高等学校计算机应用规划教材

嵌入式系统开发基础

——基于8位单片机的C语言程序设计（第二版）

- 嵌入式控制系统的定义和研究方法
- RISC微处理器、CISC微处理器简介
- IAP、ISP技术
- 光电隔离技术
- 常用串行芯片接口
- 人机界面设计

侯殿有　葛海淼　编著

清华大学出版社

高等学校计算机应用规划教材

嵌入式系统开发基础

——基于8位单片机的C语言程序设计

(第二版)

侯殿有 葛海淼 编著

清华大学出版社

北 京

内 容 简 介

嵌入式系统大多具有小巧轻薄的特点，程序代码不是很长，系统采用 8 位单片机即可满足要求。本书详细介绍了使用 8 位单片机和 C 语言进行嵌入式系统设计的方法。对于单片机技术的最新发展，如 RISC(Reduced Instruction Set Computing，精简指令集)微处理器、SoC(System on Chip，片上系统)设计技术、IAP(In Application Programming, “应用现场”可调试功能)和 ISP(In System Programming，在系统编程)功能也做了简单介绍。许多串行芯片工作体积小，能耗低，适合嵌入式系统，特别是掌上产品的使用要求，本书对此做了重点介绍。

嵌入式控制系统人机界面设计是进行嵌入式控制系统设计首先遇到的问题，也是难点。本书在详细介绍 LCD 显示汉字、曲线和 ASCII 码原理的基础上，给出了一个通用字模提取和建立小字库程序以及 3 种典型 LCD 显示驱动程序，这些资料对初学者和从事嵌入式开发工作的人员有很大的实用价值。

本书配套的电子课件、配套实验讲义、各章的习题答案和部分工具软件可以到 http://www.tupwk.com.cn/downpage 网站下载。

图书在版编目(CIP)数据

嵌入式系统开发基础：基于 8 位单片机的 C 语言程序设计 / 侯殿有，葛海淼 编著. —2 版. —北京：清华大学出版社，2014

(高等学校计算机应用规划教材)

ISBN 978-7-302-36957-8

Ⅰ. ①嵌… Ⅱ. ①侯… ②葛… Ⅲ. ①微型计算机－系统开发－高等学校－教材 ②C 语言－程序设计－高等学校－教材 Ⅳ. ①TP360.21 ②TP312

中国版本图书馆 CIP 数据核字(2014)第 135539 号

责任编辑：胡辰浩　袁建华
装帧设计：孔祥峰
责任校对：曹　阳
责任印制：
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62794504
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**19.25　　**字　　数：**481 千字
版　　次：2014 年 8 月第 1 版　　**印　　次：**2014 年 8 月第 1 次印刷
印　　数：1～3000
定　　价：38.00 元

产品编号：

第二版前言

从 1979 年 Intel 公司首次发布 MCS-51 单片机产品以来，已经过去 30 多年了，在这 30 多年中，单片机世界发生了翻天覆地的变化，出现了许多功能强大，能满足各种嵌入式产品需求的专用或通用嵌入式微处理器。

RISC(Reduced Instruction Set Computing，精简指令集)微处理器是在 CISC(Complex Instruction Set Computing，复杂指令集)微处理器基础上，近几年才发展起来的先进技术。

RISC 和 CISC 是目前设计制造微处理器的两种典型技术，虽然它们都试图在体系结构、操作运行、软硬件、编译时间等诸多因素中做出某种平衡，以求达到高效的目的，但采用的方法不同，因此，在很多方面差异很大。

RISC CPU 包含有较少的单元电路，因而面积小、功耗低；RISC 微处理器结构简单、布局紧凑、设计周期短、易于采用最新技术、用户易学易用。所有这些特点，都使 RISC 技术受到众多微处理器厂家的青睐。

随着半导体产业进入超深亚微米乃至纳米加工时代，在单一集成电路芯片上就可以实现一个复杂的电子系统，诸如手机芯片、数字电视芯片、DVD 芯片等。在未来几年内，上亿个晶体管、几千万个逻辑门有望在单一芯片上实现。

SoC(System on Chip，片上系统)设计技术始于 20 世纪 90 年代中期，随着半导体工艺技术的发展，IC 设计者能够将愈来愈复杂的功能集成到单硅片上，SoC 正是在集成电路(IC)向集成系统(IS)转变的大方向下产生的。SoC 是集成电路发展的必然趋势，是技术发展的必然，是 IC 产业的未来。

此外，CPU 的工作频率从最初的几 MHz，发展到现在的几十到上百 MHz。片上程序存储器容量从原来的最大 4KB ROM/EPROM 做到现在的 4KB Flash、甚至达到 64KB Flash，使得单片机可以不使用仿真器在线编程，即具有 IAP(In Application Programming，“应用现场”可调试功能)和 ISP(In System Programming，在系统编程，无须将存储芯片从嵌入式设备拔出即可对其编程，简称 ISP)功能。

许多单片机片上集成了WDT技术(watchdog，看门狗定时器)、SPI(Serial Peripheral Interface，串行外设接口)、A/D转换电路，使单片机无须扩展就是一个嵌入式开发系统。

还有的单片机采用多核结构，提高了浮点运算速度，增强了数字信号处理能力。

单片机工作电压范围更宽，出现了工作电压 2.2~6V 的微处理器，特别适合电池供电的手持式嵌入式产品。

在专用微处理器方面也出现了许多新产品。例如，日本脉冲马达株式会社(npm，nippon pulse motor co.，ltd)的 pcl6045b 运动控制微处理器芯片，是一种通过总线接收命令、并产生脉冲控制步进电机或脉冲驱动型伺服电机的 cmos 专用微处理器，可以广泛应用于数控机床、机器人等数控机械的运动控制中。

德州仪器公司 TI 的 MSP430，为高整合、高精度的微处理器系统，是目前具有最低功耗的 flash 16-bits RISC 微控制器。可方便地实现心电信号的采集、处理、存储和打印。

美国 Veridicom 公司的 FPS100 固态指纹传感器是一种直接接触的指纹采集器件，它是一种低功耗、低价格、高性能的电容式指纹提取微控制器。

这方面例子很多，不一一列举。

面对如此众多的单片机产品，初学者如何去学习和掌握呢？我们研制一个嵌入式产品如何选择做为核心的微处理器呢？

我们知道，单片机产品种类繁多，功能各异，但它们大多是从 MCS-51 单片机基础上发展而来，只是在功能上进行了增减，指令系统基本相同，因此，我们只要掌握了 MCS-51 单片机的硬件结构和软件编程，遇到其他处理器就比较容易处理了。

对于单片机选型，要综合考虑其功能指标。主频过高，当然运算速度快，但主频过高系统耗能就高，电磁干扰加大，且容易发热，导致系统稳定系数下降，因此，运算速度满足要求即可，不必盲目追求主频过高产品。

现在出现了许多串行芯片，这些芯片一般体积小，能耗低，适合嵌入式系统小、巧、轻、灵、薄的使用要求，特别是掌上产品的要求。但串行芯片软件编程比并行芯片复杂，时序要求严格，特别是运行速度比并行芯片慢，因此，到底使用哪种芯片要综合考虑。

虽然 8 位单片机功能很强，可以处理大多数嵌入式问题，但对于多媒体、多线程、与 Internet 的连接、图像处理等较复杂问题就比较困难，此时，应选择功能更强的单片机，如以 ARM 为代表的 32 位单片机，它有多种型号，以适应不同的应用场合。

嵌入式控制系统人机界面设计是我们进行嵌入式控制系统设计首先遇到的问题，本书对此进行了详细讨论，这部分内容是本书的重点。

本书资料主要来源是：多年 MCS-51 单片机 C 语言程序设计教学中使用的资料、作者在 20 多年科研工作和指导学生参加各种大赛中积累的经验、程序、网上资源、一些公司的产品使用说明书或技术资料。

作者对使用或借鉴资料的个人或公司表示感谢。本书由侯殿有编写 1~11 章；葛海淼编写 12~17 章。

对本书的意见和建议请与我们联系，联系信箱是 huchenhao@263.net，联系电话是 010-62796045。

设计人机界面，要提取字模建立小字库，本书通用字模提取程序功能强大、使用方便，可从 http://www.tupwk.com.cn/downpage 网站下载，使用密码是 194512125019。

目　　录

第1章　嵌入式控制系统概论

嵌入式控制系统设计是当前 IT 行业最热门的话题之一。什么是嵌入式控制系统？它们如何分类？各类系统的设计方法又有什么不同？本章将结合最常用的 MCS-51 单片机来一一回答这些问题。

1.1　单片机和嵌入式控制系统的定义、嵌入式系统的分类

单片机就是在一片半导体硅片上，集成了中央处理单元(CPU)、存储器(RAM/ROM)和各种 I/O 接口的微型计算机。这样一块集成电路芯片，具有一台微型计算机的功能，因此被称为单片微型计算机，简称单片机。

有些单片机功能比较齐全，我们称之为通用单片机；有些单片机则是专门为某一应用领域研制的，突出某一功能，例如专门的数控芯片、数字信号处理芯片等，我们称之为专用单片机。有时我们也把这两种单片机统称为微处理器。

单片机主要应用在测试和控制领域，由于单片机在使用时，通常处于测试和控制领域的核心地位，并嵌入其中，因此我们也常把单片机称为嵌入式控制器(Embedded MicroController Unit)，把嵌入某种微处理器或单片机的测试和控制系统称为嵌入式控制系统(Embedded Control System)。

在本书后面的叙述中，单片机和嵌入式控制器的意义是相同的。

嵌入式控制系统在航空航天、机械电子、家用电器、自动控制等各个领域都有广泛的应用，特别是家用电器领域更是嵌入式控制系统最大的应用领域，MP3、MP4、MP5、数码相机、扫描仪、个人 PC、车载电视、DVD、PDA(掌上电脑)等，到处都可以看到嵌入式控制系统的应用。

随着超大规模集成电路工艺和集成制造技术的不断完善，单片机的硬件集成度也在不断提高，已经出现了能够满足各种不同需要、具有各种特殊功能的单片机。在 8 位单片机得到广泛应用的基础上，16 位单片机和 32 位单片机也应运而生，特别是以 ARM 技术为基础的 32 位精减指令系统单片机(RISC Microprocessor)，由于其性能优良、价格低廉，大有取代 16 位单片机而成为高档主流机型的趋势。

嵌入式控制系统由于其内核嵌入的微处理器不同，在应用上大致可分为两个层次：在系统简单、要求不高，成本低的应用领域，大多采用以 MCS-51 为代表的 8 位单片机；随着嵌入式控制系统与 Internet 的逐步结合，PDA、手机、路由器、调制解调器等复杂的高端应用对嵌入式控制器提出了更高的要求，在少数高端应用领域，以 ARM 技术为基础的 32 位精减指令系统单片机得到越来越多的青睐。嵌入式控制系统的高端应用领域又分为代嵌入式操作

系统支持和不代嵌入式操作系统支持两种情况。

1.2　MCS-51 单片机在嵌入式控制系统中的地位和作用

1980 年，Intel 公司在 MCS-48 单片机基础上推出了 MCS-51 单片机，MCS-51 单片机包括 3 个基本型 8031、8051、8751，还包括 3 个 CMOS 工艺的低功耗型 80C31、80C51、87C51。

虽然它们是 8 位单片机，但是它们品种多、兼容性好、功能强、价格低廉、性能稳定和使用方便，特别是设计和应用资料齐全，因此受到广大工程技术人员的青睐，成为我国应用最为广泛的机种。在今后相当长的一段时间，MCS-51 单片机还是嵌入式控制系统的主流机型。

由于 MCS-51 单片机技术先进，性能稳定，世界上许多大的半导体公司也在根据 Intel 公司技术生产 MCS-51 单片机或改进型 MCS-51 单片机。因此，MCS-51 单片机已经成为 8 位单片机的实际技术标准，也是嵌入式控制系统中使用最多的嵌入式控制器。

