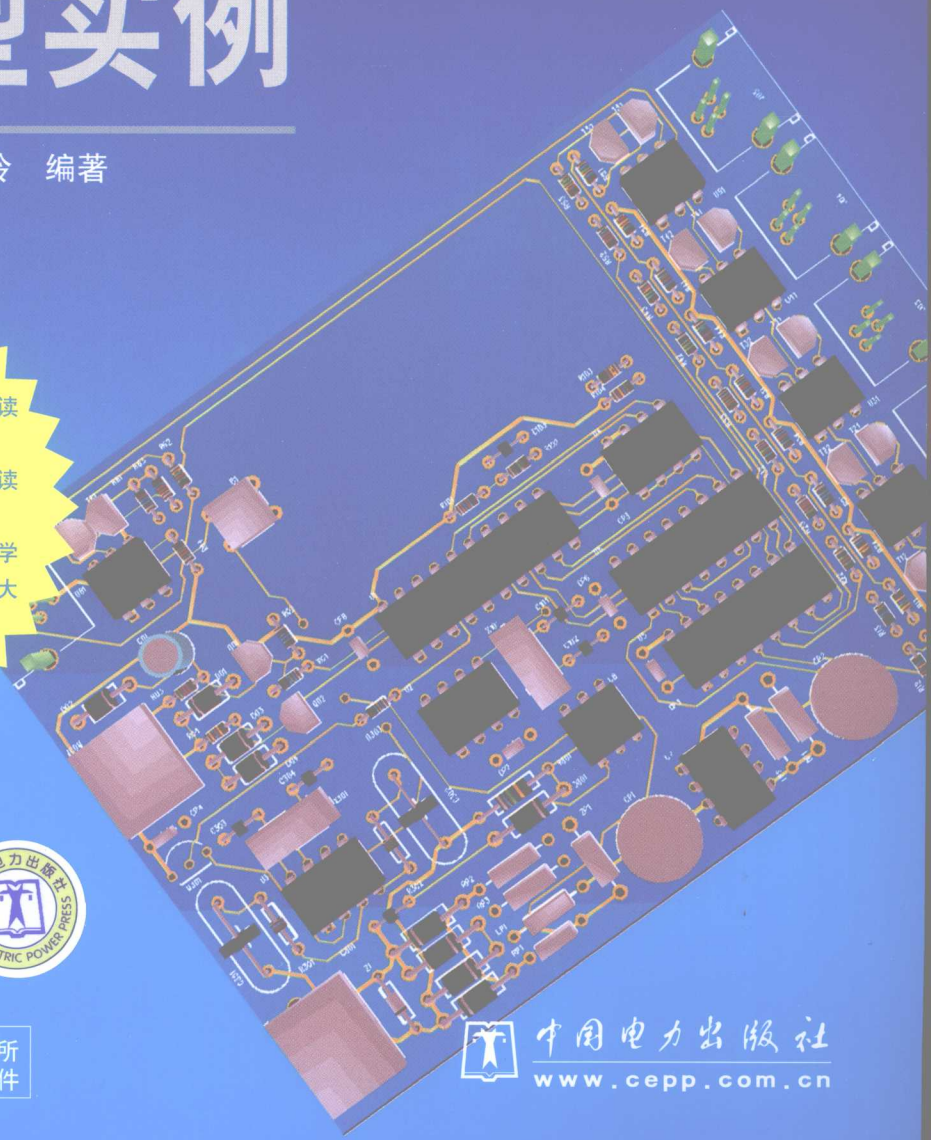
单片机C语言应用开发丛书

51单片机实用C语言程序设计与典型实例

杨国田 白 焰 董 玲 编著

- 语言文字简单、清晰，易于读者从零开始、快速掌握；
- 实例丰富，案例分析详实，读者有身临其境的感觉；
- 作者多年的实际项目和教学经验总结，真情奉献于广大读者朋友！



