第12章　AT89S51单片机的应用系统设计

12.1 单片机应用系统的设计步骤

单片机应用系统的设计，首先要经过深入细致的需求分析，周密而科学的方案论证才能使系统设计工作顺利完成。一个单片机应用系统设计，一般可分为4个阶段。

（1）明确任务和需求分析以及拟定设计方案阶段

明确系统要完成的任务十分重要，是设计工作的基础以及系统设计方案正确性的保证。

（2）硬件和软件设计阶段

根据拟定的方案，设计系统硬件电路。硬件设计的前提是必须能够完成系统的要求和保证可靠性。在硬件设计时，如能将硬件电路设计与软件设计结合起来考虑效果会更好。因为当有些问题在硬件电路中无法完成时，可直接由软件来完成（如某些软件滤波、校准功能等）；当软

件编写程序很麻烦的时候，通过稍稍改动硬件电路（或尽可能不改动）可能会使软件变得十分简单。另外在另一些要求系统实时性强、响应速度快的场合，则往往必须用硬件代替软件来完成某些功能。所以在硬件电路设计时，最好能够与软件的设计结合起来，统一考虑，合理地安排软、硬件的比例，使系统具有最佳的性能价格比。当硬件电路设计完成后，就可进行硬件电路板的绘制和焊接工作了。

接下来软件设计。正确编程方法就是根据需求分析，先绘制出软件的流程图，该环节十分重要。流程图绘制往往不能一次成功，需多次修改。

绘制可由简到繁逐步细化，先绘制系统大体上需要执行的程序模块，然后将这些模块按照要求组合在一起（如主程序、子程序以及中断服务子程序等），在大方向没问题后，再将每个模块细化，最后形成流

程图，程序编写速度就会很快，同时为后面的调试工作带来很多方便，如调试中某模块不正常，就可以通过流程图来查找问题的原因。

一定要克服不绘制流程图直接在计算机上编写程序的习惯。

设计者也可在上述软硬件设计完成后，先使用单片机软件仿真开发工具Proteus，来进行仿真设计。

用软件仿真开发工具Proteus设计的系统与用户样机在硬件上无任何联系，是一种完全用软件手段来对单片机硬件电路和软件进行设计、开发与仿真调试的开发工具。如果先在软件仿真工具的软环境下进行系统设计并调试通过，虽然还不能完全说明实际系统就完全通过，但至少在逻辑上是行得通的。

软件仿真通过后，再进行软硬件设计与实现，可大大减少设计上所走的弯路。这也是目前世界上流行的一种开发方法。

（3）硬件与软件联合调试阶段

下一步就是软硬件的联合调试。需通过硬件仿真开发工具来进行，具体的调试方法和过程，在本章后面介绍。

所有软件和硬件电路全部调试通过，并不意味系统设计成功，还需通过运行来调整系统的运行状态，例如系统中的A/D转换结果是否正确，如果不正确，是否要调零和调整基准电压等。

（4）资料与文件整理编制阶段

系统调试通过，就进入资料与文件整理编制阶段。

资料与文件包括：任务描述、设计的指导思想及设计方案论证、性能测定及现场试用报告与说明、使用指南、软件资料（流程图、子程序使用说明、地址分配、程序清单）、硬件资料（电原理图、元件布置图及接线图、接插件引脚图、线路板图、注意事项）。

文件不仅是设计工作的结果，而且是以后使用、维修以及进一步再设计的依据。因此，要精心编写，描述清楚，使数据及资料齐全。

12.2 单片机应用系统设计

介绍如何进行系统的设计。主要从硬件设计和软件设计两方面考虑。

12.2.1 硬件设计应考虑的问题

硬件设计时，应重点考虑以下问题。

1．尽可能采用功能强的芯片

（1）单片机选型。单片机的集成度越来越高，许多外围部件都已集成在芯片内，有的单片机本身就是一个系统，这可省去许多外围部件的扩展工作，使设计工作简化。

第1章已介绍较为流行的各种单片机，根据需求，选择合适机型。

例如，目前市场上较为流行的美国Cygnal公司的C8051F020 8位单片机，片内集成有8通道A/D、两路D/A、两路电压比较器，内置温度传感器、定时器、可编程数字交叉开关和64个通用I/O口、电源监测、看门狗、多种类型的串行总线（两个UART、SPI）等。用1片 C8051F020单片机，就构成一个应用系统。再如，如系统需要较大的I/O驱动能力和较强的抗干扰能力，可考虑选用AVR单片机。

（2）优先选片内有闪存的产品。例如，使用ATMEL公司的AT89S52/AT89S53/ AT89S54/ AT89S55系列产品，PHILIPS公司的89C58（内有32KB的Flash存储器）等，可省去扩展片外程序存储器的工作，减少芯片数量，缩小系统的体积。

（3）RAM容量的考虑。大多数单片机片内的RAM单元有限，当需增强软件数据处理功能时，往往觉得不足，这时可选用片内具有较大RAM容量的单片机，例如PIC18F452。

（4）对I/O端口留有余地。在样机研制出来现场试用时，往往会发现一些被忽视的问题，而这些问题是不能单靠软件措施来解决的。如有新的信号需要采集，就必须增加输入检测端；有些物理量需要控制，就必须增加输出端。如果在硬件设计之初就多设计留有一些I/O端口，这些问题就会迎刃而解。

（5）预留A/D和D/A通道。与上述I/O端口同样原因，留出一些A/D和D/A通道将来可能会解决大问题。

2．以软代硬

原则上，只要软件能做到且能满足性能要求，就不用硬件。硬件多不但增加成本，而且系统故障率也会提高。以软带硬的实质，是以时间换空间，软件执行过程需要消耗时间，因此带来的问题就是实时性下降。在实时性要求不高的场合，以软代硬是很合算的。

3．工艺设计

包括机箱、面板、配线、接插件等。须考虑到安装、调试、维修方便。另外，硬件抗干扰措施（将在本章后面介绍）也须在硬件设计时一并考虑进去。

12.2.2 典型的单片机应用系统

典型单片机应用系统框图如图12-1所示。

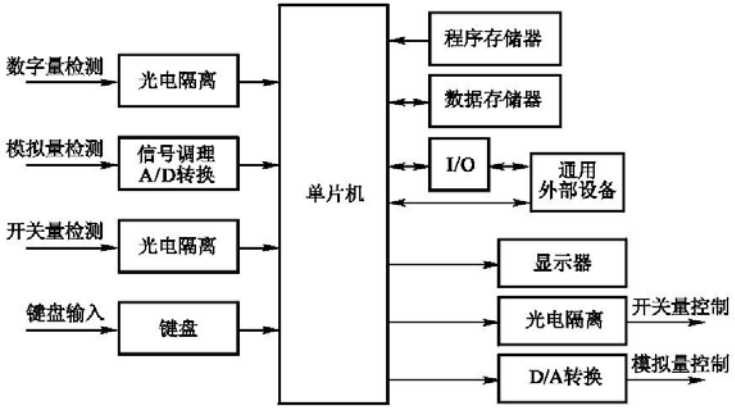


图12-1 单片机典型应用系统框图

主要由单片机基本部分、输入部分和输出部分组成。

（1）单片机基本部分

由单片机及其扩展的外设及芯片，如键盘、显示器、打印机、数据存储器、程序存储器、数字I/O等组成。

（2）输入部分

“测”的部分，被“测”的信号类型有：数字量、模拟量和开关量。模拟量输入检测的主要包括信号调理电路以及A/D转换器。A/D转换器中又包括多路切换、采样保持、A/D转换电路，目前都集成在A/D转换器芯片中，或直接集成在单片机片内。

连接传感器与A/D转换器之间的桥梁是信号调理电路，传感器输出的模拟信号要经信号调理电路对信号进行放大、滤波、隔离、量程调整等，变换成适合A/D转换的电压信号。信号放大通常由单片式仪表放大器承担。仪表放大器对信号进行放大比普通运算放大器具有更优异的性能。如何根据不同的传感器正确地选择仪表放大器来进行信号调理电路的设计，请读者参阅有关资料和文献。

（3）输出部分

是应用系统“控”的部分，包括数字量、开关量控制信号的输出和模拟量控制信号（常用于伺服控制）的输出。

12.2.3 系统设计中的总线驱动

一个AT89S51单片机应用系统有时往往是多芯片系统，如何实现AT89S51单片机对多片芯片的驱动。

在AT89S51单片机扩展多片芯片时，要注意AT89S51单片机4个并行双向口的P0~P3口的驱动能力。下面首先讨论这个问题。

AT89S51单片机的P0、P2口通常作为总线端口，当系统扩展的芯片较多时，可能造成负载过重，致使驱动能力不够，系统不能可靠地工作，所以通常要附加总线驱动器或其他驱动电路。因此在多芯片应用系统设计中首先要估计总线的负载情况，以确定是否需要对总线的驱动能力进行扩展。

图12-2为AT89S51单片机总线驱动扩展原理图。P2口需要单向驱动，常见的单向总线驱动器为74LS244。图12-3为74LS244引脚图和逻辑图。8个三态驱动器分成两组，分别由1G*和2G*控制。

