单位代码	10476
学　　号	0408180004
分 类 号	TP314

河南師範大學

硕士学位论文

MCS-51 单片机汇编语言程序 IDE 设计与实现

学科、专业：计算机软件与理论
研究方向：嵌入式系统及应用
申请学位类别：工学硕士
申　请　人：闫　娟
指导教师：毋茂盛　教授

二〇〇七年六月

摘 要

IDE 即集成开发环境(Integrated Development Environment)是进行软件开发必不可少的工具软件，一个方便、成熟、稳定、功能强大的 IDE，可显著地提高开发效率。本文主要研究了如何设计与实现一个基于 VC++的 MCS-51 单片机汇编语言程序的 IDE 系统。

该系统在 WINDOWS 环境下运行，通过编译器的两遍扫描实现对 MCS-51 汇编语言的编译，生成写入单片机的 Intel Hex 格式的目标文件，并为用户提供源码与目标码对照的列表文件（*.lst 文件）以及编译过程中出现的错误反馈信息。IDE 中的编译模块按照词法分析、语法分析、语义分析、目标代码生成等几个阶段进行，符号表的管理和出错处理贯穿编译工作的各个阶段，编译过程中源程序的各种信息被保留在各种不同的表格里。由于该系统具有程序下载功能，要用到串口通信，因此本文还介绍了串行通信的基本原理、RS－232 的定义以及串行通信的常用连接方式，然后对 Windows 的通信机制进行了讨论。在此基础上，对串行通信的 Windows 编程方法做了详细的探讨。

该软件系统将文件操作、参数设置、编辑、编译、代码下载、实验演示、帮助信息（源程序、指令表等）等功能集成在统一的开发环境中，功能齐全，界面友好；在计算机上调试好的程序既可以通过编程器写入单片机中，也可以直接通过计算机的串口下载到实验仪上，使用方便灵活,特别适合于 MCS-51 单片机的实验教学。本系统已通过河南省科技厅组织的会议鉴定，达国内领先水平。

关键词：单片机，汇编语言，IDE，编译器，串行通信

Abstract

Integrated Development Environment(IDE), plays a very important role in software developing procedure. With a mature, stable, powerful, and convenient IDE software, we will be more efficient in work. This thesis researches how to design and realize the IDE of MCS-51 assembly language based on the VC++ language.

This IDE system carries out under the WINDOWS environment, through two scanning, the complier realizes the translation of MCS-51 assembly language.The system produces the Intel Hex file and the *.lst tabulates which provides the comparison of the source code and goal code for the user, as well as appearing wrong feedback information. The complier process according to the lexical analysis, the grammar analysis, the semantic analysis, the goal code production and so on several stages carries on. The symbol table management and mistake processing pass-through each stage of the translation work.. In the translation process each kind of information of source program is retained in all sorts of different forms. This text also introduces basic mode because the IDE has the function of downloading the code .It introduces the definition of RS-232 and serial communication daily connection way of serial communication, then discusses communication mechanism in Windows, on this basis , makes detailed description of serial Windows programming method of serial communication.

This IDE soft system integrates the document operation, parameter setting, editor, compiler ,code downloading ,experiment demonstration ,help information and so on.It has a friendly interface. Besides, the object code can also be directly download to the SCM by serial port without the programmer. It is quite useful in MCU 's experiment teaching.It has get the identification of the science and technology department of Henan province,and keeps ahead of the national level.

Keywords: MCU, Assembly language, IDE, Compiler, Serial communication

独 创 性 声 明

本人郑重声明：所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得河南师范大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签名：闫娟 日期：2007.6.12

关于论文使用授权的说明

本人完全了解河南师范大学有关保留、使用学位论文的规定，即：有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权河南师范大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。（保密的学位论文在解密后适用本授权书）

