目　　录

第一章 软件使用介绍

说明：

1、菲利普/STC单片机在基本89C51单片机经过完善，增加了部分功能（增加了部分特殊寄存器），实现其功能时需初始化对应的寄存器，具体功能请参考光盘中的芯片资料介绍。

2、菲利普/STC单片机外部存储器寻址，需设置其寄存器，否则地址小于内部RAM地址范围时，寻址其内部RAM，只有寻址地址大于其内部RAM地址范围时，才自动寻址其外部存储器。

3、C51单片机下载软件为避免学生误操作，破坏掉菲利普单片机ISP下载引导区。去掉了原厂家提供的烧写软件部分功能。如需使用其它烧写功能，请使用芯片厂家提供的烧写软件（菲利普、STC）。软件在产品资料光盘中。

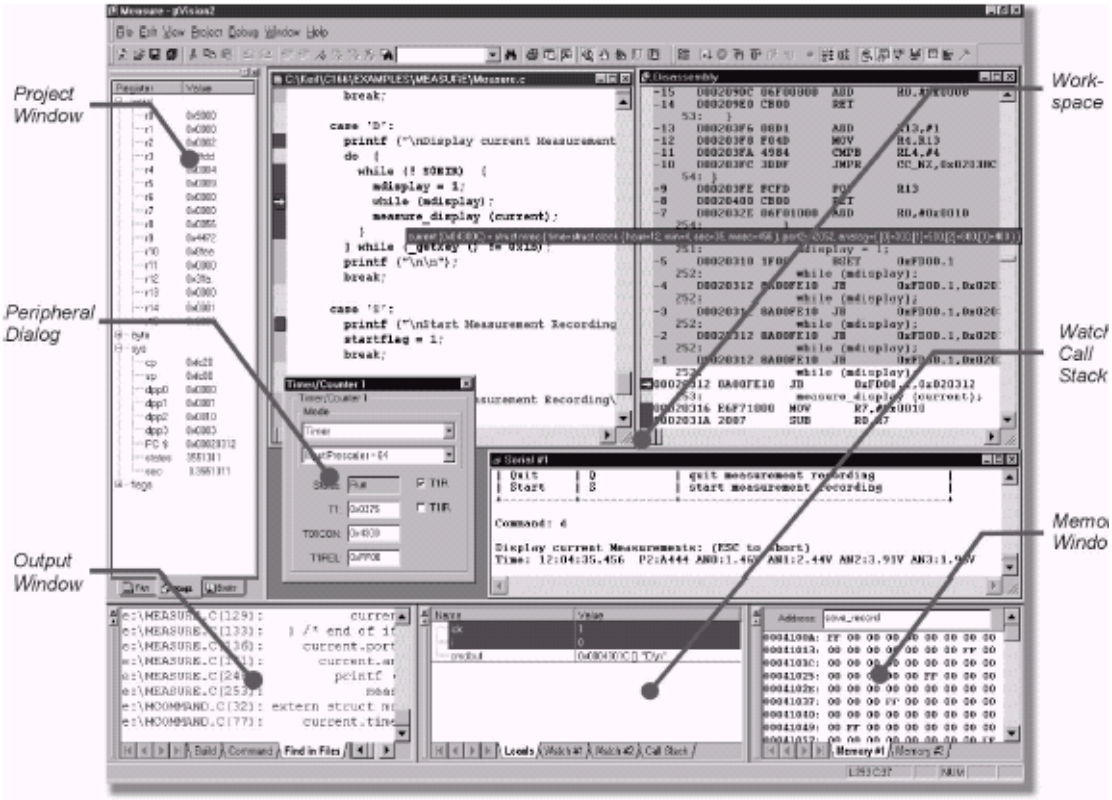
4、AT89S52在线烧写后，因使用了芯片的P1.5~P1.7管脚，如在程序中使用了该三个管脚时，下载完成后，须取下载线。

一、Keil uVsion2 软件

1、 Keil uVision2菜单条，工具条和快捷键

在 Keil uVision2 中，管理文件使用工程文件而不是以前的单一文件的模式，C51 源程序、汇编源程序、头文件等都可放在工程里统一管理。

菜单条提供各种操作菜单，如：编辑操作，项目维护，开发工具选项设置，调试程序，窗口选择和处理，在线帮助。工具条按钮允许你快速地执行 uVision2 命令。键盘快捷键（你自己可以配置）允许你执行 uVision2 命令。



uVision2 各窗口和工具条名称

下面的表格列出了 uVision2 菜单项命令，工具条图标，默认的快捷键以及他们的描述：

1.1 文件菜单和命令（File）

菜单	工具条	快捷键	描述
New		Ctrl+N	创建新文件
Open		Ctrl+O	打开已经存在的文件
Close			关闭当前文件
Save		Ctrl+S	保存当前文件
Save al			保存所有文件
Save as…			另外取名保存
Device Database			维护器件库
Print Setup…			设置打印机
Print		Ctrl+P	打印当前文件
Print Preview			打印预览
1-9			打开最近用过的文件
Exit			退出 uVision2，提示是否保存文件。

1.2 编辑菜单和编辑器命令(Edit)

菜单	工具条	快捷键	描述
Home			移动光标到本行的开始
End			移动光标到本行的末尾
Ctrl+Home			移动光标到文件的开始
Ctrl+End			移动光标到文件的结束
Ctrl+<-			移动光标到词的左边
Ctrl+->			移动光标到词的右边
Ctrl+A			选择当前文件的所有文本内容
Undo		Ctrl+Z	取消上次操作
Redo		Ctrl+Shift+Z	重复上次操作
Cut		Ctrl+X	剪切所选文本
		Ctrl+Y	剪切当前行的所有文本
Copy		Ctrl+C	复制所选文本
Paste		Ctrl+V	粘贴
Indent Selected Text			将所选文本右移一个制表键的距离
Unindent Selected Text			将所选文本左移一个制表键的距离
Toggle Bookmark		Ctrl+F2	设置/取消当前行的标签

Goto Next Bookmar	F2	移动光标到下一个标签处
Goto Previous Boo rk	Shift+F2	移动光标到上一个标签处
Clear All Bookmar		清除当前文件的所有标签
Find	Ctrl+F	在当前文件中查找文本
	F3	向前重复查找
	Shift+F3	向后重复查找
	Ctrl+F3	查找光标处的单词
	Ctrl+]	寻找匹配的大括号，圆括号，方括号（用此命令将光标放到大括号，圆括号或方括号的前面）
Replace	Ctrl+H	替换特定的字符
Find in Files…		在多个文件中查找

1.3 选择文本命令

在 uVision2 中，你可以通过按住 Shift 键和相应的光标操作键来选择文本。如，Ctrl+-> 是移动光标到下一个词，那么，Ctrl+Shift+-> 就是选择当前光标位置到下一个词的开始位置间的文本。

当然，你也可以用鼠标来选择文本，操作如下：

要选择. . .	鼠标操作
任意数量的文本	在你要选择的文本上拖动鼠标
一个词	双击此词
一行文本	移动鼠标到此行的最左边，直到鼠标变成右指向的箭头，然后单击。
多行文本	移动鼠标到此行的最左边，直到鼠标变成右指向的箭头，然后相应拖动。
一个距形框中的文本	按住 Alt 键，然后相应拖动鼠标。

1.4 视图菜单（View）

菜单	工具条	快捷键	描述
Status Bar			显示/隐藏状态条
File Toolbar			显示/隐藏文件菜单条
Build Toolbar			显示/隐藏编译菜单条
Debug Toolbar			显示/隐藏调试菜单条
Project Windo			显示/隐藏项目窗口
Output Window			显示/隐藏输出窗口
Source Browser			打开资源浏览器
Disassembly Wi w			显示/隐藏反汇编窗口
Watch & Call			显示/隐藏观察和堆栈窗口
Stack Window			
Memory Window			显示/隐藏存储器窗口
Code Coverage dow			显示/隐藏代码报告窗口
Performance Analyzer Window			显示/隐藏性能分析窗口

Symbol Window 显示/隐藏字符变量窗口

Serial Window 显示/隐藏串口 1 的观察窗口

Serial Window #2 显示/隐藏串口 2 的观察窗口

Toolbox 显示/隐藏自定义工具条

Periodic Window Update 程序运行时刷新调试窗口

Workbook Mode 显示/隐藏窗口框架模式

Options… 设置颜色，字体，快捷键和编辑器的选项

1.5 项目菜单和项目命令（Project）

菜单	工具条	快捷键	描述
New Project …			创建新项目
Import µVision1 Project …			转化 uVision1 的项目
Open Project …			打开一个已经存在的项目
Close Project…			关闭当前的项目
Target Environment			定义工具、包含文件和库的路径
Targets, Groups,Files			维护一个项目的对象、文件组和文件
Select Device for Target			选择对象的 CPU
Remove…			从项目中移走一个组或文件.
Options…		Alt+F7	设置对象、组或文件的工具选项
MCB251			
File Extensions			选择不同文件类型的扩展名
Build Target		F7	编译修改过的文件并生成应用
Rebuild Targ			重新编译所有的文件并生成应用
Translate…		Ctrl+F7	编译当前文件
Stop Build			停止生成应用的过程
1-9			打开最近打开过的项目

1.6 调试菜单和调试命令（Debug）

菜单	工具条	快捷键	描述
Start/Stop Debugging		Ctrl+F5	开始/停止调试模式
Go		F5	运行程序，直到遇到一个中断
Step		F11	单步执行程序，遇到子程序则进入
Step ov		F10	单步执行程序，跳过子程序
Step out of Current function		Ctrl+F11	执行到当前函数的结束

Stop Running　　ESC　　停止程序运行

Breakpoints…　　打开断点对话框

Insert/Remove Breakpoint　　设置/取消当前行的断点

Enable/Disable Breakpoint　　使能/禁止当前行的断点

Disable All Breakpoints　　禁止所有的断点

Kill All Breakpoints　　取消所有的断点

Show Next Statement　　显示下一条指令

Enable/Disable Trace Recording　　使能/禁止程序运行轨迹的标识

View Trace Records　　显示程序运行过的指令

Memory Map…　　打开存储器空间配置对话框

Performance Analyzer…　　打开设置性能分析的窗口

Inline Assembly…　　对某一个行重新汇编，可以修改汇编代码

Function Editor…　　编辑调试函数和调试配置文件

1.7　外围器件菜单（Peripherals）

菜单	工具条	快捷键	描述
Reset CPU			复位 CPU
Interrupt, I/O-Ports, Serial, Timer, A/D Converter, D/A Converter, I2C Controller, CAN Controller, Watchdog			打开片上外围器件的设置对话框，对话框的种类及内容依赖于你选择的 CPU，

1.8　工具菜单（Tool）

利用工具菜单，你可以配置，运行 Gimpel PC-Lint，Siemens Easy-Case 和用户程序。通过 Customize Tools Menu…菜单，你可以添加你想要添加的程序。

菜单	工具条	快捷键	描述

Setup PC-Lint…	配置 Gimpel Software 的 PC-Lint 程序
Lint	用 PC-Lint 处理当前编辑的文件
Lint all C Source Files	用 PC-Lint 处理你项目中所有的 C 源代码文件
Setup Easy-Case…	配置 Siemens 的 Easy-Case 程序
Start/Stop Easy-Case	运行/停止 Siemens 的 Easy-Case 程序
Show File (Line)	用 Easy-Case 处理当前编辑的文件
Customize Tools Menu…	添加用户程序到工具菜单中

1.9 软件版本控制系统菜单（SVCS）

用此菜单来配置和添加软件版本控制系统的命令。

菜单	工具条	快捷键	描述
Configure			配置软件版本控制系统的命令

Version Control…

1.10 视窗菜单（Window）

菜单	工具条	快捷键	描述
Cascade			以互相重叠的形式排列文件窗口
Tile Horizontally			以不互相重叠的形式水平排列文件窗口
Tile Vertically			以不互相重叠的形式垂直排列文件窗口
Arrange Icons			排列主框架底部的图标
Split			把当前的文件窗口分割为几个

1-9 激活指定的窗口对象

1.11 帮助菜单（Help）

菜单	工具条	快捷键	描述
Help topics			打开在线帮助
About μVision			显示版本信息和许可证信息

2、建立新工程的一般步骤

在 Keil IDE 中不支持单文件的处理，只有建立一个工程并对该工程进行正确的设置后，才能使用 Keil 进行编译连接仿真等操作。

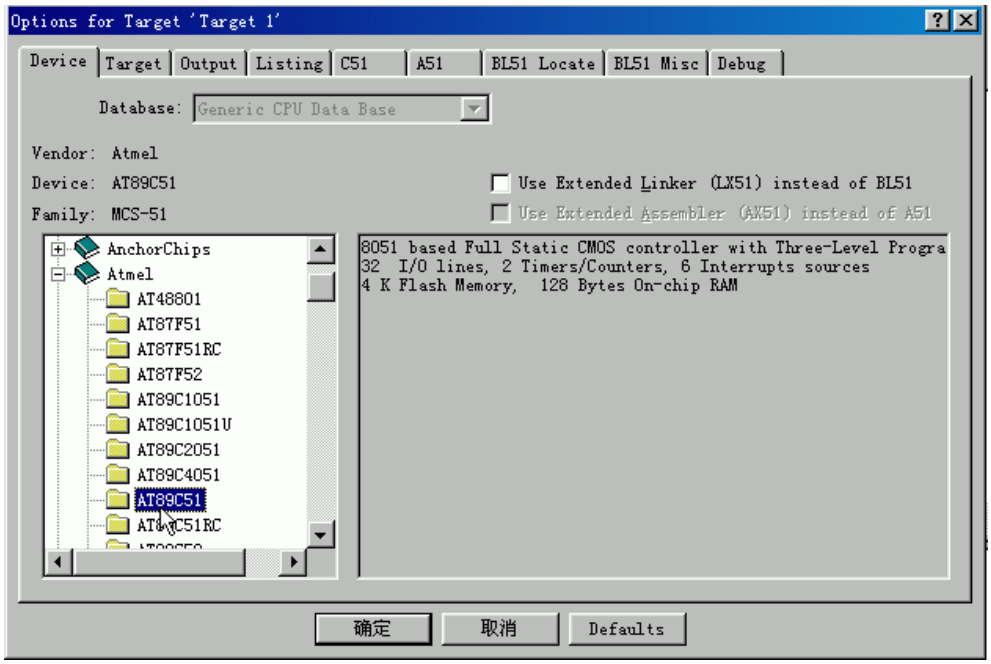
2.1 新建工程

点击菜单 Project->New Project..后，出现对话窗口；

在对话框内选择工程目录填写新工程名称点击保存新建工程。

2.2 为工程选择目标器件

在建立工程以后，还应该为工程选择合适的目标器件选择目标器件方法为：点击菜单 Project->Select Device for Targect... 后出现下面的对话框：



在此对话窗中,左边的数据库内容：窗口中厂商列表节可以单击打开显示对应器件。上图中已选择 Atmel 的 AT89C51 器件。

2.3 添加程序文件

选择目标器件完毕后可以看到在 Project 窗口出现了一个 Target1 的工程点，该目录里面还会有“Source Group1”的分组名，可以在该组下放置源程序文件。用鼠标右键点击“Source Group1”，在弹出的菜单中选择“Add files to Group Source Group1”。在弹出的对话窗口中选择待添加的程序文件，点击“Add” 即可将此文件增加到源文件组内，点击“Close”返回。如要增加新文件到文件组“Source Group1”， 要先使用“菜单 File-> New 功能建立文件，再进行“添加程序文件” 。

最好把一个工程内的所有文件放在一个目录中或分类到一个目录下的多个不同子目录中。

2.4 工作环境和参数的设置

在 Keil 的使用中，参数配置同样重要。新工程所有的配置参数都会使用缺省数值，一般可以正常运行，使用初期用户如果遇到不理解的配置参数可以不予理睬，在以后的应用中再逐步弄懂各个参数的实际用处。但工程调试参数和“输出 Hex 代码文件”一定要设置，因为 Keil 的缺省设置是不生成 Hex 代码文件。

手动将输出 Hex 文件控制打开方法如下：点击工程组窗口的工程组名再点击菜单“Project-Options for Target..”，在工程设置对话框中选择“Output” 页选中“Create Hex file”，同时也可选中“Debug Information”和“Browse Information”，点击“确定”退出后重新编译连接工程，即可生成 Hex 代码文件调试信息和浏览信息。Keil uVision2 IDE 提供了功能非常强大的开发环境，相信会给设计带来无穷的乐

3、使用Keil进行调试的基本技巧

3.1 进入和退出仿真状态

只有在用户程序编绎和连接成功后才能进行调试工作。

点击菜单 Debug->Start/Stop Debug Session IDE 将进入/退出硬件仿真状态。

3.2　　如何运行程序

在 Keil 的 IDE 中有以下几种运行方式：

A） Run 全速运行遇到断点停下或用户按动 Stop 按钮或 RST CPU 按钮停止。

B） Step info 单步跟踪运行一条指令，如果该语句为 C 中的调用子程序语句或汇编中的 CALL 指令，Step info 指令将跟踪进入子程序内部。

C） Step over 单步运行完一条指令，如果该语句为 C 中的调用子程序语句或汇编中的 CALL 指令，Step over 指令将全速完成该子程序的运行，停在下一指令处。

D） Run till Cursor Line 从当前位置运行到光标处。

另外，用户还必须注意，在 C 源程序窗口内、汇编源程序窗口内和在反汇编窗口内，以上命令表现会有所不同，请用户在使用时自行体会。

3.3　　如何设置和删除断点

设置断点/删除断点最简单的方法，是用鼠标双击待设置断点的源程序行或反汇编程序行，或用断点设置命令 “bs ……”。

3.4　　如何查看和修改寄存器的内容

仿真时主寄存器的内容显示在主寄存器窗口，用户除了可以观察以外还可自行修改，用鼠标点击选中一个单元，例如单元 DPTR， 然后再单击 DPTR 的数值位置，出现文字框后输入相应数值按回车键即可；另外的输入方法是使用命令行窗口，例如输入 A=0X34 将把 A 的数值设置为 0X34。

3.5　　如何观察和修改变量

点击“View->Watch & Call stack Window”出现相应窗口选择 Watch 1-3 中的任一个窗口，按动 F2， 在 Name 栏填入用户变量名如 Temp1 Counter 等，但必须是存在的变量。如果想修改数值可单击 Value 栏出现文本框后输入相应数值。用户可以连续修改多个不同的变量。

uVision2 IDE 提供了观察变量更简单的方法。在用户程序停止运行时，移动鼠标光标到要观察的变量上停大约一秒钟，就弹出一个“变量提示”块出来。

3.6　　如何观察存储器区域

在 Keil 中可以区域性的观察和修改所有的存储器数据，这些数据的获取从 Ky51 中获取。

Keil IDE 把 MCS-51 内核的存储器资源分成 4 个部分：

A） 内部可直接寻址 RAM data ,IDE 表示为 D:xx。

B） 间接寻址 RAM 区 idata ,IDE 表示为 I:xx。

C） RAM 区 xdata ,IDE 表示为 X:xxxx。

D） 代码区 code ,IDE 表示为 C:xxxx。

这四个区域都可以在 Keil 的 Memory Windows 中观察和修改。IDE 集成环境中点击菜单 View->Memory Windows，便会打开 Memory 窗口，Memory 窗口，可以同时显示 4 个不同的存储器区域，点击窗口下部分的编号可以相互切换显示。

在地址输入栏内输入待显示的存储器区起始地址。如 D:45h 表示从内部可直接寻址 RAM 的 45H 地址处开始显示；x:3f00H 显示外部 RAM， 从 3f00H 地址开始；c:0X1234 显示代码区域，从 1234H 地址开始。I:32H 显示内部间接寻址空间，从 32H 地址开始。

显示格式的切换：在区域显示中，缺省的显示单元为字节(byte)， 但是可以选择其他显示方式，

在 Memory 显示区域内按动鼠标右键，在弹出的菜单中可以选择的显示方式为：

Decimal　　按照十进制方式显示

Unsigned　　按照有符号的数字显示又分 char 单字节 int 整型 long 长整型

Singed　　按照无符号的数字显示又分 char 单字节 int 整型 long 长整型

ASCII　　按照 ASCII 码格式显示

Float　　按照浮点格式进行显示

Double　　按照双精度浮点格式显示

在 Memory 窗口中显示的数据可以修改，修改方法如下：在鼠标对准要修改的存储器单元，按动鼠标右键在弹出的菜单中选择“Modify Memory at 0x...”，在弹出对话框的文本输入栏内输入相应数值后按回车键，修改完成。注：代码区数据不能更改。

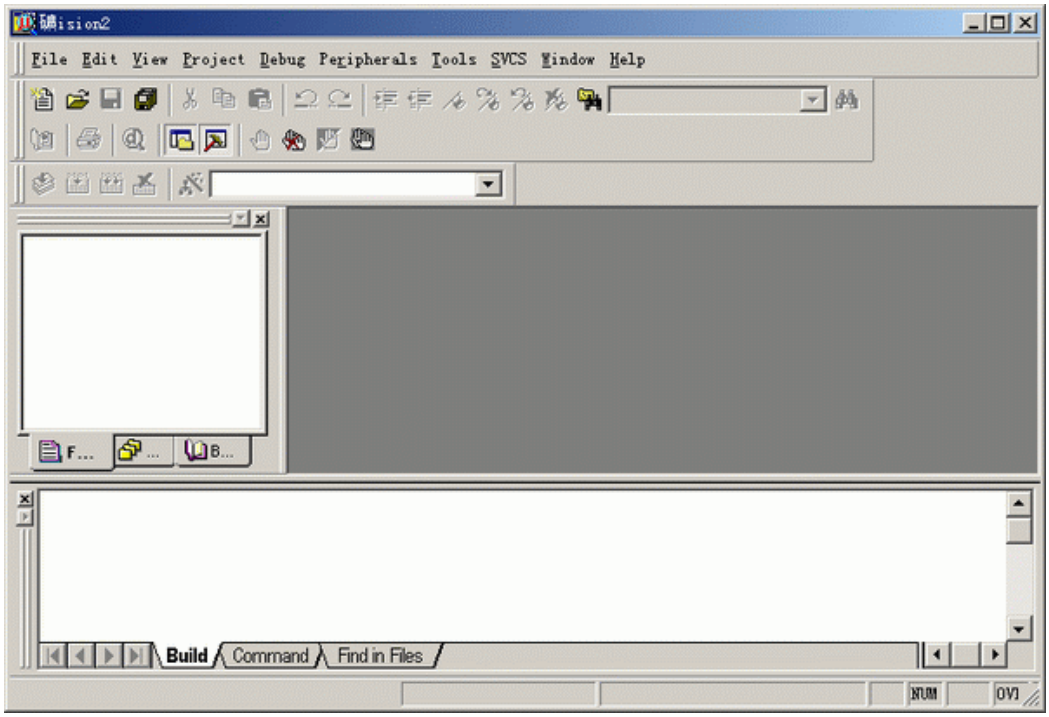
二、从一个简单实例学 Keil　Vision2 的使用

Keil C51 软件是众多单片机应用开发的优秀软件之一，它集编辑，编译，仿真于一体，支持汇编,PLM 语言和 C 语言的程序设计，界面友好，易学易用。

让我们通过一则例子学习　Keil C51 软件的使用。

进入 Keil C51 后，屏幕如下图所示。几秒钟后出现编辑界面。

启动 Keil C51 时的屏幕

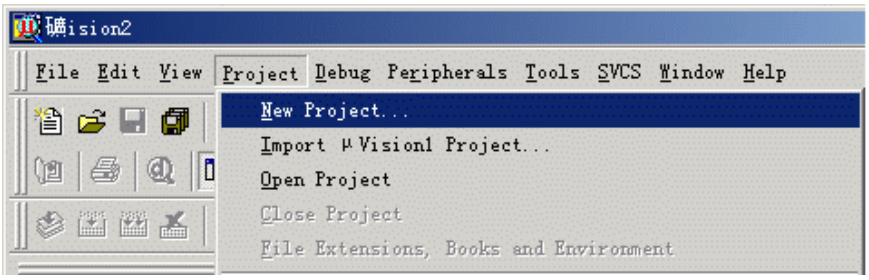


进入 Keil C51 后的编辑界面

学习程序设计语言、学习某种程序软件，最好的方法是直接操作实践。下面通过简单的编程、调试，引导大家学习 Keil C51 软件的基本使用方法和基本的调试技巧。

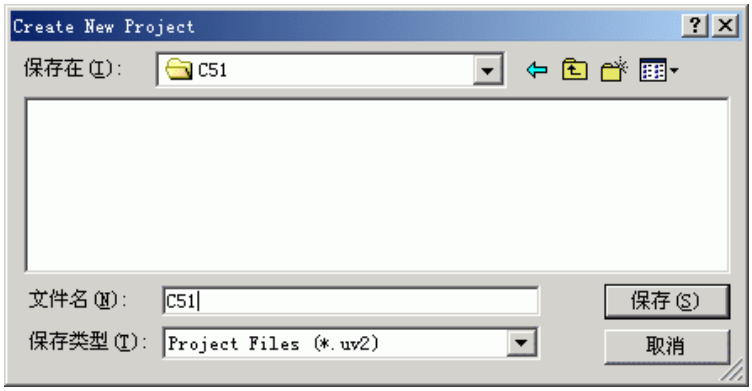
1） 建立一个新工程

单击 Project 菜单，在弹出的下拉菜单中选中 New Project 选项



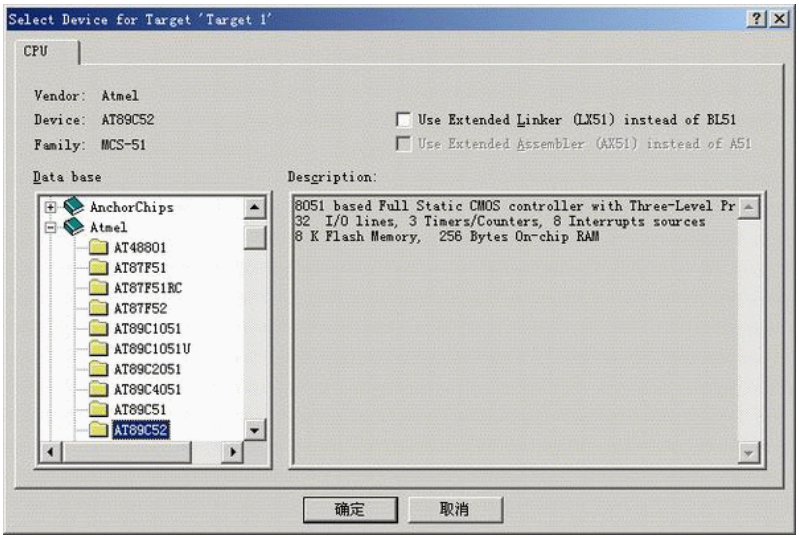
2) 然后选择你要保存的路径，输入工程文件的名字，比如保存到 C51 目录里,工程文件的名字为 C51。

如下图所示，然后点击保存。

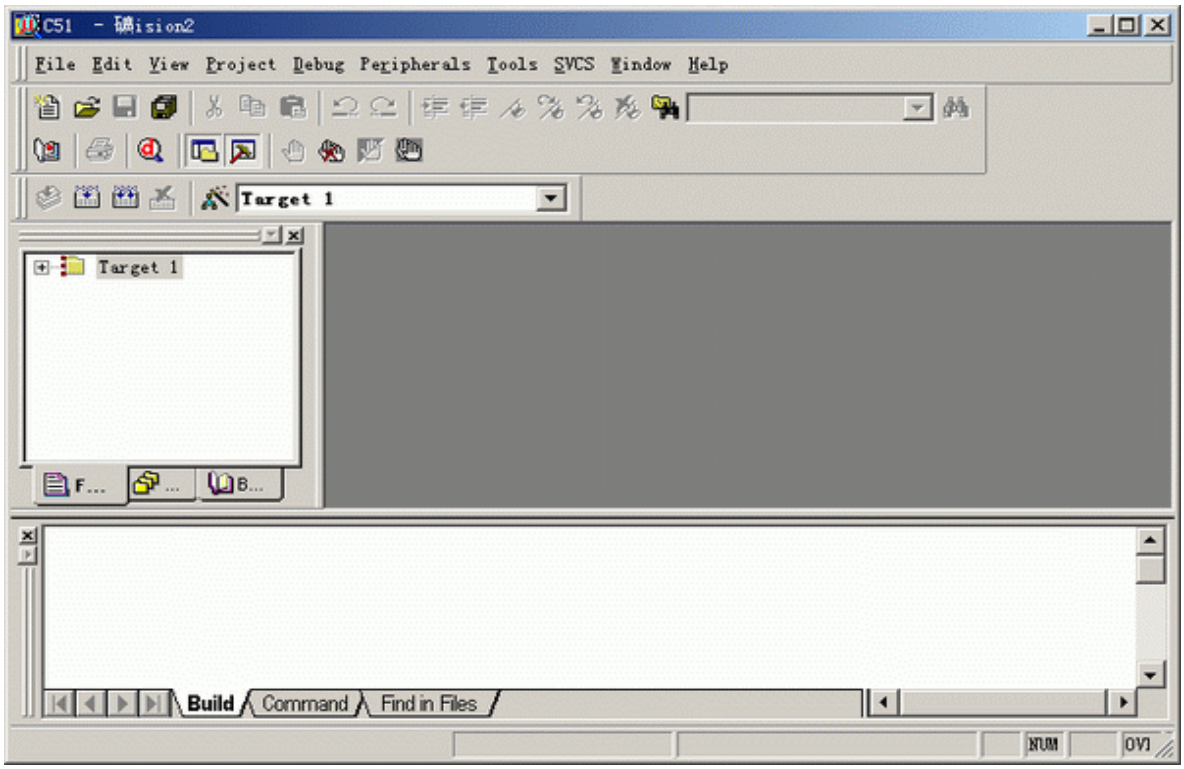


3) 这时会弹出一个对话框,要求你选择单片机的型号，你可以根据你使用的单片机来选择，keil c51 几乎支持所有的 51 核的单片机，我这里还是以大家用的比较多的 Atmel 的 89C51 来说明。

如下图所示,选择 89C51 之后，右边栏是对这个单片机的基本的说明,然后点击确定。

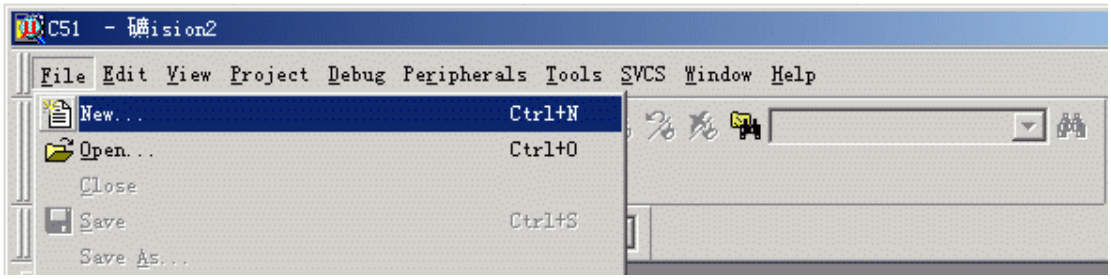


4)完成上一步骤后，屏幕如下图所示

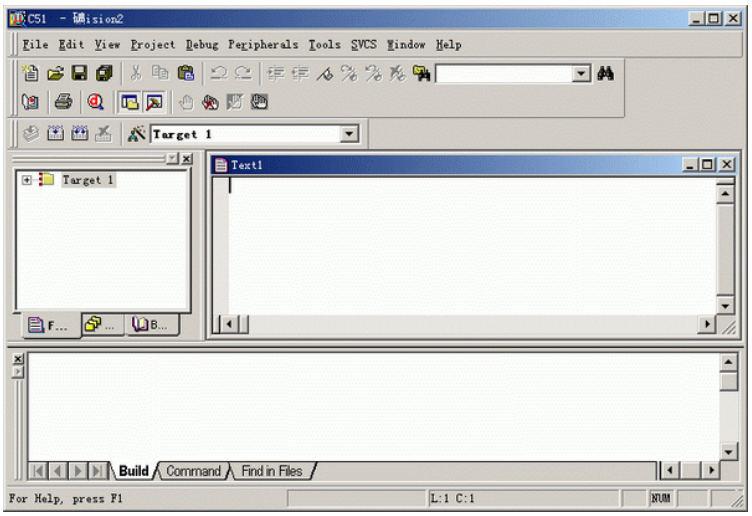


到现在为止，我们还没有编写一句程序，下面开始编写我们的第一个程序。

5)在下图中，单击“File”菜单，再在下拉菜单中单击“New”选项

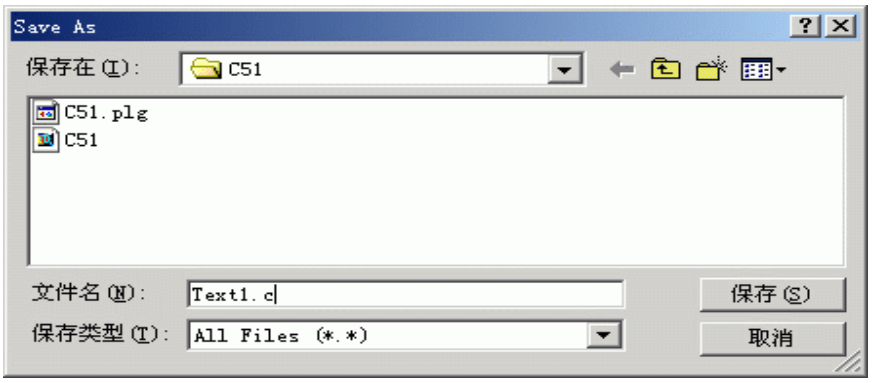


新建文件后屏幕如下图所示



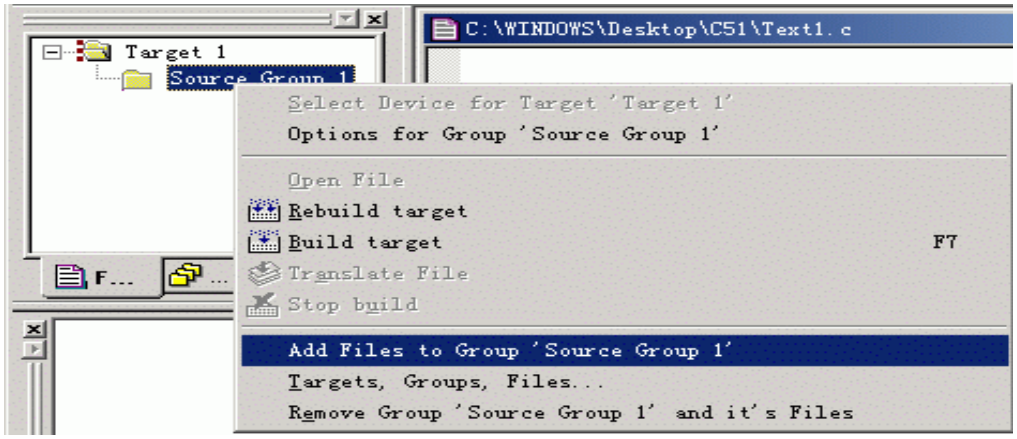
此时光标在编辑窗口里闪烁，这时可以键入用户的应用程序了，但笔者建议首先保存该空白的文件，单击菜单上的“File”，在下拉菜单中选中“Save As”选项单击，屏幕如下图所示，在“文件名”栏

右侧的编辑框中，键入欲使用的文件名，同时，必须键入正确的扩展名。注意，如果用C语言编写程序，则扩展名为(.c)；如果用汇编语言编写程序，则扩展名必须为(.asm)。然后，单击“保存”按钮。

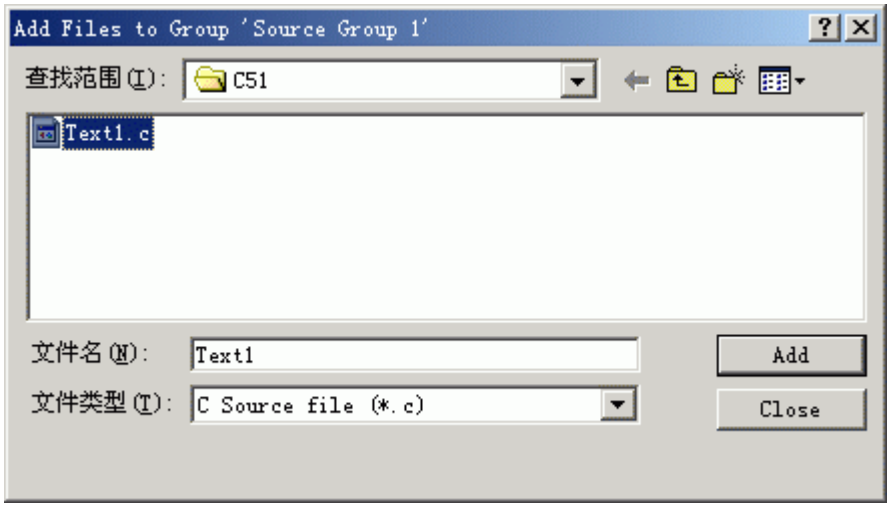


6) 回到编辑界面后，单击“Target 1”前面的“+”号，然后在“Source Group 1”上单击右键，弹出如下

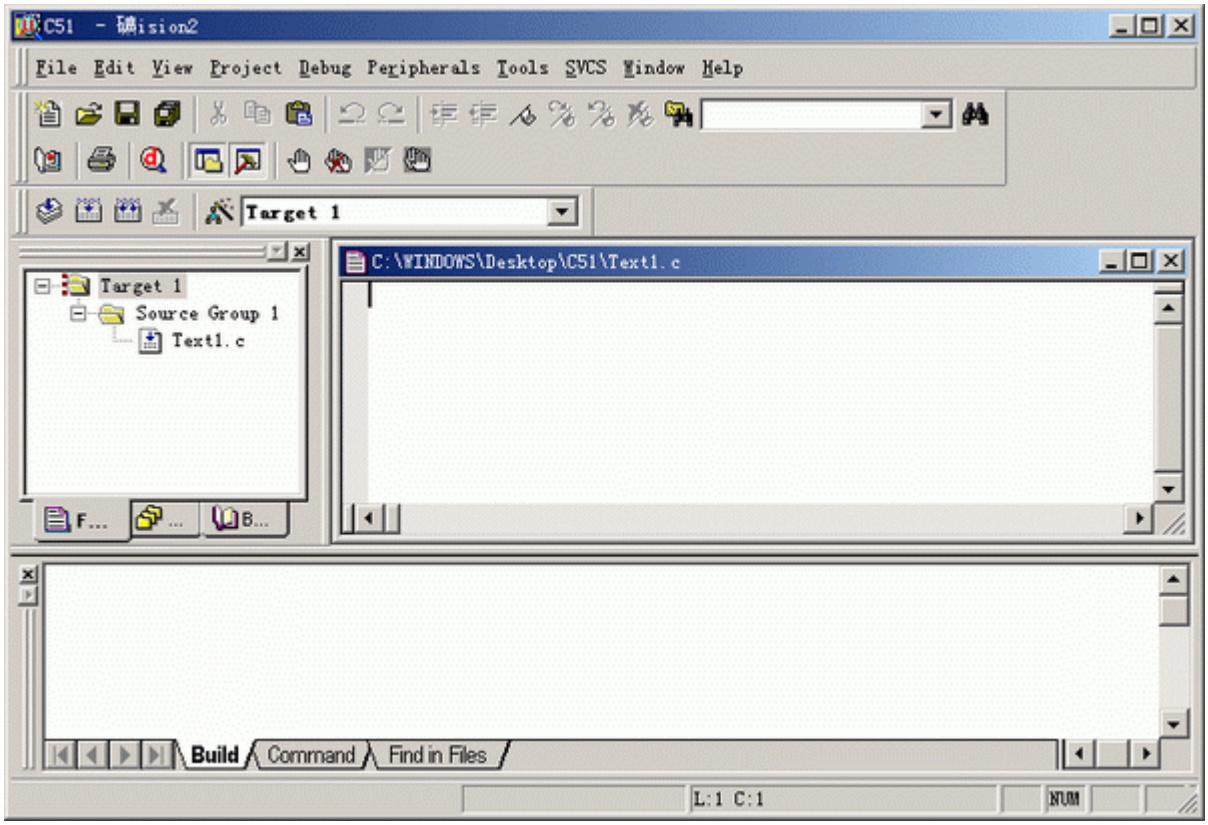
菜单



然后单击“Add File to Group ‘Source Group 1’” 屏幕如下图所示



选中 Test.c，然后单击“Add ”屏幕好下图所示

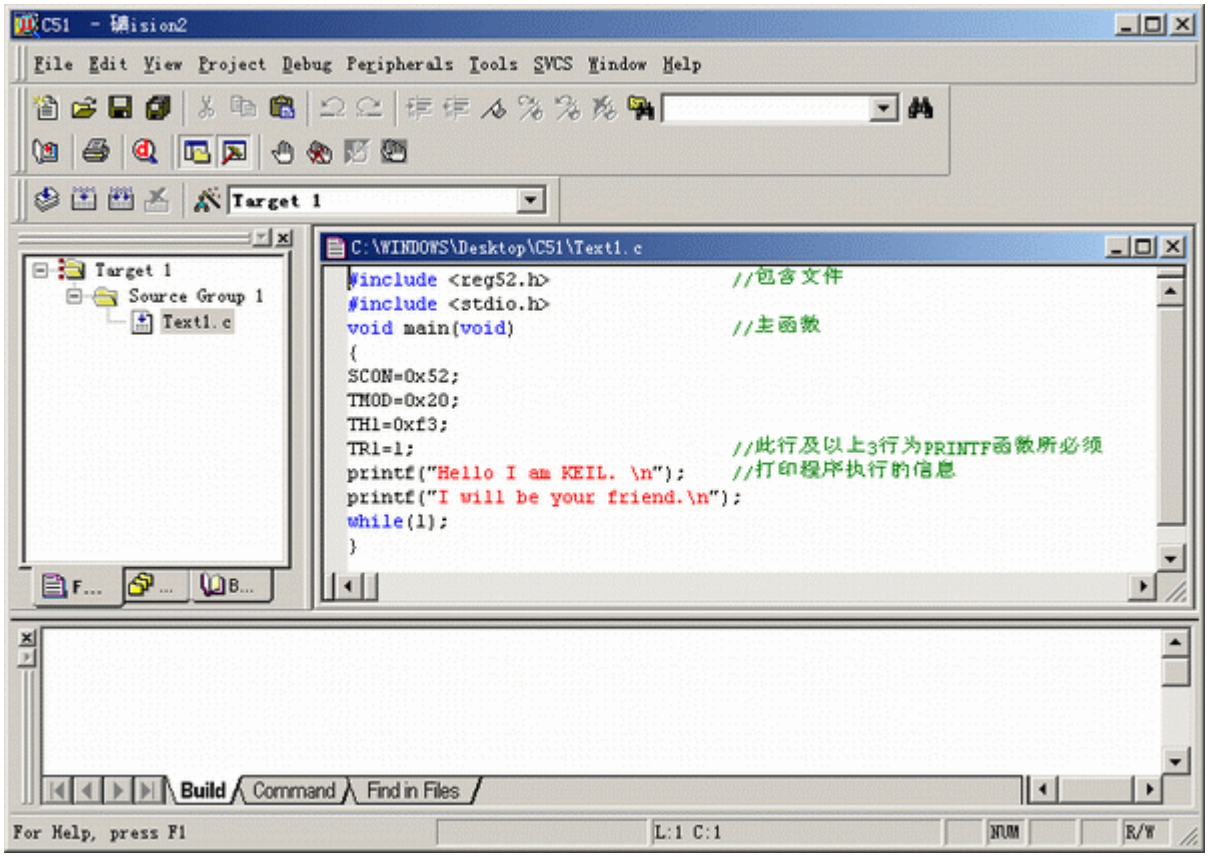


注意到“Source Group 1”文件夹中多了一个子项“Text1.c”了吗？子项的多少与所增加的源程序的多少相同。

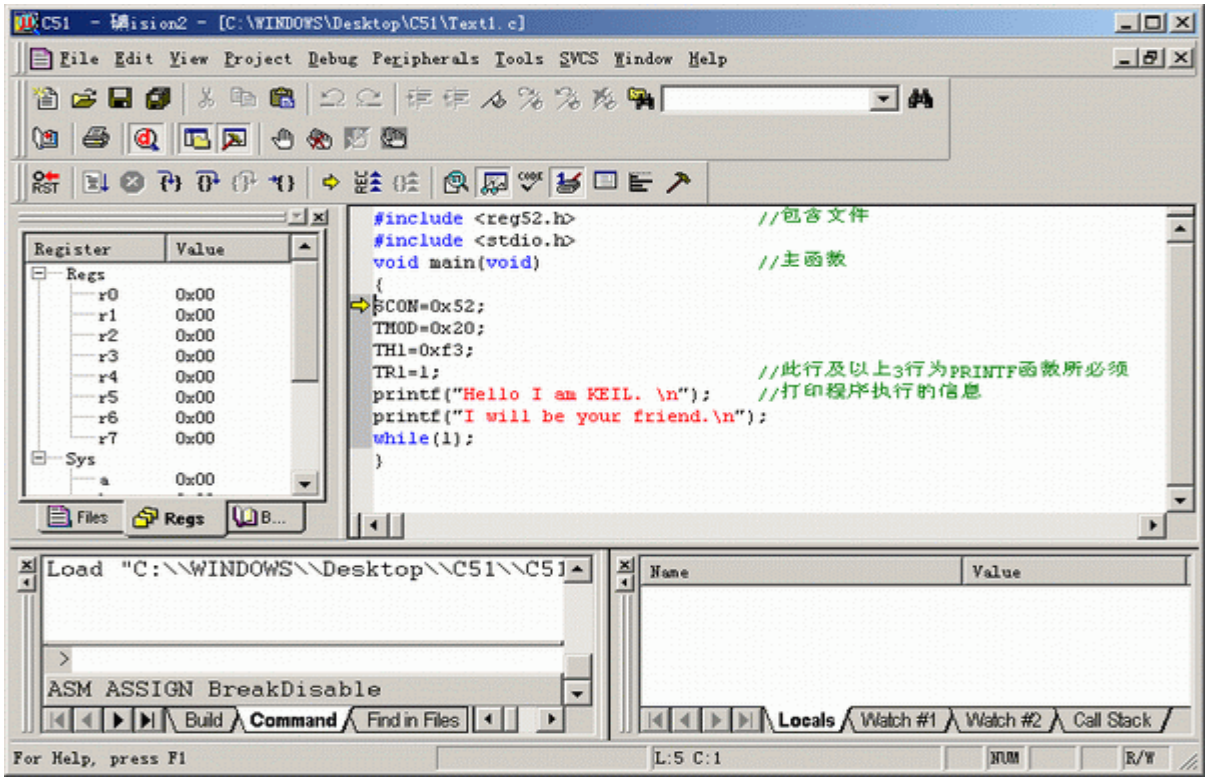
7）现在，请输入如下的 C 语言源程序：

```
#include <reg52.h>                    //包含文件
#include <stdio.h>
void main(void)                       //主函数
{
SCON=0x52;
TMOD=0x20;
TH1=0xf3;
TR1=1;                                //此行及以上 3 行为 PRINTF 函数所必须
printf("Hello I am KEIL. \n");              //打印程序执行的信息
printf("I will be your friend.\n");
while(1);
}
```

在输入上述程序时，读者已经看到了事先保存待编辑的文件的好处了吧，即 Keil c51 会自动识别关键字，并以不同的颜色提示用户加以注意，这样会使用户少犯错误，有利于提高编程效率。程序输入完毕后，如下图所示：

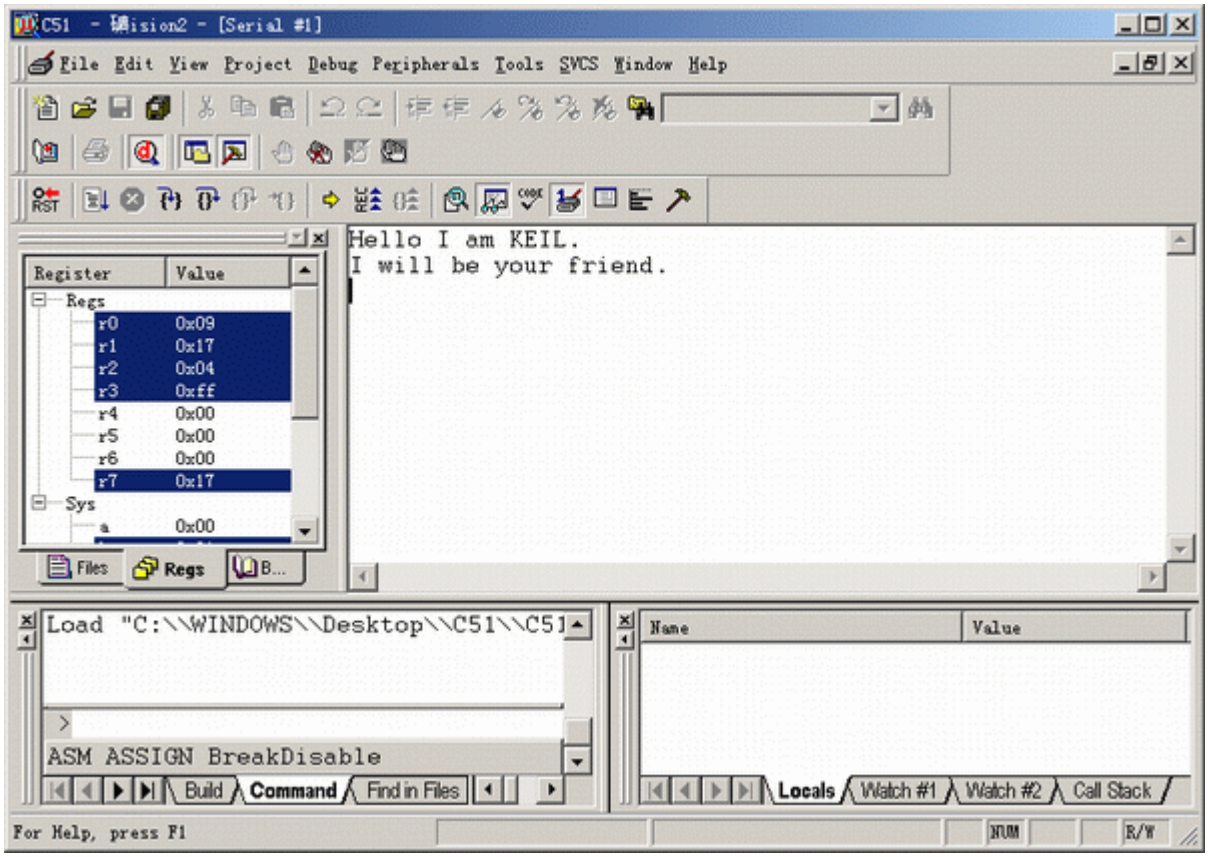


8）在上图中，单击“Project”菜单，再在下拉菜单中单击“Built Target”选项（或者使用快捷键F7），编译成功后，再单击“Project”菜单，在下拉菜单中单击“Start/Stop Debug Session”（或者使用快捷键 Ctrl+F5），屏幕如下所示：



