《单片机系统及应用》

实验讲义

2019 年 5 月

前　　言

单片机又称微控制器，它是把中央处理器（CPU）、存储器、中断系统、定时器/计数器、串行接口、并行接口等功能部件集成在一个芯片上的微型计算机。其特点是功能强、价格低、体积小、易扩展、可靠性高、灵活等被广泛应用于航空航天、军事、医疗、家电、工业控制、智能仪表等领域。

作为人才培养的高等学校，其计算机和电子信息类专业都开设了单片机方面的课程。单片机是一门应用性很强的课程，需要通过大量的实验和实践，学生才能进一步消化和理解课堂上讲授的单片机基本结构和基础理论，掌握单片机各功能部件的工作原理及其各功能部件之间相互协调的工作关系，进而掌握这门技术。

编者根据多年从事单片机教学和工程实践的经验，遵从“循序渐进，理论联系实际”的原则，从培养学生学习单片机的兴趣入手，先进行功能部件的原理性单元实验，后进行系统性设计实验，以逐步提高学生的应用开发能力。

为此，我们开发了四个实验教具，即单片机系统主机板，单片机系统键盘/显示板，并口与存储器扩充板和 A/D 和 D/A 转换板等。

本实验讲义中，设置这些实验的目的是：通过开设的实验，使学生能充分理解单片机各功能部件的工作原理，理解软件与硬件之间相辅相成的关系，掌握单片机应用系统的开发方法，为后续嵌入式系统课程的学习打下良好的理论与实践基础。

本实验讲义具有如下特点：

1、先验证性实验，后设计性实验；
2、实验内容由简到繁，由易到难；
3、前一个实验是后续实验的基础；
4、以软硬件结合的设计性实验为主，单纯语言的验证性实验为辅；
5、以理论课单元内容的实验为主，综合证性实验为辅；
6、硬件设计具有较强的灵活性，实验过程和结果生动、直观。

希望通过本课程实验，能提高学生的学习兴趣，加深学生对单片机各功能部件的理解，进而掌握解决软硬件结合的系统开发方法，为学生后续的学习和工作打下良好的基础。

目　　录

实验一　　MCS-51 单片机汇编语言编程练习

一. 实验目的

1. 熟悉单片机 Keil μVision2 集成开发环境的使用方法。

2. 理解单片机汇编语言指令的基本语法以及汇编语言程序设计的基本结构和编程方法。

3. 掌握单片机汇编语言顺序结构、分支结构和循环结构程序的设计方法。

4. 能够独立使用单片机汇编语言进行顺序结构、分支结构和循环结构程序的设计能力。对实际应用问题能够抽象出数学模型，并通过编程来实现数学模型所要完成的功能。

5. 掌握顺序结构、分支结构和循环结构程序的流程图绘制方法。

二.预习与思考

1. 预习理论教材中“顺序结构”、“分支结构”和“循环结构”等相关内容，掌握它们的实现方法。

2. 思考如何使用不同类型的单片机汇编语言指令，来完成分支、循环结构程序的设计，并总结归纳出所有具备类似分支循环功能的汇编指令。

三. 实验原理

1. 单片机汇编语言的指令分类

全球生产单片机的芯片厂商有数百家，不同厂商生产的单片机芯片都有自己特定的汇编语言指令系统对其进行支持，那么在众多不同类型的单片机汇编语言指令系统中，作为单片机的初学者是不是要一一来学，是不是需要面面俱到呢？回答是不需要。因为，尽管单片机芯片的生产厂商、芯片类型以及处理位数都不尽相同，但是所有的单片机芯片都有一个共同的祖先，那就是 Intel 公司生产的 MCS-51 系列单片机芯片。因此，只要掌握好 MCS-51 系列单片机的汇编语言指令，就可以举一反三、触类旁通，从而理解其他类型单片机芯片的汇编语言指令系统。

MCS-51 系列单片机的汇编语言指令，一共有 111 条，按照指令实现的功能不同，将这 111 条指令分成了五大类即：数据传送类指令、算术运算类指令、逻辑运算类指令、控制转移类指令以及位操作指令（也叫布尔变量操作指令）。

数据传送类指令的作用是将数据在单片机芯片内部或外部的不同部件间进行传送，它是五大类指令当中最基础、最重要，也是指令条数最多的一类指令。

算术运算类指令是使单片机进行加、减、乘、除、加 1、减 1 等不同功能的算术运算。

逻辑运算类指令单片机进行与、或、非、异或、左移、右移等不同功能的逻辑运算。

控制转移类指令的作用是控制程序的执行顺序，即控制程序是否顺序执行，何时进行分支，何时进行子程序的调用以及如何使程序不断循环执行等操作。

位操作类指令的作用是把二进制位由 1 变成 0 或者由 0 变成 1，另外还可以根据某些二进制位的值进行程序的控制与转移，位操作类是单片机特有的一类指令。

本次实验的内容，主要就是针对这些指令进行重点的练习。为了方便记忆，根据这五类指令的不同功能，将五大类指令编成了“顺口溜”，有助于初学者对五大类指令及功能的理解和记忆，总结如下：

（1）数据传送指令：28 条，作用是“传来传去”。

（2）算术操作指令：24 条，作用是“算来算去”（加减乘除等算术运算）。

（3）逻辑操作指令：25 条，作用也是“算来算去”（与、或、非、移位等逻辑运算）。

（4）控制转移指令：17 条，作用是“跳来跳去”。

（5）位操作指令：　17 条，作用是“变来变去”。

2. 单片机汇编语言程序设计的基本结构

在进行单片机汇编语言的程序设计时，通常有 4 种应用程序结构，即顺序结构、分支结构、循环结构以及主子调用结构。在具体程序设计的过程中，要根据实际情况灵活运用各种结构，有时在一个程序中需要将多种结构进行组合应用。这 4 种程序结构，如图 3.1 所示。本次实验着重顺序结构、分支结构以及主子调用结构汇编程序的设计，实验四将重点进行循环结构的汇编程序设计。

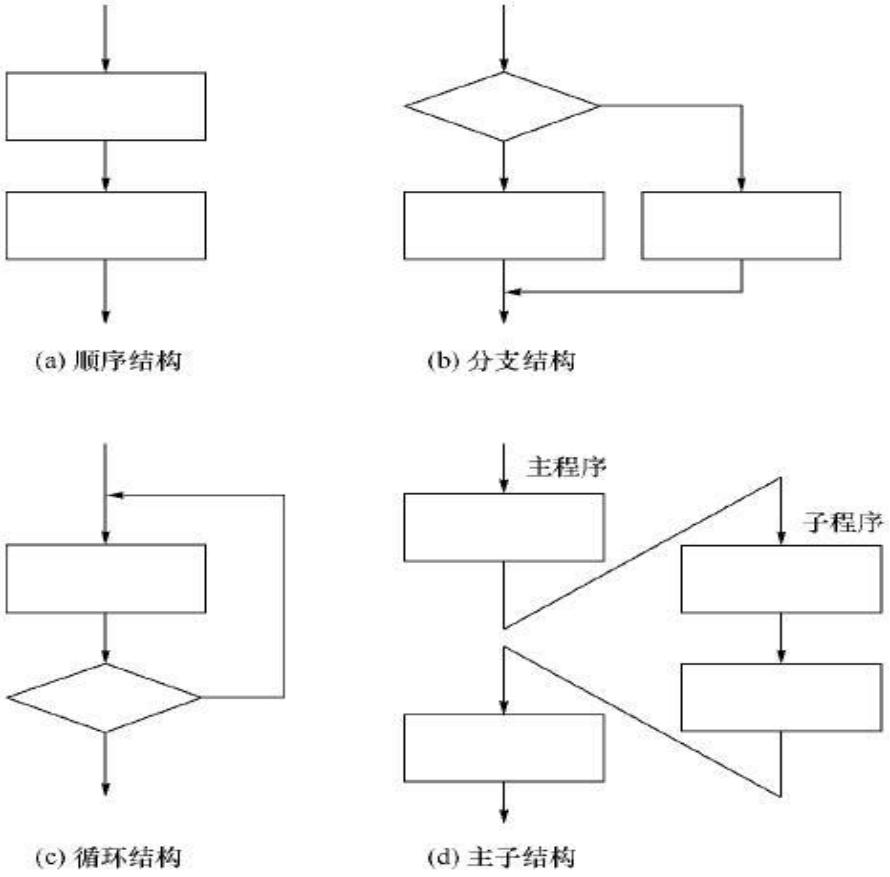


图 1.1　单片机汇编语言程序设计的的 4 种结构图

3. 单片机汇编语言程序设计的基本步骤

在掌握了单片机汇编语言的指令以及程序结构后，就要将两者结合起来进行汇编语言的程序设计了。汇编语言的程序设计是针对实际应用问题，使用 MCS-51 系列单片机的指令系统并结合汇编语言程序的设计结构，来编制汇编语言程序的过程。在程序设计的过程中，应该在保证实现程序功能的前提下，使程序占用的空间越小越好，执行速度越快越好。

汇编语言程序设计的基本步骤如下：

（1）分析应用问题，明确单片机系统的功能要求与设计目标，确定根据应用问题抽象出的算法数学模型以及具体设计思路。

（2）根据算法数学模型以及设计思路，绘制出程序实现的软件流程图。

（3）分配内、外存单元，即应用问题中的原始数据、中间数据、结果以及程序代码如何

在存储器中进行存放。

（4）按照软件程序流程图，进行汇编语言源程序的设计。

（5）使用 Keil μVision2 集成开发环境，在宿主计算机上输入程序，进行程序的汇编和调试运行，并在调试运行过程中找出错误进行更正，然后再次进行调试运行，直到程序顺利通过，得到正确的运行结果。

（6）在程序调试运行正确后，一定要及时编写程序设计文档说明，以备遗忘。

4. 实现分支结构的汇编语言指令

本次实验侧重练习分支结构的汇编语言程序设计，因此控制转移类指令是这些程序设计的核心，在这里对控制转移类指令进行简要的总结，指令的具体分类情况如下：

（1）无条件转移指令：主要包括 JMP、LJMP、SJMP 以及 AJMP 指令。

（2）条件转移指令：主要包括以下 4 小类指令。

① 累加器判 0 转移指令，JZ 和 JNZ；

② 循环转移指令，DJNZ；

③ 比较转移指令，CJNE；

④ 位转移指令，JC、JNC、JB、JNB 和 JBC 等；

（3）调用与返回指令：主要包括 LCALL、ACALL 以及 RET 指令。

从上面的总结可以看出，控制转移类指令主要包括 3 小类，即无条件转移指令、条件转移指令以及调用与返回指令。其中，无条件转移指令的作用是不需要任何条件，只要程序中遇到这样的指令，程序就会无条件转移到新的位置取指令继续执行，常用的无条件转移指令有 JMP、LJMP、SJMP、AJMP，它们的用法基本相同，不一一详细介绍。条件转移指令是最重要的也是最不容易理解的控制转移指令，它们的作用是当满足某个条件时，程序才会转移到某个新的标号地址指示的位置来执行新的程序指令，常用的条件转移指令有 4 种，前面已经叙述。子程序的调用与返回指令的作用主要用于主子结构的汇编语言程序设计。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. Keil μVision2 集成开发环境。

五. 实验内容

1. 顺序结构的汇编语言程序设计

已知单片机片内 ROM 的 50H 单元中存储的数据是 27H，请将此数据读入到单片机片内 RAM 的 60H 单元中，然后再从片内 RAM 的 60H 单元中，将这个数据写入到单片机片外 RAM 的 70H 单元中。请设计汇编语言程序、画出流程图，并调试出正确结果。具体调试要求：

（1）在 Keil μVision2 集成开发环境中，查询累加器 A、数据指针寄存器 DPTR、程序计数器 PC、通用寄存器 R0～R7 以及程序状态字寄存器 PSW 的内容。

（2）在 Keil μVision2 集成开发环境中，在存储器窗口中查询片内 RAM 和片外 RAM 存储单元的值，并给片内 ROM 的 50H 单元赋值为 27H。

（3）使用单步调试的方式来执行程序。在调试过程中，配合观察寄存器和存储器器窗口，检验程序的运行结果是否正确。

（4）连续执行程序，配合观察寄存器和存储器的窗口，检验运行结果是否正确。

【实验提示】:

此题虽然是比较简单的顺序结构汇编程序设计，不超过 10 条汇编指令就可以设计出来。但是，该题目却包含了 3 类主要的数据传输指令，即 MOV、MOVX 以及 MOVC 指令，而这 3 类指令恰好可以分别实现单片机片内 RAM、片外 RAM 以及片内和片外 ROM 中数据的传输。因此

本题的设计关键，在于理解好这 3 类汇编语言指令。此题参考源程序如下：

这里要注意，片内 ROM 的 50H 单元的数据，可以通过 Keil 软件在进入调试状态以后事先设定好。例如，本题可以在存储器的窗口中输入命令 c:0x50 然后回车，在显示出来的存储单元中找到片内 ROM 的 50H 单元，该单元通常默认的值为 00，用右键点击 00 后就会出现一个菜单，选择菜单的最后一项“更新存储器的值”，点击鼠标左键（如图 3.2 所示），在弹出的对话框中（如图 3.3 所示），输入题目中要求在片内 ROM 的 50H 单元中存放的数值 27H，然后点击确定，这时片内 ROM 的 50H 单元的值就设定好了（如图 3.4 所示）。

图 1.2　准备修改片内 ROM 的 50H 单元中的数值

图 1.3　修改片内 ROM 的 50H 单元中的数值

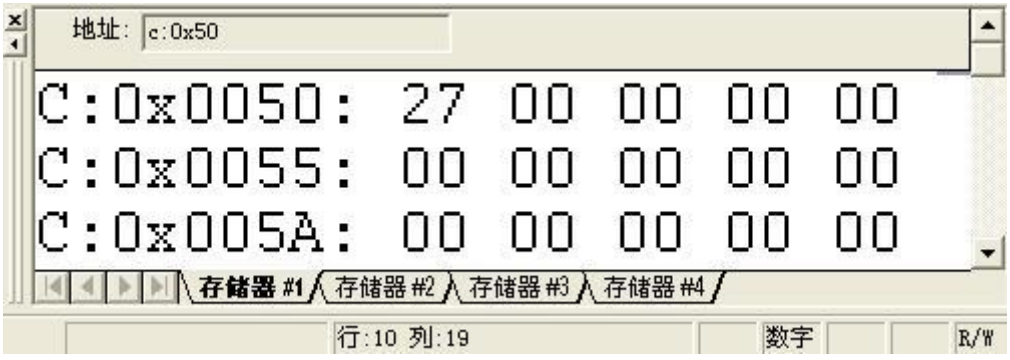


图 1.4　修改后片内 ROM 的 50H 单元中的数值为 27H

2. 分支结构的汇编语言程序设计

请完成如图 3.5 所示的符号函数功能设计。假定已知数据 X，存放在片内 RAM 的 50H 单元（X 的范围是－128～+127），通过符号函数表达式得到的结果 Y，存放在片内 RAM 的 51H

单元，请使用汇编语言的分支结构，根据图 3.6 所示的软件流程图编写程序。注意：在 Keil 软件中，负数使用补码表示，－1 的补码是 0FFH。具体调试要求：

（1）在 Keil μVision2 集成开发环境中，查询累加器 A、程序计数器 PC、通用寄存器 R0～R7 以及程序状态字寄存器 PSW 各个标志位的数据。

（2）在 Keil μVision2 集成开发环境中，查询片内 RAM 的 50H 和 51H 中的数据。

（3）使用单步调试的方式来执行不同分支的程序。在调试过程中，配合观察寄存器和存储器器窗口，检验程序的运行结果是否正确。

（4）连续执行程序，配合观察寄存器和存储器的窗口，检验运行结果是否正确。

$$Y=\begin{cases}1, & \text{当}X>0\\ 0, & \text{当}X=0\\ -1, & \text{当}X<0\end{cases},\quad(-128\le X\le 127)$$

图 1.5　符号函数的表达式

【实验提示】:

通常，符号函数仅仅是一个数学问题，但许多实际的单片机应用问题可以使用它作为数学模型。例如：关于产品的分类、包装、质量的鉴定都可以应用符号函数。如：成品打印“1”，半成品打印“0”，废品打印“-1”等等。这里可以使用 JZ、JNB、SJMP 等控制转移类指令，进行合理的搭配组合，从而完成多分支结构的程序设计，但要注意分支结构的执行顺序以及分支结果的保存，不要顺序混乱和结果丢失。另外，还要注意 Keil μVision2 集成开发环境中，负数使用补码来进行表示，调试过程中要特别留心。

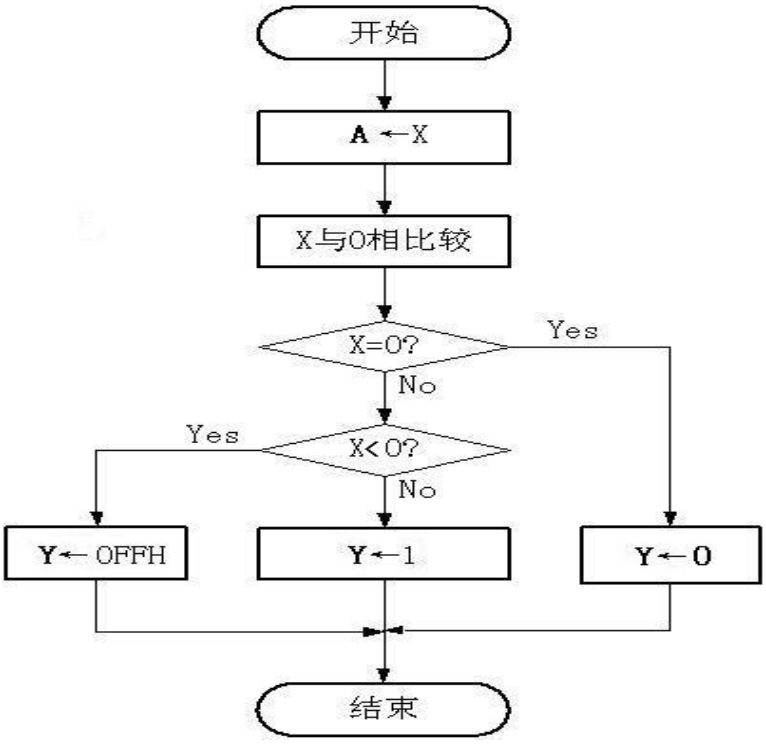


图 1.6　汇编语言程序实现符号函数的软件流程图

3. 选做题

在片内 RAM 的 30H 开始的单元中存放 5 个 2 位十六进制数,编程将它们转换成 ASCII 码,并存放在 40H 开始的单元中。

六. 实验报告

1. 通过本次实验，总结汇编语言的顺序及分支结构程序的设计方法和注意事项。
2. 写出汇编程序的源代码，给每行语句加上详细的注释，并画出程序的流程图。
3. 掌握 Keil 软件如何进行程序的分支以及给存储单元进行赋值操作。
4. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

七、实验和考核内容

1. 每人一组，独立完成。学会使用 Keil μVision2 集成开发环境之汇编语言的编程与调试，进行汇编语言源程序的设计、编译，掌握源程序编译出错提示信息的含义并加以改正，掌握目标代码的单步调试、测试数据的选择，了解运行结果是否反映程序的正确性。
2. 完成实验任务 1（顺序结构的汇编语言程序设计），结果正确，给 60 分（百分制）。
3. 完成实验任务 2（分支结构的汇编语言程序设计），结果正确，加 20 分。
4. 完成实验任务 3（选做题），结果正确，加 20 分。

实验二　　MCS-51 单片机 C 语言编程练习

一. 实验目的

1. 掌握单片机 Keil μVision2 集成开发环境的使用方法。

2. 掌握使用 C51 语言进行顺序、分支、循环结构的程序设计方法。

3. 能够使用 C51 语言，独立设计出具有一定综合性的单片机应用程序，并与汇编语言的相应程序进行比较，加深理解。

4. 对应用问题能抽象出数学模型，绘制软件程序的流程图，并能用 C51 语言实现。

二.预习与思考

1. 预习理论教材中“C51 语言程序设计”的相关内容，掌握 C51 程序设计的基本方法、思路以及设计规范。

2. 预习理论教材中“C51 程序设计”的相关例程。

3. 掌握单片机 C51 语言的调试方法，体会与标准 C 语言的异同。

4. 思考如何使用 C51 语言，对顺序、分支、循环结构的单片机应用程序进行设计，重点体会 C51 语言特有的设计方法与思想。

三. 实验原理

1. 单片机的 C51 语言简介

通常，将一些能够对 MCS—51 系列单片机进行硬件操作的 C 语言统称为 C51 语言。在众多的 C51 语言中，功能最强、最受用户欢迎的是德国 KEIL 公司的 Keil C51 语言。单片机应用系统的程序设计，既可以采用汇编语言，也可以采用 C51 语言，两者各具特色。其中，汇编语言是一种用助记符来代表机器语言的符号语言。因为它最接近机器语言，所以汇编语言对单片机的操作直接、简捷，编写的程序紧凑、执行效率较高。但是，不同种类的单片机其汇编语言存在一定的差异。在一种单片机上开发的应用程序，通常不能直接应用到另一种单片机芯片上，如果进行程序的移植，难度也比较大。与此同时，汇编语言开发的程序可读性较差，不容易理解，特别是当单片机应用系统的规模比较大时，汇编语言的编程工作量非常大，从而影响应用系统的开发效率。

相对而言，C51 语言恰好可以克服汇编语言的一些缺欠。例如，C51 语言可读性好、可移植性高，与自然语言比较接近，并且相同功能的程序使用 C51 语句的数量要远小于汇编语句。通常，C51 语言的入门学习相对于汇编语言更容易，而且在 C51 语言的程序中还可以嵌套汇编语言，从而满足对执行效率或操作的一些特殊要求。因此在单片机应用系统的开发过程中，C51 语言逐渐成为了主要的编程语言。