在计算机技术飞跃发展的今天，16 位和 32 位单片机已经出现并逐步得到推广应用，但 MCS-51 单片机的应用依然非常广泛。MCS-51 单片机的设计思想在 16 位和 32 位单片机中得到了进一步的继承和发展。

我们掌握了 MCS-51 单片机的 C 语言程序设计方法，可以完全满足一般嵌入式控制系统的设计要求，因为嵌入式控制系统大多具有小、巧、轻、灵、薄的特点，中小简单系统占嵌入式控制系统的绝大多数，少数高端应用我们遇到的较少。另外，掌握了 8 位嵌入式控制系统的设计方法也为进一步学习 16 位和 32 位嵌入式控制系统打下基础。

1.3　嵌入式控制系统的研究方法

1.3.1　交叉编译环境 Keil C

作为嵌入式控制器的单片机，不管是 8 位单片机，还是 16 位单片机或是 32 位单片机，由于受其本身资源限制，其应用程序都不能在其本身开发，要开发其应用程序，还需要一台通用计算机，如常用的 IBM-PC 机或兼容机，Windows 95/98/2000 或 XP 操作系统，16MB 以上内存，20MB 以上硬盘存储空间(运行交叉编译环境 Keil C 的最低配置)。我们也称这台通用计算机为“宿主机”，称作为嵌入式控制器的单片机为“目标机”，应用程序在“宿主机”上开发，在“目标机”上运行。“目标机”和“宿主机”之间利用计算机并口或 USB 口通过一台叫做“仿真器”的设备相连，编译好的计算机可以识别的目标程序(二进制代码程序)可以从“宿主机”传到“目标机”，这也叫程序下载，也可以从“目标机”传到“宿主机”，叫程序上传。应用程序通过“仿真器”的下载和上传，在“宿主机”上反复修改，这个过程叫做“调试”。调试好的应用程序，在“目标机”上编译成“宿主机”可以直接执行的机器码文件，通过一台叫 “固化器”的设备下载并固化到“目标机”的程序存储器中(8 位单片机常用的程序存储器是 EPROM 或 Flash)，整个下载过程，叫做烧片，也叫做程序固化。

程序固化是单片机开发的最后一步，以后“宿主机”和“目标机”就可以分离，“宿主机”的任务完成。“目标机”可以独立执行嵌入式控制器的任务。嵌入式控制系统开发过程如图 1-1 所示。

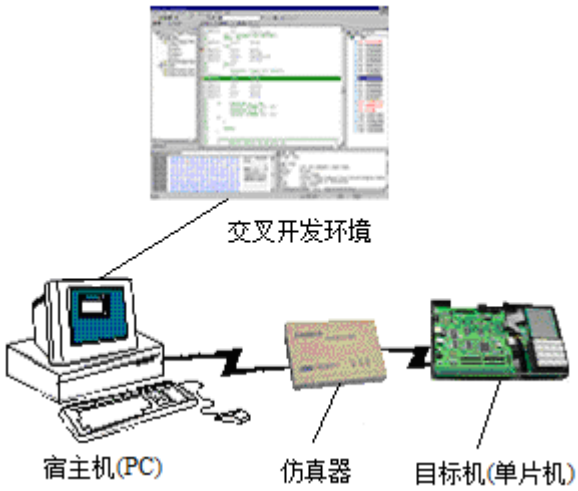


图 1-1　嵌入式控制系统开发过程

通过以上叙述可知，在“宿主机”上运行的开发工具软件的功能非常重要，我们也称这套开发工具软件为交叉编译环境或集成开发环境，交叉编译环境首先应具有类似“Word”的功能，对我们用 C 语言编写的程序进行编辑，同时它还具有调试和编译功能，可以把调试好的应用程序编译成“目标机”可以直接执行的机器码文件。

在我国，MCS-51 单片机的开发大多使用德国 Keil 公司的 μ Vision2/3 或南京伟福的 Wave6000，μ Vision2/3 也叫 Keil C51，是一款非常优秀的 MCS-51 开发工具，它功能强大、使用方便，特别是运行稳定、抗干扰和防病毒能力强，给使用者留下了深刻印象。

在清华大学出版社网站http://www.tupwk.com.cn/downpage上可下载本书免费学习参考资料，内有Keil C，供读者下载学习使用。Wave6000可以从南京伟福官方网站http://www.wave-cn.com免费下载。

1.3.2　Keil C51 的安装

打开单片机编译器文件夹，再打开 setup 子文件夹，出现图 1-2 所示的画面，双击 setup.exe 文件，出现如图 1-3 所示的选择安装类型对话框，第一次安装，选择第一项。单击 Next 按钮，出现如图 1-4 所示的选择安装版本对话框，选择 Full Version，系统就开始安装，确定安装路径“C:\Keil”和同意版权协议后，系统还要求输入产品系列号，系列号在 UP51V701.TXT 文件中。

接着我们在如图 1-5 中单击 Browse 按钮，在上一级文件夹中找到 PK51 专业开发软件路径 C51addon 文件夹，选中并确定，出现如图 1-6 所示的画面，继续单击 Next 按钮即可一步步完成安装。

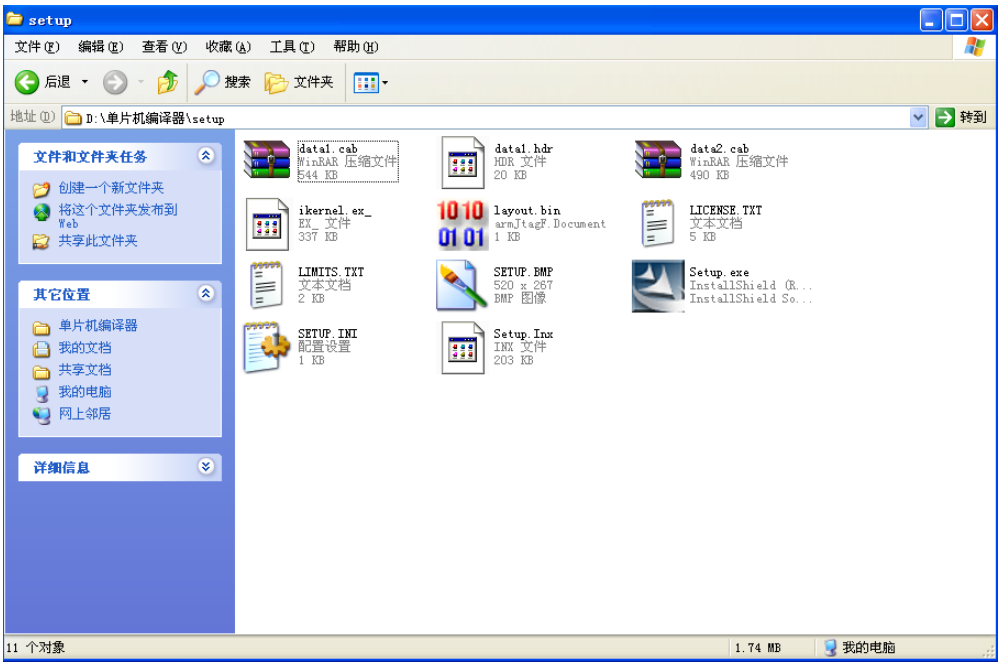


图 1-2　Keil C 安装初始画面

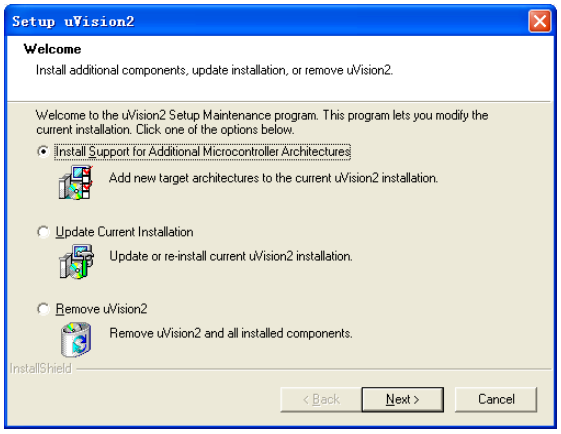


图 1-3　选择安装类型

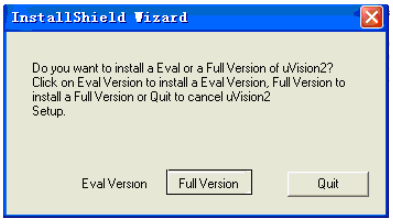


图 1-4　选择安装版本

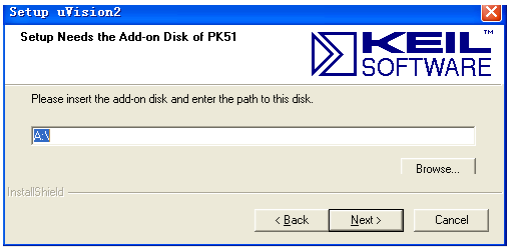


图 1-5　安装 PK51 专业开发软件

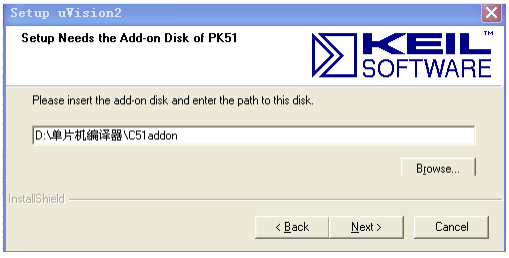


图 1-6　PK51 专业开发软件路径

1.4　程序的编辑、编译、调试和运行

1.4.1　建立项目

MCS-51 单片机程序的编辑和调试均在交叉编译环境 Keil C51 中完成，非常方便。Keil C 的快捷方式图标如图 1-7 所示，可以把它放在桌面上以方便使用。双击该图标，就会出现图 1-8 所示的交叉编译环境 Keil C 的主界面，MCS-51 单片机程序的编辑和调试均在此界面中完成，今后我们会经常在此界面上工作。

图 1-7　Keil C51 的快捷方式图标

Keil C 对程序的编辑、编译和调试，都是以“项目”为单位来进行的，在一个项目中可以包含后缀为.C 的 C 语言源文件、后缀为.h 的 C 语言头文件、后缀为.A 的汇编语言文件、后缀为.o 的机器码文件(C 语言文件经编译后形成的文件)、后缀为.LIB 的库文件(一个库文件中保存同一类功能的一些文件，这些文件又可以是后缀为.C 的 C 语言源文件、后缀为.h 的 C 语言头文件、后缀为.A 的汇编语言文件、后缀为.o 的机器码文件，还可以是另一个后缀为.LIB 的库文件)。

Keil C 在对“项目”进行编辑时，会根据每一个程序的不同后缀名调用不同的编译工具分别把它们转换为后缀为.o 的机器码文件，然后再调用链接工具文件 Link 根据“项目”结构把它们链接成一个统一的后缀为.exe 的可执行文件。

因此，使用 Keil C 进行嵌入式控制系统程序开发时，首先要新建一个项目，在开发环境主菜单中，选择 Project | New Project 命令，就会出现如图 1-9 所示的“新建项目”对话框，我们给项目起个名字：HELLO，名字的后缀 Uv2 是系统自动加的，表示这是一个 Keil C 项目。

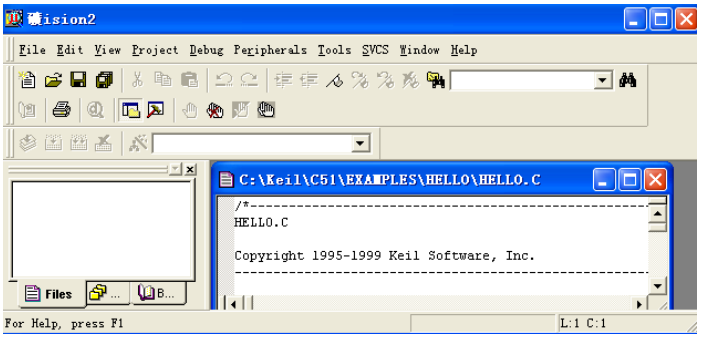


图 1-8　Keil C 的集成开发环境

选择好保存路径，单击“保存”按钮即可新建项目。之后出现“选择设备”对话框，如图 1-10 所示，就是让我们为项目选择一款单片机，例如我们选择 Intel 公司的 8031AH，就会