签名：闫娟 导师签名：毋茂盛 日期：2007.6.12

引 言

“工欲善其事，必先利其器”，精良的工具装备，是搞研发成功的基本条件。单片机 IDE (集成开发环境)是开发单片机应用系统必不可少的工具。目前，单片机 IDE 主要由世界几个大的软件公司开发，如德国的 Keil、荷兰的 Tasking，美国的 Franklin，台湾的 IAR 等等。其中 Keil 的产品 μVision2 和 Tasking 的产品 EDE 品质尤为出众，知名度颇高[1]。研究它们的设计原理和各自的优缺点，开发适合自己的 IDE，成为摆在国内计算机界面前的一个迫切任务。

在本文中，笔者用 VC++来开发实现了 MCS-51 单片机汇编语言程序的集成开发环境。首先，该系统将编辑、编译源程序和程序下载、帮助信息（源程序、指令表等）都集成在统一的开发环境中，功能齐全，界面友好；其次，该系统还提供了 24 个不同的硬件综合性或设计性实验演示。此外，该系统实现了代码下载功能，在计算机上调试好的程序可以脱离编程器，直接通过计算机的 USB 或串口下载到单片机实验设备上，实验方便灵活。其设计主要有以下几个部分：文件操作设计；编辑器的设计；编译器的设计；参数设置设计；用户登录模块设计；帮助文件模块设计；代码下载功能的设计。其中文件操作实现对文件的读、写、打印、关闭和保存等功能。编辑器实现对源文件的编辑，如查找、剪切、复制等功能。帮助文件显示单片机相关概念、指令附录、指导手册和每个实验的具体信息等。编译器可实现对 MCS-51 汇编语言的编译功能，其主要目标是将单片机汇编语言源程序文件翻译成写入单片机的 Intel HEX 文件和为用户提供源码与目标码的对照的列表（list）文件。整个编译过程按照词法分析、语法分析、语义分析、目标代码生成等几个阶段进行，通过两遍扫描实现对 MCS-51 汇编语言的翻译，并生成写入单片机的目标文件、为用户提供源码与目标码对照的列表文件的信息以及编译过程中出现的错误反馈信息[2]。代码下载功能实现将生成的目标代码（Intel Hex 格式）通过串口直接下载到单片机的功能，从而省去了以往必须要使用编程器才能下载的烦琐步骤及相关实验设备。

本课题隶属于河南省科技攻关项目《面向应用的 ES 教学实验系统研制》（项目编号：0524220074）

第一章 绪论

1.1 选题依据与意义

嵌入式技术在计算机、电子、通信、控制等学科或专业中具有极为重要的地位，对我国的经济发展和国防建设有非常重大的影响。中国有世界上最大的家用电子产品消费市场，同时，对于现代化的医疗、测控仪器和机电产品也需要有专用的嵌入式系统软件的支持。各种各样的嵌入式设备已经应用到生活的各个方面，包括 PDA、机顶盒、移动终端、工业控制设备等等。这些需求都极大地刺激了嵌入式系统的发展和产业化的进程[3]。21 世纪之初的十年被人们称为是"后 PC 时代"，嵌入式系统无处不在。

为了促进嵌入式技术在我国的发展，中国计算机学会微机专业委员会从 2000 年以来，每年都要举办一次全国性的"嵌入式系统技术与应用研讨会"。我国嵌入式技术的发展在很大程度上取决于学校的教学。为了进一步做好嵌入式系统的教学工作，2003 年 10 月由清华大学、北京航空航天大学、《单片机与嵌入式系统应用》杂志社、《电子产品世界》杂志社以及多家中、外著名公司联合在北京召开了主题为"学习嵌入式、研究嵌入式、使用嵌入式"的首届高校嵌入式系统教学研讨会。中国科学院院士沈绪榜、许居衍、中国微机单片机学会理事长陈章龙等出席了会议。清华大学邵贝贝教授、北京航空航天大学何立民教授等分别作了"嵌入式教学体系探讨"以及"嵌入式系统教学研究"等重要报告，出席会议的其它高校院、系领导、教学主任、实验中心主任、一线教师、实验教师等在会上共同研讨了嵌入式教学体系发展与改革方案。与会者普遍认为目前所使用的单片机教学实验系统大多是九十年代初研制的，已远远落后于嵌入式技术现在的发展水平。对于 MCU (Micro Controller Unit 即微控制器）系统，许多新的有很高实用价值的技术，如 CAN 总线、USB 接口、RS485、I2C、SPI 以及与 Internet 接口技术等都被排斥在实验内容之外，因此，尽快开发新一代面向应用的单片机教学实验系统变得非常迫切和重要。