9）调试程序:在上图中，单击“Debug”菜单，在下拉菜单中单击“Go”选项，（或者使用快捷键

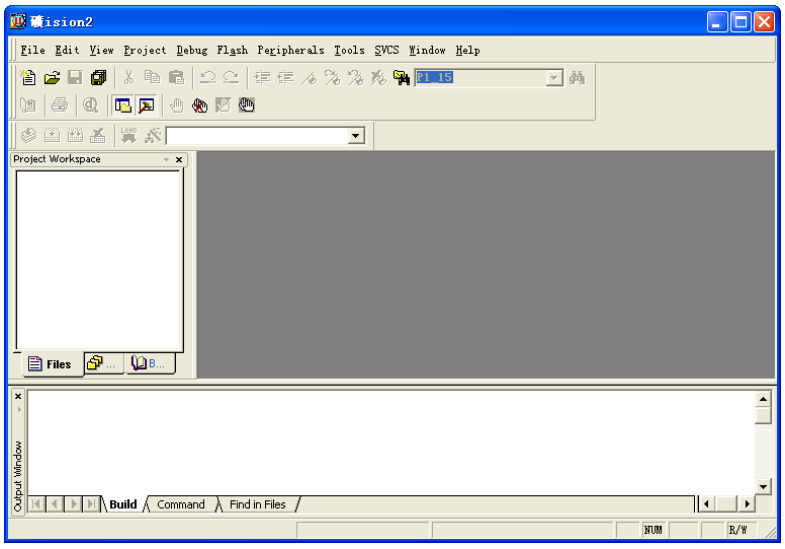
F5），然后再单击“Debug”菜单，在下拉菜单中单击“Stop Running”选项（或者使用快捷键 Esc）；再单击“View”菜单，再在下拉菜单中单击“Serial Windows #1”选项，就可以看到程序运行后的结果，其结果如下图所示



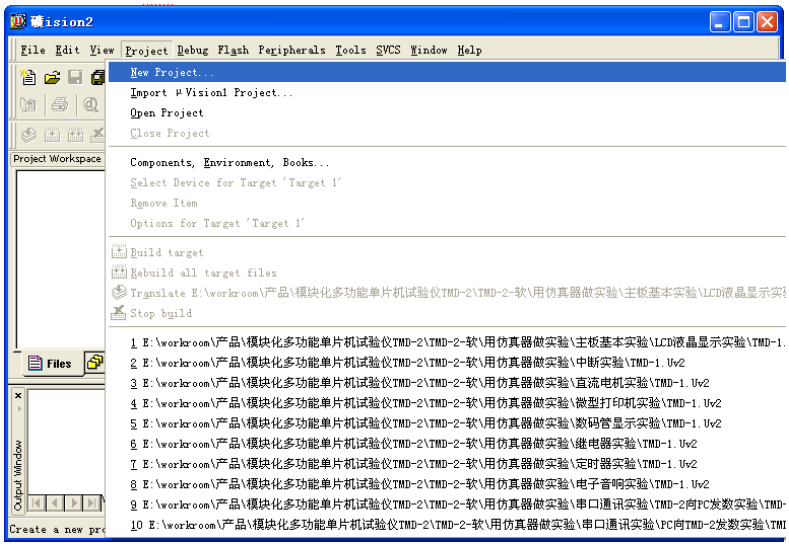
三、Keil 仿真器使用说明

1）、建立 keil 工程：

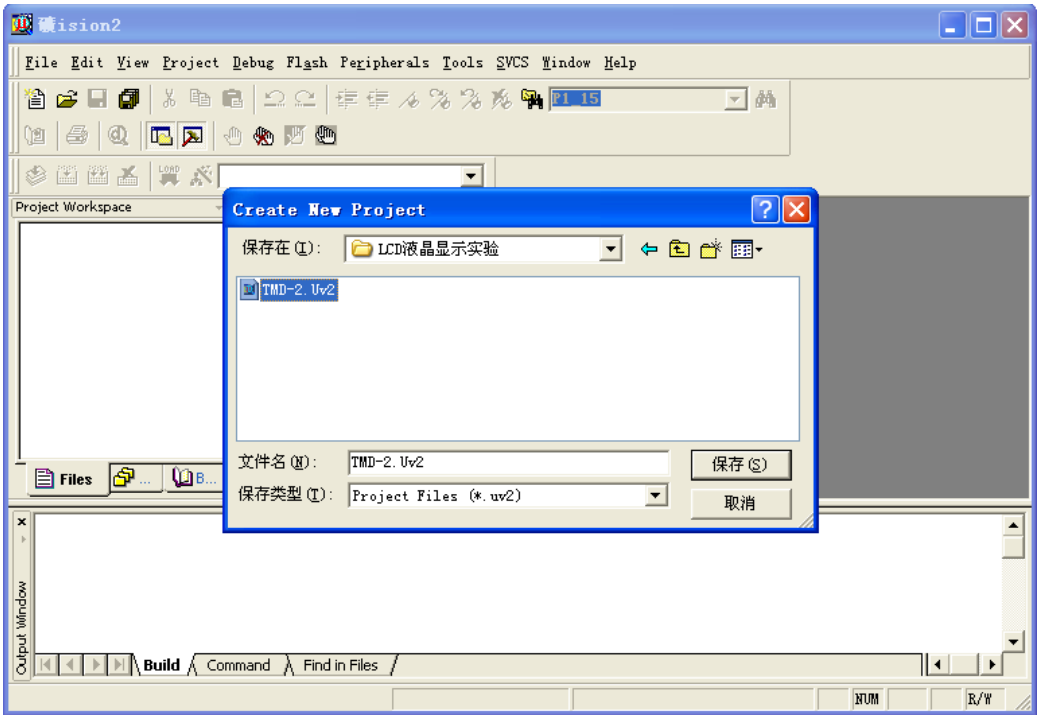
打开 keil 软件，如下图：



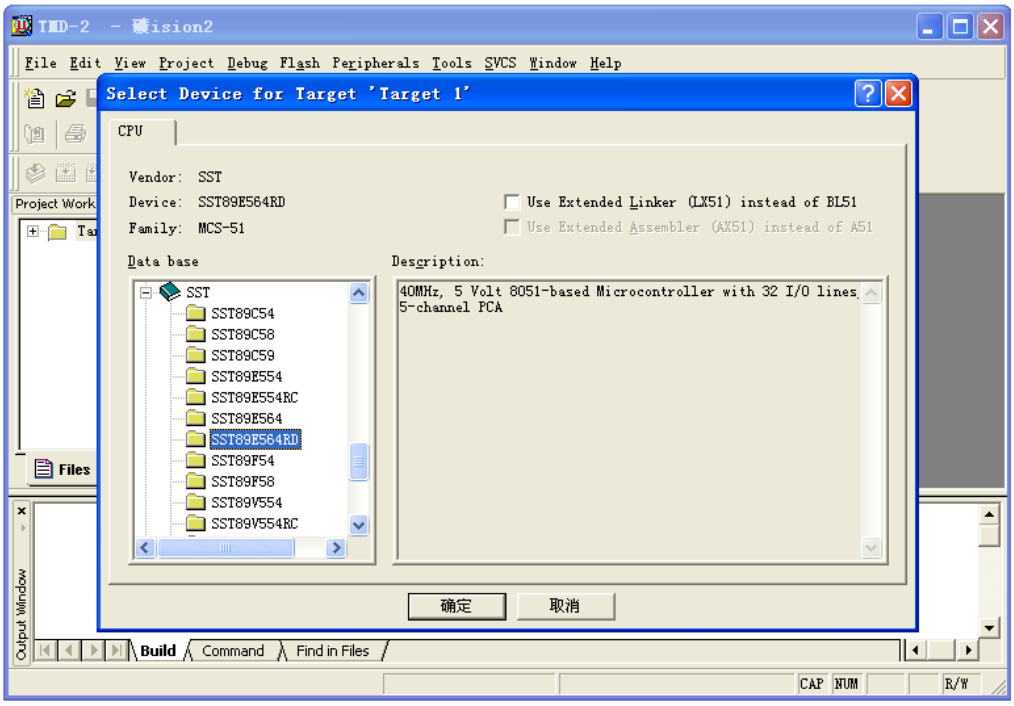
1）、创建新工程下拉菜单 Project /New Project...，如下图：



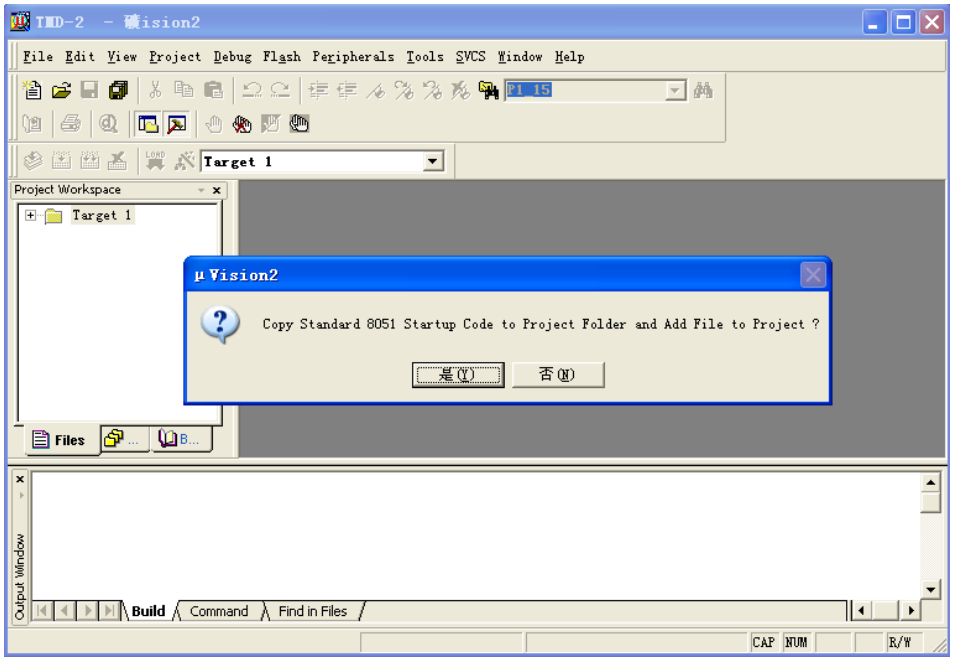
选择新建工程的名字和将要保存的路径，如下图：



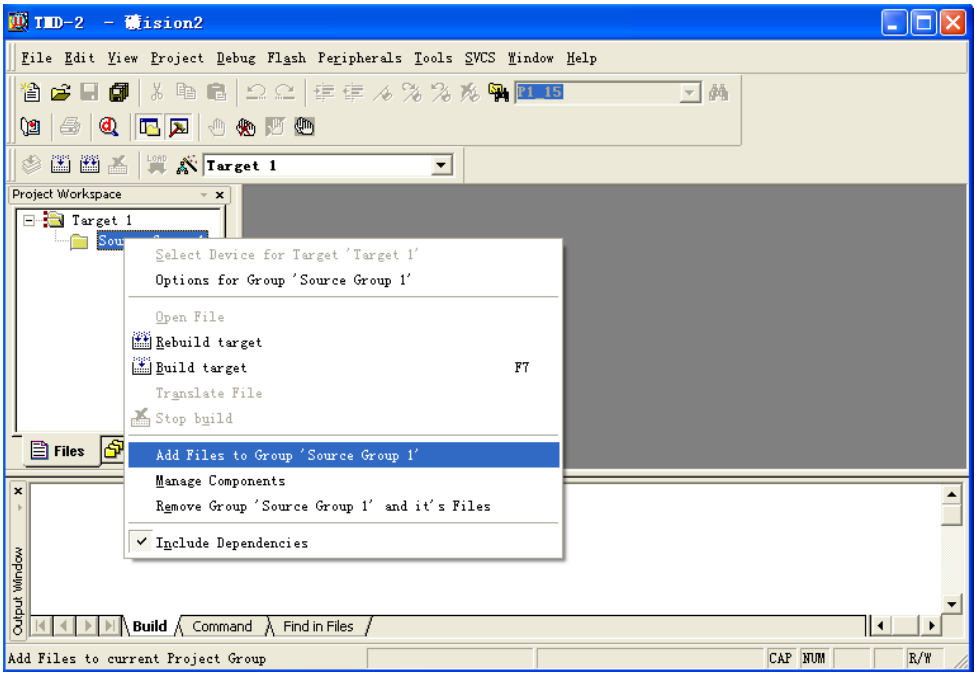
2） 选择所用单片机的型号，实验台采用 SST89E564。如下图：



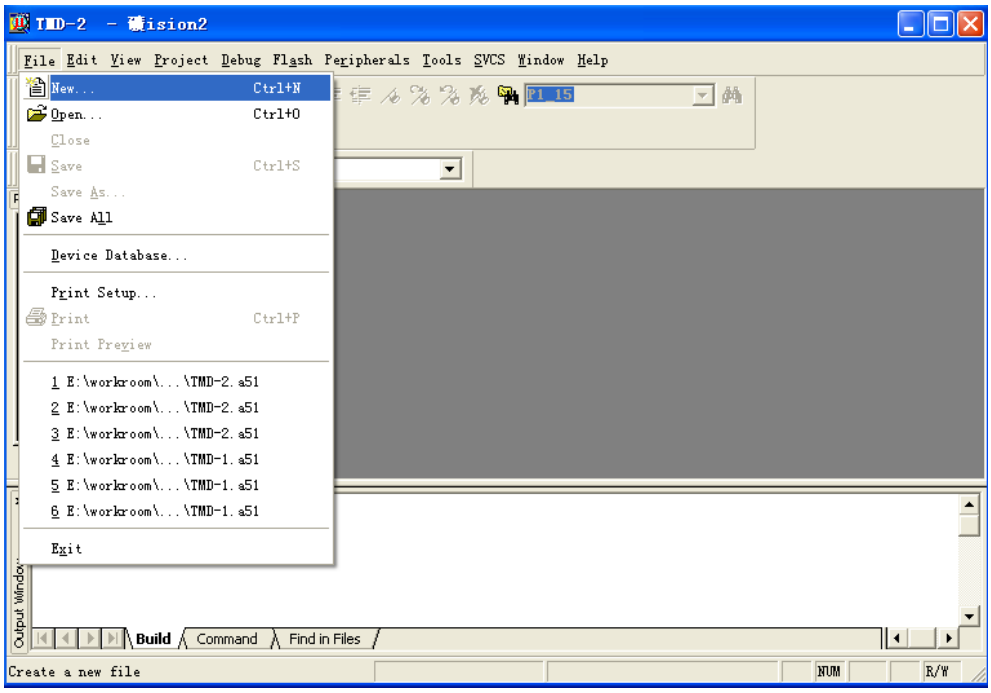
3） 选完单片机型号就会出现下图的界面，选择‘否’。



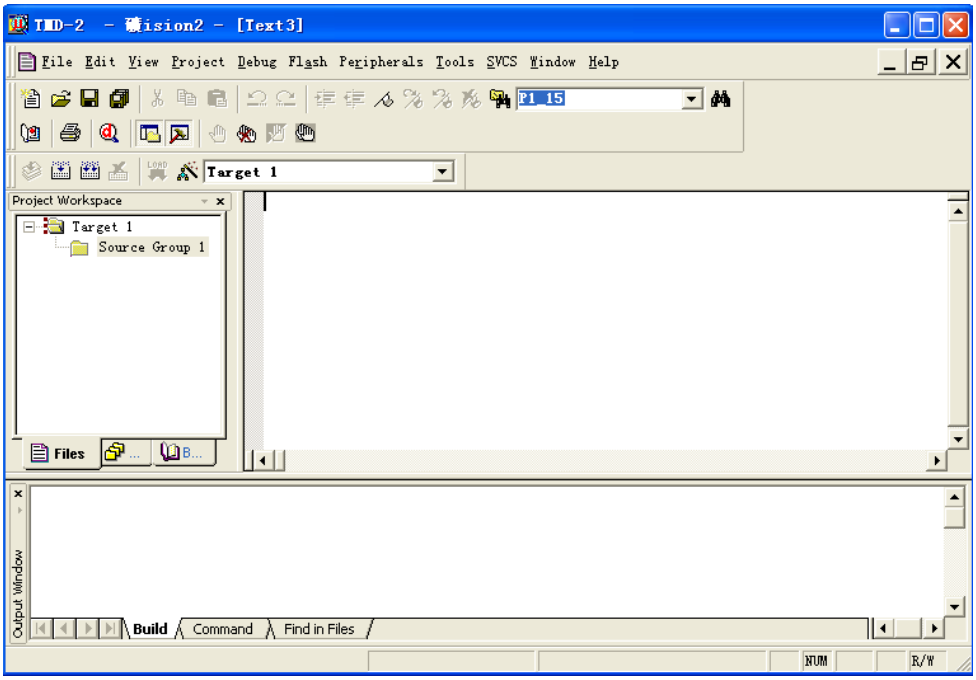
4） 以上建立了 Tsrget，再加上 xx.a51 文件就行了，建立 xx.a51 文件有两种办法，第一种是文件已经存在存储空间上了，那么就鼠标在 Source Group1 上右键，Add Files to Group’ Source Group1’选项，如下图，然后选择所要加的文件就可以了

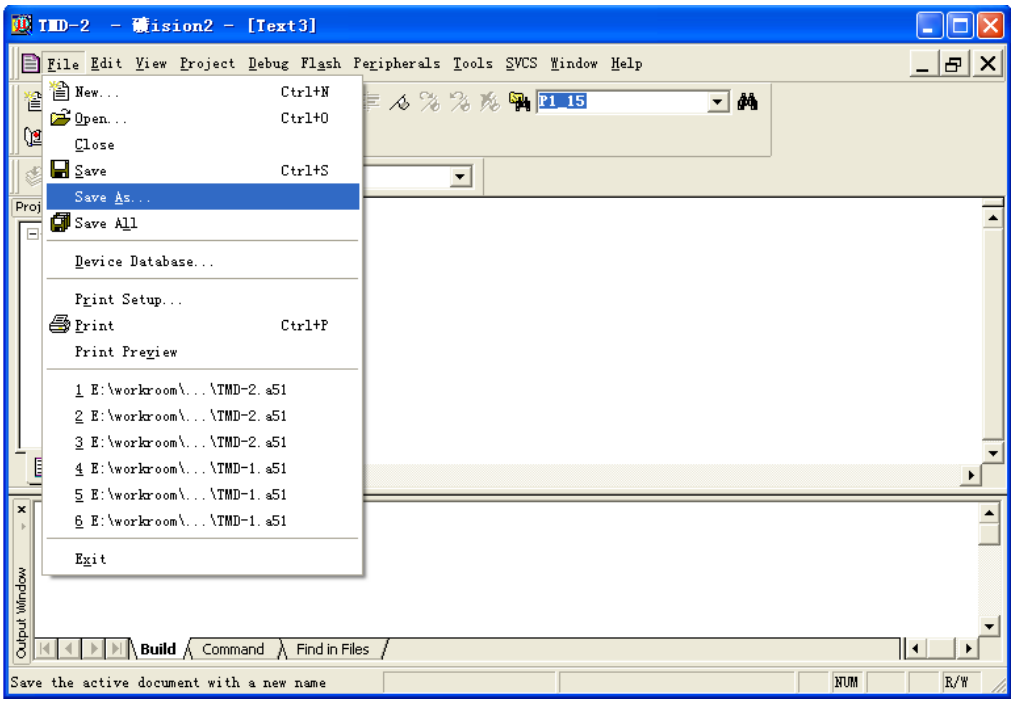


5） 建立 xx.a51 文件的第二种方法就是新建空白文件，如下图：

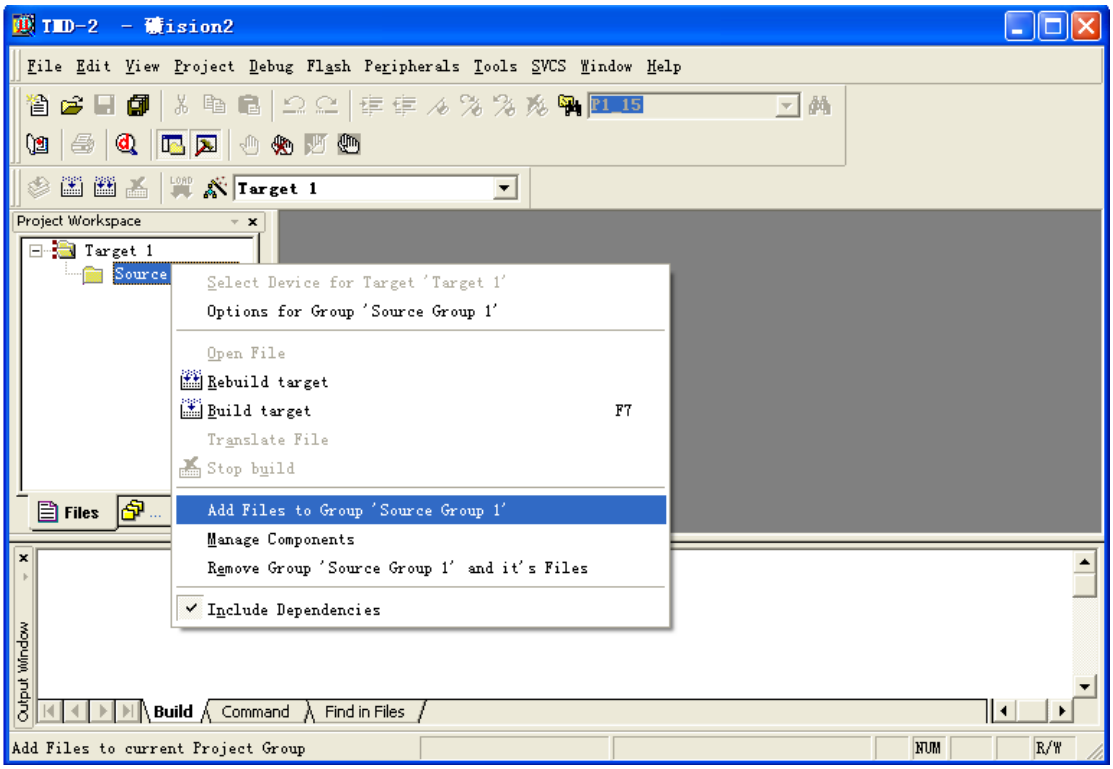


6） 新建完了文件，不要书写再将空白文件另存为 xx.a51 文件在与工程同目录下，如下图：

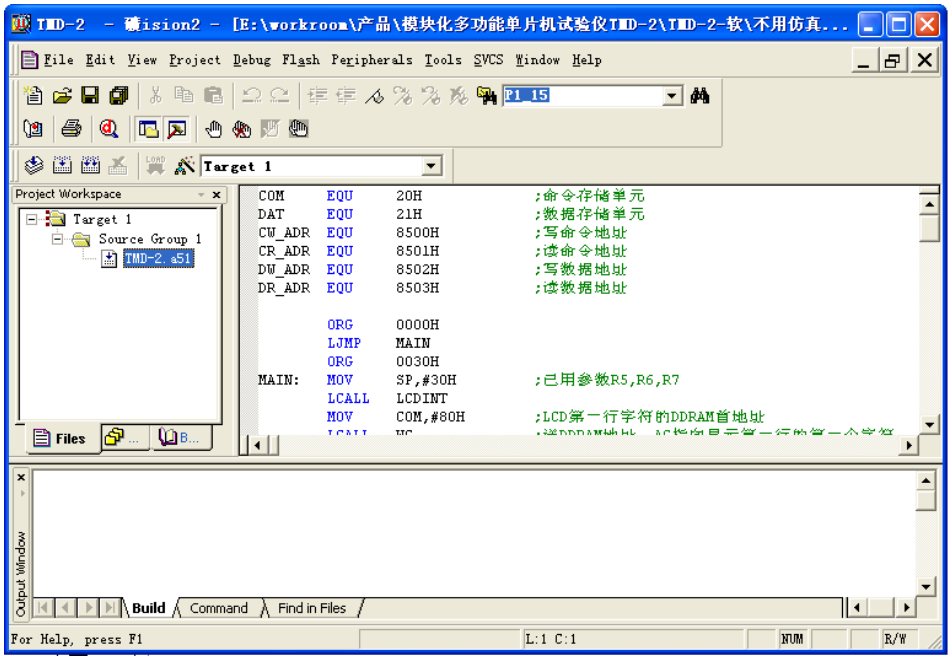




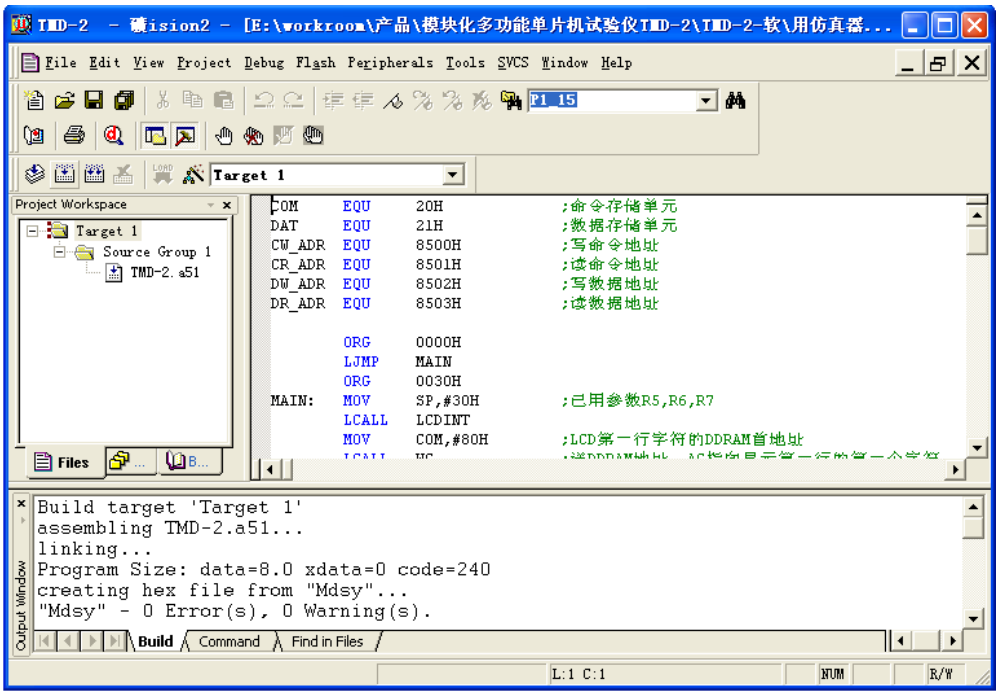
7） 最后再将 xx.a51 文件加载进来就可以了，如下图：



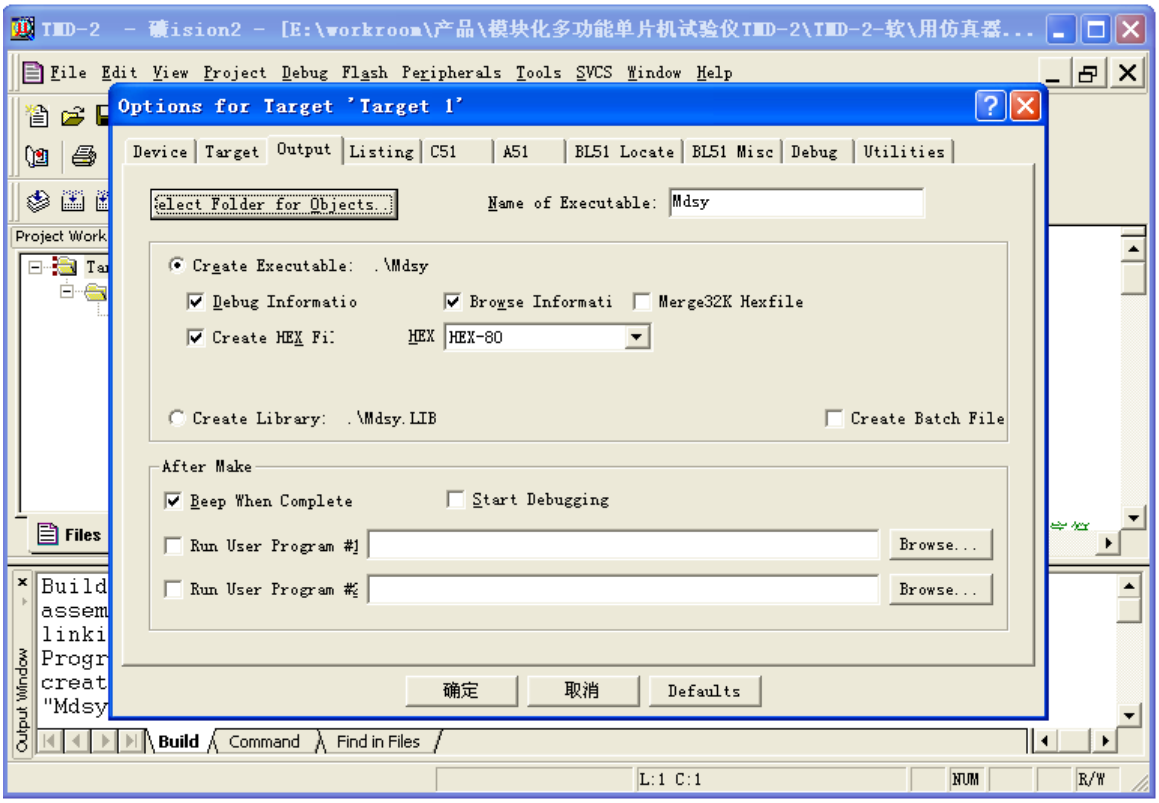
8） 双击左边工程空间里的.a51 的文件名字，右边就会出现.a51 的编辑空间，然后就可以编写需要的程序了。如下图：



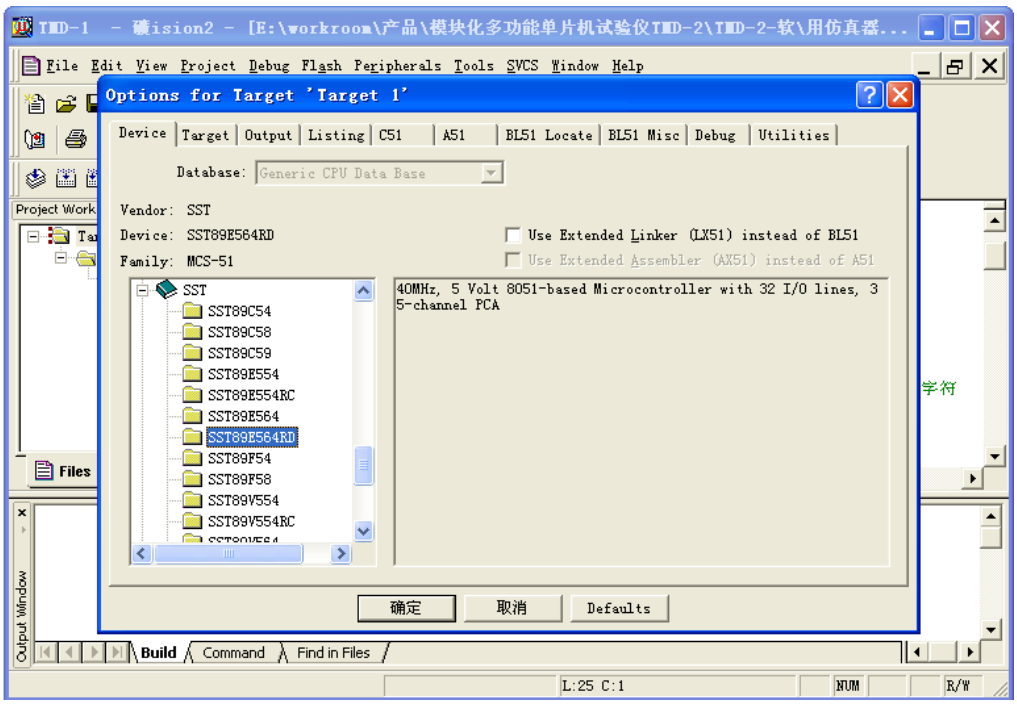
9） 程序编写完毕，需要编译，点击  快捷键，就可以了，如下图：



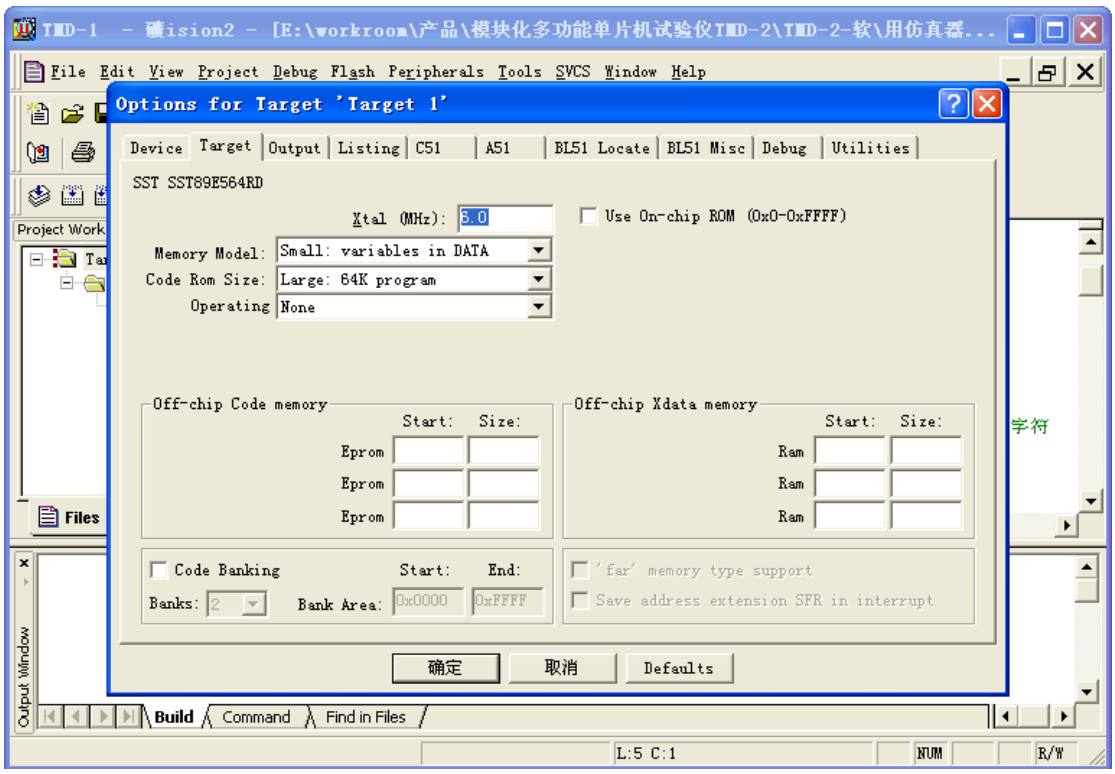
10）在最下面的 build 里可以观察有几个 Errors，几个 Warnings，有没有产生.hex 文件等信息，如果 Errors 和 Warnings 的话，说明程序还需要进一步更改。如果没有.hex 文件产生的话，那就需要在 Project/Options for Target 'Target 1'里的 Output 里设置，如下图。左边，将 Create HEX File 选项勾上即可。



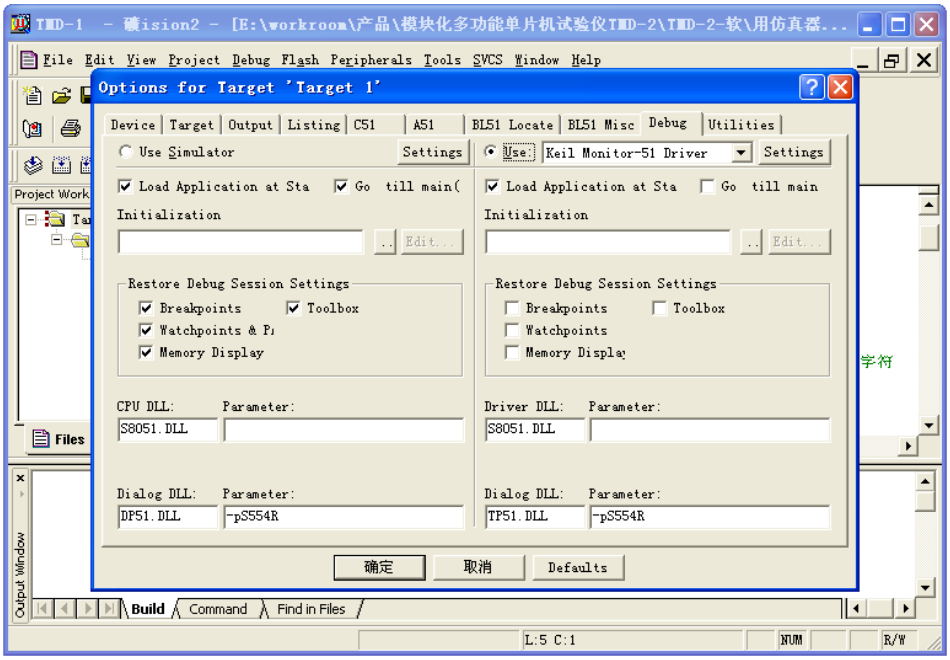
11）如果一切都没有问题，那么就可以用两种方法来执行此程序了，一种是用单片机执行，用 FlashMagic 软件在线下载的方法，介绍一下第二种方法：仿真器法。需要在 Project/Options for Target 'Target 1'里的 Device 里选择仿真器的设备型号，我们用的是 SST89E564RD，如下图：



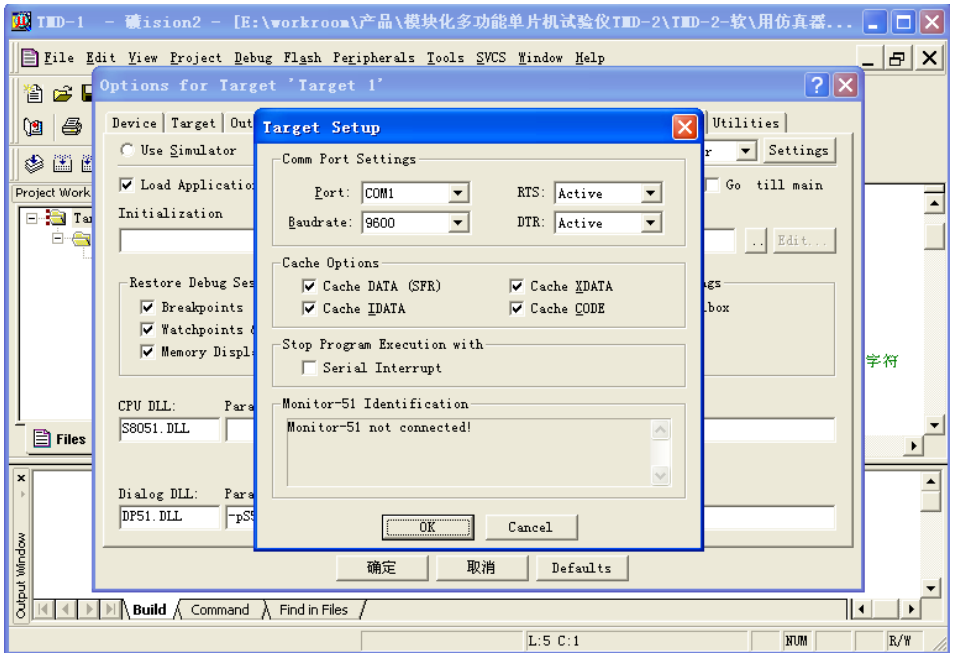
12）在 Target 里设置开发板的晶振，我们用的是 11.0592M，如下图：



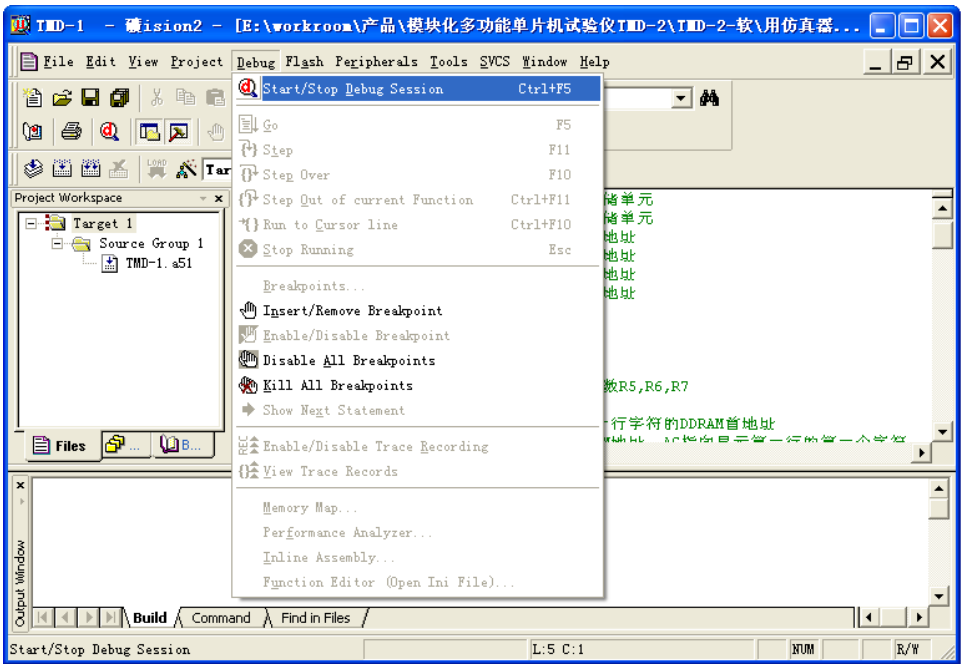
13)Debug 里设置仿真选项，右上角选中 Use[Keil Monitor-51 Driver]，下面的 Load Application at Sta 也选中，如下图：



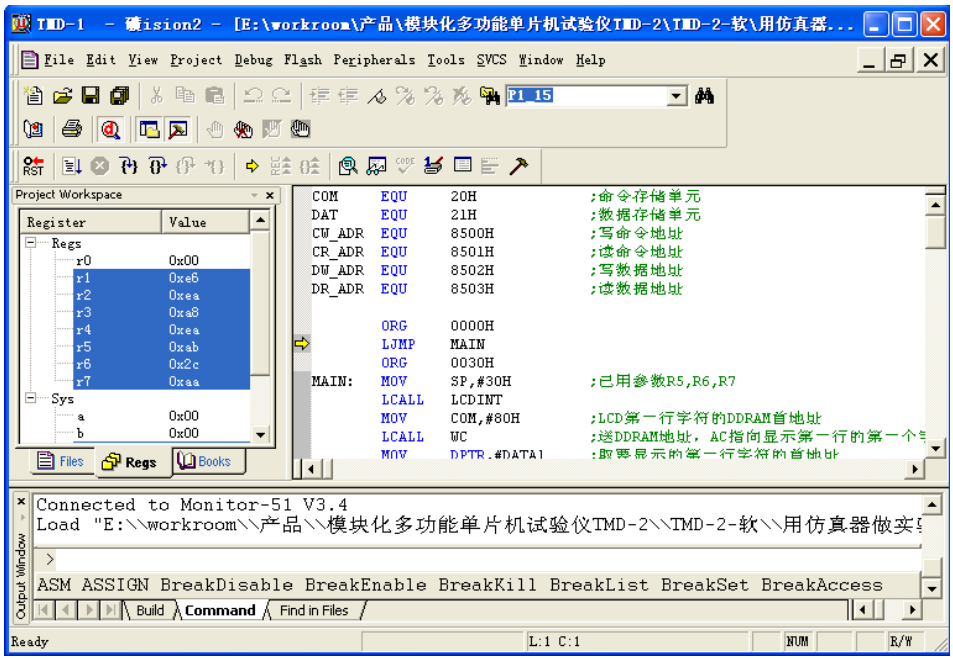
14）Settings 里设置串口和波特率等，串口选择 PC 机使用的 COM 号，波特率默认的 9600 就可以，再高也可以，RST，DTR 均选择 Active，Catch Options 必须全选择，能够提高通讯速度，否则慢的无法正常运行的。但是 Stop Program Execution with Serial Interrupt 不能选择，如下图：



16）、到此，仿真设置全部完成。联机。将核心板中部的开关拨到“仿真器”，确定电源线和串口线连接没有问题，然后上电，点击下拉菜单 Debug/Start/Stop Debug Session，如下图：



17）、进入仿真界面，如下图：



18）、介绍一下 keil 仿真器的快捷键功能：

首先，说明一下本仿真器与普通的 8051 仿真器的用法基本一样，所以在这就简单说一下。

：程序复位按钮。


：程序运行按钮，与单片机运行一样，程序不停止，只能点击  将程序挂起，然后再点击

，回到初始状态。


：Step in 单步按钮 1，程序单步执行到子程序的时候，跟踪进子程序执行。


：Step over 单步按钮 2，程序单步执行到子程序的时候，把子程序当一步执行。


：Run to Cursor Line 按钮，表示程序运行到光标所在行的位置，这个功能很方便。

19）、再次点击下拉菜单 Debug/Start/Stop Debug Session，退出仿真环境。

特别注意：

1，用 keil 仿真器调程序时千万不能按主板上的复位键，否则下到单片机里面的程序代码将会消失，再进行仿真就不得不再退出仿真环境，重新联机进入了。

2，如果想要程序返回初始状态，只需点击 

就可以了，观察  光标符号回到程序开头，程序就可以重新执行了，主板上的硬件资源状态会直接跟着改变，而不是用复位键回到刚开机的状态走。

四、Flash Magic 在系统编程（ISP）软件的使用

1．在系统编程简介

进行单片机的实验或开发时，通常需要借助编程器将调试好的目标程序写入到单片机内部程序存储器中。普通的编程器价格从几百元到几千元不等。另外，在开发过程中，程序每改动一次就要拔下电路板上的芯片编程后再插上，也比较麻烦。

随着单片机技术的发展，出现了可以在系统编程（ISP）的单片机。ISP一般是通过单片机的串行接口对内部的程序存储器进行编程，如PHILIPS 公司的P89C51RX+、P89C51RX2单片机；ATMEL公司的AT89S8252单片机；WINBOND公司的W78E516等。利用在系统编程（ISP）的单片机，单片机的实验和开发不需要编程器，单片机芯片可以直接焊接到电路板上，调试结束即成成品，甚至可以远程在线升级单片机中的程序，使得单片机应用系统的设计、生产、维护、升级等环节都发生着深刻的变革。

实验仪附带有一片PHILIPS 公司新推出的高性能8 位单片机P89C51RD2/P89C51RA2，该单片机与MCS-51单片机引脚及指令集完全兼容。该单片机最大的优点是：其片内具有64KB/8KB闪速程序存储器，1KB的片内数据存储器，且同PC机连机后，可将目标程序直接写入片内程序存储器中，不再需要专用的编程器。

PHILIPS公司P89C51RD2/P89C51RA2编程方法：

1)、在断电的情况下，将P89C51RD2/P89C51RA2单片机插入实验仪通用单片机插座并锁紧。将实验仪与单片机之间的串行通信电缆连接好：通信电缆一端接在PC 的串口COM1，另一端接实验仪的C51模块九芯针上。COM-USB通讯线一端接在PC机的USB口，另一端九芯孔接在实验仪的C51模块上九芯针上（COM-USB通讯线使用请参考“COM-USB通讯线安装及使用说明”）。

2)、将仿真器与51单片机选择开关拨到“51单片机”一侧，单片机RXD，TXD信号无任何连线。

3)、将实验仪上的编程开关拨至“COM下载”位置，编程指示灯亮。按“复位”按钮使单片复位后即可按下面的说明编程。

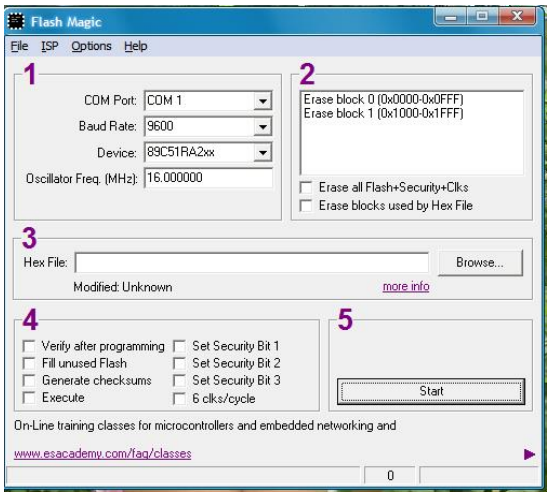
注：接上通讯线或拨下通讯线时，实验台必须在断电的情况下，否则容易损坏通讯口

2、在系统编程软件Flash Magic安装及使用说明

1）、将光盘中的FLASH Magic安装软件拷贝到硬盘上。

2）、双击Flash Magic安装软件图标，根据屏幕提示操作，程序会自动安装到PC机。

3）、运行安装完成的Flash Magic软件，如下图：



屏幕上方为主菜单，主菜单下方有屏幕被分成了5个区，分别标有1、2、3、4、5。下面对主菜单及编程方法做一说明。

主菜单简介

- File：包括打开和存储一个“HEX”文件，打开和存储一个设计文件和退出。
- ISP：包括芯片空检查；读保密位、读芯片标志字节、显示存储器内容、擦除FLASH等操作。
- Options：包括复位和高级选项操作，这两项操作在我们的实验系统中一般不用。
- HELP：软件帮助手册。