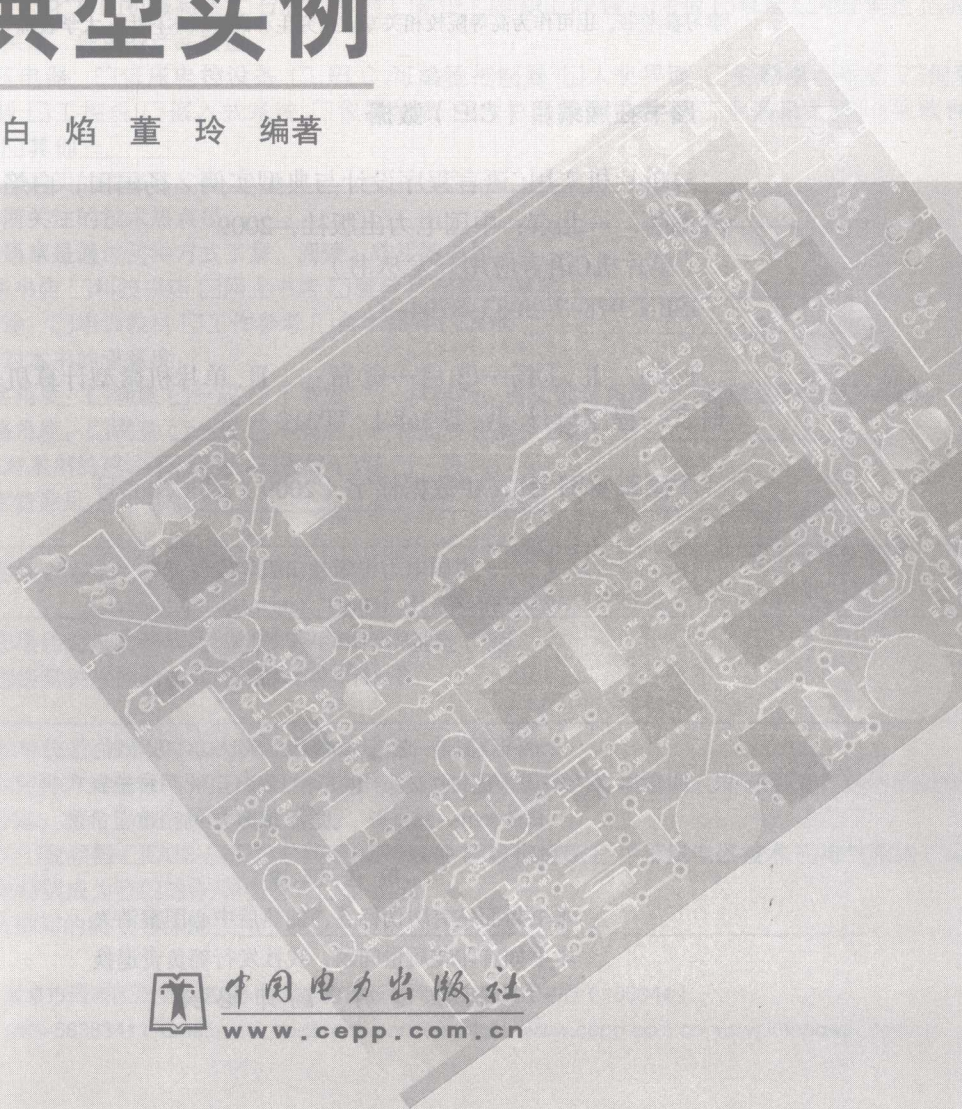
本书赠送一张光盘，内含书中所有的程序代码以及相关的文件

中国电力出版社
www.cepp.com.cn

单片机C语言应用开发丛书

51单片机实用C语言程序设计与典型实例

杨国田　白　焰　董　玲　编著



中国电力出版社
www.cepp.com.cn

内 容 提 要

本书主要介绍 8051 系列单片机的 C 语言（C51）程序设计方法、应用系统及程序设计实例，共分为 8051 系列单片机基础、C 语言基础、应用系统设计、设计案例等四个部分。其中第三部分以实例方式介绍应用系统设计的有关问题，包括人机界面设计以及常用接口器件的使用等，例如按钮、矩阵键盘、触摸屏、LCD 显示器、串行 A/D（D/A）器件、定时器、RS-232 接口等；第四部分主要介绍多个应用系统设计案例，包括红外遥控系统、射频遥控系统、电视遥控器、步进电机驱动系统、PS2/ 串口鼠标接口系统等。

本书写作时注意展示 C51 与汇编语言的内在联系，这样有利于对 C51 的自如运用，同时在案例部分给出系统设计时的种种考虑，力图使读者具有身临其境的感觉。此外，还详细介绍有关元器件参数计算的方法等，例如总线上拉电阻的阻值、电阻的额定功率核算等。