出现该设备的描述信息，如图 1-11 所示，确认后返回主界面，即完成了新建项目的工作。

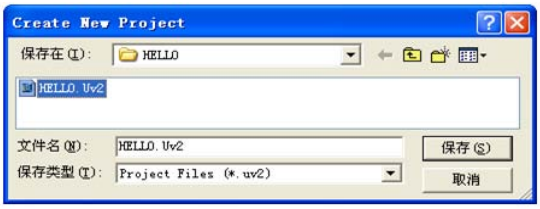


图 1-9　新建 Keil C 项目

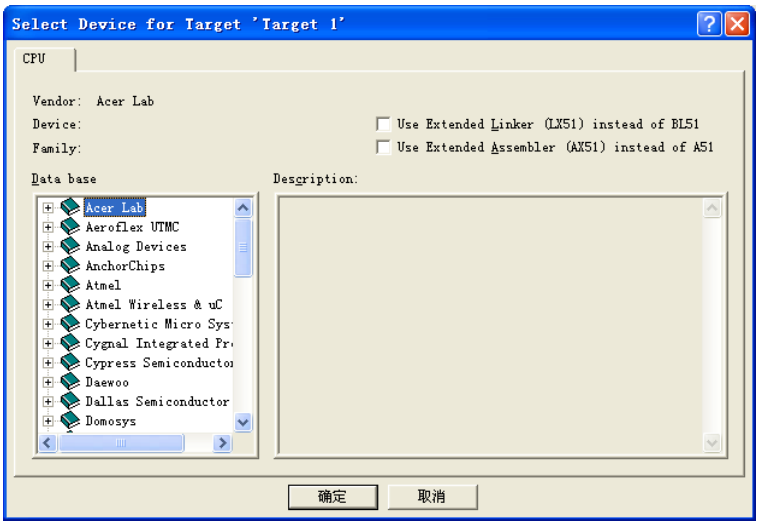


图 1-10　“选择设备”对话框

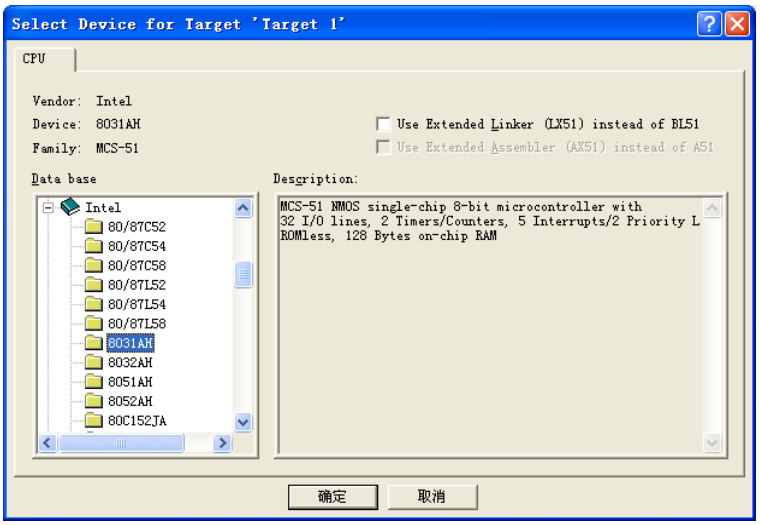


图 1-11　设备功能描述信息

1.4.2　项目的运行模式

为了使项目能够在最优化情况下运行，Keil C 对项目进行了分类，编译后代码小于 2K 的

项目为小模式(Small 模式)，其他为中模式(Compact 模式)或大模式(Large 模式)。

单片机虽然功能很强，但本身资源毕竟有限，特别是片上数据存储器，只有 128B(51 系列)或 256B(52 系列)，有时候我们必须要在片外对数据存储器进行扩展。在项目为小模式时，只使用片上数据存储器就满足系统要求了，程序用到的变量或函数调用时用到的参数可以放在片上数据存储器器中，这种情况下项目占用系统资源少、运行速度快。

虽然小模式占用系统资源少、运行速度快，但代码容量太小，在工程上一般采用大模式(Large 模式)。大模式允许数据存储器和程序代码分别为 64K，完全可以满足嵌入式控制系统的要求。中模式实际使用比较少，本书不做介绍。我们后面的例子程序均采用大模式。

在编译项目之前，要先确定使用的模式，可按如下步骤进行：

在主界面中，右击 Target1(对象 1)，从弹出的快捷菜单中选择“Options for target‘target1’”命令，打开“对象 1 设置”对话框，设置 Memory Model 为大模式：“Large Variables in XDATA”(大模式，变量放片外数据存储器)，如图 1-12 所示。

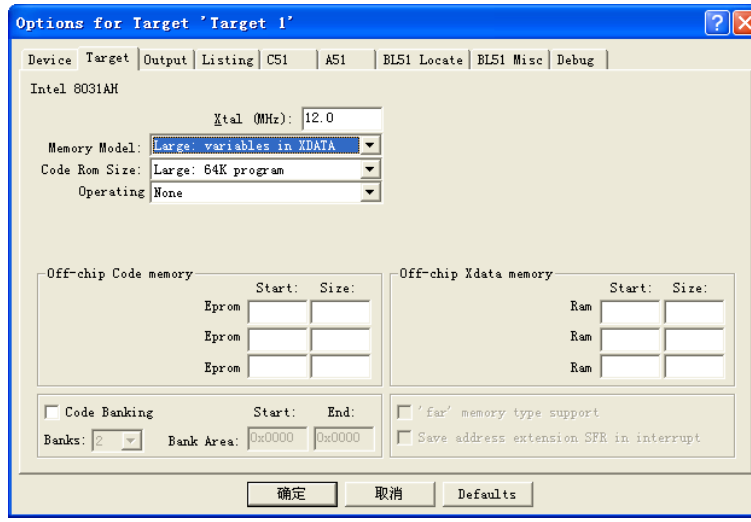


图 1-12　编译选大模式对话框

1.4.3　项目的编译模式

前面已经讲过，Keil C 可以将项目编译成后缀为.o 的机器码(也叫目标码)文件，也可以将多个目标码文件通过 Link 链接成一个后缀为.exe 的可执行文件。

如果将项目编译成后缀为.o 的机器码文件后不能直接执行，那么还要和其他后缀为.o 的机器码文件通过 Link 链接成一个后缀为.exe 的可执行文件才可执行。

一般我们的项目都较小，希望将项目编译成后缀为.o 的机器码文件后，编译器直接调用 Link 将其链接成后缀为.exe 的可执行文件，这可按如下步骤进行。

在“Options for target‘target1’”(对象 1 设置)对话框中，选中 Output 选项卡，然后选中 Debug Information 和 Create Hex Files 复选框即可，如图 1-13 所示。至此，为项目选设备和该项目编译器设置完成。

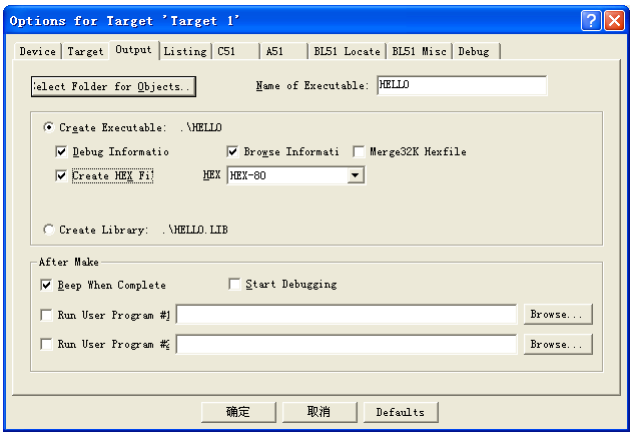


图 1-13　产生 16 进制输出文件对话框

1.4.4　项目的调试

在主界面中，右击 Source Group1，将出现如图 1-14 所示的快捷菜单，选择 Add Files to Group 'Source Group1'命令，把 C 语言源文件、头文件或汇编源文件、机器码文件、库文件加入项目中。

现在我们打开一个已建好的项目，简述一下程序的调试过程。

在主界面中选择 Project | Open Project 命令，在对话框中打开 C:/Keil/C51/EXAMPLES/HELLO.Uv2 项目，就会出现如图 1-15 所示的程序界面。界面分 4 部分，最上面是主菜单和工具栏，左面是项目工作区，显示了项目结构、帮助文档资料等。中间部分则是程序编辑区，在项目工作区双击某个文件，该文件就会在程序编辑区打开，借助于主菜单(主要是 Edit)和工具栏就可以对该文件进行编辑，Keil C 的编辑器功能非常强大，类似于小型 Word。

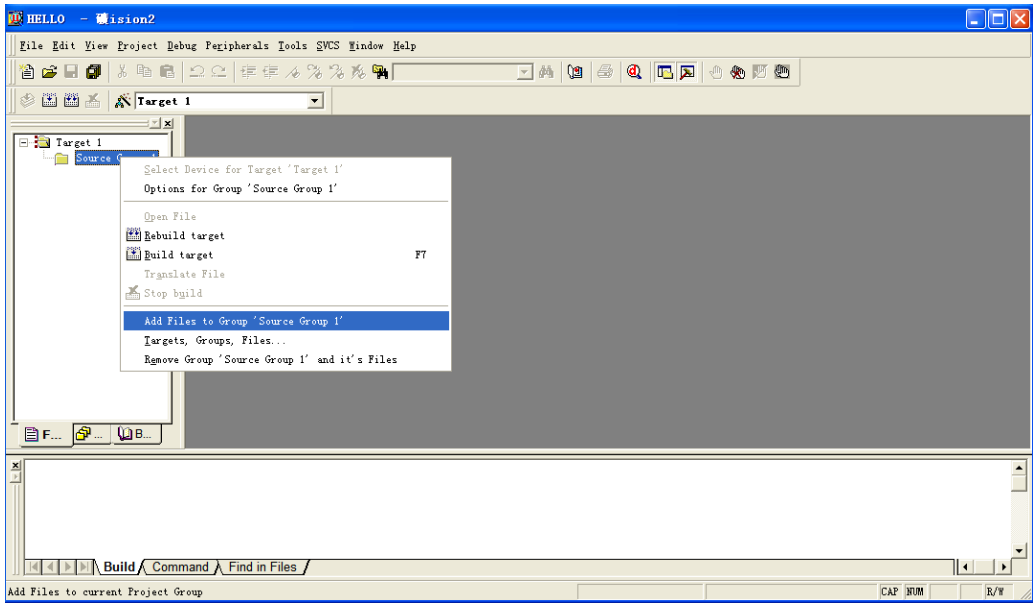


图 1-14　在项目中添加文件对话框

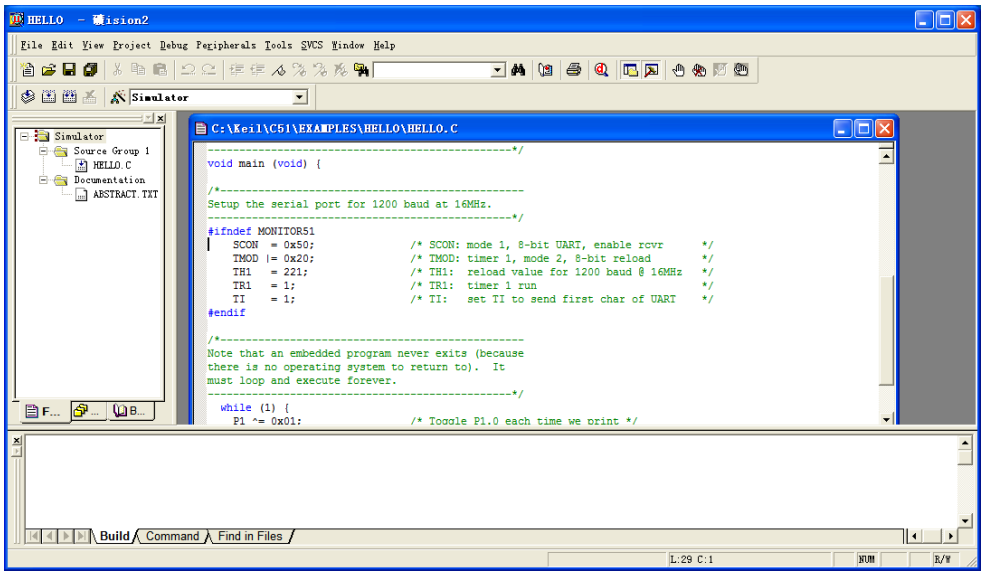


图 1-15　程序的编辑环境

程序编辑好以后，通过  3 个工具栏按钮可以将当前正在调试的文件进行编译，或链接形成机器可执行的 exe 文件。其中，  只将当前正在调试的文件进行编译，  仅对修改过的文件进行编译，它们只生成目标文件，并对项目中的每个文件进行语法检查，如果有错误会在输出栏中给出提示。  对全部文件进行编译、链接，形成机器可执行的 exe 文件。在编译过程中我们项目如果较小，常直接单击  按钮来加快编译速度。