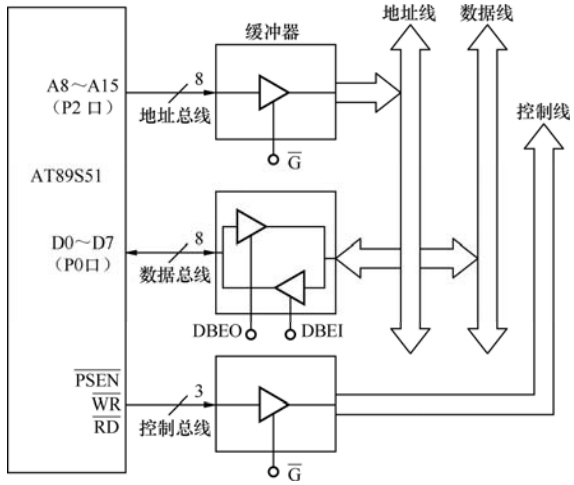
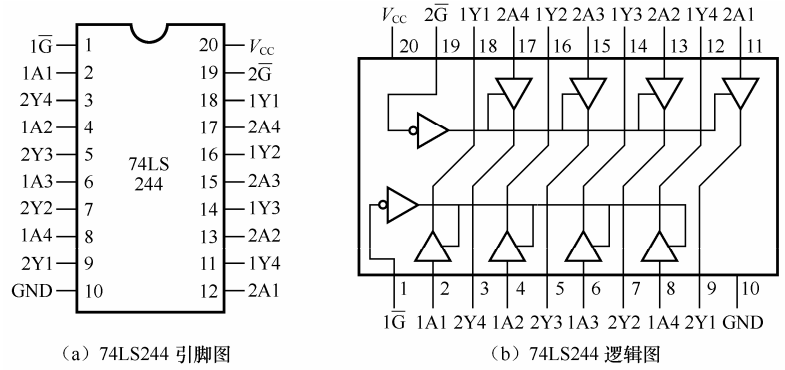


图12-2 AT89S51单片机总线驱动扩展原理图

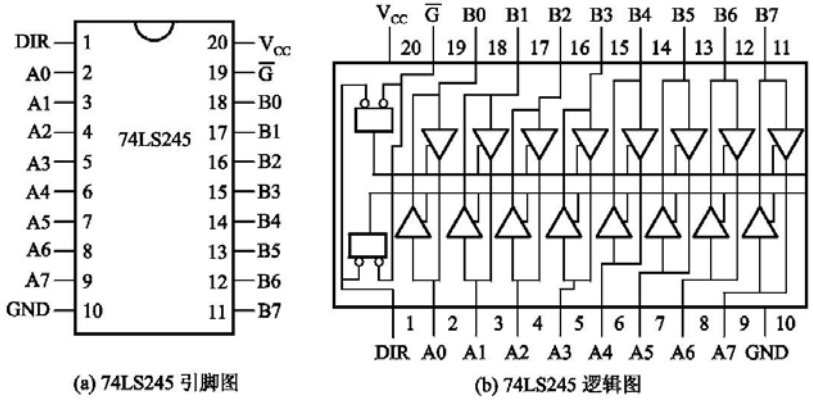
P0口作为数据总线，由于是双向传输，其驱动器应为双向驱动、三态输出，并由两个控制端来控制数据传送方向。如图12-3所示，数据输出允许控制端DBEO有效时，数据总线输入为高阻态，输出为开通状态；数据输入允许控制端DBEI有效时，则状态与上相反。常见的双向驱动器为74LS245，图12-4为其引脚和逻辑图。

16个三态门中每两个三态门组成一路双向驱动。驱动方向由、DIR两个控制端控制，控制端控制驱动器有效或高阻态，在控制端有效（=0）时，DIR控制端控制驱动器的驱动方向，DIR=0时驱动方向为从B至A，DIR=1时则相反。图12-5所示为AT89S51单片机应用系统总线驱动扩展电路图。P0口的双向驱动采用74LS245，如图12-5（a）所示；P2口的单向驱动器采用74LS244，如图12-5（b）所示。



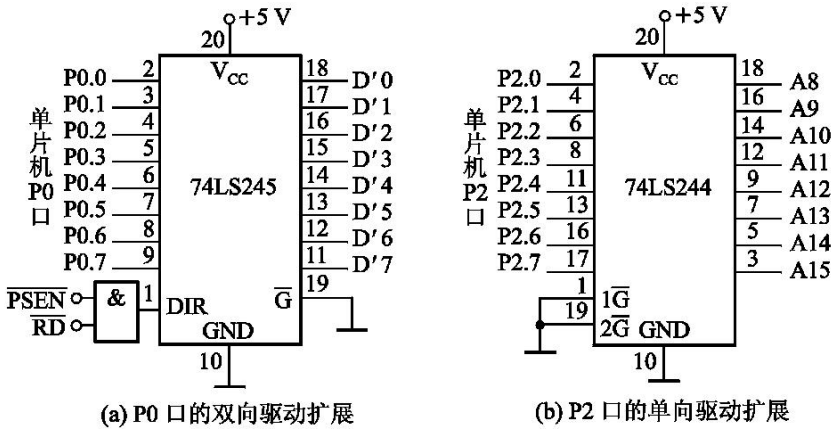
（a）74LS244 引脚图　　（b）74LS244 逻辑图

图12-3　单向驱动器74LS244引脚图和逻辑图



(a) 74LS245 引脚图　　(b) 74LS245 逻辑图

图12-4　74LS245的引脚图和逻辑图



(a) P0 口的双向驱动扩展　　(b) P2 口的单向驱动扩展

图12-5　AT89S51单片机应用系统中的总线驱动扩展电路图

P0口双向驱动器74LS245的 G* 接地，保证芯片一直处于工作状态，而输入/输出的方向控制由单片机的数据存储器的“读”控制引脚RD*和程序存储器的取指控制引脚PSEN*通过与门控制DIR引脚实现。无论是“读”数据存储器中数据（ RD*有效）还是从程序存储器中取指令（ PSEN*有效），都能保证对P0口的输入驱动；

除此以外的时间（ RD*及 PSEN*均无效），保证对P0口的输出驱动。对于P2口，因为只用作单向的地址输出，故74LS244的驱动门控制端1G* 、2G*接地。

12.2.4 AT89S51单片机的最小应用系统

AT89S51内部有4KB闪存，本身就是一个数字量输入/输出的最小应用系统。

在构建AT89S51单片机最小应用系统时，AT89S51单片机需要外接时钟电路和复位电路即可，如图12-6所示。

注意，本最小应用系统只能作为小型的数字量的测控单元。

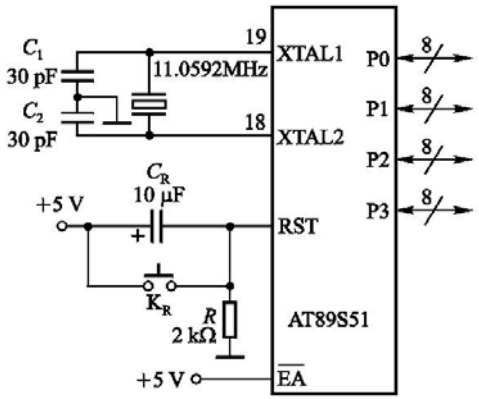


图12-6　AT89S51单片机构成的最小应用系统

12.3 单片机应用系统的仿真开发与调试

当用户样机完成硬件和软件设计，全部元器件安装完毕后，在用户样机的程序存储器中放入编写好的应用程序，系统即可运行。但应用程序运行一次性成功几乎是不可能的，多少会存在一些软件、硬件上的错误，需借助单片机的仿真开发工具进行调试，发现错误并加以改正。

AT89S51只是一个芯片，既没有键盘，又没有CRT、LED显示器，无法进行软件的开发（如编辑、汇编、调试程序等），必须借助某种开发工具（也称为仿真开发系统）所提供的开发手段。一般来说，仿真开发工具应具有如下最基本功能。

（1）用户样机程序的输入与修改。

（2）程序的运行、调试（单步运行、设置断点运行）、排错、状态查询等功能。

（3）用户样机硬件电路的诊断与检查。

（4）有较全的开发软件。用户可用汇编语言或C语言编制应用程序；由开发系统编译连接生成目标文件、可执行文件。配有反汇编软件，能将目标程序转换成汇编语言程序；有丰富的子程序可供用户选择调用。

（5）将调试正确的程序写入到程序存储器中。

下面首先介绍常用的仿真开发工具。

1. 仿真开发系统简介

通用机仿真开发系统是目前设计者使用最多的一类开发装置。这是一种通过PC机的并行口、串行口或USB口，外加在线仿真器的仿真开发系统，如图12-7。

在线仿真器一侧与PC机的串行口、并行口、或USB口相连。在线仿真器另一侧的仿真插头插入到用户样机的单片机插座上，来对样机上的单片机进行“仿真”。从仿真插头向在线仿真器看去，看到的就是一个“单片机”。 这个“单片机”是用来“代替”用户样机上的单片机。但是这个“单片机”片内程序的运行是由PC机上的软件控制的。由于在线仿真器有PC机及其仿真开发软件的强大支持，可以在PC机的屏幕上观察用户程序的运行情况，可以采用单步、设断点等手段逐条跟踪用户程序并进行修改、调试和查找软、硬件故障。

