·配套资料下载

单片机C语言

程序设计实训100例

——基于8051+Proteus仿真（第2版）

彭　伟　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

单片机C语言程序设计实训100例

——基于8051+Proteus仿真

（第2版）

彭　伟　编著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书基于 Keil μVision 程序设计平台和 Proteus 硬件仿真平台，精心编写了 100 个 8051 单片机 C 语言程序设计案例，并对各案例分别提出了难易适中的实训目标。

全书基础设计类案例涵盖 8051 单片机最基本的端口编程、定时器/计数器应用、中断程序设计和串口通信程序设计；硬件应用类案例涵盖编码与解码器件、串并与并串转换器件、并行存储器及接口扩展器件、LED 显示及驱动器件、字符液晶显示器件、图形液晶显示器件、实时日历时钟器件、A/D 与 D/A 转换器件、I^2C 接口器件、SPI 接口器件、1-Wire 总线器件及其他器件共计 12 类；综合设计类案例包括大量实用项目设计，如多功能电子日历牌、计算器、电子秤、密码锁、多点温度监测、大幅面 LED 点阵屏设计、交流电压检测、K 型热电偶及铂电阻温度计、GPS 系统、红外遥控系统、测距系统、温室监控系统、Modbus 总线系统等。

本书适用于本科、专科院校学生用于学习实践 8051 单片机 C 语言程序设计技术的教材或参考书，也可作为工程技术人员或单片机技术爱好者的学习参考书或工具书。

图书在版编目（CIP）数据

单片机 C 语言程序设计实训 100 例：基于 8051+Proteus 仿真/彭伟编著. —2 版. —北京：电子工业出版社，2012.10

ISBN 978-7-121-18655-4

Ⅰ. ①单… Ⅱ. ①彭… Ⅲ. ①单片微型计算机—C 语言—程序设计 Ⅳ. ①TP368.1 ②TP312

中国版本图书馆 CIP 数据核字（2012）第 235838 号

策划编辑：万子芬（wzf@phei.com.cn）
责任编辑：万子芬
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：28.75　字数：736 千字
印　　次：2012 年 10 月第 1 次印刷
定　　价：58.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

再版前言

不同于通用计算机应用程序设计，单片机 C 语言程序设计必须针对具体的微控制器及外围电路来进行，为此，很多公司推出了单片机实验箱等多种配套设备，其不菲的价格令人望而却步，这使得众多读者对单片机技术学习、研究与应用的愿望难以实现。

幸运的是，英国 Labcenter 公司推出的 Proteus 软件具有单片机系统仿真功能，能够很好地支持多种单片机，系统器件库包含大量元器件，并提供了多种虚拟仪器，使得仅用一台 PC 在纯软件环境中完成单片机系统设计、调试、运行成为可能，这无疑为读者学习、运用单片机 C 语言程序设计技术提供了理想平台。

本书基于 μVision 集成开发环境和 Proteus 仿真软件搭建组合平台，系统规划、精心设计了 100 个 8051 单片机 C 语言程序设计案例，所有案例均可以在仿真电路中调试和交互运行，具体内容分为以下三部分：

1．语言程序与仿真平台应用基础 包括第 1、2 章，简要介绍开发单片机 C 语言程序必须熟悉与重点掌握的内容，以及 Proteus 仿真软件的基本应用技术，为全书案例的学习提供铺垫。

2．内置资源与扩展资源应用设计 包括第 3、4 章，分别为基础设计与硬件应用两类案例。基础设计类案例涵盖 8051 单片机所有内置资源，包括端口编程、中断、定时器/计数器等；硬件应用类案例涵盖 12 类扩展资源应用技术，包括编码/解码、串并/并串转换、并行存储及接口扩展、LED 显示及驱动、字符/图形液晶、RTC、A/D 与 D/A、I2C、SPI、1-Wire 接口及其他器件。

3．资源整合与功能集成应用设计 由第 5 章提供，全部为综合型案例，如多功能电子日历、计算器、电子秤、密码锁、多点温度监测、大幅面 LED、交流电压检测、GPS 系统、红外遥控、测距、温室监控、Modbus 总线系统等。

本书第 1 版 2009 年 6 月推出后，受到读者厚爱，已连续 6 次重印。经过几年来对案例的进一步累积、优化、凝练与规范，结合最新版本仿真软件提供的新型器件及功能，并充分考虑读者的反馈意见，应电子工业出版社之约，现全新推出第 2 版。

第 2 版在保留第 1 版体例结构及撰写特色的基础上进行了以下修改：

- 大幅增加扩展硬件类案例，并进行细致归类，优化器件的分配比重。
- 调整并增加资源整合与功能集成应用类案例，突出本书的实用价值。
- 给案例源码添加行号，便于阅读定位；补充大量代码注释，便于分析研究。
- 进一步规范仿真电路、设计简介、技术资源与程序源码，并删减冗余内容。

