TPC-USB

通 用 微 机 接 口 实 验 系 统

教 师 用 实 验 指 导 书

（汇 编 +VC）

刘晓坤　郑文锋　冯一兵　金丽霞　编写

清 华 大 学 计 算 机 系
清 华 大 学 科 教 仪 器 厂
2006年11月

目　　录

第一章 TPC-USB实验系统介绍

1.1概述

在各种计算机外围接口不断推陈出新的今天，USB接口已经成为个人计算机最重要的接口方式之一，USB接口设备的应用也以惊人的速度发展，几乎新型的PC都100%支持USB技术。了解和掌握USB的应用及开发是计算机类、电子类、物理类本科生、大专生的新课题。

TPC-USB微机接口实验系统正是在这种背景下推出的。该设备在原TPC-2003A微机接口实验系统上配置了USB接口模块，直接与主机(PC)的USB接口连接，形成了一套完整的USB接口的微机接口实验系统。该系统适应当前高等院校所开设的《微机原理及其应用》和《微机接口技术》这两门课的实验，同时也提供了最新接口USB的实验，使学生在校学习期间不仅有机会接触常规接口，同时有机会接触新型的接口，为学生们今后从事微机开发应用打下基础。

1.2 TPC-USB实验系统构成及特点

该系统由一块USB总线接口模块、一个扩展实验台及软件集成实验环境组成。USB总线接口模块通过USB总线电缆与PC机相连，模块与实验台之间由一条50芯扁平电缆连接。其主要特点如下：

1.USB总线接口使用ISP1581 USB2.0高速接口芯片，完全符合USB2.0规范。提供了高速USB下的通信能力，即插即用。

2.满足《微机原理与接口技术》课程教学实验要求。实验台接口集成电路包括：可编程定时器/计数器（8253）、可编程并行接口（8255）、数/模转换器(DAC0832)、模/数转换器(ADC0809)等。外围电路包括：逻辑电平开关、LED显示、七段数码管显示、8X8双色发光二极管点阵及驱动电路、直流电机步进电机及驱动电路、电机测速用光藕电路、数字测温传感器及接口电路、继电器及驱动电路、喇叭及驱动电路。8279键盘显示控制电路。

3.在USB接口模块上扩展有DMA控制器8237及存储器，可以完成微机DMA传送以及USB的DMA传送等实验。

4.开放式结构，模块化设计支持开放实验。实验台上除固定电路外还设有用户扩展实验区。有五个通用集成电路插座，每个插座引脚都有对应的“自锁紧”插孔，利用这些插孔可以搭试更多的自己设计的实验，方便的进行课程设计。

5.功能强大的软件集成开发环境，支持Win2000;WinXP 等操作系统。可以方便的对程序进行编辑、编译、链接和调试，可以查看实验原理图，实验接线，实验程序并进行实验演示。可以增加和删除实验项目。

6.实验程序可以使8086汇编和C语言编程实验。可以对汇编程序和C语言程序进行调试.

7.系统还提供：字符、图形液晶显示实验模块；红外收发实验模块；无线通信实验模块；键盘显示实验模块等多种扩展实验模块（自选）。

8.实验台自备电源，具有电源短路保护确保系统安全。

9.使用USB接口与PC机相连，省却了打开主机箱安装接口卡的麻烦。

第二章 TPC-USB实验系统硬件环境

2.1 USB模块介绍

2.1.1 USB模块结构

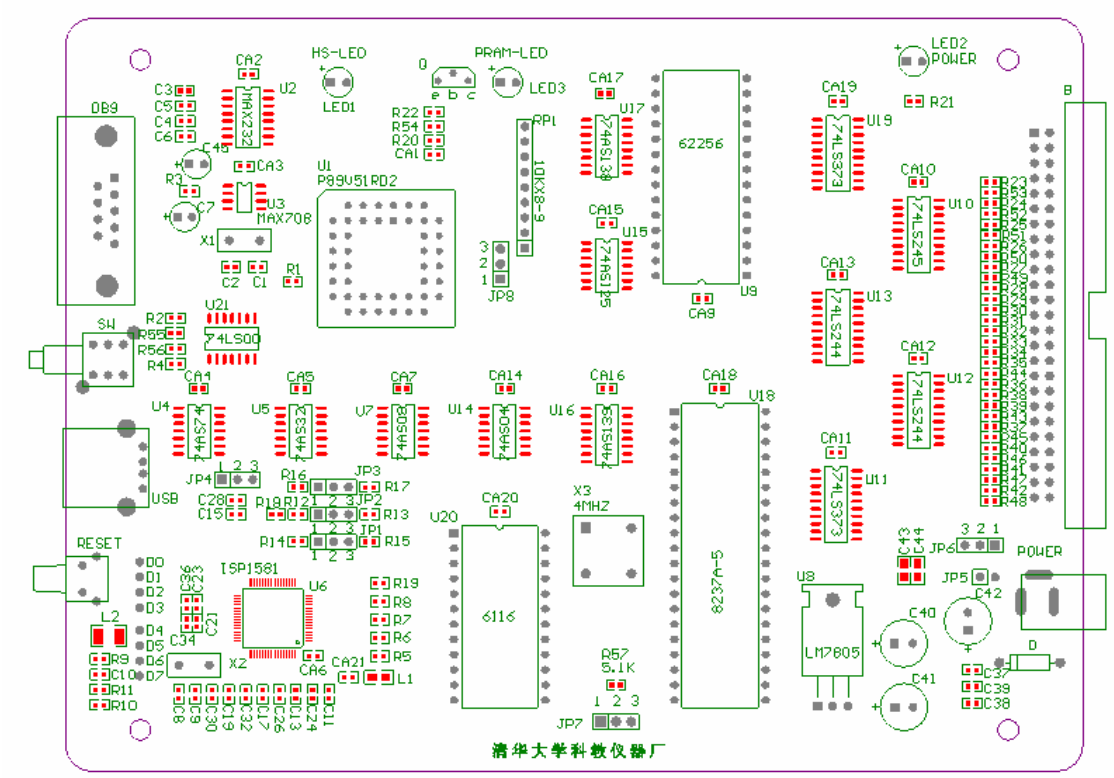


图2-1 USB模块结构图

2.1.2 USB模块功能

1.实验系统中的USB模块使用PHILIHPS的ISP1581 USB2.0高速接口芯片，符合USB2.0接口规范，提供了高速USB下的通信能力。

2.支持ISP下载，通过模块上的RS-232接口，可以对模块内部的MCU进行在线编程，对软件进行修改或在线升级。也可以通过RS-232接口下载实验程序到USB模块，进行实验。

3.模块内扩展有DMA控制器8237及存储器，可以完成微机DMA传送和USB的DMA传送实验。

4.该模块提供一个50线扁平电缆，通过该电缆将模块产生的仿ISA总线信号连到实验台上。

2.1.3 USB模块的对外接口

1.在该模块的右侧提供四个对外接口：

①9芯通用RS-232接口，需要时可连到主机的COM1或COM2，对内部的MCU在线编程，对软件升级或修改。

②USB接口，连接到主机，实验时用于信息和数据的通信。

③清零按钮（RESET），用于对模块内部电路的初始化。

④实验方式转换按钮（SW），有些实验需要将实验程序下载到USB模块运行，需要时按一下该按钮以转换实验方式。（一般情况下，用户不要按此按钮，如果需要，在实验说明中会指出）。

2.在模块的左侧提供二个对外接口：

①50线扁平电缆接口，为实验台提供仿ISA总线信号。信号安排与实验台上50芯信号插座信号一一对应。（见表十四　50芯总线插座信号）

②外接电源插孔，外接7～9V直流电源。平时USB模块与实验台相连时，使用实验台提供的电源，当USB模块单独使用或调试时，使用外接电源。

2.1.4 USB模块跳线说明

在USB模块内，用一部分跳线选择ISP1581和其它芯片的工作模式，跳线的连接说明如下：

JP1：MODE1 ISP1581 ALE/A0 功能选择。

2－3短接 低电平 选择ALE功能（地址锁存使能）

1－2短接 高电平 选择A0功能（地址数据指示）

（USB模块出厂时选择2－3短接）

JP2：M0/DA1 选择ISP1581在通用处理器模式下的读写功能。

2－3短接 低电平 选择Motorola 类型的微处理器

1－2短接 高电平 选择8051 类型的微处理器

（USB模块出厂时选择1－2短接）

JP3：BUS/DA0 选择ISP1581 总线模式

2－3短接 低电平 选择断开总线模式，AD[7:0]多路复用

1－2短接 高电平 选择通用处理器模式，AD[7:0]8位地址线

（USB模块出厂时选择2－3短接）

JP4：ISP1581 片选信号选择

2－3短接 ISP1581 片选信号由MCU 产生

1－2短接 ISP1581 片选信号由地址译码产生

（USB模块出厂时选择1－2短接）

JP5：无须用户设置

JP6：USB模块电源选择

2－3短接 选择外接电源

1－2短接 使用实验台电源

（USB模块出厂时选择1－2短接）

JP7：DMA控制器时钟选择

2－3短接 选择振荡器产生时钟

1－2短接 选择由MCU 产生时钟

（USB模块出厂时选择2－3短接）

JP8：MCU 编程方式选择

2－3短接 MCU 处于编程方式

1－2短接 MCU 处于正常工作方式

（USB模块出厂时选择1－2短接）

2.1.5 USB模块的安装

安装步骤如下：

1.关上实验台电源。

2.50线扁平电缆一端接USB模块的50芯插座，另一端接实验台50线插座。

3.USB电缆的一端接模块的USB口，另一端接主机USB口。

4.打开实验台电源。

5.系统将自行检测到模块的接入，选择用户光盘上的USB驱动程序完成驱动的安装。

安装驱动过程如下：

USB电缆接入主机，连接USB模块并加载电源后，系统将自行检测到模块的接入，提示用户发现新硬件并要求安装设备驱动：

图2-2 系统发现新硬件

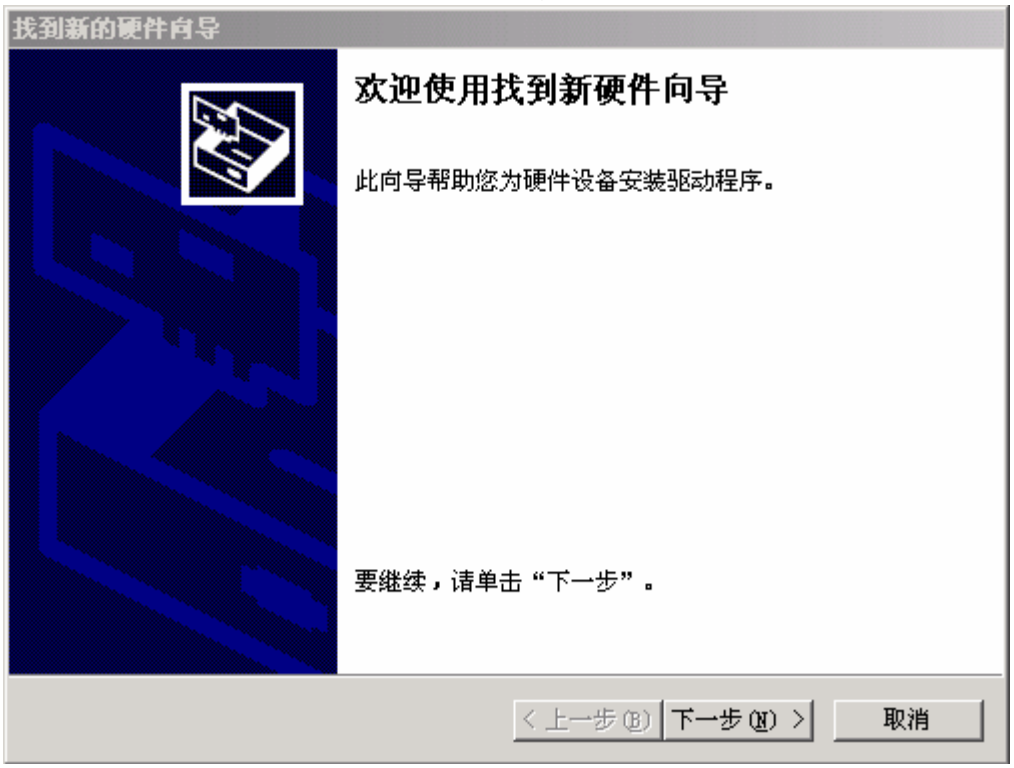


图2-3 提示找到新硬件

找到新硬件，需为此硬件指定设备驱动程序：

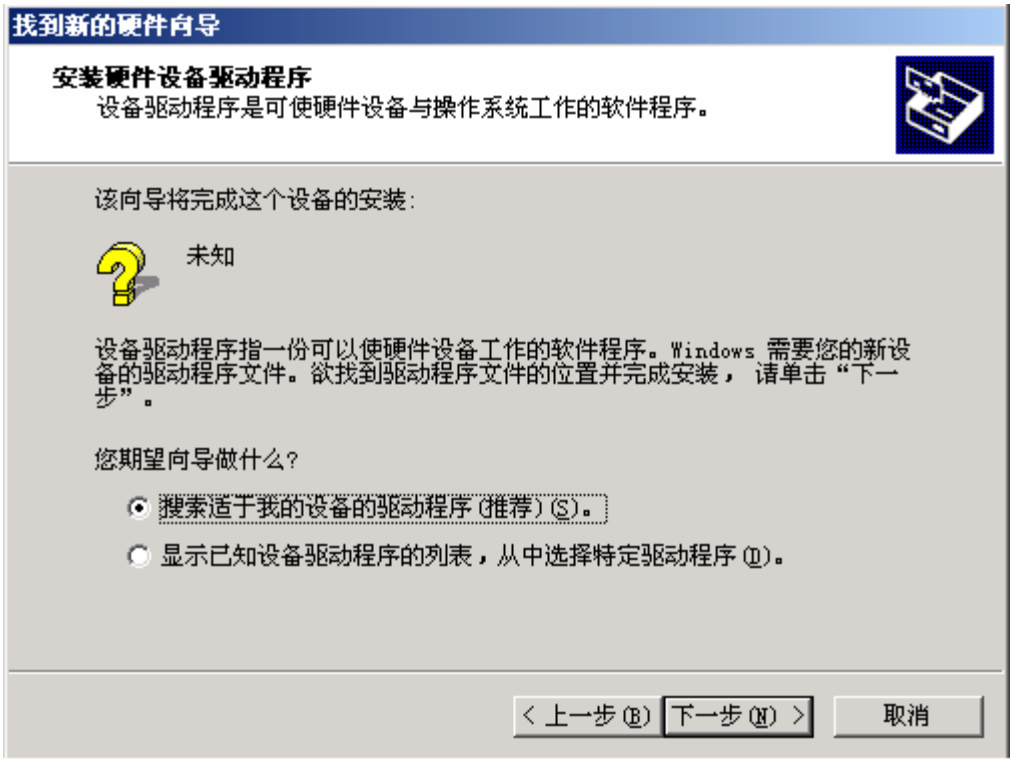


图2-4 提示按装驱动

选择驱动所在位置：(CD-ROM中“USB接口卡驱动”目录下或指定驱动所在位置)

图2-5 指定驱动所在位置

浏览驱动所在位置并选定驱动安装信息文件TPCA.inf：

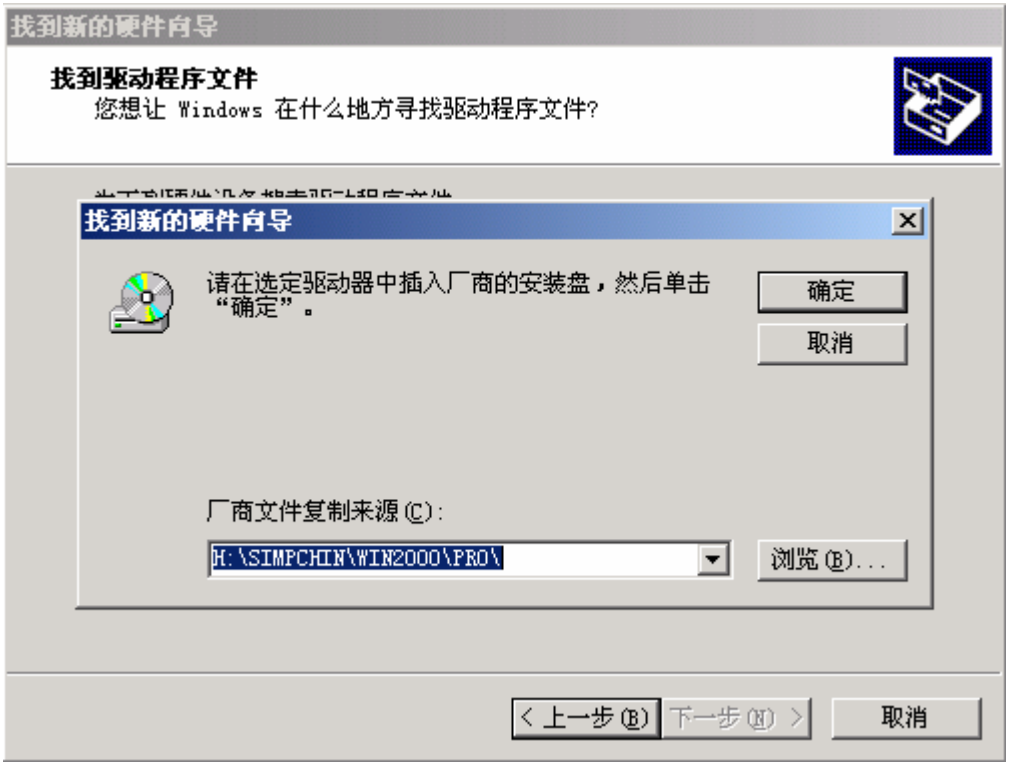


图2-6 浏览并找到驱动

选定TPCA.inf安装信息文件，并打开：

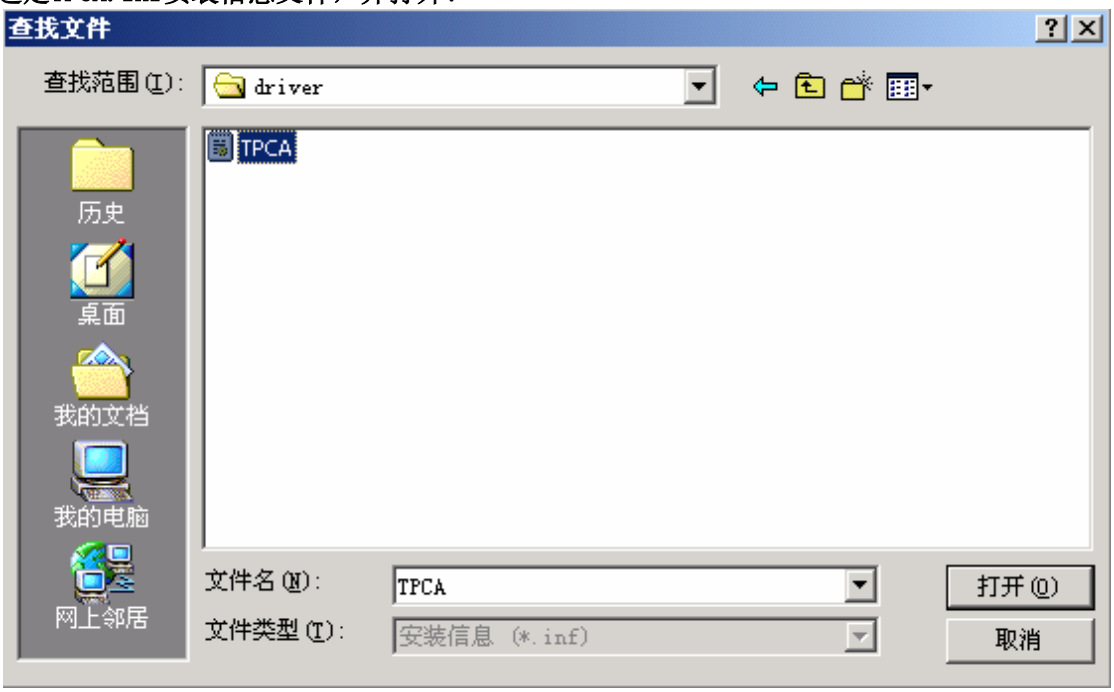


图2-7 找到驱动并选定

点击下一步，系统将自动为TPC设备安装其驱动：

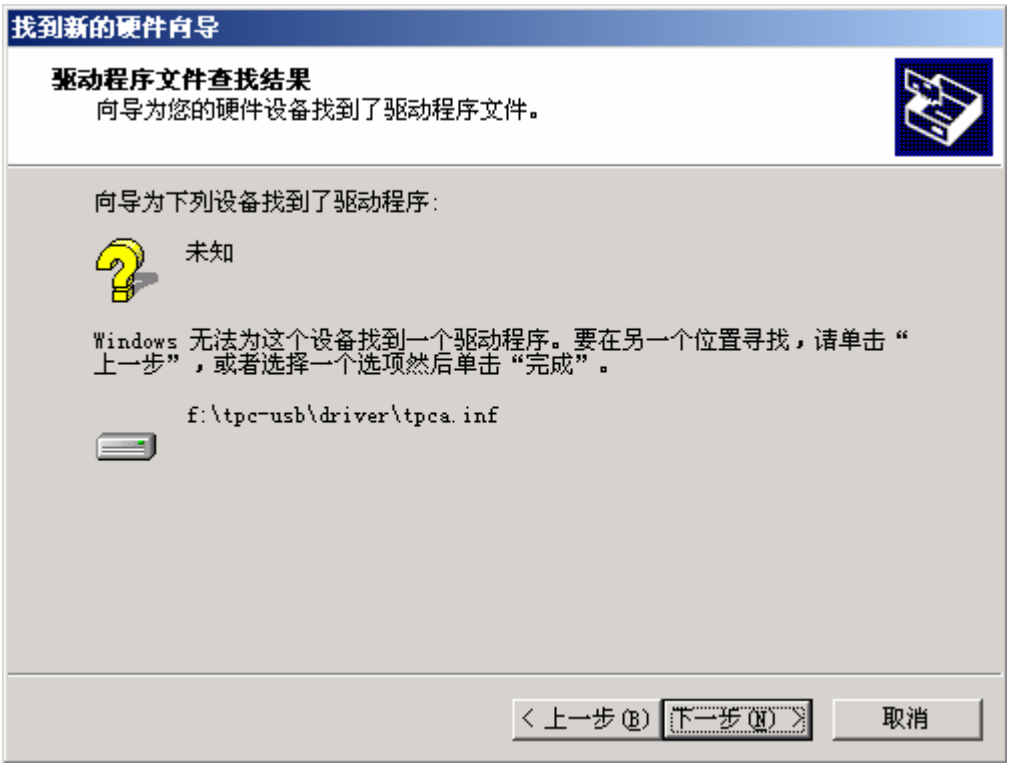


图2-8 安装驱动

驱动安装完毕：

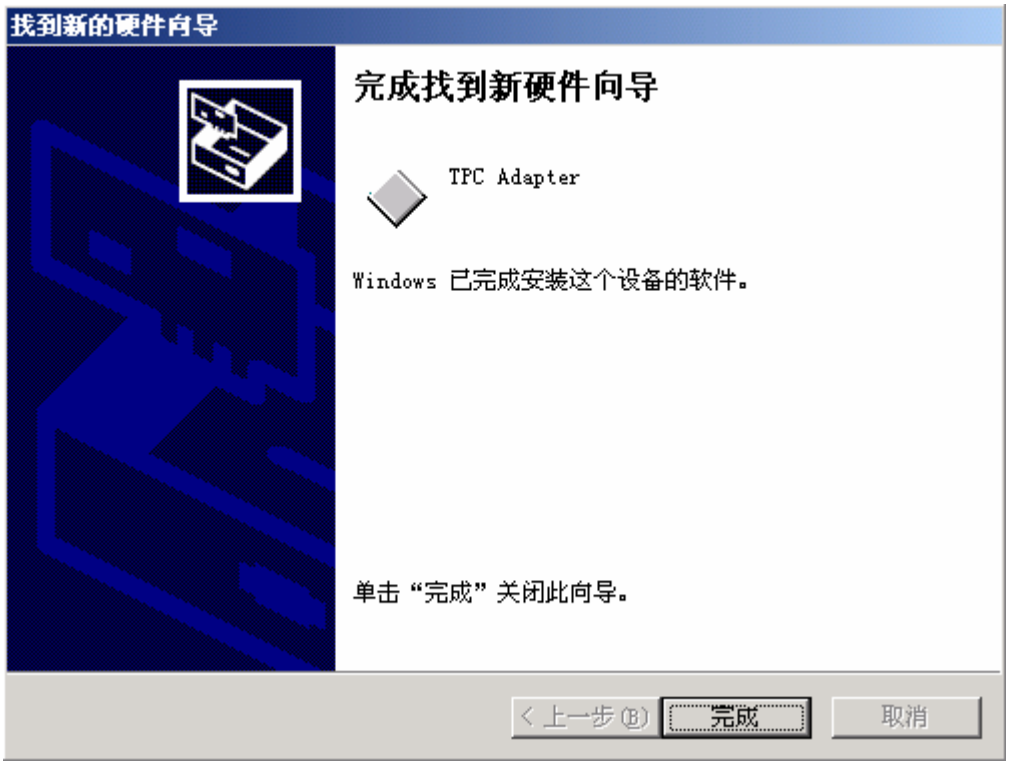


图2-9 完成安装

设备安装检测：

右键单键“我的电脑”，选择“属性”，选择硬件选项中的“设备管理器”，即可在通用串行总线控制器中找到已安装的TPC Adapter设备。**至此安装完毕。**

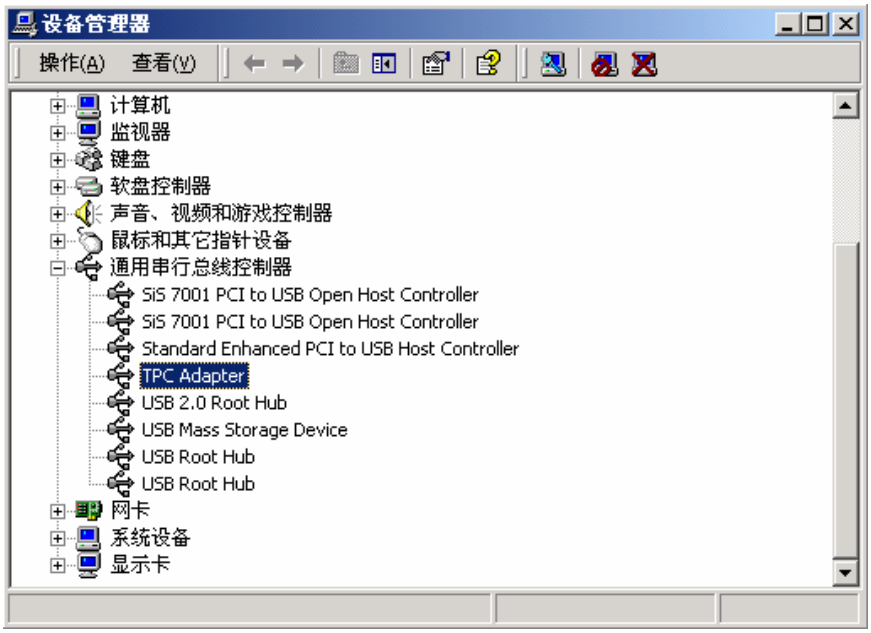


图2-10 查看安装

2.1.6 USB模块连接测试

驱动安装完成后，打开TPC-USB集成开发环境（集成开发环境的安装请参看3.2节‘集成开发环境的安装’），选定主菜单“选项”中的“硬件检测”，集成开发环境会检测到设备已连接，如果连接不正确，将会有错误提示。

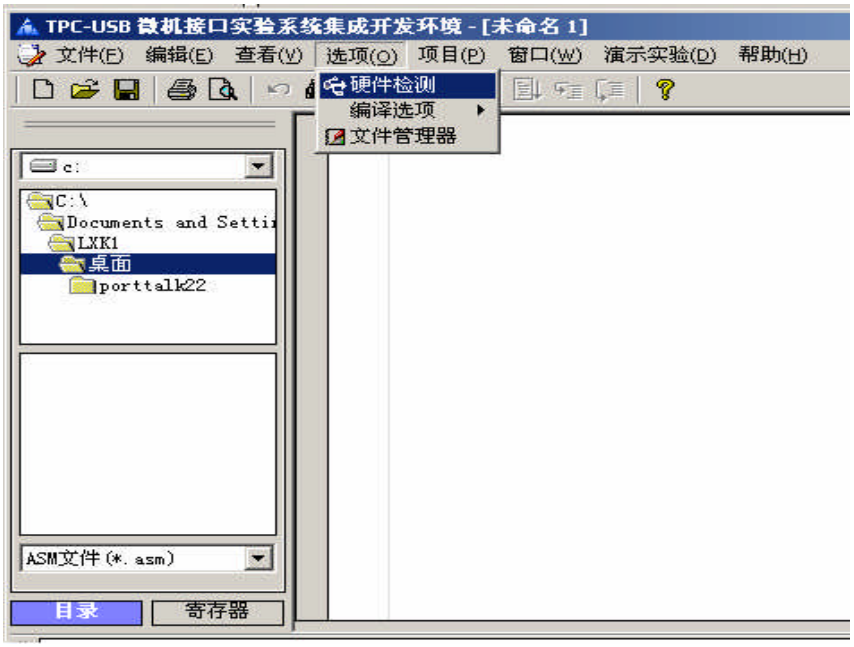


图2-11 硬件检测菜单

图2-12 正确连接提示

图2-13 硬件未连接

2.2 扩展实验台结构及主要电路

2.2.1 扩展实验台结构图

如图：图2-14 扩展实验台结构

2.2.2 实验台上包括的主要电路：

1、50芯总线信号插座及总线信号插孔

1	+5V	11	E245	21	A7	31	A1	41	ALE
2	D7	12	IOR	22	A6	32	GND	42	T/C
3	D6	13	IOW	23	A5	33	A0	43	A16
4	D5	14	AEN	24	+12V	34	GND	44	A17
5	D4	15	DACK	25	A4	35	MEMW	45	A15
6	D3	16	DRQ1	26	GND	36	MEMR	46	A14
7	D2	17	IRQ	27	A3	37	CLK	47	A13
8	D1	18	+5V	28	-12V	38	RST	48	A12
9	D0	19	A9	29	A2	39	A19	49	A10
10	+5V	20	A8	30	GND	40	A18	50	A11

50芯总线信号插座在实验台左上方，总线插座信号安排如上表。各总线信号采用“自锁紧”插孔在标有“总线”的区域引出，有数据线D0-D7、地址线A19-A0、I/O读写信号IOR IOW、存储器读写信号 MEMR MEMW、中断请求 IRQ、DMA申请DRQ、DMA回答DACK、AEN 等。

2、微机接口I/O地址译码电路

实验台上I/O地址选用280H－2BFH 64个，分8组输出：Y0-Y7，其地址分别为 280H－287H；288H－28FH；290H-297H；298H-29FH；2A0H-2A7H；2A8H-2AFH；2B0H-2B7H；2B8H-2BFH，8根输出线在实验台“I/O地址”处分别由自锁紧插孔引出。见图2-15

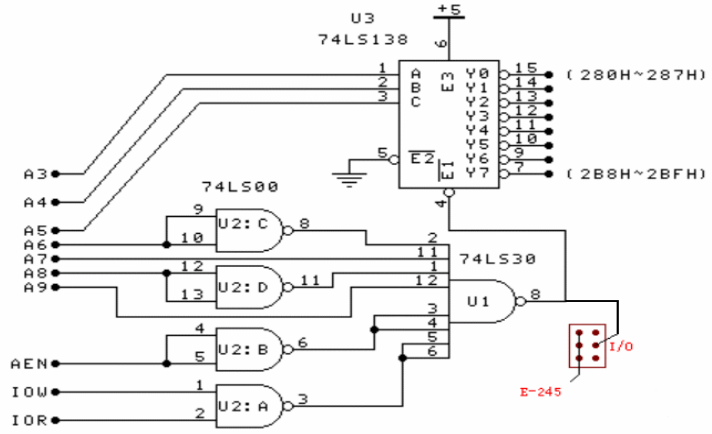


图2-15 I/O地址译码电路

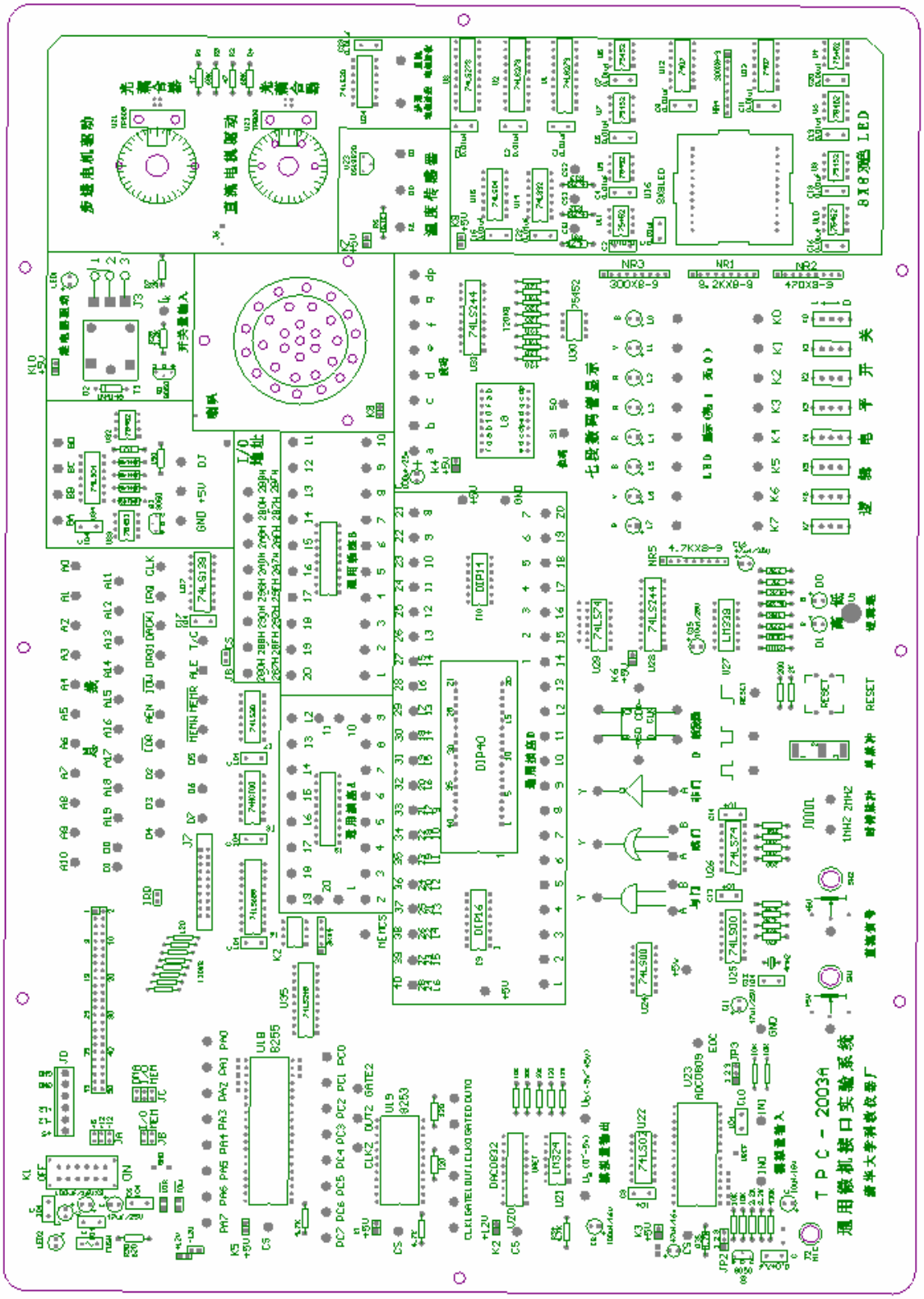


图2-14 扩展实验台结构

3、时钟电路

如图2-16所示，输出1MHZ、2MHZ两种信号，供定时器/计数器、A/D转换器、串行接口实验使用。

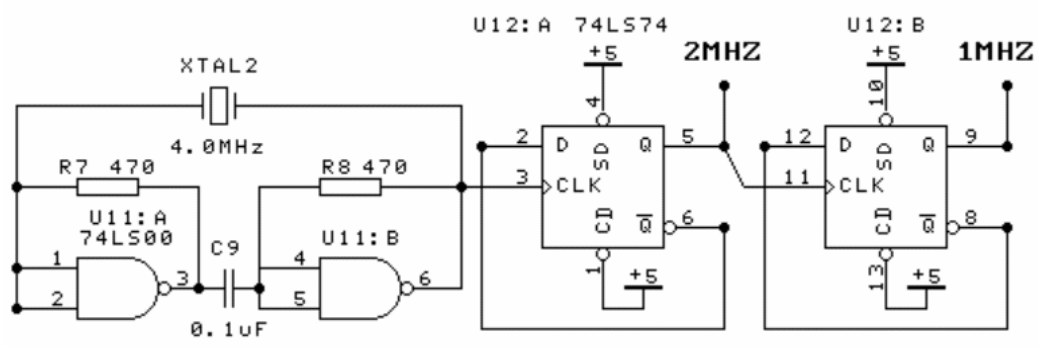


图2-16 时钟电路

4、逻辑电平开关电路

如图2-17所示，实验台右方有8个开关K0-K7,开关拨到“1”位置时开关断开，输出高电平。拨到“0”位置时开关接通输出低电平。电路中串接了保护电阻，接口电路不直接同＋5V、GND相连，有效的防止因误操作损坏集成电路现象。

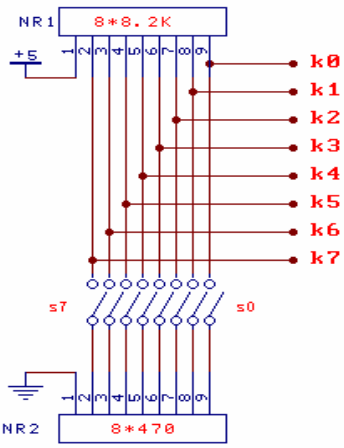


图2-17 逻辑电平开关电路

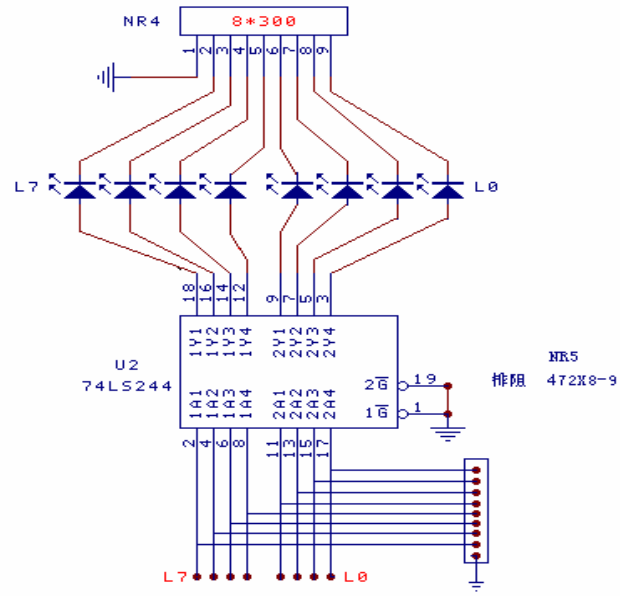


图2-18 发光二极管及驱动电路

5、LED显示电路

如图2-18所示，实验台上设有8个发光二极管及相关驱动电路(输入端L7～L0)，当输入信号为“1”时发光，为“0”时灭。

6、七段数码管显示电路

实验台设有两个共阴极数码管及驱动电路，电路图如图2-19。段码输入端：a、b、c、d、e、f、g、dp，位码输入端：S0、S1。

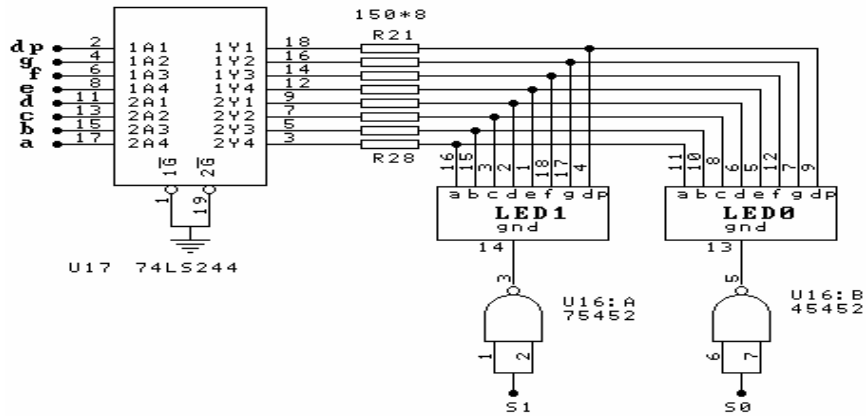


图2-19 数码管显示电路

7、单脉冲电路

如图2-20所示，采用RS触发器产生，实验者每按一次开关即可以从两个插座上分别输出一个正脉冲及负脉冲，供“中断”、“DMA”、“定时器/计数器”等实验使用。

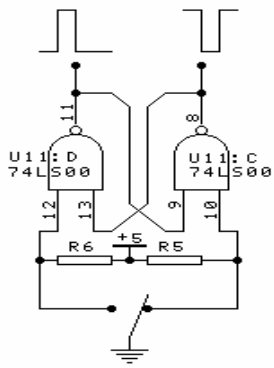


图2-20 单脉冲电路图

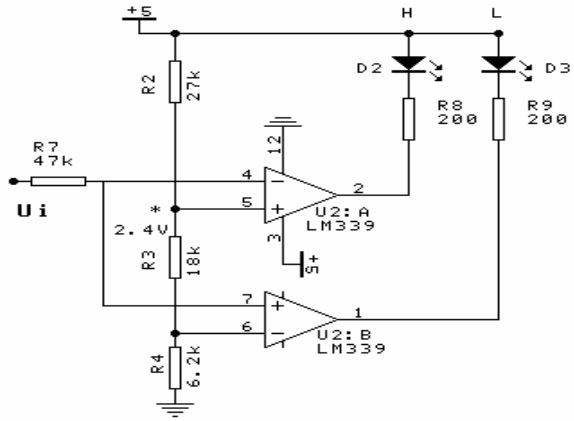


图22-21 逻辑笔电路

8、逻辑笔

如图2-21所示，当输入端Ui接高电平时红灯(H)亮，接低电平时绿灯(L)亮。

9、继电器及驱动电路

图2-22为直流继电器及相应驱动电路，当其开关量输入端“Ik”输入数字量“1”时，继电器动作，常开触点闭合红色发光二极管点亮。输入“0”时继电器常开触点断开发光二极灭。

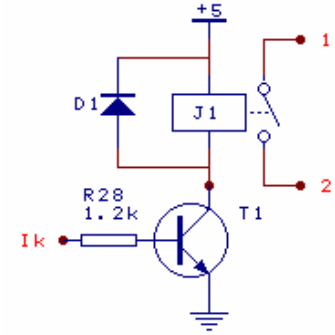


图2-22 继电器及驱动电路图

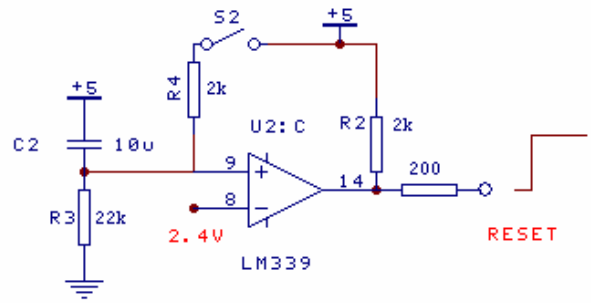


图2-23 复位电路

10、复位电路

图2-23为复位电路,实验台上有一复位电路，能在上电时，或按下复位开关RESET后，产生一个高电平的复位信号。

11、步进电机驱动电路

图2-24为步进电机的驱动电路，实验台上使用的步进电机驱动方式为二相励磁方式，BA、BB、BC、BD分别为四个线圈的驱动输入端，输入高电平时，相应线圈通电。DJ端为直流电机控制输入端。

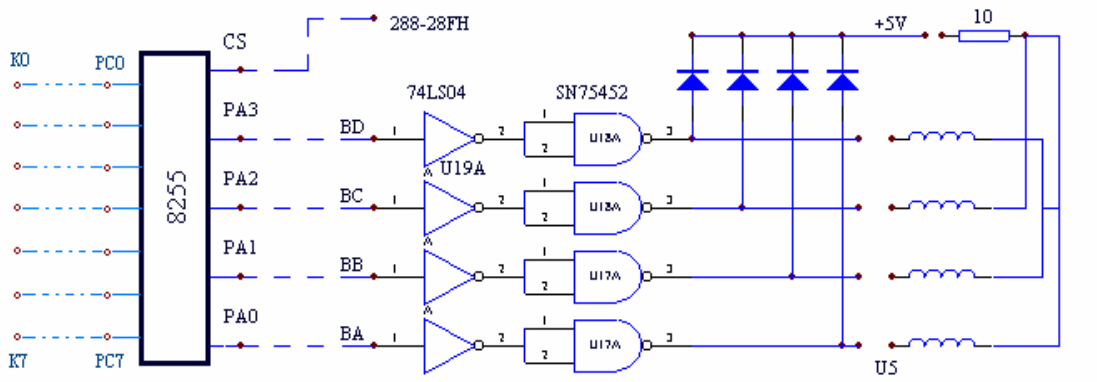


图2-24 步进电机驱动电路

12、接口集成电路

实验台上有微机原理及接口实验最常用接口电路芯片，包括：可编程定时器/计数器（8253）、可编程并行接口（8255）、数/模转换器(DAC0832)、 模/数转换器(ADC0809)，这里芯片与CPU相连的引线除去片选（CS）信号外都已连好，与外界连接的关键引脚在芯片周围用“自锁紧”插座引出，供实验使用。

13、逻辑门电路

实验台上设有几个逻辑门电路，包括“与门”、“或门”、“非门”、“触发器”供实验时选择使用。

2.2.3 用户扩展实验区

实验台上设有5个通用数字集成电路插座，其中“通用插座A”“通用插座B”为20芯，“通用插座D”为40芯活动插座以方便插拔器件。其余为14芯。插座的每个引脚都用自锁紧插孔引出。实验指导书中所列出的部分实验（简单并行接口、串行通信、集成电路测试等）电路就是利用这些插座搭试的。利用这些插座可以进行数字电路实验，也可以设计开发新的接口实验或让学生做课程设计、毕业设计等项目。

2.2.4 实验台跳线开关

为了方便实验，实验台上设有跳线开关，分以下几种：

1.实验类型选择开关JB、JC:这两个跳线开关在在实验台的左上角，50线总线插座的左下方。在TPC-USB实验系统中不起作用，用户无须设置（出T均默认接在I/O）。

2.模拟量输入选择开关JP2、JP3：在实验台ADC0809的左上角，分别用于模/数转换模拟量的输入极性选择，JP2的1、2两点短路时ADC0809的IN2可输入双极性电压（－5V～＋5V），2、3两点短路时输入单极性电压（0～＋5V）。JP3用于选择IN1的输入极性，选择方法与JP19

相同。

3.＋5V电源插针：为减轻＋5 V电源负载和各主要芯片的安全，在各主要实验电路附近都有相应的电源连接插针（标记为＋5V），当实验需要该部分电路时，用短路子短接插针即可接通＋5V电源.对用不到的电路可将短路片拔掉确保芯片安全。

2.2.5 20芯双排插座

实验台上有一个20芯双排插座J7，用于外接附加的键盘显示实验板和其它用户开发的实验板。J7各引脚信号安排如下：

19	17	15	13	11	9	7	5	3	1
D0	D1	D2	D3	D4	D5	D6	D7	IRQ	CS
RES	+5V	+5V	IOR	IOW	A0	A1	CLK	GND	GND
20	18	16	14	12	10	8	6	4	2

在J7的附近有两个短路插针标有“CS”和 “IRQ”。当“CS”的两点短接后，译码器的280H－287H连接到J7的CS端。当你扩展板上的实验需要中断信号时将“IRQ”的两端短接，不需要时应将其断开。

2.2.6 直流稳压电源

实验箱自备电源，安装在实验大板的下面，交流电源插座固定在实验箱的后测板上，交流电源开关在实验箱的右侧，交流电源开关自带指示灯，当开关打开时指示灯亮。在实验板右上角有一个直流电源开关，交流电源打开后再把直流开关拨到“ON”的位置，直流+5V +12V －12V就加到实验电路上。

主要技术指标：　输入电压　 AC 175—265V

输出电压/电流＋5V/2.5A　+12V/0.5A　-12V/0.5A

输出功率　 25W

第三章 TPC-USB集成软件开发环境

3.1 TPC-USB集成开发环境软件包

TPC-USB集成开发环境是TPC-USB实验系统所配套的软件。它提供了用户程序的编辑和编译，调试和运行，实验项目的查看和演示，实验项目的添加等功能，方便了学生和老师实验程序的编制和调试。本软件基于windows2000/XP/2003环境，界面简洁美观，功能齐全。集成开发环境主界面如图3-1：

图3-1 软件主界面

3.2 集成开发环境软件的安装

TPC-USB集成开发环境集编辑、编译、调试、实验演示等，功能齐全，使用简便，方便教师查看实验原理图及程序，利于教学。

安装步骤如下：

1．从随机所带光盘目录中找到SETUP.EXE文件，鼠标双击该文件会出现以下界面，图3.2。（为确保安全，建议先将光盘文件拷贝到硬盘某文件夹中，在硬盘上运行SETUP.EXE文件）。

图3-2

2. 输入用户名、公司名和序列号后，点击“下一步”，会出现图3-3安装选择画面。

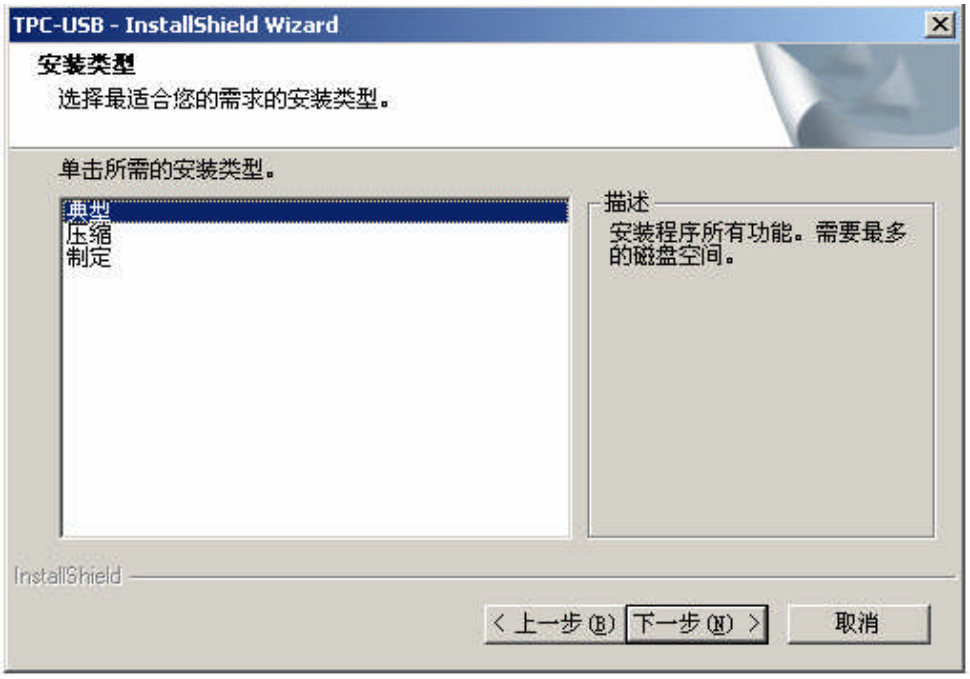


图3-3 安装选择

其中，“典型”安装是指安装主程序、实验演示程序和帮助文件。“压缩”是指只安装主程序和帮助文件，不安装实验演示程序。“制定”是在“主程序”、“实验演示”、“帮助文件”三个文件中选择你需要的安装。建议在教师实验机中选“典型”安装，在学生实验机中选“压缩”安装。

选择好以后，点击“下一步”，程序将自动将软件安装到你的机器上。

3.2.1 用户程序的编辑和编译

TPC-USB集成开发环境软件支持汇编程序(.asm文件)类型的程序开发。除了一般的编辑功能外，本软件还支持语法高亮显示，语法错误提示等功能，大大提高了程序的可读性。用户编辑好程序并保存后，即可方便地进行编译。

1.新建一个源程序

在当前运行环境下，选择菜单栏中的“文件”菜单，菜单下拉后选择“新建”，或是在工具栏中单击“新建”快捷按钮，会出现源程序编辑窗口，建议用“另存为”为文件取名保存后，就新建一个“.asm”文件。

2.打开一个源程序

当前运行环境下，选择菜单栏中的“文件”菜单，菜单下拉后选择“打开”，或是在工具栏中单击“打开”，会弹出“打开”文件选择窗口，“打开”窗口如图3-4所示：

图3-4 打开一个源程序

在窗口中“文件类型”下拉菜单中选择“ASM文档（*.asm）一项，程序即显示当前目录下所有的asm文档，单击要选择的文件，选中的文件名会显示在“文件名”中，单击“打开”则打开当前选中的文档显示在文档显示区域。点击“取消”则取消新建源文件操作。

3.编辑源程序

本软件提供了基本的编辑功能，并实现了实时的语法高亮，各项操作说明如下：

撤消

当前运行环境下，选择菜单栏中的“编辑”菜单，菜单下拉后选择“撤消”，或是在工具栏中单击“撤消”，即可撤消上一步剪切或粘贴操作。

剪切

当前运行环境下，选择菜单栏中的“编辑”菜单，菜单下拉后选择“剪切”，或是在工具栏中单击“剪切”，即可将文档显示区域中选中的内容剪切到剪贴板。

复制

当前运行环境下，选择菜单栏中的“编辑”菜单，菜单下拉后选择“复制”，或是在工具栏中单击“复制”，即可将文档显示区域中选中的内容复制到剪贴板。

粘贴

当前运行环境下，选择菜单栏中的“编辑”菜单，菜单下拉后选择“粘贴”，或是在工具栏中单击“粘贴”，即可将剪贴板中当前内容粘贴到文档显示区域光标所在处。

全选

当前运行环境下，选择菜单栏中的“编辑”菜单，菜单下拉后选择“全选”，即可将文档区域中所有内容选中。

查找

当前运行环境下，选择菜单栏中的“编辑”菜单，菜单下拉后选择“查找”，弹出查找对话框如图3-5所示：

图3-5 查找

在查找内容一栏中输入需要查找的内容，可选择“区分大小写”的查找方式，单击“查找下一个”程序则在文档显示区域中搜索与查找内容匹配的字符串，找到第一个后则高亮显示，用户点击查找下一个则继续搜索下一个匹配字符串，点击“取消”退出查找操作。

