计算机控制与应用实验讲义

《计算机控制与应用》小组编

吉林大学计算机科学与技术学院

2010年1月

前　言

本实验课供计算机专业本科高年级学生进行《计算机控制与应用》实验使用。课程设置的实验内容，基本覆盖计算机控制与应用中常见的各种应用领域。考虑到目前实际需要，同时涵盖汇编编程和C语言编程，目的是使学生了解这两方面的应用方式。同时增加了电路图和线路板的分析与制作，使学生亲手掌握实际硬件制作技术与流程。

实验采取学生亲自动手编制程序，观察测量电路输出，完成实验要求的方式。为清晰篇幅，实验原理归纳在一起。各实验题目后面都附有思考题，供同学预习、做实验及完成实验报告时参考。题目的安排不只局限于实验内容本身，力争起到开阔思路，丰富知识的目的。带星号的思考题为必做。

同学做完实验后要完成一份研讨形式的实验报告，其参考格式为：

所学到的原理及体会。

实验准备、调试与完成过程。

实验结果及难点分析讨论。

思考题

列举遇到的疑难点。

对实验设计的改进意见及创意。

实验报告要求文字简练，表达清楚，以严谨求实，认真钻研、富于创新的精神来完成。

本实验题目分为必做与选做两部分。必做题目全部完成的学生可以向指导教师申请自己想作的选做题目。

执笔：郭东伟

目录

实验一 电路原理图分析与设计

一、实验目的和要求

掌握使用计算机辅助软件分析与设计电路原理图的基本方法和流程，学习分析与设计中的通用规则；培养分析原理图的能力，能够独立设计较为简单的电路图。

二、实验原理

使用计算机辅助软件分析设计电路原理图的基本流程与软件的使用方法见附录一。MCS51系列单片机结构见附录三。

三、实验器材

1. PROTEL FOR WINDOWS SCHEMATIC EDITOR 1.5
2. 示例分析文件（微机通用接口板）

四、实验内容

1. 分析示例文件电路图。
2. 设计一个8031基本应用电路的原理图。包括8031CPU及辅助电路，外接8KEPROM，地址范围从2000H开始。使用P1口进行四路开关量输入，四路发光管LED输出。当有任意一路开关闭合，产生中断信号送入INT1。

五、实验步骤

1. 预习内容：附录一、附录三、附录五第一部分。
2. 进入Windows环境，启动软件
3. 打开示例文件，分析电路（可参阅附录五第一部分中LS系列TTL电路说明）。使用选择网络功能跟踪线路流向，判断分析电路功能。
4. 完成思考题
5. 新建一原理图，按照实验内容2要求设计电路图。注意考虑整个图纸布局。

6．功能基本完成后，对电路图元件编号。
7．进行电路电气性能检查，确认无误后请指导教师审查后，保存文件供以后使用。
8．同时，生成元件的BOM文件，估计电路的器件成本。

六、注意事项

本实验包括两个内容，都是使用应用软件完成工作。我们要学习的不是具体软件的熟练应用，因此只要按照附录要求，了解软件的基本功能就可以了。并且要预习8031单片机的基本体系结构（参见附录三－2.4）。

在第一个内容中，我们分析的是以后要使用的单片机仿真机及实验设备的基本电路。要按照线路的逻辑流向，分析涉及到的器件功能。在完成思考题后，进入第二个内容。

在设计过程中，要先绘制出原理框图，然后上机操作。要注意线路是否正确连接，以及使用的工具是否正确。开关的连接应保证无论接通或是断开，都能产生正确电平，不要产生悬空状态。发光管要采用低电平点亮方式，以确保驱动电流足够大。

在绘图过程中，要注意元件管脚与连线的连接方式，另外，对于某些连接错误（特别是LED以及电阻等元件的连接），电气性能检查不能发现。

七、思考题

1．写出示例电路图中存储器2764的寻址范围。★
2．写出示例电路图中8155及三个端口的地址。★
3．若在某个七段数码管上显示一个符号，应该如何控制输出端口。★
4．说明如何检测键盘中是否有某个键按下；当键盘中的EXE键按下后，会读入什么样的数据。★

实验二 电路图设计与线路板制作

一、 实验目的和要求

掌握通过电路原理图绘制板图的基本流程和方法，能够设计和检查较为简单的印刷电路板图纸。本实验只要求同学完成图纸的设计，了解制版的过程，不将图纸送至制版厂实际加工。

二、 实验原理

使用计算机辅助软件分析设计电路原理图的基本流程与软件的使用方法见附录一。

三、 实验器材

1. PROTEL FOR WINDOWS SCHEMATIC EDITOR 1.5
2. PROTEL FOR WINDOWS PCB EDITOR 1.0

四、 实验内容

按照在实验一中设计的8031基本应用电路原理图，进行制版图的设计。

五、 实验步骤

1. 预习内容：附录一；附录三。
2. 进入SCHEMATIC EDITOR，打开实验一完成的原理图，对于没有封装的元件填入正确的封装。
3. 生成原理图的NET文件。进入PCB EDITOR，调入此NET文件，如有错误提示，返回原理图进行修改，直到正确无误。
4. 进入PCB EDITOR，按照3英寸×4英寸或更小的尺寸在Keep Out层画出矩形闭合轮廓。然后重新调入NET文件
5. 使用手工方式或自动布局功能进行元件摆放，自动布局所需的时间比较长，建议手工布局。
6. 设置自动布线选项，进行自动布线。

7. 如果自动布线通过率不足100％，手工调整元件位置重新布线，也可手工直接连线。
8. 产生布线图的NET文件，与原理图的NET文件进行比较。直到完全吻合。

六、 注意事项与提示

1、原理图中元件要编号。

2、调入原理图NET文件后，常见的错误为封装未填或不对，可以通过生成的错误信息文件来检查。

3、可以先不画轮廓，待调入原理图NET文件正确无误后再画。如果没有轮廓，在布局和布线时将出错。

4、手工布局，也就是手工将每个元件移动到合适位置。通过元件移动来完成。

5、 注意检查布线图中是否有应连接管脚未曾连线，如果出现这种情况，需要回到原理图中进行修改并重新进行上述步骤。

6、如果原理图没有元件和封装的修改，而只是连线的修改，可以在布线完成后的图上，Unroute所有布线，然后重新调入NET文件，重利用原有布局。

七、 思考题

1．写出你所设计的电路中使数码管点亮的指令，和读入开关状态的指令。★
2．你所完成的制版图的最小尺寸是多少，是否可以改进。★
3．设电路版制作成本为0.5元/平方厘米，结合器件成本，计算你的电路图总成本。★
4．你认为在制作板图的过程中有那些值得注意的事项。
5．参阅其他参考书，说明那些问题是在设计原理图时可以忽略，而在设计板图时必须和应该考虑的。

实验三 数码管显示

一、实验目的和要求

初步学习和掌握MCS-51的体系结构和汇编语言，了解使用相应的汇编工具。了解数码管输出的原理及编程方式。

二、实验原理

参见附录三。

仿真机上面的数码管，由两个端口控制。一个是位选口，用来选择点亮哪个数码管，一个是字形口，用来输出显示用的七段字形。电路原理参见实验一的原理图。

在本设备中，端口地址与实验一的原理图有所不同，位选口为0FF21H，字形口为0FF22H。

三、实验器材和线路

单片机实验仿真机
＋5V，＋12V电源

四、实验内容

在寄存器R6中存放某十进制数的十位，R7中存放个位。要求将其转换为十六进制数，并在数码管上显示出来。

要求自己编写数码管显示子程序，使用尽量少的寄存器和内存单元。方便我们以后的实验使用。显式程序使用轮询方式不停刷新即可。

五、实验步骤

一、预习

参考附录三及其他辅助材料，学习8031汇编语言

二、程序录入

MCS51单片机汇编语言的基本格式比较简单，程序中可以使用通用寄存器或者内存单元进行计算。另外，单片机的程序没有退出到操作系统的概念，一般都是死循环程序。一个简单的计算1＋2的程序举例如下：

```
ORG 0000H ;复位起始地址
LJMP STRT ;中间地址保留给中断向量表
ORG 0040H ;程序实际起始地址
STRT: ... ;实际程序
MOV R1, #1 ;使用内部寄存器
MOV A, R1
MOV 40H,#2 ;使用内存单元
ADD A, 40H
LJMP STRT
END
```

七段字形的编码方式需要通过实验获得。这些编码作为程序中的常数，使用DB命令存放。数据使用程序存储器传送指令MOVC读取。端口输出通过数据存储器指令MOVX完成。

根据以上格式和说明，按照实验要求将程序录入。

注意在单片机程序里面，必须全使用大写，程序之间不能有空行，每行程序不能过长，另外标号只有前4个字符有效。

三、程序调试

程序调试的方法与单片机操作密切相关，参见附录及操作手册。

六、思考题

1、说明你的数码管显示子程序的调用方式，以及所影响的寄存器和内存数据单元。★

2、影响数码管显示亮度的因素有哪些，在程序中如何调整？

3、考虑使用中断方式改造数码管显示子程序。

4、说明MOVX和MOVC指令的区别。★

4、MCS51中有哪些可存取的单元，存取方式如何？它们之间的区别和联系有哪些？★

实验四 步进电机原理及应用

一、实验目的和要求

了解步进电机的工作原理，学习用单片机的步进电机控制系统的硬件设计方法，掌握定时器和中断系统的应用，熟悉单片机应用系统的设计与调试方法。

二、实验原理

见附录二。我们使用的单片机系统的频率是6M；步进电机转动一周需要48步。步进电机的四相分别接入P1的低四位。实验中使用的步进电机，依次循环定时输出01H，02H，04H，08H即可正向转动，如逆序输出上述控制字，即反向转动。转动速度取决于输出的时间间隔。

三、实验器材与线路

本实验的器材包括如下：

单片机实验仿真机
步进电机实验板
＋5V，＋12V电源

四、实验内容

编制MCS-51程序使步进电机以5转/分的速度正转2圈，停顿3秒后，以40转/分的速度持续反向旋转。本程序要求使用定时器中断来实现，不准使用程序延时的方式。

五、实验步骤

一、预习
参考辅助材料，学习8031汇编语言使用
参考附录二，了解本实验原理
二、系统连接
三、硬件调试

使用单片机的调试指令，向P1口依次写入正转的四个控制字，应使步进电机正转四步。向P1口依次写入反转的四个控制字，应使步进电机反转四步。

如上述调试出现不正常现象，应仔细检查连线和开关等，必要时找指导教师解决。**要注意由于电机的机械特性，相邻的两次输入之间必须存在一定的延时。所以直接使用程序连续写入四个控制字是不成功的。**

四、程序输入

本程序需要使用定时器定时（参见附录三－2.5），并使用中断来同步（参见附录三－2.7）。中断程序的典型例子如下：

```
ORG 0000H
LJMP STRT
ORG 000BH
LJMP TOIN ;中断向量表
ORG 0040H
STRT: ...;初始化
...
TOIN: ...;中断程序
...
RETI ;中断返回
END
```

五、程序调试及现象观测

用单步、断点、连续方式调试程序，观察状态指示灯及电机状态，检查运行结果。

六、思考题

1. 若将步进电机实验板的A、B、C、D相分别接到P1.2-P1.5，要求不变，程序需要如何修改？★
2. 如采用单双相控制，每次步进角度是多少，程序需要如何修改？★
3. 步进电机的转速取决于那些因素？有没有上、下限？
4. 如何改变步进电机的转向？★
5. 步进电机有那些规格参数，如何根据需要选择型号？

实验五 压力测量与精度分析

一、 实验目的和要求

学习压力测量的基本原理和方法，掌握模数转换器的使用。学习精度校正的基本方法。

二、 实验原理

实验中使用的A/D转换器是ICL7135，具体介绍参看附录二。原理框图参见附录六图1。ICL7135的时钟是由单片机的时钟经过48分频得到的，积分输出端BUSY接单片机INT0输入，启动转换端接高电平，处于连续工作方式。实验装置已经经过一定程度的校正，1g的重量约相当于A/D转换器的数值1。

我们使用单片机定时器的外部门控方式来测量BUSY信号的时间。转换为对应的A/D时钟脉冲的数目就是A/D转换器的输入值。BUSY信号接入单片机的INT0管脚。

三、 实验器材与线路

本实验的器材包括如下：

单片机实验仿真机

CCA－8413重量测量实验板

实验中的测量范围在0－15000g之间。

四、 实验内容

1. 编制单片机程序，测量出A/D转换器的输出值，显示在单片机数码管上。
2. 测量出零点误差，在程序中进行校正。
3. 测量不同的标准重量对应的测量值。建立表格并绘制出曲线。
4. 分析数据，计算和评价系统误差。

五、 实验步骤

一、预习
二、系统连接
三、硬件调试
四、确定标准点程序输入及调试
五、校正测量程序输入及调试

六、 实验说明

测量A/D转换器的输出值，就是测量INT0脚的高电平脉冲宽度，需要使用定时器的外部门控方式，并开启INT0中断（使用边沿触发方式），当中断到来时，说明此次测量已经完成。时钟计数除以4得到AD转换器的计数，减去10001（理由见原理），得到测量值。当测量值超过一定值后，会导致定时器溢出，在编程时需要注意这个问题。

在单片机上显示重量时，可以直接显示16进制。

七、 思考题

1、为什么时钟计数除以4得到AD转换器的计数？★
2、影响本实验测量精度的因素有那些？
3、被测重量达到多少时，会导致内部定时器溢出？说明计算过程。怎样编程来解决？★
4、你所使用的设备的线性参数是多少，说明计算方法。★
5、你所完成的实验的精度和准确度的范围在什么量级（百分比）？
6、如何使用10进制显示测量数据？

实验六　超声波测距

一、实验目的和要求

初步学习和掌握C51编程语言，了解相应的编程工具的使用。了解74HC164级联驱动数码管的使用。了解超声波测距的工作原理。

二、实验原理

C51学习参考附录四。

具体实验原理参见附录二。原理图参考附录六图2。

本实验使用的是STC89C516RD+的单片机实验板。本实验中，利用单片机的P1.0发出40kHz的方波脉冲，经过放大后输出到超声波发生器，产生超声波。超声波接受部分将接受到的回波信号经过放大、过滤等，最后输出到锁相电路中，产生INT0外部中断（边沿触发方式）。内部定时器进行准确计时，以通过计算得到准确距离。

测试距离应该在30cm~200cm之间。

三、实验器材与线路

本实验的器材包括如下：

新型单片机实验仿真机

超声波测距和温度测量实验板

四、实验内容

编制调试C51程序，通过超声波测量桌面和地面的距离，并且在74HC164级联驱动的数码管上显示。

五、实验步骤

一、预习

参考附录和其他参考资料，学习C51编程语言

参考附录，了解新型单片机的特点

参考附录，预习实验原理

二、硬件检查

三、程序输入

使用keil工具进行C51编程，使用方法类似VC6。

四、程序调试

通过ISP工具下载到实验板进行调试。

六． 思考题

1．被测量距离达到多少时，会导致内部定时器溢出？说明计算过程。怎样通过编程解决？

2. 怎样改进距离测量公式，以使得得到更精确的距离？

3. 影响74HC164驱动控制的数码管显示亮度的因素有哪些？应该怎样进行调整。

实验七　温度测量

一、实验目的和要求

学习DS18B20温度传感器的编程结构。了解温度测量的原理。加深C51编程语言的理解和学习。

二、实验原理

实验原理见附录五。原理框图见附录六图3。

本实验使用STC89C516RD+单片机实验板。单片机的P1.4与DS18B20的DQ引脚相连，进行数据和命令的传输。

单片机的P1.3连接热电阻。当P1.3为高电平时，加热热电阻。

三、实验器材与线路

本实验的器材包括如下：

新型单片机实验仿真机

超声波测距和温度测量实验板

四、实验内容

1. 编制程序测量当前教室的温度，并显示在74HC164级联控制驱动的数码管上。
2. 通过S1按键控制热电阻的加热与否，显示温度。

五、实验步骤

一．预习

参考附录和其他参考资料，学习C51编程语言

参考附录，预习DS18B20的编程结构

二．系统连接

三．程序录入

编程时注意DS18B20的时间要求，必须准确满足。根据实验原理附录中的流程图进行编程。

四．程序调式

六、思考题

1．进行精确的延时的程序有几种方法？各有什么优缺点？★
2． 参考其他资料，了解DS18B20的其他命令用法。

实验八　LCM使用

一、实验目的和要求

了解LCM的基本工作原理及编程方式。初步了解新型单片机的特点。

二、实验原理

具体实验原理参见附录五。原理框图参见附录六图4右半部分。
本实验使用STC12C5A16AD单片机实验板。

三、实验器材与线路

本实验的器材包括如下：
　新型单片机实验仿真机

四、实验内容

编制调试一程序，在LCM上显示实验人的名字。

五、实验步骤

六、预习
预习实验原理
了解新型单片机特点
设计汉字点阵代码（可以借助PCtoLCD2002字模软件）
七、系统连接
八、程序输入
程序提示：根据实验原理，编写出左半屏的命令写子程序，左半屏的数据写子程序。右半屏同样如此。剩下的程序可以调用这些子程序进行命令和数据操作。要计算每个字的显示位置并定位。
九、程序调试

六、思考题

1.如何在LCM屏幕上画上边框？
2.计算该型号LCM能够显示多少清晰汉字?
3.LCM的功耗取决于那些参数？

实验九　音频录制与播放

一、实验目的和要求

了解音频录制与播放的原理。掌握A/D转换和D/A转换。进一步掌握LCM的使用。学习新型单片机的特点。

二、实验原理

实验原理见附录五。原理框图参见附录六图4。

本实验使用STC12C5A16AD单片机实验板。A/D转换位于单片机内部。D/A转换使用DAC0832器件。音频录制时额，单片机的P1.0作为A/D转换的模拟输入通道，并将转换结果存储在单片机的EEPROM中。音频播放时，将存储在EEPROM中的数据取出，通过P0端口传递给DAC0832进行D/A转换，再经过音频调制，在扩音器中播放。

三、实验器材与线路

实验器材如下
PC机
新型单片机实验仿真机
音频处理实验板

四、实验内容

1. 通过S1按键控制录制声音，S2按键控制声音播放。
2. 在LCM上显示操作过程（例如录制声音时显示“正在录制中”）。

五、实验步骤

1．预习。
　预习新型单片机的特点，尤其A/D转换和EEPROM的使用。
　预习DAC0832的使用。
2．硬件调试。
3．编制、输入程序。

注意在重新录制声音时，应先对EEPROM进行擦除。注意声音录制和播放时采样率（即录制或播放时的转换间隔时间）。

4.程序调试

六、思考题

1. 讨论采样率的影响因素有哪些？及其对录制声音的清晰度影响。
2. 计算EEPROM针对某一采样率时录制声音的最大录制声音长度。★

附录一 使用CAD软件辅助电路板设计制作的基本方法

一. 电路原理图的设计方法

使用CAD软件辅助电路板设计，首先要根据设计目的，选择主要元件，画出草图，然后上机操作。

原理图的基本元素包括元件、连线、结点等。元件的基本属性包括元件的名称、元件的封装和元件的标号（Designator）等。软件设计者将常用的元件组织在若干个库中，如DEVICE库包括常用的分离元件、TTL库包括TTL系列的集成电路等。库中的元件包括元件的基本信息和管脚的说明及逻辑图符号，用户也可以自己定义库中没有的元件。管脚之间的电气连接是通过连线表示的。在原理图中，交叉的线是在逻辑和电气上无关的，除非使用结点（Junction）在交叉点连接。

用户在设计时在选定的图纸上按照设计放置（Place）元件和连线。放置元件时用户要先选定需要的库，然后在图纸上放置需要的元件。用户可以自由移动元件的位置和旋转。元件之间的连线是通过选择布线工具（Wiring Toolbar）,使用鼠标或键盘在电气上连接的两个管脚间连线。

在原理图画完之后要生成网络（Netlist）文件，以完成后续的操作。网络文件中包括所有使用的元件的说明和所有网络的说明。一个网络指在电气上彼此互连的所有管脚和连线。例如所有连接到一起的地线构成地线网络等。

软件一般还提供其他一些工具辅助设计和简化工作。如可以给一个网络命名，所有在图上没有连到一起，但有相同的网络名称的元素属于一个网络，就如同它们之间有线连接一样。还可以对设计按照通用的设计规范进行检查等。

二. 原理图辅助设计软件的基本操作

PROTEL FOR WINDOWS SCHEMATIC CAPTURE 1.5（以下简称SCHEMATIC EDITOR）是由Protel公司设计的原理图辅助设计软件。是在Protel For Dos 3.XX(3.12,3.16)基础上开发的，支持Windows环境和图形界面，增强了其中的一些功能

使用SCHEMATIC EDITOR的设计方法与上节的说明类似，界面的使用可以通过鼠标和菜单项操作，也可使用键盘辅助操作。下面的说明中以菜单为主。

软件的主界面左半部称为元件浏览器（Component Browser），在此处选择需要的元件库和元件。右面是设计图纸，可以在菜单项Options->Sheets中改变图纸的大小等选项。可以通过Zoom菜单改变图纸显示比例，也可使用快捷键PageUp（放大）PageDown（缩小）。