2. 单片机 C51 语言的数据类型

C51 语言的数据类型，既有与 ANSI C 语言通用的数据类型，也有自己所特有的数据类型。C51 语言的具体数据类型见表 5-1 所示。从表中可以看出 C51 语言增加了 bit、sfr、sfr16、sbit 四种新的数据类型，分别用于定义 2 进制变量、特殊功能寄存器变量、16 位特殊功能寄存器变量以及特殊功能位。另外，C51 语言还有自己特有的变量存储类型以及存储模式，这里就不一一详述，具体可以参考理论教材的相关内容。

表 2-1　C51 语言所支持的数据类型

数据类型	名　　称	长　　度	值　　域
unsigned char	无符号字符型	单字节	0～255
signed char	有符号字符型	单字节	－128～＋127
unsigned int	无符号整型	双字节	0～65 535
signed int	有符号整型	双字节	－32 768～＋32 767
unsigned long	无符号长整型	四字节	0～4 294 967 295
signed long	有符号长整型	四字节	－2 147 483 648～＋2 147 483 647
float	浮点型	四字节	±1.175494E－38～±3.402823E＋38
bit	位标量	位	0 或 1
sfr	特殊功能寄存器	单字节	0～255
sfr 16	16 位特殊功能寄存器	双字节	0～65 535
sbit	特殊功能位	位	0 或 1

3. C51 语言对单片机的 SFR 以及存储器的访问

C51 语言除了具有特殊的数据类型、存储类型以及存储模式外，C51 语言还可以对 SFR 以及片内或片外的存储器单元进行直接访问。这里的 SFR 是指 MCS-51 单片机的特殊功能寄存器。8051 单片机的 SFR 一共有 21 个，具体见表 5-2 所示。从表中可以看到，这么多的特殊功能寄存器，如果每一个都用数据类型 sfr 定义一遍再使用，就显得比较麻烦。那么如何不用定义，还能在 C51 语言中随时使用这 21 个特殊功能寄存器呢？方法也很简单，只要在每个 C51 程序中包含头文件<reg51.h>或者<reg52.h>或者<AT89X51.h>或者<AT89X52.h>中的任意一个，就可以在 C51 程序中任意使用这 21 个特殊功能寄存器。因为在这些头文件中已经将相应的特殊功能寄存器用数据类型 sfr 定义好了，所以当 C51 程序包含了上述的头文件以后就可以直接用 SFR 了，而不必再重新一个一个的来定义这些寄存器了。例如，给特殊功能寄存器 P3 口赋值为 0xff，程序可以这样编写：

```
//-------------------------------------------------------------------
    #include  <reg51.h>    //包含 21 个特殊功能寄存器地址定义的头文件
    void  main( )
    {
       P3=0xff;            //给 P3 口赋值为 0xff
    }
//-------------------------------------------------------------------
```

另外，C51 语言还可以直接访问片内或片外的存储单元，这时需要在程序中包含绝对地址访问头文件<absacc.h>。这个头文件使得 C51 语言对存储器单元的访问变得更加简便，在<absacc.h>头文件中提供了一些对存储单元进行访问的宏定义，具体如下：

（1）CBYTE[data]：该宏定义代表对单片机的片内 ROM 单元进行访问。

（2）DBYTE[data]：该宏定义代表对单片机的片内 RAM 单元进行读写操作。

（3）XBYTE[data]：该宏定义代表对单片机的片外 RAM 单元进行读写操作。

在这里举一个例子：例如要从单片机片内 ROM 的 30H 单元中读出数据，给片内 RAM 的 50H 单元，然后再从片内 RAM 的 50H 单元将该数据写入到片外 RAM 的 ABCDH 单元中。相应的 C51 语言程序可以这样编写：

```
//-------------------------------------------------------------------
    #include      <absacc.h>      //包含绝对地址访问头文件
    #define   a   CBYTE[0x30]     //定义片内 ROM 的 30H 单元
    #define   b   DBYTE[0x50]     //定义片内 RAM 的 50H 单元
    #define   c   XBYTE[0xABCD]   //定义片外 RAM 的 ABCDH 单元
    void  main( )
    {  b = a;   c = b;  }         //各个存储单元间相互赋值
//-------------------------------------------------------------------
```

表 2-2　8051 单片机的特殊功能寄存器

符　号	地　址	注　释	符　号	地　址	注　释
P0	0x80	并口 P0	IP	0xD8	中断优先控制寄存器
P1	0x90	并口 P1	PCON	0x87	波特率选择寄存器
P2	0xA0	并口 P2	SCON	0x98	串行口控制器
P3	0xB0	并口 P3	SBUF	0x99	串行数据缓冲器
PSW	0xD0	程序状态字	TCON	0x88	定时器控制寄存器
ACC	0xE0	累加器	TMOD	0x89	定时器方式选择寄存器
B	0xF0	乘除法寄存器	TL0	0x8A	定时器 0 低 8 位
SP	0x81	堆栈指针	TL1	0x8B	定时器 0 高 8 位
DPL	0x82	数据指针低 8 位	TH0	0x8C	定时器 1 低 8 位
DPH	0x83	数据指针高 8 位	TH1	0x8D	定时器 1 高 8 位
IE	0xA8	中断允许控制寄存器			

4. C51 语言对特殊功能位的访问

C51 语言除了可以对特殊功能寄存器进行整体访问以外，还可以对特殊功能寄存器的各个二进制位进行访问。通常把特殊功能寄存器中的各个二进制位，称为特殊功能位。C51 语言若想对这些特殊功能位进行操作，事先要完成两步工作：第一步，先将特殊功能位所在的特殊功能寄存器进行定义或者在程序中直接包括已经定义了特殊功能寄存器的头文件，例如 reg51.h，建议直接包含头文件；第二步，使用 C51 语言特有的数据类型 sbit 定义特殊功能位，然后用户的 C51 程序就可以对这些特殊功能位进行操作了。在这里举一个例子：例如要对单片机特殊功能寄存器 P1 口的两个特殊功能位进行赋值操作，使 P1.1=0，而 P1.6=1。相应的 C51 语言程序可以这样编写：

```
//-------------------------------------------------------------------
    #include  <reg51.h>    //包含 21 个特殊功能寄存器地址定义的头文件
    sbit    a = P1^1;      //定义 P1 口的特殊功能位 P1.1 为 a
    sbit    b = P1^6;      //定义 P1 口的特殊功能位 P1.6 为 b
```

```
    void  main( )
    {
       a = 0;              //给特殊功能位 P1.1 赋值为 0
       b = 1;              //给特殊功能位 P1.6 赋值为 1
    }
//-------------------------------------------------------------
```

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。
2. Keil μVision2 集成开发环境。

五. 实验内容

1. 顺序结构的 C51 语言程序设计

给特殊功能寄存器和特殊功能位赋值。请使用 C51 语言，给单片机的 P2 口赋值为 0x00，并使 P3 口的特殊功能位 P3.2=0，P3.3=0，P3.6=0，要求在 Keil 软件中仿真 P2 口和 P3 口的功能，并看到实际的调试结果。具体调试要求：

（1）在 Keil μVision2 环境中，掌握查看各并行口数据的方法。

（2）在 Keil μVision2 环境中，如何对并行口的各个二进制位进行赋值和观察。

（3）使用单步调试的方式来执行程序。在调试过程中，配合观察并口寄存器的窗口，检验程序的运行结果是否正确。

（4）连续执行程序，在执行的过程中，配合观察并口寄存器窗口，检验程序运行结果。

【实验提示】:

此题主要使用 C51 语言，对特殊功能寄存器以及特殊功能位进行赋值。如何在 Keil 软件中仿真调试 4 个并行 I/O 口，是此题要掌握的一个重要知识点，它将会对以后单片机硬件部分的软件仿真打下基础。此题的参考程序代码如下：

```
//-----------------------------T21.C 程序-----------------------------
//文件名称：T21.c
//程序功能：给特殊功能寄存器以及特殊功能位赋值。
//编制时间：2010 年 1 月
//--------------------------------------------------------------------
  #include  <reg51.h>       //包含定义 51 单片机寄存器的头文件
  sbit    a=P3^2;           //定义 P3 口引脚 P3.2
  sbit    b=P3^3;           //定义 P3 口引脚 P3.3
  sbit    c=P3^6;           //定义 P3 口引脚 P3.6
  void    main( )           //主函数
  {
    P2=0x00;                //给 P2 口赋值为 00
    a=0;  b=0;  c=0;        //P3.2、P3.3、P3.6 引脚赋值为 0
  }
//------------------------- T21.C 程序结束-------------------------
```

在这里，对如何使用 Keil 软件仿真调试 P0～P3 四个并行 I/O 口进行简要的介绍。当在 Keil 环境中把程序编写完毕以后，并编译为 0 个错误 0 个警告时，点击“调试菜单下的‘开始/停止调试’选项”，这时进入到调试环境。若想观察 P0～P3 这四个并行口的软件仿真结果，首先需要将这四个并行口打开，可以点击“外围设备菜单下的‘I/O-Ports’选项”，该选项会指示出 Port 0～Port 3 也就是单片机的 P0 口～P3 口，具体如图 5.1 所示。由于本题只使用 P2 和 P3 口，因此用左键点击 Port 2 和 Port 3 就会显示 P2 和 P3 口的当前状态，如图 5.2 所示。从图中可以看出，P2 和 P3 口的初始默认值都为 0xff。这里有个小技巧，每个并行口都有 8 个特殊功能位，例如 P2 口的 8 位是 P2.0～P2.7，它们在图中用小方格依次从右向左排列，其中最右侧的小方格代表 P2.0，最左侧的小方格代表 P2.7，8 个二进制位的值就是整个 P2 口的值，而每个二进制位的值可以通过它们各自小方格的状态看到。当小方格的下面画的是“对号”时，代表相应的二进制位为 1，由于 4 个并行口初始状态时，8 个小方格都是“对号”，所以每个并口的初始值都是 0xff；当相应二进制位下的小方格状态为“空白”时，那么这个小方格对应的二进制位就是 0，如果 8 个小方格的状态都为空白，则整个并口的值为 00。

当从图 5.2 中看到 P2 和 P3 口的初始状态都为 0xff 时，接下来可以按快捷键 F10 来单步执行程序，边按 F10 快捷键时，要边看 P2 和 P3 口的状态是否按照程序的设计要求进行了相应的改变。一直按 F10 键，直到程序单步执行完毕，这时执行的结果如图 5.3 所示。从图中可以看出，P2 口的 8 个小方格都是“空白”状态，因此 P2 口的值为 0x00,而从下面的 P3 口也可以看到 P3.2、P3.3 以及 P3.6 的小方格状态都是空白的，所以它们的值是 0，其他各个二进制位的状态是 1。综上所述，从图 5.3 中可以看出此时 P2=0x00，P3.2=0，P3.3=0，P3.6=0，满足题目的要求。至此，已经将 Keil 软件下如何仿真四个并行口的方法介绍完毕，希望大家多练习，全面掌握。

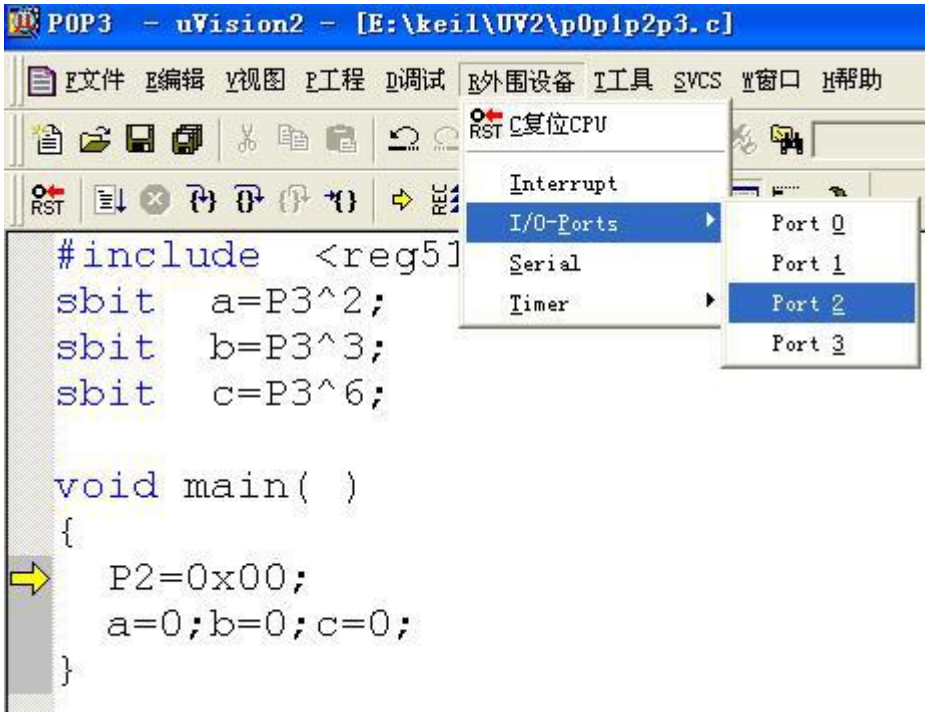


图 2.1　并行口 P0～P3 在 Keil 软件中的打开方式

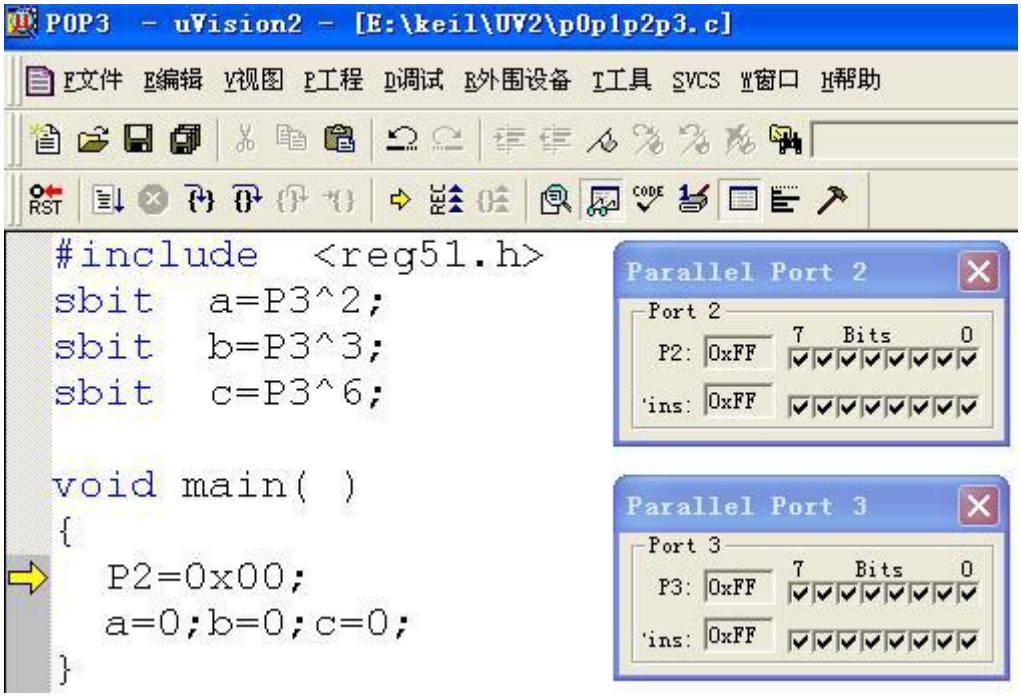


图 2.2　并行口 P2 和 P3 的初始默认值

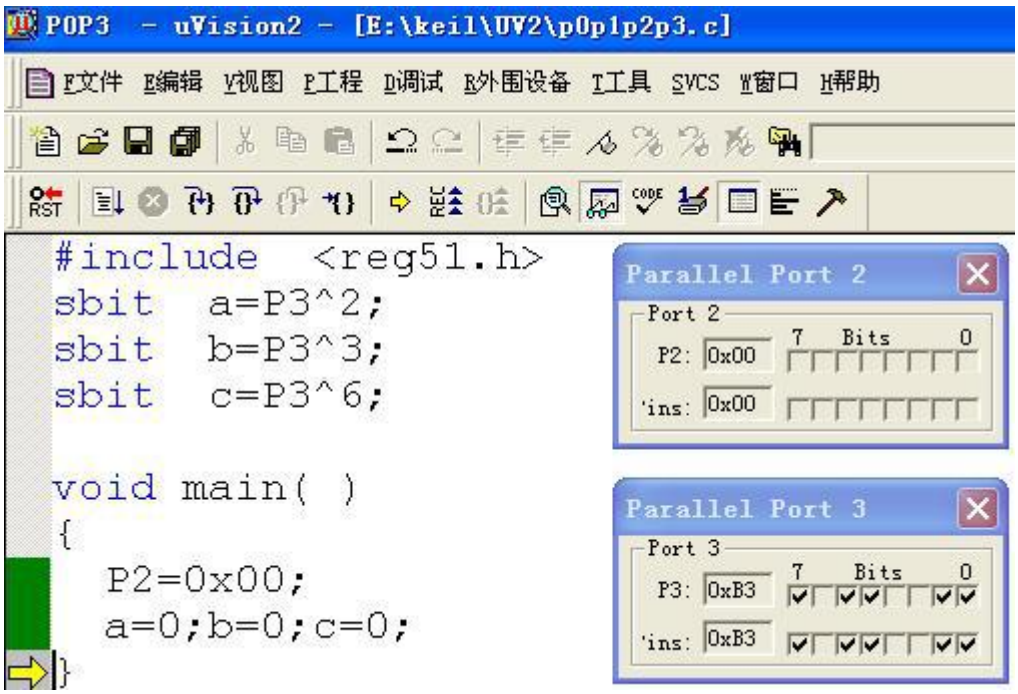


图 2.3　并行口 P2 和 P3 的执行结果图

2. 分支结构的 C51 语言程序设计

将 ASCII 码转换成 16 进制数。已知，一个 16 进制数的 ASCII 码，存放在片内 RAM 的 30H 单元，请把这个数的 16 进制表示方式存于片内 RAM 的 31H 单元中，软件流程如图 5.4 所示。具体程序的调试要求如下：

（1）在 Keil μVision2 环境中，掌握查看各并行口数据的方法。

（2）在 Keil μVision2 环境中，如何对并行口的各个二进制位进行赋值和观察。

（3）使用单步调试的方式来执行程序。在调试过程中，配合观察并口寄存器的窗口，检验程序的运行结果是否正确。

（4）连续执行程序，在执行的过程中，配合观察并口寄存器窗口，检验程序运行结果。

【实验提示】:

本题主要是对 ASCII 码与 16 进制数之间进行转换，同时加强 C51 语言的分支结构程序设计。对于 16 进制数中 0～9 的 ASCII 码值比 16 进制数本身大 30H，而 16 进制数中 A～F 的 ASCII 码值比 16 进制数本身大 37H，只要掌握好这些对应关系，使用 if-else 语句这道题目就很容易完成。

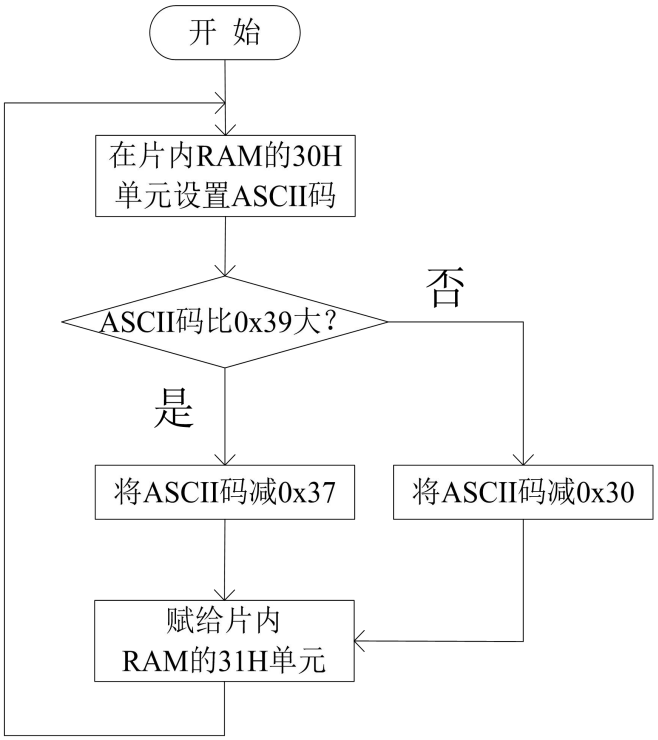


图 2.4　ASCII 码与 16 进制数转换的软件流程图

3. 循环结构的 C51 语言程序设计

片内及片外 RAM 中的块数据传送。请使用 C51 语言，将片内 RAM 的 20H～29H 单元中的数据，传送给片外 RAM 的 ABC0H～ABC9H 单元，软件流程如图 2.5 所示。具体程序的调试要求如下：

（1）在 Keil μVision2 环境中，打开片内 RAM 和片外 RAM 的存储器窗口。

（2）在 Keil μVision2 环境中，如何给片内 RAM 的 20H～29H 单元同时赋值。

（3）使用单步调试的方式执行程序，配合观察存储器窗口，检验程序运行结果是否正确。

（4）连续执行程序，在执行的过程中，配合观察存储器窗口，检验程序运行结果。

【实验提示】:

此题主要练习对片内和片外 RAM 单元的理解，以及如何查询、修改片内外 RAM 单元中所存储的数据，同时用于加强 C51 语言的循环结构程序设计。这里需要注意循环变量的

数据类型，当控制循环次数的变量类型为无符号字符型即 unsigned　char 类型时，它的大小范围在 0～255 之间，因此这时不要在 for 循环中使循环变量的值小于等于 256，否则会产生无限循环，导致程序无法正常的运行。