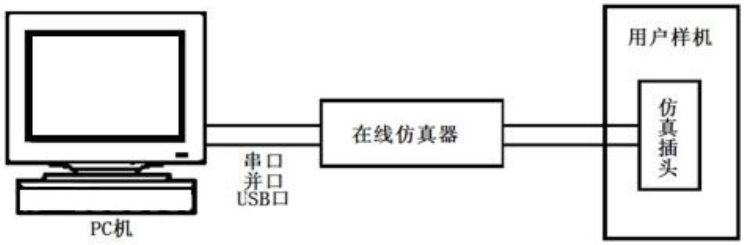


图12-7　通用机仿真开发系统

在调试用户程序时，仿真器的仿真插头必须插入用户样机空出的单片机插座中。当仿真开发系统与PC机联机后，用户可利用PC机上的仿真开发软件，在PC机上编辑、修改源程序，然后通过交叉汇编软件将其汇编成机器代码，传送到在线仿真器中的仿真RAM中。

这时用户可用单步、断点、跟踪、全速等方式运行用户程序，系统状态实时地显示在屏幕上。

待程序调试通过后，再使用仿真开发系统提供的编程器或使用专用编程器，把调试完毕的程序写入到单片机内的Flash存储器中或外扩的EPROM中。此类仿真开发系统是目前最流行的仿真开发工具。配置不同的在线仿真器，可仿真开发各种单片机。

但是随着ISP技术的普及，对于AT89S5x单片机也可不使用在线仿真器以及编程器，用户只需要在PC上修改程序，然后将修改的程序直接写入用户样机的单片机的Flash存储器中。

2. 软件仿真开发工具Proteus

软件仿真开发工具Proteus已在第4章中做了详细介绍，使用Proteus软件进行单片机系统的虚拟设计与仿真不需硬件在线仿真器，也不需要用户硬件样机，直接就可以在PC机上进行。调试完毕的软件可以将机器代码固化，一般能直接投入运行。

但Proteus是软件模拟器是使用纯软件来对用户系统仿真，不能进行用户样机硬件部分的诊断与实时在线仿真。因此在系统的开发中，一般是先用Proteus仿真软件设计出系统的硬件电路，编写程序，然后在Proteus环境下仿真调试通过。然后依照仿真的结果，完成实际的硬件设计。再将仿真通过的程序烧录用户样机的Flash存储器中，观察运行结果，如果有问题，再连接硬件仿真器去分析、调试。

3. 用户样机的源程序调试

下面介绍如何使用仿真开发工具进行汇编语言源程序编写、调试以及与用户样机硬件联调工作。

用户源程序调试过程如图12-8所示，分以下4个步骤。

（1）输入用户源程序。用户使用编辑软件WS，按照汇编语言源程序要求的格式、语法规定，把源程序输入到PC机中，并保存在磁盘上。

（2）在PC机上，利用汇编程序对用户源程序进行汇编，直至语法错误全部纠正为止。如无语法错误，则进入下一个步骤。

（3）动态在线调试。这一步对用户的源程序进行调试。上述的步骤（1）、步骤（2）是一个纯粹的软件运行过程，这一步，必须要有在线仿真器配合，才能对用户源程序进行调试。用户程序中分为与用户样机硬件无关以及与用户样机紧密相关的程序。

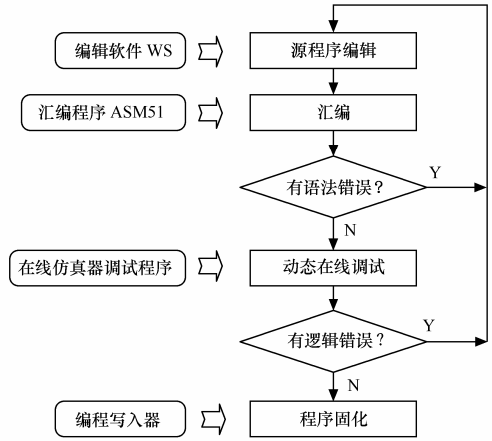


图12-8　用户样机软件设计、调试的过程

对于与用户样机硬件无关的程序，如计算程序，虽然没有语法错误，但可能存在逻辑错误，使计算结果不正确，此时必须借助于在线仿真器的动态在线调试手段，如单步运行、设置断点等，发现逻辑错误，然后返回到步骤（1）修改，直至逻辑错误纠正为止。

对于与用户样机硬件紧密相关的程序段（如接口驱动程序），一定要先把在线仿真器的仿真插头插入用户样机的单片机插座中（见图12-7），进行在线仿真调试，利用仿真开发系统提供单步、设置断点等调试手段来进行系统的调试。

部分程序段运行不正常，可能是软件逻辑上有问题，也可能是硬件有故障，必须先通过在线仿真调试程序提供的调试手段，把硬件故障排除以后，再与硬件配合，对用户程序进行动态在线调试。对于软件的逻辑错误，则返回到第一步进行修改，直至逻辑错误消除为止。在调试这类程序时，硬件调试与软件调试是不能完全分开的。许多硬件错误是通过软件的调试而发现和纠正的。

（4）将调试完毕的用户程序通过编程写入器（也称烧写器），固化在程序存储器中。

4．用户样机的硬件调试

用户样机全部焊接完毕，就可对样机的硬件进行调试。首先进行静态调试，目的是排除明显的硬件故障。

（1）静态调试

分两步：

第一步在样机加电之前，根据硬件设计图，用万用表等工具，仔细检查样机线路是否连接正确，并核对元器件型号、规格和安装是否符合要求，应特别注意电源系统的检查，防止电源的短路和极性错误，并重点检查系统总线（地址总线、数据总线、控制总线）是否存在相互之间短路或与其他信号线短路。

第二步加电后检查各芯片插座上有关引脚的电位，测量各点电平是否正常，尤其应注意AT89S51插座的各点电位，若有高压，与在线仿真器联机调试时，将会损坏在线仿真器。

具体步骤如下。

- 电源检查。当用户样机板连接或焊接完成之后，先不插主要元器件，通上电源。通常用+5V直流电源（这是TTL电源），用万用表电压挡测试各元器件插座上相应电源引脚电压数值是否正确，极性是否符合。如有错误，要及时检查、排除，以使每个电源引脚的数值都符合要求。

- **各元器件电源检查。断开电源，按正确的元器件方向插上元器件。最好是分别插入，分别通电，逐一检查每个元器件上的电源是否正确，直到最后全部插上元器件。通电后，每个元器件上电源值应正确无误。**
- **检查相应芯片的逻辑关系。通常采用静态电平检查法，即在一个芯片信号输入端加入一个相应电平，检查输出电平是否正确。单片机系统大都是数字逻辑电路，使用电平检查法可首先检查出逻辑设计是否正确，选用的元器件是否符合要求，逻辑关系是否匹配，元器件连接关系是否符合要求等。**

（2）用户样机的在线仿真与动态调试

在静态调试中，对用户样机硬件进行初步调试，只能排除一些明显的静态故障。

用户样机中的硬件故障（如各个部件内部存在的故障和部件之间连接的逻辑错误）主要靠联机在线仿真来排除的。

在断电情况下，除单片机外，插上所有的元器件，并把在线仿真器的仿真插头插入样机上AT89S51单片机的插座（见图12-7），然后分别打开用户样机和仿真器电源后便可开始联机在线仿真调试。

前面已经介绍，硬件调试和软件调试是不能完全分开的，许多硬件错误是在软件调试中发现和被纠正的。所以，在之前介绍的有关用户样机软件调试的第（3）步的动态在线调试中，即包括联机仿真、硬件在线动态调试以及硬件故障的排除。

12.4 单片机应用设计案例

本节介绍各种常用的单片机测控应用设计案例，通过这些案例使读者了解单片机系统的各种常见的应用设计。

12.4.1 单片机控制步进电机的设计

步进电机是将脉冲信号转变为角位移或线位移的开环控制元件。

非超载的情况下，电机转速、停止位置只取决于脉冲信号的频率和脉冲数，而不受负载变化的影响，给电机加一脉冲信号，电机则转过一个步距角。因而步进电机只有周期性误差而无累积误差，在速度、位置等控制领域有较为广泛的应用。

1. 控制步进电机的工作原理

驱动步进电机由单片机通过对每组线圈中的电流的顺序切换来使电机作步进式旋转，切换是单片机输出脉冲信号来实现。

调节脉冲信号频率就可改变步进电机转速；改变各相脉冲先后顺序，就可改变电机旋转方向。

步进电机驱动可采用双四拍（AB→BC→CD→DA→AB）方式，也可采用单四拍（A→B→C→D→A）方式。为使步进电机旋转平稳，还可采用单、双八拍方式（A→AB→B→BC→C→CD→D→DA→A）。

