

微机原理实验指导书

华中科技大学人工智能与自动化学院

2019.3.6

0 目录

1 简介.....	1
1.1 实验仪介绍.....	1
1.2 实验仪配置方案.....	错误!未定义书签。
1.3 功能特点.....	1
1.3.1 软件.....	1
1.3.2 硬件.....	2
2 硬件结构.....	3
2.1 电路外观.....	3
2.2 A1 区：12864 液晶显示模块电路.....	4
2.3 A2 区：16×16 LED 实验电路.....	4
2.4 A3 区：系统总线、片选区.....	5
2.5 A4 区：控制区.....	5
2.6 B1 区：语音模块 ISD1420 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：简单 I/O、SRAM 电路.....	9
2.10 B5、C5 区：扩展区.....	10
2.11 C1 区：电源区.....	10
2.12 C2 区：ADC0809 模数转换.....	10
2.13 C3 区：8251 电路.....	10
2.14 C4 区：8253 电路.....	11
2.15 D1 区：步进电机.....	11
2.16 D2 区：DAC0832 数模转换.....	11
2.17 D3 区：8255 电路、数码管驱动电路.....	12
2.18 D4 区：8237 电路.....	13
2.19 D5 区：8250 电路.....	13
2.20 D6 区：X5045.....	14
2.21 D7 区：RS232.....	14
2.22 E1 区：直流电机转速测量/控制.....	14
2.23 E2 区：光敏电阻、压力测量.....	15
2.24 E3 区：继电器.....	15
2.25 E4 区：RS485.....	16
2.26 F1 区：红外通讯.....	16
2.27 F2 区：温度测量/控制.....	16
2.28 F3 区：PWM 电压转换、功率放大电路.....	16
2.29 F4 区：发光管、开关.....	17
2.30 F5 区：键盘&LED.....	18
2.31 F6 区：0~5V 电压输出.....	18
2.32 F7 区：138 译码器.....	18

2. 33 F8 区：蜂鸣器.....	18
3 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)	错误!未定义书签。
3. 3 实验连线、演示实验、测试实验仪.....	34
3. 4 频率计 (EMU86)	35
3. 5 模拟波形发生器 (EMU86)	35
3. 6 TDS2、TDS2A (EMU86) 虚拟示波器.....	36
3. 7 Debug (行命令方式)	37
4 实验.....	39
实验一 数据传送.....	39
实验二 四字节十六进制数转十进制数.....	42
实验三 简单 I/O (16 位) 实验.....	46
实验四 静态存储器 (16 位) 读写实验.....	49
实验五 8255 控制交通灯实验.....	52
实验六 74HC138 译码器实验.....	56
实验七 8253 方波实验.....	58
实验八 8086 中断实验.....	60
实验九 简易电子琴实验.....	64
实验十 LED16 * 16 点阵实验.....	74
实验十一 数字式温度计实验 (18B20)	79
实验十二 步进电机实验.....	85
实验十三 X5045 串行 EEPROM 读写实验.....	92
实验十四 电子钟 (CLOCK)	99
附录 A 星研集成环境软件支持的软中断	105
附录 B 数码管显示、键扫描库程序说明	107

1 简介

1.1 实验仪介绍

微机原理实验采用上海星研电子科技有限公司提供的试验仪 SUN ES86PCIU+。该实验仪提供了全开放的 80x86 系统扩展总线，如：数据总线、地址总线、中断请求信号、HOLD/HLDA、读写控制信号、字节使能信号等，全面支持开放式的微机原理、接口技术各项实验。

1.2 功能特点

1.2.1 软件

实验采用星研集成环境软件，它具有如下特点：

- ◇ 完全 VC++风格。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。不同种类的 CPU 模块间采用软件切换。

- ◇ 在 WindowsXP/Vista/Windows7 的保护模式下，可使用单步、宏单步、全速断点（断点数目没有限制）、全速运行等多种调试手段，调试所有实验程序，特别是 32 位汇编程序；任意查看、修改实验仪上 8/16/32 位 SRAM 中数据、接口芯片内容。

- ◇ 支持 ASM（汇编）、C、PLM 语言，多种语言多模块混合调试：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 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**模拟调试功能**：方便学生在实验前，编写程序；编译、连接，解决语法错误；初步调试；在 Windows XP/VISTA/Windows7 环境中，提供软中断功能。

- ◇ 软件提供一个窗口：**显示当前实验程序对应的全局描述符表、页目录表、页表**，方便教学（全局描述符表、页目录表、页表均由操作系统自动产生）。支持实验仪的**监控在线更新**。

- ◇ 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。

- ◇ 支持拖拉功能（源程序中快速移动、变量名拖至观察窗等）。调试极为快速方便。

1.2.2 硬件

1、主板组成

显示部分	液晶（四种选择） 1）1602C 字符型液晶 1）12864 图形点阵液晶（标配）； 2）带汉字库的 12864 图形点阵液晶； 3）带触摸屏的 12864 液晶	自定义 8 位、 16 位总线	32K*8 位*2 片 SRAM 简单 I/O（2 片 74HC273、2 片 74HC244）
	16X16LED 点阵显示模块	输入、输出	4*4 键盘
	8 位数码管		8 个独立按键
	16 个发光二极管（红、绿、黄色）		8 个拨动开关
			单脉冲
			继电器控制
传统实验	74HC138 译码	AD、DA	可调电压
	8255、打印口		触摸屏（选配）
	8253/8254 可编程定时器/计数器	并行 AD ADC0809	
	8251 串行通信接口	并行 DA DAC0832	
	8250/16C550 串行通信接口	传感器	压力传感器或光敏电阻电路
	8259 可编程中断控制器（可扩展级联）		光电开关电机测速
语音	8237 DMA 控制器		霍尔器件电机测速（选配）
	录音、放音模块 ISD1420	控制对象	18B20 测、控温（闭环）
	自带话筒、喇叭		直流电机控制（闭环）
蜂鸣器	步进电机控制（4 相）		
通信	RS232 和 RS485 接口电路	存贮器	控温模块
	红外通信实验	其它	串行 EEPROM 及看门狗— X5045（SPI）
电源 5V 3A +12V 0.4A -12V 0.4A	PWM 脉宽调制输出接口		
CPU 8086：128K 仿真实空间	功率放大		
扩展区 三个扩展区；可根据客户需求，定制部件	逻辑笔		
辅助功能	74HC4040 分频得到 8 种频率		
	虚拟示波器、信号发生器、频率计		

2、扩展模块

实验主板右侧有三个万用扩展区（右上角是 CPU 扩展区），可根据客户需求，定制部件。

可扩展：GPS 模块；GPRS 模块；超声波模块；电流传感器模块；F/V、V/F 模块；触摸屏；USB1.1 模块；USB 2.0 模块；USB 主控；CAN 2.0 模块；10M 以太网模块；32 位 SRAM、32 位 IO 模块；数据采集模块等。

3、闭环控制

1) 旋转图形展现实验

2) 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

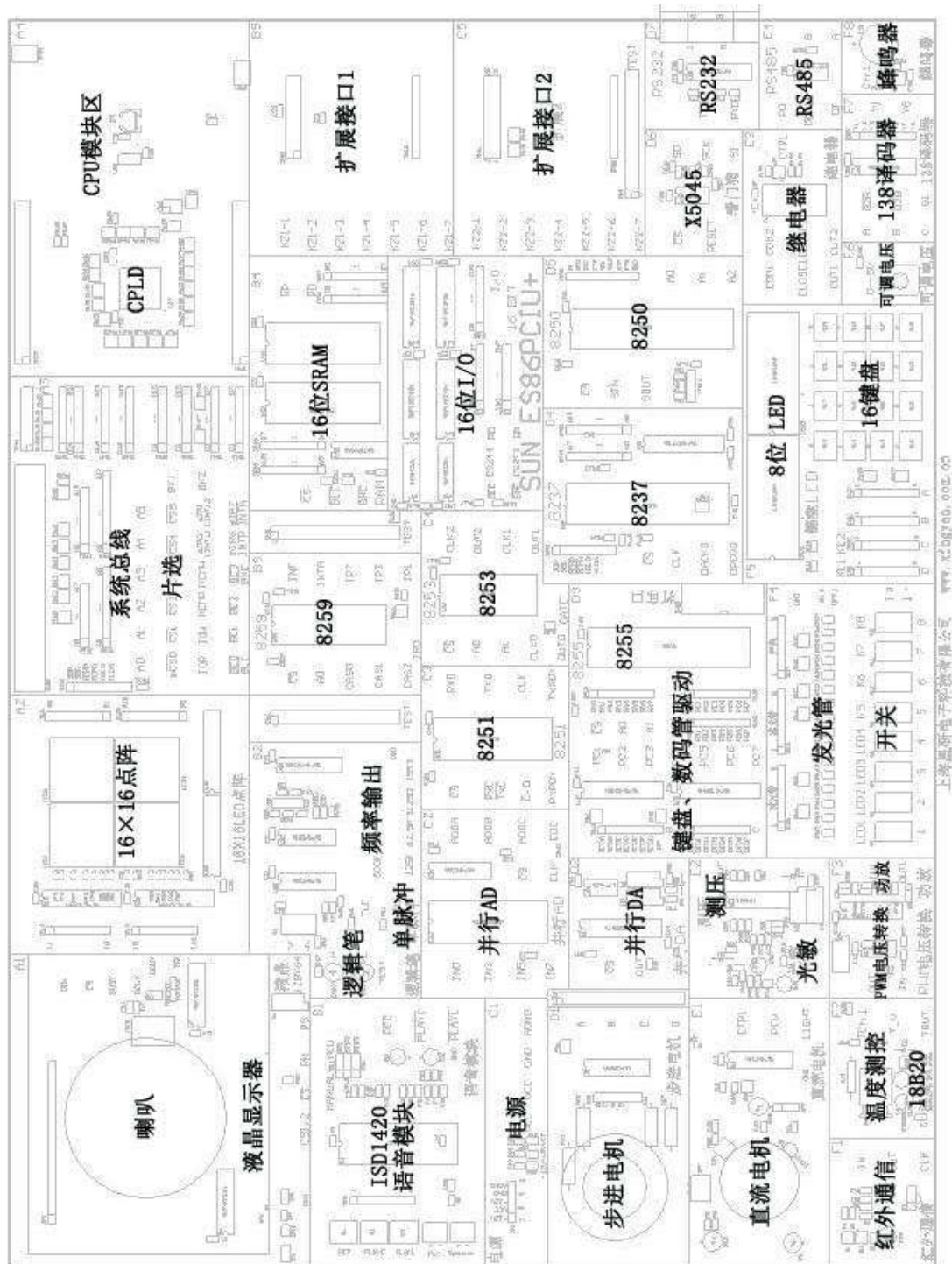
3) 数字式温度控制，通过该实验可较好认识控制在实际中的应用

4、EDA —— CPLD、FPGA 可编程逻辑实验（选配）

可设计 8255、8253、8237 等实验；提供 Verilog、VHDL 语言编写的实验范例。

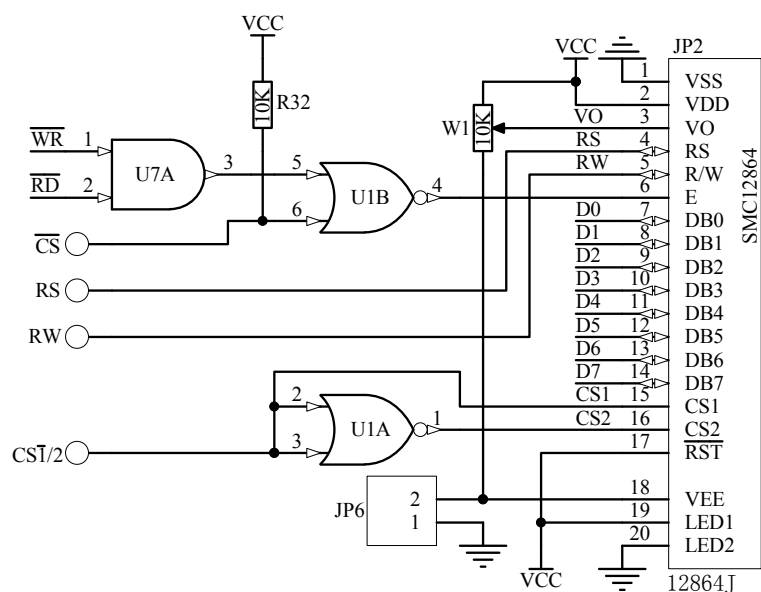
2 硬件结构

2.1 电路外观



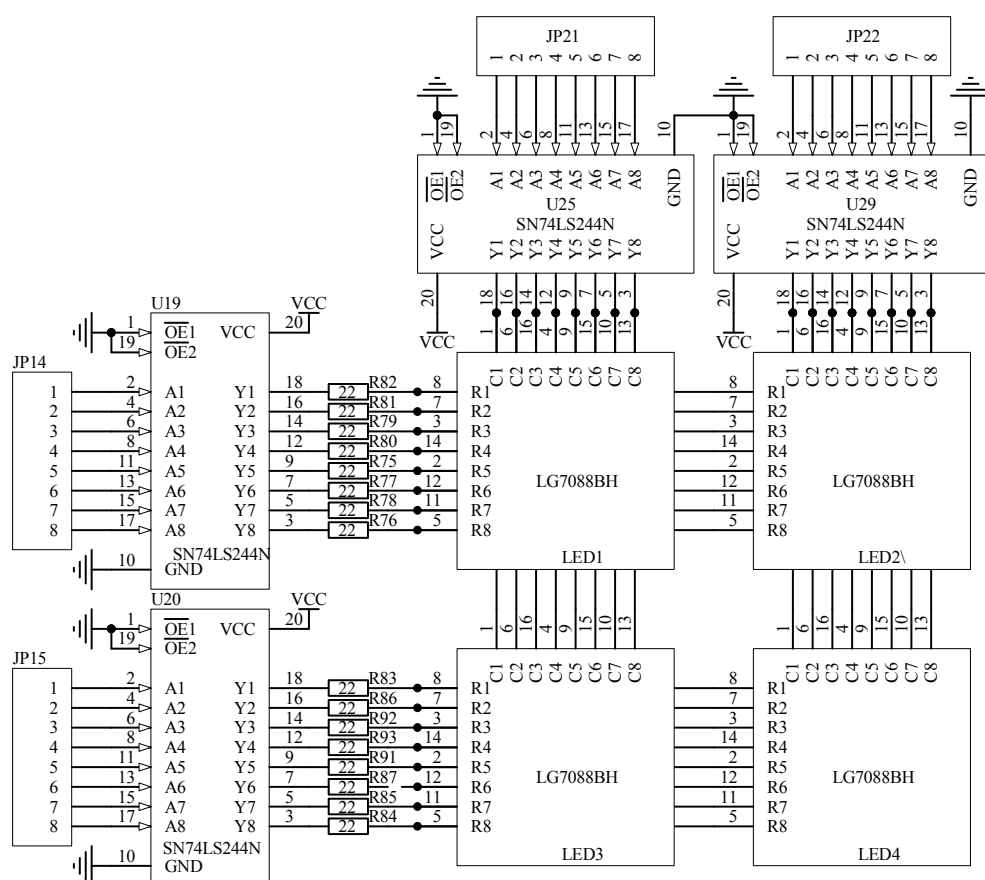
下面将逐一介绍实验仪的各个功能模块、相应的结构，在编写程序前，首先熟悉相应的硬件电路。可忽略课堂上没有介绍的硬件电路。

2. 2 A1 区：12864 液晶显示模块电路



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。

2. 3 A2 区：16×16 LED 实验电路



JP14、JP15 组成 16 根行扫描线；JP21、JP22 组成 16 根列扫描线。

2. 4 A3 区：系统总线、片选区

JP28：地址线 A0..A7；

JP32：地址线 A8..A15；

JP29：地址线 A0..A9；

JP33：地址线 A8..A17；

根据数据总线宽度，选择合适的地址线

JP41、JP42：数据总线 D0..D7

JP39、JP40：数据总线 D8..D15

JP47、JP48：数据总线 D16..D23

JP45、JP46：数据总线 D24..D31

控制线：IOR、IOW、MEMR、MEMW、HOLD、HLDA、BLE、BHE、INTR、INTA

BK1、BK2：备用

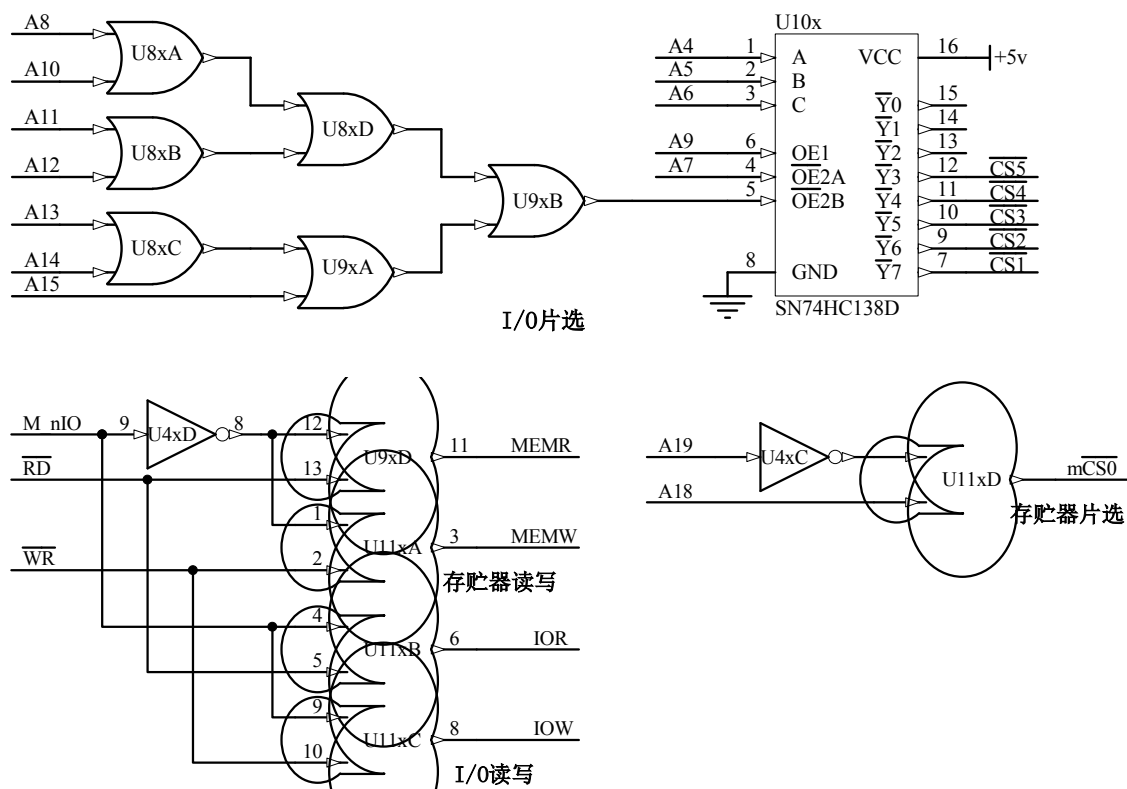
片选区

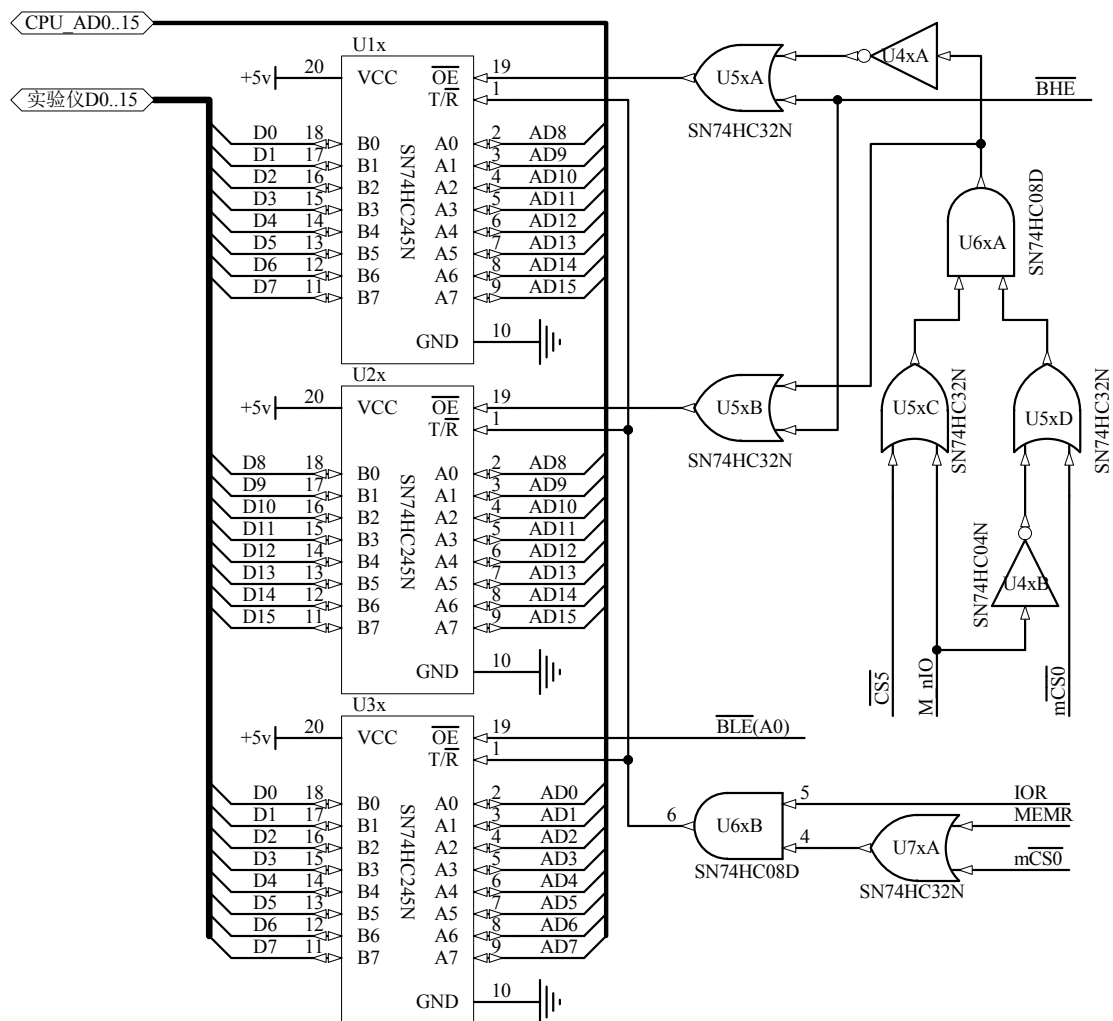
片选	地址范围	说明
mCS0	80000H~BFFFFH	存储器芯片的片选，16 位数据总线
CS1	0270H~027FH	I/O 芯片的片选，8 位数据总线
CS2	0260H~026FH	
CS3	0250H~025FH	
CS4	0240H~024FH	
CS5	0230H~023FH	I/O 芯片的片选，16 位数据总线