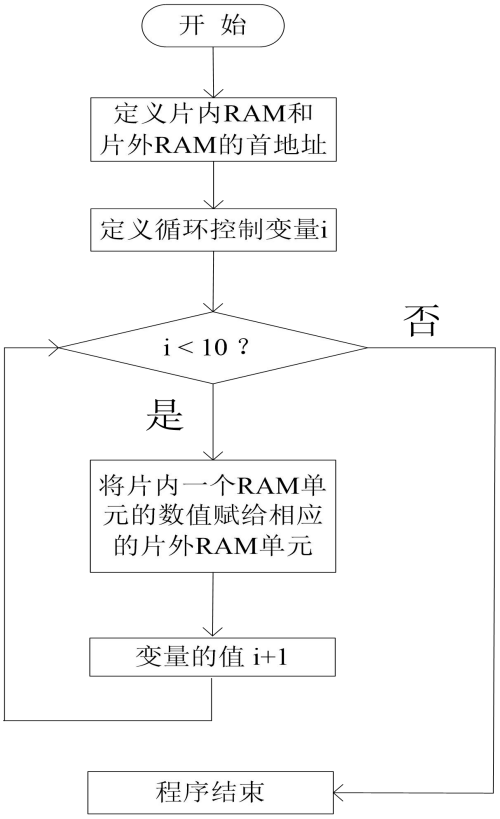


图 2.5　片内和片外存储单元中块数据的传送流程图

4. 选做题

（1）存储单元数据的相互传送。请使用 C51 语言，将单片机片内 RAM 的 32H 单元的数据与片外 RAM 的 6DH 单元的数据相加，结果保存在片外 RAM 的 50H 单元，然后再将片外 ROM 的 1234H 单元的数据与片内 RAM 的 66H 单元的数据相减,结果保存在片内 RAM 的 36H 单元中。

（2）有符号数比较大小。假设在片内 RAM 的 3AH 单元中，存储的有符号数据是 m，在片外 RAM 的 9ABDH 单元中存储的有符号数据是 n，试比较这两个数的大小，并将大数保存在片内 RAM 的 6FH 单元中。

（3）对片外 RAM 单元进行数据传输。请先将片外 RAM 的 8000H～80FFH 单元中的数据清零，然后再把数值 00H～FFH，赋给上面的 256 个外存单元。

六. 实验报告

1. 通过本次实验，总结 C51 语言不同结构的程序设计方法与调试步骤。
2. 写出 C51 程序的源代码，给每行语句加上详细的注释，并画出程序的流程图。
3. 掌握如何在 Keil 软件中观察程序、变量以及存储单元的数据变化情况。
4. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

七、实验和考核内容

1. 每人一组，独立完成。学会使用 Keil μVision2 集成开发环境之 C51 语言的编程与调试，进行汇编语言源程序的设计、编译，掌握源程序编译出错提示信息的含义并加以改正，掌握目标代码的单步调试、测试数据的选择，了解运行结果是否反映程序的正确性。

2. 完成实验任务 1（顺序结构的 C51 语言程序设计）和完成实验任务 2（分支结构的 C51 语言程序设计），结果正确，给 60 分（百分制）。

3. 完成实验任务 3（循环结构的 C51 语言程序设计），结果正确，加 20 分。

4. 完成实验任务 4（选做题），结果正确，加 20 分。

实验三　　流水灯实验

一. 实验目的

1. 熟悉单片机 P0～P3 的结构、特性及其使用方法。

2. 了解单片机系统主机板和系统键盘/显示板的电路原理，了解主机板和键盘/显示板上各器件的逻辑结构和特性。

3. 理解主机板和键盘/显示板各硬件接口的功能和特点，掌握主机板与键盘/示板之间的连接方法，并能够针对硬件的连接使用汇编语言和 C51 语言进行编程和模拟调试，实现单片机流水灯实验。

4. 掌握 STC-ISP 下载编程软件的使用方法，将模拟调试成功的*.HEX 程序在线下载到单片机的片内 ROM 中，实际观察程序运行的直观效果。

二. 预习与思考

1. 预习实验讲义附录中提供的主机板和键盘/显示板的电路原理以及电路板图，了解它们的结构。

2. 预习理论教材中“汇编语言程序设计”和“C51 程序设计”的相关例程，并结合硬件电路进行应用分析。

3. 思考如何使用汇编语言和 C51 语言对单片机的硬件进行应用程序的设计。

三. 实验原理

1. 单片机汇编语言和 C51 语言应用程序的设计步骤

在具体的单片机应用系统程序开发过程中，通常需要以下几个步骤：分析问题→设计算法→画出程序流程图→编写程序→调试验证→应用与维护，如图 3.1 所示。

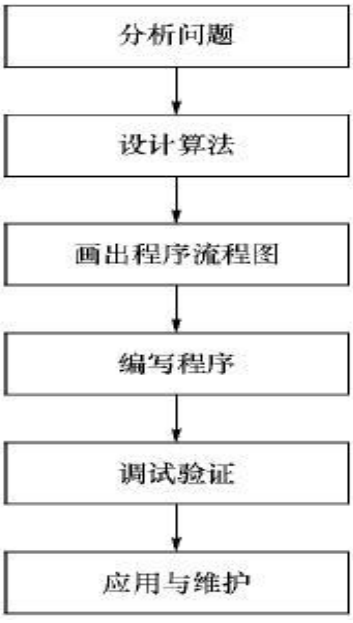


图 3.1　C51 应用程序开发的一般步骤

2. C51 语言对单片机硬件的控制

ANSI C 语言对硬件进行控制比较困难，而 C51 语言对单片机的硬件控制相对容易，且效率较高。这是因为在 C51 语言的系统程序中，提供了很多对单片机硬件的定义，这样就很方便的把单片机内部看不到的抽象硬件，用具体的软件符号表示出来，从而通过对软件符号的操作来实现硬件的工作。例如，在<reg51.h>头文件中，就把 21 个特殊功能寄存器以及它们的特殊功能位都定义出来了，这样无论是对特殊功能寄存器还是特殊功能位的操作，都相当于进行软件编程；再例如，在<absacc.h>头文件中，可以将单片机的各类存储器单元使用相应的软件关键字定义出来后，对存储单元的操作就相当于对软件中的几个关键字进行操作，从而方便了 C51 语言对单片机硬件的控制。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. 主机板一块，键盘/显示板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

3. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

五. 实验内容

请使用 89C52 单片机设计一个最小系统，以单片机的 P1 口作为输出口，外接 8 个 LED 小灯，编写汇编语言或 C51 程序，使小灯循环点亮，使用实验室提供的电路板实现。具体程序的调试要求如下：

（1）在 Keil μVision2 环境中，模拟运行编写的程序，观察 P1.0～P1.7 引脚的变化情况，判断小灯亮灭。

（2）调试完成后，用 STC-ISP 下载编程软件将生成的*。HEX 文件在线下载到单片机中。

（3）*.HEX 文件在线下载到单片机后，按复位键执行程序，检验程序运行结果。

【实验提示】:

此题主要通过用汇编语言或 C51 语言编写的程序在硬件上实现,使 LED 小灯循环点亮.需要注意的问题：

1. 主机板上的晶振与下载编程软件一致。
2. 键盘/显示板上有 2 个 LED 小灯接反，如果要实现 LED 小灯循环点亮，必须在软件方面进行调整。
3. 要考虑延时问题。

六. 需要注意的问题

两块电路板之间的电源连线有正负极，必须正确连接，否则会烧毁电路板。

七. 实验报告

1. 通过本次实验，总结汇编语言或 C51 软件程序控制单片机硬件的设计方法与调试步骤。

2. 写出汇编语言或 C51 程序的源代码，加上必要的注释，并画出程序的流程图。

3. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

实验四　　外中断实验

一. 实验目的

1. 熟悉单片机 P0～P3 的结构、特性及其使用方法。

2. 了解主机板和键盘/显示板的电路原理及其各器件的逻辑结构和特性。

3. 理解主机板和键盘/显示板各硬件接口的功能和特点，掌握主机板与键盘/显示板之间的连接方法，特别是键盘的设置和使用方法，并能够针对硬件的连接使用汇编语言和 C51 语言进行编程和模拟调试，实现单片机外中断实验。

4. 掌握 STC-ISP 下载编程软件的使用方法，将模拟调试成功的*.HEX 程序在线下载到单片机的片内 ROM 中，实际观察程序运行的直观效果。

二.预习与思考

1. 预习实验讲义附录中提供的主机板和键盘/显示板的电路原理图以及电路板图，了解它们的结构。

2. 预习理论教材中“单片机中断系统”关于外中断的相关内容，并结合硬件电路进行分析。

3. 思考如何将主机板与键盘/显示板进行连接，形成一个单片机的硬件系统，并进行相应外中断的应用程序的设计。

三. 实验原理

1. 参照附录自行将主机板和键盘/显示板进行连接，设计形成一个单片机的硬件系统。

2. 参照自行设计的硬件系统，有针对性地进行相应外中断的应用程序的设计。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. 主机板一块，键盘/显示板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

3. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

五. 实验内容

1. 请使用主机板和键盘/显示板设计一个硬件系统，以按键 S9 作为外中断请求信号，以数码管作为十进制计数显示（最右侧数码管为最低位，依次向左累计），单片机的 P0 口作为数据信号输出口（段选择），P1 口作为控制信号输出口（位选择）。每按键一次计数显示加 1，最大计数 20，然后在从 0 进行计数显示。

2. 模拟调试完成后，用 STC-ISP 下载编程软件将生成的*。HEX 文件在线下载到单片机中。

3. *.HEX 文件在线下载到单片机后，按复位键执行程序，检验程序运行结果。

【实验提示】:

此题可以使用汇编语言或 C51 语言编写的程序在硬件上实现.需要注意的问题：

（1） 主机板上的晶振与下载编程软件一致。

（2） 必须了解电路板的硬件及原理，了解连接后的硬件结构，针对硬件进行编程。

（3） 参照附录提供的说明，了解段选择和位选择信号，找出合理的数据。

六. 需要注意的问题

两块电路板之间的电源连线有正负极，必须正确连接，否则会烧毁电路板。

七. 实验报告

1. 写出汇编语言或 C51 程序的源代码，加上必要的注释，并画出程序的流程图。
2. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

实验五　　定时器实验一

一. 实验目的

1. 熟悉单片机 P0～P3 的结构、特性及其使用方法。

2. 了解主机板和键盘/显示板的电路原理，了解主机板和键盘/显示板上各器件的逻辑结构和特性。

3. 理解主机板和键盘/显示板各硬件接口的功能和特点，掌握主机板与键盘/显示板之间的连接方法，特别是数码管在键盘/显示板上的电路设置，并能够针对硬件的连接使用汇编语言和 C51 语言进行编程和模拟调试，实现单片机定时器实验。

4. 掌握 STC-ISP 下载编程软件的使用方法，将模拟调试成功的*.HEX 程序在线下载到单片机的片内 ROM 中，实际观察程序运行的直观效果。

二. 预习与思考

1. 预习实验讲义附录中提供的主机板和键盘/显示板的电路原理图以及电路板图，了解它们的结构。

2. 预习理论教材中“单片机定时器/计数器”的相关内容，了解定时器/计数器的 4 种工作方式，并结合硬件电路进行分析。

3. 思考如何将主机板与键盘/显示板进行连接，形成一个单片机的硬件系统，并进行相应定时器/计数器的应用程序的设计。

三. 实验原理

1. 参照附录自行将主机板与键盘/显示板进行连接，设计形成一个单片机的硬件系统。

2. 参照自行设计的硬件系统，有针对性地进行相应定时器/计数器应用程序的设计。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. 主机板一块，键盘/显示板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

3. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

五. 实验内容

1. 请使用主机板和键盘/显示板设计一个硬件系统，用数码管作为时钟显示，最右侧两个数码管显示“小时”，相邻两个数码管显示“分钟”，再相邻两个数码管显示“秒”，最后两个数码管显示“百分之一秒”。

2. 可自行设计进行连接，形成一个单片机硬件系统

3. 模拟调试完成后，用 STC-ISP 下载编程软件将生成的*。HEX 文件在线下载到单片机中。

4. *.HEX 文件在线下载到单片机后，按复位键执行程序，检验程序运行结果。

【实验提示】:

此题可以使用汇编语言或 C51 语言编写的程序在硬件上实现.需要注意的问题：

（1）主机板上的晶振与下载编程软件一致。

（2）必须了解电路板的硬件及原理，了解连接后的硬件结构，针对硬件进行编程。

（3）参照附录提供的说明，了解“段选择”和“位选择”信号，找出合理的数据。

（4）中断服务程序尽可能短小，执行时间短，以减少不必要的麻烦。

六. 需要注意的问题

两块电路板之间的电源连线有正负极，必须正确连接，否则会烧毁电路板。

七. 实验报告

1. 写出汇编语言或 C51 程序的源代码，加上必要的注释，并画出程序的流程图。
2. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

实验六　　定时器实验二

一. 实验目的

为使学生继续加深对定时器知识的理解和掌握，在定时器实验一的基础上， 把实验内容进一步扩充，以便为后续的实验打下良好的理论与实践基础。

二. 预习与思考

1. 复习实验五，对实验五所使用的电路加深理解。
2. 在此电路的基础上进一步改进或使用原硬件。
3. 进行相应定时器/计数器的应用程序的设计。

三. 实验原理

1. 参照附录自行将主机板与键盘/显示板进行连接，设计形成一个单片机的硬件系统。
2. 参照自行设计的硬件系统,有针对性地进行进一步相应定时器/计数器应用程序的设计。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。
2. 主机板一块，统键盘/显示板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。
3. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

五. 实验内容

1. 在实验五的实验内容基础上，进一步进行功能开发，能够用数码管显示“年、月、日”等信息（注意闰年的计算），最右侧两个数码管显示“年”，中间隔一个数码管（不亮），再相邻两个数码管显示“月”，再隔一个数码管（不亮），最后两个数码管显示“日期”。
2. 也可自行设计进行连接，也可使用实验五的单片机硬件系统
3. 模拟调试完成后，用 STC-ISP 下载编程软件将生成的*。HEX 文件在线下载到单片机中。
4. *.HEX 文件在线下载到单片机后，按复位键执行程序，检验程序运行结果。

六. 需要注意的问题

两块电路板之间的电源连线有正负极，必须正确连接，否则会烧毁电路板。

七. 实验报告

1. 写出汇编语言或 C51 程序的源代码，加上必要的注释，并画出程序的流程图。
2. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

实验七　　A/D 转换实验

一. 实验目的

1. 了解 A/D 转换的基本概念和 A/D 转换的用途；
2. 理解 A/D 转换芯片 ADC0809 的内部结构和工作原理；
3. 掌握使用 A/D 转换器实现单片机模拟量输入的软硬件设计方法。
4. 熟悉和掌握 A/D 和 D/A 转换板的结构。

二. 预习与思考

1. 预习理论教材中“A/D 转换”的相关内容；
2. 什么是 A/D 转换？它的用途？A/D 转换的主要技术指标？
3. ADC0809 如何与单片机进行连接？如何编程实现启动 A/D 转换和模拟量输入通道的切换？如何得到 A/D 转换的结果？
4. 预习实验讲义附录中提供的 A/D 和 D/A 转换板的电路原理图以及电路板图，了解它们的结构。

三. 实验原理

1. A/D 转换的基础知识

在实际测控系统中，经常需要对电压、电流、温度、压力、流量等连续变化的物理量等参数进行不间断地监测与控制。要使用单片机来实现对这些变量的监控，首先要解决这些变量和单片机之间的输入与输出问题，使用 A/D 转换和 D/A 转换来解决这些变量和单片机之间的输入与输出问题是人们常用的解决方法。

A/D 转换用于将模拟量转换为数字量。模拟量主要是指类似温度、压力、流量、速度、电流、电压等变量，这些变量的数值大小是随时间连续变化的；而数字量是指数值大小只有 0、1 两种情况，并且是随时间离散变化的量。如图 7.1 所示，左图是以电压为例的模拟量的数值变化图，右图是数字量的变化图。

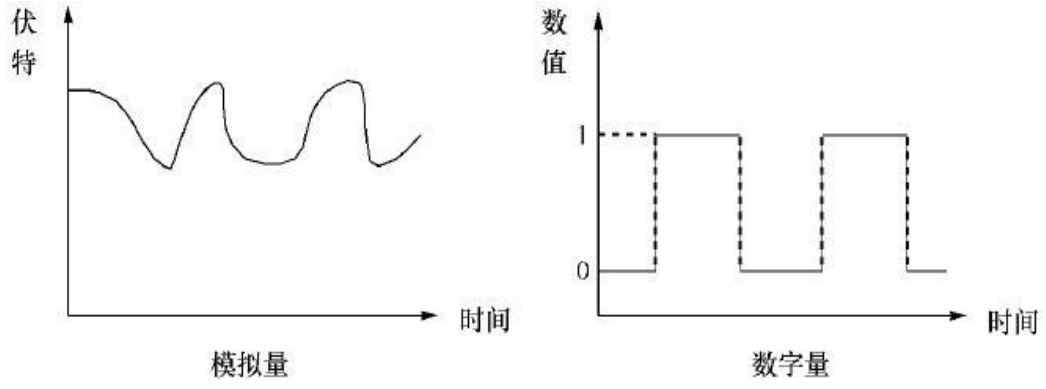


图 7.1　模拟量和数字量变化示意图

从图中可以看出，模拟量的大小随时间连续变化，而数字量是不随时间连续变化的，即它是离散的量。通常，计算机内部只能处理数字量，而不能直接处理模拟量。因此，如果当外围设备向单片机模拟量，比如电压时，必须先要进行 A/D 转换，将电压模拟量转换成由 0 或 1 组成的数字量，然后再输入到计算机机进行相应的处理。

了解 A/D 转换的基本概念后，接下来要介绍 A/D 转换的基本方法。A/D 转换有很多方法，例如：积分型 A/D 转换、逐次逼近型 A/D 转换、并行比较型/串并行比较型 A/D 转换、电容阵列逐次比较型 A/D 转换、Σ-△调制型 A/D 转换以及 V-F 型 A/D 转换等。这里以常用的积分型 A/D 转换和逐次逼近型 A/D 转换进行简要介绍。

（1）积分型 A/D 转换：是将输入的电压转换成时间或频率，然后由定时器/计数器获得数字值。积分型 A/D 转换的工作原理类似于古代的沙漏计时或燃香计时，沙漏里的沙子就好比是模拟量，而估计出来的时间就好比是数字量。这种 A/D 转换的方法，其优点是精度高、抗扰能力强，缺点是速度较低。

（2）逐次逼近型 A/D 转换：逐次逼近型 A/D 转换也是实际使用较多的方法，由一个比较器和 D/A 转换器通过逐次比较逻辑构成。当逐次增加内部的 D/A 输入值时，将其输出的电压与 A/D 转换要测量的电压进行比较。两者相等时，内部 D/A 的输入值就是 A/D 转换的结果。逐次逼近型 A/D 转换的工作原理类似于天平称重物。当增加砝码天平平衡时，砝码的重量就是被称重物的质量。在这里，重物就好比是模拟量，而砝码就好比是经过 A/D 转换后的数字量。逐次逼近型 A/D 转换的优点是速度快，缺点是抗干扰能力差。

虽然，A/D 转换的方法比较多，但是衡量一次 A/D 转换是否成功并可靠，主要有如下一些指标：转换速率、转换精度、分辨率、线性度、偏移误差、量化误差等。下面对前 3 个主要转换指标进行解释即转换速率、转换精度以及分辨率。

（1）转换速率（Conversion Rate）：是指完成一次 A/D 转换所需要时间的倒数，它是一项非常重要的指标。积分型 A/D 转换的时间是毫秒级，属于低速 A/D 转换；逐次比较型 A/D 转换的时间是微秒级，属于中速 A/D 转换；并行比较型/串并行比较型的 A/D 转换时间是纳秒级，属于高速 A/D 转换。选择何种速率的 A/D 转换器，要根据实际需要以及性价比等因素。

（2）转换精度（Conversion Accuracy）：定义为一个实际的 A/D 转换器与一个理想的 A/D 转换器在量化值上的差异。转换精度由模拟误差和数字误差组成。前者属于非固定误差，由器件质量决定；后者与 A/D 转换输出数字量的位数有关，位数越多，误差越小。

（3）分辨率(Resolution)：指模拟量变化一个最小的单位时，数字量变化的大小。通常，以 A/D 转换器的位数来表示。

一个模拟量转换成数字量需要使用 A/D 转换器(Analog to Digital Converter，ADC)。在单片机的实际应用中，可以选择两种类型的 ADC，即并行 ADC 和串行 ADC。所谓并行 ADC 是指 A/D 转换器的内部有多个比较器，可以同时将外围设备输入的模拟量数据一次转换成多位数字量输出给单片机进行处理；而串行 ADC 是指外围设备输入的模拟量数据一次转换成 1 位数字量输出给单片机来进行处理。从 A/D 转换的效率来看，并行 ADC 要优于串行 ADC，但从价格和应用方便的角度看串行 ADC 要好于并行 ADC。

2. 并行 A/D 转换器芯片 ADC0809

在实际的模拟量数据采集系统中，需要使用单片机来控制 ADC，从而完成对外围设备中运行的模拟量数据进行采集。单片机控制 ADC 进行模拟量采集的原理如图 7.2 所示。从图中看到，外围设备中运行的模拟量经过传感器处理以后，被传送到了 ADC，再经过 A/D 转换以后就会得到数字量。最后，将数字量输入给单片机来进行各种数据分析处理。下面将具体介绍 ADC0809 芯片。

