DX, RD_BUSY_AD
                                IN             AL, DX
                                TEST           AL, 80H           ;检查液晶显示是否处于忙状态
                                JNZ            WRDataL1
                                RET
WRDataL                       ENDP
;写右半屏数据, A-写入数据
WRDataR                       PROC          NEAR
                                MOV            DX, WR_DATA_AD_R
                                OUT            DX, AL
WRDataR1:                      MOV            DX, RD_BUSY_AD
                                IN             AL, DX
                                TEST           AL, 80H           ;检查液晶显示是否处于忙状态
                                JNZ            WRDataR1
                                RET
WRDataR                       ENDP

                                END            START

```

七、实验扩展及思考

实验内容：显示一幅图画，进一步熟练液晶显示的操作。

实验十三 8237 DMA 传输实验

一、实验目的与要求

了解 8237 的内部结构、工作原理；了解 8237 与 8086 的接口逻辑；掌握使用 8237，实现 DMA 传输数据。

复习本节实验内容，可尝试自行编写程序，做好实验准备工作，填写实验报告。

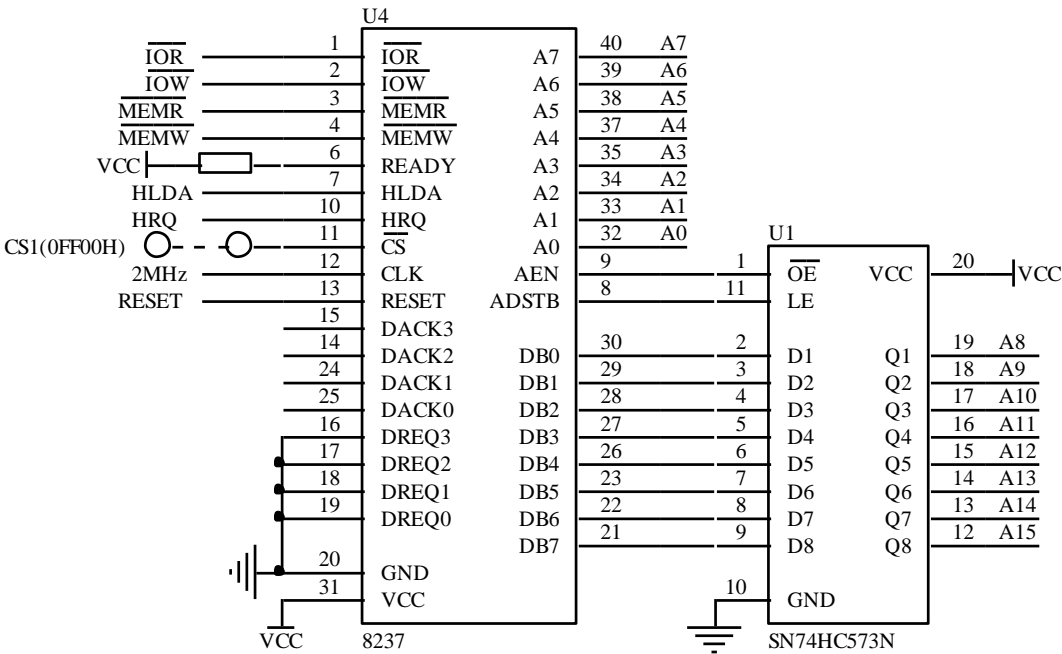
二、实验设备

STAR 系列实验仪一套、PC 机一台

三、实验内容

1、编制程序：将 DS:3000H~37FFH 内数据通过 DMA 方式传输给 DS:6000H~67FFH，并对 DS:3000H~37FFH 与 DS:6000H~67FFH 作比较。

四、实验原理图



五、实验步骤

1、连线说明：

EMU598+： CS	——	A3 区： CS2
-------------	----	-----------

2、编写、调试程序

3、DS:3000H~37FFH 与 DS:6000H~67FFH 二块数据是否完全相同

六、演示程序

```
.MODEL TINY
DMAaddr EQU 0E000H ;8237基地址
.STACK 100
.CODE
STAR: MOV AX, 0
      MOV DS, AX
      mov si, 3000h
      mov al, 0ffh
```

```

STAR3:    mov     cx, 0800h
          mov     [si], al
          inc     si
          dec     al
          loop    STAR3
          mov     al, 04h
          mov     dx, DMAAddr+8
          out     dx, al           ;禁止DMA操作
          mov     al, 00
          mov     dx, DMAAddr+1ch
          out     dx, al           ;复位
          mov     dx, DMAAddr+0ch
          out     dx, al           ;清除先/后寄存器
          mov     dx, DMAAddr+0           ;源起始地址3000H
          mov     al, 0
          out     dx, al
          mov     al, 30h
          out     dx, al
          mov     dx, DMAAddr+0ch
          out     dx, al           ;清除先/后寄存器
          mov     dx, DMAAddr+2           ;目的起始地址6000H
          mov     al, 0
          out     dx, al
          mov     al, 60h
          out     dx, al
          mov     dx, DMAAddr+0ch
          out     dx, al           ;清除先/后寄存器
          mov     dx, DMAAddr+12h
          mov     al, 0ffh           ;长度0800H
          out     dx, al
          mov     al, 07h
          out     dx, al
          mov     dx, DMAAddr+1ah
          mov     al, 88h
          out     dx, al           ;通道0方式字
          mov     al, 85h
          out     dx, al           ;通道1方式字
          mov     dx, DMAAddr+8
          mov     al, 41H
          out     dx, al           ;允许8237工作、存储器方式传送
star1:    mov     dx, DMAAddr+1eh
          mov     al, 0ch
          out     dx, al           ;允许通道0、通道1
          mov     dx, DMAAddr+18h

```

```

        mov     al, 04h
        out     dx, al           ;允许DMA操作
        mov     dx, DMAAddr+08h
        NOP
        NOP
star2:   in      al, dx
        TEST    AL, 03H
        jz      star2           ;等待DMA结束
        mov     dx, DMAAddr+0ch
        out     dx, al
        mov     dx, DMAAddr+12h ;清除先/后寄存器
        IN      AL, DX
        MOV     AH, AL
        IN      AL, DX
        CMP     AX, 0FFFFH
        JNZ     star1
        mov     dx, DMAAddr+18h
        mov     al, 00h
        out     dx, al           ;清除DMA请求
        mov     dx, DMAAddr+08h
        mov     al, 04h
        out     dx, al           ;禁止DMA操作
        mov     si, 3000h
        mov     bx, 6000h
        mov     cx, 0800h
STAR5:   mov     al, [si]
        cmp     al, ds:[bx]
        jne     False
        inc     Si
        inc     Bx
        loop    STAR5
true:    jmp     $
false:   jmp     $

        END     STAR

```

七、实验扩展及思考

1、8237 的 DMA 传输有多种工作方式，请尝试一下。

6 综合实验

实验一 简易电子琴实验

一、实验目的与要求

掌握蜂鸣器的使用方法；掌握蜂鸣器的不同发音的方法。

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

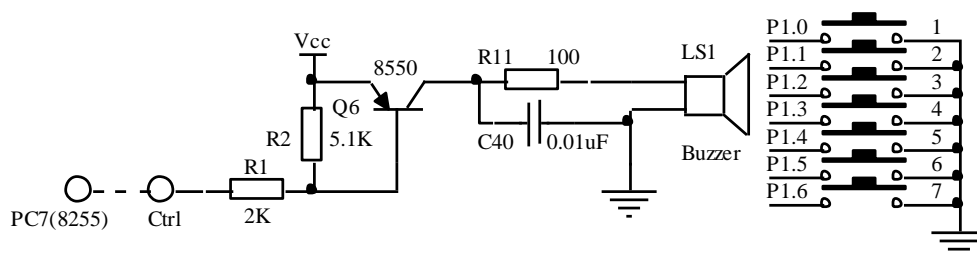
1、简易电子琴原理：

- (1) 蜂鸣器输入不同频率的方波，会发出不同的声音；
- (2) 通过按键，由 8255 控制产生不同频率的方波，从而发出不同的声音。

2. 实验过程

- (1) 通过 8255 的 PA 口，使 F5 区的 1~7 号键由低到高发出 1-7 的音阶。

四、实验原理图



五、实验步骤

1、主机连线说明：

E3 区：Ctrl	——	B4 区：PC7
B4 区：CS(8255)、A0、A1	——	A3 区：CS1、A0、A1
B4 区：JP56	——	F5 区：JP74

2、运行程序，按 F5 区的 1~7 号键，输出 7 种音阶

3、使用 F5 区的 1~7 号键，弹一首《生日快乐》。

六、演示程序

```
.MODEL          TINY
C8255           EQU          0F003H
PA8255          EQU          0F000H
.STACK          100
.DATA
```

```

Music      DW  M1, M2, M3, M4, M5, M6, M7, M7, M7, M6, M5, M4, M3, M2, M1
           DW  M1, M2, M1, M2, M3, M2, M3, M4, M3, M4, M5, M4, M5, M6, M5
           DW  M6, M7, M6, M7, M7, M6, M6, M6
           . CODE
START:     MOV      AX, @DATA
           MOV      DS, AX
           CALL     INIT8255      ;8255 初始化
           CALL     Demo          ;播放一段音乐
START1:    MOV      DX, PA8255    ;按键查询
           IN       AL, DX        ;读键值
           CMP      AL, 0FFH
           JZ       START1        ;无键
           XOR      AL, 0FFH      ;有键
           TEST     AL, 1
           JZ       START2
           CALL     Music1        ;1 号键, 调 1 号键输出
           JMP      START1
START2:    TEST     AL, 2
           JZ       START3
           CALL     Music2        ;2 号键
           JMP      START1
START3:    TEST     AL, 4
           JZ       START4
           CALL     Music3        ;3 号键
           JMP      START1
START4:    TEST     AL, 8
           JZ       START5
           CALL     Music4        ;4 号键
           JMP      START1
START5:    TEST     AL, 10H
           JZ       START6
           CALL     Music5        ;5 号键
           JMP      START1
START6:    TEST     AL, 20H
           JZ       START7
           CALL     Music6        ;6 号键
           JMP      START1
START7:    TEST     AL, 40H
           JZ       START1
           CALL     Music7        ;7 号键
           JMP      START1
Demo      PROC      NEAR
           MOV      CX, 38        ;共 38 拍
           LEA      BX, Music

```

```

Demo10:      PUSH      CX
              CALL      [BX]          ;播放该音调声音
              INC       BX
              INC       BX
              POP        CX
              LOOP       Demo10
              RET

Demo          ENDP
;节拍 1(手动按键时用)
Music1        PROC      NEAR
              CALL      W_L           ;写 0, 蜂鸣器响
              CALL      T10          ;延时 100us
              CALL      T10
              CALL      T5           ;延时 50us
              CALL      T2           ;延时 20us
              CALL      T1           ;延时 10us
              CALL      W_H           ;写 1, 蜂鸣器不响
              CALL      T10          ;延时
              CALL      T10          ;
              CALL      T5           ;
              CALL      T2           ;延时 20us
              CALL      T1           ;延时 10us
              RET

Music1        ENDP
;节拍 2, 同上
Music2        PROC      NEAR
              CALL      W_L
              CALL      T10
              CALL      T10
              CALL      T5
              CALL      T1
              CALL      W_H
              CALL      T10
              CALL      T10
              CALL      T2
              CALL      T2
              CALL      T1
              RET

Music2        ENDP
;节拍 3, 同上
Music3        PROC      NEAR
              CALL      W_L
              CALL      T10
              CALL      T10

```

	CALL	T2
	CALL	T2
	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T2
	CALL	T2
	RET	
Music3	ENDP	
;节拍 4, 同上		
Music4	PROC	NEAR
	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T2
	NOP	
	NOP	
	NOP	
	NOP	
	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T2
	NOP	
	NOP	
	NOP	
	NOP	
	RET	
Music4	ENDP	
;节拍 5, 同上		
Music5	PROC	NEAR
	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T1
	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T1
	RET	
Music5	ENDP	
;节拍 6, 同上		
Music6	PROC	NEAR
	CALL	W_L

	CALL	T10
	CALL	T5
	CALL	T2
	CALL	T2
	CALL	T1
	NOP	
	NOP	
	CALL	W_H
	CALL	T10
	CALL	T5
	CALL	T2
	CALL	T2
	CALL	T1
	NOP	
	NOP	
	RET	
Music6	ENDP	
;节拍 7, 同上		
Music7	PROC	NEAR
	CALL	W_L
	CALL	T10
	CALL	T5
	CALL	T2
	CALL	T2
	CALL	W_H
	CALL	T10
	CALL	T5
	CALL	T2
	CALL	T1
	RET	
Music7	ENDP	
;节拍 1(自动放音时用, 时间约 0. 2s)		
M1	PROC	NEAR
	MOV	CX, 1100
M10:	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T5
	CALL	T2
	CALL	T2
	NOP	
	NOP	
	NOP	
	LOOP	M11

M11:	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T5
	CALL	T2
	CALL	T1
	NOP	
	NOP	
	NOP	
	LOOP	M10
	RET	
M1	ENDP	
;节拍 2, 同上		
M2	PROC	NEAR
	MOV	CX, 1150
M20:	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T2
	CALL	T2
	CALL	T1
	CALL	T1
	LOOP	M21
M21:	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T2
	CALL	T2
	CALL	T1
	LOOP	M20
	RET	
M2	ENDP	
;节拍 3, 同上		
M3	PROC	NEAR
	MOV	CX, 1200
M30:	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T5
	LOOP	M31
M31:	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T2

	CALL	T2
	NOP	
	NOP	
	LOOP	M30
	RET	
M3	ENDP	
;节拍 4, 同上		
M4	PROC	NEAR
	MOV	CX, 1250
M40:	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T2
	CALL	T1
	PUSH	AX
	POP	AX
	PUSH	AX
	POP	AX
	LOOP	M41
M41:	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T2
	CALL	T1
	PUSH	AX
	POP	AX
	PUSH	AX
	POP	AX
	NOP	
	NOP	
	LOOP	M40
	RET	
M4	ENDP	
;节拍 5, 同上		
M5	PROC	NEAR
	MOV	CX, 1300
M50:	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T2
	CALL	T1
	PUSH	AX
	POP	AX
	LOOP	M51

M51:	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T2
	NOP	
	NOP	
	NOP	
	NOP	
	LOOP	M50
	RET	
M5	ENDP	
;节拍 6, 同上		
M6	PROC	NEAR
	MOV	CX, 1350
M60:	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T1
	NOP	
	NOP	
	LOOP	M61
M61:	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T2
	PUSH	AX
	POP	AX
	LOOP	M60
	RET	
M6	ENDP	
;节拍 7, 同上		
M7	PROC	NEAR
	MOV	CX, 1420
M70:	CALL	W_L
	CALL	T10
	CALL	T10
	CALL	T1
	LOOP	M71
M71:	CALL	W_H
	CALL	T10
	CALL	T10
	CALL	T1
	LOOP	M70
	RET	