“1”区：

COM PORT　通讯口设置，可以通过下拉菜单在其中进行选择。也可以在输入框中直接输入所连接的通讯口。

BAUD RATE 波特率设置，可以通过下拉菜单进行选择，为了下载稳定建议不要选择太高的波特率。

DEVICE　器件选择，通过下拉菜单选择与所用的51单片机型号一样的。注意区分P89C51RD2HXX与P89C51RD2XX选择的型号是不一样的。

Oscillator Freq 振荡频率设置，实验系统频率为11.0592MHz。

“2”区：

擦除选中屏幕中所要擦除的FLASH块，

“3”区：

选择和打开一个HEX文件。

“4”区：

Verify after programming　编程后校验，在写芯片时须选中该项。

Fill unused Flash　填充用不到FLASH区，

Generate checksums　产生校验和，这项功能是在选择了一个HEX文件后，FLASH MAGIC在这个HEX文件所用到的每一块FLASH块的最高地址写入一个值，这个值使这个FLASH块的校验和为55H。

Execute　执行，如果选择该项功能后，将在编程完成后自动执行固化好的程序。注意：实验系统中由于使用硬件复位操作，这项功能将不起作用。

Set security 保密位。注意不要选择该处三个选项。

“5”区：

顺序执行所选择的操作：（没有选择项将跳过）

擦除块 — 编程 - 校验 - 填充没有用到的FLASH - 产生校验和的值 - 写时钟位 - 写保密位 - 执行固化好的程序

操作完成后，将显示“Finished …”并显示出编程所用的时间。

五、Microcontroller ISP Software 在系统编程

ISP 软件使用

1．在系统编程简介

进行单片机的实验或开发时，通常需要借助编程器将调试好的目标程序写入到单片机内部程序存储器中。普通的编程器价格从几百元到几千元不等。另外，在开发过程中，程序每改动一次就要拔下电路板上的芯片编程后再插上，也比较麻烦。

随着单片机技术的发展，出现了可以在系统编程（ISP）的单片机。ISP一般是通过单片机的串行接口对内部的程序存储器进行编程，ATMEL公司的AT89S8252单片机，利用在系统编程（ISP）的单片机，单片机的实验和开发不需要编程器，单片机芯片可以直接焊接到电路板上，调试结束即成成品，甚至可以远程在线升级单片机中的程序，使得单片机应用系统的设计、生产、维护、升级等环节都发生着深刻的变革。

实验仪附带有ATEML公司ATEML89S52/ATEML89S53，该单片机与MCS-51单片机引脚及指令集完全兼容。同PC机连机后，可将目标程序直接写入片内程序存储器中，不再需要专用的编程器。

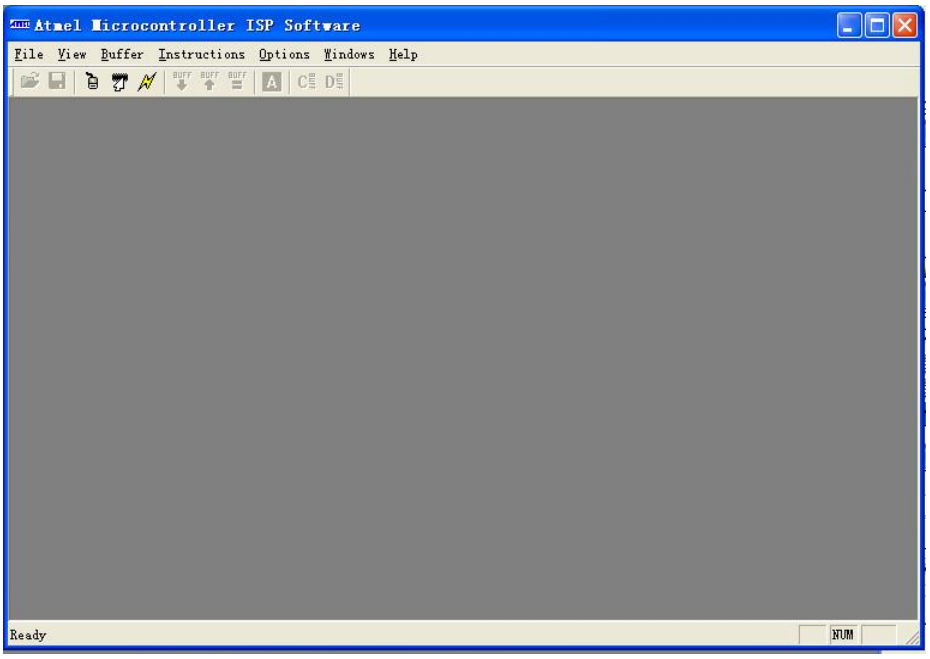
ATEML公司ATEML89S52/ATEML89S53编程方法：

1)、在断电的情况下，将ATEML89S52/ATEML89S53单片机插入实验仪通用单片机插座并锁紧。将实验仪与单片机之间的下载电缆连接好：下载电缆一端接在PC 的并口，另一端接实验仪的C51模块ISP下载口。

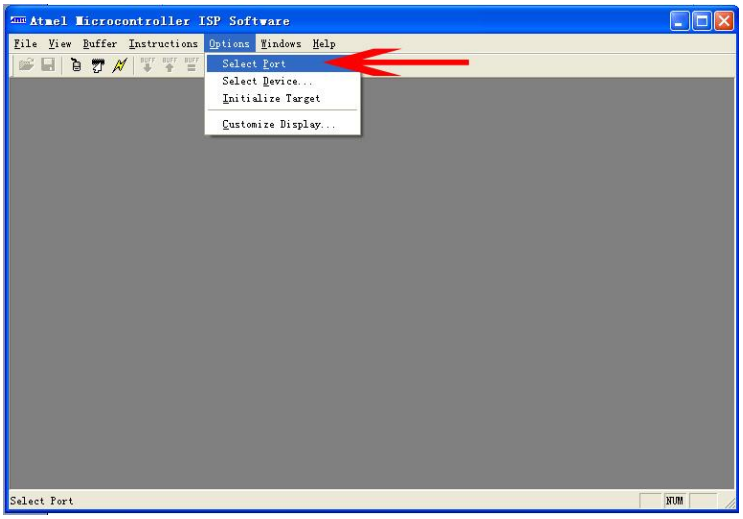
注：接上通讯线或拔下通讯线时，实验台必须在断电的情况下，否则容易损坏通讯口

2、在系统编程软件 Microcontroller ISP Software 使用说明

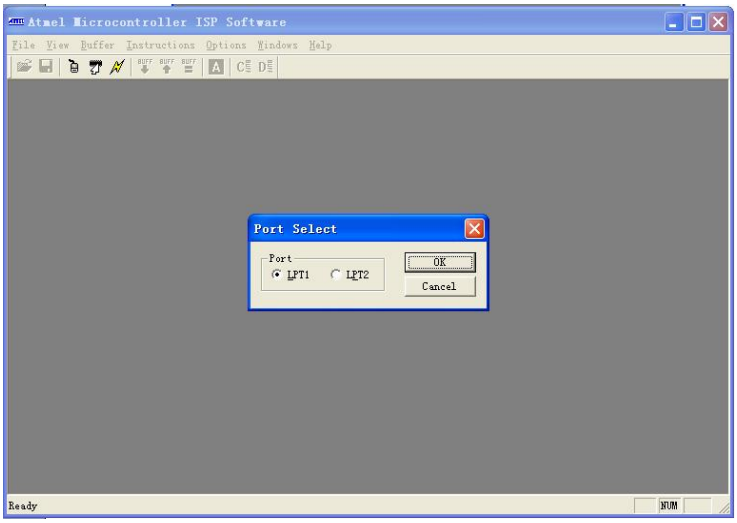
1）、运行安装完成的Microcontroller ISP Software软件，如下图：



2)、选择如下图红箭头所指菜单，（Options\Select Port菜单）设置下载端口：

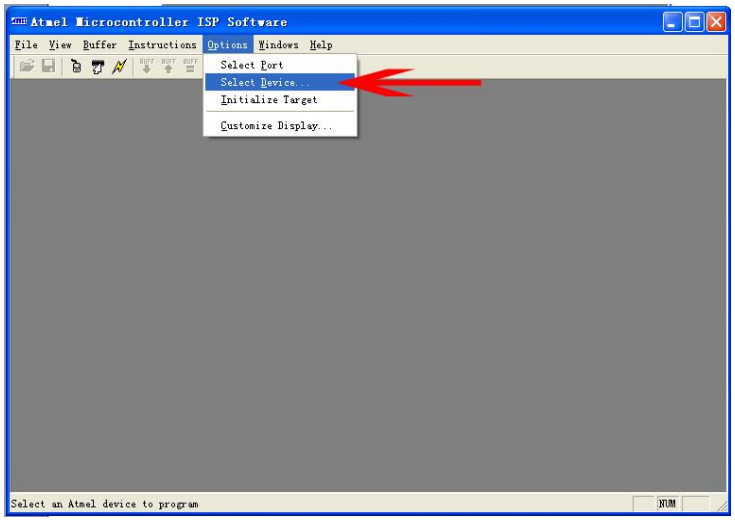


3）、选择功能菜单后如下图：

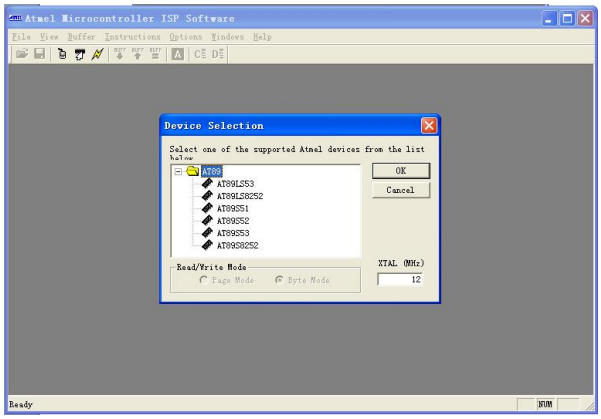


4）、选择PC端所使用的并口端号（LPT1或LPT2），点击“OK”

5）、选择下图红箭头所指功能菜单（Options\Select Device）。



6）、选择功能菜单，如下图选择使用芯片型号



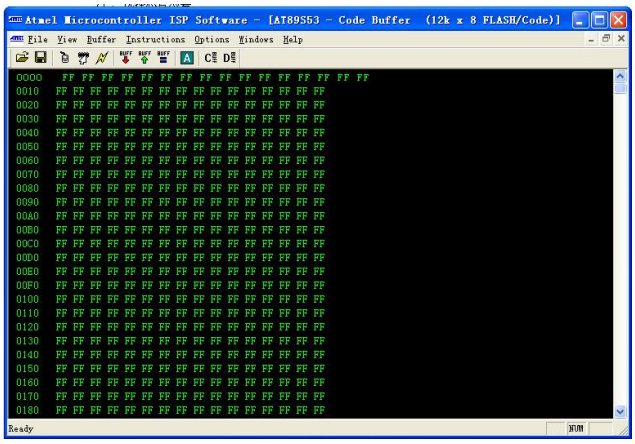
7）、选择芯片型号。

8）、选择读写模式为“Byte Mode”。

9）、选择时钟XTAL项为大板单片机所用时钟，该软件时钟只能为整数，实验系统C51单片机核心板上时钟为了方便与串口通信，采用了11.0592MHz时钟，因此该选项可以为11MHZ或12MHZ。

10）、点击OK，该软件部分型号会先与下端硬件连接，如果连接不成功会出现如下提示：

如果连接成功，如下图：



11）、上图工具栏中使用说明（从左到右顺序说明）：

1、打开需下载的十六进制文件。

2、保存缓冲区的内容。保存格式为十六进制。

3、设置软件使用端口。

4、选择设备选项，包括芯片型号、读写模式、时钟。

5、

6、写缓冲区内容到芯片。

7、读芯片内容到缓冲区。

8、比较芯片内容和缓冲区内容。

9、自动编程。

第二章、C51 硬件实验

说明：1、C51 单片机 ISP 下载时，请下载完成后再接线实验。下载完成后取出下载线。

2、PHILIPS 公司的 C51 单片机和 Atmel 公司的 C51 单片机在部分地方有些区别，请参考芯片资料后再编程，本实验系统所有实验均为 Atmel 公司的 89S52 单片机。

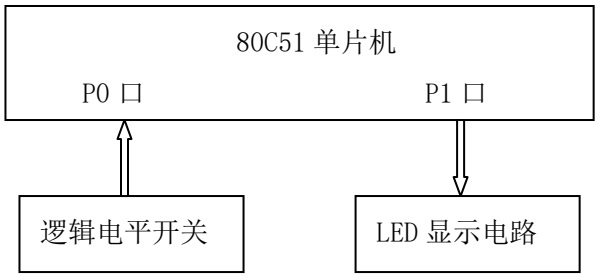
实验一　　端口 I/O 实验

一、实验目的

掌握8051单片机输入/输出端口的使用方法。

二、实验内容

1、从 8051 单片机 P0、P1、P2、P3 中任选 2 个端口，一个端口接逻辑电平开关（输入设备），另一个端口接 LED 显示电路（输出设备）。无条件将逻辑电平开关输入的数据传送给 LED 显示电路。例如，使用 P0 口输入、P1 口输出，实验电路如下。



2、任选一个端口接 LED 显示电路，编程使 8 个 LED 从左至右逐个发光（流水灯）。

3、接线：P07~P00 /C51 单片机　　接　　K7~K0 /逻辑电平开关

P17~P10 /C51 单片机　　接　　L7~L0 /LED 显示

三、实验原理

8051 单片机有 4 个 8 位的并行 I/O 端口：P0、P1、P2、P3，在不扩展存储器、I/O 端口，在不使用定时器、中断、串行口时，4 个端口，32 根口线均可用作输入或输出。作输出时，除 P0 口要加上拉电阻外，其余端口与一般的并行输出接口用法相同，但作为输入端口时，必须先向该端口写“1”。例如， P0 口接有-个输入设备，从 P0 口输入数据(注意：断开 J0)至累加器 A 中，程序段为：

```
MOV   P0, #0FFH
MOV   A,   P0
```

若将 P0.0 位的数据传送至 C 中，程序段为：

```
SETB   P0.0
```

```
        MOV    C,   P0.0
```

四、参考程序 （以 P0 口输入，P1 口输出为例）

1、 I/O 程序　　IO.ASM

```
        org     00h
main:   mov     P0,   #0ffh     ;向 P0 端口锁存器写 0FFH,准备输入
        mov     a,    p0        ;从 P0 口输入数据
        mov     P1,   a         ;将数据传至 P1 口
        sjmp    main
        end
```

2、流水灯程序　　LSD.ASM

```
        org     00h
        mov     a, #1
  loop: mov     p1,a                ;将 a 的内容通过 P1 口输出
        call    delay               ;调延时子程序
        rl      a                   ;a 左移一位
        sjmp    loop
 delay: mov     r0,   #80h          ;延时子程序
delay1: mov     r1,   #00h
delay2: djnz    r1,   delay2
        djnz    r0,   delay1
        ret
        end
```

实验二　　交通灯控制实验

一、实验目的

1、学习交通灯控制的方法。

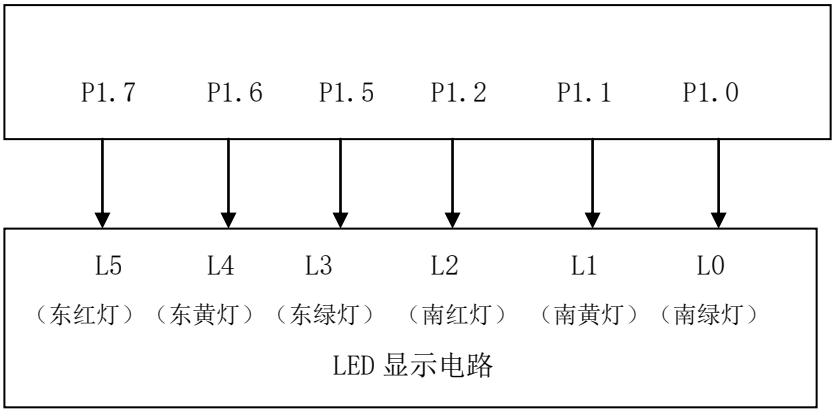
2、掌握 8051 单片机位操作指令的用法。

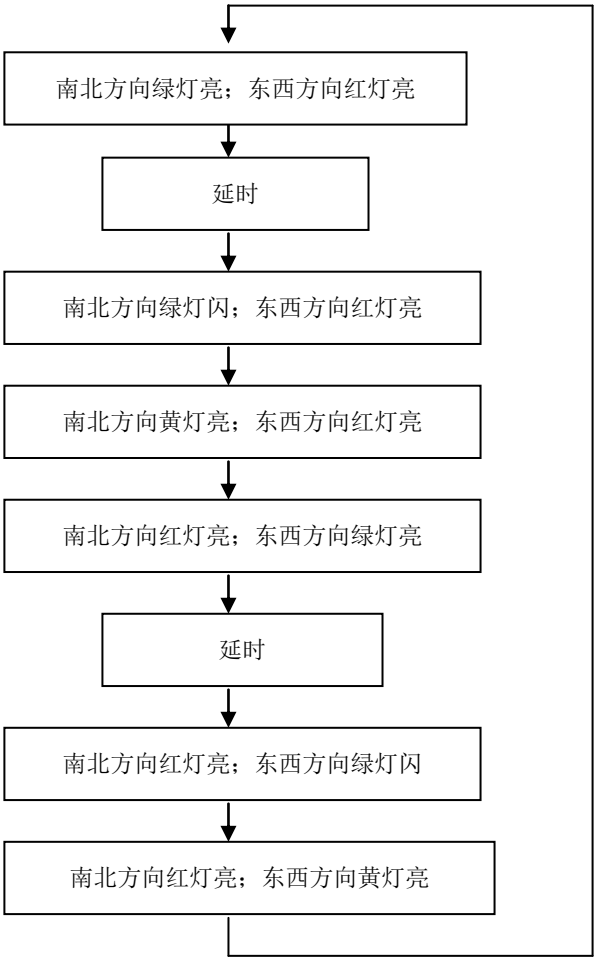
二、实验内容

1、通过单片机的P1口控制实验仪上6个LED按照交通灯的变化规律循环发光，模拟十字路口交通灯。通过一条SETB 指令，可使某一灯亮，通过一条CLR 指令，可使某一灯灭

2、接线：P17~P10　/C51单片机　　接　　L7~L0 /LED显示

三、参考程序流程图





四、参考程序　　JTD.ASM

```
        SG      EQU     P1.0        ;南北绿灯
        SY      EQU     P1.1        ;南北黄灯
        SR      EQU     P1.2        ;南北红灯
        EG      EQU     P1.6        ;东西绿灯
        EY      EQU     P1.6        ;东西黄灯
        ER      EQU     P1.7        ;东西红灯
        ORG     00h
ST1:    SETB    SG                  ;南北方向绿灯亮
        CLR     SY
        CLR     SR
        CLR     EG
        CLR     EY
        SETB    ER                  ;东西方向红灯亮
        MOV     R3,     #80         ;长延时
        CALL    DELAY
        MOV     R4,     #8          ;南北方向绿灯闪 4 次
ST2:    CPL     SG                  ;
        MOV     R3,     #2          ;短延时
        CALL    DELAY
        DJNZ    R4,     ST2
        CLR     SG                  ;南北方向绿灯灭
        SETB    SY                  ;南北方向黄灯亮
        MOV     R3,     #20         ;延时
        CALL    DELAY
ST3:    CLR     SY                  ;南北方向黄灯灭
        SETB    SR                  ;南北方向红灯亮
        CLR     ER                  ;东西方向红灯灭
        SETB    EG                  ;东西方向绿灯亮
        MOV     R3,     #80         ;长延时
        CALL    DELAY
ST4:    MOV     R4,     #8          ;东西方向绿灯闪 4 次
LP:     CPL     EG
        MOV     R3,     #2          ;短延时
        CALL    DELAY
        DJNZ    R4,     LP
        CLR     EG                  ;东西方向绿灯灭
        SETB    EY                  ;东西方向黄灯亮
```

```
        MOV     R3,     #20         ;延时
        CALL    DELAY
        SJMP    ST1                 ;转 ST1
DELAY:  MOV     R1,     #0          ;延时子程序
DELAY1: MOV     R0,     #0
DELAY2: DJNZ    R0,     DELAY2
        DJNZ    R1,     DELAY1
        DJNZ    R3,     DELAY
        RET
```

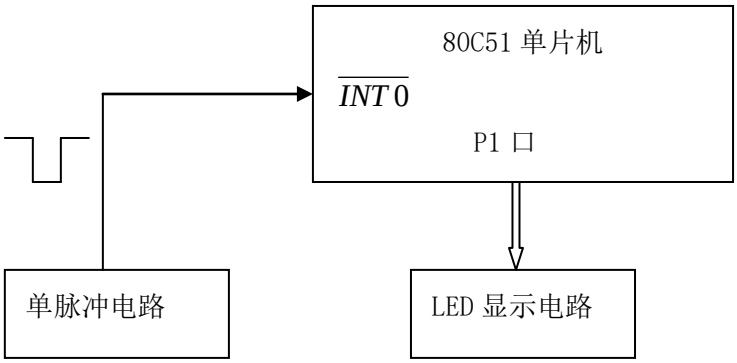
实验三　　外部中断实验

一、实验目的

掌握8051单片机外部中断的使用方法。

二、实验内容

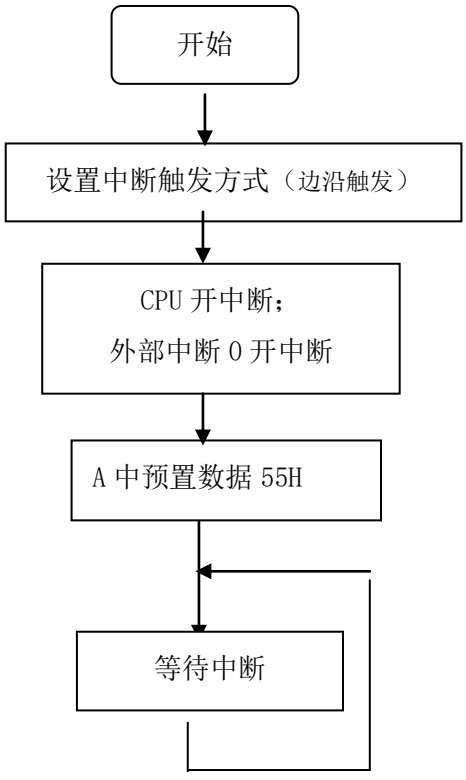
1、**实验电路如下图所示。**通过用手动逐个向 8051 单片机的 $\overline{INT0}$ 输入单脉冲，申请中断。每中断一次，依次使 8051 单片机向 P1 口输出 55H、AAH、55H……。



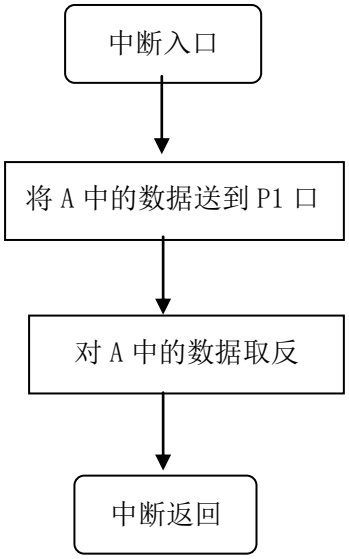
2、接线：P17~P10 /C51 单片机　　接　　L7~L0 /LED 显示

INT0 /C51 单片机　　接　　单脉冲

三、参考程序流程图



主程序流程图



中断服务程序流程图

四、参考程序　　INT0.ASM

```
        ORG     0000H
        AJMP    MAIN            ;跳转到主程序
        ORG     0003H           ;外部中断 0（ $\overline{INT0}$ ）的入口地址
        MOV     P1,     A       ;将 A 中的数据送到 P1 口显示
        CPL     A               ;对 A 中的数据取反
        RETI                    ;中断返回
        ORG     0020H           ;主程序地址
MAIN:   SETB    EA              ;CPU 开中断
        SETB    IT0             ;设置中断触发方式为边沿触发
        SETB    EX0             ;外部中断 0 开中断
        MOV     A,      #55H    ;A 中预置数据 55H
        SJMP    $               ;等待中断信号
```

实验四　　定时器实验

一、实验目的

1、 掌握8051单片机内部定时器的使用方法

2、 学习单片机控制蜂鸣器发声方法

二、实验内容

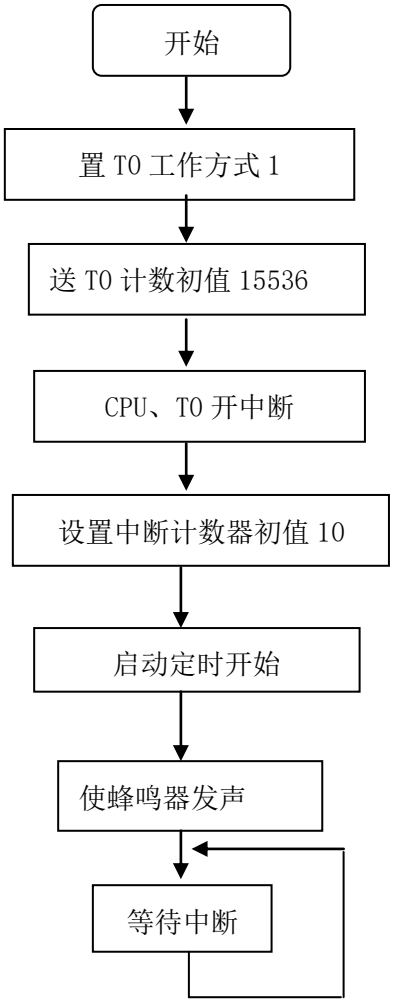
1、实验电路如图所示，用 8051 单片机内部定时器 T0 定时，控制蜂鸣器发声，使之发声 1 秒钟，停止 1 秒钟，重复循环。

2、接线：　P10 /51 单片机　接　蜂鸣器

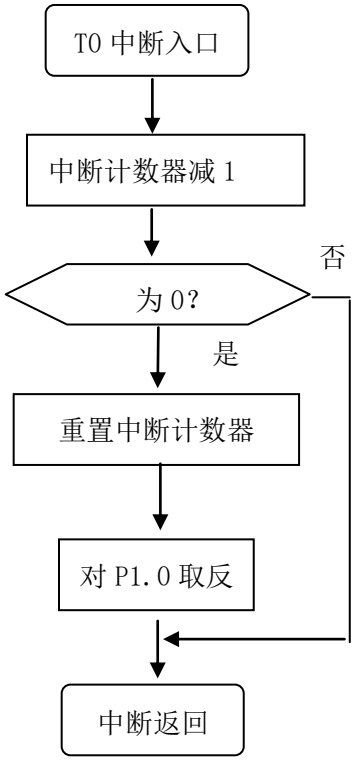
三、实验原理

当 P1.0 输出高电平时，三极管导通蜂鸣器发声，P1.0 输出低电平时，三极管截止，蜂鸣器不发声。本实验仪的晶振频率为 11.0592MHZ，定时方式时，约 1μs 计数器加 1，计数初值 15536 时，经过 50ms 溢出，溢出 20 次约为 1 秒钟。

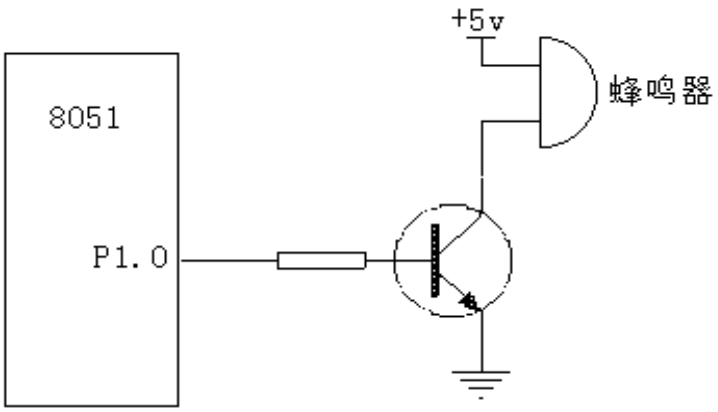
四、参考程序流程图



主程序框图



T0 中断服务程序框



五、参考程序　　DSQ.ASM

```
        ORG     00H
        AJMP    MAIN
        ORG     001BH               ;定时器 T0 中断服务程序
        DJNZ    R0,      EXIT       ;未中断 10 次，中断返回
        CPL     P1.0                ;中断 10 次到 1S，将 P1.0 取反
        MOV     R0,      #10        ;重置中断次数计数器
EXIT:   MOV     DPTR,    #15536     ;重置 T0 计数初值 15536
        MOV     TH0,     DPH
        MOV     TL0,     DPL
        RETI
 MAIN:  MOV     TMOD,    #01H       ;设置定时器 0,方式 1
        MOV     DPTR,    #15536     ;送 T0 计数初值 15536
        MOV     TH0,     DPH
        MOV     TL0,     DPL
        SETB    EA                  ;CPU 开中断
        SETB    ET0                 ;T0 开中断
        SETB    P1.0                ;蜂鸣器发声
        MOV     R0,      #10        ;R0 为中断次数计数器
        SETB    TR0                 ;启动定时开始
        SJMP    $
```

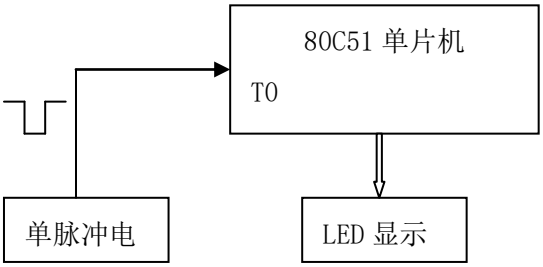
实验五　　计数器实验

一、实验目的

掌握8051单片机内部计数器使用方法。

二、实验内容

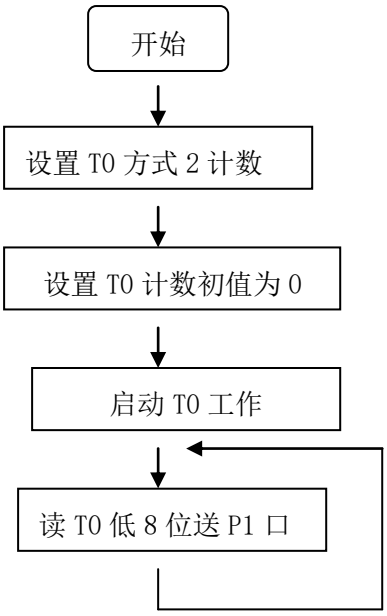
1、实验电路如图，设置8051单片机内部定时器/计数器T0计数，按方式2工作，对T0引脚（P3.4）手动输入的单脉冲进行计数。并将其计数值从P1口输出，在LED上显示出来。验证其正确性。



2、接线：　T0 /C51单片机　接　单脉冲

P17~P10　接　L7~L0 /LED 显示

三、参考流程图



四、参考程序　　JSQ.ASM

```
        ORG  0000H
        MOV  TMOD,   #06H    ;T0 计数,方式 2
        MOV  TH0,    #0      ;计数初值 0
        MOV  TL0,    #0
        SETB TR0             ;启动 T0
LOOP:   MOV  P1,     TL0     ;读 T0 送 P1 口
        AJMP LOOP
```

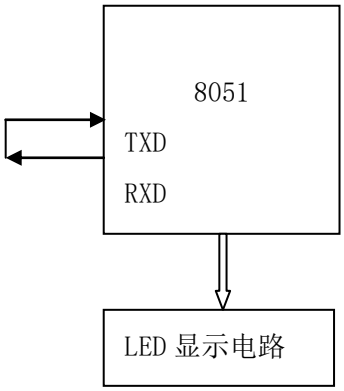
实验六　　串行口通信实验

一、实验目的

掌握单片机串行口工作方式的设定及串行通信程序设计。

二、实验内容

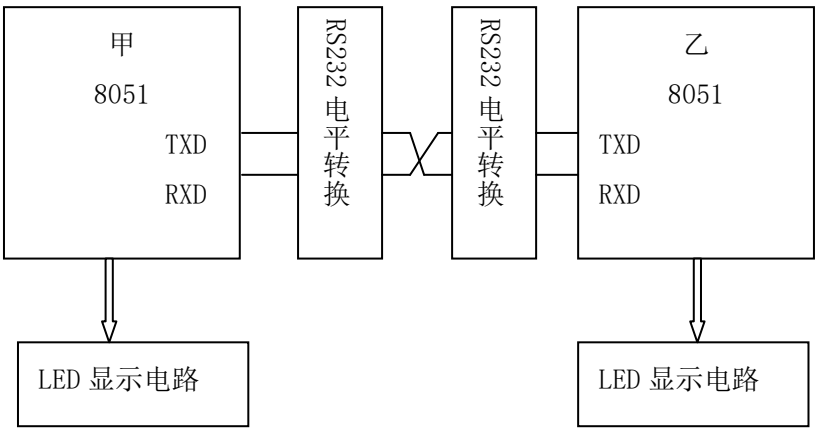
1、 自发自收实验。实验电路如图（a）所示。编程通过串行口循环发送数据00H~FFH，并自收回来在LED显示出来。



(a)

接线：　TXD /C51单片机　接　RXD /C51单片机

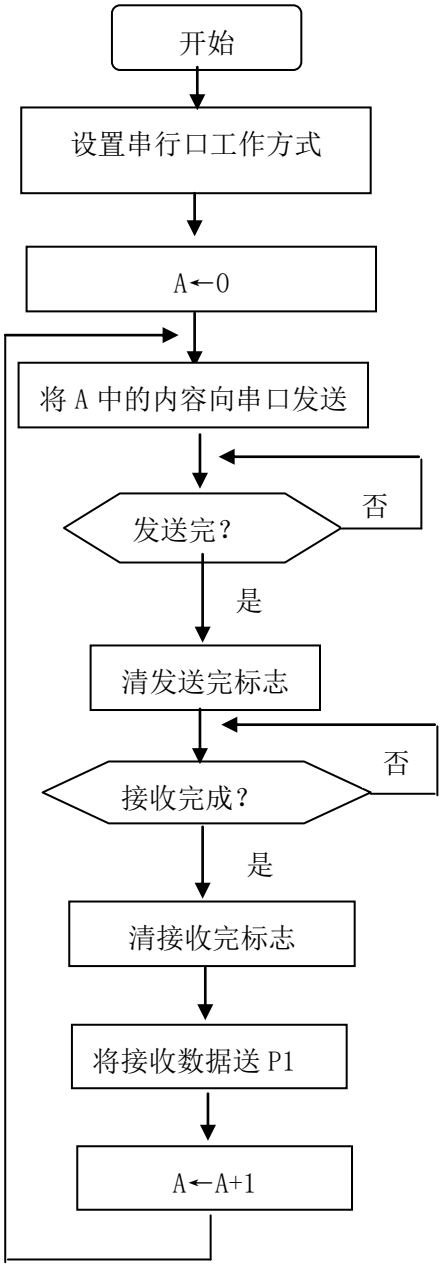
2、（选做）互发互收实验。实验电路如图（b）所示。甲单片机通过串行口循环发送数据0~FFH，乙单片机通过串行口接收并在LED显示出来。



(b)

接线：用 9 芯通讯线连接二台实验台上 C51 核心板上的 9 芯通讯头。

三、参考流程图（自发自收）



四、参考程序　　COM.ASM

```
        org     00h
        mov     scon,#10010000B     ;设置串行口方式 2 工作
        mov     a,#00h
start:  inc     a
        mov     sbuf,a              ;将 A 中的内容发送
wait0:  jnb     ti,wait0            ;等待发送完成
        clr     ti                  ;清发送中断标志
wait:   jnb     ri,wait             ;等待接收完成
        mov     a,sbuf              ;从接收缓冲器读入数据
        mov     p1,a                ;送到 P1 口显示
        clr     ri                  ;清接收中断标志
        call    delay               ;延时
        ljmp    start
delay:  mov r0,  #0                 ;延时子程序
        mov r1,  #0
lp:     djnz r1, lp
        djnz r0, lp
        ret
```

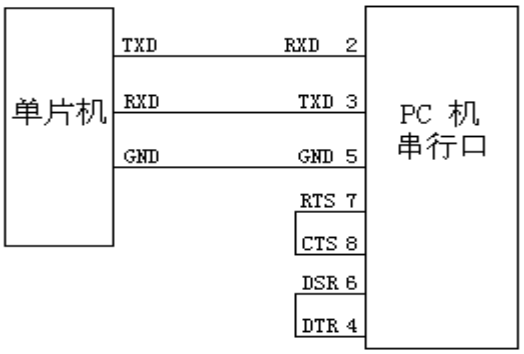
实验七　　PC 机与单片机通信

一、实验目的：

掌握利用单片机串行口与 PC 机通信的方法。

二、实验原理与内容

1、实验电路如右图所示。通过串行通信电缆将单片机的串行口与 PC 机的串行口相连。PC 机侧接 COM1 或 COM2。要求两机之间全双工通信，波特率为 9600, 字符格式为：8 个数据位，奇校验,1 个停止位。



单片机侧：P1 口接逻辑电平开关，通过逻辑电平开关输入 ASCII 码发送给 PC 机； PC 机接收的数据在串口调试助手接手窗口以对应的十六进制数显示出来。

PC 机侧：通过键盘输入字符或数字发送给单片机；接收单片机发来的数据送 P1 口显示出来。

PC 机上端控制软件可采用“串口调试助手”。

2、接线：1）单片机发送，PC 机接收

P1.7~P1.0 /C51 单片机　接　K7~K0 /逻辑电平开关

用串口通讯线连接 C51 单片机通讯口和 PC 机 COM 口

2）单片机接收，PC 机发送

P1.7~P1.0 /C51 单片机　接　L7~L0 /LED 显示

用串口通讯线连接 C51 单片机通讯口和 PC 机 COM 口

三、参考程序：

1、发送数据参考程序：(对 fosc=11.0592MHZ,波特率=9600)

```
        org     0000h
        ajmp    main
        org     0100h              ;主程序开始
main:   mov     tmod,#20h          ;设置计数器为方式 2
        mov     th1,#0fdh          ;设置计数器初值
        mov     tl1,#0fdh
        setb    tr1
        clr     ea                 ;禁止中断
        mov     scon,#50h    ;设置串行通讯方式:方式 3,允许接收
        mov     pcon,#00h            ;smod 清 0
start:  mov     p1,#0ffh           ;读入 P2 口的数据
        mov     a,p1
```

```
        mov     sbuf,a          ;发送
wait:   jnb     ti,wait         ;等待发送完成
        clr     ti              ;清发送标志
        lcall   delay           ;延时
        sjmp    start
delay:  mov     r6,#255
del1:   mov     r7,#255
        djnz    r7,$
        djnz    r6,del1
        ret
        end
```

2、接收数据参考程序

```
org     0000h
        ajmp    main
        org     0023h           ;中断子程序
        clr     es              ;关中断
        mov     a,sbuf          ;读入数据
        mov     p1,a            ;送 P1 口显示
        clr     ri              ;清除接收标志
        setb    es              ;开中断
        reti

        org     0100h           ;主程序开始
main:   mov     tmod,#20h       ;设置计数器为方式 2
        mov     th1,#0fdh       ;设置计数器初值
        mov     tl1,#0fdh
        setb    tr1
        setb    ea              ;开中断
        setb    es
        mov     scon,#50h       ;设置串行通讯方式:方式 3,允许接收
        mov     pcon,#00h       ;smod 清 0
        ajmp    $
        end
```

实验八　七段并行数码管显示

一、实验目的

1、掌握 89C51 单片机与七段 LED 显示器并行接口的设计方法。

2、掌握并行 LED 扫描显示程序的编写方法。

二、实验原理

1、并行 LED 显示原理图：

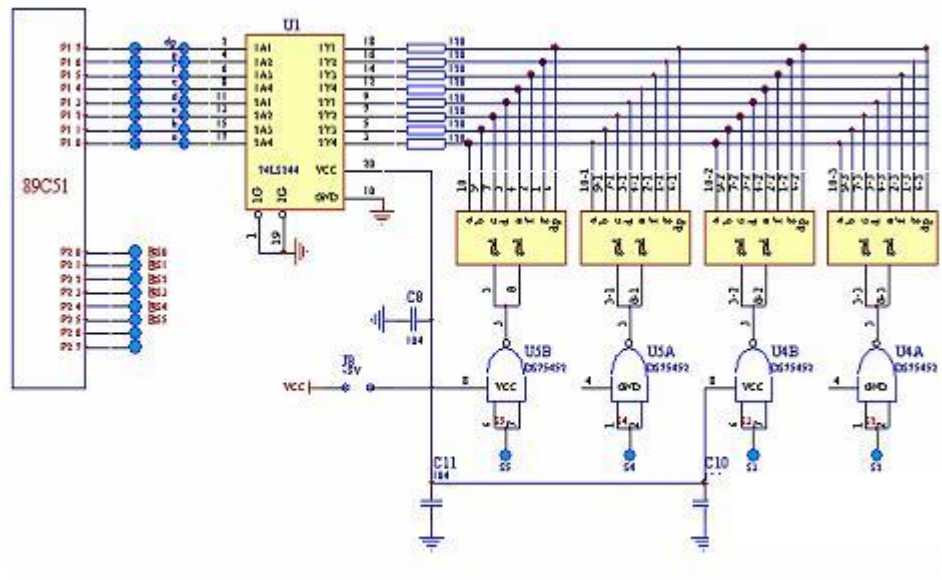
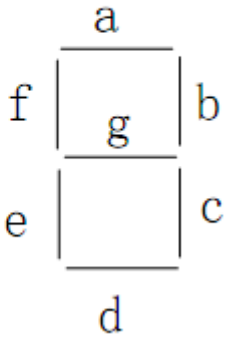


图　并行 LED 显示原理图

2、段 LED 段码表如下：

显示字型	dp	g	f	e	d	c	b	a	段码
0	0	0	1	1	1	1	1	1	3FH
1	0	0	0	0	0	1	1	0	06H
2	0	1	0	1	1	0	1	1	5BH
3	0	1	0	0	1	1	1	1	4FH
4	0	1	1	0	0	1	1	0	66H
5	0	1	1	0	1	1	0	1	6DH
6	0	1	1	1	1	1	0	1	7DH
7	0	0	0	0	0	1	1	1	07H
8	0	1	1	1	1	1	1	1	7FH
9	0	1	1	0	1	1	1	1	6FH



三、实验内容和步骤

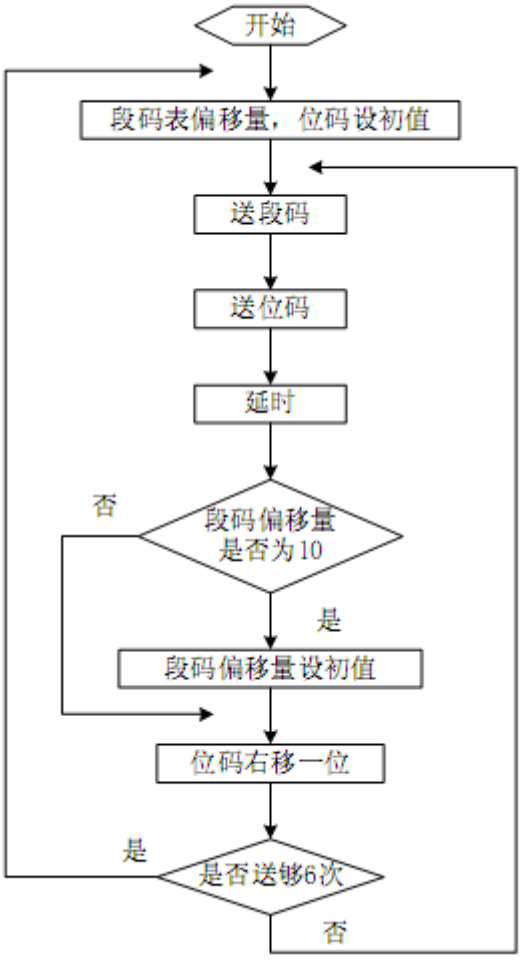
1、电路如上图，89C51 的 P1.0-P1.7 接数码管的段码 a-dp;P2.0-P2.3 接数码管的位码 S0-S3。

2、在数码管上显示“0123456789”,并顺序循环左移。

3、接线：P17~P10 /C51 单片机　接　dp~a /并行数码管

P25~P20 /C51 单片机　接　S5~S0

四、程序参考流程图



五、参考程序　LED.ASM

```
        ;并行数码管显示
        ORG     00H
        LJMP    main
        ORG     0100H
main:
        mov     20H,    #00H
        mov     21H,    #06H
        mov     22H,    #20H        ;0010 0000
        mov     23H,    #200
START:  MOV     R0,     20H         ;R0 为段码表的偏移量
        MOV     R1,     21H         ;R1 为计数器
        mov     R2,     22H         ;R2 为位码
        mov     R3,     23H
```

```
START1: MOV     DPTR,     #TABLE      ;段码表地址入 DPTR
        MOV     A,        R0          ;取偏移量
        MOVC    A,        @A+DPTR     ;取段码值

        mov     p1,A                  ;将段码送出
        mov     A,R2
        mov     p2,A                  ;将位码送出
        call    delay
        inc     r0                    ;下一段码
        cjne    R0,#0AH,S1            ;比较立即数和寄存器，不等则转
        mov     R0,#00H
S1:     mov     A,r2
        RR      A                     ;累加器循环右移
        mov     R2,A                  ;下一位码
        DJNZ    R1,START1             ;寄存器减一，不为零则转移
        call    delay
        MOV     R0,       20H
        MOV     R1,       21H
        mov     R2,       22H
        DJNZ    R3,START1              ;寄存器减一，不为零则转移
        inc     20H
        mov     A,20H
        CJNE    A,#0AH,S2
        mov     20H,#00H
S2:     SJMP    START
TABLE:  DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH;0~~9 的段码表
delay:  mov     R6,#10H
DEY0:   mov     R7,#10H
DEY1:   DJNZ    R7,DEY1
        DJNZ    R6,DEY0
        RET
        end
```

实验九 键盘实验

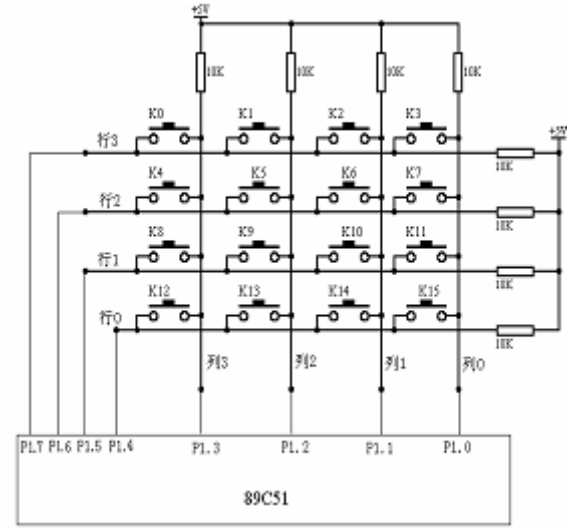
一、实验目的

掌握单片机矩阵式键盘的接口方法。

二、实验原理

单片机的键盘通常是若干个按键组成的开关矩阵。识别闭合键依靠软件实现。实验仪上设有一个 4×4 的键盘，共有 4 条行线、4 条列线，在每一条行线与列线的交叉点接有一个按键， 16 个按

键的编号为 K0~K15，结构如图所示。当某一个按键闭合时，与该键相连的行线与列线接通。识别闭合键的方法有逐行（列）扫描法及行反转法，其操作步骤如下。



1、逐行扫描法

① 将行线接微机的输出口，列线接微机的输入口。（P1 口高 4 位输出，低 4 位用于输入）

②通过输出口输出数据，逐一使 1 条行线为低电平(其余 3 行为高电平)，然后通过输入口读 4 根

列线的状态，若全为高电平，则此行无按键按下，若不全为高电平，说明这一行有键按下,且按键位于此行与电压为低电平的列线交叉点。例如，P1 口高 4 位输出 0111B(第 3 行为低电平)时，若读得列线的数据为 0111B,说明按键 K0 被按下；若读得列线的数据为 1011BH，说明按键 K1 被按下，若读得列线的数据为 1101BH，说明按键 K2 被按下。当一行没有有键按下时再用同样的办法接着扫描（检查）下一行……。

③当某一行有键按下时，通过此时行线输出及列线输入数据组合成 1 个 8 位二进制数，这个数称为键值，由键值可唯一的确定按键号码。

K0 按下时，必在行线输出 0111B，列线读得 0111B 时，其键值为 01110111B=77H；

K1 按下时，必在行线输出 0111B，列线读得 1011B 时，其键值为 01111011B=7BH；

K2 按下时，必在行线输出 0111B，列线读得 1101B 时，其键值为 01111101B =7DH；

K15 按下时，必在行线输出 1110B，列线读得 1110B 时，其键值为 11101110B =EEH；

键盘查询程序设计时，可将这 16 个按键对应的键值按照键号 0~15 连续存放（77H,7BH,7DH,7EH,B7H,BBH,BDH,BEH,D7H,DBH,DDH,DEH,E7H,EBH,EDH,EEH），构成一个数据表，通过查表即可确定键号。

2、行翻转法

① 将与行线相连的端口设置为输入，与列线相连的端口设置为输出（P2 口高 4 位输入，低 4 位用于输出）。向列线输出数据 0000B，使列线全部为低电平。然后读 4 根行线的状态，若全为高电平，说明无按键按下，返回①，若不全为高电平，说明有键按下，进入②。