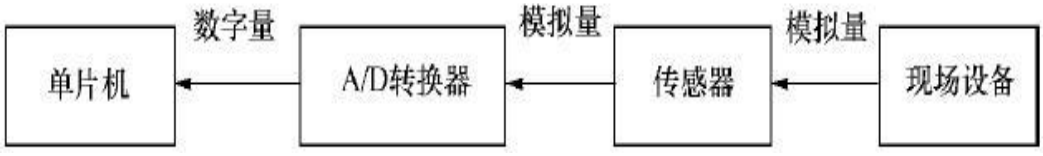


图 7.2　单片机控制 ADC 采集模拟量数据的原理图

ADC0809 芯片采用逐次逼近型转换原理设计的 8 位并行 A/D 转换器芯片。它可以将 0～+5V 的模拟量输入到 ADC 中进行转换，然后将得到的 8 位数字量转换结果，输出给单片机进

行处理。ADC0809 芯片具有 8 个模拟量的输入通道，可同时进行多路模拟信号的采集，ADC0809 芯片的外部管脚，如图 7.3 所示，各管脚功能如下：

① IN0～IN7：8 路模拟量数据输入通道管脚，每个管脚都可以输入 0～+5V 的电压模拟量。

② C、B、A：8 路模拟量输入通道的选择管脚，通常连接单片机的三根地址线。当 CBA 的值分别为 000～111 时，则对应选择模拟量的输入通道是 IN0～IN7。

③ ALE：地址输入的锁存信号管脚，高电平有效。当此管脚有效时，C、B、A 这 3 个管脚上的地址信息被输入到 ADC0809 芯片中，然后再进行相应的模拟量输入通道选择。

④ D0～D7：8 位数字量输出管脚。当 A/D 转换结束后，得到的 8 位数字量转换结果将从这 8 个管脚输出给单片机进行处理。

⑤ OE：输出使能管脚，高电平有效。当此管脚是高电平时，将转换后得到的数字量输出至单片机的数据总线上。

⑥ START：A/D 转换开始的管脚，正脉冲有效。当正脉冲持续时间大于 100ns 时，在正脉冲的上升沿对 ADC0809 的内部寄存器进行清零，而在正脉冲的下降沿开始 A/D 转换。

⑦ EOC：A/D 转换结束管脚，高电平有效。当此管脚有效时，代表本次 A/D 转换已经结束。

⑧ CLK：时钟输入管脚。输入频率的范围是 10～1200KHZ，典型值为 640KHZ。

⑨ V_{CC}、GND：芯片工作的电源和地管脚。通常，V_{CC} 接+5V 直流电源，GND 接地。

⑩ V_{REF}（+）、V_{REF}（-）：参考电压输入管脚。

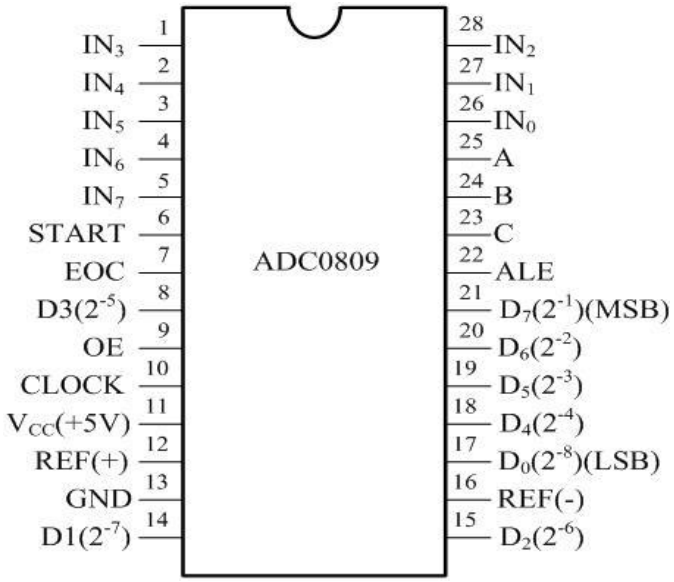


图 7.3　ADC0809 芯片的管脚图

3. 单片机控制 ADC0809 芯片的硬件设计

在单片机控制 ADC0809 芯片进行 A/D 转换设计的过程中，要注意二个问题：（1）如何确定 ADC0809 芯片在硬件电路中的地址，从而启动硬件进行 A/D 转换以及输出 A/D 转换的结果；（2）当 A/D 转换结束后，单片机如何读取 A/D 转换的结果。下面将具体介绍：

（1）ADC0809 芯片硬件地址的确定：主要是通过对单片机输入给 ADC0809 芯片控制管脚 OE、ALE、START 上的信号进行地址译码以及 C、B、A 引脚上的信号共同组合而得到的。

（2）A/D 转换结果的读取方式：通常有查询和中断两种方式。

① 查询方式：单片机不断查询 ADC0809 芯片的 EOC 管脚，当此管脚为低电平时，表示正在进行 A/D 转换，则继续查询；若为高电平时，表示 A/D 转换已经完成。此时查询结束，当 OE 管脚高电平有效时，单片机就可以从 0809 芯片的 D0～D7 管脚读取 A/D 转换结果。

② 中断方式：采用中断方式读取数据时，EOC 管脚需要经过一个“非”门连接到单片机的外部中断请求线（低电平有效）上。当正在进行 A/D 转换时，EOC 管脚处于低电平状态，

不会产生中断请求；而当 A/D 转换结束后，EOC 管脚处于高电平状态，经过反门后得到低电平，此时单片机的外部中断请求线管脚有效，这样 ADC0809 就对单片机产生了一个中断请求信号，来通知单片机 A/D 转换已经结束，此时单片机就可以将 A/D 转换的结果读走了。同时，单片机会响应 ADC0809 芯片提出的这个中断请求，于是在单片机的中断服务程序中就会对刚才由 ADC0809 转换得到的结果，进行相应的处理。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。
2. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。
3. 主机板一块，统键盘/显示板一块，A/D 和 D/A 转换板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

五. 实验内容

多通道 A/D 转换应用系统的设计

使用模数转换器芯片 ADC0809，设计一个单片机多通道模数采集系统。该数据采集系统，每个通道采集模拟信号的范围是 0V～+5V，转换得到的数字量范围是 00H～FFH。并在显示板上进行显示。查询方式和使用中断方式可由学生自选，编程语言也可可由学生自选，ADC0809 芯片的具体地址可通过跳线进行选择。

【实验提示】:

ADC0809 芯片通过 IN0～IN7 通道，将模拟电压 0～+5V 采集到 A/D 转换器 0809 中，当 START 引脚是高电平时，ADC0809 开始将采集的模拟电压信号转变为数字信号，转换后的数字信号范围是 00H～FFH。单片机通过 P0～P3 并口将转换后得到的数字量送入单片机的 CPU 中，再输出到显示板上显示数字量。

六. 实验报告

1. 通过本次实验，总结单片机控制 A/D 转换器进行模数转换的软、硬件设计方法；
2. 写出所做实验程序的源代码，加上必要的注释，并画出程序流程图；
3. 叙述程序调试过程中遇到的问题和解决方法，写出本次实验的收获和心得体会。

实验八　　D/A 转换实验

一. 实验目的

1. 了解 D/A 转换的基础知识及并行 D/A 转换器 DAC0832 芯片的应用；
2. 掌握单片机控制 D/A 转换器进行数模转换的软、硬件设计方法；
3. 掌握通过 D/A 转换器实现三角波、方波、锯齿波等不同波形的控制方法。
4. 熟悉和掌握 A/D 和 D/A 转换板的结构。

二. 预习与思考

2. 预习理论教材中“D/A 转换”的相关内容；
2. 什么是 D/A 转换？D/A 转换的用途？DAC0832 的内部结构和工作原理？
3. D/A 转换的技术指标；如何编程实现启动 D/A 转换？如何得到 D/A 转换的结果？
4. 预习实验讲义附录中提供的 A/D 和 D/A 转换板的电路原理图以及电路板图，了解它们的结构。

三. 实验原理

1. D/A 转换的基础知识

D/A 转换是指将数字量转换为模拟量的过程。D/A 转换的工作原理是以电阻解码网络为基础的，常用的电阻解码网络有二进制权电阻解码网络和 T 型电阻解码网络。转换的过程是先将各位数码按其权的大小转换为相应的模拟分量，然后使用叠加的方法把各分量合成，其和就是 D/A 转换的结果。如图 8.1 所示，是使用 T 型电阻网络来进行 D/A 转换的原理图。

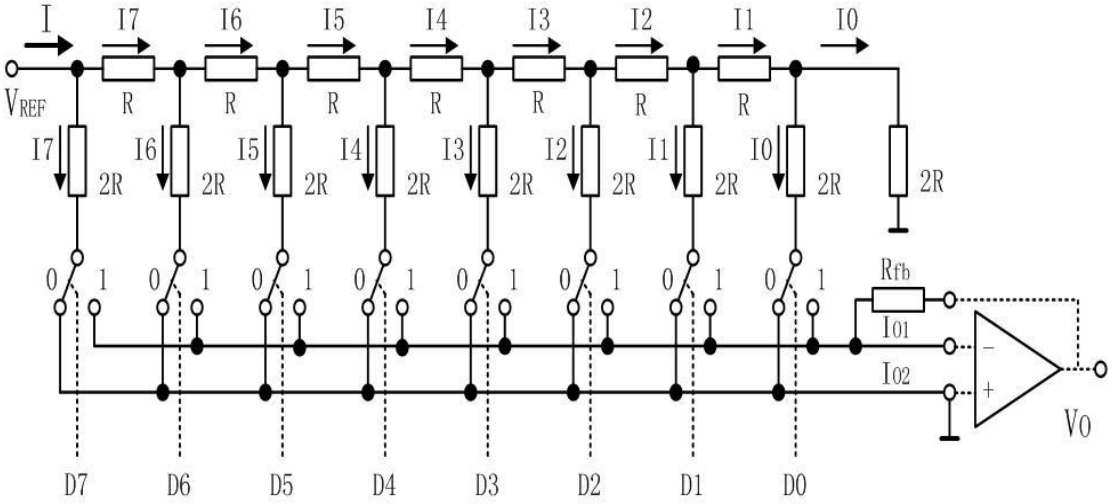


图 8.1　T 型电阻网络的 D/A 转换原理图

目前，常用的 D/A 转换主要包括电压输出型和电流输出型，下面将简要介绍这两种类型的 D/A 转换。

（1）电压输出型 D/A 转换：经过 D/A 转换后，输出的模拟量是电压。通常采用内置输出

放大器以及低阻抗的输出电压方式，也有直接从电阻阵列输出电压的情况。但由于直接输出电压的D/A转换器通常用于高阻抗负载，并且没有输出放大器的延迟，因此常把直接从电阻阵列输出电压的情况用于高速D/A转换器，其他情况用于中、低速D/A转换。

（2）电流输出型D/A转换：即经过D/A转换以后，输出的模拟量是电流，但通常此电流不直接输出给设备，一般情况都是通过外接“电流－电压”转换电路，最终得到电压信号后再传输给相应的外围设备。在外接“电流－电压”的转换电路时，有两种方法：一种是在D/A转换器的输出管脚上直接连负载电阻，从而实现“电流－电压”的转换；另一种方法是在D/A转换器的输出管脚上连接运算放大器，从而实现D/A转换后输出电压。通常，由于电流型D/A转换器，多数使用外接运算放大器的方法，所以速度较电压型D/A转换要慢些。通常，若不接运算放大器，则D/A转换后直接输出的小功率或弱电模拟信号，可以被用作大功率或强电模拟信号的控制信号。

D/A转换的技术指标主要包括以下四种：

（1）分辨率：是指数字量变化一个最小的单位时，模拟量变化的大小即D/A转换能分辩的最小输出模拟增量。通常，以D/A转换器的位数来表示。

（2）转换精度：是指在满量程的情况下，D/A转换的实际模拟输出值和理论值的接近程度。通常，转换精度与D/A转换输出的数字量位数有关，位数越多，误差越小，并且一般的D/A转换过程中转换精度都是分辨率的一半。

（3）线性度：指D/A转换的实际特性曲线和理想直线间的最大偏差。

（4）转换时间：将输入的数字量转换为稳定的模拟量输出所用的时间。

（5）输出信号的类型和信号变化范围。

一个数字量转换成模拟量需要使用D/A转换器(Digital to Analog Converter，DAC)，在单片机的实际应用中，可以选择两种类型的DAC即并行DAC和串行DAC。所谓并行DAC是指D/A转换器每次能从单片机接收到的多位数字量，然后将它们共同转换成模拟量，再传输给外围设备使用；而串行DAC是指D/A转换器每次只能从单片机接收到1位数字量，等到需要转换的所有位数字量都接收后，再启动D/A转换,并将得到的模拟量传输给外围设备进行处理。从D/A转换的效率来看，并行DAC要优于串行DAC，但从价格和应用方便的角度看串行DAC要好于并行DAC。近年来，串行DAC在应用中使用较多，但程序设计相对复杂。本实验使用的是并行D/A转换器，这里以常用的并行D/A转换器DAC0832芯片为例来进行介绍。

2. 并行D/A转换器芯片DAC0832

并行D/A转换器芯片DAC0832是一种带有输入锁存器以及输入寄存器的两级8位电流输出型D/A转换器芯片，由美国国家半导体公司研制。该芯片具有以下特性：

（1）分辨率为8位（即1/255），单一的电源供电（+5V～+15V）；

（2）具有单缓冲、双缓冲以及直通输入等3种工作方式；

（3）逻辑输入电平与TTL电平兼容，低功耗，只有20mw。

DAC0832芯片的内部结构如图8.2所示。从图中可以看出，DAC0832内部主要由三部分组成，具体如下：

（1）8位数据锁存器：用于存放来自CPU的数字量，使这个数字量得到缓冲和锁存，它的控制信号是$\overline{LE1}$。当$\overline{LE1}$由高变低时，此锁存器锁存D0～D7上来自单片机的数字量。这个锁存器所完成的动作，就好比是人们在餐桌上吃饭的时候，首先要说的“上菜”口令。

（2）8位DAC寄存器：用于存放待转换的数字量，它的控制信号是$\overline{LE2}$。这个寄存器所完成的动作，就好比是人们在餐桌上，当菜已经上齐后，要发出的“干杯”口令。此口令发出后，人们就开始吃饭了，相应的D/A转换器就开始将数字量转换为模拟量。

（3）8位D/A转换器：由8位T型电阻网络和电子开关组成，电子开关受“DAC寄存器”输出的数字量控制，T型电阻网络输出与数字量成正比的模拟电流。这个转换器所完成的动

作，就好比是在“上菜”口令和“干杯”口令都发出之后，人们就开始吃饭了。当 DAC0832 芯片将数字量转换成电流输出后，通常要再连接一个运算放大器，从而把电流转换成电压。

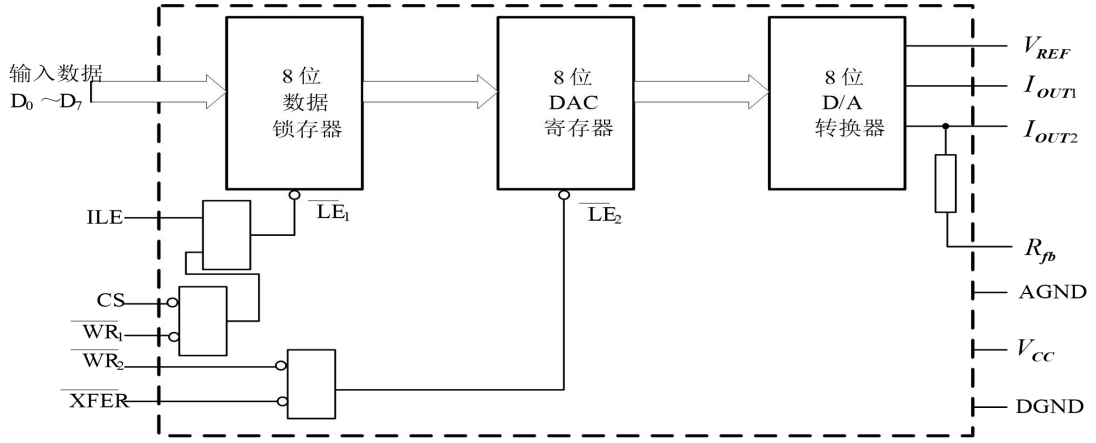


图 8.2　DAC0832 芯片的内部结构

在理解了并行 D/A 转换器芯片 DAC0832 的内部结构以后，接下来要介绍 DAC0832 的外部管脚。如图 17.3 所示，DAC0832 芯片共有 20 个管脚，各管脚的具体功能如下：

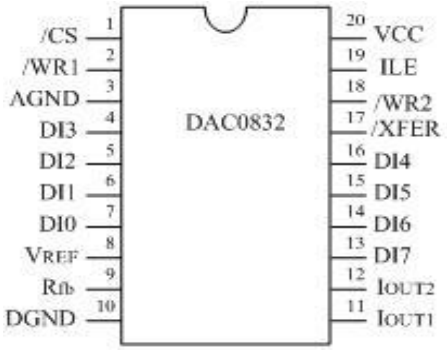


图 8.3　DAC0832 芯片的外部管脚图

（1）DI0～DI7：8 位数字量的输入管脚，通常与单片机的数据总线相连。

（2）控制管脚共有 5 个，即 ILE、$\overline{WR1}$、$\overline{CS}$、$\overline{WR2}$ 和 $\overline{XFER}$。

① ILE：数字量输入锁存控制管脚，高电平有效。

② $\overline{WR1}$、$\overline{WR2}$：写命令控制管脚，低电平有效。当 $\overline{WR1}$ 有效时，第 1 级 8 位数据锁存器打开，单片机的数字量可以写入；当 $\overline{WR2}$ 有效时，第 2 级 8 位 DAC 寄存器打开，待进行 D/A 转换的数据进入其中。

③ $\overline{CS}$：芯片选择管脚，低电平有效时，DAC0832 在硬件电路中被选中。

④ $\overline{XFER}$：数据传送控制管脚，低电平有效。一般用于多个 DAC 器的情况使用。

这里还要对控制信号作进一步的说明，其中/LE1 和/LE2 满足的逻辑关系是：

$\overline{LE1}$= ILE∩$\overline{WR1}$∩$\overline{CS}$　　$\overline{LE2}$=$\overline{WR2}$∩$\overline{XFER}$

当 $\overline{LE1}$ 为高电平时，数据锁存器状态随数据线变化，$\overline{LE1}$ 负跳变时将数据锁存在 8 位输入寄存器中。当 $\overline{LE2}$ 为高电平时，DAC 寄存器的输出随输入变化，$\overline{LE2}$ 负跳变时将 8 位输入寄存器的内容打入 DAC 寄存器并开始 D/A 转换。

（3）输出管脚：主要由 I_{OUT1}、I_{OUT2} 和 R_{fb} 共 3 个管脚组成，其中 R_{fb} 为运送放大器反馈电阻管脚，接运放输出端；I_{OUT1} 和 I_{OUT2} 是模拟电流输出管脚，且 $I_{OUT1}+I_{OUT2}$ 为常数。

当数字量为 FFH 时，I_{OUT1} 最大而 I_{OUT2} 最小；当数字量为 00H 时，I_{OUT1} 最小而 I_{OUT2} 最大。

（4）电源与接地管脚：工作电源 V_{CC} 管脚，范围是+5V～+15V；参考电压 V_{REF} 管脚，范围是 -10V～+10V；数字地管脚是 DGND；模拟地管脚是 AGND。

3. 单片机控制 D/A 转换的硬件设计

在实际应用中，并行 D/A 转换器芯片 DAC0832 有 3 种工作方式，即直通方式、单缓冲方式以及双缓冲方式，各种方式的具体含义如下：

（1）直通方式：在此方式下，DAC0832 芯片内部的 8 位输入锁存器和 8 位 DAC 寄存器的控制信号均处于有效状态，即以上两个寄存器不受控，只要有模拟量进入其中，立即直通到 D/A 转换器中进行转换，这种方式常用于不带计算机的控制系统中。

（2）单缓冲方式：在此方式下，DAC0832 芯片内部的 8 位输入锁存器和 8 位 DAC 寄存器的控制信号只有一个处于有效状态，即以上两个寄存器只有一个受控，如图 8.4 所示。