实验仪上片选、读写信号、8086 与 16 位、8 位总线外设的总线切换原理图请参考下节。

2. 5 A4 区：控制区

主控部分。

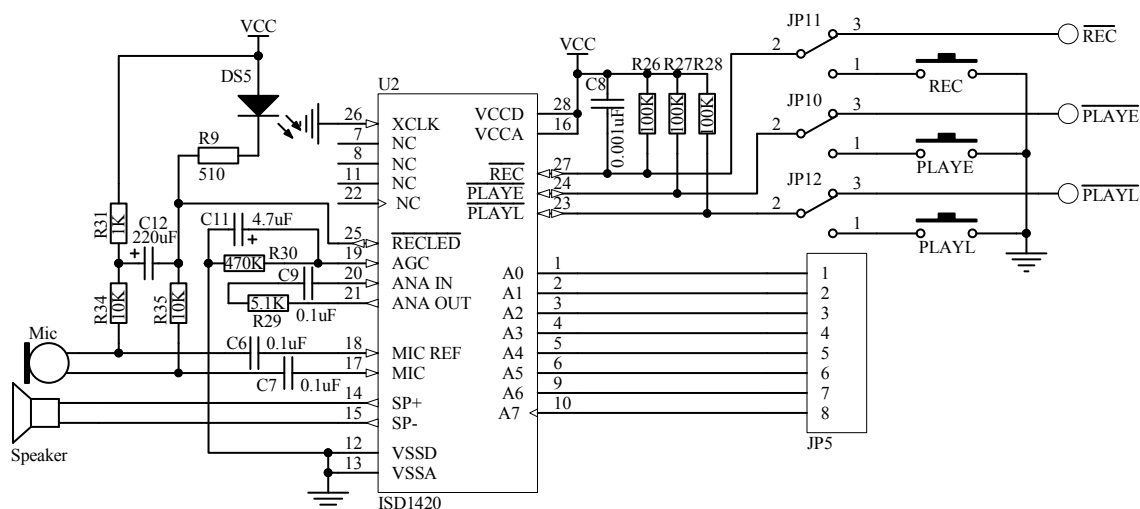




8086 是 16 位总线的 CPU，低八位数据出现在数据线 AD0..7 上，高 8 位数据出现在数据总线 AD8..15 上。8086 与 8 位外设（例如：8255）相连时，如果把 8086 的 AD0..7 直接与 8 位外设的数据线 D0..D7 相连，8086 读写外设时，8086 的最低位地址线 A0（BLE）、BHE 就不起作用，也就是浪费了一半的地址空间；如果访问 8 位外设时，最低位地址线 A0（BLE）为低电平时，8086 的数据线 AD0..AD7 与外设数据线相通，BHE 为低电平时，8086 的数据线 AD8..AD15 与外设数据线相通，就可以使用连续的地址访问外设，不会浪费地址空间。微机内部就是采用这种方式访问 8 位外设的。上图是 SUN ES86PCIU+实验仪访问 8 位、16 位外设时采用的总线切换原理图。

外设	读写	BLE (A0)	BHE	U1x	U2x	U3x	说明
16 位外设 (存储器片选 mCS0、 16 位 I/O 片 选 CS5)	读	0	0	×	√	√	D0..15 → AD0..15
		0	1	×	×	√	D0..7 → AD0..7
		1	0	×	√	×	D8..15 → AD8..15
	写	0	0	×	√	√	D0..15 ← AD0..15
		0	1	×	×	√	D0..D7 ← AD0..7
		1	0	×	√	×	D8..D15 ← AD8..15
8 位外设	读	0	1	×	×	√	D0..D7 → AD0..7
		1	0	√	×	×	D0..D7 → AD8..15
	写	0	1	×	×	√	D0..D7 ← AD0..7
		1	0	√	×	×	D0..D7 ← AD8..15

2. 6 B1 区：语音模块 ISD1420 电路



JP10、JP11、JP12：设置操作模式，MCU：CPU 控制方式；MANUAL：手动(REC、PLAYL、PLAYE)控制方式。

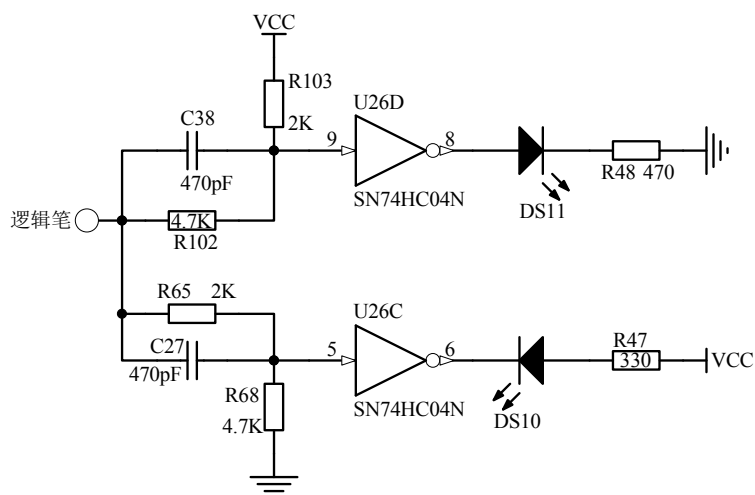
REC: 录音按键, 低电平有效;

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

JP5: 录、放音的起始地址

2.7 B2 区：逻辑笔、单脉冲、频率发生器



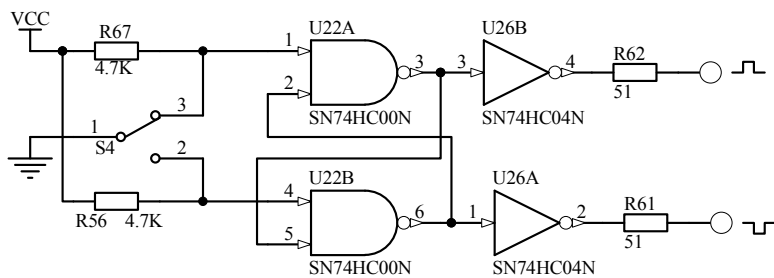
逻辑笔: 测试接口, 输入
测量信号

绿灯 (DS10): 高电平点亮

红灯 (DS11): 低电平点亮

两灯同时亮：频率信号

逻辑笔电路原理图

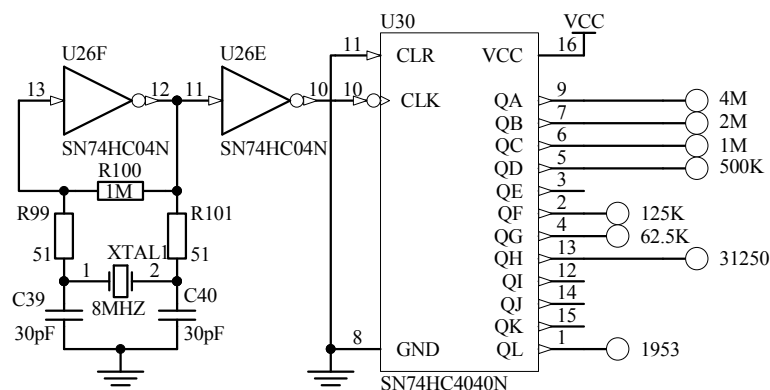


单脉冲电路原理图

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

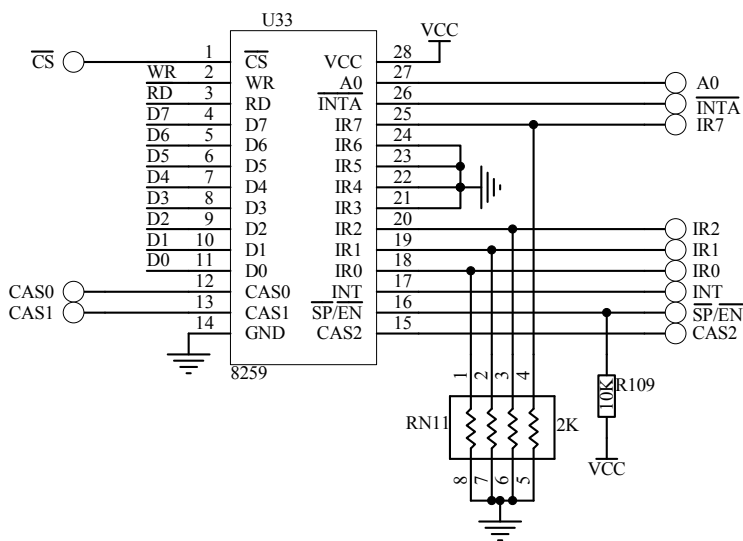


频率发生器电路原理图

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



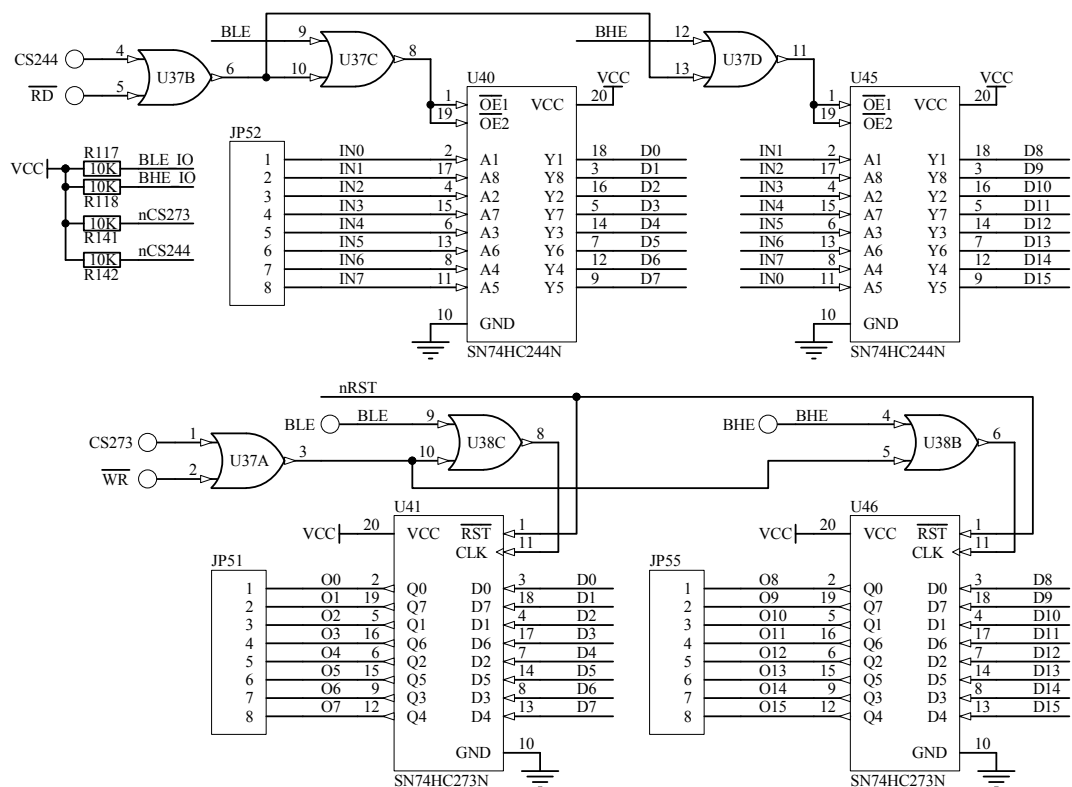
CS: 片选信号, 低电平有效;

A0: 地址信号

INR0.. INR7: 中断输入

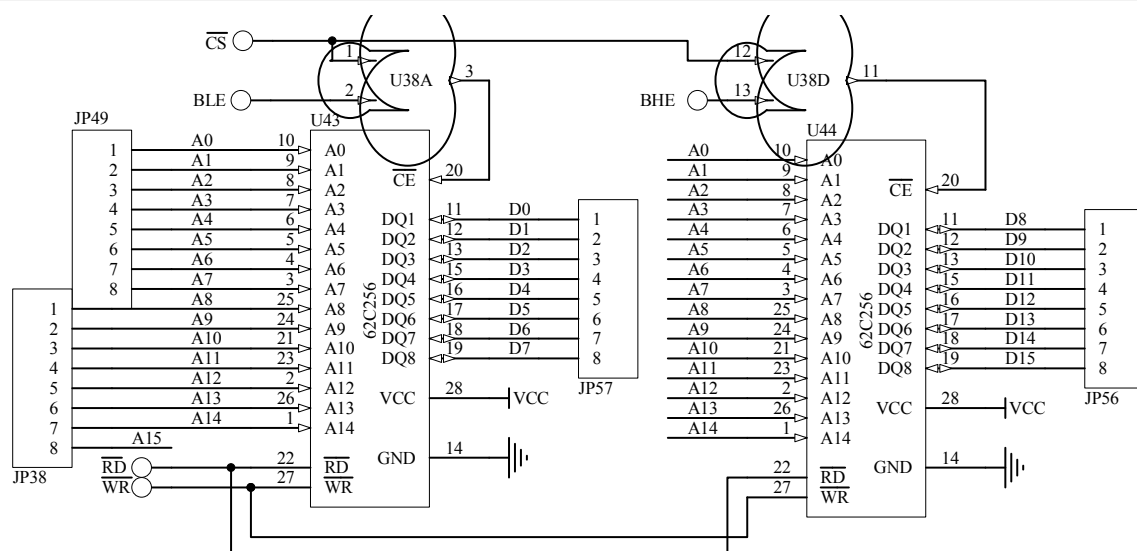
INTA: 中断响应

2. 9 B4 区：简单 I/O、SRAM 电路



可做 8 位、16 位简单 I/O 实验

CS244: 74HC244 片选信号, 低电平有效;	BLE: 低 8 位有效, 允许访问 U40、U41
CS273: 74HC273 片选信号, 低电平有效;	BHE: 高 8 位有效, 允许访问 U45、U46
RD: 读信号, 低电平有效, 读 74HC244	WR: 写信号, 低电平有效, 写 74HC273



CS: 片选信号, 低电平有效;	BLE: 低 8 位有效, 允许访问 U43
	BHE: 高 8 位有效, 允许访问 U44
RD: 读信号, 低电平有效	WR: 写信号, 低电平有效

2. 10 B5、C5 区：扩展区

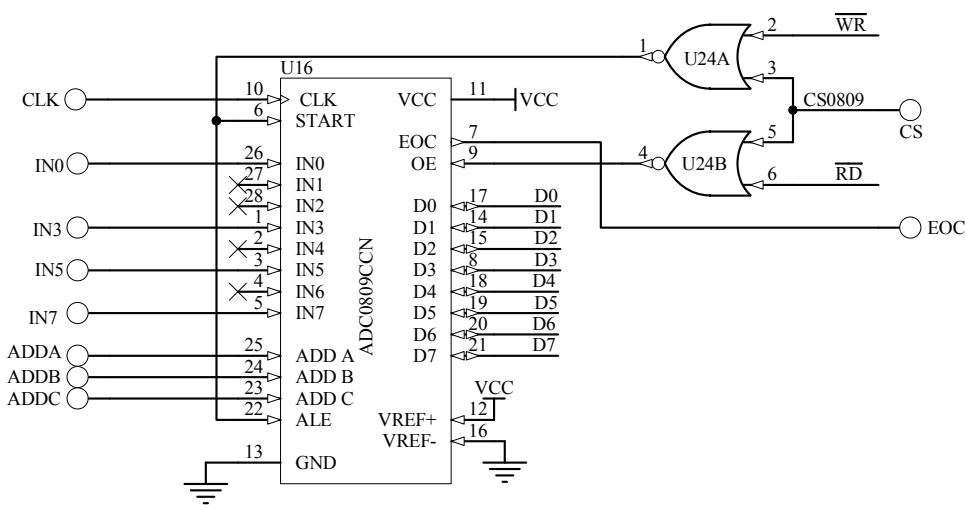
实验仪提供了二个扩展区，用来扩展 USB1.1、USB2.0、以太网、CAN 总线、非接触式 IC 卡、GPS、GPRS、双通道虚拟示波器、CPLD、FPGA 等扩展模块，其它模块正在陆续推出中。

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

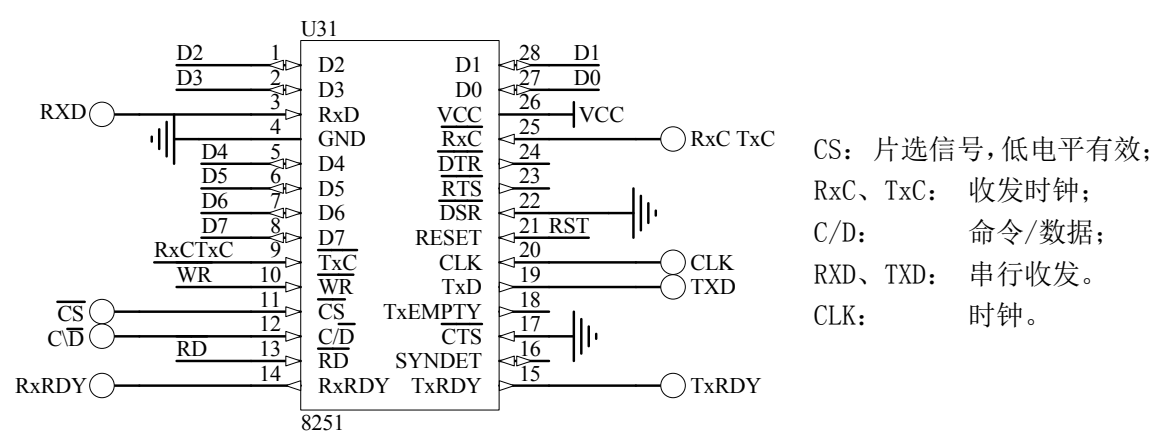
C1 区为用户提供了+5V(2A)、+12V(400mA)、-12V(400mA)等几种电源接口。DS8：+5v 指示灯；DS7：+12v 指示灯；DS6：-12v 指示灯。

2. 12 C2 区：ADC0809 模数转换

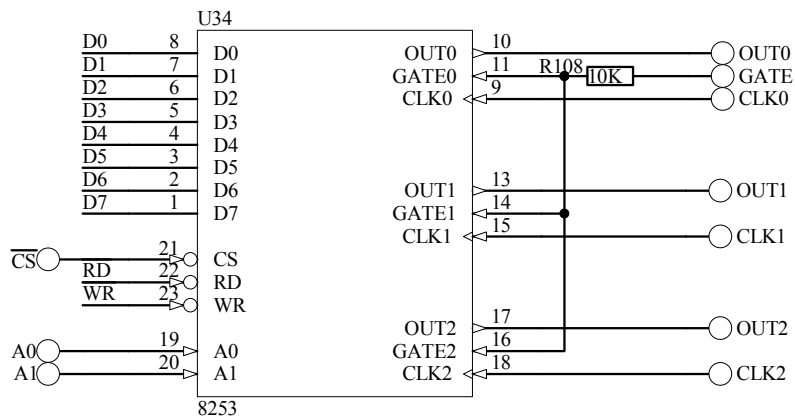


CS:	片选，低电平有效；
CLK:	输入时钟(10k—1280kHz)；
ADDA, ADDB, ADDC:	通道地址输入口；
EOC:	转换结束标志，高电平有效。
IN0、IN3、IN5、IN7:	模拟量输入

2. 13 C3 区：8251 电路

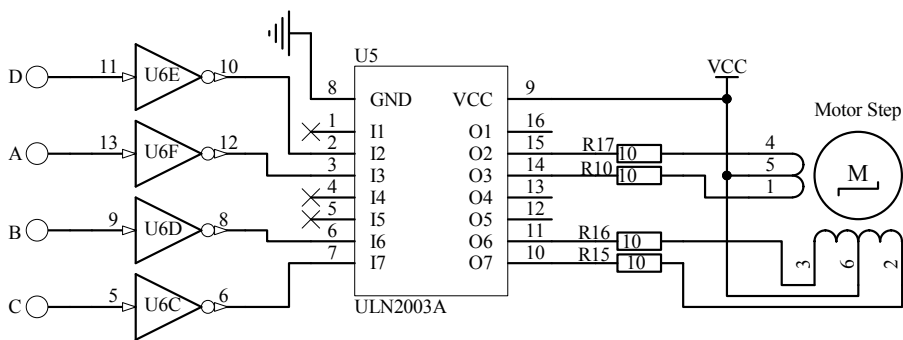


2. 14 C4 区：8253 电路

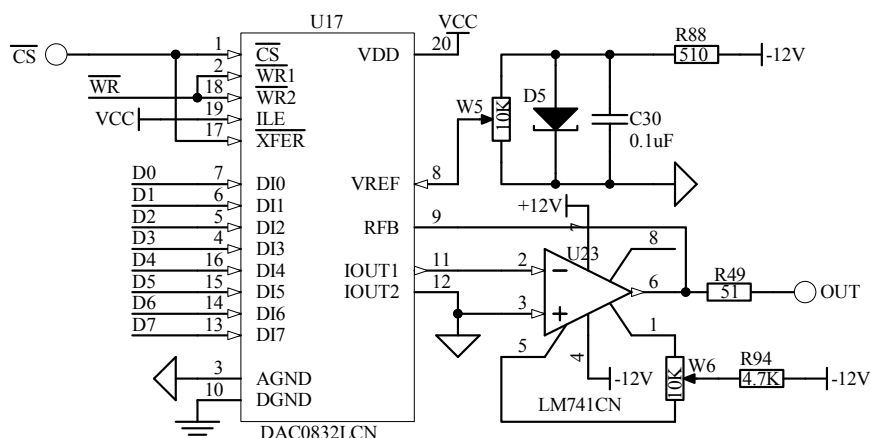


CS: 片选信号, 低电平有效;
A0、A1: 地址信号;