本书赠送 1 张光盘，内含书中涉及到的所有程序以及相关的素材文件。

本书可作为从事单片机及嵌入式开发的技术人员、51 单片机开发者和初学者的学习参考书，也可作为高等院校相关专业研究生和高年级本科生、大学教师等教材。

图书在版编目（CIP）数据

51单片机实用C语言程序设计与典型实例 / 杨国田，白焰，董玲编著. —北京：中国电力出版社，2009

（单片机C语言应用开发丛书）

ISBN 978-7-5083-8794-9

Ⅰ. 5… Ⅱ. ①杨…②白…③董… Ⅲ. 单片机微型计算机—C语言—程序设计 Ⅳ. TP368.1 TP312

中国版本图书馆CIP数据核字（2009）第067463号

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://www.cepp.com.cn）

北京市同江印刷厂印刷

各地新华书店经售

*

2009 年 7 月第一版 2009 年 7 月北京第一次印刷

787 毫米 ×1092 毫米 16 开本 15.25 印张 339 千字

印数 0001— 3000 册 定价 **28.00** 元（含 1CD）

PREFACE

前言

1980年，英特尔公司推出了后来广为普及的MCS-51单片机，近30年来，其衍生系列不断涌现，从Atmel加入FLASH ROM，到phillips加入各种外设，再到后来的Cygnal推出C8051F，使得以8051为核心的单片机在各个发展阶段的低端应用中始终扮演着一个独特的角色，其地位不断提升，资源日渐丰富，历经30年仍在生机勃勃地延续，甚至在SoC时代仍赫然占有一席之地。

有资料显示，8位机始终是嵌入式低端应用的主要机型，而且在未来相当长的时间里仍会保持这个势头。而8051系列，在8位单片机中形成了一道独特的风景线。历史最长、常盛不衰、众星捧月、不断更新，形成了既具有经典性又不乏生命力的一个单片机系列。可以说，Intel公司创建了8位机的经典系列结构。究其原因，嵌入式系统嵌入到对象体系中，并在对象环境下运行，与对象领域相关的操作主要是对外界物理参数进行采集、处理，对外界对象实现控制，并与操作者进行人机交互等，而这些任务所要求的响应速度有限，且不会随时间变化。在8位单片机能基本满足其响应速度要求后，数据宽度不成为技术发展的主要矛盾。因此，8位单片机将稳定下来，其技术发展方向转为最大限度地满足对象的采集、控制、可靠性和低功耗等品质要求。而对于8051系列，则由于其实施技术开放政策，使得这个系列历经沧桑而不老。

在相当长的历史时期里，单片机的开发以汇编语言为主，即使今天，高效的工作程序也大多不能完全摆脱汇编语言，这当然有其原因。而且Borland公司创始人之一，因使用汇编语言撰写编译器开发出Turbo Pascal（首创In-Memory Compiler）而震惊全世界的大名鼎鼎的Anders Hejlsberg先生用他的传奇经历也证明了汇编语言可以做大事。Anders先生使用汇编语言创造出了一度是全世界速度最快、品质一流的Pascal编译器，可见汇编语言的威力。但是，在Anders离开了Borland之后，几乎没有人能够修改Anders的编译器，足见Anders先生的汇编语言功力何以出神入化，并非常人所能企及。

然而在今天，单片机应用系统开发开始走向大型化、复杂化，开发周期要求越来越短，汇编语言被认为枯燥而难以维护和调试，而C语言成了人们热捧的开发语言，尤其近年来C语言几乎是大学新生的必修课，这虽然没有直接降低单片机系统开发的门槛，但是却抬高了门槛外的土地。

当然，C语言是一种优秀的高级语言，它精练，接近硬件，复杂程度适当，数据类型丰富、明确。自20世纪70年代初诞生以来，首先被K.Thompson和D.M.Ritchie两人用来

改写 UNIX 操作系统（UNIX 第 5 版）。1983 年，美国标准化协会（ANSI）又制定了 C 语言标准，称为 ANSI C。C 语言是一种结构化程序设计语言，用 C 语言写的程序层次清晰，便于按模块化方式组织，编译生成的代码质量非常高，易于维护和调试。C 语言的表现能力和处理能力极强，它不仅具有丰富的运算符和数据类型，便于实现各类复杂的数据结构，而且还可以直接访问物理内存，甚至对数据位（bit）进行操作，它的部分运算符可以直接与汇编语言的指令相对应，例如自身加、减 1 运算等，因此兼有高级语言和低级语言的特点。此外，C 语言还具有代码效率高、可移植性强等特点。总之，C 语言在速度上可与汇编语言媲美，在代码编写上又有高级语言的可读性，且易移植，它在程序设计思想上，给程序设计人员以很大的自主权。

20 世纪 80 年代中后期，C 语言开始向单片机上移植，现在早已成为专业化单片机应用系统研发的实用高级语言。类似 8051 这样的单片机，用 C 语言编写软件，尽管会牺牲一些效率和资源，但是好处颇多，不仅大大缩短开发周期，且能显著改善程序的可读性，便于程序的维护和改进，更容易研制出规模更大、性能更完善的系统。