替换

当前运行环境下，选择菜单栏中的“编辑”菜单，菜单下拉后选择“替换”，弹出替换对话框如图3-6所示：

图3-6 查找下一个

在查找内容一栏中输入需要查找的内容，可选择“全字匹配”与“区分大小写”的查找方式，在替换为一栏中输入需要替换的内容，单击“查找下一个”程序则在文档显示区域中搜索与查找内容匹配的字符串，找到第一个后则高亮显示，用户可单击“替换”将匹配的字符串替换，也可单击“全部替换”将当前文档显示区域中所有与查找内容匹配的字符串全部替换。单击“查找下一个”则继续搜索下一个匹配字符串。也可单击“取消”退出查找操作。

4.保存源程序

当前运行环境下，选择菜单栏中的“文件”菜单，菜单下拉后选择“保存”，如果是无标题文档，用户需在提示下输入文档的名称及选择保存的路径，单击确定后保存；否则程序自动保存当前文档显示区域中显示的文档。或者选择菜单栏中的“文件”菜单，菜单下拉后选择“另存为”，并在提示下输入文档的名称及选择保存的路径，单击确定后保存。

3.2.2 编译源程序

1.编译调试窗口

在当前运行环境下，选择菜单栏中的“查看”菜单，单击编译调试窗口选项或是单击工具栏中“输出窗口”按钮则可对输出栏的进行显示。若当前环境显示编译调试窗口，则单击查看输出窗口选项即可隐藏该窗口，编译调试输出窗口选项即消失；若当前隐藏编译调试窗口，则单击输出窗口选项即可显示该窗口，编译调试窗口将显示。

2.ASM编译

汇编+链接

在当前运行环境下，选择菜单栏中的“项目”菜单，选择汇编+链接选项则程序对当前ASM源文件进行汇编与链接，编译调试窗口中输出汇编与链接的结果，若程序汇编或链接有错，则详细报告错误信息。双击输出错误，集成开发环境会自动将错误所在行代码高亮显示。

开始+执行

在当前运行环境下，选择菜单栏中的“项目”菜单，选择开始+执行选项则程序对当前ASM源文件执行，程序自动运行。

3.2.3 用户程序的调试和运行

1.ASM程序的调试

寄存器窗口

在当前运行环境下，选择工作区的“寄存器”菜单，寄存器窗口即可显示。寄存器窗口中显示主要的寄存器名称及其在当前程序中的对应值，若值为红色，即表示当前寄存器的值。调试时，单步执行，寄存器会随每次单步运行改变其输出值，同样以红色显示。

开始调试

在“选项”菜单中，“编译选项”选择“调试”，然后进行进行程序的编译和链接，编译和链接成功之后，调试工具将会显示，也可以在“项目”中选择“开始/结束调试”。即可开始进行程序的调试。编译选项选择如图3-7：

在ASM程序正常链接之后，选择菜单栏中的“开始/结束调试”菜单，选择开始调试选项，则对源程序进行反汇编，进入ASM的调试状态，并在寄存器窗口中显示主要的寄存器的当前值。

设置/清除断点

在ASM的调试状态下，对程序代码所在某一行前的灰色列条单击鼠标，即对此行前设置了断点，如果清除断点，只需再在此行前的灰色列条上的断点单击鼠标，此断点标记将被清除。黄色箭头所指的行为当前单步执行到的所在行。设置/清除断点如图3-8所示：

图3-7 编译选项的选择

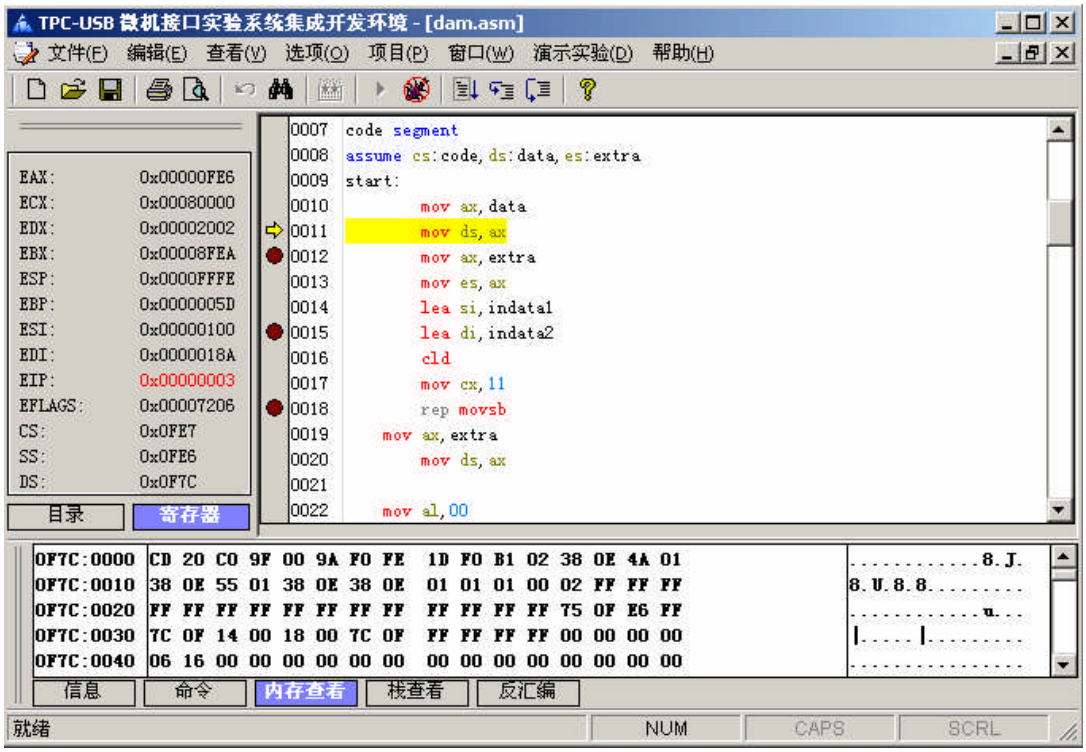


图3-8 设置/清除断点

连续运行

在ASM的调试状态下，选择“项目”菜单栏中的“连续运行”菜单或F5，则程序连续运行，直至碰到断点或程序运行结束。

单步

在ASM的调试状态下，选择“项目”菜单栏中的“单步执行”菜单或F11，则程序往后运行一条语句。

退出调试

在ASM的调试状态下，选择“项目”菜单栏中的“开始/结束调试”菜单，程序则退出ASM的调试状态。

命令调试

集成开发环境可以进行命令的调试，如图3-9：

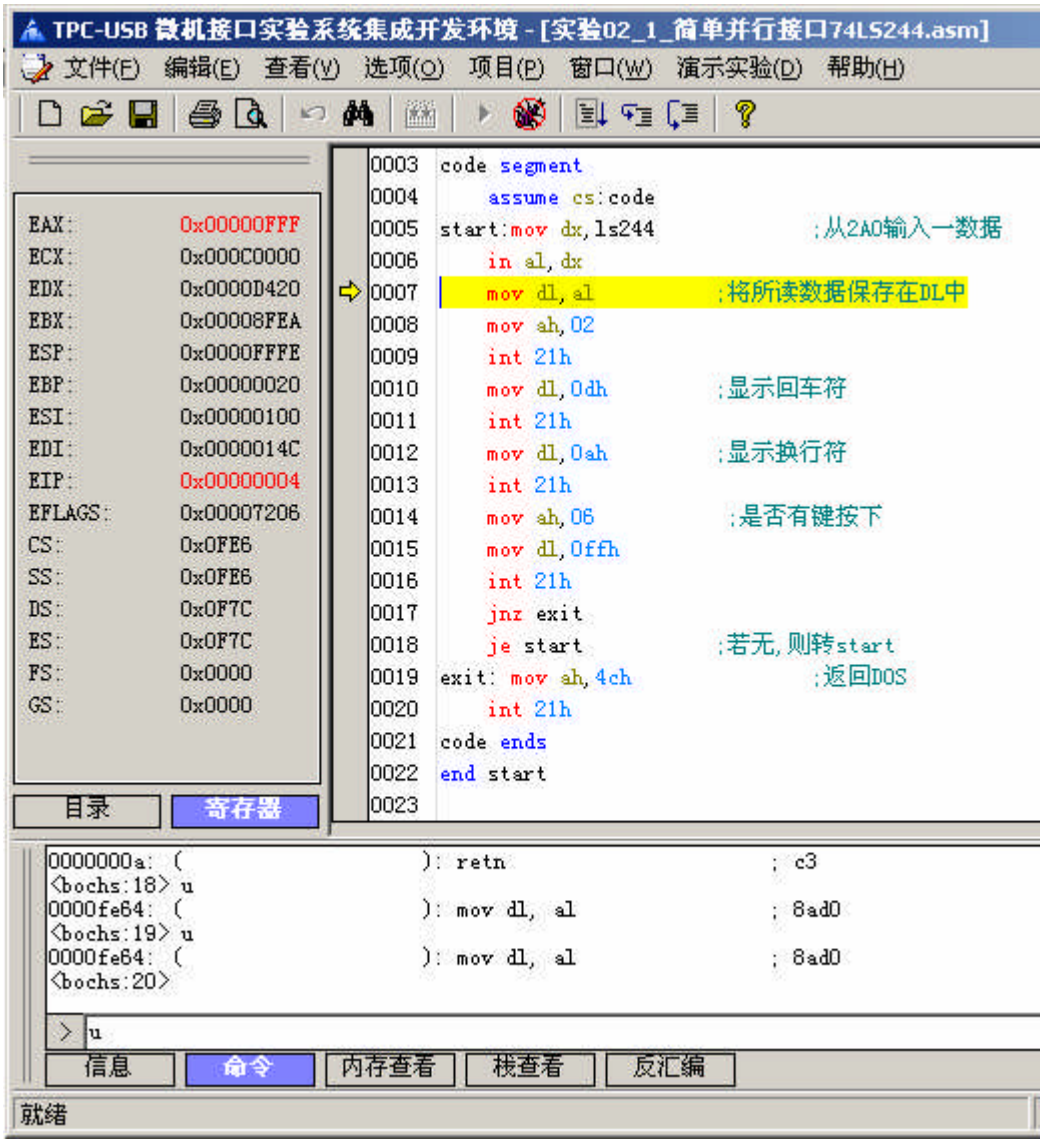


图3-9 命令调试

调试时，输出窗口可以输出编译信息，命令信息，内存查看信息，栈查看信息等。如图3-10：

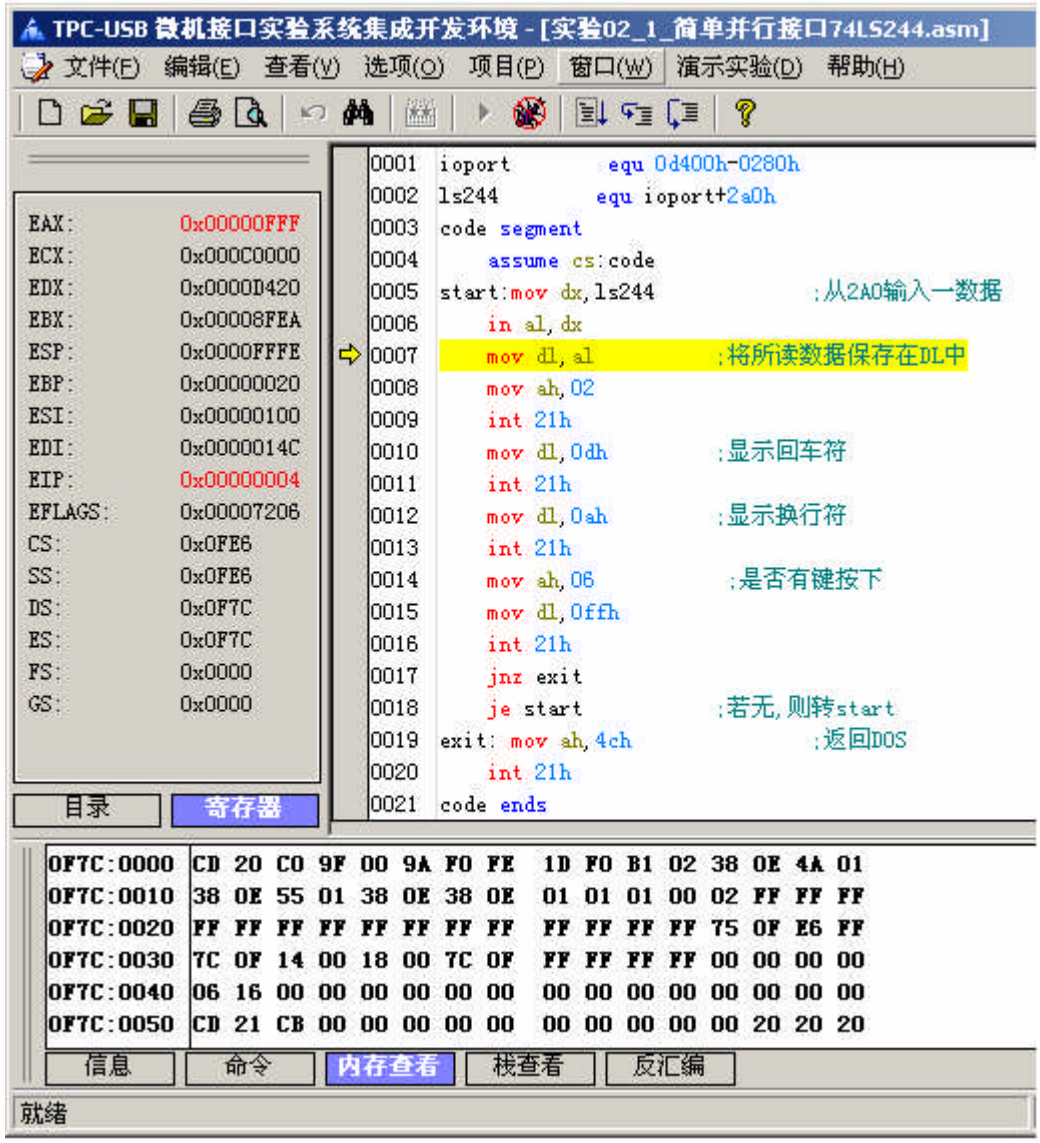


图3-10 内存查看输出窗口

3.2.4 常用调试命令

调试指令与debug稍有区别，具体调试命令如下：

bochs提供了强大的命令行调试功能，本集成开发环境在其之上包装了一个简便易用的图形界面。如果这个界面不能满足您的要求，还可以使用命令栏直接输入调试命令与bochs交互。所有调试命令bochs都提供了简要的用法说明，输入“help”（不带引号）可查看可用的命令，help ’cmd’（带引号）可查看命令cmd相关的帮助。

下面是一些常用的命令说明及示例：

1.反汇编 (u)

用法：u [/count] start end

反汇编给定的线性地址，可选参数’count’是反汇编指令的条数

例：u 反汇编当前 cs:ip 所指向的指令

u /10 从当前 cs:ip 所指向的指令起，反汇编10条指令

u /12 0xfeff 反汇编线性地址 0xfeff 处开始的12条指令

2.查看内存 (x)

用法：x /nuf addr

查看线性地址'addr'处的内存内容

nuf 由需要显示的值个数和格式标识[xduot cbhw m]组成，未指明用何种格式的情况下将使用上一次的格式。

x: 十六进制	c: 字符
d: 十进制	b: 字节
u: 无符号	h: 半字
o: 八进制	w: 字（四字节）
t: 二进制	m: 使用memory dump模式

例：x /10wx 0x234 以十六进制输出位于线性地址 0x234 处的 10 个双字

x /10bc 0x234 以字符形式输出位于线性地址 0x234 处的 10 个字节

x /h 0x234 以十六进制输出线性地址 0x234 处的 1 个字

3.查看寄存器 (info reg)

用法：info reg

查看CPU整数寄存器的内容

4.修改寄存器 (r)

用法：r reg = expression

reg 为通用寄存器

expression 为算术表达式

例：r eax = 0x12345678 对 eax 赋值 0x12345678

r ax = 0x1234 对 ax 赋值 0x1234

r al = 0x12 + 1 对 al 赋值 0x13

5.下断点 (lb)

用法：lb addr

下线性地址断点

例：lb 0xfeff 在 0xfeff 下线性地址断点，0f00:eff 所处线性地址就是 0xfeff

6.查看断点情况 (info b)

用法：info b

7.删断点 (del n)

用法：del n

删除第 n 号断点

例：del 2 删除 2 号断点，断点编号可通过前一个命令查看

8.连续运行 (c)

用法：c

在未遇到断点或是 watchpoint 时将连续运行

9.单步 (n 和 s)

用法：n

执行当前指令，并停在紧接着的下一条指令。如果当前指令是 call、ret，则相当于 Step Over。

s [count]

执行 count 条指令

10.退出 (q)

用法：q

2.C语言程序的调试

大多数实验所用的程序需要用到配套的Visual Studio生成的静态链接库(.lib)或动态链接库(.dll)文件，因此本软件采用了Visual C++的调试系统。由于版权问题，本软件没有提供Visual C++的编译和调试器，需要用户自己安装。

3.2.5 实验项目的查看和演示

本软件提供了实验项目的查看和演示功能，包括实验说明、实验原理图、实验流程图、ASM程序，并可以运行实验程序，使用户能方便快捷地了解感兴趣的实验。示例如图3-11：

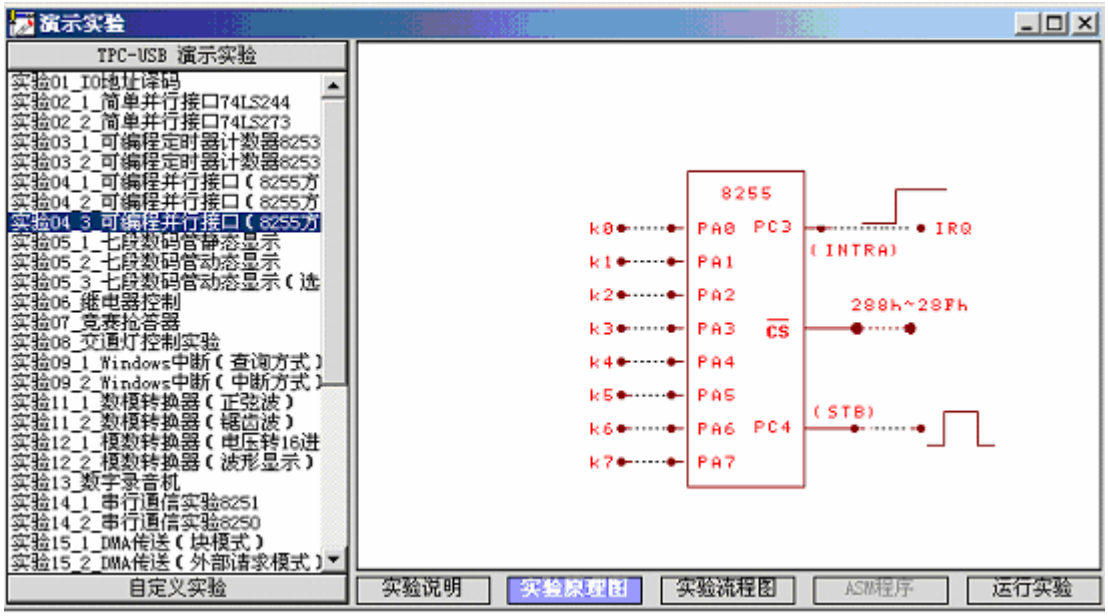


图3-11实验项目的查看和演示

各实验有几个子项，包括实验说明、实验原理图、实验流程图、ASM程序和运行实验。单击对应子项，即可查看对应的项目。

实验说明

双击实验说明子项，即可弹出对应实验的实验说明。

实验原理图

双击实验原理图子项，即可弹出对应实验的实验原理图。

实验流程图

双击实验流程图子项，即可弹出对应实验的实验流程图。

ASM程序

双击ASM程序子项，即可弹出对应实验的ASM程序源文件。

运行实验

双击运行实验子项，即可执行对应实验的可执行程序。

3.2.6 实验项目的添加和删除

除预定义的26个常用实验外，本软件还支持自定义实验，方便用户扩展实验内容。用户可以自行添加实验，被添加的实验将作为“自定义实验”的子类，之后便能在演示实验中进行查看，查看方式和预定义实验相同。

1.添加实验

在当前运行环境下，选择菜单栏“演示实验”菜单选项，在下层目录中选择自定义实验选项，则出现TPC-USB自定义实验对话框，如图3-12所示：

图3-12 自定义实验

用户可以对自定义实验进行添加和删除操作。点击添加实验按钮，则弹出添加实验对话框，如图3-13：

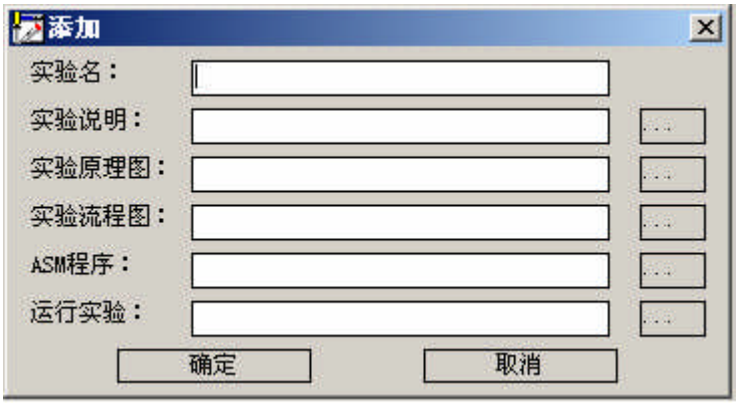


图3-13 添加实验

用户可以直接输入目标文件地址或是通过右侧的浏览按钮来选择文件，需要注意的是，添加实验时实验名称和可执行程序是必不可少的。

2. 删除实验

自定义实验是可以删除的。在当前运行环境下，选择菜单栏中的“演示实验”菜单，选择自定义实验选项，在自定义实验对话框中选定待删除的实验，点击删除实验按钮，则弹出确认对话框，确认后选定的实验将被删除，否则取消删除操作。如图3-14所示：

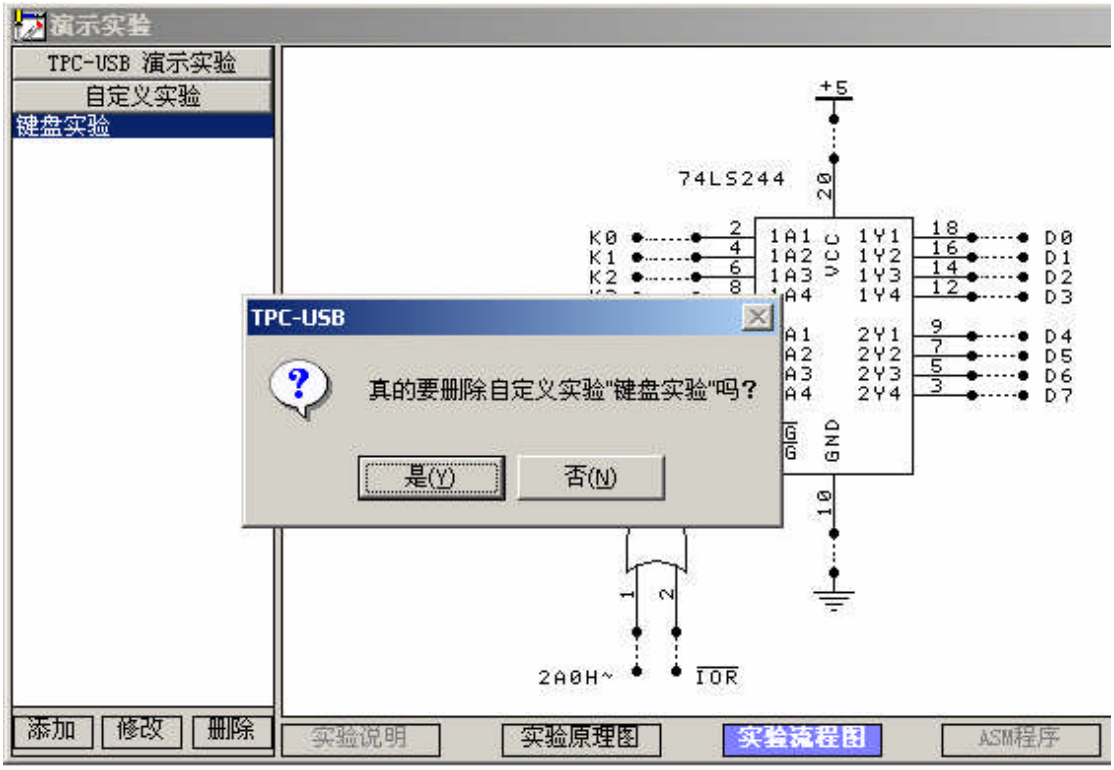


图3-14 实验删除

3.2.7 集成开发环境帮助菜单

集成开发环境帮助菜单方便教师和学生对软件使用，芯片查寻，常用命令查询等。分别如图3-14，图3-15，图1-16：

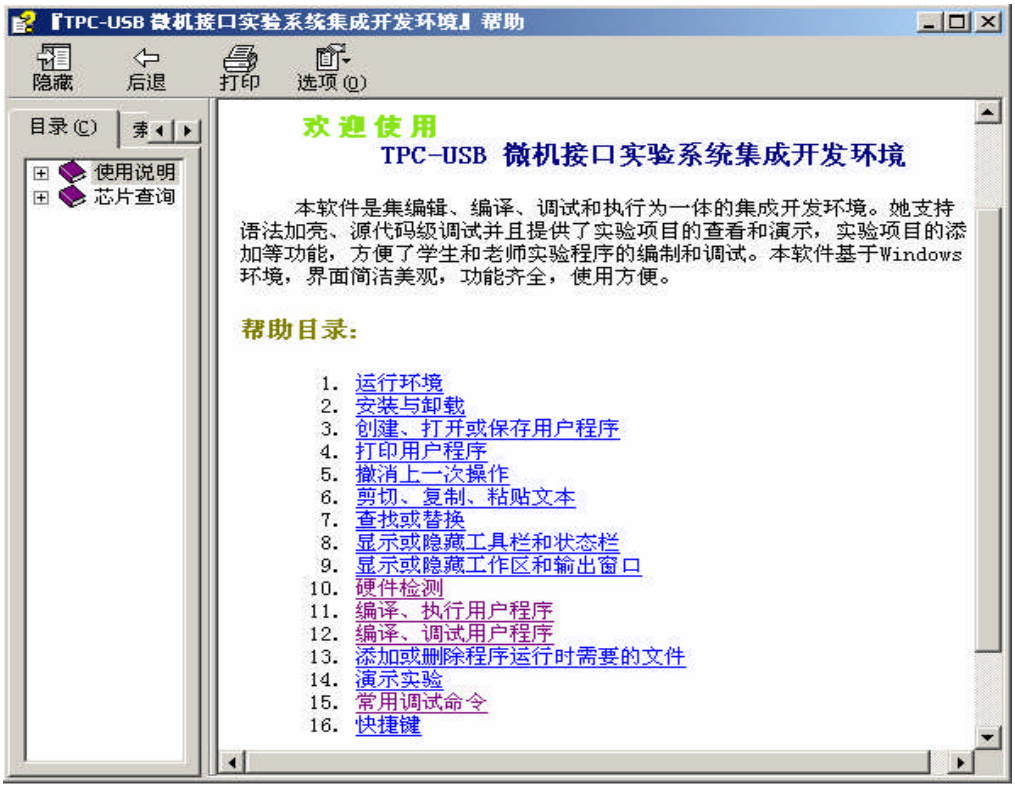


图3-14 集成开发环境帮助

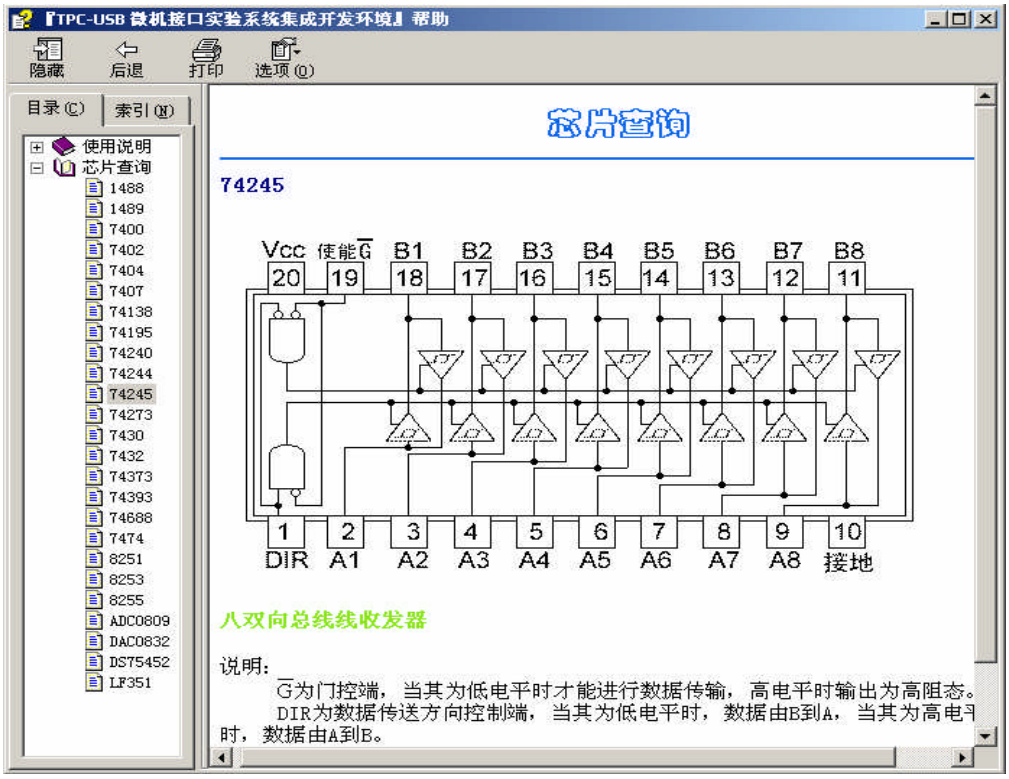


图3-15 常用芯片查询

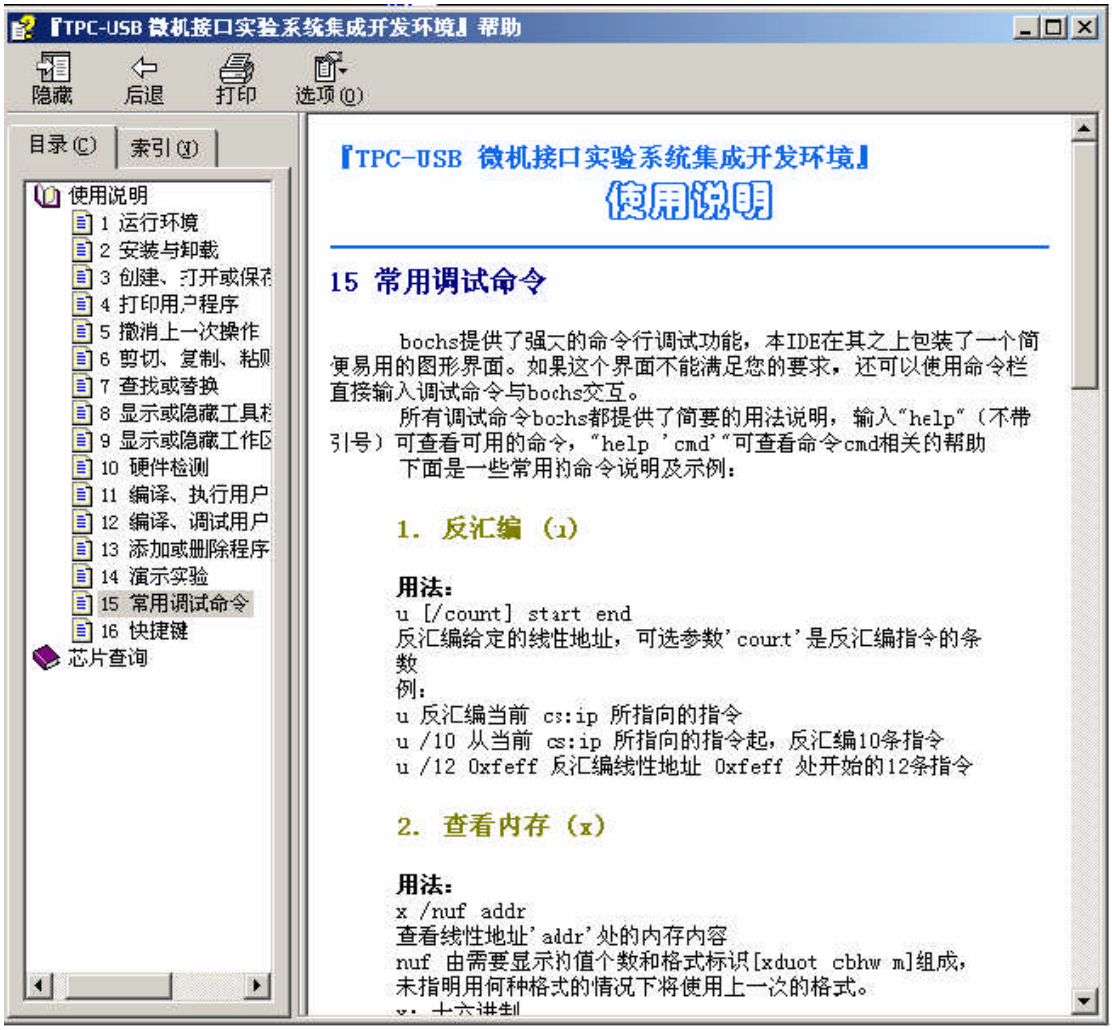

『TPC-USB 微机接口实验系统集成开发环境』帮助
隐藏
后退
打印
选项(O)
目录(C)
索引(I)
使用说明
1 运行环境
2 安装与卸载
3 创建、打开或保存
4 打印用户程序
5 撤消上一次操作
6 剪切、复制、粘贴
7 查找或替换
8 显示或隐藏工具栏
9 显示或隐藏工作区
10 硬件检测
11 编译、执行用户
12 编译、调试用户
13 添加或删除程序
14 演示实验
15 常用调试命令
16 快捷键
芯片查询
『TPC-USB 微机接口实验系统集成开发环境』
使用说明
15 常用调试命令
bochs提供了强大的命令行调试功能，本IDE在其之上包装了一个简便易用的图形界面。如果这个界面不能满足您的要求，还可以使用命令栏直接输入调试命令与bochs交互。
所有调试命令bochs都提供了简要的用法说明，输入"help"（不带引号）可查看可用的命令，"help 'cmd'"可查看命令cmd相关的帮助
下面是一些常用的命令说明及示例：
1. 反汇编 (u)
用法:
u [/count] start end
反汇编给定的线性地址，可选参数'count'是反汇编指令的条数
例:
u 反汇编当前 cs:ip 所指向的指令
u /10 从当前 cs:ip 所指向的指令起，反汇编10条指令
u /12 0xfeff 反汇编线性地址 0xfeff 处开始的12条指令
2. 查看内存 (x)
用法:
x /nuf addr
查看线性地址'addr'处的内存内容
nuf 由需要显示的值个数和格式标识[xduot cbhw m]组成，
未指明用何种格式的情况下将使用上一次的格式。

图3-16 常用调试命令

第四章 汇编实验部分

几点约定:

1、实验电路介绍中凡不加“利用通用插座”说明的均为实验台上已固定电路。

2、实验电路连线在图中均用虚线表示，实线为已连好电路。

3、所有实验程序中均加有延时子程序，请老师根据实验PC机情况，适当加、减延时程序的延时时间即可。

实验一　I/O地址译码

一、实验目的

掌握I/O地址译码电路的工作原理。

二、实验原理和内容

实验电路如图1-1所示，其中74LS74为D触发器，可直接使用实验台上数字电路实验区的D触发器,74LS138为地址译码器。译码输出端Y0～Y7在实验台上“I/O地址“输出端引出，每个输出端包含8个地址，Y0：280H～287H，Y1：288H～28FH，…… 当CPU执行I/O指令且地址在280H～2BFH范围内，译码器选中，必有一根译码线输出负脉冲。

例如：执行下面两条指令

MOV　DX，2A0H

OUT　DX，AL（或IN　AL，DX）

Y4输出一个负脉冲，执行下面两条指令

MOV　DX，2A8H

OUT　DX，AL（或IN　AL，DX）

Y5输出一个负脉冲。

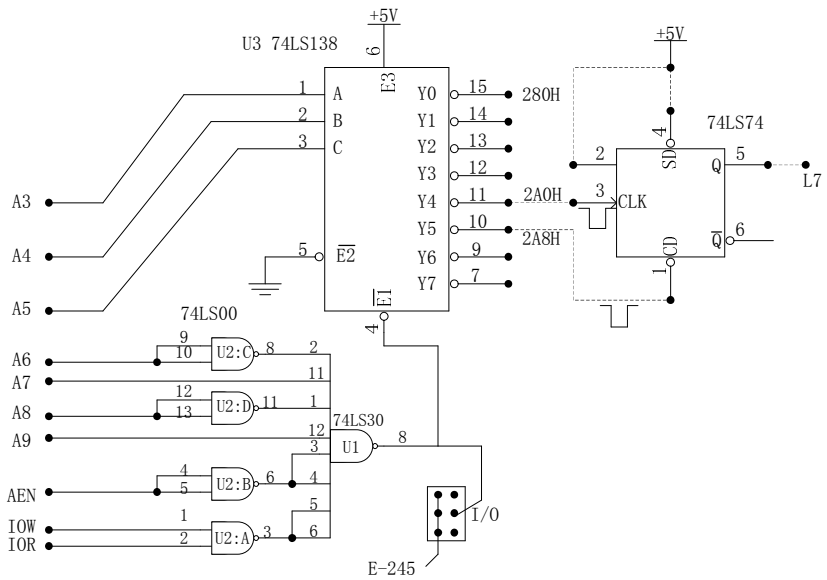


图1-1

利用这个负脉冲控制L7闪烁发光（亮、灭、亮、灭、……），时间间隔通过软件延时实现。

三、编程提示

1、实验电路中D触发器CLK端输入脉冲时，上升沿使Q端输出高电平L7发光，CD端加低电平L7灭。

2、参考程序: YMQ.ASM

```
outport1        equ 2a0h
outport2        equ 2a8h
code segment
    assume cs:code
start:
        mov dx,outport1
        out dx,al
        call delay          ;调延时子程序
        mov dx,outport2
        out dx,al
        call delay          ;调延时子程序
        mov ah,1
        int 16h
        je start
        mov ah,4ch
        int 21h
delay proc near             ;延时子程序
        mov bx,200
lll:    mov cx,0
ll:     loop ll
        dec bx
        jne lll
        ret
delay endp
code ends
end start
```

实验二　简单并行接口

一、实验目的

掌握简单并行接口的工作原理及使用方法。

二、实验内容

1、按下面图2-1简单并行输出接口电路图连接线路(74LS273插通用插座，74LS32用实验台上的“或门”)。74LS273为八D触发器，8个D输入端分别接数据总线D0～D7，8个Q输出端接LED显示电路L0～L7。

2、编程从键盘输入一个字符或数字，将其ASCⅡ码通过这个输出接口输出，根据8个发光二极管发光情况验证正确性。

3、按下面图2-2简单并行输入接口电路图连接电路(74LS244插通用插座，74LS32用实验台上的“或门”)。74LS244为八缓冲器，8个数据输入端分别接逻辑电平开关输出K0～K7，8个数据输出端分别接数据总线D0～D7。

4、用逻辑电平开关预置某个字母的ASCⅡ码，编程输入这个ASCⅡ码，并将其对应字母在屏幕上显示出来。

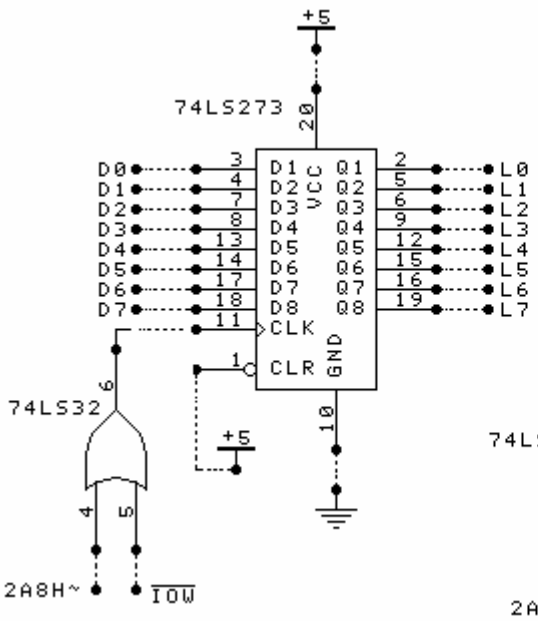


图2-1

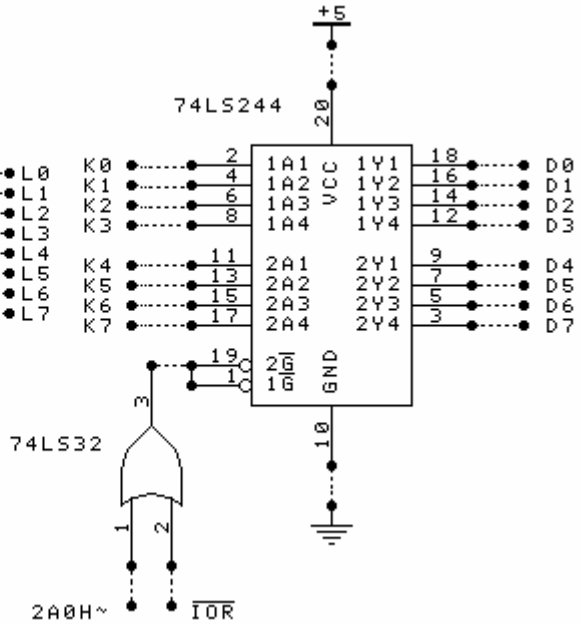


图2-2

三、编程提示

1、上述并行输出接口的地址为2A8H，并行输入接口的地址为2A0H，通过上述并行接口电路输出数据需要3条指令：

```
MOV    AL,数据
MOV    DX,2A8H
OUT    DX,AL
```

通过上述并行接口输入数据需要2条指令：

```
MOV    DX,2ADH
```

IN　　AL,DX

2、参考流程图

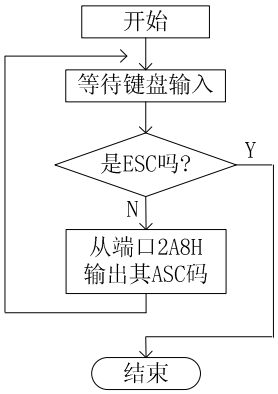


图2-3　参考程序1

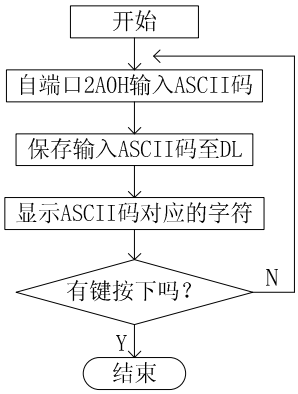


图2-4　参考程序2

3、参考程序1: E273.ASM

```
ls273    equ 2a8h
code segment
assume cs:code
start:   mov ah,2          ;回车符
         mov dl,0dh
         int 21h
         mov ah,1          ;等待键盘输入
         int 21h
         cmp al,27         ;判断是否为ESC键
         je exit           ;若是则退出
         mov dx,ls273      ;若不是,从2A8H输出其ASCII码
         out dx,al
         jmp start         ;转start
exit:    mov ah,4ch        ;返回
         int 21h
code ends
end start
```

4、参考程序2: E244.ASM

```
ls244    equ 2a0h
code segment
assume cs:code
start:
         mov dx,ls244      ;从2A0输入一数据
         in al,dx
         mov dl,al         ;将所读数据保存在DL中
```

```
        mov ah,02
        int 21h
        mov dl,0dh        ;显示回车符
        int 21h
        mov dl,0ah        ;显示换行符
        int 21h
        mov ah,06         ;是否有键按下
        mov dl,0ffh
        int 21h
        jnz exit
        je start          ;若无,则转start
exit:   mov ah,4ch        ;返回
        int 21h
code ends
end start
```

实验三　可编程定时器/计数器（8253）

一、实验目的

掌握8253的基本工作原理和编程方法，用示波器观察不同方式下的波形。

二、实验内容

1、按图3-1虚线连接电路，将计数器0设置为方式0，计数器初值为N(N≤0FH)，用手动逐个输入单脉冲，编程使计数值在屏幕上显示，并同时用逻辑笔观察OUT0电平变化(当输入N+1个脉冲后OUT0变高电平)。

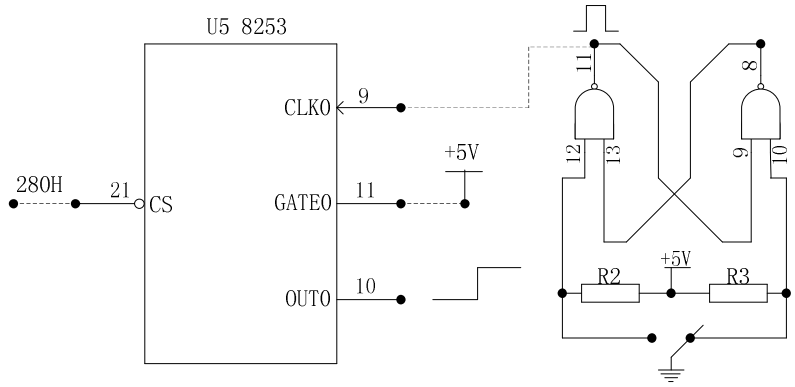


图3-1

2、按图3-2连接电路，将计数器0、计数器1分别设置为方式3，计数初值设为1000，用逻辑笔观察OUT1输出电平的变化(频率1HZ)。

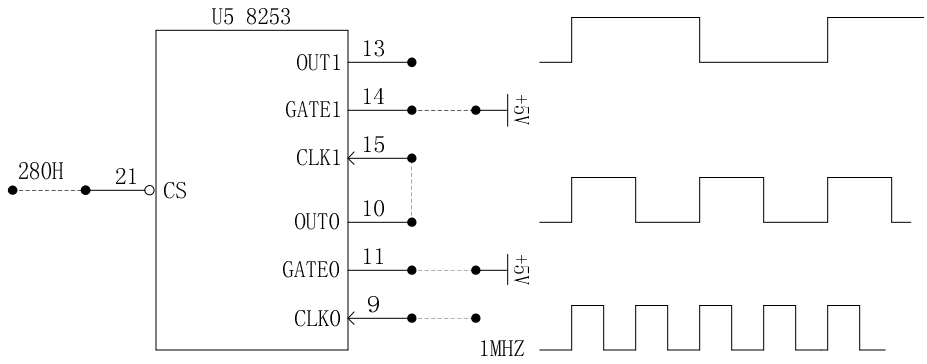


图3-2

3、按图3-3连接电路，将计数器0设置为方式3（方波），计数器设置为方式2（分频）。实现计数器0的输出为方波，计数器1的输出是计数器0输出的分频波形。人机交互界面设计：实现在显示屏幕上提示输入计数器0（方波）的参数和计数器1（分频信号）的参数。如下所示：

counter1:____

counter2:____

continue?(y/n)____

实现用键盘直接输入修改程序中方波的参数和分频信号的参数，以改变方波的宽度，分频信号的周期和分频数，不需重新修改源代码。用示波器观察计数器0和计数器1的输出波形及其关系，并在纸上画出CLK0、OUT0、OUT1的波形。

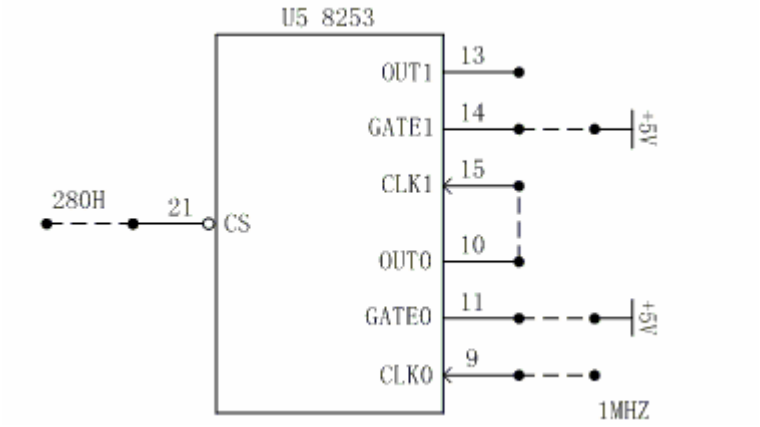


图3-3

三、编程提示

1、8253控制寄存器地址　　283H

计数器0地址　　280H

计数器1地址　　281H

CLK0连接时钟　　1MHZ

2、参考流程图(见图3-4、图3-5、图3-6):

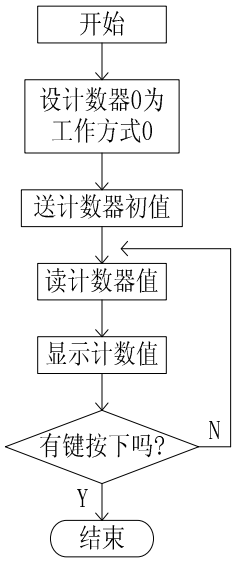


图3-4

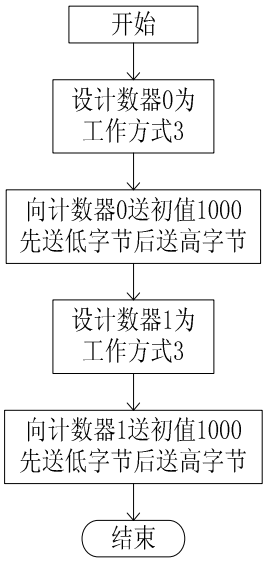


图3-5

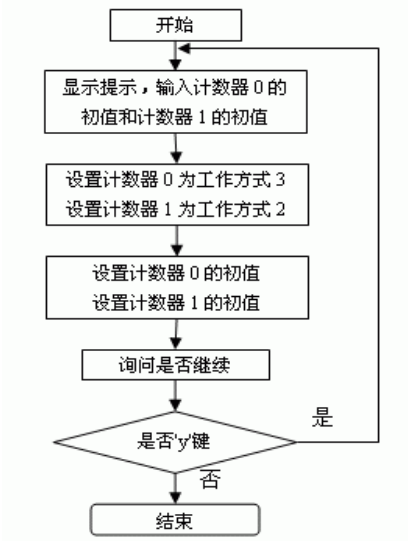


图3-6

3、参考程序1: E8253_1.ASM

```
io8253a     equ 283h
io8253b     equ 280h
code segment
assume  cs:code
start:
        mov al,14h          ;设置8253通道0为工作方式2,二进制计数
        mov dx,io8253a
        out dx,al
        mov dx,io8253b      ;送计数初值为0FH
        mov al,0fh
        out dx,al
lll:    in al,dx            ;读计数初值
        call disp           ;调显示子程序
        push dx
        mov ah,06h
        mov dl,0ffh
        int 21h
        pop dx
        jz lll
        mov ah,4ch          ;退出
        int 21h
disp proc near              ;显示子程序
```

```
        push dx
        and al,0fh          ;首先取低四位
        mov dl,al
        cmp dl,9            ;判断是否<=9
        jle  num            ;若是则为'0'-'9',ASCII码加30H
        add dl,7            ;否则为'A'-'F',ASCII码加37H
num:    add dl,30h
        mov ah,02h          ;显示
        int 21h
        mov dl,0dh          ;加回车符
        int 21h
        mov dl,0ah          ;加换行符
        int 21h
        pop dx
        ret                 ;子程序返回
disp endp
code ends
end start
```

4、参考程序2: E8253_2.ASM

```
io8253a     equ 280h
io8253b     equ 281h
io8253c     equ 283h
code segment
assume   cs:code
start:
        mov dx,io8253c      ;向8253写控制字
        mov al,36h          ;使0通道为工作方式3
        out dx,al
        mov ax,1000         ;写入循环计数初值1000
        mov dx,io8253a
        out dx,al           ;先写入低字节
        mov al,ah
        out dx,al           ;后写入高字节
        mov dx,io8253c
        mov al,76h          ;设8253通道1工作方式2
        out dx,al
        mov ax,1000         ;写入循环计数初值1000
        mov dx,io8253b
        out dx,al           ;先写低字节
```

```
            mov al,ah
            out dx,al            ;后写高字节
            mov ah,4ch           ;程序退出
            int 21h
    code ends
    end start
```

5、参考程序3: E8253_3.ASM

```
;*************************;
;*    8253 program       *;
;*************************;
data segment
     mesg0  db 13,10,'*************** 8253 program ***************',13,10,'$'
     mesg1  db 13,10,'Counter1:','$'
     mesg2  db 13,10,'Counter2:','$'
     mesg3  db 13,10,'Continue?(Y/N)','$'
     mesg4  db 13,10,13,10,'Thank You !   ',13,10,'$'
     errorm db 13,10,'Input Error ! ','$'
     counter1    dw 0
     counter2    dw 0
data ends
code segment
     assume cs:code,ds:data
main proc far
start:
        mov dx,seg data
        mov ds,dx
        mov dx,offset mesg0
        mov ah,09h
        int 21h
do:     sub bx,bx
        sub ax,ax
        mov counter1,0
        mov counter2,0            ;init
l1:
        mov dx,offset mesg1
        mov ah,09h
        int 21h
rd1 :                             ;read counter1
       mov al,0                   ;判断有无输入
```

```
        mov ah,01               ;read a char
        int 21h
        cmp al,0
        jz rd1

        cmp al,13               ;if enter
        je fdone1
        jmp tdone1
fdone1: jmp done1
tdone1: cmp al,10
        je fdone1
        cmp al,'0'               ;if input<0 or input>9 error
        jb error
        cmp al,'9'
        ja error
        push ax
        mov ax,10
        mul counter1
        mov counter1,ax          ;counter1=counter*10

        pop ax
        sub bx,bx
        mov bl,al
        sub bl,30h
        add counter1,bx           ;counter=counter+input
        jmp rd1
error:
         mov dx,offset errorm
         mov ah,09h
         int 21h
         mov dl,7
         mov ah,2
         int 21h
         jmp done3
tr:                                ;for jmp do
         mov dl,al
         mov ah,02h
         int 21h
         mov dl,10
```

```
        int 21h
        mov dl,13
        int 21h
        jmp do
l2:
        mov dx,offset mesg2
        mov ah,09h
        int 21h
rd2:
        mov al,0       ;判断有无输入
        mov ah,01  ;read counter2
        int 21h
        cmp al,0
        jz rd2
        cmp al,13           ;if enter
        je fdone2
        cmp al,10
        je fdone2
        jmp tdone2
fdone2:  jmp done2
tdone2:
        cmp al,10
        je fdone2
        cmp al,'0'
        jb error
        cmp al,'9'
        ja error

        push ax
        mov ax,10
        mul counter2
        mov counter2,ax        ;counter2=counter2*10

        pop ax
        sub bx,bx
        mov bl,al
        sub bl,30h             ;bh=0
        add counter2,bx        ;counter2=counter2+input
        jmp rd2
```

```
done1:
        jmp 12
done2:
        jmp out8253                ;after enter two counters
                                   ; set 8253 and do it
done3:
        mov dx,offset mesg3
        mov ah,09h
        int 21h
13:     mov ah,07h
        int 21h
        cmp al,'Y'
        je tr
        cmp al,'y'
        je tr
        cmp al,'N'
        je quit
        cmp al,'n'
        je quit
        mov dl,7
        mov ah,02h
        int 21h
        jmp 13
out8253:                           ;work code
        mov al,00110110b
        mov dx,283h
        out dx,al
        mov ax, counter1
        mov dx,280h
        out dx,al
        mov al,ah
        out dx,al
        mov al,01110110b
        mov dx,283h
        out dx,al
        mov ax,counter2
        mov dx,281h
        out dx,al
        mov al,ah
```

```
        out dx,al
        mov cx,2801
delay:  loop delay
        jmp done3
quit:  ;return to DOS
      mov dx,offset mesg4
      mov ah,9
      int 21h
      mov ax,4c00h
      int 21h
main endp
code ends
end start
```