2. 15 D1 区：步进电机

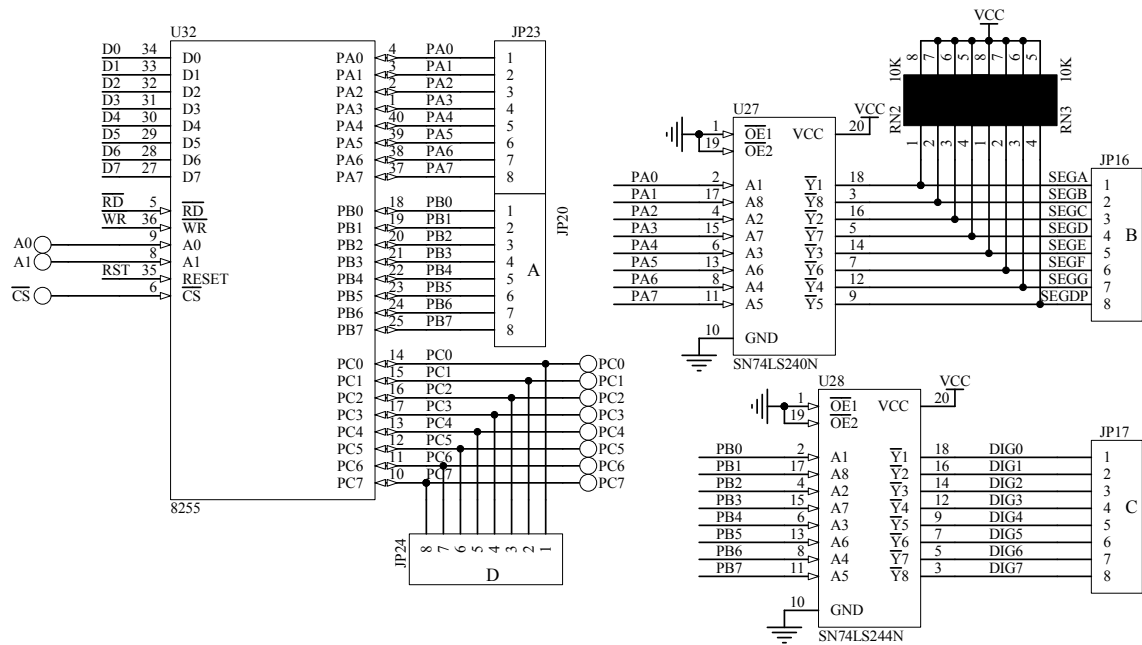


2. 16 D2 区：DAC0832 数模转换



CS: 片选, 低电平有效; OUT: 转换电压输出; 电位器 W5: 调整基准电压。

2. 17 D3 区：8255 电路、数码管驱动电路



CS: 片选信号，低电平有效；

A0、A1: 地址信号。

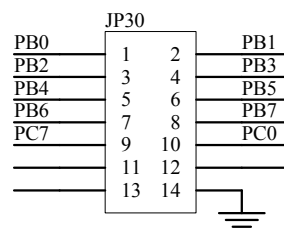
JP24: PC 口(键盘行)；

JP20: PB 口(键盘列)；

JP23: PA 口；

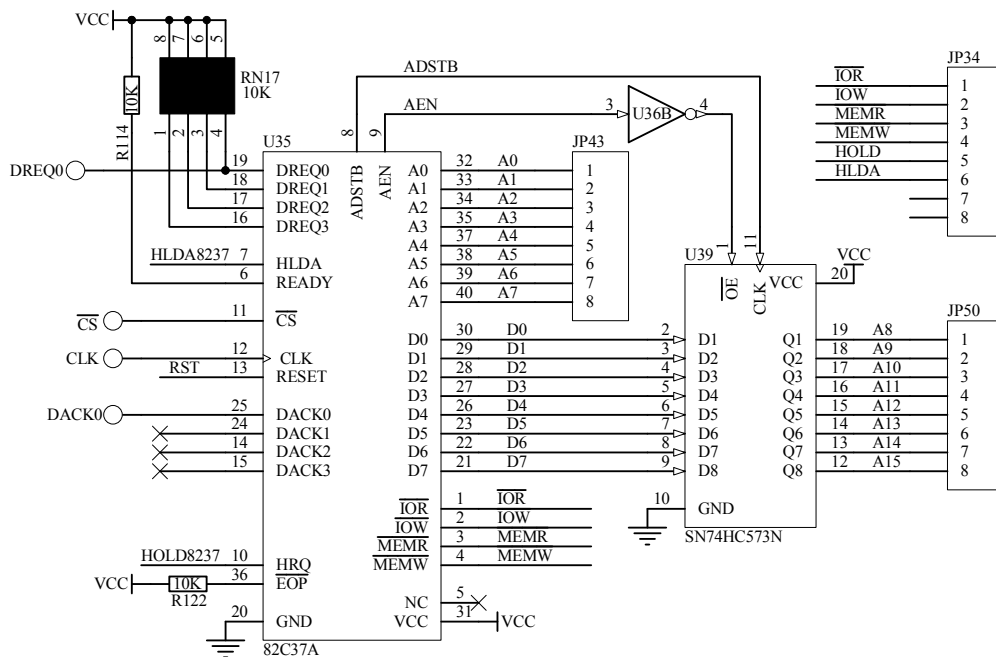
JP16: 数码管段码

JP17: 数码管段选



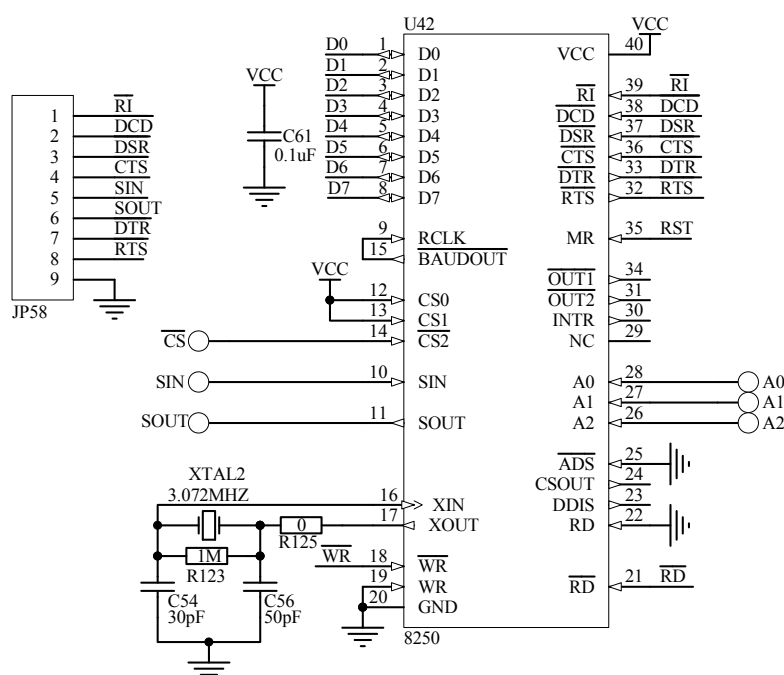
打印口

2. 18 D4 区：8237 电路



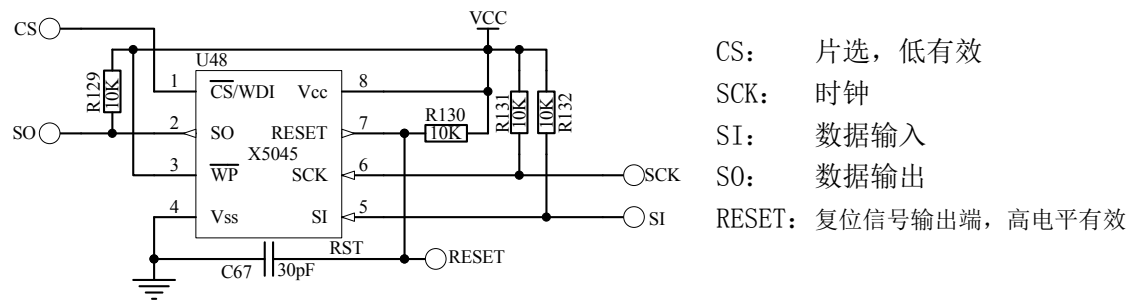
- CS: 片选信号，低电平有效；
- CLK: 时钟信号。控制 8237 内部定时和 DMA 传输时的数据传送速率
- DREQ0: DMA 请求信号；
- DACK0: DMA 响应信号；
- HOLD: 总线请求信号，高电平有效；
- HLDA: 总线响应信号，高电平有效；
- IOR: I/O 设备读信号，低电平有效；
- IOW: I/O 设备写信号，低电平有效；
- MEMR: 存储器读信号，低电平有效；
- MEMW: 存储器写信号，低电平有效；

2. 19 D5 区：8250 电路

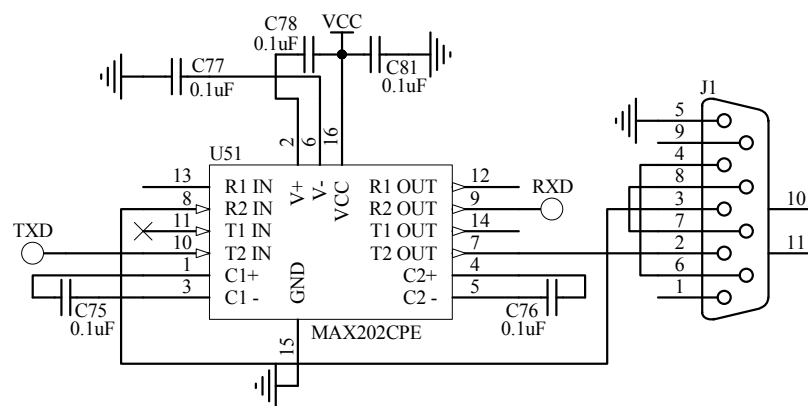


- CS: 片选信号，低电平有效；
- A0、A1、A2: 地址信号；
- SIN: 串行输入
- SOUT: 串行输出

2. 20 D6 区: X5045

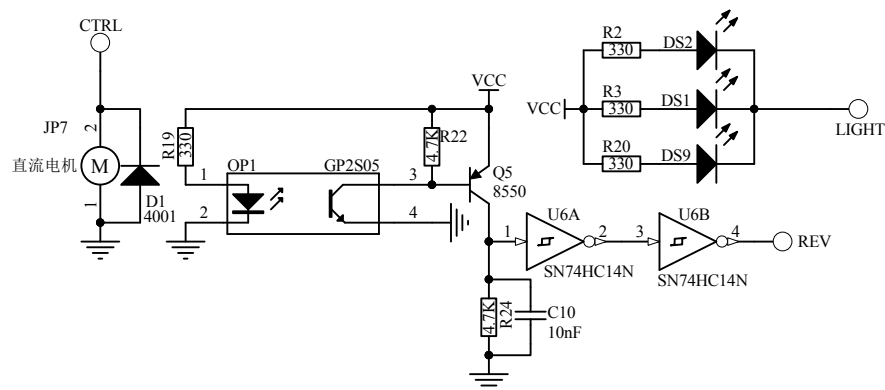


2. 21 D7 区: RS232

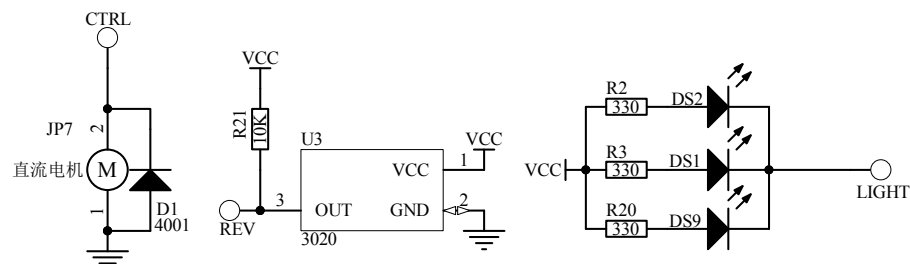


2. 22 E1 区: 直流电机转速测量/控制

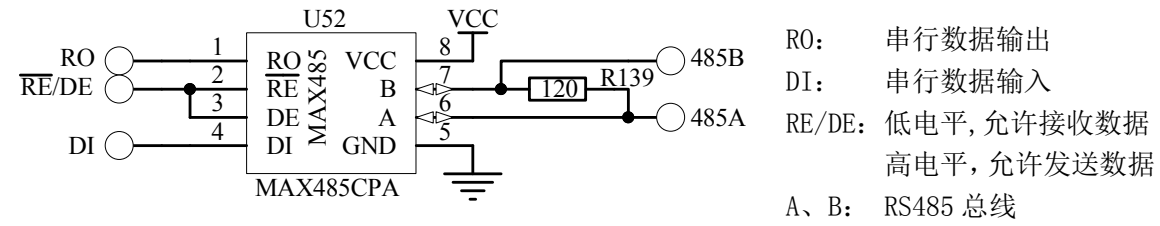
使用光电开关测速



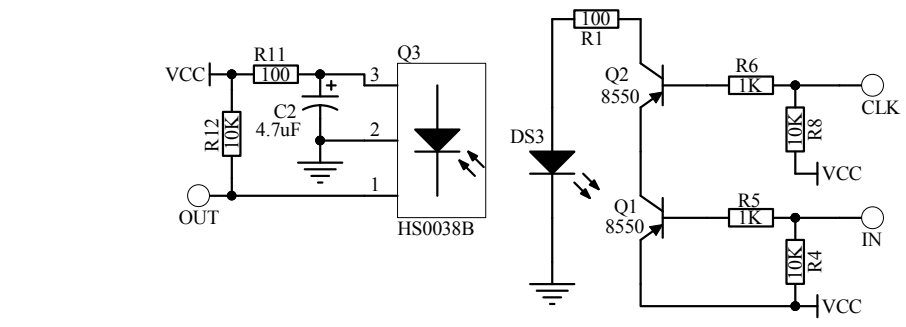
使用霍尔器件测速



2. 25 E4 区: RS485

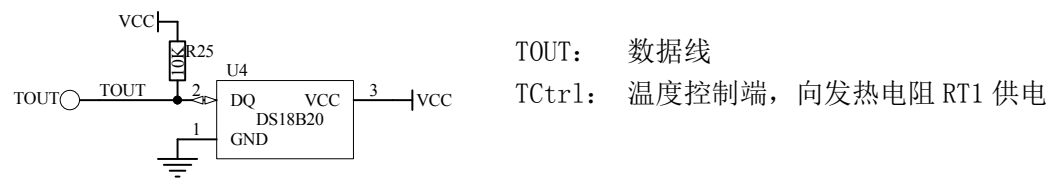


2. 26 F1 区: 红外通讯

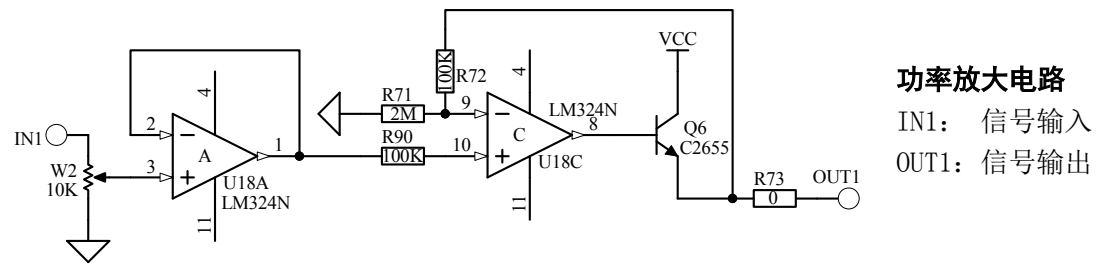
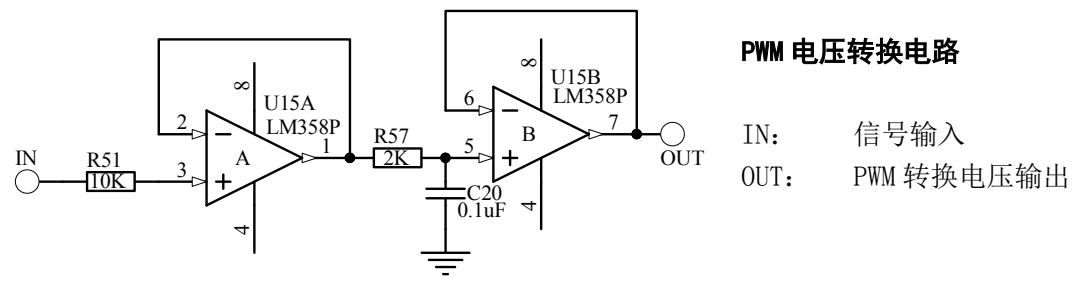


IN: 串行数据输入
OUT: 串行数据输出
CLK: 载波输入, 可接 31250 (B2 区) 频率输出

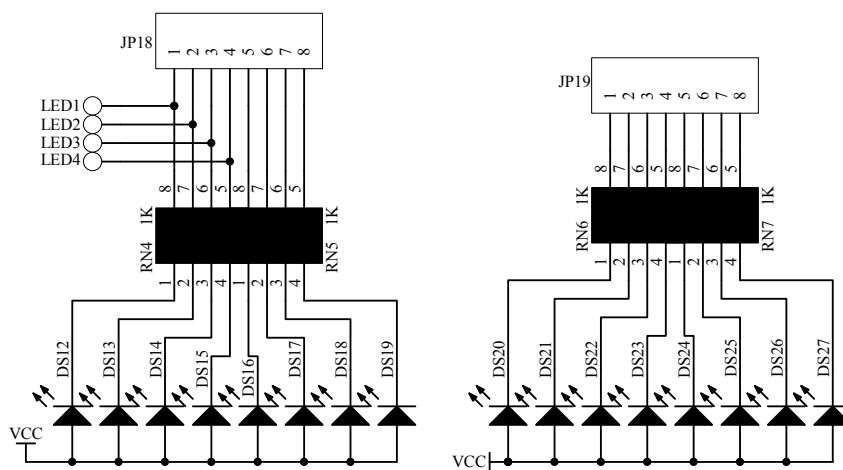
2. 27 F2 区: 温度测量/控制



2. 28 F3 区: PWM 电压转换、功率放大电路

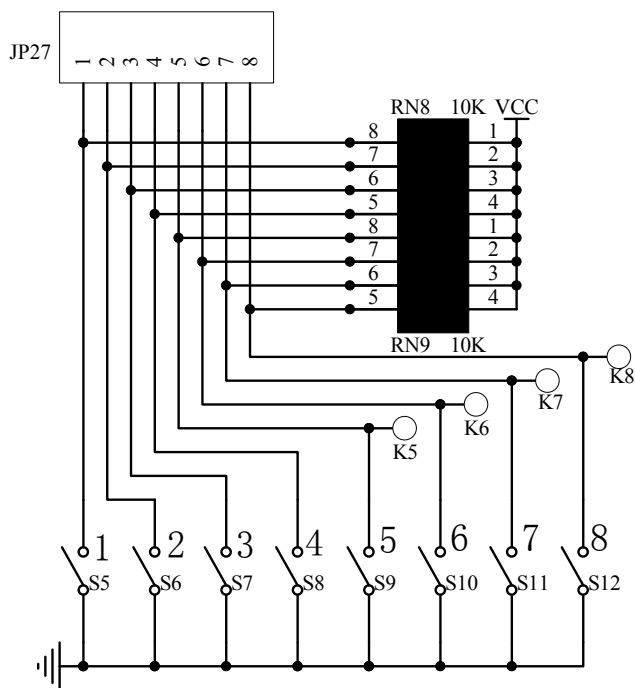


2. 29 F4 区：发光管、开关



发光管电路原理图

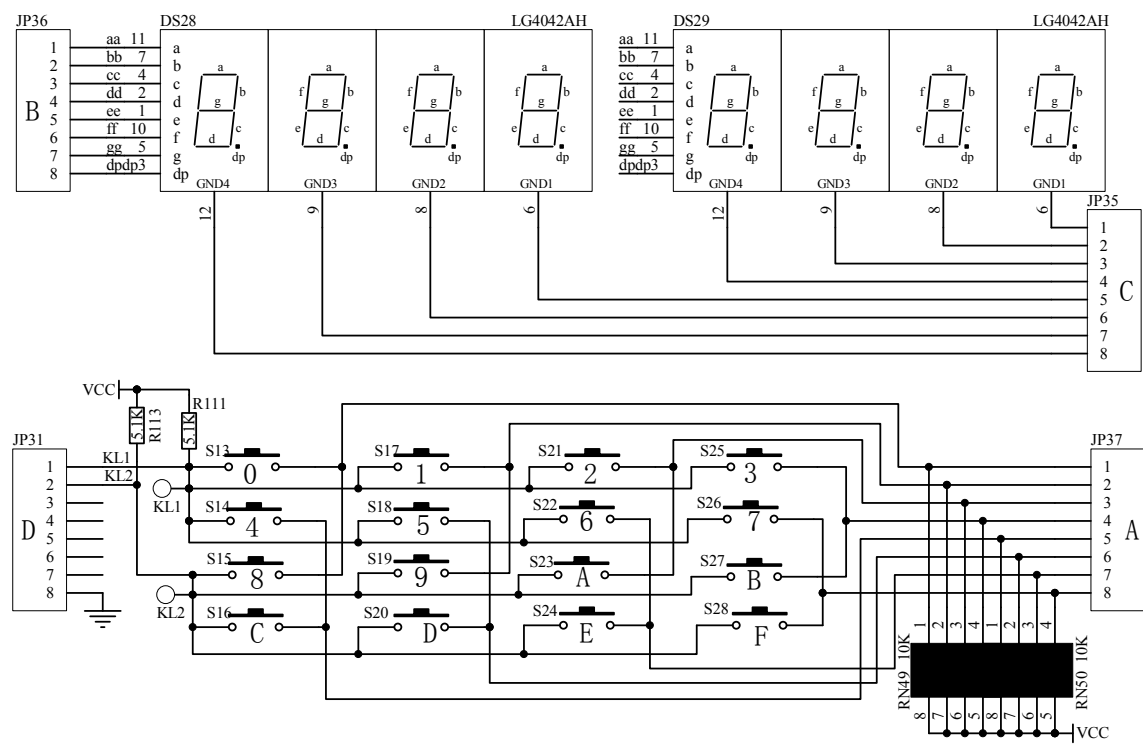
JP18、JP19：发光管控制接口，0—灯亮，1—灯灭



开关电路原理图

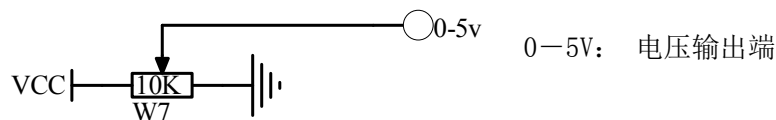
JP27：开关控制接口；闭合—0 信号，断开—1 信号

2. 30 F5 区：键盘&LED

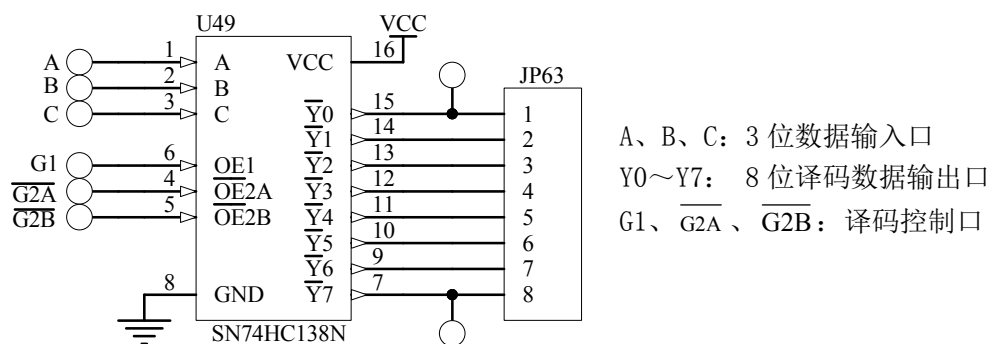


A:	按键的列线	B:	数码管段码
C:	数码管选择脚	D:	按键的行线

2. 31 F6 区：0~5V 电压输出

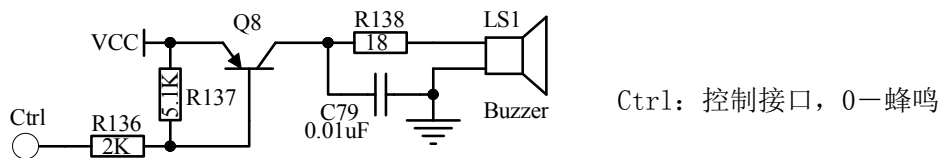


2. 32 F7 区：138 译码器



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

2. 33 F8 区：蜂鸣器



Ctrl：控制接口，0—蜂鸣

3 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，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\星研\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：



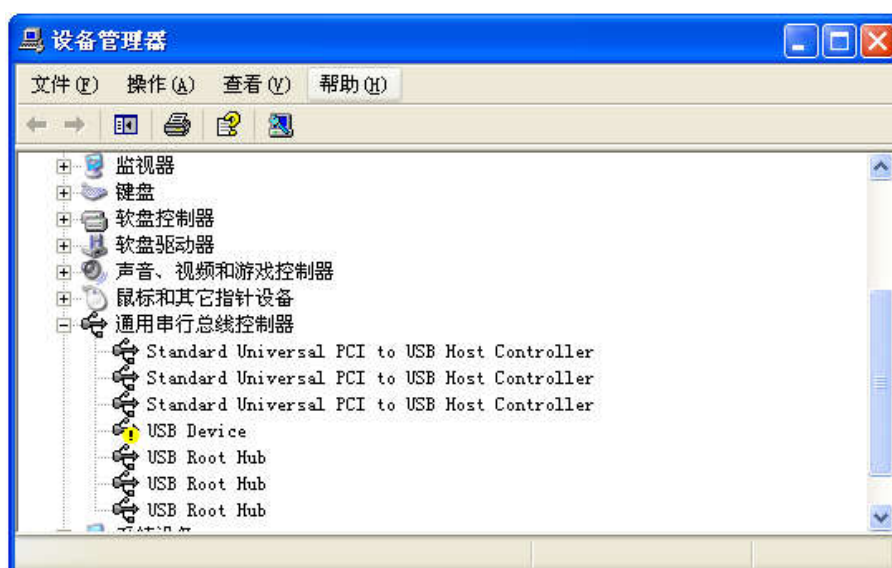
实际的界面可能有些差别，请等待该过程的结束。驱动程序的安装过程中，请勿执行其它应用程序。

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/WIN7: 在仿真器或实验仪电源打开的情况下，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/VISTA/WIN7 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 6000H~60FFH 中；然后比较。3. 2. 2 使用 Turbo C，重新编写第一个实验。