（3）双缓冲方式：如图 8.5 所示。这种方式是指 DAC0832 芯片的两个寄存器都受控，即两个寄存器均不处于直通状态，此方式也可以与单片机进行接口。

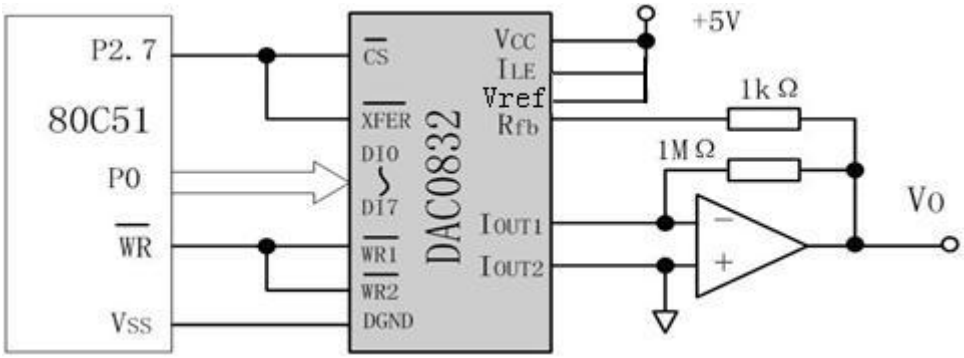


图 8.4　单缓冲单路方式 D/A 转换的控制原理图

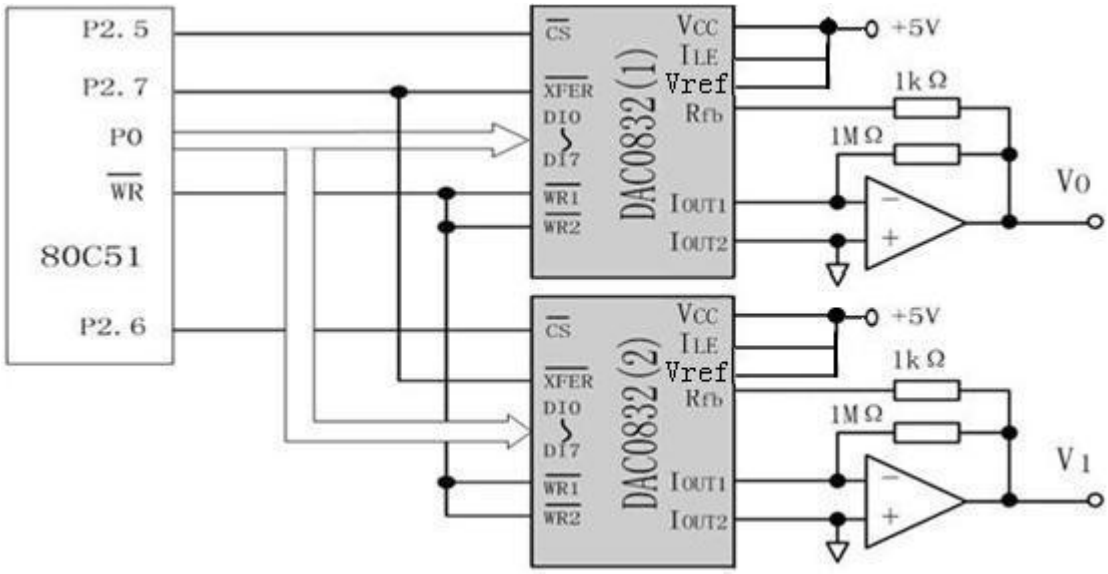


图 8.5　双缓冲多路方式 D/A 转换的控制原理图

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

3.主机板一块，键盘/显示板一块，A/D 和 D/A 转换板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

五. 实验内容

单缓冲单路 D/A 转换系统的设计

如图 8.4 所示,用现有板材设计一个单片机控制 DAC0832 芯片进行单缓冲 D/A 转换的应用系统。查询方式和使用中断方式可由学生自选，编程语言也可可由学生自选，ADC0809 芯片的具体地址可通过跳线进行选择。

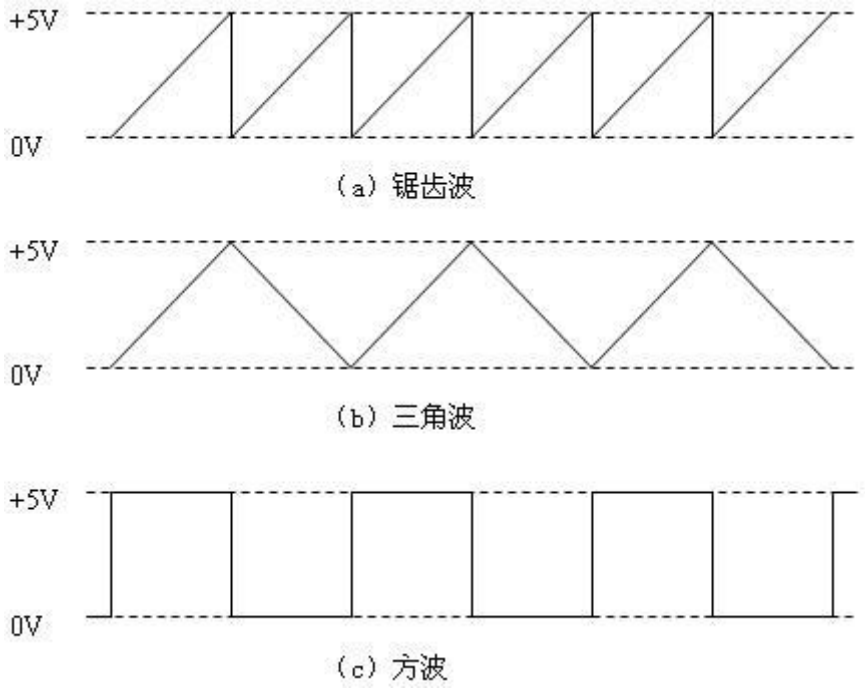


图 8.6　锯齿波、三角波以及方波的波形图

【实验提示】:

用单片机控制 DAC0832 芯片,进行单缓冲 D/A 转换。使用 DAC0832 来产生方波的波形时，只要使高、低电平的时间各自保持 1 秒钟，则小灯就会相应的点亮或者熄灭。当从某并口输入的数字量为 255 时，小灯最亮；而当某并口的值是 0 时，则小灯熄灭。因此，当某并口的值在 0～255 之间逐渐增加时，小灯的亮度应逐渐增强；而当某并口的值在 255～0 之间逐渐减少时，小灯的亮度应逐渐减弱。

要注意几个控制管脚的硬件连接。

六. 实验报告

1. 通过本次实验，总结单片机控制 D/A 转换器进行数模转换的软、硬件设计方法；
2. 写出所做实验程序的源代码，加上必要的注释，并画出程序流程图；
3. 叙述程序调试过程中遇到的问题和解决方法，写出本次实验的收获和心得体会。

实验九　　并口与存储器扩充实验

一. 实验目的

1. 了解程序、数据存储器扩充的基本原理和扩充的用途；
2. 理解 EPROM 2764 芯片和 SRAM 6264 芯片的内部结构和工作原理；
3. 理解 8255A 芯片的内部结构和工作原理；
4. 熟悉并口与存储器扩充板的结构。

二. 预习与思考

1. 预习理论教材中“程序、数据存储器扩充”的相关内容；
2. 预习理论教材中“并口扩充”的相关内容；
3. 预习实验讲义附录中提供的并口与存储器扩充板的电路原理图以及电路板图，了解它们的结构。

三. 实验原理

1、程序、数据存储器扩充的基础知识

在单片机应用系统开发中，最小系统是开发者普遍追求的。但在大多数的应用系统设计中，无论如何选型都无法实现最小系统，这就不可避免地要进行系统扩充。存储器扩充指的是外部存储器的扩充，外部存储器包括程序存储器和数据存储器，下面分别简单介绍外部程序存储器和数据存储器的扩充。

1.1 程序存储器扩充的基础知识

在单片机应用系统中，其运行的应用程序保存在 ROM 里，常见的 ROM 有 EPROM, E^2 PROM 和 Flash ROM。这里，我们以 EPROM 为例进行介绍。

1.1.1 EPROM　2764 芯片介绍

1.1.1.1　2764 芯片外部结构

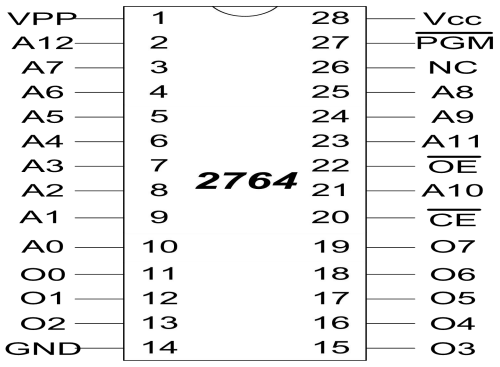


图 9.1　2764 引脚分配

1.1.1.2　2764 芯片引脚功能

①地址输入线 A_{12}～A_0：

2764 的存储容量为 8KB，故按照地址线条数和存储容量的关系（2^{13}=8192），共需 13 条地址线，即 A_{12}～A_0。2764 的地址线应和 MCS-51 单片机的 P2 和 P 0 口相接，用于传送单片机送来的地址编码信号，其中 A12 为最高位。

②数据线 O_7～O_0：

O_7～O_0是双向数据总线，O_7为最高位。在正常工作状态，O_7～O_0用于传送从 2764 中读出的数据或程序代码；在编程方式时用于传送需要写入芯片的编程代码（即程序的机器码）。

③控制线：2764 有 3 条控制线，分别是 $\overline{CE}$、$\overline{OE}$ 和 $\overline{PGM}$ 。

$\overline{CE}$ 是片选输入线，它用于控制本芯片是否工作。若给 $\overline{CE}$ 上加一个高电平，则本芯片不工作；若给 $\overline{CE}$ 上加一个低电平，则选中芯片工作。

$\overline{OE}$ 是允许输出线，它是由用户控制的输入信号线，若给 $\overline{OE}$ 上加一个高电平，则数据线 O_7～O_0处于高阻状态；若给 $\overline{OE}$ 上加一个低电平，则数据线 O_7～O_0处于读出状态。

$\overline{PGM}$ 是编程输入线，它用于控制 2764 处于正常工作状态或编程/校验状态。若给它一个高电平，则 2764 处于正常工作状态；若给它一个大于 50ms 的负脉冲，则 2764 配合 Vpp 引脚上的 21V 高压可以处于编程/校验状态。

④其他引脚线：

Vcc 为 5V±10%电源输入线；GND 为直流地线；Vpp 为编程电源输入线，当它接 5V 时，2764 处于正常工作状态，当它接 21V 高压时，2764 处于编程/校验状。NC 表示“不接”。

1.1.1.3 程序存储器扩充方法

根据理论课所讲授的知识，由单片机的并行接口 P2 提供高 8 位地址信号，并行接口 P0 分时提供低 8 位地址信号和 8 位双向数据信号。外部程序存储器的输出允许端（$\overline{OE}$）由单片机的读选通 $\overline{PSEN}$ 控制。这样，由于控制信号及使用的数据传送指令不同，即使外部数据存储器地址与外部程序存储器地址相同，也不会发生总线冲突。片选信号 $\overline{CE}$ 可以由没有用到的高位地址线经过译码得到。

图 9.2 是扩展一片 27264 的连接方法。当 MCS-51 系列单片机的 $\overline{EA}$ 引脚接低电平时，CPU 总是从外部 ROM 中取指令；当 $\overline{EA}$ 引脚接高电平时，CPU 取指令时，PC 值在内部程序存储器范围内从内部 ROM 中取指令，PC 值大于内部程序存储器范围时，CPU 从外部 ROM 中取指令。因此为了充分利用单片机资源，$\overline{EA}$ 引脚通常接高电平。下图是扩充一片 2764 的连接方法。

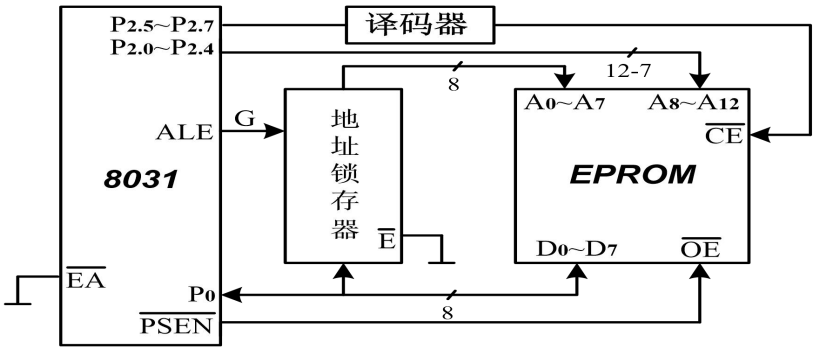


图 9.2 MCS-51 总线扩展与一片 2764 的接口电路

假定单片机片内 ROM 为 4K,要用 EPROM2764 芯片设计一个 8KB 的外部程序存储器系统，地址从 0000H 开始，应该如何实现？

用 EPROM2764 芯片扩充一个 8KB 的外部程序存储器之前，我们要知道需要几片这样的芯片。这需要通过简单公式计算就可以得出。

计算公式：(要扩充的外部程序存储器的总位数)÷(芯片总位数)

计算过程：(8KByte×8Bit)÷(8KByt×8Bit)＝1 片。

EPROM 2764 芯片与用单片机的连接：

外部程序存储器的地址从 0000H 开始，这说明 MCS-51 芯片的 $\overline{EA}$ 引脚应该接地。

①地址总线的连接

2764 芯片的低 8 位地址线 A_7～A_0：与单片机的 $P_{0.7}$～$P_{0.0}$ 经“地址锁存器”后连接。

2764 芯片的高 5 位地址线 A_{12}～A_8：直接与单片机的 $P_{2.4}$～$P_{2.0}$ 连接。

②数据总线的连接

2764 芯片的数据线 O_7～O_0：直接与单片机的 $P_{0.7}$～$P_{0.0}$ 连接。

③ROM 的读信号（允许输出线）

2764 芯片的 $\overline{OE}$：直接与单片机的 $\overline{PSEN}$ 连接。

④ROM 的片选输入线 $\overline{CE}$

到目前为止，2764 芯片的所有可用管脚中，只有片选输入信号线 $\overline{CE}$ 没有连接，所以，如何生成 $\overline{CE}$ 的逻辑是下一步要解决的问题。

2764 芯片的地址范围：0000H～1FFFH
0000H＝000　0,0000,0000,0000B
1FFFH＝000　1,1111,1111,1111B

由此可见，地址信号 A_{15}～A_{13} 应该作为芯片地址译码的输入条件，且一定要满足这样的条件：$A_{15}A_{14}A_{13}$＝000B；才能选中 2764 芯片。

∴ $\overline{CE}=A_{15}\cdot A_{14}\cdot A_{13}$

1.2 数据存储器扩充的基础知识

在单片机应用系统中，所处理的数据量非常大，其片内 RAM 无法进行存储时，就需要扩充数据存储器，常见的 RAM 有 SRAM 和 DRAM。这里，我们以 SRAM 为例进行介绍。

1.2.1 SRAM 6264 芯片介绍

1.2.1.1 SRAM 6264 芯片的外部结构

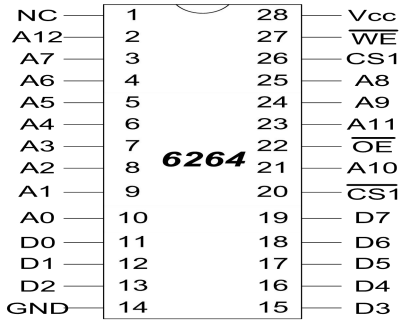


图 9.3　6264 引脚分配

1.2.1.2 SRAM 6264 芯片的引脚功能

①地址输入线 A_{12}～A_0：

6264 的存储容量为 8KBytes，A_{12} 是 6264 芯片的地址线最高位，A_0 是 6264 芯片的地址

线最低位。

②数据线 D_7～D_0：

双向三态数据总线，D_7为最高位，D_0为最低位。正常工作时，D_7～D_0用来传送 6264 的读写数据。

③控制线：6264 有 4 条控制线，分别是 CS1、$\overline{CS1}$、$\overline{OE}$ 和 $\overline{WE}$ 。

CS1 和 $\overline{CS1}$ 是片选输入线，若 CS1=1 且 $\overline{CS1}$=0 时，本芯片被选中工作；否则不被选中工作。

$\overline{OE}$ 是允许输出线，用于控制从 6264 中读出的数据是否送到数据线 D_7～D_0上。若 $\overline{OE}$ 为低电平，则读出的数据可以直接送到数据线 D_7～D_0上，否则读出的数据只能到达 6264 的内部总线。

$\overline{WE}$ 是读写命令线，若为低电平，6264 处于写入状态，若为高电平，则 6264 建立读出工作状态。

④其他引脚线：

Vcc 为 5V±10%电源输入线；GND 为直流地线；NC 表示“不接”。

1.2.1.3 工作方式

6264 有五种工作方式，其中的读出和写入方式是有效方式。下面是这五种工作方式的简单说明。

●禁止工作方式：CS1=1、$\overline{CS1}$=0、$\overline{OE}$ =0、$\overline{WE}$ =0 时，为禁止工作方式，因为 $\overline{OE}$ 和 $\overline{WE}$ 不能同时为低电平。

●读出工作方式：CS1=1、$\overline{CS1}$=0、$\overline{OE}$ =0、$\overline{WE}$ =1 时，为读出工作方式。

●写入工作方式：CS1=1、$\overline{CS1}$=0、$\overline{OE}$ =1、$\overline{WE}$ =0 时，为写入工作方式。

●选通工作方式：CS1=1、$\overline{CS1}$=0、$\overline{OE}$ =1、$\overline{WE}$ =1 时，为选通工作方式。

●未选通工作方式：CS1=1、$\overline{CS1}$=1、$\overline{OE}$ =X、$\overline{WE}$ = X 时，为未选通工作方式，其中 X 表示任意电平。

1.2.1.4 数据存储器扩充方法

外部数据存储器的扩充与外部程序存储器的一样，也是单片机的并行接口 P2 提供高 8 位地址信号，并行接口 P0 分时提供低 8 位地址信号和 8 位双向数据信号。外部数据存储器的读（$\overline{OE}$ ）和写（$\overline{WE}$ ）分别由单片机的 $\overline{RD}$（$P_{3.7}$）和 $\overline{WR}$（$P_{3.6}$）控制。

片选信号（CS1 和 $\overline{CS1}$）可以由没有用到的高位地址线经过译码得到。图 9.4 是扩展一片 6264 的连接方法。

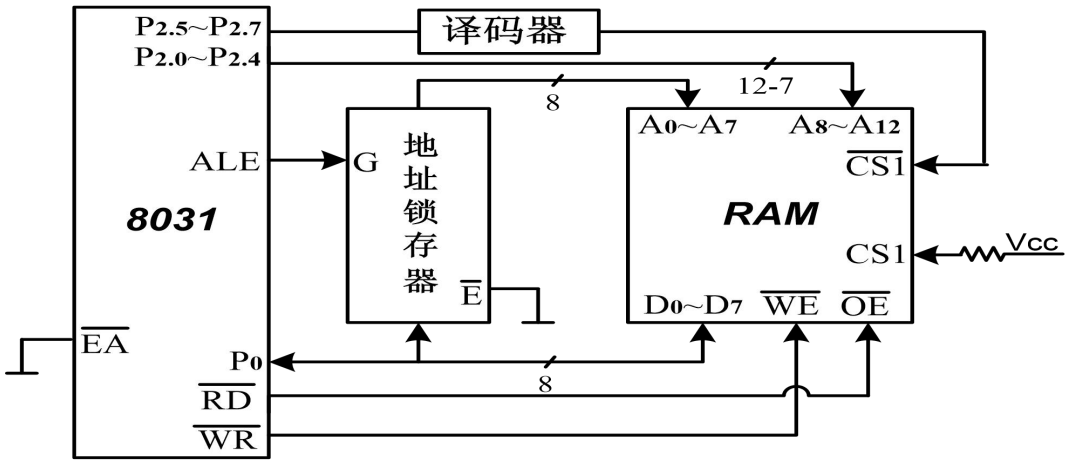


图 9.4 MCS-51 总线扩展与一片 6264 的接口电路

SRAM 6264 芯片与用单片机的连接：

假定上图外部数据存储器的地址从 2000H 开始，则其地址范围：2000H～3FFFH

即 0010,0000,0000,0000B～0011,1111,1111,1111B

①地址总线的连接

6264 芯片的低 8 位地址线 A_7～A_0：与单片机的 $P_{0.7}$～$P_{0.0}$ 经“地址锁存器”后连接。

6264 芯片的高 5 位地址线 A_{12}～A_8：直接与单片机的 $P_{2.4}$～$P_{2.0}$ 连接。

②数据总线的连接

6264 芯片的数据线 D_7～D_0：直接与单片机的 $P_{0.7}$～$P_{0.0}$ 连接。

③RAM 的读信号（允许输出线）

6264 芯片的 $\overline{OE}$：直接与单片机的的 $\overline{RD}$（$P_{3.7}$）连接。

④RAM 的写信号（读写命令线）

6264 芯片的 $\overline{WE}$：直接与单片机的的 $\overline{WR}$（$P_{3.6}$）连接。

⑤RAM 的片选输入信号线 CS1 和 $\overline{CS1}$

到目前为止，6264 芯片的所有可用管脚中，只有片选输入信号线 CS1 和 $\overline{CS1}$ 没有连接，所以，如何生成 CS1 和 $\overline{CS1}$ 的逻辑是下一步要解决的问题。

6264 芯片有二个片选输入信号 CS1 和 $\overline{CS1}$，其设计者的目的是为使用者在计算机数据存储器的设计时提供方便。在图 9.4 中，CS1=1 表示 CS1 接高电平，所以片选输入信号线 CS1 处于常有效状态。这样，只需生成 $\overline{CS1}$ 的逻辑即可。

由于 2764 芯片的地址范围：0000H～1FFFH

即地址范围：0000,0000,0000,0000B～0001,1111,1111,1111B

由此可见，地址信号 A_{15}～A_{13} 应该作为芯片地址译码的输入条件，且一定要满足这样的条件：$A_{15}A_{14}A_{13}$＝001B；才能选中 6264 芯片。

∴ $\overline{CS1}$= A_{15}・A_{14}・$\overline{A}_{13}$

1.3 数据、程序存储器混合扩充方法

按照数据存储器扩展和程序存储器扩展的原理，我们很容易联想并掌握数据、程序存储器混合扩展方法。

假定我们需要设计一个外部存储器系统，ROM 为 16KB（地址从 0000H 开始），RAM 也为 16KB（地址从 0000H 开始），这样的存储系统应该如何实现？

如果要使用 EPROM 2764 芯片扩展成 16KB 的外部程序存储器，可以通过计算得出需要几片这样的芯片：(16KByte×8Bit)÷(8KByte×8Bit)＝2 片。

同理也可以求出需要 2 片 SRAM 6264 芯片。

外部程序存储器的地址从 0000H 开始，说明 MCS-51 芯片的 $\overline{EA}$ 引脚应该接地。单片机的其它管脚（如地址总线、数据总线、$\overline{PSEN}$、$\overline{RD}$ 和 $\overline{WE}$ 等）与 SRAM 6264、EPROM 2764 的连接同前面所述。