C 语言虽好，但不是万能的；汇编语言虽枯燥，但是不可或缺。完全抛弃汇编语言至少现阶段是难以想象的，二者共存，以 C 语言为主，辅以汇编语言，容易得到最优化、理想的源程序。尤其对 8051 系列来说是这样。此外，无论哪种语言，使用者所达到的境界也是决定程序优劣的重要因素之一。

本书主要介绍 8051 系列单片机的 C 语言（C51）程序设计方法以及应用系统及程序设计实例，分为 8051 系列单片机基础、C 语言基础、应用系统设计、设计案例等四个部分，第一部分概要介绍 8051 系列单片机与系统设计有关的一些基础；第二部分结合实例系统介绍了 C 语言以及 C51 的相关基础；第三部分介绍应用系统设计的有关问题，包括人机界面设计以及常用接口器件的使用等，例如按钮、矩阵键盘、触摸屏、LCD 显示器、串行 A/D（D/A）器件、定时器、RS-232 接口等；第四部分是本书重点，主要介绍多个应用系统设计案例，包括系统设计及参数计算、程序设计等。

本书写作过程中是假设读者对 C 语言及 8051 系列单片机有初步的了解，因此书中只介绍了 C51 的特殊之处，对 51 系列单片机也只做简单介绍，而把重点放在应用系统设计及 C51 程序设计上，并介绍几个设计实例，包括红外遥控系统、射频遥控系统、电视遥控器、步进电机驱动系统、PS2/串口鼠标接口系统等。

作者多年来坚持对每个自己设计的 C 语言（包括 C51）程序编译后的汇编代码与 C 语言语句进行逐一对比分析，从而确保源代码质量，例如用 C 语言写的延时程序究竟延时时间多长，某些代码是否优化以及指针的使用有无问题等，到 Keil 的反汇编窗口看一下，就一目了然了。例如 Keil C51 的符号扩展算法，通过反汇编发现十分精巧，下面就是 char 转换成 int 型变量的编译结果：

```
MOV  A, R7            ;取得 char 变量（在 R7 中），送到累加器 A
RLC  A                ;最高位（补码的符号位，等于 1 表示负数）送到进位位 C
SUBB A, ACC(0xE0)     ;相当于用 0 减去进位位 C，如果 C 等于 1，结果为 0xFF，否则为 0
MOV  R6, A            ;将上步减法运算结果送到高位寄存器 R6，与 R7 一起形成 int 变量
```

上述指令序列的功能是字节到字的符号扩展，实质上就是根据字节变量的最高位来决

定在字变量的高位字节补 0 还是补 0xFF，上述四个指令恐怕是 8051 系列单片机最为简捷的符号扩展指令序列，如果通过分支语句来实现代码就要长得多。再有，有人通过查看反汇编代码确认 C51 中最简捷的延时语句为 while (--i)，它明显优于 while (i--)和 for(; i>0; --i)，前者编译后仅生成一条汇编指令。

汇编语言的这些作用也是作者认为它不该被抛弃的原因之一，初学者即使从 C51 开始入门，在不断的程序设计实践中也该逐步有意了解和熟悉汇编语言了。基于此，本书写作时注意展示 C51 与汇编语言的内在联系，这样有利于对 C51 的自如运用，同时在案例部分给出系统设计时的种种考虑，力图使读者具有身临其境的感觉。此外，还详细介绍有关元器件参数计算的方法等，这在其他文献中介绍较少，而开发人员往往凭经验或参考其他方案粗略确定有关参数，但实际上很多参数是可以通过计算确定的，例如总线上拉电阻的阻值、电阻的额定功率核算等。

目前，有关 8051 系列单片机、C51 的书籍和资料十分丰富，作者愿以自己 20 年的嵌入式系统开发的经验与教训与读者共同切磋，以期初学者少走弯路，练成高手。

全书由杨国田、白焰、董玲合作完成，此外，李新利、杨国利、董玫也参加了本书的写作及内容选定工作。

作　　者

2009 年 7 月于华北电力大学

目　录

>>> 第1章

51单片机基础

本章内容

- ☑ 51系列MCU的结构组成
- ☑ 特殊功能寄存器
- ☑ 存储器系统组成及特点
- ☑ 并行I/O接口
- ☑ 定时器/计数器组成及特点
- ☑ 中断子系统
- ☑ 串行I/O接口（UART）
- ☑ 系统扩展
- ☑ 51系列单片机的指令系统