改变元件库通过按元件浏览器上方的Add/Remove按钮执行，同时可以有多个库被使用。常用的库有DEVICE.LIB（常用分离器件库）、D_TTL.LIB（与DOS软件兼容的TTL集成电路库）、D_INTEL.LIB（与DOS软件兼容的INTEL芯片集成电路库）等。

放置元件使用菜单中的Place->Part执行，也可直接在元件浏览器的Components In Library列表框选择。被选择的零件必须包括在当前库中。

元件放置后可以使用鼠标单击选中，然后用鼠标拖动来移动位置，也可在选中状态按空格键旋转90度，按Y键作Y轴映射，按X键作X轴映射。双击元件可以编辑元件属性。

使用Options->Wiring Toolbar开启连线工具箱，左上角是普通连线，选择此工具后，就可以用鼠标在图纸连线。第一次左键确定连线起始点，以后的左键确定连线的转折点或结束点，第一次右键结束此条线，第二次右键结束连线状态。**要特别注意连线与其要连接的管脚必须是点与点对接，而不能使连线覆盖管脚**。

在连线工具箱中还有Junction工具用来放置结点。此工具箱中还有电源工具，用来在图纸上放置电源部件（各种电压的电源和地）。可以选择几种符号之一，注意符号不表明其含义和电气性能，必须在其Net中指定是VCC或GND或其他。

删除一个元素的方法是使用菜单中的Edit->Delete选项，然后单击此元素，但要注意的是要在删除后按鼠标右键结束此状态（这一点在许多操作中都类似）。或者在选中一个元素后按Del键。每一步操作后可以使用Edit->Undo来取消。

可以使用Edir->Select来选中一个区域或一个网络。选中的元素使用其他的颜色来标志。

使用File->Annotate来对图中的所有元件进行编号，在放置元件时缺省为U？的形式，此命令将？用排序好的数字来替换。

使用File->Report产生各种报告，常用的有：Bill Of Materal产生图纸中使用的所有元件列表报告；Electrical Rules Check产生电气规范检查报告。

使用File->Create Netlist生成此图纸的Netlist文件，供以后使用。

三．印刷电路板图设计的基本概念

在电路原理图设计完成后，为制作电路板，要进行印刷电路板（PCB）图的设计。

制作印刷电路板中涉及的基本概念有层（Layer）、封装（Pattern）、焊点（Pad）、过孔（Via）、线（Track）。

电路板上按照不同用途分为若干层，层与层之间互不交叉。根据可以布线的层数可以分为单面板，双面板和多层板。常用的双面板包括Top Layer（正面）、Bottom Layer（背面）、Top Overlay（正面印字面）和Keep Out Layer（边界轮廓）等。

PCB图中也存在称为元件（Component）的概念，但此概念与原理图中确定电路性能的含义是完全不一样的，为避免造成误解，在PCB图中改称为封装。封装是指元件的几何尺寸和物理规格是如何定义的。包括元件的大小，管脚次序，管脚的大小和管脚间的距离等数据。不同的电路器件可以有相同的封装，同一种器件生产厂商也可能根据不同需要生产不同封装的产品。在软件中将常用的封装组织为库，用户也可以根据自己的需要手工编辑或增删封装。实验中常用元件的封装可以在附录四中找到。

焊点是用来插接元件的孔，也就是元件的各个管脚在板上的几何位置；过孔是连接不同面之间走线的桥梁。焊点和过孔的属性包括大小，形状（园、正方形、长方形）和内径。

由原理图制作PCB图实际上是将原理图的元件按照其封装排列在实际尺寸的电路板上，然后按照其网络结构进行布线。当然，在电路板上，同一层交叉的线就是连在一起的。为使所有的线不相交叉，对于复杂的电路板，需要有多层走线（目前最多可达10余层），在多层间用过孔连接。这个如何安排走线和过孔的过程称为布线（Route）。

设计PCB板的过程包括指定板的轮廓、读取网络文件、元件布局、布线、反复调整和检查几个步骤。其中反复的阶段是布局和布线。布局是指元件的位置排放方式，它是布线的基础；布线是在已排放好的布局上按照连线网络连接管脚。这两个阶段实质上是将一个图在一个或多个面上无交叉地重新排放的拓扑过程。对于复杂的电路板，完全由人工进行布线是极其繁琐困难和容易出错的，CAD软件在这一方面可以辅助人的工作，它可以根据网络和元件布局自动布线（Auto Route）。自动布线虽然可以减轻人的工作量，但由于算法的限制，也有一些缺点，如不一定完全布通、有些线不是很合理、不能通过调整布局来布线等，因此仍然需要人工调节。目前有些软件也支持自动布局。具体的执行步骤参看下一节。

四．印刷电路板图辅助设计软件的操作

PROTEL FOR WINDOWS PCB DESIGNER 1.0（以下简称PCB DESIGNER）是与SCHEMATIC EDITOR配套的产品，可以用来完成印刷电路板图的辅助设计。下面根据上一节所说的流程来介绍此软件的使用，首先说明常用的操作。

在电路板的设计和制作过程中，一般使用英制长度单位如英寸，换算关系为1英寸＝2.54厘米。在使用中，英寸的单位比较大，常用mil作单位，换算为1英寸＝1000mil。如我们常见的双列直插芯片的相邻两管脚间的距离为50mil，

在PCB DESIGNER中可以使用不同颜色同时显示多个Layer，在窗口下方的状态栏中显示当前光标的位置（以mil为单位），中间的列表框可以选择当前有效的层，右方是当前焊点的属性。

图纸显示比例的调节与SCHEMATIC EDITOR类似。手工放置封装、连线、焊点和过孔可以通过菜单命令Edit->Place中的选择完成，也可以使用工具栏中的相应工具。移动一个元素通过Edit->Move命令完成。

在设计流程中，首先使用File->New命令创建一个空白的图纸，一般这个图纸的显示比例比较大，要调整到适当的大小和位置。

根据需要在Keep Out Layer划出封闭的印刷电路板的几何形状和尺寸。然后使用NetList->Load调出由原理图生成的网络文件，此操作系统将给出完成信息，一般的出错原因是由于在原理图中**没有指定元件的封装或指定的封装没有找到**等。如果是这种错误要检查原理图。

网络文件调入后，图纸上排列了电路中的所有元件（其封装），元件间的连线使用虚线表示（称为ratsnest）。下一步进行布局，即将每个元件放到合适的位置，放置的依据首先要考虑实际制作的需求（如电源的位置等），二是要考虑布线时的方便。可以手工移动元件，也可以通过命令Auto->Auto Place来自动布局。总的原则和要求就是在允许的尺寸下将各个元件分开，并将连线较多的元件集中在一起。

布局完成后开始准备自动布线。先进行自动布线的设置，进入Auto->Setup Auto Route对话框中，确定电路板的布线面（我们常用两面）已被选中（缺省为正面布竖直线，背面布水平线），其他层已被选为不使用。在Default Variables区中设定布线网格（缺省为25mil）、线宽（缺省为12mil），过孔尺寸（缺省为50mil）。在Routing Passes区中指定附加的布线算法，常用选项的有预布线（Pre-Router）：它检查图纸中已有的连线，如果它符合网络中的某根线，认为已经布好。在Advanced区中的Maze是标准的布线算法，其中的Passes决定计算的次数，计算次数越多，布线成功率越大，当然计算时间也越长。

设置好布线参数后，使用Auto->Auto Route->All开始对整个网络进行布线，屏幕出现一个对话框，表示当前的状况。

自动布线完成后已布通网络的Ratsnest消失。如果仍有未布通的网络，需要根据情况进行重新布局或手工布线。

布线完全结束后，可以使用Netlist->Clearance Check检查线和焊点间是否有间距过近等情况（尤其是有手工布线时）。然后使用Netlist->Generate根据实际物理连线生成PCB图的网络文件；在SCHEMATIC EDITOR的File->Report->Netlist Compare中可以比较两个网络是否完全一致。最后输出图纸，或直接将软盘交给制版厂家完成电路板制作。

附录二 实验原理说明

1） 步进电机及控制

1．步进电机的工作原理

步进电机是工业控制及仪表中常用的控制元件之一，它的基本功能是将数字脉冲转换为角位移，从而可以实现对执行机构的精密控制。它还具有快速启停、精确定位和步进等特点。在数控机床、打印机等领域得到广泛应用。

常见的共磁路反应式步进电机的结构中,分为定子和转子两大部分。一个三相步进电机，共有六个大极，对称的大极形成一相控制绕组。在每个大极上，面向转子的部分分布多个等间距排列的小齿。电机的转子由软磁性材料制成，外部有与大极形状相似，间距相同的小齿。

在步进电机的设计中，如果在转子上排列40个小齿，则齿间距为360/40=9°。同样，每个大极上各有5个小齿，间距也是9°。由于各极中心线之间的夹角为120°，当A相磁极与转子的齿对齐时（称为对齿），其他绕组的磁极必定**不能**与转子的齿对齐（称为错齿）。例如A相磁极与0号齿对齐时，B相磁极中心线处应正对120/9号齿，相当于13齿向后多3度。

当绕组通电后，由通电的定子与转子之间的电磁力将推动转到最大磁导率的位置，并达到稳定状态。一般而言，在只有一个绕组通电的情况下，此时转子的齿必定和此绕组的齿相对齐。如上例的情况是A相通电，如此时突然变为只有B相通电，电磁力将迫使13号齿与B相中心线对齐，这样，转子就转动了3度，电机也因此转了一步。如果按一定次序轮流通电，电机将持续转动。

2．步进电机的控制方法

步进电机的运转是由脉冲信号控制的,只要按照一定次序对线圈通电,即可实现步进电机的转动控制。考虑到各种工作条件，常用的控制方法分别叙述如下：

单三拍工作方式

当通电方式为A→B→C→A循环时,称为单三拍工作方式。所谓“单”是指每次对一相通电；所谓“三相”指每换相三次，磁场就会旋转一周，同时转子转动一个齿距。如果需要反向旋转，则通电次序也要颠倒。

这种方式控制方法简单，通电电流小，最小步进角度为齿距的三分之一；但存在着高频性能差、转矩小、易产生振荡的缺点。

双三拍工作方式

如果每次都是两相同时通电，电流切换三次，磁场旋转一周，电机前进一个齿距，称双三拍工作方式。即通电方式为Ａ Ｂ→Ｂ Ｃ→Ｃ Ａ→Ａ Ｂ循环。

由于每次都是两相通电，在这种方式下，转子齿不能与定子齿相对齐。如Ａ Ｂ相通电时，转子既不能与Ａ相齿对齐，也不能与Ｂ相齿对齐，而只能停在Ａ Ｂ相之间的中间位置，转子齿与Ａ相正差1.5度，与Ｂ相负差1.5度。同理可以推出，在双三拍工作方式中，每一步都是不对齿的。这样，它的步进位置不是稳定位置，容易引起振荡。

双三拍工作方式的电流比单三拍工作方式大，约为其1.5倍。因此其转动力矩大。另外这种方式不易产生失步。

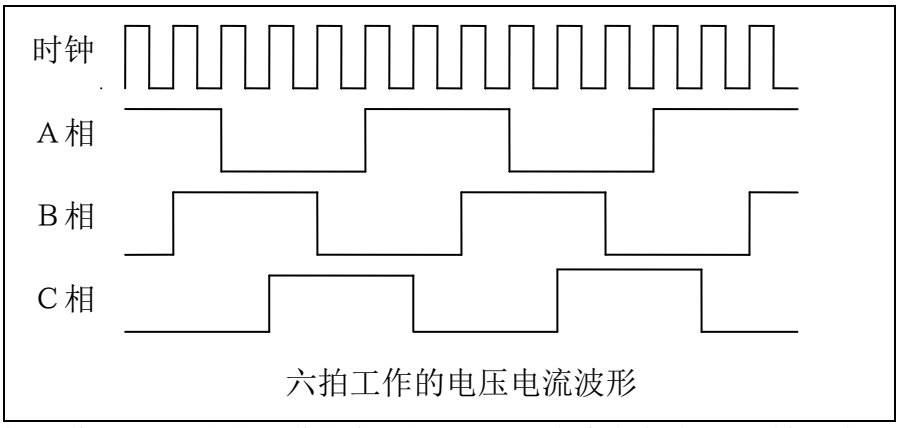
六拍工作方式

在六拍工作方式中，通电方式如下：

Ａ→Ａ Ｂ→Ｂ→Ｂ Ｃ→Ｃ→Ｃ Ａ→Ａ循环。

这时，控制电流切换六次，磁场旋转一周，电机前进一个齿距。这样每次前进的精度为齿距的六分之一，精度提高一倍。

六拍工作方式时的电压电流如图所示：



六拍工作的电压电流波形

六拍工作方式的综合工作指标最好，它具有高频性能好、转矩大、振荡小、且具有较好的平滑性。尤其是采用单片机等数字方式时，不需要复杂的硬件构成脉冲分配器，更具有实用价值。

步进电机的转动速度的控制是通过改变时钟的周期，即改变换相的频率来实现的；它转过的角度由输入的脉冲个数决定。

在我们实验中使用的步进电机是一个四相电机，其基本原理和上面介绍的类似。电机共有A、B、C、D四组绕组。类似地，也有三种控制方法，通电方式分别如下：

单相方式：Ａ→Ｂ→Ｃ→Ｄ→Ａ循环。

双相方式：Ａ Ｂ→Ｂ Ｃ→Ｃ Ｄ→Ｄ Ａ→Ａ Ｂ循环。

单双相方式：A→A B→B→B C→C→C D→D→D A→A循环。

3．步进电机的主要特性

步进电机的主要参数有电机的最大工作电压、最小启动电压、最大允许功耗及工作频率等。

步进电机的主要性能指标是频率特性曲线。曲线的纵坐标是转动力矩，横坐标是转动频率。一般电机的转动力矩随频率升高而下降。

电机的频率特性和许多因素有关，包括制造时决定的物理、几何参数和使用的方式。通过加大控制电压和降低线圈的时间常数的方法可以提高电机的最高工作频率。

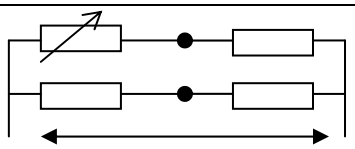
步进电机在使用时可能出现振荡、失步和阻尼等不希望出现的现象，要在实际设计中加以考虑。

2） 压力测量

压力是工程中常见的被测物理量之一，电阻应变片是一种电阻式的力传感器件，可以用于测量静态的或低频的交变应力，它具有体积小，寿命长，价格低等特点，在要求不高的场合得到广泛应用。

在金属丝的两端加以拉力后，将产生机械形变，使金属丝的长度变长，截面积相应变小，使其电阻发生相应的变化。这种导体的电阻随应力变化而变化的现象称为应变电阻效应。

电阻值的变化可以通过电桥来转换为电压量测量。当使用四个阻值相同的电阻接成桥状电路时，中间点的电压差为零。如果其中一个电阻是可随外界条件变化的电阻，它的阻值的变化引起中间点的电压差。在一定的幅度内，此差值与电阻变化率成正比。



设加在桥两端电压为U，则桥中间点的电压差为U乘以电阻变化率除以4

3） A/D转换器

A/D转换器是将模拟量转换为数字量的器件。模拟量一般是电压信号。对于电流信号可以使用运算放大器转化为电压信号。对于其他连续变化的信号量（如声、光、压力、温度、湿度等）可以通过合适的传感器转换为电信号量后进行测量。

衡量A/D的主要性能参数为：

分辨率，即输出的数字量变化一个最小刻度所对应的输入模拟量的变化值。

满刻度误差，即输出全1时输入电压与理想输入值之差。

转换速率。

转换精度。

量程，即最大输入电压。

单电压还是双电压输入。

A/D的种类很多，根据转换原理可分为双积分方式、逐次逼近式等。根据接口方式分为并行式、串行式等。

双积分A/D转换器由电子开关、积分器、比较器和控制逻辑等构成。它是将未知电压Vx转换为时间值来间接测量的。

在进行一次A/D转换时，开关先把Vx采样输入到积分器，积分器从零伏开始进行固定时间T的正向积分。然后开关将与Vx极性相反的基准电压V_{REF}输入到积分器进行反向积分。直到输出为零时停止积分。通过计数反向积分时间就可以计算出输入电压Vx在时间T内的平均值对应的数字量。

由于这种A/D要经历正、反两次积分，故转换速度较慢。

ICL7135是一种常用的四位半的双积分式A/D转换器。分辨率为50ppm（1/20000），零点误差小于10μV，零点漂移小于10μV/℃，典型输入电流1pA。可以使用单电压或双电压使用。最大输入电压15V。器件的最大时钟频率为2MHz。

ICL7135提供BCD码输出和脉冲宽度输出两种方式。器件的BUSY输出在正向积分开始时变为高电平，在反向积分过零后的第一个时钟脉冲处才变为低电平。正向积分时间为10001个时钟脉冲。BUSY信号的高电平时间对应的时钟脉冲个数减去10001即可得到对应的数字输出。

4） 测量系统的精度分析

在测量某种连续物理量时，存在着多种误差来源。在典型的应用电路中，主要包括：

1.传感器的元件特异性。

2.运算放大器的零点漂移。

3.运算放大器的共模电压。

4.某些关键元件（如电阻、电容）的实际值与标称值不符。

5.A/D转换器的误差。

在很多应用中，选用的传感器输出的电气值（电压/电流）与物理值在理论上是成线性关系的，或者在允许的误差范围内可以使用线性关系逼近。这样，测量值与实际值的关系可以使用以下公式来表示：

$$F(y) = y_0 + k \cdot y + d(y)$$

其中，y_0表示总共的零点漂移；k是线性比例系数；d(y)项表示其他的非线性关系。

要准确地测量出被测物理量，就要确定出这个公式中的几个参数，特别是零点漂移和比例系数。这包括几个步骤：在设计过程中通过选择元件参数初步确定；电路完成后通过调节其中可调节元件（如变阻器）进行调节；通过软件进行动态调节和变换。下面着重介绍软件调节的方法，这个过程也可以称为定标的过程。

定标需要对多个标准被测量进行测量，得到对应的测量值。然后对这些数据进行分析（如在坐标纸上画出对应的曲线），确定适当的系统参数。这实际上是一个数据拟合和插值的过程。

当系统的非线性项影响很小时，测量数据构成一条标准的直线关系，从中求解出直线的斜率和截距就可以得到系统的参数。

当系统的非线性项在整个量程中有一定影响，但与被测量相比没有明显的对应关系时，测量数据基本上均匀分布在一条直线的两侧。仍然能够使用线性关系来逼近。可以使用最小二乘法等方法确定这条直线的斜率和截距。

当系统中存在明显的非线性项时，可以考虑通过曲线拟合的方式求出实际的公式关系。当精度要求不是非常高的情况下，可以使用分段直线拟合的方式来进行逼近。这种情况下，确定几个关键点之后，对于每两个关键点之间的曲线使用线段来拟合，解出所有的参数后，存储为一个列表。在实际测量时，首先确定输入的范围，通过查表确定对应的参数计算出实际值。

5） 超声波测距

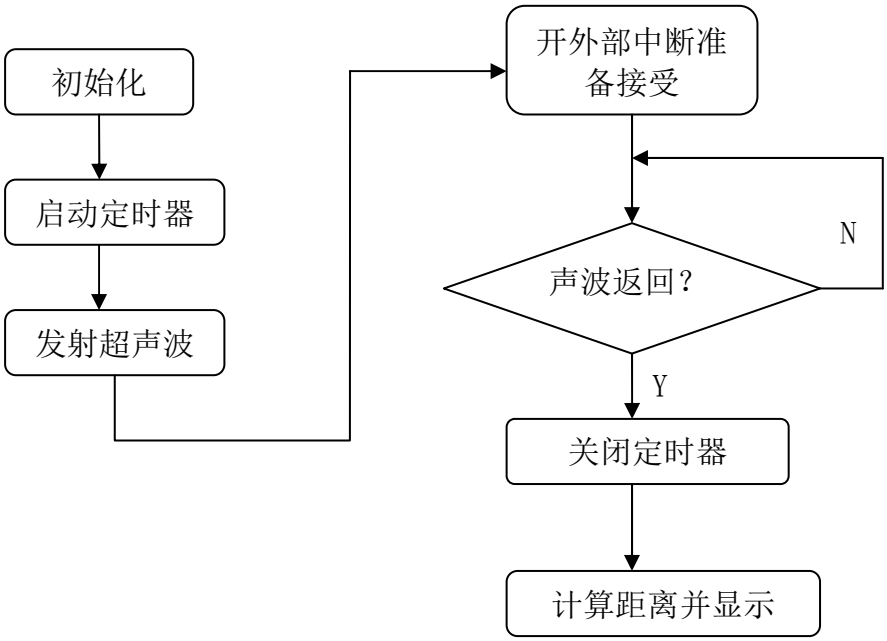
由于超声波指向性强，能量消耗缓慢，在介质中传播的距离较远，因而超声波经常用于距离的测量，如测距仪和物位测量仪等都可以通过超声波来实现。