下面分别求出 2764 和 6264 芯片的片选信号。

$2764^{\#1}$（地址范围 0000H～1FFFH）的片选信号 $\overline{CE}$：

0000H＝000 0,0000,0000,0000B

1FFFH＝000 1,1111,1111,1111B

由此可见，地址信号的 A_{15}～A_{13}作为芯片地址译码的输入条件，且一定要满足这样的条件：$A_{15}A_{14}A_{13}$＝000B；才能选中 $2764^{\#1}$芯片。

$2764^{\#2}$（地址范围 2000H～3FFFH）的片选信号 $\overline{CE}$：

2000H＝001　0,0000,0000,0000B

3FFFH＝001　1,1111,1111,1111B

同理，地址信号 A_{15}～A_{13}作为芯片地址译码的输入条件，且满足：$A_{15}A_{14}A_{13}$＝001B；方可选中 $2764^{\#2}$ 芯片。

对于 SRAM　6264，它有 2 个片选信号：即 CS1 和 $\overline{CS1}$，在实际使用中我们可以让其中一个片选信号永远满足条件，而另一个片选信号作为片选条件，这样就可以达到目的。具体让 CS1 和 $\overline{CS1}$ 当中的哪一个永远满足条件呢？这需要具体问题具体分析。

在本题中，外部程序存储器和外部数据存储器的地址都是 0000H～3FFFH，并且都是 16KB 的容量，因此它们的片选信号也应该是一致的。而 2764 芯片的片选信号 $\overline{CE}$ 是低电平有效，如果也让 6264 芯片的条件片选信号低电平有效，这样不仅实现简单，而且节省逻辑电路。

CS1=1（永远满足条件），$\overline{CS1}$ 作为作为条件片选；可以得出如下结论：

$6264^{\#1}$（地址范围 0000H～1FFFH）：片选信号 $\overline{CS1}$ 必须满足条件 $A_{15}A_{14}A_{13}$＝000B 时，该芯片方可被选中。

$6264^{\#2}$（地址范围 2000H～3FFFH）：片选信号 $\overline{CS1}$ 必须满足条件 $A_{15}A_{14}A_{13}$＝001B 时，该芯片方可被选中。

这样，2764 和 6264 两种芯片的片选信号已经确定，接下来是地址信号高 3 位（$A_{15}A_{14}A_{13}$）的译码问题。对于地址译码，通常可以通过两种方法来实现，一种方法是用基本的逻辑门电路实现，另一种方法是用现有的译码器芯片来实现。下面简单介绍这两种方法。

用基本的逻辑门电路实现地址译码：

前面我们已经得出 $2764^{\#1}$ 的片选信号 $\overline{CE}$，其地址译码要满足条件：$A_{15}A_{14}A_{13}$＝000B，并且一定要低电平输出，因此其逻辑表达式是：$\overline{CE}$ ＝A_{15}＋A_{14}＋A_{13}，有了这个逻辑表达式，用基本的逻辑门实现变得非常简单的事情。

同理，$2764^{\#2}$ 片选信号 $\overline{CE}$ 地址译码的逻辑表达式是：$\overline{CE}$ ＝A_{15}＋A_{14}＋$\overline{A}_{13}$

用译码器芯片来实现地址译码：

目前，常用的译码器芯片是 74LS139（二—四译码器）和 74LS138（三—八译码器），它们都是低电平输出。此例中，各使用 2 片 2764 和 6264，故采用二—四译码器即 74LS139 就够用了，本题的解决就是使用了二—四译码器，具体接法见图 9.5。

片选信号问题解决了，接下来是如何解决读/写信号的连接问题。

对于 2764，它是 ROM，只读不写，其内部装的是程序代码，因此它的 $\overline{OE}$ 应该与 MCS-51 的 $\overline{PSEN}$ 相连，它的 $\overline{PGM}$ 接高点平。

对于 6264，它是 RAM，既可以读又可以写，其内部装的是数据。它有两个控制信号 $\overline{OE}$ 和 $\overline{WE}$，其读出工作方式是 $\overline{OE}$ =0 且 $\overline{WE}$ =1；写入工作方式是 $\overline{OE}$ =1 且 $\overline{WE}$ =0。而 MCS-51 有读信号 $\overline{RD}$（$P_{3.7}$）和写信号 $\overline{WR}$（$P_{3.6}$），因此，我们把 MCS-51 的 $\overline{RD}$ 与 6264 的 $\overline{OE}$ 直接相连，把 MCS-51 的 $\overline{WR}$ 与 6264 的 $\overline{WE}$ 直接相连就完成了。

下图是使用 6264 和 2764 芯片扩展 16KB 数据存储器和 16KB 程序存储器的连接方法。

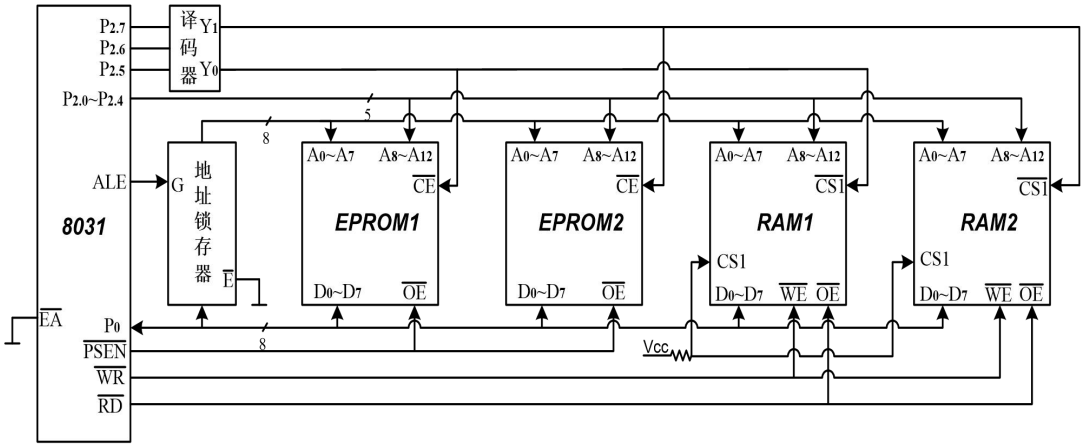


图 9.5　MCS-51 总线扩展与 2 片 6264、2 片 2764 的接口电路

2、并行接口扩充的基础知识

MCS-51 系列单片机具有 4 个 8 位并行接口，即 P0、P1、P2 和 P3，原理上这 4 个 I/O 口均可用作双向并行 I/O 口，但在实际应用中，P0 口常被用作数据总线和低 8 位地址总线，P2 口常被用作高 8 位地址总线，P3 口的某些位又常用它的第二功能，特别是无 ROM 型的单片机因为必须扩展外部程序存储器，则更是如此。所以，若一个 MCS-51 应用系统需要连接较多的并行输入/输出外围设备（如打印机、键盘、显示器等），则单片机本身所提供的 I/O 接口资源就不能满足需要了，这就不可避免地要扩充 I/O 接口。

8255A 是 Intel 公司生产的通用可编程并行 I/O 接口芯片，主要是为 8080/8085 而设计的，也可用于 MCS-51 系列单片机。8255A 具有三个 8 位并行 I/O 端口，分别为 A 口、B 口和 C 口，其中 C 口又分为高 4 位口和低 4 位口，他们都可以通过编程来改变其工作模式。目前 8255A 已成为单片机应用系统中并行接口扩充最常用的芯片。

2.1　8255A 芯片结构及引脚功能

2.1.1　8255A 的芯片外部结构

8255A 的芯片外部结构如图 8.6 所示，它主要由四部分电路组成，分别是：A 口、B 口和 C 口，A 组控制器和 B 组控制器，数据总线驱动器，读/写控制逻辑。

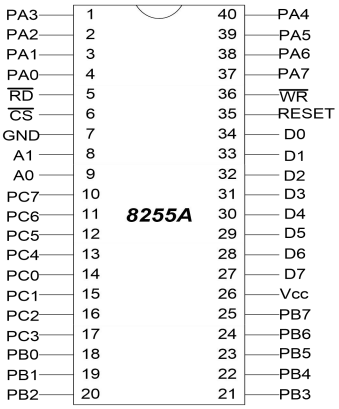


图 9.6　8255A 引脚图

① A 口、B 口和 C 口：A 口、B 口和 C 口均为 8 位 I/O 数据端口，但在结构上略有差别。A 口由一个 8 位数据输出缓冲/锁存器和一个 8 位数据输入缓冲/锁存器组成；而 B 口和 C 口各有一个 8 位数据输出缓冲/锁存器和一个 8 位数据输入缓冲器（无数据输入锁存器）

组成。在使用上，三个端口都可以直接与设备相连，分别传送外设的输入/输出数据和控制信息。但在模式 1 和模式 2 下，A 口和 B 口常被用作数据口来传送数据， C 口作为控制信号接口，高 4 位属于 A 口，传送 A 口上外设的控制/状态信息；低 4 位属于 B 口，传送 B 口上外设的控制/状态信息

② A 组控制器和 B 组控制器：两个控制器都由控制字寄存器和控制逻辑组成。控制字寄存器可以接收并存放来自 CPU 的决定端口工作方式的信息，即工作方式控制字。控制逻辑用于对 8255A 工作模式的控制。A 组控制器控制 A 口和 C 口的高 4 位，B 组控制器控制 B 口和 C 口的低 4 位。

③ 数据总线驱动器：它是一个双向三态 8 位缓冲器，用于与单片机的数据总线相连，实现单片机与 8255A 之间的数据传送。

④ 读/写控制逻辑:这部分电路可以接收单片机发来的读/写命令和选口地址,用于对 8255A 的读/写。

2.1.2 8255A 芯片的引脚功能

①数据总线：

D_7～D_0为数据总线，用于传送单片机和 8255A 之间的数据、命令和状态信息。

②并行 I/O 总线：

共有 24 条，用于与外设相连，分为三组。

A 口的 PA_7～PA_0：双向数据 I/O 总线，PA_7为最高位。可以工作于模式 0、模式 1 和模式 2，由控制字决定。

B 口的 PB_7～PB_0：双向数据 I/O 总线，PB_7为最高位。可以工作于模式 0 和模式 1，由控制字决定。

C 口的 PC_7～PC_0：双向数据/控制总线，PC_7为最高位。

③控制总线：有 6 条。

RESET：8255A 复位控制端，高电平有效。有效时芯片复位，复位状态为：控制寄存器被清除，PA、PB 和 PC 端口被设置成输入方式。

$\overline{CS}$：片选输入线，低电平有效。若$\overline{CS}$=1，则本 8255A 未被选种工作；若$\overline{CS}$=0，则被选种工作。

$\overline{RD}$：读命令输入线，低电平有效。有效时允许 CPU 读取 8255A 的数据或状态字。

$\overline{WR}$：写命令输入线，低电平有效。有效时允许 CPU 向 8255A 写入数据或控制字。

A_1和 A_0：端口选择信号输入线。通过 A_1和 A_0可以使 8255A 的 A 口、B 口、C 口和控制寄存器哪个被选中工作。

A_1A_0=00 时，A 口被选中；

A_1A_0=01 时，B 口被选中；

A_1A_0=10 时，C 口被选中；

A_1A_0=11 时，控制寄存器被选中。

④电源线：Vcc 为 5V±10%电源输入线；GND 为直流地线。

2.1.3 8255A 的控制字

8255A 有两个控制字，一个是模式控制字，一个是 C 口单一置位/复位控制字。可以使用指令把这两个控制字写入 8255A 的控制字寄存器（A_1A_0=11B），来设定 8255A 的工作模式和 C 口的各位状态。

模式控制字和 C 口单一置位/复位控制字是以最高位 D_7进行区分的。即 D_7为控制字标志位，当 D_7=1，则为模式控制字；当 D_7=0，则为 C 口单一置位/复位控制字。

① **模式控制字**

8255A 的三个端口的工作方式是由模式控制字来设定的。模式控制字的具体格式如图 9.7 所示。

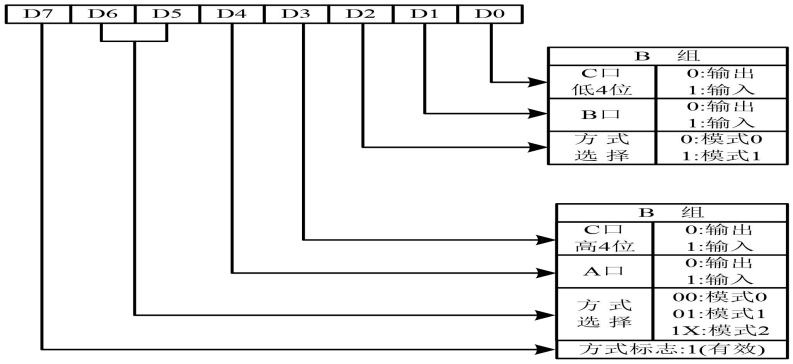


图 9.7　8255A 模式控制字

D_6～D_3为 A 组控制位。其中 D_6、D_5为 A 组模式选择位，用于设定 A 组的工作模式。

D_6D_5=00 时，A 组为模式 0；

D_6D_5=01 时，A 组为模式 1；

D_6D_5=1X 时（X 为任意值），A 组为模式 2。

D_4为 PA 口输入/输出控制位，用于设定 A 口是输入或输出。

D_4=0 时，则 A 口为输出；

D_4=1 时，则 A 口为输入。

D_3为 C 口高 4 位输入/输出控制位，用于设定 C 口高 4 位是输入或输出。

D_3=0 时，C 口高 4 位（即 PC_7～PC_4）为输出；

D_3=1 时，C 口高 4 位（即 PC_7～PC_4）为输入。

D_2～D_0为 B 组控制位。其中 D_2为 B 组模式选择位，用于设定 B 组的工作模式。

D_2=0 时，B 组为模式 0；

D_2=1 时，B 组为模式 1。

D_1为 B 口输入/输出控制位，用于设定 B 口是输入或输出。

D_1=0 时，则 B 口为输出；

D_1=1 时，则 B 口为输入。

D_0为 C 口低 4 位输入/输出控制位，用于设定 C 口低 4 位是输入或输出。

D_0=0 时，C 口低 4 位（即 PC_3～PC_0）为输出；

D_0=1 时，C 口低 4 位（即 PC_3～PC_0）为输入。

② **C 口单一置位/复位控制字**

这个控制字可以对 C 口的每位进行独立的置位或复位，以实现某些特定的功能。该控制字的格式如图 9.8 所示。

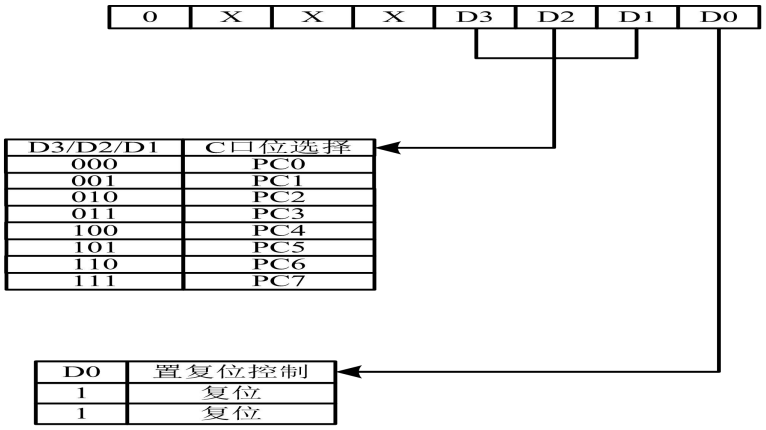


图 9.8　8255A 的 C 口单一置位/复位控制字

其中，D_7=0，它是 C 口单一置位/复位控制字的特征位。D_7=1 是方式控制字的特征位。

D_3～D_1用于选中 PC_7～PC_0中的一位，D_0是置位或复位控制位。D_0=0 时，对 PC_7～PC_0中被选中的位进行复位，D_0=1 时，对 PC_7～PC_0中被选中的位进行置位。

2.1.4　8255A 的工作模式

8255A 有三种工作模式，即模式 0、模式 1 和模式 2。我们可以通过选用某种模式控制字，并借助指令将其送给 8255A 的控制字寄存器，就可以设定 8255A 的工作模式。

① 模式 0

8255A 的工作模式 0 也被称为基本输入/输出模式。在此模式下，8255A 的 PA 口、PB 口、PC 口高 4 位和 PC 口的低 4 位均可被设定为方式 0 的输入或方式 0 的输出。方式 0 适用于无条件的数据传输设备，交换数据的双方不需要握手信息，就可以进行简单的数据传送。

② 模式 1

8255A 的工作模式 1 也被称为选通输入/输出模式，即在此模式下，有选通输入和选通输出两种工作方式。8255A 的 PA 口、PB 口常被用作设备与 CPU 之间数据的传送，PC 口用作 PA 口、PB 口的握手联络信号线，以实现在中断方式下的数据传输。PC 口的各位联络线是在设计 8255A 时已经规定好的，详细规定见下表，表中标有 I/O 的位指的是在此方式下，该位仍可用作基本输入/输出，不作联络线用。

下面我们分别按选通输入和选通输出两种情况来介绍。

▲选通输入方式：A 口和 B 口均可工作于此方式。

若 A 口和 B 口定义为模式 1 的选通输入方式，则 8255A 的内部逻辑结构如图 9.9 所示。

相应的联络线信号的意义如下：

$\overline{STB}_A$ 和 $\overline{STB}_B$：设备选通信号输入。低电平有效，下降沿将端口数据线上信息打汝端口锁存器。

IBF_A 和 IBF_B：端口锁存器满标志输出信号（该信号线与设备相连）。高电平表示端口锁存器中的数据尚未被 CPU 读取。CPU 读取后，该信号线输出低电平，表示端口锁存器已空。

$INTR_A$ 和 $INTR_B$：中断请求信号，高电平有效。

$INTE_A$ 和 $INTE_B$：8255A 端口内部中断允许触发器。当它为高电平时才允许端口发中断请求。$INTE_A$ 可由用户通过对 PC4 的单一置位/复位控制字控制，$INTE_B$ 可由用户通过对 PC2 的单一置位/复位控制字控制。

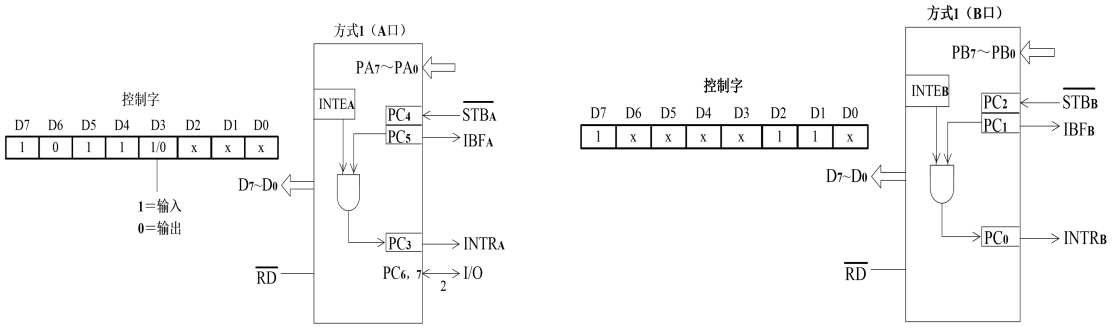


图 9.9　8255A 模式 1 输入逻辑组态

▲选通输出方式：A 口和 B 口均可工作于此模式。

若 A 口和 B 口定义为模式 1 的选通输出方式，则 8255A 的内部逻辑结构如图 9.10 所示。

相应的联络线信号的意义如下：

$\overline{OBF}_A$ 和 $\overline{OBF}_B$：输出锁存器满标志输出信号。低电平表示 CPU 已将数据写入端口，输出数据有效。设备从端口取走数据后发来的应答信号使其为高电平。

$\overline{ACK}_A$ 和 $\overline{ACK}_B$：设备输入应答信号。当 $\overline{ACK}$ 上出现设备发来的负脉冲时，表示设备已取走了端口数据。

$INTR_A$ 和 $INTR_B$：中断请求信号，高电平有效。

$INTE_A$ 和 $INTE_B$：8255A 端口内部中断允许触发器。当它为高电平时才允许端口发中断请求。$INTE_A$ 和 $INTE_B$ 分别由用户通过对 PC6 和 PC2 的单一置位/复位控制字控制。

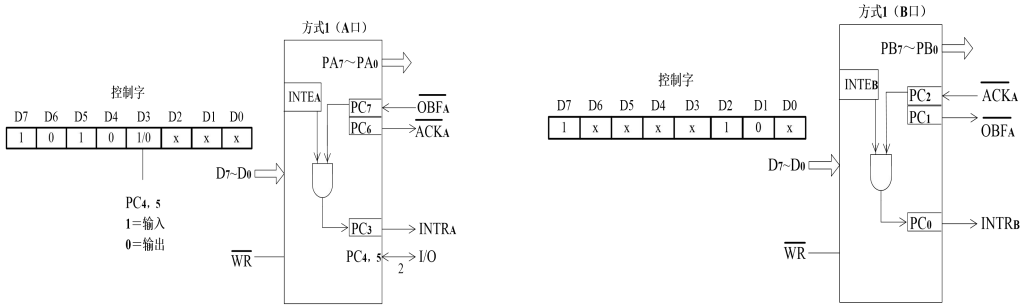


图 9.10　8255A 模式 1 输出逻辑组态

③ 模式 2

8255A 的工作模式 2 也被称为选通的双向输入/输出方式。

工作模式 2 是模式 1 的选通输入方式和模式 1 的选通输出方式的结合，并且只有 A 口才具备模式 2 的功能（B 口没有模式 2 的原因之一是它没有输入数据锁存器）。