② 翻转：将与行线相连的端口设置为输出，与列线相连的端口设置为输入（P2 口高 4 位输

出，低 4 位用于输入），然后把①中从行线得到的 4 位二进制的数向行线输出。

③ 从列线输入数据，得到一个 4 位二进制的数据，把①中得到 4 位二进制的数据作为高 4 位，与这个 4 位二进制的数据组合成的 8 位二进制数，即为键值（与逐行扫描法相同），由键值可唯一的确定按键号码。

三、实验内容

1、将键盘的 4 条行线、4 条列线与单片机 P1 口（也可任选）相连，编写键盘管理程序，要求能将按键号码在实验仪七段 LED 显示器上显示出来。可采用逐行扫描法或行翻转法。

行翻转法的接线方法：键盘的 4 条行线、4 条列线与单片机 P1 口相连，七段 LED 显示器与单片机的连接为 89C51 的 P2.0-P2.7 接数码管的段码 a-dp;码管的位码 S0 接 VCC。

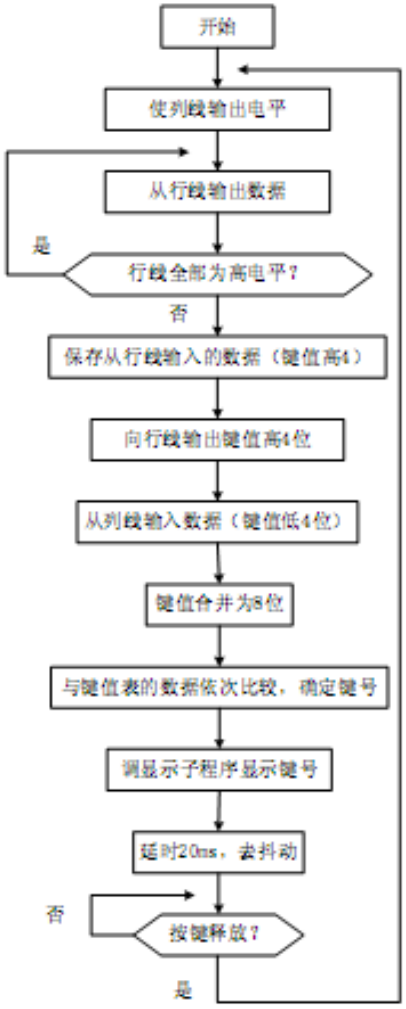
逐行扫描法的接线方法：键盘的 4 条行线、4 条列线与单片机的 P1 口相连，七段 LED 显示器与单片机的连接为 89C51 的 P2.0-P2.7 接数码管的段码 a-dp;码管的位码 S0 接 VCC。

2、接线：P27~P20 /C51 单片机　接　dp~a /并行数码管

+5V　接　S5~S0

P17~P10 /C51 单片机　接　行 3~列 0 /4X4 键盘

四、参考程序流程(行翻转法)



五、参考程序

1．JP1.ASM　　行反转法键盘扫描显示

```
        ORG 0H              ;行反转法键盘扫描显示?
KB1:    MOV P1,#0F0H        ;列线输出低电平
        MOV A,P1            ;输入 行线值
        CJNE A,#0F0H,KB2    ;若有键按下，  转 KB2?
        AJMP KB1            ;若无键按下，转 KB1
KB2:    MOV B,A             ;保存键码的高四位
        ORL A,#0FH          ;A 高四位不变，低四位置 1?
        MOV P1,A            ;键码的高四位通过行线输出?
        MOV A,P1            ;输入列线值
        ANL A,#0FH          ;屏蔽高四位
        ORL B,A             ;键码的高四位与低四位合并
        MOV DPTR,#TAB       ;DPTR 指向键码表首地址?
        MOV R3,#0           ;R3?作为键号计数器
 KB3:   MOV A,R3
        MOVC A,@A+DPTR      ;取键码
        CJNE A,B,NEXT       ;所取键码与当前按键的键码不等转移?
        CALL DELAY          ;延时 20ms
        CALL DISPLAY        ;显示键号
WAIT0:  MOV P1,#0F0H        ;以下 3 条指令为等待按键释放
        MOV A,P1
        CJNE A,#0F0H,WAIT0
        CALL DELAY
        AJMP KB1
 NEXT:  INC R3              ;键号计数器加 1
        AJMP KB3
 DELAY: MOV R0,#32H         ;延时 20ms
DELAY0: MOV R1,0C8H
DELAY1: DJNZ R1,DELAY1
          DJNZ R0,DELAY0
          RET
 ;显示子程序
DISPLAY:MOV DPTR,#TABLE     ;DPTR 指向段码表首址
        MOV A,R3
        MOVC A,@A+DPTR      ;查段码表
        MOV         p2,A        ;输出段码
        RET
  TAB:  DB 77H,7BH,7DH,7EH,0B7H,0BBH,0BDH,0BEH,0D7H
```

```
        DB 0DBH,0DDH,0DEH,0E7H,0EBH,0EDH,0EEH     ;0~F 键码
TABLE:  DB       3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH;0~~9 的段码表
        DB 77H,7CH,39H,5EH,79H,71H
        END
```

2、逐行扫描法 JIANPAN-2.ASM

```
        ORG 0H ;行扫描法键盘扫描显示
CLR P3.3
CLR P3.4
KB1:    MOV P1,#0FH ;行线输出低电平
        MOV A,P1 ;输入列线值
        CJNE A,#0FH,KB2 ;若有键按下 ,转 KB2 行扫
        AJMP KB1 ;若无键按下，转 KB1
KB2:    MOV B,#7FH
KB22:   MOV P1,B
        MOV A,P1
        ANL A,#0FH
        CJNE A,#0FH,KB3 ;此行有键按下，转 KB3
        MOV A,B
        RR A
        MOV B,A
        AJMP KB22
KB3:    ANL B,#0F0H
        ORL B,A
        MOV R3,#0H
        MOV DPTR,#TAB
KB4:    MOV A,R3
        MOVC A,@A+DPTR ;取键值
        CJNE A,B,NEXT ;所取键值与当前按键的键值不等转移
        CALL DELAY ;延时 20ms
        CALL DISPLAY ;显示键号
WAIT:   MOV P2,#0F0H ;以下 3 条指令为等待按键释放
        MOV A,P2
        CJNE A,#0F0H,WAIT
        CALL DELAY
        AJMP KB1
NEXT:   INC R3 ;键号计数器加 1
        AJMP KB4
DELAY:  MOV R0,#05H ;延时 20ms
```

```
DELAY0: MOV R1,0C8H
DELAY1: DJNZ R1,DELAY1
        DJNZ R0, DELAY0
        RET
DISPLAY: MOV DPTR,#TAB1 ;显示子程序，DPTR 指向段码表首址
         MOV A,R3
         MOVC A,@A+DPTR ;查段码表
         MOV P2, A ;将段码送出
         RET

TAB:    DB 77H,7BH,7DH,7EH,0B7H,0BBH,0BDH,0BEH,0D7H
        DB 0DBH,0DDH,0DEH,0E7H,0EBH,0EDH,0EEH     ;0~F 键码
TABLE:  DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH;0~~9 的段码表
        DB 77H,7CH,39H,5EH,79H,71H

        END
```

实验十　128X64 字符图形液晶显示

一、实验目的

1、了解字符 LCD 模块的使用方法

2、掌握 8051 单片机控制字符 LCD 模块显示程序的设计方法。

二、实验原理

字符图形 LCD 模块是一种专用显示字符、数字或符号的液晶显示模块。模块本身附有显示驱动控制电路，可以与单片机的 I/O 口线直接连接，使用方便。实验系统采用的 128X64 液晶为 ST7920 驱动控制器。

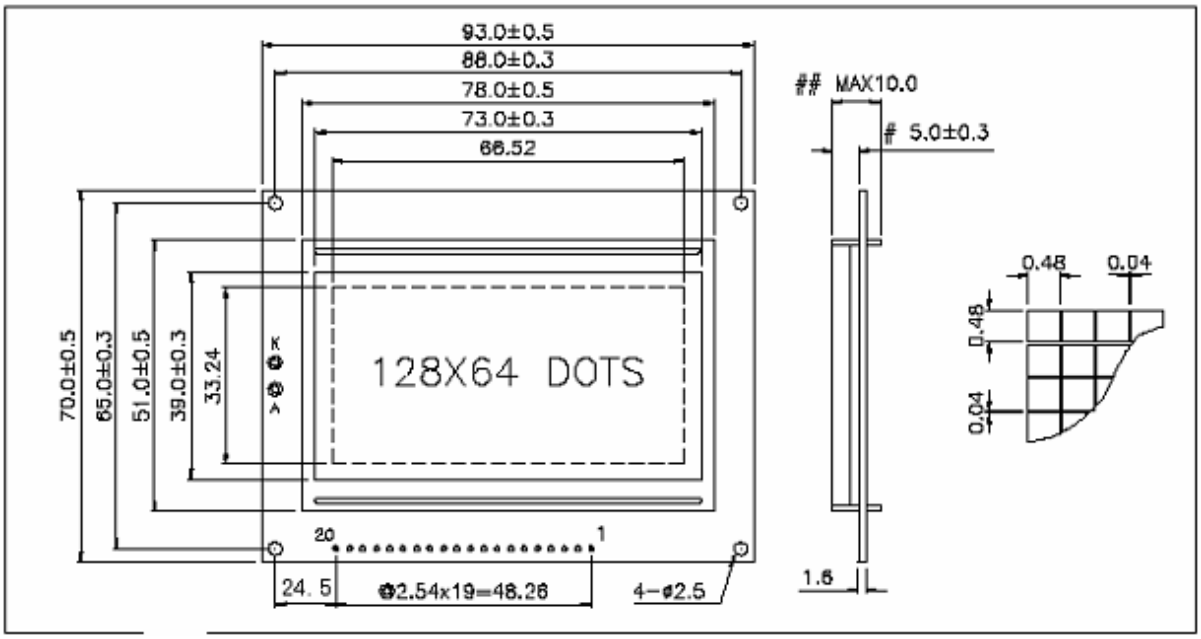
(一)、液晶显示模块概述

1. 液晶显示模块是 128×64 点阵的汉字图形型液晶显示模块，可显示汉字及图形，内置 8192 个中文汉字（16X16 点阵） 、128 个字符（8X16 点阵）及 64X256 点阵显示 RAM（GDRAM） 。可与 CPU 直接接口，提供两种界面来连接微处理机：8-位并行及串行两种连接方式。具有多种功能：光标显示、画面移位、睡眠模式等。

2. 外观尺寸：93×70×12.5mm

3. 视域尺寸：73×39mm

外形尺寸图



(二)、模块引脚说明

128X64 引脚说明

引脚	名称	方向	说明	引脚	名称	方向	说明
1	VSS	-	GND (0V)	11	DB4	I	数据 4
2	VDD	-	Supply Voltage For Logic (+5v)	12	DB5	I	数据 5
3	VO	-	Supply Voltage For LCD (悬空)	13	DB6	I	数据 6
4	RS (CS)	0	H: Data L: Instruction Code	14	DB7	I	数据 7
5	R/W (SID)	0	H: Read L: Write	15	PSB	0	H: Parallel Mode L: Serial Mode
6	E (SCLK)	0	Enable Signal	16	NC	-	空脚
7	DB0	I	数据 0	17	/RST	0	Reset Signal 低电平有效
8	DB1	I	数据 1	18	NC	-	空脚
9	DB2	I	数据 2	19	LEDA	-	背光源正极 (LED+5V)
10	DB3	I	数据 3	20	LEDK	-	背光源负极 (LED-0V)

(三)、液晶硬件接口

1、逻辑工作电压(VDD)：4.5～5.5V

2、电源地(GND)：0V

3、工作温度(Ta)：0～60℃(常温) / -20～75℃（宽温）

4、电气特性见附图 1　外部连接图（参考附图 2）三、液晶硬件接口

1、逻辑工作电压(VDD)：4.5～5.5V

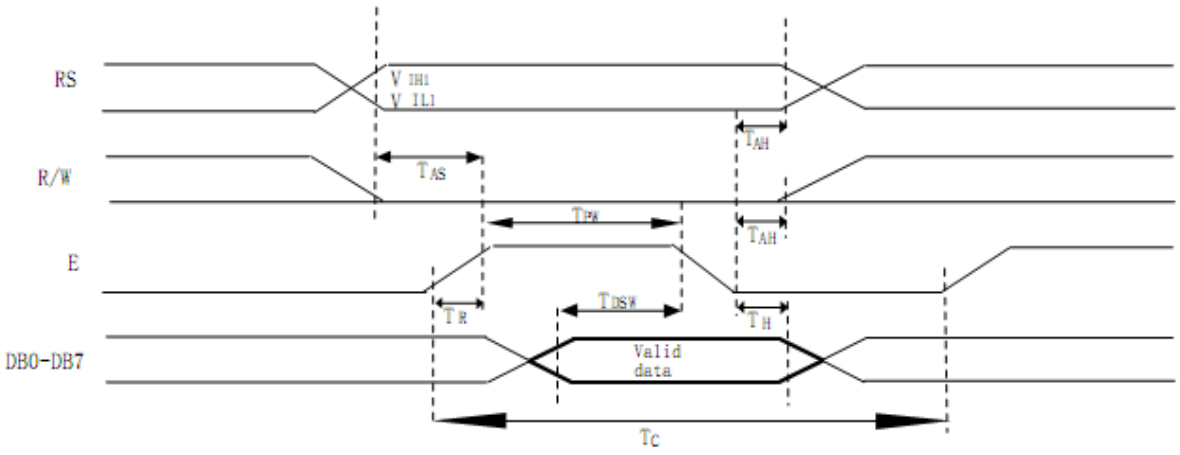
2、电源地(GND)：0V

3、工作温度(Ta)：0～60℃(常温) / -20～75℃（宽温）

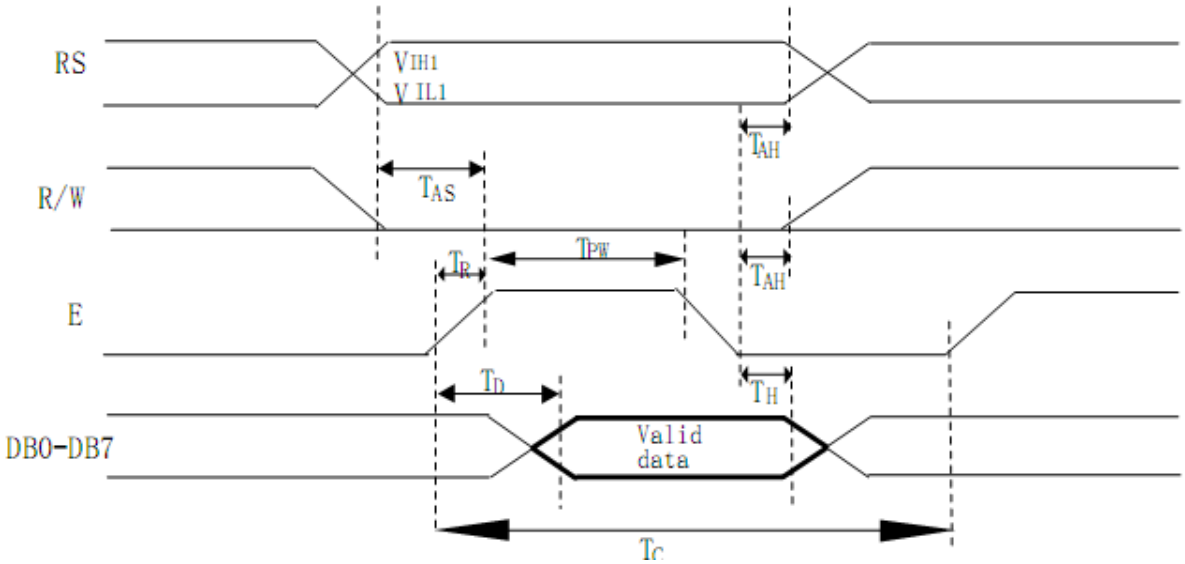
模块有并行和串行两种连接方法（时序如下）：

1、8 位并行连接时序图

MPU 写资料到模块



MPU 从模块读出资料



2、串行连接时序图

(四)、用户指令集

1、指令表 1： （RE=0：基本指令集）

指令	指令码										说明	执行时间（540KHZ）
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
清除显示	0	0	0	0	0	0	0	0	0	1	将 DDRAM 填满“20H”，并且设定 DDRAM 的地址计数器（AC）到“00H”	4.6ms
地址归位	0	0	0	0	0	0	0	0	1	X	设定 DDRAM 的地址计数器（AC）到“00H”，并且将游标移到开头原点位置；	4.6ms

											这个指令并不改变 DDRAM 的内容	
进入点设定	0	0	0	0	0	0	0	1	I/D	S	指定在资料的读取与写入时，设定游标移动方向及指定显示的移位	72us
显示状态开/关	0	0	0	0	0	0	1	D	C	B	D=1：整体显示 ON C=1：游标 ON B=1：游标位置 ON	72us
游标或显示移位控制	0	0	0	0	0	1	S/C	R/L	X	X	设定游标的移动与显示的移位控制位元；这个指令并不改变 DDRAM 的内容	72us
功能设定	0	0	0	0	1	DL	X	0 RE	X	X	DL=1 （必须设为 1） RE=1： 扩充指令集动作 RE=0： 基本指令集动作	72us
设定 CGRAM 地址	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	设定 CGRAM 地址到地址计数器（AC）	72us
设定 DDRAM 地址	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	设定 DDRAM 地址到地址计数器（AC）	72us
读取忙碌标志（BF）和地址	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	读取忙碌标志（BF）可以确认内部动作是否完成，同时可以读出地址计数器（AC）的值	0us
写资料到 RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	写入资料到内部的 RAM（DDRAM/CGRAM/IRAM/GDRAM）	72us
读出 RAM 的值	1	1	D7	D6	D5	D4	D3	D2	D1	D0	从内部 RAM 读取资料（DDRAM/CGRAM/IRAM/GDRAM）	72us

指令表—2：（RE=1：扩充指令集）

指令	指令码										说明	执行时间（540KHZ）
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
待命模式	0	0	0	0	0	0	0	0	0	1	将 DDRAM 填满“20H”，并且设定 DDRAM 的地址计数器（AC）到“00H”	72us
卷动地址或 IRAM 地址选择	0	0	0	0	0	0	0	0	1	SR	SR=1：允许输入垂直卷动地址 SR=0：允许输入 IRAM 地址	72us
反白选择	0	0	0	0	0	0	0	1	R1	R0	选择 4 行中的任一行作反白显示，并可决定反白与否	72us
睡眠模式	0	0	0	0	0	0	1	SL	X	X	SL=1：脱离睡眠模式 SL=0：进入睡眠模式	72us

扩充功能设定	0	0	0	0	1	1	X	1 RE	G	0	RE=1： 扩充指令集动作 RE=0： 基本指令集动作 G=1 ：绘图显示 ON G=0 ：绘图显示 OFF	72us
设定 IRAM 地址或卷动地址	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	SR=1：AC5—AC0 为垂直卷动地址 SR=0：AC3—AC0 为 ICON IRAM 地址	72us
设定绘图 RAM 地址	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	设定 CGRAM 地址到地址计数器（AC）	72us

备注：

1、 当模块在接受指令前，微处理顺必须先确认模块内部处于非忙碌状态，即读取 BF 标志时 BF 需为 0，方可接受新的指令；如果在送出一个指令前并不检查 BF 标志，那么在前一个指令和这个指令中间必须延迟一段较长的时间，即是等待前一个指令确实执行完成，指令执行的时间请参考指令表中的个别指令说明。

2“RE”为基本指令集与扩充指令集的选择控制位元，当变更“RE”位元后，往后的指令集将维持在最后的状态，除非再次变更“RE”位元，否则使用相同指令集时，不需每次重设“RE”位元。

具体指令介绍：

1、清除显示

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	L	L	L	L	L	L	L	H

功能：清除显示屏幕，把 DDRAM 位址计数器调整为“00H”

2、位址归位

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	L	L	L	L	L	L	H	X

功能：把 DDRAM 位址计数器调整为“00H” ，游标回原点，该功能不影响显示 DDRAM

3、位址归位

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	L	L	L	L	L	H	I/D	S

功能：把 DDRAM 位址计数器调整为“00H” ，游标回原点，该功能不影响显示 DDRAM 功能：执行该命令后，所设置的行将显示在屏幕的第一行。显示起始行是由 Z 地址计数器控制的，该命令自动将 A0-A5 位地址送入 Z 地址计数器，起始地址可以是 0-63 范围内任意一行。Z 地址计数器具有循环计数功能，用于显示行扫描同步，当扫描完一行后自动加一。

4、显示状态 开/关

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	L	L	L	L	H	D	C	B

功能： D=1；整体显示 ON　　C=1；游标 ON　　B=1；游标位置 ON

5、游标或显示移位控制

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	L	L	L	H	S/C	R/L	X	X

功能：设定游标的移动与显示的移位控制位：这个指令并不改变 DDRAM 的内容

6、功能设定

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	L	L	H	DL	X	0 RE	X	X

功能：DL=1（必须设为 1）　RE=1；扩充指令集动作　　RE=0：基本指令集动作

7、设定 CGRAM 位址

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	L	H	AC5	AC4	AC3	AC2	AC1	AC0

功能：设定 CGRAM 位址到位址计数器（AC）

8、设定 DDRAM 位址

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	L	H	AC6	AC5	AC4	AC3	AC2	AC1	AC0

功能：设定 DDRAM 位址到位址计数器（AC）

9、读取忙碌状态（BF）和位址

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	L	H	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

功能：读取忙碌状态（BF）可以确认内部动作是否完成，同时可以读出位址计数器（AC）的值

10、写资料到 RAM

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	H	L	D7	D6	D5	D4	D3	D2	D1	D0

功能：写入资料到内部的 RAM（DDRAM/CGRAM/TRAM/GDRAM）

11、读出 RAM 的值

CODE：	RW	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	H	H	D7	D6	D5	D4	D3	D2	D1	D0

功能：从内部 RAM 读取资料（DDRAM/CGRAM/TRAM/GDRAM）

12、　待命模式（12H）

CODE： RW RS DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L L L L H

功能：进入待命模式，执行其他命令都可终止待命模式

13、卷动位址或 IRAM 位址选择（13H）

CODE： RW RS DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L L L H SR

功能：SR=1；允许输入卷动位址 SR=0；允许输入 IRAM 位址

14、反白选择（14H）

CODE： RW RS DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L L H R1 R0

功能：选择 4 行中的任一行作反白显示，并可决定反白的与否

15、睡眠模式（015H）

CODE： RW RS DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L H SL X X

功能：SL=1；脱离睡眠模式 SL=0；进入睡眠模式

16、扩充功能设定（016H）

CODE： RW RS DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L H H X 1 RE G L

功能：RE=1；扩充指令集动作 RE=0；基本指令集动作 G=1；绘图显示 ON G=0；绘图显示 OFF

17、设定 IRAM 位址或卷动位址（017H）

CODE： RW RS DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L H AC5 AC4 AC3 AC2 AC1 AC0

功能：SR=1；AC5~AC0 为垂直卷动位址 SR=0；AC3~AC0 写 ICONRAM 位址

18、设定绘图 RAM 位址（018H）

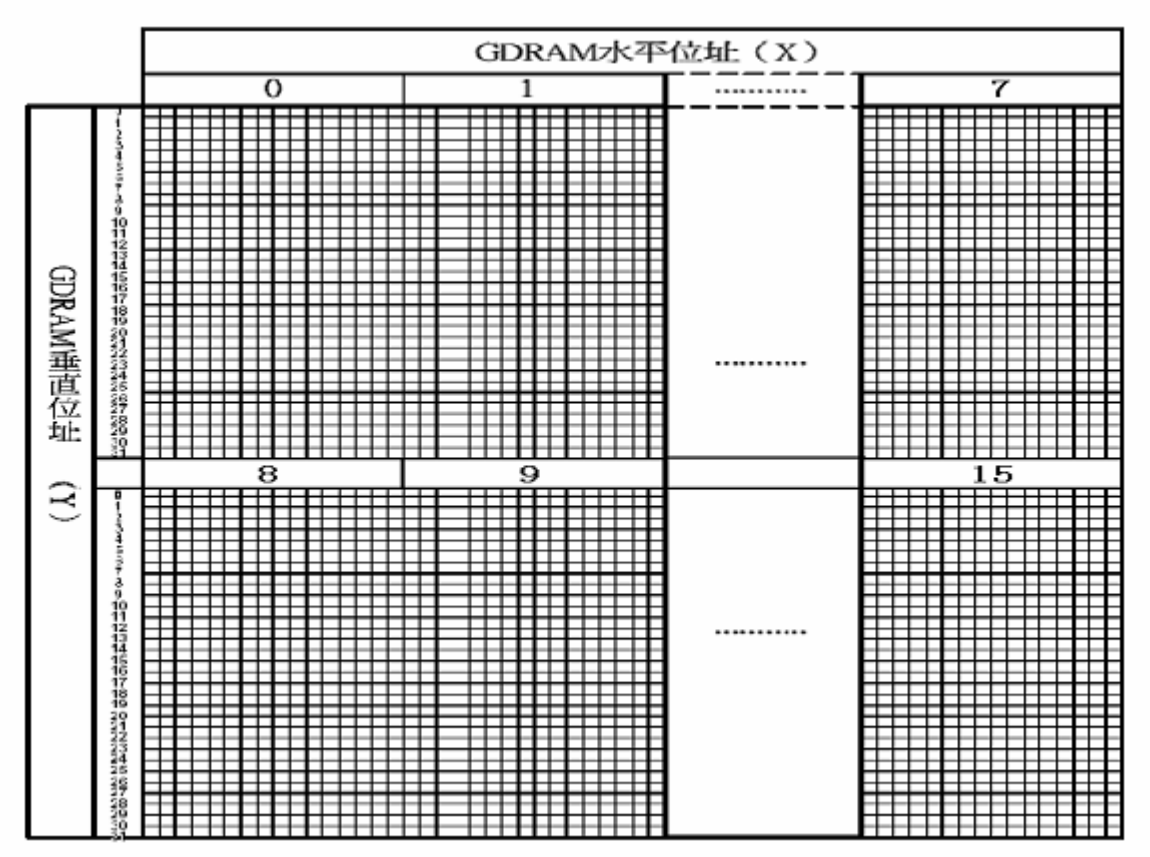
CODE： RW RS DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L H AC6 AC5 AC4 AC3 AC2 AC1 AC0

功能：设定 GDRAM 位址到位址计数器（AC）

(五)、显示坐标关系

1、图形显示坐标

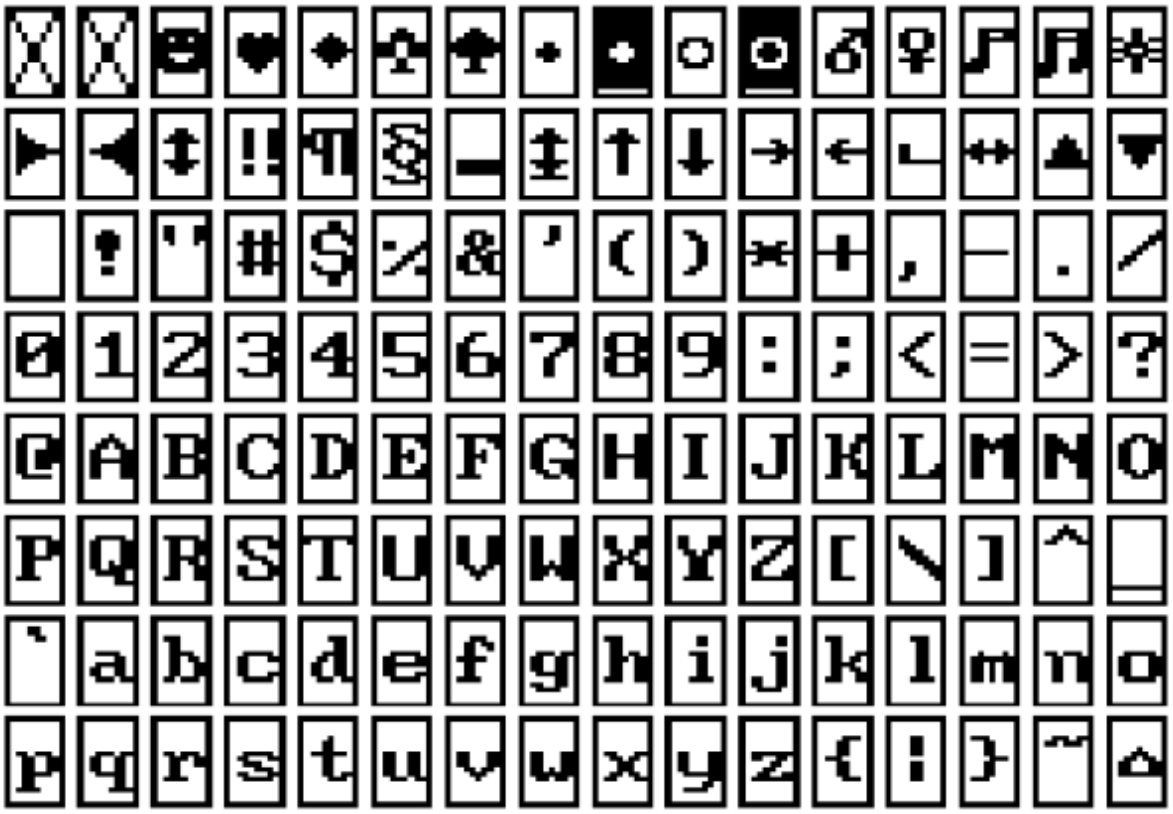


2、汉字显示坐标

	X 坐标							
Line1	80H	81H	82H	83H	84H	85H	86H	87H
Line2	90H	91H	92H	93H	94H	95H	96H	97H
Line3	88H	89H	8AH	8BH	8CH	8DH	8EH	8FH
Line4	98H	99H	9AH	9BH	9CH	9DH	9EH	9FH

3、字符表

代码（02H---7FH）



(六)、显示步骤

1、显示资料 RAM（DDRAM）

显示资料 RAM 提供 64×2 个位元组的空间，最多可以控制 4 行 16 字（64 个字）的中文字型显示，

当写入显示资料 RAM 时，可以分别显示 CGROM、HCGROM 与 CGRAM 的字型；ST7920A 可以显示三种字型 ，分别是半宽的 HCGROM 字型、CGRAM 字型及中文 CGROM 字型 ，三种字型的选择，由

在 DDRAM 中写入的编码选择，在 0000H—0006H 的编码中将自动的结合下一个位元组，组成两个位

元组的编码达成中文字型 的编码（A140—D75F） ，各种字型详细编码如下：

1、显示半宽字型 ：将 8 位元资料写入 DDRAM 中，范围为 02H—7FH 的编码。

2、显示 CGRAM 字型：将 16 位元资料写入 DDRAM 中，总共有 0000H，0002H，0004H，0006H 四种编码。

3、显示中文字形：将 16 位元资料写入 DDRAMK ，范围为 A1A1H—F7FEH 的编码。

绘图 RAM（GDRAM）

绘图显示 RAM 提供 64×32 个位元组的记忆空间，最多可以控制 256×64 点的二维也纳绘图缓冲空间，在更改绘图 RAM 时，先连续写入水平与垂直的坐标值，再写入两个 8 位元的资料到绘图 RAM，而地址计数器（AC）会自动加一；在写入绘图 RAM 的期间，绘图显示必须关闭，整个写入绘图 RAM 的步骤如下：

1、关闭绘图显示功能。

2、先将水平的位元组坐标（X）写入绘图 RAM 地址；

3、再将垂直的坐标（Y）写入绘图 RAM 地址；

4、将 D15——D8 写入到 RAM 中；

5、将 D7——D0 写入到 RAM 中；

6、打开绘图显示功能。

绘图显示的记忆体对应分布请参考表

2、游标/闪烁控制

ST7920A 提供硬体游标及闪烁控制电路，由地址计数器（address counter）的值来指定 DDRAM 中的游标或闪烁位置。

三、实验内容

1、将 LCD 与单片机连接，编程在其上面显示字符和图形

2、接线：

P2.1 /C51 单片机	接	D/I /LCD
P2.2 /C51 单处机	接	RW /LCD
P2.4 /C51 单片机	接	E /LCD
P07~P00 /C51 单片机	接	D7~D0 /LCD
P2.5 /C51 单片机	接	PSB
P3.5 /C51 单片机	接	RST

四、参考程序

LCD.ASM

```
;******************OCM4x8C 并口******************
RS          EQU         P2.1
RW          EQU         P2.2
E           EQU         P2.4
PSB         EQU         P2.5
RST         EQU         P3.5

LCD_X       EQU         30H
LCD_Y       EQU         31H
LCD_X1      EQU         32H
LCD_Y1      EQU         33H
COUNT       EQU         34H
COUNT1      EQU         35H
COUNT2      EQU         36H
COUNT3      EQU         37H
LCD_DATA    EQU         38H
LCD_DATA1   EQU         39H
LCD_DATA2   EQU         3AH
```

```
;*********************************************
                ORG     0000H
                LJMP    MAIN
                ORG     0100H
MAIN:           MOV     SP,#5FH
                LCALL   DELAY2
                SETB    PSB
                SETB    RST
;********************初始化**********************
LGS0:           MOV     A,#34H                  ;34H--扩充指令操作
                LCALL   SEND_I
                MOV     A,#30H                  ;30H--基本指令操作
                LCALL   SEND_I
                MOV     A,#01H                  ;清除显示
                LCALL   SEND_I
                MOV     A,#06H                  ;指定在资料写入或读取
时，光标的移动方向
                LCALL   SEND_I
                MOV     A,#0CH                  ;开显示,关光标,不闪烁
                LCALL   SEND_I
;*********************************************
LGS1:           MOV     DPTR,#TAB1          ;显示汉字和字符
                MOV     COUNT,#28
                MOV     A,#80H
                LCALL   SEND_I
LGS11:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   SEND_D
                INC     DPTR
                DJNZ    COUNT,LGS11
                LCALL   DELAY3

;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
LGS5:           MOV     DPTR,#TAB5          ;显示图形
                LCALL   PHO_DISP
                LCALL   DELAY3
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                MOV     A,#34H
```

```
                    LCALL     SEND_I
                    MOV       A,#30H
                    LCALL     SEND_I
                    LJMP      LGS1
;*********************************************
PHO_DISP:  MOV      LCD_X,#80H          ;全屏显示图形子程序
                    MOV       LCD_Y1,#80H
                    MOV       COUNT3,#02H
PHO_DISP1:MOV       LCD_X1,LCD_X
                    MOV       COUNT2,#20H
PHO_DISP2:MOV       COUNT1,#08H
PHO_DISP3:LCALL     WR_ZB
                    CLR       A
                    MOVC      A,@A+DPTR
                    LCALL     SEND_D
                    INC       DPTR
                    CLR       A
                    MOVC      A,@A+DPTR
                    LCALL     SEND_D
                    INC       DPTR
                    INC       LCD_X1
                    DJNZ      COUNT1,PHO_DISP3
                    MOV       LCD_X1,LCD_X
                    INC       LCD_Y1
                    DJNZ      COUNT2,PHO_DISP2
                    MOV       LCD_X,#88H
                    MOV       LCD_Y1,#80H
                    DJNZ      COUNT3,PHO_DISP1

                    MOV       A,#36H
                    LCALL     SEND_I
;                   LCALL     DELAY1
                    MOV       A,#30H
                    LCALL     SEND_I
                    RET


;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
WR_ZB:          MOV     A,#34H
                LCALL   SEND_I
;               LCALL   DELAY1
                MOV     A,LCD_Y1
                LCALL   SEND_I
                MOV     A,LCD_X1
                LCALL   SEND_I
                MOV     A,#30H
                LCALL   SEND_I
                RET
;*********************************************
SEND_D:         LCALL   CHK_BUSY        ;写数据子程序
                SETB    RS
                CLR     RW
                MOV     P0,A
                SETB    E
                NOP
                NOP
                CLR     E
                NOP
                MOV     P0,#0FFH
                RET

SEND_I:         LCALL   CHK_BUSY        ;写指令子程序
                CLR     RS
                CLR     RW
                MOV     P0,A
                SETB    E
                NOP
                NOP
                CLR     E
                NOP
                MOV     P0,#0FFH
                RET

CHK_BUSY: CLR   RS                      ;测忙碌子程序
                SETB    RW
                SETB    E
```

```
                    JB      P0.7,$
                    CLR     E
                    RET
;*********************************************
DELAY3:             MOV     R5,#0AH
DEL31:              MOV     R6,#0FFH
DEL32:              MOV     R7,#0FFH
DEL33:              DJNZ    R7,DEL33
                    DJNZ    R6,DEL32
                    DJNZ    R5,DEL31
                    RET

DELAY2:             MOV     R6,#0CH
DEL21:              MOV     R7,#18H
DEL22:              DJNZ    R7,DEL22
                    DJNZ    R6,DEL21
                    RET

DELAY1:             MOV     R6,#06H
DEL11:              MOV     R7,#08H
DEL12:              DJNZ    R7,DEL12
                    DJNZ    R6,DEL11
                    RET
;*********************************************
TAB1:       DB 'QHFC 实验系统'
            DB 'TEL:010-81765272'


TAB5:
DB   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
DB   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
DB   00,00,00,00,01H,80H,00,60H,04H,00,38H,00,00,00,00,00
DB   00,00,00,00,03H,0E0H,30H,78H,0FH,00,78H,00,00,00,00,00
DB   00,00,00,00,06H,0A0H,38H,0F8H,0FH,00,78H,0FFH,0E0H,00,00,00
DB   00,00,00,00,1DH,20H,3CH,18H,0EH,00,60H,80H,10H,00,00,00
DB   00,00,00,00,19H,30H,07H,00,08H,00,40H,80H,18H,00,00,00
DB   00,00,00,00,30H,10H,07H,0FH,0F8H,00,00,40H,0CH,00,00,00
DB   00,00,00,00,60H,0F0H,00,7FH,8CH,00,00,40H,03H,00,00,00
```

```
DB   00,00,00,00,0C1H,80,01H,0C0H,0F7H,0F0H,00,40H,01H,80H,00,00
DB   00,00,00,01H,86H,03H,0FFH,00,10H,1CH,00,0C0H,00,0C0H,00,00
DB   00,00,00,01H,04H,3FH,0FFH,0C0H,10H,01H,0C0H,0C0H,40H,70H,0FH,00
DB   00,00,00,01H,0FH,0FFH,0FFH,0F3H,0DFH,0F8H,70H,80H,60H,1EH,7FH,00
DB   00,00,3CH,01H,8FH,0FFH,0FFH,0FFH,0C0H,02H,18H,80H,40H,02H,00,00
DB   00,00,1FH,00,0FFH,0FFH,0FFH,0FFH,38H,03H,86H,80H,0C0H,01H,00,00
DB   00,00,03H,0E0H,0FFH,0FFH,0FFH,0FFH,0F8H,00,83H,00,0C0H,01H,80H,00
DB   00,00,00,00,3FH,0FFH,0FFH,0FFH,0E0H,00,83H,03H,80H,00,80H,00
DB   00,0FH,0F0H,00H,7FH,0FFH,0FFH,0FFH,0F8H,00,83H,07H,02H,00,80,00
DB   00,03H,0E0H,00,7FH,0F0H,07H,0FFH,0FCH,01H,83H,0FEH,06H,03H,80H,00
DB   00,00,00,00,7FH,0F8H,01H,0FFH,0FEH,01H,02H,06H,0EH,0FH,80H,00
DB   00,00,00,00,7FH,0FCH,00H,7FH,0FFH,0FFH,0EH,03H,1CH,0EH,00,00
DB   00,00,00,00,7FH,0FFH,00,7FH,0FFH,0F8H,00,01H,0E8H,0C6H,00,00
DB   00,03H,0FFH,0C0H,7FH,0FFH,0C0H,3FH,0FFH,0FCH,00,00H,63H,83H,00,00
DB   00,01H,0FEH,00,3FH,0FFH,0F8H,3FH,0FFH,0FFH,80H,00,3EH,03H,00,00
DB   00,00,00,00,1FH,0FFH,0FEH,1FH,0FFH,0FFH,0C0H,00,1CH,01H,80H,00
DB   00,00,00,00,1FH,0FFH,0FFH,0FFH,0FFH,0FFH,0F0H,00,0CH,00,80H,00
DB   00,00,00,00,0FH,0FFH,0FFH,0FFH,0FFH,0FFH,0F0H,00,07H,0F0H,80H,00
DB   00,00,00,00,0FH,0FFH,0FFH,0FFH,0FFH,0FFH,0FCH,00,03H,80H,80H,0FEH
DB   00,00,00,00,18H,0FFH,0FFH,0FFH,0FFH,0FFH,0FCH,00,1FH,0C0H,83H,82H
DB   00,00,00,00,33H,0BFH,0FFH,0FFH,0FFH,0FFH,0C4H,00,41H,0FFH,0FEH,06H
DB   00,00,00,00,0E6H,0FH,0FFH,0FFH,0FFH,0FFH,0FH,00,70H,00,00,3CH
DB   00,00,00,01H,8CH,0DH,0FFH,0FFH,0FFH,0F8H,09H,80H,1FH,0E0H,00,3CH
DB   00,00,00,03H,1FH,8EH,00,0FFH,0FFH,80H,19H,80H,00,3FH,0F0H,04H
DB   00,00,00,0CH,77H,0F3H,0E0H,07H,0FFH,00,71H,80H,00,33H,18H,04H
DB   00,00,00,39H,0C7H,0E0H,38H,00,00,00,0C1H,80H,00,22H,04H,0CH
DB   00,00,00,67H,07H,0C0H,0FH,00,00,07H,81H,80H,00,46H,03H,0F8H
DB   00,00,03H,0FCH,07H,0FCH,00,0E0H,00,0CH,01H,80H,00,1CH,00,00
DB   00,0FCH,07H,0E0H,03H,0FFH,00,1FH,0FFH,0F0H,01H,80H,00,30H,00,00
DB   3BH,0FFH,0FFH,80H,00,0FFH,0C0H,00,00,00,01H,80H,00,60H,00,00
DB   7FH,0FFH,0F8H,0F8H,00,7FH,0F8H,00,00,00,03H,0C0H,00,60H,00,00
DB   7FH,0FFH,0FEH,0EH,00,3FH,0FEH,00,00,00,0FH,0C0H,00,0C0H,00,00
DB   3FH,0FFH,0F9H,0FCH,0FFH,0CFH,0FFH,80,00,00,1FH,0C0H,00,80H,00,00
DB   07H,0FFH,0C0H,7BH,0E0H,7FH,0FFH,0E0H,00,00,3FH,00,03H,87H,0FFH,0E0H
DB   00,00,01H,0C0H,3FH,83H,0FFH,0FEH,00H,03H,0FEH,00,0EH,04H,7FH,0E0H
DB   00,00,07H,00,00,7EH,0FH,0FFH,0FFH,0FFH,0FCH,00,38H,00,7FH,00
DB   00,00,1FH,00,00,7FH,0FEH,7FH,0FFH,0FFH,0F8H,00,60H,00,00,00
DB   00,00,3FH,0E0H,07H,0FFH,83H,0FFH,0DFH,0FFH,0C0H,01H,80H,00,00,00
```

```
DB   00,00,3FH,0FCH,7CH,7FH,0C0H,00,3FH,0FDH,00,03H,00,00,00,00
DB   00,00,7FH,0FFH,80H,7FH,0E0H,00,00,06H,03H,0FEH,00,00,00,00
DB   00,00,0FFH,0FFH,80H,7FH,0E0H,00,03H,0E3H,07H,0E0H,00,00,00,00
DB   00,00,0FFH,0FEH,00,3FH,0E0H,00,3EH,3FH,8CH,00,00,00,00,00
DB   00,00,0FFH,0FEH,00,08H,60H,00,0E0H,0BH,0FFH,80H,00,00,00,00
DB   00,00,7FH,0FFH,00,38H,60H,01H,80H,7FH,0F8H,0FFH,00,00,00,00
DB   00,00,7FH,0FFH,0FFH,0C0H,0E0H,00,0F0H,0C0H,1EH,00,0C0H,00,00,00
DB   00,00,30H,00,0FH,0E0H,0C0H,00,67H,00,03H,0C6H,60H,00,00,00
DB   00,00,18H,00,0FH,0FFH,80H,00,3CH,00,00,23H,30H,00,00,00
DB   00,00,0CH,00,0FH,0FFH,00,00,00,00,00,19H,10H,00,00,00
DB   00,00,03H,0C0H,0FH,0F8H,00,00,00,00,00,08H,10H,00,00,00
DB   00,00,00,7FH,0FFH,0E0H,00,00,00,00,00,07H,0F0H,00,00,00
DB   00,00,00,0FH,0FCH,00,00,00,00,00,00,00,00,00,00,00
DB   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
DB   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
DB   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
DB   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00

                    END
```

实验十一　　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、掌握单片机控制双色点阵 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 点阵的红绿两色 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。

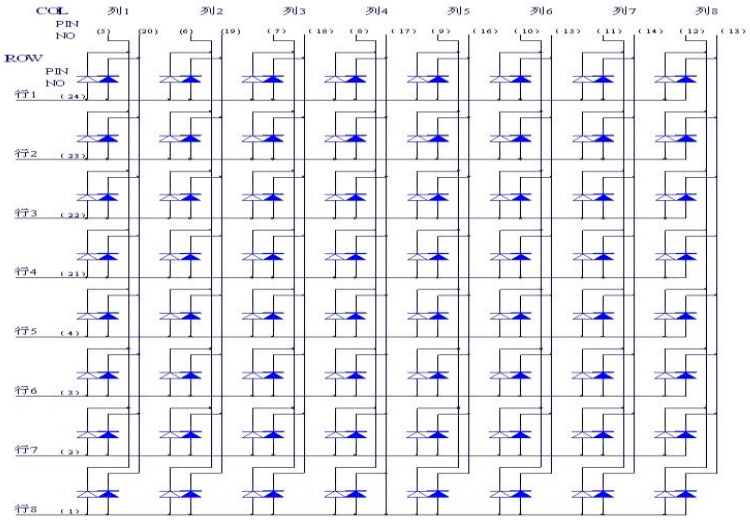


图 双色点阵

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。

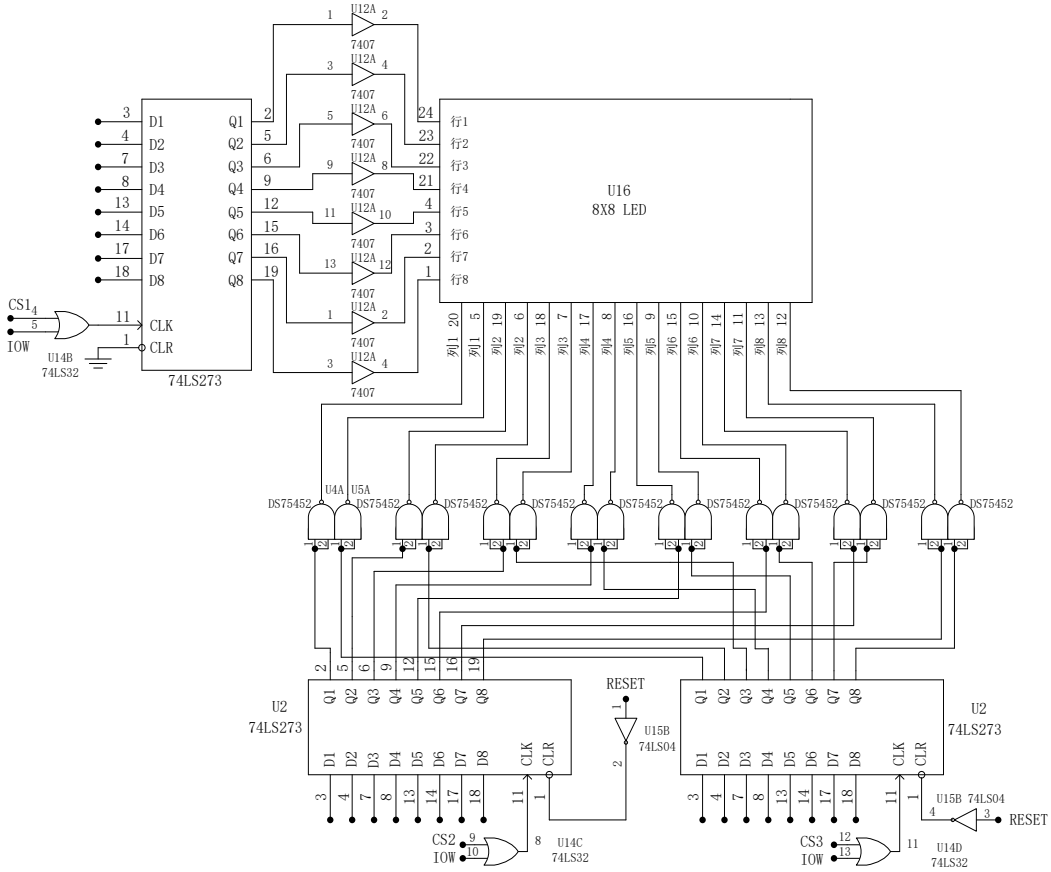
⑦ 行代码输出 44H； 红色列代码输 40H；第七列 2 个红色 LED 发光。

⑧ 行代码输出 24H； 红色列代码输 80H；第八列 2 个红色 LED 发光。

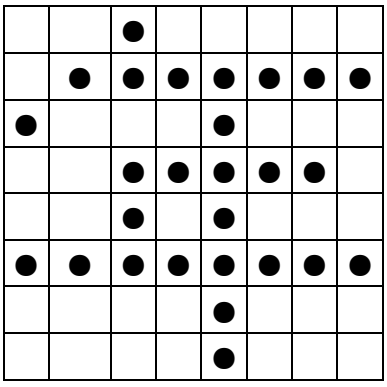
在步骤①～⑧之间可插入几 ms 的延时，重复进行①～⑧即可在 LED 上稳定的显示出红色