利用超声波在空气中的传播速度为已知，测量声波在发射后遇到障碍物反射回来的时间，根据发射和接收的时间差计算出发射点到障碍物的实际距离。理想情况下，超声波在空气中的传播速度为340m/s（不考虑温度等因素），根据计时器记录的时间t，就可以计算出发射点距障碍物的距离(s)，即：s=340t/2。

超声波测距分为两部分：超声波发生器部分和超声波接受部分。超声波发生器可以分为两大类：一 类是用电气方式产生超声波，一类是用机械方式产生超声波。本实验中采用压电式发生器。压电式超声波发生器实际上是利用压电晶体的谐振来工作的。它有两个压电晶片和一个共振板。当它的两极外加脉冲信号，其频率等于压电晶片的固有振荡频率时，压电晶片将会发生共振，并带动共振板振动，便产生超声波。反之，如果两电极间未外加电压，当共振板接收到超声波时，将压迫压电晶片作振动，将机械能转换为电信号，这时它就成为超声波接收器了。

实验产生误差的原因有时间误差和超声波传播速度误差两大方面。其中后者影响较大，超声波的传播速度受空气的密度所影响，空气的密度越高则超声波的传播速度就越快，而空气的密度又与温度有着密切的关系。限制系统的最大可测距离存在四个因素：超声波的幅度,反射面的质地,反射面和入射声波之间的夹角以及接收器的灵敏度。

超声波测距的流程图如下所示：


初始化
启动定时器
发射超声波
开外部中断准备接受
声波返回?
N
Y
关闭定时器
计算距离并显示

附录三　MCS-51系统结构与指令系统

第一章 概述与历史

随着半导体技术的发展，能够在一个芯片上制造出上百万个晶体管，于是出现了由大规模集成电路组成的中央处理器，以及大容量的集成电路半导体存储器，通用和专用的输入输出接口电路，由这些大规模集成电路组成各种类型的微型计算机（Microcomputer）。

从七十年代中期开始，出现了以一个大规模集成电路为主组成的微型计算机：单片微型计算机，简称为单片机（single chip microcomputer）。由于单片机面向控制，特别适合于控制领域，因而又名微控制器（Microcontroller）。

从1976年起，Intel公司开始生产MCS系列单片机。1976年推出MCS－48系列8位单片机，其典型产品为8048。在1980年推出MCS－51系列单片机，这是一个高性能的8位单片机，在各个方面性能都得到大大增强。MCS－51系列的典型产品为8051/8031，其内部资源为：

8位CPU

128字节RAM数据存贮器

32位I/O线

两个16位的定时器/计数器

一个全双工异步串行口

五个中断源，两个中断优先级

64K程序存储器空间

64K外部数据存贮器空间

片内振荡器，频率范围为1.2MHz到12MHz

Intel公司于1984年推出16位MCS－96系列单片机，它的典型产品为8397BH。

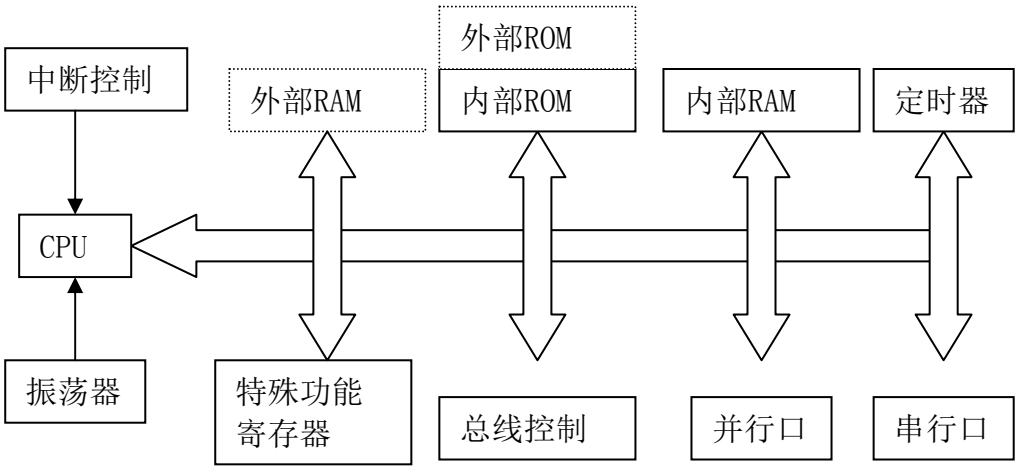
虽然单片机产生已经很长时间，但由于在智能仪表、家用电器等实时控制和分布式控制领域需要这种结构简单，控制灵活，并不需要很快的计算速度和存储容量的计算机，因此在计算机控制等领域单片机仍然具有不可替代的地位。

第二章 MCS－51系列单片机系统结构

自从MCS－51单片机问世以来，该系列已经发展到几十种型号，它们都具有相同的内部核心结构和基本相同的外部引脚，其中以8031为代表产品。我们以MCS－51表示这些以8031为内核的单片机。

2.1 总体结构

MCS－51是一个集成了CPU和常用的外围器件的芯片，结构简图如下。引脚说明见附录。它们具有的硬件资源在第一章中已经说明。



2.2 中央处理器CPU

2.2.1 运算器

运算器主要包括ALU、累加器、B寄存器、程序状态寄存器PSW，十进制调整电路和布尔处理器等。8031可以执行的运算功能包括：

算术运算：加减乘除；

逻辑运算：与、或、异或、非；

布尔运算：置1、置0、取反；

增量、减量运算；

十进制调整；

数据传送。

2.2.2 控制器

控制器包括时钟发生器、定时控制逻辑、复位电路、指令译码器、指令寄存器、程序计数器PC、数据指针DPTR、堆栈指针等部件。

① 时钟电路

时钟电路是计算机的心脏，它控制着计算机的工作节奏。MCS－51单片机内有一个由反相器构成的振荡器，引脚XTAL1是反相器的输入端，XTAL2是反相器的输出端。

MCS－51的时钟可以利用其内部的振荡器产生，只要在引脚XTAL1和XTAL2上外接定时反馈电路，内部振荡器便自激振荡，产生时钟输出到内部的定时控制逻辑。定时反馈电路一般为石英晶振和电容组成的并联电路。振荡频率主要由晶振的频率决定，一般在0.5MHz到16MHz之间，典型值为12MHz和11.0592MHz。电容的主要作用是帮助振荡器起振，对振荡频率也有微小的影响。典型值为30pF。MCS－51的工作时钟也可以由外部提供。

② 复位电路

计算机在启动时都需要复位，使CPU和系统中的其他部件都处于一个确定的初始状态，并从这个状态开始工作。

MCS－51的RST引脚就是复位信号施密特输入端。当RST端保持16个时钟周期以上的高电平时，就可以使CPU复位，内部寄存器处于确定的初始状态，当RST为低电平后，退出复位，系统从初始状态开始工作。程序从0000H处开始执行，一般须在此处放一条长转移指令转移的真正的程序入口（原因见中断一节）。

复位后的内部寄存器状态

寄存器	内容	寄存器	内容
PC	0000H	TMOD	00H
ACC	00H	TCON	00H
B	00H	TH0	00H
PSW	00H	TL0	00H
SP	07H	TH1	00H
DPTR	0000H	TL1	00H
P0－P3	0FFH	SCON	00H
IP	00H	SBUF	不定
IE	00H	PC	0000H

在RST引脚外只要接一个电阻和电容构成的复位电路就能实现上电自动复位。对应CHMOS器件只要接一个电容即可。复位电路在上电时的高电平时间应包括电源电压上升时间（约10ms）和内部复位时间（12个时钟周期），为可靠起见，一般为20ms，这个时间取决于电路的时间常数RC，当时钟周期为12MHz时，典型值为R=10K，C=10μF。

除上电复位以外，常常需要手工复位，将一个常开按钮并联于上电复位电路，开关闭合时就能实现器件复位。

在单片机应用系统中，单片机的外围电路也需要和单片机同步复位，保证单片机对外围电路有效初始化。由于单片机的RST端是施密特输入，而

一般接口电路是TTL电平输入，需要把RST电路信号进行整形后转换为系统复位信号输出到各外围电路。同时应注意有些器件的复位引脚是低电平有效的，需要逻辑转换。

③ CPU定时

CPU取出一条指令到该指令执行完成的时间为指令周期。指令周期是以机器周期为单位的。MCS－51的一个机器周期由六个状态构成（S1－S6），每一个状态由两个时钟周期构成（P1，P2）。因此一个机器周期由12个时钟周期组成，若晶振频率为12MHz，则一个机器周期为1微秒。

大多数MCS－51的指令执行时间为一个机器周期，小部分指令为2个机器周期，只有乘除法需要4个机器周期。指令需要的指令周期在指令表中有详细说明。通过计算指令的执行周期可以精确控制程序的执行时间。

2.3 存储器组织

MCS－51有五个独立的寻址空间。

64K字节程序存储器空间（0－0FFFFH）

64K外部数据存储器空间（0－0FFFFH）

256字节内部RAM空间（0－0FFH）

256位寻址空间（0－0FFH）

工作寄存器区

这些寻址空间中，工作寄存器区重合在内部RAM的前128字节空间中，后128字节是内部特殊功能寄存器（SFR）空间，位寻址区的前128个地址重合在内部RAM中，后128个地址重合在SFR中的一部分寄存器中。MCS－51系列中不同型号的单片机，特殊功能寄存器的定义和使用不完全一致。

2.3.1 程序存储器

MCS－51系列的单片机有些型号在内部有4K字节的程序存储器（如8051和8751），有些有8K或16K内部程序存储器（如83C51），有些则没有（如8031）。不论哪种型号的单片机，都可以通过外部存储器扩展到64K的程序空间。

对于内部有程序存储器的单片机，如果引脚EA接VCC，则程序计数器PC的值在内部寻址空间范围之内时，CPU从内部的程序存储器读取指令；否则访问外部的程序存储器。如果EA接地，则总是从外部程序存储器读取指令。对于内部没有程序存储器的单片机，EA必须接地，只能从外部读取指令。

单片机外部扩展的程序存储器一般为EPROM（紫外线可擦除电可编程的只读存储器），引脚PSEN（低有效）输出外部程序存储器的选通信号。

2.3.2 外部数据存储器

MCS－51的外部程序存储器和数据存储器是分离的，对于CPU来说，程序存储器是只读的，通过PSEN信号控制；数据存储器是可读写的，通过RD和WR信号控制。在程序存储器中也可以固化一些数据（如预先放置的表格），

但读取这些数据的指令与读取数据存储器的指令也是不同的。这一点是和一般的微机系统不一样的。这是由于单片机应用的领域，程序都是固定的，为了防止可能的对程序的破坏而采取的一种保护措施。

在一般情况下，单片机都不需要很多的数据存储器，这时内部的128字节RAM基本是够用的。在有些情况下，需要大量的数据存储器，这时可以外接最大直至64K的外部数据存储器。

外部数据存储器的寻址空间也可以直接作为扩展I/O口的寻址空间使用，对此CPU使用相同的操作指令来读写。在简单的扩展中，常使用位选的方式来简单扩展端口。

2.3.3 内部RAM数据存储器

MCS－51单片机的内部RAM的寻址空间为256字节，实际提供给用户的通用RAM空间一般为前128字节。内部RAM的不同地址区域从功能和用途可以区分为：内部工作寄存器区，位寻址区，堆栈和通用数据存储区。

内部RAM的0－1FH为4组工作寄存器区，每个区有8个工作寄存器（R0－R7）。在同一时刻，只能使用一组工作寄存器，这是通过程序状态字PSW的地3，4位来控制的。例如当此两位为00时，使用第0组工作寄存器，对应于00H到07H的内部RAM空间。也就是说，这时指令中使用R0与直接使用00单元是等价的，不过使用工作寄存器的指令简单，且执行快。

程序通过修改PSW的这两位，就可以选择一个工作寄存器区，这个特性提高了MCS－51上下文切换的速度，对于提高CPU响应中断的速度和现场保护与恢复是很有利的。

内部RAM的20H－2FH为位寻址区域，这16个单元的每一位都对应一个位地址，占据位地址空间的0－7FH。每一位都可以独立置位、清除、取反等操作，也可以作为条件转移的条件使用。这个特性对应单片机应用于开关量控制和判断中是很有用处的。在SFR中的地址为8的整数倍的寄存器也可以位寻址，这样可以方便控制SFR中的某一位。

在中断和子程序调用中都需要堆栈。MCS－51的堆栈理论上可以设置在内部RAM的任意区域，但由于0－1FH和20－2FH区域有上面说的特殊功能，因此一般设置在30H以后。堆栈的栈顶位置由堆栈指针寄存器SP指出，SP在复位后为07H，一般在程序初始化阶段将其调整到需要的位置。堆栈是向上增长的，也就是从当前栈顶位置向地址增加的位置扩展，而这种扩展没有限制，在实际编程中要注意留出足够的区域以免发生冲突。

在内部RAM中，所有的单元都可以作为通用的数据存储器使用，存放输入的数据或计算的中间结果等。如果程序不使用其他组的工作寄存器、位寻址单元等，这些地址都可以自由使用。

2.3.4 特殊功能寄存器SFR

MCS-51内部的I/O口锁存器及定时器、串行口、中断等各种控制寄存器和状态寄存器等都称为特殊功能寄存器。它们控制着各种CPU周边设备，保存着它们的状态。SFR离散地分布在80H－0FFH的特殊功能寄存器地址空间中。不同型号的单片机内部I/O功能有所不同，实际存在的特殊功能寄存器数量差别较大。特殊功能寄存器空间中有些单元是空着的，在有些控制寄存器中的某些位也没有定义。这些单元是为了以后新型的单片机保留的，为了软件与以后的新型号兼容，用户程序不要对这些空单元进行操作。

特殊功能寄存器中地址为8的倍数的寄存器可以进行位寻址，它们的位地址位于80H到0FFH的地址空间。它们的位地址恰好等于SFR的地址加上位的次序（如PSW第7位CY的位地址是PSW的地址0D0H加上7，即0D7H）。

2.4 I/O口

MCS－51单片机有4个双向8位输入输出口P0－P3口。每一个口由口锁存器、输出缓冲器、输入缓冲器组成。这些位的复位后的状态都是1。其中P0口是三态双向I/O口，P1、P2、P3口内部有提升电阻，称为准双向口。

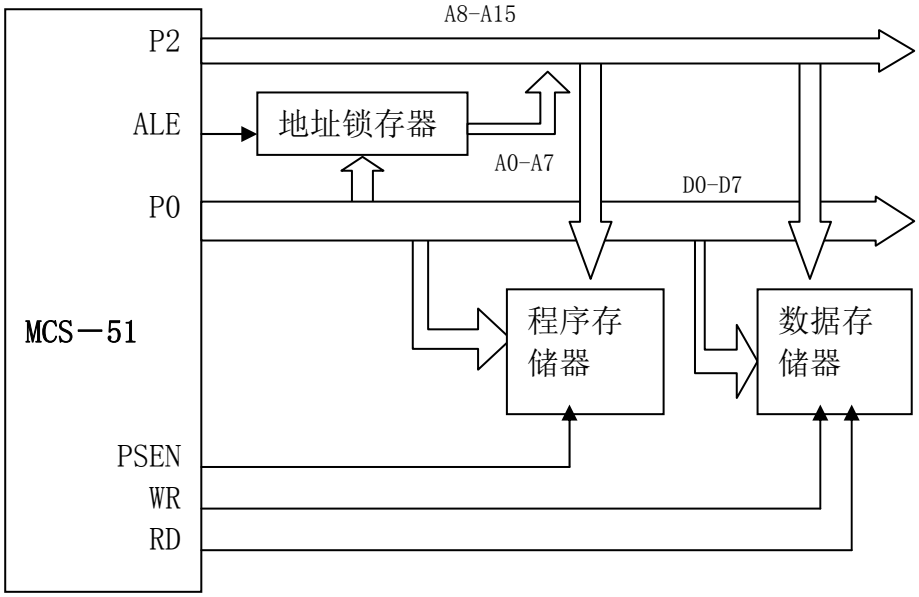
P1口为准双向口。它的每一位都可以分别定义为输入或输出线。作为输出时，程序直接向P1口的该位输出0或1，外部电路就锁存为低或高电平。作为输入端使用时，必须向其输出1，读取时，此位的状态就由外电路决定。

P3口为多功能口，用户可以使用系统预设的功能，这时必须向其输出1；也可以作为普通I/O口使用，这时使用方法与P1口一致。当系统定义的功能如下表所示：

P3口系统功能定义		
P3.0	RXD	串行输入口
P3.1	TXD	串行输出口
P3.2	INT0	外部中断0输入
P3.3	INT1	外部中断1输入
P3.4	T0	计数器0外部输入
P3.5	T1	计数器1外部输入
P3.6	WR	外部数据存储器写
P3.7	RD	外部数据存储器读

P2口也有两种功能，一种功能是作为外接程序/数据存储器的地址高位线输出，一种是普通的I/O口。对于内部没有程序存储器或需要外接的应用，P2口就不能作为输入输出口使用。

P0口为三态双向I/O口。和P2口类似，当外接程序/数据存储器时，必须作为地址/数据口使用，否则也可以作为普通输入输出口使用。当P0口作为地址/数据口使用时，在机器周期的前半时刻输出程序地址（或外部数据地址）的地址低8位，后半部分输入输出数据。这是通过ALE线来控制的。典型的总线扩展方法如图所示。



2.5 定时器/计数器

由于在绝大多数的应用中都需要使用定时器/计数器，在MCS－51中设立了两个16位定时器/计数器T0，T1，在某些型号中还有第三个定时器/计数器T2。

和定时器/计数器相关的特殊功能寄存器有以下：TH0，TL0，TH1，TL1，TMOD，TCON。通过对它们的设置和读写就可以控制使用定时器/计数器。

TH0，TL0为T0的16位计数的高8位和低8位；TH1，TL1为T1的16位计数的高8位和低8位；TMOD是方式寄存器；TCON是状态和控制寄存器。

2.5.1 方式寄存器TMOD

TMOD的格式如下，高4位和低4位分别控制T1和T0，它们的含义是完全相同的：

D7	D6	D5	D4	D3	D2	D1	D0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
T1方式字段				T0方式字段			

工作方式选择位M1M0确定了定时器的工作方式，其具体含义稍后详述。

定时计数选择位C/T＝0时为定时方式。这时，以振荡器的12分频信号作为计数信号，也就是每一个机器周期定时器内的值加1。若晶振为12MHz，则定时器的计数频率为1MHz。定时器从初值开始加1到溢出所需的时间是固定的，也就是（$2^{定时器最大位数}$－定时器初值）×定时周期，所以称为定时方式。

C/T＝1为外部计数方式，这种方式采用外部引脚（T0为P3.4，T1为P3.5）的输入脉冲作为计数脉冲。内部硬件在每一个机器周期的S5P2采样外部引脚的状态，当上一个周期为高电平，此周期为低电平，也就是发生负跳变时计数器加1。外部计数事件的最高计数频率为晶振频率的1/24，外部输入脉冲的高电平和低电平时间必须持续一个机器周期以上。计数器的用途通常是为了测量外部脉冲的周期、频率等。

门控位GATE为1时，定时器的计数受外部输入电平的控制（INT0控制T0，INT1控制T1），如外部控制电平为0，定时器不计数。GATE为0时定时器不受外部控制。这种方式可以使定时器的启动与停止受外部电路控制。

2.5.2 控制寄存器TCON

控制寄存器TCON的高4位存放定时器的运行控制位和溢出标志位，低4位存放外部中断控制位（详见中断一节），格式如下：

D7	D6	D5	D4	D3	D2	D1	D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

运行控制位TR0和TR1分别控制两个定时器是否允许计数。此位由软件置位和清零。当GATE为0时，TR为1时允许计数；当GATE为1时，只有TR为1且外部控制电平为高时，定时器才计数。

溢出标志位TF0和TF1标志定时器是否已经计数到最大值，发生溢出。这一位可以由程序来查询和清零。如果CPU响应定时器中断则此位为1时，发生定时器中断，在中断响应时由硬件清零。

2.5.3 定时器的工作方式

定时器有4种工作方式，其中方式3只适用于T0。下面详细讨论这4种工作方式。

当方式控制字M1M0为00时处于方式0。此时定时器为一13位的计数器，由TL的低5位和TH的8位构成，当13位计数溢出时置位TF。

M1M0为01时为方式1，方式1与方式0的区别仅在于它是16的定时/计数器，由TH和TL联合构成。

方式2，即M1M0＝10是自动恢复初值的8位计数器。这时TL作为8位计数器，TH作为计数初值寄存器。这种方式下，定时器发生溢出后自动恢复为TH中的初值，不需CPU干预，因此定时更加精确。方式2经常作为串行口的波特率发生器使用（见下节）。