河南省科技攻关项目《面向应用的 ES 教学实验系统研制》（项目编号：0524220074）就是一个针对目前教学实验系统的现状，开发和研究新一代教学实验系统的项目。本课题实现了该项目中的软件编制。本文将用 VC++6.0 开发单片机汇编

语言程序的IDE。

IDE即集成开发环境(Integrated Development Environment),IDE软件是进行软件开发必不可少的工具软件。不仅仅是普通PC机软件开发有此需求，在进行嵌入式软件的开发过程中，如果有一个方便、成熟、稳定、功能强大的IDE，能更加显著地提高开发效率[4]。本文主要介绍了在探索嵌入式系统软件IDE开发技术的过程中，如何设计与实现一个基于VC++的MCS-51单片机汇编语言的IDE。

本系统在MCS-51单片机上实现了代码下载功能，在计算机上调试好的程序既可以通过编程器写入单片机或EPROM中，也可以直接通过计算机的串口下载到应用系统中。系统运行时，程序可存放在单片机内部，也可存放在单片机外部的EPROM或Flash存储器中，非常方便灵活。

1.2 国内外研究现状

学习单片机汇编语言理应有一个好的集成式的程序开发系统，其界面直观且能提高效率，可使初学者和程序员省去频繁输入命令的麻烦。

单片机IDE(集成开发环境)是开发单片机应用系统必不可少的工具之一。目前，单片机IDE主要由世界几大专门面向嵌入式系统的软件公司提供：德国的 Keil、荷兰的Tasking，美国的Franklin，台湾的 IAR 等等。其中Keil的产品μVision2和Tasking的产品EDE品质尤为出众，知名度颇高。研究它们的设计原理和各自的优缺点，开发适合自己的IDE，成为摆在国内计算机界面前的一个迫切任务。

虽然目前市场上有许多MCS-51汇编语言编译器，就其编辑、编译、连接、调试、运行等主要功能来说，大家虽然都已基本实现了，但也有不足之处，比如说有的系统不能实现文件下载，有的费时费力不够直观且极易出错等，这些都在一定程度上为初学者带来了不便。而且，目前对汇编语言编译器的开发大多采用Turbo c、VB语言等，但是随Visual C++的广泛应用，其强大的功能、便捷性和直观性日益体现出来，其"可视"的资源编辑器与MFC类以及应用程序向导，为快速高效地开发出功能强大的Windows应用程序提供了极大的方便[5]。同时，在编译器的整体设计中要利用串口进行数据通讯。而在串口通信的各种开发工具中间，VC由于其功能的强大和灵活，同时也得到了Microsoft的最大支持，利用RS-232进行数据信号的采集和传递也成为VC编程的一大热点[6]。所以一般在涉及硬件操作的通讯编程中，大都推荐使用VC作为

开发工具。鉴于此，本文将用VC++6.0来设计和实现面向应用的单片机教学实验系统中的MCS-51单片机汇编语言IDE，此IDE不仅应具有编辑、编译功能，而且还将具备专门为ES教学实验系统研制的能将编译过的汇编程序（即Intel Hex格式）通过串口直接下载到单片机的功能。这样一来，不但使编写程序变的更加便捷、直观，同时也省去了以往必须要使用编程器才能使程序下载到单片机的繁琐步骤及相关实验设备。