本书特点：

- 全书代码均调试通过，可作为 8051 单片机的“代码宝典”或“代码手册”使用。

- 大量“核心源码”及“可移植源码”可直接应用到工程项目中。
- 案例电路可实时仿真运行，增加了学习过程的趣味性并提振信心。
- 所规划的案例设计循序渐进，所设定的实训目标难易适中。

这些特点使读者“漫步”于系统电路、程序逻辑、工作时序、接口设计、读写控制、功能实现之中，便可轻松提高8051单片机C语言程序设计水平与系统开发效率。

在本书选题、撰写到出版的全过程中，学院领导、学院教务、科研等部门均给予了重要支持，并提供了项目资助，在此对学院及部门领导的关心与支持表示由衷感谢！

由于作者水平有限，且全书撰写任务极其繁重，书中错漏之处在所难免，在此真诚欢迎读者多提宝贵意见，以期不断改进。作者邮箱pw95aaa@foxmail.com。

本书所有案例的配套资料压缩包可到电子工业出版社华信教育资源网（http://www.hxedu.com.cn）免费下载，其中包括案例的仿真电路、C语言源程序框架及编译生成的HEX文件。

彭　伟

2012年10月于武昌

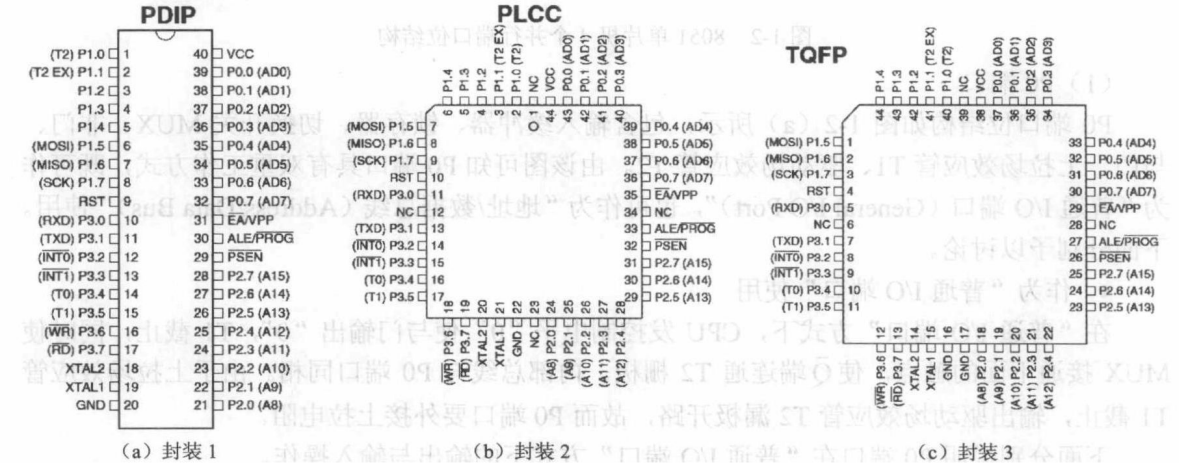
目 录

第 1 章　8051 单片机 C 语言程序设计概述

开发 8051 单片机应用系统时，使用 C 语言可以大幅提高开发效率，缩短开发周期，所编写的程序可读性好且易于移植，选用 C 语言开发单片机应用系统程序已经成为必然趋势。为引导读者深入学习 8051 单片机内置资源、扩展硬件开发技术及综合型项目设计技术，本书给出了 100 项 Keil μVision 开发平台下编写的单片机 C 程序及 Proteus 仿真电路，各案例后面均提出了两个以上的难易适中的实训要求。阅读本书要求读者已经学习了 8051 单片机 C 语言程序设计基本技术，本章仅介绍使用 C 语言设计单片机应用系统必须参考和重点掌握的内容，这些内容会给阅读、调试、研究全书案例及进行设计实训提供重要帮助。

1.1　8051 单片机引脚

图 1-1 给出了 8051 单片机的几种不同封装形式及引脚分布，本节将对单片机 4 个双向输入/输出（I/O）端口引脚、控制引脚、晶振及电源引脚进行介绍。



（a）封装 1　　（b）封装 2　　（c）封装 3

图 1-1　8051 单片机不同封装形式及引脚图

图 1-2 给出了 8051 的 4 个 8 位并行端口 P0、P1、P2、P3 的位结构，这些端口都是双向的，每个端口位均包含：两个三态输入缓冲器、一个输出锁存器及一个场效应管（FET）驱动器，其中 P0 端口还有一个上拉场效应管。位结构中的两个输入缓冲器分别受内部“读锁存器”和“读引脚”信号控制，“位锁存器”用典型的“D 型触发器”表示。