51 系列单片机自 20 世纪 80 年代初在 Intel 公司诞生以来，迅速推广开来，得到了广大开发人员和学生的喜爱，直至今天，许多高校的课堂上仍然在讲授 51 单片机课程。在计算机技术飞速发展的年代，一个 8 位单片机，历经 20 余年，仍然在嵌入式系统研发当中广泛使用，真可以说是一个奇迹。不仅如此，51 单片机还派生了很多相同内核以及改进内核的新型号，例如大内存、带有 A/D、扩展语音处理模块、带有高速 I/O 等新产品。这种现状部分源于该系列单片机优秀的设计、低廉的价格、简便易得的开发工具。当然首先是该系列单片机即使在今天仍然能满足大量的实际需求，包括简单的语音及视频处理。

本书重点在于 C51 程序设计及应用，因此不准备系统地介绍 51 单片机的原理，而只是对 51 系列单片机的相关基础做一个简单介绍。

1.1 51 系列 MCU 的结构组成

51 系列单片机最原始的型号是 8031，内部集成 EPROM 存储器后称为 8751，后来广泛使用的主要是 89C51/89C52 等，这是 Atmel 公司生产的兼容型号，89C51 系列主要是内部的 EPROM 换成了 FLASH 存储器，可以多次反复擦写，而 89C52 则在此基础上增加一个定时器。近年 89S51/89S52 等则占据主流，它们支持串行写入，简化了开发工具和编程器。各种型号的内核及指令系统基本相同，因此这里将以 89C52 为例介绍其相关的基础。

概括起来，51 单片机的内部结构可以用图 1-1 来描述。

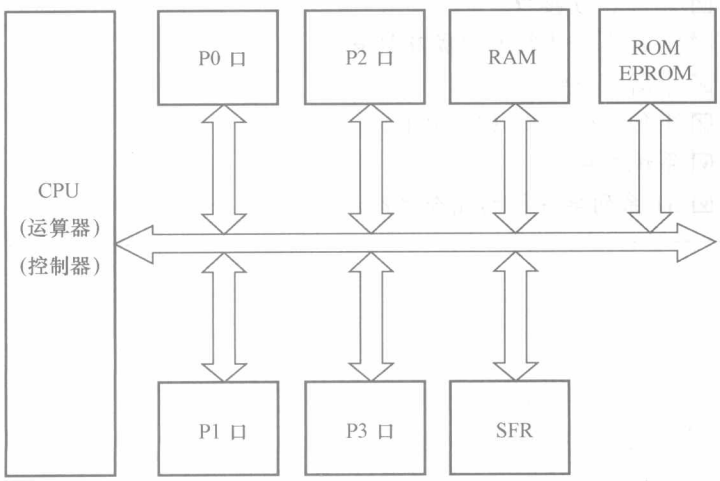


图 1-1　51 系列 MCU 的内部结构简图

51 系列单片机内部具体结构如图 1-2 所示，由 CPU、并行 I/O、串行 I/O、定时器/计数器、内部 RAM 存储器、内部 ROM 存储器、时钟震荡电路等组成，部分型号还带有其他模块。正是这些集成在内部的模块决定了它是一个单片机。