形成 exe 文件后，还要对 exe 进行调试，反复修改，才能最后形成正确程序。

exe 文件的调试也在此环境中完成，单击工具栏按钮图标  ，会出现如图 1-16 所示的 exe 文件调试界面。

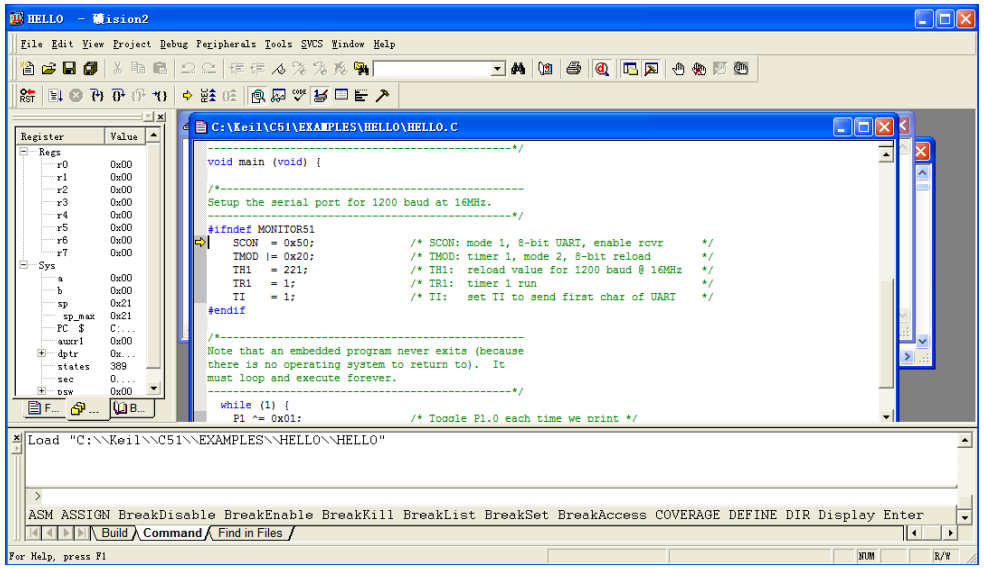


图 1-16　EXE 文件调试环境

调试环境的主要功能有连续执行程序、单步执行程序并进入函数内部、单步执行程序不进入函数内部，仅把函数当成一条语句、执行到光标处等功能，还可以在程序运行中对某些变量和存储器跟踪观察、显示反汇编结果等，如图 1-17 所示。

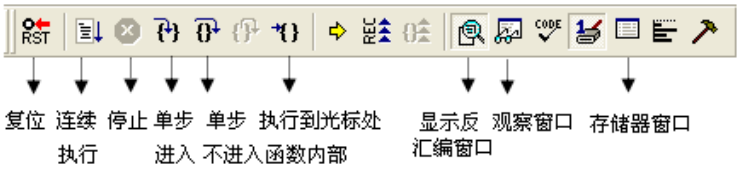


图 1-17　调试环境的主要功能

还可以在程序一处或多处设置断点，使程序执行到断点处停止，也可以取消一处或多处已设置的断点，还可以在程序运行中对中断、I/O 口、串口、定时器状态进行观察，如图 1-18 所示。

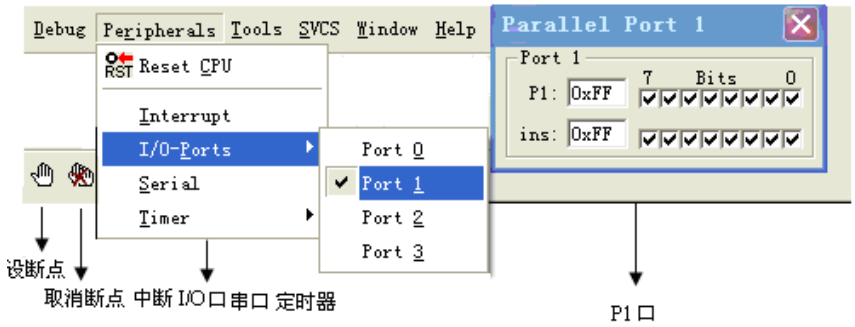


图 1-18　调试环境的其他功能

嵌入式开发基础是一门实践性非常强的课程，在强调理论课学习的同时，必须安排一定的实验课，如：广州周立功公司DP-51PRO单片机仿真实验仪，南京伟福公司的LAB6000 都是比较好的教学实验系统。

1.5　系统软件资源

Keil C 除了给我们提供非常丰富的编辑和编译工具外，还给我们提供了一些非常宝贵的库函数，这些库函数是以头文件的形式给出的。

每个头文件中都含有几个常用的函数，如果要使用其中的函数，可以采用预处理命令 #include 将有关的头文件包含进来。

使用库函数可以大大简化用户的程序工作从而提高编程效率，由于 MCS-51 系列单片机本身的特点，某些库函数的参数和调用格式与 ANSIC 标准有所不同。如果在调用一个函数过程中又出现了直接或间接调用该函数本身，则称之为函数的递归调用。并不是所有的函数都可以递归调用，我们称能进行递归调用的函数具有再入属性(reentrant)。

为了节省篇幅，我们只列出几个常用的库函数做简单介绍，更详细的内容读者可以参见

c:\keil\c51\INC 文件夹。

1. 寄存器库函数 reg51.h/reg52.h

在 reg51.h 头文件中定义了 MCS-51、reg52.h 头文件中定义了 MCS-52 的所有特殊功能寄存器和寄存器相应的位，定义时都用大写字母。当在程序中包含了寄存器库函数 reg51.h 或 reg52.h 后，就可以直接在程序中使用 MCS-51 或 MCS-52 的特殊功能寄存器和寄存器相应的位，引用时都用大写字母。

因为我们经常使用 MCS-52 的兼容机型 89C52，所以引用 reg52.h 即可。

2. 字符函数 ctype.h

ctype.h 头文件中包含十几个函数，功能有：检查参数字符是否为英文字母、数字字符、参数字符是否在 0x00~0x7f 之间、是否为可打印字符、参数字符是否为标点、空格和格式字符、参数字符是否大写或小写英文字母、检查参数字符是否为十六进制数字、大写或小写英文字母互相转换等。

3. 一般输入/输出函数 stdio.h

c51 库中包含的输入/输出函数都在 stdio.h 文件中，stdio.h 库中的所有函数都依赖 MCS-51 的串行口，使用 stdio.h 库中的所有函数时，串口必须进行初始化。例如，以 2400 波特率(时钟频率为 12mhz)，初始化程序为：

```
SCON=0x52;
TMOD=0x20;
TH1=0xf3;
TR1=1;
```

当然也可以用其他波特率。关于串口驱动程序将在后面介绍。

4. 内部函数 intrins.h

头文件 intrins.h 中包含 4 个函数，功能有：变量 var 循环左移 n 位、变量 var 循环右移 n 位、产生一个 MCS-51 单片机的 nop 指令(时间和主频有关，常用做短延时)、对字节中的一位进行测试等。

5. 标准函数 stdlib.h

头文件 stdlib.h 中包含 8 个函数，功能有：将字符串 string 转换成浮点数值或长整型数值、返回 n 个具有 len 长度的内存指针，如果无内存空间可用，则返回 null，对被 callon、malloc 或 realloc 函数分配的存储器区域进行初始化等。

6. 字符串函数 string.h

头文件 string.h 中包含 17 个函数，功能有：复制字符串、搜索字符串、比较字符串等。

7. 数学函数 math.h

头文件 math.h 中包含 6 个函数，功能有：计算绝对值，计算 e 为底的 i 的幂，随机数发生器，计算余弦、正弦、正切值，计算反余弦值、反正弦值、反正切值，计算双曲余弦值、双曲正弦值、双曲正切值等。

8. 绝对地址访问函数 absacc.h

头文件 absacc.h 中的函数功能有：CBYTE 以字节形式对 CODE 区寻址，DBYTE 以字节形式对 DATA 区寻址，PBYTE 以字节形式对 PDATA 区寻址，XBYTE 以字节形式对 XDATA 寻址，CWORD 以字形式对 CODE 区寻址，DWORD 以字形式对 DATA 区寻址，PWORD 以字形式对 PDATA 区寻址，XWORD 以字形式对 XDATA 寻址。例如 XBYTE[0x0001]是以字节形式对片外 RAM 的 0x0001 单元访问。

当 CPU 对某存储器或外围设备进行访问时，都必须通过绝对地址访问函数 absacc.h 来进行。

绝对地址访问函数的使用将在后面介绍。

1.6 习　　题

1. 什么是单片机？8 位单片机和 32 位单片机的典型机型各是什么机器?
2. 什么是嵌入式控制系统？它有哪些应用?
3. 什么是交叉开发环境？MCS-51 单片机的交叉开发环境有哪两种?
4. 什么是“宿主机”？ 什么是“目标机”？
5. 简述嵌入式控制系统的开发过程。
6. 举几个身边的实例，说明嵌入式控制系统的应用？
7. 从网上下载某版本的 Keil C51，安装在用户的个人电脑中。
8. 把 Keil C 的快捷方式放在桌面上，双击运行 Keil C51，在主菜单中选择 Project | Open Project 命令，打开 C:/Keil/c51/examples/hello 项目，初步了解和认识 Keil C 和程序调试过程。

第2章　MCS-51单片机系统和系统扩展

单片机产品种类繁多，功能各异，但它们大多是从 MCS-51 单片机的基础上发展而来，只是在功能上进行了增减，指令系统基本相同，因此，我们只要掌握了 MCS-51 单片机的硬件结构和软件编程，遇到其他处理器就比较容易处理了。本章将介绍 MCS-51 单片机的硬件结构，关于 MCS-51 单片机的软件编程将在后面章节介绍。

2.1　MCS-51 系列单片机

MCS-51单片机是美国 Intel 公司于1980年推出的高性能8位单片机，它有51和52两个系列。

对于 51 系列，主要有 8031、8051、8751 三种机型，它们的引脚和指令系统完全相同，只是片内程序存储器的容量不同，8031 芯片内部没有 ROM，8051 芯片内部有 4KB 的 ROM，8751 芯片内部有 4KB 的 EPROM。51 子系列的特点如下：