```

M7                ENDP
;写 0 (8255. PC. 7=0)
W_L              PROC    NEAR
                  MOV     DX, C8255
                  MOV     AL, 0EH
                  OUT     DX, AL
                  RET
W_L              ENDP
;写 1 (8255. PC. 7=1)
W_H              PROC    NEAR
                  MOV     DX, C8255
                  MOV     AL, 0FH
                  OUT     DX, AL
                  RET
W_H              ENDP
;8255 初始化
INIT8255         PROC    NEAR
                  MOV     DX, C8255
                  MOV     AL, 90H           ;PC. 7 输出, PA 输入
                  OUT     DX, AL
                  MOV     DX, C8255
                  MOV     AL, 0FH
                  OUT     DX, AL
                  RET
INIT8255         ENDP
;延时 10us
T1              PROC    NEAR
                  RET
T1              ENDP
;延时 20us
T2              PROC    NEAR
                  CALL     T1
                  RET
T2              ENDP
;延时 50us
T5              PROC    NEAR
                  CALL     T2
                  CALL     T2
                  RET
T5              ENDP
;延时 100s
T10             PROC    NEAR
                  CALL     T2
                  CALL     T2

```

	CALL	T5
	RET	
T10	ENDP	
	END	START

七. 实验扩展及思考题

设计一个简易电子播放器实验程序，使用蜂鸣器，回放一段音乐。

实验二 LED8 * 8 点阵实验

一、实验目的与要求

- 1、熟悉 8255 的功能，了解点阵显示的原理及控制方法；
- 2、学会使用 LED 点阵，通过编程显示不同字符；
- 3、认真预习本节实验内容，可尝试自行编写程序，做好实验准备工作，填写实验报告。

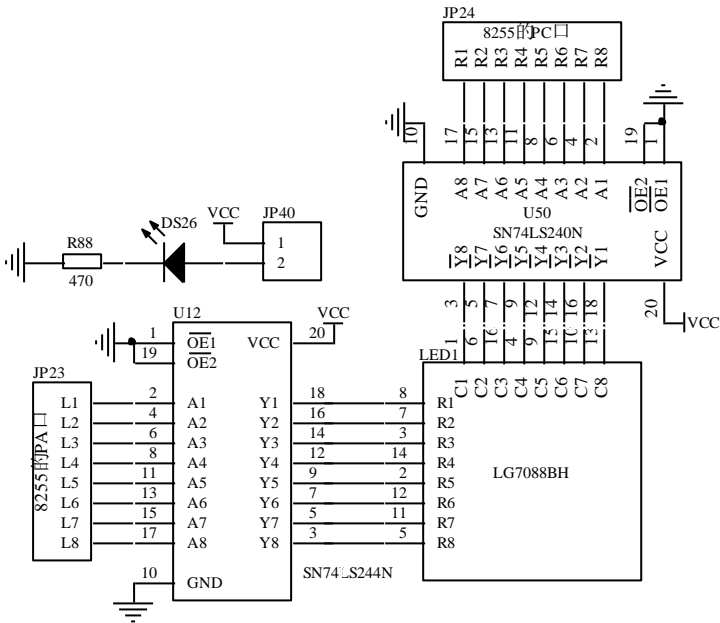
二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

- 1、编写程序，用 8255 的 PA 控制 8X8 点阵的行；8255 的 PC 口控制 8X8 点阵的列；显示字符。
- 2、按图连接线路；运行程序，观察实验结果，学会编程控制 LED 点阵显示字符。

四、实验原理图



五、实验步骤

- 1、主机连线说明：

B4 区：CS (8255)、A0、A1	——	A3 区：CS1、A0、A1
B4 区：JP56	——	A2 区：JP23 (列输出线)
B4 区：JP52	——	A2 区：JP24 (行输出线)

(注意连线方向)

- 2、运行程序，观察实验结果。运行演示程序将会看到字符“WELCOME TO XINGYAN”在点阵上自下而上循环移动显示。

六、演示程序

```
.MODEL TINY
```

```

ADDR_8255_PA    EQU        0F000H                ;8255 PA口
ADDR_8255_PB    EQU        0F001H                ;8255 PB口
ADDR_8255_PC    EQU        0F002H                ;8255 PC口
ADDR_8255_C     EQU        0F003H                ;8255控制口
ADDR_8155_PA    EQU        0E101H                ;8155 PA口
ADDR_8155_C     EQU        0E100H                ;8155控制口
LINE1           EQU        ADDR_8255_PA          ;行线1
LINE2           EQU        ADDR_8255_PB          ;行线2
ROW1            EQU        ADDR_8255_PC          ;列线1
ROW2            EQU        ADDR_8155_PA          ;列线2

        .STACK    100
        .DATA

HUAN    DB  00H, 0C0H, 00H, 0C0H, 0FEH, 0C0H, 07H, 0FFH, 0C7H, 86H, 6FH, 6CH, 3CH, 60H, 18H, 60H
        DB  1CH, 60H, 1CH, 70H, 36H, 0F0H, 36H, 0D8H, 61H, 9CH, 0C7H, 0FH, 3CH, 06H, 00H, 00H
YING    DB  60H, 00H, 31H, 0C0H, 3FH, 7EH, 36H, 66H, 06H, 66H, 06H, 66H, 0F6H, 66H, 36H, 66H
        DB  37H, 0E6H, 37H, 7EH, 36H, 6CH, 30H, 60H, 30H, 60H, 78H, 00H, 0CFH, 0FFH, 00H, 00H
SHI     DB  00H, 00H, 06H, 30H, 07H, 30H, 0FH, 0FFH, 0CH, 30H, 1FH, 0FFH, 3BH, 33H, 7BH, 33H
        DB  1BH, 0FFH, 1BH, 33H, 19H, 0B0H, 18H, 0E0H, 18H, 60H, 18H, 0FCH, 19H, 8FH, 1FH, 03H
YONG    DB  00, 0, 1FH, 0FEH, 18H, 0C6H, 18H, 0C6H, 18H, 0C6H, 1FH, 0FEH, 018H, 0C6H, 18H, 0C6H
        DB  18H, 0C6H, 1FH, 0FEH, 18H, 0C6H, 18H, 0C6H, 30H, 0C6H, 30H, 0C6H, 60H, 0DEH, 0C0H, 0CCH
XING    DB  00H, 00H, 1FH, 0FCH, 18H, 0CH, 1FH, 0FCH, 18H, 0CH, 1FH, 0FCH, 01H, 80H, 19H, 80H
        DB  1FH, 0FEH, 31H, 80H, 31H, 80H, 6FH, 0FCH, 01H, 80H, 01H, 80H, 7FH, 0FFH, 00H, 00H
YAN     DB  0, 0, 0FFH, 0FFH, 18H, 0CCH, 18H, 0CCH, 30H, 0CCH, 30H, 0CCH, 7FH, 0FFH, 7CH, 0CCH
        DB  0FCH, 0CCH, 3CH, 0CCH, 3CH, 0CCH, 3DH, 8CH, 3DH, 8CH, 33H, 0CH, 06H, 0CH, 0CH, 0CH
SHIO    DB  01H, 80H, 00H, 0C0H, 3FH, 0FFH, 3CH, 06H, 67H, 0CCH, 06H, 0C0H, 0CH, 0C0H, 07H, 0C0H
        DB  06H, 0C0H, 7FH, 0FFH, 00H, 0C0H, 01H, 0E0H, 03H, 30H, 06H, 18H, 1CH, 1CH, 70H, 18H
YANO    DB  00H, 00H, 0FCH, 60H, 0CH, 60H, 6CH, 0F0H, 6CH, 0D8H, 6DH, 8FH, 6FH, 0F8H, 7EH, 00H
        DB  06H, 0C6H, 07H, 66H, 3FH, 0ECH, 0E7H, 0ECH, 06H, 18H, 1FH, 0FFH, 0CH, 00H, 00H, 00H
YI      DB  0CH, 0C0H, 0CH, 60H, 18H, 7CH, 1BH, 6CH, 33H, 0CH, 73H, 18H, 0F1H, 98H, 31H, 98H
        DB  30H, 0F0H, 30H, 0F0H, 30H, 60H, 30H, 0F0H, 31H, 98H, 33H, 0FH, 3EH, 06H, 30H, 00H
NONE    DB  00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H
        DB  00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H, 00H

        .CODE

START:   MOV        AX, @DATA
        MOV        DS, AX
        MOV        ES, AX
        NOP
        CALL       INIT_IO
        CALL       TEST_LED                ;调用测试子程序, 测试LED是否全亮
        CALL       CLEAR

;滚动显示多个字符
CHS_SHOW:    MOV        CX, 9
            LEA        SI, HUAN
CHS_1:      PUSH     CX

```

```

                MOV        CX, 16
CHS_2:          CALL        DISP_CH
                INC        SI
                INC        SI
                LOOP       CHS_2
                POP        CX
                LOOP       CHS_1
                JMP        CHS_SHOW
;显示一个16*16点阵字子程序, 字型码放在DPTR指出的地址
DISP_CH          PROC        NEAR
                PUSH       CX
                MOV        CX, 8
DISP_CH_1:      CALL        DISP1
                LOOP       DISP_CH_1
                POP        CX
                RET
DISP_CH          ENDP
;显示一个16*16点阵字子程序, 字型码放在显示缓冲区XBUF
DISP1            PROC        NEAR
                PUSH       SI
                PUSH       CX
                MOV        CX, 16                ;计数器, 16列依次被扫描
                MOV        BL, 0FEH            ;上边列输出值
                MOV        BH, 0FFH            ;下边列输出值
REPEAT:          MOV        DX, LINE1
                MOV        AL, BL
                OUT         DX, AL                ;上边列输出
                MOV        DX, LINE2
                MOV        AL, BH
                OUT         DX, AL                ;下边列输出
                LODSB
                CALL        ADJUST                ;调整AL, 将AL中二进制数旋转180度
                MOV        DX, ROW1
                OUT         DX, AL                ;左边行输出
                LODSB
                CALL        ADJUST                ;调整AL, 将AL中二进制数旋转180度
                MOV        DX, ROW2
                OUT         DX, AL                ;右边行输出
                CALL        DL10MS
                CALL        CLEAR
                STC
                RCL         BL, 1
                RCL         BH, 1                ;循环移位BX, 行线扫描输出0
                LOOP       REPEAT

```