当M1M0＝11时为方式3，它只适用于T0，若T1被设置为方式3时，它将停止计数。在此种方式下，T0被分为两个独立的8位计数器TL0和TH0。TL0使用T0的所有状态和控制位，可以作为定时器或计数器使用。TH0被固定为一个8位定时器方式，使用T1的状态控制位TR1和TF1。TR1为1时允许TH0计数，当TH0计数溢出时置位标志TF1。一般来说，当T1用作串行口的波特率发生器时（详见串行接口一节），T0才在需要的情况下使用方式3，以增加一个计数器，这时T1在计数溢出时不置位TF1。

2.6 串行接口

CPU与外部的接口方式有并行和串行两种。并行通信中信息传输线的位数和传送的数据位数相等，通信数据快，适合于近距离通信。串行通信仅需要一个数据线，一根地线（全双工需要发送和接收两根数据线），通信成本小，速度相对慢，适合于远距离通信。

串行通信有两种方式：同步通信和异步通信。

异步通信方式是按字符传送的，在字符前有低电平的起始位，在后面有高电平的停止位，字符之间没有固定的间隔长度。这种方式通信效率比较低，但实现比较简单。

同步通信是按数据块传送的，在数据块之前有特定的同步字符，后面有校验字符。同步通信的字符之间没有间隔，通信效率比较高。

在串行通信时，通信的双方必须事先商定每秒传送的数据位数，称之为波特率（Baud Rate），已经其他一些设置，如奇偶校验位、停止位等。

MCS－51具有内置的全双工异步串行接口，可以同时发送和接收数据。在某些型号中还增加了通信功能更强的串行接口。

2.6.1 串行接口的构成和特性

在MCS－51的外部引脚中，由P3.0（RXD）和P3.1（TXD）作为串行接口的输入和输出线。在方式0时，RXD作为数据输入输出线，TXD为时钟输出线。

串行口的内部有数据接收缓冲器和数据发送缓冲器，数据接收缓冲器只能读出不能写入，数据发送缓冲器只能写入不能读出，这两个数据缓冲器都用符号SBUF表示，地址是99H。

特殊功能寄存器SCON存放串行口的状态和控制信息，串行口用定时器T1作为波特率发生器，特殊功能寄存器PCON的最高位SMOD为串行口波特率的倍率控制位。

SCON的地址为98H，具有位寻址功能。其格式如下：

D7	D6	D5	D4	D3	D2	D1	D0
SM1	SM0	SM2	REN	TB8	RB8	TI	RI

由SM1和SM0组成串行口的方式控制位，分别对应方式0到3。SM2位为允许多机通信控制位。REN为允许串行接收位，由程序置位来允许接收。在方式2和3中，TB8为发送的第9位数据，RB8为接收的第9位数据。对于方式1，

RB8是接收的停止位。TI是发送中断标志，由硬件在方式0串行发送第8位结束时置位，或在其他方式下发送停止位时置位，必须由软件清零。RI是接收中断标志，由硬件在方式0接收到第8位结束时置位，或在其他方式下接收到停止位时置位，必须由软件清零。

当PCON的最高位SMOD为1时使波特率加倍。

2.6.2 串行接口的工作方式

① 方式0

方式0是外接移位寄存器的工作方式，用于扩展I/O接口。在这种方式下，数据由RXD串行输入或输出，由TXD端输出移位脉冲。波特率固定为振荡器的1/12。

当用作输出时，一般外接串行输入并行输出移位寄存器（如74LS164等），移位寄存器的串行数据端接RXD，时钟端接TXD。当CPU对SBUF写入数据后，就启动了串行口的输出。内部的定时逻辑从下一个机器周期开始，在每个机器周期的S6P2阶段使内部的移位寄存器右移1位送入RXD，左边移入0（也就是说最低位最先移出）；而TXD在S3、S4、S5期间为0，S6、S1、S2期间为1。当8位都移出后，置位发送中断标志TI，完成了一个字节的发送。

当用作输入时，一般外接并行输入串行输出移位寄存器（如74LS166等），移位寄存器的串行数据端接RXD，时钟端接TXD。当REN＝1，RI＝0时开始启动串行口接收。TXD引脚在每个机器周期跳变一次，使外部数据通过RXD输入到内部移位寄存器中。当8次移位完成后，数据写入SBUF，停止输出移位脉冲，置位接收中断标志RI，完成一个字节的输入。

②方式1

方式1是一种8位异步通信口方式。TXD为数据输出线，RXD为数据输入线。传送一个字节的格式为：前面一位低电平的起始位，中间从低位到高位的8位数据，后面一位高电平的停止位，共计10位。

在使用方式1输出时，当CPU向SBUF写入一个数据，就启动串行口发送。内部的移位寄存器根据波特率将数据按照帧格式不断输出到TXD，当输出结束后，置位发送中断标志TI

在REN＝1时，就允许串行口接收。接收器以所选波特率的16倍的速率采样RXD端的电平，当检测到RXD端出现负跳变时，就启动接收逻辑。接收器把接收一位的时间（即波特率的倒数）16等分，在每位时间的第7、8、9三个状态检测RXD的电平，并取其中的至少两个相同的值作为数据输入，这样可以防止干扰。如果第一位接收到的不是0，则起始位无效，重新进入等待起始位状态。9次接收之后，将数据装入SBUF和RB8（停止位），并置位RI。如接收的最后一位不是1或RI＝1，则本次接收的数据无效，重新等待。

③ 方式2和方式3

在这两种方式下，串行口是一个9位的异步通信口。区别在于方式2的波特率固定为振荡器频率的1/64或1/32，而方式3的波特率由T1的溢出速率决定。

在这两种方式下，一帧信息包括11位，数据位为9位，其余和方式1一致。附加的第9位数据最后发送或接收，它在发送时为SCON的TB8，接收时存到SCON的RB8。

在这两种方式下，发送的过程与方式1完全一致，只是多发送一位而已。接收的过程也大致类似，区别在于接收到数据后，除了判断RI标志外还要判断本机SCON的SM2标志。如果SM2＝0则接收数据（将接收的数据送入SBUF和RB8，置位RI）；当SM2＝1时，只有第9位数据为1时才接收数据，否则就丢失数据，不置位RI，等待下一次接收。

2.6.3 波特率

在方式0下，波特率是振荡器频率除以12。

在方式2下，波特率由振荡器的频率和SMOD决定。如果SMOD为0，波特率等于振荡器频率的1/64；当SMOD为1时，为振荡器频率的1/32。

在方式1和方式3下，波特率由定时器T1决定（在某些型号中也可以由第三个定时器T2决定，这里不讨论）。由于定时器是可编程的，因此可选的波特率范围也比较大，这两种方式也比较常用。

这时，波特率由下式决定：

波特率＝2^{smod}*(T1的溢出频率)/32

T1作为波特率发生器时，通常应禁止T1中断，工作在定时方式，选择方式2。这种方式是自动重装初始值的方式，可以保证定时的精确性。这种工作模式下，波特率的计算公式为：

波特率＝2^{smod}*振荡器频率/[32*12*(256-TH1)]

当振荡器频率为11.0592MHz时，对于常用的波特率如1.2K，9.6K等都可以计算出精确的定时器初值，因此这个频率常常被使用。

2.7 中断系统

现代的计算机都具有中断功能，能够对外界异步发生的事件进行及时处理。MCS－51中不同型号的单片机的中断数量是不同的，典型（基本）的8031有5个中断源，具有两个中断优先级，可以实现二级中断服务程序嵌套。可以分别指定每一个中断源的中断级别和中断允许字。

2.7.1 中断源

8031的5个中断源是INT0，INT1上的外部中断源，定时器T0，T1的溢出中断和串行口的发送接收中断等三个内部的中断源。

INT0和INT1上输入的两个外部中断源和它们的触发方式锁存在特殊功能寄存器TCON的低四位，高四位是T0和T1的运行控制位和溢出标志位。它的结构见前面的说明。

IEn是外部中断的请求源标志。当外部中断向CPU请求中断时置位，当CPU响应此中断由硬件清零。

ITn是外部中断的触发方式控制位。当IT＝0时是电平触发方式，外部引脚出现低电平时产生中断请求。当IT＝1时是边沿触发方式。其详细区别后面叙述。

内部中断的中断请求标志分别在不同器件的控制寄存器中。TCON的TF0和TF1标志T0和T1的溢出中断请求，这两个标志都由硬件置位和清零。串行口有两个中断原因：接收中断RI和发送中断TI，它们逻辑或之后作为一个中断源。当串行口发送完一个字符后硬件置位TI，接收到一个字符后硬件置位RI。要注意当CPU响应串行口中断后，硬件并不将这两个标志清零，必须在中断服务程序中将其清零（这是为了在一个中断程序中能够区分两种中断原因）。

这些标志也可以由软件置位和清零，可以达到由程序模拟外部中断的目的。

2.7.2 中断控制

每一个中断源的开放和屏蔽是由中断允许寄存器IE控制的。IE的字节地址是0A8H，也是可以位寻址的特殊功能寄存器。其格式如下：

D7	D6	D5	D4	D3	D2	D1	D0
EA	空	空	ES	ET1	EX1	ET0	EX0

EA是整个CPU的中断允许标志。当EA＝1时，CPU可以响应中断；当EA＝0时，屏蔽所有中断申请。

ES是串行口中断允许位；ET1和ET0是T1和T0的中断允许位；EX1和EX0是外部中断INT1和INT0的中断允许位。这些位为1时允许响应对应的中断申请，为0时屏蔽之。

MCS－51有两级中断优先级。一个正在执行的中断服务程序可以被高优先级的中断申请所打断，但不能被相同优先级或低优先级的中断申请所中断，这些新的中断申请必须在现在执行的中断处理程序结束后返回主程序，并执行一条指令后才能得到响应。MCS－51内部有不可存取的优先级状态触发器来标志CPU是否处于运行何种优先级的中断服务程序。

MCS－51内部有一个中断优先级寄存器IP（地址为0B8H），定义各个中断源的优先级，格式如下：

D7	D6	D5	D4	D3	D2	D1	D0
空	空	空	PS	PT1	PX1	PT0	PX0

从D4开始，按顺序分别为串行中断、T1、INT1、T0，INT0的中断优先级标志，若为1，表示此中断的优先级为高，否则为低。

当CPU接收到同样优先级的几个中断申请时，按照以下的次序决定中断响应的优先次序：INT0，T0，INT1，T1，串行中断。

CPU复位后，IE，IP的内容均为0，禁止所有中断。由初始化程序对IE、IP编程，以根据需要决定中断的允许与优先级。

2.7.3 中断响应过程

CPU在每一个机器周期的S6阶段采样所有允许的中断申请，如果发现有中断申请，并且没有下列条件的阻止，将在下一个机器周期响应激活的最高级中断申请。

- CPU正在处理相同的或更高级别的中断；
- 现在的机器周期不是所执行指令的最后一个机器周期（多周期指令）；
- 正在执行的指令不是中断返回指令RETI，或是对IE，IP的写操作指令；

CPU响应中断时，先置位响应的优先级状态触发器（防止其他相同级别或低级别中断的执行）；将中断请求标志清零（TI和RI除外）；把当前程序计数器PC的内容压入堆栈（但不保护PSW），然后根据中断源的类型，将程序转移到对应的中断服务程序入口地址。各中断服务程序的入口地址是固定的，如下所示：

中断源	入口地址
INT0	0003H
T0	000BH
INT1	0013H
T1	001BH
串行口中断	0023H
定时器T2（8052等型号）	002BH

通常在中断的入口处，放置一条长跳转指令，转移到用户设计的中断处理程序中。

CPU在执行中断处理程序一直遇到RETI指令为止，RETI表示中断处理程序的结束。CPU在执行这条指令时，把响应中断时所置的优先级状态触发器清零，从堆栈的顶部弹出两个字节到程序计数器PC，CPU重新执行被打断的主程序。

在中断处理过程中，必须由用户程序自己保护和恢复现场。

CPU响应一个外部中断的时间取决于当前执行的指令，如果当前执行高级别的中断处理程序，就要等待到这个程序结束。如果中断响应没有被延迟，在3－8个周期内就可以响应这个中断。

2.7.4 外部中断触发方式选择

若外部中断定义为电平触发方式，外部中断申请触发器的状态随着CPU在每个机器周期采样到的外部中断输入线的电平变化而变化。外部要申请中断时，必须保持有效（低电平），直到CPU实际响应该中断为止，同时在

中断服务程序返回之前，外部中断必须无效（高电平），否则CPU将再次响应该中断。所以电平触发方式适合于外部中断以低电平输入的，并且CPU能够清除外部中断输入源的状态。

外部中断定义为边沿触发方式，外部中断申请触发器能锁存外部中断输入线上的负跳变，即使CPU暂时不能响应，中断申请标志也不会丢失。在这种方式下，只有在相邻的两次采样中外部中断由高转为低，才置位中断申请触发器。因此外部输入的脉冲宽度必须至少保持12个时钟周期，才不会丢失中断。这种方式适合于以脉冲方式输入的外部中断请求。

习题和思考题

1．什么是堆栈，堆栈应该设在什么地方。
2．在8031的内部RAM中，那些单元适合作数据缓冲区。
3．若晶振的频率是11.0592MHz，使用T0产生1微秒的定时，可以使用那些方式，分别写出各种方式的初始化程序。
4．若晶振的频率是12MHz，如何测量外部20-1KHz之间的方波周期，若频率约为0.5MHz时，又应如何测量。讨论每种方式的误差范围。
5．如何使用T0实现1秒钟的定时。
6．MCS－51的中断处理程序能否放在64K范围的任意位置，若能，如何实现。
7．串行口的0方式输出，能否外接多个74LS164，若不可以说明原因，若能，画出实现方式的逻辑框图，说明数据输出方法。

第三章 MCS－51指令系统

计算机的指令系统是一套控制计算机操作的编码，称为机器语言。计算机只能识别和执行机器语言的指令。为了容易使人们所理解，便于记忆和使用，通常用符号指令来描述计算机的指令系统，称为汇编语言。将汇编语言翻译为机器语言的程序是汇编程序。这一章中将使用Intel的标准格式汇编指令来介绍MCS－51的指令系统。

3.1 指令格式

3.1.1 汇编指令

MCS－51汇编指令由标号、操作码助记符字段和操作数字段组成。指令格式如下：

[标号：] 操作码 [操作数1],[操作数2],[操作数3]

第一部分是标号，一般可以省略。标号必须以冒号结束。

第一部分是指令操作码助记符，它由二至五个英文字母组成。

第二部分是操作数，它以一个或几个空格与操作码分开，操作数之间由逗号分隔。根据指令不同，可能有一个、二个、三个或没有操作数。

3.1.2 伪指令

大多数汇编程序都提供许多伪指令供用户使用。大多数伪指令汇编时不产生机器语言指令，仅仅为汇编程序提供信息。最常用的伪指令有以下几条。

一、定位伪指令

ORG xxxx

xxxx为十进制或十六进制数。它指出了该伪指令后的指令的汇编地址，即生成的机器指令存放的存储器地址。在一个汇编语言源程序中允许使用多条伪指令，但一般情况下，指令区域不能重叠。

二、定义字节伪指令

DB X1[, X2...]

X为单字节数据，它为十进制或十六进制数，X也可以是由单引号括起来的字符串。该伪指令把后面定义的数据送到目标程序存储器，通常用于定义一个常数表。

三、字定义伪指令

DW W1[,W2...]

本指令与上一个伪指令作用相类似，用以存放双字节数据。通常可以定义一个地址表。

四、汇编结束指令

END

该伪指令指出结束汇编，汇编程序不再对后面的内容进行处理。

五、注释

汇编程序允许用户在源程序中添加注释。注释以分号开始。

3.1.3 常用的缩写符号

在下面描述MCS－51的指令系统时，我们使用下面的缩写符号，含义如下：

A 累加器ACC

AB　　累加器ACC和辅助寄存器B组成的寄存器对，构成一个16位2进制数。

Direct　直接地址

＃data　立即数，表示一个常数

@ 间接寻址

$ 当前指令起始地址

(X)　X寄存器内容

((X))　由X寄存器寻址的内部RAM存储器单元内容

注意本章中所说的寄存器一般指工作寄存器R0－R7

在MCS－51汇编语言中，一般可以直接使用SFR的符号寻址这些寄存器，也可以使用类似IE.2的形式寻址SFR中的位地址。

3.2 寻址方式

寻址方式决定一条指令中存取参与运算的数据的方式。MCS－51有五种寻址方式：寄存器寻址、直接寻址、寄存器间接寻址、立即寻址、和基址变址寻址。

3.2.1 寄存器寻址

由指令指出某一个寄存器的内容作为操作数称为寄存器寻址。一般在指令中有3位来表示所用的寄存器（R0－R7）。累加器ACC、B、DPTR和C（布尔处理中的累加器）也可以使用寄存器寻址方式访问，只是对它们的寻址都隐含在相应的指令中。例如指令：

INC R0　；表示将R0的内容增一。

3.2.2 直接寻址

在指令中含有操作数的直接地址，该地址指出了参与运算的数据所在的字节单元地址（内部RAM）或位地址。

直接寻址方式访问以下三种存储空间：

特殊功能寄存器（只能用直接寻址方式访问）；

内部数据存储器；

位地址空间；

例如指令：

ANL A，41H；将ACC的内容和内部RAM中41H单元的内容相逻辑与，结果写回ACC中

本指令使用了两个操作数，A是寄存器寻址，隐含在指令中；操作数2是直接寻址，出现在指令码中。

3.2.3 寄存器间接寻址

由指令指出某一个寄存器的内容作为操作数的地址，这种寻址方式叫做寄存器间接寻址。

寄存器间接寻址使用R0，R1作为内部RAM的寻址指针；使用DPTR作为外部扩展数据存储器的寻址指针。使用符号@表示。

例如指令：

MOV @R0，44H

若此时R0的内容为32H，则将内部RAM中44H单元的内存传送到内部RAM中32H单元中。第一个操作数使用寄存器间接寻址。

3.2.4 立即寻址

在指令中包含操作数本身。用符号#表示，例如指令：

MOV A， #75H

将常数75H传送到累加器A中。

3.2.5 基址变址寻址

这种方式以16位的程序计数器PC或数据指针DPTR作为基寄存器，以8位的累加器A作为变址寄存器，基址寄存器和变址寄存器的内容相加作为16位的地址访问程序存储器。如：

MOVC A，@A+PC

MOVC A, @A+DPTR

3.2.6转移指令寻址

在转移指令中,欲转移到的程序地址的寻址方式有4种。即长转移寻址、短转移寻址、相对转移寻址和基址变址寻址。

长转移寻址直接在指令中指出16位的目标地址，可以转移到程序存储器的任意位置，而不管当前指令的位置。类似于8086的段间转移指令。

短转移寻址可以转移到与下条指令存放在同一个2K区域内的一个地址。在指令执行时，首先将PC加2，然后将PC的高5位和指令中指出的11位低位地址相连作为目标地址。类似于8086的段内转移指令。

相对转移寻址在指令中给出目标地址与当前PC的单字节有符号相对位移。执行时，首先将PC加2，然后将指令中的有符号相对地址与PC相加计算出目标地址。因此，目标地址可以在本指令的前128个字节到后127个字节之间。

转移指令的基址变址寻址使用DPTR和A，只有一条指令：

JMP @A+DPTR

这条指令将DPTR中的16位数与A中的8位无符号数相加得到目标地址，通常用于查表式的多地址跳转。

3.3 指令类型

MCS－51汇编语言有5大类，33种操作功能，42种操作码助记符来描述。每一种功能可以有多种寻址方式，这样共有111种指令。按字节数分有49条单字节指令，45条双字节指令和17条3字节指令。按指令执行时间分，有64条单周期指令、45条双周期指令和2条4周期指令。

按功能分类，MCS－51的指令系统可分为：

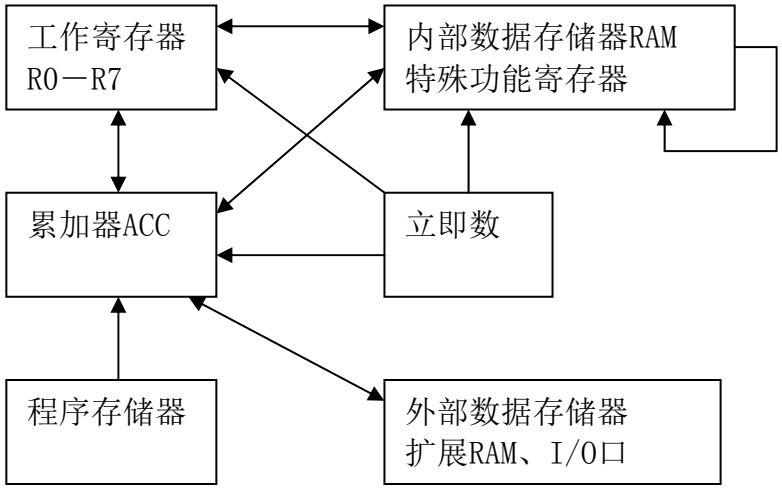
- ◆ 数据传送指令
- ◆ 算术运算指令
- ◆ 逻辑运算指令
- ◆ 位操作指令
- ◆ 控制转移指令