各种工作方式时序见图12-9。

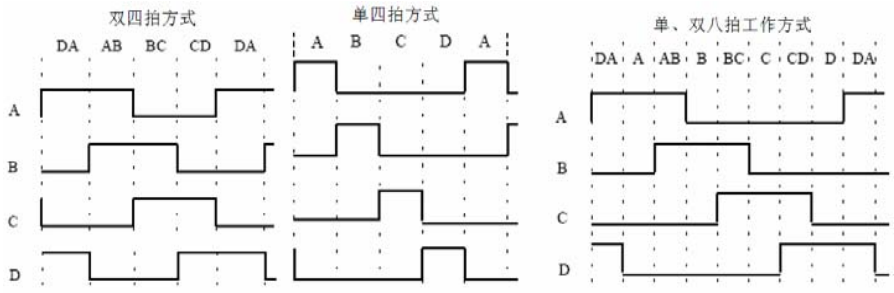


图12-9　各种工作方式时序图

图12-9脉冲信号是高电平有效，但实际控制时公共端是接在Vcc上，所以实际控制脉冲是低电平有效。

2. 电路设计与编程

【例12-1】单片机对步进电机控制的原理电路见图12-10。编写程序，用四路I/O口输出实现环形脉冲分配，控制步进电机按固定方向连续转动。同时，通过“正转”和“反转”两个按键来控制电机的正转与反转。按下“正转”按键，步进电机正转；按下“反转”按键，步进电机反转；松开按键，电机停止转动。

ULN2003是高耐压、大电流达林顿阵列系列产品，7个NPN达林顿管组成。多用于单片机、智能仪表、PLC等控制电路中。

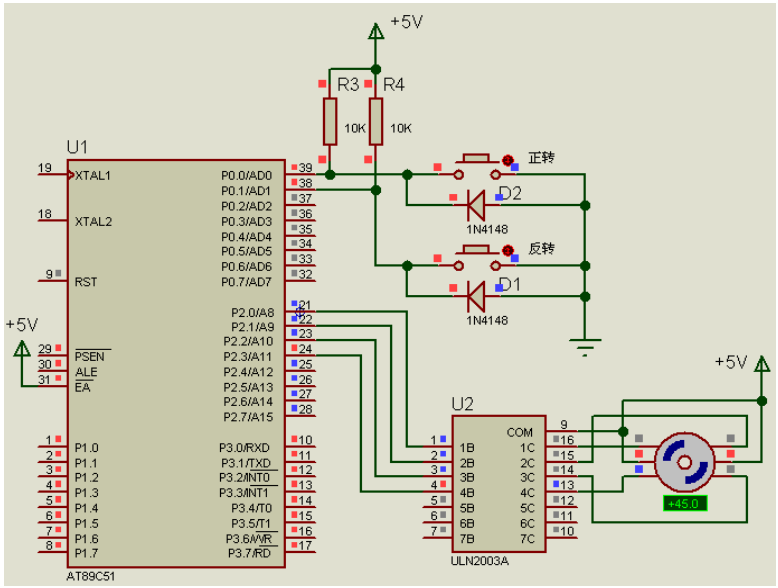


图12-10　单片机控制步进电机接口电路

在5V电压下能与TTL和CMOS电路直接相连，可直接驱动继电器等负载。具有电流增益高、工作电压高、温度范围宽、带负载能力强等特点。输入5V的TTL电平，输出可达500mA/50V。适于各类高速大功率驱动的系统。

参考程序：

```
#include "reg51.h"
#define uchar unsigned char
#define uint unsigned int
#define out P2
sbit pos=P0^0;                    //定义检测正转控制位P0.0
sbit neg=P0^1;                    //定义检测反转控制位P0.1
void delayms(uint);
```

```
uchar code turn[]={0x02,0x06,0x04,0x0c,0x08,0x09,0x01,0x03};
                                        //步进脉冲数组
void main(void)
{
        uchar i;
        out=0x03;
        while(1)
        {
                if(!pos)      //如果正转按键按下
                {
                        i=i< 8?i+1: 0;  //如果i<8，则i= i+1;否则i=0
                        out=turn[i];
                        delayms(50);
                }
```



```
else if(!neg)
                {
                        i = i > 0 ? i-1: 7;
                        out=turn[i];
                        delayms(50);
                }
        }
}

void delayms(uint j)                                    //函数功能：延时
{
        uchar i;
        for(;j>0;j--)
        {
                i=250;
                while(--i);
                i=249;
                while(--i);
        }
}
```


12.4.2 单片机控制直流电机

直流电机多用在无交流电源、方便移动场合，具有低速大力矩等特点。如何用单片机控制直流电机。

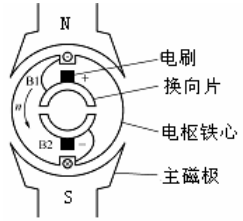
1. 控制直流电机的工作原理

对直流电机可精确控制其旋转速度或转矩，通过两个磁场相互作用产生旋转。结构见图12-11（a），定子装设一对直流励磁的静止主磁极N和S，在转子上装设电枢铁心。定子与转子间有一气隙。在电枢铁心上放置了由两根导体连成的电枢线圈，线圈首端和末端分别连到两个圆弧形铜片上，此铜片称为换向片。由换向片构成的整体称为换向器。

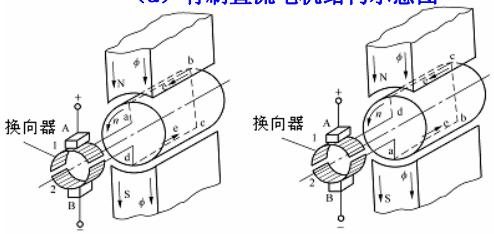
换向器固定在转轴上，换向片与转轴间互相绝缘。在换向片上放置一对固定不动的电刷B1和B2，当电枢旋转时，电枢线圈通过换向片和电刷与外电路接通。

定子通过永磁体或受激励电磁铁产生一固定磁场，由于转子由一系列电磁体构成，当电流通过其中一个绕组时会产生一个磁场。

对有刷直流电机，转子上换向器和定子电刷在电机旋转时为每个绕组供给电能。通电转子绕组与定子磁体有相反极性，因而相互吸引，使转子转动至与定子磁场对准的位置。当转子到达对准位置时，电刷通过换向器为下一组绕组供电，从而使转子维持旋转运动，见图12-11（b）。



（a）有刷直流电机结构示意图



（i）导体ad处于N极下　　（ii）导体ad处于S极下

（b）有刷直流电机工作示意图

图12-11　直流电机工作示意图

直流电机转速与施加电压成正比，转矩与电流成正比。由于必须在工作期间改变直流电机的速度，直流电机控制是一较困难问题。直流电机高效运行的常见方法是施加一个 PWM（脉宽调制）脉冲波，其占空比对应于所需速度。电机起到了一个低通滤波器作用，PWM信号相对容易产生，这种驱动方式使用更为广泛。

2. 电路设计与编程

【例12-2】原理电路见图12-12。使用单片机两个I/O脚控制直流电机转速和旋转方向。其中P3.7脚输出PWM信号控制直流电机转速；P3.6脚控制直流电机旋转方向。

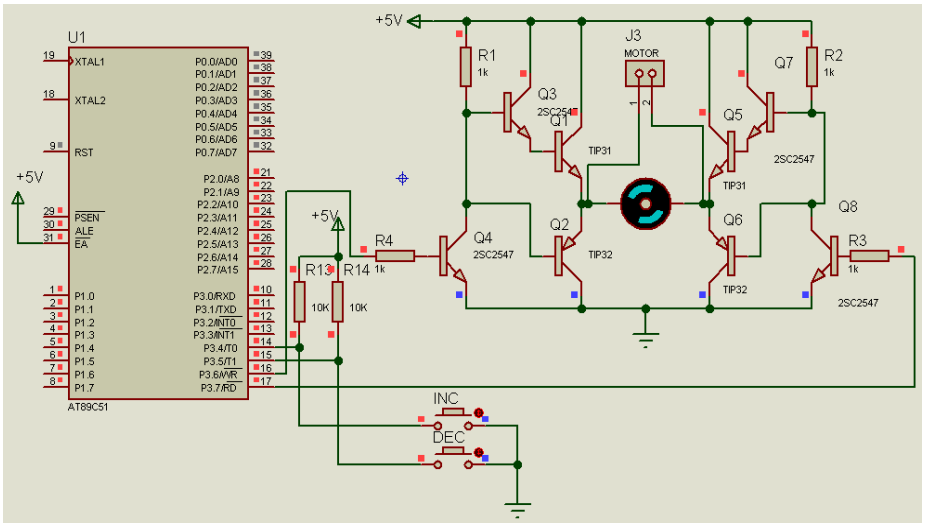


图12-12　单片机控制直流电机的接口电路

当P3.6=1时，P3.7发送PWM波，直流电机正转。且可通过“INC”和“DEC”两个按键来增大和减少直流电机转速。反之，P3.6=0时，P3.7发送PWM信号，直流电机反转。

因此，增大和减小电机转速，实际上是通过按下 “INC”或“DEC”按键来改变输出PWM信号占空比，控制直流电机转速。图12-12中驱动电路使用了NPN低频、低噪声小功率达林顿管 2SC2547。