1.3 论文内容

第一章主要介绍选题依据与意义、国内外研究现状、研究方法等；第二章主要探讨了一些理论基础和相关概念，如单片机的相关理论知识，一般 IDE 的主要功能、开发环节，还有对编译器的相关概念、理论知识的介绍，同时对 HEX 文件的格式做了简单介绍等；第三章主要研究了该软件系统的具体设计，首先介绍了对编程工具 VC++6.0 的选择和其特点，以及编辑器的界面设计和具体实现的步骤，研究了编译器设计模块中的词法分析、语法分析和符号表管理等具体模块的设计，然后介绍了串口通信的基本概念、Windows 串口通信 API 函数以及串口通信的设计实现过程和一些人性化设计，如用户登录模块、实验演示模块和帮助文件模块，最后简单介绍了该系统的使用方法；第四章主要介绍了该系统在高校单片机实践教学中的应用；第五章给出了作者的结论和建议。

第二章 MCS-51 单片机汇编语言程序 IDE 的设计

2.1 单片机概述

单片机又称微控制器，在工业控制中占据很重要的地位，目前8位单片机仍占有单片机市场60%以上的份额。MCS-51系列单片机属于8位单片机，具有很强的功能，适用范围广，既可构成功能很强的复杂系统，也可组成较简单的应用系统[7]。单片机靠写入存储器中的机器语言程序的执行来实现对各种系统的控制。

随着微电子工艺水平的提高，近二十余年来单片微型计算机有了飞跃的发展。世界上著名的集成电路制造商纷纷推出各自的产品，单片机型号之多，已达到难以统计的地步。以著名的芯片制造业巨头——Intel 公司为例，早期推出的 MCS-48 系列单片机是一种功能简单、寻址范围很有限的低性能8位单片机，这类单片机早已被稍后推出的MCS-51系列8位单片机所取代。在MCS-51系列单片机的内核8051/80C51的基础上，ATMEL公司、Philips公司、Siemens公司等很多大公司，纷纷推出了名目繁多的派生芯片。这类单片机是目前世界上应用最广的一类单片机。它的繁衍发展之路也是其它系列单片机发展的共同道路[8]。归纳起来，它沿两条路发展：

① 改进集成电路制造工艺，提高芯片的工作速度，降低工作电压和降低功耗。早期的 8051 的最高振荡器频率为 12MHz，一个机器周期为1μs。Intel 80C51ZX 的一个机器周期仅为1/6μs。早期的8051的正常工作电压都是5V，而 Philips 公司的80CL51/80CL410可工作于1.8V的低电压。80CL51/80CL410是全静态设计的，当芯片采用外部时钟时，可工作于直流状态，即可把外部时钟完全关掉仍能够保持住芯片内部状态；当时钟重新加上时，芯片将继续正常工作。当外部时钟停止时，芯片的消耗电流只有 1μA。

② 在保留共同的 CPU 体系结构、最基本的外设装置（如异步串行口、定时器等）和一套公用的指令系统的基础上，根据不同的应用领域，把不同的外设装置集成到芯片内，在同一家族内衍生出各种型号的单片机。Intel 公司在 8051/80C51 的基础上，已衍生出 10 种共 50 多个型号的芯片；Philips公司在 80C51 的基础上衍生出 20 多种近 50 个型号的单片机。

2.2 MCS-51 单片机指令系统及其指令格式概述

MCS-51单片机指令系统具有功能强、指令短、执行快等特点，共有111条指令和8条伪指令。从功能上可划分成数据传送、算术操作、逻辑操作、程序转移位操作等五大类；从空间属性上分为单字节指令(49条)、双字节指令(46条)和最长的三字节指令(只有16条)。从时间属性上可分成单机器周期指令(64条)、双机器周期指令(45条)和只有乘、除法两条4个机器周期的指令。可见，MCS-51单片机指令系统在存储空间和执行时间方面具有较高的效率。

指令系统中的指令描述了不同的操作，不同操作对应不同的指令。但结构上，每条指令通常由操作码和操作数两部分组成。操作码表示计算机执行该指令将进行何种操作，操作数表示参加操作的数的本身或操作数所在的地址。MCS-51单片机的指令有无操作数、单操作数、双操作数三种情况。汇编语言指令有如下的格式：

[标号：] 操作码助记符 [目的操作数][，源操作数] [；注释]