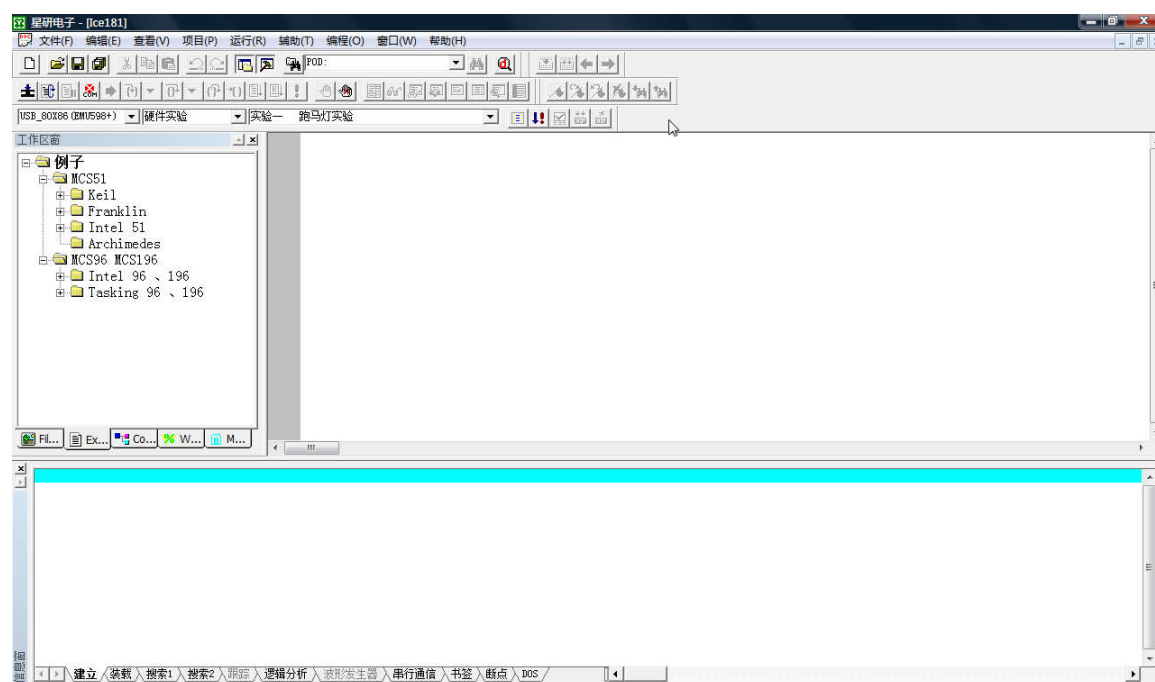
3. 2. 1 数据传送程序（ASM）

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。

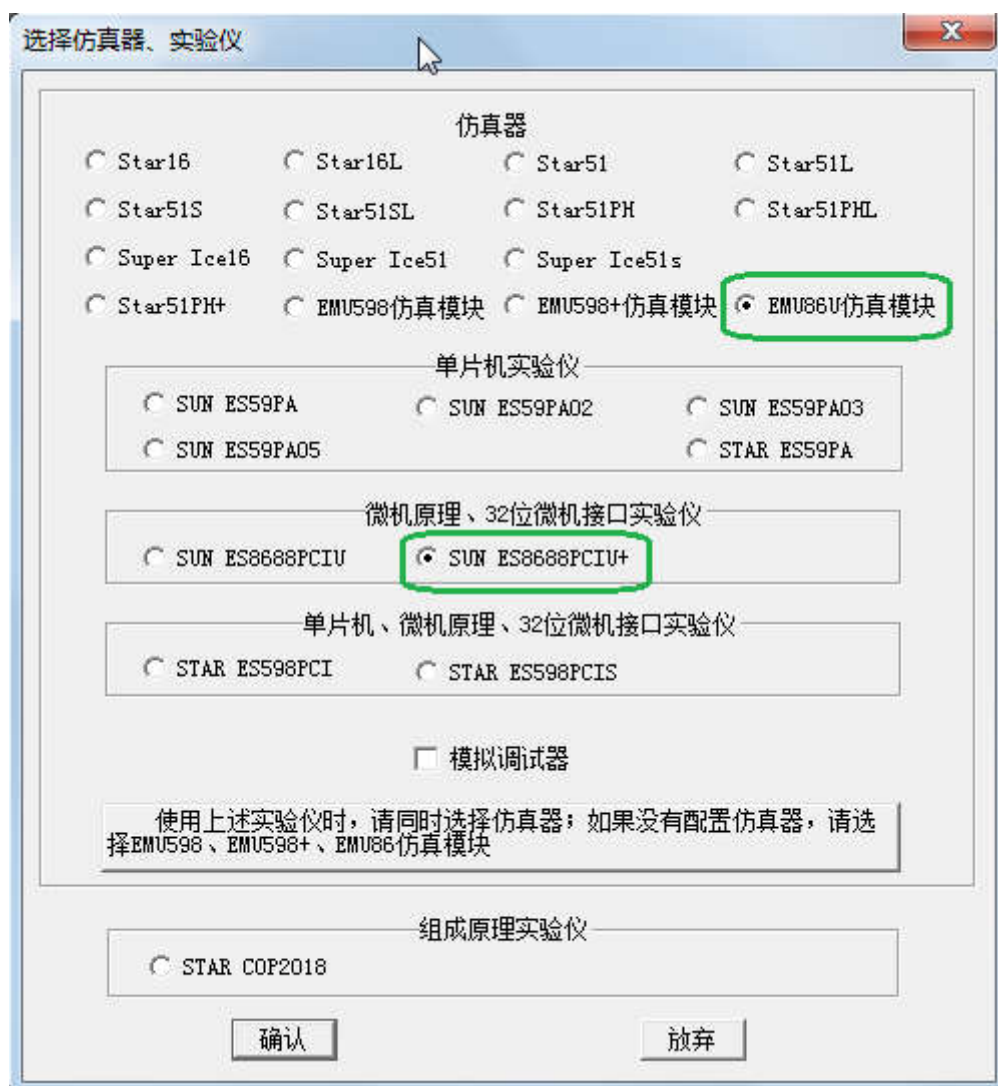
本实例是将数据段中 3000H~30FFH 单元的内容传送给 6000H~60FFH 单元中；再将它读出与 3000H~30FFH 单元中数据比较。程序是用汇编语言来编写。下面介绍相应的操作步骤：

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：SUN ES86PCIU+；仿真器：EMU86U 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。

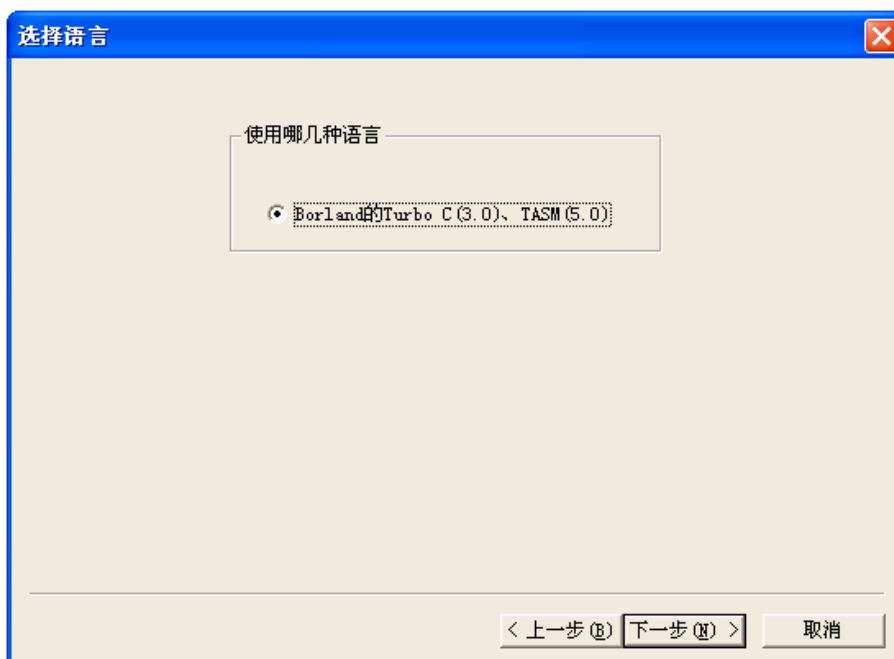
2、设置缺省项目

执行 [主菜单 » 辅助 » 缺省项目]，出现一个对话框：

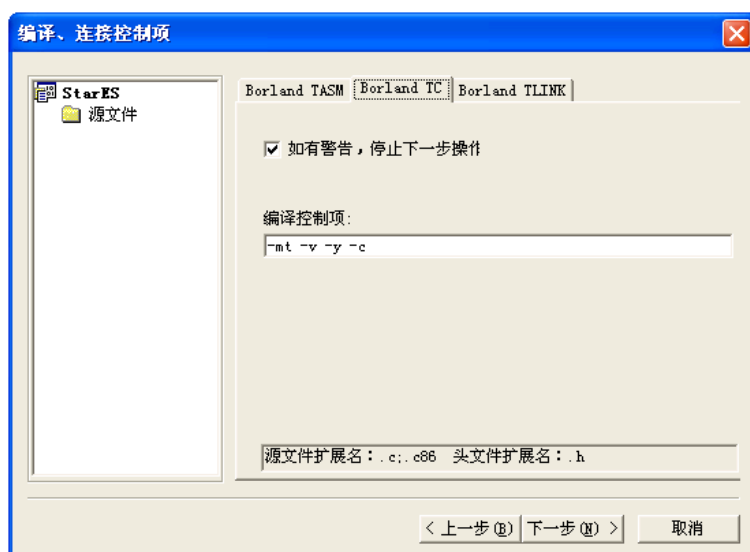


请选择“8086（EMU86）”。

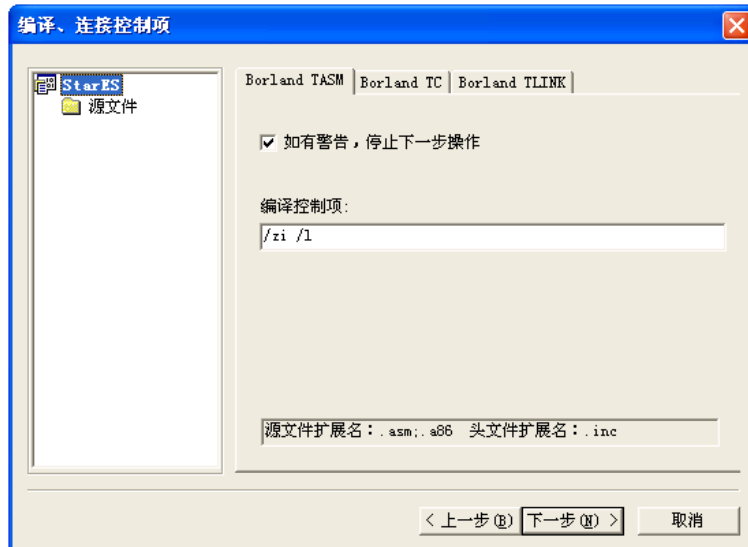
点击进入下一步：“选择语言”



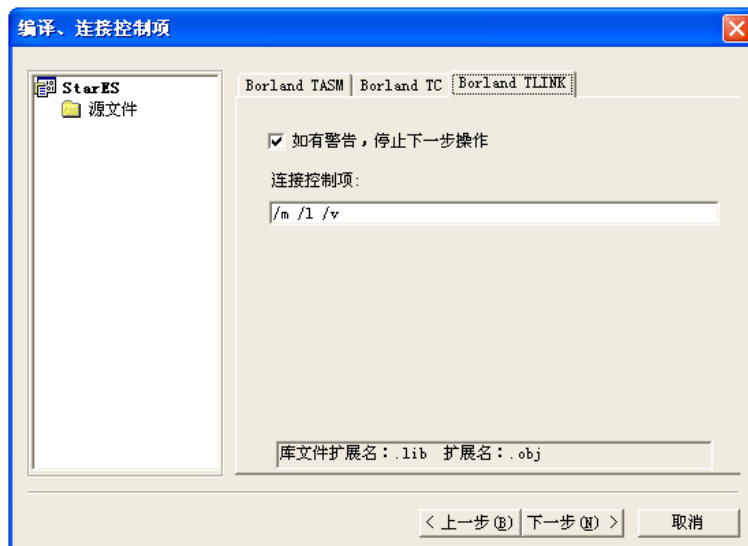
您可以根据自己的需要以及程序的类型作相应的选择，本实例选择Borland公司的Turbo C (3.0)、TASM (5.0)（请确定在选择语言之前已经安装好相应的编译软件）。然后再点击进入下一步：“编译、连接控制项”



memory model 请选择 tiny, 缩写为 mt (也可以选择其它模式); 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。

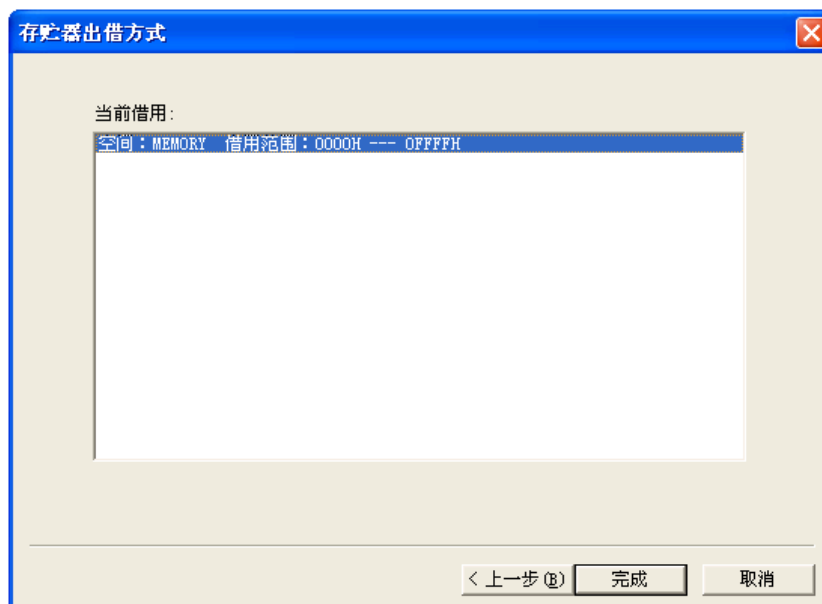


如果需要源程序级别调试，必须使用 /zi /l 控制项。
一般不必改变 Tasm 的编译控制项。



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”

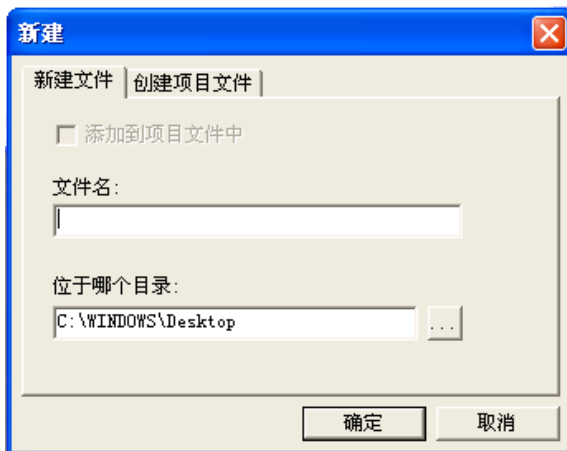


仿真模块 EMU86 提供 64K 仿真 RAM (IS61LV6416)，作程序段 (CS)、数据段 (DS)、附加段 (ES)、堆栈段 (SS) 使用。

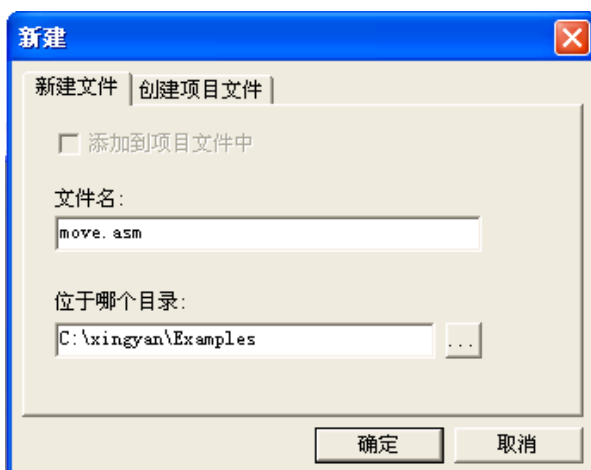
B4 区的 SRAM (二片 62256) 也可以作数据段 (DS)、附加段 (ES)、堆栈段 (SS) 使用，但需要连接地址、数据总线、读信号、写信号。

3、建立源文件

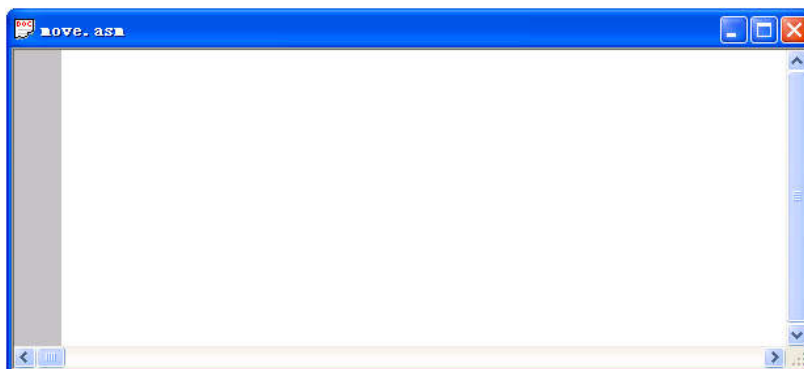
下面我们建立源文件，执行 [主菜单 » 文件 » 新建]，（或者点击图标）打开窗口如下：



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
```

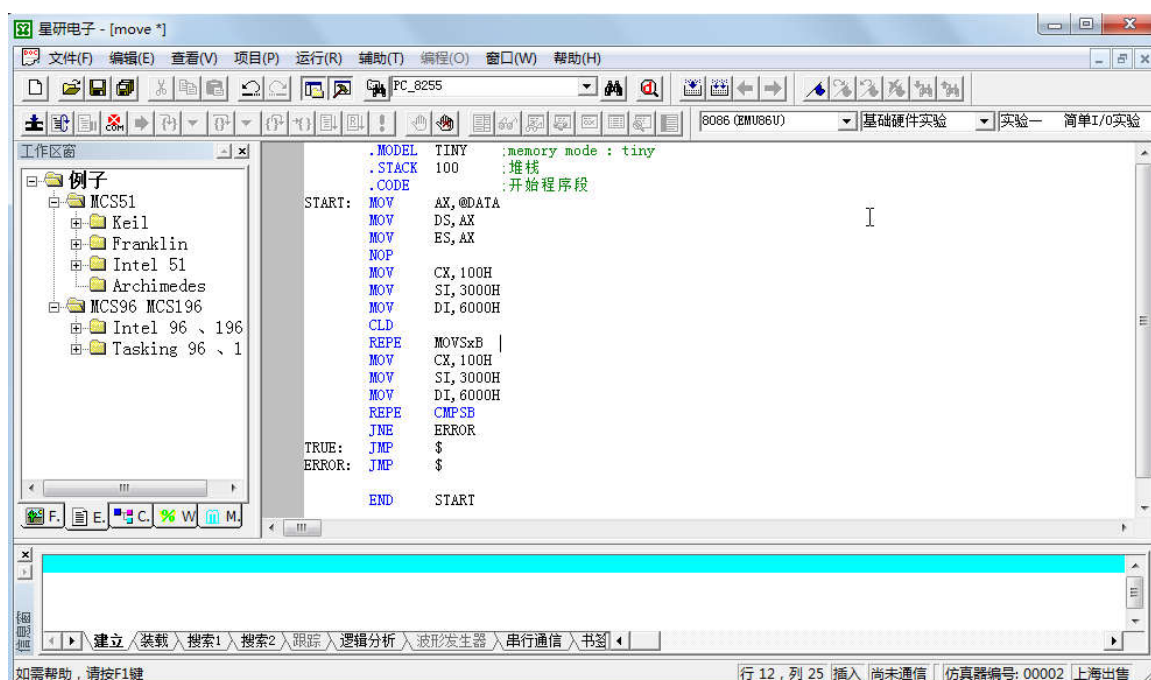
```

MOV        DS, AX
MOV        ES, AX
NOP
MOV        CX, 100H
MOV        SI, 3000H
MOV        DI, 6000H
CLD
REPE       MOVSB
MOV        CX, 100H
MOV        SI, 3000H
MOV        DI, 6000H
REPE       CMPSB
JNE        ERROR
TRUE:      JMP        $
ERROR:     JMP        $