```

        POP        CX
        POP        SI
        RET
DISP1   ENDP
INIT_IO PROC        NEAR
        MOV        DX, ADDR_8255_C      ;8255控制字地址
        MOV        AL, 80H              ;设置8255的PA、PB、PC口为输出口
        OUT        DX, AL               ;写控制字
        MOV        DX, ADDR_8155_C      ;8155控制字地址
        MOV        AL, 03H              ;设置8155的PA口为输出
        OUT        DX, AL               ;写控制字
        RET
INIT_IO ENDP
CLEAR   PROC        NEAR
        MOV        AL, 0FFH
        MOV        DX, LINE1
        OUT        DX, AL
        MOV        DX, LINE2
        OUT        DX, AL
        MOV        AL, 0
        MOV        DX, ROW1
        OUT        DX, AL
        MOV        DX, ROW2
        OUT        DX, AL
        RET
CLEAR   ENDP
;测试LED子程序, 点亮LED并延时1S
TEST_LED PROC        NEAR
        MOV        DX, LINE1
        XOR        AL, AL
        OUT        DX, AL
        MOV        DX, LINE2
        OUT        DX, AL
        MOV        AL, 0FFH
        MOV        DX, ROW1
        OUT        DX, AL
        MOV        DX, ROW2
        OUT        DX, AL
        CALL        DL500ms
        CALL        DL500ms
        RET
TEST_LED ENDP
;调整AL中取到的字型码的一个字节, 将最高位调整位最低位, 最低位调整为最高位
ADJUST PROC        NEAR

```

	PUSH	CX
	MOV	CX, 8
ADJUST1:	RCL	AL, 1
	XCHG	AL, AH
	RCR	AL, 1
	XCHG	AL, AH
	LOOP	ADJUST1
	MOV	AL, AH
	POP	CX
	RET	
ADJUST	ENDP	
DL10ms	PROC	NEAR
	PUSH	CX
	MOV	CX, 133
	LOOP	\$
	POP	CX
	RET	
DL10ms	ENDP	
DL500ms	PROC	NEAR
	PUSH	CX
	MOV	CX, 0FFFFH
	LOOP	\$
	POP	CX
	RET	
DL500ms	ENDP	
	END	START

七、实验扩展及思考

- 1、如果并行扩展口线不多，能不能用串并转换方式替代 8255 和 8155，若可以具体如何连线，若不行，还有其它方法吗？
- 2、修改程序，使显示的字符从左至右动态循环显示。

实验二 LED8 * 8 双色点阵实验(选配)

一、实验目的与要求

- 1、熟悉 8255 的功能，了解点阵显示的原理及控制方法；
- 2、学会使用 LED 点阵，通过编程显示不同字符；
- 3、认真预习本节实验内容，可尝试自行编写程序，做好实验准备工作，填写实验报告。

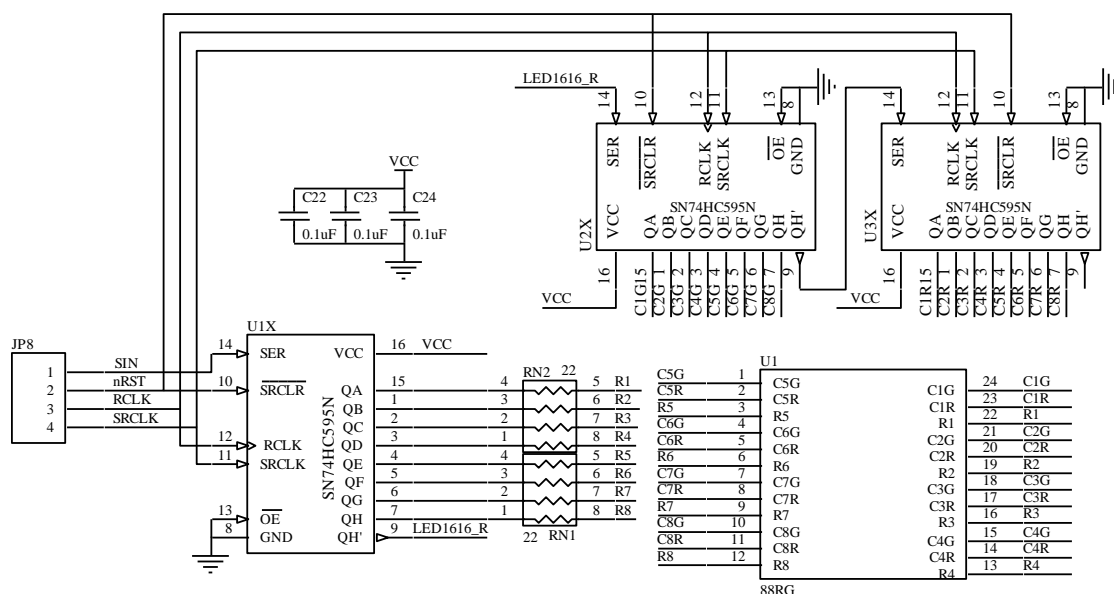
二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

- 1、编写程序，用 8255 的 PA 通过 3 片 74HC595 控制 8*8 双色点阵；显示字符
- 2、按图连接线路；运行程序，观察实验结果，学会编程控制 LED 点阵显示字符。

四、实验原理图



五、实验步骤

- 1、主机连线说明：

B4 区：CS (8255)、A0、A1	——	A3 区：CS1、A0、A1
B4 区：JP56 (PA 口)	——	F6 区：JP23

说明：PA. 7—Sin；PA. 6—nRST；PA. 5—RCLK；PA. 4—SRCLK。

- 2、运行程序，观察实验结果。运行演示程序将会看到字符“WELCOME TO XINGYAN”在点阵上自下而上循环移动显示。

六、演示程序

```
.MODEL TINY
ADDR_8255_PA EQU 0F000H ;8255 PA 口
ADDR_8255_C EQU 0F003H ;8255 控制口
```



```

        . STACK      100
        . DATA
CHAR_TAB: DB      000H, 082H, 092H, 092H, 0AAH, 0AAH, 044H, 044H ;W
          DB      000H, 03EH, 002H, 002H, 01EH, 002H, 002H, 03EH ;E
          DB      000H, 002H, 002H, 002H, 002H, 002H, 002H, 03EH ;L
          DB      000H, 038H, 044H, 002H, 002H, 002H, 044H, 038H ;C
          DB      000H, 038H, 044H, 082H, 082H, 082H, 044H, 038H ;O
          DB      000H, 0C6H, 0C6H, 0AAH, 0AAH, 092H, 092H, 082H ;M
          DB      000H, 03EH, 002H, 002H, 01EH, 002H, 002H, 03EH ;E
          DB      000H, 000H, 000H, 000H, 000H, 000H, 000H, 000H ;
          DB      000H, 07FH, 008H, 008H, 008H, 008H, 008H, 008H ;T
          DB      000H, 038H, 044H, 082H, 082H, 082H, 044H, 038H ;O
          DB      000H, 000H, 000H, 000H, 000H, 000H, 000H, 000H ;
          DB      000H, 022H, 022H, 014H, 008H, 014H, 022H, 022H ;X
          DB      000H, 01CH, 008H, 008H, 008H, 008H, 008H, 01CH ;I
          DB      000H, 042H, 046H, 04AH, 052H, 062H, 042H, 042H ;N
          DB      000H, 038H, 044H, 002H, 072H, 042H, 044H, 038H ;G
          DB      000H, 022H, 022H, 014H, 014H, 008H, 008H, 008H ;Y
          DB      000H, 018H, 018H, 024H, 024H, 07EH, 042H, 042H ;A
          DB      000H, 042H, 046H, 04AH, 052H, 062H, 042H, 042H ;N
          DB      000H, 000H, 000H, 000H, 000H, 000H, 000H, 000H ;

COM      DB      ?          ;8 位公共端, 高电平有效
RG       DW      ?          ;高 8 位红色, 低 8 位绿色, 低电平有效, 对应纵向 8 个发光管
MODE     DB      ?          ;0--显示红色;1--显示绿色;2--显示橙色

        . CODE
START:   MOV      AX, @DATA
          MOV      DS, AX
          NOP
          CALL     Init8255
          CALL     TEST_LED    ;调用测试子程序, 测试 LED 是否全亮
          CALL     CLEAR      ;清除显示

;滚动显示多个字符
CHS_SHOW: MOV      MODE, 00H
          MOV      CX, 12H
          LEA      SI, CHAR_TAB
CHS_1:   PUSH     CX
          MOV      CX, 8
CHS_2:   CALL     DISP_CH
          INC      SI
          LOOP     CHS_2
          POP      CX

```

```

                CMP        CX, 12
                JNE        CHS_3
                MOV        MODE, 01H
                JMP        CHS_4
CHS_3:          CMP        CX, 9
                JNE        CHS_4
                MOV        MODE, 02H
CHS_4:          LOOP       CHS_1
                JMP        CHS_SHOW

```

;显示一个 8*8 点阵字母子程序, 字型码放在 SI 指出的地址

```

DISP_CH        PROC        NEAR
                PUSH        CX
                MOV        CX, 10H
DISP_CH_1:      CALL        DISP1
                LOOP       DISP_CH_1
                POP         CX
                RET
DISP_CH        ENDP

```

;显示一个 8*8 点阵字母子程序, 字型码放在显示缓冲区 XBUFF

```

DISP1          PROC        NEAR
                PUSH        SI
                PUSH        CX
                MOV        CX, 8           ;计数器, 8 行依次被扫描
                MOV        COM, 01H       ;控制显示哪一列
REPEAT:         XOR        AX, AX
                LODSB
                NOT         AX
;               CALL        ADJUST         ;调整 AL, 将 AL 中二进制数旋转 180 度
                CMP        MODE, 00H
                JZ          DISP1_1
                CMP        MODE, 01H
                JNZ         DISP1_2
                XCHG        AH, AL
                JMP         DISP1_1
DISP1_2:        MOV        AH, AL
DISP1_1:        MOV        RG, AX
                CALL        Line_Row
                CALL        DL10MS
                CALL        CLEAR
                ROL         COM, 1         ;循环移位, 显示 8 列
                LOOP       REPEAT
                POP         CX

```

```

        POP        SI
        RET
DISP1   ENDP

;8 位数据在 BL 中
w8bits  PROC      NEAR
        PUSH      CX
        MOV        CX, 8
        MOV        DX, ADDR_8255_PA
w8bits_1: MOV        AL, 7FH          ;bSin = 0;
        TEST       BL, 80H
        JZ         w8bits_2
        MOV        AL, 0FFH          ;bSin = 1
w8bits_2: OUT        DX, AL
        AND        AL, 0EFH          ;bSRclk = 0
        OUT        DX, AL
        OR         AL, 10H           ;bSRclk = 1
        OUT        DX, AL
        SHL        BL, 1
        LOOP       w8bits_1
        POP        CX
        RET
w8bits  ENDP

;16 位数据在 BX 中
w16bits PROC      NEAR
        PUSH      CX
        MOV        CX, 10H
        MOV        DX, ADDR_8255_PA
w16bits_1: MOV        AL, 7FH          ;bSin = 0;
        TEST       BX, 8000H
        JZ         w16bits_2
        MOV        AX, 0FFH          ;bSin = 1
w16bits_2: OUT        DX, AL
        AND        AL, 0EFH          ;bSRclk = 0
        OUT        DX, AL
        OR         AL, 10H           ;bSRclk = 1
        OUT        DX, AL
        SHL        BX, 1
        LOOP       w16bits_1
        POP        CX
        RET
w16bits ENDP

CLEAR   PROC      NEAR

```

```

        PUSH        AX
        MOV         AX, 0BFH                ;bRst = 0
        MOV         DX, ADDR_8255_PA
        OUT         DX, AL
        MOV         AL, 0FFH                ;bRst = 1
        OUT         DX, AL
        MOV         AL, 0DFH                ;bRclk = 0
        OUT         DX, AL
        MOV         AL, 0FFH                ;bRclk = 1
        OUT         DX, AL
        POP         AX
        RET
CLEAR    ENDP
Line_Row PROC    NEAR
        MOV         BX, RG
        CALL        w16bits
        MOV         BL, COM
        CALL        w8bits
        MOV         AL, 0DFH                ;bRclk = 0
        MOV         DX, ADDR_8255_PA
        OUT         DX, AL
        MOV         AL, 0FFH                ;bRclk = 1
        MOV         DX, ADDR_8255_PA
        OUT         DX, AL
        RET
Line_Row ENDP
;测试 LED 子程序
TEST_LED PROC    NEAR
        MOV         COM, 0FFH
        MOV         RG, 00FFH
        CALL        Line_Row                ;点阵发红色
        CALL        DL500ms
        CALL        DL500ms
        MOV         RG, 0FF00H
        CALL        Line_Row                ;点阵发绿色
        CALL        DL500ms
        CALL        DL500ms
        MOV         RG, 55AAH
        CALL        Line_Row                ;点阵一半发红色, 一半发绿色, 相间
        CALL        DL500ms
        CALL        DL500ms
        MOV         RG, 0AA55H
        CALL        Line_Row                ;点阵一半绿色发, 一半发红色, 相间, 与上一步相反
        CALL        DL500ms

```

```

CALL    DL500ms
MOV     RG, 0000H
CALL    Line_Row      ;点阵发橙色
CALL    DL500ms
CALL    DL500ms
RET
TEST_LED ENDP
;调整 AL 中取到的字型码的一个字节, 将最高位调整位最低位, 最低位调整为最高位
ADJUST  PROC    NEAR
        PUSH    CX
        MOV     CX, 8
ADJUST1: RCL     AL, 1
        XCHG    AL, DL
        RCR     AL, 1
        XCHG    AL, DL
        LOOP    ADJUST1
        MOV     AL, DL
        POP     CX
        RET
ADJUST  ENDP
DL10ms  PROC    NEAR
        PUSH    CX
        MOV     CX, 133
        LOOP    $
        POP     CX
        RET
DL10ms  ENDP
DL500ms PROC    NEAR
        PUSH    CX
        MOV     CX, 0FFFFH
        LOOP    $
        POP     CX
        RET
DL500ms ENDP
Init8255 PROC    NEAR
        MOV     DX, ADDR_8255_C
        MOV     AL, 80H
        OUT     DX, AL
        RET
Init8255 ENDP

```

END START

七、实验扩展及思考

- 1、修改程序，使显示的字符从左至右动态循环显示。

实验三 数字式温度计实验(18B20)

一、实验目的

掌握一线串行接口的读写操作；掌握数字温度计 DS18B20 的使用

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

1、DS18B20:

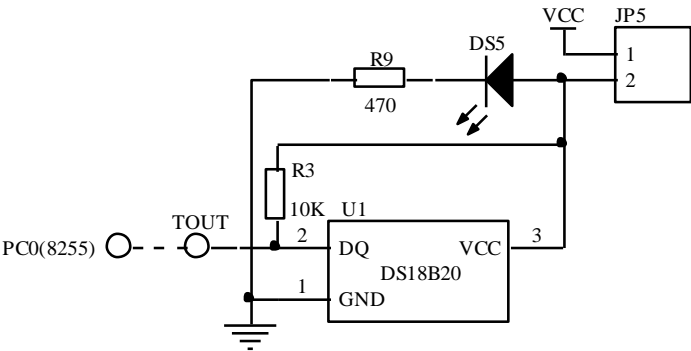
- (1) 一线串行接口数字式温度计
- (2) 温度测量范围-55℃-125℃，-10℃-85℃内误差±0.5℃
- (3) 9-12 位转换精度，转换时间 100ms-750ms，通常为 500ms

2、实验过程

- (1) 应用 DS18B20 制作一个数字温度计，通过 DS18B20 测量温度，8279 控制 LED（F4 区）

动态显示温度

四. 实验原理图



五. 实验步骤

1、主机连线说明:

E4 区: TOUT	——	B4 区: PC0 (8255)
B4 区: CS (8255)、A0、A1	——	A3 区: CS1、A0、A1
D3 区: CS、A0	——	A3 区: CS5、A0
D3 区: CLK	——	B2 区: 2M

2、使用 DS18B20 测量温度，将读出的十六进制温度值转换为十进制数

3、通过 LED (F4 区) 动态显示温度，温度数据通过 DS18B20 获取。可用手指贴住 DS18B20 (E4 区)，温度显示会随之上升。

六. 演示程序