实验四　可编程并行接口（一）（8255方式0）

一、实验目的

1、通过实验，掌握8255工作于方式0以及设置A口为输出口，C口为输入口的方法。

2、掌握8255三个数据端口与被测IC芯片的硬件连接方法。

3、通过实验掌握用8255并行口模拟集成电路测试仪，对集成电路进行逻辑测试的方法。

二、实验内容

1、实验电路如图4-1，8255C口接逻辑电平开关K0～K7，A口接LED显示电路L0～L7。

2、编程从8255C口输入数据，再从A口输出。

3、按图4-2连接硬件电路，以测试3-8译码器74LS138为例。

人机交互界面设计：按照集成电路74LS138芯片的引脚图显示在屏幕上，如下所示：

```
+------------------------------+
|   A    B    C   G1  G2A+G2B  |
|            74LS138           |
|Y0  Y1  Y2  Y3  Y4  Y5  Y6  Y7|
+------------------------------+
      Test Again ?(Y/N)','$'
```

原理图为图4-3，流程图为4-4：

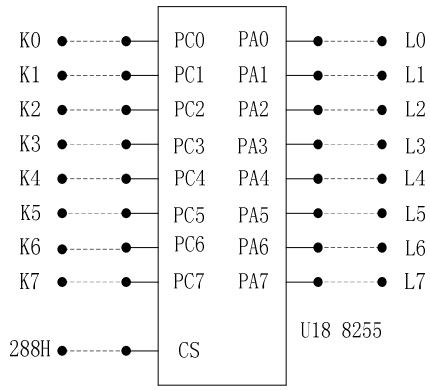

K0
K1
K2
K3
K4
K5
K6
K7
288H
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
CS
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
U18 8255
L0
L1
L2
L3
L4
L5
L6
L7

图4-1

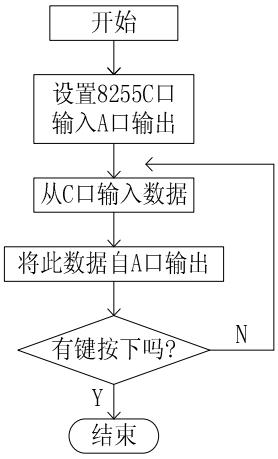

开始
设置8255C口
输入A口输出
从C口输入数据
将此数据自A口输出
有键按下吗?
N
Y
结束

图4-2

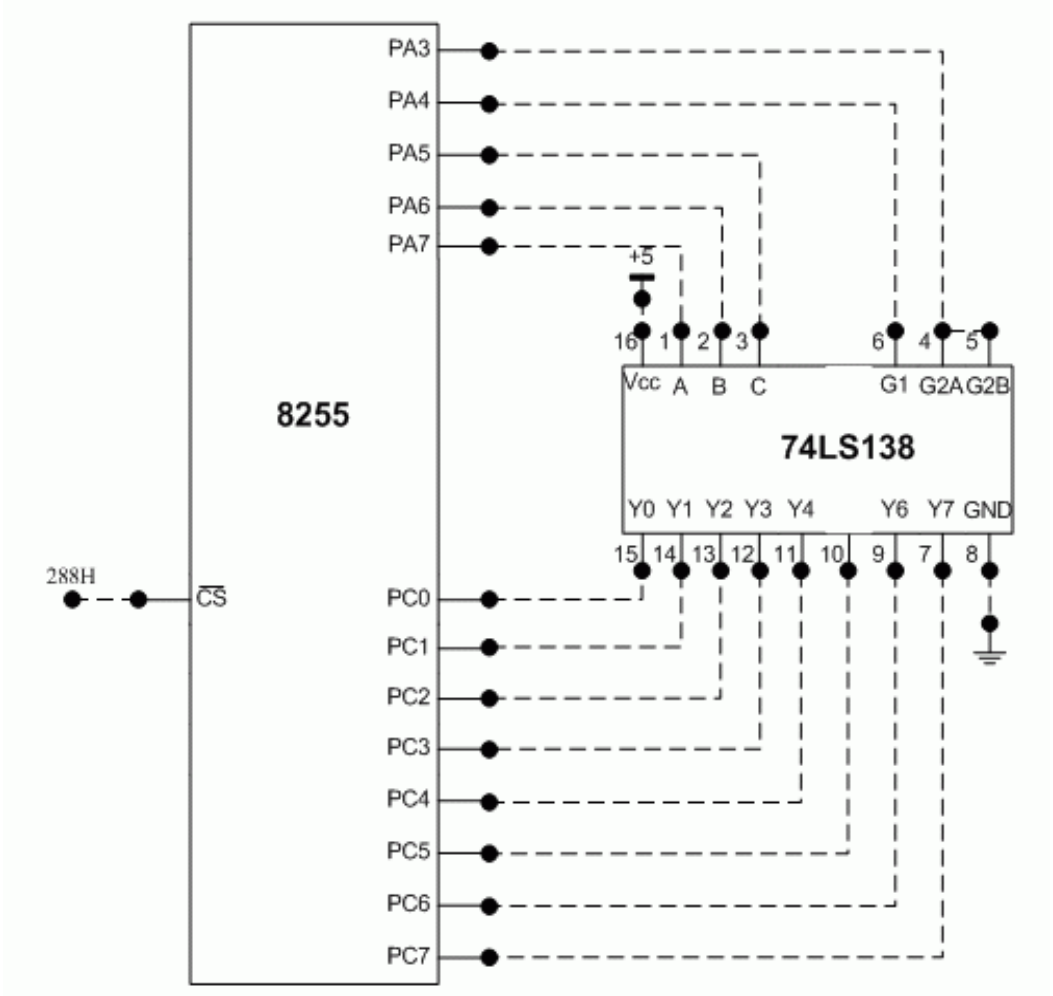

8255
PA3
PA4
PA5
PA6
PA7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
288H
CS
+5
74LS138
Vcc A B C G1 G2A G2B
Y0 Y1 Y2 Y3 Y4 Y6 Y7 GND
16 1 2 3 6 4 5
15 14 13 12 11 10 9 7 8

图4-3

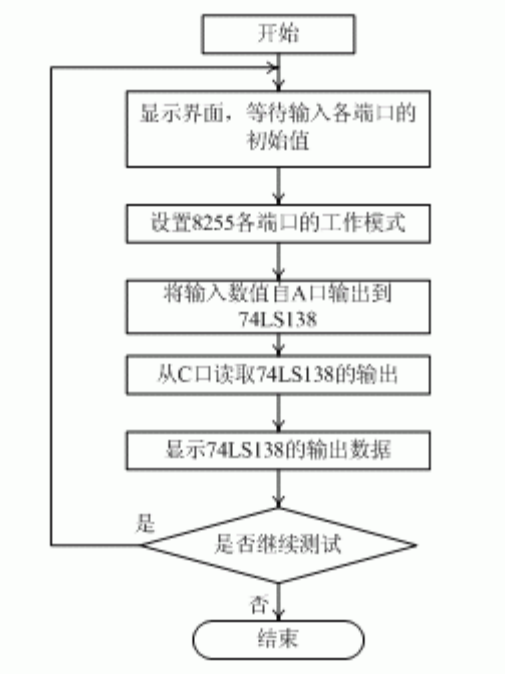


图4-4

三、编程提示

1、8255控制寄存器端口地址　　28BH
　A口的地址　　288H
　C口的地址　　28AH

2、参考流程图(见图4-2、图4-4):

3、参考程序1:　E8255.ASM

```
io8255a     equ 288h
io8255b     equ 28bh
io8255c     equ 28ah
code   segment
assume cs:code
start:
        mov dx,io8255b          ;设8255为C口输入,A口输出
        mov al,8bh
        out dx,al
inout:  mov dx,io8255c          ;从C口输入一数据
        in al,dx
        mov dx,io8255a          ;从A口输出刚才自C口
        out dx,al               ;所输入的数据
        mov dl,0ffh             ;判断是否有按键
        mov ah,06h
        int 21h
```

```
        jz inout                ;若无,则继续自C口输入,A口输出
        mov ah,4ch              ;否则返回
        int 21h
 code ends
 end start
```

4、参考程序2: E8255_1.ASM

```
data   segment
  chip db 13,10
       db 13,10
       db '              Program to test the chip of 74LS138',13,10
       db 13,10
       db 13,10
       db '               +-------------------------------+',13,10
       db '               |   A    B    C   G1  G2A+G2B  |',13,10
       db '               |                               |',13,10
       db '               |           74LS138             |',13,10
       db '               |                               |',13,10
       db '               |Y0  Y1  Y2  Y3  Y4  Y5  Y6  Y7|',13,10
       db '               +-------------------------------+',13,10,'$'
  mess db 'After you have ready,Please press any key !','$'
  mes2 db 'Test Again ?(Y/N)','$'
  InA  db 0
  OutC db 0
  cll  db '                                                ','$'
data   ends

;*****************************************
code   segment
       assume cs:code,ds:data
start: mov      ax,data
       mov      ds,ax
again: call     cls
       call     InputB
;Output CTRLcode(write) to 28Bh
       mov      dx,28bh
       mov      al,10001011b
       out      dx,al
;Output In to 288h
       mov      dx,288h
```

```
        mov     al,InA
        out     dx,al
        call    OutputC
jmp1:   mov     ah,2
        mov     dh,15
        mov     dl,20
        int     10h
        mov     ah,09
        lea     dx,mes2
        int     21h
        mov     ah,1
        int     21h
        cmp     al,'y'
        je      again
        cmp     al,'n'
        je      exit
        mov     ah,2
        mov     dh,15
        mov     dl,0
        int     10h
        lea     dx,cll
        mov     ah,9
        int     21h
        jmp     jmp1
exit:   mov     ah,4ch
        int     21h

;****************************************
InputB proc    near
        mov     ah,2
        mov     bh,0
        mov     dx,0
        int     10h
        mov     ah,09
        lea     dx,chip
        int     21h
        mov     ah,2
        mov     bh,0
        mov     dh,15
```

```
        mov     dl,10
        int     10h
        mov     ah,09h
        lea     dx,mess
        int     21h
        mov     ah,0ch
        mov     al,08h
        int     21h
wait1:  mov     ah,0Bh
        int     21h
        cmp     al,0
        jne     wait1
        mov     ah,2
        mov     bh,0
        mov     dh,15
        mov     dl,10
        int     10h
        lea     dx,cll
        mov     ah,9
        int     21h
        mov     dh,4
        mov     dl,18
jmp3:   push    dx
        mov     ah,2
        mov     bh,0
        int     10h
jmp4:   mov     ah,7
        int     21h
        cmp     al,'1'
        jne     jmp2
        mov     ah,2
        xchg    al,dl
        int     21h
        mov     cl,1
        mov     bl,InA
        sal     bl,cl
        add     bl,1
        mov     InA,bl
        jmp     jmp5
```

```
jmp2:  cmp     al,'0'
       jne     jmp4
       mov     ah,2
       xchg    al,dl
       int     21h
       mov     cl,1
       mov     bl,InA
       sal     bl,cl
       mov     InA,bl
jmp5:  pop     dx
       add     dl,5
       cmp     dl,43
       jb      jmp3
       mov     cl,3
       mov     bl,InA
       sal     bl,cl
       mov     InA,bl
       ret
InputB endp
;*****************************************
cls    proc    near
       mov     ah,6
       mov     al,0
       mov     ch,0
       mov     cl,0
       mov     dh,24
       mov     dl,79
       mov     bh,7
       int     10h
       ret
cls    endp
;*****************************************
OutputC proc     near
       mov      dx,28ah
       in       al,dx
       mov      OutC,al
       mov      dh,12
       mov      dl,16
j:     push     dx
```

```
        mov     ah,2
        mov     bh,0
        int     10h
        mov     al,OutC
        mov     bl,01h
        and     bl,al
        mov     cl,1
        shr     al,cl
        mov     OutC,al
        add     bl,30h
        xchg    bl,dl
        mov     ah,2
        int     21h
        pop     dx
        add     dl,4
        cmp     dl,46
        jb      j
        ret
OutputC endp
;*****************************************
code    ends
        end     start
```

实验五　七段数码管

一、实验目的

掌握数码管显示数字的原理

二、实验内容

1、静态显示:按图5-1连接好电路，将8255的A口PA0～PA6分别与七段数码管的段码驱动输入端a～ g 相连，位码驱动输入端S1接+5V(选中)，S0、dp接地(关闭)。编程从键盘输入一位十进制数字(0～9)，在七段数码管上显示出来。

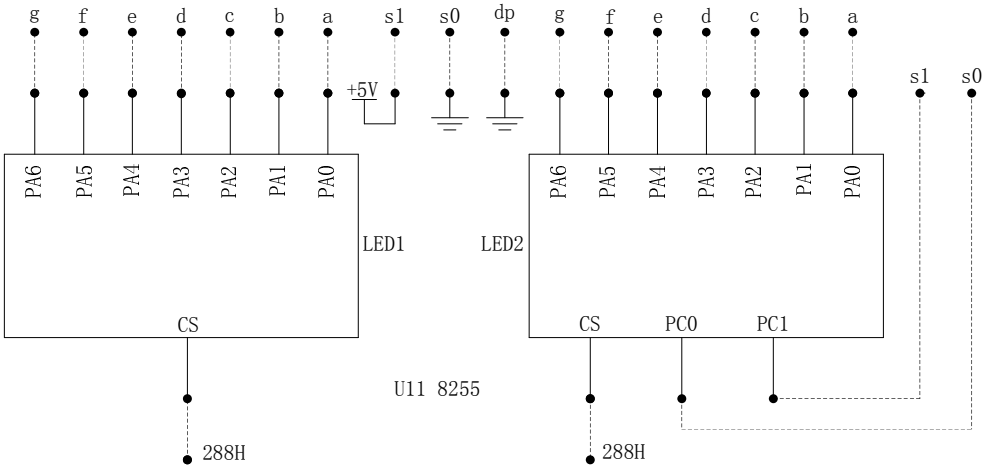


图5-1

图5-2

2、动态显示:按图5-2连接好电路，七段数码管段码连接不变，位码驱动输入端S1，S0接8255 C口的PC1，PC0。编程在两个数码管上显示“56”。

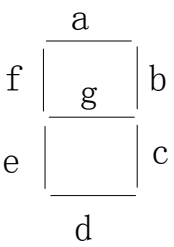
3、动态显示(选作):使用图5-2的电路，编程在两个数码管上循环显示“00-99”。

三、编程提示

1、实验台上的七段数码管为共阴型，段码采用同相驱动，输入端加高电平,选中的数码管亮，位码加反相驱动器，位码输入端高电平选中。

2、七段数码管的字型代码表如下表：

显示字形	g	e	f	d	c	b	a	段码
0	0	1	1	1	1	1	1	3fh
1	0	0	0	0	1	1	0	06h
2	1	0	1	1	0	1	1	5bh
3	1	0	0	1	1	1	1	4fh
4	1	1	0	0	1	1	0	66h
5	1	1	0	1	1	0	1	6dh
6	1	1	1	1	1	0	1	7dh
7	0	0	0	0	1	1	1	07h
8	1	1	1	1	1	1	1	7fh
9	1	1	0	1	1	1	1	6fh



3、参考流程图(见图5-3、图5-4)

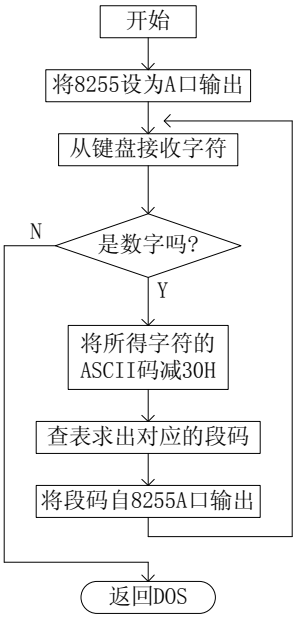


图5-3

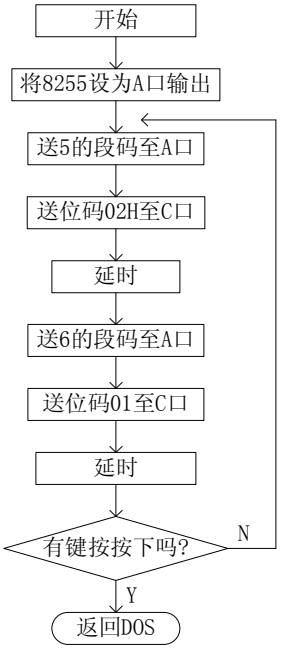


图5-4

4、参考程序1: LED1.ASM

```
data  segment
io8255a      equ 288h
io8255b      equ 28bh
led          db  3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh
mesg1        db  0dh,0ah,'Input a num (0--9),other key is exit:',0dh,0ah,'$'
data ends
code  segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        mov dx,io8255b          ;使8255的A口为输出方式
        mov ax,80h
        out dx,al
sss:    mov dx,offset mesg1     ;显示提示信息
        mov ah,09h
        int 21h
        mov ah,01               ;从键盘接收字符
        int 21h
        cmp al,'0'              ;是否小于0
```

```
        jl  exit                ;若是则退出
        cmp al,'9'              ;是否大于9
        jg  exit                ;若是则退出
        sub al,30h              ;将所得字符的ASCII码减30H
        mov bx,offset led       ;bx为数码表的起始地址
        xlat                    ;求出相应的段码
        mov dx,io8255a          ;从8255的A口输出
        out dx,al
        jmp sss                 ;转SSS
exit:   mov ah,4ch              ;返回
        int 21h
code ends
end start
```

5、参考程序2: LED2.ASM

```
data  segment
io8255a     equ 28ah
io8255b     equ 28bh
io8255c     equ 288h
led         db  3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh ;段码
buffer1     db  6,5             ;存放要显示的个位和十位
bz          dw  ?               ;位码
data ends
code  segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        mov dx,io8255b          ;将8255设为A口输出
        mov al,80h
        out dx,al
        mov di,offset buffer1   ;设di为显示缓冲区
loop2:  mov bh,02
lll:    mov byte ptr bz,bh
        push di
        dec di
        add di, bz
        mov bl,[di]             ;bl为要显示的数
        pop di
        mov al,0
```

```
        mov dx,io8255a
        out dx,al
        mov bh,0
        mov si,offset led       ;置led数码表偏移地址为SI
        add si,bx               ;求出对应的led数码
        mov al,byte ptr [si]
        mov dx,io8255c          ;自8255A的口输出
        out dx,al
        mov al,byte ptr bz      ;使相应的数码管亮
        mov dx,io8255a
        out dx,al
        mov cx,3000
delay:  loop delay              ;延时
        mov bh,byte ptr bz
        shr bh,1
        jnz lll
        mov dx,0ffh
        mov  ah,06
        int  21h
        je  loop2               ;有键按下则退出
        mov dx,io8255a
        mov al,0                ;关掉数码管显示
        out dx,al
        mov ah,4ch              ;返回
        int 21h
code ends
end start
```

6、参考程序3: LED3.ASM

```
data  segment
io8255a     equ 28ah
io8255b     equ 28bh
io8255c     equ 288h
led         db  3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh ;段码
buffer1     db  0,0             ;存放要显示的十位和个位
bz          dw  ?               ;位码
data ends
code  segment
assume cs:code,ds:data
start:
```

```
        mov ax,data
        mov ds,ax
        mov dx,io8255b          ;将8255设为A口输出
        mov al,80h
        out dx,al
        mov di,offset buffer1   ;设di为显示缓冲区
loop1:  mov cx,030h             ;循环次数
loop2:  mov bh,02
lll:    mov byte ptr bz,bh
        push di
        dec di
        add di, bz
        mov bl,[di]             ;bl为要显示的数
        pop di
        mov bh,0
        mov si,offset led       ;置led数码表偏移地址为SI
        add si,bx               ;求出对应的led数码
        mov al,byte ptr [si]
        mov dx,io8255c          ;自8255A的口输出
        out dx,al
        mov al,byte ptr bz      ;使相应的数码管亮
        mov dx,io8255a
        out dx,al
        push cx
        mov cx,100
delay:  loop delay              ;延时
        pop cx
        mov al,00h
        out dx,al
        mov bh,byte ptr bz
        shr bh,1
        jnz lll
        loop loop2              ;循环延时
        mov ax,word ptr [di]
        cmp ah,09
        jnz set
        cmp al,09
        jnz set
        mov ax,0000
```

```
        mov [di],al
        mov [di+1],ah
        jmp loop1
set:    mov ah,01
        int 16h
        jne exit                ;有键按下则转exit
        mov ax,word ptr [di]
        inc al
        aaa
        mov [di],al             ;al为十位
        mov [di+1],ah           ;ah中为个位
        jmp loop1
exit:   mov dx,io8255a
        mov al,0                ;关掉数码管显示
        out dx,al
        mov ah,4ch              ;返回
        int 21h
code ends
end start
```

实验六　继电器控制

一、实验目的

1、了解微机控制直流继电器的一般方法。

2、进一步熟悉使用8255、8253。

二、实验内容

实验电路如图6-1，按虚线连接电路：CLK0接1MHZ，GATE0，GATE1，接+5V，OUT0接CLK1，OUT1接PA0，PC0接继电器驱动电路的开关输入端Ik。编程使用8253定时，让继电器周而复始的闭合5秒钟(指示灯灯亮)，断开5秒钟(指示灯灯灭)。

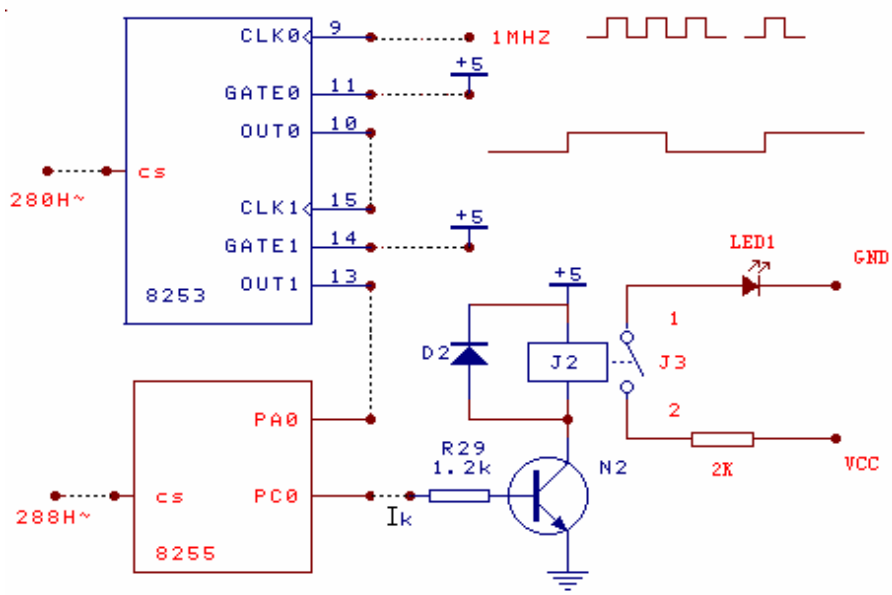


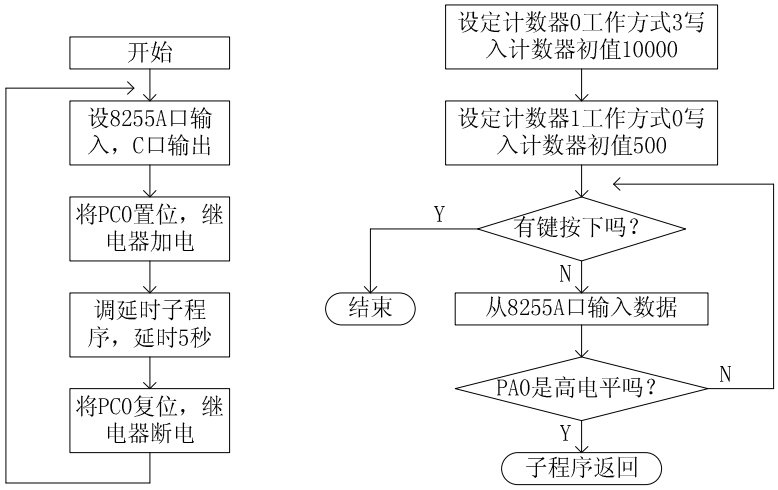
图6-1

三、编程提示

1、将8253计数器0设置为方式3、计数器1设置为方式0并联使用，CLK0接1MHZ时钟，设置两个计数器的初值(乘积为5000000)启动计数器工作后，经过5秒钟OUT1输出高电平。通过8255A口查询OUT1的输出电平，用C口PC0输出开关量控制继电器动作。

2、继电器开关量输入端输入“1”时，继电器常开触点闭合，电路接通，指示灯发亮，输入“0”时断开，指示灯熄灭。

3、参考流程图(见图6-2)：



(A)主程序　　(B)延时子程序

图6-2

4、参考程序: JDQ.ASM

```
io8255a     equ 280h
io8255b     equ 281h
io8255c     equ 283h
io8255d     equ 288h
io8255e     equ 28bh
code segment
assume cs:code
start:
        mov dx,io8255e      ;设8255为A口输入,C口输出
        mov al,90h
lll:    out dx,al
        mov al,01           ;将PC0置位
        out dx,al
        call delay          ;延时5s
        mov al,0            ;将PC0复位
        out dx,al
        call delay          ;延时5s
        jmp lll             ;转lll
delay proc near             ;延时子程序
        push dx
        mov dx,io8255c      ;设8253计数器为方式3
        mov al,36h
        out dx,al
        mov dx,io8255a
        mov ax,10000        ;写入计数器初值10000
        out dx,al
        mov al,ah
        out dx,al
        mov dx,io8255c
        mov al,70h          ;设计数器1为工作方式0
        out dx,al
        mov dx,io8255b
        mov ax,500          ;写入计数器初值500
        out dx,al
        mov al,ah
        out dx,al
lll2:   mov ah,06           ;是否有键按下
        mov dl,0ffh
```

```
        int 21h
        jne exit            ;若有则转exit
        mov dx,io8255d
        in  al,dx           ;查询8255的PA0是否为高电平
        and al,01
        jz  l12             ;若不是则继续
        pop dx
        ret                 ;定时时间到，子程序返回
exit:   mov ah,4ch
        int 21h
delay endp
code ends
end start
```

实验七　竞赛抢答器

一、实验目的

1、了解微机化竞赛抢答器的基本原理。

2、进一步学习使用并行接口。

二、实验内容

图7-1为竞赛抢答器(模拟)的原理图，逻辑开关K0～K7代表竞赛抢答按钮0～7号，当某个逻辑电平开关置“ 1 ”时，相当某组抢答按钮按下。在七段数码管上将其组号(0～7)显示出来，并使微机扬声器响一下。

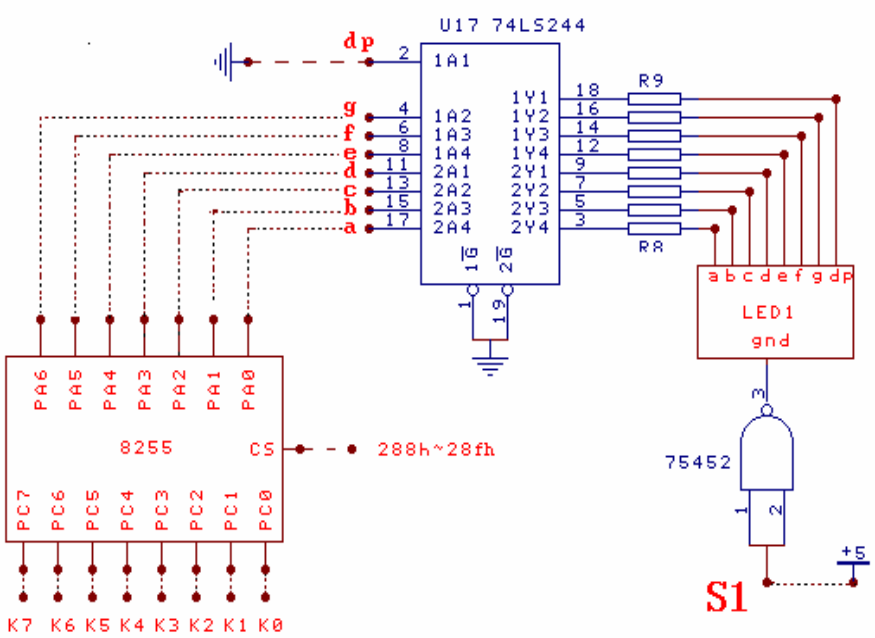


图7-1

三、编程提示

设置8255为C口输入、A口输出，读取C口数据，若为0表示无人抢答，若不为0则有人抢答。根据读取数据可判断其组号。从键盘上按空格键开始下一轮抢答，按其它键程序退出。

四、参考流程图（见图7-2）

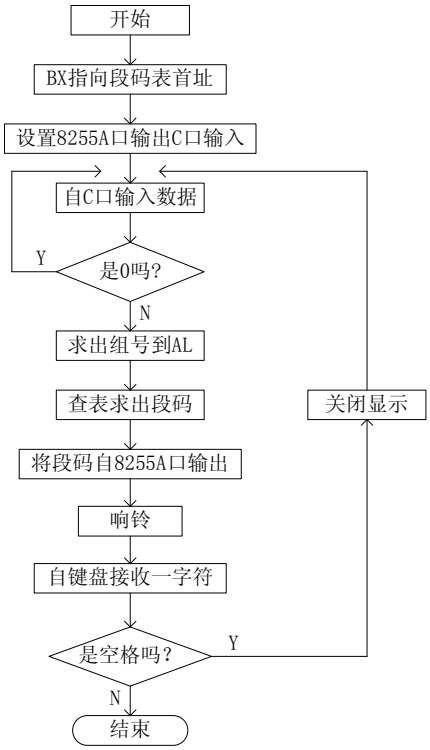


图7-2

五、参考程序： QDQ.ASM

```
data segment
io8255a     equ 28ah
io8255b     equ 28bh
io8255c     equ 288h
led         db  3fh,06h,5bh,4fh,66h,6dh,7dh,07h ;数码表
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        mov dx,io8255b          ;设8255为A口输出,C口输入
        mov ax,89h
        out dx,al
        mov bx,offset led       ;使BX指向段码管首址
sss:    mov dx,io8255a
        in  al,dx               ;从8255的C口输入数据
        or  al,al               ;比较是否为0
        je  sss                 ;若为0,则表明无键按下，转sss
        mov cl,0ffh             ;cl作计数器,初值为-1
rr:     shr al,1
        inc cl
        jnc rr
        mov al,cl
        xlat
        mov dx,io8255c
        out dx,al
        mov dl,7                ;响铃 ASCII码为07
        mov ah,2
        int 21h
wai:    mov ah,1
        int 21h
        cmp al,20h              ;是否为空格
        jne eee                 ;不是,转eee
        mov al,0                ;是,关灭灯
        mov dx,io8255c
        out dx,al
        jmp sss
```

```
eee:    mov ah,4ch              ;返回
        int 21h
code  ends
end start
```

实验八　交通灯控制实验

一、实验目的

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

二、实验内容

如图8-1，L7、L6、L5作为南北路口的交通灯与PC7、PC6、PC5相连，L2、L1、L0作为东西路口的交通灯与PC2、PC1、PC0相连。编程使六个灯按交通灯变化规律亮灭。

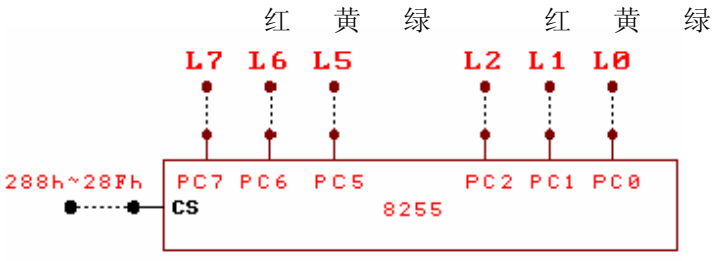


图8-1

三、编程提示：十字路口交通灯的变化规律要求：

（1） 南北路口的绿灯、东西路口的红灯同时亮30秒左右。

（2） 南北路口的黄灯闪烁若干次，同时东西路口的红灯继续亮。

（3） 南北路口的红灯、东西路口的绿灯同时亮30秒左右。

（4） 南北路口的红灯继续亮、同时东西路口的黄灯亮闪烁若干次。

（5） 转(1)重复。

四、参考流程图

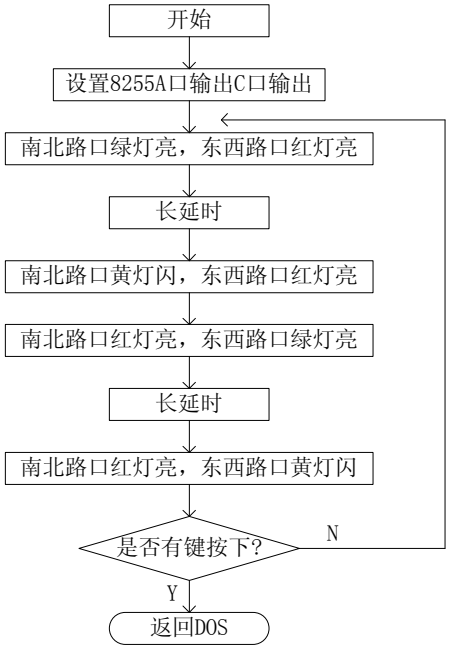


图8-2

五、参考程序：JTD.ASM

```
data segment
io8255a     equ 28ah
io8255b     equ 28bh
portc1      db  24h,44h,04h,44h,04h,44h,04h    ;六个灯可能
            db  81h,82h,80h,82h,80h,82h,80h    ;的状态数据
            db  0ffh                           ;结束标志
data ends
code  segment
assume  cs:code,ds:data
start:  mov ax,data
        mov ds,ax
        mov dx,io8255b
        mov al,90h
        out dx,al           ;设置8255为C口输出
        mov dx,io8255a
re_on:  mov bx,0
on:     mov al,portc1[bx]
        cmp al,0ffh
        jz  re_on
        out dx,al           ;点亮相应的灯
```

```
        inc bx
        mov cx,200          ;参数赋初值
        test al,21h         ;是否有绿灯亮
        jz  de1             ;没有,短延时
        mov cx,2000         ;有,长延时
de1:    mov di,9000         ;di赋初值9000
de0:    dec di              ;减1计数
        jnz de0             ;di不为0
        loop de1
        push dx
        mov ah,06h
        mov dl,0ffh
        int 21h
        pop dx
        jz  on              ;没有,转到on
exit:   mov ah,4ch          ;返回
        int 21h
code ends
end start
```

实验九　中断

一、实验目的

1、掌握PC机中断处理系统的基本原理。

2、学会编写中断服务程序。

二、实验原理与内容

1、实验原理

PC机用户可使用的硬件中断只有可屏蔽中断，由8259中断控制器管理。中断控制器用于接收外部的中断请求信号，经过优先级判别等处理后向CPU发出可屏蔽中断请求。IBMPC、PC/XT

机内有一片8259中断控制器对外可以提供8个中断源：

中断源	中断类型号	中断功能
IRQ0	08H	时钟
IRQ1	09H	键盘
IRQ2	0AH	保留
IRQ3	0BH	串行口2
IRQ4	0CH	串行口1
IRQ5	0DH	硬盘
IRQ6	0EH	软盘
IRQ7	0FH	并行打印机

8个中断源的中断请求信号线IRQ0～IRQ7在主机的62线ISA总线插座中可以引出，系统已设定中断请求信号为“边沿触发”，普通结束方式。对于PC/AT及286以上微机内又扩展了一片8259中断控制，IRQ2用于两片8259之间级连，对外可以提供16个中断源：

中断源	中断类型号	中断功能
IRQ8	070H	实时时钟
IRQ9	071H	用户中断
IRQ10	072H	保留
IRQ11	073H	保留
IRQ12	074H	保留
IRQ13	075H	协处理器
IRQ14	076H	硬盘
IRQ15	077H	保留

TPC-USB实验板上，固定的接到了3号中断IRQ3上，即进行中断实验时，所用中断类型号为0BH。

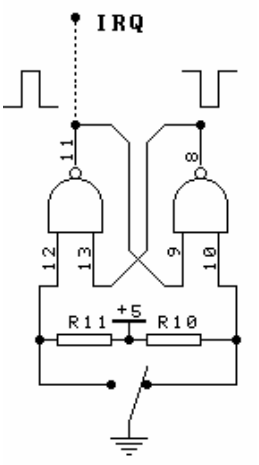


图9-1

2、实验内容

实验电路如图9-1，直接用手动产单脉冲作为中断请求信号(只需连接一根导线)。要求每按一次开关产生一次中断，在屏幕上显示一次“TPCA Interrupt!”，中断10次后程序退出。

三、参考流程图

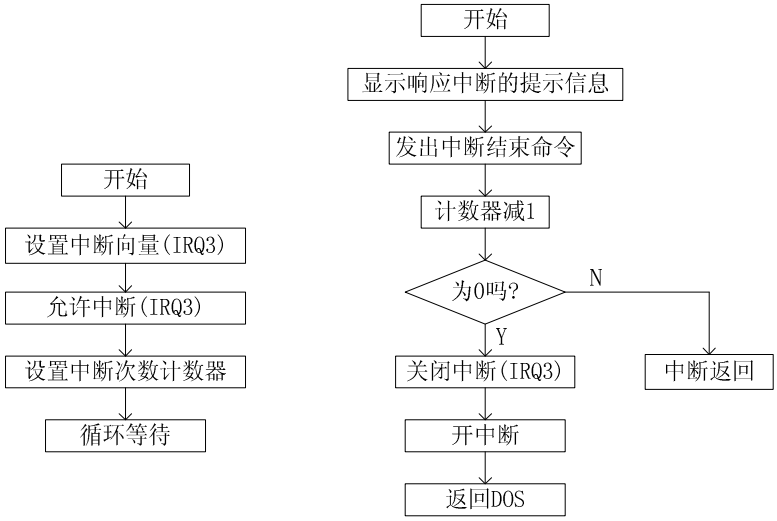


图9-2

四、参考程序：INT.ASM

```
data segment
mess db 'TPCA interrupt!',0dh,0ah,'$'
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,cs
        mov ds,ax
        mov dx,offset int3
        mov ax,250bh
        int 21h                 ;设置IRQ3的中断矢量
        in al,21h               ;读中断屏蔽寄存器
        and al,0f7h             ;开放IRQ3中断
        out 21h,al
        mov cx,10               ;记中断循环次数为10次
        sti
ll:     jmp ll
int3:                           ;中断服务程序
        mov ax,data
        mov ds,ax
        mov dx,offset mess
        mov ah,09               ;显示每次中断的提示信息
        int 21h
```

```
        mov al,20h
        out 20h,al              ;发出EOI结束中断
        loop next
        in al,21h
        or al,08h               ;关闭IRQ3中断
        out 21h,al
        sti                     ;置中断标志位
        mov ah,4ch              ;返回DOS
        int 21h
next:   iret
code ends
end start
```

实验十　可编程并行接口（二）（8255方式 1 ）

一、实验目的

1、掌握8255工作方式 1 时的使用及编程。

2、进一步掌握中断处理程序的编写。

二、实验内容

1、按图10-1，8255方式1的输出电路连好线路。

2、编程:每按一次单脉冲按钮产生一个正脉冲使8255产生一次中断请求，让CPU进行一次中断服务:依次输出01H、02H，04H，08H，10H，20H，40H，80H使L0～L7依次发光，中断 8 次结束。

3、按图10-2，8255方式1输入电路，连好线路。

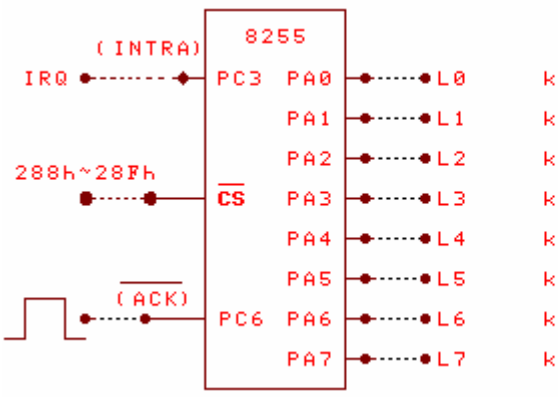


图10-1 输出电路

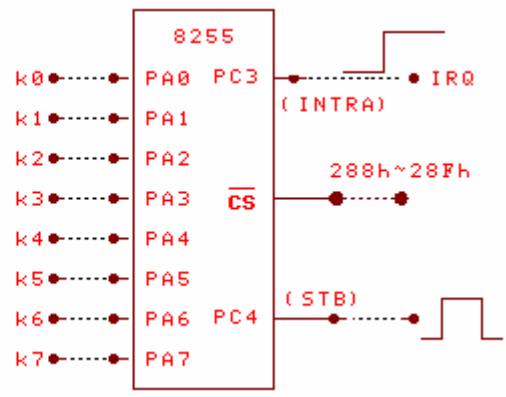
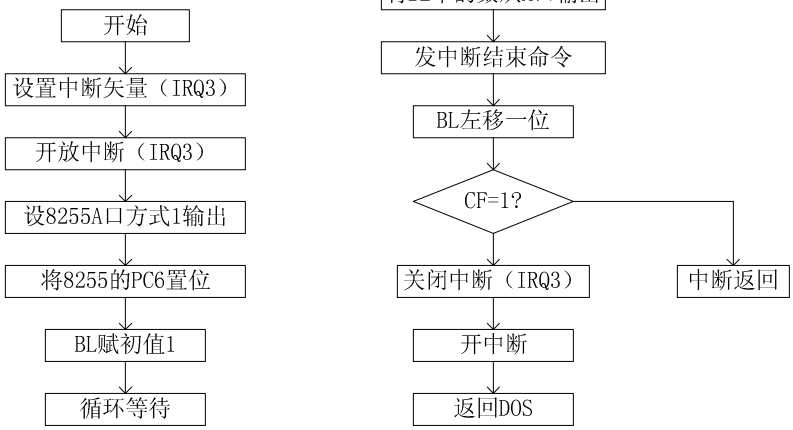


图10-2 输入电路

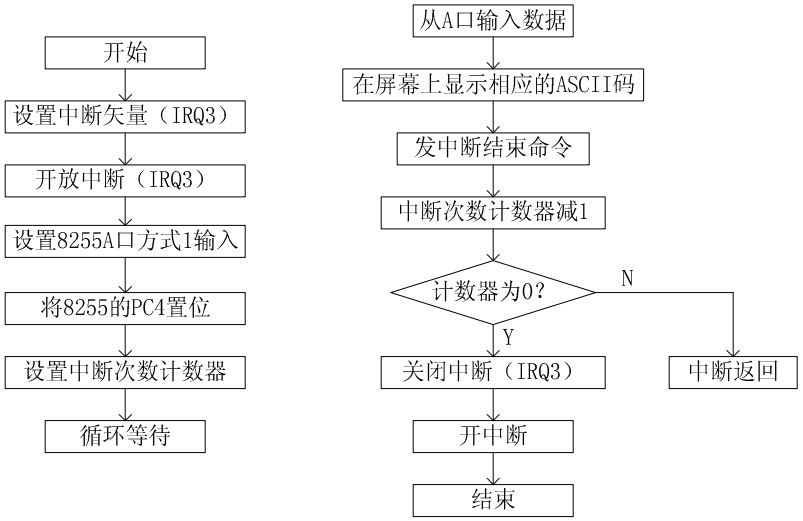
4、编程:每按一次单脉冲按钮产生一个正脉冲使8255产生一次中断请求，让CPU进行一次中断服务:读取逻辑电平开关预置的ASCII码，在屏幕上显示其对应的字符，中断 8 次结束。

三、参考流程图:（如图10-3、图10-4）



（A） 输出主程序　　（B） 输出中断服务程序

图10-3



（A） 输出主程序　　（B） 输出中断服务程序

图10-4

四、参考程序１: E8255-1o.ASM

```
code segment
assume cs:code
start:
        mov ax,cs
        mov ds,ax
        mov dx,offset int_proc
        mov ax,250bh          ;设外部中断int_proc类型号为0BH
        int 21h
        mov dx,21h
        in al,dx
```

```
        and al,0f7h          ;开放IRQ3中断
        out dx,al
        mov dx,28bh          ;置8255为A口方式1输出
        mov al,0a0h
        out dx,al
        mov al,0dh           ;将PC6置位
        out dx,al
        mov bl,1
ll:     jmp ll               ;循环等待
int_proc:
        mov al,bl
        mov dx,288h          ;将AL从8255的A口输出
        out dx,al
        mov al,20h
        out 20h,al
        shl bl,1
        jnc next             ;中断次数小于8，返回主程序
        in al,21h
        or al,08h            ;关闭IRQ7中断
        out 21h,al
        sti                  ;开中断
        mov ah,4ch           ;返回DOS
        int 21h
next:   iret
code ends
end start
```

五、参考程序 2： E8255-1i.ASM

```
code segment
assume cs:code
start:
        mov ax,cs
        mov ds,ax
        mov dx,offset int_proc  ;设置IRQ3中断矢量
        mov ax,250bh
        int 21h
        mov dx,21h
        in al,dx
        and al,0f7h             ;开放IRQ7中断
        out dx,al
```

```
        mov dx,28bh             ;设8255为A口方式1输入
        mov al,0b8h
        out dx,al
        mov al,09h
        out dx,al
        mov bl,8                ;BL为中断次数计数器
ll:     jmp ll
int_proc:                       ;中断服务程序
        mov dx,288h             ;自8255A口输入一数据
        in al,dx
        mov dl,al               ;将所输入的数据保存到DL
        mov ah,02h              ;显示ASCII码为DL的字符
        int 21h
        mov dl,0dh              ;回车
        int 21h
        mov dl,0ah              ;换行
        int 21h
        mov dx,20h              ;发出EOI结束命令
        mov al,20h
        out dx,al
        dec bl                  ;计数器减1
        jnz next                ;不为0则返回主程序
        in al,21h
        or al,08h
        out 21h,al              ;关IRQ3中断
        sti                     ;开中断
        mov ah,4ch              ;返回DOS
        int 21h
next:   iret
code ends
end start
```

实验十一　数/模转换器

一、实验目的

了解数/模转换器的基本原理，掌握DAC0832芯片的使用方法。

二、实验内容

1、实验电路原理如图11-1，DAC0832采用单缓冲方式，具有单双极性输入端(图中的Ua、Ub),利用debug输出命令(Out 290 数据)输出数据给DAC0832，用万用表测量单极性输出端Ua及双极性输出端Ub的电压，验证数字与电压之间的线性关系。

2、编程产生以下波形(从Ub输出，用示波器观察)

（1）锯齿波

（2）正弦波

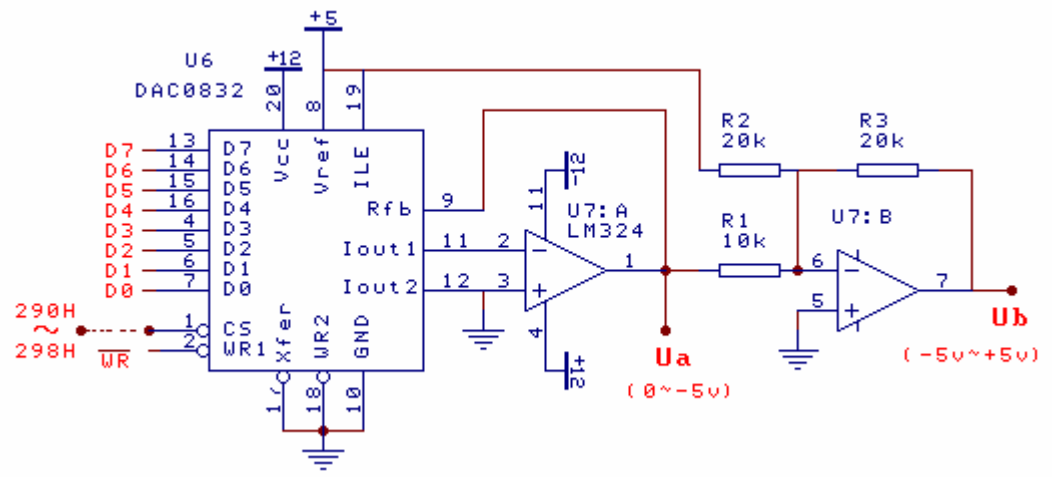


图11-1

三、编程提示

1、8位D/A转换器DAC0832的口地址为290H，输入数据与输出电压的关系为：

$$Ua = -\frac{U_{REF}}{256} \times N$$

$$Ub = -\frac{2U_{REF}}{256} \times N - 5$$

(UREF表示参考电压,N表示数数据)，这里的参考电压为ＰＣ机的＋５Ｖ电源。

2、产生锯齿波只须将输出到DAC0832的数据由0循环递增。产生正弦波可根据正弦函数建一个下弦数字量表，取值范围为一个周期，表中数据个数在16个以上。

四、参考流程图:

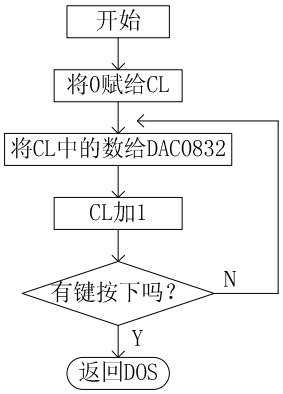


图11-2 锯齿波

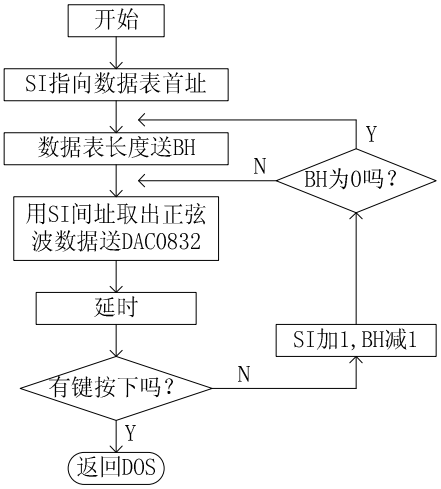


图11-3 正弦波

五、参考程序1: DA_1.ASM

```
io0832a      equ 290h
code segment
assume cs:code
start:
        mov cl,0
        mov dx,io0832a
lll:    mov al,cl
        out dx,al
        inc cl                  ;cl加1
        inc cl
        inc cl
        inc cl
        inc cl
        inc cl
        inc cl
        inc cl
        push dx
        mov ah,06h              ;判断是否有键按下
        mov dl,0ffh
        int 21h
        pop dx
        jz  lll                 ;若无则转LLL
        mov ah,4ch              ;返回
        int 21h
code ends
```

```
        end start
```

六、参考程序 2： DA_2.ASM

```
        data  segment
        io0832a     equ 290h
        sin         db 80h,96h,0aeh,0c5h,0d8h,0e9h,0f5h,0fdh
                    db 0ffh,0fdh,0f5h,0e9h,0d8h,0c5h,0aeh,96h
                    db 80h,66h,4eh,38h,25h,15h,09h,04h
                    db 00h,04h,09h,15h,25h,38h,4eh,66h       ;正弦波数据
        data ends
        code segment
        assume cs:code,ds:data
        start:
                mov ax,data
                mov ds,ax
        ll:     mov si,offset sin       ;置正弦波数据的偏移地址为SI
                mov bh,32               ;一组输出32个数据
        lll:    mov al,[si]             ;将数据输出到D/A转换器
                mov dx,io0832a
                out dx,al
                mov ah,06h
                mov dl,0ffh
                int 21h
                jne exit
                mov cx,1
        delay:  loop delay              ;延时
                inc si                  ;取下一个数据
                dec bh
                jnz lll                 ;若未取完32个数据则转lll
                jmp  ll
        exit:   mov ah,4ch              ;退出
                int 21h
        code ends
        end start
```

实验十二　模/数转换器

一、实验目的

了解模/数转换的基本原理，掌握ADC0809的使用方法。

二、实验内容

1、实验电路原理图如图12-1。通过实验台左下角电位器RW1输出0～5Ｖ直流电压送入ADC0809通道0(IN0),利用debug的输出命令启动Ａ/Ｄ转换器，输入命令读取转换结果，验证输入电压与转换后数字的关系。