下面我们分别介绍各类指令

3.4 数据传送指令

数据传送指令用于在存贮器空间的各个地址之间传送数据。

数据传送的方向如图所示，对A和工作寄存器的操作最多。另外不是所



有操作方向的所有寻址方式都是有效的。

3.4.1 内部数据传送指令

1） 以累加器A为目的操作数的指令

```
MOV A,Rn
MOV A,direct
MOV A,@Rn
MOV A, #data
```

2） 以Rn为目的操作数的指令

MOV Rn, A
MOV Rn, direct
MOV Rn,#data

3) 以直接寻址单元为目的操作数的指令

MOV direct, A
MOV direct, Rn
MOV direct, direct
MOV direct, @Ri
MOV direct, #data

4) 以寄存器间接寻址为目的操作数的指令

MOV @Ri, A
MOV @Ri, direct
MOV @Ri, #data

5) 栈操作指令

PUSH direct
POP direct

在堆栈指令中，SP指向栈顶的位置。执行压栈指令时，首先将SP加1，然后将直接地址指出的内容传送到SP寻址的内部RAM中。退栈指令的过程与此相反。

6) 交换指令

XCH A, Rn
XCH A, direct
XCH A,@Ri
XCHD A,@Ri

前三条交换指令将A的内容与对应寻址单元中的内容相交换。最后一条指令将A的低四位与R0或R1指出的间接寻址单元中的内容的低四位相交换，各自的高四位不变。

3.4.2 累加器A与外部数据存储器传送指令

MOV DPTR, #data16
MOVX A, @DPTR
MOVX @DPTR, A
MOVX A, @Ri
MOVX @Ri, A

外部数据存储器的存取指令是通过DPTR或@Ri寻址的。上面的第一条指令可以将一个16位立即数赋予DPTR。下两条指令使用DPTR指出的地址存取外部数据存储器。也可以使用@Ri来寻址，这时地址的高位等于当前程序地址的高位，低位是Ri的内容。

3.4.3 查表指令

MOVC A, @A+PC

这条指令以PC作为基址寄存器，A的内容作为无符号数和PC内容（下一条指令的起始地址）相加得到的16位地址，由该地址指出的程序存储器单元内容送到累加器。此指令常用于查表，要求表格整个存放在查表指令以下的256字节之内。

MOVC A, @A+DPTR

此指令也用于查表，以DPTR作为基址寄存器，A的内容和DPTR相加作为程序存储器的地址，此地址内容送到A。这种查表指令比较方便使用，表格可以存放到任意地址，可在多处使用。但表格的大小仍不能超过256字节。

3.5 算术运算指令

MCS－51的算术运算指令有加、减、乘、除和增量、减量指令。算术运算的结果不仅反映在结果寄存器中，还反映在程序状态寄存器PSW中。PSW是一个特殊功能寄存器，字节地址是0D0H，可位寻址，位寻址地址是0D0H－0D7H。它反映了算术运算指令的结果状态，可以作为条件转移指令的条件。其格式和各位功能如下：

D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	F1	P

CY：进位标志。同时也是布尔处理机的累加器C。如果运算结果最高位有进位或借位时，该位置1，否则清零。

AC：辅助进位标志。如果运算结果的低四位有进位或借位时置位，否则清零。此标志一般用于二进制运算的十进制调整。

OV：溢出标志。如果操作结果有进位进入最高位但最高位没有产生进位或者最高位产生进位而低位没有向最高位进位，这时OV标志置位，否则清零。OV标志是有符号运算（补码）的溢出标志，而CY作为无符号数运算的溢出标志。

P：奇偶标志。这时ACC的奇偶标志。如果ACC的8位模2和为1（即有奇数个1），则P标志置位，否则清零。由于P总是表示A的奇偶性，只随ACC的内容而变化，所以一个数写入PSW，P位的值不变。

RS1，RS0：工作寄存器区选择位。前面已经说明。

F0：用户标志位。用户可以作为软件标志自由使用。

F1：保留，用户也可自己使用。

3.5.1 加法指令

ADD A，Rn

ADD A, direct

ADD A, @Ri

ADD A, #data

这些是将寻址方式指出的内容与A的内容相加，结果仍送入A。PSW各位标志根据运算结果改变。

ADDC A，Rn

ADDC A, direct

ADDC A, @Ri

ADDC A, #data

这些指令在作加法时把寻址方式指出的内容、A的内容和进位标志CY相加送入A。通常用于多字节加法。

3.5.2 减法指令

SUBB A, Rn

SUBB A, direct

SUBB A, @Ri

SUBB A, #data

减法指令只有带借位的减法一种。它从累加器中减去指定的变量和进位标志CY，结果在累加器中。PSW各位标志根据运算结果而定。如果需要进行普通减法，可以先将CY清零。

3.5.3 十进制调整指令

DA A

这条指令将累加器中的数据根据AC标志将其调整为压缩BCD码。可以对两个压缩BCD运算后的结果调整为正确的十进制数据。

3.5.4 增量、减量指令

INC A

INC Rn

INC direct

INC @Ri

INC DPTR

DEC A

DEC Rn

DEC direct

DEC @Ri

这些指令将指定的变量加1（或减1）存放到原位置。不影响任何标志。当使用这些指令改变输出口的内容时，原始数据从口的锁存器中读入，不从引脚读入。

3.5.5 乘除法指令

MUL AB

这条指令把A和B中的无符号8位数相乘，其16位积的低位存放在A中，高位字节在B中。如果积大于255，则置位溢出标志OV，否则清零。进位标志CY总是清零。

DIV AB

这条指令的功能是把A的8位无符号整数除以B中的8位无符号整数，所得商的整数部分存放在A中，余数存放在B中。

如果除数（B）为零，则结果A和B中的内容不定，并置位溢出标志OV。进位标志CY总是清零。

3.6 逻辑运算指令

3.6.1 累加器A的逻辑操作指令

CLR A

这条指令的功能是将A清零，不影响CY，AC，OV等标志。

CPL A

将A的每一位逻辑取反。不影响标志。

RL A

将A的内容向左循环移位1位，位7移入位0。不影响标志。

RLC A

将A和进位标志一起向左循环移位1位。位7移入CY，CY移入位0。不影响标志。

RR A

类似RL指令，方向是向右移位。

RRC A

类似RLC指令，方向是向右移位。

SWAP A

这条指令将累加器A的高四位和低四位相互交换。

3.6.2 两个操作数的逻辑操作指令

ANL A, R0

ANL A, direct

ANL A, @Ri

ANL A, #data

ANL direct, A

ANL direct, #data

上述指令在指定的变量之间基于位的逻辑与操作。结果存放在前一个变量中。不影响标志。

ORL A, R0

ORL A, direct

ORL A, @Ri

ORL A, #data
ORL direct, A
ORL direct, #data

上述指令在指定的变量之间基于位的逻辑或操作。结果存放在前一个变量中。不影响标志。

XRL A, R0
XRL A, direct
XRL A, @Ri
XRL A, #data
XRL direct, A
XRL direct, #data

上述指令在指定的变量之间基于位的逻辑异或操作。结果存放在前一个变量中。不影响标志。

这些指令在修改输出口（P0－P3）时，原始数据从锁存器读入，不读引脚状态。

3.7 位操作指令

MCS－51系列单片机内有一个布尔处理机，它以CY（PSW.7）作为累加器C，以RAM和SFR内的位寻址区的位单元作为操作数，进行位变量的传送，修改和逻辑运算指令。

3.7.1 位传送指令

MOV C, bit
MOV bit, C

使用位传送指令中一个操作数必须是C，另一个是直接寻址位单元。

3.7.2 位修改指令

CLR C
CLR bit
SETB C
SETB bit
CPL C
CPL bit

这些指令将C或位单元的内容清零、置1和取反。

3.7.3 位变量逻辑操作指令

ANL C, bit
ANL C, /bit

逻辑与（AND）指令将C与位地址bit中内容相与送入C中。后面指令中在位地址前的“/”符号是将此位地址内容取反后（但不改变其本身）参与运算。

ORL C, bit
ORL C, /bit
进行逻辑或。其他类似ANL指令。

3.8 控制转移指令

3.8.1 无条件转移指令

转移指令使用的寻址方式在前面已经描述。

长跳转指令LJMP addr16使用16位绝对地址寻址，可以跳转到程序的任意地点。

短跳转指令AJMP addr11使用11位短转移寻址，可以跳转到和下条指令处于相同的2K页面的其他位置。

相对转移SJMP rel使用相对转移寻址，可以跳转到本条指令的前128个字节到后127个字节之内。

基址变址寻址JMP @A+DPTR可以实现查表转移。

3.8.2 条件转移指令

条件转移指令是依照某种特定条件实现转移的指令。条件满足时跳转到指令指出的目的地址，不满足时执行下面的指令。条件转移指令都是使用相对转移指令。

1) 测试条件符号转移指令
JZ rel
如果累加器为零则转移；
JNZ rel
如果累加器不为零则转移；
JC rel
如果CY=1则转移；
JNC rel
如果CY=0则转移；
JB bit, rel
如果直接寻址单元的值为1，则转移；
JNB bit, rel
如果直接寻址单元的值为0，则转移；
JBC bit, rel
如果直接寻址单元的值为1，则将此位清零，然后转移；

2) 比较不相等转移指令
CJNE A, direct, rel
CJNE A, #data, rel
CJNE Rn, #data, rel
CJNE @Ri, #data, rel

这些指令比较两个操作数是否相等，如果不相等则转移。如果第一操作数小于第二个操作数，则置位CY，否则CY清零。

3） 减量转移指令

DJNZ Rn, rel

DJNZ direct, rel

这组指令把源操作数减1，结果回送到源操作数中；如果结果不为0则转移。不改变标志位。

3.8.3 调用和返回指令

调用和返回指令是用于子程序设计的。在调用时地址是靠堆栈来保存返回地址的，这样可以实现子程序的嵌套。

短调用指令 ACALL addr11使用11位短转移寻址，长调用指令LCALL addr16使用16位长转移寻址。它们在执行时，都将当前地址压入堆栈，并将栈指针加2。接着将目标地址装入PC中。

子程序返回指令RET从堆栈中弹出PC的高位和低位字节，把栈指针减2，返回到子程序调用的位置继续执行。不影响任何标志。

中断返回指令RETI除了执行RET的功能外，还清除内部相应的中断状态寄存器（该寄存器由CPU在中断响应时由硬件置位）。因此，中断服务程序必须由RETI结束。CPU执行RETI指令后至少再执行一条指令，才能响应新的中断请求。

空操作指令NOP不执行任何操作。一般在延迟等程序中用于调整CPU的执行时间。

习题和思考题

1. 有多少种指令可以将A的内容清零，写出每种指令，分析其对各种标志的影响。
2. 如果条件转移指令的目标地址距离当前地址的距离大于128字节，如何转移。
3. 实现循环10次的方法有哪些。
4. 编制一段程序，将存放于30H，31H中的4位压缩BCD码作为减数，减去存放于32H和33H的4位压缩BCD码（十进制减法），结果存放在35H和36H中，如果结果为负数，置位CY。
5. 编制一段程序，将存放在30H－33H中的四位ASCII码数字转换为二进制数，存放于34H－35H中。
6. 编制一段程序，读取P1口的低四位，取反后输出到P1的高四位，保持P1的低四位不变。
7. 编制一段程序，计数P1.0位上的负跳变（从1到0），要求对于小于8个机器周期的干扰性跳变不予计数。

第四章　新型单片机增加功能

STC89C516RD+和STC12C5A16AD是两种扩展的单片机。

STC89C516RD+ 增加以下功能：

1）增强型 6 时钟/机器周期，12 时钟/机器周期 8051 CPU
2）用户应用程序空间 64K 字节
3）片上集成 1280 字节 RAM
4）增加 P4 端口
5）增加看门狗定时器
6）支持 ISP 和 IAP

STC12C5A16AD增加以下功能：

1）增强型 8051 CPU，单时钟/机器周期，指令代码完全兼容传统 8051，速度比普通 8051 快 8~12 倍
2）用户应用程序空间 16K 字节
3）片上集成 1280 字节 SRAM
4）内有 A/D 转换，10 位 ADC，共 8 路，转换速度可达 250K/S(每秒钟 25 万次)
5）PWM（2 路）/PCA（可编程计数器阵列，2 路）
6）增加 P4 端口
7）增加看门狗定时器
8）支持 ISP 和 IAP
9）有 EEPROM 功能（其实是 Flash）

附录四　单片机C语言程序设计

C51是一种在MCS-51单片机上的特定的C语言，能对单片机的硬件资源进行灵活快捷地操作，具备汇编语言的功能，同时不失高级语言的可读性好、可移植性好等特点，可以方便调用成熟的库函数（或程序模块）等现有资源，，因此广泛应用于控制工程、信号处理等众多领域。

C51具有C语言的功能，与标准C语言没有什么本质的区别，完全的支持标准C语言全部指令，同时C51添加了许多用来优化8051指令结构的指令，对C语言进行扩展，在大多数应用环境下，C51程序的执行效率已经非常接近汇编语言程序。

第一章　C51对标准C语言的扩展

1.1　数据类型

C51相对于C语言扩展了以下几个特殊的数据类型：

1. bit 型变量

bit变量可以用于定义一个位变量，但不能定义位指针，也不能定义位数组。它的值是一个二进制位（即0或1），类似一些高级语言中的bool类型中的true和false。

2. sfr 特殊功能寄存器

sfr特殊功能寄存器，占用一个字节内存单元，值域为0～255，利用它可以访问51系列单片机内部的所有特殊功能寄存器。如用sfr P1 = 0x90这一句定义变量P1为P1端口在片内的寄存器，在后面的语句中可以用P1=255（对P1端口的所有引脚置高电平）之类的语句来对特殊功能寄存器进行操作。

3．sfr16特殊功能寄存器

sfr16特殊功能寄存器占用两个内存单元，值域为0～65535。sfr16和sfr一样用于操作特殊功能寄存器，所不同的是它用于操作占两个字节的寄存器，如定时器T0和T1。

4．sbit型变量

sbit型变量可以访问和操作芯片内部RAM中的可寻址或特殊功能寄存器中的可寻址位。

在MCS-51系列单片机系统中，有时要访问特殊功能寄存器中的某些位，这时采用关键字sbit定义可位寻址特殊功能寄存器的位寻址对象。定义方法有如下3种：

(1) sbit 位变量名 = 位地址 这种定义是将为的绝对地址赋给位变量，位地址必须位于0x80～0xFF之间。例如：sbit var=0xD2;

(2) sbit 位变量名 = 特殊功能寄存器名^位位置 当可寻址位位于特殊功能寄存器中时，可以采用这种方法，这里位位置是一个0～7之间的常数。例如：sbit var=PSW^2;

(3) sbit 位变量 =字节地址^位位置 这种定义是以一个常数（字节地址）作为基地址，该常数必须在0x80～0xFF之间，位位置则是一个0～7之间的常数。例如 var=0xD0^2;

单片机中的特殊功能寄存器和特殊功能寄存器可寻址位，已被预定义放在文件reg51.h中，在程序的开头只需加上#include<reg51.h>或#include<reg52.h>即可。另外，sbit还可访问单片机内20H～2FH范围内的位对象。

1.2 变量存储类型

8051存储区可分为内部数据存储区、外部数据存储区以及程序存储区，各自的寻址方式不同。

C51将内部数据存储区分为三种不同的存储类型，分别用data，idata和bdata来表示。51系列及其派生系列最多可有256B的内部存储区，其中低128B可以直接或间接寻址，高128B（从0x80到0xFF）只能间接寻址，从20H～2FH的128b可位寻址；C51能对其进行读写操作。对存放在低128B直接寻址的变量用data说明；对存放在整个内部数据存储区（256B）间接寻址的变量用idata说明；对存放在20H～2FH的位寻址的变量用bdata说明。

C51将外部数据存储区分为两种不同的存储类型，分别用xdata和pdata来表示。外部数据存储区最多可有64KB，C51可对其进行读/写操作。Xdata可以指定外部数据存储区64KB内的任何地址，而pdata仅指示1页（或256B）的外部数据存储去。访问外部数据存储区比访问内部数据存储区慢，因为外部数据存储区是通过数据指针加载地址来间接访问的。

C51用code来表示程序存储区类型，程序存储区是只读不写的。值得注意的是：程序存储区可能在8051CPU内或者在外部或者在内部都有，具体要看设时选择的CPU的型号来决定程序存储区在CPU内，外的分布情况，以及根据程序容量决定是否需要程序存储器的扩展。

C51数据存储类型与8051系列单片机实际存储空间的对应关系如表所示。

变量存储类型

存储类型	与存储空间对应关系
data	直接寻址片内数据存储区，访问速度快（128B）
bdata	可位寻址片内数据存储区,允许位与字节混可访问(16B)
idata	间接寻址片内数据存储区，可访问片内全部RAM地址空间（256B）
pdata	分页寻址片外数据存储区（256B），通过P0口的地址对其寻址（在汇编语言中由MOVX @Rn访问）
xdata	片外数据存储区(64KB)(在汇编语言中由MOVX @DPTR访问）
code	程序代码存储区(64KB)(在汇编语言中由MOVC @DPTR访问）

下面用实例来说明如何定义各种存储类型变量。

(1)char data var1；表示字符变量var1被定义在8051单片机内数据存储区中（地址为00H～0FFH）。

(2)bit bdata var2；表示位变量var2被定义在8051单片机内数据存储区的位寻址区中（地址为20H～2FH）。

(3)float idata var3，var4；表示浮点变量var3，var4被定义在8051单片机内数据存储区中，并且只能用间接寻址方式访问。

(4)int pdata var5；表示整型变量var5被定义在片外数据存储区中，他的高字节地址保存在P2口中。

(5)unsigned int xdata var5；表示无符号整型变量var5给定义在片外数据存储区中。

1.3 存储器模式

C51提供了三种存储器模式：small，compact和large模式，用来决定变量的默认存储类型，参数传递区和无明确存储类型说明变量的存储类型。

在small模式下，所有的默认变量都位于内部RAM中（这和使用data定义存储类型方式的结果一样）。如果有函数被声明为再入函数，编译器会在内部RAM中为他们分配空间。这种模式的优势为数据的存取速度很快，因此在程序设计中应该尽量使用small模式。不足之处：只能使用120B的存储空间（总共128B，但至少有8B被存储器组使用）。一般来说如果系统多徐的内存小于内部RAM数时，都应以small模式进行编译。

compact模式下，所有的默认变量都位于外部RAM区的一页内（这和使用pdata定义存储类型方式的结果一样）。该存储类型适用于变量不超过256B的情况，这是由寻址方式决定的。和small模式相比，效率较低，对变量的访问速度相对慢一些，但比large模式快。对数据的寻址是通过R0和R1（这两个寄存器用来存放地址的低位字节）进行间接寻址，如果要使用多于256B的变量，高位字节（指出具体哪一页）可用P2口指定。

在large模式中，所有默认变量可放在多达64KB的外部RAM中（这和使用xdata定义存储类型方式的结果一样），均使用数据指针DPTR进行寻址。其优点是空间大，可存变量多；缺点是速度较慢，尤其对于两个以上的多字节变量的访问速度来说更是如此。

1.4指针

C51支持两种指针：通用指针和存储器指针。

1.通用指针

通用指针用三个字节进行存储：第一个字节为存储器类型,其编码如下表所示，第二个字节为16位偏移地址的高字节，第三字节为16位偏移地址的低字节。它的声明和使用与标准C语言一样。

存储器类型编码

存储器类型	idata/data/bdata	xdata	pdata	code
值	0x00	0x01	0xFE	0xFF

```
如， char *  string;      //定义了一个指向char型数据的指针，而指
                          //针string本身则根据不同的存储器模式存
                         //放在相应的区域
      int *  number;     //定义了一个指向int型整数的指针，而指针
                         //number本身则根据不同的存储器模式存放
                         //在相应的区域
```

此外，C51也可以使用前面介绍的关键字（见上表）对指针的存储位置进行声明。

```
如， char * xdata ptr;   //定义了一个指向char型数据的指针，而指
                        //针ptr本身存放在外部存储区
      int * idata varptr;  //定义了一个指向int型整数的指针，而
                           //指针varptr本身存放在内部数据存储
                          //区，用间接方式寻址
```

2．存储器指针

C51允许编程者规定指针指向的存储区域，因此这种指针又称为存储器指针，包含了对数据类型和数据空间的说明。

```
如，char data  * str;        //定义指向data区中char型数据
                             //变量的指针
    int xdata *number;       //定义指向xdata外部数据存储区
                             //中int型整数的指针
```

同时与普通指针一样，也可以使用前面介绍的关键字（见上表）对存储器指针的存储位置进行声明。

```
如，char data  * xdata ptrr; //存放在xdata区的指针，指向data
                             //区中的字符型数据
     int xdata * xdata varptr; //存放在xdata区的指针，指向
                             // xdata区中的int型数据
```

2.指针转换

C51编译器可以在存储器指针和通用指针之间转换，应注意以下几个问题。

(1) 当存储器指针作为一个实参传递给使用通用指针的函数时，指针自动转换。

(2) 一个存储器指针作为一个函数的参数，如果没有说明函数原型，存储器指针常

常被转换为通用指针。如果调用希望一个短指针作为参数，可能会发生错误。为了避免这种错误，可以使用预编译命令：#include 文件和所有外部函数的原型。这样可以确保编译器进行必须的类型转换并检测出类型转换错误。