```
.MODEL      TINY
Con_8255    EQU      0F003H
PC_8255     EQU      0F002H
EXTRN      Display8:NEAR
            .STACK    300
            .DATA
buffer      DB          8 DUP (0)           ;温度临时存放区
            .CODE
START:      MOV        AX,@DATA
```

```

MOV DS, AX
MOV ES, AX
NOP
CALL Init8255
MAIN: CALL START_Temperature ;向DS18B20发送读温度指令
JB MAIN
CALL DelayTime
CALL RD_Temperature ;读出温度值,并转换为BCD码
CALL DIS_BCD ;提取温度数据,转换为非压缩型BCD码,并显示
JMP MAIN
;温度转换/显示
DIS_BCD PROC NEAR
MOV BX, AX
LEA DI, buffer+7
STD
MOV AL, 10H ;10H表示不需要显示
STOSB
STOSB
STOSB
STOSB
TEST AH, 08H
JNZ DIS_BCD1
STOSB ;正数
JMP DIS_BCD2
DIS_BCD1: MOV AL, 11H
STOSB ;负数
NEG BX
DIS_BCD2: ;将温度整数位转换为ASCII
;将温度的个位与十位合在BH中
SHL BX, 1
SHL BX, 1
SHL BX, 1
SHL BX, 1
MOV AX, 10
XCHG AL, BH
DIV BH
CMP AL, 0
JNZ DIS_BCD3 ;判断温度的十位是否为0进行相应处理
MOV AL, 10H ;十位为0
XCHG AL, [DI+1]
STOSB
JMP DIS_BCD4
DIS_BCD3: STOSB
DIS_BCD4: MOV AL, AH
OR AL, 80H ;小数点

```

```

                                STOSB
                                XOR        AL, AL                ;转换小数部分
                                TEST       BL, 10H
                                JZ         DIS_BCD5
                                MOV        AL, 6
DIS_BCD5:                      TEST       BL, 20H
                                JZ         DIS_BCD6
                                ADD        AL, 12H
                                DAA
DIS_BCD6:                      TEST       BL, 40H
                                JZ         DIS_BCD7
                                ADD        AL, 25H
                                DAA
DIS_BCD7:                      TEST       BL, 80H
                                JZ         DIS_BCD8
                                ADD        AL, 50H
                                DAA
DIS_BCD8:                      MOV        CL, 4
                                ROR        AL, CL
                                AND        AL, 0FH
                                STOSB
                                CLD
                                LEA        SI, buffer            ;显示温度
                                CALL       Display8
                                RET
DIS_BCD                        ENDP
;延时程序
DelayTime                      PROC        NEAR
                                XOR        CX, CX
                                LOOP       $
                                LOOP       $
                                LOOP       $
                                RET
DelayTime                      ENDP
;写 0
W_L                            PROC        NEAR
                                PUSH       AX
                                MOV        DX, Con_8255
                                MOV        AL, 80H
                                OUT        DX, AL
                                POP        AX
                                RET
W_L                            ENDP
;写 1

```



```

W_H      PROC      NEAR
          PUSH      AX
          MOV        DX, Con_8255
          MOV        AL, 01H
          OUT        DX, AL
          POP        AX
          RET
W_H      ENDP
;DS18B20复位初始化子程序
INIT_18B20  PROC      NEAR
          CALL      W_L      ;主机发出501us复位低脉冲
          MOV        CX, 136
          LOOP       $
          MOV        DX, Con_8255
          MOV        AX, 89H
          OUT        DX, AL      ;PC输入状态
          DEC        DX
          DEC        DX
          MOV        CX, 15
INIT_18B20_1: IN      AL, DX
          TEST       AL, 01H
          JZ         INIT_18B20_2
          LOOP       INIT_18B20_1
          STC        ;置位标志位，表示DS18B20不存在
          RET
INIT_18B20_2: MOV      CX, 136
          LOOP       $
          CLC        ;复位标志位, 表示DS18B20存在
          RET
INIT_18B20  ENDP
;写操作
WRITE_18B20  PROC      NEAR
          MOV        CX, 8      ;一共8位数据
WRI:        PUSH     AX      ;0->PC0      CALL      W_L
          MOV        DX, Con_8255
          MOV        AL, 80H
          OUT        DX, AL
          POP        AX
          ROR        AL, 1
          JNB        WRI1
          PUSH     AX      ;1->PC0      CALL      W_H
          MOV        DX, Con_8255
          MOV        AL, 01H
          OUT        DX, AL

```

```

WRI2:      POP      AX
           PUSH     CX                ;延时55us
           MOV      CX, 7
           LOOP     $
           POP      CX
           CALL     W_H
           LOOP     WRI
           RET

WRI1:      PUSH     CX
           NOP
           POP      CX
           JMP      WRI2

WRITE_18B20 ENDP
;读操作
READ_18B20 PROC    NEAR
MOV        CX, 8                ;数据一共有8位
Read:      MOV      DX, Con_8255
           MOV      AL, 80H
           OUT      DX, AL        ;0->PC0
           MOV      AL, 89H
           OUT      DX, AL        ;输入状态
           NOP
           NOP
           NOP
           NOP
           MOV      DX, PC_8255
           IN       AL, DX
           ROR      AL, 1
           RCR      BL, 1
           PUSH     CX
           MOV      CX, 11
           LOOP     $
           POP      CX
           LOOP     Read
           MOV      AL, BL
           RET

READ_18B20 ENDP
; 判断DS18B20是否存在, 启动DS18B20      ;CY为判断标志
START_Temperature: CALL    INIT_18B20      ;先复位DS18B20
                   JB      GET_T
                   MOV      AL, 0CCH        ;跳过ROM匹配
                   CALL     WRITE_18B20
                   MOV      AL, 44H        ;发出温度转换命令

```

```

                CALL        WRITE_18B20
                CLC
GET_T:          RET
; 读出转换后的温度值, 存在AX
RD_Temperature: CALL        INIT_18B20           ;准备读温度前先复位
                MOV         AL, 0CCH           ;跳过ROM匹配
                CALL        WRITE_18B20
                MOV         AL, 0BEH           ;发出读温度命令
                CALL        WRITE_18B20
                CALL        READ_18B20         ;读出温度
                MOV         AH, AL             ;存放到AX
                CALL        READ_18B20
                XCHG        AL, AH
                RET
Init8255        PROC        NEAR
                MOV         DX, Con_8255
                MOV         AL, 80H
                OUT         DX, AL
                DEC         DX
                DEC         DX
                MOV         AL, 0FFH
                OUT         DX, AL
                RET
Init8255        ENDP

                END         START

```

七、实验扩展及思考题

实验内容：读取 DS18B20 内部 64 位识别码，了解多个 DS18B20 协同工作原理

实验四 步进电机实验

一、实验目的与要求

- 1、了解步进电机的基本原理，掌握步进电机的转动编程方法
- 2、了解影响电机转速的因素有那些

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

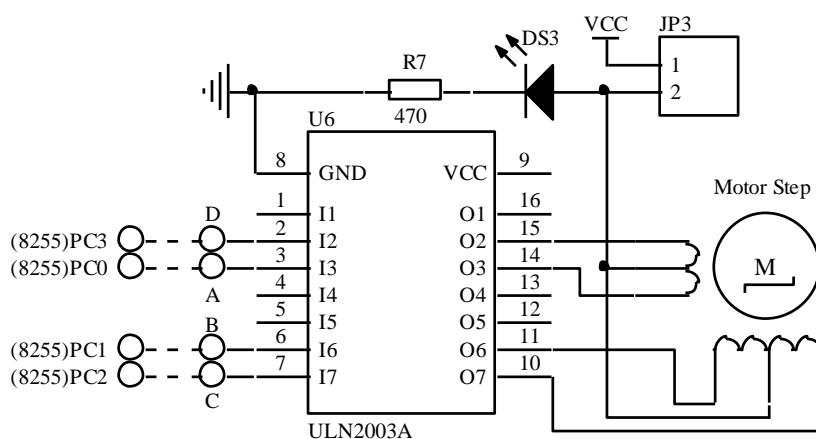
编写程序：使用 F4 区的键盘控制步进电机的正反转、调节转速，连续转动或转动指定步数；将相应的数据显示在 F4 区的数码管上。

四、控制原理

步进电机的驱动原理是通过它每相线圈的电流的顺序切换来使电机作步进式旋转，驱动电路由脉冲来控制，所以调节脉冲的频率便可改变步进电机的转速，微控制器最适合控制步进电机。另外，由于电机的转动惯量的存在，其转动速度还受驱动功率的影响，当脉冲的频率大于某一值（本实验为 $f > 100\text{Hz}$ ）时，电机便不再转动。

实验电机共有四个相位（A, B, C, D），按转动步骤可分单 4 拍（A→B→C→D→A），双 4 拍（AB→BC→CD→DA→AB）和单双 8 拍（A→AB→B→BC→C→CD→D→DA→A）。

五、实验原理图



六、实验步骤

- 1、主机连线说明：

D1 区：A、B、C、D	——	B4 区：PC0、PC1、PC2、PC3
D3 区：CLK	——	B2 区：2M
D3 区：CS、A0	——	A3 区：CS5、A0
B3 区：CS、A0	——	A3 区：CS1、A0
B3 区：INT、INTA	——	EMU598+：INTR、INTA
D5 区：CS(8253)、A0、A1	——	A3 区：CS2、A0、A1
D5 区：GATE0	——	C1 区：VCC
D5 区：CLK0	——	B2 区：1M
D5 区：OUT0	——	B3 区：IR0
B4 区：CS(8255)、A0、A1	——	A3 区：CS3、A0、A1

- 2、调试程序，查看运行结果是否正确

七、演示程序

```

.MODEL TINY
EXTRN Display8:NEAR, SCAN_KEY:NEAR
I08259_0 EQU 0F000H
I08259_1 EQU 0F001H
Con_8253 EQU 0E003H
T0_8253 EQU 0E000H
I08255_Con EQU 0D003H ;CS3
I08255_PC EQU 0D002H
.STACK 100
.DATA
StepControl DB 0 ;下一次送给步进电机的值
buffer DB 8 DUP(0) ;显示缓冲区，8个字节
buffer1 DB 8 DUP(0) ;显示缓冲区，8个字节
SpeedNo DB 0 ;选择哪一级速度
StepDelay DB 0 ;转动一步后，延时常数
StartStepDelay DB 0 ;若选择速度过快，延时由长到短，最终使用对应延时常数
StartStepDelay1 DB 0 ;StartStepDelay
bFirst DB 0 ;有没有转动过步进电机
bClockwise DB 0 ; =1 顺时针方向 =0 逆时针方向转动
bNeedDisplay DB 0 ;已转动一步，需要显示新步数
StepCount DW 0 ;需要转动的步数
StepDelayTab: DB 250, 125, 83, 62, 50, 42, 36, 32, 28, 25, 22, 21
.CODE
START: MOV AX, @DATA
MOV DS, AX
MOV ES, AX
NOP
MOV bFirst, 1 ;有没有转动过步进电机
MOV bClockwise, 1 ;顺时针方向
MOV StepControl, 33H ;下一次送给步进电机的值
MOV SpeedNo, 5 ;第五级速度
CALL Init8255
CALL Init8253
CALL Init8259
CALL WriIntver
MOV buffer, 0 ;显示缓冲器初始化
MOV buffer+1, 0
MOV buffer+2, 0
MOV buffer+3, 0
MOV buffer+4, 10H
MOV AL, SpeedNo
MOV buffer+5, AL
MOV buffer+6, 10H
MOV buffer+7, 0

```

```

STAR2:      LEA      SI,buffer
            LEA      DI,buffer1
            MOV      CX,8
            REP      MOVSB
            LEA      SI,buffer1
            CALL     Display8
STAR3:      CALL     Scan_Key
            JB       STAR5
            CMP      bNeedDisplay,0
            JZ       STAR3
            MOV      bNeedDisplay,0
            CALL     Step_SUB_1
            JMP      STAR2
STAR5:      CLI                      ;终止步进电机转动
            CMP      AL,10
            JNB      STAR1
            MOV      AH,buffer+2
            MOV      buffer+3,AH
            MOV      AH,buffer+1
            MOV      buffer+2,AH
            MOV      AH,buffer
            MOV      buffer+1,AH
            MOV      buffer,AL
            JMP      STAR2
STAR1:      CMP      AL,14
            JNB      STAR3
            LEA      SI,DriverTab
            SUB      AL,10
            SHL      AL,1
            XOR      AH,AH
            MOV      BX,AX
            JMP      CS:[SI+BX]
DriverTab:  DW      Direction          ;转动方向
            DW      Speed_up          ;提高转速
            DW      Speed_Down        ;降低转速
            DW      Exec              ;步进电机根据方向、转速、步数开始转动
Direction:  CMP      bClockwise,0
            JZ       Clockwise
            MOV      bClockwise,0
            MOV      buffer+7,1
AntiClockwise:  CMP      bFirst,0
            JZ       AntiClockwise1
            MOV      StepControl,91H
            JMP      Direction1

```

```

AntiClockwise1:  MOV     AL, StepControl
                  ROR     AL, 2
                  MOV     StepControl, AL
                  JMP     Direction1
Clockwise:       MOV     bClockwise, 1
                  MOV     buffer+7, 0
                  CMP     bFirst, 0
                  JZ      Clockwise1
                  MOV     StepControl, 33H
                  JMP     Direction1
Clockwise1:      MOV     AL, StepControl
                  ROL     AL, 2
                  MOV     StepControl, AL
Direction1:      JMP     STAR2
Speed_up:        MOV     AL, SpeedNo
                  CMP     AL, 11
                  JZ      Speed_up2
Speed_up1:       INC     AL
                  MOV     SpeedNo, AL
                  MOV     buffer+5, AL
Speed_up2:       JMP     STAR2
Speed_Down:      MOV     AL, SpeedNo
                  CMP     AL, 0
                  JZ      Speed_Down1
                  DEC     AL
                  MOV     SpeedNo, AL
                  MOV     buffer+5, AL
Speed_Down1:     JMP     STAR2
Exec:            MOV     bFirst, 0
                  CALL    TakeStepCount
                  LEA     BX, StepDelayTab
                  MOV     AL, SpeedNo
                  XLAT
                  MOV     StepDelay, AL
                  CMP     AL, 50
                  JNB     Exec1
                  MOV     AL, 50
Exec1:           MOV     StartStepDelay, AL
                  MOV     StartStepDelay1, AL
                  STI
                  JMP     STAR2
TIMER0:         PUSH    AX
                  PUSH    DX
                  DEC     StartStepDelay

```