(1) 8 位 CPU。

(2) 片内有 128B 的数据存储器。

(3) 程序存储器寻址空间为 64KB。

(4) 数据存储器寻址空间为 64KB。

(5) 128 个用户可以按“位”寻址的位空间。

(6) 21 个特殊功能寄存器。

(7) 4 个 8 位的 I/O 口。

(8) 两个 16 位的定时/计数器。

(9) 两个优先级的 5 个中断源。

(10) 一个全双工的串行接口。

(11) 片内带振荡器电路，频率范围 1.2M~12M，外加晶振就可工作。

(12) 单一+5V 电源。

对于 52 系列，有 8032、8052、8752 三种机型，与 51 系列不同之处在于片内数据存储器增加到 256B；8032 芯片内部带没有 ROM，8052 芯片内部有 8KB ROM，8752 芯片内部有 8KB EPROM。52 系列芯片内部带有 3 个 16 位的定时/计数器，6 个中断源。

在实际使用中，由于 8031(8032)内部没有 ROM，8051(8052)内部仅有 ROM，不方便程序修改，因此，在不需要系统扩展的嵌入式控制系统中，8751(8752)使用较多。

2.2　MCS-51 单片机的外部引脚和总线

MCS-51 系列单片机有 40 个引脚，采用双列直插式封装，引脚结构如图 2-1 所示。各引脚功能介绍如下。

2.2.1　输入/输出引脚

1. P0 口(引脚 39~32)

P0 口包括 P0.0~P0.7 共 8 个引脚，在不扩展 I/O 口和片外存储器时，做 I/O 口使用；在有 I/O 扩展或片外存储器时，分时做 8 位数据线和地址总线的低 8 位(A0~A7)。

2. P1 口(引脚 1~8)

P1 口包括 P1.0~P1.7 共 8 个引脚，是 MCS-51 单片机唯一的专用 I/O 口线。在简单的嵌入式控制系统中，经常用它来控制外围设备。

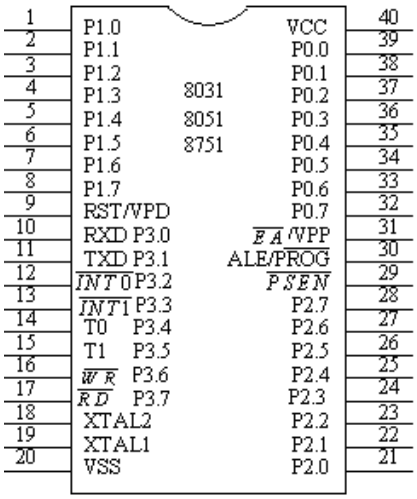


图 2-1　MCS-51 系列单片机引脚

3. P2 口(21~28 脚)

P2口包括 P2.0~P2.7共8个引脚，在系统进行存储器扩展时，做地址总线的高8位(A8~A15)，如果存储器扩展时不需要16位地址总线，如扩展容量小于64K，则省下的 P2口可做普通 I/O 口线。

4. P3 口(10~17 脚)

P3 口包括 P3.0~P3.7 共 8 个引脚，除做普通 I/O 口线外，每个引脚还有第二功能，详见后面的介绍。

2.2.2　MCS-51 单片机的控制线

MCS-51 单片机的控制线主要有如下几个引脚。

1. ALE/PROG(30 脚)

该信号是地址锁存信号输出端，ALE 在每个机器周期输出两次，在访问片外程序存储器时，下降沿用于锁存 P0 口输出的地址总线低 8 位(A0~A7)；在不访问片外程序存储器时，可作为时钟脉冲使用或做定时。在访问片外数据存储器时，ALE 脉冲会跳空一次，此时不能做时钟脉冲。对于有片内 EPROM 的 51 单片机，该引脚做编程脉冲 PROG 的输入端。

2. $\overline{PSEN}$ (29 脚)

片外程序存储器读选通信号，低电平有效。在从片外程序存储器读指令或数据时，该信号每个机器周期有效两次，指令或数据通过 P0 口读回。在访问片外数据存储器时，该信号无效。

3. RST/V_{PD} (9 脚)

RST 是复位信号，高有效，当 RST 保持 10ms 以上高电平时，可保证系统可靠复位。V_{PD} 是备用电源，该引脚可接备用电源，当V_{CC} 故障或系统电压过低时保证 RAM 中的数据不丢失。

4. $\overline{EA}$ / V_{PP} (31 引脚)

$\overline{EA}$ 是片外程序存储器选择端，$\overline{EA}$ =0，选片外程序存储器，$\overline{EA}$ =1，选片内程序存储器。V_{PP} 是片内 EPROM 编程电源，一般接+21V。

5. V_{CC} 和V_{SS} (40 和 20 引脚)

V_{CC} 是+5V，V_{SS} 是 5V 地。

6. XTAL1 和 XTAL2(19 和 18 脚)

外接晶振，MCS-51 常用 6M 或 11.0592M。

2.2.3　MCS-51 单片机的片外总线

MCS-51 单片机很多引脚是为了系统扩展而设的，这些引脚构成了片外地址总线、数据总线和控制总线。

1. 地址总线

地址总线宽度为 16 位，可以访问 64K 的存储器空间。由 P0 经锁存器提供地址总线的低

8 位(A0~A7)，由 P2 提供地址总线的高 8 位(A8~A15)。

2. 数据总线

数据总线宽度为 8 位，由 P0 口提供。

3. 控制总线

控制总线由 P3 口在第二功能状态下提供，此外还有 RST、$\overline{EA}$、ALE 和$\overline{PSEN}$四根控制线。

2.2.4　MCS-51 单片机存储器结构

MCS-51单片机存储器分为程序存储器和数据存储器两种，每种存储器又有片上和片外之分。下面分别进行介绍。

1. 程序存储器

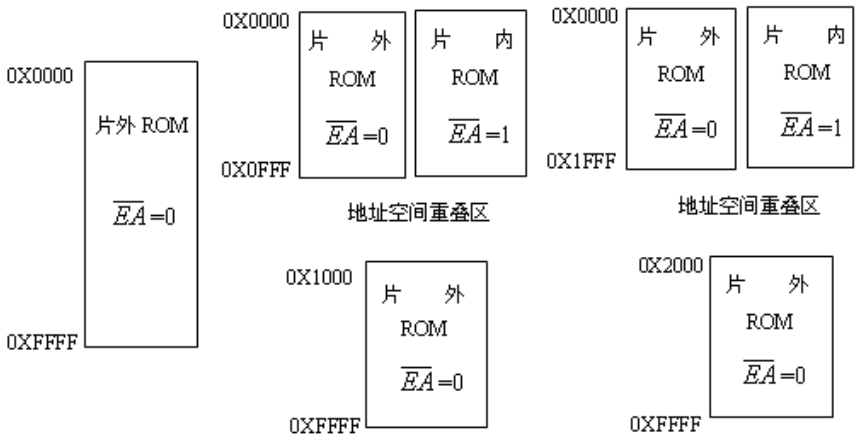
(1) 程序存储器的编址与访问

程序存储器用于存放单片机工作时的程序，单片机工作时，首先由用户编制好程序和表格、常数，把它存放在程序存储器中，然后在控制器的控制下，一次从程序存储器中取出指令送到 CPU 中执行，实现相应的功能。为此，设有一个专用的程序计数器 PC，用于存放要执行的指令的地址，它具有自动计数的功能，每取出一条指令，它的内容会自动加 1，指向下一条要执行的指令，从而实现从程序存储器中依次取出指令来执行。由于 MCS-51 单片机的程序计数器为 16 位，因此，程序存储器的地址空间为 64KB。

MCS-51单片机的程序存储器，从物理结构上可以分为片内和片外两种，对于片内程序存储器，在 MCS-51系列中，不同的芯片各不相同，8031和8032内部没有 ROM，8051内部有4KB 的 ROM，8751内部有4KB 的 EPROM，8052内部有8KB 的 ROM， 8752内部有8KB 的 EPROM。

对于内部没有 ROM 的8031和8032芯片，工作时只能扩展外部 ROM，最多可扩展64KB，地址范围为0X0000~0XFFFF，对于内部有 ROM 的芯片，根据情况也可以扩展外部 ROM ，但内部的 ROM 和外部的 ROM 共用64KB 的存储空间。其中，片内程序存储器地址空间和片外程序存储器的低地址空间重叠。8051、8751重叠区域为0X0000~0X0FFF，8052、8752重叠区域为0X0000~0X1FFF，MCS-51程序存储器编址如图2-2所示。

单片机在执行指令时，对于低地址部分，是从片内程序存储器中取指令，还是从片外程序存储器中取指令，是根据单片机芯片上的片外程序存储器选用引脚$\overline{EA}$ 电平的高低来决定的，$\overline{EA}$ 接低电平，则从片外程序存储器取指令，$\overline{EA}$ 接高电平，则从片内程序存储器取指令。对于 8031 和 8032 芯片，$\overline{EA}$ 只能保持低电平，指令只能从片外程序存储器中取得。



(A) 片内没有 ROM　　(B) 片内有 4KBROM　　(C) 片内有 8KBROM

图 2-2　程序存储器编址

(2) 程序存储器的 7 个特殊地址

在 64KB 程序存储器中，有 7 个单元有特殊用途，第一个单元是 0X0000 单元，因 MCS-51 系列单片机复位后 PC 的内容为 0X0000，故单片机复位后将从 0X0000 单元开始执行程序。程序存储器的 0X0000 单元地址是系统程序的启动地址，用户一般放一条绝对转移指令，转到用户设计的主程序的起始地址。另外 6 个单元对应于 6 个中断源(51 子系列为 5 个)。分别对应中断服务程序的入口地址，具体情况如表 2-1 所示。在用 C 语言编写中断服务程序时，这些地址是看不到的，这些地址是用中断向量号来代表的，具体情况如表 2-1 所示。

表 2-1　中断的入口地址

中断源	汇编编程入口地址	C 编程中断向量代码
外部中断 0	0003H	0
定时/计数器 0	000BH	1
外部中断 1	0013H	2
定时/计数器 1	001BH	3
串行口	0023H	4
定时/计数器 2(仅 52 子系列有)	002BH	5

表 2-1 所示的 6 个地址之间仅隔 8 个单元，用于存放中断服务程序往往不够用。用汇编语言编程时，这里通常放一条绝对转移指令，转到真正的中断服务程序。用 C 语言编程时则由编译器决定，用户不用考虑。

2. 数据存储器

数据存储器在单片机中用于存取程序执行时所需的数据，它从物理结构上可分片上数据存储器和片外数据存储器。两个部分在编址和访问方式上各不相同，其中，片内数据存储器又可分为多个部分，采用多种方式访问。

(1) 片内数据存储器

MCS-51系列单片机的片内数据存储器除了 RAM 块之外，还有特殊功能寄存器(SFR)块。对于51子系列，前者有128个字节，编址为0X00~0X7F；后者也占128个字节，编址为0X80~0XFF；二者连续不重叠。对于52子系列，前者有256字节，编址为0X00~0XFF；后者也有128字节，编址为0X80~0XFF；后者与前者的后128字节编址重叠，访问时通过不同的指令来区分。

片内数据存储器按功能可分为以下几个组成部分：工作寄存器组区、位寻址区、一般 RAM 区和特殊功能寄存器区，其中还包括堆栈区。具体分配情况如图 2-3 所示。

这里我们使用 C 语言编写控制程序，虽然我们使用的 C 语言(这里使用的 C 语言和一般 C 语言有稍许差别，后面会详细介绍)有将变量指定存储到数据存储器某个具体地址的功能，但我们经常不用，而是充分发挥 C 语言与硬件无关性的优点，由编译器来决定。

① 工作寄存器组区

0X00~0X1F 单元为工作寄存器组区，共 32 个字节，工作寄存器也称为通用寄存器，用于临时寄存 8 位信息。工作寄存器共有 4 组，称为 0 组、1 组、2 组和 3 组。每组 8 个寄存器，依次用 R0~R7 表示。也就是说， R0 可能表示 0 组的第一个寄存器(地址为 0X00)，也可能表示 1 组的第一个寄存器(地址为 0X08)，还可能表示 2 组、3 组的第一个寄存器(地址分别为 0X10、0X18)，使用哪一组当中的寄存器由程序状态寄存器 PSW 中的 RS0 和 RS1 两位来选择。