伪指令不要求计算机做任何操作，也没有对应的机器码，不产生目标程序，不影响程序的执行，仅仅是能够帮助进行汇编的一些指令。它主要用来指定程序或数据的起始位置，给出一些连续存放数据的地址或为中间运算结果保留一部分存储空间以及表示源程序结束等等。不同版本的汇编语言，伪指令的符号和含义可能有所不同，但基本用法是相似的[9]。如表2-1所示：

表 2-1 伪指令表

序号	名称	说明
1	ORG	汇编起始指令
2	DB	定义字节指令
3	DW	定义字指令
4	DS	定义空间指令
5	EQU	等值指令
6	BIT	位地址符号命令
7	END	汇编结束指令
8	DATA	数据地址赋值命令

2.3 集成开发环境概述

IDE的全称是Integrated Development Environment，直译为集成开发环境，它包括编辑、编译、调试等功能。

2.3.1 IDE 的主要功能

通过对于上述IDE软件的分析，以及其他成熟的比如Visual Studio等软件的分析，可以看出，一个理想中的IDE软件应带有下面的功能。

（1）代码编辑。使用IDE的过程中，用户60%的时间都是在进行代码编辑。理想的IDE应该尽量在代码编辑中提供对用户的帮助，改进开发效率。具体的手段包括提高程序的可读性(改善显示效果)、简化用户的输入动作(如代码自动生成)、方便用户对程序的阅读(函数列表、交叉索引等)。

（2）Project管理及编译控制(此处不是指一般意义上的项目管理，而是一个软件包的编译选项等等)。在使用IDE编辑软件的时候，有些软件引入了项目(Project)的概念。例如Visual Studio就有专门的项目文件来记录当前软件的路径、编译信息、文件信息、类信息等等[10]。也有不少Linux下面的IDE并没有专门引入Project的概念，它们认为使用GNU的Makefile系统就可以很方便的实现软件的项目管理和编译控制，再加上Automake和Autoconf等生成Makefile的辅助程序，在编译软件的过程中不需要其他额外信息了。本文也未引入Project管理，着重研究编译控制。

（3）平台管理(一般IDE没有，嵌入式需要针对不同开发板及环境预设一些基本的Profile，方便用户选择使用)。针对嵌入式开发环境的目标平台多样性的特点，可以引入平台信息管理功能。针对每个CPU体系、Processor的型号、开发板的名称，将所有的平台信息分类记录在平台信息管理模块中。用户使用一个新的开发平台的过程中，就可以选取相应的配置文件，比较简单地获得他人已经在本平台下成功使用过的开发环境、操作系统等。这样就能辅助开发人员很快地进入最终应用开发的阶段。

（4）软件调试。IDE的调试功能是不可或缺的。早期的Turbo C 2.0软件，就以简便且强大的调试功能而深受好评，广泛流行。软件调试，通常是对本地运行的软件，利用操作系统提供的Trace功能，控制它的执行，并获取它的内部信息，从而检查出软件执行过程中所出现的错误。而在嵌入式系统开发中，还必须支持远程调试。需要说明的是，在本文中，并未能够实现上面的全部功能，主要着重点在于代码编辑、调

试、连接和下载部分。也就是在PC机上控制目标板中程序执行，从而检查软件错误。另外，嵌入式开发中，还可能利用到强大的JTAG在线调试功能，利用硬件断点寄存器等方式实现对系统软件的调试。

2.3.2 单片机应用系统软件开发步骤

综上所述，在嵌入式软件开发中，由于目标平台的资源有限性，所有的开发工作都是在Host PC上面完成的，所以，导致嵌入式系统中每一个软件的开发过程必须经历下面的环节：

（1）源代码编辑

嵌入式开发主要使用的是C，C++、汇编等类型的源程序。因此，使用一个好的代码编辑软件可以提高这个环节的工作效率。

（2）编译生成目标程序

在源代码编辑完成之后，必须要进行编译、出错处理、连接，然后生成目标代码。

（3）下载或编程

代码总是要在目标平台上才可以执行的。所以编译生成的目标程序首先必须要下载到开发板。代码下载的步骤比较复杂。例如，下载Boot loader的动作是根据Processor所支持的方式来实现的，每个平台各不相同；而下载Linux kernel的动作又是根据Boot loade：所支持的方式来实现的，每个Boot loader也各不相同；下载应用程序的方式又是根据操作系统所提供的方式来实现的，是网络方式还是串口方式，通讯协议又是哪种，都无法确定。所以，这个步骤难于统一完成。本文采用的是通过串口将目标文件下载到单片机上。