当 CPU 发出“写锁存器”脉冲信号到 D 触发器 CL 端时，内部总线的值将送入 D 触发器。

当 CPU 发出“读锁存器”脉冲信号到缓冲器 1 时，触发器 Q 端输出至内部总线上。

当 CPU 发送“读引脚”脉冲信号到缓冲器 2 时，外部引脚输入值将置于内部总线上。

某些指令读端口时将激活“读锁存器”信号，另一些指令则激活“读引脚”信号。对于“读-改-写”这样的指令操作，CPU 发出的将是“读锁存器”信号。具体执行“读锁存器”还

是“读引脚”将由 CPU 根据不同指令自动处理。有关细节将在 1.3 节“特殊功能寄存器”中讨论。

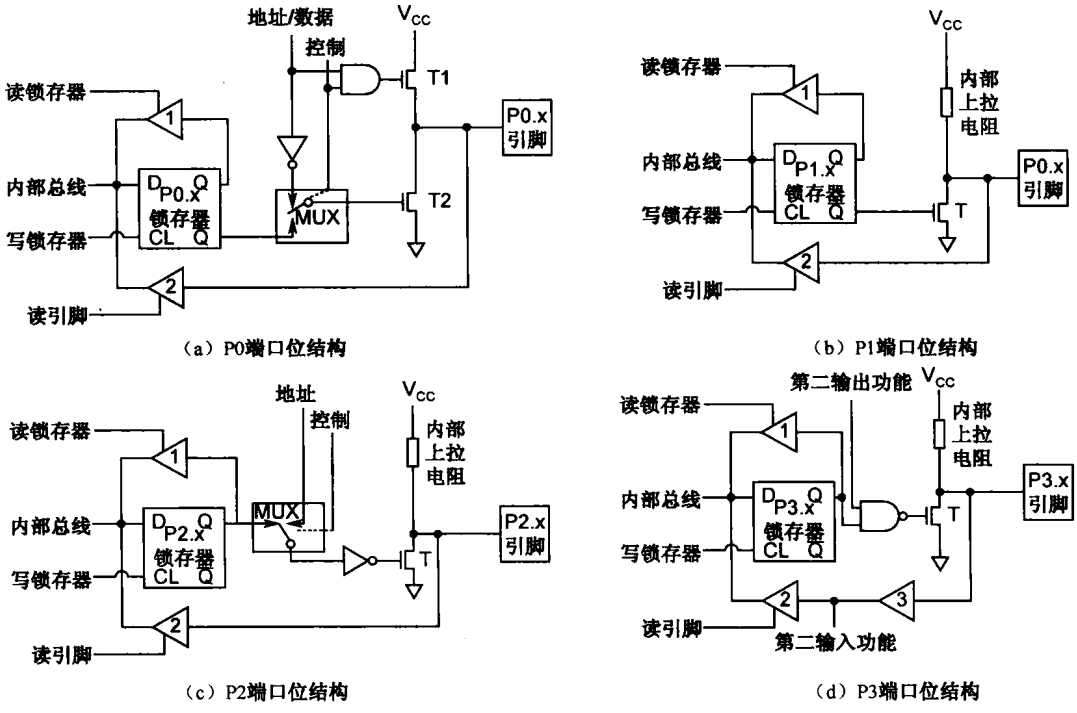


图 1-2　8051 单片机 4 个并行端口位结构

（1）P0 端口

P0 端口位结构如图 1-2（a）所示，包含输入缓冲器、锁存器、切换开关 MUX、非门、与门、上拉场效应管 T1、驱动场效应管 T2。由该图可知 P0 端口具有双重工作方式，既可作为“普通 I/O 端口（General I/O Port）”，也可作为“地址/数据总线（Address/Data Bus）”使用。下面分别予以讨论。

- 作为“普通 I/O 端口”使用

在“普通 I/O 端口”方式下，CPU 发控制电平“0”使与门输出“0”，T1 截止，同时使 MUX 接通下面的触点，使 $\overline{Q}$ 端连通 T2 栅极，内部总线与 P0 端口同相。由于上拉场效应管 T1 截止，输出驱动场效应管 T2 漏极开路，故而 P0 端口要外接上拉电阻。

下面分别说明 P0 端口在“普通 I/O 端口”方式下的输出与输入操作。

输出操作：以 3.3 节“花样流水灯”为例，8 只 LED 阴极接 P0 端口，阳极通过限流电阻接 V_{CC}，这一组限流电阻同时扮演了上拉电阻的角色。如果将这一组 LED 阳极接 P0 端口，阴极串接限流电阻后接 GND，尽管 P0 端口仍然在输出 0、1 序列，但 8 只 LED 却无法实现演示效果，因为场效应管 T2 没有上拉电阻。此时可仍将 LED 阳极接 P0 端口，阴极接 GND，电阻则改为一端接 P0 端口，另一端接 V_{CC} 上拉。