在用 C 语言编程时仅在编写中断服务函数时可以选择使用哪一组工作寄存器组。就是在中断向量后面加一个 0~3 的数字表示使用 0 组~3 组的哪一个工作寄存器组。但我们也可以不加这个数字，由编译器自动选择。其他情况下，则不用选择工作寄存器组。

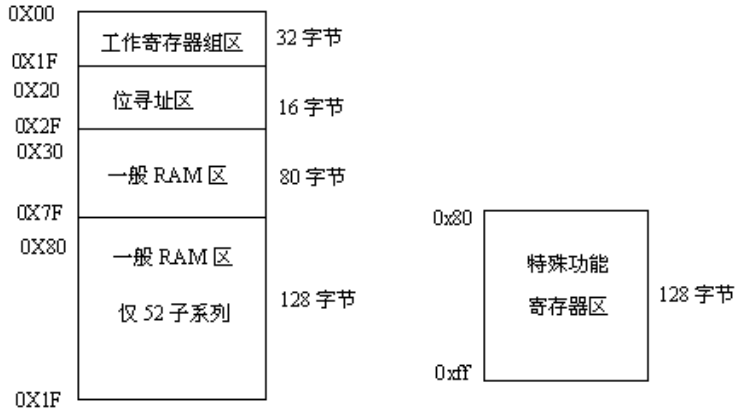


图 2-3　片内数据存储器分配

② 位寻址区

0X20~0X2F 为位寻址区，共 16 个字节，128 位。这 128 位每位都可以按位方式使用，每一位都有一个位地址，位地址范围为 0X00~0X7F ，用 C 语言编程时，在用到位变量时我们只是通过位变量数据类型(后面介绍)来定义，具体这个位变量存放在什么地址是由编译器

决定的。也就是说，我们可以使用位变量，但可以不管它们的地址。位寻址区地址分配如表 2-2 所示。

表 2-2　位寻址区地址分配表

位 / 位地址 / 字节地址	D7	D6	D5	D4	D3	D2	D1	D0
0X20	07	06	05	04	03	02	01	00
0X21	0F	0E	0D	0C	0B	0A	09	08
0X22	17	16	15	14	13	12	11	10
0X23	1F	1E	1D	1C	1B	1A	19	18
0X24	27	26	25	24	23	22	21	20
0X25	2F	2E	2D	2C	2B	2A	29	28
0X26	37	36	35	34	33	32	31	30
0X27	3F	3E	3D	3C	3B	3A	39	38
0X28	47	46	45	44	43	22	41	40
0X29	4F	4E	4D	4C	4B	4A	49	48
0X2A	57	56	55	54	53	52	51	50
0X2B	5F	5E	5D	5C	5B	5A	59	58
0X2C	67	66	65	64	63	62	61	60
0X2D	6F	6E	6D	6C	6B	6A	69	68
0X2E	77	76	75	74	73	72	71	70
0X2F	7F	7E	7D	7C	7B	7A	99	78

③ 一般 RAM 区

0X30~0X7F 是一般 RAM 区，也称为用户 RAM 区，共 80 个字节，对于 52 子系列，一般 RAM 区从 0X30~0XFF 单元。另外，对于前两区中未用的单元也可作为用户 RAM 单元使用。

④ 堆栈区与堆栈指针

堆栈是按先入后出、后入先出(FILO)的原则进行管理的一段存储区域。在 MCS-51 单片机中，堆栈占片内数据存储器的一段区域。在用 C 语言编程时，由编译器管理堆栈区与堆栈指针。

(2) 特殊功能存储器

特殊功能存储器(SFR)也称为专用寄存器，专门用于控制、管理片内算术逻辑部件、并行 I/O 接口、串行口、定时/计数器、中断系统等功能模块的工作。用户在编程时可以给其设定值，但不能移作他用。SFR 分布在 0X80~0XFF 的地址空间，与片内数据存储器统一编址。除 PC 外，51 子系列有 18 个特殊功能寄存器，其中 3 个为双字节，SFR 共占用 21 个字节；52 子系列有 21 个特殊功能寄存器，其中 5 个是双字节，SFR 共占用 26 个字节。它们的分配情况如下：

- CPU 专用寄存器：累加器 A，寄存器 B，程序状态寄存器 PSW，堆栈指针 SP，数据指针 DPTR。

- 并行接口：P0~P3。
- 串行接口：串行控制寄存器 SCON，串口数据缓冲器 SBUF，电源控制寄存器 PCON。
- 定时/计数器：方式寄存器 TMOD，控制寄存器 TCON，初值寄存器 TH0、TL0、TH1、TL1。
- 中断系统：中断允许寄存器 IE，中断优先寄存器 IP。

特殊功能寄存器的名称、在 C 语言中引用方法如表 2-3 所示。有一些特殊功能寄存器在 C 语言中是看不到的，表中没列出。

在 C 语言程序中，特殊功能寄存器在头文件“REG51.H”中定义，使用这些特殊功能寄存器，在程序中必须引用这个头文件。特殊功能寄存器除累加器 A 和寄存器 B 是通用寄存器，在程序中可以随便使用外，其他特殊功能寄存器都是专用的，不能移作它用。

表 2-3　特殊功能寄存器的名称、在 C 语言中引用方法

名称	按字节引用符号	C 语言程序中按位引用符号							
		D7	D6	D5	D4	D3	D2	D1	D0
堆栈指针	SP								
定时器/计数器控制	TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
定时器/计数器 T0	TL0、TH0								
定时器/计数器 T1	TL1、TH1								
P0 口	P0	P0^7	P0^6	P0^5	P0^4	P0^3	P0^2	P0^1	P0^0
P1 口	P1	P1^7	P1^6	P1^5	P1^4	P1^3	P1^2	P1^1	P1^0
P2 口	P2	P2^7	P2^6	P2^5	P2^4	P2^3	P2^2	P2^1	P2^0
P3 口	P3	P3^7	P3^6	P3^5	P3^4	P3^3	P3^2	P3^1	P3^0
电源控制	PCON	SMOD							
串口控制	SON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
串口数据	SBUF								
中断允许	IE	EA		ET2	ES	ET1	EX1	ET0	EX0
中断优先级	IP			PT2	PS	PT1	PX1	PT0	PX0
累加器 A	ACC	ACC^7	ACC^6	ACC^5	ACC^4	ACC^3	ACC^2	ACC^1	ACC^0
寄存器 B	B	B^7	B^6	B^5	B^4	B^3	B^2	B^1	B^0

在表 2-3 中，特殊功能寄存器 P0、P1、P2、P3、A、B 可以使用字节符号按字节引用，也可以按位引用，编译器规定按位引用的格式是：字节符号^位，如表中所示，例如，引用 P0 的 D0 位，使用 P0^0，但这样做在程序中容易和异或符号“^”混淆，直接用“^”引用，编译器会报错，所以要用位变量 sbit(位数据类型，后面介绍)重新换一个名字定义；累加器 A 的字节符号是 ACC，在程序中写成 A，编译器也会报错，寄存器 B 的字节符号还是 B；其他特殊功能寄存器按字节引用时使用字节符号，如果有位名称也可以按位引用，按位引用时使

用位符号。所有特殊功能寄存器无论是字节名称，还是位名称规定都要大写。虽然我们暂时还没学习用 C 语言编写 MCS-51 的驱动程序，但下面的例子基本还可以看懂特殊功能寄存器的引用方法。

例 2-1　特殊功能寄存器的引用。

```
#include <reg52.h>   //特殊功能寄存器在头文件“reg52.h”中定义
sbit  acc7=ACC^7;    //定义 ACC 的 bit7 位为 acc7
sbit  p0_0=p0^0;     //定义 P0 的 bit0 位为 p0_0
sbit  p1_7=p1^7;     //定义 P1 的 bit7 位为 p1_7
sbit  p2_0=p2^0;     //定义 P2 的 bit0 位为 p2_0
sbit  b0=B^0;        //定义 B 的 bit0 位为 b0
//-------------------------------------------------------------------------
//主函数
//-------------------------------------------------------------------------
void main(void)
{
  ACC=0xff;          //引用累加器 a
  acc7=0;            //引用累加器 ACC 的 bit7
  P0=0xff;           //引用 P0
  p0_0=0;            //引用 P0 bit0
  p1_7=1;            //引用 P1 bit7
  p2_0=0;            //引用 P2bit0
  B=0xff;            //引用寄存器 B
  b0=0;              //引用寄存器 B bit0
}
```

(3) 存储器地址访问

在 MCS-51 中，程序存储器和数据存储器的地址空间都是 0x0000~0xffff，片上数据存储器和片外数据存储器的低 256 字节的地址空间是相同的。但由于访问时使用不同的指令，由不同的控制信号控制，所以这些地址不会混淆。

2.3　MCS-51 单片机的最小系统

所谓最小系统，是指一个真正可用的单片机最小配置系统。对于单片机内部资源已能满足系统需要的，可直接采用最小系统。MCS-51单片机根据片内有无程序存储器最小系统分两种情况。

2.3.1　8051/8751 的最小系统

8051/8751片内有4K 的 ROM/EPROM，因此，只需要外接晶体振荡器和复位电路就可构成最小系统。

该最小系统的特点如下：

(1) 由于片外没有扩展存储器和外设，P0、P1、P2、P3 都可以作为用户 I/O 口使用。

(2) 片内数据存储器有 128 个字节，地址空间为 0x00~0x7F，没有片外数据存储器。

(3) 内部有 4KB 程序存储器，地址空间为 0x0000~0x0FFF，没有片外程序存储器，$\overline{EA}$ 应接高电平。

(4) 可以使用两个定时/计数器 T0 和 T1，一个全双工的串行通信接口，5 个中断源。

2.3.2 8031 最小应用系统

8031 芯片内无程序存储器，因此，在构成最小应用系统时不仅要外接晶体振荡器和复位电路，还应外扩程序存储器。

该最小系统的特点如下：

(1) 由于 P0、P2 在扩展程序存储器时作为地址线和数据线，不能作为 I/O 线，因此，只有 P1、P3 作为用户 I/O 口使用。

(2) 片内数据存储器同样有 128 个字节，地址空间为 00H~7FH，没有片外数据存储器。

(3) 内部无程序存储器，只能在片外扩展程序存储器，其地址空间随扩展芯片容量不同而不同。如使用的是2764芯片，容量为8K字节，地址空间为0x0000~0x1FFF。在实际应用中，由于小容量的程序存储器产量逐渐减少，价格反而比大容量的程序存储器贵，所以在扩展程序存储器时往往选择容量最大的27512芯片，容量为64K字节，地址空间为0x0000~0xFFFF。

8031 由于片内没有程序存储器，只能使用片外程序存储器，$\overline{EA}$ 只能接低电平。

(4) 同样可以使用两个定时/计数器 T0 和 T1，一个全双工的串行通信接口，5 个中断源。

2.4 MCS-51 单片机系统扩展

MCS-51单片机系统扩展包括程序存储器扩展、数据存储器扩展、I/O口扩展、定时/计数器扩展、中断系统扩展和串行口扩展。

2.4.1 存储器扩展概述

1. MCS-51 单片机的存储器扩展能力

由于 MCS-51 单片机地址总线宽度为 16 位，所以片外可扩展的存储器最大容量为 64KB，地址为 0x0000~0xFFFF。因为程序存储器和数据存储器通过不同的控制信号和指令进行访问，允许两者的地址空间重叠，所以片外可扩展的程序存储器与数据存储器都为 64KB。

另外，在 MCS-51 单片机中，扩展的外部设备与片外数据存储器统一编址，即外部设备占用片外数据存储器的地址空间。因此，片外数据存储器同外部设备总的扩展空间是 64KB。

2. 存储器扩展的一般方法

存储器除按照读写特性不同分为程序存储器和数据存储器之外，每种存储器都有不同的种类。程序存储器又可分为掩膜 ROM、光可擦除 EPROM 或电可擦除 EEPROM 以及最近使用较多的快闪存储器 FLASH 等。