```

JNZ     TIMER0_1
MOV     AL, StartStepDelay1
CMP     AL, StepDelay
JZ      TIMER0_2
DEC     AL
MOV     StartStepDelay1, AL
TIMER0_2: MOV     StartStepDelay, AL
MOV     AL, StepControl
MOV     DX, I08255_PC
OUT     DX, AL
CMP     bClockwise, 0
JNZ     TIMER0_3
ROR     AL, 1
JMP     TIMER0_4
TIMER0_3: ROL     AL, 1
TIMER0_4: MOV     StepControl, AL
CMP     StepCount, 0
JZ      TIMER0_1
MOV     bNeedDisplay, 1
DEC     StepCount
JNZ     TIMER0_1
add     sp, 8 ;小写部分不允许使用单步、单步进入命令
popf
cli
pushf
sub     sp, 8
nop
TIMER0_1: MOV     DX, I08259_0
MOV     AL, 20H
OUT     DX, AL
POP     DX
POP     AX
IRET
Step_SUB_1 PROC    NEAR
MOV     CX, 4
LEA     BX, buffer
Step_SUB_1_1: DEC     BYTE PTR [BX]
CMP     BYTE PTR [BX], 0FFH
JNZ     Step_SUB_1_2
MOV     BYTE PTR [BX], 9
INC     BX
LOOP    Step_SUB_1_1
Step_SUB_1_2: RET
Step_SUB_1 ENDP

```


TakeStepCount	PROC	NEAR	
	MOV	AL, buffer+3	;转动步数送入StepCount
	MOV	BX, 10	
	MUL	BL	
	ADD	AL, buffer+2	
	MUL	BL	
	ADD	AL, buffer+1	
	ADC	AH, 0	
	MUL	BX	
	ADD	AL, buffer	
	ADC	AH, 0	
	MOV	StepCount, AX	
	RET		
TakeStepCount	ENDP		
Init8255	PROC	NEAR	
	MOV	DX, I08255_Con	
	MOV	AL, 80H	
	OUT	DX, AL	;8255 PC输出
	DEC	DX	
	MOV	AL, 0FFH	
	OUT	DX, AL	;0FFH->8255 PC
	RET		
Init8255	ENDP		
Init8253	PROC	NEAR	
	MOV	DX, Con_8253	
	MOV	AL, 35H	
	OUT	DX, AL	;计数器T0设置在模式2状态, BCD码计数
	MOV	DX, T0_8253	
	MOV	AL, 10H	
	OUT	DX, AL	
	MOV	AL, 02H	
	OUT	DX, AL	;CLK0/210
	RET		
Init8253	ENDP		
Init8259	PROC	NEAR	
	MOV	DX, I08259_0	
	MOV	AL, 13H	
	OUT	DX, AL	
	MOV	DX, I08259_1	
	MOV	AL, 08H	
	OUT	DX, AL	
	MOV	AL, 09H	
	OUT	DX, AL	
	MOV	AL, 0FEH	

```

                                OUT    DX, AL
                                RET
Init8259                      ENDP
WriIntver                     PROC    NEAR
                                PUSH    ES
                                MOV     AX, 0
                                MOV     ES, AX
                                MOV     DI, 20H
                                LEA     AX, TIMERO
                                STOSW
                                MOV     AX, CS
                                STOSW
                                POP     ES
                                RET
WriIntver                     ENDP

                                END     START

```

八、实验扩展及思考

- 1、怎样改变电机的转速？
- 2、通过实验找出电机转速的上限，如何能进一步提高最大转速？
- 3、怎样能使电机反转？

实验五 直流电机测速实验

一. 实验目的

了解直流电机工作原理；了解光电开关的原理；掌握使用光电开关测量直流电机转速。

二. 实验设备

STAR 系列实验仪一套、PC 机一台。

三. 实验内容

1、转速测量原理：



图 1 强反射



图 2 弱反射

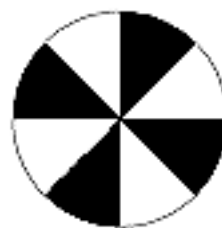


图 3 转盘

本转速测量实验采用反射式光电开关，通过计数转盘通断光电开关产生的脉冲，计算出转速

(1) 反射式光开关工作原理：光电开关发射光，射到测量物体上，如果强反射，如图 1，光电开关接收到反射回来的光，则产生高电平 1；弱反射，如图 2，光电开关接收不到反射回来的光，则产生弱电平 0。

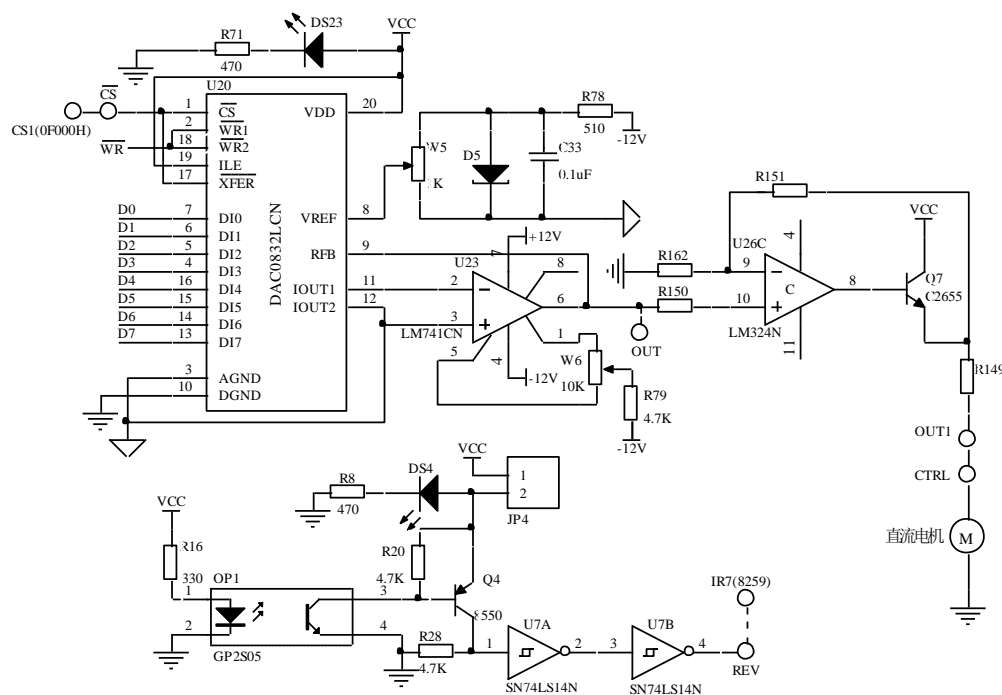
(2) 实验方法：本实验转速测量用的转盘在下表面做成如图 3 样子的转盘，白部分为强反射区，黑部分为弱反射区，转盘每转一圈，产生 4 个脉冲，每 1/4 秒计数出脉冲数，即得到每秒的转速。(演示程序中，LED 显示的是每秒钟转速)

2、实验过程

(1) 由 DAC0832 给电机供电，使用光电开关，测量电机转速，再经调整，最终将转速显示在 LED 上。

(2) 通过按键调节电机转速，随之变化的转速动态显示 LED 上

四. 实验原理图



五、实验步骤

1、主机连线说明：

B3 区：CS、A0	——	A3 区：CS1、A0
B3 区：INT、INTA	——	EMU598+：INTR、INTA
D5 区：CS、A0、A1	——	A3 区：CS2、A0、A1
D5 区：GATE0、GATE1	——	C1 区：VCC
D5 区：CLK0	——	B2 区：31250Hz
D5 区：CLK1	——	B2 区：1M
D5 区：OUT0	——	B3 区：IR0
E2 区：CS	——	A3 区：CS3
E2 区：OUT1	——	E1 区：CTRL
E1 区：REV	——	B3 区：IR1
D3 区：CLK	——	B2 区：2M
D3 区：CS、A0	——	A3 区：CS5、A0

2、由 DAC0832 经功放电路驱动直流电机，计数光电开关通次数并经过换算得出直流电机的转速，并将转速显示在 LED 上。

3、F4 区的 0、1 号按键控制直流电机转速快慢，（最大转速 $\approx 96\text{r/s}$ ，5V，误差 $\pm 1\text{r/s}$ ）

六、演示程序

```

.MODEL          TINY
EXTRN           Display8:NEAR, SCAN_KEY:NEAR, GetKeyA:NEAR
I08259_0         EQU          0F000H
I08259_1         EQU          0F001H
Con_8253         EQU          0E003H
T0_8253         EQU          0E000H
T1_8253         EQU          0E001H
DA0832          EQU          0D000H
VoltageOffset    EQU          5                ;0832 调整幅度
.STACK          200
.DATA
buffer          DB            8 DUP(0)        ;显示缓冲区，8 个字节
buffer1         DB            8 DUP(0)        ;显示缓冲区，8 个字节
VOLTAGE         DB            0                ;转换电压数字量
Count           DW            0                ;一秒转动次数
NowCount        DW            0                ;当前计数值
kpTime          DW            0                ;保存上一次采样时定时器的值
bNeedDisplay    DB            0                ;需要刷新显示
.CODE
START:          MOV          AX, @DATA
                MOV          DS, AX
                MOV          ES, AX
                NOP
                MOV          bNeedDisplay, 1    ;显示初始值
                MOV          VOLTAGE, 99H      ;初始化转换电压输入值，99H-3.0V

```

```

MOV      Count, 0          ;一秒转动次数
MOV      NowCount, 0       ;当前计数值
MOV      kpTime, 0         ;保存上一次采样时定时器的值
CALL     DAC0832           ;初始 D/A
CALL     Init8253
CALL     Init8259
CALL     WriIntver
STI
MAIN:    CALL     GetKeyA    ;按键扫描
JNB      Main1
JNZ      Key1
Key0:    MOV      AL, VoltageOffset;0 号键按下，转速提高
ADD      AL, VOLTAGE
CMP      AL, VOLTAGE
JNB      Key0_1
MOV      AL, 0FFH          ;最大
Key0_1:  MOV      VOLTAGE, AL
CALL     DAC0832           ;D/A
JMP      Main2
Key1:    MOV      AL, VOLTAGE ;1 号键按下，转速降低
SUB      AL, VoltageOffset
JNB      Key1_1
XOR      AL, AL            ;最小
Key1_1:  MOV      VOLTAGE, AL
CALL     DAC0832           ;D/A
JMP      Main2
Main1:   CMP      bNeedDisplay, 0
JZ       MAIN
MOV      bNeedDisplay, 0    ;1s 定时到刷新转速
Main2:   CALL     RateTest   ;计算转速/显示
JMP      MAIN              ;循环进行实验内容介绍与测速功能测试
;转速测量/显示
RateTest: MOV      AX, Count
MOV      BL, 10
DIV      BL
CMP      AL, 0
JNZ      RateTest1
MOV      AL, 10H           ;高位为 0，不需要显示
RateTest1: MOV      buffer, AH
MOV      buffer+1, AL
MOV      AL, VOLTAGE        ;给 0832 送的数据
AND      AL, 0FH
MOV      buffer+4, AL
MOV      AL, VOLTAGE

```

	AND	AL, 0F0H	
	ROR	AL, 4	
	MOV	buffer+5, AL	
	MOV	buffer+2, 10H	;不显示
	MOV	buffer+3, 10H	
	MOV	buffer+6, 10H	
	MOV	buffer+7, 10H	
	LEA	SI, buffer	
	LEA	DI, buffer1	
	MOV	CX, 8	
	REP	MOVSB	
	LEA	SI, buffer	
	CALL	Display8	;显示转换结果
	RET		
Timer0Int:	PUSH	AX	
	PUSH	DX	
	MOV	bNeedDisplay, 1	
	MOV	AX, NowCount	
	SHR	AX, 1	
	SHR	AX, 1	
	MOV	Count, AX	;转一圈产生 4 个脉冲, Count=NowCount/4
	MOV	NowCount, 0	
	MOV	DX, I08259_0	
	MOV	AL, 20H	
	OUT	DX, AL	
	POP	DX	
	POP	AX	
	IRET		
CountInt:	PUSH	AX	
	PUSH	DX	
	MOV	DX, Con_8253	
	MOV	AL, 40H	
	OUT	DX, AL	;锁存
	MOV	DX, T1_8253	
	IN	AL, DX	
	MOV	AH, AL	
	IN	AL, DX	
	XCHG	AL, AH	;T1 的当前值
	XCHG	AX, kpTime	
	SUB	AX, kpTime	
	CMP	AX, 100	
	JB	CountInt1	;前后二次采样时间差小于 100, 判断是干扰
	INC	NowCount	
CountInt1:	MOV	DX, I08259_0	

	MOV	AL, 20H	
	OUT	DX, AL	
	POP	DX	
	POP	AX	
	IRET		
Init8253	PROC	NEAR	
	MOV	DX, Con_8253	
	MOV	AL, 34H	
	OUT	DX, AL	;计数器 T0 设置在模式 2 状态, HEX 计数
	MOV	DX, T0_8253	
	MOV	AL, 12H	
	OUT	DX, AL	
	MOV	AL, 7AH	
	OUT	DX, AL	;CLK0=31250Hz, 1s 定时
	MOV	DX, Con_8253	
	MOV	AL, 74H	
	OUT	DX, AL	;计数器 T1 设置在模式 2 状态, HEX 计数
	MOV	DX, T1_8253	
	MOV	AL, 0FFH	
	OUT	DX, AL	
	MOV	AL, 0FFH	
	OUT	DX, AL	;作定时器使用
	RET		
Init8253	ENDP		
Init8259	PROC	NEAR	
	MOV	DX, I08259_0	
	MOV	AL, 13H	
	OUT	DX, AL	
	MOV	DX, I08259_1	
	MOV	AL, 08H	
	OUT	DX, AL	
	MOV	AL, 09H	
	OUT	DX, AL	
	MOV	AL, 0FCH	
	OUT	DX, AL	
	RET		
Init8259	ENDP		
WriIntver	PROC	NEAR	
	PUSH	ES	
	MOV	AX, 0	
	MOV	ES, AX	
	MOV	DI, 20H	
	LEA	AX, Timer0Int	
	STOSW		