（4）调试

通常的软件调试，都是在本机运行调试软件(比如gdb)来完成的。在嵌入式环境中，开发板上可能由于资源所限，无法运行gdb。因此，嵌入式环境中的软件调试，通常采用的是软件远程调试的方法或者JTAG在线(In-Circuit)调试的方法。前者，即软件远程调试，是在开发板上运行一个小巧的调试服务器(例如gdbserver)，由PC机上的gdb通过串口或者Ethernet进行连接。所有的符号分析以及源代码查看、动作控制都是在PC机上完成的，而具体的执行则是由gdbserver进行控制，并将状态信息反馈回PC上的gdb，进行信息处理之后，再显示给用户以供分析[11]。这种方式提供了基本的调试能力，对于一般应用软件来说，已经完全够用了。

2.3.3 开发系统软件实现方案

单片机开发系统软件包括下面两个方面的内容：单片机程序的建立（Build）和仿真调试（Debug）。

（1）单片机程序的建立

建立过程可分成两步：编译和连接/定位。编译只把源程序翻译成机器码，此过程由编译器完成。经过编译产生的机器码的地址是浮动的，即地址是相对的，并不能在单片机中执行，要用连接定位器（简称连接器）将它们在单片机的绝对地址空间中进行定位，如Intel 196单片机代码必须从2080H单元开始放置，中断向量地址都是特定单元，连接器进行连接时要把相应的代码装载到特定的地址中[12]。

目前编译器和连接器一般是由单片机厂家提供，也有一大批由从事编译器/连接器开发的第三方供应商提供，各厂家的软件都有些差异和各自的特色。Intel 公司早期是提供 16 位单片机的编译器和连接器的，后来把这一任务转交给了第三方供应商，如 Tasking 公司。目前国内的开发者大多采用国外的编译器和连接器，也有某些汇编语言的编译器是自行开发的。

（2）单片机程序的调试

只有编译器和连接器并不能进行开发，还要一个仿真运行的环境，能够将单片机程序放入仿真 RAM 或单片机的程序存储器中，控制单片机的运行，如单步运行、到断点、停止，并将运行过程中的状态和内部寄存器内容读出来[13]。早期的开发环境是分离地运行不同的指令。如在DOS操作系统下，先是用文本编辑软件写源程序，然后在 DOS 系统下运行编译和连接程序，完成后再进入仿真软件进行仿真调试。但目前的开发软件大都将所有功能集中于一体，并建立在 Windows 平台上，这种软件就是本文要研究的集成开发环境（IDE）[14]。

一般采用可视化的编程工具实现单片机集成开发环境的设计，如用 Visual Basic、Visual C++等。软件环境主要是进行调试的一个环境，它的优劣决定单片机开发的效率。所以这种软件应当具有丰富的功能和友好的界面，而且要便于调试和发现程序中的错误。笔者用 Visual C++设计了一个使用相当方便和可靠的集成开发环境。