数据存储器又可分为静态 SRAM 和动态 DRAM。因此，存储器芯片有多种。另外，即使同一种类的存储器芯片，容量不同，其引脚情况也不相同。尽管如此，存储器芯片与单片机扩展连接具有共同的规律。即不论何种存储器芯片，其引脚都呈三总线结构，与单片机连接都是三总线对接。

3. 存储器芯片的控制线

对于程序存储器，一般来说，具有输出允许控制线 OE，它与单片机的 PSEN 信号线相连。除此之外，对于 EPROM 芯片还有编程脉冲输入线(PRG)、编程状态线(REAY/BUSY)，PRG 应与单片机在编程方式下的编程脉冲输出线相连；REAY/BUSY 在单片机查询输入/输出方式下，与一根 I/O 接口线相连；在单片机中断工作方式下，与一个外部中断信号输入线相连。对于数据存储器，一般都有输出允许控制线 OE 和写控制线 WE，它们分别与单片机的读信号线 RD 和写信号线 WR 相连。

4. 存储器芯片的数据线

数据线的数目由芯片的字长决定。1 位字长的芯片数据线只有 1 根；4 位字长的芯片数据线有 4 根；8 位字长的芯片数据线有 8 根。现在单片机存储器扩展使用的芯片字长基本上是 8 位。连接时，存储器芯片的数据线与单片机的数据总线(P0.0~P0.7)按由低位到高位的顺序相连。

5. 存储器芯片的地址线

地址线的数目由芯片的容量决定，容量与地址线数目满足关系式：$Q=2^n$，存储器芯片的地址线与单片机的地址总线(A0~A15)按由低位到高位的顺序相连。一般来说，存储器芯片的地址线数目总是少于等于单片机地址总线的数目，因此连接后，单片机的高位地址线有剩余。剩余地址线一般可作为译码线，译码输出与存储器芯片的片选信号 CE 相连。存储器芯片有一根或几根片选信号线。对存储器芯片访问时，片选信号必须有效，即选中存储器芯片。片选信号线与单片机系统的译码输出相连后，就决定了存储器芯片的地址范围。

2.4.2　存储器扩展的讨论

在单片机发展的早期，由于8031片内没有程序存储器、8051和8751片内分别有4KB ROM和4KB EPROM，片内数据存储器只有128B，因此，在实际使用时往往要进行程序存储器和数据存储器的扩展，这方面介绍的资料很多，我们不再赘述。

随着计算机技术的发展、特别是存储器技术的发展，许多 MCS-51 单片机的兼容机型程序存储器都采用了快闪存储器 FLASH 来代替 ROM 或 EPROM。在这些兼容机型中，STC 和 ATMEL 公司的 89C51/89C52 单片机在 8 位机市场上占有很大份额。

89C51/89C52 单片机除了程序存储器采用了快闪存储器 FLASH 来代替 ROM 或 EPROM 之外，ISP(In System Programming，在系统编程)和 IAP(In Application Programming，“应用现场”可调试功能)是应用在 FLASH 程序存储器基础上的一种编程模式，它可以在应用程序正常运行的情况下，通过 IAP 程序对另外一段程序 FLASH 空间进行读/写操作，甚至可以控

制对某段、某字节的读/写操作，这给数据存储和硬件现场升级带来了很大的灵活性。IAP 和 ISP 功能是 STC 89C51/89C52 单片机的特别重要的功能。

89C51/89C52 单片机片上程序存储器和数据存储器的容量有多种，可根据系统需要选择，因此，1 片 89C51/89C52 单片机不需要存储器扩展就是 1 个最小嵌入式控制系统。

STC 89C51/89C52 单片机由于采用 ISP 技术，不使用仿真器进行编程，减小了开发成本，特别适合大学生学习使用。

STC 89C51/89C52 单片机在管脚和编程上与 MCS-51 单片机完全兼容，今后我们就以 STC 89C51/89C52 单片机为例来讲述问题。

关于 STC 89C51/89C52 单片机的应用我们在第 3 章详细讲解。

2.5　输入/输出口扩展和使用

MCS-51 单片机有 4 个并行 I/O 接口，STC 89C51/89C52 单片机在 MCS-51 单片机的基础上增加了一个 P4 口，每个口 8 位，但这些接口并不能完全提供给用户使用，对于片内有程序存储器的 8051/8751/STC 89C51/STC 89C52 单片机，在不扩展外部资源，不使用串行口、外中断、定时/计数器时，才能使用 5 个并行 I/O 接口。如果片外要扩展，则 P0、P2 口要用来做数据、地址总线，P3 口中的某些位也要用来做第二功能信号线。留给用户的 I/O 线很少。因此，在大部分嵌入式应用中都要进行 I/O 口扩展。

I/O 扩展接口种类很多，其功能可分为简单 I/O 接口和可编程 I/O 接口。简单 I/O 接口扩展通过数据缓冲器、锁存器来实现，结构简单，价格便宜，但功能简单。可编程 I/O 接口扩展通过可编程接口芯片实现，电路复杂，价格相对较高，但功能全，使用灵活。不管是简单 I/O 接口还是可编程 I/O 接口，与其他外部设备一样都是与片外数据存储器统一编址。占用片外数据存储器的地址空间，通过片外数据存储器的访问方式访问。

本章只介绍简单 I/O 接口，可编程 I/O 接口将在后面章节中介绍。

2.5.1　简单 I/O 接口扩展

通常通过数据缓冲器、锁存器来扩展简单 I/O 接口。例如，利用 74LS373 、74LS244、74LS273、74LS245 等芯片都可以做简单的 I/O 扩展。实际上，只要具有输入三态、输出锁存的电路，就可以用作 I/O 接口扩展。

如图 2-4 所示是利用 74LS373 和 74LS244 扩展的简单 I/O 接口。其中，74LS373 扩展并行输出口，74LS244 扩展并行输入口。74LS373 是一个带输出三态门的 8 位锁存器，具有 8 个输入端 D0~D7，8 个输出端 Q0~Q7，G 高电平时，则把输入端的数据锁存于内部锁存器，$\overline{OE}$ 为输出允许端，低电平时把锁存器中的内容通过输出端输出。

74LS244 是单向数据缓冲器，带两个控制端 $\overline{1G}$ 和 $\overline{2G}$，当它们为低电平时，输入端 D0~D7 的数据输出到 Q0~Q7。

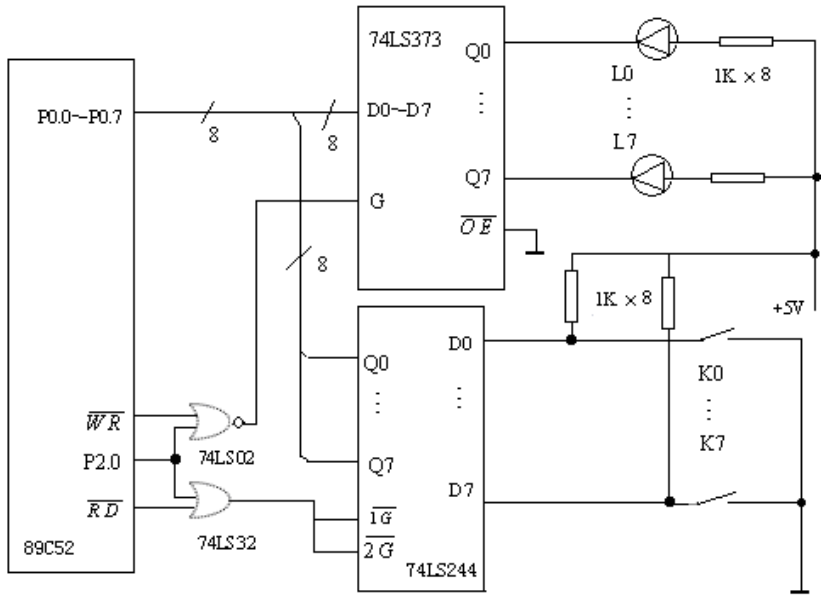


图 2-4　简单 I/O 接口扩展

图中 74LS373 的控制端 G 是由 8051 单片机的写信号$\overline{WR}$和 P2.0 通过或非门 74LS02 后相连，输出允许端$\overline{OE}$直接接地，所以，当 74LS373 输入端有数据时直接通过输出端输出。当执行向片外数据存储器写指令时，指令中片外数据存储器的地址使 P2.0 为低电平，则控制端 G 有效，数据总线上的数据就送到 74LS373 的输出端。74LS244 的控制端$\overline{1G}$和$\overline{2G}$连在一起，8051 单片机的读信号$\overline{RD}$和 P2.0 通过或门 74LS32 与$\overline{1G}$和$\overline{2G}$相连，当执行从片外数据存储器读指令时，指令中片外数据存储器的地址使 P2.0 为低电平，则控制端$\overline{1G}$和$\overline{2G}$有效，74LS244 的输入端的数据通过输出端送到数据总线，然后传送到 8051 单片机的内部。

例 2-2　在图 2-4 中，输入接口接 K0~K7 八个开关，输出接口接 L0~L7 八个发光二极管，现读入 8 个开关状态，通过输出口 L0~L7 来显示，程序如下：

```
#Include<absacc.h>
#define uchar unsigned char
#define uint unsigned int
//---------------------------------------------------------------------
//主函数
//---------------------------------------------------------------------
void main(void)
{
  :
  :
  uchar i;
  i=XBYTE[0xfeff];
  XBYTE[0xfeff]=i;
}
```

由于我们还没有学习 C 语言编程(第 4 章开始介绍)，这里先简单介绍一下程序。

单片机和通用计算机一样，对外围设备进行“端口”管理，“端口”就是外围设备的地址，一个外围设备有一个或多个“端口” 地址，单片机对外围设备进行访问必须通过“端口”地址来进行。Keil C 对“端口”地址即外围设备进行访问是通过头文件 absacc.h 中定义的几个“宏”来进行的，所以在程序中要引用这个头文件。XBYTE[0xfeff] 表示该外围设备的字节地址是 0xfeff。

头文件 absacc.h 和外围设备“端口”地址的概念我们在后面讲述指针时详细介绍。

程序中的 i=XBYTE[0xfeff]是“读”指令，控制线 $\overline{RD}$ 为低电平有效；译码线 P2.0 为低，其他地址线无关，我们规定为 1，所以有效地址是 0xfeff；$\overline{RD}$ 和 P2.0 为低，“或”的结果是低，$\overline{1G}$ 和 $\overline{2G}$ 有效，74LS244 芯片 Q0~Q7 数据送数据线，读入 i。

程序中的 XBYTE[0xfeff]= i 是“写”指令，控制线 $\overline{WR}$ 为低电平有效；译码线 P2.0 为低，其他地址线无关，我们规定为 1，所以有效地址也是 0xfeff；$\overline{WR}$ 和 P2.0 为低，“或非”的结果是高，G 有效，芯片 74LS373 被选中。数据线上数据 i 写入 74LS373，由于 74LS373 输出允许 $\overline{OE}$ 直接接地，所以数据线上数据 i 直接通过 74LS373 的 Q0~Q7 输出控制 L0~L7。

在计算机术语中，“读”和“写”都是以 CPU 为中心的，“读”就是把外围设备数据赋给 CPU，“写”就是将 CPU 中的数据赋给外围设备。

2.5.2　I/O 口在 TTL 电路中使用

前面讲过，MCS-51 单片机有 4 个 8 位的并行输入/输出接口，STC 89C51/89C52 单片机在 MCS-51 单片机的基础上增加了一个 P4 口。P0、P1、P2、和 P3、P4 这 5 个口既可以做并行输入/输出数据通道，也可以按“位”方式使用，并行端口是 MCS-51 单片机控制外部设备的主要通道。