END        START



```

输入源程序，如下图：



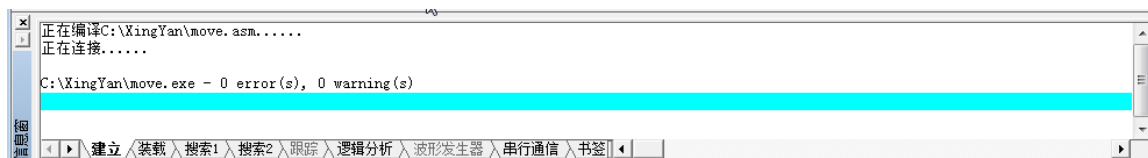
这样一个源文件就建立好了。

4. 编译、连接文件

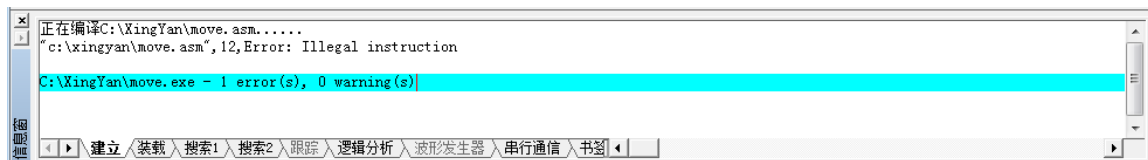
首先选择一个源文件，然后可以编译、连接文件了。对文件编译，如果没有错误，再与库文件连接，生成代码文件（DOB、EXE 文件）。编译、连接文件的方法有如下二种：（1）使用[主菜单 » 项目 » 编译、连接]或[主菜单 » 项目 » 重新编译、连接]；（2）点击图标或来“编译、连接”或“重新编译连接”。

“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

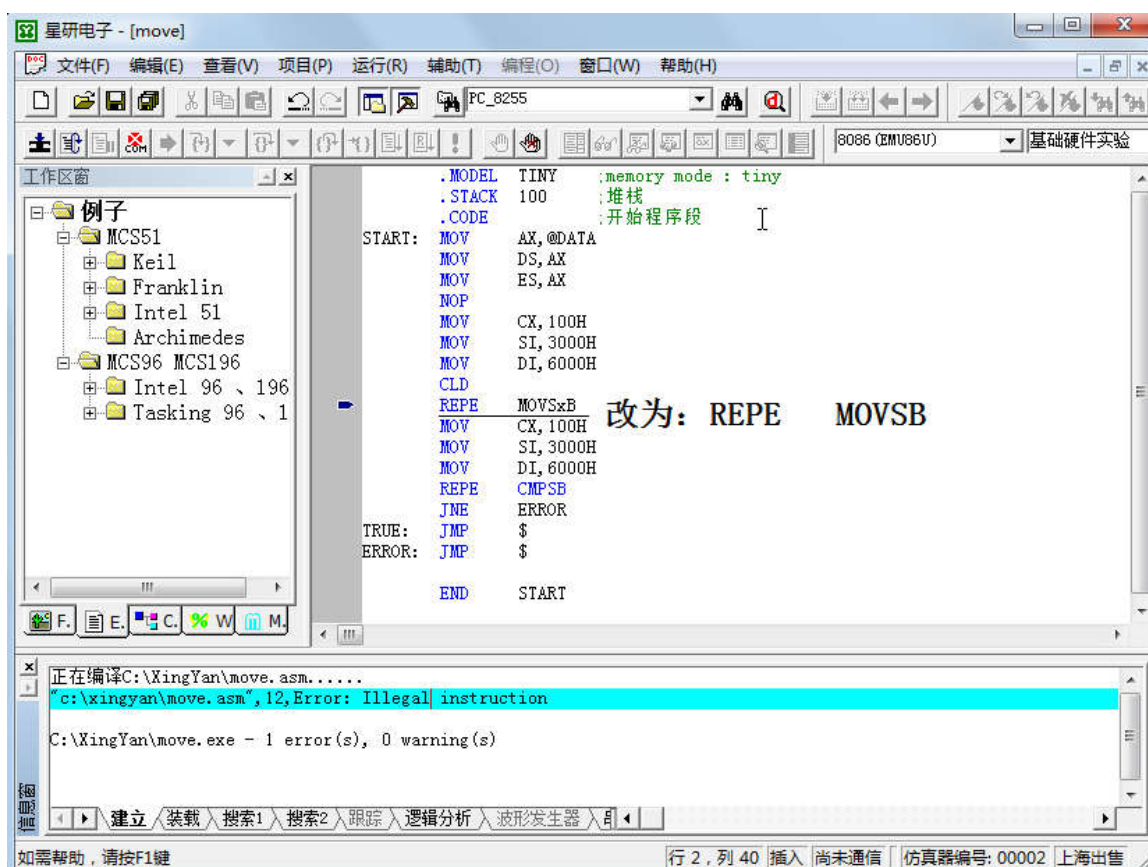
代码文件（DOB、EXE 文件）。编译、连接过程中产生的信息显示在信息窗的“建立”视中。编译没有错误的信息如下：



若有错误则出现如下信息框：



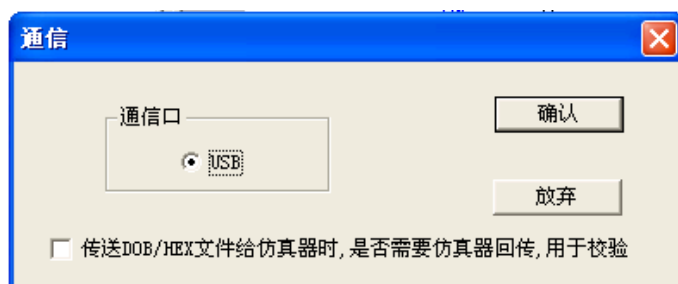
有错误、警告信息，用鼠标左键双击错误、警告信息或将光标移到错误、警告信息上，回车，系统自动打开对应的出错文件，并定位于出错行上。



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：

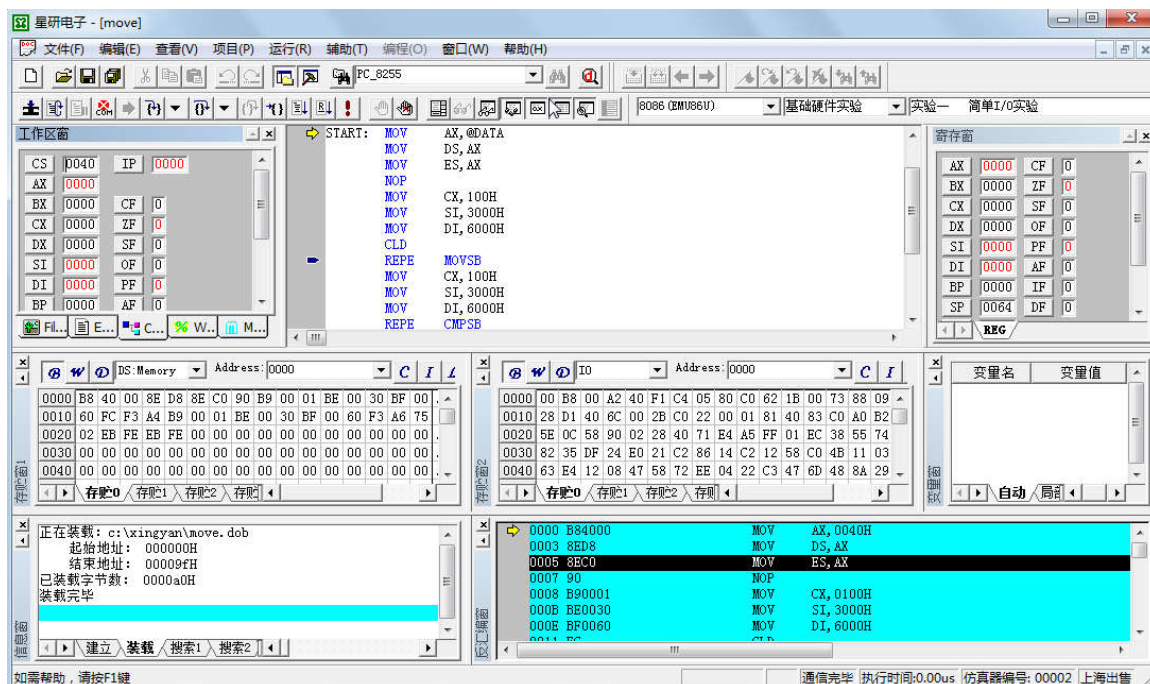


仿真器、实验仪配套的通信线可以与微机 USB 口相连, 即为 USB 通信线, 请选择 USB。
对于最下面一行的**校验**, 通常您不必选中它, 可以提高传送 DOB、HEX、BIN 文件时的速度。
在进入调试状态以前, 你还必须确定实验仪与微机的正确连接, 电源接通, 开关打开。

编译、连接正确后, 可以开始调试程序。进入调试状态方法有:

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

进入后的窗口如下:


















在整个图片中可以看到相对应的窗口信息。在“工作区窗”的“CommonRegister”中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。

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）。

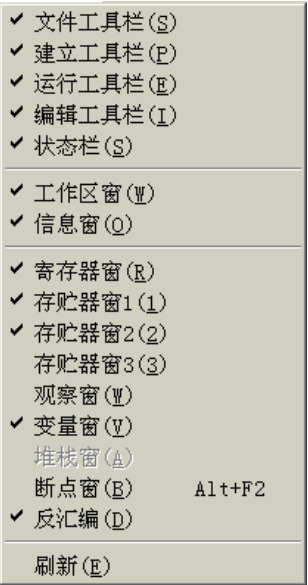
注意：欲终止微机与仿真器之间通信，功能键 **ESC** 是一个很方便的键，它的效果比点击相
应的图标的效果要好。建议用户多用 **ESC** 键。在系统运行“连续单步”或者“连续单步进入”
时 **ESC** 键被禁止，这时用户可以按键盘的其他任意键停止其运行。

5. 调试的方法及技巧

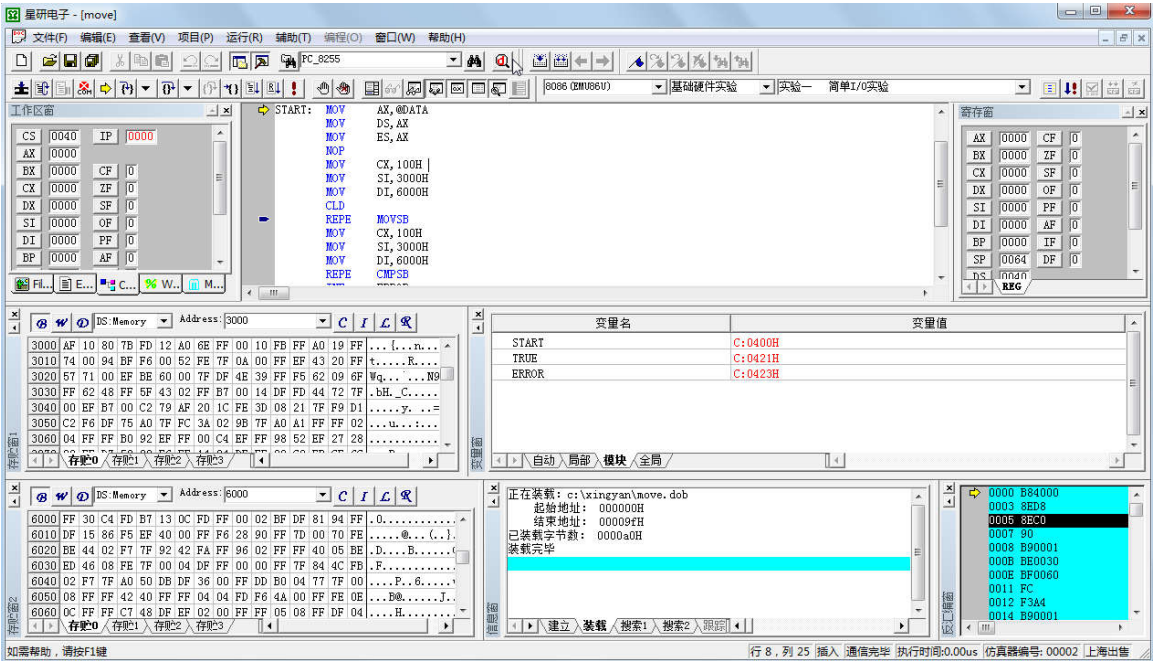
一般来说，用户的程序或多或少的会有一些逻辑错误，所使用的仿真器、实验仪和星研集
成软件可以帮助用户很快的定位，很快的查出相应的错误。

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。

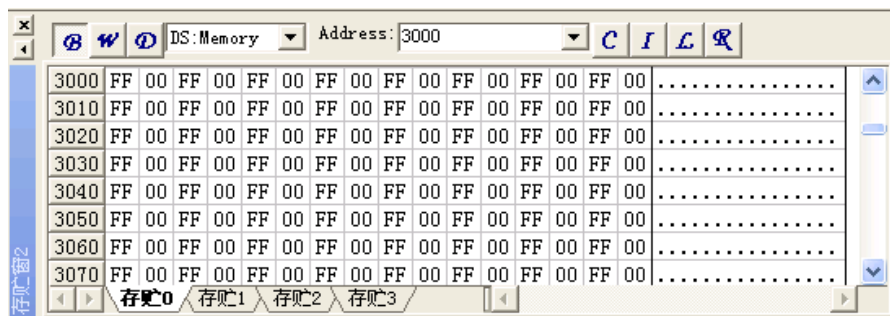
进入调试界面后，由于本次操作需要观察二个数据块：数据段 3000H~30FFH，数据段 6000H~60FFH，可以打开一到二个存储器窗口，具体操作是：[主菜单>>查看]



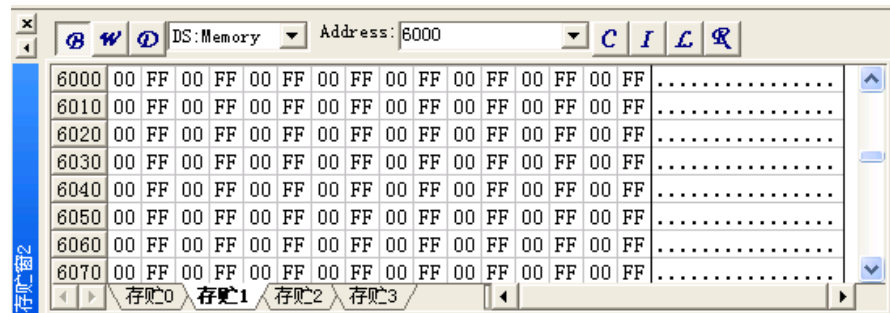
然后根据你的需要打开不同的窗口。调整后的调试界面为：



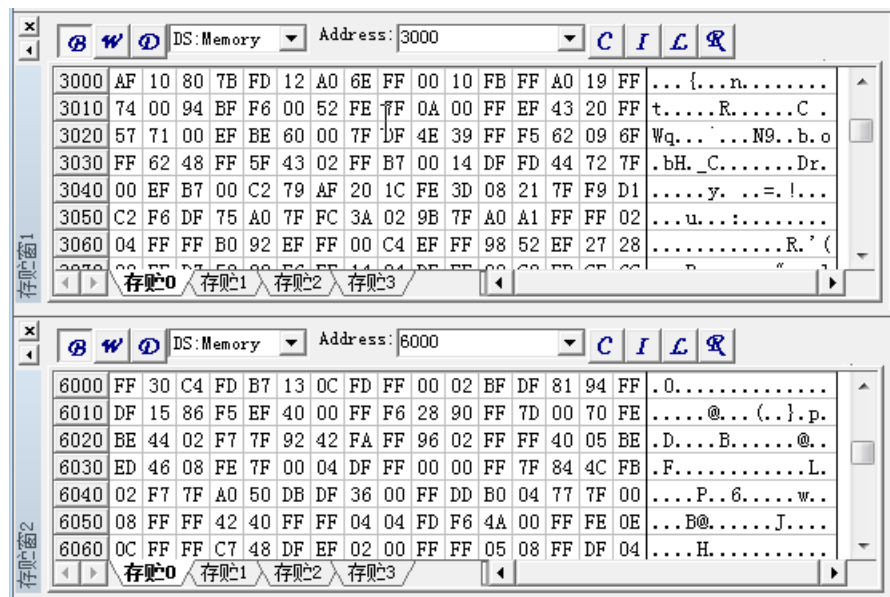
由于我们本次操作主要是观察存储器窗口，所以我们拉大了这两个存储器窗口的大小。每个窗口设置了4个分页项：**存储0** **存储1** **存储2** **存储3**，我们可以在不同的分页项设置不同的观察数据空间以及地址范围。在**DS: Memory**中可以选择CS: Memory, DS: Memory, I/O，根据需要可以做不同的选择。在**Address: 0000**中可以直接输入地址，然后按回车，就可以直接转到我们输入的地址的窗口上面观察数据。由于我们在此程序中的写入数据的RAM空间分别为DS: 3000H~30FFH、DS: 6000H~60FFH，故我们建立的分页项如下：
存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



存储 1 分页项:

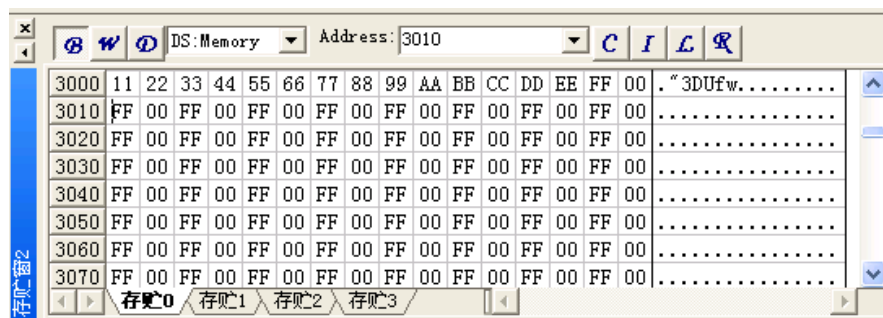


我们这样设置界面的目的就是当用户要观察不同地址段的数据时，只要切换一下分页项就行了。由于本次程序需要同时观察 DS: 3000~30FFH、DS: 6000H~60FFH，所以打开二个存储器窗。如图：



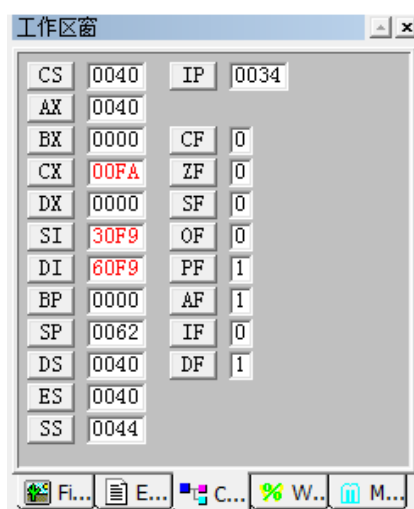
软件中总共存在 3 个存储器窗。可以同时观察三个不同的地址。

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：



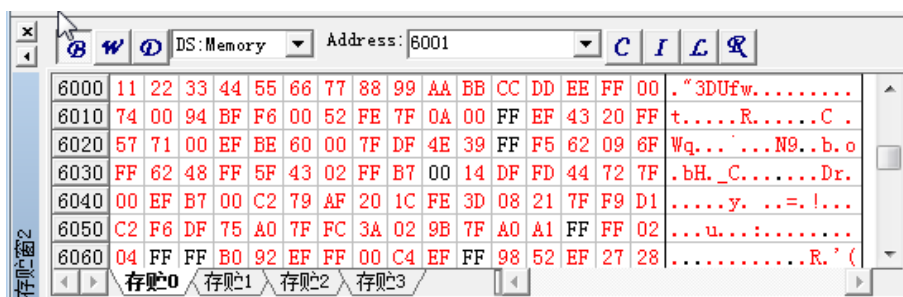
一般刚刚写好的程序，在进入调试状态后，执行“单步”或者“单步进入”，我们推荐您能记住这些操作的相对应的功能键，这样您就在调试程序的过程中很方便。

在刚才的调试程序中我们多次执行“单步进入(F7)”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 CX、DI、SI 的数值的变化，每一次 MOVSB, CX 减一，DI、SI 加一，从 DS: [SI] 取出的数值复制给 ES: [DI] 中：11、22、33、44、55、66……。

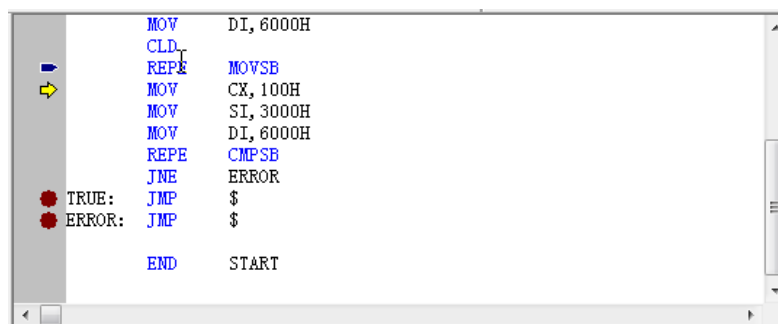
我们可以看到存储器窗口中的相对应的 RAM 的数据的变化。比如





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。

把光标移动到第 13 行 (MOV CX, 100H) 上，点击图标 (功能键 F4)，全速运行到光标行，检查 DS: 6000H~60FFH 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。

将光标移到第 18 行 (TRUE: JMP \$) 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；在 19 行 (ERROR: JMP \$) 上设置断点。

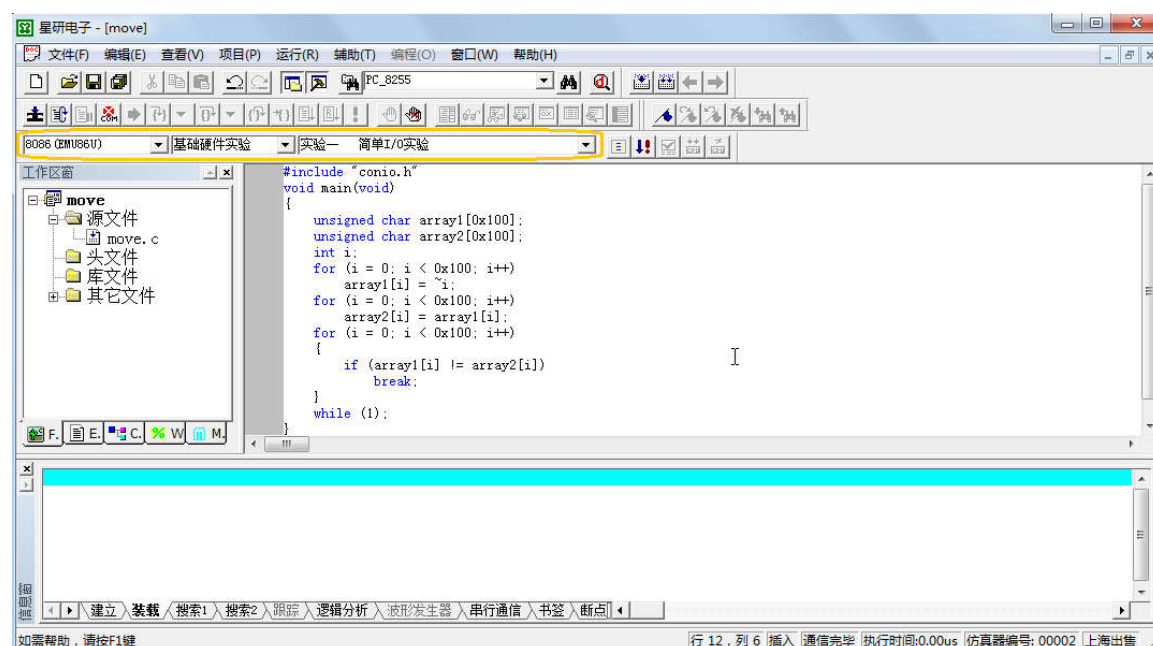



点击图标（功能键 F9），CPU 全速断点运行到光标处停下，如果在第 18 行上 CPU 停止运行，表示运行结果正确；如果 CPU 运行到第 19 行，停止运行，与期望值不符，表示程序有问题，将光标移到第 13 行上（具体操作是：用鼠标点击该行，然后再点击图标），使用单步进入命令 F7 或连续单步进入命令 Ctrl + F7，检查结果，判断程序出错原因。

3.3 实验连线、演示实验、测试实验仪


演示实验

选择仿真器或仿真模块时，必须正确选择购买的实验仪，选择完毕后，会出现一个工具条



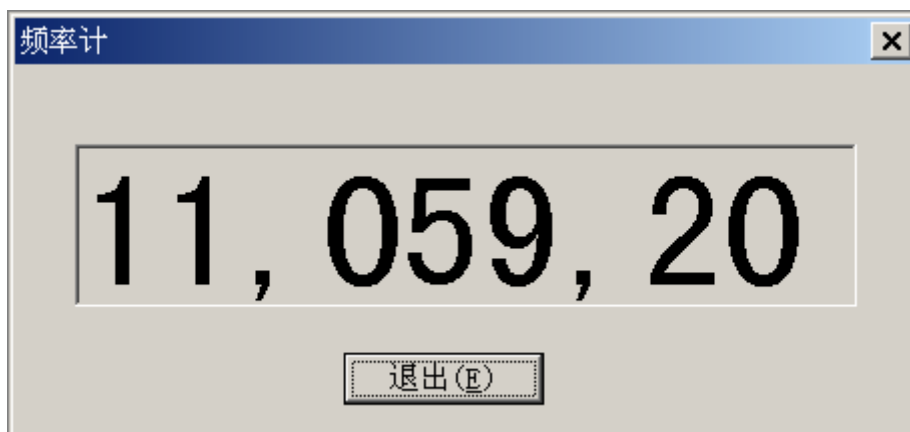
首先选择哪一类实验，例如：软件实验、基础硬件实验、综合实验；然后选择实验，点击, 显示该实验对应的连线说明：



点击, 星研软件自动将该实验的机器码装入实验仪, 并全速运行该程序; 如果按照上述连线后, 没有出现实验结果, 可以怀疑与该实验相关的芯片出问题。

3. 4 频率计 (EMU86)

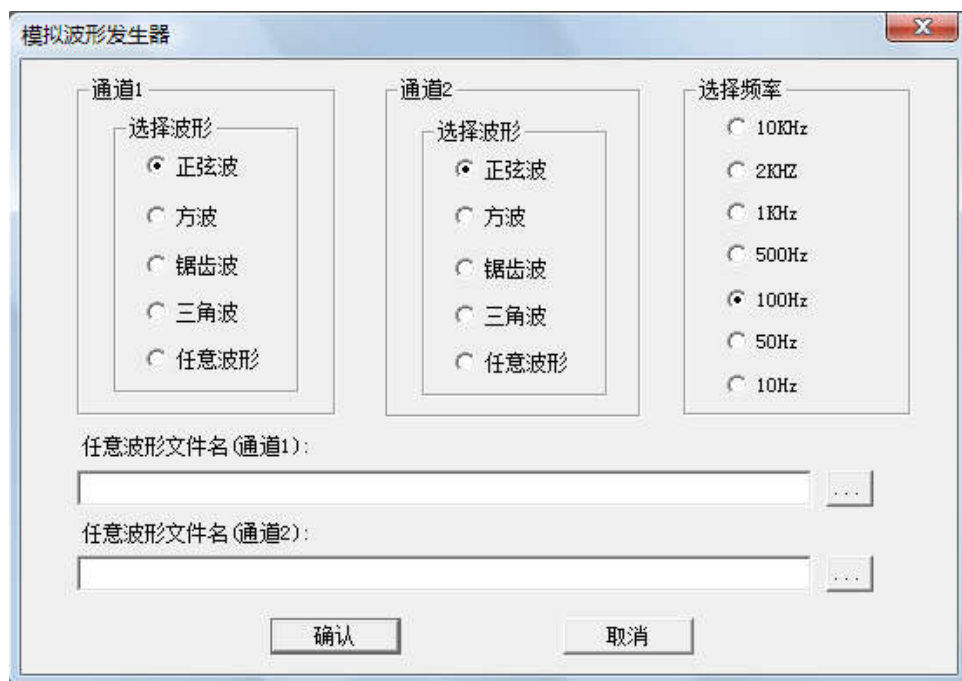
如果您需要测试 CPU 的振荡频率、您电路中其它信号的频率, 在调试状态, 您可以选择频率计功能: [主菜单 » 分析手段 » 频率计]。它可以测试 100M 以内的信号。



EMU86 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 (EMU86)

EMU86 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]。



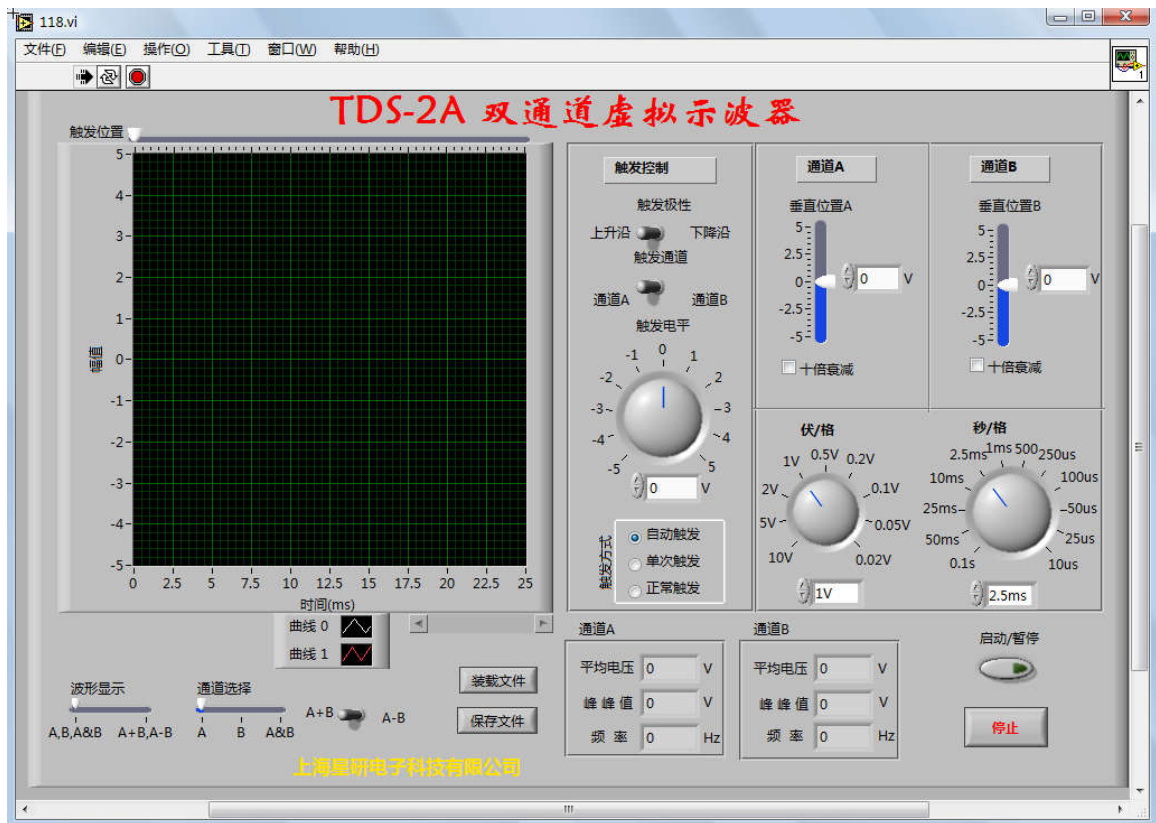
EMU86 仿真模块的 W1、W2 对应于通道 1、通道 2。

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（EMU86）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(EMU86)”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。

3. 7 Debug（行命令方式）

星研集成环境软件提供类似于 Dos 中行命令调试软件 Debug 的功能，操作完全一样，支持 R、D、E、F、M、I、O、A、U、G、T 命令。

运行星研集成环境软件，点击工具条的 ，进入调试状态，点击“信息窗”的“Debug”标签，相当于在 Dos 中启动了 Debug 命令。在 Debug 视中，可执行 R、D、E、F、M、I、O、A、U、G、T 命令，观察、调试程序。

4 实验

本实验课程学时为 32 个学时，共设计了十四个实验，其中实验一~实验八为必做实验，分配 24 个学时，实验九~实验十四为选做实验，任一实验小组从这六个实验中选做一个实验，分配 8 个学时。通过对这些实验程序的编写、调试，使学生掌握 8086 的指令系统、存储器的连接和 I/O 接口电路等，了解程序设计过程，掌握汇编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

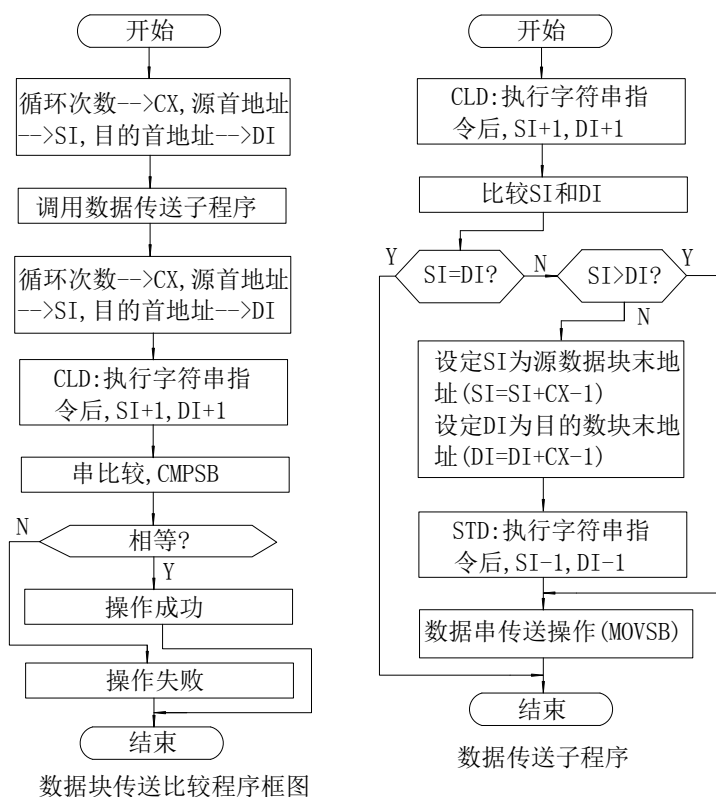
一、实验目的

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