在此模式下，A 口成为 8 位双向三态数据总线，C 口的 PC_7～PC_3 用来作为 A 组输入/输出的握手信号，而 B 口和 PC_2～PC_0 则可以被设置为方式 0 或方式 1 下工作。

A 口定义为工作模式 2 时，8255A 的内部逻辑结构如图 9.11 所示，此模式下所涉及的联络信号意义与模式 1 相同。

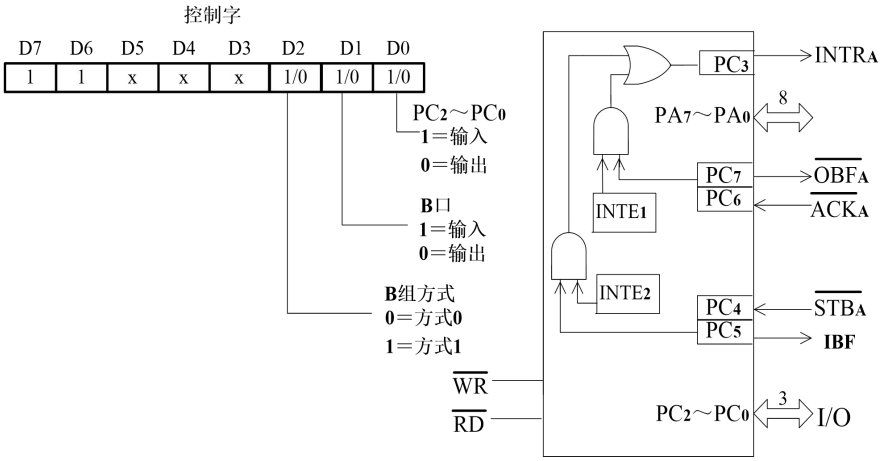


图 9.11　8255A A 口模式 2 逻辑组态

表 9.1　8255A 端口及工作状态选择表

C 口各位	模式 1		模式 2
	选通输入方式	选通输出方式	双向（输入/输出）方式
PC7	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$
PC6	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$
PC5	IBF_A	I/O	IBF_A
PC4	$\overline{STB}_A$	I/O	$\overline{STB}_A$
PC3	$INTR_A$	$INTR_A$	$INTR_A$
PC2	$\overline{STB}_B$	$\overline{ACK}_B$	由 B 口工作模式决定
PC1	IBF_B	$\overline{OBF}_B$	由 B 口工作模式决定
PC0	$INTR_B$	$INTR_B$	由 B 口工作模式决定

3、8255A 与 MCS-51 单片机的连接方法

8255A 与 MCS-51 单片机的连接有两种方法，一种是总线扩充连接法，一种是直接连接法。直接连接法如下图 9.12 所示。

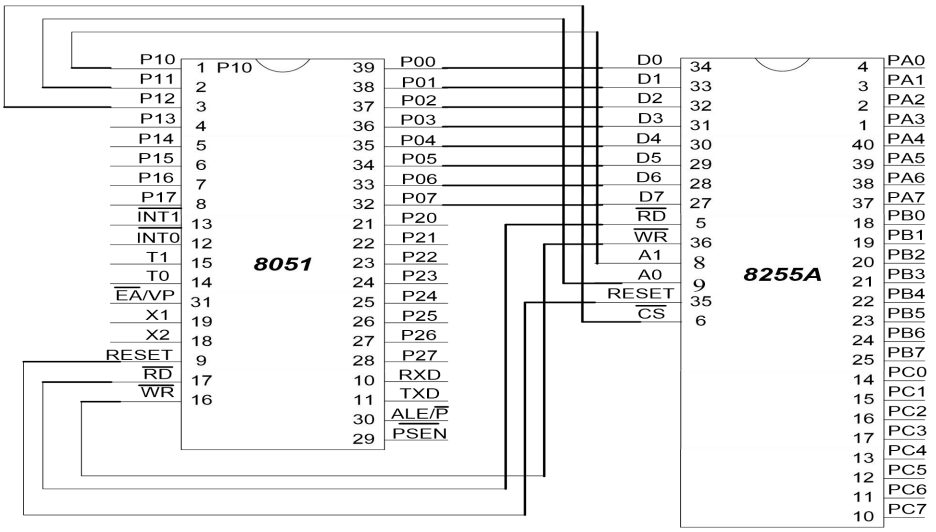


图 9.12　8255A 与单片机的直接连接

这种连接方法是将单片机的一个 I/O 口直接与 8255A 的数据总线相连，另外占用其它 I/O 口的 3 条线进行片选和端口选择。采用这种方式连接的优点是：结构简单明了。缺点是：因为要人为地控制片选信号，在编程方面要相对复杂。

总线扩充连接法如图 8.13 所示。

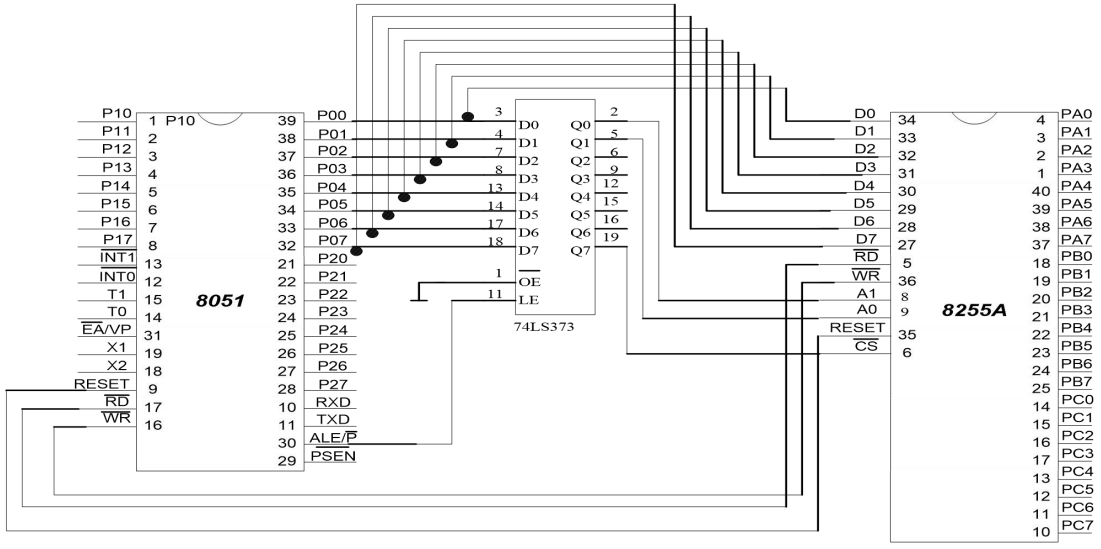


图 8.13　8255A 与单片机的总线扩充连接

总线扩充连接法是用 74LS373 或 74LS273 作为锁存器将低 8 位地址进行锁存，借用其中的 3 位作为 8255A 的片选和端口选择信号。这种连接方法的好处是只占用了一组 I/O 口换取了 3 个 I/O 口，缺点是多用了一个锁存器，浪费了许多外部存储单元资源（8255A 的地址不唯一）。

当然，如果应用系统比较复杂且需要外部资源较多，就必须通过精确的地址译码作为片选信号，这样势必要使用高 8 位地址。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

3.主机板一块，统键盘/显示板一块，并口与存储器扩充板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

五. 实验内容

1、数据存储器扩充设计

用单片机系统主机板、单片机系统键盘/显示板和并口与存储器扩充板设计一个单片机应用系统。将片内 RAM 30H～3FH 单元中的数据搬移到片外 RAM 1000H 开始的 16 个单元中。搬移后要通过键盘的控制，在数码管上分别可以显示片内 RAM 30H～3FH 单元和片外 RAM 1000H～100FH 中的数据，验证它们是否一致。

【实验提示】:

可以定义二个功能键，一个用于显示 30H～3FH 单元中的数据，另一个用于显示 1000H～100FH 单元中的数据。前 5 位数码管显示单元地址（十六进制），最后 2 位数码管显示单元中的数据（十六进制）。显示时，两个数之间必须要有一定的时间间隔。

2、并口扩充设计

用单片机系统主机板、单片机系统键盘/显示板和并口与存储器扩充板设计一个单片机应用系统。任意使用 8255A 的一个接口，实现实验三要求的的流水灯实验。

六. 实验报告

1. 通过本次实验，总结单片机与并口与存储器扩充板连接的软、硬件设计方法；
2. 写出所做实验程序的源代码，加上必要的注释，并画出程序流程图；
3. 叙述程序调试过程中遇到的问题和解决方法，写出本次实验的收获和心得体会。

实验十　　键盘输入接口实验

一. 实验目的

1. 理解矩阵键盘的基础知识及其工作原理。
2. 掌握单片机矩阵键盘的软件和硬件设计方法。
3. 进一步加深对具有矩阵键盘的单片机应用系统的开发方法的认识。

二. 预习与思考

1. 进一步熟悉单片机系统键盘、显示板的电路原理图以及电路板图。
2. 预习理论教材“单片机应用中的人机接口”的相关内容。
3. 理解如何判断键盘被按下以及如何获取按键的键值等方法。
4. 掌握键盘扫描的方法及其工作原理。
5. 理解为什么键盘要进行消抖？掌握键盘消抖的方法。

三. 实验原理

1. 基础知识

键盘是计算机系统不可缺少的输入设备，计算机操作者通过键盘可以向计算机输入各种操作命令或数据，计算机捕捉到按键信息后会进行相应的处理。同样，在单片机应用系统中，单片机捕捉到按键信息后，也会进行相应的处理。

从结构上分，可分为独立式键盘和行列矩阵式键盘。独立式键盘是指键盘中的各个按键的输入相互独立，每个按键都单独用一条线与计算机连接。独立式键盘的优点是连接简单，读取便捷。缺点是占用资源多，同样的资源所连接的按键少。

通常情况下，计算机系统都采用行列矩阵式键盘。所谓行列矩阵式键盘是将按键的输入接成行和列，按行列接入计算机系统中，从而形成一个具有“行 X 列”的矩阵。

2. 行列矩阵式键盘的工作原理

在行列矩阵式键盘中，人们能够使用什么办法来鉴别是否有按键被按下？是哪个按键被按下呢？在实际的应用中，人们可以通过使用软件方法或件硬方法来解决前面提出的问题。在没有按键被按下时，任意两条相互交叉的行与列线不连通，只有某个键盘被按下时，对应的行与列线才连通，处于这个行与列交叉点的按键就是被按下的按键。

3. 矩阵式键盘的扫描方法

详细内容请参见教材。

4. 键盘的消抖

由于按键基本都是机械开关式的，当进行敲击按键操作时，会由于反作用等原因产生多次敲击，而不是一次敲击，这种现象就是键盘的抖动。如果不及时排除，就可能会出现多次读取按键被按下的情况发生，导致产生误操作。因此，必须进行键盘的消抖。

键盘的消抖方法有硬件消抖和软件消抖两种。硬件消抖方法是使用消抖电路来实现，具体方法可以查阅教材。使用软件进行消抖也是人们常用的方法。根据对人们按键的动作分析，一般按键至少要持续 100ms，而抖动的时间大约是 10ms。这样，判断按键是否被按下时，可以采用如下方法：当第一次读到按键的键值时，先不要急于下结论，要至少在 10ms 以后再次读取键值，此时键盘的抖动已经过去，如果前后两次读取的键值相同，则可以认定这个按键被按下。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. 主机板一块，键盘/显示板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

3. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

五. 实验内容

1. 参照附录中单片机键盘/显示板上键盘排列结构，按照“逐行扫描法”或“行反转法”的原理，所以汇编语言或 C51 语言进行编程，实现对按键键值的获取。

2. 参照附录自行将主机板和键盘/显示板进行连接，设计形成一个单片机的硬件系统。

3. 参照自行设计的硬件系统，有针对性地进行应用程序的设计，即通过使用按键实现对定时器实验二中年、月、日、时、分、秒等参数进行设置。

4. 模拟调试完成后，用 STC-ISP 下载编程软件将生成的*。HEX 文件在线下载到单片机中。

5. *.HEX 文件在线下载到单片机后，按复位键执行程序，检验程序运行结果。

六. 需要注意的问题

两块电路板之间的电源连线有正负极，必须正确连接，否则会烧毁电路板。

七. 实验报告

1. 写出汇编语言或 C51 程序的源代码，加上必要的注释，并画出程序的流程图。

2. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

实验十一　　单片机应用系统综合实验一

一. 实验目的

1. 熟悉主机板、键盘/显示板的电路原理图以及电路板图。

2. 掌握各板之间的连接方法。

3. 理解主机板和统键盘/显示板各硬件接口的功能和特点，并能够针对硬件的连接使用汇编语言和 C51 语言进行编程和模拟调试，实现单片机应用系统综合实验。

4. 掌握 STC-ISP 下载编程软件的使用方法，将模拟调试成功的*.HEX 程序在线下载到单片机的片内 ROM 中，实际观察程序运行的直观效果。

二. 预习与思考

1. 预习实验讲义附录中提供的主机板和键盘/显示板的电路原理图以及电路板图，了解它们的结构。

2. 预习理论教材中“单片机定时器/计数器”的相关内容，了解定时器/计数器的 4 种工作方式，并结合硬件电路进行分析。

3. 思考如何将主机板和键盘/显示板进行连接，形成一个单片机的硬件系统，并进行相应定时器/计数器的应用程序的设计。

三. 实验原理

1. 参照附录自行将主机板和键盘/显示板进行连接，设计形成一个单片机的硬件系统。

2. 参照自行设计的硬件系统，有针对性地进行相应定时器/计数器应用程序的设计。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. 主机板一块，键盘/显示板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

3. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

五. 实验内容

1. 请使用/主机板和键盘/显示板设计一个硬件系统，实现一个交通路口红绿灯的控制。

2. 用 2-3 个数码管显示相对双向绿灯亮的倒计时时间（秒），在绿灯亮的最后几秒时，绿灯要闪烁，即亮灭交替，亮灭时间均为 0.5 秒，然后变成红灯。

3. 每组同学可自行设计进行连接，形成一个单片机硬件系统。

4. 模拟调试完成后，用 STC-ISP 下载编程软件将生成的*。HEX 文件在线下载到单片机中。

5. *.HEX 文件在线下载到单片机后，按复位键执行程序，检验程序运行结果。

【实验提示】:

此题可以使用汇编语言或 C51 语言编写的程序在硬件上实现.需要注意的问题：

（1）主机板上的、单片机型号、晶振要与下载编程软件一致。

（2）必须了解电路板的硬件及原理，了解连接后的硬件结构，针对硬件进行编程。

（3）中断服务程序尽可能短小，执行时间短，以减少不必要的麻烦。

（4）红绿灯的显示要与数码管显示同步。

六. 需要注意的问题

两块电路板之间的电源连线有正负极，必须正确连接，否则会烧毁电路板。

七. 实验报告

1. 画出两块电路板的连接图。
2. 写出汇编语言或 C51 程序的源代码，加上必要的注释，并画出程序的流程图。
3. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

八. 分组进行实验

每组二人，自由组合，每个人写自己的实验报告，除了实验报告中所要求的内容外，还要写出你在小组中所承担的工作。

实验十三　　单片机应用系统综合实验二

一. 实验目的

1. 熟悉主机板、键盘/显示板和 A/D、D/A 转换扩充板的电路原理图以及电路板图。

2. 掌握各示板之间的连接方法。

3.能够针对硬件的连接使用汇编语言和 C51 语言进行编程和模拟调试,实现单片机应用系统综合实验。

4. 掌握 STC-ISP 下载编程软件的使用方法，将模拟调试成功的*.HEX 程序在线下载到单片机的片内 ROM 中，实际观察程序运行的直观效果。

二. 预习与思考

1. 预习实验讲义附录中提供的主机板、键盘/显示板和 A/D、D/A 转换扩充板的电路原理图以及电路板图，了解它们的结构。

2. 预习理论教材中“单片机定时器/计数器”的相关内容，了解定时器/计数器的 4 种工作方式，并结合硬件电路进行分析。

3. 思考如何将主机板、键盘/显示板和 A/D、D/A 转换扩充板进行连接，并进行模/数转换，进行现场模拟量的采集，经数/模转换，输出数字量进行相应控制。完成这个应用系统的硬件设计，并针对硬件结构，完成应用程序的设计。

三. 实验原理

1. 参照附录自行将主机板、键盘/显示板和 A/D、D/A 转换扩充板进行连接，设计形成一个单片机的硬件系统。

2. 参照自行设计的硬件系统，有针对性地进行相应应用程序的设计。

四. 实验设备和器件

1. PC 机一台，操作系统为 Windows XP，内存 256MB 以上，硬盘 10GB 以上。

2. 主机板一块，键盘/显示板一块，A/D、D/A 转换扩充板一块，USB 线一条，9 针串口连接线一条，排电缆连接线若干条。

3. Keil μVision2 集成开发环境和 STC-ISP 下载编程软件。

五. 实验内容

1. 请使用单片机系统主机板和单片机系统键盘、显示板和 A/D、D/A 转换扩充板设计一个硬件系统，实现一个单点的集控制指标设置、模拟现场信息采集、现场信息显示和回控的控制系统。

2. 每组同学可自行设计进行连接，形成一个单片机硬件系统。

3. 模拟调试完成后，用 STC-ISP 下载编程软件将生成的*。HEX 文件在线下载到单片机中。

4. *.HEX 文件在线下载到单片机后，按复位键执行程序，检验程序运行结果。

【实验提示】:

此题可以使用汇编语言或 C51 语言编写的程序在硬件上实现.需要注意的问题：

（1）单片机系统主机板上的晶振与下载编程软件一致。

（2）必须了解电路板的硬件及原理，了解连接后的硬件结构，针对硬件进行编程。

（3）控制指标设置：借助键盘进行设置单点的运行指标。

（4）模拟现场信息采集：对 A/D 编程，实现现场的信息采集。

（5）现场信息显示：显示单点的现场信息和单点的设置信息。

（6）回控：对 D/A 编程，根据采集。

六. 需要注意的问题

两块电路板之间的电源连线有正负极，必须正确连接，否则会烧毁电路板。

七. 实验报告

1. 画出两块电路班的连接图，并说明两块电路班的连接接口都起什么作为。
2. 写出汇编语言或 C51 程序的源代码，加上必要的注释，并画出程序的流程图。
3. 叙述程序调试过程中遇到的困难以及解决方法，写出本次实验的收获和心得体会。