“年”字。若想显示绿色“年”，只需把红色列码改为绿色列码即可。

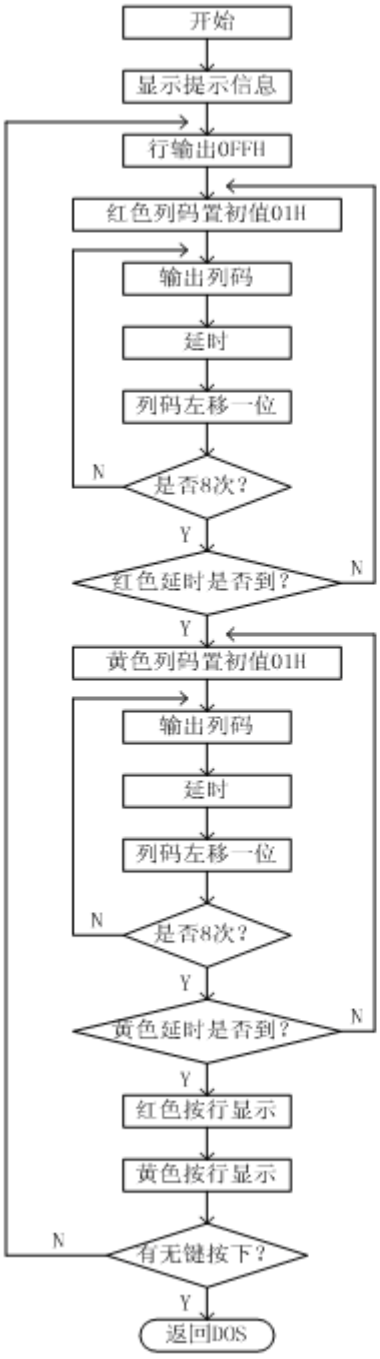
三、实验内容:



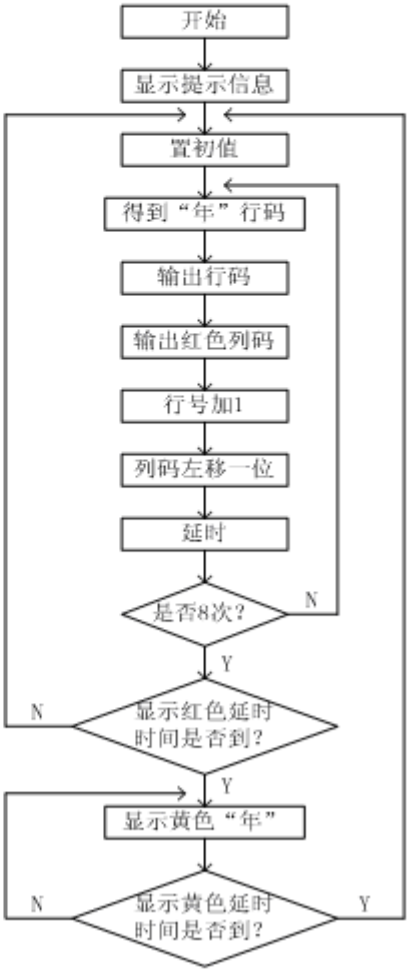
1. 实验仪上的点阵LED及驱动电路如图所示，行代码输出的数据通过行驱动器7407加至点阵的8条行线上，红和绿列代码的输出数据通过驱动器ULN2803反相后分别加至红和黄的列线上。
2. 接线方法：单片机P2口接点阵的“行D7~D0”，单片机的P1口接点阵的“红D7~D0”或“绿D7~D0”，接“红D7~D0”时，点阵显示数据为红色，接“绿D7~D0”时，点阵显示数据为绿色。
3. 编程重复使 LED 点阵红色逐列点亮，再绿色逐列点亮，再红色逐行点亮，绿色逐行点亮。
4. 编程在 LED 上重复显示红色“年”和绿色“年”
5. 点阵显示模块工作于非总线模式，行代码、红代码、绿代码分别通过行驱动器 7407 和列反相驱动器 ULN2803 加到行线和列线上。如图



四、流程图



1、逐行、逐列显示参考流程图 1



2、显示“年”参考流程 2

五、参考程序

参考程序 1　LEDD.ASM　逐行逐列显示

```
        ORG 0H
START:  MOV P2,#0FFH   ;阳极全部加高电平
        MOV A,#01H     ;A 初值为 80H,为最左一列(第 7 列)阴极输出低电平准备?
LOOP:   MOV P1,A       ;使一列阴极为低电平
        CALL DELAY     ;延时
        RLC A          ;A 右移一位，为下一列阴极输出低电平作准备
```

```
        JNC LOOP        ;八列未完转移?
        CLR  C
        MOV P1,#0FFH    ;阴极全部加低电平
        mov a,#80h      ;为最上面一行输出高电平准备
LOOP1:  MOV P2,A        ;使一行阳极极为高电平
        CALL DELAY      ; 延时
        RRC A           ; A 右移一位，为下一行阳极输出高电平作准备
        JNC LOOP1       ; 八行未完转移?
        CLR C
        AJMP START
DELAY:  MOV R1,#0C8H    ;延时子程序
DEY0:   MOV R0,#0H
DEY1:   DJNZ R0,DEY1
        DJNZ R1,DEY0
        RET
        END
```

参考程序 2　LEDD-HZ.ASM　　显示汉字“年”

```
        ORG 0H          ;显示'年?
 START:MOV DPTR,#DATA1  ;指向点阵代码首址
        MOV R2,#01H     ;80H 使第 7 列（最左一列）阴极为低电平??
        CLR C
 DISP: MOV A,#0
        MOVC A,@A+DPTR  ;取一列代码
        MOV P2,A        ;加至阳极
        MOV  P1,R2      ;使一列阴极为低电平
        CALL DELAY      ;延时
        INC DPTR        ;指向下一列代码?
        MOV A,R2        ;R2 右移一位，为下一列阴极输出低电平作准备
        RLC A           ;
        MOV R2,A        ;
        JNC DISP        ;8 列未完,转
        AJMP START
DELAY: MOV R1,#0        ;延时子程序
DELY0: DJNZ R1,DELY0
        RET
DATA1: ;DB 24H,44H,0DCH,54H,7FH,54H,54H,44H;年的点阵代码
      DB 44H,54H,54H,7FH,54H,0DCH,44H,24H
        END
```

实验十二　　继电器控制实验

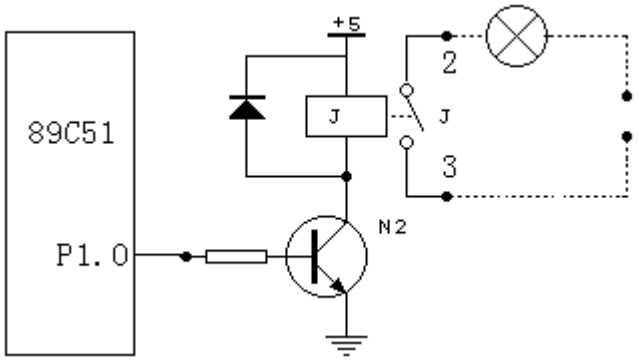
一、实验目的

掌握单片机控制直流继电器的方法。

二、实验原理及内容

1、实验电路如图，继电器常开触点 2、3 之间串联一个红色发光二极管， 当 P1.0 输出高电平时，三极管导通，继电器得电，常开触点闭合，灯泡发光。当 P1.0 输出低电平时，三极管截止，继电器失电，灯泡不发光。编程通过定时器 T0 定时，使继电器周而复始的闭合 5 秒钟（灯亮），断开 5 秒钟（灯灭）。

2、接线：　P10 /C51 单片机　　接　　继电器



三、参考程序　　JDQ.ASM

```
        ORG     00H
        SETB    P1.0            ;使继电器得电，常开触点闭合
START:  CALL    DELAY5S         ;延时 5 秒钟
        CPL     P1.0            ;对 P1.0 口取反
        SJMP    START
DELAY5S:MOV     R0,     #50     ;延时 5 秒钟子程序,送循环初值
DELAY1: MOV     TMOD,   #01H    ;选用 T0,方式 0
        MOV     DPTR,   #15536  ;计数初值
        MOV     TL0,    DPL     ;送计数初值，100ms 溢出一次。
        MOV     TH0,    DPH
        SETB    TR0             ;启动定时
        JNB     TF0,    $       ;等待 T0 溢出
        CLR     TF0             ;清除溢出标志
        DJNZ    R0,     DELAY1  ;T0 溢出不到 50 次，转移
        RET
```

实验十三　　直流电机控制实验

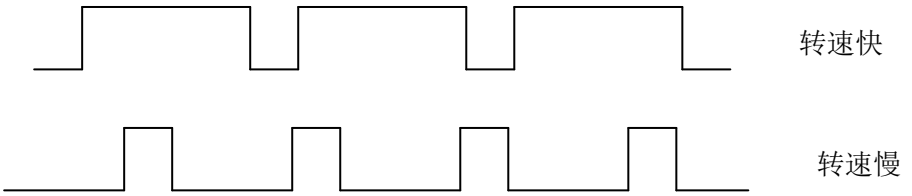
一、实验目的

了解单片机控制直流电机的基本原理。

二、实验原理

1、直流电机的转动原理

直流电机的转动方向是由加在电机上的电压正负极性决定的。电压为正，则顺时针方向转，电压为负，则逆时针方向转，转速的改变是通过改变所加脉冲的占空比越大，转速越快。示意图如下：本实验所用的直流电机，只能单方向转动。

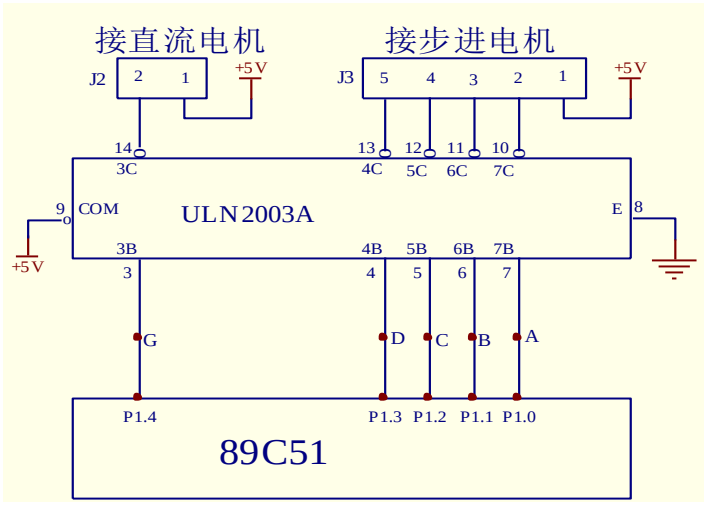


三、实验电路与内容

1、实验电路图所示，ULN2003A 为达林顿晶体管阵列，内含 7 个达林顿晶体管，xB（x=1~7）为达林顿晶体管的输入（基极）端，xC 为达林顿晶体管的输出（集电极）端。实验内容：

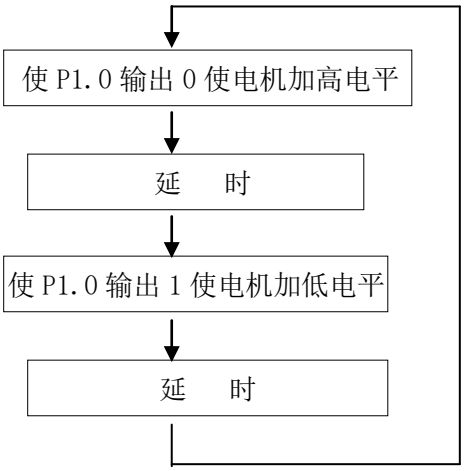
1 按实验电路图连接直流电机，编程使直流电机转动。

2、接线：P10　/C51 单片机　　接　　直流电机



四、参考流程图

2、直流电机转



五、参考程序　　ZLDJ.ASM

```
1、      ORG     00H
START:   CLR     P1.0              ;置 P1.0 低电平,使直流加低电平
         MOV     R0,     #250      ;延时
         CALL    DELAY
         SETB     P1.0             ;P1.0 高电平，使直流加高电平
         MOV     R0,     #150      ;低电平延时
         CALL    DELAY             ;延时
         SJMP    START
DELAY:   DJNZ    R0,     DELAY     ;延时子程序
         RET
```

实验十四　　步进电动机控制实验

一、实验目的

了解单片机控制步进电机的基本原理。

二、实验原理

1、步进电机驱动原理

步进电机驱动原理是通过对每相线圈中的加电顺序切换来使电机作步进式旋转。驱动电流由脉冲信号来控制，调节脉冲信号的频率便可改变步进电机的转速。本实验所用的步进电机型号为：20BY-0(四相四拍)永磁步进电动机，电机线圈由四相组成，即：ϕ1(A 相);ϕ2(B 相);ϕ3(C 相);ϕ4(D 相)，驱动方式为二相激励方式，各线圈通电顺序与步进方向的关系如下表。

相 顺序	A 相	B 相	C 相	D 相	逆时针转动 ↑ | ↓ 顺时针转动
0	1	1	0	0	
1	0	1	1	0	
2	0	0	1	1	
3	1	0	0	1	

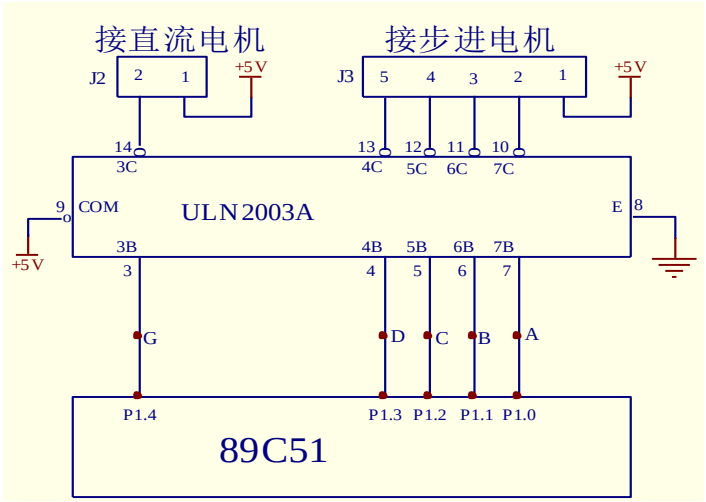
若首先向 A、B 相加电，接着 B、C 相加电；C、D 相加电、D、A 相加电，之后又返回到 A、B 相，电机按顺时针方向旋转。反之，电机按逆时针方向旋转。每一次加电，转动的角度（步进角）为 18 度。通过改变加电的频率(步进电机脉冲频率),可改变步进电机的速度。通过改变加电的顺序，可改变转动的方向。

三、实验电路与内容

实验电路图所示，ULN2003A 为达林顿晶体管阵列，内含 7 个达林顿晶体管，xB（x=1~7）为达林顿晶体管的输入（基极）端，xC 为达林顿晶体管的输出（集电极）端。实验内容：

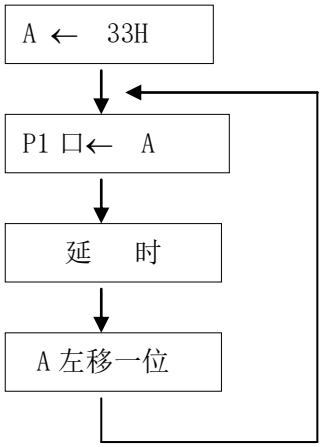
1、按实验电路图连接步进电机，编程输出脉冲序列，使步进电机按顺时针、逆时针转动。

2、接线：　P13~P10 /C51 单片机　　接　　BD~BA /步进电机



四、参考流程图

1、步进电机按顺时针转动



五、参考程序　　BJDJ.ASM

```
1、  ORG     OOH
     MOV     A,     #33H          ;A 相、B 相加电
START: MOV   P1,     A
     CALL    DELAY                ;延时
     RL      A
     SJMP    START
DELAY: MOV   RO,     #OFFH         ;延时子程序
     MOV     R1,     #O
DELAY1: DJNZ R1,     DELAY1
     DJNZ     RO,     DELAY1
     RET
     END
```

实验十五　PS2 键盘控制

一、实验目的

1、掌握 8051 单与 PS2 键盘控制编程方法

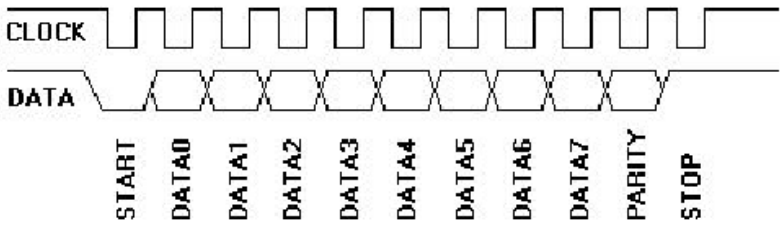
2、掌握 PS2 键盘编程方法

二、实验内容

1、从键盘/鼠标发送到主机的数据在时钟信号的下降沿 当时钟从高变到低的时候 被读取 从主机发送到键盘/鼠标的数据在上升沿 当时钟从低变到高的时候 被读取 不管通讯的方向怎样 键盘/鼠标总是产生时钟信号 如果主机要发送数据 它必须首先告诉设备开始产生时钟信号 这个过程在下一章节中被描述 最大的时钟频率是 33kHz 而且大多数设备工作在 10 20kHz 如果你要制作一个 PS/2 设备 我推荐你把频率控制在 15kHz 左右 这就意味着时钟应该是高 40 微秒低 40 微秒

2、键盘和鼠标使用一种每帧包含 11 位的串行协议 这些位含义是

1 start bit.　This is always 0.	1 个起始位，总是为 0
8 data bits, least significant bit first.	8 个数据位，低位在前
1 parity bit (odd parity).	1 个校验位，奇校验
1 stop bit.　This is always 1.	1 个停止位，总是为 1



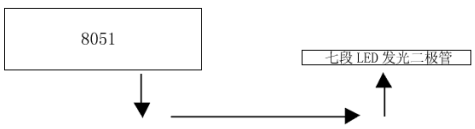
3、接线：P20 /C51 单片机　　接　　DAT /PS2 键盘

P21 /C51 单片机　　接　　CLK /PS2 键盘

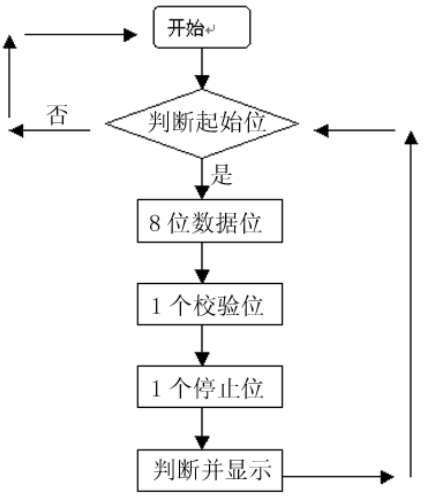
P17~P10 /C51 单片机　接　　dp~a /并行数码管

VCC　　接　　S0/并行数码管

AT Computer		
Signals	DIN41524, Female at Computer, 5-pin DIN 180°	6-pin Mini DIN PS2 Style Female at Computer
Clock	1	5
Data	2	1
nc	3	2,6
GND	4	3
+5V	5	4
Shield	Shell	Shell



三、流程图



四、参考程序

PS2.ASM

```
DAT    EQU   P2.0
CLK    EQU   P2.1

          ORG   0000H
          AJMP  KEY
          ORG   0100H
KEY:   JB   DAT,KEY
          JB   CLK,KEY
       MOV  R0,#9
       MOV  R1,#00H
       MOV  A,#00H
KEY0:  JB  CLK,KEY0
       JB  DAT,KEY1
       CLR  C
       RRC  A
       MOV  R1,A
       DJNZ R0,KEY3
       LJMP KEY2
KEY1:  SETB C
       RRC  A
       MOV  R1,A
       DJNZ R0,KEY3
       LJMP KEY2
```

```
KEY3:   JNB  CLK,KEY3
        LJMP KEY0
KEY2:   LCALL DISP
        LCALL DELAY
        LCALL DELAY
        LJMP KEY
DISP:   MOV  R6,#0
          MOV  R7,#16
DI0:    MOV  A,R6
        MOV  DPTR,#TAB
        MOVC A,@A+DPTR
        MOV  B,R1
        CJNE A,B,DI1
        LJMP DI2
DI1:    INC  R6
        DJNZ R7,DI0
        MOV  R6,#16
DI2:    MOV          DPTR,        #TABLE
        MOV          A,R6
        CJNE    A,#16,DI3
        LJMP    DI4
DI3: MOVC        A,@A+DPTR
        MOV          P1,          A
DI4:    RET
DELAY:  MOV  R6,#200
DEL:    MOV  R7,#255
        DJNZ R7,$
        DJNZ R6,DEL
        RET
TABLE:  DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH;0~~9 的段码表
            DB       77H,7CH,39H,5EH,79H,71H
TAB:    DB   45H,16H,1EH,26H,25H,2EH,36H,3DH,3EH,46H,1CH,32H,21H,23H,24H,2BH   ;PS 键
        盘 0~F 的键码
        DB   0FFH,0FFH,0FFH
        END
```

实验十六　DS18B20 测温实验

1-Wire 总线编程知识

简单介绍一些 1－Wire 总线编程时常用的子程序。

```
;18B20 初始化子程序，使用 R7
Initial:  CLR     DQ          ;数据线为低 700 微秒
          LCALL   DLY700
          SETB    DQ                      ;置数据线为高，以便接收 18B20 信号
I1:       MOV     C,DQ              ;等待数据线为低
          JC      I1
          LCALL   DLY700            ;延时 700 微秒
          RET
;向 18B20 写 8 位数据子程序，被写的数在 ACC 中，使用 R2、R7
W8BIT:    MOV     R2,#8             ;设置计数器 R2 为 8
W1:       CLR     DQ                      ;使数据线为低
          RRC     A                       ;右移，将被写位移到进位 C
          JNC     W2                      ;被写位为 0，转 W2
          SETB    DQ                      ;被写位为 1，使数据线为高
W2:       LCALL   DLY70             ;延时 70 微秒
          SETB    DQ                      ;使数据线为高
          DJNZ    R2,W1             ;如果 8 位数据没写完，转 W1
          RET
;从 18B20 读 8 位数据子程序，读的数在 ACC 中
;使用 R2、R7
R8BIT:    MOV     R2,#8             ;设置计数器 R2 为 8
RR1:      CLR     DQ                      ;使数据线为低，启动读过程
          NOP                             ;等待 4 微秒
          SETB    DQ                      ;使数据线为高，以便从 18B20 读数据
          NOP                             ;等待 4 微秒
          MOV     C,DQ              ;将数据读入进位 C
          RRC     A                       ;通过右移将进位 C 送入 ACC
          LCALL   DLY70             ;延时 70 微秒
          DJNZ    R2,RR1            ;如果 8 位数据没读完，转 RR1
          RET
```

1-Wire 总线的其它知识在第一章里有介绍，这里就不再赘述了。

一、实验目的

通过 1-Wire 总线接口的测温芯片 DS18B20 学习 1-Wire 总线

二、实验电路原理图及其说明

主机用 P1.2 模拟 1-Wire 总线

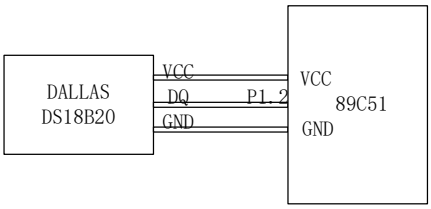


图　用 P1.2 模拟 1-Wire 总线

这里简单介绍一下 DALLAS 公司的可编程数字测温芯片 DS18B20。

特性描述：

1．通过 1-Wire 总线与 CPU 通讯

2．有唯一 64 位串行码存储在 ROM 里，可允许多个设备在一条 1-Wire 总线上工作

3．可以从数据线上得到电源供应，范围为 3V~5.5V

4．测温范围-55℃～ +125℃（-67℉～+125℉）

5．从-10℃～ +85℃范围，精度是+ / -0.5℃

6．表示温度的数据用户可以从 9 位～12 位任意选择

7．所采数据转换为 12 位数字信号可在 750ms 内完成

8．用户可自定义非易失性的报警设置

9．报警搜索命令可以识别和定位温度超过程序限定的设备

10．与 DS1822 软件兼容

11．应用方向如恒温调控，工业系统，数字温度计，或者任何温控敏感系统

12．8 脚封装或者 3 脚封装

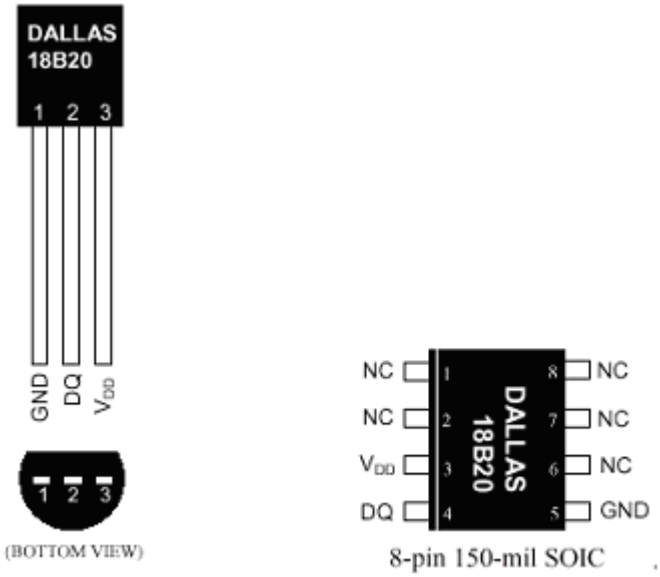


图 4-3-3 **封装形式**

管脚描述：

DQ：数据输入输出脚

GND：地 ground

VCC：电源 power

NC：未用管脚

结构特点：

1． DS18B20 有一个显著的结构特点：当外部电源掉电时，其仍可正常工作。因为当数据线 DQ 为高电平 HI 时，可通过 DQ 脚上的上拉电阻为芯片供电，当数据线 DQ 为低电平时，可通过内部的 Cpp 充放电为芯片供电。

2．每一个 DS18B20 都有唯一一个 64 位编码存储在 ROM 里。64 位编码格式如下：

DS18B20 的另一个显著特点是：64 位激光 ROM 编码

8-BIT FAMILY CODE 包含了 1-Wire 的家族码 28H。

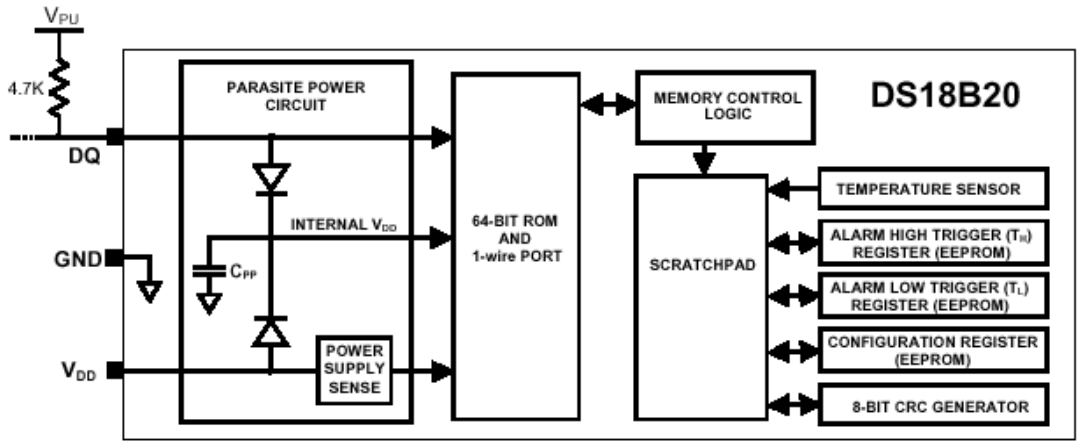


图 4-3-4　DS18B20 结构图

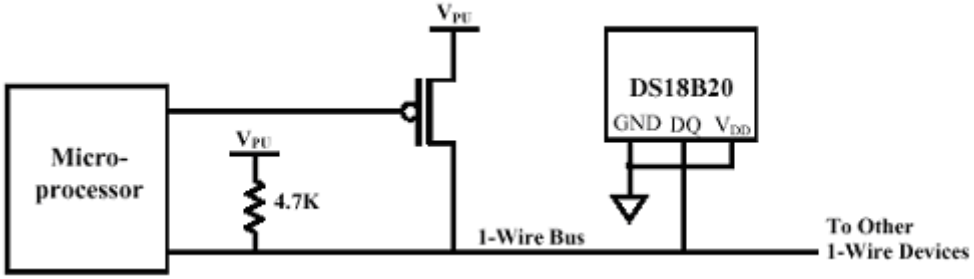


图 4-3-5　DS18B20 借电方式供电的情况

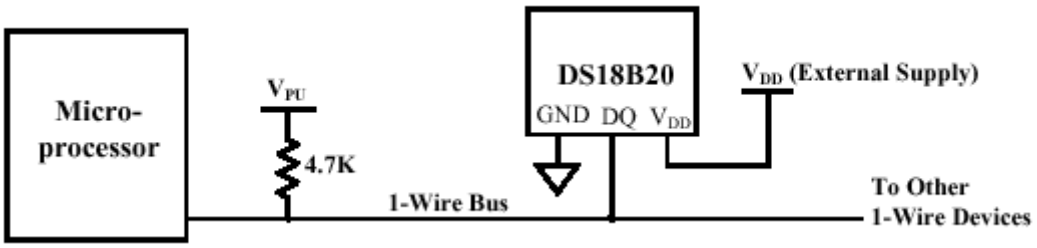


图 4-3-6　DS18B20 由外部电源供电的情况

48-BIT SERIAL NUMBER 包含了唯一的一串数值。

8-BIT CRC 是最重要的，它是一个由前 56 位计算出来的纠错码。

测温操作：

用 9 位、10 位、11 位、12 位表示温度数据的精度分别为 0.5℃、0.25℃、0.125℃和 0.0625℃。默认的是 12 位表示。上电时 DS18B20 为低电平 IDLE 状态，若初始化温度测量和 A-to-D 转换，需输入转换命令 Convert T(44H)。转换后的数据存到 2 个字节的温度寄存器，然后返回 IDLE 状态。DS18B20 可以通过报警搜索命令 ECH 来检查报警标志 flag，任何一个带有报警 flag 功能的 DS18B20 都会对这个命令反映，所以主机可以定位是哪个设备报警的。

TH 和 TL 寄存器的格式：

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
S	2^6	2^5	2^5	2^5	2^2	2^1	2^0

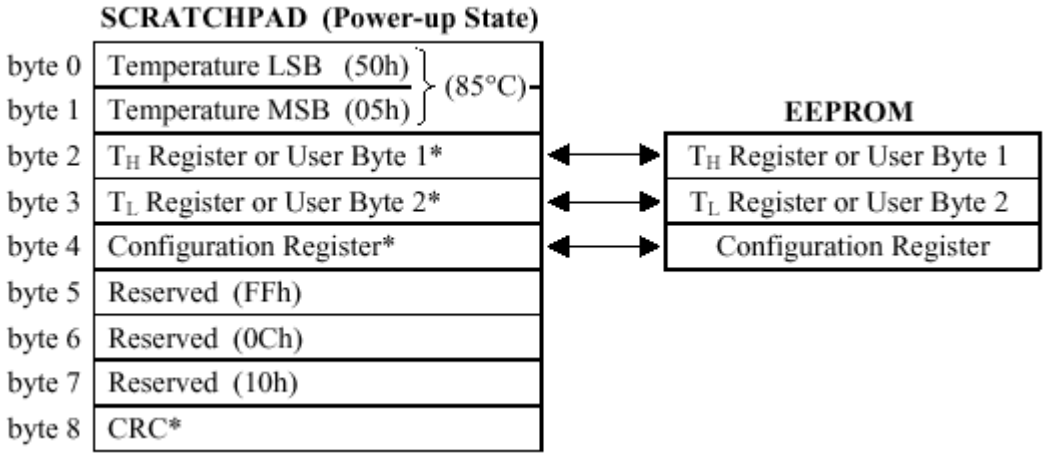
温度寄存器的格式：

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
MS Byte	S	S	S	S	S	2^6	2^5	2^4

报警操作：

在温度转换之后，温度值要与温度极限值寄存器（TH,TL）里的值比较，由于温度值是 12 位的，TH 和 TL 是 8 位寄存器，所以温度值的 4～11 位参与比较，无论是高于 TH 还是低于 TL，均符合报警条件，DS18B20 里的报警标志 flag 被设置报警，在每次温度测量之后报警 flag 又被复位，取消报警状态。

DS18B20 的中间结果寄存器的存储结构：



Temperature LSB ：温度低字节寄存器

Temperature MSB ：温度高字节寄存器

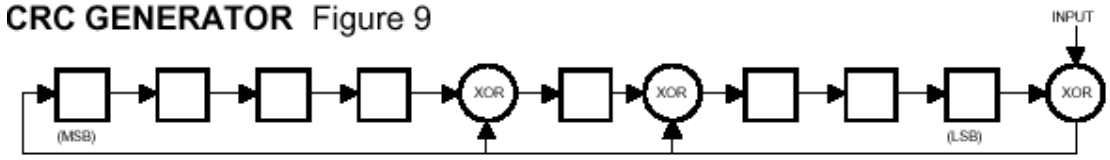
TH Register or User Byte 1*：高温极限寄存器或者用户自定义字节 1

TL Register or User Byte 2*：低温极限寄存器或者用户自定义字节 2

CRC*:是 64 位 ROM 码中的 8-BIT CRC，它可通过如下公式计算得到：

$$CRC = X^8 + X^5 + X^4 + 1$$

多项式发生器如图：



1-Wire 总线上的主机再次计算 CRC，然后与 DS18B20 通过以上的多项式发生器计算的 CRC 比较，以此方法纠错。

Configuration Register*：配置寄存器

配置寄存器格式：

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	R1	R0	1	1	1	1	1

用户可以通过配置寄存器中的 R0、R1 来设置温度转换方案。

表 4-3-1　DS18B20 转换方案列表

R1	R0	Resolution	Max Conversion Time	
0	0	9-bit	93.75 ms	(t_{CONV}/8)
0	1	10-bit	187.5 ms	(t_{CONV}/4)
1	0	11-bit	375 ms	(t_{CONV}/2)
1	1	12-bit	750 ms	(t_{CONV})

表 4-3-2　DS18B20 的指令系统

命　　令	描　　述	协议	1-Wire 总线的反应
转换温度	启动温度转换	44H	DS18B20 将转换状态传给主机（借电供电的情况不能用此命令）
读暂存器	读暂存器	BEH	DS18B20 传输 9 个数据字节给主机
写暂存器	写数据到暂存器的 byte2、3、4（TH,TL 和 configuration registers）	4EH	主机传输 3 个字节给 DS18B20
复制暂存器	从暂存器中复制 TH,TL 和 configuration registers 到 EEPROM.	48H	无
回读 EEPROM	把 TH,TL 和 configuration registers 的值从 EEPROM 回读至暂存器中	B8H	DS18B20 传输设置状态给主机
Read Power Supply	把 DS18B20 的电源模式发信号给主机	B4H	DS18B20 传输供电状态给主机

表 4-3-3 温度与数据的关系表

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C*	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FE6Fh
-55°C	1111 1100 1001 0000	FC90h

三、实验内容

4 个数码管 3 位显示温度值。

四、实验步骤

1、实验接线

P3.2/C51　　接　　DOI/DS18B20

P1.7~P1.0　　接　　dp~a/并行数码管

P2.3~P2.0　　接　　S3~S0/并行数码管

2、实验现象

数码管显示当前环境温度

说明：因温度传感器和实验箱本身会发热，显示温度与实际温度有一定的差异。

如需实际可考虑在传感上加上散热片采集更接近实际的温度，传感器应远离有发热源的位置。可以程序调节差异值。

五、参考程序

```
N    EQU        8                    ;读出 ROM 的位数
DQ   EQU        P3.2                 ;单总线的数据线
TH   EQU        40                   ;最高极限温度
TL   EQU        10                   ;最低极限温度
SROMC EQU       0F0H                 ;搜索 ROM 命令
MROMC EQU       55H                  ;匹配 ROM 命令
LROMC EQU       0CCH                 ;跳过读 ROM 命令
RROMC EQU       33H                  ;读 ROM 命令
CTC EQU         44H                  ;转换温度命令
WREGC EQU       4EH                  ;写暂存器命令
RREGC EQU       0BEH                 ;读暂存器命令
CONFIGEQU       1FH                  ;配置寄存器的值,9 位温度值
    DSEG        AT 30H
DAT:  DS        1                    ;温度值存储单元
POINT:DS        1                    ;小数点后温度值的存储单元
ROMA: DS        8                    ;器件 A 的 ROM 存储单元
BUF:  DS        9                    ;暂存器中数据存储单元
BUFT: DS        4                    ;数码管动态显示的 4 个数据缓冲单元
BTH:  DS        1                    ;显示 ROM 中某一字节高四位
BTL:  DS        1                    ;显示 ROM 中某一字节低四位
    CSEG        AT 0000H
    LJMP        MAIN
    ORG         0100H
MAIN: MOV       SP,#70H
    MOV         PSW,#00H
    LCALL       Initial
    MOV         A,#LROMC   ;写跳过 ROM 命令
    LCALL       W8BIT
    MOV         A,#WREGC   ;写暂存器命令，可写入 TH,TL,Configuration
    LCALL       W8BIT
    MOV         A,#TH                ;写 18B20 的 TH 寄存器
    LCALL       W8BIT
    MOV         A,#TL                ;写 18B20 的 TL 寄存器
    LCALL       W8BIT
    MOV         A,#CONFIG  ;写 18B20 的配置寄存器,9 位温度值
    LCALL       W8BIT
```

```
RETN: LCALL     Initial             ;18B20 初始化
    MOV         A,#LROMC   ;写跳过 ROM 命令
    LCALL       W8BIT
    MOV         A,#CTC              ;写转换温度命令
    LCALL       W8BIT
BUSY: LCALL     Disply              ;显示温度
    CLR         DQ                  ;读 18B20 的忙位
    NOP
    SETB        DQ
    NOP
    MOV         C,DQ
    LCALL       DLY70               ;延时 70 微秒
    JNC         BUSY                ;如果 18B20 忙，转 BUSY
    LCALL       Initial             ;初始化
    MOV         A,#LROMC   ;写跳过 ROM 命令
    LCALL       W8BIT
    MOV         A,#RREGC   ;写读 18B20 暂存器命令
    LCALL       W8BIT
    MOV         R3,#2               ;设置计数器 R3=2,从 18B20 读 2 个字节的温度值
    MOV         R1,#BUF             ;置 R1 温度值缓冲区
RER:  LCALL     R8BIT               ;读温度值
    MOV         @R1,A               ;将温度值存入缓冲区
    INC         R1                  ;温度值缓冲区地址加 1
    DJNZ        R3,RER              ;如 2 个字节温度值没读完，转 RER
    LCALL       STORE               ;温度转换并存储子程序
    LCALL       Disply              ;温度显示子程序，只所以调用 2 次该子程序
                                    ;是为了增强数码管的亮度
    SJMP        RETN
    ;18B20 初始化子程序，使用 R7
Initial:CLR     DQ                  ;数据线为低 700 微秒
    LCALL       DLY700
    SETB        DQ                  ;置数据线为高，以便接收 18B20 信号
I1: MOV         C,DQ                ;等待数据线为低
    JCI1
    LCALL       DLY700              ;延时 700 微秒
    RET
    ;向 18B20 写 8 位子程序，被写的数在 ACC 中
    ;使用 R2、R7
```

```
W8BIT:MOV       R2,#8               ;设置计数器 R2 为 8
W1: CLR         DQ                  ;使数据线为低
    RRC         A                   ;右移将被写位移如进位 C
    JNC         W2                  ;被写位为 0，转 W2
    SETB        DQ                  ;被写位为 1，使数据线为高
W2: LCALL       DLY70               ;延时 70 微秒
    SETB        DQ                  ;使数据线为高
    DJNZ        R2,W1               ;如果 8 位数据没写完，转 W1
    RET
    ;读 8 位子程序，读的数在 ACC 中
    ;使用 R2、R7
R8BIT:MOV       R2,#8               ;设置计数器 R2 为 8
RR1:  CLR       DQ                  ;使数据线为低，启动读过程
    NOP                             ;等待 4 微秒
    SETB        DQ                  ;使数据线为高，以便从 18B20 读数据
    NOP                             ;等待 4 微秒
    MOV         C,DQ                ;将数据读入进位 C
    RRC         A                   ;通过右移将进位 C 送入 ACC
    LCALL       DLY70               ;延时 70 微秒
    DJNZ        R2,RR1              ;如果 8 位数据没读完，转 RR1
    RET
    ;温度值存储子程序,使用 R0
STORE:MOV       R0,#BUF             ;将温度缓冲区首址送 R0
    MOV         A,@R0
    MOV         B,A
    ANL         A,#08H              ;保留小数位
    MOV         POINT,A
    MOV         A,B
    ANL         A,#0F0H             ;保留高 4 位
    MOV         B,A
    INC         R0
    MOV         A,@R0
    ANL         A,#0FH              ;保留低 4 位
    ORL         A,B                 ;组成 1 个字节
    SWAP        A
    MOV         R0,#BUFT
    JBACC.7,NEG ;如果温度为负,转 NEG
    MOV         DAT,A               ;温度为正值,直接存到 DAT 里
```

```
    MOV     A,#00H          ;正号显示
    MOV     @R0,A
    JMP     NST
NEG:  CPL   A               ;将相应的负值转换为其绝对值
    MOV     B,A
    MOV     A,POINT
    CPL     A
    CLR     ACC.4
    ADD     A,#08H
    MOV     C,ACC.4
    XCH     A,B
    ADDC    A,#0
    MOV     DAT,A           ;把负温度值 1-8 位变正存储
    MOV     POINT,B         ;把负温度值第 9 位变正存储
    MOV     A,#40H          ;负号显示
    MOV     @R0,A
NST:  INC   R0
    MOV     A,DAT           ;取温度值
    MOV     B,#10           ;十位数在 ACC 中,个位数在 B 中
    DIV     AB
    MOV     DPTR,#TAB  ;将十位数转换为数码管字形码值存入 BUFT 中
    MOVC    A,@A+DPTR
    MOV     @R0,A
    INC     R0              ;字形码缓冲区地址加 1
    MOV     A,B
    MOVC    A,@A+DPTR  ;将个位数字形码连同小数点存入 BUFT 中
    SETB    ACC.7
    MOV     @R0,A
    INC     R0              ;字形码缓冲区地址加 1
    MOV     A,POINT         ;取小数点后的值字形码存入缓冲区 BUFT 中
    JBACC.3,FLOAT
    MOV     A,#3FH
    MOV     @R0,A
    JMP     E
FLOAT:MOV   A,#6DH
    MOV     @R0,A
E:  RET
```

```
    ;显示温度子程序，使用了 DPTR、R0、R5。
    ;BUFT 中存放的是温度值的字形码
Disply:     MOV     R0,#BUFT
        MOV     A,@R0
        MOV     P1,A        ;送出正负位的 7 段代码
        SETB    P2.3        ;开正负位显示
        CALL    DISP
        CLR     P2.3
        MOV     A,@R0
        MOV     P1,A        ;送出十位的 7 段代码
        SETB    P2.2        ;开十位显示
        CALL    DISP
        CLR     P2.2
        MOV     A,@R0
        MOV     P1,A        ;送出个位的 7 段代码
        SETB    P2.1        ;开个位显示
        CALL    DISP
        CLR     P2.1
        MOV     A,@R0
        MOV     P1,A        ;送出小数位的 7 段代码
        SETB    P2.0        ;开小数位显示
        CALL    DISP
        CLR     P2.0
        RET
    ;显示一个数字子程序
DISP:   MOV     R5,#250             ;等待 600 微秒
        DJNZ    R5,$
        INC     R0                  ;BUFT 字形码缓冲区地址加 1,准备显示下一个数字
        RET
DLY700:     MOV     R7,#157             ;延时 700 微秒子程序，使用 R7
    DJNZ        R7,$
DLY70:MOV       R7,#17              ;延时 70 微秒子程序，使用 R7
    DJNZ        R7,$
    RET

    ;延时 1 分钟子程序，使用 R5、R6、R7
DLY1M:MOV       R5,#240
LOOP1:MOV       R6,#250
```

```
LOOP2:MOV       R7,#250
      DJNZ      R7,$
      DJNZ      R6,LOOP2
      DJNZ      R5,LOOP1
      RET

TAB:    DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH          ;a~f 字形码
      END
```

实验十七　串行 A/D-D/A(PCF8591)转换实验

一、PCF8591 介绍

PCF8591 为带 I2C 总线接口的 8 位 A/D、8 位 D/A 转换器。它有四个 A/D 输入通道：AD0，AD1，AD2，AD3。八个 PCF8591 可以同时挂在 I2C 总线上。八个 PCF8591 可用硬件地址线 A2，A1，A0 加以区别。四个 A/D 输入通道：IN0，IN1，IN2，IN3。

PCF8591 有一路 D/A 输出：可用程序实现，从 OUT 输出方波、锯齿波，及其它波形。

现简述一下 PCF8591

I　PCF8591－I2C 总线的串行 A/D、D/A 转换芯片简介

(1) 芯片封装和管脚描述

SYMBOL	PIN	DESCRIPTION
AINO	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I^2C-bus data input/output
SCL	10	I^2C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

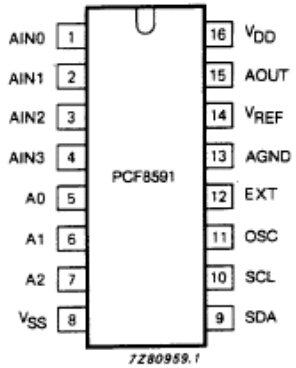


Fig.2 Pinning diagram.

PCF8591 封装及管脚说明

(2) 功能描述

如图 4-4-3 所示，PCF8591 读写命令的格式如下

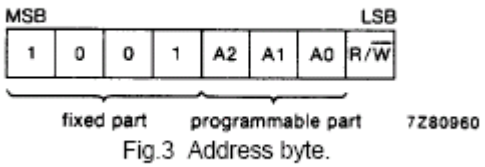


Fig.3 Address byte.