```

_STACK      SEGMENT      STACK
              DW          100 DUP(?)

_STACK      ENDS
DATA        SEGMENT
DATA        ENDS
CODE        SEGMENT
START       PROC          NEAR
              ASSUME      CS:CODE, DS:DATA, SS:_STACK
              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
              REP         MOVSB
Move1:       REP         MOVSB
Return:      RET
Move        ENDP
START       ENDP
CODE        ENDS
              END          START

```

六、思考题

- 1、子程序 Move 中为什么比较 SI、DI？

七、实验报告

- 1、单步运行各条指令并记录相关寄存器的数值。
- 2、注释每一条指令的功能

实验二 四字节十六进制数转十进制数

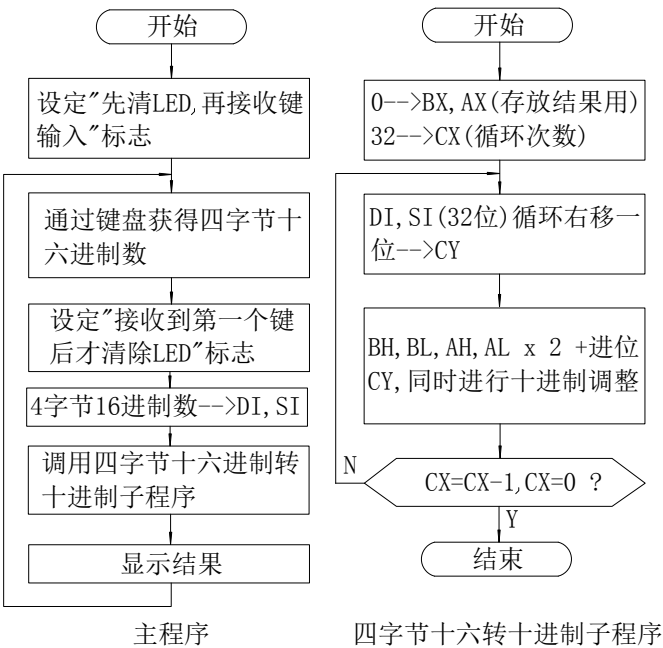
一、实验目的

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：CS、A0、A1	——	A3 区：CS1、A0、A1
D3 区：PC0、PC1	——	F5 区：KL1、KL2
D3 区：JP20、B、C	——	F5 区：A、B、C

2、在 F5 区的键盘上输入 8 位十六进制数

3、结果显示在 F5 区的数码管上

五、程序清单

```
EXTRN      InitKeyDisplay:NEAR, Display8:NEAR, GetKey:NEAR
EXTRN      F1:BYTE
_STACK     SEGMENT      STACK
```

```

        DW          1000 DUP(?)
_STACK
ENDS

_DATA
SENMENT WORD PUBLIC 'DATA'
BUFFER DB          8 DUP(?)
_DATA
ENDS

CODE
START PROC NEAR
        ASSUME CS:CODE, DS:_DATA, SS:_STACK
        MOV      AX, _DATA
        MOV      DS, AX
        MOV      ES, AX
        NOP
        CALL     InitKeyDisplay      ;对键盘、数码管扫描控制器8255初始化
        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

START          ENDP
CODE           ENDS
END            START

```


六、实验扩展及思考

- 1、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。
- 2、绘制本实验的详细实验电路图。
- 3、注释每各条指令的功能

实验三 简单 I/O (16 位) 实验

一、实验目的与要求

- 1、了解外设的扩展方法，掌握外设的读写时序。
- 2、了解 74HC273、74HC244 的功能，掌握它们的使用方法。
- 3、掌握 CPU 对 16 位外设的访问方法
- 4、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

SUN 系列实验仪一套、PC 机一台

三、实验内容

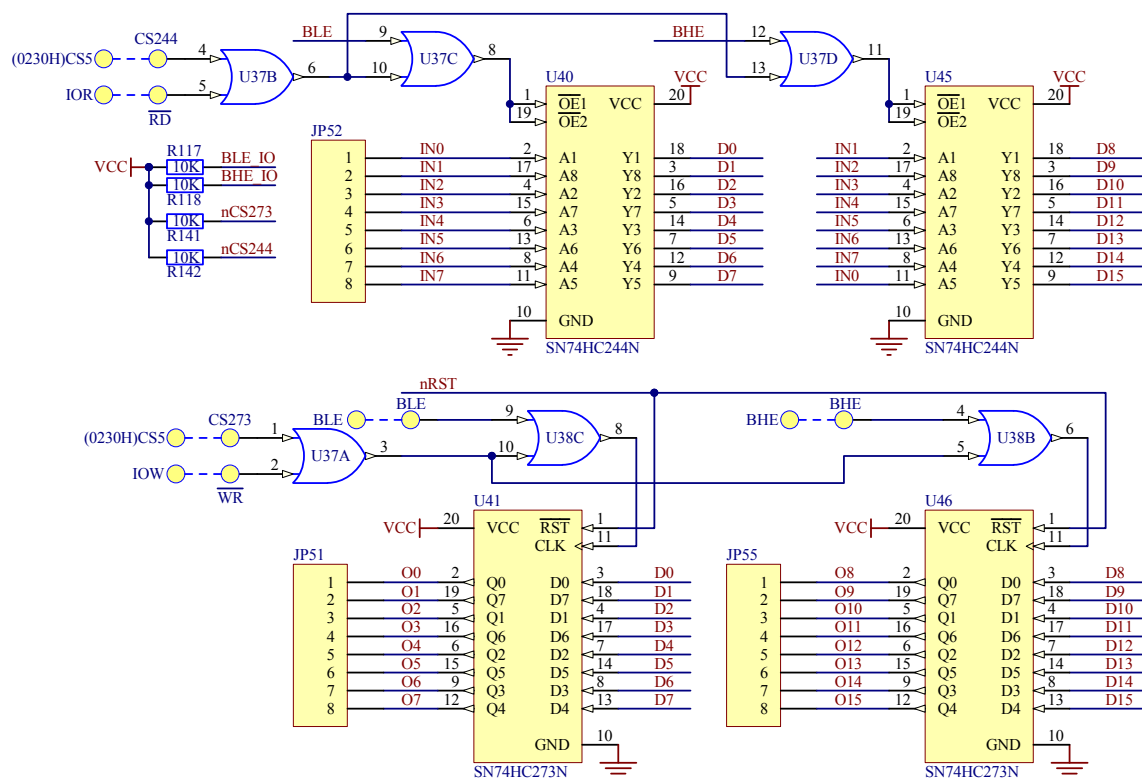
1、说明：二片 74HC244 组成 16 位的只读外设，二片 74HC273 组成 16 位的只写外设，它们都可以按字节或字方式操作。实验仪具有 16 位数据总线 D0..D15、BLE（低电平有效，选中挂在低 8 位数据总线上外设）、BHE（低电平有效，选中挂在高 8 位数据总线上外设）；BLE、BHE 同时有效，对外设字方式读写，BLE 或 BHE 有效，对外设字节方式读写。

二片 74HC273 的输出端与 F4 区的 16 个发光二极管相连；低位 74HC244 的输入端与 F4 区的 8 个拨动开关相连，8 个拨动开关循环左移一位后与高位 74HC244 的输入端相连。

2、编写程序：将 B4 区的二片 74HC244 中数据读出、写入二片 74HC273 中；然后逐一点亮挂在 74HC273 上的 16 个发光二极管；循环执行

3、连接线路验证功能，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(I/O) 区：CS273、CS244	——	A3 区：CS5、CS5
B4(I/O) 区：BLE、BHE	——	A3 区：BLE、BHE
B4(I/O) 区：RD、WR	——	A3 区：IOR、IOW
B4(I/O) 区：JP51、JP55	——	F4 区：JP18、JP19(发光管)
B4(I/O) 区：JP52	——	F4 区：JP27（开关）
B4 区：JP57(D0..D7)	——	A3 区：JP42(D0..D7)
B4 区：JP56(D8..D15)	——	A3 区：JP40(D8..D15)

2、观察实验结果，拨动开关状态是否与点亮的发光二极管一致，是否循环点亮 16 个发光二极管。

六、演示程序

```

I0244      EQU      0230H      ;244(16位)片选
I0273      EQU      0230H      ;273(16位)片选
_STACK     SEGMENT      STACK
            DW      100 DUP(?)

_STACK     ENDS
_DATA      SEGMENT      WORD PUBLIC 'DATA'
_DATA      ENDS
CODE       SEGMENT
START      PROC          NEAR
            ASSUME      CS:CODE, DS:_DATA, SS:_STACK
            MOV         AX, _DATA
            MOV         DX, I0244
            IN          AX, DX      ;读取开关数据(16位, K0 K7 K6 K5 K4 K3
                                   K2 K1 K7 K6 K5 K4 K3 K2 K1 K0)

            MOV         DX, I0273
            OUT         DX, AX
            CALL        Delay
            CALL        Delay
            CALL        Delay
            CALL        Delay
            CALL        Delay
            CALL        Delay
            CALL        Delay
            MOV         DX, I0273
            MOV         AX, 0FFFEH
START1:    OUT         DX, AX
            CALL        Delay
            TEST        AX, 8000H
            JZ          START
            ROL         AX, 1
            JMP         START1
Delay      PROC          NEAR      ;延时
Delay1:    XOR         CX, CX

```

	LOOP	\$
	RET	
Delay	ENDP	
START	ENDP	
CODE	ENDS	
	END	START

七、实验扩展及思考

- 1、请按照字、字节方式画出读（74HC244）写（74HC273）的时序。
- 2、以上程序中，使用 16 位方式读写外设，请按照 8 位方式，重编程序。
- 4、绘制本实验的详细实验电路图。
- 5、注释每各条指令的功能

实验四 静态存贮器(16 位)读写实验

一、实验目的与要求

- 1、了解静态存贮器的特性、扩展方法，掌握存贮器的读写时序。
- 2、掌握 CPU 对 16 位静态存贮器的访问方法
- 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

SUN 系列实验仪一套、PC 机一台

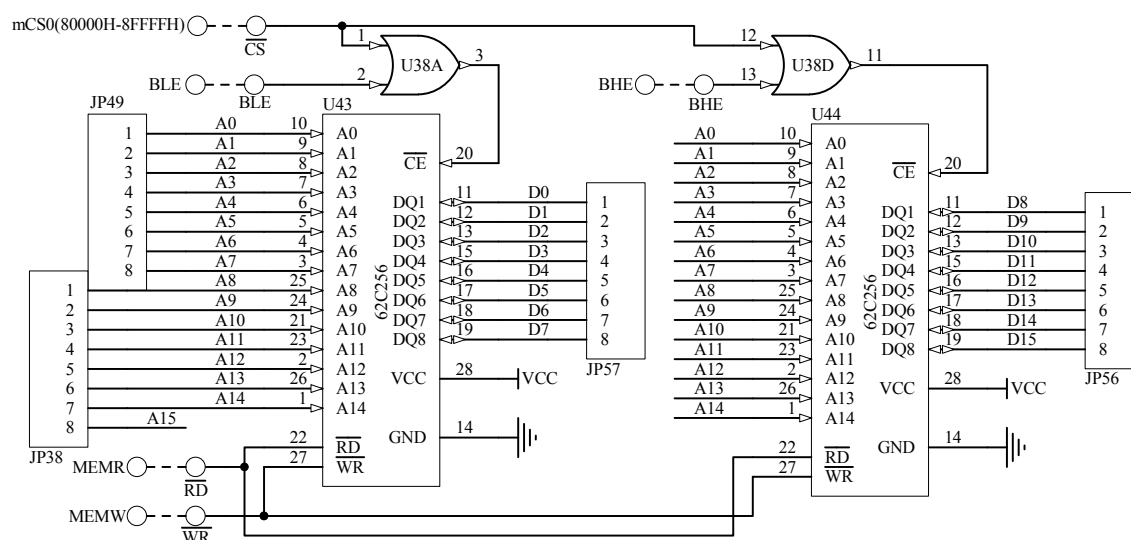
三、实验内容

1、说明：实验仪选用常用的静态存贮器芯片 62256（32K×8 位），二片组成 32K×16 位，共 64K 字节。实验仪具有 16 位数据总线 D0..D15、20 位地址线 A0..A19、BLE（低电平有效，选择低 8 位存贮器芯片）、BHE（低电平有效，选择高 8 位存贮器芯片）；BLE、BHE 同时有效，对存贮器字方式读写，BLE 或 BHE 有效，对存贮器字节方式读写。扩展 16 位存贮器时，不使用地址线 A0。

2、编写程序：将 B4 区的静态存贮器 3000H-30FFH 单元中数据复制到 6000H-60FFH 的单元中，并校验，检测写入的数据是否正确。

3、连接线路验证功能，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 (RAM) 区：CS、BLE、BHE	——	A3 区：mCS0、BLE、BHE
B4 (RAM) 区：RD、WR	——	A3 区：MEMR、MEMW
B4 (RAM) 区：JP49 (A0..A7)	——	A3 区：JP29 (A1..A8)
B4 (RAM) 区：JP38 (A8..A15)	——	A3 区：JP33 (A9..A16)
B4 (RAM) 区：JP57 (D0..D7)	——	A3 区：JP42 (D0..D7)
B4 (RAM) 区：JP56 (D8..D15)	——	A3 区：JP40 (D8..D15)

2、通过星研软件的存储器窗、寄存器窗等，观察运行结果。

六、演示程序

;数据块移动(16位存储器实验)

```

_STACK      SEGMENT      STACK
              DW          100 DUP(?)