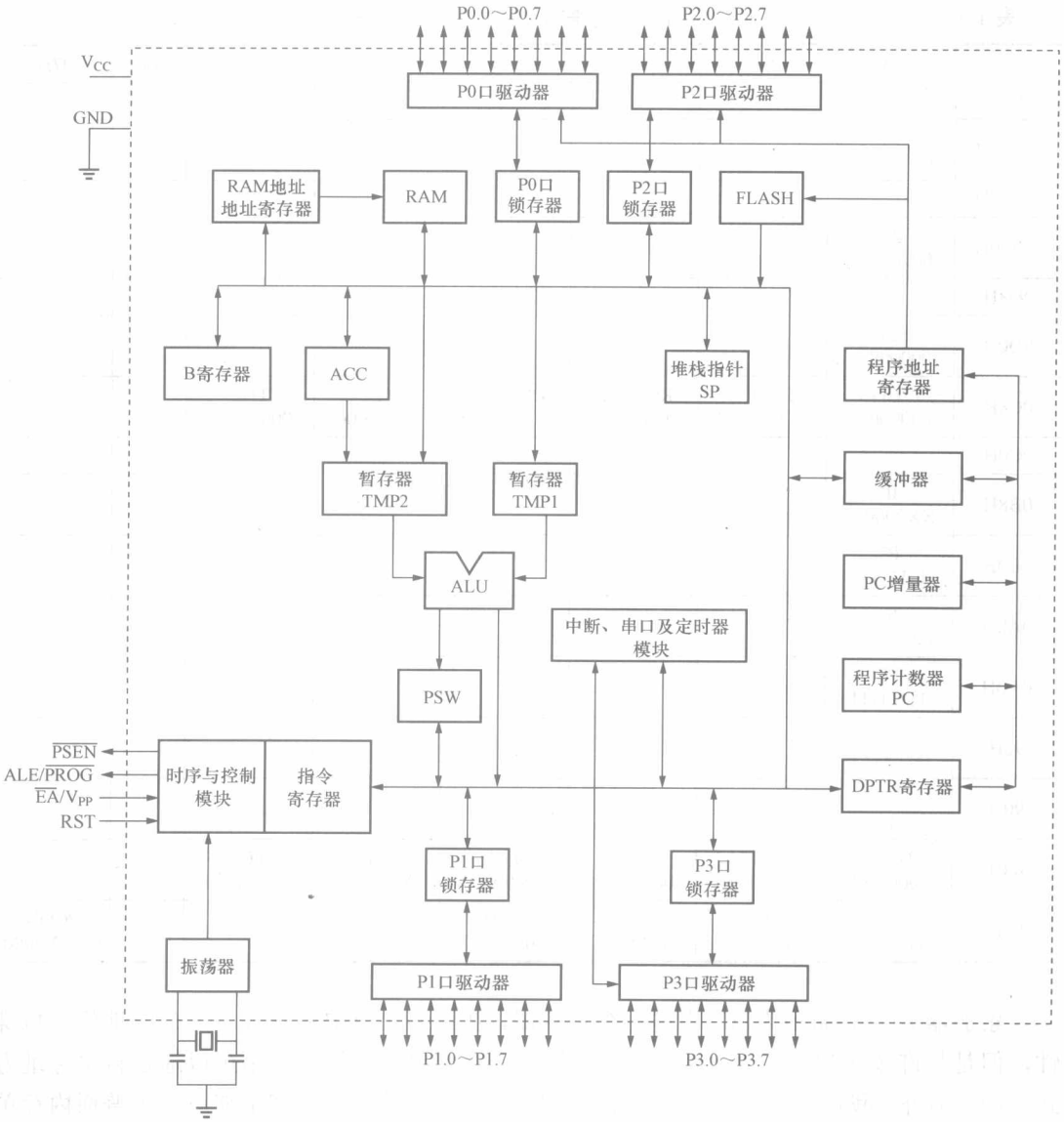


图 1-2　51 系列 MCU 的结构组成

1.2　特殊功能寄存器

51 单片机与其他单片机类似，内部集成了 I/O 接口器件，理论上讲，这些器件可以纳入数据存储器空间，作为一个或几个单元来访问，但是 51 系列 MCU 没有这么做，而是通过一系列特殊功能寄存器来访问和操作，这些寄存器占据与内部存储器高端重叠的地址空间，即 80H～FFH 之间的部分地址，但是要求必须通过直接寻址方式访问，而把间接寻址方式留给高端 RAM 使用。这意味着无法通过 R0/R1 间接寻址去访问特殊功能寄存器，事实上由于特殊功能寄存器使用上的特点，通常也不需要这么做。51 系列 MCU 特殊功能寄存器的分布见表 1-1。

表 1-1　　51 系列 MCU 特殊功能寄存器地址分布图

	00H	01H	02H	03H	04H	05H	06H	07H
0F8H								
0F0H	B 00000000							
0E8H								
0E0H	ACC 00000000							
0D8H								
0D0H	PSW 00000000							
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000		
0C0H								
0B8H	IP XX000000							
0B0H	P3 11111111							
0A8H	IE 0X000000							
0A0H	P2 11111111							
98H	SCON 00000000	SBUF XXXXXXXX						
90H	P1 11111111							
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000		
80H	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000

累加器 ACC，简写为 A，也是一个寄存器，在许多指令中具有独一无二的地位与特殊性，但是与许多 CPU 不同的是，它也占据一个存储单元地址，甚至可以通过直接寻址方式访问。另外，我们可以对累加器 A 进行逻辑移位等操作，但是却不能对一个普通内存单元进行这种操作。

寄存器 B，它也是一个只支持直接寻址方式的内存单元，但是在乘除法指令中，它却扮演一个特殊的角色，配合累加器进行运算，而且是除了累加器 A 以外，唯一可以参与乘除运算的寄存器。

程序状态字 PSW，用于辅助运算以及进行某些设定，它存在的最充分的理由是多位数运算中需要处理进位和借位，由于 CPU 内部总线宽度一般是 8 的倍数，内部资源通常具有 8 个位，于是 PSW 中剩余的空位自然就被其他功能位所占据，内部 7 个有效数据位可以分为三类，一类是运行结果特征指示位，包括 CY、AC、OV、P，具体作用参看表 1-2；第二类是寄存器体选择位，包括 RS1、RS0，用于选择/激活四个寄存器体中的一个；第三类只有一个位，即 F0，它相当于一个只有一个数据位的内存单元，不过它寄生在 PSW 内部。PSW 程序状态字各位说明见表 1-3。