PCF8591 读写命令的格式

高 4 位是固定的，A0～A2 就是 PCF8591 芯片管脚 5～7，其值取决于硬件电路的连接，如图 4-4-1。R/＃W 是表示数据的传输方向，R/＃W=1 表示‘读’，即数据由 PCF8591 向主机；R/＃W=0 表示‘写’，即数据由主机向 PCF8591。

总上所述，PCF8591 的读命令是 91H，写命令是 90H。

PCF8591 的控制字节：

0	Bit6	Bit5	Bit4	0	Bit2	Bit1	Bit0

Bit1 、Bit0 用来选择 A/D 通道：

00　　选择 AIN0

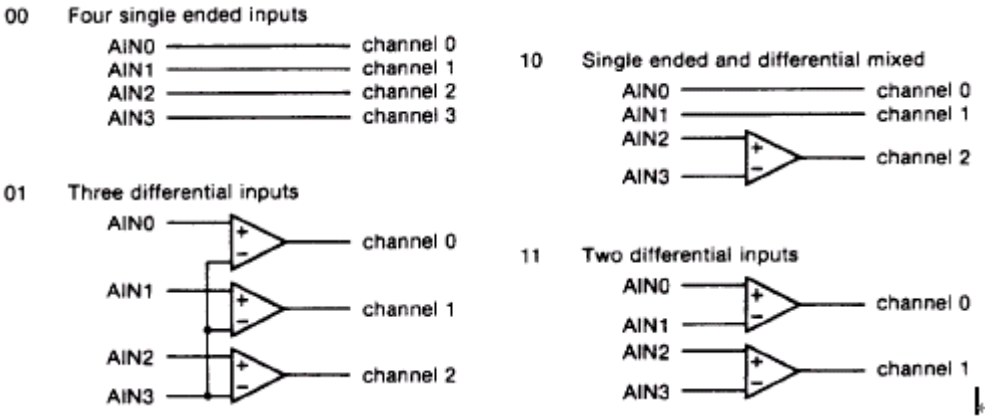
01　　选择 AIN1

02　　选择 AIN2

03　　选择 AIN3

Bit2：指向模拟通道的指针自动加一的控制位，Bit2＝1，打开此项功能；Bit2＝0，禁止此项功能。此实验中，Bit2＝0。

Bit5、Bit4 用来选择模拟输入的模式：

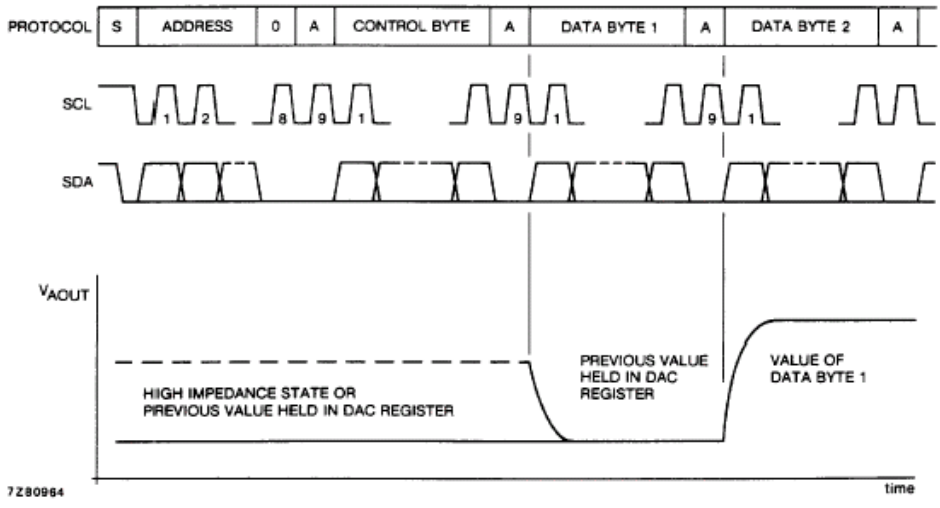


PCF8591 电压输入方式图

Bit6：模拟输出使能标志位。Bit6＝1，激活此项；Bit6＝0 禁止此项。Bit6 与 Bit2 是配合使用的，详细的请查看相关资料，此实验选 Bit6＝0。

注意：PCF8591 的控制字节上电复位，其值默认是‘0’，符合这个实验的要求，所以程序里没有重新设置其值，若读者需要，可根据自己的情况设置。

D/A 转换时序：



D/A 转换时序图

一、实验目的

通过芯片 PCF8591，来学习串行 AD/DA 转换。

二、实验内容

1、PCF8591 模数转换实验。

2、PCF8591 数模转换实验。

三、实验步骤

1、接线

SDA ------ P1.3

SCK ------ P1.4

IN0 ------ 2（10K 电位器）

+5V ------ 1（10K 电位器）

GND ------ 3（10K 电位器）

四、参考程序

1、PCF8591 模数转换实验。随着旋转电位器，模拟输入 0 到+5V 之间变化，在 4 个数码管上显示 A/D 转换后的结果，按十进制显示电压值。

```
SDA     EQU     P1.3        ;I2C 数据线
SCL     EQU     P1.4        ;I2C 时钟线
ADRW    EQU     90H         ;PCF8591 写,PCF8591 的地址是 92H,最低位为 0 表示写
ADRR    EQU     91H         ;PCF8591 读,PCF8591 的地址是 92H,最低位为 1 表示读
DAT     EQU     30H
BUFTD   EQU     31H
        ORG     0000H
        LJMP    MAIN
        ORG     0100H
MAIN:   MOV     SP,#60H
        MOV     PSW,#00H
LOOP:   LCALL   ADCBYT      ;调用读模数转换后的数据子程序
        MOV     DAT,A       ;将读出的电压值放在 DAT 中
        LCALL   STORE       ;将读出的数据转换成电压值,并形成字形码送;BUFTD 缓冲区
        LCALL   DISP
        LCALL   DELAY
        SJMP    LOOP
DISP:   MOV     R7,#4
        MOV     R0,#BUFTD
DISP1:  MOV     A,@R0
        MOV     SBUF,A
        JNB     TI,$
        INC     R0
```

```
        MOV     R6,#50
        DJNZ    R6,$
        DJNZ    R7,DISP1
        RET
DELAY:  MOV     R6,#255
DEL0:   MOV     R7,#255
        DJNZ    R7,$
        DJNZ    R6,DEL0
        RET
;启动 PCF8951 进行模数转换并读数据子程序
;使用参数 F0,R0
ADCBYT: LCALL   STAR                ;发出 I2C 总线数据传输命令
        MOV     A,#ADRR             ;发出读 PCF8951 命令,启动 A/D 转换
        LCALL   WRBYT
        LCALL   CACK                ;调用应答位检测子程序, F0=0 正常应答,F0=1 非正
常应答
        JB      F0,ADCBYT  ;非正常应答转 ADCBYT 重新开始
        LCALL   RDBYT               ;读上次转换后的电压值(8 位)
        LCALL   MNACK               ;主机发送非应答位子程序
        LCALL   STOP                ;结束这次 I2C 数据传输
        RET
;字形码表生成子程序,使用 R0、DPTR、B
STORE:  MOV     R0,#BUFTD  ;将电压值字形码缓冲区的首址送 R0
        MOV     A,#0FFH             ;送 0FF 到字形码缓冲区的第一字节,即不显示
        MOV     @R0,A
        INC     R0                  ;字形码缓冲区地址加 1
        MOV     A,DAT                 ;取出电压值
        MOV     B,#51                 ;除 51 得到电压值的个位
        DIV     AB
        MOV     DPTR,#TAB  ;从字形码表中取出个位的字形码
        MOVC    A,@A+DPTR
        ANL     A,#0FEH           ;加上小数点
        MOV     @R0,A
        INC     R0                    ;字形码缓冲区地址加 1
        MOV     A,B                 ;余数送 A
        CJNE    A,#50,E2
        DEC     A                   ;等于 50 应减 1,以免移位后变成 100,这里做近似处理
E2:     RL      A                     ;余数*2,得到真正的十进制余数
```

```
        MOV     B,#10               ;得到小数点后的第一位数在 A 中,第二位在 B 中
        DIV     AB
        CALL    FINDOUT     ;将小数点后的第一位字形码存入字形码表中
        MOV     A,B
        CALL    FINDOUT     ;将小数点后的第二位字形码存入字形码表中
        RET
;将数字转换成字形码送字形码缓冲区子程序
;使用 DPTR、ACC、R0
;入口:ACC 中存放被转换数字
FINDOUT:MOV     DPTR,#TAB  ;取字形码表首地址到 DPTR
        MOVC    A,@A+DPTR  ;转换成字形码并放入字形码缓冲区中
        MOV     @R0,A
        INC     R0                  ;字形码缓冲区地址加 1
        RET
;向 SDA 线上发送一个数据字节子程序，使用了 R0
WRBYT:  MOV     R0,#08H             ;8 位数据长度送 R0 中
WLP:    RLC     A                   ;发送数据左移，把发送位送入进位 C 中
        JC      WRE1                ;判断发送"1"还是"0"，发送"1"转 WRE1
        SJMP    WRE0                ;发送"0"转 WRE0
WLP1:   DJNZ    R0,WLP              ;8 位是否发送完，未完转 WLP
        RET                         ;8 位发送完，返回
;发送"1"程序段
WRE1:   SETB    SDA                 ;置数据信号 SDA 为高
        SETB    SCL                 ;置时钟信号 SCL 为高
        NOP                         ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        SETB    SCL
        NOP
        NOP
        CLR     SCL                 ;使时钟信号 SCL 为低
        CLR     SDA                 ;使数据信号 SDA 为低
        SJMP    WLP1
;发送"0"程序段
WRE0:   CLR     SDA                 ;使数据信号 SDA 为低
        NOP
        NOP
        SETB    SCL                 ;置时钟信号 SCL 为高
        NOP                         ;NOP 为满足 I2C 协议所需的等待时间
```

```
        NOP
        CLR     SCL                         ;使时钟信号 SCL 为低
        SJMP    WLP1
;从 SDA 线上读取一个数据字节子程序,放入 R2 中
RDBYT:  MOV     R0,#08H             ;8 位数据长度送入 R0
RLP:    SETB    SDA                 ;置 SDA 为输入方式
        SETB    SCL                 ;置时钟信号 SCL 为高,使 SDA 上数据有效
        CLR     C                   ;默认读数为 0，置进位 C=0
        JNB     SDA,RD0     ;读数为 0，转 RD0
        SETB    C           ;读数为 1，置进位 C=1
RD0:    MOV     A,R2        ;将读到的数（在进位 C 中），通过循环左移放到 R2 中
        RLC     A
        MOV     R2,A
        CLR     SCL
        DJNZ    R0,RLP              ;没有读完 8 位数据转 RLP
        RET                         ;读完 8 位数据返回
;START 信号子程序
STAR:   SETB    SCL                 ;置时钟信号 SCL 为高
        SETB    SDA                 ;置数据信号 SDA 为高
        NOP                         ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        NOP
        NOP
        CLR     SDA                 ;使数据信号 SDA 为低
        NOP                         ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        NOP
        NOP
        CLR     SCL                 ;使时钟信号 SCL 为低
        RET
;主机发送应答位子程序
MACK:   CLR     SDA                 ;使数据信号 SDA 为低
        SETB    SCL                 ;置时钟信号 SCL 为高
        NOP                         ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL                 ;使时钟信号 SCL 为低
        SETB    SDA                 ;置数据信号 SDA 为高
        RET
```

```
;主机发送非应答位子程序
MNACK:    SETB    SDA              ;置数据信号 SDA 为高
          SETB    SCL              ;置时钟信号 SCL 为高
          NOP                      ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          CLR     SCL              ;使时钟信号 SCL 为低
          CLR     SDA              ;使数据信号 SDA 为低
          RET
;应答位检测子程序，F0=0 表示收到了正常应答
          ;F0=1 表示收到非正常应答
CACK:     CLR     SCL
          SETB    SDA              ;置数据信号 SDA 为高，以便接收数据
          SETB    SCL              ;置时钟信号 SCL 为高
          NOP
          NOP
          NOP
          NOP
          CLR     F0               ;默认为正常应答
          JNB     SDA,CEND   ;正常应答转 CEND
          SETB    F0               ;置错误应答标志
CEND:     CLR     SCL              ;使时钟信号 SCL 为低
          NOP                      ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          RET
;STOP 信号子程序
STOP:     CLR     SDA              ;使数据信号 SDA 为低
          SETB    SCL              ;置时钟信号 SCL 为高
          NOP                      ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          SETB    SDA              ;置数据信号 SDA 为高
          NOP                      ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          CLR     SCL              ;使时钟信号 SCL 为低
          RET

TAB:      DB 03H,9FH,25H,0DH,99H,49H,41H,1FH,01H,09H          ;0~9 字形码值
          END
```

2、PCF8591 数模转换实验。模拟信号输出正弦波

```
SDA       EQU       P1.3        ;I2C 数据线
SCL       EQU       P1.4        ;I2C 时钟线
ADRW      EQU       90H         ;PCF8591 写,PCF8591 的地址是 92H,最低位为 0 表示写
ADRR      EQU       91H         ;PCF8591 读,PCF8591 的地址是 93H,最低位为 1 表示读
CONTL     EQU       40H         ;PCF8591 控制字节,允许 D/A 转换输出
DSEG      AT 20H
DAT:      DS        1           ;PCF8591D/A 转换的数据存储单元
CSEG      AT 0000H
          LJMP      MAIN
          ORG       0100H
MAIN:     MOV       SP,#60H
          MOV       PSW,#00H
START:    MOV       R3,#0
          MOV       R4,#32
LOOP:     MOV       DPTR,#SIN_DAT
          MOV       A,R3
          MOVC      A,@A+DPTR
          MOV       DAT,A ;#VALUE       ;将数值送往 DAT 单元,供 D/A 转换使用
          LCALL     DACBYT      ;进行 D/A 转换
          INC       R3
          DJNZ      R4,LOOP
          LJMP      START
;D/A 转换子程序
;已用参数 F0,R0
DACBYT:   LCALL     STAR                ;发出启动传输命令
          MOV       A,#ADRW     ;发出对 PCF8591 寻址和写命令
          LCALL     WRBYT
          LCALL     CACK                ;调用应答子程序
          JB        F0,DACBYT   ;如果错误应答，转 DACBYT 重新开始
          MOV       A,#CONTL    ;发出对 PCF8591 的控制命令
          LCALL     WRBYT
          LCALL     CACK                ;调用应答子程序
          JB        F0,DACBYT   ;如果错误应答，转 DACBYT 重新开始
          MOV       A,DAT               ;从 DAT 中取出被转换数据
          LCALL     WRBYT               ;向 PCF8591 写要 D/A 转换的数据
          LCALL     CACK                ;调用应答子程序
          JB        F0,DACBYT   ;如果错误应答，转 DACBYT 重新开始
```

```
        LCALL   STOP                ;发送结束，发出停止命令
        RET
;向 SDA 线上发送一个数据字节子程序，使用了 R0
WRBYT:  MOV     R0,#08H             ;8 位数据长度送 R0 中
WLP:    RLC     A                   ;发送数据左移，把发送位送入进位 C 中
        JC      WRE1                ;判断发送"1"还是"0"，发送"1"转 WRE1
        SJMP    WRE0                ;发送"0"转 WRE0
WLP1:   DJNZ    R0,WLP              ;8 位是否发送完，未完转 WLP
        RET                         ;8 位发送完，返回
;发送"1"程序段
WRE1:   SETB    SDA                 ;置数据信号 SDA 为高

        NOP                         ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        SETB    SCL
        NOP
        NOP
        CLR     SCL                 ;使时钟信号 SCL 为低
        CLR     SDA                 ;使数据信号 SDA 为低
        SJMP    WLP1
;发送"0"程序段
WRE0:   CLR     SDA                 ;使数据信号 SDA 为低
        NOP
        NOP
        SETB    SCL                 ;置时钟信号 SCL 为高
        NOP                         ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL                         ;使时钟信号 SCL 为低
        SJMP    WLP1
;从 SDA 线上读取一个数据字节子程序,放入 R2 中
RDBYT:  MOV     R0,#08H             ;8 位数据长度送入 R0
RLP:    SETB    SDA                 ;置 SDA 为输入方式
        SETB    SCL                 ;置时钟信号 SCL 为高,使 SDA 上数据有效
        CLR     C                   ;默认读数为 0，置进位 C=0
        JNB     SDA,RD0     ;读数为 0，转 RD0
        SETB    C           ;读数为 1，置进位 C=1
RD0:    MOV     A,R2        ;将读到的数（在进位 C 中），通过循环左移放到 R2 中
RLC     A
```

```
MOV       R2,A
          CLR       SCL
          DJNZ      R0,RLP          ;没有读完 8 位数据转 RLP
          RET                       ;读完 8 位数据返回
;START 信号子程序
STAR:     SETB      SCL             ;置时钟信号 SCL 为高
          SETB      SDA             ;置数据信号 SDA 为高
          NOP                       ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          NOP
          NOP
          CLR       SDA             ;使数据信号 SDA 为低
          NOP                       ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          NOP
          NOP
          CLR       SCL             ;使时钟信号 SCL 为低
          RET

;主机发送应答位子程序
MACK:     CLR       SDA             ;使数据信号 SDA 为低
          SETB      SCL             ;置时钟信号 SCL 为高
          NOP                       ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          CLR       SCL             ;使时钟信号 SCL 为低
          SETB      SDA             ;置数据信号 SDA 为高
          RET

;主机发送非应答位子程序
MNACK:    SETB      SDA             ;置数据信号 SDA 为高
          SETB      SCL             ;置时钟信号 SCL 为高
          NOP                       ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          CLR       SCL             ;使时钟信号 SCL 为低
          CLR       SDA             ;使数据信号 SDA 为低
          RET
;应答位检测子程序，F0=0 表示收到了正常应答
          ;F0=1 表示收到非正常应答
```

```
CACK:     CLR     SCL
          SETB    SDA                 ;置数据信号 SDA 为高，以便接收数据

          SETB    SCL                 ;置时钟信号 SCL 为高
          NOP
          NOP
          NOP
          NOP
          CLR     F0                  ;默认为正常应答
          JNB     SDA,CEND    ;正常应答转 CEND
          SETB    F0                  ;置错误应答标志
CEND:     CLR     SCL                 ;使时钟信号 SCL 为低
          NOP                         ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          RET
;STOP 信号子程序
STOP:     CLR     SDA                 ;使数据信号 SDA 为低
          SETB    SCL                 ;置时钟信号 SCL 为高
          NOP                         ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          SETB    SDA                 ;置数据信号 SDA 为高
          NOP                         ;NOP 为满足 I2C 协议所需的等待时间
          NOP
          CLR     SCL                 ;使时钟信号 SCL 为低
          RET
SIN_DAT:
    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

          END
```

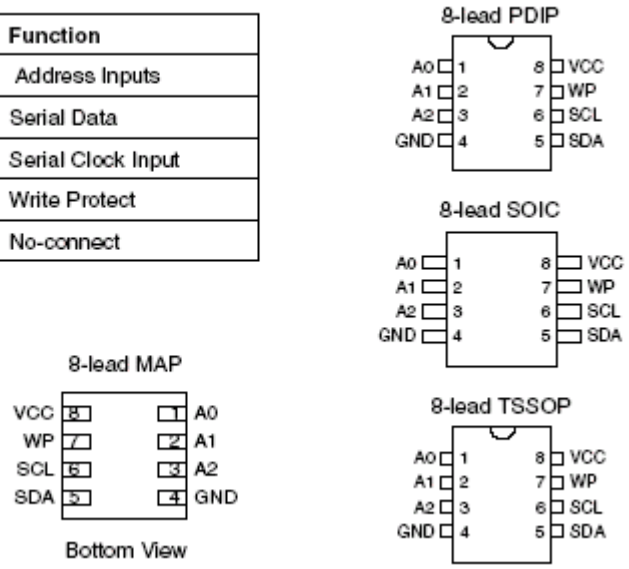
实验十八　　串行 EEPROM 实验

AT24C02－I2C 总线的 2K Serial EEPROM 芯片简介

1、AT24C02 的封装和管脚描述及结构图见下图。

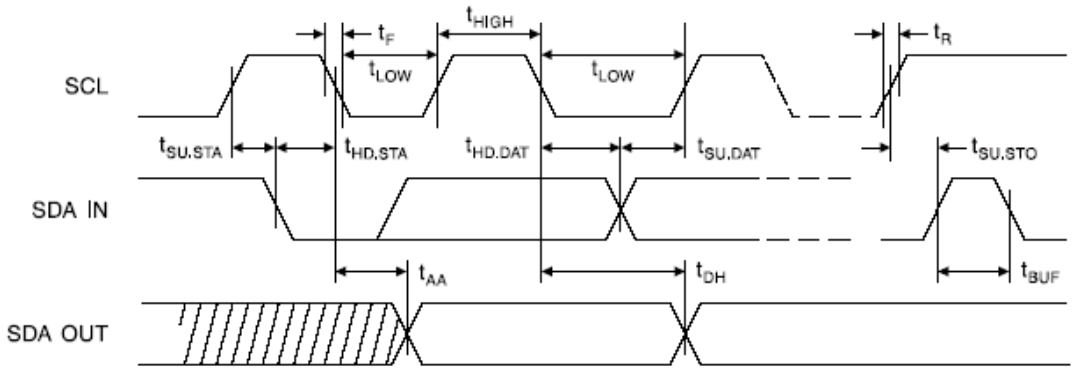
Pin Configurations

Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No-connect

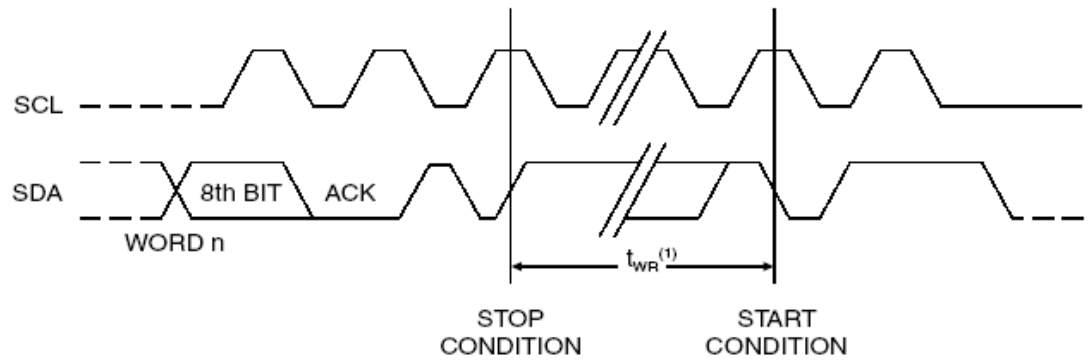


AT24C02 封装及管脚说明图

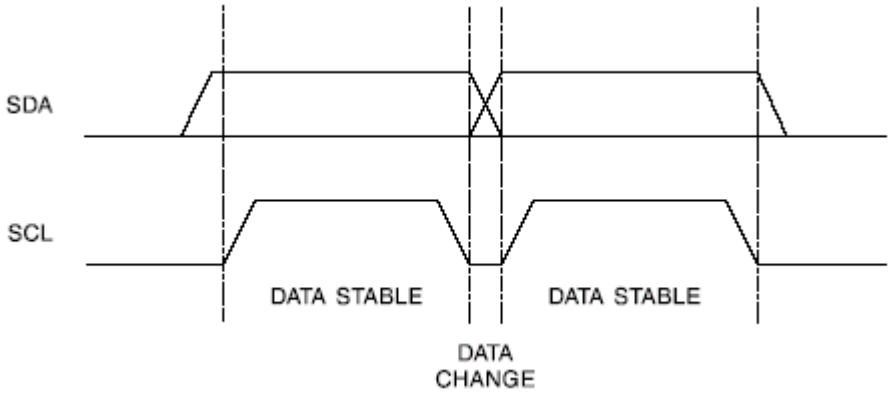
2、AT24C02 时序图



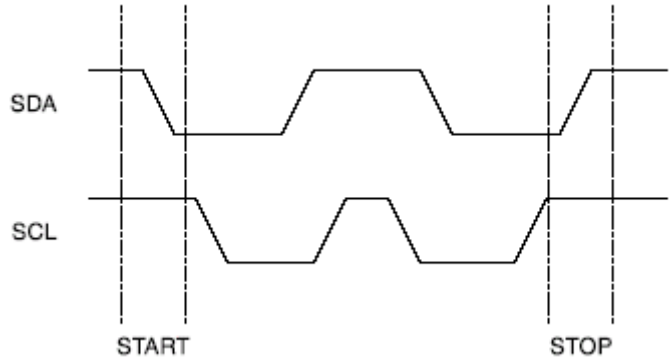
AT24C02 总线时序



AT24C02 写时序

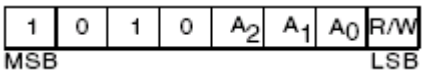


AT24C02 无效数据



AT24C02 的起始和终止

3、24C02 读写命令的确定：

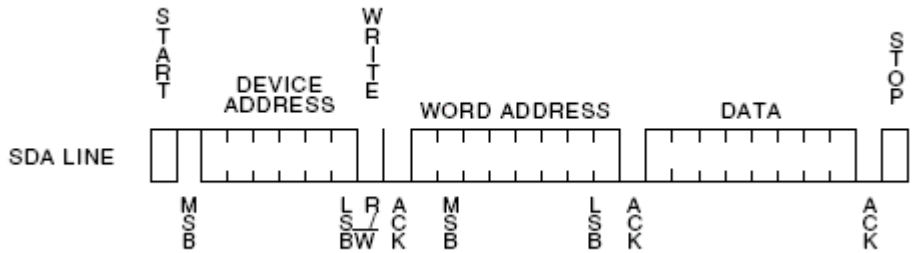


A0～A2 是 AT24C02 管脚 1～3，其值由硬件电路的连接决定的。

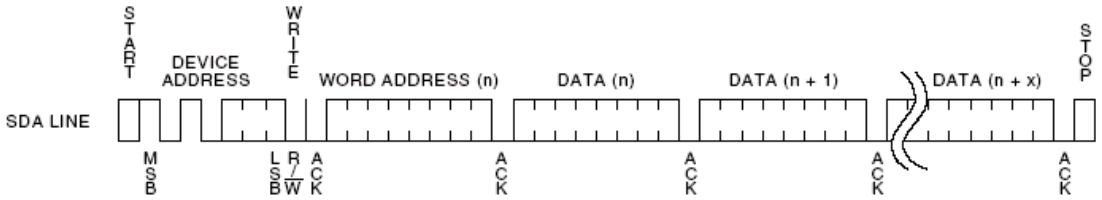
R/#W 表示的是对 24C02 而言数据传输的方向：R/#W＝1，对 24C02 读操作，数据由 24C02 向主机；R/#W＝0，对 24C02 写操作，数据由主机向 24C02 。

总上所述，图 4-4-1 中 24C02 的读命令是 0A5H，写命令是 0A4H。

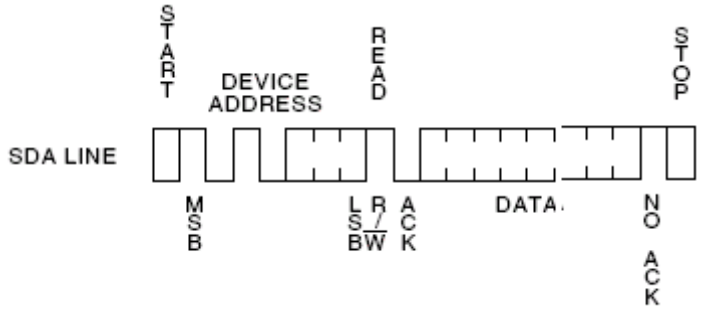
4、AT24C02 字节操作序列：



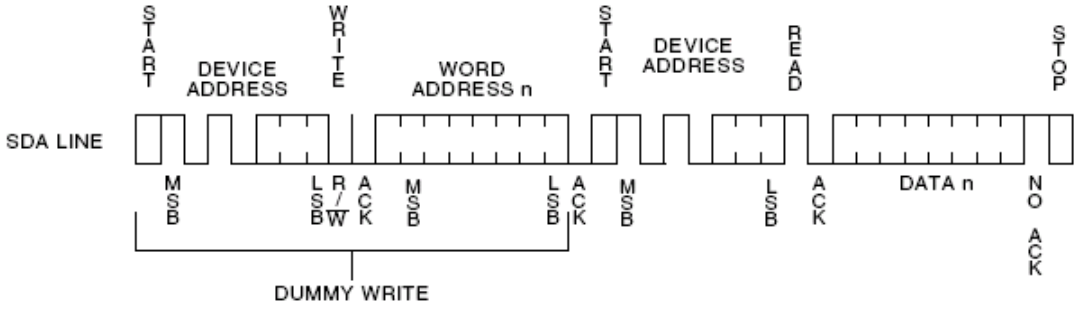
AT24C02 字节写



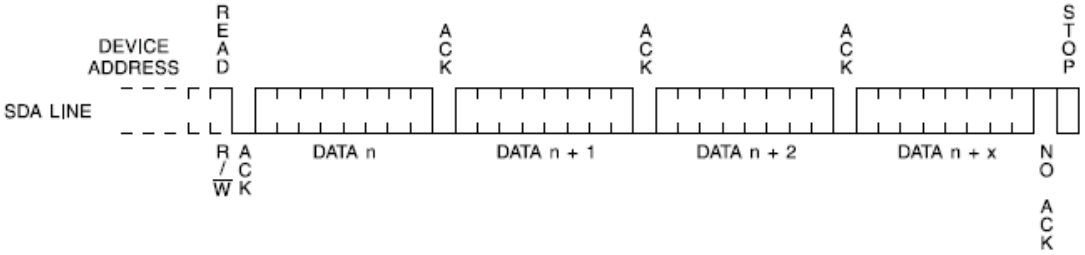
AT24C02 页写



AT24C02 当前地址读



AT24C02 任意地址读



AT24C02 连续地址读

一、实验目的

1、学习串行 EEPROM 的读写方法

2、了解 I^2 总线的控制方法

二、实验内容

向串行 EEPROM 的 10H 地址起依次写入 01H、02H、04H、08H、10H、20H、40H、80H。再从串行 EEPROM 的 10H 地址依次读出数据并送 P1 口显示

1、实验接线

P1.7~P1.0/C51　　接　　L7~L0/发光二极管

P2.6/C51　　接　　SDA/I^2 总线实验区

P2.7/C51　　接　　SCL/I^2 总线实验区

2、实验现象

LED 发光二极依次从右到左依次亮灭

三、参考程序

```
SDA    EQU     P2.6              ;I2C 数据线
SCL    EQU     P2.7              ;I2C 时钟线
EPMW   EQU     0A2H              ;AT24C02 写命令,地址为 A2H,最低位为 0 表示写
EPMR   EQU     0A3H              ;AT24C02 读命令,地址为 A3H,最低位为 1 表示
BUFRD  EQU     020H              ;从 AT24C02 读出的数据存储缓冲区
       ORG     0000H
       LJMP    MAIN
       ORG     0100H
MAIN:  LCALL   STOEPM            ;把 TAB 表 8 个字节连续存储到 EEPROM(AT2402)
       LCALL   DELY0             ;两次读、写操作之间需延时≤10mS,此处延时 10mS
       LCALL   RFROEPM    ;从 EEPROM(AT2402 里读出 8 个字节并存储到 BUFRD 缓冲区
       MOV     R4,#8
       MOV     R0,#BUFRD
LOOP:  MOV     A,@R0
       INC     R0
       MOV     P1,A
       LCALL   DELAY
       LCALL   DELAY
       DJNZ    R4,LOOP
       AJMP    $
TAB:   DB      01H,02H,04H,08H,10H,20H,40H,80H
       ;向 AT2402 写 8 个字节子程序
       ;选用 1 区的 R0--R7,以免与主程序冲突
       ;写的 8 个字节存放在 TAB 表中
STOEPM:
       LCALL   STAR              ;发出启动传输命令
       MOV     A,#EPMW           ;向 AT2402 发出写命令
       LCALL   WRBYT
       LCALL   CACK              ;调用应答子程序
       JB      F0,STOEPM    ;如果错误应答，转 STOEPM 重新开始
       MOV     A,#10H       ;向 AT2402 传送 WORD ADDRESS 00H,从地址 0 开始写
       LCALL   WRBYT
```

```
        LCALL   CACK            ;调用应答子程序
        JB      F0,STOEPM       ;如果错误应答，转 STOEPM 重新开始
        MOV     DPTR,#TAB       ;将 TAB 表的首地址取到 DPTR
        MOV     R6,#0                   ;TAB 中每个字节的偏移量 0 送 R6
        MOV     R7,#8                   ;AT2402 中的一页的字节数,R7 作为计数器使用
SEND:   MOV     A,R6                    ;从 TAB 表中取出 1 个字节送 AT2402
        MOVC    A,@A+DPTR
        INC     R6                      ;TAB 表的偏移量加 1
        LCALL   WRBYT
        LCALL   CACK                    ;调用应答子程序
        JB      F0,STOEPM       ;如果错误应答，转 STOEPM 重新开始
        DJNZ    R7,SEND
        LCALL   STOP                    ;传输结束，发出停止命令
        RET
        ;从 AT2402 读出 8 个字节子程序
        ;选用 1 区的 R0--R7,以免与主程序冲突
        ;读出的 8 个字节放在 BUFRD 缓冲区中
RFROEPM:
        LCALL   STAR                    ;发出启动传输命令
        MOV     A,#EPMW                 ;向 AT2402 发出写命令
        LCALL   WRBYT
        LCALL   CACK                    ;调用应答子程序
        JB      F0,RFROEPM              ;如果错误应答，转 RFROEPM 重新开始
        MOV     A,#10H          ;向 AT2402 传送 WORD ADDRESS 00H,从地址 0 开始写
        LCALL   WRBYT
        LCALL   CACK                    ;调用应答子程序
        JB      F0,RFROEPM              ;如果错误应答，转 RFROEPM 重新开始
        LCALL   STAR
        MOV     A,#EPMR                 ;向 AT2402 发出读命令
        LCALL   WRBYT
        LCALL   CACK                    ;调用应答子程序
        JB      F0,RFROEPM          ;如果错误应答，转 RFROEPM 重新开始
        MOV     R3,#8               ;读 8 个字节的计数器使用
RDN:    MOV     R1,#BUFRD           ;BUFRD 缓冲区首地址送 R1
RDN1:   LCALL   RDBYT                   ;从 AT2402 中读出 1 个字节
        MOV     @R1,A                   ;将读出的字节存入 BUFRD 缓冲区
        DJNZ    R3,ACK              ;如果 8 个字节没读完,转 ACK
        LCALL   MNACK                   ;8 个字节读完,接收结束，发送非应答信号
```

```
        LCALL   STOP            ;发送停止传输信号
        RET
ACK:    LCALL   MACK            ;发送应答信号
        INC     R1              ;缓冲区地址加 1
        SJMP    RDN1
        ;向 SDA 线上发送一个数据字节子程序，使用了 R0
        ;选用 1 区的 R0--R7
WRBYT:  MOV     R0,#08H         ;8 位数据长度送 R0 中
WLP:    RLC     A               ;发送数据左移，使发送位送入进位 C 中
        JC      WRE1            ;判断发送“1”还是“0”，发送“1”转 WRE1
        SJMP    WRE0            ;发送“0”转 WRE0
WLP1:   DJNZ    R0,WLP          ;8 位是否发送完，未完转 WLP
        RET                     ;8 位发送完，返回
        ;发送“1”程序段
WRE1:   SETB    SDA             ;置数据信号 SDA 为高
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        CLR     SDA             ;使数据信号 SDA 为低
        SJMP    WLP1
        ;发送“0”程序
WRE0:   CLR     SDA             ;使数据信号 SDA 为低
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        SJMP    WLP1
        ;从 SDA 线上读取一个数据字节子程序,放入 R2 中
        ;使用 1 区 R0--R7
RDBYT:
        MOV     R0,#08H         ;8 位数据长度送入 R0
RLP:    SETB    SDA             ;置 SDA 为输入方式
        SETB    SCL             ;置时钟信号 SCL 为高,使 SDA 上数据有效
        CLR     C               ;默认读数为 0，置进位 C=0
        JNB     SDA,RD0         ;读数为 0，转 RD0
        SETB    C               ;读数为 1，置进位 C=1
RD0:    MOV     A,R2            ;将读到的数（在进位 C 中），通过循环左移放到 R2 中
```

```
        RLC     A
        MOV     R2,A
        CLR     SCL
        DJNZ    R0,RLP          ;没有读完 8 位数据转 RLP
        RET                     ;返回
        ;START 信号子程序
STAR:   SETB    SCL             ;置时钟信号 SCL 为高
        SETB    SDA             ;置数据信号 SDA 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SDA             ;使数据信号 SDA 为低
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        RET
        ;主机发送应答位子程序
MACK:   CLR     SDA             ;使数据信号 SDA 为低
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        SETB    SDA             ;置数据信号 SDA 为高
        RET
        ;主机发送非应答位子程序
MNACK:  SETB    SDA             ;置数据信号 SDA 为高
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        CLR     SDA             ;使数据信号 SDA 为低
        RET
        ;应答位检测子程序，F0=0 表示收到了正常应答
        ;F0=1 表示收到非正常应答
CACK:   SETB    SDA             ;置数据信号 SDA 为高，以便接收数据

        SETB    SCL             ;置时钟信号 SCL 为高
        CLR     F0              ;默认为正常应答
        JNB     SDA,CEND        ;正常应答转 CEND
```

```
        SETB    F0              ;置错误应答标志
CEND:   CLR     SCL             ;使时钟信号 SCL 为低
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        RET
        ;STOP 信号子程序
STOP:   CLR     SDA             ;使数据信号 SDA 为低
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        SETB    SDA             ;置数据信号 SDA 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        RET
DELAY:  MOV     R5,#100         ;延时 1 秒子程序，使用参数 R7、R6、R5。
DELY0:  MOV     R7,#10          ;延时 10mS
DELY1:  MOV     R6,#250         ;延时 1mS
        DJNZ    R6,$
        DJNZ    R7,DELY1
        DJNZ    R5,DELY0
        RET
        END
```

实验十九 PCF8583 电子日历与时钟实验

PCF8583 时钟芯片介绍

1、PCF8583 的封装及管脚描述：

SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

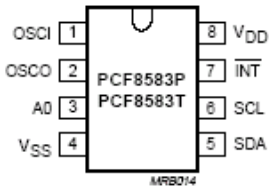
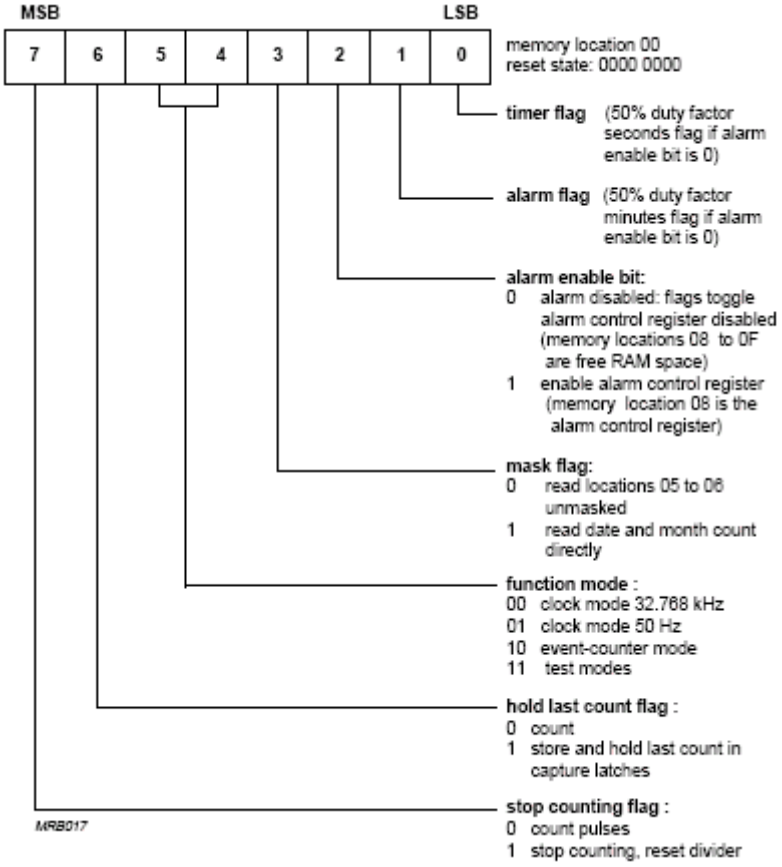


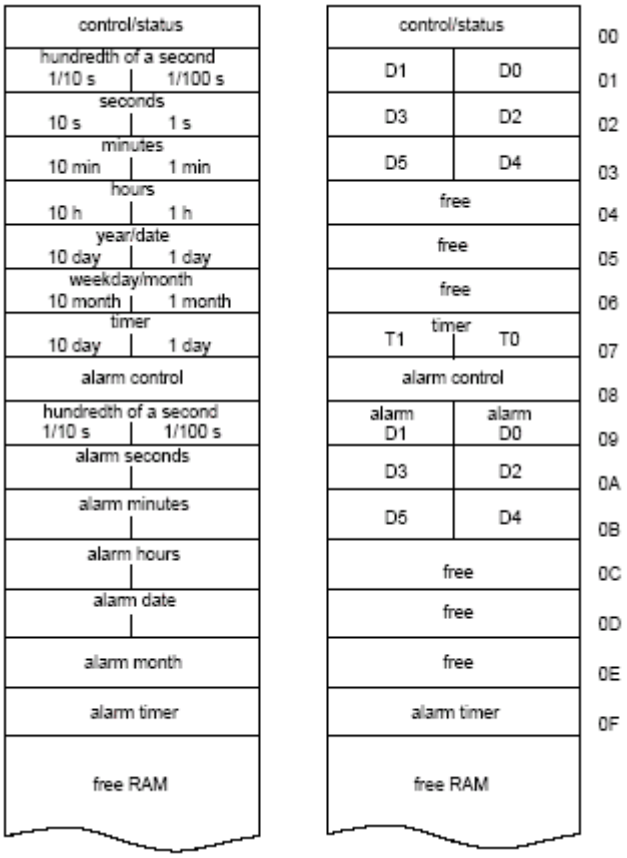
Fig.2 Pinning diagram.

PF8583 封装及管脚说明

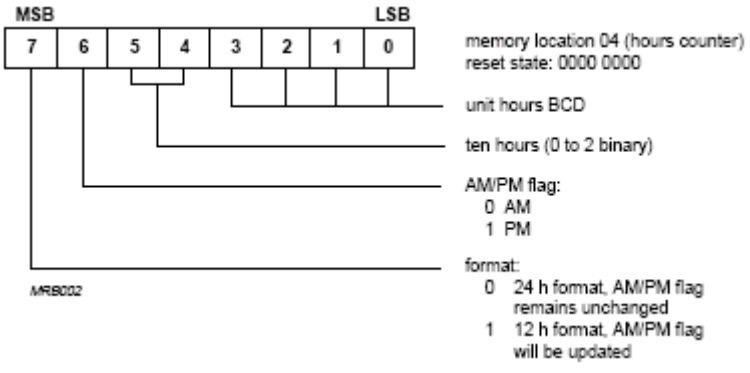
2、PCF8583 的功能描述：



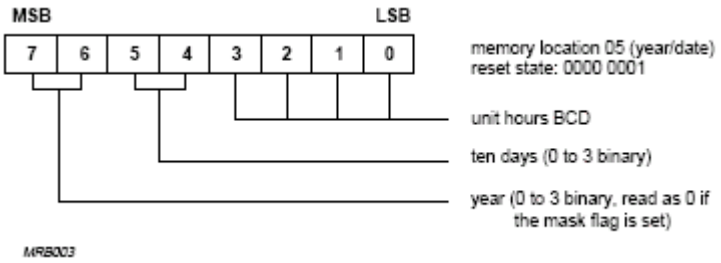
PCF8583 的控制/状态寄存器



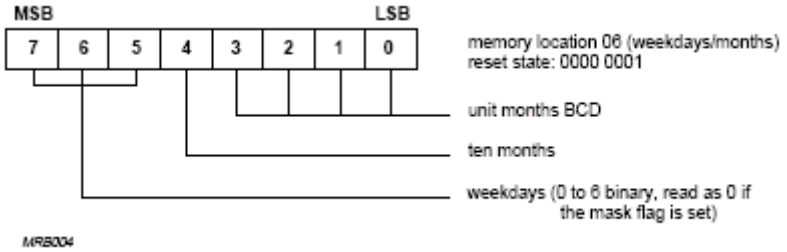
PCF8583 的寄存器存储结构



时指针格式



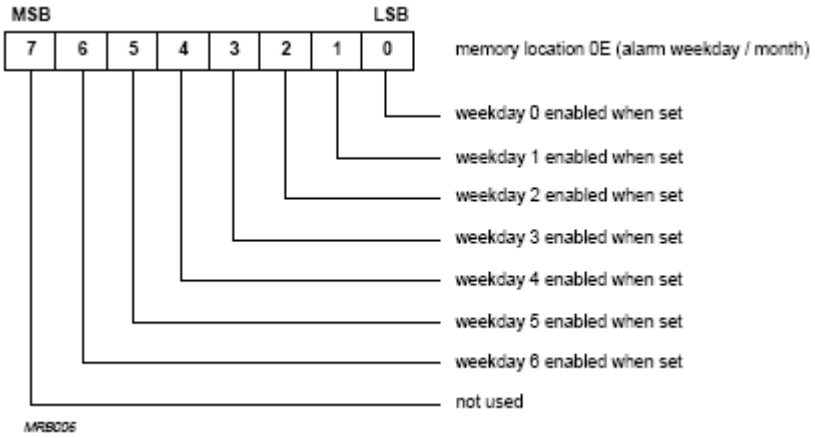
年/日指针格式



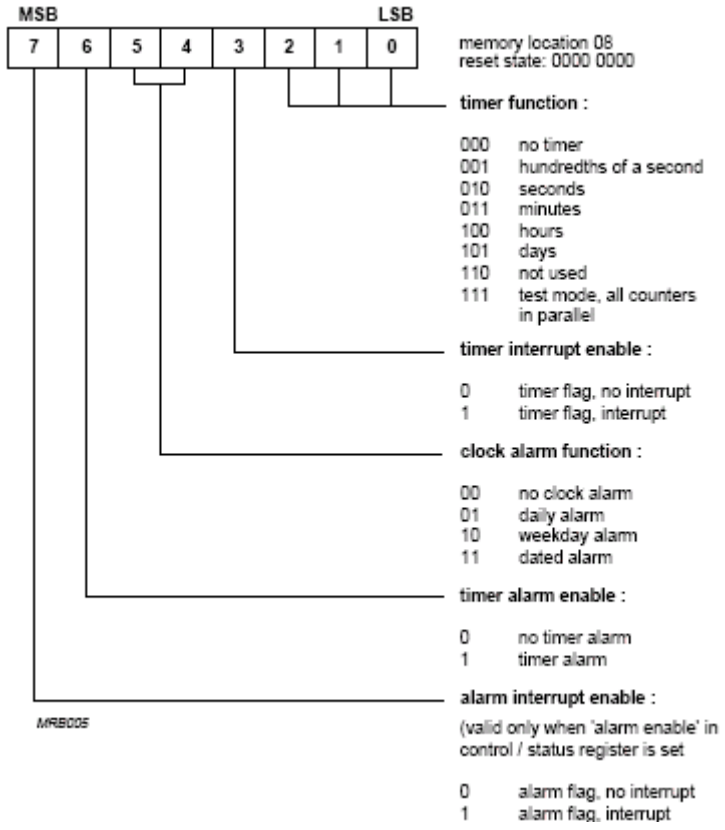
周/月指针格式

UNIT	COUNTING CYCLE	CARRY TO NEXT UNIT	CONTENTS OF THE MONTH COUNTER
hundredths of a second	00 to 99	99 to 00	
seconds	00 to 59	59 to 00	
minutes	00 to 59	59 to 00	
hours (24 h)	00 to 23	23 to 00	
hours (12 h)	12 AM; 01 AM to 11 AM; 12 PM; 01 PM to 11 PM	 11 PM to 12 AM	
date	01 to 31 01 to 30 01 to 29 01 to 28	31 to 01 30 to 01 29 to 01 28 to 01	1, 3, 5, 7, 8, 10, 12 4, 6, 9, 11 2, year = 0 2, year = 1, 2, 3
months	01 to 12	12 to 01	
year	0 to 3		
weekdays	0 to 6	6 to 0	
timer	00 to 99	no carry	