_STACK      ENDS
_DATA      SEGMENT      WORD PUBLIC 'DATA'
_DATA      ENDS
CODE      SEGMENT
START      PROC          NEAR
              ASSUME      CS:CODE, DS:_DATA, SS:_STACK
              MOV         AX, 8000H          ;存储器扩展空间段地址
              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
              START      ENDP
              CODE        ENDS
              END         START

```

七、实验扩展及思考

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、本实验与软件实验一的异同点。

3、绘制本实验的详细实验电路图。

4、注释每条指令的功能

实验五 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 8086 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。

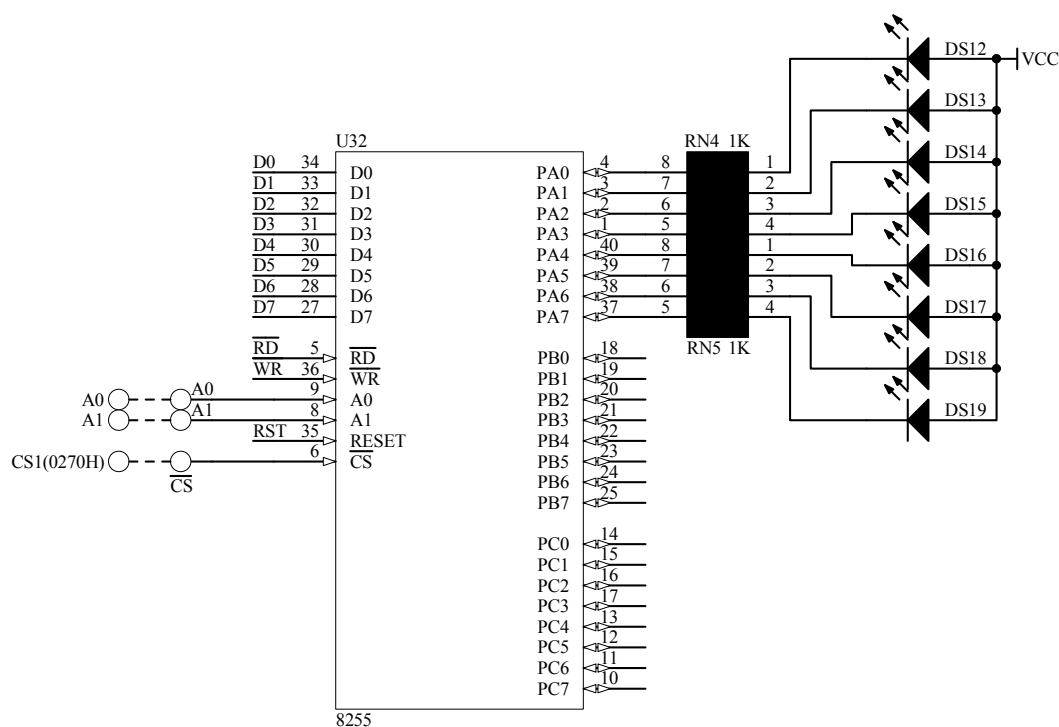
二、实验设备

SUN 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：

D3 区：CS、A0、A1	——	A3 区：CS1、A0、A1
D3 区：JP23 (PA 口)	——	F4 区：JP18

- 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

```
COM_ADD      EQU      0273H
PA_ADD       EQU      0270H
PB_ADD       EQU      0271H
PC_ADD       EQU      0272H
_STACK      SEGMENT STACK
```



```

        DW          100 DUP(?)
_STACK
ENDS
_DATA
SEGMENT WORD PUBLIC 'DATA'
LED_Data
DB          10111110B          ;东西绿灯，南北红灯
DB          10111111B          ;东西绿灯闪烁，南北红灯
DB          10111101B          ;东西黄灯亮，南北红灯
DB          11101011B          ;东西红灯，南北绿灯
DB          11111011B          ;东西红灯，南北绿灯闪烁
DB          11011011B          ;东西红灯，南北黄灯亮

_DATA
ENDS
CODE
SEGMENT
START
PROC      NEAR
ASSUME    CS:CODE, DS:_DATA, SS:_STACK
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		
START	ENDP		
CODE	ENDS		
	END	START	

七、实验扩展及思考

1、 如何对 8255 的 PC 口进行位操作？

- 2、绘制本实验的详细实验电路图。
- 3、注释每各条指令的功能

实验六 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

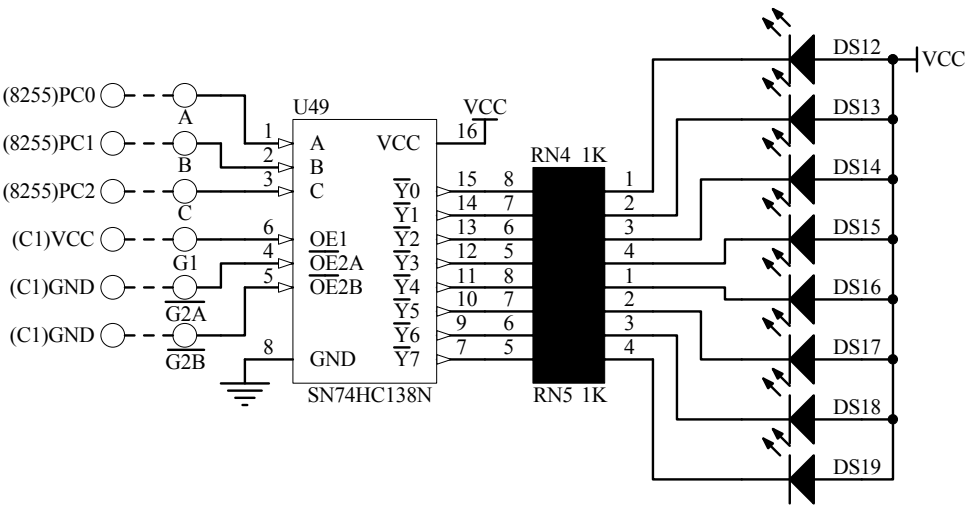
二、实验设备

SUN 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:
 - F7 区: A、B、C ——— D3 区: PC0、PC1、PC2
 - F7 区: G1、G2A、G2B ——— C1 区: VCC、GND、GND
 - F7 区: JP63 ——— F4 区: JP18 (LED 指示灯)
 - D3 区: CS、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

```
Con_8255      EQU      0273H      ;8255控制口
PC_8255       EQU      0272H      ;8255 PC口
_STACK        SEGMENT STACK
DW            100 DUP(?)
```

```

_STACK      ENDS
CODE        SEGMENT
START       PROC      NEAR
            ASSUME     CS:CODE, SS:_STACK
            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
START       ENDP
CODE        ENDS
            END        START

```

七、实验扩展及思考

- 1、 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？
- 2、 绘制本实验的详细实验电路图。
- 3、 注释每各条指令的功能

实验七 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 8086 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

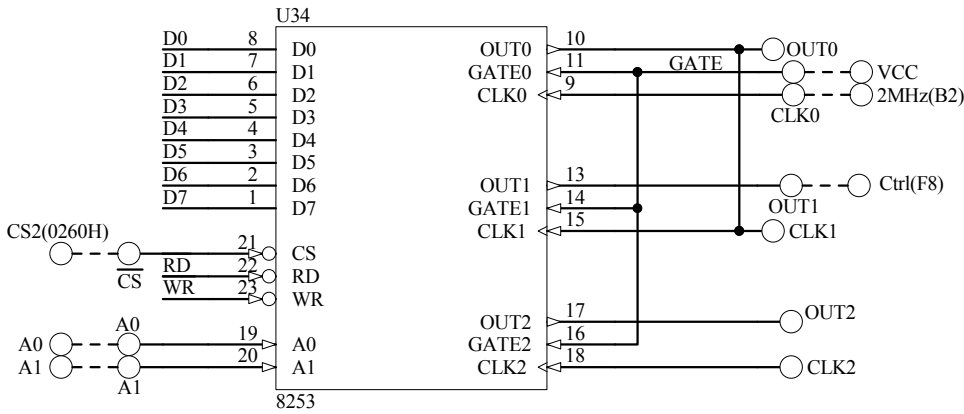
SUN 系列实验仪一套、PC 机一台

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：

C4 区：CS、A0、A1	——	A3 区：CS2、A0、A1
C4 区：CLK0	——	B2 区：2M
C4 区：OUT0	——	C4 区：CLK1
C4 区：OUT1	——	F8 区：Ctrl(蜂鸣器)
C4 区：GATE	——	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

```
COM_ADDR EQU 0263H
T0_ADDR EQU 0260H
T1_ADDR EQU 0261H
```

```
_STACK SEGMENT STACK
```

```

                                DW          100 DUP(?)
_STACK                          ENDS
CODE                            SEGMENT
START                          PROC      NEAR
                                ASSUME     CS:CODE, SS:_STACK
                                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, T0_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
START                          ENDP
CODE                            ENDS
                                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。
- 3、绘制本实验的详细实验电路图。
- 4、注释每各条指令的功能

实验八 8086 中断实验

一、实验目的与要求

- 1、了解 8086 内部响应中断的机制；掌握中断向量的作用。
- 2、利用实验仪上单脉冲、74HC244 电路，不使用 8259，实现一个中断实例。
- 3、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

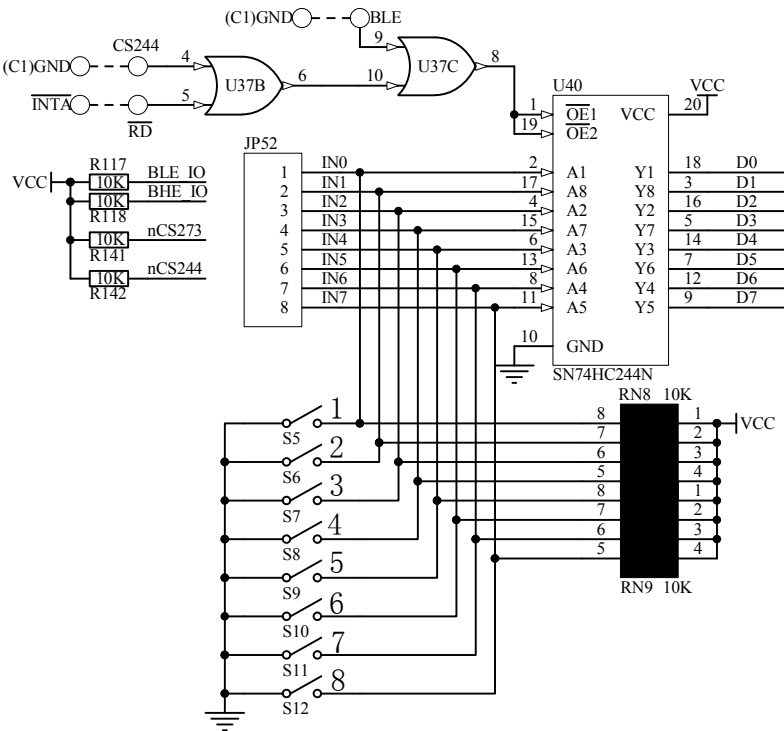
SUN 系列实验仪一套、PC 机一台

三、实验内容

1、编制程序：拨动单脉冲开关，“**11**”送给 8086 的 INTR，触发中断；8086 通过 INTA 信号，读取中断向量；8086 计数中断次数，显示于 F5 区的数码管上

注意：给 INTR 高电平信号，8086 就会相应中断，所以实验开始前，保证单脉冲开关给 8086 低电平；中断程序中，加一个较长的延时程序，在中断结束前，有时间拨动单脉冲开关，恢复给 8086 低电平。

四、实验原理图



本实验，通过 F4 区的 8 个拨动开关，给 74HC244 设定中断向量；本实验的中断向量是 08H，即 IN7-IN0 位数据是 00001000。同学可以自定义中断向量，实验程序中处理中断向量部分程序作相应调整


五、实验步骤

1、连线说明：

B4 区：CS244、BLE	——	C1 区：GND
B4 区：RD (IO 区)	——	A3 区：INTA
A3 区：INTR	——	B2 区：单脉冲 11
B4 区：JP57 (D0..D7)	——	A3 区：JP41
B4 区：JP52 (IN0..7)	——	F4 区：JP27 (1..8)

D3 区: CS、A0、A1	——	A3 区: CS1、A0、A1
D3 区: PC0、PC1	——	F5 区: KL1、KL2
D3 区: JP20、B、C	——	F5 区: A、B、C

2、运行程序

3、实验开始前，保证单脉冲开关给 8086 低电平；运行程序；向下拨动开关（触发中断），立即向上拨动开关，产生一个“”，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

```

EXTRN          InitKeyDisplay:NEAR, Display8:NEAR
_STACK         SEGMENT          STACK
                DW               100 DUP(?)

_STACK         ENDS
_DATA         SEGMENT          WORD PUBLIC 'DATA'
BUFFER        DB               8 DUP(?)
Counter        DB               ?
ReDisplayFlag  DB               0
_DATA         ENDS

CODE          SEGMENT
START         PROC              NEAR
                ASSUME          CS:CODE, DS:_DATA, SS:_STACK
                MOV              AX, _DATA
                MOV              DS, AX
                MOV              ES, AX
                NOP
                CALL             InitKeyDisplay      ;对键盘、数码管控制器8255初始化
                CALL             WriIntver
                MOV              Counter, 0          ;中断次数
                MOV              ReDisplayFlag, 1    ;需要显示
                STI               ;开中断

START1:       LEA               SI, Buffer
                CALL             Display8
                CMP              ReDisplayFlag, 0
                JZ               START1
                CALL             LedDisplay
                MOV              ReDisplayFlag, 0
                JMP              START1

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	
	RET		
LedDisplay	ENDP		
INT_0:	PUSH	DX	
	PUSH	AX	
	MOV	AL, Counter	
	ADD	AL, 1	
	DAA		
	MOV	Counter, AL	
	MOV	ReDisplayFlag, 1	
	CALL	LedDisplay	
DELAY:	PUSH	BX	
	PUSH	CX	
	PUSH	DI	
	PUSH	SI	
	MOV	CX, 20	
DELAY1:	LEA	SI, Buffer	
	CALL	Display8	
	loop	DELAY1	
	POP	SI	
	POP	DI	
	POP	CX	
	POP	BX	
	POP	AX	

	POP	DX
	IRET	
START	ENDP	
CODE	ENDS	
	END	START

七、实验扩展及思考

- 1、绘制本实验的详细实验电路图。
- 2、扼要注释每指令的功能

实验九 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

SUN 系列实验仪一套、PC 机一台。

三、实验内容

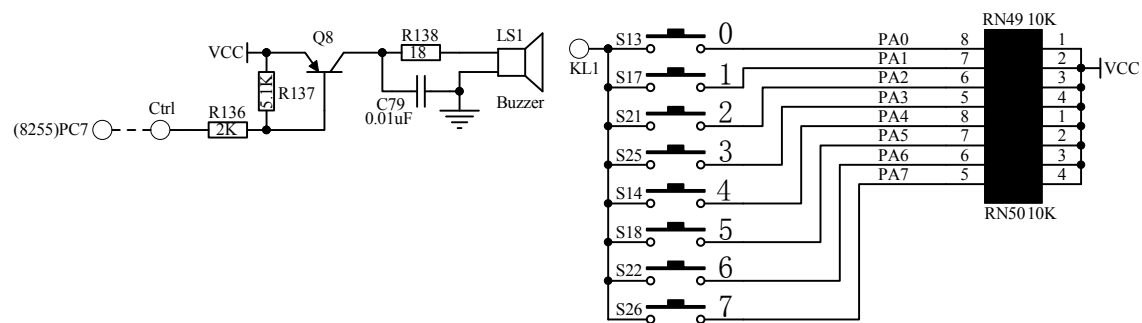
1、简易电子琴原理：

- (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 CPU 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：

D3 区：CS、A0、A1	——	A3 区：CS1、A0、A1
D3 区：PC7	——	F8 区：Ctrl
D3 区：JP23（PA 口）	——	F5 区：JP37(A)
C1 区：GND	——	F5 区：KL1

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

```
C8255          EQU          0273H
PA8255          EQU          0270H
_STACK          SEGMENT      STACK
                DW           100 DUP(?)
_STACK          ENDS
_DATA           SEGMENT      WORD PUBLIC 'DATA'
Music           DW   M1, M2, M3, M4, M5, M6, 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
_DATA
CODE
START  PROC      NEAR
        ASSUME    CS:CODE, DS:_DATA, SS:_STACK
        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, 2
        JZ        START2
        CALL      Music1           ;1 号键, 调 1 号键输出
        JMP       START1
START2: TEST      AL, 4
        JZ        START3
        CALL      Music2           ;2 号键
        JMP       START1
START3: TEST      AL, 8
        JZ        START4
        CALL      Music3           ;3 号键
        JMP       START1
START4: TEST      AL, 10H
        JZ        START5
        CALL      Music4           ;4 号键
        JMP       START1
START5: TEST      AL, 20H
        JZ        START6
        CALL      Music5           ;5 号键
        JMP       START1
START6: TEST      AL, 40H
        JZ        START7
        CALL      Music6           ;6 号键
        JMP       START1
START7: TEST      AL, 80H
        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	T1	;延时 10us
	CALL	T1	;延时 10us
	CALL	W_H	;写 1, 蜂鸣器不响
	CALL	T10	;延时
	CALL	T10	;
	CALL	T5	;
	CALL	T1	;延时 10us
	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

START     ENDP
CODE      ENDS
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。

实验十 LED16 * 16 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。

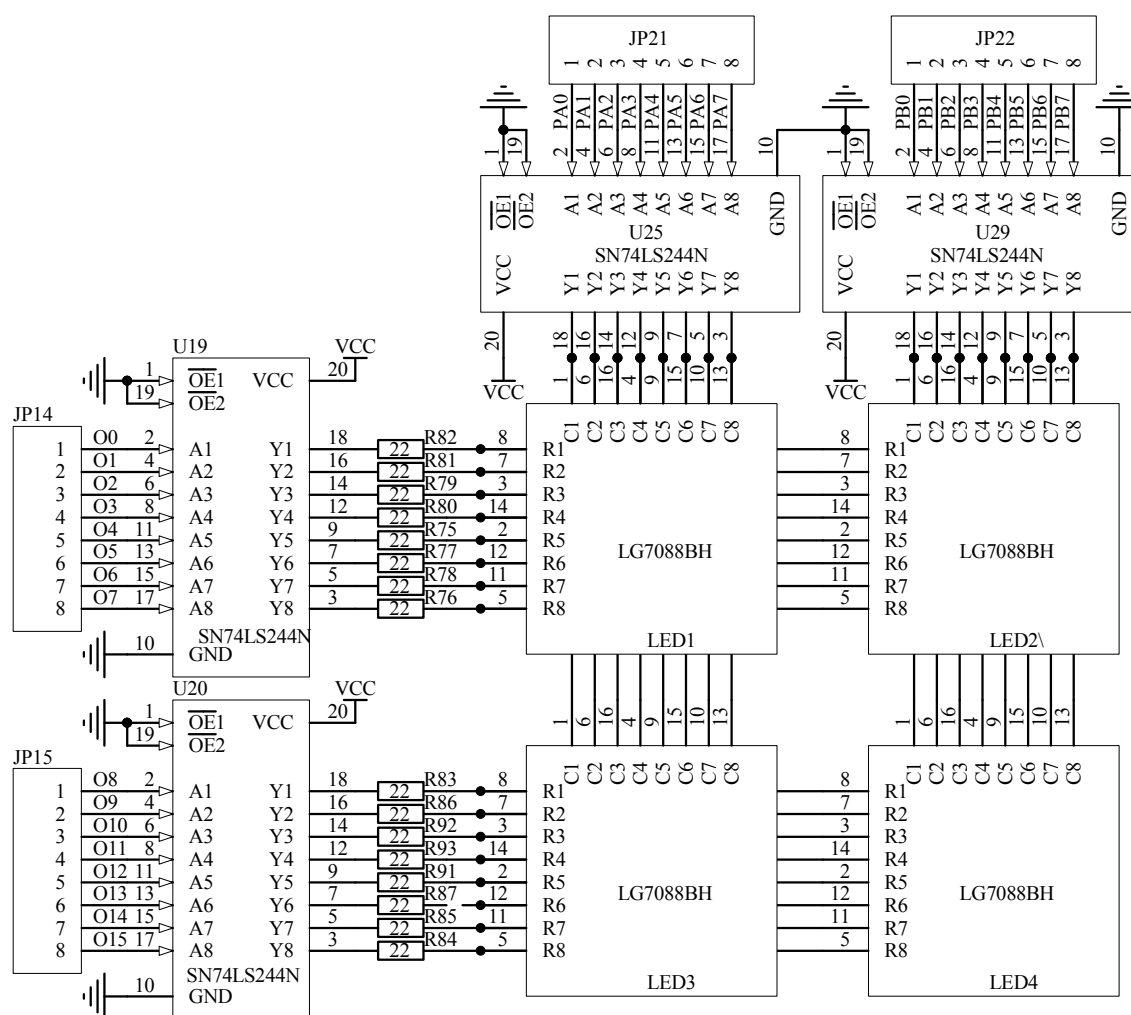
二、实验设备

SUN 系列实验仪一套、PC 机一台。

三、实验内容

- 1、编写程序，用 B4 区的二片 74HC273 控制 16X16 点阵的行；8255 的 PA、PB 口控制 16X16 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：

D3 区: CS (8255)、A0、A1	——	A3 区: CS1、A0、A1
D3 区: JP23(PA)、JP20(PB)	——	A2 区: JP21、JP22 (列输出线)
B4 区: JP57(D0..D7)	——	A3 区: JP42(D0..D7)
B4 区: JP56(D8..D15)	——	A3 区: JP40(D8..D15)
B4(I/O) 区: CS273、BLE、BHE	——	A3 区: CS5、BLE、BHE
B4(I/O) 区: RD、WR	——	A3 区: IOR、IOW
B4(I/O) 区: JP51、JP55	——	A2 区: JP14、JP15 (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“欢迎使用星研实验仪”在点阵上自下而上循环移动显示。

六、演示程序

```

ADDR_8255_PA    EQU            270H                ;8255 PA口
ADDR_8255_PB    EQU            271H                ;8255 PB口
ADDR_8255_C     EQU            273H                ;8255控制口
ADDR_273        EQU            230H                ;IO区74HC273(16位I/O)
LINE            EQU            ADDR_273            ;行线1, 行线2
ROW1            EQU            ADDR_8255_PA        ;列线1
ROW2            EQU            ADDR_8255_PB        ;列线2

_STACK          SEGMENT      STACK
                DW            100 DUP(?)
_STACK          ENDS
_DATA           SEGMENT      WORD PUBLIC '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

```

```

_DATA          ENDS
CODE           SEGMENT
START          PROC      NEAR
                ASSUME   CS:CODE, DS:_DATA, SS:_STACK
                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, LINE
                MOV      AX, BX
                OUT      DX, AX        ;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
	RET		
INIT_IO	ENDP		
CLEAR	PROC	NEAR	
	MOV	AX, 0FFFFH	
	MOV	DX, LINE	
	OUT	DX, AX	
	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, LINE	
	XOR	AX, AX	
	OUT	DX, AX	
	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

START                          ENDP
CODE                           ENDS
                                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。

实验十一 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

SUN 系列实验仪一套、PC 机一台。

三、实验内容

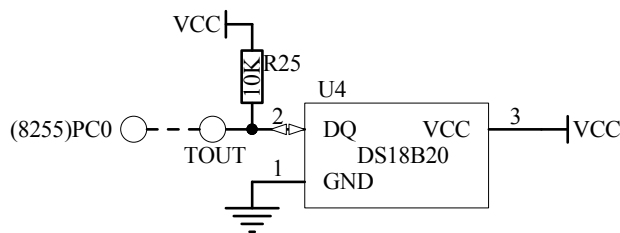
1、DS18B20：

- (1) 一线串行接口数字式温度计
- (2) 温度测量范围 -55°C – 125°C ， -10°C – 85°C 内误差 $\pm 0.5^{\circ}\text{C}$
- (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，B4 区 74HC273 控制 LED (F5 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：

F2 区：TOUT	——	D3 区：PC0 (8255)
D3 区：CS (8255)、A0、A1	——	A3 区：CS1、A0、A1
B4 区：JP57 (D0..D7)	——	A3 区：JP42 (D0..D7)
B4 区：JP56 (D8..D15)	——	A3 区：JP40 (D8..D15)
B4 (I/O) 区：CS273、BLE、BHE	——	A3 区：CS5、BLE、BHE
B4 (I/O) 区：RD、WR	——	A3 区：IOR、IOW
B4 (I/O) 区：JP51、JP55	——	F5 区：B、C

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 (F5 区) 动态显示温度，温度数据通过 DS18B20 获取。可用手指贴住 DS18B20 (G1 区)，温度显示会随之上升。

六. 演示程序

```

Con_8255      EQU      0273H
PC_8255       EQU      0272H
EXTRN         InitKeyDisplay:NEAR, Display8A:NEAR
_STACK        SEGMENT   STACK
                DW      100 DUP(?)
_STACK        ENDS
_DATA         SEGMENT   WORD PUBLIC 'DATA'
    
```

```

buffer      DB      8 DUP(0)          ;温度临时存放区
_DATA      ENDS
CODE       SEGMENT
START      PROC      NEAR
            ASSUME    CS:CODE, DS:_DATA, SS:_STACK
            MOV       AX, _DATA
            MOV       DS, AX
            MOV       ES, AX
            NOP
            CALL      InitDisBuffer    ;给显示缓冲区赋初值，消隐
            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
RET
DIS_BCD  ENDP
;延时程序
DelayTime PROC      NEAR
MOV      CX, 70
L1:      LEA      SI, buffer
CALL     Display8A
LOOP     L1
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
        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
POP        AX
WRI2:      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
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
InitDisBuffer PROC    NEAR
PUSH       DS
POP        ES
LEA        DI, buffer
MOV        AX, 1010H
MOV        CX, 4
CLD
REP        STOSW
RET
InitDisBuffer ENDP

START      ENDP
CODE       ENDS
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

实验十二 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

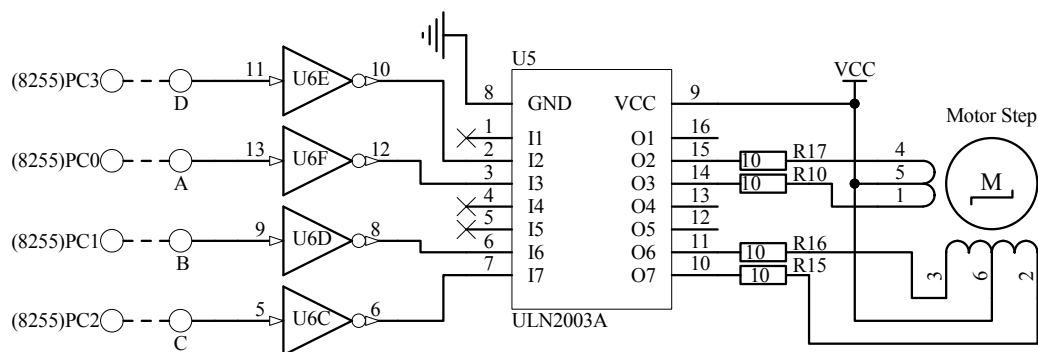
编写程序：使用 F5 区的键盘控制步进电机的正反转、调节转速，连续转动或转动指定步数；将相应的数据显示在 F5 区的数码管上。

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	——	D3 区：PC4、PC5、PC6、PC7
D3 区：CS、A0、A1	——	A3 区：CS1、A0、A1
D3 区：PC0、PC1	——	F5 区：KL1、KL2
D3 区：JP20 (PB)、B、C	——	F5 区：A、B、C
B3 区：CS、A0	——	A3 区：CS3、A0
B3 区：INT、INTA	——	A3 区：INTR、INTA
C4 区：CS (8253)、A0、A1	——	A3 区：CS2、A0、A1
C4 区：GATE	——	C1 区：VCC
C4 区：CLK0	——	B2 区：1M
C4 区：OUT0	——	B3 区：IR0

- 2、调试程序，查看运行结果是否正确

七、演示程序（完整程序见目录 STEP）

- 1、步进电机控制程序（STEP.ASM）

```
EXTRN          InitKeyDisplay:NEAR, GetKeyA:NEAR, Display8:NEAR
```

```

I08259_0      EQU      0250H
I08259_1      EQU      0251H
Con_8253      EQU      0263H
T0_8253       EQU      0260H
I08255_Con    EQU      0273H          ;CS3
I08255_PC     EQU      0272H

_STACK        SEGMENT  STACK
              DW        100 DUP(?)