参考程序如下：

```
#include "reg51.h"
#include "intrins.h"
#define uchar unsigned char
#define uint unsigned int
sbit INC=P3^4;
sbit DEC=P3^5;
sbit DIR=P3^6;
sbit PWM=P3^7;
void delay(uint);
int PWM= 900;

void main(void)
{
        DIR=1;
        while(1)
        {
        if(!INC)
        PWM=PWM>0 ? PWM-1 : 0;          //如果PWM>0，则PWM=PWM-1；否则PWM=0
```



```
if(!DEC)
        PWM=PWM<1000?PWM+1:1000; //如PWM<1000，则PWM=PWM+1；否
                                 //则PWM=1000
        PWM=1;                          //产生PWM的信号高电平
        delay(PWM);                     //延时
        PWM=0;                          //产生PWM的信号低电平
        delay(1000-PWM);                //延时
   }
}
void delay(uint j)
{
   for(;j>0;j--)
   {
        _nop_();
   }
}
```


12.4.3 频率计的制作

1. 工作原理

利用单片机定时器/计数器可实现信号频率测量。频率测量有测频法和测周法两种。测频法利用外部电平变化引发的外部中断，测算1s内出现的次数，从而实现对频率测量；测周法是通过测算某两次电平变化引发的中断间的时间，再求倒数，从而实现对频率测定。总之，测频法是直接根据定义来测定频率，测周法是通过测定周期间接测定频率。理论上，测频法适于较高频率测量，测周法适于较低频率测量。本例采用测频法。

2. 电路设计与软件编程

【例12-3】设计以单片机为核心的频率测量装置，测量加在P3.4脚上数字时钟信号频率，并在外部扩展的6位LED数码管上显示测量频率值。原理电路与仿真见图12-13。

本频率计测量的信号由数字时钟源“DCLOCK”产生，在电路中添加数字时钟源的具体操作与设置见第4章。手动改变被测时钟信号源频率，观察是否与LED数码管上显示的测量结果相同。

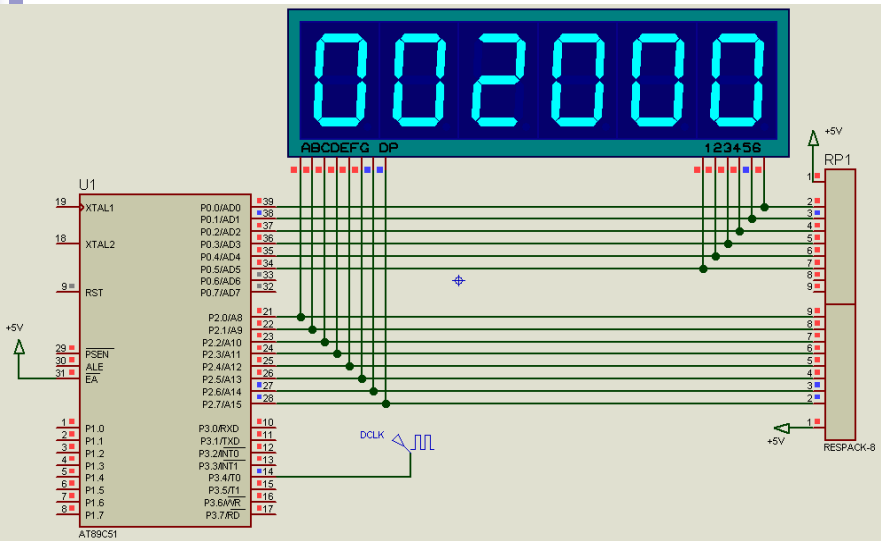


图12-13　频率计原理电路与仿真

参考程序如下。

```
#include<reg51.h>
sfr16 DPTR=0x82;                                    //定义寄存器DPTR
unsigned char cnt_t0,cnt_t1,qian,bai,shi,ge,bb,wan,shiwan;
unsigned long freq;                                 //定义频率
unsigned char code table[]={0x3f,0x06,0x5b,0x4f, 0x66,0x6d, 0x7d,
              0x07, 0x7f,0x6f,0x77,0x7c, 0x39,0x5e,0x79,0x71};
                                                    //共阴数码管段码表
void  delay_1ms(unsigned int z) //函数功能：延时约1ms
{
        unsigned char i,j;
        for(i=0;i<z;i++)
        for(j=0;j<110;j++);
}
```



```
void  init()            //函数功能：定时器/计数器及中断系统初始化
{
        freq=0;                 //频率赋初值
        cnt_t1=0;
        cnt_t0=0;
        IE=0x8a;
        //开中断，T0，T1中断
        TMOD=0x15;              //T0为定时器方式1，T1为计数器于方式1
        TH1=0x3c;               //T1定时50ms
        TL1=0xb0;
        TR1=1;                  //开启定时器T1
        TH0=0;                  //T0清0
        TL0=0;
        TR0=1;                  //开启定时器T0
}
void  display(unsigned long freq_num)  //函数功能：驱动数码管显示
 {
    shiwan=freq_num%1000000/100000;
    wan=freq_num%100000/10000;
```



```
        qian=freq_num%10000/ 1000;   //显示千位
        bai=freq_num%1000/100;       //显示百位
        shi=freq_num%100/10;         //显示十位
        ge=freq_num%10;              //显示个位
        P0=0xdf;
              //P0口是位选
        P2=table[shiwan];            //显示十万位
        delay_1ms(5);
        P0=0xef;
        P2=table[wan];               //显示万位
        delay_1ms(3);
        P0=0xf7;
        P2=table[qian];              //显示千位
        delay_1ms(3);
```

```
        P0=0xfb;
        P2=table[bai];                          //显示百位
        delay_1ms(3);
        P0=0xfd;
        P2=table[shi];                          //显示十位
        delay_1ms(3);
        P0=0xfe;
        P2=table[ge];                           //显示个位
        delay_1ms(3);
}
void  main()                                    //主函数
{
        P0=0xff;                                //初始化P0口
        init();                                 //计数器初始化
        while(1)
        {
                if(cnt_t1==19)                  //定时1s
                {
```



```
                cnt_t1=0;                       //定时完成后清0
                TR1=0;                          //关闭T1定时器，定时1S完成
                delay_1ms(141);                 //延时较正误差,通过实验获得
                TR0=0;                          //关闭T0
                DPL=TH00;                       //利用DPTR读入其值
                DPH=TH0;
                freq=cnt_t0*65535;
                freq=freq+DPTR;                 //计数值放入变量
        }

 display(freq);                                 //调用显示函数
  }

}
void  t1_func()  interrupt 3                    //定时器T1的中断函数
{
        TH1=0x3c;
        TL1=0xb0;
        cnt_t1++;
}
```



```
void  t0_func()  interrupt 1                    //定时器T0的中断函数
{
        cnt_t0++;
}
```


12.4.4 电话拨号的模拟

1. 模拟电话拨号的设计要求

设计模拟电话拨号时的状况，把模拟电话键盘拨出的某一电话号码，显示在LCD显示屏上。电话键盘除了0～9的10个数字键外，还有“*”键用于实现删除功能，即删除一位最后输入的号码；“#”键用于清除显示屏上所有的数字显示。此外还要求每按下一个键，发出声响，表示按下该键。还有LCD显示器，显示所拨的电话号码。

2. 电路设计与编程

【例12-6】设计一模拟电话拨号时的电话键盘及显示装置，把电话键盘拨出的电话号码及其他信息，显示在LCD显示屏上。电话键盘共12个键，除了0～9的10个数字键外，还有“*”键用于删除最后输入的1位号码的功

能；“#”键用于清除显示屏上所有的数字显示。此外还要求每按下一个键，蜂鸣器要发出声响，以表示按下该键。显示信息共2行，第1行为设计者信息，第2行显示所拨的电话号码。

本例的电话拨号键盘采用4×3矩阵键盘，共12个键。拨号号码显示采用LCD 1602 液晶显示模块。因此涉及了单片机与4×3矩阵式键盘以及与16×2的液晶显示屏的接口设计，还有各种驱动程序的编制。液晶显示屏采用LCD1602 (即Proteus中的LM016L)。