启动IN0开始转换：O 0298 0

读取转换结果：　　I　0298

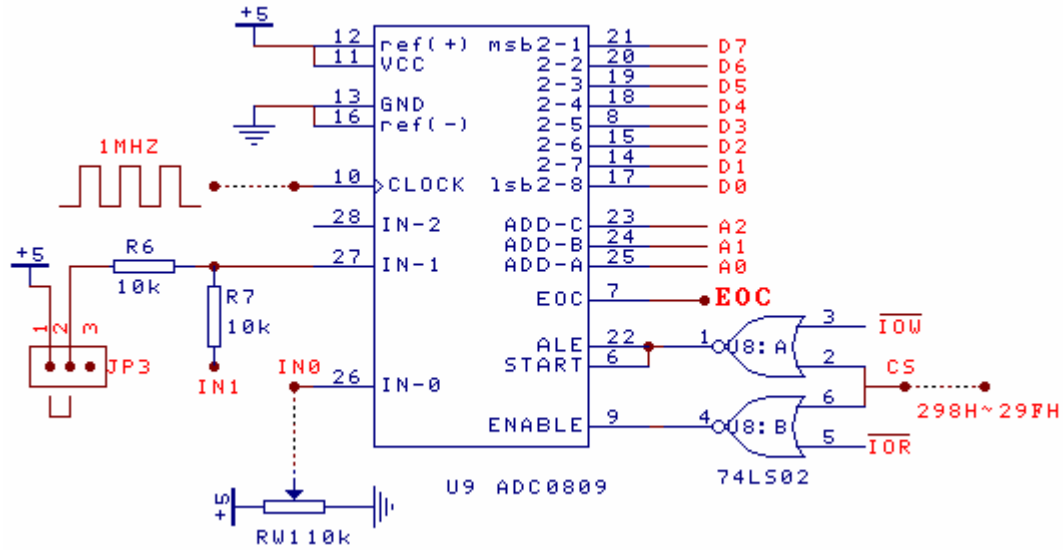


图12-1 模数转换电路

2、编程采集IN0输入的电压,在屏幕上显示出转换后的数据(用16进制数)。

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

三、实验提示

1、ADC0809的IN0口地址为298H，IN1口地址为299H。

2、IN0单极性输入电压与转换后数字的关系为：

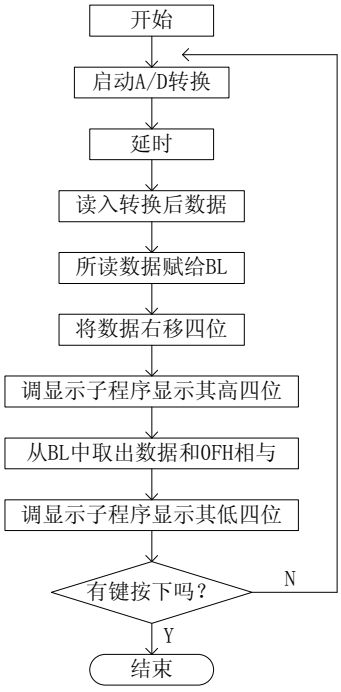
$$N = \frac{U_i}{U_{REF}/256}$$

其中Ui为输入电压，UREF为参考电压，这里的参考电压为PC机的＋５Ｖ电源。

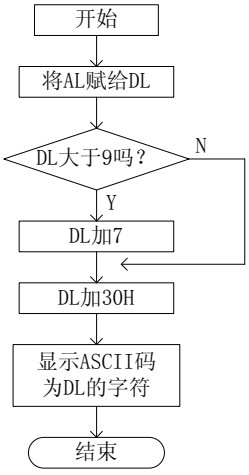
3、一次A/D转换的程序可以为

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

四、参考流程图（见图12-2、图12-3）



(A) 主程序



（B）显示子程序

图12-2

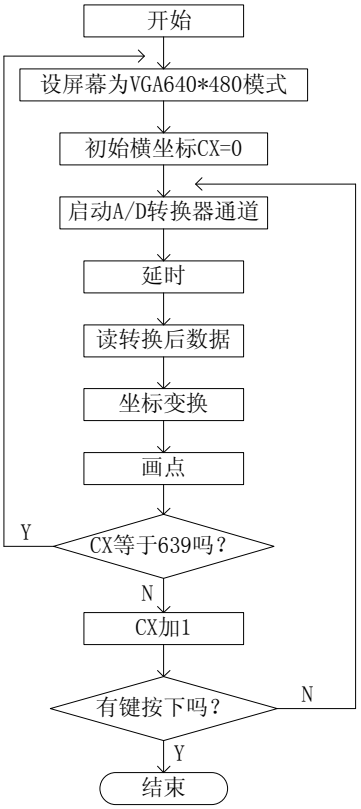


图12-3

五、参考程序１： AD_1.ASM

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

```
disp endp
code ends
end start
```

六、参考程序 2： AD_2.ASM

```
io0809b     equ 299h
code segment
assume    cs:code
start:  mov ax,0012h            ;设屏幕显示方式为VGA 640X480模示
        int 10h
start1: mov ax,0600h
        int 10h                 ;清屏
        and cx,0                ;cx为横坐标
draw:   mov dx,io0809b          ;启动A/D转换器通道1
        out dx,al
        mov bx,200              ;延时
delay:  dec bx
        jnz delay
        in al,dx                ;读入数据
        mov ah,0
        mov dx,368              ;dx为纵坐标
        sub dx,ax
        mov al,0ah              ;设置颜色
        mov ah,0ch              ;画点
        int 10h
        cmp cx,639              ;一行是否满
        jz start1               ;是则转start
        inc cx                  ;继续画点
        push dx
        mov ah,06h              ;是否有键按下
        mov dl,0ffh
        int 21h
        pop dx
        je draw                 ;无,则继续画点
        mov ax,0003             ;有恢复屏幕为字符方式
        int 10h
        mov ah,4ch              ;返回
        int 21h
code ends
end start
```

实验十三　串行通讯

一、实验目的

1、了解串行通讯的基本原理。

2、掌握串行接口芯片8251的工作原理和编程方法。

二、实验内容

1、按图13-1连接好电路,(8251插通用插座)其中8253计数器用于产生8251的发送和接收时钟，TXD和RXD连在一起。

2、编程：从键盘输入一个字符,将其ASCII码加 1 后发送出去,再接收回来在屏幕上显示，实现自发自收。

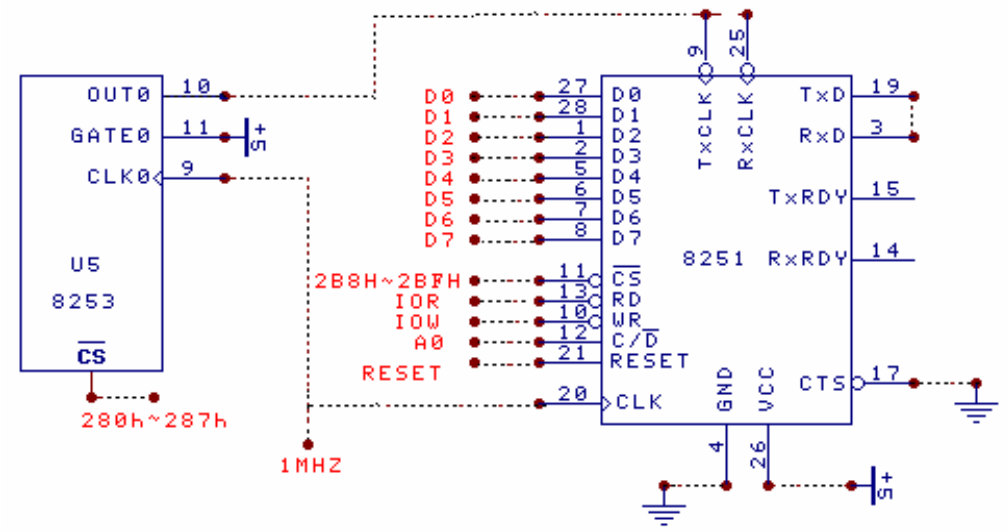


图13-1 串行通讯电路

三、实验提示

1、图示电路8251的控制口地址为2B9H,数据口地址为2B8H。

2、8253计数器的计数初值=时钟频率/(波特率×波特率因子)，这里的时钟频率接1MHz，波特率若选1200，波特率因子若选16，则计数器初值为52。

3、收发采用查询方式。

四、参考流程图（见图13-2）

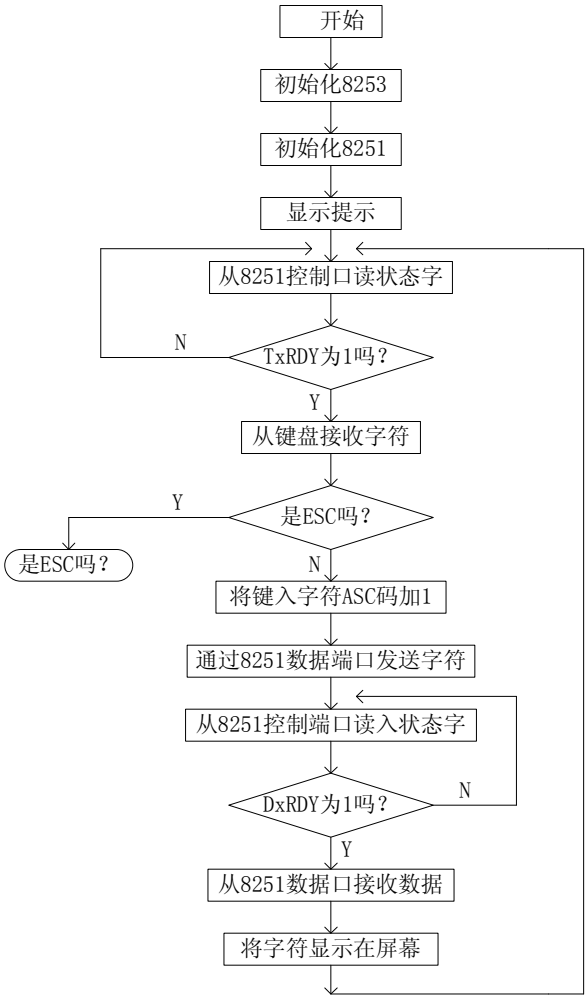


图13-2

五、参考程序： E8251.ASM

```
data segment
io8253a equ 280h
io8253b equ 283h
io8251a equ 2b8h
io8251b equ 2b9h
mes1    db  'you can play a key on the keybord!',0dh,0ah,24h
mes2    dd  mes1
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
```

```
        mov dx,io8253b      ;设置8253计数器0工作方式
        mov al,16h
        out dx,al
        mov dx,io8253a
        mov al,52           ;给8253计数器0送初值
        out dx,al
        mov dx,io8251b      ;初始化8251
        xor al,al
        mov cx,03           ;向8251控制端口送3个0
delay:  call out1
        loop delay
        mov al,40h          ;向8251控制端口送40H,使其复位
        call out1
        mov al,4eh          ;设置为1个停止位,8个数据位,波特率因子为16
        call out1
        mov al,27h          ;向8251送控制字允许其发送和接收
        call out1
        lds dx,mes2         ;显示提示信息
        mov ah,09
        int 21h
waiti:  mov dx,io8251b
        in al,dx
        test al,01          ;发送是否准备好
        jz waiti
        mov ah,01           ;是,从键盘上读一字符
        int 21h
        cmp al,27           ;若为ESC,结束
        jz exit
        mov dx,io8251a
        inc al
        out dx,al           ;发送
        mov cx,40h
s51:    loop s51            ;延时
next:   mov dx,io8251b
        in al,dx
        test al,02          ;检查接收是否准备好
        jz next             ;没有,等待
        mov dx,io8251a
        in al,dx            ;准备好,接收
```

```
        mov dl,al
        mov ah,02           ;将接收到的字符显示在屏幕上
        int 21h
        jmp waiti
exit:   mov ah,4ch          ;退出
        int 21h
out1 proc near              ;向外发送一字节的子程序
        out  dx,al
        push cx
        mov  cx,40h
gg:     loop gg             ;延时
        pop  cx
        ret
out1 endp
code ends
end start
```

实验十四　DMA传送

一、实验目的

1、掌握PC机工作环境下进行DMA方式数据传送(Block MODE和Demand Mode)(块传送、外部请求传送)方法。

2、掌握DMA的编程方法。

二、实验内容

1、按照图14-1将实验箱通用插座D区6116连接好。编程将USB的6116缓冲区D4000H,偏移量为0的一块数据循环写入字符A～Z，用Block MODE DMA方式传送到实验箱的6116的缓冲区D6000H上，并察看送出的数据是否正确。

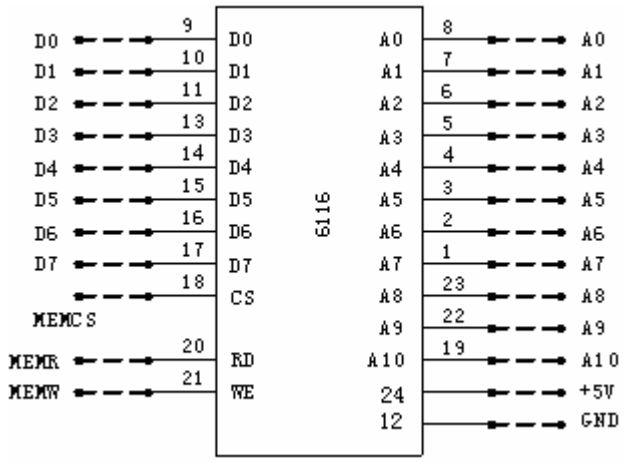


图14-1

2、用通用插座按图14-2连接好电路（74LS74利用实验台上的D触发器）。编程将主机内存缓冲区D4000H,偏移量为0的10个数据，使用Demand Mode DMA方式从内存向外设传送。

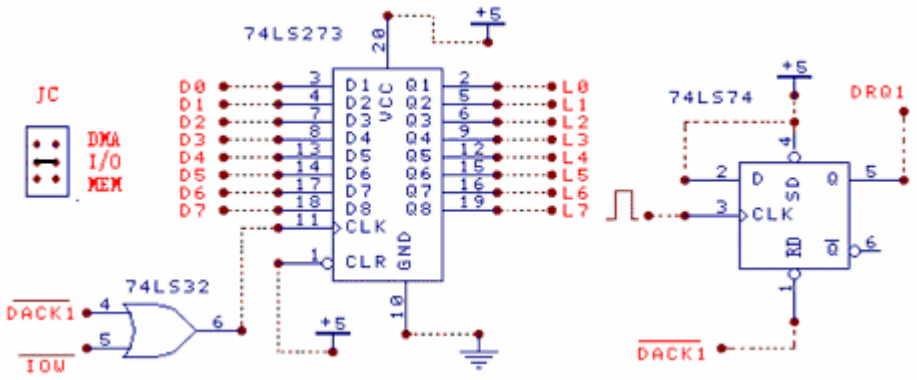


图14-2

3、用通用插座按图14-3连接好电路（74LS74利用实验台上的D触发器）。编程在主机内存缓冲区D4000H,偏移量为0的位置开辟数据缓冲区，使用Demand Mode DMA方式从外设向内存传送8个数据并存入数据缓冲区，编程不断显示缓冲区的数据。

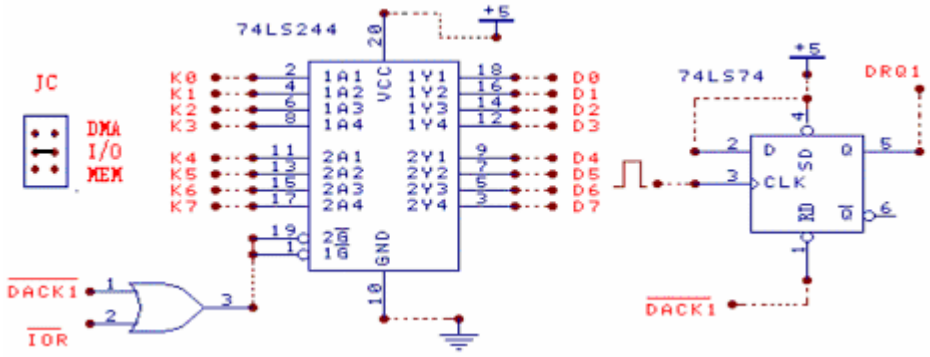


图14-3

三、实验提示

1、ＤＭＡ请求是由单脉冲输入到Ｄ触发器，由触发器的Ｑ端向DRQ1发出的。CPU响应后发出$\overline{\text{DACK}}$1，将触发器Q置成低电平以撤消请求。

2、汇编程序中，为避免与系统8237有冲突，TPC-USB模块上的8237端口范围为10H-1F，即按通常模式进行DMA编程时，对8237所有端口均加10H。

四、参考流程图（见图14-4、图14-5、图14-6）

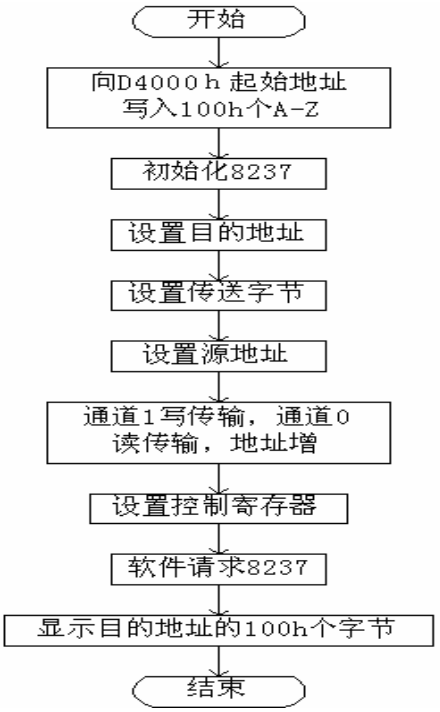


图14-4

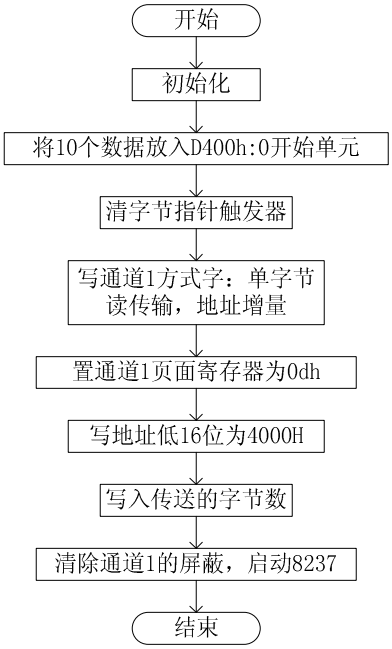


图14-5

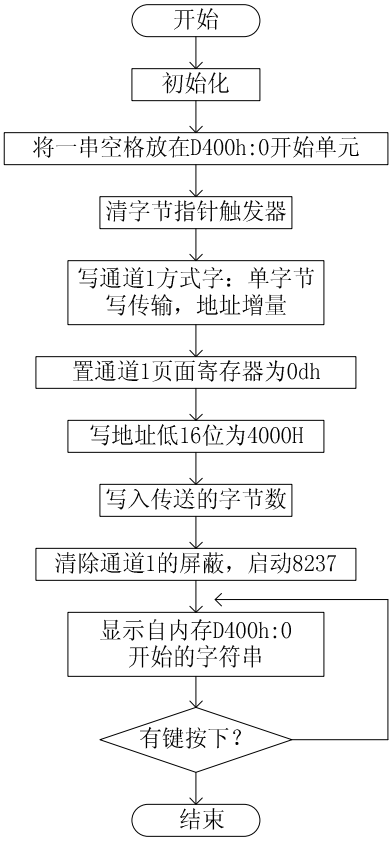


图14-6

五、参考程序1： DMA.asm

```
code segment
    assume cs:code
start:
        mov ax,0D000h
        mov es,ax
        mov bx,4000h
        mov cx,0ffh;100h
        mov dl,40h
rep1:   inc dl
        mov es:[bx],dl
        inc bx
        cmp dl,5ah
        jnz ss1
        mov dl,40h
        ss1: loop rep1

        mov dx,18h          ;关闭8237
        mov al,04h
        out dx,al
        mov dx,1dh          ;复位
        mov al,00h
        out dx,al
        mov dx,12h          ;写目的地址低位
        mov al,00h
        out dx,al
        mov dx,12h          ;写目的地址高位
        mov al,60h;
        out dx,al
        mov dx,13h          ;传送字节数低位
        mov al,0ffh;
        out dx,al
        mov dx,13h          ;传送字节数高位
        mov al,0;1h
        out dx,al
        mov dx,10h          ;源地址低位
        mov al,00h
        out dx,al
        mov dx,10h          ;源地址高位
```

```
        mov al,40h
        out dx,al
        mov dx,1bh          ;通道1写传输,地址增
        mov al,85h
        out dx,al
        mov dx,1bh          ;通道0读传输,地址增
        mov al,88h
        out dx,al
        mov dx,18h          ;DREQ低电平有效,存储器到存储器,开启8237
        mov al,41h
        out dx,al
        mov dx,19h          ;通道1请求
        mov al,04h
        out dx,al

        mov cx,0F000h
        delay: loop delay

        mov ax,0D000h;---
        mov es,ax
        mov bx,06000h
        mov cx,0ffh;
rep2:   mov dl,es:[bx]
        mov ah,02h
        int 21h
        inc bx
        loop rep2
        mov ax,4c00h
        int 21h
        code ends
        end start
```

六、参考程序2: DMA_0.asm

```
data segment
outdata db 01,02,04,08,10h,20h,40h,80h,0ffh,00h
data ends
extra segment at 0d400h
ext db 10 dup(?)
extra ends
code segment
```

```
    assume cs:code,ds:data,es:extra
start:
        mov ax,data
        mov ds,ax
        mov ax,extra
        mov es,ax
        lea si,outdata
        lea di,ext
        cld
        mov cx,10
        rep movsb
        out 1ch,al          ;清字节指针
        mov al,49h          ;写方式字
        out 1bh,al
        mov al,0dh          ;置地址页面寄存器
        out 83h,al
        mov al,0            ;写入基地址低十六位
        out 12h,al
        mov al,40h
        out 12h,al
        mov al,0ah          ;写入传送的字节数10
        out 13h,al          ;先写低字节
        mov al,00h
        out 13h,al          ;后写高字节
        mov al,01           ;清通道屏蔽，启动DMA
        out 1ah,al
        mov ah,4ch
        int 21h
        code ends
        end start
```

七、参考程序3： DMA_I.asm

```
data segment
indata1 db 8 dup(30h),0dh,0ah,24h
data ends
extra segment at 0d400h
indata2 db 11 dup(?)
extra ends
code segment
    assume cs:code,ds:data,es:extra
```

```
start:
        mov ax,data
        mov ds,ax
        mov ax,extra
        mov es,ax
        lea si,indata1
        lea di,indata2
        cld
        mov cx,11
        rep movsb
        mov ax,extra
        mov ds,ax
        mov al,00
        out 1ch,al          ;清字节指针
        mov al,45h          ;写方式字
        out 1bh,al
        mov al,0dh          ;置地址页面寄存器
        out 83h,al
        mov al,00
        out 12h,al          ;写入基地址的低十六位
        mov al,40h
        out 12h,al
        mov ax,7            ;写入传送的8个字节数
        out 13h,al          ;先写低字节
        mov al,ah
        out 13h,al          ;后写高字节
        mov al,01           ;清通道屏蔽
        out 1ah,al          ;启动DMA
sss:    lea dx,indata2
lll:    mov ah,09
        int 21h
        mov ah,1
        int 16h
        je sss
exit:   mov ah,4ch
        int 21h
        code ends
        end start
```

实验十五　集成电路测试

一、实验目的

通过对四与非门(74LS00)集成电路芯片的测试,了解测试一般数字集成电路方法，进一步熟悉可编程并行接口8255 的使用。

二、实验内容

1、按图15-1连接电路，其中74LS00插在通用插座上。

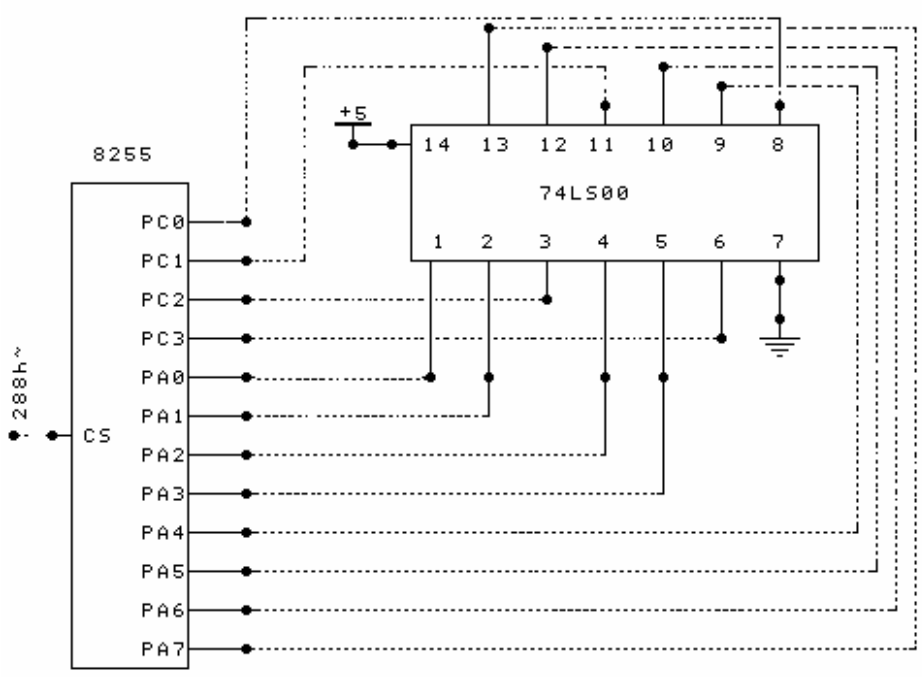


图15-1 集成电路测试电路图

2、编程提示:

将8255设置在方式0下工作，使A口输出，C口输入，先后通过A口分别给每一个门电路送入四个测试信号(00、01、10、11)，相应从C 口读出每一个门电路的输出结果，与正常值(1、1、1、0)进行比较，若相等，则此芯片好，否则为坏。

三、参考流程图　（见图15-2）

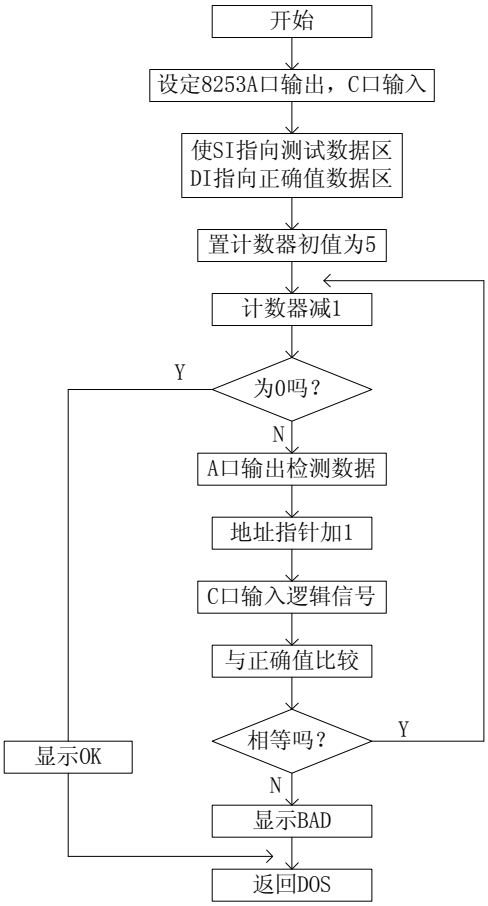


图15-2 集成电路测试流程图

四、参考程序： JC.ASM

```
data    segment
io8255a     equ 288h
io8255b     equ 28ah
io8255c     equ 28bh
se          db  00000000b       ;检测时发送的数据
            db  01010101b
            db  10101010b
            db  11111111b
ac0         db  00001111b       ;74LS00正确时检测时接收的数据
            db  00001111b
            db  00001111b
            db  00000000b
outbuf      db  'THE CHIP IS OK',07h,0ah,0dh,'$'
news        db  'THE CHIP IS BAD',07h,0ah,0dh,'$'
data ends
```

```
code segment
assume cs:code,ds:code
start:
        mov ax,data
        mov ds,ax
        mov dx,io8255c          ;对8255进行初始化编程
        mov al,89h              ;使A口输出,C口输入
        out dx,al
        mov di,offset ac0       ;DI中存放接收数据的缓冲区首址
        mov si,offset se        ;SI中存放发收数据的缓冲区首址
        mov cx,05h              ;发送四个字节
again:  dec cx
        jz exit                 ;如果四个数值都相等,则显示提示信息
        mov dx,io8255a
        mov al,[si]
        mov bl,[di]
        out dx,al               ;发送数据
        inc si
        inc di
        mov dx,io8255b
        in al,dx                ;读芯片的逻辑输出
        and al,0fh
        cmp al,bl
        je again                ;若正确就继续
error:  mov dx,offset news      ;若有错,芯片有问题
        mov ah,09h              ;显示错误的提示信息
        int 21h
        jmp ppp
exit:   mov dx,offset outbuf    ;显示正确的提示信息
        mov ah,09h
        int 21h
ppp:    mov ah,4ch              ;返回
        int 21h
code ends
end start
```

实验十六　电子琴

一、实验目的

1、通过8253产生不同的频率信号，使PC机成为简易电子琴。

2、了解利用8255和8253产生音乐的基本方法。

二、实验内容

实验电路如图16-1，8253的CLK0接1MHZ时钟，GATE0接8255的PA1，OUT0和8255的PA0接到与门的两个输入端，K8跳线连接喇叭，编程使计算机的数字键1、2、3、4、5、6、7作为电子琴按键，按下即发出相应的音阶。

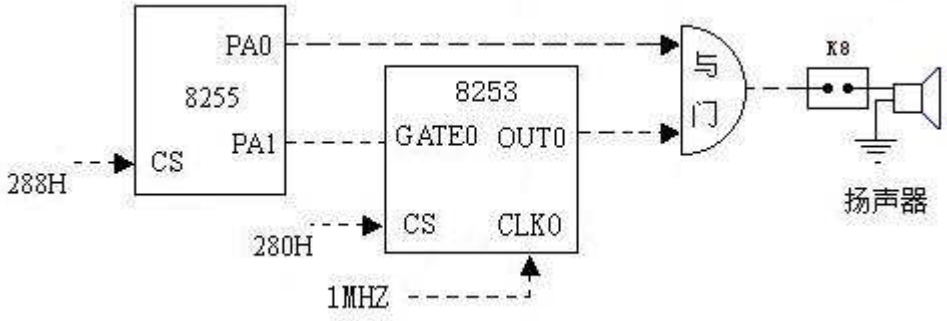


图16-1 电子琴电路

三、编程提示:

1、利用8255的PA0口来施加控制信号给与门，用来控制扬声器的开关状态。再利用设置不同的计数值，使8253产生不同频率的波形，使扬声器产生不同频率的音调，达到类似与音阶的高低音变换。对于音乐，每个音阶都有确定的频率。

各音阶标称频率值:

音 阶	1	2	3	4	5	6	7	$\dot{1}$
低频率(单位:Hz)	262	294	330	347	392	440	494	524
高频率(单位:Hz)	524	588	660	698	784	880	988	1048

四、参考流程图 （见图16-2）

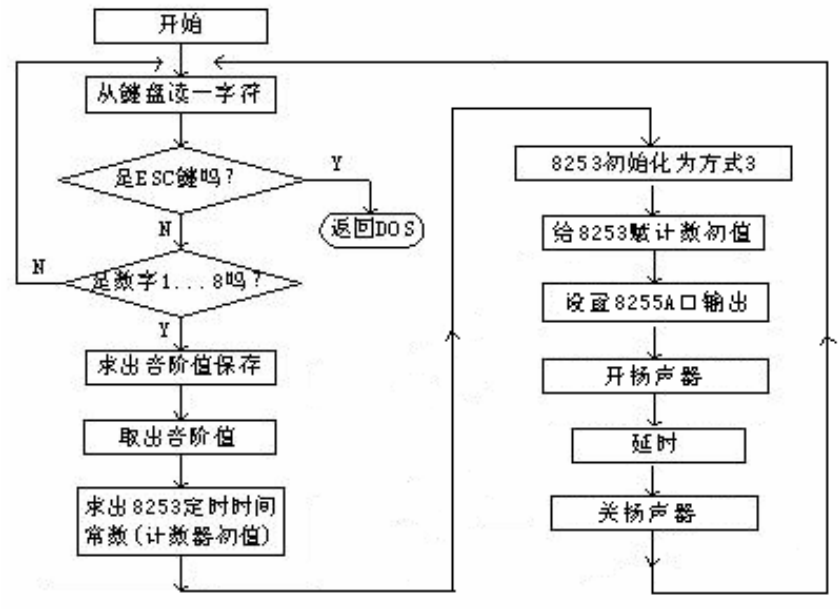


图16-2 主程序

五、参考程序： DZQ.ASM

```
data segment
        io8255a     equ 288h
        io8255b     equ 28bh
        io8253a     equ 280h
        io8253b     equ 283h
table dw 524,588,660,698,784,880,988,1048;高音的
;table dw 262,294,330,347,392,440,494,524;低音的
msg db 'Press 1,2,3,4,5,6,7,8,ESC:',0dh,0ah,'$'
data ends

code segment
assume cs:code,ds:data
start:
    mov ax,data
    mov ds,ax

    mov dx,offset msg
    mov ah,9
    int 21h                 ;显示提示信息
sing:
    mov ah,7
    int 21h                 ;从键盘接收字符,不回显
```

```
cmp al,1bh
je finish              ;若为ESC键,则转finish
cmp al,'1'
jl  sing
cmp al,'8'
jg sing                ;若不在'1'-'8'之间转sing

sub al,31h
shl al,1               ;转为查表偏移量
mov bl,al              ;保存偏移到bx
mov bh,0

mov ax,4240H           ;计数初值 = 1000000 / 频率, 保存到AX
mov dx,0FH
div word ptr[table+bx]
mov bx,ax

mov dx,io8253b            ;设置8253计时器0方式3, 先读写低字节, 再读写高字节
mov al,00110110B
out dx,al

mov dx,io8253a
mov ax,bx
out dx,al              ;写计数初值低字节

mov al,ah
out dx,al              ;写计数初值高字节

mov dx,io8255b            ;设置8255 A口输出
mov al,10000000B
out dx,al

mov dx,io8255a
mov al,03h
out dx,al              ;置PA1PA0 = 11(开扬声器)
call delay             ;延时
mov al,0h
out dx,al              ;置PA1PA0 = 00(关扬声器)
```

```
    jmp sing
finish:
    mov ax,4c00h
    int 21h

delay proc near          ;延时子程序
    push cx
    push ax
    mov ax,15
x1: mov cx,0ffffh
x2: dec cx
    jnz x2
    dec ax
    jnz x1
    pop ax
    pop cx
    ret
delay endp
code ends
    end start
```

实验十七　8250串行通讯实验

一、实验目的

1、进一步了解串行通信的基本原理。

2、掌握串行接口芯片8250的工作原理和编程方法。

二、实验内容

1、按图17-1连接线路，图中8250芯片插在通用插座上。

2、编程:从键盘输入一个字符，将其ASCII码加1后发送出去，再接收回来在屏幕上加1后的字符显示出来，实现自发自收。

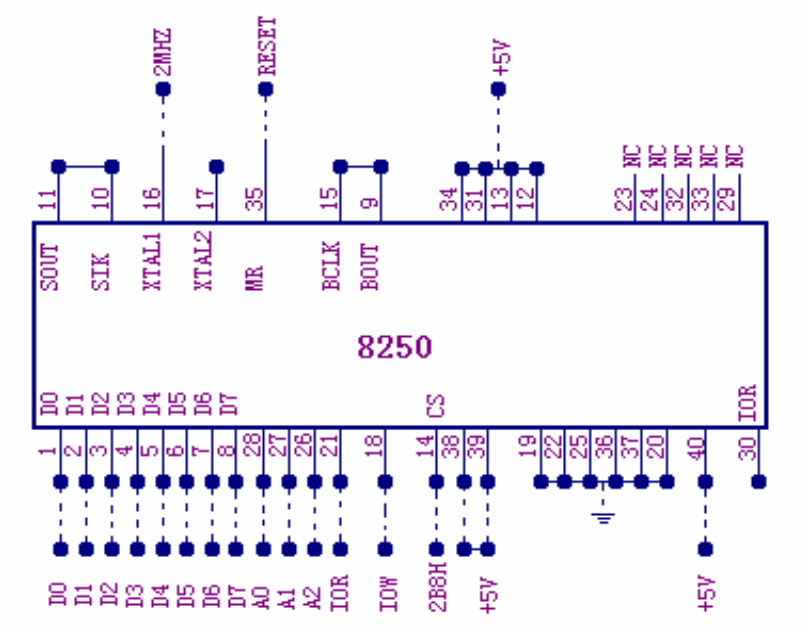


图17-1

三、实验提示

1、8250介绍：

INC8250是一个可编程序异步通讯单元芯片，在微机系统中起串行数据的输入输出接口作用。此外，它还包含有可编程序波特率发生器，它可用1～65535的因子对输入时钟进行分频，以产生波特率十六倍的输入输出时钟。

2、8250时钟接2MHZ，若选波特率为9600，波特率因子为16，则因子寄存器中分频数为13。所以因子寄存器低字节送13，高字节为00H。

3、图中CS接02B8H～02BFH:

下表为各寄存器选择地址一览表。表中DLAB为线控制寄存器的最高位，也叫因子寄存器存取位。当DLAB为0时选接收数据缓冲器，发送数据寄存器和中断允许寄存器。当DLAB为1时选因子寄存器的低字节和高字节。

DLAB	A2	A1	A0	选中寄存器
0	0	0	0	接收缓冲器（读）发送保持寄存器（写）
0	0	0	1	中断允许寄存器
X	0	1	0	中断标志寄存器（仅用于读）
X	0	1	1	线控制寄存器
X	1	0	0	MODEM控制寄存器
X	1	0	1	线状态寄存器
X	1	1	0	MODEM状态寄存器
X	1	1	1	无
1	0	0	0	因子寄存器（低字节）
1	0	0	1	因子寄存器（高字节）

4、收发采用查询方式。

四、参考流程

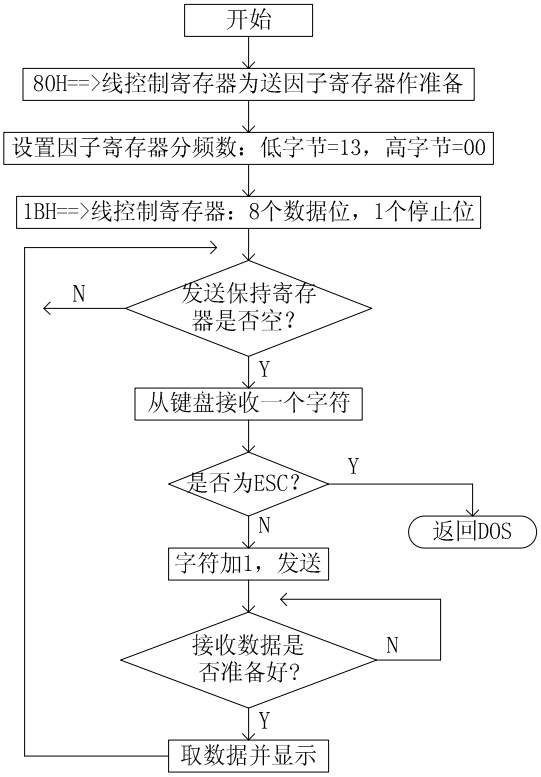


图17-2

五、参考程序：E8250.ASM

```
data segment
port    equ 2b8h
port1   equ 2b9h
port3   equ 2bbh
port5   equ 2bdh
mes     db  'you can play a key on the keyboard!',0ah,0dh
        db  'esc quit to dos!',0ah,0dh,'$'
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        mov al,80h
        mov dx,port3
        out dx,al
```

```
        mov al,13               ;set light divisor
        mov dx,port
        out dx,al
        mov al,00               ;set low divisor 9600 boud
        mov dx,port1
        out dx,al
        mov al,00011011b        ;8 bits 1 stop
        mov dx,port3
        out dx,al
        mov al,00h
        mov dx,port1
        out dx,al               ;interrupt enable all off
        mov dx,offset mes
        mov ah,09h
        int 21h
waiti:  mov dx,port5
        in  al,dx               ;get line status
        and al,20h
        test al,20h
        jz waiti
        mov ah,01
        int 21h
        cmp al,27
        jz  exit
        mov dx,port
        inc al
        out dx,al
        mov cx,10h
s50:    loop s50
next:   mov dx,port5
        in  al,dx
        and al,01
        test al,01
        jz  next
        mov dx,port
        in  al,dx
        mov dl,al
        mov ah,02
        int 21h
```

```
        jmp waiti
exit:   mov ah,4ch
        int 21h
code ends
end start
```

实验十八　步进电机控制实验

一、实验目的

1、了解步进电机控制的基本原理。

2、掌握控制步进电机转动的编程方法。

二、实验内容

1、按图18-1连接线路，利用8255输出脉冲序列，开关K0～K6控制步进电机转速，K7控制步进电机转向。8255 CS接288H～28FH。PA0～PA3接BA～BD；PC0～PC7接K0～K7。

2、编程:当K0～K6中某一开关为“1”(向上拨)时步进电机启动。K7向上拨电机正转，向下拨电机反转。

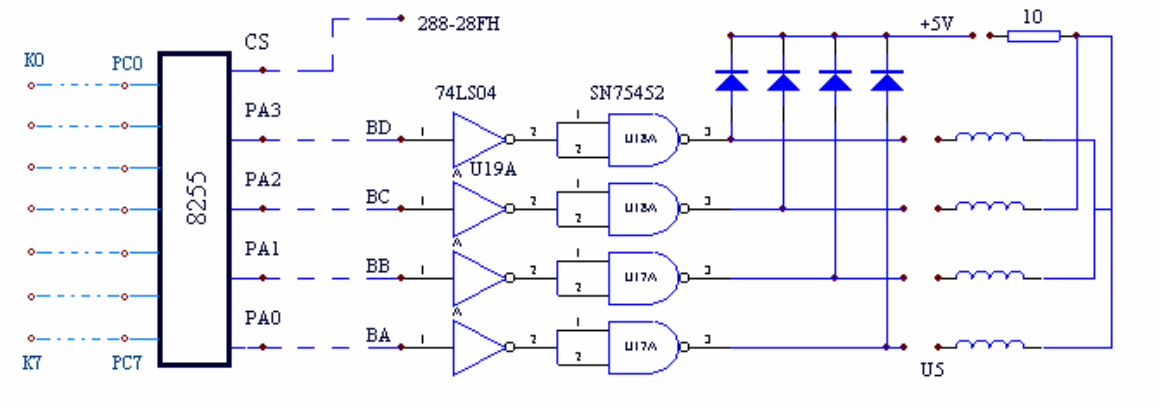


图18-1

三、实验说明

步进电机驱动原理是通过对每相线圈中的电流的顺序切换来使电机作步进式旋转。驱动电路由脉冲信号来控制，所以调节脉冲信号的频率便可改变步进电机的转速。

如图18-2所示：本实验使用的步进电机用直流＋5V电压，每相电流为0.16A，电机线圈由四相组成，即:Φ1(BA)； Φ2(BB)； Φ3(BC)； Φ4(BD)

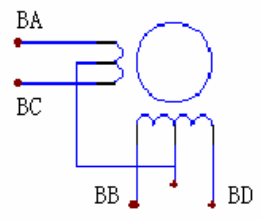


图18-2

驱动方式为二相激磁方式，各线圈通电顺序如下表。

顺序 \ 相	φ 1	φ 2	φ 3	φ 4
0	1	1	0	0
1	0	1	1	0
2	0	0	1	1
3	1	0	0	1

反时针方向回转

正时针方向回转

表中首先向φ1线圈－φ2线圈输入驱动电流，接着φ2－φ3，φ3－φ4，φ4－φ1，又返回到φ1－φ2，按这种顺序切换，电机轴按顺时针方向旋转。

实验可通过不同长度延时来得到不同频率的步进电机输入脉冲，从而得到多种步进速度。

四、参考流程图

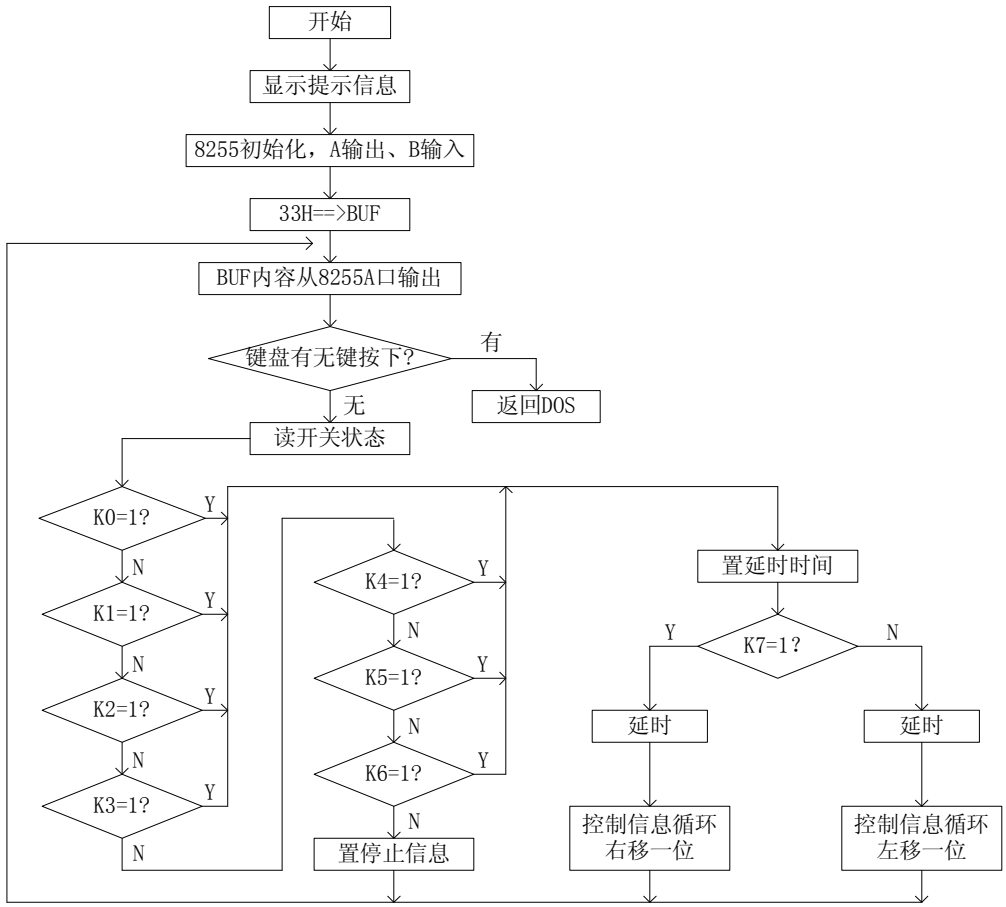


图18-3

五、参考程序：BJDJ.ASM

```
data segment
p55a    equ 288h    ;8255 a port output
p55c    equ 28ah    ;8255 c port input
p55ctl  equ 28bh    ;8255 coutrl port
```

```
buf     db  0
mes     db  'k0-k6 are speed contyol',0ah,0dh
        db  'k6 is the lowest speed ',0ah,0dh
        db  'k0 is the highest speed',0ah,0dh
        db  'k7 is the direction control',0ah,0dh,'$'
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,cs
        mov ds,ax
        mov ax,data
        mov ds,ax
        mov dx,offset mes
        mov ah,09
        int 21h
        mov dx,p55ctl
        mov al,8bh
        out dx,al                   ;8255 c input, a output
        mov buf,33h
out1:   mov al,buf
        mov dx,p55a
        out dx,al
        push dx
        mov ah,06h
        mov dl,0ffh
        int 21h                     ;any key pressed
        pop dx
        je  in1
        mov ah,4ch
        int 21h
in1:    mov dx,p55c
        in  al,dx                   ;input switch value
        test al,01h
        jnz k0
        test al,02h
        jnz k1
        test al,04h
        jnz k2
```

```
        test al,08h
        jnz k3
        test al,10h
        jnz k4
        test al,20h
        jnz k5
        test al,40h
        jnz k6
stop:   mov dx,p55a
        mov al,0ffh
        jmp out1
k0:     mov bl,10h
sam:    test al,80h
        jz zx0
        jmp nx0
k1:     mov bl,18h
        jmp sam
k2:     mov bl,20h
        jmp sam
k3:     mov bl,40h
        jmp sam
k4:     mov bl,80h
        jmp sam
k5:     mov bl,0c0h
        jmp sam
k6:     mov bl,0ffh
        jmp sam
zx0:    call delay
        mov al,buf
        ror al,1
        mov buf,al
        jmp out1
nx0:    call delay
        mov al,buf
        rol al,1
        mov buf,al
        jmp out1
delay proc near
delay1: mov cx,05a4h
```

```
delay2: loop delay2
        dec bl
        jnz delay1
        ret
delay endp
code ends
end start
```

实验十九　小直流电机转速控制实验

一、 实验目的

1、进一步了解DAC0832的性能及编程方法。

2、了解直流电机控制的基本方法。

二、 实验内容

1、按图19-1线路接线。DAC0832的CS接290H～297H，Ub接DJ插孔，8255 CS接288H～28FH。

2、编程利用DAC0832输出一串脉冲，经放大后驱动小直流电机，利用开关K0～K5控制改变输出脉冲的电平及持续时间，达到使电机加速，减速之目的。

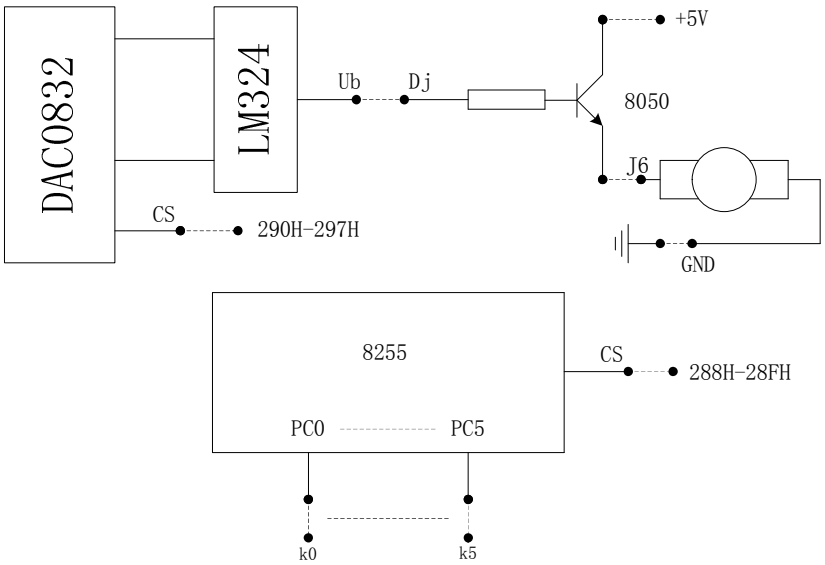


图19-1

三、实验原理简述

小直流电机的转速是由Ub输出脉冲的占空比来决定的，正向占空比越大转速越快，反之越慢。见下图19-2例:

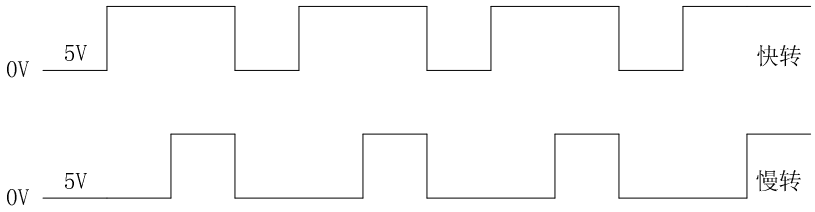


图19-2

在本实验中，模拟量输出Ub为双极性，当输入数字量小于80H时输出为负，输入等于80H时为0V，输入大于80H时输出为正。因而本实验中，DAC0832输入数字量只有2个(80H和FFH)，通过不同的延迟时间达到改变小电机转速的目的。

四、参考程序框图

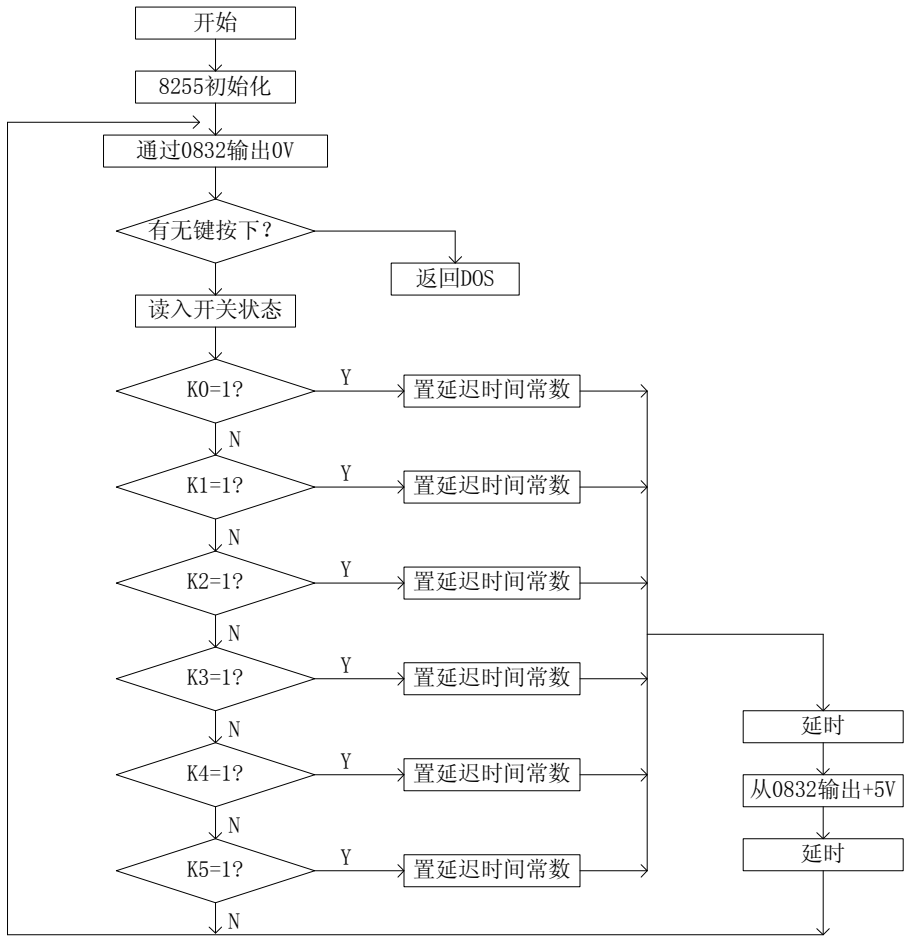


图19-3

五、参考程序： ZLDJ.ASM

```
data segment
port1       equ 290h
port2       equ 28bh
port3       equ 28ah
buf1        dw  ?
buf2        dw  ?
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        mov dx,port2
        mov al,8bh
        out dx,al               ;8255 port c input
lll:    mov al,80h
        mov dx,port1
        out dx,al               ;d/a output 0v
        push dx
        mov ah,06h
        mov dl,0ffh
        int 21h
        pop dx
        je intk                 ;not any key jmp intk
        mov ah,4ch
        int 21h                 ;exit to dos
intk:   mov dx,port3
        in al,dx                ;read switch
        test al,01h
        jnz k0
        test al,02h
        jnz k1
        test al,04h
        jnz k2
        test al,08h
        jnz k3
        test al,10h
        jnz k4
```

```
        test al,20h
        jnz k5
        jmp lll
k0:     mov buf1,0400h
        mov buf2,0330h
delay:  mov cx,buf1
delay1: loop delay1
        mov al,0ffh
        mov dx,port1
        out dx,al
        mov cx,buf2
delay2: loop delay2
        jmp lll
k1:     mov buf1,0400h
        mov buf2,0400h
        jmp delay
k2:     mov buf1,0400h
        mov buf2,0500h
        jmp delay
k3:     mov buf1,0400h
        mov buf2,0600h
        jmp delay
k4:     mov buf1,0400h
        mov buf2,0700h
        jmp delay
k5:     mov buf1,0400h
        mov buf2,0800h
        jmp delay
code ends
end start
```

实验二十　　键盘显示控制器实验

一、实验目的

1、掌握8279键盘显示电路的基本功能及编程方法。

2、掌握一般键盘和显示电路的工作原理。

3、进一步掌握定时器的使用和中断处理程序的编程方法。

二、实验内容

1、本实验的实验电路如图20-1，它做在一块扩展电路板上，用一根20芯扁平电缆与实验台上扩展插头J7相连。

2、编程1:使得在小键盘上每按一个健，6位数码管上显示出相应字符，它们的对应关系如下:

小键盘		显示	小键盘		显示
0	—	0	C	—	C
1	—	1	D	—	d
2	—	2	E	—	E
3	—	3	F	—	F
4	—	4	G	—	q
5	—	5	M	—	冂
6	—	6	P	—	p
7	—	7	W	—	凵
8	—	8	X	—	刂
9	—	9	Y	—	乛
A	—	冂	R	—	返回
B	—	b			