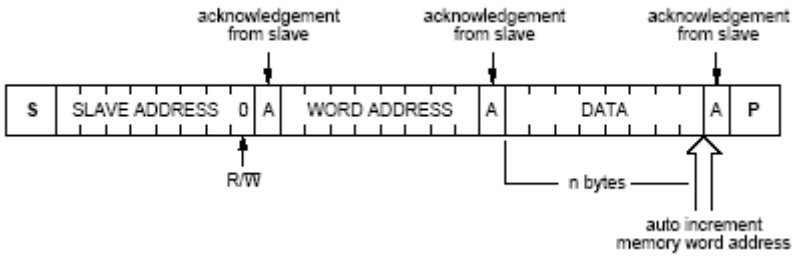
PCF8583 指针的循环周期表



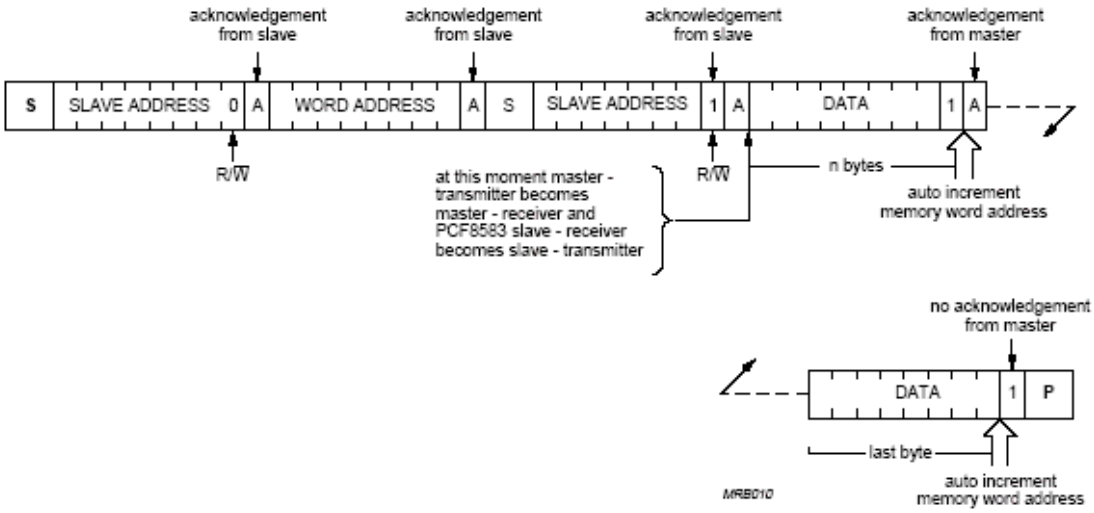
报警星期选择的寄存器



报警控制寄存器（时钟模式）



日历时钟写模式（主机发送/从机接收/写数据）



日历时钟读模式（主机写入数据地址/读数据）

详细资料请查片光盘中的芯片资料介绍

一、实验目的

学习 PCF8583 电子日历时钟芯片的功能及原理。

二、实验内容

1、普通的电子日历与时钟的实验。

2、有报警功能的电子日历与时钟（定时报警）的实验。

三、实验步骤

四、参考程序

普通的电子日历与时钟实验。实验结果：在 4 个串行数码管上显示分和秒。从 12 分 34 秒开始。(用户可自行在 LCD 上显示完整的日历和报警功能)

```
SDA       EQU     P1.3              ;I2C 数据线
SCL       EQU     P1.4              ;I2C 时钟线
SLAW2     EQU     0A0H              ;PCF8583 写命令,地址 A2H,最低位为 0 表示写
SLAR2     EQU     0A1H              ;PCF8583 读命令,地址 A2H,最低位为 1 表示读
WADDR     EQU     00H               ;PCF8583 写入数据首地址
CONTRL    EQU     08H               ;PCF8583 控制字节,不允许报警,直接读月、日
HSECOND EQU       00H               ;百分之一秒初始值
SECOND    EQU     34H               ;秒初始值
MINUTE    EQU     12H               ;分初始值
HOUR      EQU     15H               ;时初始值
DATE      EQU     06H               ;日初始值
MONTH     EQU     06H               ;月初始值

SLA       EQU     30H               ;
NUMBYT    EQU     31H               ;
BUFTC     EQU     32H               ;
```

```
BUFTD     EQU     34H       ;取数暂共 8 个字节依次为  日十位、日个位、小时十位、小时个数
                            ;分十位、分个位、秒十位、秒个位.
ADRBUF    EQU     40H                    ;两个缓冲区首地址存储单元初始化时使用
BUF       EQU     41H                    ;临时缓冲区

          ORG     0000H
          LJMP      MAIN
          ORG       0100H
MAIN:     LCALL     INITIAL              ;PCF8583 初始化子程序,用到参数 F0,R0,R1,R2,R3
          MOV       R1,#BUFTC             ;连续读秒分时日放进 BUFTD 子程序
LOOP:     MOV       R3,#2                ;
          LCALL     STORE                ;从 PCF8583 中读出时间值,并放入 BUFTD 中
          LCALL   DISP
          LCALL   DELAY
          SJMP      LOOP

DISP:     MOV     R7,#4           ;显示时从分的十位开始
          MOV     R0,#BUFTD+4     ;串行数码只有 4 位，只显示分和秒，
DISP1:    MOV     A,@R0
          MOV     SBUF,A
          JNB     TI,$
          INC       R0
          MOV     R6,#150
          DJNZ    R6,$
          DJNZ    R7,DISP1
          RET

DELAY:    MOV     R6,#200
DEL0:     MOV     R7,#255
          DJNZ    R7,$
          DJNZ    R6,DEL0
          RET
          ;PCF8583 设定初始值子程序
          ;设定初始值 1/100 秒、秒、分、时、日、月
          ;用到参数 F0,R0,R1,R2,R3
INITIAL:   LCALL     STAR                ;发出启动传输命令
          MOV       DPTR,#MEMORY        ;将 PCF8583 初始化表传送到 BUF 缓冲区
          MOV       R1,#BUF
```

```
        MOV     R2,#0
        MOV     R3,#8
INPUT:  MOV     A,R2
        MOVC    A,@A+DPTR
        MOV     @R1,A
        INC     R1
        INC     R2
        DJNZ    R3,INPUT
        MOV     NUMBYT,#8  ;将 PCF8583 初始化表写入 PCF8583
        MOV     SLA,#SLAW2
        MOV     ADRBUF,#BUF
        LCALL   WRNBYT
        RET
        ;PCF8583 存储单元初始化表
MEMORY: DB      WADDR,CONTRL,HSECOND,SECOND,MINUTE,HOUR,DATE,MONTH

;模拟 I2C 总线发送 N 个字节数据，N 存放在 NUMBYT 中
        ;数据区的第一个字节为从器件中一个起始的地址
        ;数据区的第二个字节是控制寄存器字节
WRNBYT: LCALL   STAR                ;发出启动传输命令
        MOV     A,SLA               ;发出发送数据命令
        LCALL   WRBYT
        LCALL   CACK                ;调用应答子程序
        JB      F0,WRNBYT  ;如果错误应答，转 WRNBYT 重新开始
        MOV     R1,ADRBUF  ;放发送数据区的首地址在 R1 中

WRDAT:  MOV     A,@R1               ;从数据区中取出 1 个字节发送
        LCALL   WRBYT
        LCALL   CACK                ;调用应答子程序
        JB      F0,WRNBYT  ;如果错误应答，转 WRNBYT 重新开始
        INC     R1                  ;数据区地址加 1
        NOP
        NOP
        NOP
        NOP
        DJNZ    NUMBYT,WRDAT        ;发送没有结束转 WRDAT 继续发送
        LCALL   STOP                ;发送结束，发出停止命令
        RET
```

```
        ;从 PCF8583 读秒分时日并存入相应的缓冲区子程序
        ;破坏参数 F0,R0,R1,R2,R3
        ;入口:R3=2
        ;读取数据从秒的个位开始读取，为方便显示因此暂存从#BUFTD+7 开始减 1
STORE:  MOV     R3,#2
        LCALL   STAR                ;发出启动传输命令
        MOV     A,#SLAW2    ;发 PCF8583 写命令
        LCALL   WRBYT
        LCALL   CACK                ;调用应答子程序
        JB      F0,STORE    ;如果错误应答，转 STORE 重新开始
        MOV     A,#02H              ;发送读 PCF8583 数据的首地址 02H
        LCALL   WRBYT
        LCALL   CACK                ;调用应答子程序
        JB      F0,STORE    ;如果错误应答，转 STORE 重新开始
        NOP
        NOP
        LCALL   STAR
        MOV     A,#SLAR2    ;发送 PCF8583 读命令
        LCALL   WRBYT
        LCALL   CACK                ;调用应答子程序
        JB      F0,STORE    ;如果错误应答，转 STORE 重新开始
        MOV     R1,#BUFTD+7         ;将 BUFTD 第四个字节的地址送 R1
REWR:   LCALL   RDBYT               ;从 PCF8583 读 1 个字节,第一次为秒,第二次为
        LCALL   MACK                ;发送应答信号
        MOV     B,A
        ANL     A,#0FH              ;分离出读出字节的低 4 位存入缓冲区
        LCALL   FTAB
        MOV     A,B
        ANL     A,#0F0H             ;分离出读出字节的高 4 位存入缓冲区
        SWAP    A
        LCALL   FTAB
        LCALL   RDBYT               ;从 PCF8583 读 1 个字节,第一次为分,第二次为时
        LCALL   MACK                ;发送应答信号
        MOV     B,A
        ANL     A,#0FH              ;分离出读出字节的低 4 位存入缓冲区
        LCALL   FTAB
        MOV     A,B
        ANL     A,#0F0H             ;分离出读出字节的高 4 位存入缓冲区
```

```
        SWAP    A
        LCALL   FTAB
        NOP
        NOP
        NOP
        DJNZ    R3,REWR
        LCALL   MNACK           ;接收结束，发送非应答信号
        LCALL   STOP            ;发送停止信号
        RET

        ;将数字转换为字形码并存入缓冲区子程序
        ;入口:A 为数字,R1 为缓冲区指针
        ;出口:缓冲区指针减 1
FTAB:   MOV     DPTR,#TAB  ;从字形码表 TAB 中取出相应的字形码
        MOVC    A,@A+DPTR
        MOV     @R1,A           ;将字形码存入缓冲区
        DEC     R1              ;缓冲区指针减 1
        RET
;-------------------------------------------------------------------------------
;           模拟串行数据线发送和读取数据子程序
;-------------------------------------------------------------------------------
        ;向 SDA 线上发送一个数据字节子程序，使用了 R0
WRBYT:  MOV     R0,#08H         ;8 位数据长度送 R0 中
WLP:    RLC     A               ;发送数据左移，使发送位送入进位 C 中
        JC      WRE1            ;判断发送“1”还是“0”，发送“1”转 WRE1
        SJMP    WRE0            ;发送“0”转 WRE0
WLP1:   DJNZ    R0,WLP          ;8 位是否发送完，未完转 WLP
        RET                     ;8 位发送完，返回
          ;发送“1”程序段
WRE1:   SETB    SDA             ;置数据信号 SDA 为高
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP
        NOP
        NOP
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        NOP
        NOP
```

```
        CLR     SDA             ;使数据信号 SDA 为低
        SJMP    WLP1
            ;发送“0”程序段
WRE0:   CLR     SDA             ;使数据信号 SDA 为低
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        NOP
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        NOP
        NOP
        SJMP    WLP1

        ;从 SDA 线上读取一个数据字节子程序,放入 R2 中
RDBYT:  MOV     R0,#08H         ;8 位数据长度送入 R0
RLP:    SETB    SDA             ;置 SDA 为输入方式
        SETB    SCL             ;置时钟信号 SCL 为高,使 SDA 上数据有效
        CLR     C               ;默认读数为 0，置进位 C=0
        JNB     SDA,RD0         ;读数为 0，转 RD0
        SETB    C               ;读数为 1，置进位 C=1
RD0:    MOV     A,R2            ;将读到的数（在进位 C 中），通过循环左移放到
R2 中
        RLC     A
        MOV     R2,A
        NOP
        NOP
        NOP
        NOP
        CLR     SCL
        DJNZ    R0,RLP          ;没有读完 8 位数据转 RLP
        RET                     ;读完 8 位数据返回

        ;STAR 信号子程序
STAR:   SETB    SCL             ;置时钟信号 SCL 为高
        SETB    SDA             ;置数据信号 SDA 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
```

```
        CLR     SDA             ;使数据信号 SDA 为低
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        RET
        ;主机发送应答位子程序
MACK:   CLR     SDA             ;使数据信号 SDA 为低
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        SETB    SDA             ;置数据信号 SDA 为高
        RET
        ;主机发送非应答位子程序
MNACK:  SETB    SDA             ;置数据信号 SDA 为高
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        CLR     SDA             ;使数据信号 SDA 为低
        RET
        ;应答位检测子程序，F0=0 表示收到了正常应答
        ;F0=1 表示收到非正常应答
CACK:   SETB    SDA             ;置数据信号 SDA 为高，以便接收数据
        SETB    SCL             ;置时钟信号 SCL 为高
        CLR     F0              ;默认为正常应答
        NOP
        JNB     SDA,CEND    ;正常应答转 CEND
        SETB    F0              ;置错误应答标志
CEND:   CLR     SCL             ;使时钟信号 SCL 为低
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        RET
        ;STOP 信号子程序
STOP:   CLR     SDA             ;使数据信号 SDA 为低
        SETB    SCL             ;置时钟信号 SCL 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
```

```
        SETB    SDA             ;置数据信号 SDA 为高
        NOP                     ;NOP 为满足 I2C 协议所需的等待时间
        NOP
        CLR     SCL             ;使时钟信号 SCL 为低
        RET

TAB:    DB 03H,9FH,25H,0DH,99H,49H,41H,1FH,01H,09H;0~~9 的段码表
        END
```

实验二十　串行 EEPROM 及看门狗实验

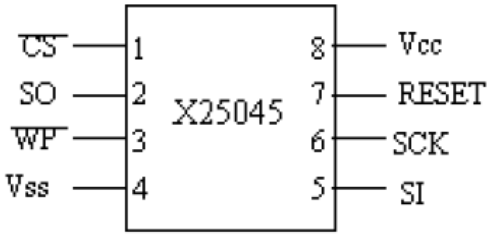
一、实验目的

1、 学习串行 EEPROM（X5045）的使用方法

2、 学习串行看门狗的使用方法

二、实验原理

X25045 是美国Xicor 公司的生产的集成电路，它将E^2PROM、看门狗定时器、电压监控三种功能组合在单个芯片之内。因其体积小、占用I/O 口少等优点已被广泛应用于工业控制、仪器仪表等领域，是一种理想的单片机外围芯片。X25045 引脚如图所示。引脚功能如下：



CS：片选择输入，

SO：串行输出，数据由此引脚逐位输出。

SI：串行输入，数据或命令由此引脚逐位写入X25045。

SCK：串行时钟输入，其上升沿将数据或命令写入，下降沿将数据输出。

WP：写保护输入，当它低电平时， 写操作被禁止。

Vss：地；

Vcc：电源电压；

RESET：复位输出。

X25045 读写操作之前，需要先向它发指令，指令名及指令格式如下表所示。

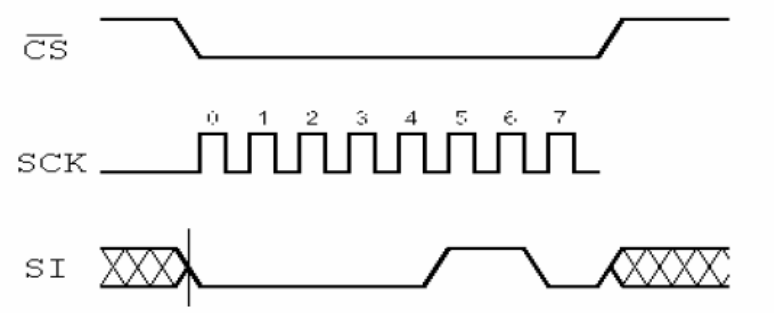
X25045 读写操作之前，需要先向它发指令，指令名及指令格式如下表所示。

指令名	指令格式	操作
WREN	00000110	设置写使能锁存器 （允许写操作）
WRDI	00000100	复位写使能锁存器 （禁止写操作）
RDSR	00000101	读状态寄存器
WRSR	00000001	写状态寄存器
READ	0000$A_8$011	把开始于所选地址的 存储器中的数据读出
WRITE	0000$A_8$010	把数据写入开始于 所选地址的存储器

1、对X25045 的E^2PROM 写操作

X25045 中有512×8 的串行E^2PROM，通过X25045 的*CS* 、SCK、SI 等引脚控制对X25045 写。写操作过程如下：

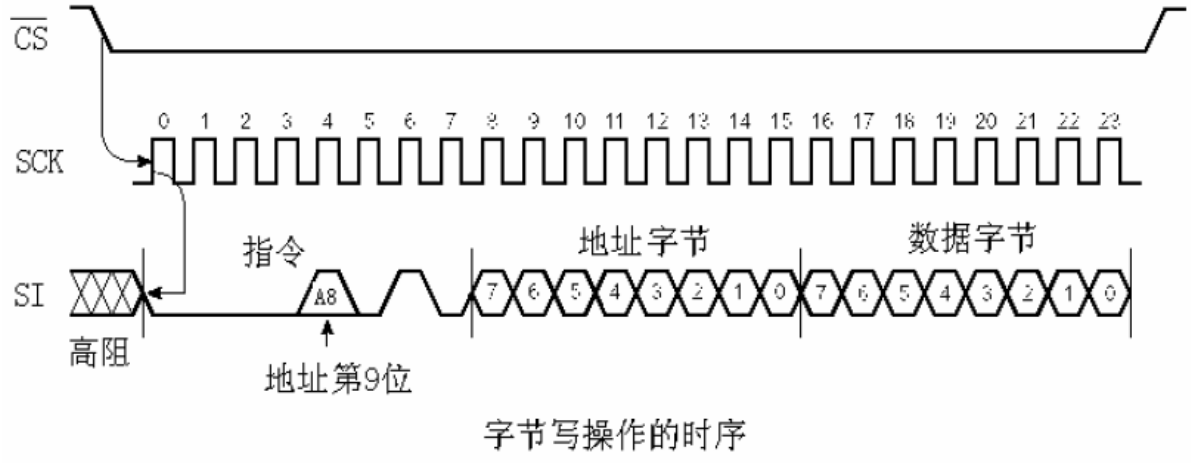
① 首先向 X5045 发设置写使能锁存器指令：00000110，时序如下：



写使能锁存器时序

② 然后向X5045 发送写操作指令：0000$A_8$010，其中A8 为地址的第9 位。

③ 向 X5045 发送 E^2PROM 的地址（低 8 位）及数据，在写入数据之后将 *CS* 置高。时序如下：



字节写操作的时序

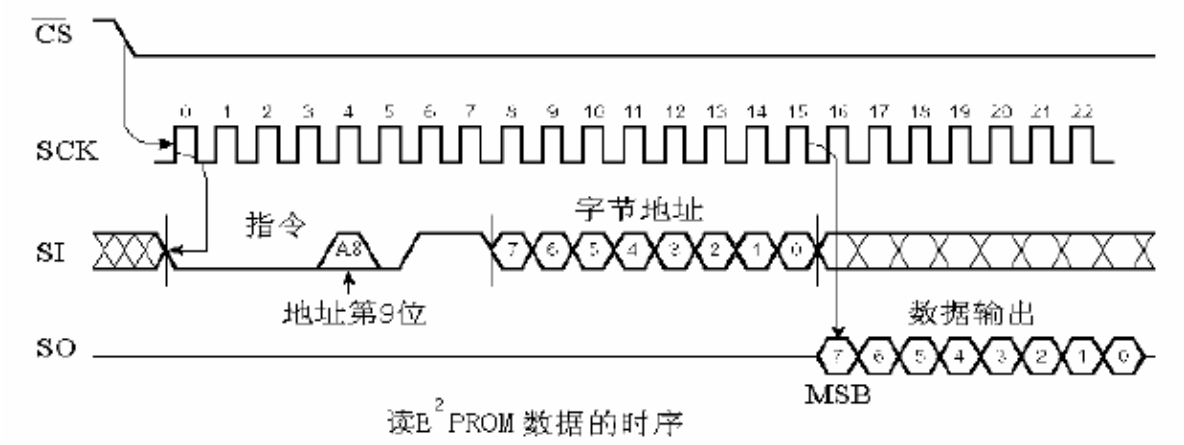
④ 将*CS* 置高后，延时2ms，数据被写入E^2PROM 中。

2、X25045 的E^2PROM 读操作

① 向X5045 发送读操作指令：0000$A_8$010，其中A8 为地址的第9 位。

② 向X5045 发送E^2PROM 的地址（低8 位）。

③ 从 X5045 的 SO 逐位接收 X25045 传出的数据。对 X25045 的 E^2PROM 读操作时序如下：



读E²PROM 数据的时序

① X25045 看门狗的使用方法

X25045 中包含有一个看门狗定时器，在看门狗定时器预置的时间内没有总线活动， X25045 将从RESET 输出一个复位信号。预置时间的方法是向状态寄存器WD1、WD0 位写入数据。

WD1、WD0=00， 1.4 秒；

WD1、WD0=01， 0.6 秒；

WD1、WD0=10， 0.2 秒；

WD1、WD0=11， 禁止。

X25045 状态寄存器及其它各位的意义如下：

D7	D6	D5	D4	D3	D2	D1	D0
X	X	WD1	WD0	BL1	BL0	WEL	WIP

WIP 位表示X25045 是否忙于向E_2PROM 写数据。该位是只读位，WIP 为0 表示没有写操作在进行，可向E_2PROM 写数据；WIP 为1 时表示正在进行写操作，此时不能向E_2PROM 写数据。

WEL 位表示写使能锁存器的状态。该位是只读位，由WRDI 指令复位，写使能锁存器被复位时，向E_2PROM 写操作被禁止。

BL1、BL0 位用来确定E_2PROM 的块保护范围。

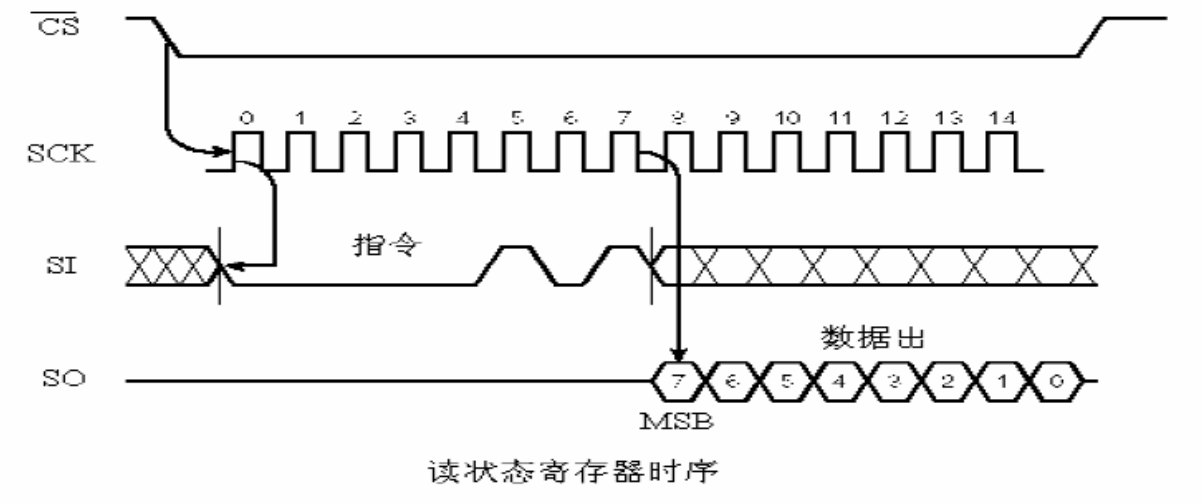
BL1、BL0 =00， 无写保护

BL1、BL0 =01， 180H~1FFH

BL1、BL0 =10， 100H~1FFH

BL1、BL0 =11， 000H~1FFH

X25045 状态寄存器的写时序和读时序如下。



读状态寄存器时序

三、实验内容

1、向看门狗写入数据，使看门狗定时复位。

2、接线：

CS/X5045	接	P2.4/C51
SCK/X5045	接	P2.5/C51
SI/X5045	接	P2.6/C51
SO/X5045	接	P2.7/C51
/WP/X5045	接	VCC

3、实验现象

看门狗的REST定时输出一个高电平

四、参考程序

```
        CS      EQU     P2.4            ;片选信号由P2.2产生
        SCK     EQU     P2.5            ;时钟信号由P2.3产生
        SI      EQU     P2.6            ;SO由P2.1产生
        SO      EQU     P2.7            ;SI由P2.0产生
        ADDR    EQU     07FH            ;使用X25045地址为7FH的单元
        DAT     EQU     0AAH            ;写入数据为0AAH
        ORG     0000H                   ;主程序
          AJMP    MAIN
          ORG     0100H
MAIN:   MOV     A,      #06H            ;发送写使能命令
        CLR     CS
        ACALL    TRAN                    ;调用发送子程序
        SETB    CS                      ;写使能命令发送结束
        MOV     A,      #01H            ;发送写状态字指令
        CLR     CS                      ;
        ACALL    TRAN                    ;
```

```
        MOV     A,      #00000000b          ;设置看门狗的超时周期为1.4秒
        ACALL    TRAN                       ;
        SETB    CS                          ;结束
        SJMP    $
TRAN:   MOV     R0,     #08H                ;发送一字节数据子程序
TRAN1:  RLC     A
        MOV     SI,     C                   ;将C中的数据输出到SI
        CLR     SCK                         ;SCK产生一个上跳变
        SETB    SCK
        DJNZ    R0,     TRAN1               ;8位未发送完,转移
        CLR     SI                          ;发送结束
        RET
        END
```

实验二十一　串行数码管显示

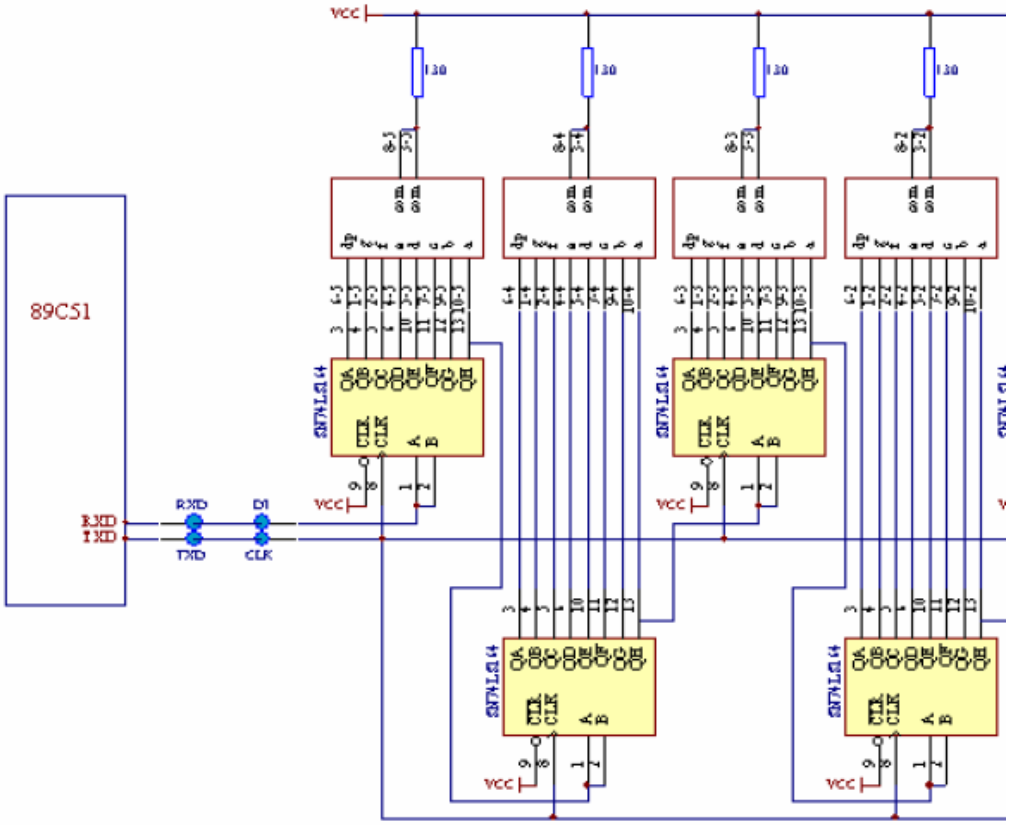
一、实验目的

1、掌握89C51 单片机与七段LED 显示器接口的一种方法。

2、掌握利用89C51 单片机串行口扩展并行输出口的方法。

二、实验原理

七段LED 显示器是单片机应用系统中最常用的输出设备。单片机与七段LED 显示器的接口方法很多。实验仪上设有一种在智能化仪表中常用的显示电路，原理如图所示。采用6 个共阳极的LED 显示器，通过6 个串联的串入/并出移位寄存器74HC164 直接驱动，该电路可以同89C51单片机的串行口（或2 根I/O 线）直接相连，通过串行口将显示段码逐位送出，即可显示。该电路的优点是占用单片机I/O 口线少，且软件设计简单。例如：欲在左起第1 个七段LED 上显示”0”，只要通过传送指令将“0”的段码“03H”通过串行口送出即可。



三、实验内容和步骤

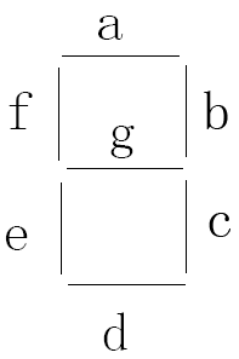
1、按图示电路将单片机与显示电路连接：RXD 连DAT；TXD 连CLK。

2、编程使七段LED 显示器上循环显示9~0。

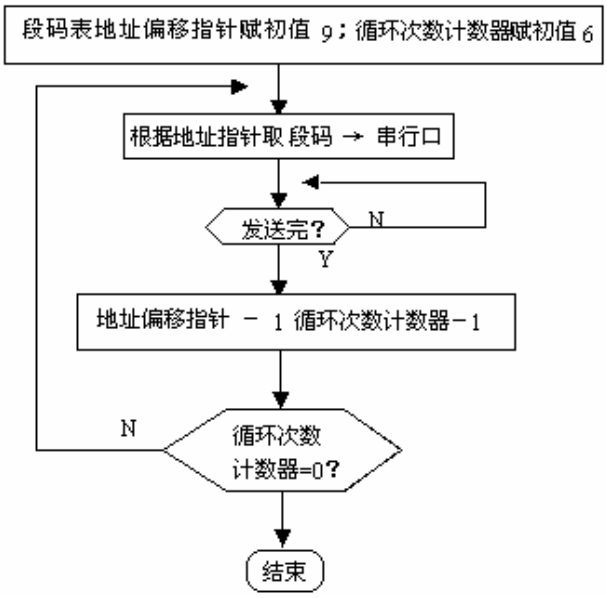
3、接线注意：若使用在线可编程芯片，RXD 连DAT；TXD 连CLK 须在编程后再接，否则不能在线可编程。

4、七段 LED 段码表如下：

字符	a	b	c	d	e	f	g	dp	段码
0	0	0	0	0	0	0	1	1	03
1	1	0	0	1	1	1	1	1	9F
2	0	0	1	0	0	1	0	1	25
3	0	0	0	0	1	1	0	1	0D
4	1	0	0	1	1	0	0	1	99
5	0	1	0	0	1	0	0	1	49
6	0	1	0	0	0	0	0	1	41
7	0	0	0	1	1	1	1	1	1F
8	0	0	0	0	0	0	0	1	01
9	0	0	0	0	1	0	0	1	09



四、程序参考流程图



五、实验参考程序：

```
            ORG 0000H
            AJMP  MAIN

            ORG 0100H
MAIN:       MOV R2, #09H  ;段码偏移指针值为 9
LOOP:       MOV A,R2
            MOV R0,A
            MOV R1, #04H  ;循环次数 4，显示 4 位数
START:      MOV DPTR, #TABLE
            MOV A, R0
            MOVC A, @A+DPTR
            MOV SBUF, A
            JNB TI, $
            DEC R0
```

```
        CJNE   R0,#0FFH,ST2
        MOV   R0,#09H
ST2:      DJNZ R1,START
        LCALL   DELAY
          DEC    R2
          CJNE   R2,#0FFH,NEXT
          MOV    R2,#09H
NEXT:     LJMP   LOOP
DELAY:    MOV    R5,#5
DEL0:     MOV    R6,#255
DEL1:     MOV    R7,#255
          DJNZ   R7,$
          DJNZ   R6,DEL1
         DJNZ   R5,DEL0
          RET
TABLE:    DB 03H,9FH,25H,0DH,99H,49H,41H,1FH,01H,09H;0~~9 的段码表
          END
```

实验二十二　BCD 码数管码显示实验

一、实验目的

1、掌握89C51 单片机与BCD码数码管显示接口的一种方法。

2、掌握利用89C51 单片机串行口扩展并行输出口的方法。

二、实验内容

1、实验接线

D / BCD 码数码管　　接　　P1.3

C / BCD 码数码管　　接　　P1.2

B / BCD 码数码管　　接　　P1.1

A / BCD 码数码管　　接　　P1.0

2、实验现象

BCD 码数码管循环显示 0~9

三、参考程序

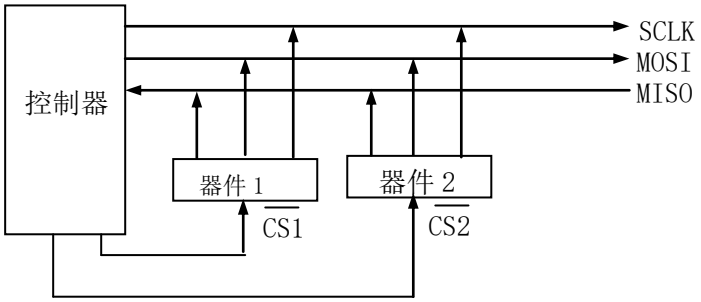
```
            ORG 0000H
            AJMP  MAIN

            ORG 0100H
MAIN:       MOV     R2, #00H        ;段码偏移指针值为 9
LOOP:       MOV     A,R2            ;输出 BCD 码数据
            MOV     P1,A
            LCALL   DELAY           ;延时
            INC     R2
            CJNE    R2,#0AH,LOOP    ;判断输出的数据是否大于 9
            MOV     R2,#00
            LJMP    LOOP

DELAY:      MOV   R5,#5
DEL0:       MOV   R6,#255
DEL1:       MOV   R7,#255
            DJNZ  R7,$
          DJNZ  R6,DEL1
          DJNZ  R5,DEL0
          RET
TABLE:      DB 03H,9FH,25H,0DH,99H,49H,41H,1FH,01H,09H;0~~9 的段码表
            END
```

实验二十三　语音芯片 ISD4002 录放音实验

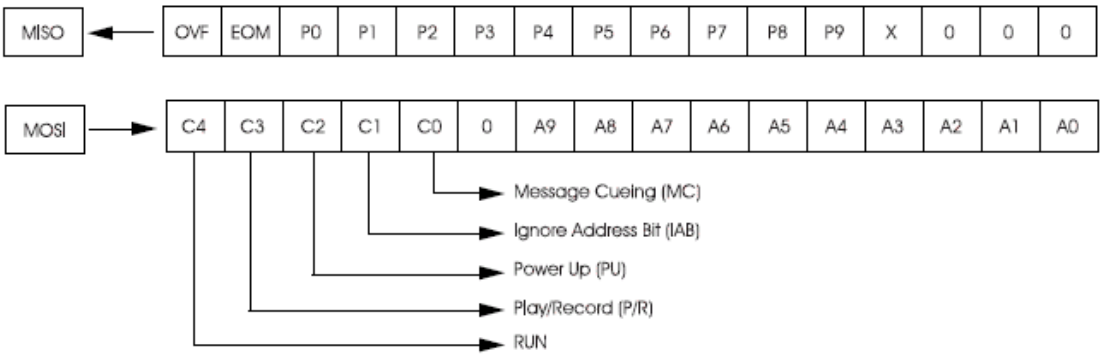
关于 SPI 总线接口与语音芯片的基础知识

SPI(Serial Periperal Interface)是 MOTOROLA 推出的同步串行扩展接口。该接口共使用四条信号线：主机输出片选信号线/SS，主机输出时钟信号线 SCLK，主机输出/从机输入的数据线 MOSI，主机输入/从机输出的数据线 MISO。串行时钟 SCLK 用于同步 MOSI 和 MISO 传输信号。SPI 串行扩展接口是全双工同步通信口。主机方式传送数据的最高速度达到 1.05Mb/s。

SPI 组成的系统如下图：



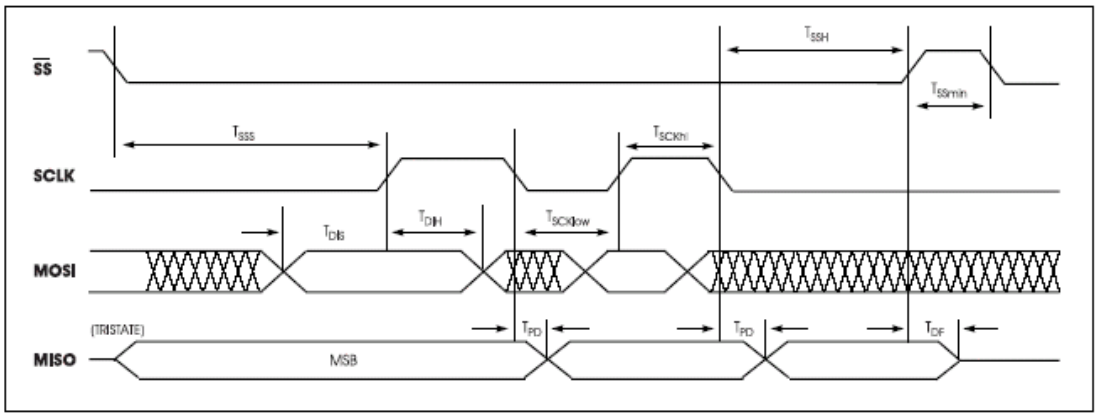
SPI 组成的系统图



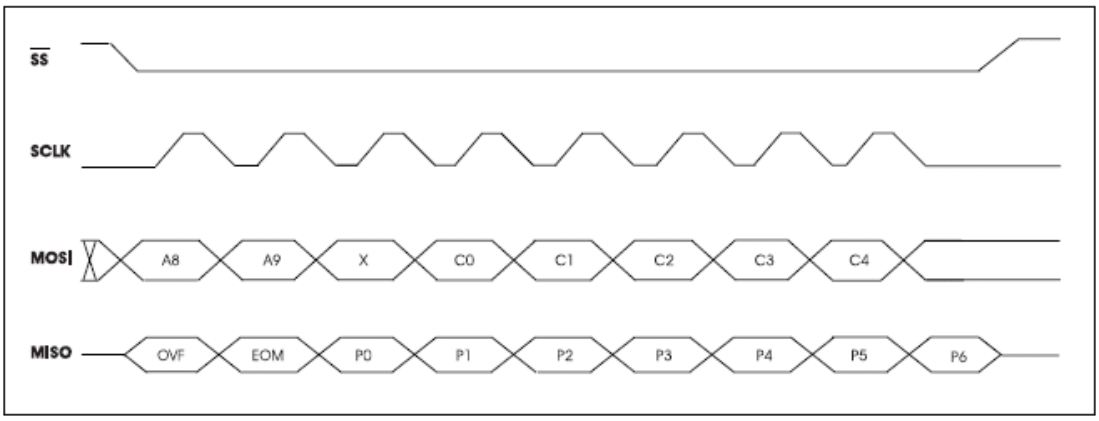
SPI　PORT

Control Register	Bit	Device Function	Control Register	Bit	Device Function
RUN		Enable or Disable an operation	PU		Master power control
=	1	Start	=	1	Power-Up
=	0	Stop	=	0	Power-Down
P/$\overline{R}$		Selects play or record operation	IAB[1]		Ignore address control bit
=	1	Play	=	1	Ignore input address register (A9–A0)
=	0	Record	=	0	Use the input address register contents for an operation (A9–A0)
MC		Enable or Disable Message Cueing	P9–P0		Output of the row pointer register
=	1	Enable Message Cueing	A9–A0		Input address register
=	0	Disable Message Cueing			

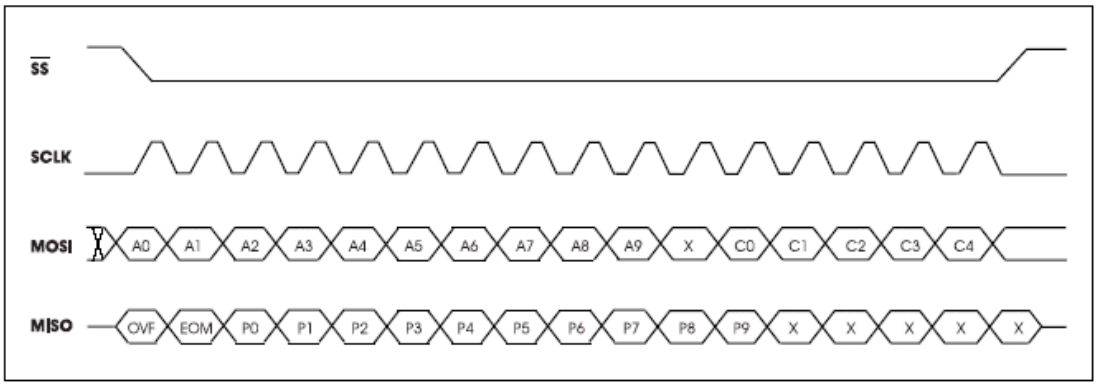
SPI 控制寄存器



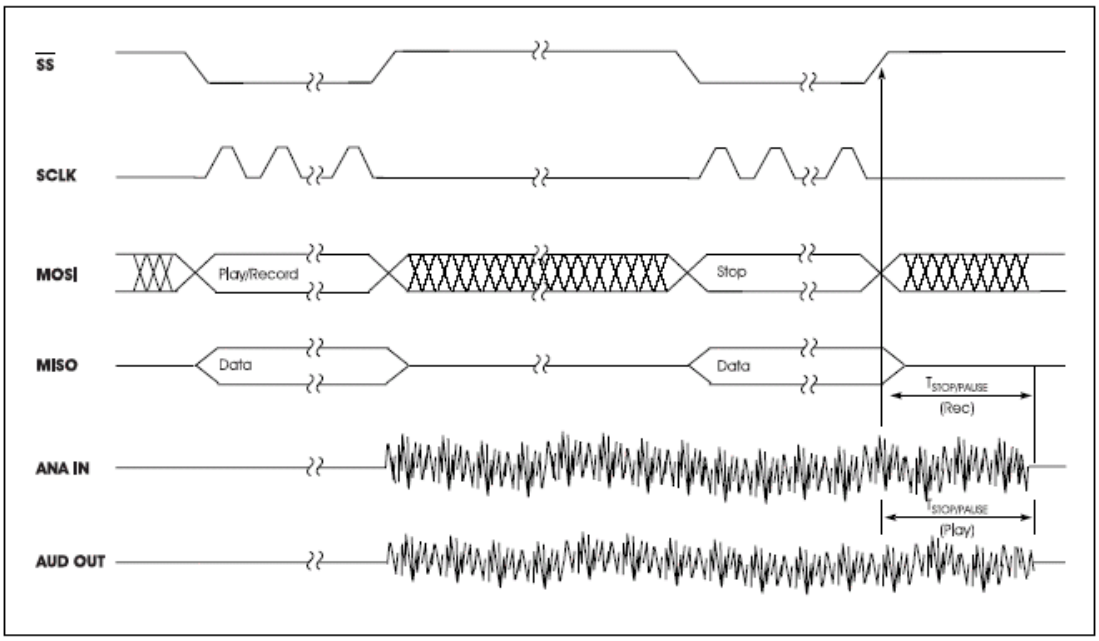
SPI 总线时序图



SPI 总线 8 位命令格式



SPI 总线 16 位命令格式



录 / 放和停止时序图

SPI 总线上可挂接多种具有 SPI 外围接口的器件。我们挑选一个具有广阔应用前景的器件 ISD4002 作为实验对象。作为典型实例，结合器件自身的接口电路加以详细讲解，有利于加深对 SPI 接口技术的理解，便于进一步开发利用多样化的外围接口的器件。

单片机通过 SPI 总线接口用特定的指令来控制语音芯片做录放音的动作，再把单片机和语音芯片嵌入到通信设备及智能仪器、治安报警、儿童玩具中去，就可做成语音播放的机器，应用范围很广。掌握了单片机控制语音芯片的技术，自己做个语音留言器或复读机就不困难。我们先来介绍关于语音录放电路的基础知识。

ISD 公司的专利技术成功实现了将模拟数据储存在半导体存储器中。这种突破性的 EEPROM 存储方法可以将模拟语音数据直接写入存储单元，不需要经过 A/D 或 D/A 转换。这种技术产生了两个效果：比同等的数字方式具有更大的集成度; 存储的模拟数据不丢失。

SPI 总线典型信号的模拟子程序：

```
;设置录音起始地址为 100H 的子程序 SETREC1
;设置录音起始地址为 200H 的子程序 SETREC2
;设置放音起始地址为 100H 的子程序 SETPLY1
;设置放音起始地址为 200H 的子程序 SETPLY2
  SETREC1:MOV     COMH,#0A1H;设置录音起始地址为 100H 的子程序
          SJMP    SETPLYA
  SETREC2:MOV     COMH,#0A2H;设置录音起始地址为 200H 的子程序
          SJMP    SETPLYA
  SETPLY1:MOV     COMH,#0E1H;设置放音起始地址为 100H 的子程序
          SJMP    SETPLYA
  SETPLY2:MOV     COMH,#0E2H;设置放音起始地址为 200H 的子程序
          SJMP    SETPLYA
```

```
SETPLYA:MOV     COML,#00H
        CALL    COMM
        RET
;主机向 ISD4002 连续发送 2 个字节指令的子程序 COMM
;操作指令（双字节）高 8 位在 COMH 中，低 8 位在 COML 中。
COMM:   CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
        MOV     A,COML
        ACALL   TRANSFE   ;发送命令低字节
        MOV     A,COMH
        CALL    TRANSFE   ;发送命令高字节
        SETB    SS                          ;置 SS 为高,结束 SPI 总线通讯
        RET
        ;送连续放音指令的子程序
PLAY:   MOV     A,#0F0H             ;F0H 为放音指令
        CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
        CALL    TRANSFE   ;发送放音指令
        SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
        RET
        ;送停止指令的子程序 STOP
STOP:   MOV     A,#30H              ;30H 为停止指令
        CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
        CALL    TRANSFE   ;发送停止指令
        SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
        RET
        ;送停止并掉电指令的子程序
STOPPWD: MOV    A,#10H              ;10H 为停止并掉电指令
        CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
        CALL    TRANSFE   ;发送掉电指令
        SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
        RET
        ;送连续录音指令的子程序
REC:    MOV     A,#0B0H             ;B0H 为录音指令
        CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
        CALL    TRANSFE   ;发送录音指令
        SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
        RET
        ;送上电指令(20H)的子程序
POWERUP: MOV    A,#20H              ;20H 为上电指令
```

```
        CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
        CALL    TRANSFE    ;发送上电指令
        SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
        CALL    DEL25mS
        RET
        ;主机向从机串行传送 1 个字节数据的子程序
        ;使用了 R7
        ;输出的字符在 ACC 中
TRANSFE: MOV    R7,#8               ;置接收的位数为 8
BITOUT: CLR     SCLK                ;将时钟 SCLK 置低
        RRC     A                   ;将 ACC 的最低位移到进位 C,作为发送位
        MOV     MOSI,C              ;将发送位送 MOSI 发送
        SETB    SCLK                ;置时钟 SCLK 为高
        DJNZ    R7,BITOUT  ;8 位数据发送完否?未完转 BITOUT 继续发送
        RET
        ;延时 2 秒子程序，使用参数 R0、R7、R6。
DELAY:  MOV     R0,#20
DELY0:  MOV     R7,#100             ;延时 0.1 秒
DELY1:  MOV     R6,#250             ;延时 1mS
        DJNZ    R6,$
        DJNZ    R7,DELY1
        DJNZ    R0,DELY0
        RET
        ;延时 25 毫秒子程序，使用参数 R7、R6。
DEL25mS: MOV    R7,#25
DEL25:  MOV     R6,#250             ;延时 1mS
        DJNZ    R6,$
        DJNZ    R7,DEL25
        RET
```

以上均为标准程序段，仔细理解吃透，对 ISD4002 语音芯片的编程非常有好处，只需编辑主程序，按 ISD4002 语音芯片的时序图调用各子程序段，即可完成一定的录音功能。

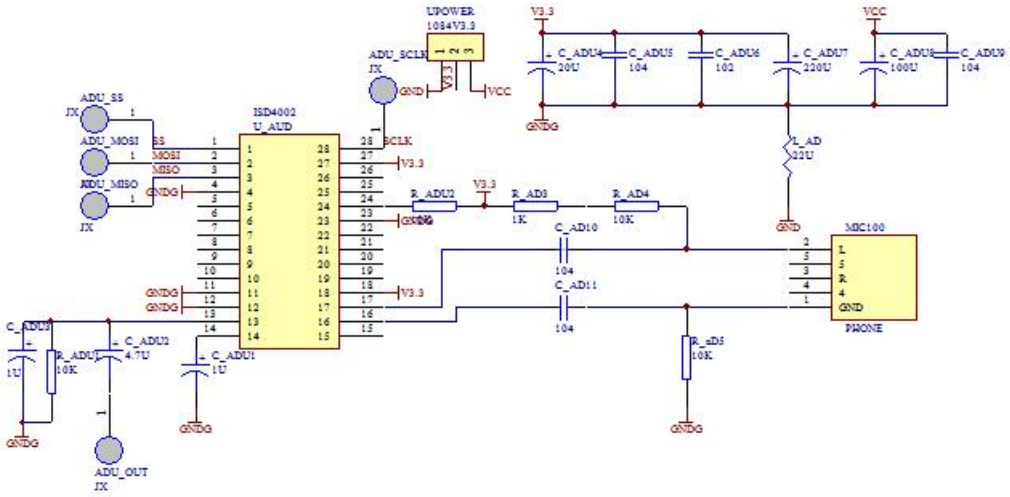
一、实验目的

掌握 SPI 总线接口的 ISD4002 语音芯片的录放音操作，SPI 总线的编程技巧，熟悉 ISD4002 的应用电路图。

二、实验电路原理图及其说明

通过读取 2 个按钮的控制状态，再通过 SPI 总线把地址和控制字节写入 ISD4002。从而实现单

片机与语音芯片联合完成一定功能的模块。



原理图

三、语音芯片－ISD4002 介绍。

1、语音芯片 ISD4002 的特点：

（1）、语音芯片可持续录音、放音。以 ISD4002 芯片为例，持续录放时间可达 120 秒，ISD4004 最长可达 16 分钟，可供多种情况选择；

（2）、断电后信息仍然存储，不会丢失，无需后备电池。信息可保存 100 年之久；

（3）、操作简单，无需专用编程器或语音开发器；

（4）、单电源供电。典型电压为+3V；

（5）、易于与单片机接口；

（6）、内部自带自动音量控制（AGC）电路及滤波电路，输出音质良好。

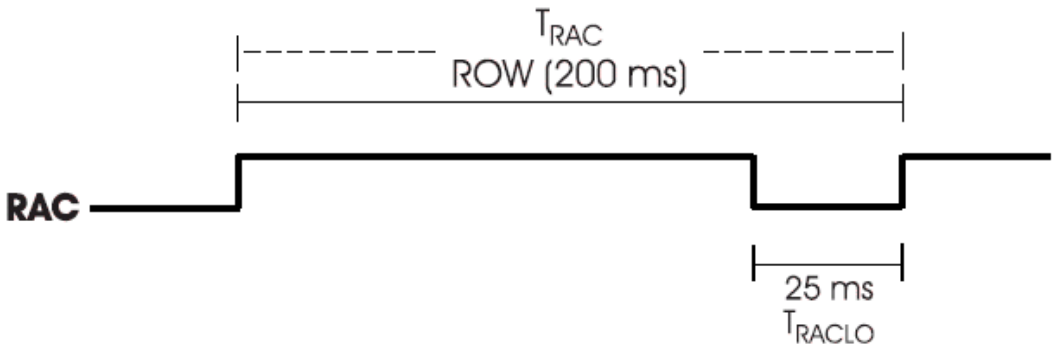
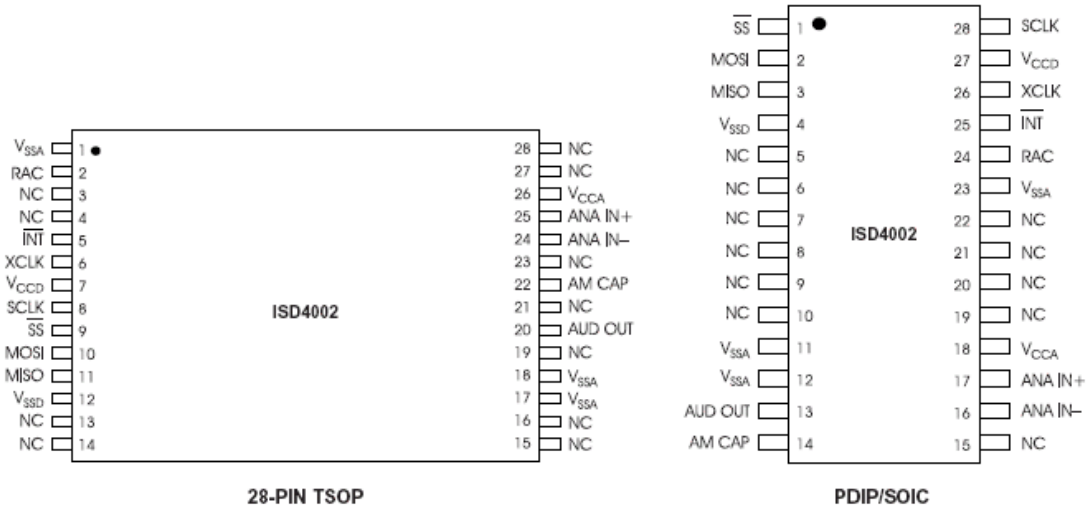


图 4-6-9　RAC 周期

2、 封装及管脚描述：



ISD4002 的封装

表 4-6-1 引脚功能

管　　脚	功　　能
/SS（1 脚）	从设备选则
MOSI（2 脚）	数据输入线
MISO（3 脚）	数据输出线
VSSD（4 脚）	数字地
NC（5～10 脚）	未用
VSSA（11～12 脚）	模拟地
AUD OUT（13 脚）	语音输出
AM CAP（14 脚）	接 1U 电容可去除噪音
NC（15 脚）	未用
ANA IN-（16 脚）	模拟负输入
ANA IN+（17 脚）	模拟正输入
VCCA	模拟电源
NC（19～22）	未用
VSSA （23 脚）	模拟地
RAC（24 脚）	序列地址时钟，每段信息 0.2S，4002 可存 600 段
/INT（25 脚）	信息结束或溢出中断
XCLK（26 脚）	外部时钟输入
VCCD （27 脚）	数字电源
SCLK（28 脚）	串行时钟