```

                                MOV     AX, CS
                                STOSW
                                LEA     AX, CountInt
                                MOV     DI, 3CH
                                STOSW
                                MOV     AX, CS
                                STOSW
                                POP     ES
                                RET
WriIntver    ENDP
;数模转换, A-转换数字量
DAC0832     PROC     NEAR
                                MOV     DX, DA0832
                                MOV     AL, VOLTAGE
                                OUT     DX, AL
                                RET
DAC0832     ENDP
                                END     START

```

七. 实验扩展及思考题

实验内容：在日光灯或白炽灯下，将转速调节到 25、50、75，观察转盘有什么现象出来

实验六 旋转图形实验

一、实验目的

了解旋转图形如何呈现的原理；加深了解控制在实际中的应用。

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三. 实验内容

1、实验原理

一个旋转中的图形，当每转到一定角度时，被照亮一下，不断重复此过程，就可以看到图形稳定的图像（虽然该图形在旋转中）。如果推迟照亮(推迟时间很短)，就会出现图形在慢慢旋转。

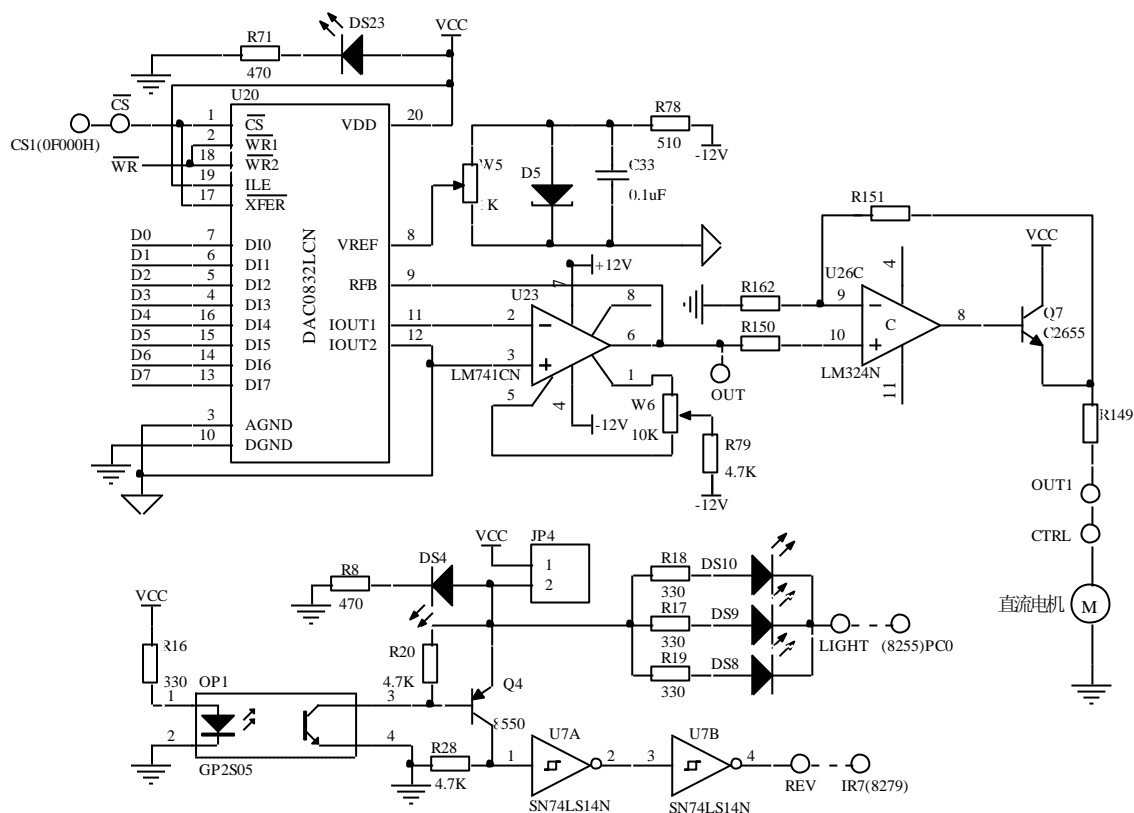
本实验利用光电开关来判断转盘转到某个固定角度的时刻，给出中断信号

2、实验过程

(1) 由光电开关结合中断给出控制信号, 通过 CPU 控制直流电机边上的小灯闪烁, 将直流电机驱动的旋转转盘上的图形呈现出来, 并令图形也在旋转(缓慢的)。

* 该实验类似于：夏天，快速旋转的风扇里能看到缓慢旋转的扇叶的假像。

四. 实验原理图



五. 实验步骤

1、主机连线说明：

B3 $\overline{\text{X}}$: CS、A0	——	A3 $\overline{\text{X}}$: CS1、A0
B3 $\overline{\text{X}}$: INT、INTA	——	EMU598+: INTR、INTA
B3 $\overline{\text{X}}$: IR7	——	E1 $\overline{\text{X}}$: REV
E2 $\overline{\text{X}}$: CS	——	A3 $\overline{\text{X}}$: CS3

E2 区: OUT1	——	E1 区: CTRL
B4 区: CS(8255)、A0、A1	——	A3 区: CS2、A0、A1
B4 区: PC0	——	E1 区: LIGHT

2、检测光电开关产生的中断信号，判断转盘转到固定的角度，计算得出延时，延时过后闪亮(几十微秒)直流电机上的小灯一次(确保小灯照在转盘上)。

3、每转一圈产生 4 个中断，演示程序运行后，可以看到转盘中的图形在转动。

六. 演示程序

```
.MODEL      TINY
I08259_0    EQU      0F000H
I08259_1    EQU      0F001H
Io8255_Con  EQU      0E003H      ;8255 控制口
Io8255_PC   EQU      0E002H      ;PC. 0 控制小灯
DA0832      EQU      0D000H      ;0832 输出口地址
Voltage     EQU      085H        ;直流电机输入电压，85-->50r/s

.STACK      100
.DATA
Count       DW      0            ;计数值缓冲器
FlashFlag   DB      0            ;是否点亮小灯标志

.CODE
START:      MOV      AX, @DATA
            MOV      DS, AX
            MOV      ES, AX
            nop
            MOV      Count, 1
            MOV      FlashFlag, 0      ;不点亮小灯
            CALL     Init8255          ;初始化 8255, 熄灭小灯
            CALL     Init8259
            CALL     WriIntver
            CALL     DAC0832
            STI
MAIN:        CMP      FlashFlag, 0      ;是否点亮小灯
            JZ        MAIN
            MOV      FlashFlag, 0
            CALL     FlashLED
            JMP       MAIN

;点灯中断标志服务程序
Int_LED:     PUSH     AX
            PUSH     DX
            MOV      FlashFlag, 1      ;发光标志，1 有效
            MOV      DX, I08259_0
            MOV      AL, 20H
            OUT      DX, AL
            POP      DX
```

```

        POP        AX
        IRET

;点亮小灯
FlashLED    PROC    NEAR
        CALL      DelayLED
        MOV       DX, Io8255_PC
        MOV       AL, 0
        OUT       DX, AL                ;点亮小灯
        CALL      DelayTime
        MOV       DX, Io8255_PC
        MOV       AL, 0FFH
        OUT       DX, AL
        RET                          ;熄灭小灯
FlashLED    ENDP

;小灯点亮前要延时，产生旋转中的图形在旋转的效果
DelayLED    PROC    NEAR
        MOV       CX, Count            ;每次发光前的延时，由 Count 内的值决定
DelayLED1:   CALL      Delay            ;延时
        LOOP     DelayLED1
        INC       Count
        MOV       AX, Count
        CMP       AX, 200
        JNZ       DelayLED2
        MOV       Count, 1            ;延时达到转盘转过 1/4 圈的时间, 要重新设定
DelayLED2:   RET
DelayLED    ENDP
;延时
Delay        PROC    NEAR
        PUSH     CX
        MOV      CX, 1
        LOOP     $
        POP      CX
        RET
Delay        ENDP
;小灯闪亮的时间, 控制图形的亮度和清晰度
DelayTime    PROC    NEAR
        PUSH     CX
        MOV      CX, 48
        LOOP     $
        POP      CX
        RET
DelayTime    ENDP
Init8259     PROC    NEAR
        MOV      DX, Io8259_0

```

```

MOV        AL, 13H
OUT        DX, AL
MOV        DX, I08259_1
MOV        AL, 08H
OUT        DX, AL
MOV        AL, 09H
OUT        DX, AL
MOV        AL, 7FH
OUT        DX, AL
RET
Init8259   ENDP
WriIntver  PROC    NEAR
PUSH       ES
MOV        AX, 0
MOV        ES, AX
MOV        DI, 3CH
LEA        AX, Int_LED
STOSW
MOV        AX, CS
STOSW
POP        ES
RET
WriIntver  ENDP
;设置直流电机供电电压
DAC0832    PROC    NEAR
MOV        DX, DA0832
MOV        AL, Voltage
OUT        DX, AL
RET
DAC0832    ENDP
Init8255    PROC    NEAR
MOV        DX, Io8255_Con
MOV        AL, 80H
OUT        DX, AL                ;PC 口输出
MOV        AL, 0FFH
DEC        DX
DEC        DX
OUT        DX, AL                ;1->PC.0
RET
Init8255    ENDP
END        START

```

七. 实验扩展及思考题

实验内容：上面实验中的图形的旋转方向是同向的，通过控制小灯闪烁时机，可以实现旋转中的图形反向旋转，有兴趣者可自行尝试

实验七 ISD1110 语音模块实验

一、实验目的

了解 ISD1110 的性能；与 8088 的接口逻辑；掌握手动和 CPU 控制两种录音、放音的基本功能。

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

1、ISD1110 语音模块 (B1 区)：

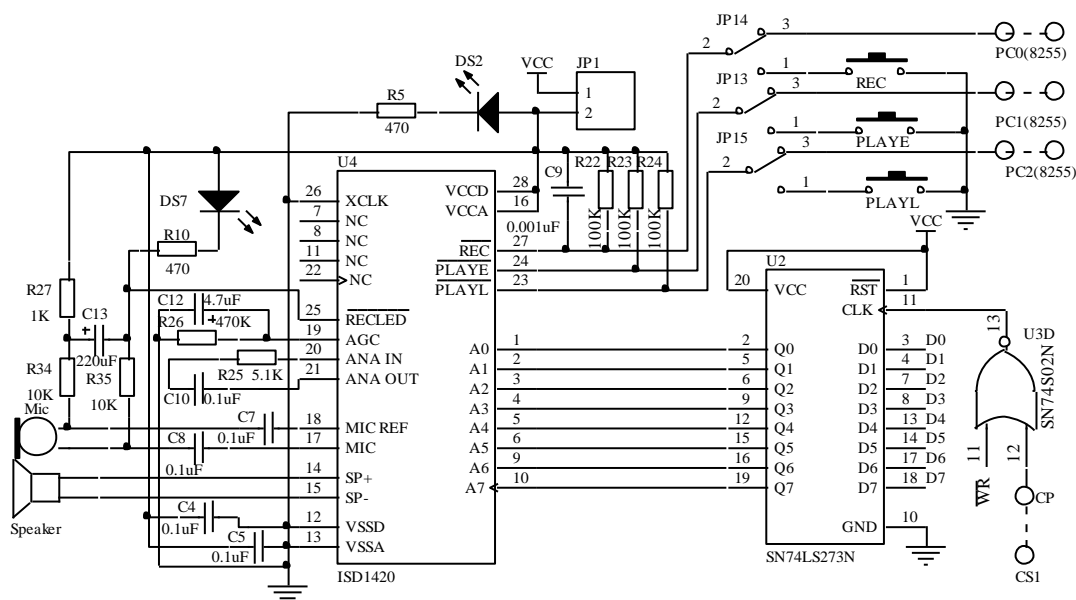
(1) 10 秒录放音长度，具有不掉电存储功能；(2) 可分 1—160 段录放音片段

2、具体操作

(1) 手动控制方式，通过 B1 区按键 REC 录音和按键 PLAYE、PLAYL 放音

(2) MCU 控制方式，通过 F5 区 8 个按键控制录、放音：1~4 号键录音各 5 秒；然后通过 5~8 号键放音，放音内容顺序对应 1~4 号键的录音内容

四、实验原理图



五、实验步骤

1、主机连线说明：

B1 区：REC	——	B4 区：PC0 (8255) 录音控制
B1 区：PLAYE	——	B4 区：PC1 (8255) 电平放音控制
B1 区：PLAYL	——	B4 区：PC2 (8255) 触发放音控制，下降沿触发
B1 区：CP	——	A3 区：CS1
B4 区：CS (8255)、A0、A1	——	A3 区：CS2、A0、A1
B4 区：JP56 (8255 的 PA 口)	——	F5 区：JP74

2、将 JP13, JP14, JP15 跳向 “MCU”，8088 控制，运行演示程序，1~4 号键录音，5~8 号键放音。

六、演示程序

.MODEL TINY