(3) 可以强行改变指针类型。

1.5 绝对地址的访问

C51提供了如下三种访问绝对地址的方法。

1.绝对宏

在C51自带的ABSACC.H头文件中已经给出了指向不同存储区首地址的指针(包括子类型和字节类型)的宏定义。在程序中只要加入语句“#include <absacc.h>”，就可以直接使用已定义的关键字进行程序设计，访问存储器的相应单元。

```
如，rval=CBYTE[0x0006];     //读程序存储区地址0006H的字节内容
     rval=XWORD[0x0002];  //读外部数据存储区地址0004H（2×
```

```
                                    //sizeof（unsigned int）=4）的字内容
DBYTE[0x0002]=5;                    //向内部数据区0002H写入字节内容5
PWORD[0x0002]=57;                   //向pdata存储区地址0004H（2×sizeof
                                    //（unsigned int）=4）写入字内容为57
```

2. _at_关键字

_at_关键字可以把变量定位在51单片机某个固定的地址空间上。用法：直接在数据定义后加上_at_ const即可，其中const是常量（指明定位变量的地址）。

注意：(1) 绝对变量不能被初始化。

(2) bit型函数及变量不能用_at_指定。

如，char xdata text[256] _at_ 0xE000; //指定数组text开始于xdata
//的0xE000H

提示，如果外部绝对变量是I/O端口等可自行变化数据，需要使用volatile关键字进行描述。

3. 连接定位控制

此方法是利用连接控制指令code、xdata、pdata、data和bdata对“段”地址进行指定，如要指定某具体变量地址，则有很大局限性。

1.6函数的使用

C51中函数的定义方式与标准C语言是相同的，由于C51在标准C语言的基础上扩展了若干专用的关键字，因此可以将其应用于函数的定义中。

C51的函数定义格式如下。

```
[return_type] funcname ([args])  [{small/compact/large}]
[reentrant]
[interrupt n] [using n]
```

其中，return_type: 函数返回值类型，默认为int

funcname: 函数名

args: 函数参数列表

{small/compact/large}: 函数模式选择

reentrant: 表示函数是可递归的或是可重入的

interrupt: 中断函数（n为中断源编号）

using: 指定函数所用的寄存器组（n为0～3的整数）

1. 中断函数

单片机的中断系统十分重要，C51中断函数可以来声明中断和编写中断服务程序，中断过程通过使用interrupt关键字和中断源编号（0～31）来实现，其中基本中断源号为0～4。

函数格式为：

```
return_type funcname interrupt  n using n
```

其中，interrupt n中的n为中断源编号，中断源编号告诉编译器中断程序的入口地址，两者之间一一对应，基本中断号的对应关系如下表所示。

基本中断源编号与入口地址对应关系

中断源编号	中断源描述	入口地址
0	外部中断0	0003H
1	定时器/计数器0溢出	000BH
2	外部中断1	0013H
3	定时器/计数器1溢出	001BH
4	串行口中断	0023H

using n中的n对应四组通用寄存器中的一组，函数入口时将其所在的寄存器组保存，函数返回时再恢复寄存器组。C51中可使用using指定寄存器组，n的取值为0～3的常整数，分别表示51单片机内的4个寄存器组。此关键字一般在不同优先级别的中断函数中很有用，这样可以不用在每次中断的时候都对所有寄存器进行保存。

在使用中断函数时，编译器会进行以下操作。

(1) 在函数被激活的时候，如果有需要，将会把 ACC、B、DPH、DPL 以及 PSW 中的

内容压入堆栈中进行保存。

(2) 所有正在使用的寄存器都会被压入堆栈中进行保存，除非使用了 using 关键字。

(3) 在函数退出时，所有使用的寄存器，以及特殊寄存器，都会出栈，以恢复函数

执行前的状态。

(4) 函数结束时会调用 51 的 RETI 指令。

2. 重入函数

C51为了节省内部数据空间，提供了一种压缩堆栈，即为每个函数设定一个空间用于存放局部变量。函数中的每个变量都放在这个空间的固定位置，当此函数被递归调用时，会导致变量被覆盖。

在某些应用中一般函数是不可取的，因此C51允许将函数定义成重入函数。重入函数，又称作再入函数，是一种可以在函数体内间接调用其自身的一种函数。编译器采用模拟堆栈的方法，即在每次函数调用时，局部变量都会被单独保存，因此重入函数可被递归调用和多重调用，而不必担心变量被覆盖。

函数格式为：

```
funcname (args) reentrant
```

如，

```
int calc (int i,int b) reentrant
{
     int x;
     x=table[i];
     return(x*b);
}
```

注意：

(1) 重入函数不能传递 bit 类型参数。

(2) 重入函数不能被 alien 关键字定义的函数所调用。

(3) 在编译时：重入函数建立的是模拟堆栈区，small 模式下模拟堆栈区位于 idata 区，

compact模式下模拟堆栈区位于pdata区，large模式下模拟堆栈区位于xdata区。

(4) 在同一程序中可以定义和使用不同存储器模式的重入函数，任意模式的重入函数不能调用不同存储器模式的重入函数，但可以调用普通函数。

(5) 实际参数可以传递给间接调用的重入函数，无重入属性的间接调用函数不能包含调用参数。

第二章　C51程序举例

例1　仿真开关灯

设有按键S1,按下S1, led亮,再按下S1, led灭。

实验连线如实验一所示。其中按键S1与P1.0相连，led灯与P1.1相连。

程序如下：

```
#include <reg52.h>
bit flag;
sbit s1=P1^0;
sbit led=P1^1;
void main()
{
flag=0;
while(1)
{
  if(s1==0)    //判断S1 是否被按下,是则执行内嵌的语句
 {
        flag=～flag;
        while(s1==0);    //判断按键是否被松开,如果没有等待
```

```
    }
   if(flag==0)
  led=0;
   else
  led=1;
   }
   }
```

例2 通过串口进行I/O控制，要求从键盘接受以字符串，并将接到的字符串通过串口进行输出。

```
#include<reg51.h>
#include<stdio.h>
void main(void)
{
   unsigned char idata string[80];
   SCON=0x50;            //设置串口工作方式1，通过串口输出显示
   TMOD=0x20;      //定时器1用于串行波特率发生器，工作在方式2
   TH1=0xF3;
   TL0=0xF3;
   TR1=1;           //启动定时器1
   TI=1;             //发送第一个字符

   printf("please input strings:\n");
   gets(string,80);
   printf("%s \n",string);         //将字符输出到屏幕
   while(1);
}
```

例3 定时器和中断，设主频为12MHz,利用定时器T0定时，使P1.0口输出频率为100MHz的连续方波。

要产生100MHz的正方波，定时时间为5ms，即每5ms P1.0求反一次，即每个方波周期为10ms如果主频为12MHz，定时器工作在方式0，可以计算出初值为X=0C78H，转化为TH0=0x63，TL0=0x18。

程序如下：

```
 #include<reg51.h>
void main()
{
    P1=0;              //清P1口
    TMOD=0x00;       //T0使用定时模式，工作在方式0，无门控位
```

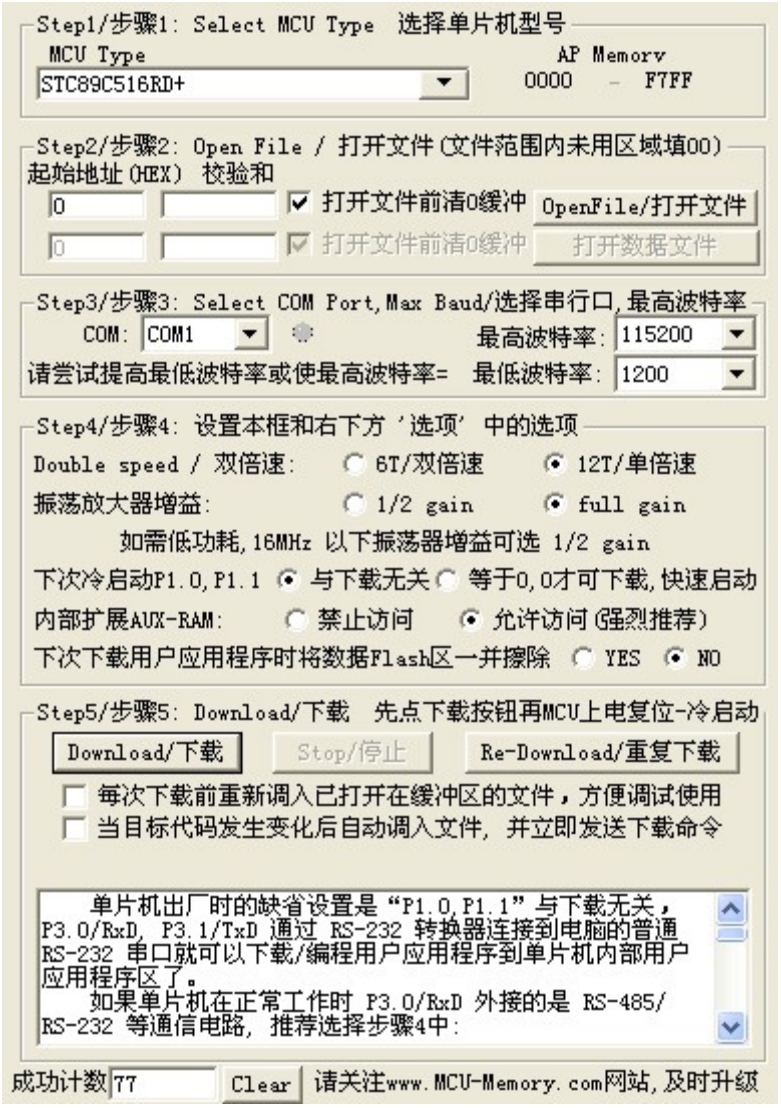
```
    TH0=0x63;          //为T0填入初值，定时时间5ms
    TL0=0x18;
    TR0=1;             //启动T0
    ET0=1;             //允许定时器0中断
    EA=1;              //cpu开放中断
    While(1);
}
//T0溢出中断处理函数
Void timer0_int() interrupt 1 using 2
//T0溢出中断，使用寄存器组2
{
   TH0=0x63;
   TL0=0x18;
   P1 ^= 0x01;              //P1.0取反，产生方波
}
```

第三章 C51软件使用及工作流程

本书实验中，C51编程使用keil软件和STC-ISP下载软件两款软件。

keil软件的操作类似于VC6，这里不再重点介绍。当把程序输入，调试完成时，我们需要生成.hex文件，以供下载软件将其下载到单片机中去。单击“工程”菜单，在下拉菜单中单击“目标选项”，在弹出的对话框中勾选中“输出”选项卡中的“生成hex文件”选项即可。

STC-ISP软件用于将.hex文件下载到单片机中。界面如下图：



Step1：在下载之前先要在MCU Type选择你所使用的单片机型号，本书实验中可能用到STC89C516RD+或者是STC12C5A16AD。

Step2：选择要下载的.hex文件。

Step3：选择串行通讯口，并设置分辨率。

Step4：按默认设置。

Step5：点击“Download/下载”开始对.hex文件进行装载，软件装载准备好会要求我们打开实验板电源正式进行下载，下载完成后软件会有提示下载是否成功。

第四章 C51常用库函数

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）。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。

最常用的头文件有：

CTYPE.H (字符函数头文件)；

STDIO.H(一般I/O函数头文件)；

STRING.H(字符串函数头文件)；

STDLIB.H(标准函数)；

MATH.H(数学函数)；

ABSACC.H(绝对地址访问头文件)；

INTRINS.H(内部函数头文件)，其中的_NOP_()函数经常用于延时；

STDARG.H(参数变量表头文件)；

SETJMP.H(全程跳转头文件)；

REGxxx.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。

文件reg52.h如下：

```
/*--------------------------------------------------------------------
------------
REG52.H

Header file for generic 80C52 and 80C32 microcontroller.
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
--------------------------------------------------------------------
----------*/

#ifndef __REG52_H__
#define __REG52_H__

/*  BYTE Registers  */
sfr P0    = 0x80;
sfr P1    = 0x90;
```

```
sfr P2    = 0xA0;
sfr P3    = 0xB0;
sfr PSW   = 0xD0;
sfr ACC   = 0xE0;
sfr B     = 0xF0;
sfr SP    = 0x81;
sfr DPL   = 0x82;
sfr DPH   = 0x83;
sfr PCON  = 0x87;
sfr TCON  = 0x88;
sfr TMOD  = 0x89;
sfr TL0   = 0x8A;
sfr TL1   = 0x8B;
sfr TH0   = 0x8C;
sfr TH1   = 0x8D;
sfr IE    = 0xA8;
sfr IP    = 0xB8;
sfr SCON  = 0x98;
sfr SBUF  = 0x99;

/*  8052 Extensions  */
sfr T2CON  = 0xC8;
sfr RCAP2L = 0xCA;
sfr RCAP2H = 0xCB;
sfr TL2    = 0xCC;
sfr TH2    = 0xCD;


/*  BIT Registers  */
/*  PSW  */
sbit CY    = PSW^7;
sbit AC    = PSW^6;
sbit F0    = PSW^5;
sbit RS1   = PSW^4;
sbit RS0   = PSW^3;
sbit OV    = PSW^2;
sbit P     = PSW^0; //8052 only
```

```
/*  TCON  */
sbit TF1   = TCON^7;
sbit TR1   = TCON^6;
sbit TF0   = TCON^5;
sbit TR0   = TCON^4;
sbit IE1   = TCON^3;
sbit IT1   = TCON^2;
sbit IE0   = TCON^1;
sbit IT0   = TCON^0;

/*  IE  */
sbit EA    = IE^7;
sbit ET2   = IE^5; //8052 only
sbit ES    = IE^4;
sbit ET1   = IE^3;
sbit EX1   = IE^2;
sbit ET0   = IE^1;
sbit EX0   = IE^0;

/*  IP  */
sbit PT2   = IP^5;
sbit PS    = IP^4;
sbit PT1   = IP^3;
sbit PX1   = IP^2;
sbit PT0   = IP^1;
sbit PX0   = IP^0;

/*  P3  */
sbit RD    = P3^7;
sbit WR    = P3^6;
sbit T1    = P3^5;
sbit T0    = P3^4;
sbit INT1  = P3^3;
sbit INT0  = P3^2;
sbit TXD   = P3^1;
sbit RXD   = P3^0;
```

```
/*  SCON  */
sbit SM0   = SCON^7;
sbit SM1   = SCON^6;
sbit SM2   = SCON^5;
sbit REN   = SCON^4;
sbit TB8   = SCON^3;
sbit RB8   = SCON^2;
sbit TI    = SCON^1;
sbit RI    = SCON^0;

/*  P1  */
sbit T2EX  = P1^1; // 8052 only
sbit T2    = P1^0; // 8052 only

/*  T2CON  */
sbit TF2    = T2CON^7;
sbit EXF2   = T2CON^6;
sbit RCLK   = T2CON^5;
sbit TCLK   = T2CON^4;
sbit EXEN2  = T2CON^3;
sbit TR2    = T2CON^2;
sbit C_T2   = T2CON^1;
sbit CP_RL2 = T2CON^0;

#endif
```

附录五　器件说明

1）实验中使用的一般器件说明

分离元器件（参考价 0.2/件）

名称	英文名	封装	估计价格
电阻	RES2	AXIAL0.?(注1)	0.2元/件
普通电容	CAP	同上	0.2元/件
开关	SW_SPST	同上	1元/件
	SW_DIP4	DIP8	4元/件
晶振	CRYSTAL	AXIAL0.?(注1)	0.5元/件
二极管	DIODE	AXIAL0.?(注1)	0.2元/件
电解电容	ELECTRO1	RAD0.？ (注2)	0.5元/件
发光管	LED	LEDAXIAL	0.2元/件

注1：AXIAL0.?表示一系列矩形双管脚封装，如AXIAL0.3，AXIAL0.4等，后面的数字代表两管脚之间的距离（以英寸表示）。在实际中可以根据器件的具体规格使用，比较通用的规格是AXIAL0.3。

注2：RAD0.?代表一系列圆形区域，双管脚封装。具体使用同注1。

分离元器件的价格区间变动较大，特别是指标要求相差较大的情况下。

中大规模集成电路（参考价 10元/片）

1．名称：DAC0832；功能：数模转换器；封装：DIP20；

管脚号	名称	说明	管脚号	名称	说明
1	CS	片选	10	GND	
2	WR1		11	IOUT1	输出1
3	GND	地	12	IOUT2	输出2
4-7	DI3-DI0	数据输入	17	Xfer	启动
13-16	DI7-DI4		18	WR2	
8	VREF	基准电压	19	ILE	
9	RF		20	VCC	电源

2．名称：8031；功能：CPU；封装：DIP40；

管脚定义

管脚号	名称	说明	管脚号	名称	说明
1-8	P1.0-P1.7		18	X1	晶振
9	RESET	复位	19	X2	
10	RXD		20	GND	地
11	TXD		21-28	P2.0-P2.7	
12	INT0		29	PSEN	
13	INT1		30	ALE	
14	T0		31	EA	
15	T1		39-32	P0.0-P0.7	
16	WR	写	40	VCC	电源
17	RD	读			

3. 名称：ADC0809；功能：8输入模数转换器；封装：DIP28；

管脚号	名称	说明	管脚号	名称	说明
1—5	IN3-IN7	输入	15	D5	
6	START	启动	16	Ref-	基准电压
7	EOC	结束转换	17	D7	
8	D4		18	D3	
9	OE	使能	19	D2	
10	CLK	时钟	20	D1	
11	VCC	电源	21	D0	
12	Ref+	基准电压	23-25	ADD1-3	输入选择
13	GND	地	26-28	IN0-IN2	输入
14	D6				

小规模集成电路 （参考价 2元/片）

1．名称：74LS138；功能：3—8译码器；封装：DIP16；

管脚号	名称	说明	管脚号	名称	说明
1	A	输入	7	Y0	输出

2	B	输入	8	GND	地
3	C	输入	9－15	Y1－Y7	输出
4	E1	使能 低有效	16	VCC	电源
5	E2	使能 低有效			
6	E3	使能 高有效			

2．名称：74LS244；功能：三态总线；封装：DIP20；

管脚号	名称	说明	管脚号	名称	说明
2,4,6,8	1A1－1A4	1输入	1	1G	1允许
18,16,14,12	1Y1－1Y4	1输出	19	2G	2允许
11,13,15,17	2A1－2A4	2输入	10	GND	地
9,7,5,3	2Y1－2Y4	2输出	20	VCC	电源

3．名称：74LS240；功能：三态总线驱动器；封装：DIP20；

功能及管脚信息与74LS244类似，不同点在于，本电路将输入反相后输出。

4．名称：74LS374；功能：三态锁存器；封装：DIP20；

管脚号	名称	说明	管脚号	名称	说明
3,4,7,8,13,14,17,18	D0-D7	输入	2,5,6,9,12,15,16,19	Q0－Q7	输出
1	OE	使能	11	CLK	时钟
10	GND	地	20	VCC	电源