3、编程2:中断编程

利用实验台上提供的定时器8253和扩展板上提供的8279以及键盘和数码显示电路，设计一个电子钟。由8253中断定时，小键盘控制电子钟的启停及初始值的预置。

电子钟显示格式如下：

XX. XX. XX.　　由左向右分别为时、分、秒

要求具有如下功能：

①、C键:清除，显示全零。

②、G键:启动，电子钟计时。

③、D键:停止，电子钟停止计时。

④、P键:设置时、分、秒值。输入时依次为时、分、秒，同时应有判断输入错误的能力，若输入有错，则显示:E－－－－－－。此时敲P键可重新输入预置值。

⑤、E键:程序退出。

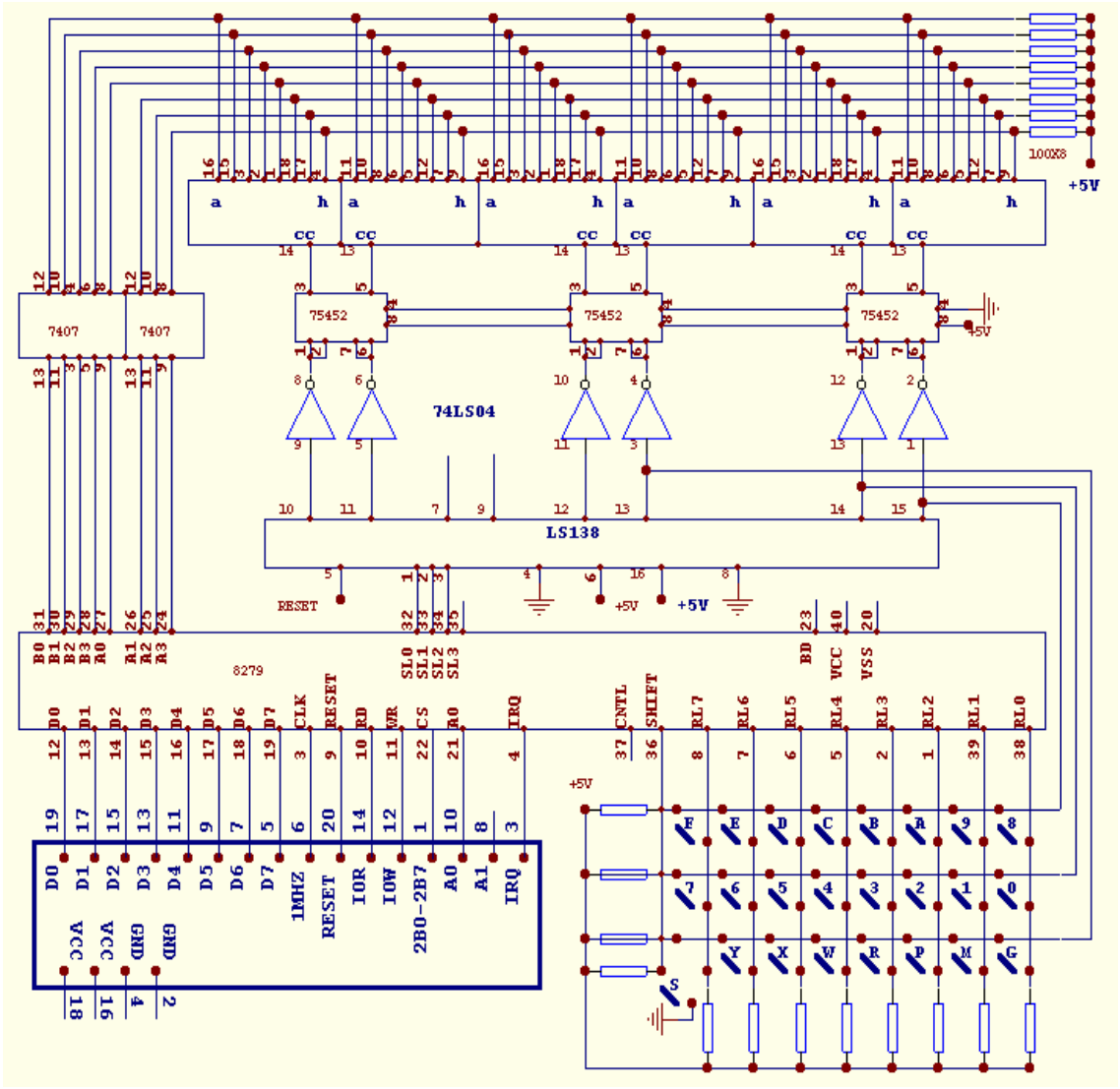


图20-1

三、编程1 接线方法

用一根20芯扁平电缆将实验扩展板与实验台上扩展插头J7相连。

四、编程2 接线方法

1、用一根20芯扁平电缆将实验扩展板与实验台上扩展插头J7相连。

2、实验台上8253 CLK0 接 1MHZ，GATE0 和 GATE1接＋5V，OUT0 接 CLK1，OUT1 接 IRQ，CS接 280H～287H。

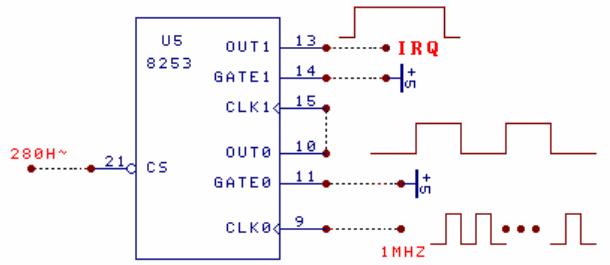
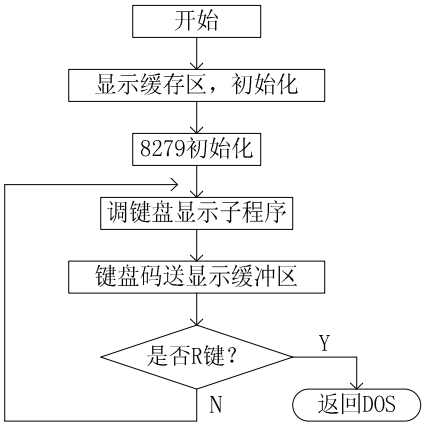
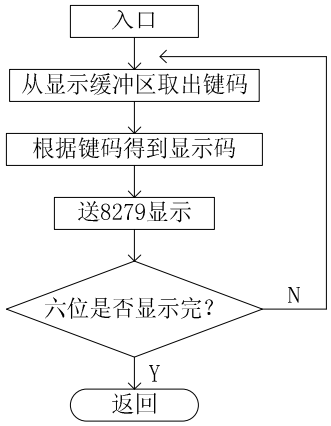


图20-2

五、编程1 参考流程



主程序流程图:



显示子程序流程图:DISP

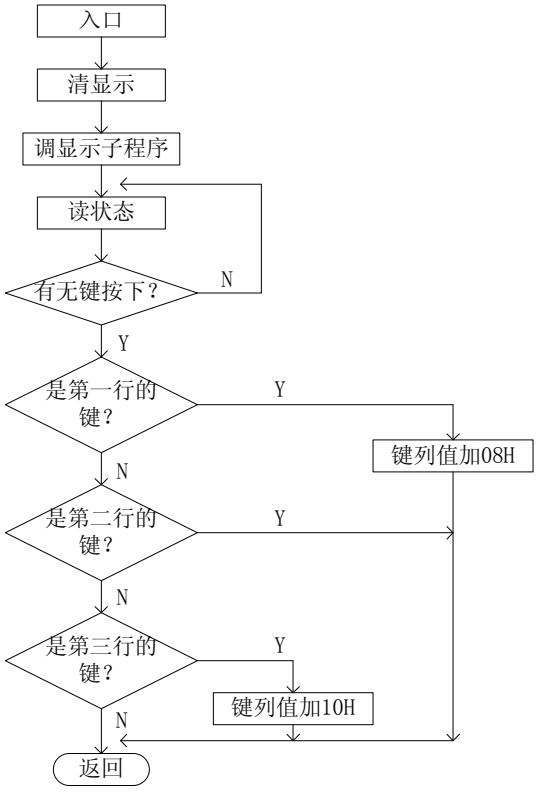


图20-3 键盘显示子程序流程图:KEY2

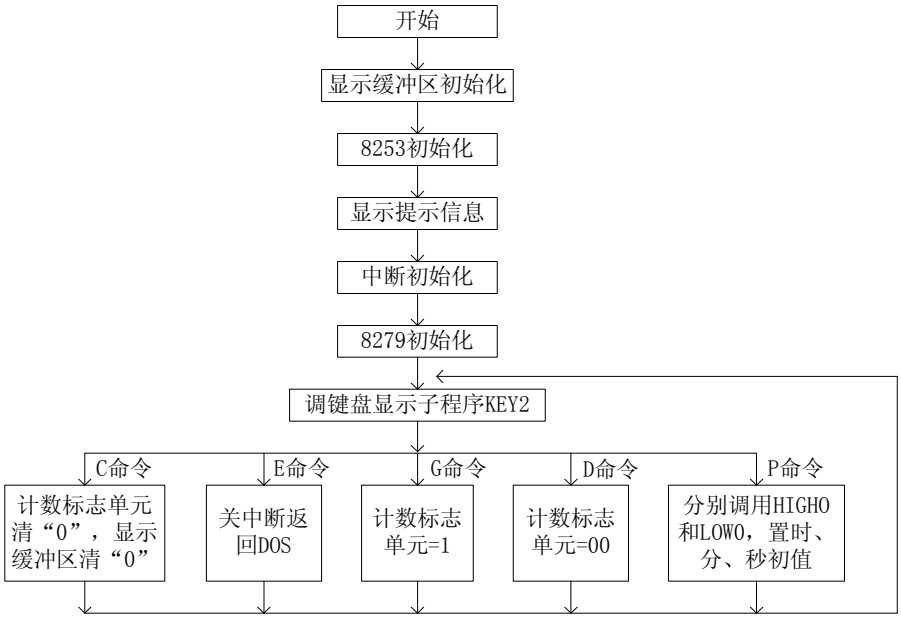
六、编程1 参考程序: JPXSH1.ASM

```
port0   equ 2b0h   ;8279 data port
port1   equ 2b1h   ;8279 ctrl port
data segment
sec1    db  0      ;hour hight
sec2    db  0      ;houp low
min1    db  0      ;min  hight
min2    db  0      ;min low
hour1   db  0      ;sec hight
hour2   db  0      ;sec low
led     db  3fh,06,5bh,4fh,66h,6dh,7dh,07,7fh,6fh,77h,7ch,39h,5eh,79h
        db  71h,67h,37h,73h,31h,3eh,36h,66h
data ends
code segment
main proc far
assume cs:code,ds:data
start:
        cli
        mov ax,data
        mov ds,ax
        mov sec1,0
        mov sec2,0
        mov min1,0
        mov min2,0
        mov hour1,0
        mov hour2,0
        mov dx,port1
        mov al,0d3h
        out dx,al                   ;8279 clear
        mov al,2ah
        out dx,al                   ;8279 clock
        mov al,40h
        out dx,al                   ;read fifo ram command
        mov al,00h
        out dx,al                   ;keybord disply mode
        mov al,80h
        out dx,al                   ;write ram command
key1:   call key2                   ;call keybord and disply
next2:  mov hour2,al
```

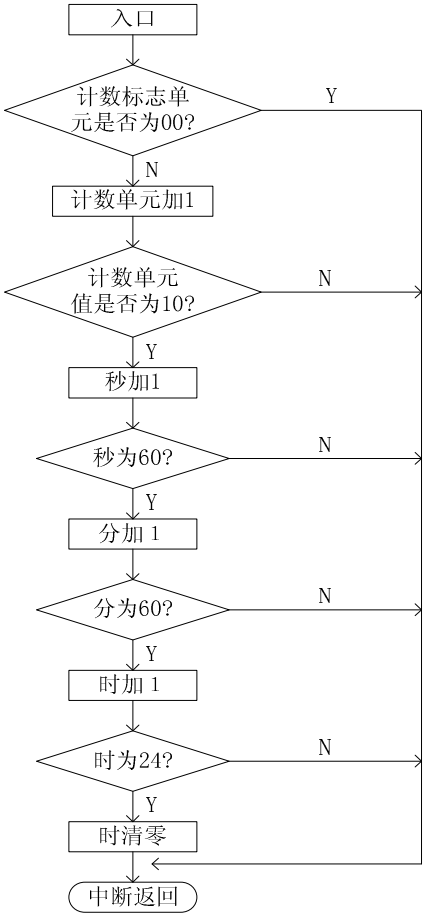
```
        mov hour1,al
        mov min2,al
        mov min1,al
        mov sec2,al
        mov sec1,al
        push ax
        mov ah,1
        int 16h
        jne toexit
lp0:    pop ax
        cmp al,13h                  ;'r' command
        jnz lp1
toexit: mov ax,4c00h                ;quit to dos
        int 21h
lp1:    jmp key1
main endp
key2 proc near
        mov dx,port1
        mov al,0d1h
        out dx,al                   ;clear display
wrep:   call disp
        mov dx,port1
        in al,dx
        and al,07h
        jz wrep
keyn:   mov dx,port0
        in al,dx
        mov bl,al
        and al,07h
        and bl,38h
        mov cl,03
        shr bl,cl
        cmp bl,00h
        jnz line1
        add al,08h
        jmp quit1
line1:  cmp bl,01h
        jnz line2
        jmp quit1
```

```
line2:  add al,10h
quit1:  ret
key2 endp
disp proc near
        push cx
        mov ax,data
        mov ds,ax
        mov dx,port1
        mov al,90h
        out dx,al
        mov si,offset sec1
        mov cx,0006
        mov bx,offset led
disp1:  cld
        lodsb
        xlat
        mov dx,port0
        out dx,al
        loop disp1
        pop cx
        ret
disp endp
code ends
end start
```

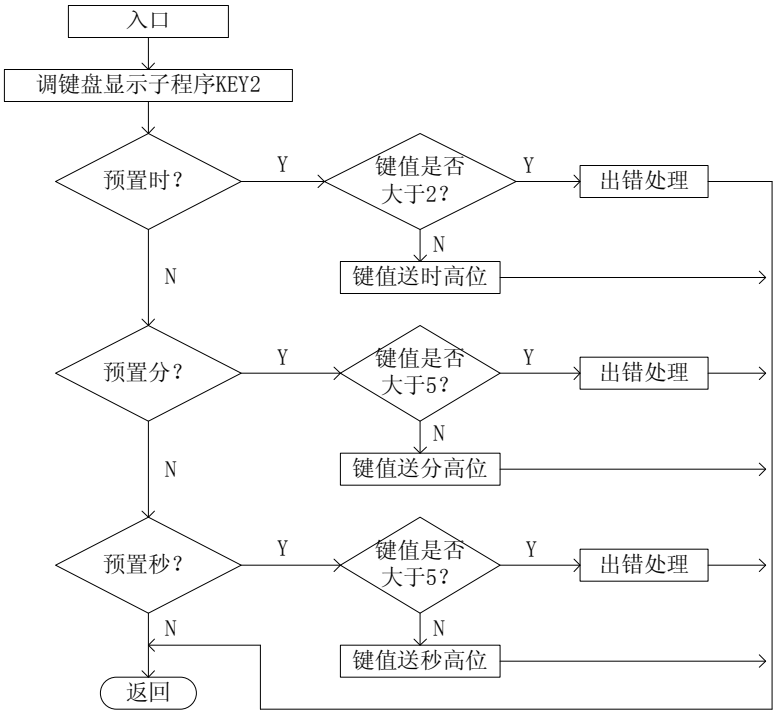
七、编程2 参考流程



主程序流程图：



中断处理子程序：



预置时、分、秒高位子程序HIGH0:

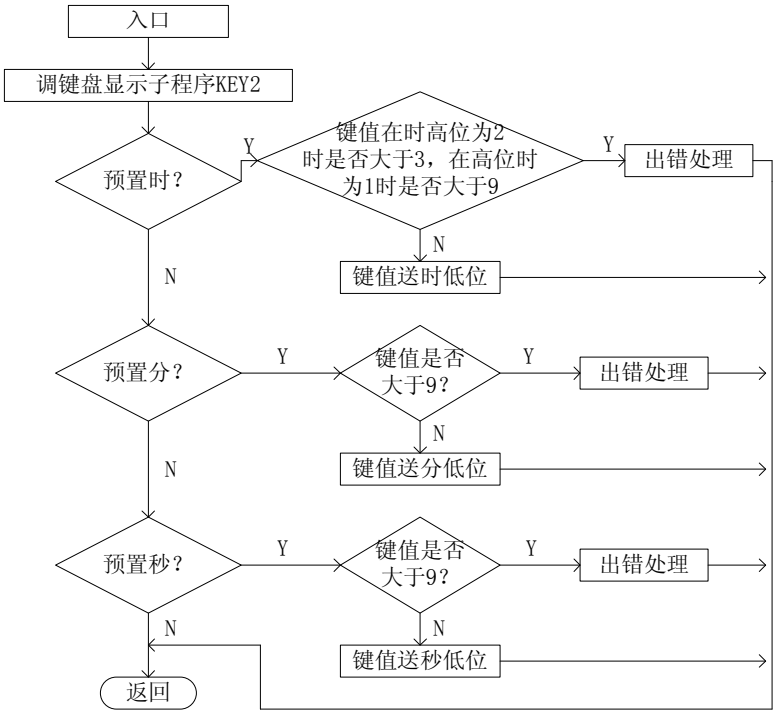


图20-4 预置时、分、秒低位子程序LOW0:

八、编程2 参考程序： JPXSH.ASM

```
inta00  equ 20h      ;8259a port，口地址
inta01  equ 21h      ;8259a port，口地址
```

```
port0   equ 2b0h    ;8279 data port, 8279数据口
port1   equ 2b1h    ;8279 ctrl port, 8279控制口
time0   equ 280h    ;8253 time0 port, 8253定时器0口地址
time1   equ 281h    ;8253 time1 port, 8253定时器1口地址
timec   equ 283h    ;8253 ctrl port, 8253控制口地址
stacks segment stack
sta dw 512 dup(?)
top equ length sta
stacks ends
data segment
csreg   dw  ?
ipreg   dw  ?
irq_times   dw  00h
buf     db  0       ;count, 计数单元
sign    db  0       ;flage, 计数标志
sec1    db  0       ;hour hight, 秒高位
sec2    db  0       ;houp low, 秒低位
min1    db  0       ;min  hight, 分高位
min2    db  0       ;min low, 分低位
hour1   db  0       ;sec hight, 时高位
hour2   db  0       ;sec low, 时低位
err1    db  0       ;error flage, 出错标志
hms     db  0       ;00 is hour,11 is min,22 is sec, 预置时、分、秒标志
led     db  3fh,06,5bh,4fh,66h,6dh,7dh,07,7fh,6fh,79h,40h
mes     db  'pleas first create the irq pulse!',0ah,0dh,0ah,0dh
        db  'in small keybord:',0ah,0dh
        db  'c--clear to zero;  g--go ahead',0ah,0dh
        db  'd--stop the disply;e--exit',0ah,0dh
        db  'p--position the beginning time',0ah,0dh,'$'
data ends
code segment
main proc far
assume  cs:code,ds:data,ss:stacks,es:data
start:  cli
        mov ax,data
        mov ds,ax
        mov buf,0
        mov sign,01
        mov sec1,0
```

```
mov sec2,0
mov min1,0
mov min2,0
mov hour1,0
mov hour2,0
mov err1,0
mov dx,timec          ;8253初始化
mov al,36h
out dx,al
mov dx,time0
mov ax,1000
out dx,al
mov al,ah
out dx,al
mov dx,timec
mov al, 74h
out dx,al
mov ax,100
mov dx,time1          ;定时器1每0.1秒中断一次
out dx,al
mov al,ah
out dx,al
mov ax,stacks
mov ss,ax
mov sp,top
mov ax,data
mov ds,ax
mov es,ax
mov dx,offset mes
mov ah,09
int 21h

mov ax,cs
mov ds,ax
mov dx,offset int_proc
mov ax,250bh
int 21h
in al,21h
and al,0f7h
```

```
        out 21h,al

        mov dx,port1
        mov al,0d3h
        out dx,al           ;8279 clear, 清零
        mov al,2ah
        out dx,al           ;8279 clock, 置时钟命令
        mov al,40h
        out dx,al           ;read fifo ram command,fifo ram命令
        mov al,00h
        out dx,al           ;keybord disply mode, 置键盘显示模式
        mov al,80h
        out dx,al           ;write ram command, 写ram命令
        sti
key1:   call key2           ;call keybord and disply, 调键盘显示子命令
        cmp hour2,0ah                       ;err flage
        jz next2
next1:  cmp al,0ch          ;'c' command,'c'命令否
        jnz lp0
next2:  mov sign,00h
        mov hour2,00h
        mov hour1,00h
        mov min2,00h
        mov min1,00h
        mov sec2,00h
        mov sec1,00h
lp0:    cmp al,0eh          ;'e' command,'e'命令退出程序
        jnz lp1
        mov sign,00h
        jmp exit
lp1:    cmp al,10h          ;'g' command,'g'命令否
        jnz lp2
        mov sign,01h
        jmp key1
lp2:    cmp al,0dh          ;'d' command,'d'命令否
        jnz seti
        mov sign,00h
key3:   jmp key1
seti:   cmp al,12h          ;'p' command,'p'命令否
```

```
        jnz key1
        mov sign,00h          ;add 1 flage,'00'为预置时标志
        mov hms,00h           ;hour flage
        call high0
        cmp err1,01h
        jz key3
        call low0
        cmp err1,01h
        jz key3
        mov hms,11h           ;min flage,'11'为预置分标志
        call high0
        cmp err1,01h
        jz key3
        call low0
        cmp err1,01h
        jz key3
        mov hms,22h           ;sec flage,'22'为预置秒标志
        call high0
        cmp err1,01h
        jz key3
        call low0
        jmp key1
exit:   in al,21h             ;关中断IRQ3
        or al,08h
        out 21h,al
        sti
        mov ax,4c00h
        int 21h
main endp
int_proc proc far
        cli
        push ax
        push bx
        push cx
        push dx
        push si
        push di
        push ds
        cmp sign,00           ;sign is add 1 flage，是否允许计数
```

```
        jz endt1
        inc buf             ;buf is count, 计数单元加1
        cmp buf,10
        jl endt
        mov buf,0
        inc sec1
        cmp sec1,10
        jl endt
        mov sec1,0
        inc sec2
        cmp sec2,6
        jl endt
        mov sec2,0
        inc min1
        cmp min1,10
        jl endt
        mov min1,0
        inc min2
        cmp min2,6
        jl endt
        mov min2,0
        inc hour1
        cmp hour2,2
        jl hh
        cmp hour1,4
        jl endt
        mov hour1,0
        mov hour2,0
endt1:  jmp endt
hh:     cmp hour1,10
        jl endt
        mov hour1,0
        inc hour2
endt:   mov al,20h          ;send EOI
        mov dx,inta00
        out dx,al
        mov cx,0ffffh
loopx:  nop
        loop loopx          ;延时
```

```
        pop ds
        pop di
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
        mov al,20h
        out 20h,al
        iret
int_proc endp
key2 proc near
        mov dx,port1
        mov al,0d1h
        out dx,al           ;clear display, 清显示
wrep:   call disp           ;调显示子程序
        mov dx,port1
        in al,dx
        and al,07h
        jz wrep
keyn:   mov dx,port0        ;读状态
        in al,dx
        mov bl,al
        and al,07h
        and bl,38h
        mov cl,03
        shr bl,cl
        cmp bl,00h          ;是否第一行键
        jnz line1
        add al,08h
        jmp quit1
line1:  cmp bl,01h
        jnz line2           ;是否第二行键
        jmp quit1
line2:  add al,10h
quit1:  ret
key2 endp
disp proc near
        push cx
```

```
        mov ax,data
        mov ds,ax
        mov dx,port1
        mov al,90h
        out dx,al
        mov si,offset sec1
        mov cx,0006
        mov bx,offset led
disp1:  cld
        lodsb
        xlat
        mov dx,port0
        out dx,al
        loop disp1
        pop cx
        ret
disp endp
errs proc near
        mov hour2,0ah
        mov hour1,0bh       ;error
        mov min2,0bh        ;disply 'E-----'显示出错标志
        mov min1,0bh
        mov sec2,0bh
        mov sec1,0bh
        mov err1,01h        ;err flage, 标记出错
        ret
errs endp
high0 proc near
        call key2
        mov err1,00
        cmp hms,00h         ;hms is hour min sc flage, 预置时、分、秒
        jnz min0
        cmp al,02h          ;00 is hour, 预置时
        jg error            ;11 is min, 预置分
        mov hour2,al        ;22 is sec, 预置秒
        jmp hqut1
min0:   cmp hms,11h
        jnz sec0
        cmp al,05h
```

```
        jg error
        mov min2,al
        jmp hqut1
sec0:   cmp al,05h
        jg error
        mov sec2,al
hqut1:  ret
error:  call errs
        ret
high0 endp
low0 proc near
        call key2           ;get hour min sec low, 预置时、分、秒低位
        mov err1,00
        cmp hms,00h
        jnz min3
        mov dl,hour2
        cmp dl,01
        jg hour3
        cmp al,09h
        jg error
        mov hour1,al
        jmp lqut1
hour3:  cmp al,03h
        jg error
        mov hour1,al
        jmp lqut1
min3:   cmp hms,11h
        jnz sec3
        cmp al,09h
        jg error
        mov min1,al
        jmp lqut1
sec3:   cmp al,09h
        jg error
        mov sec1,al
lqut1:  ret
low0 endp
code ends
end start
```

实验二十一　　存储器读写实验

一、实验目的

1、熟悉6116静态RAM的使用方法，掌握PC机外存扩充的手段。

2、通过对硬件电路的分析，学习了解总线的工作时序。

二、实验内容

1、　　硬件电路如下:

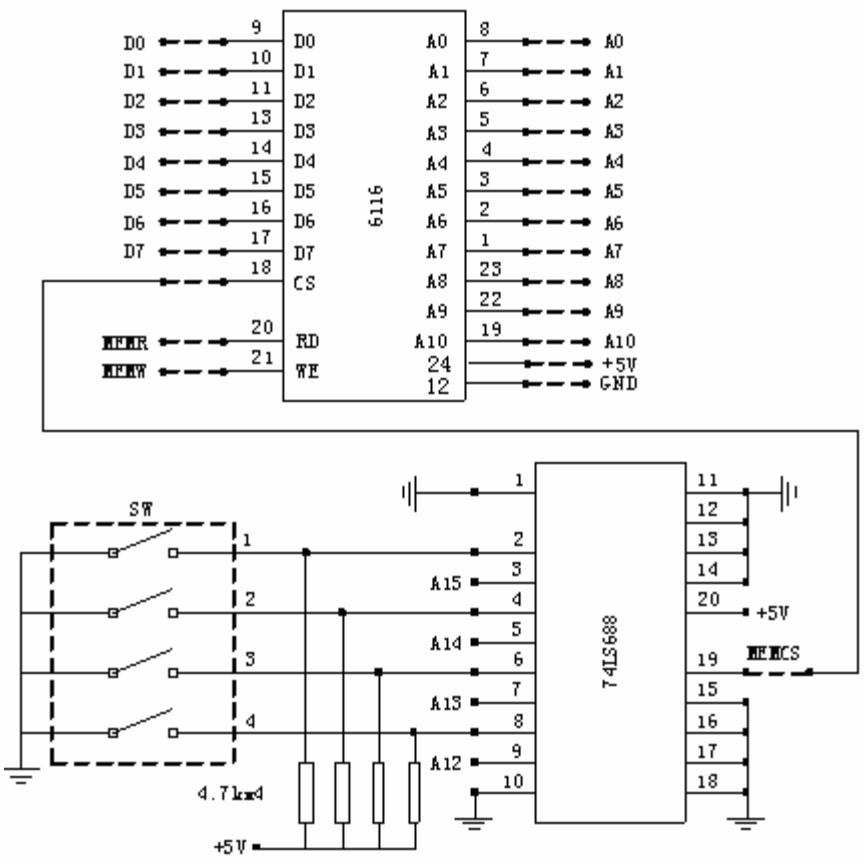


图21-1

2、编制程序，将字符A-Z循环写入扩展的6116RAM中，然后再将6116的内容读出来显示在主机屏幕上。

三、编程提示

1、注意:TPC-USB已为扩展的6116指定了段地址:0d000H。

2、TPC-USB模块外扩储器的地址范围为0D4000H-0D7fffH。

3、通过片选信号的产生方式，确定6116RAM在PC机系统中的地址范围。因为段地址已指定，所以其地址为CS=A15 and A14 and A13 and A12，实验台上设有地址选择微动开关K2，

拨动开关，可以选择4000-7fff的地址范围。编制程序，从0d6000H开始循环写入100h个A-Z。

开关状态如下:

1	2	3	4	地址
ON	OFF	ON	OFF	d4000h
ON	OFF	OFF	ON	d6000h

四、程序框图

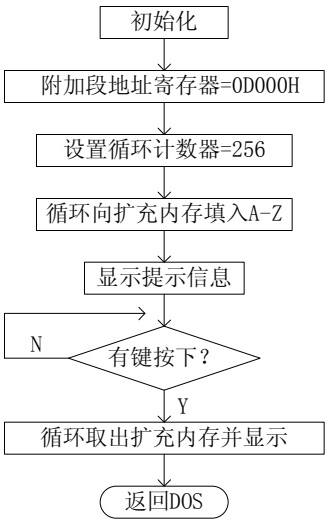


图21-2

五、程序清单: MEMRWEX.ASM

```
data segment
message db 'please enter a key to show the contents!',0dh,0ah,'$'
data ends
code segment
assume cs:code,ds:data,es:data
start:
        mov ax,data
        mov ds,ax
        mov ax,0d000h
        mov es,ax
        mov bx,06000h
        mov cx,100h
        mov dx,40h
rep1:   inc dl
        mov es:[bx],dl
        inc bx
        cmp dl,5ah
        jnz ss1
        mov dl,40h
```

```
ss1:    loop rep1
        mov dx,offset message
        mov ah,09
        int 21h
        mov ah,01h
        int 21h
        mov ax,0d000h
        mov es,ax
        mov bx,06000h
        mov cx,0100h
rep2:   mov dl,es:[bx]
        mov ah,02h
        int 21h
        inc bx
        loop rep2
        mov ax,4c00h
        int 21h
    code ends
    end start
```

实验二十二　双色点阵发光二极管显示实验

一、实验目的

1、了解双色点阵LED显示器的基本原理。

2、掌握PC机控制双色点阵LED显示程序的设计方法。

二、实验原理

点阵LED显示器是将许多LED类似矩阵一样排列在一起组成的显示器件，双色点阵LED是在每一个点阵的位置上有红绿或红黄或红白两种不同颜色的发光二极管。当微机输出的控制信号使得点阵中有些LED发光，有些不发光，即可显示出特定的信息，包括汉字、图形等。车站广场由微机控制的点阵LED大屏幕广告宣传牌随处可见。

实验仪上设有一个共阳极8×8点阵的红黄两色LED显示器，其点阵结构如图所示。该点阵

对外引出24条线，其中8条行线，8条红色列线，8条黄色列线。若使某一种颜色、某一个LED发光，只要将与其相连的行线加高电平，列线加低电平即可。

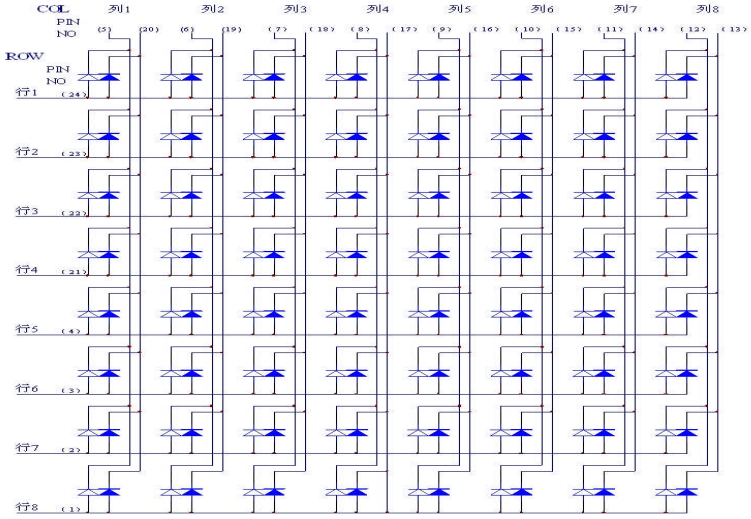


图22-1

例如欲显示汉字“年”，采用逐列循环发光。首先由“年”的点阵轮廓，确定点阵代码(如图所示)根据“年”的点阵代码，确定逐列循环发光的顺序如下:

① 行代码输出 44H； 红色列代码输 01H；第一列2个红色LED发光。

② 行代码输出 54H； 红色列代码输 02H；第二列3个红色LED发光。

③ 行代码输出 54H； 红色列代码输 04H；第三列3个红色LED发光。

④ 行代码输出 7FH； 红色列代码输 08H；第四列7个红色LED发光。

⑤ 行代码输出 54H； 红色列代码输 10H；第五列3个红色LED发光。

⑥ 行代码输出 DCH； 红色列代码输 20H；第六列5个红色LED发光。

⑦ 行代码输出 44H； 红色列代码输 40H；第七列2个红色LED发光。

⑧ 行代码输出 24H； 红色列代码输 80H；第八列2个红色LED发光。

在步骤①～⑧之间可插入几ms的延时，重复进行①～⑧即可在LED上稳定的显示出红色“年”字。若想显示黄色“年”，只需把红色列码改为黄色列码即可。

三、实验内容:

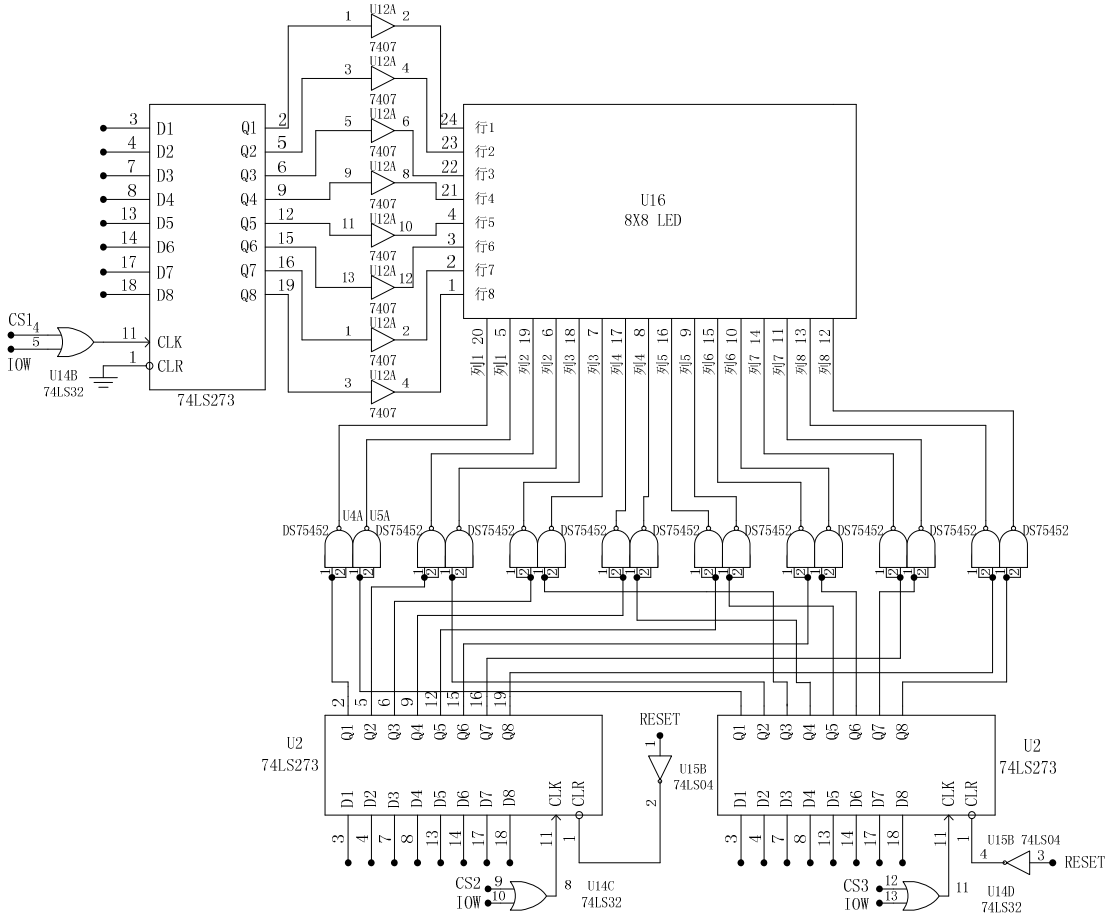


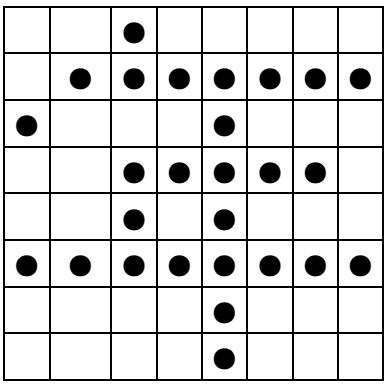
图22-2

1、实验仪上的点阵LED及驱动电路如下图所示，行代码、红色列代码、黄色列代码各用一片74LS273锁存。行代码输出的数据通过行驱动器7407加至点阵的8条行线上，红和黄列代码的输出数据通过驱动器DS75452反相后分别加至红和黄的列线上。行锁存器片选信号为CS1，红色列锁存器片选信号为CS2，黄色列锁存器片选信号为CS3。

2、接线方法:行片选信号 CS1 接 280H；红列片选信号 CS2 接 288H；黄列片选信号 CS3 接 290H。

3、编程重复使LED点阵红色逐列点亮，再黄色逐列点亮，再红色逐行点亮，黄色逐行点亮。

4、编程在LED上重复显示红色“年”和黄色“年”。



四、逐行、逐列显示参考流程图

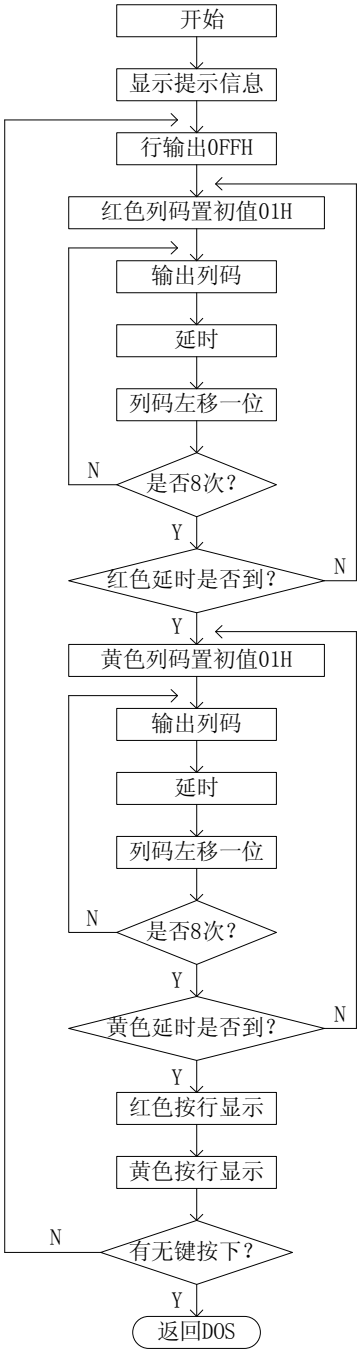


图22-3

五、逐行逐列显示参考程序 11588.asm

```
proth   equ 280h
protlr  equ 288h
protly  equ 290h
data segment
```

```
mess    db  'strike any key,return to dos!',0ah,0dh,'$'
count   db  07
count1  dw  0000
buff    db  24h,22h,3bh,2ah,0feh,2ah,2ah,22h
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        mov dx,offset mess
        mov ah,09
        int 21h                         ;显示提示信息
agn:    mov al,0ffh
        mov dx,proth
        out dx,al
        mov ah,01
        mov cx,0008h
agn1:   shl ah,01
        mov dx,protlr
        mov al,ah
        out dx,al                       ;红行亮
        push cx
        mov cx,0030h
d5:     call delay1
        loop d5
        pop cx
        loop agn1
        mov ah,01
        int 16h
        jnz a2
        mov ah,01
        mov cx,0008h
agn2:   mov dx,protly
        mov al,ah
        out dx,al
        push cx
        mov cx,0030h                    ;黄行亮
d4:     call delay1
```

```
        loop d4
        pop cx
        shl ah,01
        loop agn2
        mov ah,01
        int 16h
        jnz a2
        mov al,0ffh
        mov dx,protlr
        out dx,al                    ;红列亮
        mov ah,01
        mov cx,0008h
agn3:   mov dx,proth
        mov al,ah
        out dx,al
        push cx
        mov cx,0030h
d2:     call delay1
        loop d2
        pop cx
        shl ah,01
        loop agn3
        mov al,00h
        mov dx,protlr
        out dx,al
        mov ah,01
        int 16h
        jnz a2
        mov al,0ffh
        mov dx,protly
        out dx,al                    ;黄列亮
        mov ah,01
        mov cx,0008h
agn4:   mov dx,proth
        mov al,ah
        out dx,al
        push cx
        mov cx,0030h
d1:     call delay1
```

```
        loop d1
        pop cx
        shl ah,01
        loop agn4
        mov ah,01
        int 16h
        jnz a2
        jmp agn
delay1 proc near                    ;延迟子程序
        push cx
        mov cx,4000h
ccc:    loop ccc
        pop cx
        ret
delay1 endp
a2:     mov ah,4ch                  ;返回
        int 21h
code ends
end start
```

六、显示“年”参考流程

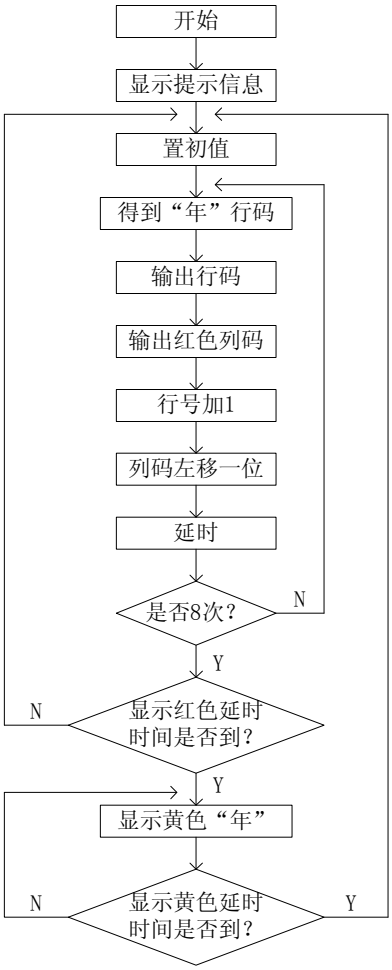


图22-4

七、显示“年”参考程序 11588-1.ASM

```
proth  equ 280h
protlr equ 288h
protly equ 290h
data segment
mess db 'Strike any key,return to DOS!',0AH,0DH,'$'
min1 db 00h,01h,02h,03h,04h,05h,06h,07h
count db 0
buff db 44h,54h,54h,7fh,54h,0dch,44h,24h
data ends
code segment
    assume cs:code,ds:data
start:  mov ax,data
        mov ds,ax
        mov dx,offset mess
```

```
        mov ah,09
        int 21h             ;显示提示信息
agn:    mov cx,80h
 d2:    mov ah,01h
        push cx
        mov cx,0008h
        mov si,offset min1
next:   mov al,[si]
        mov bx,offset buff
        xlat                ;得到第一行码
        mov dx,proth
        out dx,al
        mov al,ah
        mov dx,protlr
        out dx,al           ;显示第一行红
        shl ah,01
        inc si
        push cx
        mov cx,8fffh
delay2: loop delay2         ;延时
        pop cx
        loop next
        pop cx
        call delay
        loop d2
        mov al,00
        mov dx,protlr
        out dx,al
        mov ah,01           ;有无键按下
        int 16h
        jnz a2
agn1:   mov cx,80h          ;agn1为显示黄色
d1:     mov si,offset min1
        mov ah,01
        push cx
        mov cx,0008h
next1:  mov al,[si]
        mov bx,offset buff
        xlat
```

```
        mov dx,proth
        out dx,al
        mov al,ah
        mov dx,protly
        out dx,al
        shl ah,01
        inc si
        push cx
        mov cx,8fffh
delay1: loop delay1
        pop cx
        loop next1
        pop cx
        call delay
        loop d1
        mov al,00
        mov dx,protly
        out dx,al
        mov ah,01
        int 16h
        jnz a2
        jmp agn             ;黄色红色交替显示
delay proc near             ;延迟子程序
        push cx
        mov cx,0ffffh
ccc:    loop ccc
        pop cx
        ret
delay endp
a2:     mov ah,4ch
        int 21h             ;返回
        code ends
        end start
```

第五章 VC++实验部分

1、基本输入输出-基本输入输出函数简介

1、Startup();

语法：BOOL Startup()

功能描述:查询TPC-USB微机接口实验装置是否可用，如果可用则打开。

参数:无

返回值:如果设备存在并且可用，则返回True，否则返回False

备注:应用程序在对TPC-USB做任何操作之前必须调用该函数，应用程序结束时必须使用Cleanup函数关闭该设备。

2、void Cleanup();

语法:void Cleanup()

功能描述:关闭设备。

参数:无

返回值:无

备注:应用程序结束时必须使用Cleanup函数关闭该设备。它和Startup成对使用。

3、PortReadByte;

语法:BOOL PortReadByte(DWORD address, BYTE *pdata);

功能描述:读TPC-USB某个的IO端口值。

参数:

address:指明要读的IO端口地址。

pdata: 该函数执行完后，address所指明的端口值被填入该地址。

返回值:如果读成功，则返回True，否则返回False。

备注:应用程序使用该函数前必须先调用Startup函数。

例子:

```
BYTE    data;
DWORD   address = 0x283;
if (!Startup())
       {
    //ERROR  ..出错处理
       }
       if(!PortReadByte(address,&data))
       {
    //ERROR  .. 出错处理
       }
       //SUCCESS  ..成功，此时data里存放地址为address的IO端口的值
```

4、PortWriteByte;

语法:BOOL PortWriteByte(DWORD address, BYTE data);

功能描述:将给定值写入TPC-USB所指明的IO端口。

参数:

address:指明要写的硬件IO端口地址

data:　该函数执行完后，data将被写入address所指明的IO端口

返回值:如果读成功，则返回True，否则返回False。

备注:应用程序使用该函数前必须先调用Startup。

例子:

```
BYTE    data;
DWORD   address = 0x283;
if (!Startup())
      {
    //ERROR  ..出错处理
      }
      if(!PortReadByte(address,&data))
      {
    //ERROR  ..出错处理
      }
      //SUCCESS  ..此时已经将值data写入address所指明的IO端口
```

几点约定:

1、实验电路介绍中凡不加“利用通用插座”说明的均为实验台上已固定电路。

2、实验电路连线在图中均用虚线表示，实线为已连好电路。

3、所以实验程序均加有延时子程序，请老师根据实验PC机情况，适当加减延时衡的延时时间即可。

实验一　I/O地址译码

一、实验目的

掌握I/O地址译码电路的工作原理。

二、实验原理和内容

实验电路如图1-1所示，其中74LS74为D触发器，可直接使用实验台上数字电路实验区的D触发器,74LS138为地址译码器。译码输出端Y0～Y7在实验台上“I/O地址“输出端引出，每个输出端包含8个地址，Y0：280H～287H，Y1：288H～28FH，……　当CPU执行I/O指令且地址在280H～2BFH范围内，译码器选中，必有一根译码线输出负脉冲。

例如：执行下面一条指令

```
PortWriteByte(0X2a0,0X10);
```

Y4输出一个负脉冲，执行下面一条指令

```
PortWriteByte(0X2a8,0X10);
```

Y5输出一个负脉冲。

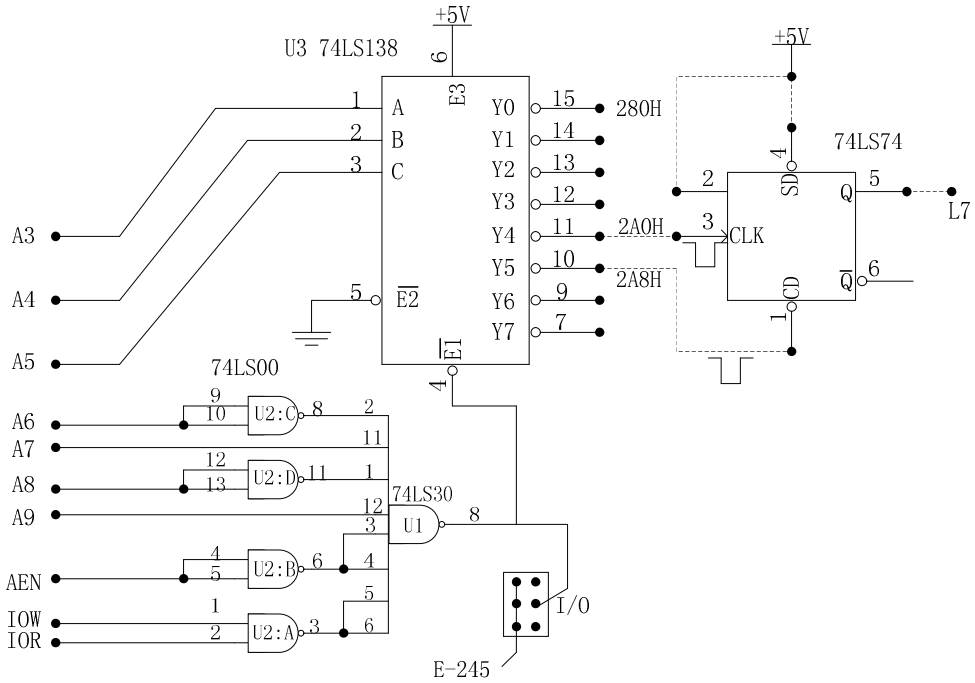


图1-1

利用这个负脉冲控制L7闪烁发光（亮、灭、亮、灭、……），时间间隔通过软件延时实现。

三、编程提示

1、实验电路中D触发器CLK端输入脉冲时，上升沿使Q端输出高电平L7发光，CD端加低电平L7灭。

2、参考程序：YMQ.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Press any key to exit!");
    while(!kbhit())
    {
        PortWriteByte(0x2a0,0x10);
```

```
        Sleep(1000);
        PortWriteByte(0x2a8,0x10);
        Sleep(1000);
    }
    Cleanup();
}
```

实验二　简单并行接口

一、实验目的

掌握简单并行接口的工作原理及使用方法。

二、实验内容

1、按下面图2-1简单并行输出接口电路图连接线路(74LS273插通用插座，74LS32用实验台上的“或门”)。74LS273为八D触发器，8个D输入端分别接数据总线D0～D7，8个Q输出端接LED显示电路L0～L7。

2、编程从键盘输入一个字符或数字，将其ASCⅡ码通过这个输出接口输出，根据8个发光二极管发光情况验证正确性。

3、按下面图2-2简单并行输入接口电路图连接电路(74LS244插通用插座，74LS32用实验台上的“或门”)。74LS244为八缓冲器，8个数据输入端分别接逻辑电平开关输出K0～K7，8个数据输出端分别接数据总线D0～D7。

4、用逻辑电平开关预置某个字母的ASCⅡ码，编程输入这个ASCⅡ码，并将其对应字母在屏幕上显示出来。

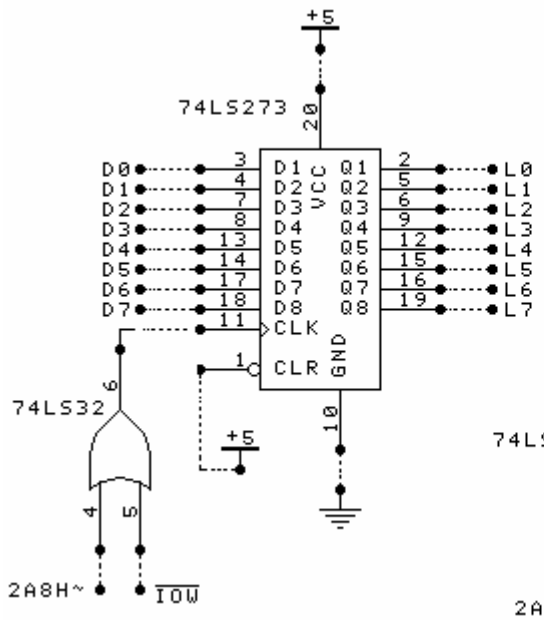


图2-1

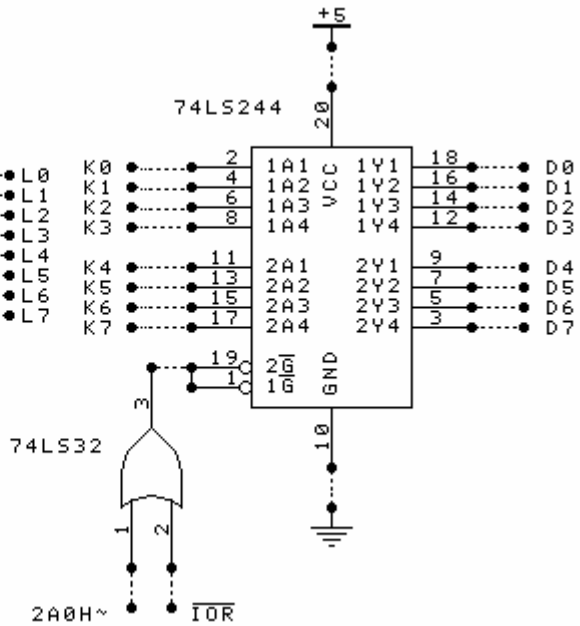


图2-2

三、编程提示

1、上述并行输出接口的地址为2A8H，并行输入接口的地址为2A0H，通过上述并行接口电路输出数据需要一条指令：

PortWriteByte(0X2a8,BYTE data)；

通过上述并行接口输入数据需要一条指令：

PortWriteByte(0X2a0,& data)；

2、参考流程图

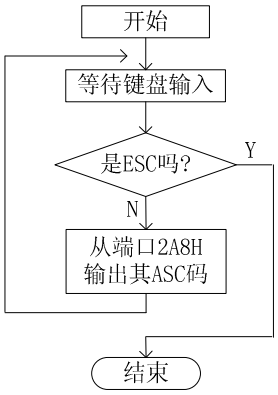


图2-3

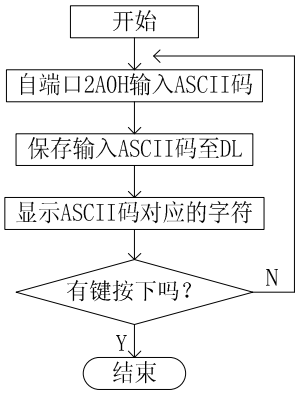


图2-4

3、参考程序1: E273.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    char k;
    if (!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("ESC is to exit!\n");
    while((k=getch())!=27)
    {
        printf("%x\n",k);
        PortWriteByte(0x2a8,(BYTE)k);
    }
    Cleanup();
}
```