```

        . STACK      100                ;堆栈段
ISD1420_AD1 EQU      00H                ;1号键录放音起始地址, 每次录音5s
ISD1420_AD2 EQU      28H                ;2号键录放音起始地址
ISD1420_AD3 EQU      50H                ;3号键录放音起始地址
ISD1420_AD4 EQU      78H                ;4号键录放音起始地址
ISDCOMM EQU      0F000H ;录放音地址/操作模式输入地址, 0F000H接CS1
I8255_Ctr EQU      0E003H                ;8255控制端口地址
I8255_PA EQU      0E000H                ;键盘数据输入口
I8255_PC EQU      0E002H                ;ISD1420控制输出口
. DATA
KeepMode DB      7                ;保存REC、PLAYE、PLAYL当前值
bNewKey DB      0                ;有键按下标志位, 清0-无键按下
KEYno DB      0
KeyTab DW      KEY1, KEY2, KEY3, KEY4, KEY5, KEY6, KEY7, KEY8 ;录音键放音键子程序入口
. CODE                ;程序段
START: MOV      AX, @DATA
        MOV      DS, AX
        NOP
        CALL     MainInit          ;主程序初始化
Main:   CALL     ScanKey            ;扫描按键
        JNB      Main
Main1:  CALL     KeyRun              ;按键处理
        CMP      bNewKey, 0        ;是否有新的键按下
        JZ       Main
        MOV      bNewKey, 0        ;清按键标志
        JMP      Main1            ;循环进行实验内容介绍与ISD1420功能测试
;主程序初始化
MainInit PROC      NEAR
        MOV      bNewKey, 0        ;有键按下标志位, 清0-无键按下
        MOV      DX, I8255_Ctr    ;8255初始化
        MOV      AL, 90H          ;PA为输入, PC的低四位为输出
        OUT      DX, AL
        CALL     ISD_INIT
        RET
MainInit ENDP
;录放音子程序
KEY1 PROC      NEAR
        MOV      AL, ISD1420_AD1  ;1号键录音首地址
        CALL     KEY_REC
        RET
KEY1 ENDP
KEY2 PROC      NEAR
        MOV      AL, ISD1420_AD2  ;2号键录音首地址
        CALL     KEY_REC

```

	RET		
KEY2	ENDP		
KEY3	PROC	NEAR	
	MOV	AL, ISD1420_AD3	;3号键录音首地址
	CALL	KEY_REC	
	RET		
KEY3	ENDP		
KEY4	PROC	NEAR	
	MOV	AL, ISD1420_AD4	;4号键录音首地址
	CALL	KEY_REC	
	RET		
KEY4	ENDP		
;录音子程序			
KEY_REC	PROC	NEAR	
	MOV	CX, 20	;录音时间长度, 5s
	CALL	ISD_REC	;调用录音子程序
KEY_REC1:	CALL	Delay_025S	;延时
	CMP	bNewKey, 0	;检测按键是否有键按下
	JNZ	KEY_REC2	
	LOOP	KEY_REC1	;录音时间, 根据CX的值决定
	CALL	ISD_STOP	;停止录音
KEY_REC2:	RET		
KEY_REC	ENDP		
;放音子程序			
KEY5	PROC	NEAR	
	MOV	AL, ISD1420_AD1	;5号键放音首地址
	CALL	KEY_PLAY	
	RET		
KEY5	ENDP		
KEY6	PROC	NEAR	
	MOV	AL, ISD1420_AD2	
	CALL	KEY_PLAY	
	RET		
KEY6	ENDP		
KEY7	PROC	NEAR	
	MOV	AL, ISD1420_AD3	;7号键放音首地址
	CALL	KEY_PLAY	
	RET		
KEY7	ENDP		
KEY8	PROC	NEAR	
	MOV	AL, ISD1420_AD4	;8号键放音首地址
	CALL	KEY_PLAY	
	RET		
KEY8	ENDP		

```

KEY_PLAY      PROC
MOV           CX, 20
CALL         ISD_PLAY      ;调用录音子程序
KEY_PLAY1:    CALL         Delay_025S    ;用于进度显示的时间参照
CMP          bNewKey, 0
JNZ          KEY_PLAY2     ;检测按键是否有键按下
LOOP         KEY_PLAY1
KEY_PLAY2:    RET
KEY_PLAY      ENDP
KeyRun        PROC      NEAR
LEA          BX, KeyTab    ;有键按下, 跳到相应处理程序
MOV          AL, KEYno     ;KEYno--按键值
SHL          AL, 1         ;×2倍
XOR          AH, AH
ADD          BX, AX
CALL         [BX]          ;[BX]=对应按键子程序入口地址
RET
KeyRun        ENDP
;按键扫描
ScanKey       PROC      NEAR
MOV          DX, I8255_PA  ;8255. PA----检测按键输入
IN           AL, DX        ;键扫描
CMP          AL, 0FFH
JNZ          ScanKey1
ScanKey4:     CLC           ;无键按下
RET
ScanKey1:     CALL         ScanKey2     ;有按键, 取抖动处理
JNB          ScanKey4
ScanKey3:     MOV          BL, KEYno
CALL         Delay20ms      ;消抖动
CALL         Delay20ms
CALL         ScanKey2
JNB          ScanKey4
CMP          BL, KEYno
JNZ          ScanKey3
ScanKey5:     MOV          DX, I8255_PA
ScanKey6:     IN           AL, DX
CMP          AL, 0FFH
JNZ          ScanKey6
STC
RET
ScanKey       ENDP
;按下的键(1~8)转化为对应的键值(0~7), 便于多分支程序处理
ScanKey2      PROC      NEAR

```


	PUSH	BX	
	XOR	BL, BL	
	MOV	DX, I8255_PA	
	IN	AL, DX	
	TEST	AL, 01H	
	JZ	ScanKey21	
	INC	BL	
	TEST	AL, 02H	
	JZ	ScanKey21	
	INC	BL	
	TEST	AL, 04H	
	JZ	ScanKey21	
	INC	BL	
	TEST	AL, 08H	
	JZ	ScanKey21	
	INC	BL	
	TEST	AL, 10H	
	JZ	ScanKey21	
	INC	BL	
	TEST	AL, 20H	
	JZ	ScanKey21	
	INC	BL	
	TEST	AL, 40H	
	JZ	ScanKey21	
	INC	BL	
	TEST	AL, 80H	
	JZ	ScanKey21	
	CLC		
	JMP	ScanKey22	
ScanKey21:	STC		;有键按下，置有键按下标志
	MOV	KEYno, BL	;获得键值
ScanKey22:	POP	BX	
	RET		
ScanKey2	ENDP		
;延时			
Delay20ms	PROC	NEAR	
	PUSH	CX	
	MOV	CX, 2640	
	LOOP	\$	
	POP	CX	
	RET		
Delay20ms	ENDP		
;延时0.25s（兼有键盘检测功能）			
Delay_025S	PROC	NEAR	

```

                PUSH    AX
                PUSH    CX
                PUSH    DX
                MOV     CX, 33000
                LOOP    $
                MOV     CX, 33000
                LOOP    $
                CALL    ScanKey
                JNB     DL1S_2
                MOV     bNewKey, 1
DL1S_2:         POP     DX
                POP     CX
                POP     AX
                RET
Delay_025S     ENDP
;录音子程序
;AL--存放操作方式设置值, CX--录几秒
ISD_INIT      PROC    NEAR
                MOV     DX, I8255_PC
                MOV     AL, KeepMode
                OR       AL, 7                ;语音模块初始化, 关闭录放音功能
                OUT     DX, AL
                MOV     KeepMode, AL
                MOV     DX, ISDCOMM
                XOR     AL, AL
                OUT     DX, AL                ;允许手动录放音, 当A6, A7为高时, 无法手动放音
                RET
ISD_INIT      ENDP
;操作模式, AL-操作模式设置值
ISD_MODE      PROC    NEAR
                PUSH    AX
                CALL    ISD_STOP            ;初始化, REC, PLAYE, PLAYL置位, 设置操作模式
                MOV     DX, ISDCOMM        ;设置操作模式:分段录音
                POP     AX
                OUT     DX, AL                ;设置操作模式命令在AL中
                MOV     DX, I8255_PC
                MOV     AL, KeepMode
                AND     AL, 0FBH
                OUT     DX, AL
                OR       AL, 4
                OUT     DX, AL                ;给PLAYL一个上升沿, 锁存命令
                MOV     KeepMode, AL
                RET
ISD_MODE      ENDP

```

```

;录音
ISD_REC      PROC      NEAR
MOV          DX, ISDCOMM
OUT          DX, AL      ;设置录音起始地址
MOV          DX, I8255_PC
MOV          AL, KeepMode
AND          AL, 0FEH
OUT          DX, AL      ;REC变低，即开始录音
MOV          KeepMode, AL
RET
ISD_REC      ENDP

;放音子程序
;AL--放哪段音
ISD_PLAY      PROC      NEAR
PUSH         AX
CALL         ISD_STOP    ;暂停之前的录放音操作
POP          AX
MOV          DX, ISDCOMM ;设置放音起始地址
OUT          DX, AL
MOV          DX, I8255_PC
MOV          AL, KeepMode
AND          AL, 0FDH
OUT          DX, AL      ;0->PLAYE 开始放音, 边沿放音模式
OR           AL, 2
OUT          DX, AL      ;1->PLAYE
MOV          KeepMode, AL
RET
ISD_PLAY      ENDP

;停止录放音
ISD_STOP      PROC      NEAR
MOV          DX, I8255_PC
MOV          AL, KeepMode
AND          AL, 0FBH
OUT          DX, AL      ;PLAYL: 一个负脉冲停止放音
OR           AL, 4
OUT          DX, AL
CALL         Delay50ms
OR           AL, 3        ;1->REC, PLAYE
OUT          DX, AL      ;关闭所有操作指令
MOV          KeepMode, AL
MOV          DX, ISDCOMM
XOR          AL, AL
OUT          DX, AL      ;允许手动录放音, 当A6, A7为高时，无法手动放音
RET

```

```

ISD_STOP      ENDP
;延时
Delay50ms     PROC      NEAR
                PUSH     CX
                MOV      CX, 13200
                LOOP     $
                POP      CX
                RET
Delay50ms     ENDP

                END      START

```

七. 实验扩展及思考题

实验名称：公交车的报站功能

实验内容：利用掌握分段录音和放音控制，实现公交车的报站功能，有兴趣者可自行尝试

实验八 恒温控制实验

一、实验目的

了解温度控制的概念，初步接触控制在实际中的应用

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

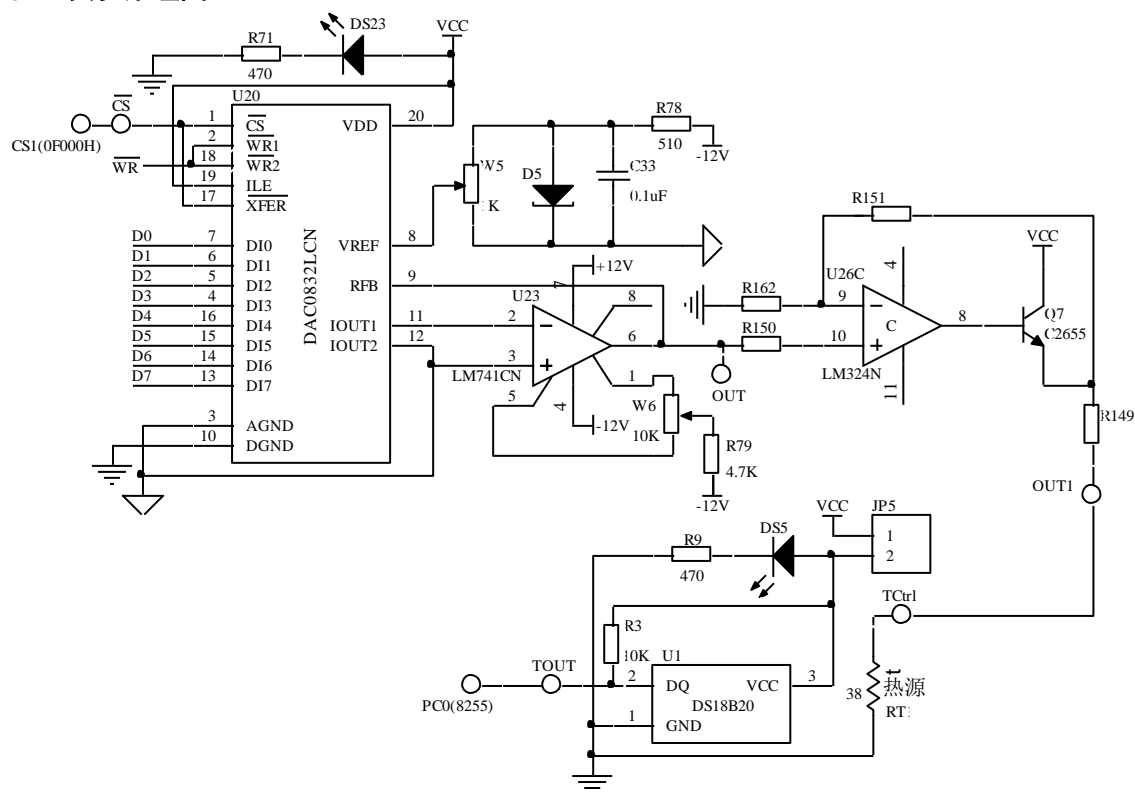
1、温度控制原理

- (1) 通过 DS18B20 测量温度
- (2) 通过 DAC0832 输出电压供给发热电阻 RT1 控制温度
- (3) 比较测量温度和设置温度(要求达到的温度)，判断控制温度增减方向，通过 DAC0832 控制 RT1 的发热量逐渐控制温度达到设置温度

2. 实验过程

- (1) 使用 DS18B20 制作一个数字温度计，使用 LED 显示温度
- (2) 使用键盘随意设定某一温度，通过 DAC0832 输出电压供给发热电阻 RT1，使发热电阻 RT1 固定在该温度。

四、实验原理图



五、实验步骤

1、主机连线说明：

E4 区：TOUT	——	B4 区：PC0 (8255)
E4 区：TCtrl	——	E2 区：OUT1
B4 区：CS、A0、A1	——	A3 区：CS1、A0、A1

D3 区: CS、A0	——	A3 区: CS5、A0
D3 区: CLK	——	B2 区: 2M
E2 区: CS	——	A3 区: CS2

2、温度控制方式

(1) 使用 DS18B20 测量温度，将测量温度与设置温度比较，得出温差，以此调整 DAC0832 的输出电压，从而控制发热电阻的发热量。

(2) 调整 DAC0832 的输出电压：

如果温差超过 10℃，调节 DAC0832 的输出电压，使每秒变化不小于 0.5℃；

如果温差超过 5℃，调节 DAC0832 的输出电压，使每秒变化不小于 0.2℃；

如果温差超过 2℃，调节 DAC0832 的输出电压，使每秒变化在 0.1℃~0.2℃；

如果温差小于 2℃，调节 DAC0832 的输出电压，使每秒变化在 0.1℃范围内。

温度对发热电阻的变化，反应会滞后，在开始加温、开始降温时，应考虑变化趋势。

(3) 循环上述过程，逐渐控制达到所设置的温度。此过程中，当前温度动态显示。

(4) 设置控制温度的范围：室温~最大加热温度，设置在室温+10℃为宜。

3. 实验过程中，将会看到

(1) 通过 DS18B20 测量的当前温度(十进制)会显示在 LED 上(右边 4 位)

(2) 通过 16 键键盘(F4 区)输入设置温度，同时 LED(左边 3 位)会显示输入的键值(十进制)

(3) 测量温度会逐渐向设置温度靠近，最终在设置温度附近浮动。

*遮掩 DS18B20 (E4 区)，减小外界影响，可以提高控制精度

六、演示程序

略

七、实验扩展及思考题

实验内容：上面实验采用改变 D/A 供电大小的方法控制，还有采用控制通断热源供电时间的方法，控制温度，有兴趣者可自行尝试。

实验九 电子钟 (CLOCK)

一、实验目的

进一步熟悉 8253、8259、8279

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

- 1、使用 8253 定时功能，产生 0.5S 的定时中断给 8259
- 2、在 F4 区的数码管上显示时间
- 3、允许设置时钟初值

四、实验步骤

- 1、主机连线说明：

D3 区 : CLK	——	B2 区: 2M
D3 区 : CS、A0	——	A3 区: CS5、A0
B3 区 : CS、A0	——	A3 区: CS1、A0
B3 区 : INT、INTA	——	EMU598+: INTR、INTA
B3 区 : IR0	——	D5 区 : OUT0
D5 区 : CS (8253)、A0、A1	——	A3 区: CS2、A0、A1
D5 区 : GATE0	——	C1 区: VCC
D5 区 : CLK0	——	B2 区: 62.5K

- 2、运行程序，按 F4 区的 F 键，设置时钟初值；
- 3、观察 F4 区数码管上显示的时间是否正确。

五、演示程序