5．名称：MC1413；功能：驱动器；封装：DIP16

注：此集成电路为7组三极管驱动器，当输入管脚为高时，对应的三极管导通，在输出管脚与地之间形成通路。

管脚号	名称	说明	管脚号	名称	说明
1－7	IN1-IN7	输入	16－10	OUT1-OUT7	输出
8	GND	地	9	VCC	电源

6．名称：74LS125；功能：四独立三态门；封装：DIP14；

管脚：2入3出1控制、5入6出4控制、9入8出10控制、12入11出13控制，7为地，14为电源。

7．名称：74LS32；功能：四独立或门；封装：DIP14；

管脚：1、2入3出，4、5入6出，9、10入8出，12、13入11出，7为地，14为电源。

8. 名称：74LS08；功能：四独立与门；封装：DIP14；
管脚：1、2入3出，4、5入6出，9、10入8出，12、13入11出，7为地，14为电源。

9. 名称：74LS00；功能：四独立与非门；封装：DIP14；
管脚：1、2入3出，4、5入6出，9、10入8出，12、13入11出，7为地，14为电源。

10. 名称：74LS04；功能：六独立非门；封装：DIP14；
管脚：1入2出，3入4出，5入6出，9入8出，11入10出，13入12出，7为地，14为电源。

11. 名称：74LS02；功能：四独立或非门；封装：DIP14；
管脚：2、3入1出，5、6入4出，8、9入10出，11、12入13出，7为地，14为电源。

12. 名称：74LS86；功能：四独立异或门；封装：DIP14；
管脚：1、2入3出，4、5入6出，9、10入8出，12、13入11出，7为地，14为电源。

2）重要器件说明和功能介绍

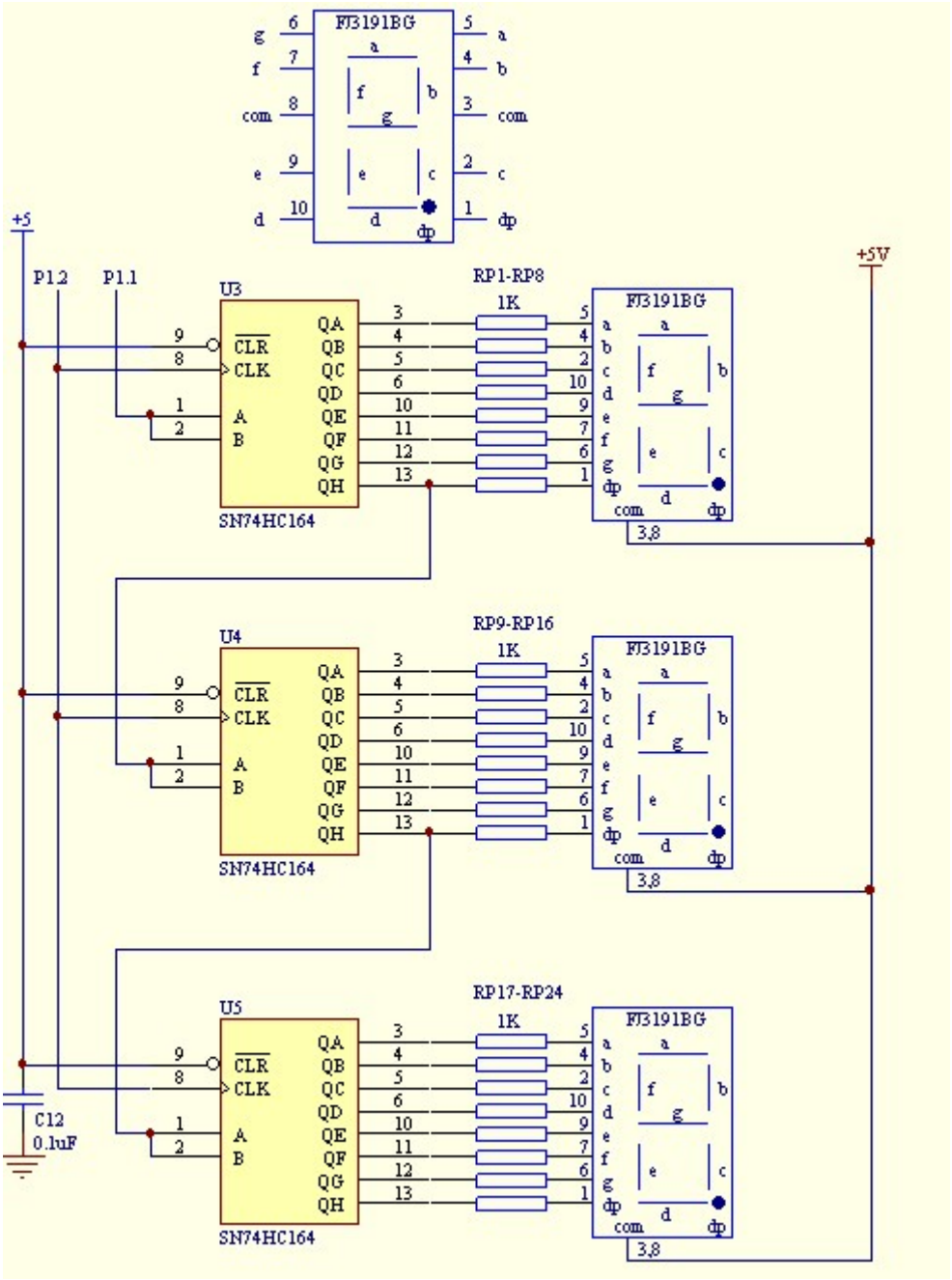
1. 74HC164级联驱动数码管显示

74HC164是高速CMOS 器件。74HC164是8位边沿触发式移位寄存器，串行输入数据，然后并行输出。数据通过两个输入端（A或B）之一串行输入；任一输入端可以用作高电平使能端，控制另一输入端的数据输入。两个输入端或者连接在一起，或者把不用的输入端接高电平，一定不要悬空。

时钟（CLK）每次由低变高时，数据右移一位，输入到Q0，Q0 是两个数据输入端（A和B）的逻辑与，它将上升时钟沿之前保持一个建立时间的长度。

主复位(CLR)输入端上的一个低电平将使其它所有输入端都无效，同时非同步地清除寄存器，强制所有的输出为低电平。

本实验采用3个74HC164级联控制三个数码管的显示，具体实验原理如下图所示。其中使用单片机P1.2作为模拟串口数据，使用P1.3模拟串口时钟，CLR端接高电平。使用上一个74HC164的Q7作为下一个74HC164的输入端。


FJ3191BG
g
f
com
e
d
a
b
com
c
dp
+5
P1.2
P1.1
U3
RP1-RP8
1K
CLR
CLK
A
B
QA
QB
QC
QD
QE
QF
QG
QH
SN74HC164
3,8
+5V
U4
RP9-RP16
U5
RP17-RP24
C12
0.1uF

2. DS18B20的介绍

本书实验中使用DS18B20这一种单总线式数字温度传感器，它具有微型化、低功耗、高性能、搞干扰能力强、易配处理器等优点，可直接将温度转化成串行数字信号给单片机处理。它的温度测量范围为－55℃～＋125℃，可编程为9位～12位A/D转换精度，固有测温分辨率0.5℃，测温分辨率最高可达0.0625℃，被测温度用符号扩展的16位数字量方式串行输出。DS18B20有三个引脚，其中只有一位数据输入输出操作引脚，通常称为DQ。

DS18B20工作过程如下：初始化——ROM操作命令——存储器操作命令——处理数据。其中初始化序列过程如下：将DS18B20的数据总线拉低500μs以上，然后释放，DS18B20收到信号后等待16～60μs左右，之后发出60～240μs的存在低脉冲，单片机收到此信号表示初始化成功。

ROM操作命令有：

指令	代码	解释
Read ROM（读ROM）	[33H]	此命令允许总线主机读DS18B20的8位产品系列编码，唯一的48位序列号，以及8位的CRC。
Match ROM（匹配ROM）	[55H]	此命令后继以64位的ROM数据序列，允许总线主机对多点总线上特定的DS18B20寻址。
Skip ROM（跳过ROM）	[CCH]	在单点总线系统中，此命令通过允许总线主机不提供64位ROM编码而访问存储器操作来节省时间。
Search ROM（搜索ROM）	[F0H]	此命令允许总线控制器用排除法识别总线上的所有从机的64位编码。
Alarm search（告警搜索）	[ECH]	此命令的流程与搜索ROM命令相同。但是，仅在最近一次温度测量出现告警的情况下，DS18B20才对此命令作出响应。

存储器操作命令如下：

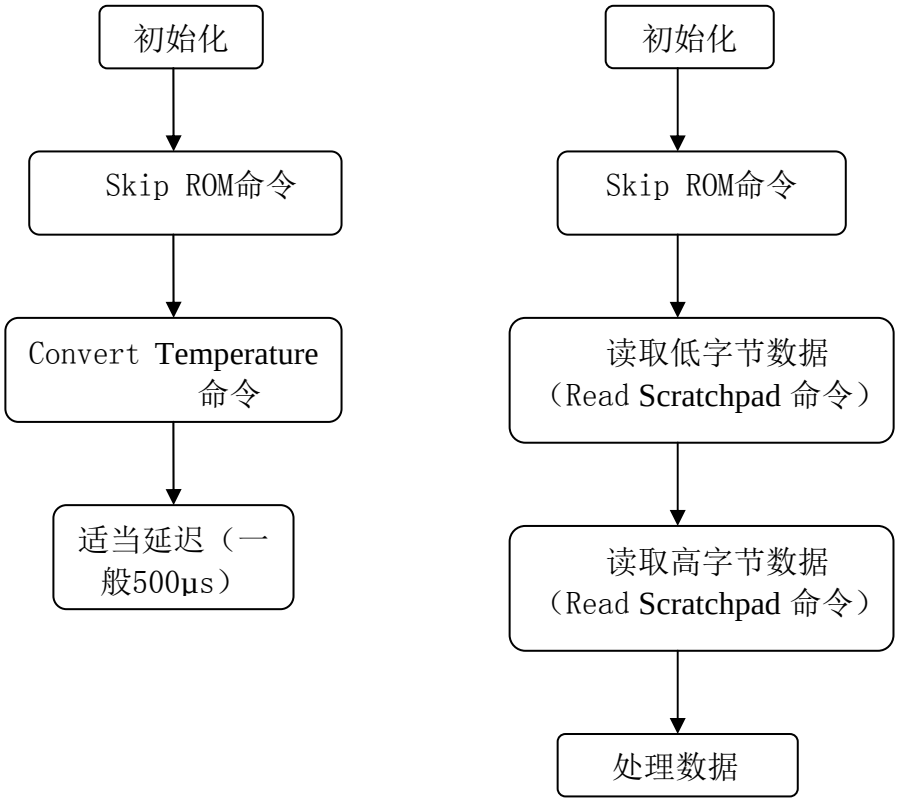
命令	代码	解释
Write Scratchpad (写暂存存储器)	[4EH]	此命令向DS18B20的暂存器中写入数据，开始位置在地址2。接下来写入的两个字节将被存到暂存器中的地址位置2和3。
Read Scratchpad (读暂存存储器)	[BEH]	此命令读取暂存器的内容。读取将从字节0开始，一直进行下去，直到第9（字节8，CRC）字节读完。

Copy Scratchpad (复制暂存存储器)	[48H]	此命令把暂存器的内容拷贝到DS18B20的EPROM存储器里，即把温度报警触发字节存入非易失性存储器里。
Convert Temperature (温度变换)	[44H]	这条命令启动一次温度转换而无需其他数据。
Recall EPROM (重新调出)	[B8H]	此命令把贮存在EPROM中温度触发器的值重新调至暂存存储器。
Read Power Supply (读电源)	[B4H]	对于在此命令发送至DS18B20之后所发出的第一读数据的时间片，器件都会给出其电源方式的信号：“0”=寄生电源供电，“1”=外部电源供电。

处理数据

DS18B20 的高速暂存存储器共有 9 个字节。当温度转换命令发布后，经转换所得的温度值以二字节补码形式存放在高速暂存存储器的第 0 和第 1 个字节。这是 12 位精度转换后得到的 16 位数据，其中前面 5 位为符号位。如果测得的温度大于或等于 0，这 5 位为 0，只要将测到的数值乘于 0.0625 即可得到实际温度；如果温度小于 0，这 5 位为 1，测到的数值需要取反加 1 再乘于 0.0625 即可得到实际温度。单片机可通过单线接口读到该数据，读取时低位在前，高位在后。

单片机控制 DS18B20 完成温度转换和读取数据的具体流程分别如下：



写操作（包括写指令）具体过程如下：通过单总线采取移位的方式来向DS18B20写入数据，按照从低位到高位的顺序每次一位的方式写进去，需要满足写时间间隙的要求。在写数据时间隙的前15μs数据总线需要是被单片机拉置低电平，而后则将是芯片对总线数据的采样时间，采样时间在15～60μs，采样时间内如果单片机将总线拉高则表示写“1”，如果单片机将总线拉低则表示写“0”。每一位的发送都应该有一个至少 15μs的低电平起始位，随后的数据“0”或“1”应该在45μs 内完成。整个位的发送时间应该保持在60～120uS，否则不能保证通信的正常。

读操作（包括读数据）具体过程如下：也是通过移位的方法从DS18B20中读取数据，按照从低位到高位的顺序每次一位的方式读入，需要满足读时间间隙的要求。读时间隙时控制时的采样时间应该更加的精确才行，读时间隙时也是必须先由单片机产生至少 1μs的低电平，表示读时间的起始。随后在总线被释放后的15μs 中DS18B20 会发送内部数据位，这时单片机如

果发现总线为高电平表示读出“1”,如果总线为低电平则表示读出数据“0”。注意，必须在读间隙开始的 15μs 内读取数据位才可以保证通信的正确。

3.LCM的使用

YM12864C 是一种图形点阵液晶显示器。它主要采用动态驱动原理由行驱动控制器和列驱动器两部分组成了128(列)×64(行)的全点阵液晶显示。此显示器采用了 COB 的软封装方式，通过导电橡胶和压框连接LCD,使其寿命长，连接可靠。

YM12864C是全屏幕点阵,点阵数为 128(列)×64(行),可显示 8(每行)×4(行)个(16×16点阵)汉字，也可完成图形，字符的显示。与 CPU接口采用5 条位控制总线和8 位并行数据总线输入输出，适配M6800系列时序。内部有显示数据锁存器，自带上电复位电路。

一. 硬件说明

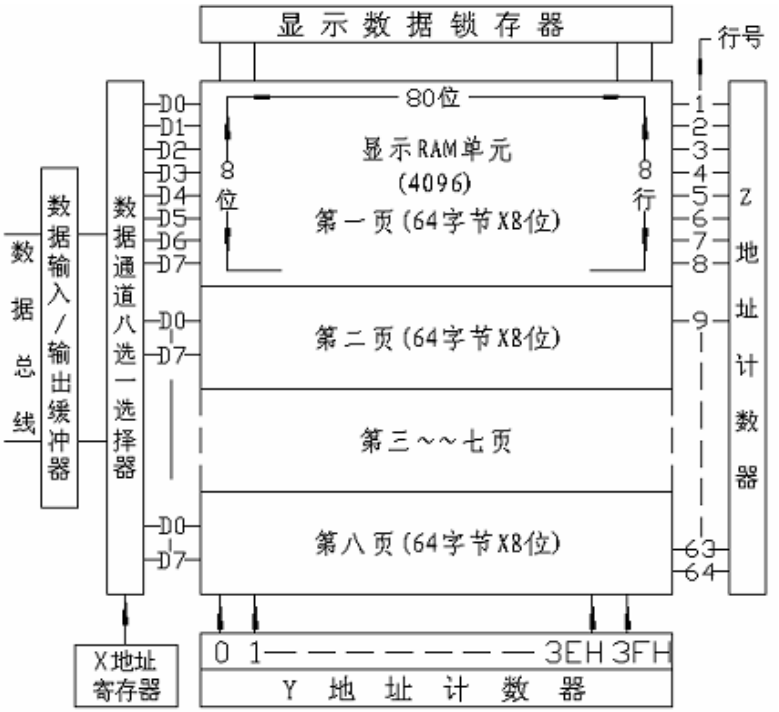
引脚特性如下表所示：

引脚号	引脚名称	级别	引脚功能描述
1	CS1	H/L	片选信号，当/CS1=L时,液晶左半屏显示
2	CS2	H/L	片选信号，当/CS2=L时,液晶右半屏显示
3	VSS	0V	电源地
4	VDD	+5V	电源电压
5	V0	0至-10V	LCD驱动负电压，要求VDD-VLCD=10V
6	RS	H/L	寄存器选择信号
7	R/W	H/L	读/写操作选择信号
8	E	H/L	使能信号
9	DB0	H/L	八位三态并行数据总线
10	DB1		
11	DB2		
12	DB3		
13	DB4		
14	DB5		
15	DB6		
16	DB7		
17	RES	H/L	复位信号，低电平有效
18	VOUT	-10V	输出-10V的负电压（单电源供电）
19	LED+(EL)	+5V	背光电源,Idd≤960mA
20	LED-(EL)	0V	

YM12864C的液晶分为左边和右边两个64×64的子屏，分别通过CS1和CS2选通，每个子屏相应的内部寄存器是相互独立的。

二．寄存器功能说明

LCM的内部寄存器如下图所示：



内部扫描时钟实时将DDRAM数据通过光学震荡显示在LCM液晶屏上，微处理器将数据通过数据总线在时序电路的控制下写入DDDRAM某个单元，DDRAM单元的选择通过X，Y，Z 3个地址寄存器决定。CS1和CS2决定片选子屏，X地址寄存器决定子屏中显示单元页位置，从上至下，每8行，即一个字节为一页，范围从D0h ～ D7h ；Y地址寄存器决定子屏中显示单元列位置，范围从0 ～3Fh；Z地址寄存器决定行滚动的首行地址，首行范围从1～64 。下面将详细介绍每个寄存器的功能和使用方法。

1）显示数据RAM(DDRAM)

DDRAM（64×8×8 bits）是存储图形显示数据的。此RAM的每一位数据对应显示面板上一个点的显示（数据为 H）与不显示（数据为 L）。

2）I/O缓冲器（DB0～DB7）

I/O 缓冲器为双向三态数据缓冲器。是 LCM（液晶显示模块）内部总线与单片机总线的结合部。其作用是将两个不同时钟下工作的系统连接起来，实现通讯。I/O缓冲器在片选信号/CS 有效状态下,I/O 缓冲器开放，实现 LCM（液晶显示模块）与单片机之间的数据传递。当片选信号为无效

状态时，I/O缓冲器将中断LCM（液晶显示模块）内部总线与单片机数据总线的联系，对外总线呈高阻状态，从而不影响单片机的其他数据操作功能。

3）输入寄存器

输入寄存器用于接收在 单片机 运行速度下传送给 LCM（液晶显示模块）的数据并将其锁存在输入寄存器内，其输出将在LCM（液晶显示模块）内部工作时钟的运作下将数据写入指令寄存器或显示存储器内。

4）输出寄存器

输出寄存器用于暂存从显示存储器读出的数据，在单片机读操作时，输出寄存器将当前锁存的数据通过I/O缓冲器送入单片机数据总线上。

5）指令寄存器

指令寄存器用于接收单片机发来的指令代码，通过译码将指令代码置入相关的寄存器或触发器内。

6）状态字寄存器

状态字寄存器是 LCM（液晶显示模块）与单片机通讯时唯一的“握手”信号。状态字寄存器向单片机表示了 LCM（液晶显示模块）当前的工作状态。尤其是状态字中的“忙”标志位是单片机在每次对 LCM（液晶显示模块）访问时必须要读出判别的状态位。当处于“忙”标志位时，I/O 缓冲器被封锁，此时单片机对 LCM（液晶显示模块）的任何操作（除读状态字操作外）都将是无效的。

7）X地址寄存器

X 地址寄存器是一个三位页地址寄存器，其输出控制着 DDRAM 中 8 个页面的选 择，也是控制着数据传输通道的八选一选择器。X 地址寄存器可以由单片机以指令形式设置。X 地址寄存器没有自动修改功能，所以要想转换页面需要重新设置 X 地址寄存器的内容。

8）Y地址计数器

Y 地址计数器是一个 6 位循环加一计数器。它管理某一页面上的 64 个单元。Y地址计数器可以由单片机以指令形式设置，它和页地址指针结合唯一选通显示存储器的一个单元，Y地址计数器具有自动加一功能。在显示存储器读/写操作后 Y地址计数将自动加一。当计数器加至3FH后循环归零再继续加一。

9）Z地址计数器

Z 地址计数器是一个 6 位地址计数器，用于确定当前显示行的扫描地址。Z 地址计数器具有自动加一功能。它与行驱动器的行扫描输出同步，选择相应的列驱动的数据输出。

10）显示起始行寄存器

显示起始行寄存器是一个6位寄存器，它规定了显示存储器所对应显示屏上第一行的行号。该行的数据将作为显示屏上第一行显示状态的控制信号。

11）显示开/关触发器

显示开/关触发器的作用就是控制显示驱动输出的电平以控制显示屏的开关。在触发器输出为“关”电平时，显示数据锁存器的输入被封锁并将输出置“0”,从而使显示驱动输出全部为非选择波形，显示屏呈不显示状态。在触发器输出为“开”电平时，显示数据锁存器被控制，显示驱动输出受显示驱动数据总线上数据控制，显示屏将呈显示状态。

12）复位端/RES

复位端/RES 用于在 LCM（液晶显示模块）上电时或需要时实现硬件电路对 LCM（液晶显示模块）的复位。该复位功能将实现：设置显示状态为关显示状态，显示起始寄存器清零。显示RAM第1行对应显示屏上的第1行，在复位期间状态字中RESET位置1。

三．软件功能说明

3.1指令表

指令名称	控制信号		控制代码							
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
显示开关设置	0	0	0	0	1	1	1	1	1	D
显示起始行设置	0	0	1	1	L5	L4	L3	L2	L1	L0
页面地址设置	0	0	1	0	1	1	1	P2	P1	P0
列地址设置	0	0	0	1	C5	C4	C3	C2	C1	C0
读取状态字	0	1	BUSY	0	ON/OFF	RESET	0	0	0	0
写显示数据	1	0	数据							
读显示数据	1	1	数据							

详细解释各个指令功能

1）读状态字

状态字是单片机了解LCM（液晶显示模块）当前状态，或 LCM 向单片机提供其内部状态的唯一的信息渠道。

BUSY 表示当前 LCM 接口控制电路运行状态。BUSY=1 表示LCM 正在处理单片机发过来的指令或数据。此时接口电路被封锁，不能接受除读状态

字以外的任何操作。BUSY=0表示 LCM 接口控制电路已外于“准备好”状态，等待单片机的访问。

ON/OFF 表示当前的显示状态。ON/OFF 1 表示关显示状态，ON/OFF 0 表示开显 示状态。

RESET 表示当前LCM 的工作状态，即反映/RES 端的电平状态。当/RES 为低电平状态时，LCM 处于复位工作状态，标志位 RESET=1。当/REST 为高电平状态时，LCM 为正常工作状态，标志位 RESET=0。