输入操作：如果此前内部总线刚刚输出了低电平，此时锁存器 Q=0，$\overline{Q}$=1，驱动场效应管 T2 导通，接口呈现低电平，此时无论 P0 端口外部信号是“1”还是“0”，从 P0 端口引脚读取的信号都将为“0”，这显然将无法正确读取接口引脚信号。

故而，在执行接口输入操作前，应先向 P0 端口锁存器写“1”，D 触发器的 $\overline{Q}$ 端输出“0”使 T2 截止，外部引脚处于悬浮(Float/FLT)状态，变为高阻抗输入（此时 T1 也是截止的）。

3.10 节“继电器及双向可控硅控制照明设备”案例中，使用 P0.0 作为 K1 按键输入引脚，P0.0 外接了上拉电阻（因为 T1 截止，T2 漏极开路），但读取按键输入前，整个源代码中并未出现向 P0.0 写“1”的操作，这是因为单片机上电时默认已经向所有接口（P0、P1、P2、P3）全部写入“1”，仿真调试时观察到初始时 P0 为灰色，这显然就是因为初始上电时，内部默认写入 P0 锁存器的“1”已经截止了 T2。

- 作为“地址/数据总线”使用

P0 端口作为“地址/数据总线”使用时通过 CPU 内部写控制信号“1”实现，它使 MUX 连接上面的触点，“地址/数据”信号通过“非门”连接到 T2 的栅极，由于控制信号为“1”，“地址/数据”信号通过“与门”连接 T1 的栅极，实际上相当于“直接”或称“同相”连接到 T1 的栅极。“地址/数据总线”方式下的输出与输入操作分别说明如下：

输出操作：输出的“地址”或“数据”信号将通过“与门”驱动 T1，并同时通过“非门”驱动 T2，例如，输出 0 时：T1 截止，T2 导通；反之，输出 1 时：T1 导通，T2 截止，从而以推挽方式实现信号输出。

输入操作：从外部设备读取输入“数据”时，数据信号将通过缓冲器 2 进入内部总线，数据输入时，CPU 将通过写控制信号“0”使 T1 截止，此时相当于瞬间又自动回到了“普通 I/O 端口”方式，随即 CPU 自动向 P0 锁存器写“1”截止 T2，并通过“读引脚”控制缓冲器读取外部数据，此时的“普通 I/O 方式”下的差别与此前讨论过的 I/O 方式有两个差别：一是无须外部上拉，二是向 P0 锁存器写“1”截止 T2 的操作是自动完成。

在“地址/数据总线”应用方式下，P0 端口复用数据总线（D0～D7）及地址总线的低 8 位（A0～A7），数据及低 8 位地址分时复用 P0 端口，低 8 位地址由 ALE（Address Latch Enable，地址锁存允许）信号的下降沿锁存到外部地址锁存器中（如 74LS373），地址的高 8 位则通过 P2 端口输出，不经过锁存器，输出地址后，紧接着的数据输出或输入操作将在 $\overline{\text{WR}}$/P3.6 及 $\overline{\text{RD}}$/P3.7 的自动控制下完成。第 4 章 62256RAM、8255 等相关案例均涉及了 P0 端口的“地址/数据总线”应用。

小结：P0 在“普通 I/O 端口”方式下为准双向接口，因为输出“0”后改为输入时，需要先输出“1”，然后才能成为输入端口。而在“地址/数据总线”方式下 P0 为真正的双向 I/O 端口。

（2）P1 端口

P1 端口是通用的准双向 I/O 端口，由一个输出锁存器、两个三态输入缓冲器和一个输出驱动场效应管及内部上拉电阻组成。P1 端口与 P0 端口作为“普通 I/O 端口”使用时的原理相似，它相当于 P0 端口省去了“与门”、“非门”、“MUX”，且上拉场效应管 T1 由内部上拉电阻代替，P1 端口不再需要外接上拉电阻。与 P0 用作普通 I/O 的操作一样，作为输入端口使用时，除了初始时不需要向接口写“1”截止驱动场效应管以外，如果以后曾向接口输出过“0”，则每当由“写”操作改为“读”操作时，都需要先向接口写“1”截止场效应管，然后才能正常读取输入的数据。