除此以外，51 系列 MCU 还有四个特殊的寄存器群，即 4 套独立的 R0-R7 寄存器，它们虽然不属于 SFR，但是它们具有自己独特的特点。

首先，它们就位于内部 RAM 的最低端，每一套占据 8 个字节，因此，可以通过直接寻址、间接寻址以及寄存器名字来访问，某一具体时刻只有一套是当前寄存器体，但是指令无能力区分，只是固定访问当前寄存器体，具体哪一个是当前寄存器体，由 PSW 寄存器中的 RS1、RS0 来选择。

其次，R0、R1 两个寄存器还可以作为间接寻址寄存器，通过它们可以访问 00～FFH 之间的内部 RAM，也可以访问外部数据存储器，甚至他们自身的内容。例如：

```
MOV     A, @R0 或 MOV     A, @R1     ; 读取内部 RAM 存储器某一单元内容
MOVX    A, @R0 或 MOVX    A, @R1     ; 读取外部 RAM 存储器某一单元内容
```

因此，可以说 R0、R1 具有多种身份：数据寄存器、间接寻址寄存器、内存单元。

在 51 系列 MCU 中，这些寄存器扮演着重要角色。

表 1-2　　PSW 程序状态字的组成

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

表 1-3　　PSW 程序状态字各位说明

符　号	位　置	说　明
CY	PSW.7	进位标志位。反映最近一次运算的进位或借位
AC	PSW.6	辅助进位标志（用于 BCD 指令），D3 向 D4 的进位
F0	PSW.5	通用标志位，可供用户程序使用
RS1	PSW.4	寄存器工作体选择标志位，两个位可以在四个寄存器体之中选择一个。 RS1:RS0=00：选择 BANK0(00H～07H)；=01：选择 BANK1(08H～0FH)； RS1:RS0=10：选择 BANK2(10H～17H)；=11：选择 BANK3(18H～1FH)
RS0	PSW.3	
OV	PSW.2	溢出标志位，最近一次运算结果溢出时，该位置 1
—	PSW.1	厂家保留
P	PSW.0	奇偶校验标志位。每个指令周期由硬件置 1 或清 0，它指示累加器中 1 的个数是单数还是双数，即偶校验

1.3　存储器系统组成及特点

历史上曾经出现过存储器系统结构简单的 CPU，例如 6502，它只有一套空间，无论程序存储器、数据存储器还是 I/O 端口，都必须安排在这一套空间中，访问时所使用的指令也是完全相同的。而 51 系列则相对复杂得多，它拥有四套存储空间，其中包括最多 64KB 外部程序存储空间、64KB 外部数据空间、内部程序存储器空间、128B/256B 内部数据存储空间、特殊功能寄存器等，不同的存储空间需要采用不同的指令访问，只有内部、外部程序存储器采用相同的访问指令，事实上它们也确实占据同一个空间，这一点初学者需要

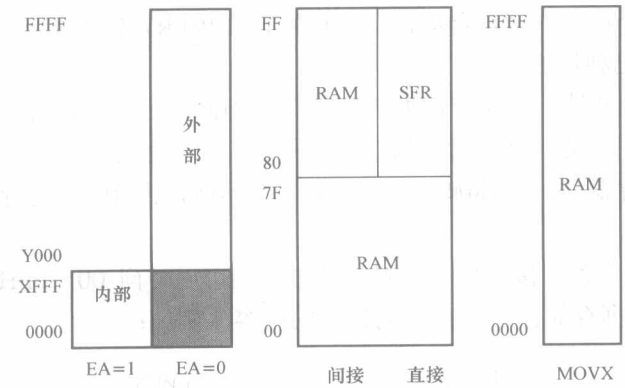


图 1-3　51 系列单片机存储空间分布情况

一个适应过程。具体的存储空间分布情况如图 1-3 所示。

一、内部 RAM 存储区

内部集成的高速 RAM 存储器，与 64KB 外部数据存储器无关，可以通过两种寻址方式访问，一是寄存器间接寻址（通过 R0、R1 寄存器），二是直接寻址（也叫直接地址）。对于 51 系列，内部的 RAM 为 128B，地址在 00H～7FH 之间，80H～FFH 之间散布着特殊功能寄存器，这些特殊功能寄存器只能通过直接寻址方式访问，但是对于 8032、8752、89C52 等型号尾数数字不为 1 的子系列，内部有 256B 字节的 RAM，地址范围为 00H～FFH，因此地址为 80H 以上的存储器，因空间与特殊功能寄存器所占据的空间重叠，必须采用寄存器间接寻址方式访问，而占据同一空间的特殊功能寄存器则仍然只能采用直接寻址方式访问。