八. 分组进行实验

每组二人，自由组合，每个人写自己的实验报告，除了实验报告中所要求的内容外，还要写出你在小组中所承担的工作。

单片机系统开发板部件介绍

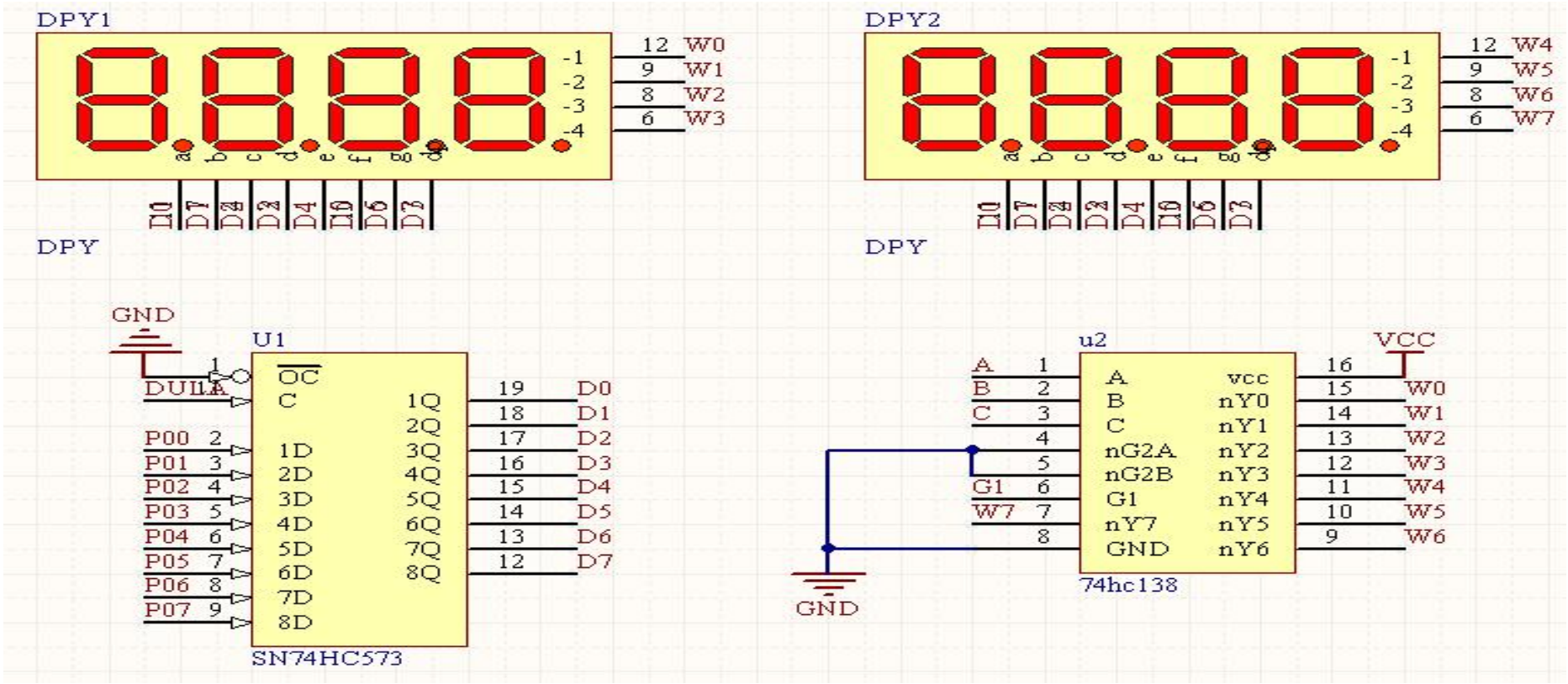
1、通过串口线完成在线下载

2、通过 USB 口线提供+5V 电源

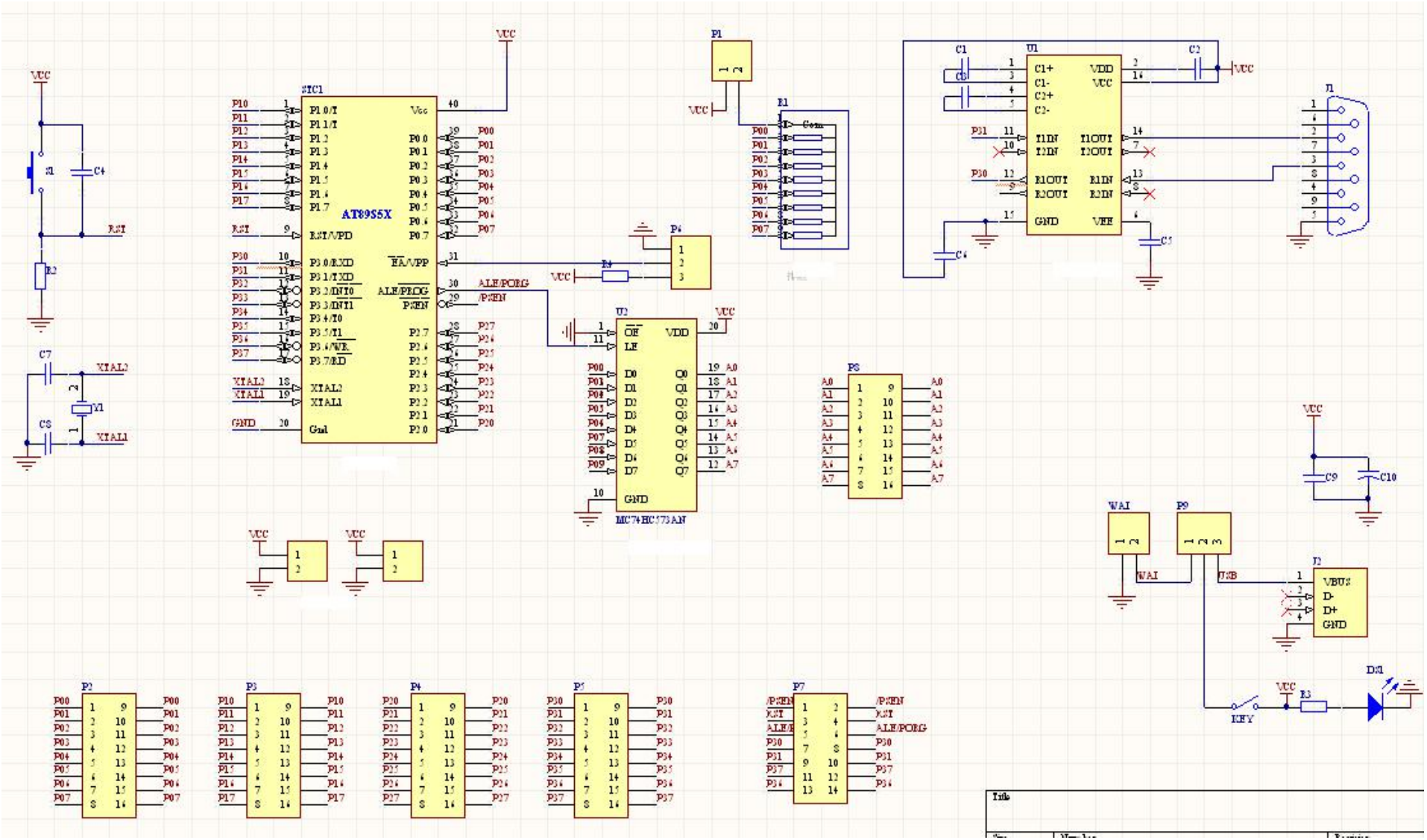
3、CPU 板上的 51 单片机 P0-P3 口全部引出。

4、键盘、显示板上有 8 位数码管，8 个指示灯、16 个按键

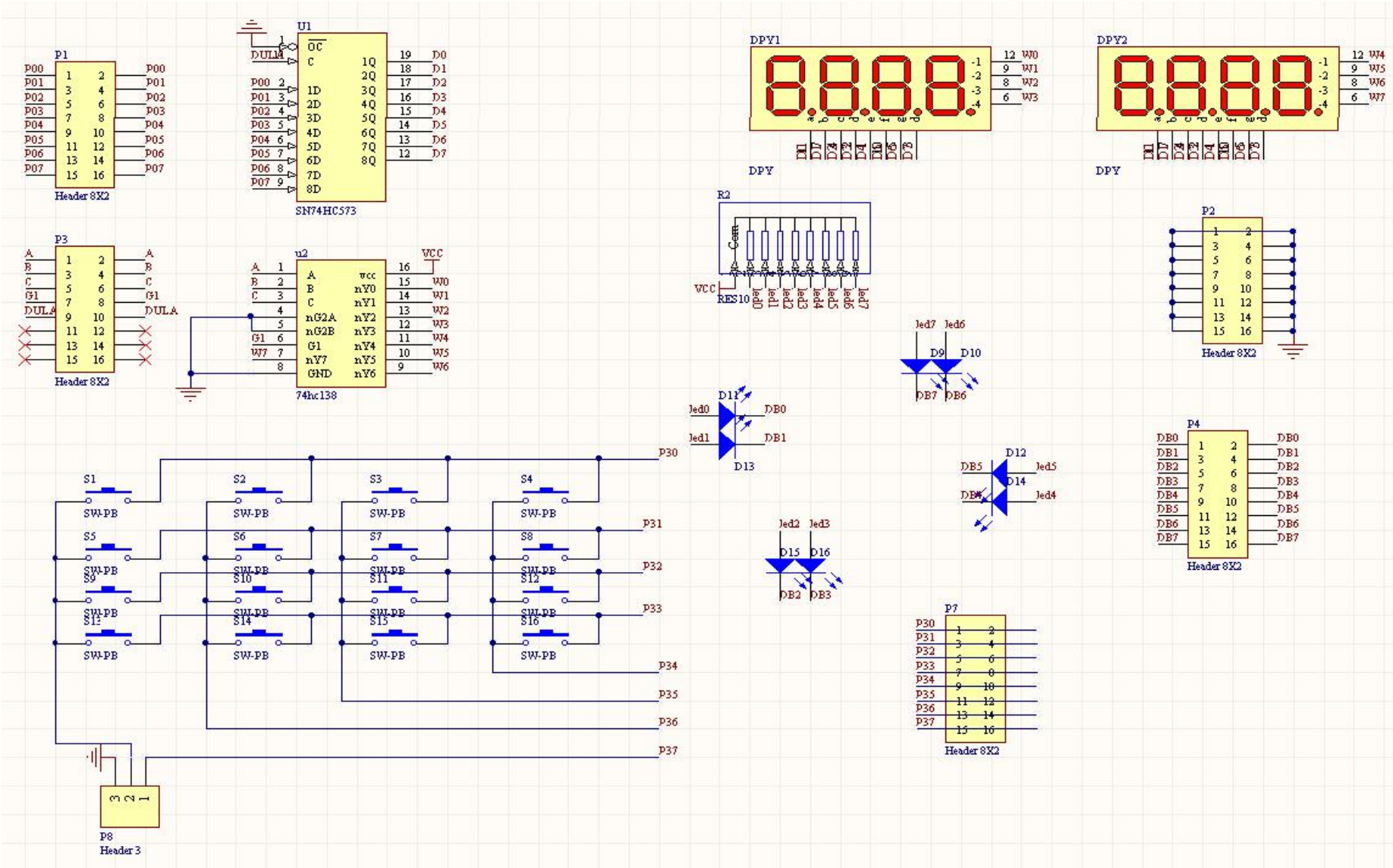
5、从下列数码管图可以看出：数据与段的关系为：数据 D7～D0 对应 dot g f e d c b a 段



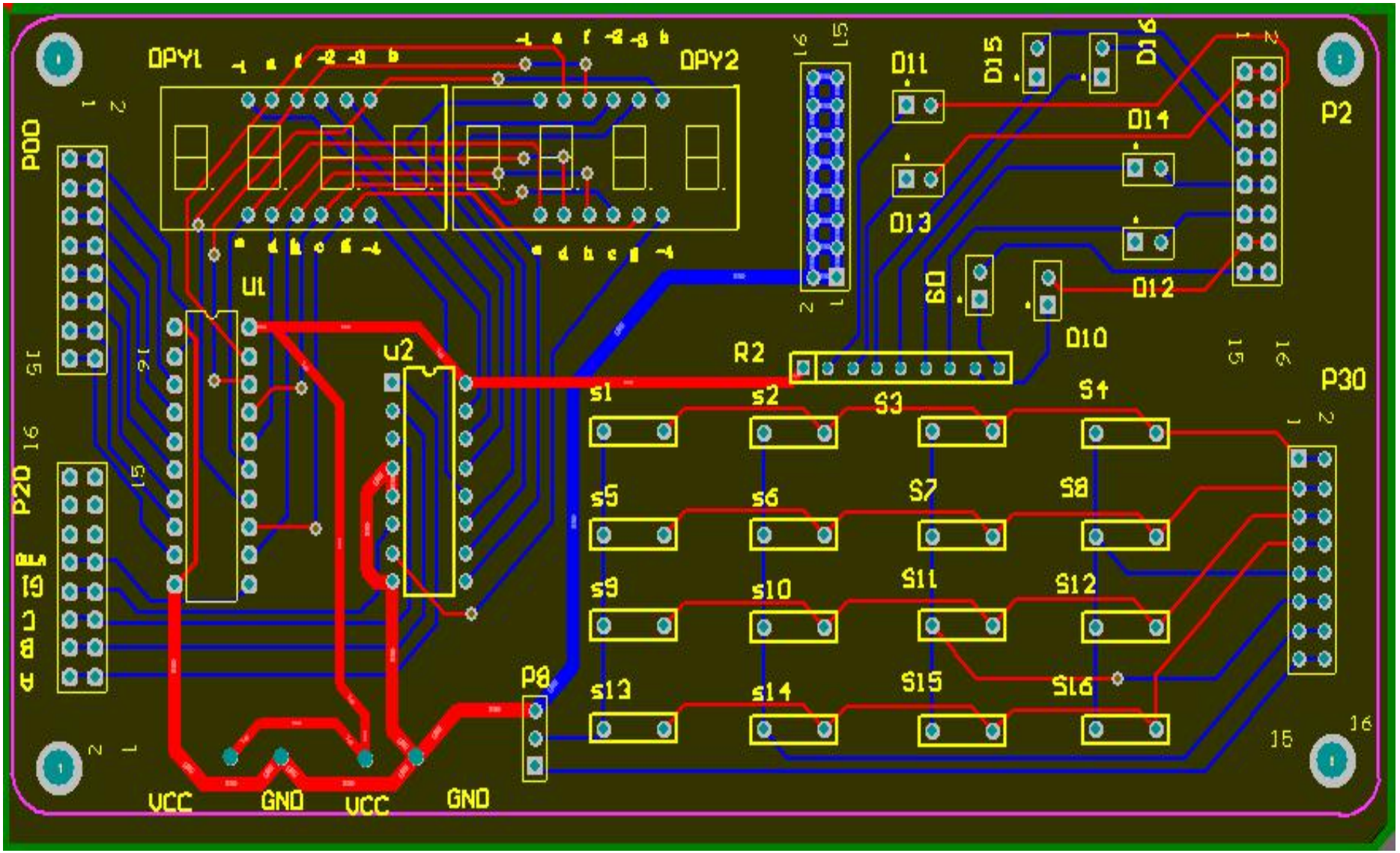
单片机系统主机板原理图



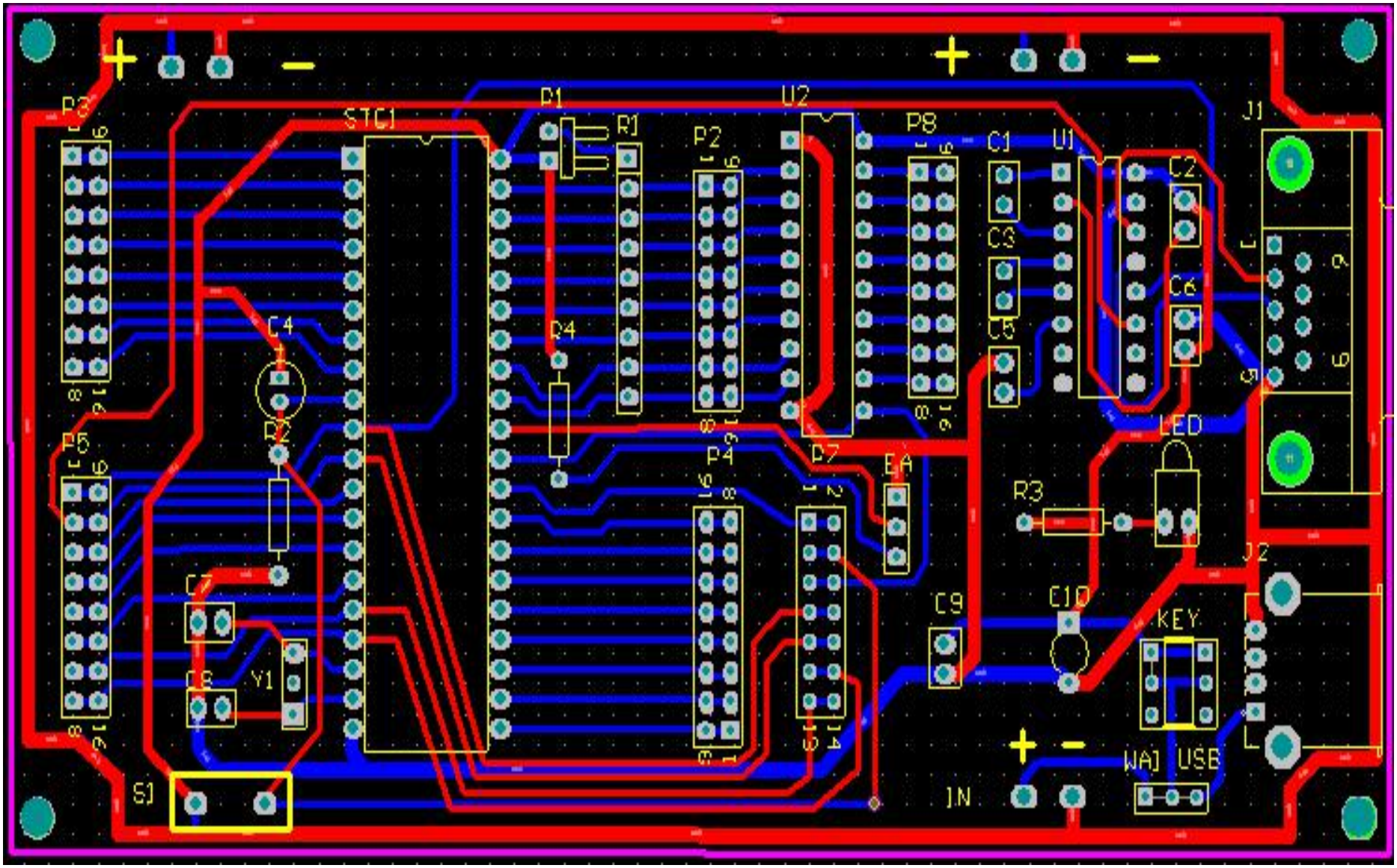
单片机系统键盘、显示板原理图



单片机系统键盘、显示印刷电路板图



单片机系统主机印刷电路板图



印刷电路板各图说明

一、 单片机系统主机电路板

1、十六针排电缆连接插座

共有五个（P2、P3、P4、P5、P8），横向两针短路，具体连接说明如下：

P2 插座：与单片机的 P0 口连接（上数 P0.0～P0.7），可作为普通的 I/O 口使用，也可作为 8 位数据总线（上数 D0～D7）；

P3 插座：与单片机的 P1 口连接（上数为 P1.0～P1.7），只能作为普通的 I/O 口使用；

P4 插座：与单片机的 P2 口连接（下数为 P2.0～P2.7），既可作为普通的 I/O 口，也可作为高 8 位地址总线（上数 A8～A15）；

P5 插座：与单片机的 P3 口连接（上数为 P3.0～P3.7），既可作为普通的 I/O 口，也可作为控制总线；

P8 插座：单片机的 P0 口的低 8 位地址信号经过 U2 芯片（74HC573 锁存器）输出到 P8 插座（上数为 A0～A7），锁存信号 ALE。

2、十四针排电缆连接插座

P7 插座：上数连接信号分别为/PSEN、RST、ALE、P3.0（RXD）、P3.1（TXD）、P3.7（/RD）、P3.6（/WR）

3、双针跳线座

P1 跳线座：短路连接时使排电阻 R1 与 5VDC 电源连接，给 P0 口加上拉电阻（P0 口为普通的 I/O 口时），作为数据总线时必须断开。

4、三针跳线座

P6 跳线座：上面 2 针短路连接时（/EA=0），使用单片机外部 ROM；下面 2 针短路连接时（/EA=1），使用单片机内部 ROM。

5、三针跳线座（按键 KEY 下面）

与 WAI 端连接时，使用外部 5VDC 电源；与 USB 端连接时，使用来自计算机 USB 接口的 5VDC 电源。

6、LED 灯

它是电路板加电指示灯。

7、按键 KEY

5VDC 电源加电开关。

8、按键 S1

单片机复位开关。

注意：以上各连接插座的信号可查看单片机系统主机板原理图了解。

二、 单片机系统键盘、显示电路板

1、十六针排电缆连接插座

共有五个（P00、P20、P2、P30 和八段数码管右侧未做标记的插座），横向两针短路，具体连接说明如下：

P00 插座：数码管的段选信号。上数第 1～8 对针分别经过 U1 芯片（74HC573 锁存器）与数码管的 a～f 段连接。八段数码管为共阴极的，即 a～f 段只有接高电平时才能点亮。

注意：U1 芯片的第 1 脚为输出使能，低电平有效，电路板上该引脚接地，即输出使能常有效。第 11 脚为锁存使能 LE，高电平写入，低电平锁存，LE 为 P20 的下数第 5 对针（DULA）。

P20 插座：它是数码管的位选信号，即选择哪个数码管。A、B、C、G1 是 U2 芯片（74HC138 译码器）的引脚，P20 的下数第 1 对针与 U2 第 1 脚 A 连接，下数第 2 对针与 U2 第 1 脚 B 连接，下数第 3 对针与 U2 第 1 脚 C 连接，下数第 4 对针与 U2 第 6 脚 G1 连接（高电平有效），U2 芯片的 4、5 脚接地（常有效）。下数第 5 对针（DULA）接 U1 芯片（74HC573 锁存器）的第 11 脚 LE，LE 是 74HC573 锁存器的锁存信号，高电平有效（即高电平写入）。其它 3 对针悬空。74HC138 的译码输出（/Y0）选通左数第 1 个数码管，/Y1 选通左数第 2 个数码管，…，/Y7 选通左数第 8 个数码管。

P2 插座：与 8 个 LED 灯连接，P2 插座输入低电平时对应的 LED 灯亮。上数第 1～8 对针分别 D13、D11、D15、D16、D14、D12、D10、D9 等 LED 灯。排电阻 R2 是这些 LED 灯的限流电阻，R2 两端分别接 LED 灯和 5VDC 电源。

注意：由于 P2 插座的上数第 1、2 两对针接反，导致 D13、D11LED 灯点亮的顺序是反的，可以借助软件改正。

P30 插座：4X4 键盘扫描信号。上数第 1～4 对针分别为行 1～4 的扫描信号，上数第 5～8 对针分别为列 4～1 的扫描信号。

八段数码管右侧未做标记的插座：所有针接地，用于测试排电缆。测试方法：此插座与本电路板上的 P2 插座对接，通电即可。如果排电缆是好的，则所有 LED 灯全亮。

2、三针跳线座 P8

它是左数第一列键选择硬跳线。上面 2 针（2、3）短路连接时，这列 4 个按键的一端接地，它们可作为接地按键使用（如外中断信号等）；下面 2 针（1、2）短路连接时，这一列的 4 个按键作为键盘阵列的第一列，它们的一端与 P30 插座的第 8 对针连接。

外中断 0 信号连接举例：如果使用外中断 0 进行外中断实验，则可以将键盘、显示印刷电路板上的 P30 插座与主机印刷电路板的 P5 插座对接，P5 插座上数第三对针是/INT0，而 P30 插座上数第三对针是对应第三行的四个按键，这样 S9 就是给/INT0 提供中断信号的按键。

STC-ISP 下载编程软件的使用方法

1. 首先，所编程序在 UV2 上编译无误，模拟运行无误。
2. 在 UV2 的界面上，点击“目标 1”左侧的图标，系统弹出“目标“目标 1”属性”菜单栏，再点击“输出”菜单，选择“E 生成 HEX 文件”。注意，此时一定要查看“N 执行文件名”菜单栏提供的具体文件名，作为下载依据。
3. 运行 STC-ISP 下载编程软件。
4. 查看左上角“MCU type”菜单栏，其芯片类型应与电路板上的类型一致。
5. 点击“OpenFile/打开文件”，选择“N 执行文件名”(.hex 文件)。
6. 再点击“DownLoad/下载”。注意：点击“DownLoad/下载”前，关闭电路板电源，待右下角菜单栏出现“Chinese:正在尝试与 MCU/单片机 握手连接 ...”或“仍在连接中，请给 MCU 上电...”字样时，接通电路板电源。
7. 按下主机电路板上的“复位”键，即可观察程序运行是否达到原设计要求。