本设计原理电路及仿真见图12-14 。

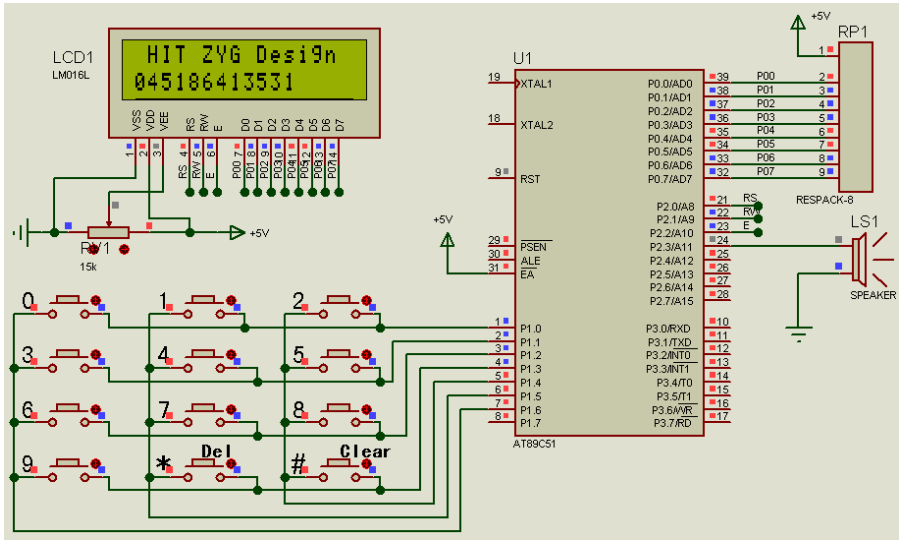


图12-14　电话拨号的模拟

参考程序如下。

```
#include<reg51.h>
#define uint unsigned int
#define uchar unsigned char
uchar keycode,DDram_value=0xc0;
sbit rs=P2^0;
sbit rw=P2^1;
sbit e =P2^2;
sbit speaker=P2^3;
uchar code table[]={0x30,0x31,0x32,0x33,0x34,
0x35,0x36,0x37,0x38,0x39, 0x20};
uchar code table_designer[]=" HIT ZYG Design ";
//第1行显示设计者信息
```



```
void lcd_delay();
void delay(uint n);
void lcd_init(void);
void lcd_busy(void);
void lcd_wr_con(uchar c);
void lcd_wr_data(uchar d);
uchar checkkey(void);
uchar keyscan(void);
void main()
{
    uchar num;
       lcd_init();
       lcd_wr_con(0x80);
```

```
for(num=0;num<=14;num++)
        {
                lcd_wr_data(table_designer[num]);
        }
        while(1)
        {
                keycode=keyscan();
                if((keycode>=0)&&(keycode<=9))
{
                lcd_wr_con(0x06);
                lcd_wr_con(DDram_value);
                lcd_wr_data(table[keycode]);
                DDram_value++;
        }
```



```
 else if(keycode==0x0b)
          {
                uchar i,j;
                j=0xc0;
                for(i=0;i<=15;i++)
                {
                      lcd_wr_con(j);
                      lcd_wr_data(table[10]);
                }
void lcd_delay()        //函数功能：液晶显示延时
{
  uchar y;
  for(y=0;y<0xff;y++)
  {
        ;
  }
}
```



```
 void lcd_init(void)                    //函数功能：液晶初始化
 {
    lcd_wr_con(0x01);
    lcd_wr_con(0x38);
```



```
lcd_wr_con(0x0c);
   lcd_wr_con(0x06);
}
void lcd_busy(void)             //函数功能：判液晶是否忙
{
        P0=0xff;
        rs=0;
        rw=1;
        e=1;
        e=0;
        while(P0&0x80)
        {
                e=0;
```

```
e=1;
        }
        lcd_delay();
}

void lcd_wr_con(uchar c)        //函数功能：向液晶显示器写入命令
{
lcd_busy();
        e=0;
        rs=0;
        rw=0;
        e=1;
```



```
        P0=c;
        e=0;
        lcd_delay();
}
void lcd_wr_data(uchar d) //函数功能：/向液晶写数据
{                       lcd_busy();
        e=0;
        rs=1;
        rw=0;
        e=1;
        P0=d;
        e=0;
        lcd_delay();
}
```



```
void delay(uint n)                          //函数功能：延时
{
        uchar i;
        uint j;
        for(i=50;i>0;i--)
    for(j=n;j>0;j--);
}

uchar checkkey(void)                //函数功能：检测键有无按下
{
        uchar temp;
        P1=0xf0;
        temp=P1;
```



```
temp=temp&0xf0;
        if(temp==0xf0)
        {
           return(0);
        }
        else{
           return(1);
        }
}
uchar keyscan(void)  //函数功能：键盘扫描并返回所按下的键盘号
{
   uchar hanghao,liehao,keyvalue,buff;
   if(checkkey()==0)
```

```
{
        return(0xff);              //无键按下，返回0xff
   }
   else                            //无键按下，返回0xff
   {
        uchar sound;
        for(sound=50;sound>0;sound--){
              speaker=0;
              delay(1);
              speaker=1;
              delay(1);
               }
               P1=0x0f;
```



```
              buff=P1;
              if(buff==0x0e)
              {
                    hanghao=0;
              }
              else if(buff==0x0d)
              {
                    hanghao=3;
              }
              else if(buff==0x0b)
              {
                    hanghao=6;
              }
```



```
   else if(buff==0x07)
                {
                   hanghao=9;
                }
                P1=0xf0;
                buff=P1;
                if(buff==0xe0)
                {
                      liehao=2;
                }
                else if(buff==0xd0)
                {
                      liehao=1;
                }
```



```
   else if(buff==0xb0)
                 {
                    liehao=0;
                 }
                 keyvalue=hanghao+liehao;
                 while(P1!=0xf0);
                 return(keyvalue);
    }
}
```

12.4.5　8位竞赛抢答器设计

目前，各类竞赛中大多用到竞赛抢答器，以单片机为核心配上抢答按钮开关以及数码管显示器并结合编写的软件，很容易制作一个竞赛抢答器，且修改方便。

1.设计要求

设计一个以单片机为核心8位竞赛抢答器，要求如下：

（1）抢答器同时供8名选手或8个代表队比赛，分别用8个按钮S0～S7表示。

（2）设置一个系统清除和抢答控制开关S，该开关由主持人控制。

（3）抢答器具有锁存与显示功能。即选手按动按钮，锁存相应的编号，且优先抢答选手的编号一直保持到主持人将系统清除为止。

（4）抢答器具有定时抢答功能，且一次抢答时间由主持人设定（如30秒）。当主持人启动“开始”键后，定时器进行减计时，同时扬声器发出短暂声响，声响持续时间为0.5s左右。

（5）参赛选手在设定的时间内进行抢答，抢答有效，定时器停止工作，显示器上显示选手的编号和抢答剩余时间，并保持到主持人将系统清除为止。

（6）如定时时间已到，无人抢答，本次抢答无效，系统报警并禁止抢答，定时显示器上显示00。

通过键盘改变可抢答时间，可把定时时间变量设为全局变量，通过键盘扫描程序使每按下一次按键，时间加1（超过30时置0）。同时单片机不断进行按键扫描，当参赛选手的按键按下时，用于产生时钟信号的定时计数器停止计数，同时将选手编号（按键号）和抢答时间分别显示在LED上。

2. 电路设计与仿真

【例12-5】8位竞赛抢答器的原理电路与仿真见图12-15。晶振频率为12MHz。图中为剩余18秒时，7号选手抢答成功。

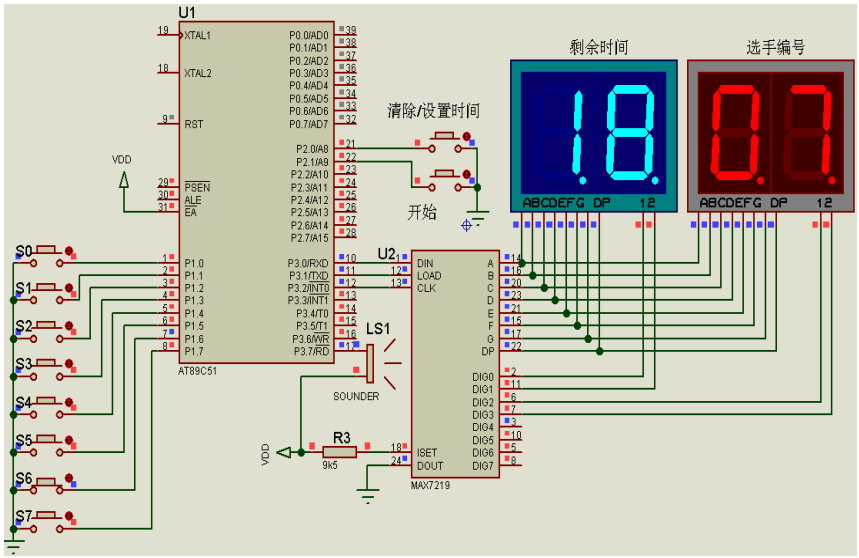


图12-15　　8位竞赛抢答器原理电路与仿真

图中MAX7219是一串行接收数据的动态扫描显示驱动器。MAX7219驱动8位以下LED显示器时，它的DIN、LOAD、CLK端分别与单片机三条口线(P3.0～ P3.2)相连。

MAX7219采用16位数据串行移位接收方式，即单片机将16 位二进制数逐位发送到DIN端，在CLK的每个上升沿将一位数据移入MAX7219 内移位寄存器，当16 位数据移入完后，在LOAD脚信号上升沿将16 位数据装入MAX7219内相应位置，对送入的数据进行BCD译码并显示。本例对MAX7219进行相应的初始化设置，具体请查阅有关MAX7219技术资料。