以 4.25 节“ADC0832 模数转换与 LCD 显示”为例，ADC0832 的 DI 与 DO 引脚合并连接到单片机的 P1.2 引脚，源程序中 A/D 转换函数内有如下代码：

```
105     //第 3 个时钟脉冲上升沿之前，设 DI = 0/1，分别对应于选择 CH0/CH1
106     CLK = 0;  DIO = 0;  _nop_(); _nop_();
107     CLK = 1;            _nop_(); _nop_();
108     //第 3 个时钟脉冲下降沿之后置 DI = 1，释放数据线，准备接收
109     //P1 端口读取数据时需要先写 1，否则总线将出现逻辑冲突(黄色方块闪烁)
110     CLK = 0;  DIO = 1;  _nop_(); _nop_();
```

由于第 110 行之前对 DIO（P1.2）的操作均为输出，用于启动及配置 ADC0832，这些工作的最后一步是第 106 行，它置 DIO（P1.2）为 0，故而在第 110 行一定要向 DIO（P1.2）写 1，使端口内部驱动场效应管 T 被截止，从而释放数据线，以便开始串行读取 ADC0832 输出给单片机的 A/D 转换值。所谓释放数据线，指该数据线当前处于被接口内部或外部上拉电阻拉为高电平的状态，不再受接口内部电路控制。类似的，第 4 章有关 1-Wire 总线器件 DS18B20、DS2405 等案例也需要在串行输出后切换为输入时，在相应接口上写“1”来释放总线，准备读取数据。

（3）P2 端口

P2 端口与 P1 端口相比多出了一个转换控制部分，当 P2 与 P0 配合作为“地址/数据总线”方式下的高 8 位地址线（A8～A15）使用时，CPU 将写控制信号“1”使 MUX 切换到右边，在“地址/数据总线”方式下，无论 P2 端口剩余多少地址线，均不能被用于普通 I/O 操作。

反之，CPU 通过写控制信号“0”将 MUX 切换到左边，使之工作于“普通 I/O 端口”方式。作为“普通 I/O 端口”使用时，P2 锁存器 Q 端输出通过“非门”驱动场效应管，相当于 P1 端口中的通过 $\overline{Q}$ 直接驱动场效应管，在“普通 I/O 端口”方式下，P2 与 P1 同为准双向 I/O 端口。

（4）P3 端口

P3 端口为具有双重功能的 I/O 端口，与 P1 相比，它增加了第二 I/O 功能。

作为“普通 I/O 端口”使用时，CPU 将第二输出功能控制线保持为“1”，锁存器 Q 端通过“与非门”（此时等价于“非门”）驱动场效应管，相当于 P1 端口中的通过 $\overline{Q}$ 直接驱动场效应管，或相当于 P2 端口中通过 Q 端经“非门”驱动场效应管。在这种方式下，其读/写操作与 P1、P2 相同。

下面接着讨论 P3 端口处于第二重功能时的相关操作：

当处于第二功能输出时，CPU 自动向 P3 锁存器写“1”，由于 Q=1，“与非门”相当一个“非门”，此时的输出将仅仅由第二功能输出线决定，例如，由 UART 模块通过 TXD 输出的 SBUF 寄存器串行数据及 $\overline{RD}$ 、$\overline{WR}$ 引脚输出的读/写控制信号。

当处于第二功能输入时，CPU 除自动向 P3 锁存器写“1”，置 Q=1 以外，还将向第二功能输出线写“1”，以保证 Q 和第二功能输出线经过“与非门”后输出 0，使得场效应管 T 被截止，此时所读取的 P3 端口引脚信号将通过缓冲器 3 直接进入第二功能输入端，例如，从 RXD、INT0、INT1、T0、T1 引脚读取的信号将通过第二功能输入端分别进入单片机内部的串行模块、外部中断处理模块、定时器/计数器模块进行处理。

在下述情况下，P3 端口相应引脚将处于第二功能状态：

- 启动串行通信模块（RXD/TXD）；
- 使能处部中断输入（INT0/INT1）；
- 定时器/计数器配置为外部计数状态（T0/T1）；
- 访问外部扩展存储器或接口扩展器件（$\overline{WR}$ / $\overline{RD}$ ）。

在第二功能下，P3.0～P3.7 引脚分别对应于 RXD、TXD、$\overline{INT0}$ 、$\overline{INT1}$ 、T0、T1、$\overline{WR}$ 及 $\overline{RD}$，第 3 章有关串口、外部中断及定时器/计数器的案例分别涉及前 6 位，第 4 章有关 62256 存储器扩展和 8255 接口扩展等案例连接了最后 2 位 $\overline{WR}$ 及 $\overline{RD}$，它们用于总线控制。

（5）8051 单片机的其他相关引脚

8051 的控制引脚有 RST，ALE/$\overline{PROG}$，$\overline{PSEN}$，$\overline{EA}$ /VPP，本书多数案例中的 RST（Reset）引脚连接了系统复位外围电路，只有部分案例未在 RST 引脚连接复位电路，但这并不影响