典型的访问方式有：

RAM：对于整个内部 RAM，即 00H～FFH 之间的某个单元，可以采用间接寻址方式访问，例如指令

```
MOV  A, @R0      ; 读取内部 RAM 存储器某一单元内容
```

对于低端内部 RAM，即 00H～7FH 之间的某个单元，还可以采用直接寻址方式访问，例如指令

```
MOV  A, 61H      ; 读取内部 RAM 地址为 61H 的存储单元的内容
```

注意，该指令与下面指令的区别。

```
MOV  A, #61H     ; 立即数 61H（97）送累加器
```

后者是将立即数 61H 送到累加器 A。

如果对高端地址单元通过直接寻址方式访问，即便确实存在 RAM 存储器（例如 52 系列），CPU 也不会访问 RAM，而是直接访问相应的特殊功能寄存器，如果你访问的地址恰好没有特殊功能寄存器，那么访问的结果是无意义的。例如执行指令

```
MOV  A, 90H
```

CPU 将会把 P0 口的内容返回到累加器 A。

二、外部数据存储空间

51/52 系列拥有 64KB 完整的、独立的外部数据存储空间，该空间必须通过两条 MOVX 指令之一去访问

```
MOVX    A, @R0
```

或

```
MOVX    A, @DPTR
```

在指令执行过程中，CPU会在外部地址总线上输出16位的地址（P2输出A15～A8、P0通过锁存器输出A7～A0），范围是0000H～FFFFH，这对于第二条指令，很好理解，因为DPTR本身就是一个16位寄存器，然而第一条指令只有一个8位的R0寄存器，它只能包含8位的地址，另外8位从何而来？事实上，51系列CPU在执行这条指令时，把R0的内容作为A7～A0送到低8位地址总线即P0口，而要求指令执行前，程序必须已经在P2口放置了正确的高8位地址，即A15～A8。因此，这条指令适合进行页面（高8位地址完全相同的256个字节存储空间）访问，即需要访问同一个页面内的若干字节时，只需要改变R0，即可访问到所需要的字节。

指令对应的总线行为应该予以注意，当执行MOVX类指令时，CPU首先输出高8位地址到P2（这时它实际上是地址总线），同时将低8位地址送到P0口（这时它实际上是地址总线），然后通过ALE信号通知外部锁存器（通常是74LS373/74LS573/74HC373/74HC573等类似芯片）将A7～A0锁存输出，结果P2和锁存器的输出和起来构成A15～A0，之后，根据读还是写操作，输出RD或者WR负脉冲，从而完成一次操作。

注意51系列与8086系列不同，它没有独立的I/O空间，I/O接口器件也必须以存储器的身份与CPU通信，当然也使用前述的两条指令。这一点在分析一些51程序时也应该特别注意，类似访问外部存储器的指令，可能是访问I/O器件的。

三、内部程序存储空间

某些型号的51单片机内部集成程序存储器，目前主要是FLASH存储器，容量大小不一，通常有2KB、4KB、8KB、16KB、32KB等不同值，地址范围固定从0000H开始，不同容量则结束地址不同，例如4KB容量的到0FFFH结束，32KB容量的到7FFFH结束，余者类推。如果指令访问到超出结束地址的单元，则CPU自动转去访问外部程序存储器。指令必须放在程序空间，如果把数据（通常是表格）固化到程序存储器中，可以通过MOVC类指令提取并送到累加器A。

内部程序存储器的访问对用户是透明的，即在外部总线上没有任何反映，只有当访问的地址超出结束地址时，才会自动使用总线访问外部的程序存储器，具体总线行为可以参看下面的内容，即外部程序存储空间。

内部程序存储空间具有加密功能（也称锁定功能），即通过使用加密位，可以防止程序被读出，以保护知识产权。

四、外部程序存储空间

当内部无程序存储器或者访问超出内部结束地址的程序空间时，CPU通过总线访问外部程序存储器，外部存储器可能的空间是64KB减去内部存储空间大小，因此地址范围可能是X000～FFFFH，当内部无程序存储器时，外部程序存储器容量可达最大值64KB，地址范围0000H～FFFFH，如果内部已有4KB，那么外部有效地址范围将是1000H～FFFFH，余者类推。

访问外部程序存储器的过程在输出地址方面与访问外部数据存储器非常相似，但是控