参考程序如下：

```
#include<reg51.h>
sbit DIN=P3^0;                  //与max7219接口定义
sbit LOAD=P3^1;
sbit CLK=P3^2;
sbit key0=P1^0;                 //8路抢答器按键
sbit key1=P1^1;
sbit key2=P1^2;
sbit key3=P1^3;
sbit key4=P1^4;
sbit key5=P1^5;
sbit key6=P1^6;
sbit key7=P1^7;
sbit key_clear=P2^0;            //主持人时间设置、清除
sbit begin=P2^1;                //主持人开始按键
```



```
sbit sounder=P3^7;                          //蜂鸣器
unsigned char second=30;                    //秒表计数值
unsigned char counter=0;                //counter每100，minite加1
unsigned char people=0;                 //抢答结果
unsigned char
num_add[]={0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};
                                        //max7219读写地址、内容
unsigned char
num_dat[]={0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89};
unsigned char keyscan()                 //键盘扫描函数
{
unsigned char keyvalue,temp;
keyvalue=0;
P1=0xff;
temp=P1;
```



```
if(~(P1&temp))
{
switch(temp)
{
        case 0xfe:
        keyvalue=1;
        break;
        case 0xfd:
        keyvalue=2;
        break;
        case 0xfb:
        keyvalue=3;
        break;
        case 0xf7:
        keyvalue=4;
        break;
```

```
                    case 0xef:
                    keyvalue=5;
                    break;
                    case 0xdf:
                    keyvalue=6;
                    break;
                    case 0xbf:
                    keyvalue=7;
                    break;
                    case 0x7f:
                    keyvalue=8;
                    break;
                    default:
                    keyvalue=0;
                    break;
            }
        }
       return keyvalue;
       }
```



```
void max7219_send(unsigned char add,unsigned char dat)
                                    // 函数功能：向max7219写命令
{
        unsigned char   ADS,i,j;
        LOAD=0;
        i=0;
        while(i<16)
        {
        if(i<8)
                {
                        ADS=add;
                }
        else
                {
                        ADS=dat;
                }
```



```
                for(j=8;j>=1;j--)
                {
                        DIN=ADS&0x80;
                        ADS=ADS<<1;
                        CLK=1;
                        CLK=0;
                }
                i=i+8;
        }
        LOAD=1;
        }

void max7219_init()                     //函数功能：max7219初始化
{
        max7219_send(0x0c,0x01);
        max7219_send(0x0b,0x07);
        max7219_send(0x0a,0xf5);
        max7219_send(0x09,0xff);
}
```



```
void time_display(unsigned char x)              //函数功能：时间显示
{
        unsigned char i,j;
        i=x/10;
        j=x%10;
        max7219_send(num_add[1],num_dat[j]);
        max7219_send(num_add[0],num_dat[i]);
}
 void scare_display(unsigned char x)//函数功能：抢答结果显示
{
        unsigned char i,j;
        i=x/10;
        j=x%10;
```

```
        max7219_send(num_add[3],num_dat[j]);
        max7219_send(num_add[2],num_dat[i]);
}
void holderscan()
        //函数功能：抢答时间设置，0-60s
{
        time_display(second);
        scare_display(people);
        if(~key_clear)
        //如果有键按下，改变抢答时间
        {
        while(~key_clear);
                if(people)
                //如果抢答结果没有清空，抢答器重置
                {
                        second=30;
```



```
        people=0;
                }
                if(second<60)
                {
                        second++;
                }
                else
                {
                        second=0;
                }
        }
}
void timer_init()                       //定时器T0初始化
{
        EA=1;
        ET0=1;
        TMOD=0x01;                      //定时器T0方式0定时
```



```
TH0=0xd8;               //装入定时器定时常数，设定10ms中断一次
TL0=0xef;
}
 void main()
{
while(1)
{
 do
{
 holderscan();
 }while(begin);         //开始前进行设置,若未按下开始键
 while(~begin);         //防抖
 max7219_init();                //芯片初始化
 timer_init();                  //中断初始化
 TR0=1;                                 //开始中断
  do
```



```
{
 time_display(second);
 scare_display(people);
 people=keyscan();
  }while((!people)&&(second)); //运行直到抢答结束或时间结束
  TR0=0;
  }
}
void timer0() interrupt 1
                //定时器T0中断函数
{
        if(counter<100)
        {
        counter++;
                if(counter==50)
                {
                        sounder=0;
                }
```

```
}
else
{
        sounder=1;
        counter=0;
        second=second-1;
}
        TH0=0xd8;       //重新装载
        TL0=0xef;
        TR0=1;
}
```


12.4.6 基于时钟/日历芯片DS1302的电子钟设计

在单片机应用系统中，有时往往需要一个实时时钟/日历作为测控时间基准。时钟/日历集成电路芯片多种，设计者只需选择合适芯片即可。本节介绍最为常见的时钟/日历芯片DS1302的功能、特性以及单片机的硬件接口设计及软件编程。

1. 工作原理

（1） 基本性能

时钟/日历芯片DS1302是美国DALLAS公司推出的涓流充电时钟芯片，功能特性如下。

- 能计算2100年前的年、月、日、星期、时、分、秒的信息；每月的天数和闰年天数可自动调整；时钟可设置为24或12小时格式。
- 与单片机间采用单线同步串行通信。

- 31字节的8位静态RAM。
- 功耗低，保持数据和时钟信息时功率小于1mW；可选的涓流充电能力。
- 读/写时钟或RAM数据有单字节和多字节两种传送方式。

DS1302引脚见图12-7。

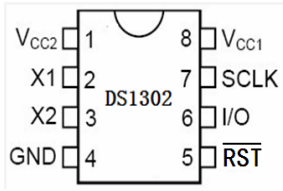


图12-7　DS1302的引脚

各引脚功能如下：

- I/O：数据输入/输出。
- SCLK：同步串行时钟输入。
- RST*：芯片复位，1—芯片的读/写使能，0—芯片复位并被禁止读/写。
- VCC2：主电源输入，接系统电源。
- VCC1：备份电源输入引脚，通常接2.7~3.5V电源。当VCC2>VCC1+0.2V时，芯片由VCC2供电；当VCC2<VCC1时，芯片由VCC1供电。
- GND：地
- X1，X2：接32.768KHz晶振引脚。

单片机与DS1302间无数据传输时，SCLK保持低电平，此时如果 从低变为高时，即启动数据传输，此时SCLK的上升沿将数据写入DS1302，而在SCLK的下降沿从DS1302读出数据。 为低时，则禁止数据传输,读/写时序如图12-8所示。数据传输时，低位在前，高位在后。

2. DS1302的命令字格式

单片机对**DS1302**的读/写，都必须由单片机先向**DS1302**写入一个命令字**(8**位**)** 发起，命令字的格式见表12-1。

表 12-1 DS1302 的命令字格式

D7	D6	D5	D4	D3	D2	D1	D0
1	RAM/$\overline{CK}$	A4	A3	A2	A1	A0	RD/$\overline{W}$

命令字各位功能：

D7：必须为逻辑**1**，如为**0**，则禁止写入**DS1302**。

D6：**1**—读/写**RAM**数据，**0**—读/写时钟/日历数据。

D5～D1：为读/写单元的地址；

D0：**1**—对**DS1302**读操作，**0**—对**DS1302**写操作。

注意，命令字（8位）总是低位在先，命令字每1位都是在SCLK上升沿送出。

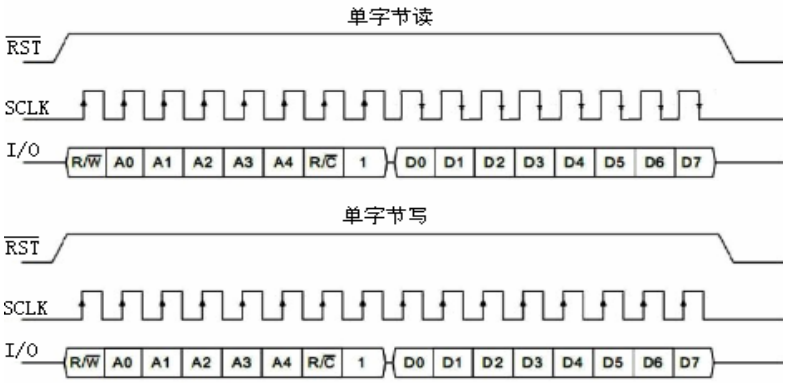


图12-18 **DS1302**读/写时序

3. DS1302的内部寄存器

片内各时钟/日历寄存器以及其它的功能寄存器见表12-2。通过向寄存器写入命令字实现对**DS1302**操作。

例如，如要设置秒寄存器的初始值，需要先写入命令字**80H**(见表12-2)，然后再向秒寄存器写入初始值；如要读出某时刻秒值，需要先写入命令字**81H**，然后再从秒寄存器读取秒值。

表**12-2**中各寄存器 “取值范围”**1**列存放的数据均为**BCD**码。

表 12-2　主要寄存器、命令字与取值范围及各位内容

寄存器名（地址）	命令字		取值范围	各 位 内 容							
	写	读		D7	D6	D5	D4	D3～D0			
秒寄存器（00H）	80H	81H	00～59	CH	10SEC			SEC			
分寄存器（01H）	82H	83H	00～59	0	10MIN			MIN			
小时寄存器（02H）	84H	85H	01～12 或 00～23	12/24	0	AP	HR	HR			
日寄存器（03H）	86H	87H	01～28, 29, 30, 31	0	0	10DATE		DATE			
月寄存器（04H）	88H	89H	01～12	0	0	0	10M	MONTH			
星期寄存器（05H）	8AH	8BH	01～07	0	0	0	0	DAY			
年寄存器（06H）	8CH	8DH	01～99	10YEAR				YEAR			
写保护寄存器（07H）	8EH	8FH		WP	0	0	0	0			
涓流充电寄存器（08H）	90H	91H		TCS	TCS	TCS	TCS	DS	DS	RS	RS
时钟突发寄存器（3EH）	BEH	BFH									