4、参考程序2: E244.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    byte data;
    if (!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    while(!kbhit())
    {
        PortReadByte(0x2a0,&data);
        printf("%x\n",data);
    }
    Cleanup();
}
```

实验三　可编程定时器/计数器（8253）

一、实验目的

掌握8253的基本工作原理和编程方法。

二、实验内容

1、按图3-1虚线连接电路，将计数器0设置为方式0，计数器初值为N(N≤0FH)，用手动逐个输入单脉冲，编程使计数值在屏幕上显示，并同时用逻辑笔观察OUT0电平变化(当输入N+1个脉冲后OUT0变高电平)。

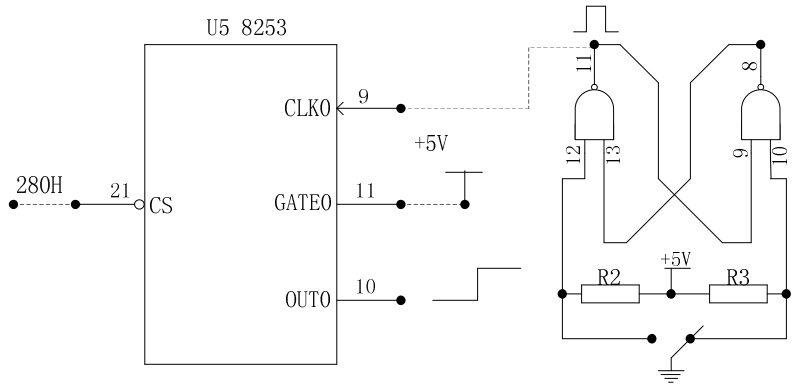


图3-1

2、按图3-2连接电路，将计数器0、计数器1分别设置为方式3，计数初值设为1000，用逻辑笔观察OUT1输出电平的变化(频率1HZ)。

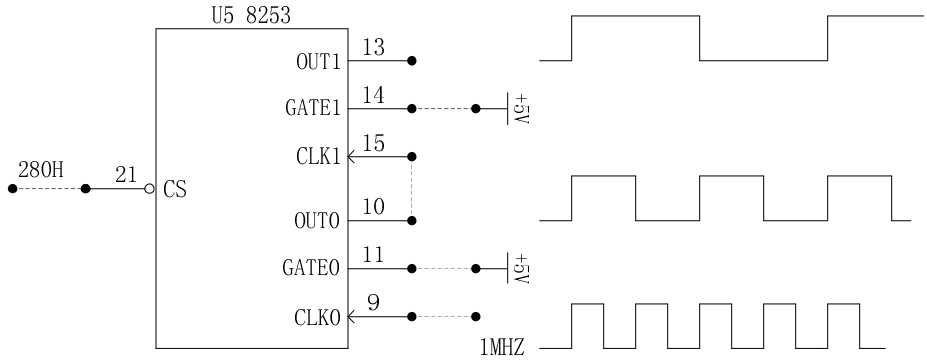


图3-2

三、 编程提示

1、8253控制寄存器地址 283H

计数器0地址 280H

计数器1地址 281H

CLK0连接时钟 1MHZ

2、参考流程图(见图3-3、图3-4):

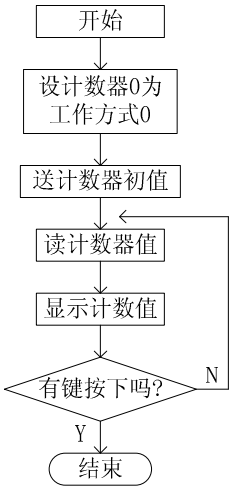


图3-3

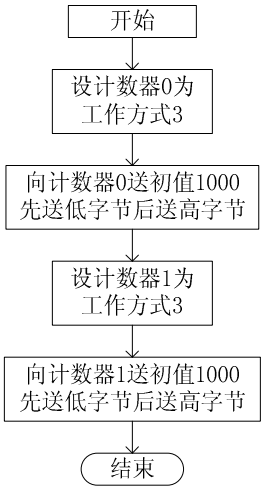


图3-4

3、参考程序1: E8253_1.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    byte data;
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    PortWriteByte(0x283,0x14);
    PortWriteByte(0x280,0x0f);
    while(!kbhit())
    {
        PortReadByte(0x280,&data);
        printf("%d\n",data);
    }
    Cleanup();
}
```

4、参考程序2: E8253_2.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
```

```
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    PortWriteByte(0x283,0x36);
    PortWriteByte(0x280,1000%256);
    PortWriteByte(0x280,1000/256);
    PortWriteByte(0x283,0x76);
    PortWriteByte(0x281,1000%256);
    PortWriteByte(0x281,1000/256);
    Cleanup();
    printf("Press any key to exit!\n");
}
```

实验四　可编程并行接口（一）（8255方式0）

一、实验目的

掌握8255方式0的工作原理及使用方法。

二、实验内容

1、实验电路如图4-1，8255C口接逻辑电平开关K0～K7，A口接LED显示电路L0～L7。

2、编程从8255C口输入数据，再从A口输出。

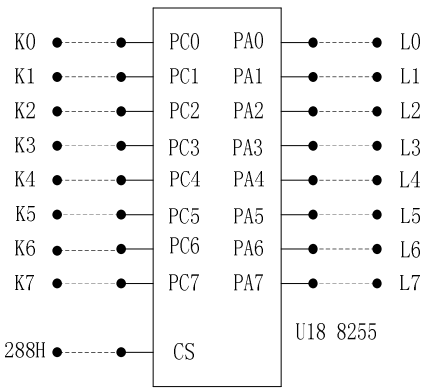


图4-1

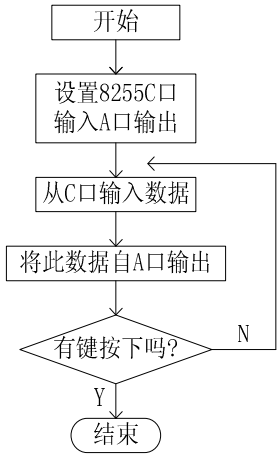


图4-2

三、编程提示

1、8255控制寄存器端口地址　　28BH
　A口的地址　　288H
　C口的地址　　28AH

2、参考流程图(见图4-2):

3、参考程序:　E8255.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    byte data;
    if (!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Press any key to exit!\n");
    while(!kbhit())
    {
        PortWriteByte(0x28b,0x8b);
        PortReadByte(0x28a,&data);
        PortWriteByte(0x288,data);
    }
    Cleanup();
}
```

实验五　七段数码管

一、实验目的

掌握数码管显示数字的原理

二、实验内容

1、静态显示:按图5-1连接好电路，将8255的A口PA0～PA6分别与七段数码管的段码驱动输入端a～g相连，位码驱动输入端S1接+5V(选中)，S0、dp接地(关闭)。编程从键盘输入一位十进制数字(0～9)，在七段数码管上显示出来。

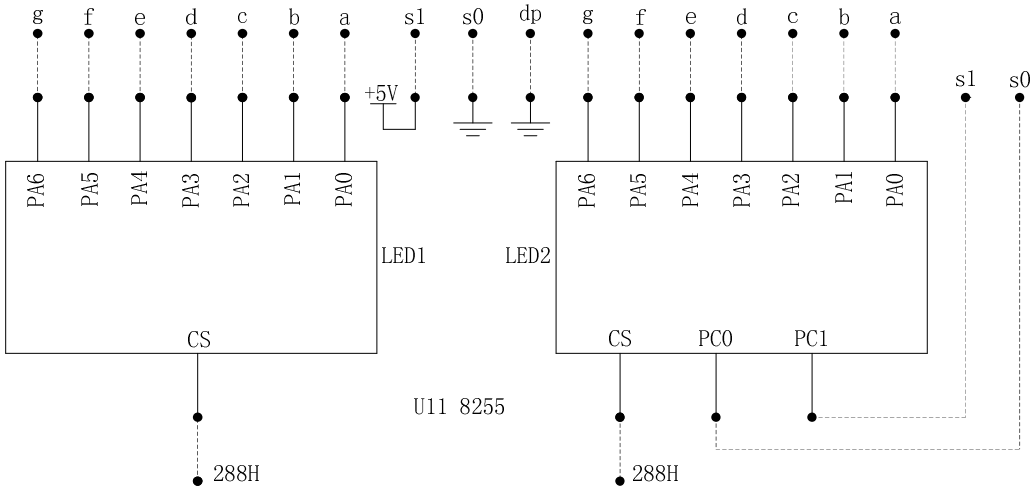


图5-1　　图5-2

2、动态显示:按图5-2连接好电路，七段数码管段码连接不变，位码驱动输入端S1，S0接8255 C口的PC1，PC0。编程在两个数码管上显示“56”。

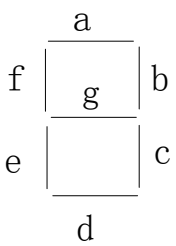
3、动态显示(选作):使用图23的电路，编程在两个数码管上循环显示“00-99”。

三、编程提示

1、实验台上的七段数码管为共阴型，段码采用同相驱动，输入端加高电平,选中的数码管亮，位码加反相驱动器，位码输入端高电平选中。

2、七段数码管的字型代码表如下表：

显示字形	g	e	f	d	c	b	a	段码
0	0	1	1	1	1	1	1	3fh
1	0	0	0	0	1	1	0	06h
2	1	0	1	1	0	1	1	5bh
3	1	0	0	1	1	1	1	4fh
4	1	1	0	0	1	1	0	66h
5	1	1	0	1	1	0	1	6dh
6	1	1	1	1	1	0	1	7dh
7	0	0	0	0	1	1	1	07h
8	1	1	1	1	1	1	1	7fh
9	1	1	0	1	1	1	1	6fh



3、参考流程图(见图5-3、图5-4)

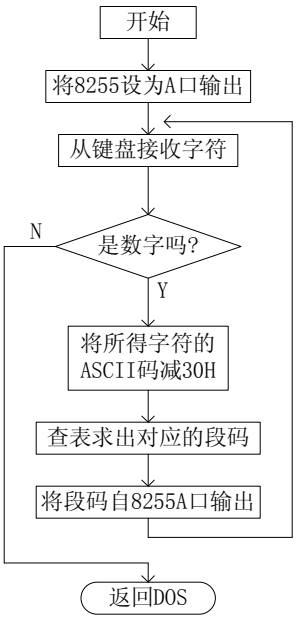


图5-3

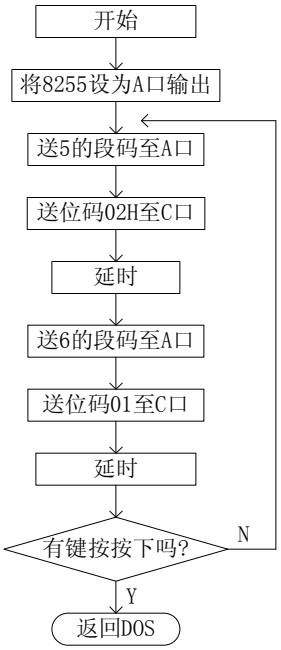


图5-4

4、参考程序1: LED1.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
char led[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
void main()
{
    int out;
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    PortWriteByte(0x28b,0x80);
    printf("\nInput a number(0-9),other key is to exit!:\n");
    while(true)
    {
        out=getch();
        if(out<0x30||out>0x39) break;
        printf("%c\n",out);
        PortWriteByte(0x288,led[out-48]);
```

```
    }
    Cleanup();
}
```

5、参考程序2: LED2.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Please enter any key return!");
    do{
        PortWriteByte(0x28b,0x82);
        PortWriteByte(0x288,0x6d);
        PortWriteByte(0x28a,0x02);
        Sleep(1000);
        PortWriteByte(0x288,0x7d);
        PortWriteByte(0x28a,0x01);
        Sleep(1000);
      }while(!kbhit());
    PortWriteByte(0x28a,0x00);
    Cleanup();
}
```

6、参考程序3: LED3.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
char led[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
void main()
{
    int i,j;
    printf("Press any key to begin!\n\n");
    getch();
```

```
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Please enter any key return!");
    do{
        PortWriteByte(0x28b,0x82);
        for(i=0;i<10;i++)
        {
            for(j=0;j<10;j++)
            {
                PortWriteByte(0x288,led[j]);
                PortWriteByte(0x28a,0x01);
                Sleep(60);
                PortWriteByte(0x288,led[i]);
                PortWriteByte(0x28a,0x02);
                Sleep(60);
                if(kbhit()) break;
            }
        }
    }while(!kbhit());
    PortWriteByte(0x28a,0x00);
    Cleanup();
}
```

实验六　继电器控制

一、实验目的

1、了解微机控制直流继电器的一般方法。

2、进一步熟悉使用8255、8253。

二、实验内容

实验电路如图6-1，按虚线连接电路：CLK0接1MHZ，GATE0，GATE1，接+5V，OUT0接CLK1，OUT1接PA0，PC0接继电器驱动电路的开关输入端Ik。编程使用8253定时，让继电器周而复始的闭合5秒钟(指示灯灯亮)，断开5秒钟(指示灯灯灭)。

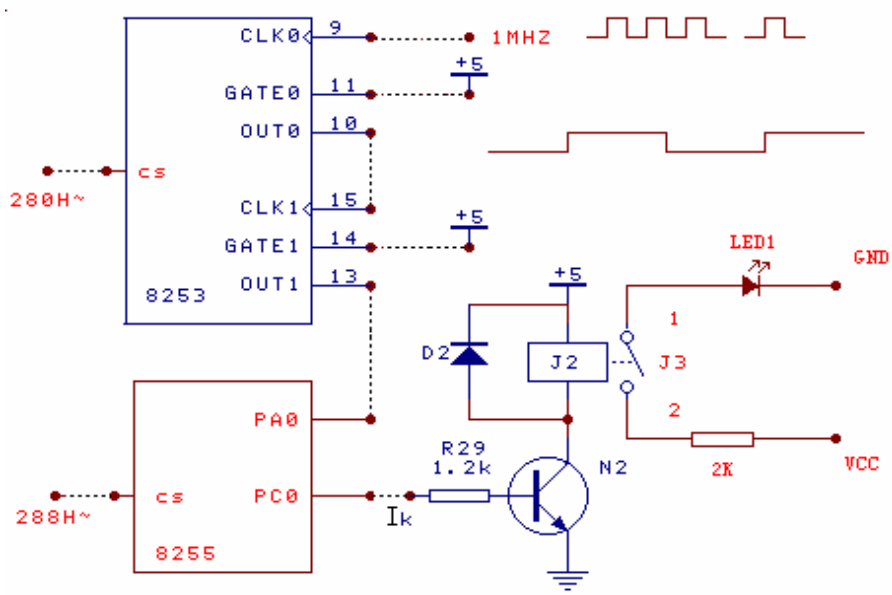


图6-1

三、编程提示

1、将8253计数器0设置为方式3、计数器1设置为方式0并联使用，CLK0接1MHZ时钟，设置两个计数器的初值(乘积为5000000)启动计数器工作后，经过5秒钟OUT1输出高电平。通过8255A口查询OUT1的输出电平，用C口PC0输出开关量控制继电器动作。

2、继电器开关量输入端输入“1”时，继电器常开触点闭合，电路接通，指示灯发亮，输入“0”时断开，指示灯 熄灭。

3、参考流程图(见图6-2)：

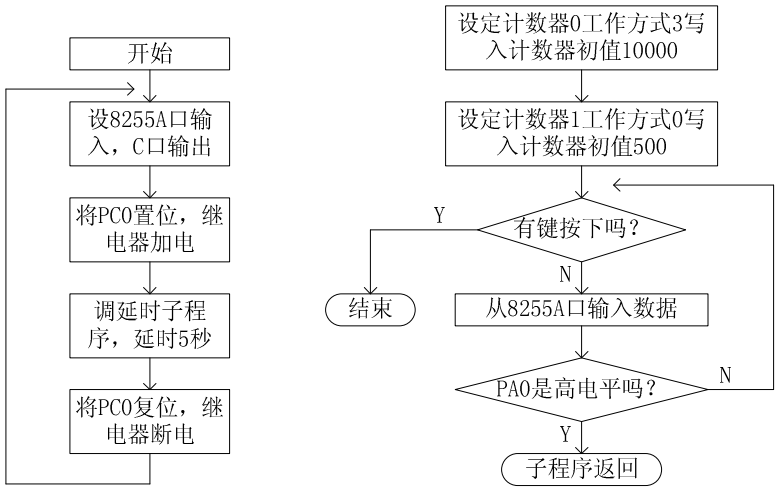


图6-2

4、参考程序：JDQ.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
```

```
#pragma comment(lib,"..\\ApiEx.lib")
void m_delay();
void main()
{
 if (!Startup())
 {
     printf("ERROR: Open Device Error!\n");
     return;
 }
 printf("press any key to return!\n");
 PortWriteByte(0x28b,0x90);
 while(true)
 {
     PortWriteByte(0x28b,1);
     m_delay();
     PortWriteByte(0x28b,0);
     m_delay();
 }
 Cleanup();
}
void m_delay()
{
     byte data;
     PortWriteByte(0x283,0x36);
     PortWriteByte(0x280,1000%256);
     PortWriteByte(0x280,1000/256);
     PortWriteByte(0x283,0x70);
     PortWriteByte(0x281,1000%256);
     PortWriteByte(0x281,1000/256);
     do{
         if(kbhit())
             exit(0);
         PortReadByte(0x288,&data);
       }while(!(data&0x01));
 }
```

实验七　竞赛抢答器

一、实验目的

1、了解微机化竞赛抢答器的基本原理。

2、进一步学习使用并行接口。

二、实验内容

图7-1为竞赛抢答器(模拟)的原理图，逻辑开关K0～K7代表竞赛抢答按钮0～7号，当某个逻辑电平开关置“ 1 ”时，相当某组抢答按钮按下。在七段数码管上将其组号(0～7)显示出来，并使微机扬声器响一下。

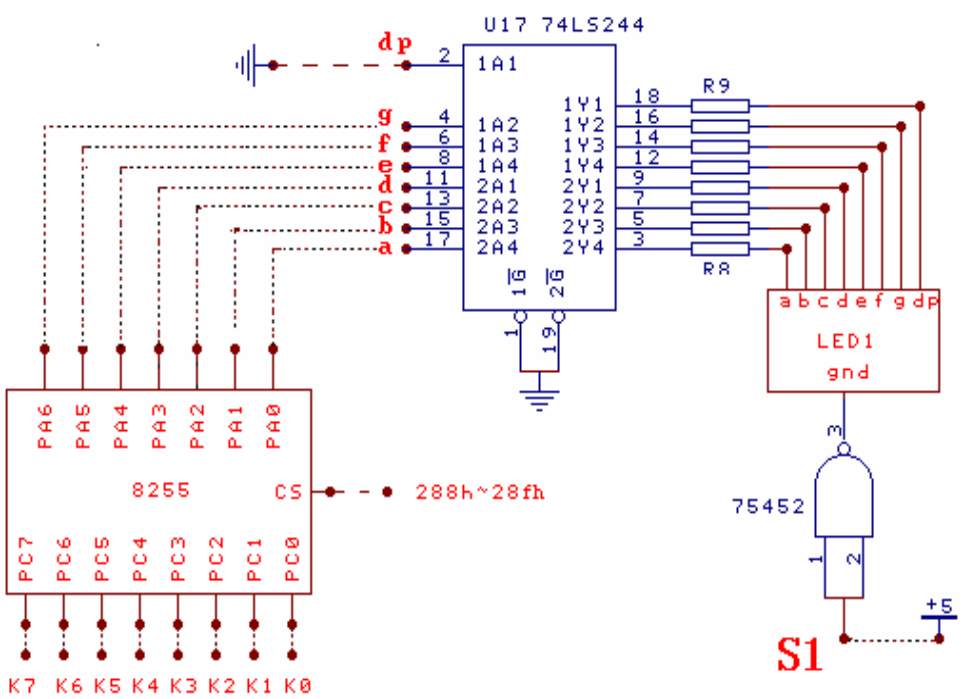


图7-1

三、编程提示

设置8255为C口输入、A口输出，读取C口数据，若为0表示无人抢答，若不为0则有人抢答。根据读取数据可判断其组号。从键盘上按空格键开始下一轮抢答，按其它键程序退出。

四、参考流程图 （见图7-2）

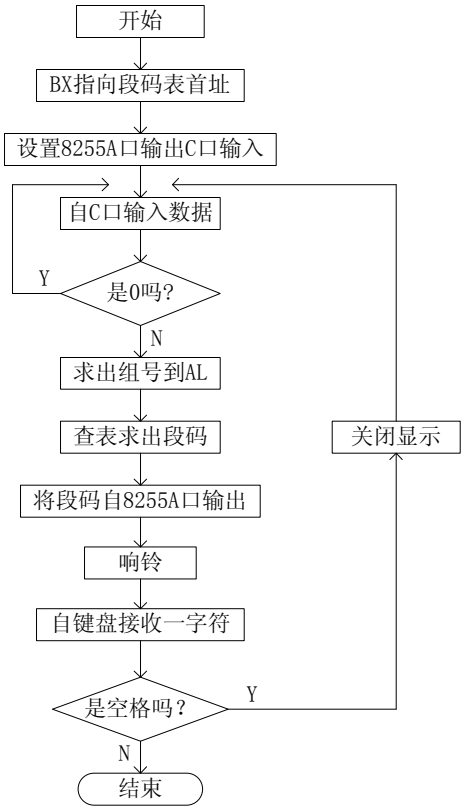


图7-2

五、参考程序： QDQ.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
int led[8]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07};
int num[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
int  i=0;
void main()
{
    BYTE    data;
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("ESC is to exit!");
    PortWriteByte(0x28b,0x89);
    for(;;)
```

```
    {
        do{
            PortReadByte(0x28a,&data);
        }while(!data);
        for(i=0;i<8;i++)
        {
            if(data==num[i])
            {
                printf("\7");
                break;
            }
        }
        if(i<8)
        {
            PortWriteByte(0x288,led[i+1]);
            data=0;
        }
        if(getch() == 27)
            exit(0);
        PortWriteByte(0x288,0);
    }
    Cleanup();
}
```

实验八　交通灯控制实验

一、实验目的

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

二、实验内容

如图8-1，L7、L6、L5作为南北路口的交通灯与PC7、PC6、PC5相连，L2、L1、L0作为东西路口的交通灯与PC2、PC1、PC0相连。编程使六个灯按交通灯变化规律亮灭。

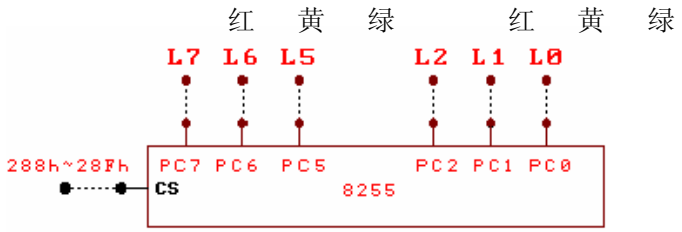


图8-1

三、编程提示：十字路口交通灯的变化规律要求：

（1） 南北路口的绿灯、东西路口的红灯同时亮30秒左右。
（2） 南北路口的黄灯闪烁若干次，同时东西路口的红灯继续亮。
（3） 南北路口的红灯、东西路口的绿灯同时亮30秒左右。
（4） 南北路口的红灯继续亮、同时东西路口的黄灯亮闪烁若干次。
（5） 转(1)重复。

四、参考流程图

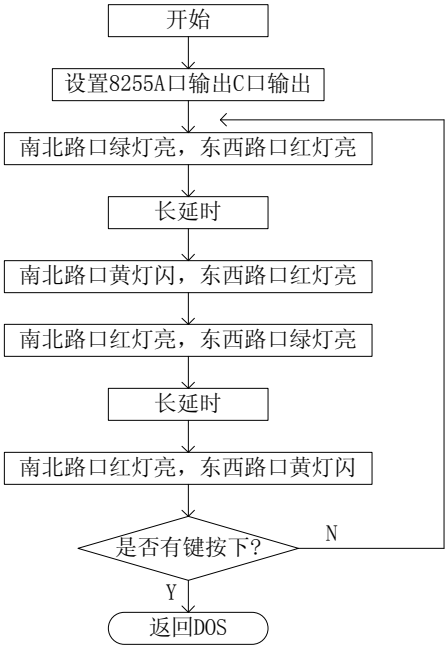


图8-2

五、参考程序： JTD.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    int i;
    int portc[]={0x24,0x44,0x04,0x44,0x04,0x44,0x04,
            0x81,0x82,0x80,0x82,0x80,0x82,0x80,0xff};
    if(!Startup())
    {
        printf("ERROR:Open Device Error!\n");
        return;
    }
    printf("Enter any key will return!\n");
```

```
    PortWriteByte(0x28b,0x80);
    for(;;)
    {
        for(i=0;i<14;i++)
        {
            PortWriteByte(0x28a,portc[i]);
            if(kbhit())
                exit(0);
            if(portc[i]&0x21)
                Sleep(1800);
            else
                Sleep(600);
        }
    }
    Cleanup();
}
```

2、中断-中断函数简介

1、EnableIntr;

语法:BOOL EnableIntr();

功能描述:将微机实验装置的中断输入设为有效，执行此函数后，PLX9054将接受微机实验装置上的中断请求，然后根据该请求申请一个PCI中断。

参数:无

返回值:如果成功，则返回True，否则返回False

备注:应用程序在调用该函数之前，必须先调用Startup函数。

2、DisableIntr;

语法:BOOL DisableIntr();

功能描述:将微机实验装置的中断输入设为无效，执行此函数后，PLX9054将不相应微机实验装置上的中断请求

参数:无

返回值:如果成功，则返回True，否则返回False

备注:应用程序在调用该函数之前，必须先调用Startup函数。

3、RegisterISR;

语法:BOOL RegisterISR(ISR_ROUTINE pfuncISR);

功能描述:注册中断服务程序，当微机实验箱上的中断输入有效时，且实验箱上的中断输入使能，程序将会执行该中断服务程序。

参数:pfuncISR:该参数即为中断服务函数名

返回值:如果成功，则返回True，否则返回False

备注:应用程序在调用该函数之前，必须先调用Startup函数。

实验九　中断

一、实验目的

1、了解Windows下中断处理过程。

2、比较中断和查询两种数据交换方法的效率差别。

二、实验内容

用查询和中断方式分别实现控制指示灯，实验电路如图9-1。要求直接用手动产生的单脉冲作为中断请求信号，每按一次单脉冲产生一次中断，让指示灯显示一秒钟的0x55，否则让指示灯显示0xAA。然后在任务栏比较中断和查询方式下CPU利用率的差别。

1、用查询方法

将8255的A口设为输出接指示灯，C口设为输入，将PC0接正脉冲输入，线路连接如下图所示：

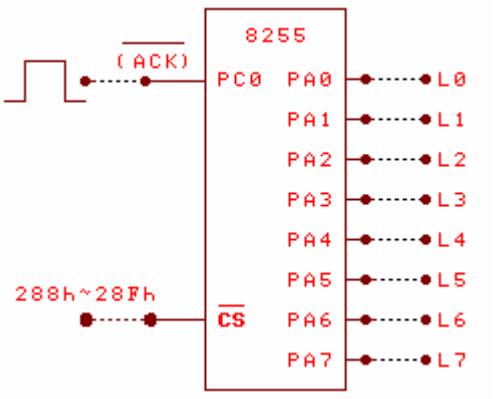


图9-1

2、用中断方法

将8255的A口设为输出，IRQ直接接到正脉冲

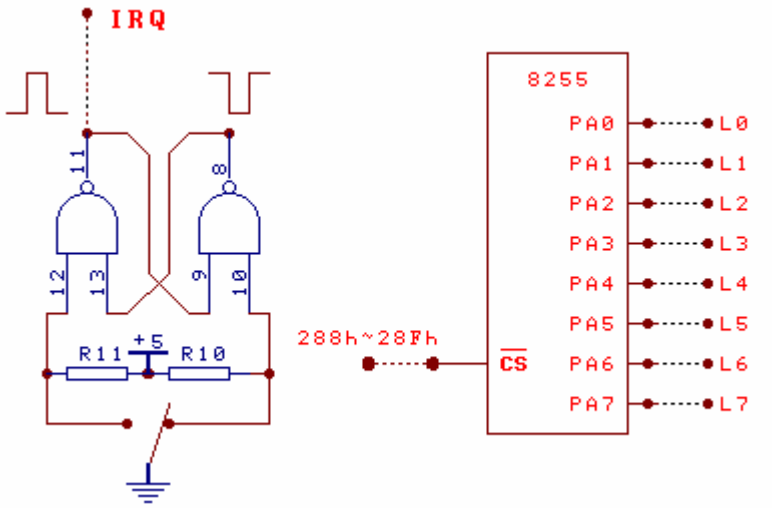


图9-2

三、编程提示

1、查询方式流程图：

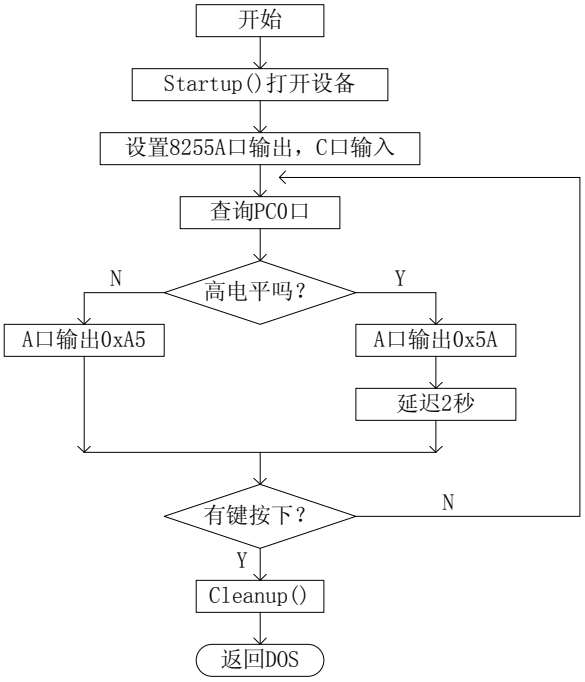


图9-3

2、中断方式流程图：

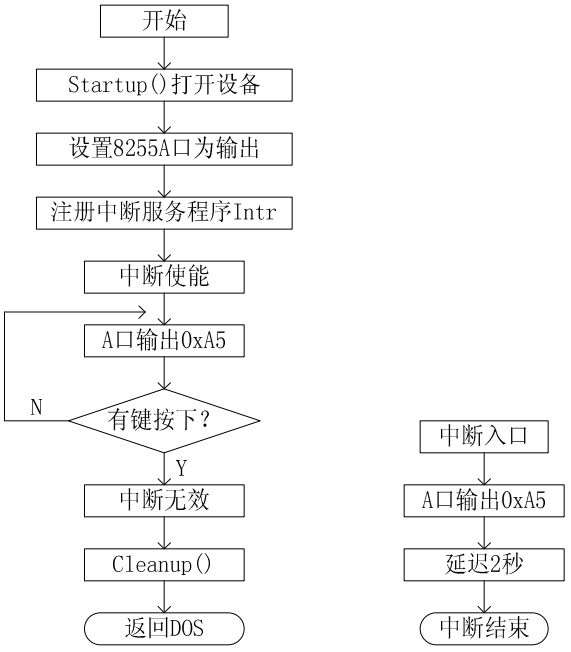


图9-4

四、参考程序(查询方式)：INT0.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    BYTE    data;
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Please Press DMC! Press any key to exit!\n");
    PortWriteByte(0x28b,0x8b);
    while(!kbhit())
    {
        PortReadByte(0x28a,&data);
        if(data&0x01)
        {
            PortWriteByte(0x288,0x55);
            Sleep(1*1000);
        }
        PortWriteByte(0x288,0xaa);
    }
    Cleanup();
}
```

五、参考程序(中断方式):

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEX.h"
#pragma comment(lib,"..\\ApiEx.lib")
int i;
void MyISR()
{
    PortWriteByte(0x288,0x55);
    Sleep(1000);
    printf("%d\n",i++);
}
void main()
```

```
{
    printf("Press any key to begin!\n\n");
    getch();
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Please Press DMC! Press any key to exit!\n");
    PortWriteByte(0x28b,0xa0);
    RegisterLocalISR(MyISR);
    EnableIntr();
    while(!kbhit())
    {
        PortWriteByte(0x288,0xaa);
        Sleep(1000);
    }
    DisableIntr();
    Cleanup();
}
```

实验十　可编程并行接口（二）（8255方式１）

一、实验目的

1、掌握8255工作方式１时的使用及编程。

2、进一步掌握中断处理程序的编写。

二、实验内容

1、按图10-1，8255方式1的输出电路连好线路。

2、编程：每按一次单脉冲按钮产生一个正脉冲使8255产生一次中断请求，让CPU进行一次中断服务：依次输出01H、02H，04H，08H，10H，20H，40H，80H使L0～L7依次发光，中断８次结束。

3、按图10-2，8255方式1输入电路，连好线路。

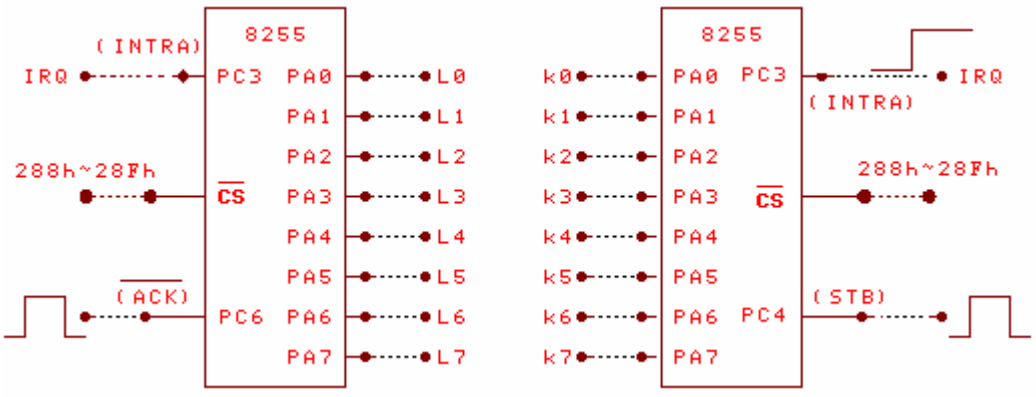
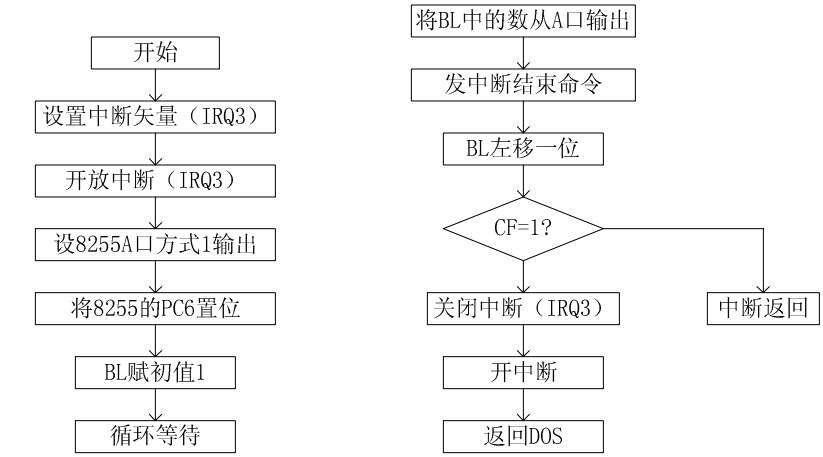


图10-1 输出电路　　图10-2 输入电路

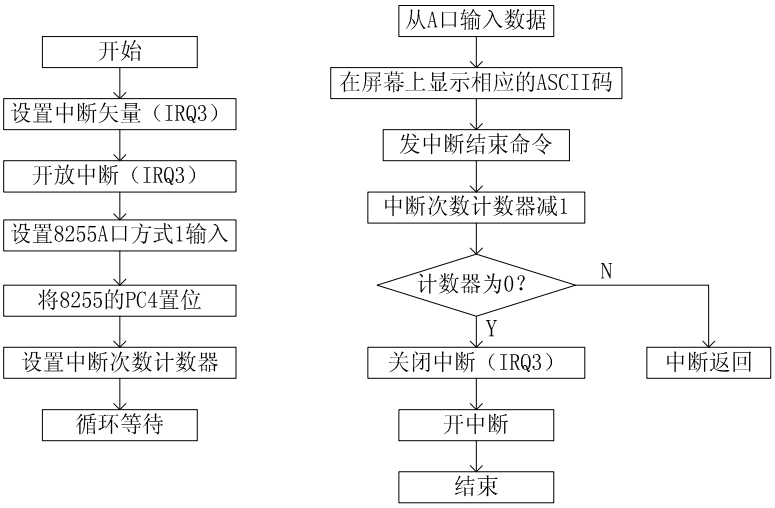
4、编程:每按一次单脉冲按钮产生一个正脉冲使8255产生一次中断请求，让CPU进行一次中断服务:读取逻辑电平开关预置的ASCII码，在屏幕上显示其对应的字符，中断８次结束。

三、参考流程图:（如图10-3，图10-4）



（A） 输出主程序　　（B） 输出中断服务程序

图10-3



（A） 输出主程序　　（B） 输出中断服务程序

图10-4

四、参考程序１： E8255-1o.CPP

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
byte Count=0x01;
void IntS();
void main()
{
    if(!Startup())
    {
        printf("ERROR:Open Device Error!\n");
        return;
    }
    printf("Press DMC !Press any key to exit!\n");
    RegisterLocalISR(IntS);
    EnableIntr();
    PortWriteByte(0x28b,0xa0);
    PortWriteByte(0x28b,0x0d);
    while(!kbhit());
    DisableIntr();
    Cleanup();
}
void IntS()
{
```

```
        PortWriteByte(0x288,Count);
        printf("This is a Interrupt!Out=%x\n",Count);
        Count<<=1;
        if(Count==0)
            exit(0);
        Sleep(1000);
    }
```

五、参考程序 2： E8255-1i.CPP

```
    #include <stdio.h>
    #include <conio.h>
    #include "..\\ApiEx.h"
    #pragma comment(lib,"..\\ApiEx.lib")
    int Count=8;
    void IntS();
    void main()
    {
        if(!Startup())
        {
            printf("ERROR:Open Device Error!\n");
            return;
        }
        printf("Press DMC !Press any key to exit!\n");
        RegisterLocalISR(IntS);
        EnableIntr();
        PortWriteByte(0x28b,0xb8);
        PortWriteByte(0x28b,0x09);
        while(!kbhit());
        DisableIntr();
        Cleanup();
    }
    void IntS()
    {
        byte data;
        PortReadByte(0x288,&data);
        printf("This is a Interrupt!In=%x\n",data);
        Count--;
        if(Count==0)
            exit(0);
}
```

实验十一　数/模转换器

一、实验目的

了解数/模转换器的基本原理，掌握DAC0832芯片的使用方法。

二、实验内容

1、实验电路原理如图11-1，DAC0832采用单缓冲方式，具有单双极性输入端(图中的Ua、Ub),利用debug输出命令(Out 290 数据)输出数据给DAC0832，用万用表测量单极性输出端Ua及双极性输出端Ub的电压，验证数字与电压之间的线性关系。

2、编程产生以下波形(从Ub输出，用示波器观察)

（1）锯齿波

（2）正弦波

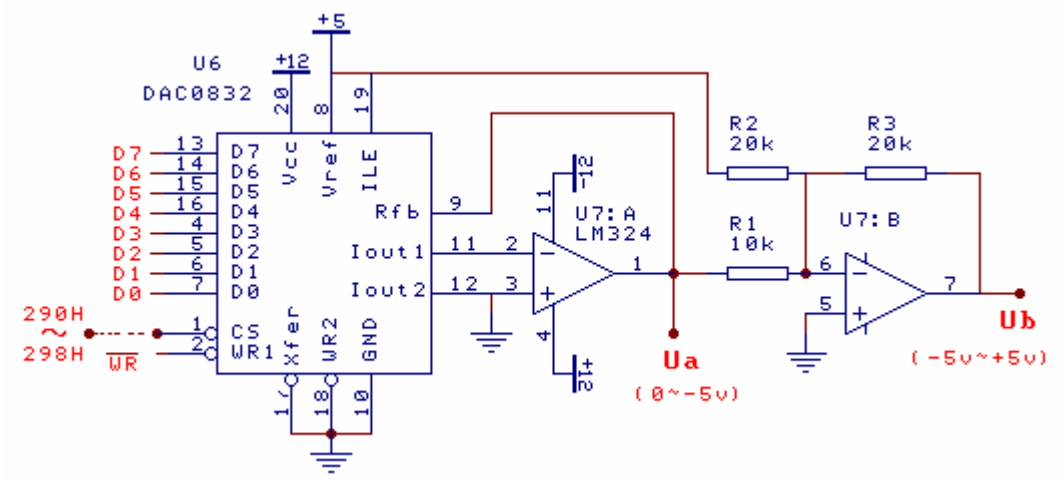


图11-1

三、编程提示

1、8位D/A转换器DAC0832的口地址为290H，输入数据与输出电压的关系为：

$$Ua = -\frac{U_{REF}}{256} \times N$$

$$Ub = -\frac{2U_{REF}}{256} \times N - 5$$

(UREF表示参考电压,N表示数数据)，这里的参考电压为ＰＣ机的＋５Ｖ电源。

2、产生锯齿波只须将输出到DAC0832的数据由0循环递增。产生正弦波可根据正弦函数建一个下弦数字量表，取值范围为一个周期，表中数据个数在16个以上。

四、参考流程图:

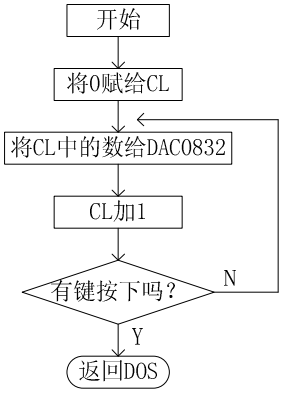


图11-2 锯齿波

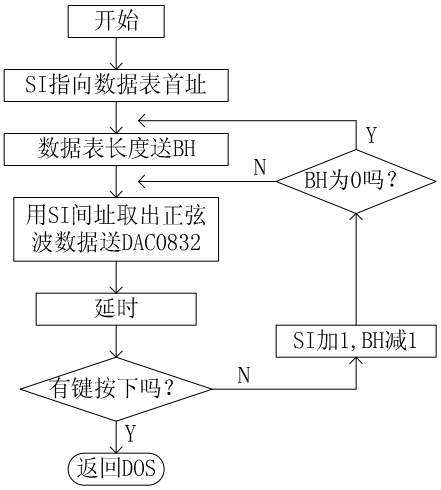


图11-3 正弦波

五、参考程序 1： DA_1.CPP

```
#include<conio.h>
#include<stdio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    char    i = 0;
    printf("Press any key to exit!\n");
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    do{
        PortWriteByte(0x290,i++);
      }while(!kbhit());
    Cleanup();
}
```

六、参考程序 2： DA_2.CPP

```
#include<conio.h>
#include<stdio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
```

```
    int i;
    int m_sin[]={0x80,0x96,0x0ae,0x0c5,0x0d8,0x0e9,0x0f5,0x0fd,0x0ff,
        0x0fd,0x0f5,0x0e9,0x0d8,0x0c5,0x0ae,0x96,0x80,0x66,0x4e,0x38,
        0x25,0x15,0x09,0x04,0x00,0x04,0x09,0x15,0x25,0x38,0x4e,0x66};
    printf("Press any key to begin!\n\n");
    getch();
    printf("Press any key to exit!\n");
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    do{
        for(i=0;i<32;i++)
        {
            PortWriteByte(0x290,m_sin[i]);
            if(kbhit())
                break;
        }
      }while(!kbhit());
    Cleanup();
}
```

实验十二　模/数转换器

一、实验目的

了解模/数转换的基本原理，掌握ADC0809的使用方法。

二、实验内容

1、实验电路原理图如图12-1。通过实验台左下角电位器RW1输出0～5Ｖ直流电压送入ADC0809通道0(IN0),利用debug的输出命令启动Ａ/Ｄ转换器，输入命令读取转换结果，验证输入电压与转换后数字的关系。

启动IN0开始转换：　PortWriteByte(0x298,0);

读取转换结果：　　PortReadByte(0x298,&data);

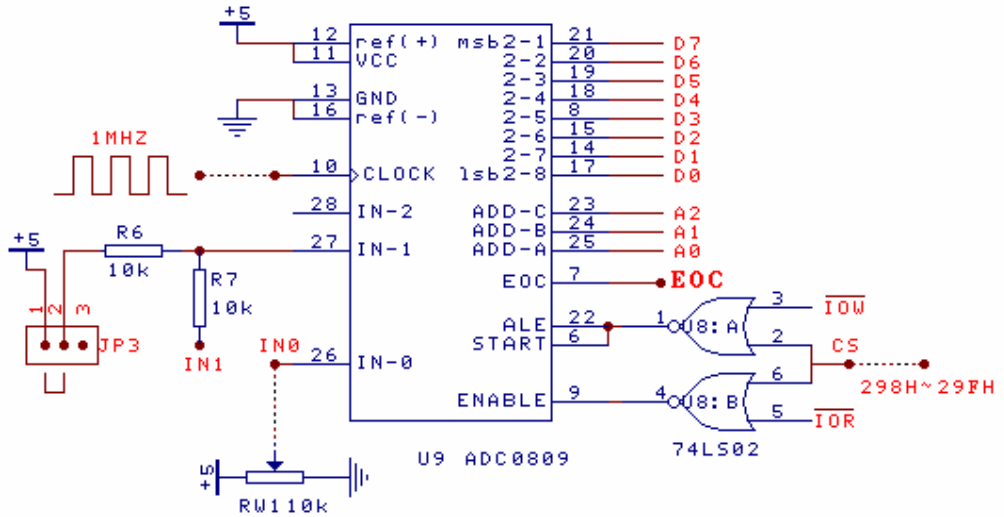


图12-1 模数转换电路

2、编程采集IN0输入的电压,在屏幕上显示出转换后的数据(用16进制数)。

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

三、实验提示

1、ADC0809的IN0口地址为298H，IN1口地址为299H。

2、IN0单极性输入电压与转换后数字的关系为：

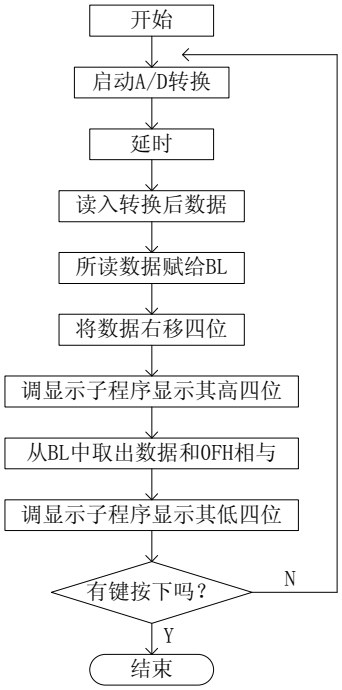
$$N = \frac{U_i}{U_{REF}/256}$$

其中Ui为输入电压，UREF为参考电压，这里的参考电压为PC机的＋５Ｖ电源。

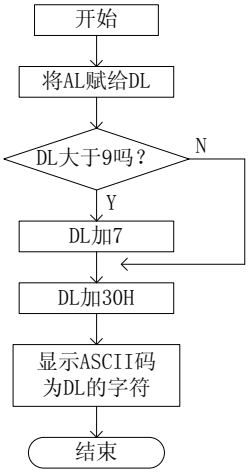
3、一次A/D转换的程序可以为

```
PortWriteByte(口地址，数据);  //启动转换
延时
PortReadByte(口地址，&data);  //读取转换结果放在data中
```

四、参考流程图（见图12-2,图12-3）



(A) 主程序



（B）显示子程序

图12-2

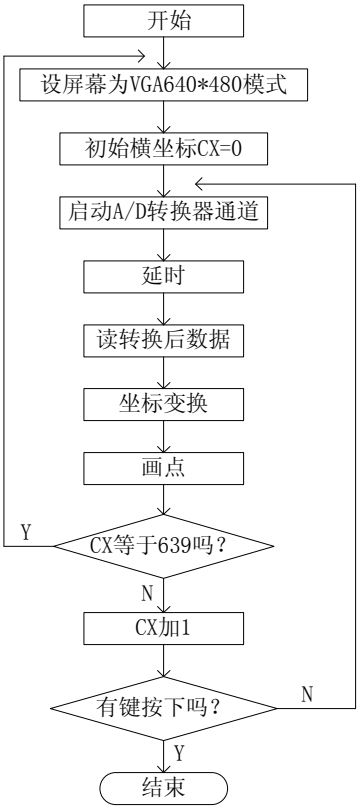


图12-3

五、参考程序 1： AD_1.CPP

```
#include <conio.h>
#include <stdio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    byte    data;
    printf("Press any key to begin!\n\n");
    getch();
    printf("Press any key to exit!\n");
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    while(!kbhit())
    {
        PortWriteByte(0x298,0x00);
        Sleep(70);
        PortReadByte(0x298,&data);
        printf("%d\n",data);
    }
    Cleanup();
}
```

六、参考程序 2： AD_2.CPP

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("press any key to start!\n");
    getch();
```

```
    byte data;
    printf("Press any key to exit!\n");
    while(!kbhit())
    {
        PortWriteByte(0x299,00);
        Sleep(70);
        PortReadByte(0x299,&data);
        for(data=data/4;data>0;data--)
        {
            printf("*");
        }
        printf("\n");
    }
    Cleanup();
}
```

实验十三　串行通讯

一、实验目的

1、了解串行通讯的基本原理。

2、掌握串行接口芯片8251的工作原理和编程方法。

二、实验内容

1、按图13-1连接好电路,(8251插通用插座)其中8253计数器用于产生8251的发送和接收时钟，TXD和RXD连在一起。

2、编程：从键盘输入一个字符,将其ASCII码加 1 后发送出去,再接收回来在屏幕上显示，实现自发自收。

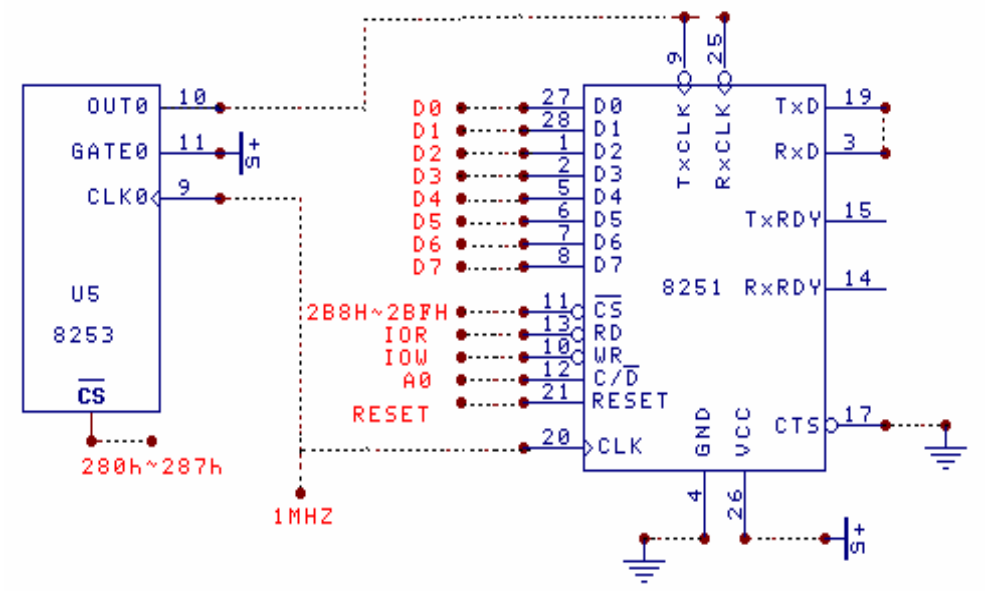


图13-1 串行通讯电路

三、实验提示

1、图示电路8251的控制口地址为2B9H,数据口地址为2B8H。

2、8253计数器的计数初值=时钟频率/(波特率×波特率因子)，这里的时钟频率接1MHz，波特率若选1200，波特率因子若选16，则计数器初值为52。

3、收发采用查询方式。

四、参考流程图（见图13-2）

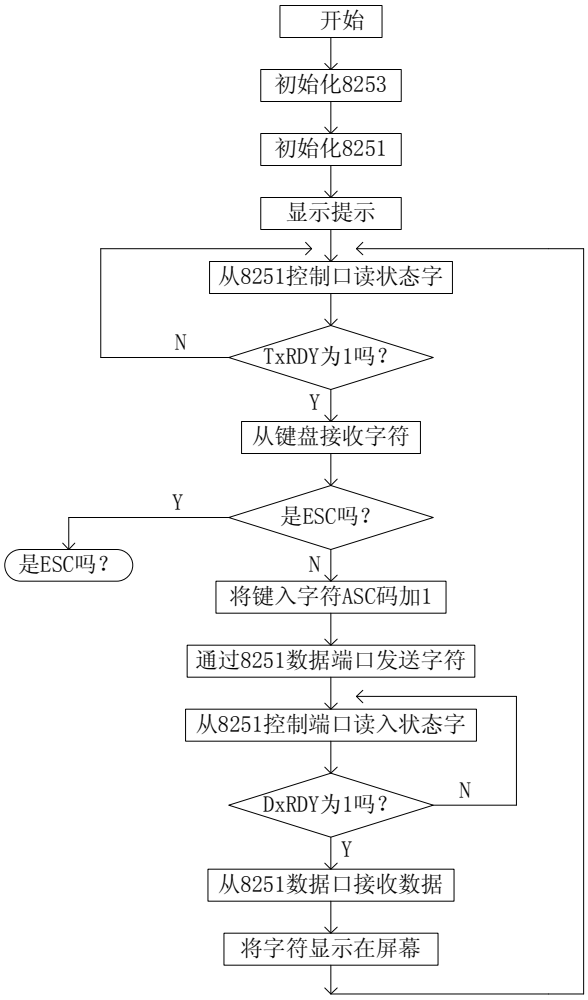


图14-2

五、参考程序： E8251.CPP

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    int i;
    BYTE data;
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
```