用 C 语言编写控制程序，不用对 I/O 口的内部结构了解的太多，只要能正确使用即可。

1. P0 口的操作

和 IBM_PC 机不同，MCS-51 单片机的程序存储器和数据存储器是分开的，用户程序固化在 EEPROM 或 FLASH 为介质的程序存储器中，运行中的数据存放在 RAM 中，它们各占 64K 的地址空间，由于访问指令和控制信号不同，因此，对程序存储器和数据存储器的访问地址不会混淆。访问 64K 的地址空间，地址线必须有 16 条。MCS-51 把 P0 作为地址总线的低 8 位(A0~A7)，P2 作为地址总线的高 8 位(A8~A15)，由于单片机体积小，出线困难，MCS-51 把 P0 也作为 8 位数据线(D0~D7)。在程序执行过程中，P0 上先出现地址线，然后通过一个控制信号(ALE)将此地址总线的低 8 位(A0~A7)锁存在一个锁存器(74373)中，通过锁存器输出提供给外部地址总线，然后 P0 上出现的信号就是 8 位数据线(D0~D7)。

用 C 语言编写控制程序，对这种分时使用 P0 的细节可以不用知道，我们只要正确使用 C 语言编写程序，编译器和 MCS-51 单片机硬件会自动完成这些操作，我们把 MCS-51 数据线和地址线理解成和 IBM_PC 一样也行。

如果嵌入式控制系统很简单，不需要 P0 参与地址译码，那么 P0 口就只用来做数据线。

如图 2-5 所示为液晶显示器 T6963C 控制电路，T6963C 片选用 P2.7，P0 只做数据线。

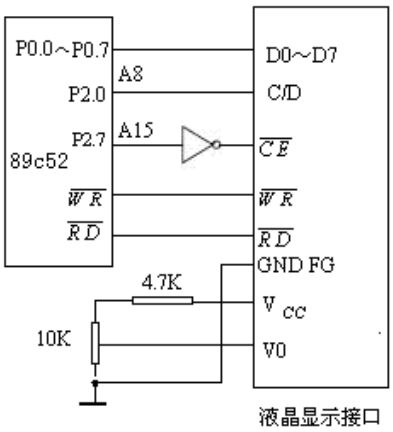


图 2-5　简单译码电路，P0 只做数据线

还有许多简单场合，嵌入式控制系统不需要地址线和数据线，P0 口也可以做 8 位 I/O 口使用。此时，P0 口具有驱动 8 个 TTL 门电路的负载能力。

由于 P0 口的输出是漏电极开路(相当于集电极开路)，外输出电路应加上拉电阻。

由于 P0 口的结构，在做输入时应先向 P0 口写“1”。P0 口在输出时具有锁存功能，输入时具有缓冲功能。

2. P1 口的操作

P1 口的操作基本与 P0 口相同，但它只能做通用 I/O 口使用，也是在简单嵌入式控制系统中用的最多的并行口。

它与 P0 口的不同点是内部有上拉电阻，做输出时不用再加上拉电阻，P0 口具有驱动 4 个 TTL 门电路的负载能力。

由于 P1 口的结构，做输入时应先向 P1 口写“1”。P1 口在输出时具有锁存功能，输入时具有缓冲功能。

3. P2 口的操作

P2 口的操作基本同 P0 口和 P1 口，当外围设备，主要是程序存储器或数据存储器需要地址总线高 8 位时，P2 口只能做地址总线使用，否则，它也可以做 I/O 口，此时，它的用法与 P1 口相同。

4. P3 口的操作

P3 口除做普通 I/O 口之外，它的每一位还具有第二功能，具体如表 2-4 所示。

表 2-4　P3 口第二功能

P3 口引脚	第二功能
P3.0	RXD　串行输入
P3.1	TXD　串行输出
P3.2	INT0　外部中断 0 输入,低有效

(续表)

P3 口引脚	第二功能	
P3.3	INT1	外部中断 1 输入,低有效
P3.4	T0	定时器/计数器 0 外部计数脉冲输入
P3.5	T1	定时器/计数器 1 外部计数脉冲输入
P3.6	WR	外部数据存储器写
P3.7	RD 成	外部数据存储器读

当系统复位或上电时，P3 口处于第二功能状态；当执行 I/O 操作指令时，又变回普通 I/O 口，此时，它的用法和负载能力同 P1 口。

5. P4 口的操作

P4 口的操作基本与 P1 口相同，详见第 3 章。

2.5.3　I/O 口在外围设备中使用

在简单的嵌入式控制系统中，并行口特别是 P1、P4 口可以直接与外围设备相连做输入和输出。但由于并行口负载能力弱，同时为了保护微处理器，这种连接一般需要经过隔离器件来完成。

光电隔离器件是使用最方便、价格低廉、隔离效果很好的一种隔离器件。

如光电隔离器件 TLP521-4，原理图如图 2-6 所示，它的工作电压范围为 5~50V，工作电流为 5~50mA，除了做光电隔离外，还可以起到功率转换作用。

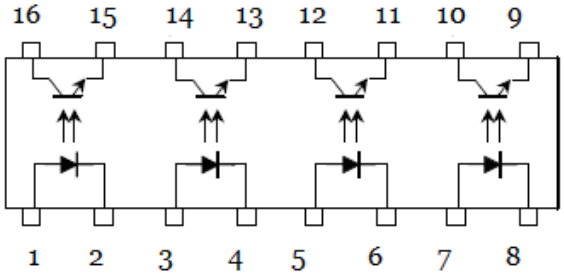


图 2-6　光电隔离器件 TLP521-4 原理图

如 P1.0 经光电隔离器件做输出，控制一个工作电压为 24V 的中间继电器，具体电路可按如图 2-7 所示进行设计，当 P1.0 输出高电平时，发光二极管有电流流过并发光，三极管导通并饱和，集电极和发射极接通，集电极输出低电平给中间继电器；当 P1.0 输出低电平时，发光二极管无电流流过，集电极和发射极截止，集电极输出高电平给中间继电器。

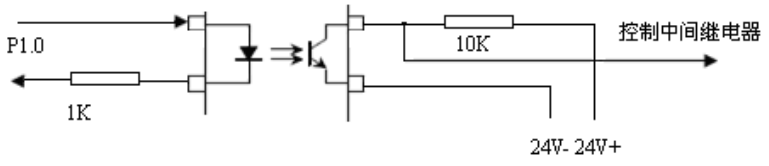


图 2-7　P1.0 经光电隔离器件做输出

如果 P1.1 经光电隔离器件做输入，输入信号是一个 24V 的开关量，具体电路可按如图 2-8 所示进行设计，原理同上，只是限流电阻不同。

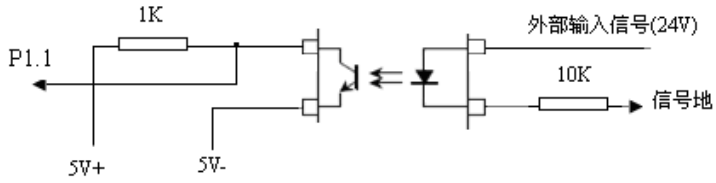


图 2-8　P1.1 经光电隔离器件做输入

下面，我们结合如图 2-9 所示的一个具体实例来看一下 TLP521-4 的应用：有一台机械加工设备，它的纵向进给距离需要精密控制，同时，纵向进给距离和速度要给相应仪表进行显示。我们在纵向驱动的液压马达的轴上，安装一台光电码盘做反馈器件，液压马达每转一周，光电码盘会发出两路相位差 90 度、个数为 2000、电平为 5~12V 的脉冲。我们通过 TLP521-4 的隔离将此信号送给工作电压不同的单片机和各个仪表，如图 2-9 所示。

在这里，TLP521-4 起着保护单片机的隔离作用；同时还将一路光电码盘反馈信号给需要反馈信号的单片机(进给距离精密控制)和几个仪表(进给距离和速度显示)的信号分配作用；由于单片机和几个仪表的工作电压不同，TLP521-4 还起到了电平转换作用。

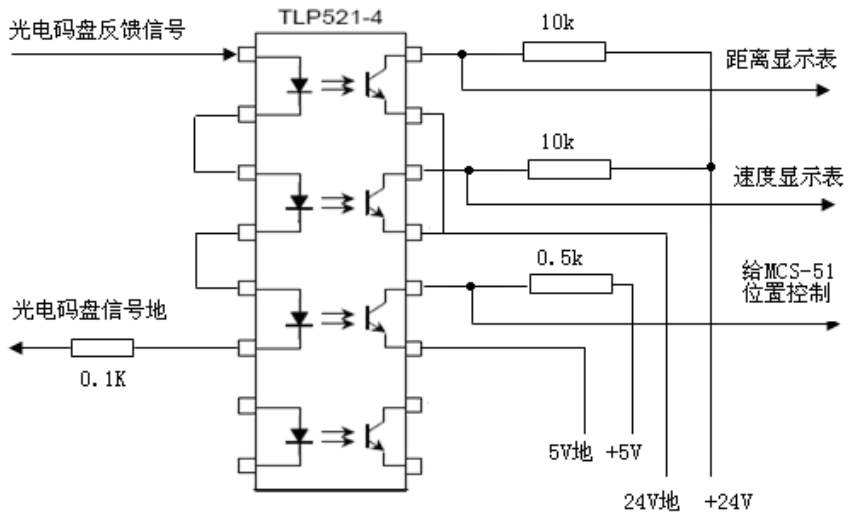


图 2-9　TLP521-4 的应用

除使用光电器件做隔离和功率转换之外，单片机还经常使用继电器、可控硅(也叫晶闸管)、晶体管、固态继电器等器件做隔离和功率转换。

如图 2-10 和图 2-11 所示，显示了使用晶体管和固态继电器做隔离和功率转换的实例。

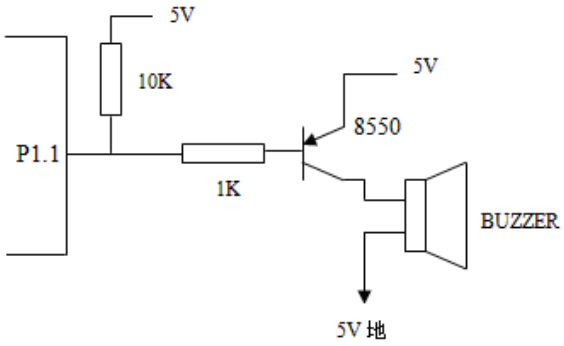


图 2-10　P1.1 通过晶体管放大电路控制蜂鸣器

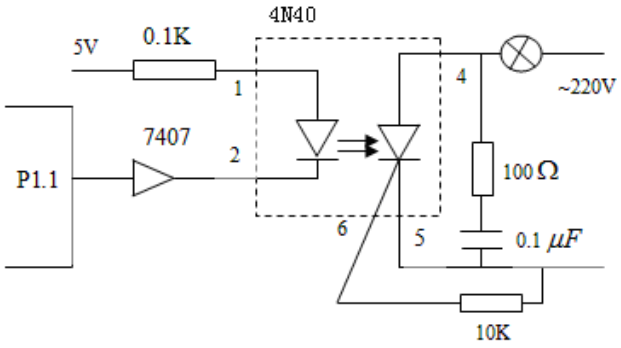


图 2-11　P1.1 通过固态继电器控制大功率设备

2.6　习　　题

1. MCS-51 单片机有几个系列？它们的区别是什么？
2. MCS-51 单片机有哪些特性？
3. MCS-51 单片机有几个并行口？它们的用法各是什么？
4. P0 口作为数据线，如何分时做地址总线的低 8 位？
5. MCS-51 单片机的数据线、控制线和地址线有哪些？
6. MCS-51 单片机存储器结构和 IBM_PC 机有什么不同?
7.信号线 $\overline{EA}$ 有什么作用?
8. MCS-51 单片机数据存储器地址在什么情况下会重叠?
9. MCS-51 单片机特殊功能寄存器有哪些？如何按字节或位来访问？
10. 简述什么是 8031、8051、8751 最小系统？
11. MCS-51 单片机如何进行程序存储器和数据存储器的扩展？
12. MCS-51 单片机如何利用数据缓冲器和锁存器进行简单 I/O 扩展？