Proteus 仿真。ALE 引脚在存储器扩展和 I/O 端口扩展案例中用到。另外，由于全书所有案例程序都绑定在 8051 单片机内部程序 ROM 中，故而仿真电路中 $\overline{EA}$ 全部接高电平。

XTAL1、XTAL2 引脚在本书多数案例连接有 12MHz 或 11.0592MHz 振荡器，在串行通信案例中多数选择 11.0592MHz 振荡器。Proteus 默认晶振连接时不影响仿真运行，因为振荡器频率可在芯片属性中设置。主电源引脚 V_{SS}，V_{CC} 分别接 GND 和+5V，Proteus 仿真时电源已默认连接，仿真电路原理图中电源连接全部默认，注意在实物电路中不要忽略了电源与晶振连接。

1.2 数据与程序内存

8051 单片机存储器组织结构如图 1-3 所示，存储器分为以下 4 部分：

① 片内程序存储器（ROM）4KB；

② 片外程序存储器（ROM）64KB（含内部 4KB）；

③ 片内数据存储器（RAM）256B；

④ 片外数据存储器（RAM）64KB。

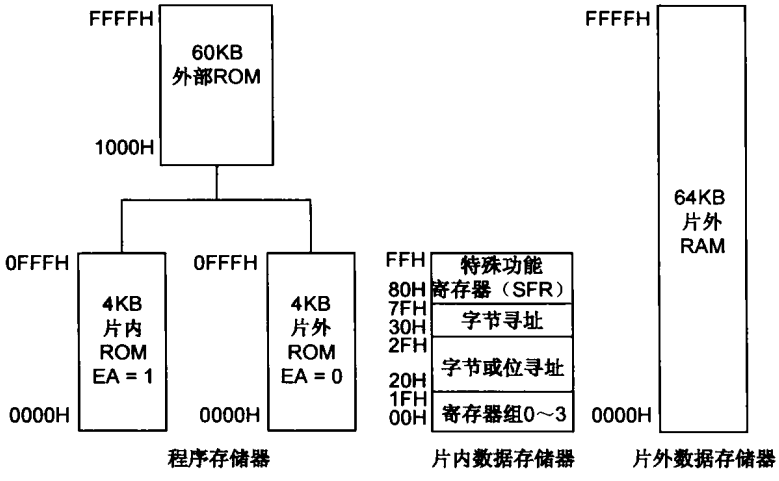


图 1-3　8051 单片机存储器组织结构

这 4 部分中的前 2 部分是程序 ROM，片内程序 ROM 用于存放 8051 控制程序，本书各案例中 8051 单片机绑定的都是由 C 语言程序编译链接生成的 HEX 程序文件，这相当在实物电路中向 4KB 的 8051 单片机片内程序 ROM 中烧写 HEX 程序，要从片内程序 ROM 的 4KB 存储器取指令时，注意将 $\overline{EA}$ 接高平。8051 片外程序 ROM 可扩展至 64KB，地址范围为 0000H～0FFFH 的 4KB 片内程序 ROM 地址是重叠的，使用过程中要用 $\overline{EA}$ 的取值来区分，对于带片内程序 ROM 空间的 8051 单片机，$\overline{EA}$ 可接高电平，单片机运行时将从片内程序 ROM 的 0000H 地址开始执行，当程序计数器（PC）的值超过 0FFFH 时，自动转换到片外程序 ROM 的 1000H～FFFFH 地址空间执行。对于没有片内程序 ROM 的 8051 单片机，例如，8031 单片机，程序存放于片外程序 ROM，其 $\overline{EA}$ 引脚固定接低电平。

8051 单片机的数据 RAM 空间虽然很小，但它起着非常重要的作用，256 字节被分为两个区，00H～7FH 的 128 字节空间是真正的 RAM 区，可读/写各种数据；80H～FFH 的 128B 空间大部分专门用于特殊功能寄存器（Special Function Register，SFR），8051 单片机在这个