```
    printf("You can Press a key to start:\n");
    getch();
    printf("ESC is exit!\n");
    PortWriteByte(0x283,0x16);
    Sleep(1*100);
    PortWriteByte(0x280,52);
    Sleep(1*100);
    for(i=0;i<3;i++)
    {
        PortWriteByte(0x2b9,0);
        Sleep(1*100);
    }
    PortWriteByte(0x2b9,0x40);
    Sleep(1*100);
    PortWriteByte(0x2b9,0x4e);
    Sleep(1*100);
    PortWriteByte(0x2b9,0x27);
    Sleep(1*100);
    for(;;)
    {
        do{
            PortReadByte(0x2b9,&data);
          }while(!(data&0x01));
        data = getch();
        if(data == 0x1b) exit(0);
        putchar(data);
        PortWriteByte(0x2b8,data+1);
        Sleep(1*100);
        do{
            PortReadByte(0x2b9,&data);
          }while(!(data&0x02));
        PortReadByte(0x2b8,&data);
        printf("%c",data);
    }
    Cleanup();
}
```

3、DMA及RAM操作函数简介

1、Write8237

语法: bool Write8237(WORD address, BYTE data);

功能描述:写TPC-USB模块上8237的某个端口。

参数:

address:指明要写的8237端口地址

data: 该函数执行完后，data将被写入address所指明的8237端口

返回值:如果读成功，则返回True，否则返回False。

备注:应用程序使用该函数前必须先调用Startup函数。

2、Read8237

语法: bool Read8237(WORD address, BYTE* pdata);

功能描述:读TPC-USB模块上8237某个端口值。

参数:

address:指明要读的8237端口地址。

pdata: 该函数执行完后，address所指明的端口值被填入该地址。

返回值:如果读成功，则返回True，否则返回False。

备注:应用程序使用该函数前必须先调用Startup函数。

3、MemReadByte

语法: bool MemReadByte(WORD address, BYTE *pdata);

功能描述:读存储器。

参数:

address:指明要读的存储器地址。

pdata: 该函数执行完后，address所指明的端口值被填入该地址。

返回值:如果读成功，则返回True，否则返回False。

备注:应用程序使用该函数前必须先调用Startup函数。

4、MemWriteByte

语法: bool MemWriteByte(WORD address, BYTE data);

功能描述:写存储器。

参数:

address:指明要写的存储器地址。

pdata: 该函数执行完后，data将被填入address所指明的存储器地址。

返回值:如果读成功，则返回True，否则返回False。

备注:应用程序使用该函数前必须先调用Startup函数。

实验十四　DMA传送

一、实验目的

1、掌握PC机工作环境下进行DMA方式数据传送(Block MODE和Demand Mode)(块传送、外部请求传送)方法。

2、掌握DMA的编程方法。

二、实验内容

1、用通用插座按图14-1将6116电路连接好。编程将主机内存缓冲区D4000H,偏移量为0的一块数据循环写入字符A～Z，用Block MODE DMA方式传送到实验箱上的RAM6116上，并察看送出的数据是否正确。其中扩展6116首地址为D6000H。

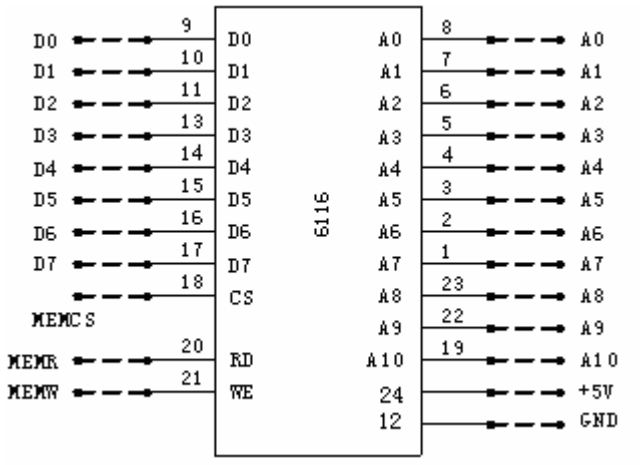


图14-1

2、用通用插座按图14-2连接好电路（74LS74利用实验台上的D触发器）。编程将主机内存缓冲区D4000H,偏移量为0的10个数据，使用Demand Mode DMA方式从内存向外设传送。

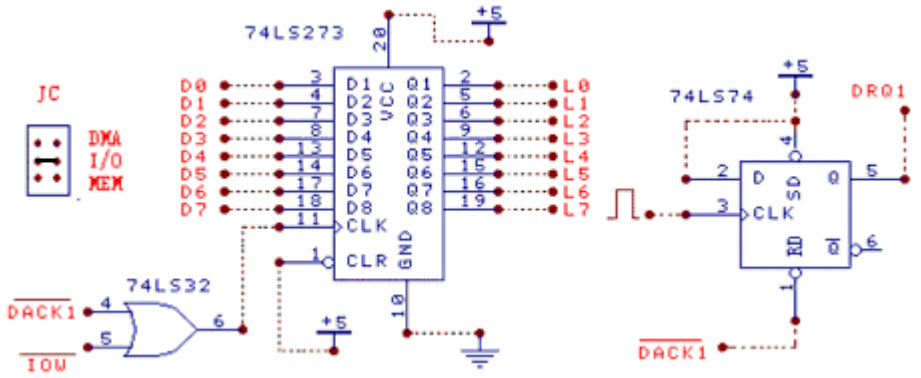


图14-2

3、用通用插座按图14-3连接好电路（74LS74利用实验台上的D触发器）。编程在主机内存缓冲区D4000H,偏移量为0的位置开辟数据缓冲区，使用Demand Mode DMA方式从外设向内存传送8个数据并存入数据缓冲区，编程不断显示缓冲区的数据。

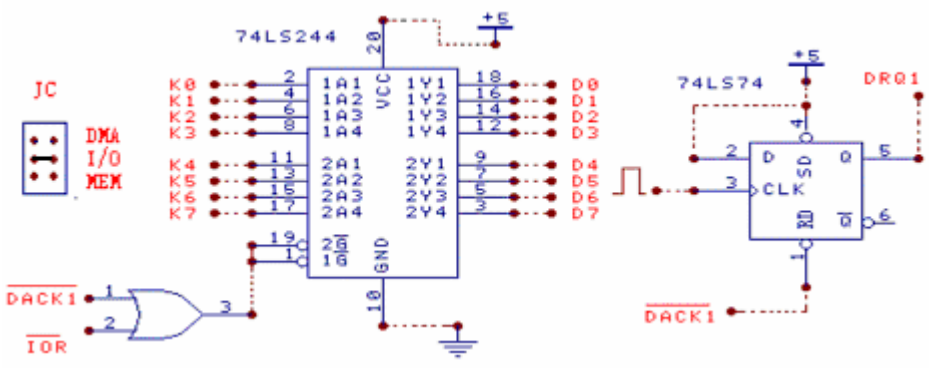


图14-3

三、实验提示

1、 DMA请求是由单脉冲输入到D触发器，由触发器的Q端向DRQ1发出的。CPU响应后发出$\overline{\text{DACK}}$1，将触发器Q置成低电平以撤消请求。

2、TPC-USB模块上内存范围为0D4000H-0D7fffH。

四、参考流程图（见图14-4、图14-5、图14-6）

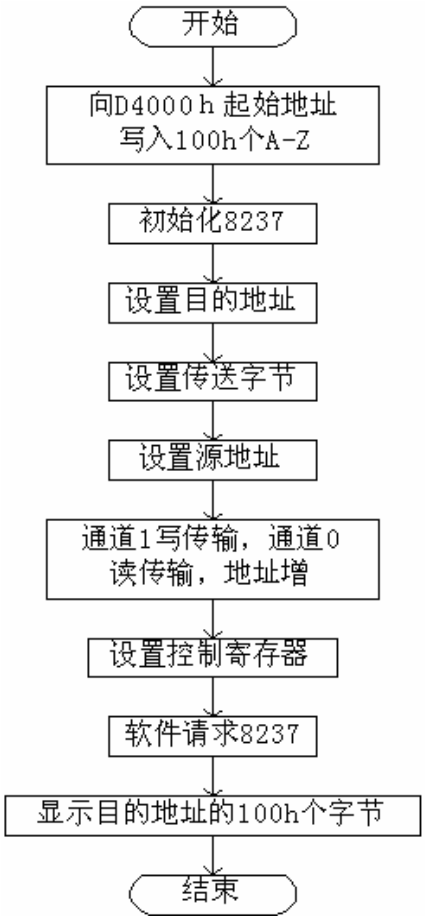


图14-4

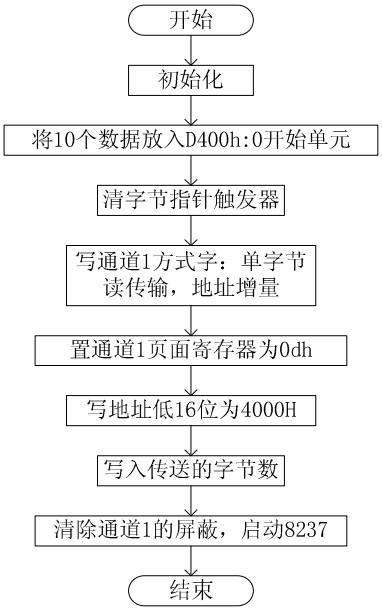


图14-5

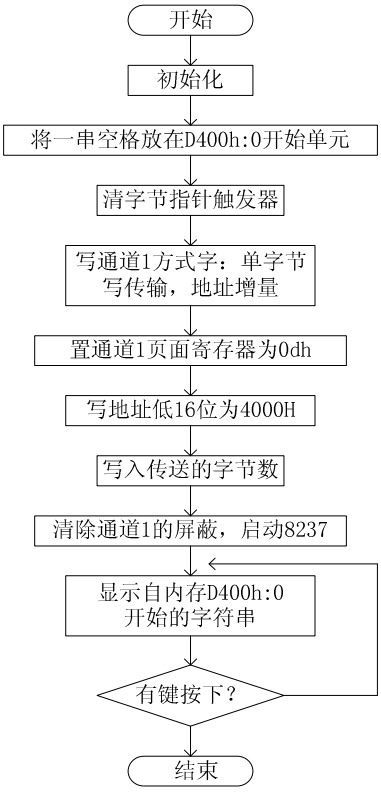


图14-6

五、参考程序1： DMA.CPP

```
/*************************/
/*   DMA传送（块模式）   */
/*************************/
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")

void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
        int addr;
        int addr1;
        BYTE b;
```

```
        BYTE b1;
        BYTE alpha='A';
        BYTE alpha1='a';
    printf("Writing pattern (0xff ~ 0x00)..\n");
    for (addr = 0x00,alpha='A'; addr < 0x100; addr ++)
    {
        //往USB6116空间写入100个A~Z
        MemWriteByte(addr, alpha++);
        //往实验箱扩展6116空间写入100个a
        MemWriteByte(addr+0x2000, alpha1);
        if(alpha>'Z')
        {
            alpha='A';
        }
    }
Sleep(100);
    //显示源地址数据
    for (addr = 0x00; addr < 0x100; addr ++)
    {
    MemReadByte(addr,&b);
    printf("%c ",b);
    if ( (addr + 1) %16 ==0) printf("\n");
    }
printf("..............................\n");
    //显示目的地址数据
for (addr1 = 0x00; addr1 < 0x100; addr1 ++)
    {
    MemReadByte(addr1+0x2000,&b1);
    printf("%c ",b1);
    if ( (addr1 + 1) %16 ==0) printf("\n");
    }
    printf("configuring 8237...\n");
    Sleep(100);
        Write8237(0x08, 0x04);// 关闭，0x08控制寄存器
        Write8237(0x0D, 0x00);// 复位
        Write8237(0x02, 0x00);// 目地址低位
        Write8237(0x02, 0x60);// 目的地址高位41
    Sleep(100);
        Write8237(0x03, 0xFF);// 传输数据长度低位
```

```
        Write8237(0x03, 0x00);// 传输数据低位
        Sleep(100);
        Write8237(0x00, 0x00);// 源地址低位
        Write8237(0x00, 0x40);// 源地址高位
        Sleep(100);
        Write8237(0x0B, 0x85);// mode，传输方式10000101，通道0块传送
        Write8237(0x0B, 0x88);// mode (0)？
        Sleep(100);
        Write8237(0x08, 0x41);// command word，控制字
        Sleep(100);
        Write8237(0x0A, 0x01);// 启动
        Write8237(0x0A, 0x00);
        Sleep(100);
        Write8237(0x09, 0x04);
    for (addr = 0x00; addr < 0x100; addr ++)//显示目的地址数据
    {
        MemReadByte(addr+0x2000, &b);
        printf("%c ",b);
        if ( (addr + 1) %16 ==0) printf("\n");
    }
    Cleanup();
}
```

六、参考程序2： DMA_O.CPP

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
BYTE outdata[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0xff,0x00};
int Dmaout()
{
    int addr;
    printf("Writing pattern (0xff ~ 0x00)..\n");
    for (addr = 0; addr < 10; addr ++)
    {
        MemWriteByte(addr, outdata[addr]);
    }
    printf("configuring 8237...\n");
    Write8237(0x0C, 0x00);      // clear
    Write8237(0x0b, 0x49);      // mode
```

```
    Write8237(0x02, 0x00);      // address
    Write8237(0x02, 0x40);
    Write8237(0x03, 0x00);      //count
    Write8237(0x03, 0x0a);
    Write8237(0x0A, 0x01);      // clear mask
    Sleep(100);
    return 1;
}
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    Dmaout();
    Cleanup();
}
```

七、参考程序3： DMA_I.CPP

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
BYTE indata[]={0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x0d,0x0a};
int Dmain()
{
    int addr;
    printf("Writing pattern (0xff ~ 0x00)..\n");
    for (addr = 0; addr < 10; addr ++)
    {
        MemWriteByte(addr, indata[addr]);
    }
    printf("configuring 8237...\n");
    Write8237(0x0C, 0x00);// clear
    Write8237(0x0b, 0x45);// mode
    Write8237(0x02, 0x00);// address
    Write8237(0x02, 0x40);
    Write8237(0x03, 0x00);//count
    Write8237(0x03, 0x07);
```

```
    Write8237(0x0A, 0x01);// clear mask
    Sleep(100);
    for(;;)
    {
        for (addr = 0; addr < 8; addr ++)
        {
            MemReadByte(addr, &indata[addr]);
        }
        printf("%s",indata);
    }
    return 1;
}
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    Dmain();
    Cleanup();
}
```

实验十五　集成电路测试

一、实验目的

通过对四与非门(74LS00)集成电路芯片的测试,了解测试一般数字集成电路方法，进一步熟悉可编程并行接口8255 的使用。

二、实验内容

1、按图15-1连接电路，其中74LS00插在通用插座上。

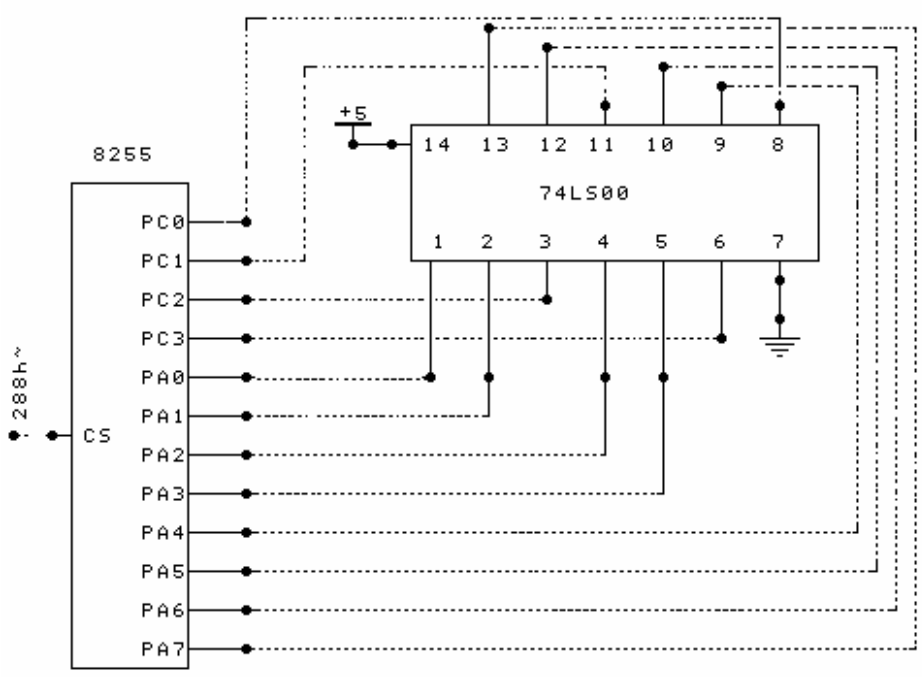


图15-1 集成电路测试电路图

2、编程提示:

将8255设置在方式0下工作，使A口输出，C口输入，先后通过A口分别给每一个门电路送入四个测试信号(00、01、10、11)，相应从C 口读出每一个门电路的输出结果，与正常值(1、1、1、0)进行比较，若相等，则此芯片好，否则为坏。

三、参考流程图　（见图15-2）

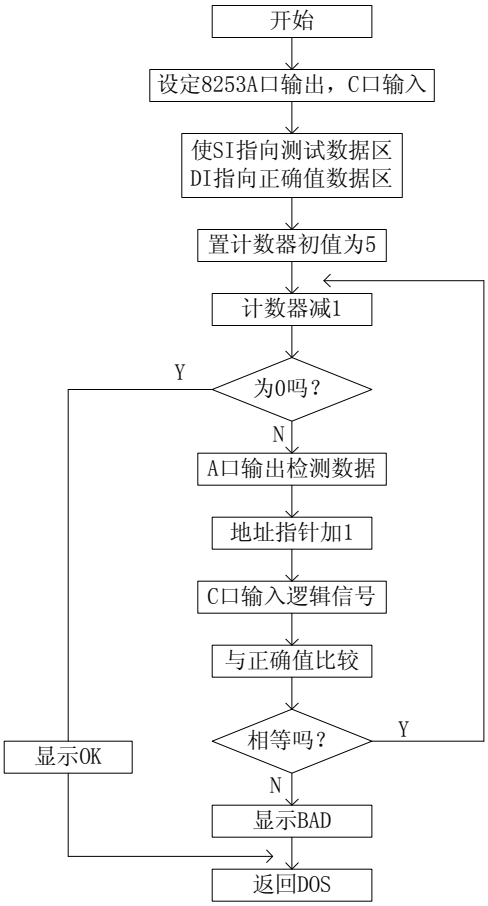


图15-2 集成电路测试流程图

四、参考程序： JC.CPP

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    int i;
    BYTE    data;
    int send[4]={0xff,0xaa,0x55,0x0};
    int reci[4]={0x0,0xf,0xf,0xf};
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
```

```
    PortWriteByte(0x28b,0x89);
    for(i=0;i<4;i++)
    {
        PortWriteByte(0x288,send[i]);
        PortReadByte(0x28a,&data);
        if(data!=reci[i])
        {
            printf("THE CHIP IS BAD!\7\n");
            exit(0);
        }
    }
    Cleanup();
    printf("THE CHIP IS OK!\7\n");
}
```

实验十六　电子琴

一、实验目的

1、通过8253产生不同的频率信号，使PC机成为简易电子琴。

2、了解利用8255和8253产生音乐的基本方法。

二、实验内容

实验电路如图16-1，8253的CLK0接1MHZ时钟，GATE0接8255的PA1，OUT0和8255的PA0接到与门的两个输入端，K8跳线连接喇叭，编程使计算机的数字键1、2、3、4、5、6、7作为电子琴按键，按下即发出相应的音阶。

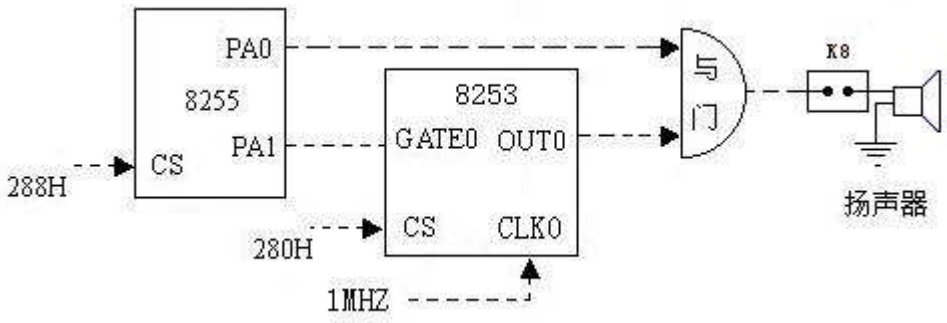


图16-1 电子琴电路

三、编程提示:

1、利用8255的PA0口来施加控制信号给与门，用来控制扬声器的开关状态。再利用设置不同的计数值，使8253产生不同频率的波形，使扬声器产生不同频率的音调，达到类似与音阶的高低音变换。对于音乐，每个音阶都有确定的频率。

各音阶标称频率值:

音 阶	1	2	3	4	5	6	7	$\dot{1}$
低频率(单位:Hz)	262	294	330	347	392	440	494	524
高频率(单位:Hz)	524	588	660	698	784	880	988	1048

四、参考流程图 （见图16-2）

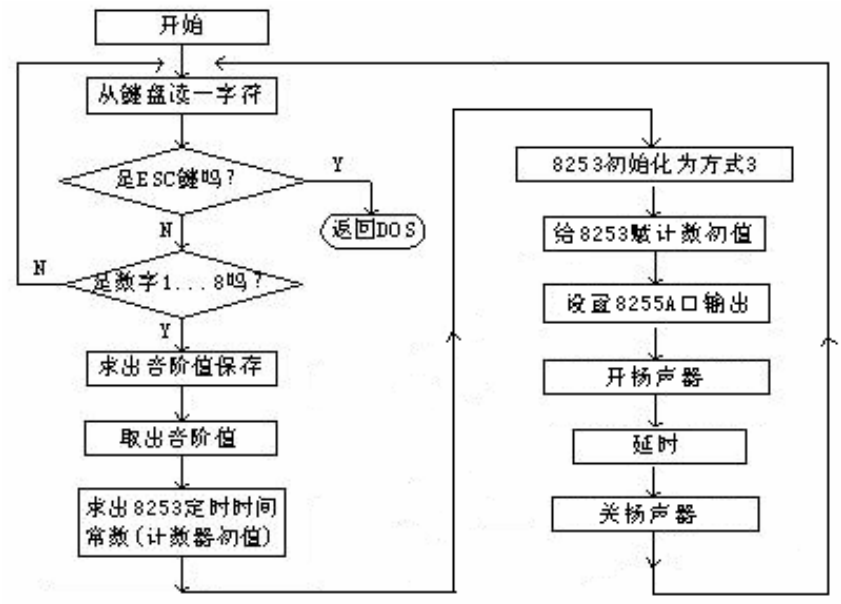


图16-2 主程序

四、**参考程序:** DZQ.cpp

```
/**********************************/
/*        电子琴 (低频率)          */
/**********************************/
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")

unsigned short time[]={0xf5,0x4c,0xd9,0x2f,0xf7,0xe0,0xe8,0x99};//计数值的低位
unsigned short time1[]={0x0e,0x0d,0x0b,0x0b,0x9,0x8,0x7,0x7};//计数值的高位
//{3829,3404,3033,2863,2551,2272,2024,1945};//*8253发不同音的计数器初值，将其换算
//为以上16进制
//1000000/各音阶标称频率值＝计数初值
void de_lay(unsigned short i,unsigned short j);
void main()
{
```

```
    int i,k;
    if(!Startup())      /*打开设备*/
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Press 1,2,3,4,5,6,7,8\n");
    printf("Press other key to exit!\n");

            while((k=getch())!=27)
                {
                if(0x31< k <0x38) //break;
                    {
                        PortWriteByte(0x283,0x36);//00010110,8253控制字,分高低位
传送
                        Sleep(10);
                        de_lay(time[k-0x31],time1[k-0x31]);
                        Sleep(10);
                        PortWriteByte(0x28b,0x80);//10000000, 8255控制字
                        Sleep(10);
                        PortWriteByte(0x288,0x03);//设置8255A口, 开扬声器
                        Sleep(500);             /*延时*/
                        PortWriteByte(0x288,0x00);//设置8255A口, 关扬声器
                    }
                }
    Cleanup();      /*关闭设备*/
}

void de_lay(unsigned short i,unsigned short j)
{
    PortWriteByte(0x280,i);     /*输出计数值低位*/
    Sleep(50);
    PortWriteByte(0x280,j);     /*输出计数值高位*/
}
```

实验十七　8250串行通讯实验

一、实验目的

1、进一步了解串行通信的基本原理。

2、掌握串行接口芯片8250的工作原理和编程方法。

二、实验内容

1、按图17-1连接线路，图中8250芯片插在通用插座上。

2、编程:从键盘输入一个字符，将其ASCII码加1后发送出去，再接收回来在屏幕上加1后的字符显示出来，实现自发自收。

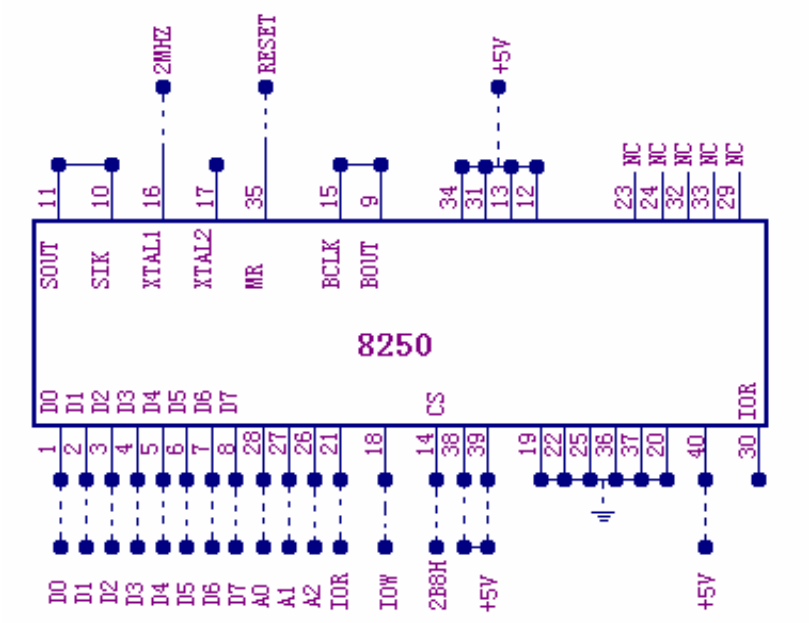


图17-1

三、实验提示

1、8250介绍：

INC8250是一个可编程序异步通讯单元芯片，在微机系统中起串行数据的输入输出接口作用。此外，它还包含有可编程序波特率发生器，它可用1～65535的因子对输入时钟进行分频，以产生波特率十六倍的输入输出时钟。

2、8250时钟接2MHZ，若选波特率为9600，波特率因子为16，则因子寄存器中分频数为13。所以因子寄存器低字节送13，高字节为00H。

3、图中CS接02B8H～02BFH:

下表为各寄存器选择地址一览表。表中DLAB为线控制寄存器的最高位，也叫因子寄存器存取位。当DLAB为0时选接收数据缓冲器，发送数据寄存器和中断允许寄存器。当DLAB为1时选因子寄存器的低字节和高字节。

DLAB	A2	A1	A0	选中寄存器
0	0	0	0	接收缓冲器（读）发送保持寄存器（写）
0	0	0	1	中断允许寄存器
X	0	1	0	中断标志寄存器（仅用于读）
X	0	1	1	线控制寄存器
X	1	0	0	MODEM控制寄存器
X	1	0	1	线状态寄存器
X	1	1	0	MODEM状态寄存器
X	1	1	1	无
1	0	0	0	因子寄存器（低字节）
1	0	0	1	因子寄存器（高字节）

4、收发采用查询方式。

四、参考流程

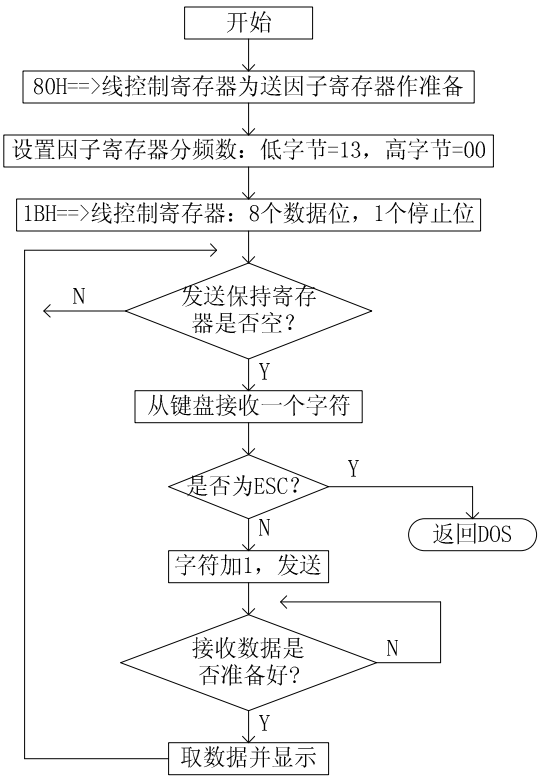


图17-2

五、参考程序：E8250.ASM

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
#define ioport  0x2b8
```

```
void main()
{
    BYTE    data;
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    PortWriteByte(ioport+3,0x80);
    Sleep(1*100);
    PortWriteByte(ioport,0x0d);
    Sleep(1*100);
    PortWriteByte(ioport+1,0x00);
    Sleep(1*100);
    PortWriteByte(ioport+3,0x1b);
    Sleep(1*100);
    PortWriteByte(ioport+1,0x00);
    Sleep(1*100);
    printf("You can play a key on the keybort!ESC is exit!\n");
    for(;;)
    {
        do{
            PortReadByte(ioport+5,&data);
          }while(!(data & 0x20));
        data = getch();
        if(data == 0x1b) exit(0);
        putchar(data);
        PortWriteByte(ioport,data+1);
        Sleep(1*100);
        do{
            PortReadByte(ioport+5,&data);
          }while(!(data & 0x01));
        PortReadByte(ioport,&data);
        putchar(data);
    }
    Cleanup();
}
```

实验十八　步进电机控制实验

一、实验目的

1、了解步进电机控制的基本原理。

2、掌握控制步进电机转动的编程方法。

二、实验内容

1、按图18-1连接线路，利用8255输出脉冲序列，开关K0～K6控制步进电机转速，K7控制步进电机转向。8255 CS接288H～28FH。PA0～PA3接BA～BD；PC0～PC7接K0～K7。

2、编程:当K0～K6中某一开关为“1”(向上拨)时步进电机启动。K7向上拨电机正转，向下拨电机反转。

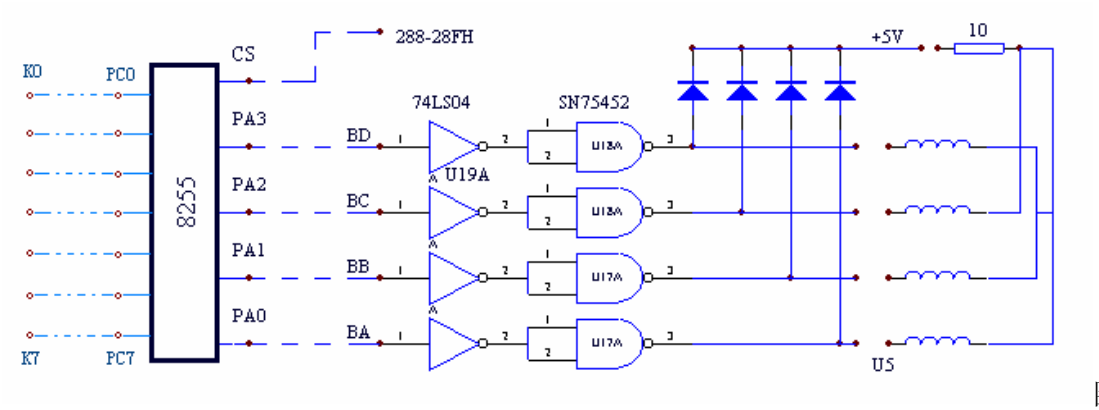


图18-1

三、实验说明

步进电机驱动原理是通过对每相线圈中的电流的顺序切换来使电机作步进式旋转。驱动电路由脉冲信号来控制，所以调节脉冲信号的频率便可改变步进电机的转速。

如图18-2所示：本实验使用的步进电机用直流＋5V电压，每相电流为0.16A，电机线圈由四相组成，即:Φ1(BA)； Φ2(BB)； Φ3(BC)； Φ4(BD)

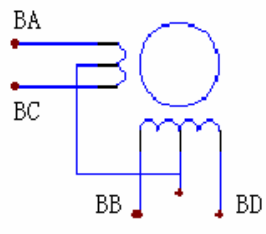


图19-2

驱动方式为二相激磁方式，各线圈通电顺序如下表。

顺序 \ 相	Φ 1	Φ 2	Φ 3	Φ 4
0	1	1	0	0
1	0	1	1	0
2	0	0	1	1
3	1	0	0	1

反时针方向回转

正时针方向回转

表中首先向φ1线圈－φ2线圈输入驱动电流，接着φ2－φ3，φ3－φ4，φ4－φ1，又返回到φ1－φ2，按这种顺序切换，电机轴按顺时针方向旋转。

实验可通过不同长度延时来得到不同频率的步进电机输入脉冲，从而得到多种步进速度。

四、参考流程图

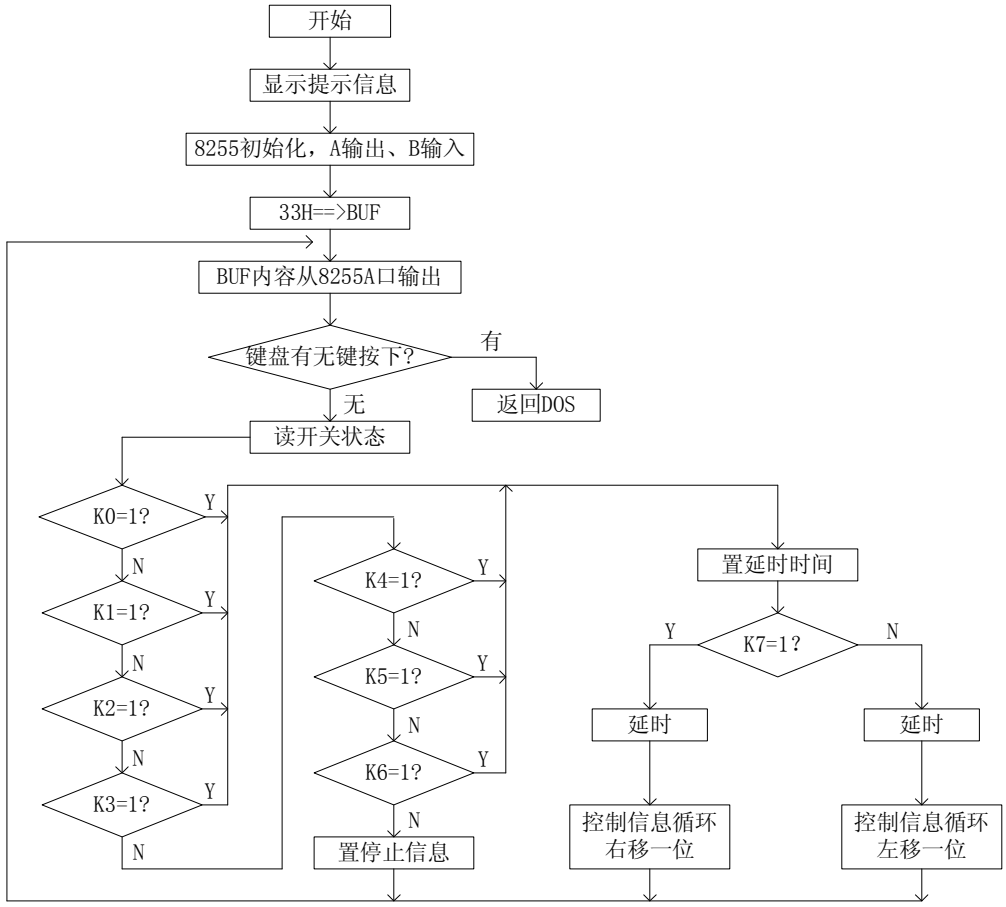


图18-3

五、参考程序： BJDJ.ASM

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    BYTE    data;
    int buf = 0x33,d;
    printf("Press any key to begin!\n\n");
    getch();
    printf("press any key to return! \n");
```

```
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    PortWriteByte(0x28b,0x8b);
    while(!kbhit())
    {
        PortReadByte(0x28a,&data);
        if (data & 1) d = 400;
        else if (data & 2) d =  200;
        else if (data & 4) d =  100;
        else if (data & 8) d =  80;
        else if (data & 16) d = 40;
        else if (data & 32) d = 20;
        else if (data & 64) d = 10;
        else d = 0;
        if (d != 0)
        {
            Sleep(d);
            if (data & 128)
                buf = ((buf&1)<<7)|(buf>>1);
            else
                buf = ((buf&128)>>7)|(buf<<1);
            PortWriteByte(0x288,buf);
        }
        else
            PortWriteByte(0x288,0xff);
    }
    Cleanup();
}
```

实验十九　小直流电机转速控制实验

一、实验目的

1、进一步了解DAC0832的性能及编程方法。

2、了解直流电机控制的基本方法。

二、实验内容

1、按图19-1线路接线。DAC0832的CS接290H～297H，Ub接DJ插孔，8255 CS接288H～28FH。

2、编程利用DAC0832输出一串脉冲，经放大后驱动小直流电机，利用开关K0～K5控制改变输出脉冲的电平及持续时间，达到使电机加速，减速之目的。

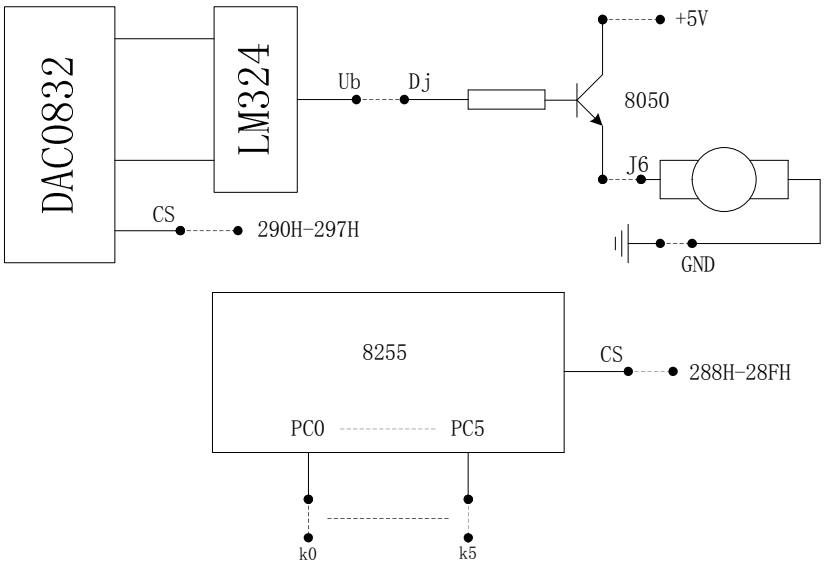


图19-1

三、实验原理简述

小直流电机的转速是由Ub输出脉冲的占空比来决定的，正向占空比越大转速越快，反之越慢。见下图19-2:

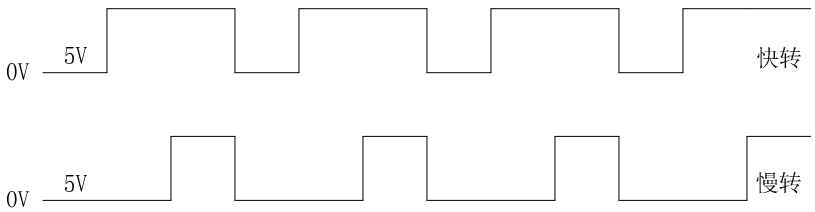


图19-2

在本实验中，模拟量输出Ub为双极性，当输入数字量小于80H时输出为负，输入等于80H时为0V，输入大于80H时输出为正。因而本实验中，DAC0832输入数字量只有2个(80H和FFH)，通过不同的延迟时间达到改变小电机转速的目的。

四、参考程序框图

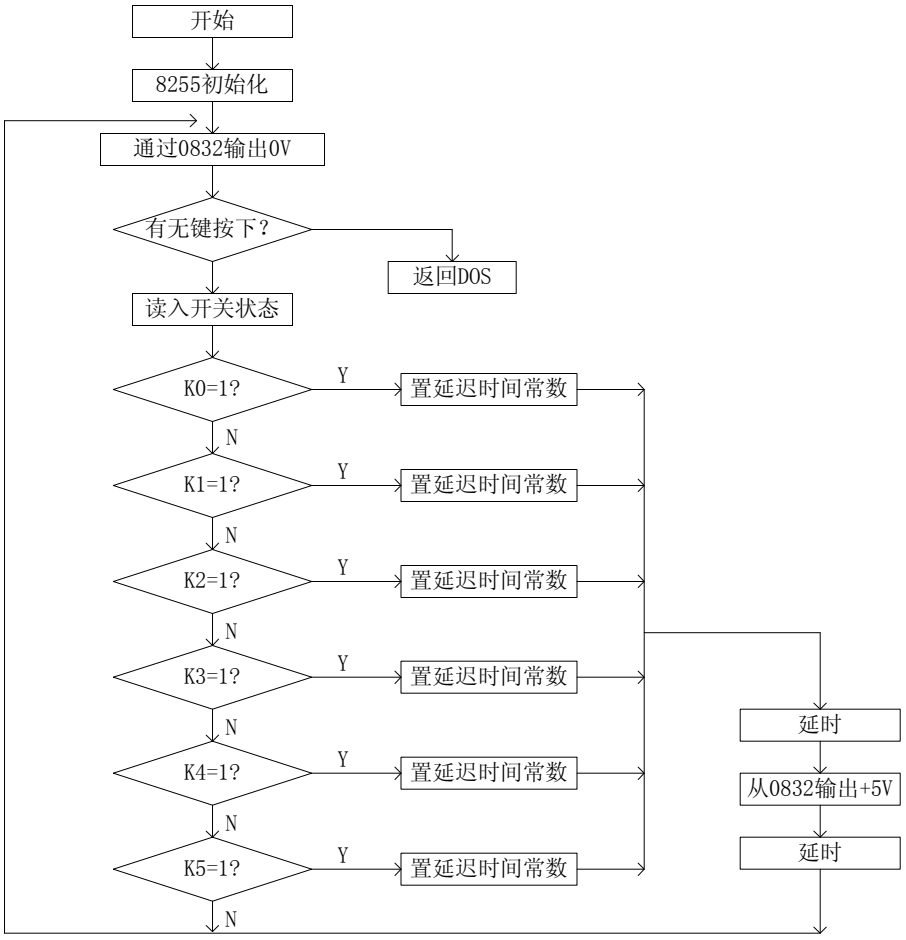


图19-3

五、参考程序： ZLDJ.ASM

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
void main()
{
    BYTE    data;
    int d;
    printf("Press any key to begin!\n\n");
    getch();
    printf("press any key to return!\n");
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
```

```
        return;
    }
    PortWriteByte(0x28b,0x8b);
    while(!kbhit())
    {
        PortWriteByte(0x290,0x80);
        PortReadByte(0x28a,&data);
        if (data & 1) d = 15;
        else if (data & 2) d = 25;
        else if (data & 4) d = 35;
        else if (data & 8) d = 50;
        else if (data & 16) d = 70;
        else if (data & 32) d = 80;
        else d = 0;
        Sleep(200);
        PortWriteByte(0x290,0xff);
        Sleep(d);
    }
    Cleanup();
}
```

实验二十　　键盘显示控制器实验

一、实验目的

1、掌握8279键盘显示电路的基本功能及编程方法。

2、掌握一般键盘和显示电路的工作原理。

3、进一步掌握定时器的使用和中断处理程序的编程方法。

二、实验内容

1、本实验的实验电路如图20-1，它做在一块扩展电路板上，用一根20芯扁平电缆与实验台上扩展插头J7相连。

2、编程1:使得在小键盘上每按一个健，6位数码管上显示出相应字符，它们的对应关系如下:

小键盘		显示	小键盘		显示
0	—	0	C	—	C
1	—	1	D	—	d
2	—	2	E	—	E
3	—	3	F	—	F
4	—	4	G	—	q

5	—	5	M	—	冂
6	—	6	P	—	p
7	—	7	W	—	凵
8	—	8	X	—	丨丨
9	—	9	Y	—	乛
A	—	冃	R	—	返回
B	—	b			

3、编程2:中断编程

利用实验台上提供的定时器8253和扩展板上提供的8279以及键盘和数码显示电路，设计一个电子钟。由8253中断定时，小键盘控制电子钟的启停及初始值的预置。

电子钟显示格式如下：

XX. XX. XX.　　由左向右分别为时、分、秒

要求具有如下功能：

①、C键:清除，显示全零。

②、G键:启动，电子钟计时。

③、D键:停止，电子钟停止计时。

④、P键:设置时、分、秒值。输入时依次为时、分、秒，同时应有判断输入错误的能力，若输入有错，则显示:E－－－－－－。此时敲P键可重新输入预置值。

⑤、E键:程序退出。

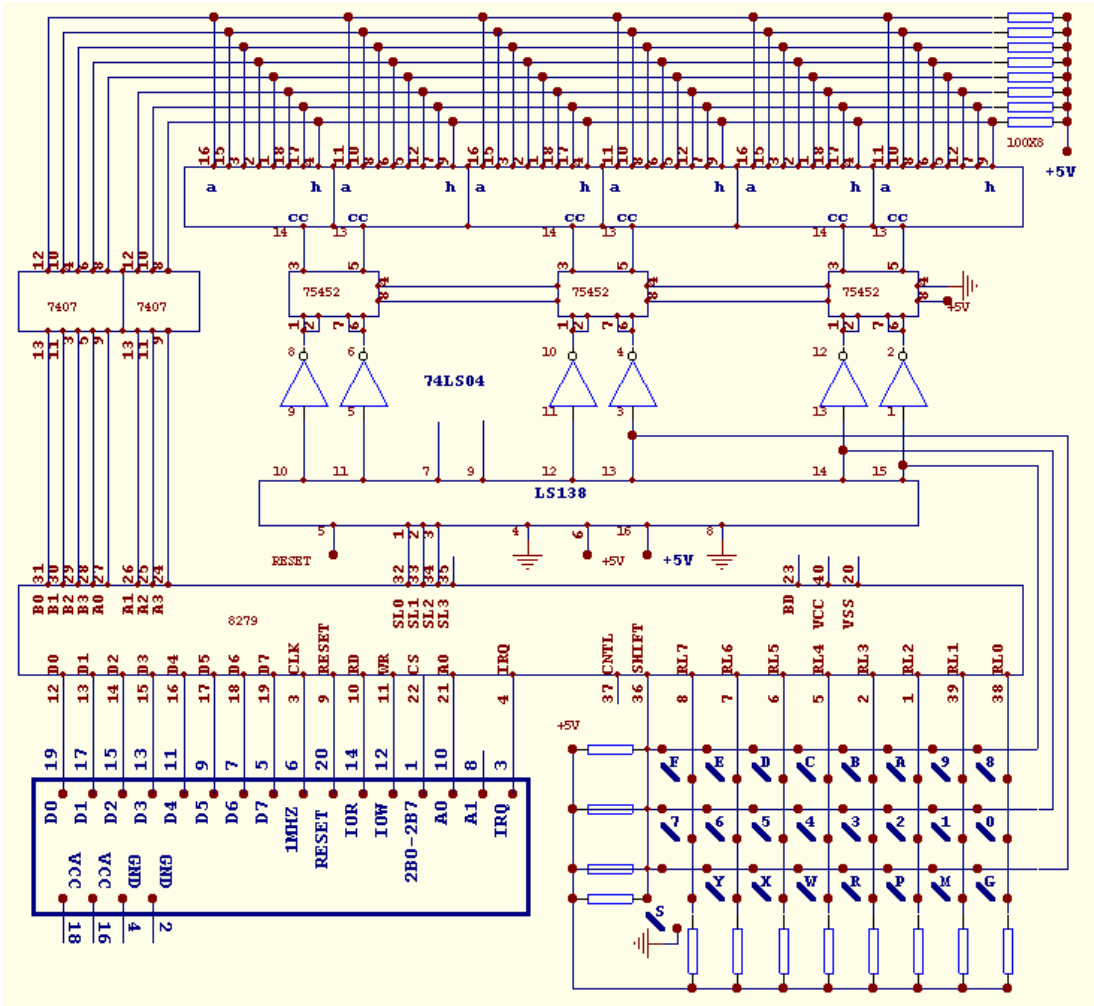


图20-1

三、编程1 接线方法

用一根20芯扁平电缆将实验扩展板与实验台上扩展插头J7相连。

四、编程2 接线方法

1、用一根20芯扁平电缆将实验扩展板与实验台上扩展插头J7相连。

2、实验台上8253 CLK0 接 1MHZ，GATE0 和 GATE1接＋5V，OUT0 接 CLK1，OUT1 接 IRQ，CS接 280H～287H。

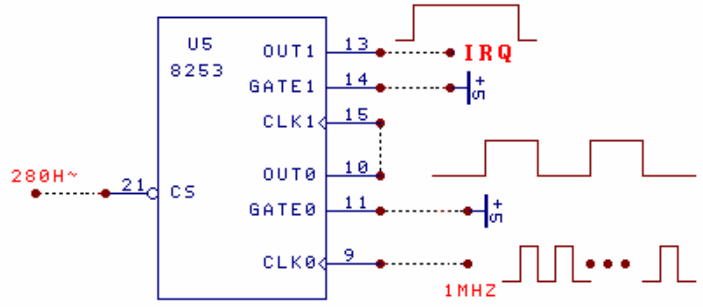
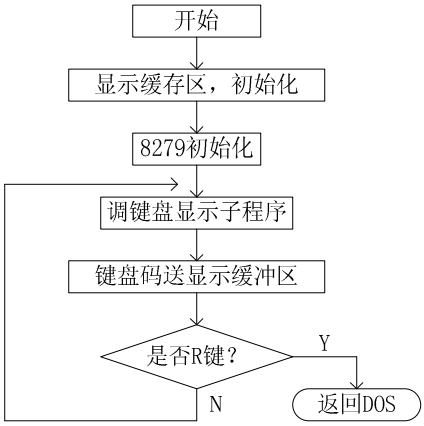
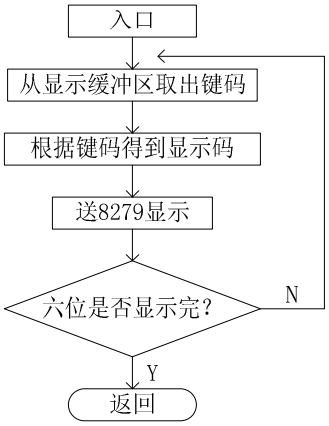


图20-2

五、编程1 参考流程



主程序流程图:



显示子程序流程图:DISP

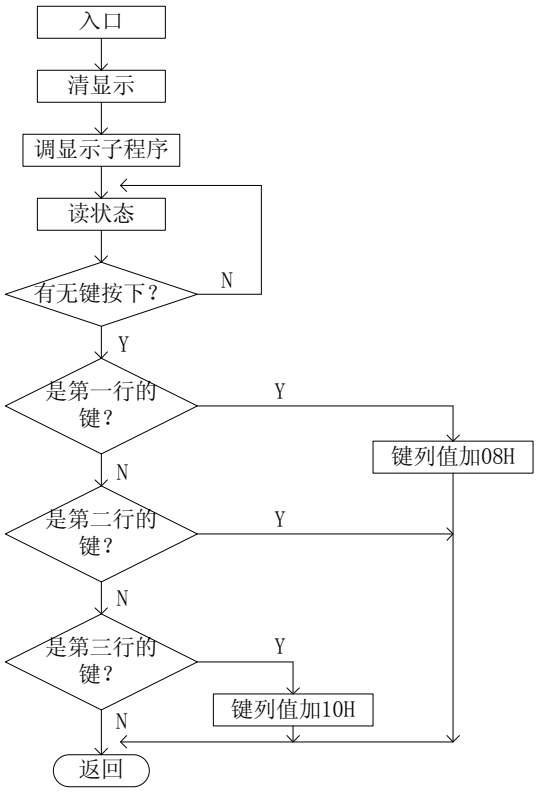


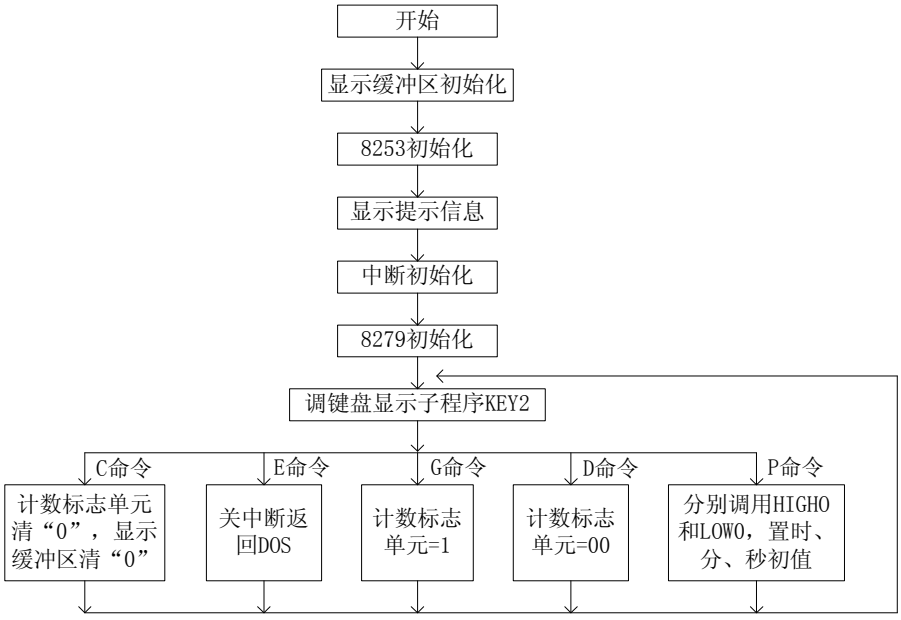
图20-3 键盘显示子程序流程图:KEY2

六、编程1 参考程序：JPXSH1.CPP

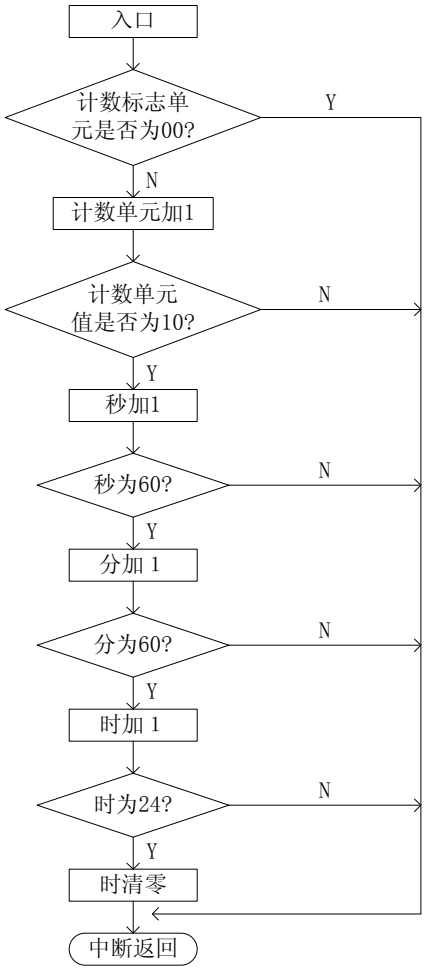
```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
#define ioport  0x2b0       //8279 数据口
#define ioport1 0x2b1       //8279 控制口
int led[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,
        0x7C,0x39,0x5E,0x79,0x71,0x67,0x37,0x73,0x31,0x3E,0x36,0x66,0x80};
int keyin=0;
void key();
void disp();
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    printf("Press small keybord,'R' is exit!\n");
    PortWriteByte(ioport1,0xd3);
    PortWriteByte(ioport1,0x2a);
    PortWriteByte(ioport1,0x40);
    PortWriteByte(ioport1,0x00);
    PortWriteByte(ioport1,0x80);
    do{
        key();
      }while(!(keyin==0x13));
    Cleanup();
}
void key()
{
    BYTE    data;
    PortWriteByte(ioport1,0xd1);
    Sleep(100);
    do{
        disp();
        PortReadByte(ioport1,&data);
      }while(!(data&0x07));
```