_STACK        ENDS
_DATA         SEGMENT  WORD PUBLIC '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
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
_DATA         ENDS

CODE          SEGMENT  WORD PUBLIC 'CODE'
START        PROC      NEAR
              ASSUME    CS:CODE, DS:_DATA, SS:_STACK
              MOV       AX, _DATA
              MOV       DS, AX
              MOV       ES, AX
              NOP
              CALL      InitKeyDisplay ;初始化键盘、数码管控制器（8255）
              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
CALL     Display8
STAR3:   CALL     GetKeyA
JB       STAR5
CMP      bNeedDisplay, 0
JZ       STAR3
MOV      bNeedDisplay, 0
CALL     Adjust_Step
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      STAR2
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	TIMERO_1	
	MOV	AL, StartStepDelay1	
	CMP	AL, StepDelay	
	JZ	TIMERO_2	
	DEC	AL	
	MOV	StartStepDelay1, AL	
TIMERO_2:	MOV	StartStepDelay, AL	
	MOV	AL, StepControl	
	MOV	DX, I08255_PC	
	OUT	DX, AL	
	CMP	bClockwise, 0	
	JNZ	TIMERO_3	
	ROR	AL, 1	
	JMP	TIMERO_4	
TIMERO_3:	ROL	AL, 1	
TIMERO_4:	MOV	StepControl, AL	
	CMP	StepCount, 0	
	JZ	TIMERO_1	
	MOV	bNeedDisplay, 1	
	DEC	StepCount	
	JNZ	TIMERO_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		
TIMERO_1:	MOV	DX, I08259_0	
	MOV	AL, 20H	
	OUT	DX, AL	
	POP	DX	
	POP	AX	
	IRET		
Adjust_Step	PROC	NEAR	
	MOV	CX, 4	
	LEA	DI, buffer	
	CLD		
	MOV	AX, StepCount	
	MOV	BX, 10	
Adjust_Step1:	XOR	DX, DX	
	DIV	BX	

	XCHG	AX, DX	
	STOSB		
	MOV	AX, DX	
	LOOP	Adjust_Step1	
	RET		
Adjust_Step	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, 81H	
	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, 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, TIMER0
                                STOSW
                                MOV        AX, CS
                                STOSW
                                POP        ES
                                RET
WriIntver                     ENDP

START                         ENDP
CODE                          ENDS
                                END        START

```

2、键盘、数码管扫描程序(K_D.ASM)，请参阅基础实验九“8255键盘显示实验”

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？

实验十三 X5045 串行 EEPROM 读写实验

一、实验目的与要求

了解 SPI 总线；掌握 SPI 总线的数据读写操作；掌握对 X5045 单字节读写、页写模式和连续读取模式的操作

学会建立项目，使用多个模块文件完成一个功能，便于移植。

二、实验设备

SUN 系列实验仪一套、PC 机一台。

三、实验内容

1、X5045 (D6 区)：

(1) 内置 4K bit 三线串行接口 EEPROM，支持 16 字节页写模式和连续读取功能。

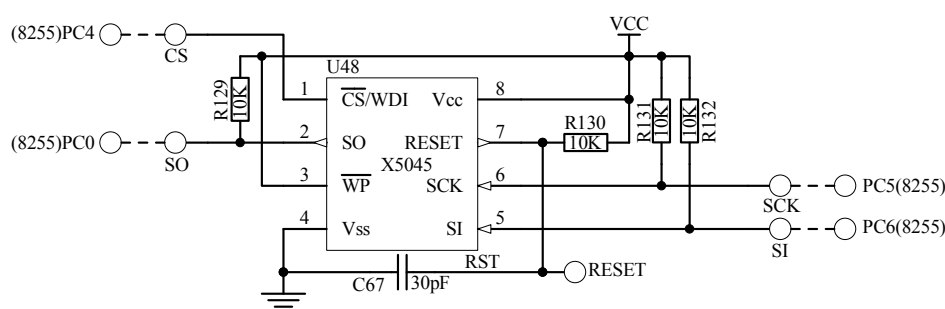
(2) 内置看门狗和低电压检测功能。

2、实验过程

(1) 写满 X5045 内部整个 4K bit 串行 EEPROM，起始写入地址为 00H，之后地址与数据都以 +1 递增，直到写满整个 EEPROM

(2) 检验写入数据是否正确并显示结果，正确：点亮 F4 区的发光管 (DS12-DS19)，错误：熄灭发光管。

四、实验原理图



五、实验步骤

1、主机连线说明：

D3 区：CS、A0、A1	——	A3 区：CS1、A0、A1
D3 区：PC4、PC5、PC6	——	D6 区：CS、SCK、SI
D3 区：PC0	——	D6 区：S0
D3 区：JP23 (PA)	——	F4 区：JP18

2、运行程序，正确：点亮 F4 区的发光管 (DS12-DS19)，错误：熄灭发光管。

六、演示程序 (完整程序见目录 X5045)

```
;CS      PC4      片选，低有效
;SCLK    PC5      时钟输入
;SI      PC6      数据输入
;S0      PC0      数据输出
CON_8255 EQU      0273H
PC_8255 EQU      0272H
;1、X5045 子程序 (X5045.ASM)
;(1)对 CS、SCLK、SI 置 1、清零、读 S0 子程序
```


CS_H	PROC	NEAR	输出高电平
	PUSH	AX	
	MOV	DX, CON_8255	
	MOV	AL, 09H	
	OUT	DX, AL	
	POP	AX	
	RET		
CS_H	ENDP		
CS_L	PROC	NEAR	;输出低电平
	PUSH	AX	
	MOV	DX, CON_8255	
	MOV	AL, 08H	
	OUT	DX, AL	
	POP	AX	
	RET		
CS_L	ENDP		
SCLK_H	PROC	NEAR	;输出高电平
	PUSH	AX	
	MOV	DX, CON_8255	
	MOV	AL, 0BH	
	OUT	DX, AL	
	POP	AX	
	RET		
SCLK_H	ENDP		
SCLK_L	PROC	NEAR	;输出低电平
	PUSH	AX	
	MOV	DX, CON_8255	
	MOV	AL, 0AH	
	OUT	DX, AL	
	POP	AX	
	RET		
SCLK_L	ENDP		
SI_H	PROC	NEAR	;输出高电平
	PUSH	AX	
	MOV	DX, CON_8255	
	MOV	AL, 0DH	
	OUT	DX, AL	
	POP	AX	
	RET		
SI_H	ENDP		
SI_L	PROC	NEAR	;输出低电平
	PUSH	AX	
	MOV	DX, CON_8255	
	MOV	AL, 0CH	

```

        OUT        DX, AL
        POP        AX
        RET
SI_L    ENDP
R_S0    PROC      NEAR
        PUSH      AX
        MOV        DX, PC_8255
        IN         AL, DX
        TEST       AL, 01H
        POP        AX
        RET
R_S0    ENDP
; (2) 串行发送 8 位数据, 用于指令/数据的发送
XTRAN   PROC      NEAR
X_WMid:  PUSH      DX
        PUSH      CX
        PUSH      AX
        MOV        CX, 08H
        MOV        AH, 80H
XTRAN1:  CALL       SCLK_L           ; 上升沿写入
        TEST       AH, AL
        JZ         XTRAN2
        CALL       SI_H
        JMP        XTRAN3
XTRAN2:  CALL       SI_L
XTRAN3:  CALL       SCLK_H
        ROR        AH, 1
        LOOP       XTRAN1
        POP        AX
        POP        CX
        POP        DX
        RET
XTRAN   ENDP
; (2) 串行接收 8 位数据, 用于读取数据
XRECE   PROC      NEAR
X_RMId:  PUSH      DX
        PUSH      CX
        MOV        CX, 08H
        XOR        AL, AL
XRECE1:  ROL        AL, 01H
        CALL       SCLK_H           ; 下降沿读出
        CALL       SCLK_L
        CALL       R_S0
        JZ         XRECE2

```

```

                OR            AL, 01H
XRECE2:         LOOP         XRECE1
                POP          CX
                POP          DX
                RET
XRECE          ENDP
; (3)读状态
X_RSDR         PROC         NEAR
                CALL         SCLK_L           ;读状态，用于查忙
                CALL         CS_L
                MOV          AL, 05H
                CALL         XTRAN
                CALL         XRECE
                CALL         CS_H
                RET
X_RSDR         ENDP
; (4)写状态，控制看门狗，存储内容的保护范围
X_WRSR         PROC         NEAR
                PUSH         AX
                CALL         X_WREN           ;必须先开启允许写
                CALL         SCLK_L
                CALL         CS_L
                MOV          AL, 01H           ;写状态寄存器命令 01H
                CALL         XTRAN
                POP          AX               ;A 中放设置命令
                CALL         XTRAN
                CALL         CS_H
X_WRSR1:       CALL         X_RSDR
                TEST         AL, 01H
                JNZ          X_WRSR1
                RET
X_WRSR         ENDP
; (5)允许写操作
X_WREN         PROC         NEAR
                CALL         SCLK_L
                CALL         CS_L
                MOV          AL, 06H           ;允许写指令 06H
                CALL         XTRAN           ;发送 8 位数据子程序
                CALL         CS_H
                RET
X_WREN         ENDP
; (6)禁止写操作
X_WRDI         PROC         NEAR
                CALL         SCLK_L

```

```

CALL CS_L
MOV AL, 04H ;禁止写指令 04H
CALL XTRAN ;发送 8 位数据子程序
CALL CS_H
RET
X_WRDI ENDP
; (7) 开始写
X_WBeg PROC NEAR
PUSH AX
MOV AL, 02H ;写低 256 个字节数据
JNB X_WBeg1
MOV AL, 0AH
X_WBeg1: PUSH AX
CALL X_WREN ;每次要写入数据，都必须开允许写操作
CALL SCLK_L ;写操作
CALL CS_L
POP AX
CALL XTRAN
POP AX ;起始地址
CALL XTRAN
RET
X_WBeg ENDP
; (8) 写数据
; X_WMid 与 XTRAN 完全一样
; (9) 结束写
X_WEnd PROC NEAR
CALL CS_H ;写数据结束后，关闭允许写
X_WEnd1: CALL X_RSDDR
TEST AL, 01H
JNZ X_WEnd1
RET
X_WEnd ENDP
; (10) 开始读
X_RBeg PROC NEAR
CALL SCLK_L
CALL CS_L
PUSH AX
MOV AL, 03H ;读低 2K Bit 数据
JNB X_RBeg1
MOV AL, 0BH ;读高 2K Bit 数据
X_RBeg1: CALL XTRAN
POP AX ;起始地址
CALL XTRAN
RET

```

```

X_RBeg      ENDP
; (11)读数据
; X_RMid 与 XRECE 完全一样
; (12)结束读
X_REnd      PROC      NEAR
              CALL      CS_H
              RET

X_REnd
;测试 X5045
;写数据，X5045 内的串行 EEPROM 分为高、低各 256 个字节，要分别写入
WLOOP1      EQU      10H      ;写内循环次数
WLOOP2      EQU      20H      ;写外循环次数
RLOOP1      EQU      10H      ;读内循环次数
RLOOP2      EQU      20H      ;读外循环次数
T_Data      EQU      0FEH      ;起始数据 FEH
T_X5045_WR  PROC      NEAR
              MOV      AH, T_Data      ;起始数据，之后不数据值会不断增加
              MOV      BL, 00H      ;串行 EEPROM 地址起始地址
              MOV      CX, WLOOP2      ;外循环次数 20H
              MOV      AL, 30H
              CALL      X_WRSR      ;写状态寄存器
XWR:         CLC      ;低 256 字节
              CMP      CX, 11H
              JNB      XWR1
              STC      ;高 256 字节
XWR1:        PUSH     CX
              MOV      CX, WLOOP1      ;内循环次数 10H
              MOV      AL, BL
              CALL      X_WBeg
XWR2:        MOV      AL, AH      ;欲写数据放入 A
              INC      AH
              CALL      X_WMid      ;调用写操作子程序
              LOOP     XWR2
              CALL      X_WEnd      ;结束本次写操作
              ADD      BL, WLOOP1
              POP      CX
              LOOP     XWR
              RET
T_X5045_WR  ENDP
;读数据，可以一次连续地读出高、低 256 字节串行 EEPROM 里的数据，并检验
T_X5045_RD  PROC      NEAR
              LEA      DI, Buffer5045
              MOV      AH, T_Data      ;起始检验数据
              MOV      BL, 00H      ;串行 EEPROM 地址起始地址

```

```

MOV      CX, RLOOP2      ;外循环次数 20H
CLC
MOV      AL, BL          ;从低 256 字节的 00H 开始读
CALL     X_RBeg          ;开始读
XRD1:    PUSH     CX
MOV      CX, RLOOP1      ;内循环次数 10H
XRD2:    CALL     X_RMid   ;调用读操作子程序
STOSB
CMP      AL, AH
JNZ      XRD3            ;比较判断写入的数据是否正确
INC      AH
LOOP     XRD2
POP      CX
LOOP     XRD1
CALL     X_REnd          ;读数据结束
CLC      ;为测试结果的判断标志，C=0 正确；C=1 错误
RET
XRD3:    CALL     X_REnd
POP      CX
STC
RET
T_X5045_RD  ENDP

```

七、实验扩展及思考

实验内容：上面实验采用连续读写方式操作，现在对 X5045 中的串行 EEPROM 进行单字节的写操作和读操作，进一步熟练串行数据的读写，有兴趣者可尝试。

实验十四 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 F5 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：CS、A0、A1	——	A3 区：CS1、A0、A1
D3 区：PC0、PC1	——	F5 区：KL1、KL2
D3 区：JP20 (PB)、B、C	——	F5 区：A、B、C
B3 区：CS、A0	——	A3 区：CS3、A0
B3 区：INT、INTA	——	A3 区：INTR、INTA
B3 区：IR0	——	C4 区：OUT0
C4 区：CS (8253)、A0、A1	——	A3 区：CS2、A0、A1
C4 区：GATE	——	C1 区：VCC
C4 区：CLK0	——	B2 区：62.5K

- 2、运行程序，按 F5 区的 F 键，设置时钟初值；
- 3、观察 F5 区数码管上显示的时间是否正确。

五、演示程序

```

EXTRN          InitKeyDisplay:NEAR, Display8:NEAR, GetKeyA:NEAR, GetKeyB:NEAR
I08259_0        EQU            0250H
I08259_1        EQU            0251H
Con_8253        EQU            0263H
T0_8253         EQU            0260H

_STACK         SEGMENT         STACK
                DW              200 DUP(?)
_STACK         ENDS

_DATA          SEGMENT         WORD PUBLIC '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
_DATA      ENDS

CODE        SEGMENT
START       PROC      NEAR
            ASSUME     CS:CODE, DS:_DATA, SS:_STACK
            MOV        AX, _DATA
            MOV        DS, AX
            MOV        ES, AX
            NOP
            CALL       InitKeyDisplay ;对键盘、数码管扫描控制器8255初始化
            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, 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, Timer0Int	
	STOSW		
	MOV	AX, CS	
	STOSW		
	POP	ES	
	RET		
WriIntver	ENDP		
START	ENDP		
CODE	ENDS		
	END	START	

附录 A 星研集成环境软件支持的软中断

输 入	功 能	输 出
INT 21H 软中断		
AH = 01H, AL = 00H	带回显的字符输入 (无字符输入时将等待输入)	AL = 键值
AH = 01H, AL = 01H	带回显的 16 进制数输入 (无输入时将等待输入)	AL = 键值
AH = 01H, AL = 02H	带回显的 10 进制数输入 (无输入时将等待输入)	AL = 键值
AH = 02H, DL = 8 位数据(通常是 ASCII 代码)	字符输出(输出一字符到信息窗的 Dos 标签视中)	
AH = 06H, DL=0FFH	请求输入 (它与 INT 21H, AH=01H 区别, 它 不会等待键输入)	如果之前有键按下: 零标志=清除, AL=键值; 如果之前没有键按下, 零标志=置 1
AH = 06H, DL= 00H-0FEH(通常是 ASCII 代码)	字符输出(输出一字符到信息窗的 Dos 标签视中)	
AH = 09H DS: DX = 段: 偏移地址	输出字符串(送一字符串到信息窗 的 Dos 标签视中, 字符串以 ' \$ ' 字 符(24H)结尾)	
AH = 0AH AL = 0(接收任意字符) DS: DX = 段: 偏移地址 缓冲区的第一个字节存放它能保 存的最大字符数(1 至 255)	缓冲输入(从键盘读一行并放入用 户定义的缓冲区)	缓冲区的第一个字节说明它能保 存的最大字符数(1 至 255), 该值 由用户设置, 第二个字节返回实际 输入的字符数(回车除外)
AH = 0AH AL = 0(接收 16 进制数) DS: DX = 段: 偏移地址 缓冲区的第一个字节存放它能保 存的最大字符数(1 至 255)	缓冲输入(从键盘读一行并放入用 户定义的缓冲区)	缓冲区的第一个字节说明它能保 存的最大字符数(1 至 255), 该值 由用户设置, 第二个字节返回实际 输入的字符数(回车除外)
AH = 0AH AL = 2(接收 10 进制数) DS: DX = 段: 偏移地址 缓冲区的第一个字节存放它能保 存的最大字符数(1 至 255)	缓冲输入(从键盘读一行并放入用 户定义的缓冲区)	缓冲区的第一个字节说明它能保 存的最大字符数(1 至 255), 该值 由用户设置, 第二个字节返回实际 输入的字符数(回车除外)
AH = 0BH	取输入状态(检查是否可以从键盘 缓冲区取一个字符)	AL = 00H(没有输入), AL = 0FFH(有字符输入)
AH = 0CH	先清键盘缓冲区, 然后, 如果 AL = 01H、06H、07H、 08H、0AH, 相当于 INT 21H, AH = AL	
AH = 25H, AL = 中断号 DS: DX = 中断处理过程段: 位移	置中断向量	
AH = 35H, AL = 中断号	取中断向量(得到当前中断处理程 序地址)	ES: BX = 中断处理程序段: 位移
AH = 4CH	带返回码结束程序	
AH = 0FEH, AL = 0	读取 GDTR 寄存器	EDI: 缓冲区首地址

EDI = 存放缓冲区首地址		
AH = 0FEH, AL = 1 EBX = 全局描述符表首地址 CX = 读取长度 EDI = 存放缓冲区首地址	读取全局描述符表	EDI: 缓冲区首地址
AH = 0FEH, AL = 2	如果操作系统是 Winxp SP2 或更新, 返回: AL = 1; 操作系统比 Winxp SP2 来得早, 返回: AL = 0	操作系统是 Winxp SP2 或更新, 返 回: AL = 1; 操作系统比 Winxp SP2 来得早, 返回: AL = 0
AH = 0FEH, AL = 3, EDI = 存放缓冲区首地址	读 CR3	EDI: 缓冲区首地址(存放 CR3)
AH = 0FEH, AL = 4 CX = 读取字节数, EBX = 物理地址, EDI = 存放缓冲区首地址	读内存	CY = 1, 读取成功; CY = 0, 读取失败
AH = 0FEH, AL = 5 EDI = 存放缓冲区首地址	读 PCI9052 板卡三块 Memory 对应的 虚拟地址、物理地址	第一个双字: 虚拟地址—— PCI9052 MEMORY 基地址(用于访问 局部配置寄存器) 第二个双字: 物理地址—— PCI9052 MEMORY 基地址 第三个双字: 虚拟地址—— PCI9052 板卡上 8 位 Memory 空间 基地址 第四个双字: 物理地址—— PCI9052 板卡上 8 位 Memory 空间 基地址 第五个双字: 虚拟地址—— PCI9052 板卡上 32 位 Memory 空间 基地址 第六个双字: 物理地址—— PCI9052 板卡上 32 位 Memory 空间 基地址
AH = 0FEH, AL = 6	读当前系统的 DS	DS→EAX
AH = 0FFH, DX=多少毫秒	延时 DX 毫秒	
INT 10H 软中断		
AH = 0	清屏	
AH = 2, DH = 行, DL = 列	设置光标位置 (用字符坐标确定光标位置)	
AH = 3	读光标位置	DH = 行, DL = 列

附录 B 数码管显示、键扫描库程序说明

1、InitKeyDisplay 初始化键盘、LED 控制器 8255

2、Display8 显示子程序（8255 对数码管扫描）

输入参数：SI—指向 8 字节显示缓冲区；
如果需要显示小数点，8 位 16 进制数的最高位为 1，例如：80H；
如果某位不需要显示，符值 10H；
如果需要显示负号“-”，符值 11H

例子：10H, 10H, 03H, 82H, 00H, 00H, 00H, 00H 显示为：“ 32.0000”

3、Display8A 显示子程序（简单 I/O 的二片 74HC273 对数码管扫描）

输入参数：SI—指向 8 字节显示缓冲区；
如果需要显示小数点，8 位 16 进制数的最高位为 1，例如：80H；
如果某位不需要显示，符值 10H；
如果需要显示负号“-”，符值 11H

例子：10H, 10H, 03H, 82H, 00H, 00H, 00H, 00H 显示为：“ 32.0000”

4、GetBCDKey 接收一组压缩 BCD 码键值

输入参数：DI — 指向接收缓冲区
CX — 接收键值数量
F1 -- 0 : 先清除显示，再接收键输入
-- 1 : 接收到第一个按键后, 先清除原来显示的内容, 再显示键值
变量 F1 已在库文件中定义

例子：
EXTRN F1:BYTE ;F1 已在库文件中定义
.....
MOV F1, 0 ;先清除显示，再接收键输入
LEA DI, augend
MOV CX, 4 ;按键次数
CALL GetBCDKey ;得到双字节十进制数

5、GetKey 接收一组压缩 16 进制键值

输入参数：DI — 指向接收缓冲区
CX — 接收键值数量
F1 -- 0 : 先清除显示，再接收键输入
-- 1 : 接收到第一个按键后, 先清除原来显示的内容, 再显示键值
变量 F1 已在库文件中定义

6、GetKeyA 接收一个 16 进制键值，如果没有按键，立即返回

输出：CY -- 0, 没有按键；
CY -- 1, AL--键值

7、GetKeyB 接收一个 16 进制键值，如果没有按键，一直等待

输出： AL -- 键值

以太网、USB1.1、USB2.0、CAN、GPS、GPRS 等模块说明请参阅光盘中说明