在指令设置和数据读写时要注意状态字中的 BUSY 标志。只有在 BUSY=0 时，单片机对 LCM 的操作才能有效。因此单片机在每次对 LCM 操作之前，都要读出状态字判断BUSY是否为“0”。若不为“0”，则单片机需要等待，直至BUSY=0 为止。

2)显示开关设置

该指令设置显示开/关触发器的状态，由此控制显示数据锁存器的工作方式，从而控制显示屏上的显示状态。D位为显示开/关的控制位。当D=1 为开显示设置，显示数据锁存器正常工作，显示屏上呈现所需的显示效果。此时在状态字中ON/OFF=0。当D=0为关显示设置，显示数据锁存器被置零，显示屏呈不显示状态，但显示存储器并没有被破坏，在状态字中ON/OFF=1。

3)显示起始行设置

该指令设置了显示起始行寄存器的内容。LCM 通过/CS 的选择分别具有 64 行显示的管理能力，该指令中 L5～L0 为显示起始行的地址，取值在 0～3FH（1～64 行）范围内，它规定了显示屏上最顶一行所对应的显示存储器的行地址。如果定时间隔地，等间距地修改（如加一或减一）显示起始行寄存器的内容，则显示屏将呈现显示内容向上或向下平滑滚动的显示效果。

4)页面地址设置

该指令设置了页面地址—X 地址寄存器的内容。LCM将显示存储器分成 8 页，指令代码中 P2～P0 就是要确定当前所要选择的页面地址，取值范围为 0～7H ，代表1～8 页。该指令规定了以后的读/写操作将在哪一个页面上进行。

5)列地址设置

该指令设置了 Y 地址数计数器的内容，LCM 通过/CS 的选择分别具有64 列显示的管理能力，C5～C0=0～3FH（1～64）代表某一页面上的某一单元地址，随后的一次读或写数据将在这个单元上进行。Y 地址计数器具有自动加一功能，在每一次读/写数据后它将自动加一，所以在连续进行读/写数据时，Y 地址计数器不必每次都设置一次。页面地址的设置和列地址的设置将显示存储器单元唯一地确定下来，为后来的显示数据的读/写作了地址的选通。

6)写显示数据

该操作将 8位数据写入先前已确定的显示存储器的单元内。操作完成后列地址计数器自动加一。

7)读显示数据

该操作将 LCM 接口部的输出寄存器内容读出，然后列地址计数器自动加一。

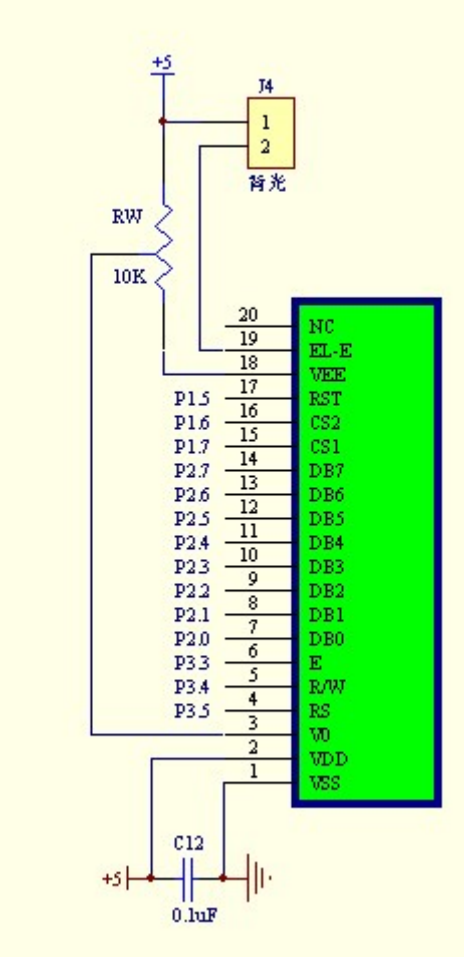
3.2控制时序表

CS1	CS2	RS	R/W	E	DB7~DB0	功能
X	X	X	X	0	高阻	总线释放
1	1	0	0	下降沿	输入	写指令代码
1	1	0	1	1	输出	读状态字
1	1	1	0	下降沿	输入	写显示数据
1	1	1	1	1	输出	读显示数据

3.3 DDRAM表

	CS1=1					CS2=1					
Y=	0	1	. . .	62	63	0	1	. . .	62	63	行号
X=0 ↓ X=7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7
	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7
	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7

四．硬件电路图如下



五．图形和字模生成软件

在液晶显示中，自定义图形和文字的字模对应的字节表需要使用专门的字模软件来生成。可以使用PCtoLCD2002字模软件提取。

六．程序设计

YM12864C的使用过程包括初始化和读写操作。下面举例给出C51编写的左半屏写数据。

```
void data_w_l(){
        P2=0xff;
        cs2=0;
        cs1=1;
        rs=0;
        rw=1;
        e=1;
```

```
                do{;}
                while(p27==1);     //直到读状态字BUSY位为0
                e=0;
                rs=1;
                rw=0;
                P2=lcm_data;       //将数据写入左半屏
                e=1;
_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();
_nop_();_nop_();
_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();_nop_();
_nop_();_nop_();
                                        //延时写入数据
                e=0;
                cs1=0;
              }
```

4.STC12C5A16AD的A/D转换

STC12C5A16AD的A/D转换口在P1口(P1.7～P1.0),有8路10位高速A/D转换器，速度可以达到250KHz (25万次/次)。上电复位后P1口为弱上拉型I/O口，可以通过软件设置将8路中的任何一路设置为A/D转换，不需要作为A/D使用的口可继续作为I/O口使用。

P1ASF(地址为9Dh)：需要作为A/D使用的端口需先将P1ASF特殊功能寄存器中的相应位置设为‘1’，将相应的端口设置为模拟功能。（该寄存器是只写寄存器，度无效。）

ADC_CONTR(地址为0BCh)：A/D转换控制特殊功能寄存器，复位值00000000。

B7	B6	B5	B4	B3	B2	B1	B0
ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0

CHS2,CHS1,CHS0：三位进行模拟输入通道选择。当数值为000时，选择P1.0作为A/D输入来用，当数值为001时，选择P1.1作为A/D输入来用，以此类推。

ADC_START：模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换，转换结束后为0.

ADC_FLAG ::模数转换器转换结束标志位，当A/D转换完成后，ADC_FLAG为1，需要用软件清零。

SPEED1，SPEED0 ： 模数转换器转换速度控制位。当数值为11时，A/D转换周期为90个时钟周期;当数值为10时,A/D转换周期为180个时钟周期；当数值为01时，A/D转换周期为360个时钟周期；当数值为00时，A/D转换周期为540个时钟周期；

ADC_POWER ： 数模转换器电源控制位。数值为0时，关闭ADC电源；当数值为1时，打开A/D转换器电源。

AUXR1（地址为0A2h）寄存器的ADRJ位（即B2位）：A/D转换结果寄存器(ADC_RES,ADC_RESL)的数据格式调整控制位。

ADC_RES(地址为0BDh),ADC_RESL(地址为0BEh) ：A/D转换结果特殊功能寄存器。

当ADRJ位为0时，10位A/D转换结果的高8位放在ADC_RES中，低2位放在ADC_RESL的低2位。

当ADRJ位为1时，10位A/D转换结果的高2位存放在ADC_RES低2位中，低8位存放在ADC_RESL中。

假定Vin为模拟输入通道输入电压，Vcc为单片机实际工作电压。

ADRJ=0，模数转换结果计算公式如下：取10位结果（ADC_RES[7:0]，ADC_RESL[1:0]）=

1024 × Vin / Vcc

ADRJ=0，模数转换结果计算公式如下：取8位结果（ADC_RES[7:0]）= 1024 × Vin / Vcc

ADRJ=1，模数转换结果计算公式如下：取10位结果（ADC_RESL[1:0]，ADC_RES[7:0]）=

1024 × Vin / Vcc

IE ：中断允许寄存器

	B7	B6	B5	B4	B3	B2	B1	B0
IE	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0

如果要允许A/D转换中断则需要将相应的控制位置1：

1）将EADC置1，允许ADC中断，这是ADC中断的中断控制位。

2）将EA置为1，打开单片机总中断控制位，此位不打开，也是无法产生ADC中断的。

5. STC12C5A16AD的EEPROM应用

一. IAP及EEPROM新增寄存器介绍

IAP_DATA(地址为C2h) : ISP/IAP操作时的数据寄存器，ISP/IAP从EEPROM读出的数据放在此处，向EEPROM写的数据也要放到此处。

IAP_ADDRH(地址为C3h) : ISP/IAP操作时的地址寄存器的高八位。

IAP_ADDRL(地址为C4h) : ISP/IAP操作时的地址寄存器的低八位。

IAP_CMD(地址为C5h) : ISP/IAP操作时的命令模式寄存器，需要命令触发寄存器触发方可有效。寄存器的高六位保留，低两位做命令字。当命令字为00时，处于待机模式；为01时，进行字节读操作；为10时，进行字节写操作；为11时，进行扇区擦出工作。

IAP_TRIG(地址为C6h) : ISP/IAP操作时的命令触发寄存器。

在IAPEN（IAP_CONTR.7）=1时，对IAP_TRIG先写入5Ah，再写入A5h，ISP/IAP命令才会生效。

IAP_CONTR(地址为C7h) : ISP/IAP控制寄存器。

B7	B6	B5	B4	B3	B2	B1	B0	复位值
IAPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

IAPEN : ISP/IAP功能允许位。0：禁止ISP/IAP编程改变EEPROM，1：允许编程改变EEPROM。

SWBS : 0：软件选择从用户主程序区启动，1：从ISP程序区启动。

SWRST: 0：不操作，1：产生软件系统复位，硬件自动清零。

CMD_FAIL: 如果送了ISP/IAP命令，并对ISP_TRIG送5Ah/A5h触发失败，则为1，需由软件清零。

WT2，WT1，WT0：三位用来设置等待时间。当系统时钟<1MHz,设置为7；当系统时钟<2MHz,设置为6；当系统时钟<3MHz,设置为5；当系统时钟<6MHz,设置为4；当系统时钟<12MHz,设置为3；当系统时钟<20MHz,设置为2；当系统时钟<24MHz,设置为1；当系统时钟<30MHz,设置为0。

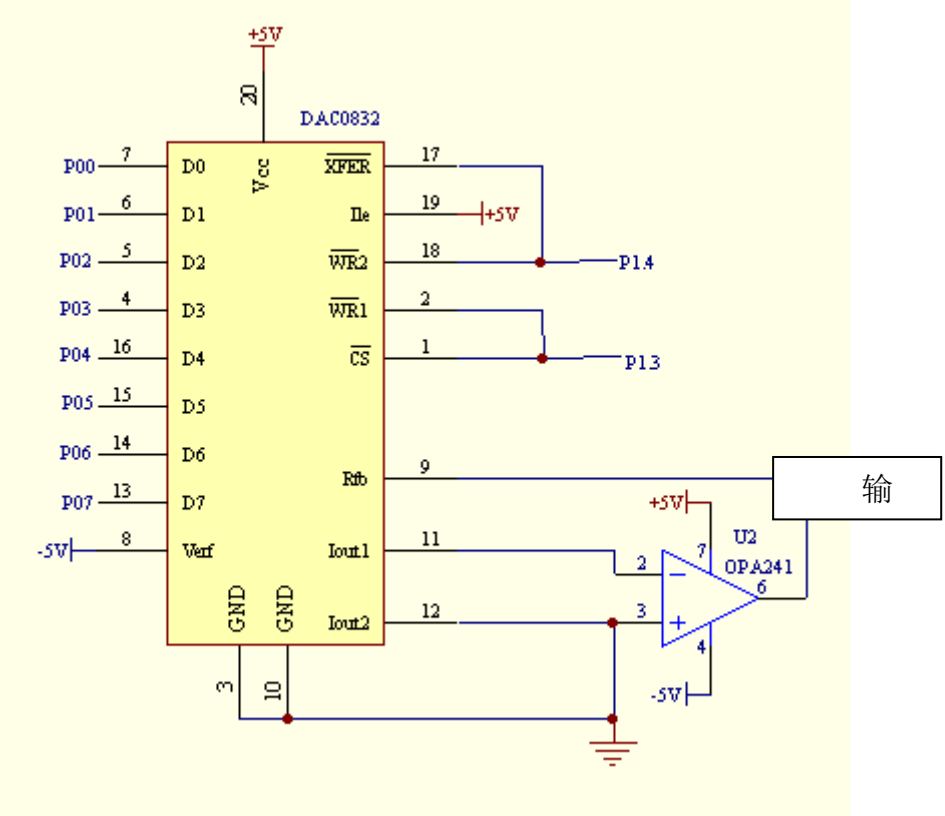
二. EEPROM 地址

STC12C5A16AD的EEPROM具有8K字节数，扇区数为16，起始扇区首地址为0000h，结束扇区末位地址为1FFFh。每一扇区大小为100h。

6.数模转换器DAC0832的介绍与使用

在数字电子技术的很多应用场合往往需要把数字量转换为模拟量，称为数/模转换器（D/ A转换器，简称DAC）。本实验将采用大规模集成电路DAC0832实现D / A转换。ADC0832 为8位分辨率A/D转换芯片，其最高分辨可达256级，可以适应一般的模拟量转换要求。DAC0832有8个输入端，每个输入端是8位二进制数的一位，有一个模拟输出端，输入可有2^8 ＝256个不同的二进制组态，输出为256个电压之一，即输出电压不是整个电压范围内任意值，而只能是256个可能值。输出电压与输入的数字量成正比，实现了从数字量到模拟量的转换。

本书实验该器件的实验原理图如下所示。



各引脚功能如下：

D0～D7：数据输入线。

CS：输入寄存器选择信号，低电平有效。

WR1：写信号1，低电平有效。

XFER：数据传送信号，低电平有效。

WR2：写信号2，低电平有效。
Verf：基准电源输入引脚。
Rfb：反馈信号输入引脚，反馈电阻在芯片内部。
Iout1,Iout2：电流输出引脚。
VCC：电源输入引脚。
AGND：模拟接地信号。
NGND：数字接地信号。

由于DAC0832输出的是电流，要转换为电压，本书实验通过外接运算放大器实现转换。D0～D7连接单片机P0口，CS与WR1连接单片机P1.3口，XFER与WR2连接单片机P1.4口。编写C51程序时，只需将P1.3，P1.4拉低即可根据P0口数据进行数据转换形成电压值，注意转换速率（往P1口送数据的时间间隔）。

附录六 实验设备原理框图

图1：压力测量和精度分析

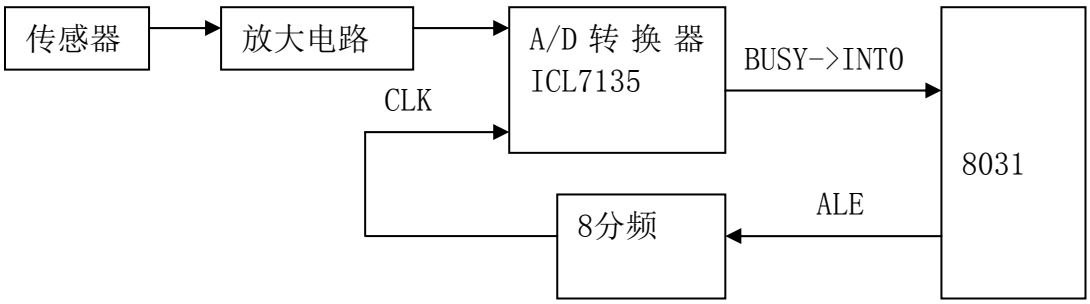


图2：超声波测距

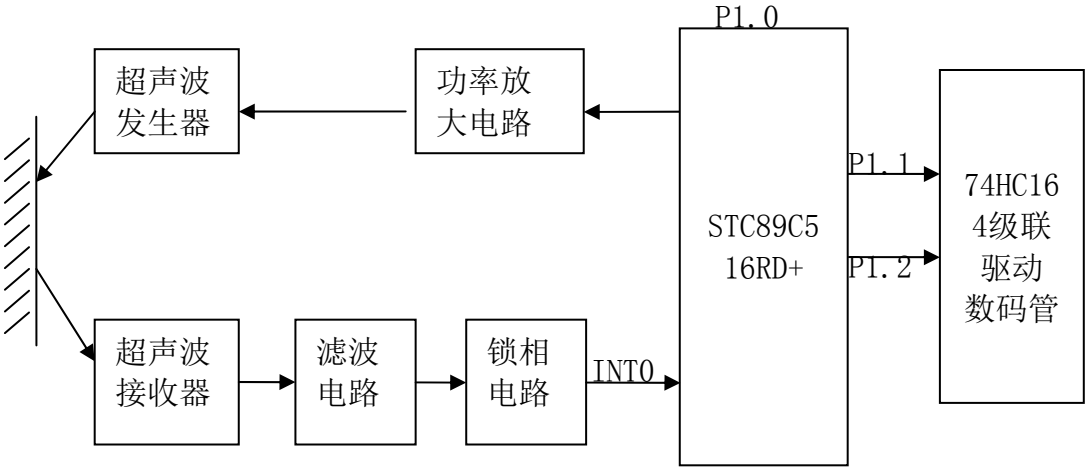


图3：温度测量

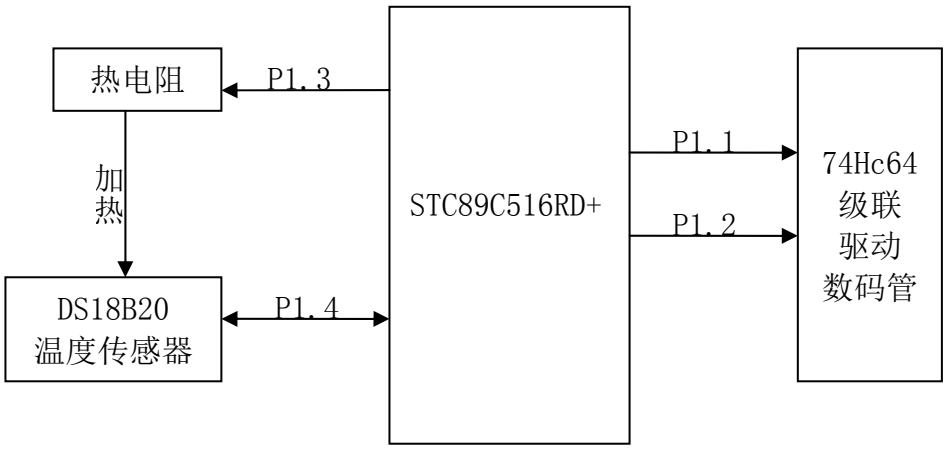
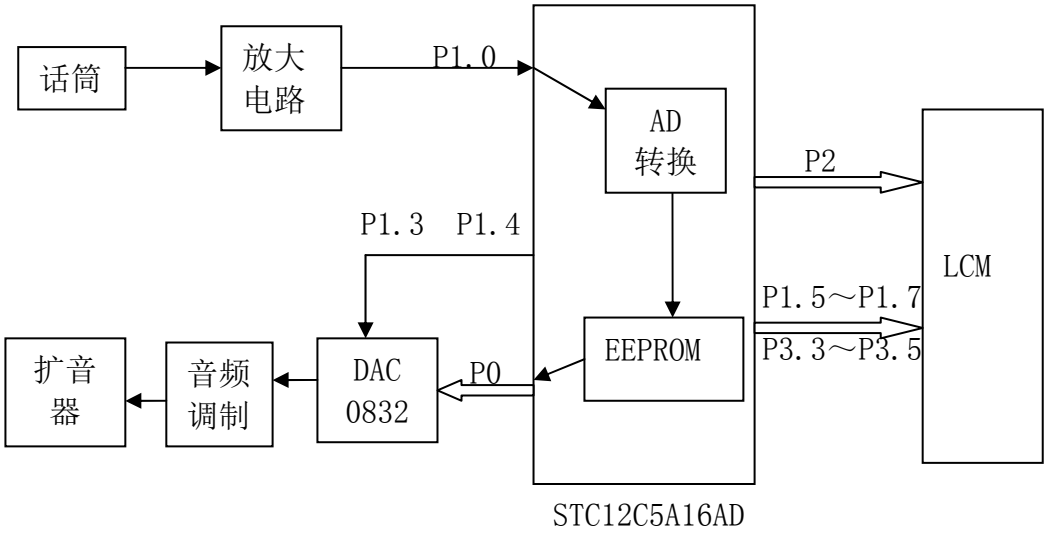

热电阻
P1.3
加热
DS18B20
温度传感器
P1.4
STC89C516RD+
P1.1
P1.2
74Hc64
级联
驱动
数码管

图4：音频录制与播放


话筒
放大
电路
P1.0
AD
转换
P2
LCM
P1.3 P1.4
P1.5～P1.7
P3.3～P3.5
EEPROM
扩音
器
音频
调制
DAC
0832
P0
STC12C5A16AD