空间安排了 21 个特殊功能寄存器（SFR），不论是用汇编语言还是用 C 语言编写单片机程序，这些特殊功能寄存器都要重点熟练掌握。

1.3 特殊功能寄存器

由图 1-3 可知，片内数据 RAM 中 80H～FFH 空间大部用于特殊功能寄存器，表 1-1 列出了 8051 单片机的所有特殊功能寄存器。

表 1-1 8051 单片机的特殊功能寄存器

寄存器	寄存器位								地址	说明
	D7	D6	D5	D4	D3	D2	D1	D0		
P0*	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	0x80	P0 端口
P1*	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	0x90	P1 端口
P2*	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	0xA0	P2 端口
P3*	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	0xB0	P3 端口
PSW*	CY	AC	F0	RS1	RS0	OV	-	P	0xD0	程序状态字
ACC*									0xE0	累加器
B*									0xF0	乘法寄存器
SP									0x81	堆栈指针
DPL									0x82	数据存储器地址低字节
DPH									0x83	数据存储器地址高字节
PCON	SMOD				GF1	GF0	PD	IDL	0x87	电源控制及波特率选择
TCON*	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0x88	定时器控制
TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	0x89	定时器方式选择
TL0									0x8A	T/C 0 低字节
TL1									0x8B	T/C 1 低字节
TH0									0x8C	T/C 0 高字节
TH1									0x8D	T/C 1 高字节
IE*	EA			ES	ET1	EX1	ET0	EX0	0xA8	中断允许寄存器
IP*				PS	PT1	PX1	PT0	PX0	0xB8	中断优先级寄存器
SCON*	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	0x98	串行口控制器
SBUF									0x99	串行数据缓冲

上述特殊功能寄存器分别用于以下功能单元。

CPU：ACC、B、PSW、SP、DPTR。

并行端口：P0、P1、P2、P3。

中断系统：IE、IP。

定时器/计数器模块：TMOD、TCON、T0（TH0，TL0）、T1（TH1，TL1），其中 TCON 还含有外部中断标志位及外部中断触发方式配置位。

串行通信模块：SCON、SBUF、PCON。

这些特殊功能寄存器在开发单片机 C 语言程序时将被大量使用。对于可位寻址的 SFR，例如，ACC 寄存器，它常被用于判断字节中各位的状态，要判断某字节第 3 位是否为 1，可将 ACC 看成 1 字节变量，ACC 获取该字节的值后可直接判断 ACC3 是否为 1，当然需要先有

定义 sbit ACC3 = ACC^3。

又如，程序状态字（Program Status Word，PSW）寄存器，头文件 reg51.h 已包含了该寄存器的各位定义，其部分内容如下：

```
/*  BIT Register  */
/*  PSW   */
sbit CY  = 0xD7;
sbit AC  = 0xD6;
sbit F0  = 0xD5;
sbit RS1 = 0xD4;
sbit RS0 = 0xD3;
sbit OV  = 0xD2;
sbit P   = 0xD0;
```

PSW 寄存器中的 CY 和 F0 在 C 语言程序中仍然被直接大量使用，如要将字节变量 d（假定为 10101101）由高位开始逐位串行发送，可对 d 进行 8 次左移，并逐次发送 CY，即

```
for (i = 0 ; i < 8 ; i++)
{
  d <<= 1; DQ = CY;
}
```

代码中 DQ 为某外部芯片读/写串行数据的引脚定义，这段代码利用汇编语言程序中常用的进位标志位 CY，可以方便地获取移出的各位。

表 1-1 中带有*号的特殊功能寄存器是可位寻址的，Keil\C51 下头文件 reg51.h 对 P0～P3 以外的可位寻址的各寄存器位给出了独立定义，如果要直接引用 4 个 I/O 端口的各引脚，需要在程序中用 sbit 进行单独定义，当然也可以将头文件 reg51.h（keil\c51\inc）改成 at89x52.h（keil\c51\inc\atmel），4 个接口的各引脚在该头文件中均被单独定义，如 P0.1 被定义为 P0_1，尽管其中有些定义对 8051 单片机是无效的，但这并不影响程序正常编译运行。

1.1 节“8051 单片机引脚”曾提到端口结构中的“读锁存器”与“读引脚”的问题，这些问题的解释涉及 8051 单片机的相关特殊功能寄存器。此前已经提到，凡“读-改-写”这类指令，CPU 将执行“读锁存器”，否则将为“读引脚”，下面通过实例说明这一问题。

以汇编指令 ORL P2, A 为例，它相当于 C 语言的 P2 = P2 | ACC 或 P2 |= ACC，现假设在 P2.0 引脚接有一开关，执行程序之前先合上开关使 P2.0 引脚接地，然后依次执行：

```
P2 = 0x33(即 0011 0011); ACC = 0x44(即 0100 0100); P2 = P2 | ACC; while (1);
```

其中最后的 while（1）相当于汇编指令 JMP $，它使程序执行到该位置后进入死循环，以便观察单片机接口状态。上述语句执行后，P2 端口引脚输出应为 0x77（0111 0111）还是 0x76（0111 0110）呢？

实际情况是如果开关仍然合上，通过 Proteus 仿真软件可观察到 P2 端口引脚状态为 0x76；如果开关断开，则 P2 端口引脚状态变为 0x77，显然，对于 P2 = P2 | ACC 这样的“读-改-写”语句，CPU 控制执行的是“读锁存器”，因为如果是“读引脚”，所读取的 P2 将为 0x32 而不是 0x33，从而最终结果将恒为 0x76 而不可能在断开开关时变为 0x77。