- CH：时钟暂停位，1-振荡器停止，DS1302为低功耗方式；0-时钟开始工作。
- 10SEC：秒的十位数字，SEC为秒的个位数字
- 10MIN：分的十位数字，MIN为分的个位数字
- 12/24：12或24小时方式选择位
- AP：小时格式设置位， 0-上午模式（AM）；1-下午模式（PM）
- 10DATE：日期的十位数字，DATE为日期的个位数字
- 10M：月的十位数字，MONTH为日期的个位数字
- DAY：星期的个位数字

- 10YEAR：年的十位数字，YEAR为年的十位数字

表12-2中后3个寄存器的功能及特殊位符号的意义说明如下。

写保护寄存器：该寄存器的D7位WP是写保护位，其余7位（D0～D6）置为0。在对时钟/日历单元和RAM单元进行写操作前，WP必须为0，即允许写入。当WP为1时，用来防止对其它寄存器进行写操作。

涓流充电寄存器：慢充电寄存器，用于管理对备用电源的充电。

TCS：当4位TCS=1010时，才允许使用涓流充电寄存器，其他任何状态都将禁止使用涓流充电器。

DS：两DS位用于选择连接在VCC2和VCC1间的二极管数目。

01-选择1个二极管；10-选择2个二极管；

11或00-涓流充电器被禁止。

RS：两位RS位用于选择涓流充电器内部在VCC2和VCC1之间的连接电阻。RS=01，选择R1（2kΩ）；RS=10时，选择R2（4kΩ）；RS=11时，选择R3（8kΩ）；RS=00时，不选择任何电阻。

时钟突发寄存器：单片机对DS1302除单字节数据读/写外，还可采用突发方式，即多字节连续读/写。在多字节连续读/写中，只要对地址为3EH的时钟突发寄存器进行读/写操作，即把对时钟/日历或RAM单元的读/写设定为多字节方式。该方式，读/写都开始于地址0的D0位。当多字节方式写时钟/日历时，必须按照数据传送的次序写入最先的8个寄存器；但是以多字节方式写RAM时，没有必要写入所有的31个字节，每个被写入的字节都被传输到RAM，无论31个字节是否都被写入。

2. 电路设计与编程

【例12-5】制作一个使用时钟/日历芯片DS1302并采用LCD1602显示的日历/时钟，基本功能如下。

（1）显示6个参量的内容，第一行显示：年、月、日；第二行显示：时、分、秒。

（2）闰年自动判别。

（3）键盘采用动态扫描方式查询，参量应能进行增1修改，由“启动日期与时间修改”功能键k1与6个参量修改键组合来完成增1修改。即先按一下k1，然后按一下被修改参量键，即可使该参量增1，修改完毕，再按一下k1表示修改结束确认。

本例时钟/日历原理电路与仿真见图12-19。LCD1602分两行显示日历与时钟。

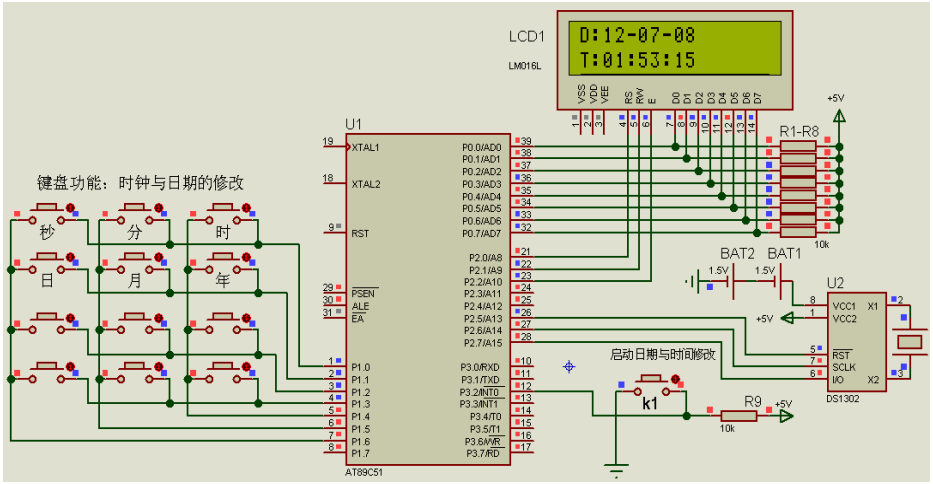


图12-19 LCD显示的时钟/日历原理电路及仿真

图12-19的4×3矩阵键盘，只用到了其中2行键共6个，余下的按键，本例未使用，可用于将来键盘功能扩展。

参考程序如下：

```
#include<reg51.h>
#include "LCD1602.h"                    //液晶显示器LCD1602头文件
#include "DS1302.h"                     //时钟/日历芯片DS1302头文件
#define uchar unsigned char
#define uint unsigned int
bit key_flag1=0,key_flag2=0;
SYSTEMTIME adjusted;                    // 此处为结构体定义
```

```
uchar sec_add=0,min_add=0,hou_add=0;
uchar day_add=0,mon_add=0,yea_add=0;
uchar data_alarm[7]={0};
int key_scan()                  // 函数功能：键盘扫描，判是否有键按下
{
        int i=0;
        uint temp;
        P1=0xf0;
        temp=P1;
        if(temp!=0xf0)
        {
        i=1;
        }
        else
        {
        i=0;
        }

        return i;
}
```



```
uchar key_value()       //函数功能：获取按下的按键值
{
  uint m=0,n=0,temp;
  uchar value;
  uchar v[4][3]={'2','1','0','5','4','3','8','7','6','b','a','9'};
  P1=0xfe;temp=P1; if(temp!=0xfe)m=0;
//采用分行、分列扫描的形式获取按键键值
  P1=0xfd;temp=P1; if(temp!=0xfd)m=1;
  P1=0xfb;temp=P1; if(temp!=0xfb)m=2;
  P1=0xf7;temp=P1; if(temp!=0xf7)m=3;
  P1=0xef;temp=P1; if(temp!=0xef)n=0;
  P1=0xdf;temp=P1; if(temp!=0xdf)n=1;
  P1=0xbf;temp=P1; if(temp!=0xbf)n=2;
```



```
   value=v[m][n];
    return value;
}
void adjust(void)
//函数功能：修改各参量
{
        if(key_scan()&&key_flag1)
        switch(key_value())
        {
        case '0':sec_add++;break;
        case '1':min_add++;break;
        case '2':hou_add++;break;
        case '3':day_add++;break;
        case '4':mon_add++;break;
        case '5':yea_add++;break;
```



```
        default: break;
        }
    adjusted.Second+=sec_add;
    adjusted.Minute+=min_add;
    adjusted.Hour+=hou_add;
    adjusted.Day+=day_add;
    adjusted.Month+=mon_add;
    adjusted.Year+=yea_add;
    if(adjusted.Second>59)
        {
                adjusted.Second=adjusted.Second%60;
                adjusted.Minute++;
        }
        if(adjusted.Minute>59)
        {
                adjusted.Minute=adjusted.Minute%60;
                adjusted.Hour++;
        }
```

```
if(adjusted.Hour>23)
        {
                adjusted.Hour=adjusted.Hour%24;
                adjusted.Day++;
        }
        if(adjusted.Day>31)
                adjusted.Day=adjusted.Day%31;
        if(adjusted.Month>12)
                adjusted.Month=adjusted.Month%12;
        if(adjusted.Year>100)
                adjusted.Year=adjusted.Year%100;
}
```



```
void changing(void) interrupt 0 using 0
//中断处理函数，修改参量，或修改确认
{
        if(key_flag1)
        key_flag1=0;
        else
        key_flag1=1;
}

main()                  //主函数
{
        uint i;
        uchar p1[]="D:",p2[]="T:";
        SYSTEMTIME T;
        EA=1;
        EX0=1;
        IT0=1;
        EA=1;
```



```
        EX1=1;
        IT1=1;
        init1602();
        Initial_DS1302() ;
        while(1){
                write_com(0x80);
                write_string(p1,2);
                write_com(0xc0);
                write_string(p2,2);
                DS1302_GetTime(&T) ;
                adjusted.Second=T.Second;
                adjusted.Minute=T.Minute;
                adjusted.Hour=T.Hour;
                adjusted.Week=T.Week;
```



```
        adjusted.Day=T.Day;
        adjusted.Month=T.Month;
        adjusted.Year=T.Year;
        for(i=0;i<9;i++)
       {
                adjusted.DateString[i]=T.DateString[i];
                adjusted.TimeString[i]=T.TimeString[i];
        }
                adjust();
                DateToStr(&adjusted);
                TimeToStr(&adjusted);
                write_com(0x82);
        write_string(adjusted.DateString,8);
        write_com(0xc2);
        write_string(adjusted.TimeString,8);
                delay(10);
        }
}
```

程序中，使用了自编的液晶显示器LCD1602的头文件“LCD1602.h”，由于液晶显示器LCD1602是单片机应用系统经常用到的器件，因此将其常用到的驱动函数等函数，写成一个头文件，如果以后在其他项目中也用到LCD1602，只需将该头文件包含进来即可，这样程序的编写提供了方便。同理对时钟/日历芯片DS1302的控制，也可自编头文件“DS1302.h”，以后在其他项目中将该头文件包含进来即可。上述两个头文件清单见附录1与附录2。