Part Number	Sample Rate	Required Clock
ISD4002-120	8.0 KHz	1024 KHz
ISD4002-150	6.4 KHz	819.2 KHz
ISD4002-180	5.3 KHz	682.7 KHz
ISD4002-240	4.0 KHz	512 KHz

外部输入时钟表

四、实验内容

ISD4002 录放音实验。

五、实验步骤

1、接线：

2、按下录音按钮，对着麦克风录音，录音时间要小于 30 秒，然后使录音按钮弹起。等 5 秒后再按下录音按钮，对着麦克风录音，录音时间要小于 60 秒，然后使录音按钮弹起。

3、录音结束后至少等 5 秒，按一下放音按钮，放第一段录音，放完后等 5 秒再按一次放音按钮，放第二段录音。放完第二段后，等 5 秒再按放音按钮，则重放第一段，这样可以一直重复放下去。

4、如果要重新录音，必须按主板上的复位按钮。重复步骤 3～5。

六、参考程序

1. ISD4002 录放音实验。

```
COML    EQU     20H             ;16 位命令的低字节存储单元
COMH    EQU     21H             ;16 位命令的高字节存储单元
MISO    EQU     P1.0            ;主机输入，从设备输出信号
SS      EQU     P1.1            ;从设备片选信号
MOSI    EQU     P3.0            ;主机输出，从设备输入信号
SCLK    EQU     P3.1            ;从设备的时钟脉冲
KEY_L   EQU     P1.2            ;录音按键
KEY_F   EQU     P1.3            ;放音按键
        CSEG    AT 0000H
        LJMP    MAIN
        ORG     0100H
MAIN:   MOV     SP,#60H
        ;接下来是从指定的两个地址开始录两段语音，第 1 段录音不能超过 30 秒,
```

```
        ;第 2 段录音不能超过 60 秒,且要录满两段后才可播放,第一段录音结束后,
        ;至少要等 5 秒,才能进行第二段录音
RECP:   JB      KEY_L,RECP          ;判断录音键是否按下？按下录音开始
        CALL    POWERUP             ;上电指令,包括延时 25ms，录音需连续上电 2 次
        CALL    POWERUP             ;上电指令,包括延时 25ms
        CALL    DEL25mS             ;再延时 25 毫秒
        CALL    SETREC1             ;设置录音起始地址为 100H
        CALL    REC                 ;录音,该段最长不能超过 30s,超过的无效!
RECP2:  JNB     KEY_L,RECP2         ;判断录音键是否弹起？没弹起转 RECP2 继续录音
        CALL    STOP                ;录音键弹起,停止录音
RECP3:  JB      KEY_L,RECP3         ;判断录音键是否按下？按下录音开始
        CALL    POWERUP             ;上电指令,包括延时 25 ms，录音需连续上电两次
        CALL    POWERUP             ;上电指令,包括延时 25 ms
        CALL    DEL25mS             ;再延时 25 ms
        CALL    SETREC2             ;设置录音起始地址为 100H
        CALL    REC                 ;录音,该段最长不能超过 60s,超过的无效!
RECP4:  JNB     KEY_L,RECP4         ;判断录音键是否弹起？没弹起转 RECP2 继续录音
        CALL    STOP                ;录音键弹起,则停止录音
        ;下面是放音程序段,按下放音按钮,从地址 100H 开始放音,再按放音按钮
        ;从第二段地址 200H 放音,再按放音按钮,又从第一段开始放音,一直继续下去。
        ;放下一段录音时,必须在上一段放音结束后等 5 秒后才能按放音按钮.
PLYP11:
        JB      KEY_F,PLYP11        ;放音键未按下转 PLYP1
        CALL    POWERUP             ;重新加电,包括延时 25 毫秒
        CALL    SETPLY1             ;设置放音起始地址为 000H
        CALL    PLAY                ;放音子程序
        CALL    DELAY               ;延时 2 秒
PLYP2:
        JB      KEY_F,PLYP2         ;放音键未按下转 PLYP2
        CALL    POWERUP             ;重新加电,包括延时 25 毫秒
        CALL    SETPLY2             ;设置放音起始地址为 200H
        CALL    PLAY                        ;放音子程序
        CALL    DELAY                       ;延时 2 秒
        SJMP    PLYP1                       ;转 PLYP1 放音第一段
        ;设置录音起始地址为 100H 的子程序 SETREC1
        ;设置录音起始地址为 200H 的子程序 SETREC2
        ;设置放音起始地址为 100H 的子程序 SETPLY1
        ;设置放音起始地址为 200H 的子程序 SETPLY2
```

```
SETREC1:  MOV     COMH,#0A1H          ;设置录音起始地址为 100H 的子程序
          SJMP    SETPLYA
SETREC2:  MOV     COMH,#0A2H          ;设置录音起始地址为 200H 的子程序
          SJMP    SETPLYA
SETPLY1:  MOV     COMH,#0E1H          ;设置放音起始地址为 100H 的子程序
          SJMP    SETPLYA
SETPLY2:  MOV     COMH,#0E2H          ;设置放音起始地址为 200H 的子程序
          SJMP    SETPLYA
SETPLYA:  MOV     COML,#00H
          CALL    COMM
          RET
          ;主机向 ISD4002 连续发送 2 个字节指令的子程序 COMM
          ;操作指令（双字节）高 8 位在 COMH 中，低 8 位在 COML 中。
COMM:     CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
          MOV     A,COML
          ACALL   TRANSFE             ;发送命令低字节
          MOV     A,COMH
          CALL    TRANSFE             ;发送命令高字节
          SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
          RET
          ;送连续放音指令的子程序
PLAY:     MOV     A,#0F0H             ;F0H 为放音指令
          CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
          CALL    TRANSFE             ;发送放音指令
          SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
          RET
          ;送停止指令的子程序 STOP
STOP:     MOV     A,#30H              ;30H 为停止指令
          CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
          CALL    TRANSFE             ;发送停止指令
          SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
          RET
          ;送停止并掉电指令的子程序
STOPPWD:  MOV     A,#10H              ;10H 为停止并掉电指令
          CLR     SS                  ;置 SS 为低,使 SPI 总线能够进行通讯工作
          CALL    TRANSFE             ;发送掉电指令
          SETB    SS                  ;置 SS 为高,结束 SPI 总线通讯
          RET
```

```
        ;送连续录音指令的子程序
REC:    MOV     A,#0B0H         ;B0H 为录音指令
        CLR     SS              ;置 SS 为低,使 SPI 总线能够进行通讯工作
        CALL    TRANSFE         ;发送录音指令
        SETB    SS              ;置 SS 为高,结束 SPI 总线通讯
        RET
        ;送上电指令(20H)的子程序
POWERUP: MOV    A,#20H          ;20H 为上电指令
        CLR     SS              ;置 SS 为低,使 SPI 总线能够进行通讯工作
        CALL    TRANSFE         ;发送上电指令
        SETB    SS              ;置 SS 为高,结束 SPI 总线通讯
        CALL    DEL25mS
        RET
        ;主机向从机串行传送 1 个字节数据的子程序
        ;使用了 R7
        ;输出的字符在 ACC 中
TRANSFE: MOV    R7,#8           ;置接收的位数为 8
BITOUT: CLR     SCLK            ;将时钟 SCLK 置低
        RRC     A               ;将 ACC 的最低位移到进位 C,作为发送位
        MOV     MOSI,C                  ;将发送位送 MOSI 发送
        SETB    SCLK                    ;置时钟 SCLK 为高
        DJNZ    R7,BITOUT       ;8 位数据发送完否?未完转 BITOUT 继续发送
        RET
        ;延时 2 秒子程序，使用参数 R0、R7、R6。
DELAY:  MOV     R0,#20
DELY0:  MOV     R7,#100         ;延时 0.1 秒
DELY1:  MOV     R6,#250         ;延时 1mS
        DJNZ    R6,$
        DJNZ    R7,DELY1
        DJNZ    R0,DELY0
        RET
        ;延时 25 毫秒子程序，使用参数 R7、R6。
DEL25mS: MOV    R7,#25
DEL25:  MOV     R6,#250         ;延时 1mS
        DJNZ    R6,$
        DJNZ    R7,DEL25
        RET
        END
```

实验二十四　　扩展 RAM 实验

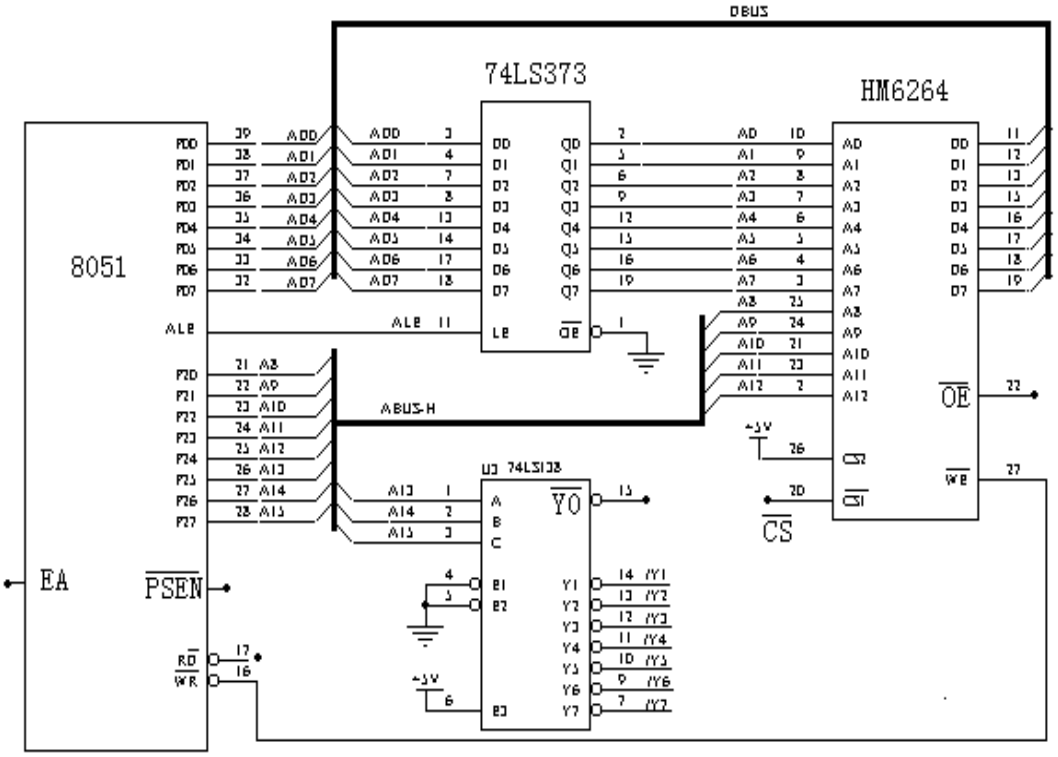
一、实验目的

1、掌握8051单片机扩展外部数据存储器的方法。

2、掌握8051内部RAM和外部RAM之间的数据传送的方法。

二、实验原理与内容

1、 实验仪上有一片容量为 2K×8 位的 RAM 型号为 6116， 6116 与 8051 单片机的连接电路如图所示。



把 6264 作为外部 RAM（扩展外部数据存储器）。

编写数据传送程序，使 8031 内部 RAM30H~3FH 单元置初值 00H~0FH，然后传送到外部 RAM(6264)的 F000H~F00FH 单元中，再将 F000H~F00FH 中内容求和送到 P1 口显示。

三、参考程序　　RAM.ASM

```
        org     00h
        mov r0, #10h                ; r0 为循环计数器
        mov a,  #0
        mov r1, #30h                ; r1 为片内 RAM 地址指针
WriAga: mov @r1, a                  ; 写片内 RAM 一个单元
        inc a
        inc r1                      ; 地址指针加 1
        djnz r0, WriAga             ; 循环计数器减 1，不等于 0，转移
```

```
        mov dptr,#0             ; DPTR 为片外 RAM 地址指针
        mov r2,  #10h           ; r2 为循环计数器
        mov r0,  #30h           ; r0 为片内 RAM 地址指针
ReaWri: mov a,   @r0            ; 读片内 RAM 一个单元
        movx     @dptr,a        ; 写入外部 RAM 一个单元
        inc dptr                ; 片外 RAM 地址指针加 1
        inc r0                  ; 片内 RAM 地址指针加 1
        djnz r2, ReaWri         ; 循环计数器减 1，不等于 0，转移
        mov r0,  #10h           ; r0 为循环计数器
        mov dptr,#0             ; DPTR 为片外 RAM 地址指针
        mov 40h, #0             ; 40H 为和数的存放单元
addAga: movx a,  @dptr          ; 读片外 RAM 一个单元到 a 中
        MOV B,   A              ; 寄存到 b 中
        MOV A,   40H
        add a,   B              ; 求和
        MOV 40H, A              ; 存和数
        inc      dptr           ; 片外 RAM 地址指针加 1
        djnz r0, addAga         ; 循环计数器减 1，不等于 0，转移
        mov p1,  40H            ; 将和送 P1 口显示
        sjmp     $
```

实验二十五　扩展可编程定时器/计数器 8254

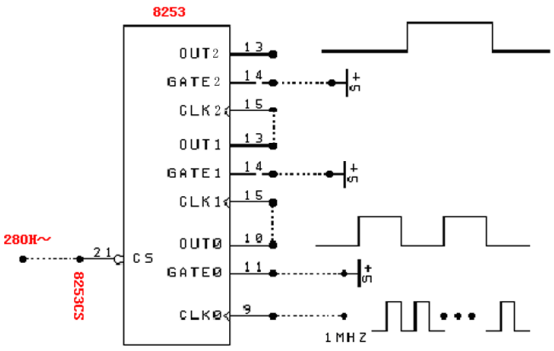
一、实验目的

掌握 8051 单片机扩展可编程定时器/计数器 8253 的方法。

掌握可编程定时器/计数器 8253 的方法。

二、实验内容

按下图虚线连接电路，将计数器0、计数器1、计数器2、分别设置为方式3，分别进行2、25、40分频，用逻辑笔观察OUT1输出电平的变化。



三、编程提示

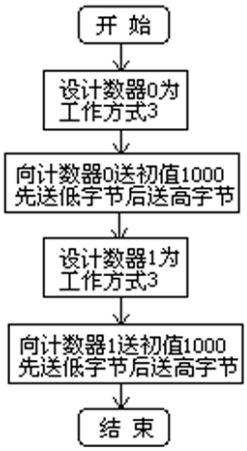
1、　8253 控制寄存器地址　　E003

计数器 0 地址　　E000H

计数器 1 地址　　E001H

CLK0 连接时钟　　1MHZ

2、参考流程图



四、参考程序

```
AD8253A        EQU        0280H                ;8253 计数器 0 地址
```

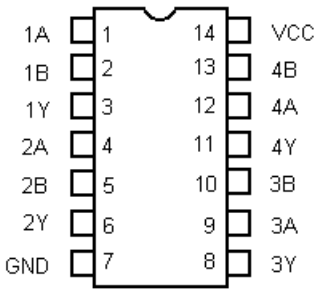
```
AD8253B         EQU     0281H               ;8253 计数器 1 地址
AD8253C         EQU     0282H               ;8253 计数器 2 地址
ADRCTRL         EQU     0283H               ;8253 控制寄存器地址
        CSEG    AT 0000H
        LJMP    MAIN
        ORG     0100H
MAIN:   MOV     SP,#60H
        MOV     PSW,#00H
        LCALL   DIVI2               ;使用计数器 0 对 CLK0 时钟(1MHz)进行 2 分频
        LCALL   DIVI25              ;使用计数器 1 对 CLK1 时钟(OUT0)进行 25 分频
        LCALL   DIVI40              ;使用计数器 2 对 CLK2 时钟(OUT1)进行 40 分频
        SJMP    $
        ;对 CLK0(1MHz)进行 2 分频子程序;使用了 DPTR、ACC
DIVI2:  MOV     DPTR,#ADRCTRL       ;写 8253 控制寄存器
        MOV     A,#16H      ;控制字:选择 0 号计数器,仅读/输入低字节,选择方式 3
                            ;采用 16 位 2 进制计数器
        MOVX    @DPTR,A
        MOV     DPTR,#AD8253A;对 1M 时钟 CLK0 进行 2 分频
        MOV     A,#2
        MOVX    @DPTR,A
        RET
        ;对 CLK1 时钟(OUT0)进行 25 分频子程序   ;使用了 DPTR、ACC
DIVI25: MOV     DPTR,#ADRCTRL       ;写 8253 控制寄存器
        MOV     A,#56H      ;控制字: 1 号,仅读/输入低字节,方式 3; 16 位 2 进制计数器
        MOVX    @DPTR, A
        MOV     DPTR,#AD8253B       ;对时钟 CLK1(OUT0)进行 25 分频
        MOV     A,#25
        MOVX    @DPTR,A
        RET
        ;对 CLK2 时钟(OUT1)进行 40 分频子程序,最终得到的频率是 500HZ
DIVI40: MOV     DPTR,#ADRCTRL       ;写 8253 控制寄存器
        MOV     A,#96H      ;控制字: 1 号,仅读/输入低字节,方式 3; 16 位 2 进制计数器
        MOVX    @DPTR,A
        MOV     DPTR,#AD8253C       ;对时钟 CLK2(OUT1)进行 40 分频
        MOV     A,#40
        MOVX    @DPTR,A
        RET
        END
```

附录

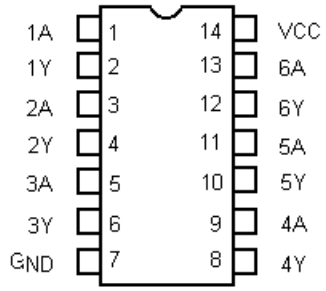
附录一　常用实验器件引脚图

1.　四 2 输入正与非门 74LS00

$Y = \overline{AB}$



$Y = \overline{A}$



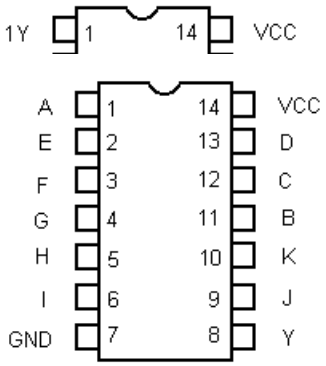
2．六反相器 74LS04

3．四 2 输入正或非门 74LS28

4．4-2-3-2 与或非门 74LS64

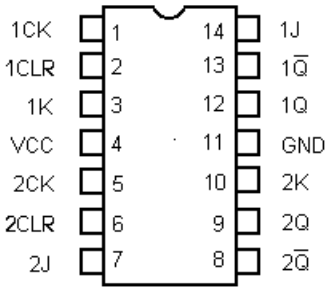
5．双 J-K 触发器（带清除端）74LS73A

$Y=\overline{ABCD+EF+GHI+JK}$



真 值 表

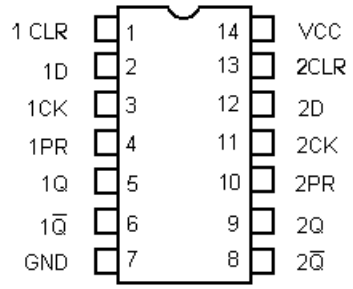
输入				输出	
清除	时钟	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\overline{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	反转	
H	H	X	X	Q_0	$\overline{Q}_0$



6．双 D 型正边沿触发器（带预置和清除端）74LS74

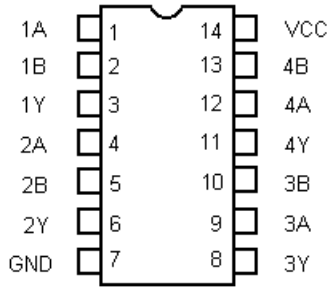
真 值 表

输入				输出	
预置	清除	时钟	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$



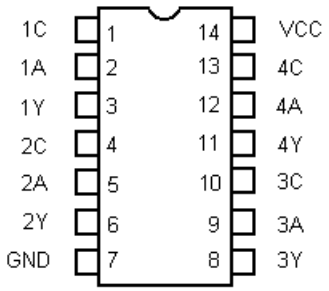
7．四 2 输入异或门 74LS86

$Y=A \oplus B = A\overline{B} + \overline{A}B$



8．三态输出的四总线缓冲门 125

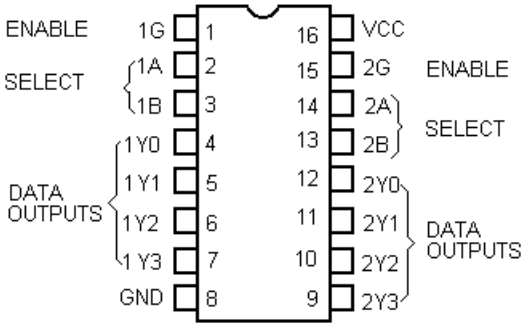
正逻辑
Y = A
C为高时输出截止



9．双 2—4 线译码器/分配器 139

真值表

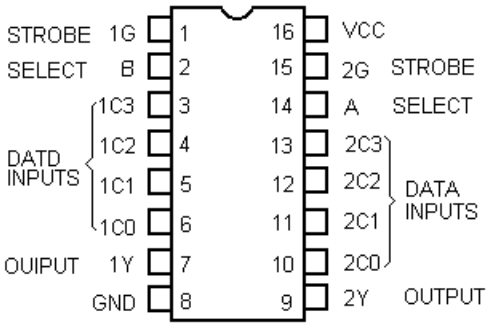
输入端			输出端			
允许 G	选择 B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L



10．双 4—1 线数据选择器/多路开关 74LS153

真 值 表

选择输入		数据输入				选通	输出
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

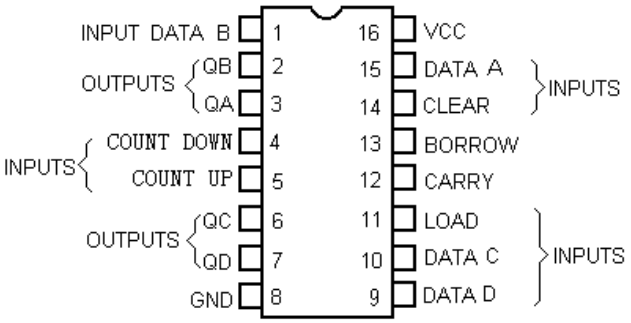


11．同步十进制加/减计数器（双时钟带清除端）74LS192

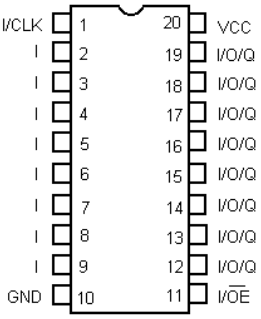
1. CLEAR是高有效。

2. LOAD下降沿预置数据。

3. QD是最高位,QA是最低位。



12．GAL16V8



附录二 Keil C库函数

C-51 软件包的库包含标准的应用程序，每个函数都在相应的头文件（.h）中有原型声明。如果使用库函数，必须在源程序中用预编译指令定义与该函数相关的头文件（包含了该函数的原型声明）。例如：

#include

#include

如果省掉头文件，编译器则期望标准的C 参数类型，从而不能保证函数的正确执行。

1 CTYPE.H：字符函数

在CTYPE.H 头文件中包含下列一些库函数：

函数名： isalpha

原型： extern bit isalpha(char)

功能： isalpha 检查传入的字符是否在‘A’-‘Z’和‘a’-‘z’之间，如果为真返回值为1，否则为0。

函数名： isalnum

原型： extern bit isalnum(char)

功能： isalnum 检查字符是否位于‘A’-‘Z’，‘a’-‘z’或‘0’-‘9’之间，为真返回值是1，否则为0。

函数名： iscntrl

原型： extern bit iscntrl(char)

功能： iscntrl 检查字符是否位于0x00～0x1F 之间或0x7F，为真返回值是1，否则为0。

函数名： isdigit

原型： extern bit isdigit(char)

功能： isdigit 检查字符是否在‘0’-‘9’之间，为真返回值是1，否则为0。

函数名： isgraph

原型： extern bit isgraph(char)

功能： isgraph 检查变量是否为可打印字符，可打印字符的值域为0x21～0x7E。若为可打印，返回值为1，否则为0。

函数名： isprint

原型： extern bit isprint(char)

功能：除与isgraph 相同外，还接受空格字符（0X20）。

函数名： ispunct

原型： extern bit ispunct(char)

功能： ispunct 检查字符是否位为标点或空格。如果该字符是个空格或32 个标点和格式字符之一（假定使用ASCII 字符集中128 个标准字符），则返回1，否则返回0。Ispunct 对下列字符返回1：

（空格）！“$%^&()+,-./:<=>?_[‘~{}

函数名： islower

原型： extern bit islower(char)

功能： islower 检查字符变量是否位于‘a’-‘z’之间，为真返回值是1，否则为0。

函数名： isupper

原型： extern bit isupper(char)

功能： isupper 检查字符变量是否位于‘A’-‘Z’之间，为真返回值是1，否则为0。

函数名： isspace

原型： extern bit isspace(char)

功能： isspace 检查字符变量是否为下列之一：空格、制表符、回车、换行、垂直制表符和送纸。为真返回值是1，否则为0。

函数名： isxdigit

原型： extern bit isxdigit(char)

功能： isxdigit 检查字符变量是否位于‘0’-‘9’，‘A’-‘F’或‘a’-‘f’之间，为真返回值是1，否则为0。

函数名： toascii

原型： toascii(c)((c)&0x7F);

功能：该宏将任何整型值缩小到有效的ASCII 范围内,它将变量和0x7F 相与从而去掉低7 位以上所有数位。

函数名： toint

原型： extern char toint(char)

功能： toint 将ASCII 字符转换为16 进制，返回值0 到9 由ASCII 字符‘0’到‘9’得到，10 到15 由ASCII 字符‘a’-‘f’（与大小写无关）得到。

函数名： tolower

原型： extern char tolower(char)

功能： tolower 将字符转换为小写形式，如果字符变量不在‘A’-‘Z’之间，则不作转换，返回该字符。

函数名： _tolower

原型： tolower(c);(c-‘A’+‘a’)

功能：该宏将0x20 参量值逐位相或。

函数名： toupper

原型： extern char toupper(char)

功能： toupper 将字符转换为大写形式，如果字符变量不在‘a’-‘z’之间，则不作转换，返回该字符。

函数名： _toupper

原型： _toupper(c);((c)-‘a’+’A’)

功能： _toupper 宏将c 与0xDF 逐位相与。

2 STDIO.H：一般I/O 函数

C51 编译器包含字符I/O 函数，它们通过处理器的串行接口操作，为支持其它I/O机制，只需修改getkey()和putchar()函数，其它所有I/O 支持函数依赖这两个模块，不需要改动。在使用8051 串行口之前，必须将它们初始化，下例以2400波特率，12MHz 初始化串口：

SCON=0x52

TMOD=0x20

TR1=1

TH1=0Xf3

其它工作模式和波特率等细节问题可以从8051 用户手册中得到。

函数名： _getkey

原型： extern char _getkey();

功能： _getkey()从8051 串口读入一个字符，然后等待字符输入，这个函数是改变整个输入端口机制应作修改的唯一一个函数。

函数名： getchar

原型： extern char _getchar();

功能： getchar()使用_getkey 从串口读入字符，除了读入的字符马上传给putchar 函数以作响应外，与_getkey 相同。

函数名： gets

原型： extern char *gets(char *s, int n);

功能：该函数通过getchar 从控制台设备读入一个字符送入由‘s’指向的数据组。考虑到ANSI 标准的建议，限制每次调用时能读入的最大字符数，函数提供了一个字符计数器‘n’，在所有情况下，当检测到换行符时，放弃字符输入。

函数名： ungetchar

原型： extern char ungetchar(char);

功能： ungetchar 将输入字符推回输入缓冲区，因此下次gets 或getchar 可用该字符。ungetchar 成功时返回‘char’，失败时返回EOF，不可能用ungetchar 处理多个字符。

函数名： _ungetchar

原型： extern char _ungetchar(char);

功能： _ungetchar 将传入字符送回输入缓冲区并将其值返回给调用者，下次使用getkey时可获得该字符，写回多个字符是不可能的。

函数名： putchar

原型： extern putchar(char);

功能： putchar 通过8051 串口输出‘char’，和函数getkey 一样，putchar是改变整个输出机制所需修改的唯一一个函数。

函数名： printf

原型： extern int printf(const char*, …);

功能： printf 以一定格式通过8051 串口输出数值和串，返回值为实际输出的字符数，参量可以是指针、字符或数值，第一个参量是格式串指针。

注：允许作为printf 参量的总字节数由C51 库限制，因为8051 结构上存贮空间有限，在SMALL 和COMPACT 模式下，最大可传递15 个字节的参数（即5 个指针，或1 个指针和3 个长字节），在LARGE 模式下，至多可传递40 个字节的参数。格式控制串包含下列域（方括号内的域是可能的）：

%[flags][width][.precision]type“width”域定义了参量欲显示的字符数，它必须是一个十进制数，如果实际显示的字符数小于“width”，输出左端补以空格，如果“width”域以0 开始，则左端补0。

“flag”域用来定义下面选项：

Falg　意义

-　输出左齐

+　输出值如果是有符号数值，则加上+/-号

‘ ‘(空格)

输出值如果为正则左边补以空格显示

#

如果它与0，x 或X 联用，则在输出前加上字符0、0x，0X。当与值类型g、G、f、e、E 联用时，‘#’使输出数产生一个十进制小数点。

b,B

它们与格式类型d、i、o、u、x、X 联用，这样参量类型被接受为‘[unsigned]char’，如：%bu，%bd 或%bx。

L,L

它们与格式类型d、i、o、u、x、X 联用，这样参量类型被接受为‘[unsigned]long’，如：%lu，%ld 或%lx。

*

下一个参量不作输出。

“type“域定义参量如下类型：

字符

类型

输出格式

d

int

有符号十进制数（16 位）

U

int

无符号十进制数

o

int

无符号八进制数

X，x

int

无符号十六进制数

f

float

[-]dddd.dddd 形式的浮点数

e，E

float

[-]d.ddddE[sign]dd 形式的浮点数

g，G

float

e 或f 形式浮点数，看哪一种输出形式好。

c

char

字符

s

pointer

指向一个带结束符号的串

p

pointer

带存贮器指示符和偏移的指针。M:aaaa。

M:=C(ode),D(ata),I(data),P(data) aaaa:指针偏移值。

例子：

printf(“Int-Val%d,Char-Val%bd,Long-Val%d”,I,c,l);

printf(“String%s,Character%c”,array,character);

printf(“Pointer%p”,&array[10]);

函数名： sprintf

原型： extern int sprintf(char *s，const char*，…);

功能： sprintf 与printf 相似，但输出不显示在控制台上，而是通过一个指针S，送入可寻址的缓冲区。

注：sprintf 允许输出的参量总字节数与printf 完全相同。

函数名： puts

原型： extern int puts(const char*，…);

功能： puts 将串‘s’和换行符写入控制台设备，错误时返回EOF，否则返回一非负数。

函数名： scanf

原型： extern int scanf(const char*，…);

功能： scanf 在格式串控制下，利用getcha 函数由控制台读入数据，每遇到一个值（符号格式串规定），就将它按顺序赋给每个参量，注意每个参量必须都是指针。scanf返回它所发现并转换的输入项数。若遇到错误返回EOF。格式串包括：

l 空格、制表符等，这些空白字符被忽略。
l 字符，除需匹配的“%”（格式控制字符）外。
l 转换指定字符“%”，后随几个可选字符；赋值抑制符“*”，一个指定最大域宽的数。
注：scanf 参量允许的总字节数与printf 相同，格式控制串可包括下列域（方括号内是可选的）：
%[flags][width]type
格式串总是以百分号开始，每个域包含一个或多个字符或数。“width”域定义了参量可接受的字符数，“width”必须是一个正十进制数。如果实际输入字符数量小于“width”，则不会进行填充。‘flag’域用来定义下面选项：
Flag
意义
*
输入被忽略
b，h
它们用作格式类型d，i，o，u 和x 的前缀，用这些变量可定义参量是字符指针还是无符号字符指针。如%bu，%bd，%bx。
L
它们被作格式类型d，i，o，u 和x 的前缀，使用这个前缀可定义参量是长指针还是无符号字长指针。如%lu，%ld，%lx。
“type”域定义参量为如下类型：
描述符
类型
输入格式
d
ptr to int
有符号十进制数（16 位）
i
ptr to int
如C 中记号一样，整型值
u
ptr to int
无符号十进制数
o
ptr to int
无符号八进制数
x
ptr to int
无符号十六进制数

f，e，g

ptr to float

浮点数

c

ptr to char

一个字符

s

ptr to string

一个字串

例子：

scanf(“%d%bd%ld”,&i,&c,&l);

scanf(“%f”,&f);

scanf(“%3s,%c”,&string[0],&character);

函数名： sscanf

原型： extern int sscanf(const *s,const char*，…);

功能： sscanf 与scanf 方式相似,但串输入不是通过控制台，而是通过另一个以空结束的指针。

注：sscanf 参量允许的总字节数由C-51 库限制，这是因为8051 处理器结构内存的限制，在SMALL和COMPACT 模式，最大允许15 字节参数（即至多5 个指针，或2 个指针，2 个长整型或1 个字符型）的传递。在LARGE 模式下，最大允许传送40 个字节的参数。

3 STRING.H：串函数

串函数通常将指针串作输入值。一个串就包括2 个或多个字符。串结以空字符表示。在函数memcmp，memcpy，memchr，memccpy，memmove 和memset 中，串长度由调用者明确规定，使这些函数可工作在任何模式下。

函数名： memchr

原型： extern void *memchr(void *s1， char val，int len);

功能： memchr 顺序搜索s1 中的len 个字符找出字符val，成功时返回s1 中指向val 的指针，失败时返回NULL。

函数名： memcmp

原型： extern char memcmp(void *s1， void *s2，int len);

功能： memcmp 逐个字符比较串s1 和s2 的前len 个字符。相等时返回0，如果串s1 大于或小于s2，则相应返回一个正数或负数。

函数名： memcpy

原型： extern void *memcpy(void *dest， void *src，int len);

功能： memcpy 由src 所指内存中拷贝len 个字符到dest 中，返回批向dest 中的最后一个字符的指针。如果src 和dest 发生交迭，则结果是不可预测的。

函数名： memccpy

原型： extern void *memccpy(void *dest， void *src，char val，int len);

功能： memccpy 拷贝src 中len 个字符到dest 中，如果实际拷贝了len 个字符返回NULL。拷贝过程在拷贝完字符val 后停止，此时返回指向dest 中下一个元素的指针。

函数名： memmove

原型： extern void *memmove(void *dest， void *src，int len);

功能： memmove 工作方式与memcpy 相同，但拷贝区可以交迭。

函数名： memset

原型： extern void *memset(void *s， char val，int len);

功能： memset 将val 值填充指针s 中len 个单元。

函数名： strcat

原型： extern char *strcat(char *s1， char *s2);

功能： strcat 将串s2 拷贝到串s1 结尾。它假定s1 定义的地址区足以接受两个串。返回指针指向s1 串的第一字符。

函数名： strncat

原型： extern char *strncat(char *s1， char *s2，int n);

功能： strncat 拷贝串s2 中n 个字符到串s1 结尾。如果s2 比n 短，则只拷贝s2。

函数名： strcmp

原型： extern char strcmp(char *s1， char *s2);

功能： strcmp 比较串s1 和s2，如果相等返回0，如果s1s2 则返回一个正数。

函数名： strncmp

原型： extern char strncmp(char *s1， char *s2，int n);

功能： strncmp 比较串s1 和s2 中前n 个字符，返回值与strncmp 相同。

函数名： strcpy

原型： extern char *strcpy(char *s1， char *s2);

功能： strcpy 将串s2 包括结束符拷贝到s1，返回指向s1 的第一个字符的指针。

函数名： strncpy

原型： extern char *strncpy(char *s1， char *s2，int n);

功能： strncpy 与strcpy 相似，但只拷贝n 个字符。如果s2 长度小于n，则s1 串以‘0’补齐到长度n。

函数名： strlen

原型： extern int strlen(char *s1);

功能： strlen 返回串s1 字符个数（包括结束字符）。

函数名： strchr，strpos

原型： extern char *strchr(char *s1， char c);

extern int strpos（char *s1，char c）；

功能： strchr 搜索s1 串中第一个出现的‘c’字符，如果成功，返回指向该字符的别指针，搜索也包括结束符。搜索一个空字符返回指向空字符的指针而不是空指针。strpos 与strchr 相似，但它返回字符在串中的位置或-1，s1 串的第一个字符位置是0。

函数名： strrchr，strrpos

原型： extern char *strrchr(char *s1， char c);extern int strrpos（char *s1，char c）；

功能： strrchr 搜索s1 串中最后一个出现的‘c’字符，如果成功，返回指向该字符的指针，否则返回NULL。对s1 搜索也返回指向字符的指针而不是空指针。strrpos 与strrchr 相似，但它返回字符在串中的位置或-1。

函数名： strspn，strcspn，strpbrk，strrpbrk

原型： extern int strspn(char *s1， char *set);

extern int strcspn(char *s1，char *set);

extern char *strpbrk(char *s1,char *set);

extern char *strpbrk(char *s1,char *set);

功能： strspn 搜索s1 串中第一个不包含在set 中的字符，返回值是s1 中包含在set 里字符的个数。如果s1 中所有字符都包含在set 里，则返回s1 的长度（包括结束符）。如果s1 是空串，则返回0。strcspn 与strspn 类似，但它搜索的是s1 串中的第一个包含在set 里的字符。strpbrk 与strspn 很相似，但它返回指向搜索到字符的指针，而不是个数，如果未找到，则返回NULL。strrpbrk 与strpbrk 相似，但它返回s1 中指向找到的set 字集中最后一个字符的指针。

4 STDLIB.H：标准函数

函数名： atof

原型： extern double atof(char *s1);

功能： atof 将s1 串转换为浮点值并返回它。输入串必须包含与浮点值规定相符的数。C51 编译器对数据类型float 和double 相同对待。

函数名： atol

原型： extern long atol(char *s1);

功能： atol 将s1 串转换成一个长整型值并返回它。输入串必须包含与长整型值规定相符的数。

函数名： atoi

原型： extern int atoi(char *s1);

功能： atoi 将s1 串转换为整型数并返回它。输入串必须包含与整型数规定相符的数。

5 MATH.H：数学函数

函数名： abs，cabs，fabs，labs

原型： extern int abs(int val);

extern char cabs(char val);

extern float fabs(float val);

extern long labs(long val);

功能： abs 决定了变量val 的绝对值，如果val 为正，则不作改变返回；如果为负，则返回相反数。这四个函数除了变量和返回值的数据不一样外，它们功能相同。

函数名： exp，log，log10

原型： extern float exp(float x);

extern float log(float x);

extern float log10(float x);

功能：exp 返回以e 为底x 的幂，log 返回x 的自然数（e=2.718282），log10 返回x 以10为底的数。

函数名： sqrt

原型： extern float sqrt(float x);

功能： sqrt 返回x 的平方根。

函数名：rand，srand

原型： extern int rand(void);

extern void srand（int n）；

功能： rand 返回一个0 到32767 之间的伪随机数。srand 用来将随机数发生器初始化成一个已知（或期望）值，对rand 的相继调用将产生相同序列的随机数。

函数名： cos，sin，tan

原型： extern float cos(flaot x);

extern float sin(flaot x);

extern flaot tan(flaot x);

功能： cos 返回x 的余弦值。Sin 返回x 的正弦值。tan 返回x 的正切值，所有函数变量范围为-π/2～+π/2，变量必须在±65535 之间，否则会产生一个NaN 错误。

函数名： acos，asin，atan，atan2

原型： extern float acos(float x);

extern float asin(float x);

extern float atan(float x);

extern float atan(float y,float x);

功能： acos 返回x 的反余弦值，asin 返回x 的正弦值，atan 返回x 的反正切值，它们的值域为-π/2～+π/2。atan2 返回x/y 的反正切，其值域为-π～+π。

函数名： cosh，sinh，tanh

原型： extern float cosh(float x);

extern float sinh(float x);

extern float tanh(float x);

功能： cosh 返回x 的双曲余弦值；sinh 返回x 的双曲正弦值；tanh 返回x 的双曲正切值。

函数名： fpsave，fprestore

原型： extern void fpsave(struct FPBUF *p);

extern void fprestore (struct FPBUF *p);

功能： fpsave 保存浮点子程序的状态。fprestore 将浮点子程序的状态恢复为其原始状态，当用中断程序执行浮点运算时这两个函数是有用的。

6 ABSACC.H：绝对地址访问

函数名： CBYTE，DBYTE，PBYTE，XBYTE

原型： #define CBYTE((unsigned char *)0x50000L)

#define DBYTE((unsigned char *)0x40000L)

#define PBYTE((unsigned char *)0x30000L)

#define XBYTE((unsigned char *)0x20000L)

功能： 上述宏定义用来对8051 地址空间作绝对地址访问，因此，可以字节寻址。CBYTE寻址CODE区，DBYTE 寻址DATA 区，PBYTE 寻址XDATA 区（通过MOVX @RO 命令），XBYTE 寻址XDATA 区（通过MOVX @DPTR 命令）。

例：下列指令在外存区访问地址0x1000

xval=XBYTE[0x1000];

XBYTE[0X1000]=20;

通过使用#define 指令，用符号可定义绝对地址，如符号X10 可与XBYTE[0x1000]地址相等：

#define X10 XBYTE[0x1000]。

函数名： CWORD，DWORD，PWORD，XWORD

原型： #define CWORD((unsigned int *)0x50000L)

#define DWORD((unsigned int *)0x40000L)

#define PWORD((unsigned int *)0x30000L)

#define XWORD((unsigned int *)0x20000L)

功能：这些宏与上面相似，只是它们指定的类型为unsigned int。通过灵活的数据类型，所有地址空间都可以访问。

7 INTRINS.H：内部函数

函数名： _crol_，_irol_，_lrol_

原型： unsigned char _crol_(unsigned char val,unsigned char n);

unsigned int _irol_(unsigned int val,unsigned char n);

unsigned int _lrol_(unsigned int val,unsigned char n);

功能：_crol_，_irol_，_lrol_以位形式将val 左移n 位，该函数与8051“RLA”指令相关，上面几个函数不同于参数类型。

例：

```
#include
main()
{
unsigned int y;
y=0x00ff;
y=_irol_(y,4);
}
```

函数名： _cror_，_iror_，_lror_

原型： unsigned char _cror_(unsigned char val,unsigned char n);

unsigned int _iror_(unsigned int val,unsigned char n);

unsigned int _lror_(unsigned int val,unsigned char n);

功能：_cror_，_iror_，_lror_以位形式将val 右移n 位，该函数与8051“RRA”指令相关，上面几个函数不同于参数类型。

例：

```
#include
main()
{
unsigned int y;
y=0x0ff00;
y=_iror_(y,4);
}
```

函数名： _nop_

原型： void _nop_(void);

功能：_nop_产生一个NOP 指令,该函数可用作C 程序的时间比较。C51 编译器在_nop_函数工作期间不产生函数调用，即在程序中直接执行了NOP 指令。

例：

```
P()=1;
_nop_();
P()=0;
```

函数名： _testbit_

原型：bit _testbit_(bit x);

功能：_testbit_产生一个JBC 指令，该函数测试一个位，当置位时返回1，否则返回0。如果该位置为1，则将该位复位为0。8051 的JBC 指令即用作此目的。_testbit_只能用于可直接寻址的位；在表达式中使用是不允许的。

8 STDARG.H：变量参数表

C51 编译器允许再入函数的变量参数（记号为“…”）。头文件STDARG.H 允许处理函数的参数表,在编译时它们的长度和数据类型是未知的。为此，定义了下列宏。

宏名： va_list

功能： 指向参数的指针。

宏名： va_stat(va_list pointer,last_argument)

功能： 初始化指向参数的指针。

宏名： type va_arg(va_list pointer,type)

功能：返回类型为type 的参数。

宏名： va_end(va_list pointer)

功能： 识别表尾的哑宏。

9 SETJMP.H：全程跳转

Setjmp.h 中的函数用作正常的系列数调用和函数结束，它允许从深层函数调用中直接返回。

函数名： setjmp

原型： int setjmp(jmp_buf env);

功能： setjmp 将状态信息存入env 供函数longjmp 使用。当直接调用setjmp 时返回值是0，当由longjmp 调用时返回非零值，setjmp 只能在语句IF 或SWITCH 中调用一次。

函数名：longjmp

原型：longjmp(jmp_bufenv,intval);

功能：longjmp恢复调用setjmp时存在env中的状态。程序继续执行，似乎函数setjmp已被执行过。由setjmp返回的值是在函数longjmp中传送的值val，由setjmp调用的函数中的所有自动变量和未用易失性定义的变量的值都要改变。

10REGxxx.H：访问SFR和SFR-BIT地址

文件REG51.H，REG52.H和REG552.H允许访问8051系列的SFR和SFR-bit的地址，这些文件都包含#include指令，并定义了所需的所有SFR名以寻址8051系列的外围电路地址，对于8051系列中其它一些器件，用户可用文件编辑器容易地产生一个“.h”文件。

下例表明了对8051PORT0和PORT1的访问：

```
#include
main(){
if(p0==0x10)p1=0x50;
}
```

KeilC51库函数原型列表

1.1.CTYPE.H

bitisalnum(charc);

bitisalpha(charc);

bitiscntrl(charc);

bitisdigit(charc);

bitisgraph(charc);

bitislower(charc);

bitisprint(charc);

bitispunct(charc);

bitisspace(charc);

bitisupper(charc);

bitisxdigit(charc);

bittoascii(charc);

bittoint(charc);

chartolower(charc);

char__tolower(charc);

chartoupper(charc);

char__toupper(charc);

2.2.INTRINS.H

unsignedchar_crol_(unsignedcharc,unsignedcharb);

unsignedchar_cror_(unsignedcharc,unsignedcharb);

unsignedchar_chkfloat_(floatual);

unsignedint_irol_(unsignedinti,unsignedcharb);

unsignedint_iror_(unsignedinti,unsignedcharb);

unsignedlong_irol_(unsignedlongl,unsignedcharb);

unsignedlong_iror_(unsignedlongL,unsignedcharb);

void_nop_(void);

bit_testbit_(bitb);

3.3.STDIO.H

chargetchar(void);

char_getkey(void);

char*gets(char*string,intlen);

intprintf(constchar*fmtstr[,argument]…);

charputchar(charc);

intputs(constchar*string);

intscanf(constchar*fmtstr.[,argument]…);

intsprintf(char*buffer,constchar*fmtstr[;argument]);

intsscanf(char*buffer,constchar*fmtstr[,argument]);

charungetchar(charc);

voidvprintf(constchar*fmtstr,char*argptr);

voidvsprintf(char*buffer,constchar*fmtstr,char*argptr);

4.4.STDLIB.H

floatatof(void*string);

intatoi(void*string);

longatol(void*string);

void*calloc(unsignedintnum,unsignedintlen);

voidfree(voidxdata*p);

voidinit_mempool(void*data*p,unsignedintsize);

void*malloc(unsignedintsize);

intrand(void);

void*realloc(voidxdata*p,unsignedintsize);

voidsrand(intseed);

5.5.STRING.H

void*memccpy(void*dest,void*src,charc,intlen);

void*memchr(void*buf,charc,intlen);

charmemcmp(void*buf1,void*buf2,intlen);

void*memcopy(void*dest,void*SRC,intlen);

void*memmove(void*dest,void*src,intlen);

void*memset(void*buf,charc,intlen);

char*strcat(char*dest,char*src);

char*strchr(constchar*string,charc);

charstrcmp(char*string1,char*string2);
char*strcpy(char*dest,char*src);
intstrcspn(char*src,char*set);
intstrlen(char*src);
char*strncat(char8dest,char*src,intlen);
charstrncmp(char*string1,char*string2,intlen);
charstrncpy(char*dest,char*src,intlen);
char*strpbrk(char*string,char*set);
intstrpos(constchar*string,charc);
char*strrchr(constchar*string,charc);
char*strrpbrk(char*string,char*set);
intstrrpos(constchar*string,charc);
intstrspn(char*string,char*set);

C51强大功能及其高效率的重要体现之一在于其丰富的可直接调用的库函数，多使用库函数使程序代码简单，结构清晰，易于调试和维护，下面介绍C51的库函数系统。

本征库函数(intrinsicroutines)和非本征证库函数

C51提供的本征函数是指编译时直接将固定的代码插入当前行，而不是用ACALL和LCALL语句来实现，这样就大大提供了函数访问的效率，而非本征函数则必须由ACALL及LCALL调用。

C51的本征库函数只有9个，数目虽少，但都非常有用，列如下：
crol,_cror_：将char型变量循环向左(右)移动指定位数后返回
iror,_irol_：将int型变量循环向左(右)移动指定位数后返回
lrol,_lror_：将long型变量循环向左(右)移动指定位数后返回
nop：相当于插入NOP
testbit：相当于JBCbitvar测试该位变量并跳转同时清除。
chkfloat：测试并返回源点数状态。
使用时，必须包含#inclucle<intrins.h>一行。
如不说明，下面谈到的库函数均指非本征库函数。

1.专用寄存器include文件

例如8031、8051均为REG51.h其中包括了所有8051的SFR及其位定义，一般系统都必须包括本文件。
2.绝对地址include文件absacc.h
该文件中实际只定义了几个宏，以确定各存储空间的绝对地址。
3.动态内存分配函数，位于stdlib.h中
4.缓冲区处理函数位于“string.h”中
其中包括拷贝比较移动等函数如：

memccpymemchrmemcmpmemcpymemmovememset
这样很方便地对缓冲区进行处理。
5.输入输出流函数，位于“stdio.h”中
流函数通8051的串口或用户定义的I/O口读写数据，缺省为8051串口，如要修改，比如改为LCD显示，可修改lib目录中的getkey.c及putchar.c源文件，然后在库中替换它们即可。__