```

.MODEL          TINY
EXTRN           Display8:NEAR, GetKeyA:NEAR, GetKeyB:NEAR
I08259_0        EQU          0F000H
I08259_1        EQU          0F001H
Con_8253        EQU          0E003H
T0_8253         EQU          0E000H

.STACK          200

.DATA

halfsec         DB           0                ;0.5秒计数
Sec              DB           0                ;秒
Min              DB           0                ;分
hour             DB           0                ;时
buffer           DB           8 DUP (0)        ;显示缓冲区, 8个字节
buffer1          DB           8 DUP (0)        ;显示缓冲区, 8个字节
bNeedDisplay     DB           0                ;需要刷新显示
number           DB           0                ;设置哪一位时间

```

bFlash	DB	0	;设置时是否需要刷新
	. CODE		
START:	MOV	AX, @DATA	
	MOV	DS, AX	
	MOV	ES, AX	
	NOP		
	mov	sec, 0	;时分秒赋初值23:58:00
	mov	min, 58	
	mov	hour, 23	
	MOV	bNeedDisplay, 1	;显示初始值
	CALL	Init8253	
	CALL	Init8259	
	CALL	WriIntver	
	STI		
MAIN:	CALL	GetKeyA	;按键扫描
	JNB	Main1	
	CMP	AL, 0FH	;设置时间
	JNZ	Main1	
	CALL	SetTime	
Main1:	CMP	bNeedDisplay, 0	
	JZ	MAIN	
	CALL	Display_LED	;显示时分秒
	MOV	bNeedDisplay, 0	;1s定时到刷新转速
Main2:	JMP	MAIN	;循环进行实验内容介绍与测速功能测试
SetTime	PROC	NEAR	
	LEA	SI, buffer1	
	CALL	TimeToBuffer	
	MOV	Number, 0	
Key:	CMP	bFlash, 0	
	JZ	Key2	
	LEA	SI, buffer1	
	LEA	DI, buffer	
	MOV	CX, 8	
	REP	MOVSB	
	CMP	halfsec, 0	
	JNZ	FLASH	
	MOV	BL, number	
	NOT	BL	
	AND	BX, 07H	
	LEA	SI, buffer	
	MOV	BYTE PTR [SI+BX], 10H	;当前设置位置产生闪烁效果
FLASH:	LEA	SI, buffer	
	CALL	Display8	
	MOV	bFlash, 0	

Key2:	CALL	GetKeyA	
	JNB	Key	
	CMP	AL, 0EH	;放弃设置
	JNZ	Key1	
	JMP	Exit	
Key1:	CMP	AL, 0FH	
	JZ	SetTime8	
SetTime1:	CMP	AL, 10	
	JNB	Key	;无效按键
	CMP	number, 0	
	JNZ	SetTime2	
	CMP	AL, 3	;调整时的十位数
	JNB	Key	
	MOV	buffer1 + 7, AL	
	JMP	SetTime7	
SetTime2:	CMP	number, 1	
	JNZ	SetTime3	
	CMP	buffer1 + 7, 1	;调整时的个位数
	JZ	SetTime2_1	
	CMP	AL, 4	
	JNB	Key	
SetTime2_1:	MOV	buffer1 + 6, AL	
	INC	number	
	JMP	SetTime7	
SetTime3:	CMP	number, 3	
	JNZ	SetTime4	
	CMP	AL, 6	;调整分的十位数
	JNB	Key	
	MOV	buffer1 + 4, AL	
	JMP	SetTime7	
SetTime4:	CMP	number, 4	
	JNZ	SetTime5	
	MOV	buffer1 + 3, AL	;调整分的个位数
	INC	number	
	JMP	SetTime7	
SetTime5:	CMP	number, 6	
	JNZ	SetTime6	
	CMP	AL, 6	;调整秒的十位数
	JB	SetTime5_1	
	JMP	Key	
SetTime5_1:	MOV	buffer1 + 1, AL	
	JMP	SetTime7	
SetTime6:	MOV	buffer1, AL	;调整秒的个位数
SetTime7:	INC	number	

```

                                CMP        number, 8
                                JNB        SetTime8
                                MOV        bFlash, 1                ;需要刷新
                                JMP        Key
SetTime8:                     MOV        AL, buffer1 + 1            ;确认
                                MOV        BL, 10
                                MUL        BL
                                ADD        AL, buffer1
                                MOV        sec, AL                ;秒
                                MOV        AL, buffer1 + 4
                                MUL        BL
                                ADD        AL, buffer1 + 3
                                MOV        min, AL                ;分
                                MOV        AL, buffer1 + 7
                                MUL        BL
                                ADD        AL, buffer1 + 6
                                MOV        hour, AL                ;时
                                JMP        Exit
Exit:                          RET
SetTime                        ENDP
;hour min sec转化成可显示格式
TimeToBuffer    PROC          NEAR
                                MOV        AL, sec
                                XOR        AH, AH
                                MOV        BL, 10
                                DIV        BL
                                MOV        [SI], AH
                                MOV        [SI + 1], AL            ;秒
                                MOV        BYTE PTR [SI + 2], 10H ;这位不显示
                                MOV        AL, min
                                XOR        AH, AH
                                DIV        BL
                                MOV        [SI + 3], AH
                                MOV        [SI + 4], AL            ;分
                                MOV        BYTE PTR [SI + 5], 10H ;这位不显示
                                MOV        AL, hour
                                XOR        AH, AH
                                DIV        BL
                                MOV        [SI + 6], AH
                                MOV        [SI + 7], AL            ;时
                                RET
TimeToBuffer    ENDP
;显示时分秒
Display_LED     PROC          NEAR

```

```

                                LEA      SI, buffer
                                CALL     TimeToBuffer
                                LEA      SI, buffer
                                CALL     Display8           ;显示
                                RET
Display_LED      ENDP
;0.5s产生一次中断
Timer0Int:      PUSH     AX
                PUSH     DX
                MOV      bFlash, 1
                INC      halfsec
                CMP      halfsec, 2
                JNZ      Timer0Int1
                MOV      bNeedDisplay, 1
                MOV      halfsec, 0
                INC      sec
                CMP      sec, 60
                JNZ      Timer0Int1
                MOV      sec, 0
                INC      min
                CMP      min, 60
                JNZ      Timer0Int1
                MOV      min, 0
                INC      hour
                CMP      hour, 24
                JNZ      Timer0Int1
                MOV      hour, 0
Timer0Int1:     MOV      DX, I08259_0
                MOV      AL, 20H
                OUT      DX, AL
                POP      DX
                POP      AX
                IRET
Init8253      PROC      NEAR
                MOV      DX, Con_8253
                MOV      AL, 34H
                OUT      DX, AL           ;计数器T0设置在模式2状态, HEX计数
                MOV      DX, T0_8253
                MOV      AL, 12H
                OUT      DX, AL
                MOV      AL, 7AH
                OUT      DX, AL         ;CLK0=62.5kHz, 0.5s定时
                RET
Init8253      ENDP

```

Init8259	PROC	NEAR
	MOV	DX, IO8259_0
	MOV	AL, 13H
	OUT	DX, AL
	MOV	DX, IO8259_1
	MOV	AL, 08H
	OUT	DX, AL
	MOV	AL, 09H
	OUT	DX, AL
	MOV	AL, 0FEH
	OUT	DX, AL
	RET	
Init8259	ENDP	
WriIntver	PROC	NEAR
	PUSH	ES
	MOV	AX, 0
	MOV	ES, AX
	MOV	DI, 20H
	LEA	AX, Timer0Int
	STOSW	
	MOV	AX, CS
	STOSW	
	POP	ES
	RET	
WriIntver	ENDP	
	END	START

实验十 光敏电阻测量光照强度实验

一、实验目的

- 1、了解如何利用光敏电阻测量光照强度。
- 2、掌握使用光敏电阻和 ADC0809 制作简易的光照强度测量仪。

二、实验设备

STAR 系列实验仪一套、PC 机一台。

三、实验内容

1、光照强度测量原理：

(1) 在实验仪上有两个光敏电阻，其中有一个光敏电阻表面贴上黑胶布，让其不能对光照的变化产生明显的电阻值改变。另外一个光敏电阻没有贴上黑胶布能迅速地感应光照强度的变化。本实验利用光敏电阻测量光照强度，外界光照强度发生变化引起了光敏电阻的阻值改变。

(2) 光敏电阻 R41 与完全相同的光敏电阻 R57（被黑胶布遮住）组成的电桥(图 1)， ΔR_x 为照度变大时电阻的增加值。将电阻变化转换为电桥产生的压差 U_o 的变化，经放大电路将产生的压差 U_o 放大，得到可以被 A/D 转换的电压，整个系统的实际电路图如图 2 所示。

(3) 本实验假设光照强度与产生的压差 U_o 成正比，设定某一标准光照强度与某一压差对应，以此对应关系换算产生其它压差的照度。从而达到测量光照强度的目的。

如果你需要精确测量光照强度，先将电压值与光照强度对应关系制成表，根据测得电压值通过查表，得出光照强度值。

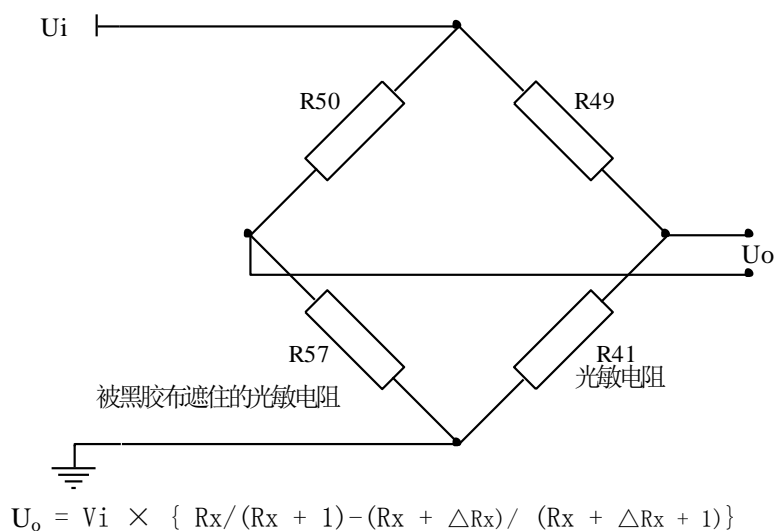


图 1 电压差产生原理图

2、实验过程

(1) 由光照强度产生的模拟电压信号转换为数字信号，然后转换为照度(单位是勒克斯)显示在 LED 上；

(2) 校准照度测量器：在一定的光强度下，产生 200 数字量的电压，以此对应关系（照度—电压）将其它光强度转换为勒克斯值，显示在 LED 上。