```
    PortReadByte(ioport,&data);
    keyin = data & 0x07;
    data = data & 0x38;
    data>>=3;
    if(data==0)
        keyin = keyin+0x08;
    else if(data!=1)
        keyin = keyin+0x10;
}
void disp()
{
    int i;
    PortWriteByte(ioport1,0x90);
    for(i=0;i<6;i++)
    {
        PortWriteByte(ioport,led[keyin]);
    }
}
```

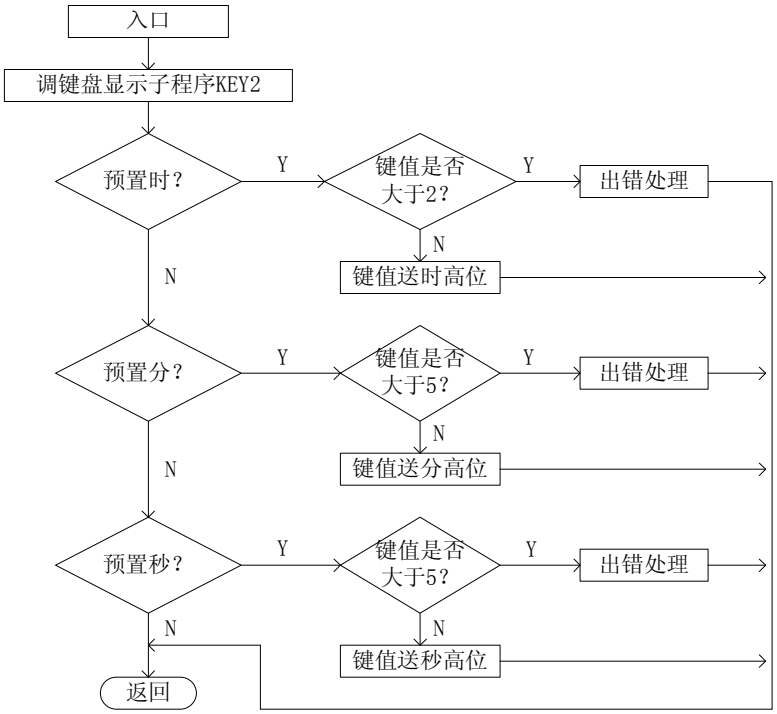
七、编程2 参考流程



主程序流程图:


入口
计数标志单元是否为00?
Y
N
计数单元加1
计数单元值是否为10?
N
Y
秒加1
秒为60?
N
Y
分加 1
分为60?
N
Y
时加 1
时为24?
N
Y
时清零
中断返回

中断处理子程序:



预置时、分、秒高位子程序HIGH0:

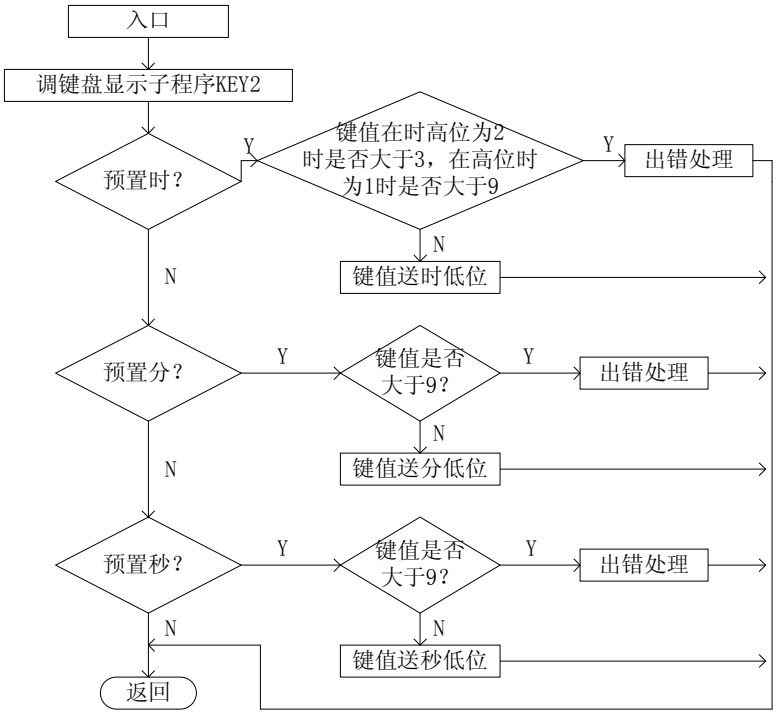


图20-4 预置时、分、秒低位子程序LOW0:

八、编程2 参考程序： JPXSH.CPP

```
#include <stdio.h>
#include <conio.h>
```

```
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
#define ioport  0x2b0
#define ioport1 0x2b1
#define time    0x280
#define time1   0x281
#define timec   0x283
int buf=0;
int sign=1;
int sec1=0,sec2=0;
int min1=0,min2=0;
int hour1=0,hour2=0;
int err1=0;
int hms;
int led[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,
        0x39,0x5E,0x79,0x71,0x67,0x37,0x73,0x31,0x3E,0x36,0x66,0x80,0x40};
int keyin=0;
void key();
void disp();
void high();
void low();
void int_proc();
void error();
void main()
{
    printf("Press any key to begin!\n\n");
    getch();
    printf("Press small keybord:\n");
    printf("C--CLEAR TO ZERO;\n");
    printf("G--GO AHEAD\n");
    printf("D--STOP THE DISPLY;\n");
    printf("P--POSITION THE BEGINNING TIME\n");
    printf("E--EXIT\n");
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    PortWriteByte(timec,0x36);
```

```
PortWriteByte(time,1000%256);
PortWriteByte(time,1000/256);
PortWriteByte(timec,0x74);
PortWriteByte(time1,100%256);
PortWriteByte(time1,100/256);
PortWriteByte(ioport1,0xd3);
PortWriteByte(ioport1,0x2a);
PortWriteByte(ioport1,0x40);
PortWriteByte(ioport1,0x00);
PortWriteByte(ioport1,0x80);
RegisterLocalISR(int_proc);
EnableIntr();
do{
    key();
    if((hour2==0x0a)|(keyin==0x0c))
    {
        sign = 0;
        hour2 = 0;
        hour1 = 0;
        min2 = 0;
        min1 = 0;
        sec2 = 0;
        sec1 = 0;
    }
    if(keyin==0x10)
    {
        sign = 0x01;
    }else if(keyin==0x0d)
    {
        sign = 0x00;
    }else if(keyin==0x12)
    {
        sign = 0x00;
        hms = 0x00;
        high();
        if(err1!=0x01)
        {
            low();
            if(err1!=0x01)
```

```
                {
                    hms = 0x11;
                    high();
                    if(err1!=0x01)
                    {
                        low();
                        if(err1!=0x01)
                        {
                            hms = 0x22;
                            high();
                            if(err1!=0x01)
                            {
                                low();
                            }
                        }
                    }
                }
            }
        }
    }while(!(keyin==0x0e));
    DisableIntr();
    Cleanup();
}
void int_proc()
{
    if(sign!=0)
    {
        buf++;
        if(buf==10)
        {
            buf = 0;
            sec1++;
            if(sec1==10)
            {
                sec1 = 0;
                sec2++;
                if(sec2==6)
                {
                    sec2 = 0;
```

```
                    min1++;
                    if(min1==10)
                    {
                        min1 = 0;
                        min2++;
                        if(min2==6)
                        {
                            min2 = 0;
                            hour1++;
                            if((hour1==4)&(hour2==2))
                            {
                                hour2 = 0;
                                hour1 = 0;
                            }
                            if(hour1==10)
                            {
                                hour1 = 0;
                                hour2++;
                            }
                        }
                    }
                }
            }
        }
    }
}
void key()
{
    BYTE    data;
    PortWriteByte(ioport1,0xd1);
    Sleep(100);
    do{
        disp();
        PortReadByte(ioport1,&data);
    }while(!(data&0x07));
    PortReadByte(ioport,&data);
    keyin = data & 0x07;
    data = data & 0x38;
    data>>=3;
```

```
    if(data==0)
        keyin = keyin+0x08;
    else if(data!=1)
        keyin = keyin+0x10;
}
void disp()
{
    PortWriteByte(ioport1,0x90);
    Sleep(100);
    PortWriteByte(ioport,led[sec1]);
    PortWriteByte(ioport,led[sec2]);
    PortWriteByte(ioport,led[min1]);
    PortWriteByte(ioport,led[min2]);
    PortWriteByte(ioport,led[hour1]);
    PortWriteByte(ioport,led[hour2]);
}
void high()
{
    key();
    err1 = 00;
    if(hms==0x00)
    {
        if(keyin<=0x02)
        {
            hour2 = keyin;
        }else
        {
            error();
        }
    }else if(hms==0x11)
    {
        if(keyin<=0x05)
        {
            min2 = keyin;
        }else
        {
            error();
        }
    }else if(keyin<=0x05)
```

```
    {
        sec2 = keyin;
    }else
    {
        error();
    }
}
void error()
{
    err1 = 0x01;
    hour2 = 0x0e;
    hour1 = 0x18;
    min2 = 0x18;
    min1 = 0x18;
    sec2 = 0x18;
    sec1 = 0x18;
}
void low()
{
    key();
    err1 = 00;
    if(hms==0x00)
    {
    if(((keyin<=0x09)&(hour2==0))|((keyin<=0x09)&(hour2==1))|((keyin<=0x03)&
        (hour2==2)))
        {
            hour1 = keyin;
        }else
        {
            error();
        }
    }else if(hms==0x11)
    {
        if(keyin<=0x09)
        {
            min1 = keyin;
        }else
        {
            error();
```

```
        }
    }else if(keyin<=0x09)
    {
        sec1 = keyin;
    }else
    {
        error();
    }
}
```

实验二十一　　存储器读写实验

一、实验目的

1、熟悉6116静态RAM的使用方法，掌握PC机外存扩充的手段。

2、通过对硬件电路的分析，学习了解总线的工作时序。

二、实验内容

1、硬件电路如下:

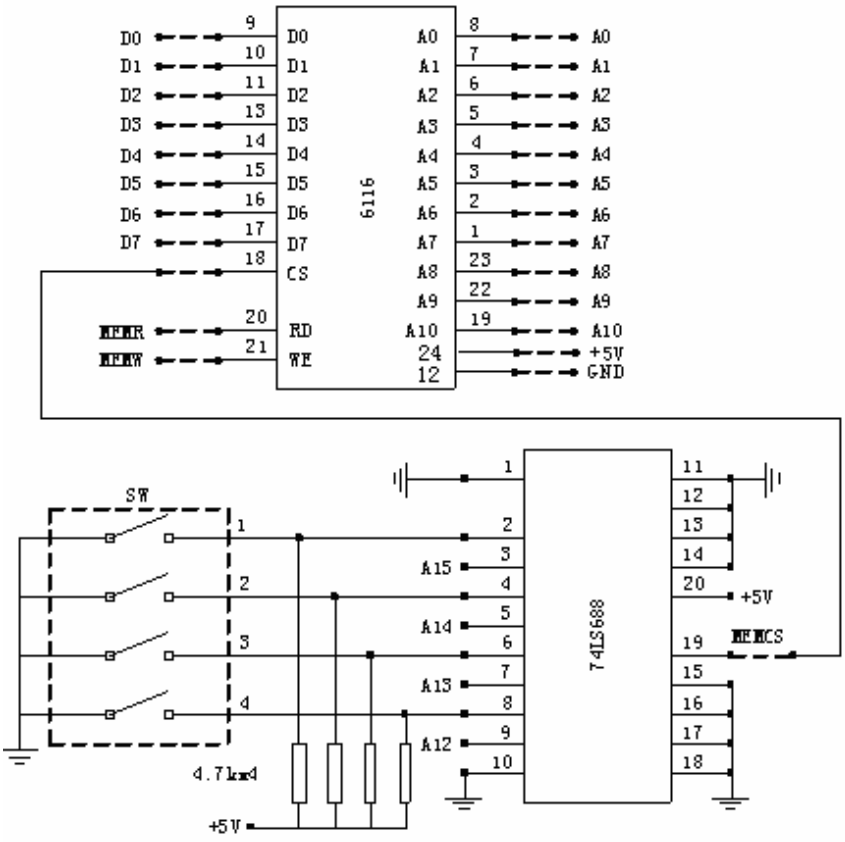


图21-1

2、编制程序，将字符A-Z循环写入扩展的6116RAM中，然后再将6116的内容读出来显示在主机屏幕上。

三、编程提示

1、注意:TPC-USB已为扩展的6116指定了段地址:0D000H。

2、TPC-USB模块外扩储器的默认起始地址为D4000H，即在程序中如访问D6000H，只需通过零地址addr=0，再加上2000H即可，即addr+2000H。（注TPC-USB模块外扩存储器的范围为0D4000H-0D7fffH。

3、通过片选信号的产生方式，确定6116RAM在PC机系统中的地址范围。因为段地址已指定，所以其地址为CS=A15 and A14 and A13 and A12，实验台上设有地址选择微动开关K2，拨动开关，可以选择4000-7fff的地址范围。编制程序，从0d6000H开始循环写入100h个A-Z。

开关状态如下:

1	2	3	4	地址
ON	OFF	ON	OFF	d4000h
ON	OFF	OFF	ON	d6000h

四、程序框图

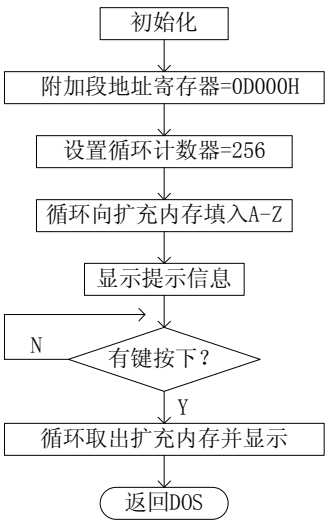


图21-2

五、程序清单： MEMRWEX.CPP

```
#include <stdio.h>
#include <conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
int Memrw()
{
    int addr;
    BYTE b;
    BYTE alpha='A';
    for (addr = 0,alpha='A'; addr < 0x100; addr ++)
```

```
    {
        MemWriteByte(addr+0x2000, alpha++);
        if(alpha>'Z')
        {
            alpha='A';
        }
    }
    Sleep(100);
    printf("please enter a key to show the content of mem!\n");
    getch();
    for (addr = 0; addr < 0x100; addr ++)
    {
    MemReadByte(addr+0x2000, &b);
    printf("%c ",b);
    if ( (addr + 1) %16 ==0) printf("\n");
    }
    if (addr != 0x200) return 0;
    return 1;
}
void main()
{
    if(!Startup())
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
    Memrw();
    Cleanup();
}
```

实验二十二　双色点阵发光二极管显示实验

一、实验目的

1、了解双色点阵LED显示器的基本原理。

2、掌握PC机控制双色点阵LED显示程序的设计方法。

二、实验原理

点阵LED显示器是将许多LED类似矩阵一样排列在一起组成的显示器件，双色点阵LED是在每一个点阵的位置上有红绿或红黄或红白两种不同颜色的发光二极管。当微机输出的控制信号使得点阵中有些LED发光，有些不发光，即可显示出特定的信息，包括汉字、图形等。车站

广场由微机控制的点阵LED大屏幕广告宣传牌随处可见。

实验仪上设有一个共阳极8×8点阵的红黄两色LED显示器，其点阵结构如图所示。该点阵对外引出24条线，其中8条行线，8条红色列线，8条黄色列线。若使某一种颜色、某一个LED发光，只要将与其相连的行线加高电平，列线加低电平即可。

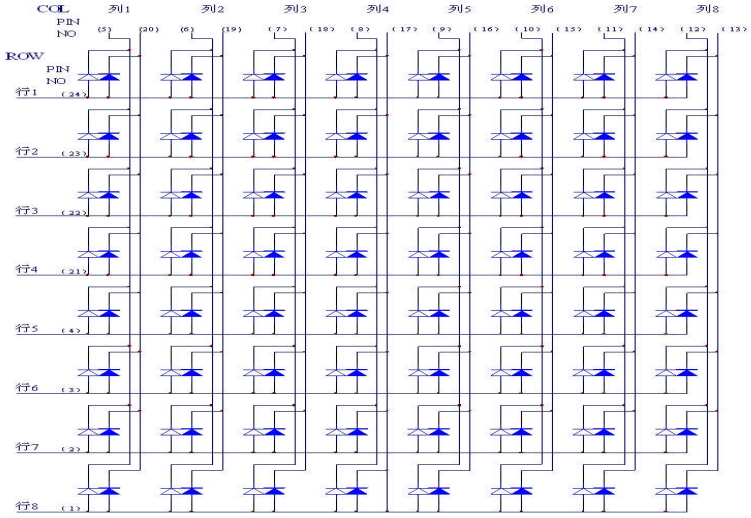


图22-1

例如欲显示汉字“年”，采用逐列循环发光。首先由“年”的点阵轮廓，确定点阵代码(如图所示)根据“年”的点阵代码，确定逐列循环发光的顺序如下:

① 行代码输出 44H； 红色列代码输 01H；第一列2个红色LED发光。

② 行代码输出 54H； 红色列代码输 02H；第二列3个红色LED发光。

③ 行代码输出 54H； 红色列代码输 04H；第三列3个红色LED发光。

④ 行代码输出 7FH； 红色列代码输 08H；第四列7个红色LED发光。

⑤ 行代码输出 54H； 红色列代码输 10H；第五列3个红色LED发光。

⑥ 行代码输出 DCH； 红色列代码输 20H；第六列5个红色LED发光。

⑦ 行代码输出 44H； 红色列代码输 40H；第七列2个红色LED发光。

⑧ 行代码输出 24H； 红色列代码输 80H；第八列2个红色LED发光。

在步骤①～⑧之间可插入几ms的延时，重复进行①～⑧即可在LED上稳定的显示出红色“年”字。若想显示黄色“年”，只需把红色列码改为黄色列码即可。

三、实验内容:

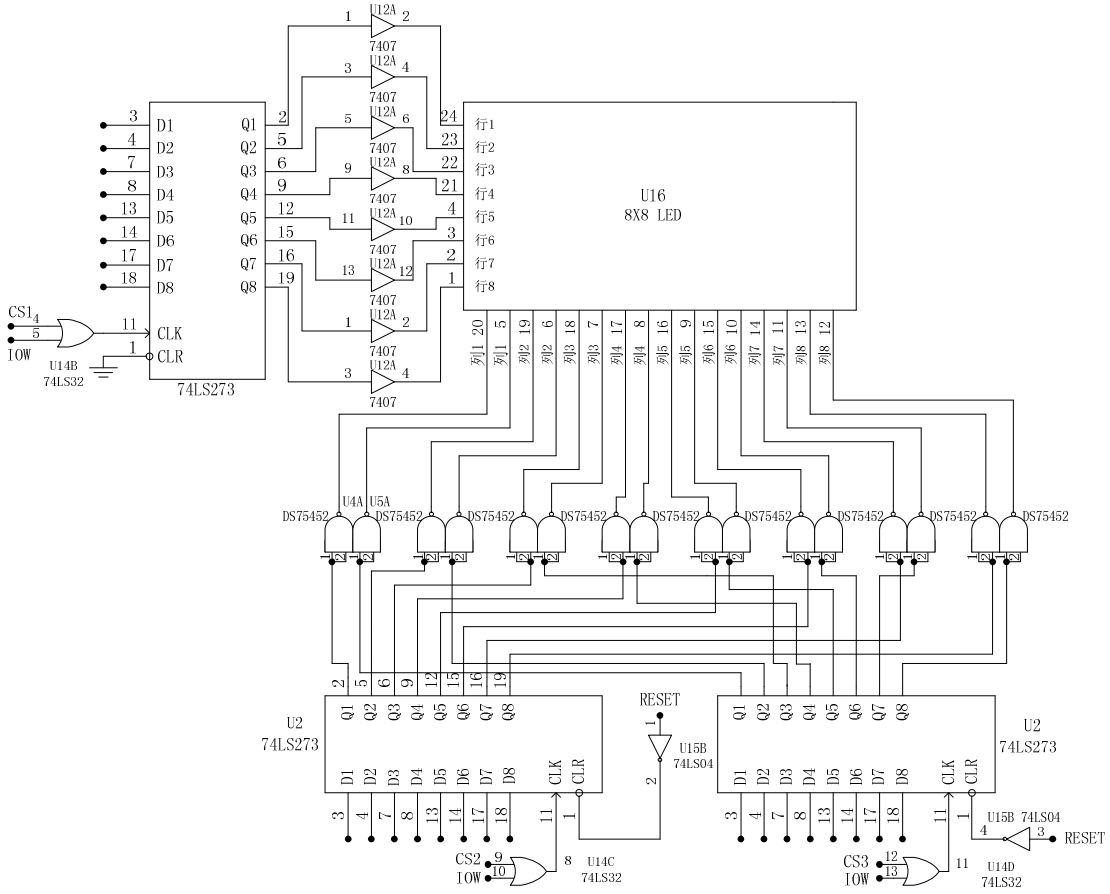


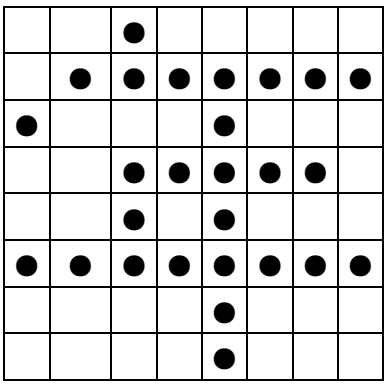
图22-2

1、实验仪上的点阵LED及驱动电路如下图所示，行代码、红色列代码、黄色列代码各用一片74LS273锁存。行代码输出的数据通过行驱动器7407加至点阵的8条行线上，红和黄列代码的输出数据通过驱动器DS75452反相后分别加至红和黄的列线上。行锁存器片选信号为CS1，红色列锁存器片选信号为CS2，黄色列锁存器片选信号为CS3。

2、接线方法:行片选信号 CS1 接 280H；红列片选信号 CS2 接 288H；黄列片选信号 CS3 接 290H。

3、编程重复使LED点阵红色逐列点亮，再黄色逐列点亮，再红色逐行点亮，黄色逐行点亮。

4、编程在LED上重复显示红色“年”和黄色“年”。



四、逐行、逐列显示参考流程图

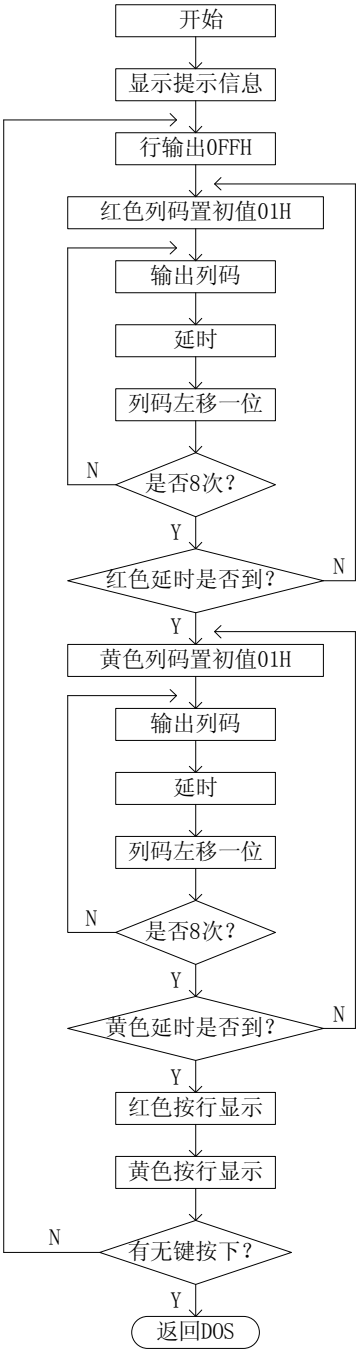


图22-3

五、　　逐行逐列显示参考程序

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
```

```
#define porth 0x280 //行
#define portlr 0x288//红列
#define portly 0x290//黄列
void main()
{
    if(!Startup())          //打开设备
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }
while(!kbhit())
{
    int ah=0x01;                        //红行亮
    for(int c1=0;c1<8;c1++)
    {
        PortWriteByte(porth,0xff);
        PortWriteByte(portlr,ah);
        Sleep(50);
        ah=ah<<1;
    }
    PortWriteByte(portlr,0x00);
    ah=0x01;                            //黄行亮
    for(int c2=0;c2<8;c2++)
    {
        PortWriteByte(porth,0xff);
        PortWriteByte(portly,ah);
        Sleep(50);
        ah=ah<<1;
    }
    PortWriteByte(portly,0x00);
    ah=0x01;                            //红列亮
    for(int c3=0;c3<8;c3++)
    {
        PortWriteByte(portlr,0xff);
        PortWriteByte(porth,ah);
        Sleep(50);
        ah=ah<<1;
    }
```

```
    PortWriteByte(portlr,0x00);
        ah=0x01;                          //黄列亮
    for(int c4=0;c4<8;c4++)
    {
        PortWriteByte(portly,0xff);
        PortWriteByte(porth,ah);
        Sleep(50);
        ah=ah<<1;
    }
    PortWriteByte(portly,0x00);
}
Cleanup();
}
```

六、显示“年”参考流程

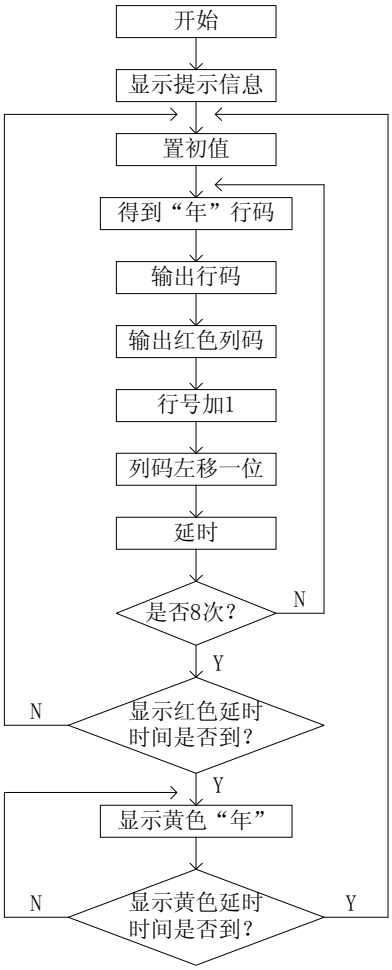


图22-4

七、显示“年”参考程序 11588-1.ASM

```
#include<stdio.h>
#include<conio.h>
#include "..\\ApiEx.h"
#pragma comment(lib,"..\\ApiEx.lib")
#define porth 0x280
#define portlr 0x288
#define portly 0x290
unsigned int buff[]={0x44,0x54,0x54,0x7f,0x54,0x0dc,0x44,0x24};

void main()
{
    if(!Startup())          //开启设备
    {
        printf("ERROR: Open Device Error!\n");
        return;
    }

while(!kbhit())
{
for(int count1=10;count1>=0;count1--)//显示红色“年字”
    {
        int ah=0x01;
        for(int c0=0;c0<8;c0++)
        {
        PortWriteByte(porth,buff[c0]);
        PortWriteByte(portlr,ah);
        Sleep(2);
        PortWriteByte(portlr,0x00);
        ah=ah<<1;
        }
        PortWriteByte(portlr,0x00);
    }

for(int count2=10;count2>=0;count2--)//显示黄色“年字”
    {
        int al=0x01;
        for(int c1=0;c1<8;c1++)
        {
```

```
        PortWriteByte(porth,buff[c1]);
        PortWriteByte(portly,al);
        Sleep(2);
        PortWriteByte(portly,0x00);
        al=al<<1;
        }
        PortWriteByte(portly,0x00);
    }
}
Cleanup();
}
```

附录一、随机光盘实验程序名称表

汇编程序与实现相同/类似功能的C程序文件对照

	汇编程序	说明及对应C程序	
1	Ymq.asm	io地址译码	Ymq.cpp
2	E244.asm	简单并行口244	E244.cpp
3	E273.asm	简单并行口273	E273.cpp
4	E8253_1.asm	可编程定时器_计数器	E8253_1.cpp
5	E8253_2.asm	可编程定时器_计数器	E8253_2.cpp
6	E8255.asm	可编程并行口1_8255方式0	E8255.cpp
7	Led1.asm	七段数码管1	LED1.cpp
8	Led2.asm	七段数码管2	LED2.cpp
9	Led3.asm	七段数码管3	LED3.cpp
10	Jdq.asm	继电器	JDQ.cpp
11	Qdq.asm	竞赛抢答器	QDQ.cpp
12	Jtd.asm	交通灯控制	JTD.cpp
13	Int.asm	中断	INT.cpp
14	E8255_1o.asm	可编程并行口2_8255方式1_1	E8255_1o.cpp
15	E8255_1i.asm	可编程并行口2_8255方式1_2	E8255_1i.cpp
16	Da_1.asm	数模转换1	DA_1.cpp
17	Da_2.asm	数模转换2	DA_2.cpp
18	Ad_1.asm	模数转换1	AD_1.cpp
19	Ad_2.asm	模数转换2	AD_2.cpp
20			
21	E8251.asm	8251串行通信	E8251.cpp
22	Dma.asm	DMA传送1	DMA.cpp
23	Dma_o.asm	DMA传送2	DMA_O.cpp
24	Dma_i.asm	DMA传送3	DMA_I.cpp
25	Jc.asm	集成电路测试	JC.cpp
26	Dzq.asm	电子琴	DZQ.cpp
27	E8250.asm	8250串行通信	E8250.cpp
28	Bjdj.asm	步进电机	BJDJ.cpp
29	Zldj.asm	直流电机	ZLDJ.cpp
30	Jpxsh1.asm	8279 键盘显示控制器1	JPXSH1.cpp
31	Jpxsh.asm	8279 键盘显示控制器2	JPXSH.cpp
32	Memrwex.asm	存储器读写	MEMRWEX.cpp
33	11588.ASM	点阵实验一	11588.cpp
34	11588-1.ASM	点阵实验二	11588-1.cpp

35	KEY.ASM	8255A并行口键盘扫描实验一
36	KEY1.ASM	8255A并行口键盘扫描实验二
37	LED.ASM	总线控制LED显示实验一
38	LED4.ASM	总线控制LED显示实验二
39	KL.ASM	微机接口、键盘、LED综合实验－I/O编程
40	KL1.ASM	微机接口、键盘、LED综合实验－中断编程
41	CHLCD.ASM	字符液晶模块实验一
42	CHLCD1.ASM	字符液晶模块实验二
43	GRLCD.ASM	测试液晶模块及屏幕像素程序的编程
44	GRLCD1.ASM	汉字字符显示编程
45	GRLCD2.ASM	图形显示编程
46	GRLCD3.ASM	特效显示编程
47	LCDCTR.ASM	液晶显示屏手动控制对比度实验
48	LCDCTR1.ASM	液晶显示屏程序控制对比度实验
49	LCDBL.ASM	图形液晶程序控制背光实验
51	BMKEY.ASM	薄膜按键开关实验
52	Wuxian.asm	无线通讯实验
53	Hongwai-q.asm	红外模块实验

附录二：TPC-USB通用微机接口实验系统硬件实验指导（汇编程序）

汇编语言支持winnt/2000/xp。

参照实验指导书安装TPC-USB模块及其驱动程序后才能正常运行程序。

说明：开关K向上为“1”，向下为“0”。

一、I/O地址译码

连线：CD---2A8H-2AFH　SD-- +5V　D-- +5V　Q---L7　CLK---2A0H-2A7H

运行程序：YMQ地址译码

运行结果：发光二极管L7有规律的进行连续闪烁。

二、简单并行接口

（1）将74LS273插在相关插座上

连线：D0-D7---74LS273的（3、4、7、8、13、14、17、18）脚　或门输入A---2A8H-2AFH

或门输入B---/IOW　或门输出Y---11脚　10脚---GND　1、20脚-- +5V

L0-L7---74LS273的（2、5、6、9、12、15、16、19）脚

运行程序：E273简单并行口程序

运行结果：拨动开关K，相应置“1”的开关所对应的灯L亮，否则灭。

（2）将74LS244插在相关插座上

连线：K0-K7---74LS244（2、4、6、8、11、13、15、17）脚　或门输入A---/IOR

或门输入B---2A0H-2A7H　或门输出Y---1、19脚　10脚---GND　20脚-- +5V

D0-D7---74LS244（18、16、14、12、9、7、5、3）脚

运行程序：E244简单并行口程序

运行结果：用开关输入字母的ASCII码值，在屏幕上显示对应的字母。

三、可编程定时器/计数器

（1）连线：CLK0---正脉冲　CS---280H-287H　GATE0-- +5V

运行程序：E8253_1可编程定时器/计数器程序

运行结果：压单脉冲键，每压一次在屏幕上循环显示“1-9”、“A-F”，把逻辑笔插入逻

辑孔，用控测端测试OUT0，压15次单脉冲键后，可以看到OUT0的电平变化一次。

（2）连线：CS---280H-287H　GATE1-- +5V　GATE0-- +5V　CLK1---OUT0　CLK0---1MHZ

运行程序：E8253_2可编程定时器/计数器程序

运行结果：把逻辑笔插入逻辑孔，用探测端测试OUT1，可以看到OUT1的电平有规律的进行高低交替变化。

（3）连线：CS---280H-287H　GATE1-- +5V　GATE0-- +5V　CLK1---OUT0　CLK0---1MHZ

运行程序：E8253_3可编程定时器/计数器程序

运行结果：如下所示：

counter1:____

counter2:____

continue?(y/n)____

用示波器观察不同方式下的波形，并在纸上画出CLK0、OUT0、OUT1的波形。

四、可编程并行接口（一）8255

（1）连线：PC0-PC7---K0-K7　CS---288H-28FH　PA0-PA7---L0-L7

运行程序：E8255可编程并行口（一）程序

运行结果：拨动开关K，相应置“1”的开关所对应的灯L亮，否则灭。

（2）连线：利用通用插座插好74LS138芯片，将8255的PC0-PC7---138的（15～7）脚、8255CS---288H-28FH　分别将8255的PA3-PA7---138的（1、2、3、6、4）脚、138的16脚--＋5、8脚—GND、4脚--5脚对连。

运行程序：E8255_1可编程并行口（一）程序

运行结果：

```
+-------------------------------+
|   A    B    C   G1  G2A+G2B  |
|                               |
|             74LS138           |
|                               |
|Y0  Y1  Y2  Y3  Y4  Y5  Y6  Y7|
+-------------------------------+
       Test Again ?(Y/N)','$'
```

显示由8255的A口输出，由C口读入的数据。

五、七段数码管

（1）连线：PA0-PA6---a-g　S1-- +5　S0---GND　DP---GND　CS---288H-28FH

运行程序：LED1七段数码管程序

运行结果：从电脑键盘上输入0-9，并在七段数码管上直接显示出来。

（2）连线：PA0-PA6---a-g　S1---PC1　S0---PC0　DP---GND　CS---288H-28FH
运行程序：LED2七段数码管程序
运行结果：在七段数码管上按秒循环显示00-99。

六、继电器控制

连线：CLK0---1MHZ　GATE0-- +5V　OUT0---CLK1　GATE1-- +5V
8253CS---280H-287H　OUT1---PA0　8255CS---288H-28FH　PC0---IK（右上角）
运行程序：JDQ继电器程序
运行结果：继电器周而反复的闭合5秒，同时指示灯亮，断开5秒，灯灭。

七、竞赛抢答器

连线：DP-GND　PA0-PA6---a-g　PC0-PC7---K0-K7　S1-- +5V
8255CS---288H-28FH
运行程序：QDQ竞赛抢答器程序
运行结果：当某开关置“1”，表示有按键按下，在数码管上显示其组号，同时喇叭响一下。

八、交通灯控制实验

连线：DP-GND　PC0-PC7---K0-K7　S1-- +5V
8255CS---288H-28FH
运行程序：JTD交通灯控制程序
运行结果：指示灯按交通灯规律进行亮灭。

九、中断

连线：单脉冲---IRQ
运行程序：INT中断程序
运行结果：每按一次中断，输出“TPC-USB INTERRUPT！”

十、可编程并行接口（二）8255

（1）连线：PC3---IRQ　PC6---单脉冲　PA0-PA7---L0-L7　8255CS---288H-28FH
运行程序：E8255-1o可编程并行接口程序
运行结果：按一次单脉冲键，让CPU进行一次中断，使L0-L7依次发光，8次后结束。
（2）连线：PA0-PA7---K0-K7　PC3---IRQ　PC4---单脉冲　8255CS---288H-28FH
运行程序：E8255-1i可编程并行接口程序
运行结果：按一次单脉冲键，让CPU进行一次中断，读取开关预置的ASCII码，在屏幕上显示其对应的字母，中断8次后结束。

十一、数/模转换器

（1）连线：DAC0832CS---290H-297H
运行程序：DA_1数模转换程序
运行结果：用示波器观察，单极输出端Ua及双极输出端Ub，均输出剧齿波。
（2）连线：DAC0832CS---290H-297H
运行程序：DA_2数模转换程序
运行结果：用示波器观察，单极输出端Ua及双极输出端Ub，均输出正弦波。

十二、模/数转换器

（1）连线：ADC0809CS---298H-29FH　RW1---IN0
运行程序：AD_1模/数转换程序
运行结果：采集IN0输入的电压，在屏幕上显示转换后的数据。
（2）连线：ADC0809CS---298H-29FH　RW1---IN0
运行程序：AD_2模/数转换程序
运行结果：将JP3（左下方）的1、2短路使IN2处于双机性工作方式，给IN1输入一个低频

交流信号（幅度为+5 V），编程采集这个数据并在屏幕上显示对应的幅度。

十三、串行通讯

将8251芯片插在相应的通用插座上

连线：8253OUT0---8251的9、25脚　GATE0-- +5V　8253CS---280H-287H

CLK0---1MHZ　8251的3脚---19脚　17脚--- GND　26脚-- +5V　4脚---GND

20脚---1MHZ　21脚--- RESET　12脚---A0　10脚---/IOW　13脚---/IOR

11脚---2B8H-2BFH　27、28、1、2脚---D0-D3　5-8脚---D4---D7

运行程序：E8251串行通信程序

运行结果：从键盘输入一个字符，将其ASCII码加1后发出去，再接收回来在计算机屏幕上显示，实现自收发。（如：敲a，显示ab）。

十四、DMA传送

（1） 连线： D0-D7---扩展6116（9、10、11、13、14、15、16、17）脚

A0-A10---扩展6116（8、7、6、5、4、3、2、1、23、22、19）脚

MEMW--- 6116的21脚、MEMR--- 6116的20脚、MEMCS---6116的18脚

6116的24脚---＋5、12脚---GND。

运行程序：DMA存储器到实验箱上扩展6116块传送程序

运行结果：显示源数据区256个A-Z，显示目的数据区256个a,显示传送后目的256个A-Z。

（2）将芯片74LS273插在相应的通用插座上

连线：D0-D7---74LS273（3、4、7、8、13、14、17、18）脚　或门的输入A---/DACK1

或门的输入B----/IOW　或门的输出Y---11脚　1、20脚-- +5V　10脚--- GND

L0-L7---2、5、6、9、12、15、16、19脚　D、SD-- +5V　Q--- DRQ1　CD---/DACK1

CLK---正单脉冲

运行程序：DMA_O写程序

运行结果：按压单脉冲键，使发光二极管L0-L7依次发光，10次后结束。

（3）将芯片74LS244插在通用插座上

连线：K0-K7---74LS244的2、4、6、8、11、13、15、17脚　或门输入A---/DACK1

D0-D7---74LS244的18、16、14、12、9、7、5、3脚　或门输入B---/IOR

20脚---+5V　10脚---GND　或门输出Y---1、19脚　CLK---正单脉冲

D、SD---+5V　CD---/DACK1　Q---DRQ1

运行程序：DMA_I读程序

运行结果：每压一次单脉冲键，便传送一次开关输入的值，显示开关所对应的ASCII码的字母，传送8次后结束。

十五、集成电路测试

将被测试的芯片74LS00插在通用插座上

连线：8255CS---288H-28FH　PC0~PC3---8、11、3、6脚　14脚-- +5V

PA0-PA7---1、2、4、5、9、10、12、13脚　7脚---GND

运行程序：JC集成电路测试程序

运行结果：检测芯片的好坏，好则显示“This chip is ok”，否则显示“This chip is bad”。

十六、电子琴

连线：8253CS---280H-287H　CLK0---1MHZ　GATE0—8255 PA1　OUT0---与门一入端口　8255CS---288H-28FH　PA0---与门二入端　与门输出端－－K8 喇叭

运行程序：DZQ电子琴程序

运行结果：在计算机的键盘上按数字键可以象弹钢琴一样发出不同音调。

十七、8250串行通讯实验

将8250芯片插在40芯通用插座上

连线：D0-D7---1-8脚　A0-A2---28-26脚　38、39、40脚---+5V

/IOR---21脚　/IOW---18脚　14脚---2B8H-2BFH

19、22、25、36、37、20脚---GND　11脚---10脚　15脚---9脚

16脚---2MHZ　35脚---RESET　34、31、13、12脚-- +5V

运行程序：E8250串行通讯程序

运行结果：从键盘输入一个字符，将其ASCII码值加1后发出去，再自动接收回来在计算机的屏幕上显示出来，实现自收发。

十八、步进电机控制实验

连线：8255CS---288H-28FH　PA0-PA3---BA-BD　PC0-PC7---K0-K7

运行程序：BJDJ步进电机程序

运行结果：K0-K6控制步进电机的转速，K7控制其转向。

十九、直流电机转速控制实验

连线：DAC0832CS---290H-297H　Ub---DJ　8255CS---288H-28FH

K0-K5---PC0-PC5

运行程序：ZLDJ直流电机程序

运行结果：K0-K5控制直流电机的转速。

二十、键盘显示控制实验

（1）连线：将小键盘上的20芯扁平电缆和实验台上的J7相连（注意接线别接反）。

运行程序：JPXSH1键盘显示程序

运行结果：按小键盘上键，数码管上显示相应的字符（如按7，则显示7）。

（2）连线：将小键盘上的20芯扁平电缆和实验台上的J7相连。

8253CS---280H-287H　CLK0---1MHZ　GATE0、GATE1---+5V　OUT0---CLK1

OUT1---IRQ

运行程序：JPXSH键盘显示程序

运行结果：按电子钟格式显示：XX　XX　XX　时、分、秒

C键：清除　显示全零　G键：启动　电子钟计时

D键：停止　停止计时　P键：可以自由设置时、分、秒并且可以判断设置的数据的对错，出错显示E-----，此时敲P键可重新输入预置值

E键：程序退出，返回

二十一、存储器读写

扩展6116插在40芯通用插座上

连线：6116的8-1脚---A0-A7　23脚步---A8　22脚---A9　19脚---A10

9-11脚---D0-D2　13-17脚---D3-D7　24脚-- +5V　12、18脚---GND

21脚---/MEMW　20脚---MEMR

运行程序：MEMRWEX

运行结果：显示256个A-Z

二十二、双色点阵发光二极管显示实验

连线:行片选信号 CS1 接 280H;红列片选信号 CS2 接 288H;黄列片选信号 CS3 接 290H。

运行程序：11588、11588－1

运行结果：

1、 LED点阵红色逐列点亮，再黄色逐列点亮，再红色逐行点亮，黄色逐行点亮。

2、 LED上重复显示红色“年”和黄色“年”。

（以上仅供参考，更多内容请参看实验指导书）

附录三：TPC-USB通用微机接口实验系统硬件实验指导（C语言程序）

C语言程序支持winnt/2000/xp。
参照实验指导书安装TPC-USB模块及其驱动程序后才能正常运行程序。

说明：开关K向上为“1”，向下为“0”。

一、I/O地址译码

连线：CD---2A8H-2AFH SD-- +5V D-- +5V Q---L7 CLK---2A0H-2A7H
运行程序：YMQ地址译码
运行结果：发光二极管L7有规律的进行连续闪烁。

二、简单并行接口

（1）将74LS273插在相关插座上
连线：D0-D7---74LS273的（3、4、7、8、13、14、17、18）脚 或门输入A---2A8H-2AFH
或门输入B---/IOW 或门输出Y---11脚 10脚---GND 1、20脚-- +5V
L0-L7---74LS273的（2、5、6、9、12、15、16、19）脚
运行程序：E273简单并行口程序
运行结果：拨动开关K，相应置“1”的开关所对应的灯L亮，否则灭。
（2）将74LS244插在相关插座上
连线：K0-K7---74LS244（2、4、6、8、11、13、15、17）脚 或门输入A---/IOR
或门输入B---2A0H-2A7H 或门输出Y---1、19脚 10脚---GND 20脚-- +5V
D0-D7---74LS244（18、16、14、12、9、7、5、3）脚
运行程序：E244简单并行口程序
运行结果：用开关输入字母的ASCII码值，在屏幕上显示对应的字母。

三、可编程定时器/计数器

（1）连线：CLK0---正脉冲 CS---280H-287H GATE0-- +5V
运行程序：E8253_1可编程定时器/计数器程序
运行结果：压单脉冲键，每压一次在屏幕上循环显示“1-9”、“A-F”，把逻辑笔插入逻
辑孔，用控测端测试OUT0，压15次单脉冲键后，可以看到OUT0的电平变化一次。
（2）连线：CS---280H-287H GATE1-- +5V GATE0-- +5V CLK1---OUT0
CLK0---1MHZ
运行程序：E8253_2可编程定时器/计数器程序
运行结果：把逻辑笔插入逻辑孔，用探测端测试OUT1，可以看到OUT1的电平有规律的进行高低交替变化。

四、可编程并行接口（一）8255

连线：PC0-PC7---K0-K7 CS---288H-28FH PA0-PA7---L0-L7
运行程序：E8255可编程并行口（一）程序
运行结果：拨动开关K，相应置“1”的开关所对应的灯L亮，否则灭。

五、七段数码管

（1）连线：PA0-PA6---a-g S1-- +5 S0---GND DP---GND CS---288H-28FH
运行程序：LED1七段数码管程序
运行结果：从电脑键盘上输入0-9，并在七段数码管上直接显示出来。
（2）连线：PA0-PA6---a-g S1---PC1 S0---PC0 DP---GND CS---288H-28FH
运行程序：LED2七段数码管程序
运行结果：在七段数码管上按秒循环显示00-99。

六、继电器控制

连线：CLK0---1MHZ　GATE0-- +5V　OUT0---CLK1　GATE1-- +5V

8253CS---280H-287H　OUT1---PA0　8255CS---288H-28FH　PC0---IK（右上角）

运行程序：JDQ继电器程序

运行结果：继电器周而反复的闭合5秒，同时指示灯亮，断开5秒，灯灭。

七、竞赛抢答器

连线：DP-GND　PA0-PA6---a-g　PC0-PC7---K0-K7　S1-- +5V

8255CS---288H-28FH

运行程序：QDQ竞赛抢答器程序

运行结果：当某开关置“1”，表示有按键按下，在数码管上显示其组号，同时喇叭响一下。

八、交通灯控制实验

连线：DP-GND　PC0-PC7---K0-K7　S1-- +5V

8255CS---288H-28FH

运行程序：JTD交通灯控制程序

运行结果：指示灯按交通灯规律进行亮灭。

九、中断

（1）连线：8255PA0-PA7---L0-L7　PC0---正单脉冲　8255CS---288H-28FH

运行程序：INT0查询法中断程序

运行结果：　单脉冲每按一次，指示灯显示0x55，否则指示灯显示0xAA

(2)连线：8255PA0-PA7---L0-L7　8255CS---288H-28FH　正单脉冲---IRQ

运行程序：INT1中断法中断程序

运行结果：单脉冲每按一次，指示灯显示0x55，否则指示灯显示0xAA

十、可编程并行接口（二）8255

（1）连线：PC3---IRQ　PC6---单脉冲　PA0-PA7---L0-L7　8255CS---288H-28FH

运行程序：E8255-1o可编程并行接口程序

运行结果：按一次单脉冲键，让CPU进行一次中断，使L0-L7依次发光，8次后结束。

（2）连线：PA0-PA7---K0-K7　PC3---IRQ　PC4---单脉冲　8255CS---288H-28FH

运行程序：E8255-1i可编程并行接口程序

运行结果：按一次单脉冲键，让CPU进行一次中断，读取开关预置的ASCII码，在屏幕上显示其对应的字母，中断8次后结束。

十一、数/模转换器

（1）连线：DAC0832CS---290H-297H

运行程序：DA_1数模转换程序

运行结果：用示波器观察，单极输出端Ua及双极输出端Ub，均输出剧齿波。

（2）连线：DAC0832CS---290H-297H

运行程序：DA_2数模转换程序

运行结果：用示波器观察，单极输出端Ua及双极输出端Ub，均输出正弦波。

十二、模/数转换器

（1）连线：ADC0809CS---298H-29FH　RW1---IN0

运行程序：AD_1模/数转换程序

运行结果：采集IN0输入的电压，在屏幕上显示转换后的数据。

（2）连线：ADC0809CS---298H-29FH　RW1---IN0

运行程序：AD_2模/数转换程序

运行结果：将JP3（左下方）的1、2短路使IN2处于双机性工作方式，给IN1输入一个低频

交流信号（幅度为+5 V），编程采集这个数据并在屏幕上显示对应的幅度。

十三、串行通讯

将8251芯片插在相应的通用插座上

连线：8253OUT0---8251的9、25脚　GATE0-- +5V　8253CS---280H-287H

CLK0---1MHZ　8251的3脚---19脚　17脚--- GND　26脚-- +5V　4脚---GND

20脚---1MHZ　21脚--- RESET　12脚---A0　10脚---/IOW　13脚---/IOR

11脚---2B8H-2BFH　27、28、1、2脚---D0-D3　5-8脚---D4---D7

运行程序：E8251串行通信程序

运行结果：从键盘输入一个字符，将其ASCII码加1后发出去，再接收回来在计算机屏幕上显示，实现自收发。（如：敲a，显示ab）。

十四、DMA传送

（1）连线： D0-D7---扩展6116（9、10、11、13、14、15、16、17）脚

A0-A10---扩展6116（8、7、6、5、4、3、2、1、23、22、19）脚

MEMW--- 6116的21脚、MEMR--- 6116的20脚、MEMCS---6116的18脚

6116的24脚---＋5、12脚---GND。

运行程序：DMA存储器到实验箱上扩展6116块传送程序

运行结果：显示源数据区256个A-Z，显示目的数据区256个a,显示传送后目的256个A-Z。

（2）将芯片74LS273插在相应的通用插座上

连线：D0-D7---74LS273（3、4、7、8、13、14、17、18）脚　或门的输入A---/DACK1

或门的输入B----/IOW　或门的输出Y---11脚　1、20脚-- +5V　10脚--- GND

L0-L7---2、5、6、9、12、15、16、19脚　D、SD-- +5V　Q--- DRQ1　CD---/DACK1

CLK---正单脉冲

运行程序：DMA_O写程序

运行结果：按压单脉冲键，使发光二极管L0-L7依次发光，10次后结束。

（3）将芯片74LS244插在通用插座上

连线：K0-K7---74LS244的2、4、6、8、11、13、15、17脚　或门输入A---/DACK1

D0-D7---74LS244的18、16、14、12、9、7、5、3脚　或门输入B---/IOR

20脚---+5V　10脚---GND　或门输出Y---1、19脚　CLK---正单脉冲

D、SD---+5V　CD---/DACK1　Q---DRQ1

运行程序：DMA_I读程序

运行结果：每压一次单脉冲键，便传送一次开关输入的值，显示开关所对应的ASCII码的字母，传送8次后结束。

十五、集成电路测试

将被测试的芯片74LS00插在通用插座上

连线：8255CS---288H-28FH　PC0~PC3---8、11、3、6脚　14脚-- +5V

PA0-PA7---1、2、4、5、9、10、12、13脚　7脚---GND

运行程序：JC集成电路测试程序

运行结果：检测芯片的好坏，好则显示“This chip is ok”，否则显示“This chip is bad”。

十六、电子琴

连线：8253CS---280H-287H　CLK0---1MHZ　GATE0—8255 PA1　OUT0---与门一入端口　8255CS---288H-28FH　PA0---与门二入端　与门输出端－－K8 喇叭

运行程序：DZQ电子琴程序

运行结果：在计算机的键盘上按数字键可以象弹钢琴一样发出不同音调。

十七、8250串行通讯实验

将8250芯片插在40芯通用插座上

连线：D0-D7---1-8脚　A0-A2---28-26脚　38、39、40脚---+5V

/IOR---21脚　/IOW---18脚　14脚---2B8H-2BFH

19、22、25、36、37、20脚---GND　11脚---10脚　15脚---9脚

16脚---2MHZ　35脚---RESET　34、31、13、12脚-- +5V

运行程序：E8250串行通讯程序

运行结果：从键盘输入一个字符，将其ASCII码值加1后发出去，再自动接收回来在计算机的屏幕上显示出来，实现自收发。

十八、步进电机控制实验

连线：8255CS---288H-28FH　PA0-PA3---BA-BD　PC0-PC7---K0-K7

运行程序：BJDJ步进电机程序

运行结果：K0-K6控制步进电机的转速，K7控制其转向

十九、直流电机转速控制实验

连线：DAC0832CS---290H-297H　Ub---DJ　8255CS---288H-28FH

K0-K5---PC0-PC5

运行程序：ZLDJ直流电机程序

运行结果：K0-K5控制直流电机的转速。

二十、键盘显示控制实验

（1）连线：将小键盘上的20芯扁平电缆和实验台上的J7相连（注意接线别接反）。

运行程序：JPXSH1键盘显示程序

运行结果：按小键盘上键，数码管上显示相应的字符（如按7，则显示7）。

（2）连线：将小键盘上的20芯扁平电缆和实验台上的J7相连。

8253CS---280H-287H　CLK0---1MHZ　GATE0、GATE1---+5V　OUT0---CLK1

OUT1---IRQ

运行程序：JPXSH键盘显示程序

运行结果：按电子钟格式显示：XX　XX　XX　时、分、秒

C键：清除　显示全零　G键：启动　电子钟计时

D键：停止　停止计时　P键：可以自由设置时、分、秒并且可以判断　设置的数据的对错，出错显示E-----，此时敲P键可重新输入预置值E键：程序退出，返回

二十一、存储器读写

扩展6116插在40芯通用插座上

连线：6116的8-1脚---A0-A7　23脚步---A8　22脚---A9　19脚---A10

9-11脚---D0-D2　13-17脚---D3-D7　24脚-- +5V　12、18脚---GND

21脚---/MEMW　20脚---MEMR

运行程序：MEMRWEX

运行结果：显示256个A-Z

二十二：双色点阵发光二极管显示实验

连线：行片选信号 CS1 接 280H；红列片选信号 CS2 接 288H；黄列片选信号 CS3 接 290H。

运行程序：11588、11588－1

运行结果：

3、LED点阵红色逐列点亮，再黄色逐列点亮，再红色逐行点亮，黄色逐行点亮。

4、LED上重复显示红色“年”和黄色“年”。

（以上仅供参考，更多内容请参看实验指导书）