可见，CPU“读锁存器”而非“读引脚”，可避免此前输出到“端口引脚”的值可能被外部改变的情况，“读锁存器”可确保读取的是此前输出的原始值，而不是输出后被外部改变过的值。

再如指令 MOV A, P1，相当于 C 语言的 ACC = P1，CPU 显然将发出“读引脚”信号。仍假设运行程序之前开关先被合上，P2.0 接地，依次执行下述语句：

```
P2 = 0x33; ACC = P2; P2 = ACC | 0x44; while (1);
```

观察单片机 P2 端口，可知其状态为 0x76，且断开开关后仍为 0x76，可见对于 ACC = P2，

CPU 控制执行的是“读 P2 引脚”上的 0x32，而不是“读 P2 锁存器”中的 0x33。

在所设计的单片机系统中，如果接口输出值可能被外部改变，而程序要求每次均在上次输出值的基础上运算处理后输出，而不是在输出发生变化值的基础上运算处理再输出，这时就要考虑到所编写的 C 程序生成的指令是“读-改-写”类指令，还是仅为“读”类指令。

具体生成为何类指令，需要通过 Keil μVision 的反编译功能查看，而不能凭直觉判断。例如：

P2 = P2 | i 反编译后的汇编指令如下，其中 0x08 为编译器分配给变量 i 的地址：

```
C:0x001C    E508    MOV A,  0x08
C:0x001E    42A0    ORL P2, A
```

显然，由于编译生成了“读-改-写”类指令“ORL P2, A”，故 CPU 处理该语句时将执行“读锁存器”操作而不是“读引脚”操作。

再如 P2 = （P2 & 0xF8）| i，其反编译后的汇编指令如下，其中 0x08 仍为变量 i 的地址：

```
C:0x0020    E5A0    MOV A,  P2
C:0x0022    54F8    ANL A,  #0xF8
C:0x0024    4508    ORL A,  0x08
C:0x0026    F5A0    MOV P2, A
```

可见，CPU 处理该语句时执行的将是“读引脚”而不读“读锁存”操作。

对于 P0～P3 端口的“读”/“写”操作还应注意，由于初始上电时它们全部被写入“1”，故可以不需要任何初始化而直接读取端口外部数据。但如果端口曾输出过“0”，之后又需要逆向读入外部数据，此时则需要在相应端口位上先输出“1”，1.1 节“8051 单片机引脚”已经讨论过这样操作的原理。

1.4 外部中断、定时器/计数器及串口应用

使用 C 语言开发 8051 单片机程序时，除了要控制 4 个双向 I/O 端口外，还要掌握 8051 单片机外部中断、定时器/计数器及串口程序设计。对于定时器/计数器及串口功能模块，它们既可以工作于中断方式，也可以工作于非中断方式，本书针对不同的工作方式分别提供了相应案例。图 1-4 给出了 8051 单片机的中断系统结构图。

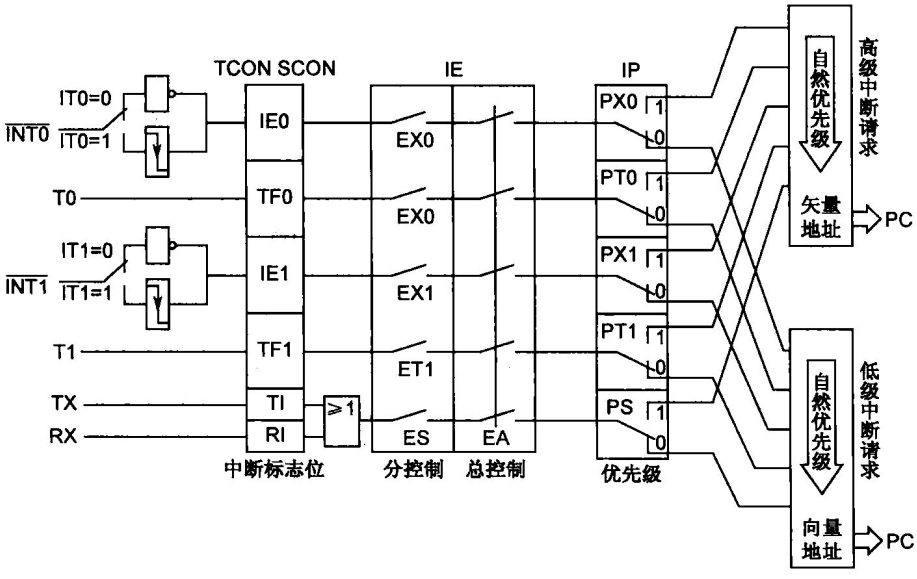


图 1-4 8051 单片机中断系统结构图