2.4 编译器概述

在整个IDE的开发中，编译模块的开发至关重要。

从理论上说，构造专用计算机来直接执行用某种高级语言编写的程序是可能的，但是，目前的机器能执行的是非常低级的语言，即机器语言。那么一个基本的问题是，高级语言最终是怎样在计算机上执行的呢？答案是编译。编译，代表从面向人的源语言表示的算法到面向硬件的目标语言表示的算法的一个等价变换。通常情况下，人们将能够完成一种语言到另一种语言变换的软件称为翻译器，而本文要研究的编译器就是其中的一类。编译器是程序员将命令翻译成可以在计算机上执行的代码的软件开发工具，它的特点是目标语言比源语言低级。编译器的工作可以分成若干阶段，每个阶段把源程序从一种表达形式变换成另一种表达形式。在计算机研发方面，编译器与处理器的研发必须协调统一地进行，因为只有编译器的设计充分利用了处理器架构的优势，处理器才能达到最大效能。同时，一个好的编译器可为软件开发和维护提供更有力的支持，可以构造出各种程序分析、理解和开发工具，这些工具对于新的软件开发模式的实践也有很好的支持。由于编译系统具有上述两个重要意义，因此一个硬件系统如果缺乏一个良好的编译系统的支持，一方面不能充分发挥它的性能，另一方面，相关的各种软件无法有效地开发。一个成功的硬件系统（如CPU、并行计算系统等）不仅要有很好的硬件特性，而且要有丰富的软件支持，否则就不可能在市场上生存。目前，这些软件的开发，除了极少的核心部分可能使用汇编语言外，绝大部分的程序是用高级语言开发的，将这些高级语言程序翻译成正确的、高效的、在目标系统上运行的目标程序，必须依赖于编译器的支持。而且，不同的硬件系统有不同特点，同一个软件在不同的硬件平台上的高效执行所要实施的程序优化也是不同的，面向具体硬件平台的优化编译技术可以很好地解决这方面的问题。

编译器是一种特殊的程序，它可以把以特定编程语言写成的程序变为机器可以运行的机器码。在文本编辑器中写好一个程序，然后程序员可以运行相应的编译器，通过指定需要编译的文件的名称就可以把相应的源文件（通过一个复杂的过程）转化为机器码了。编译器，是将便于人编写，阅读，维护的高级计算机语言翻译为计算机能识别，运行的低级机器语言的程序。它将源程序（Source program）作为输入，翻译产生使用目标语言（Target language）的等价程序[15]。源程序一般为高级语言或汇编语言（High-level language），如 Pascal，C++等，而目标语言则是目标机器的目标代码（Object code），有时也称作机器代码（Machine code）。本文中，笔者将目标程序代码定为 Intel hex 格式。

一个现代编译器的主要工作流程如下：

源程序（source code）→预处理器（preprocessor）→编译器（compiler）→汇编程序（assembler）→目标程序（object code）→连接器（链接器，Linker）→可执行程序（executables）。

2.5 HEX 文件格式简介

本文研究的 IDE 软件系统会生成两种文件，即可执行文件和列表文件。其中可执行文件的格式就是标准 Intel HEX 格式，其优点是可执行程序既可以直接下载到应用系统中，也可以通过编程器写入到单片机中，适用范围更宽。Intel HEX 文件经常被用于将程序或数据传输存储到 ROM、EPROM，大多数编程器和模拟器使用 Intel HEX 文件[16]。Intel hex 文件常用来保存单片机或其他处理器的目标程序代码。它保存物理程序存储区中的目标代码映象。

Intel HEX 文件是记录文本行的 ASCII 文本文件，由若干行组成。在 Intel HEX 文件中，每一行是一个 HEX 记录，由十六进制数组成的机器码或者数据常量组成，两个字符表示一个字节的值。

每个记录都具有如下的形式：

":LLAAAATTDD...DDCC"

":" 记录的起始标志；

"LL" 记录长度，表示该记录中的数据字节数；

"AAAA" 数据装入的首地址（16 位）；

"TT" 记录类型，00 表示数据记录，01 表示文件结束；

"DD" 数据值（字节）；

"CC" 校验和。即将其本身与记录中除起始标志外的所有字节相加应为 0。

如最小系统实验的可执行文件第一行内容为：

"3A 3130 30303030 3030 3735 3831 3630 3738 3041 4432 3932 3734 3038 3131 3235 4332 3932 3734 3038 3131 3231 0D0A"，其中：

第 0 字节"3A"为":"，即记录起始标志；

第 1、2 字节"3130"为记录长度，其值为：10H，即该记录含有 16 个字节的数据；

第 3、4、5、6 字节"30303030"为记录装入首地址，即"0000H"；

第 7、8 字节"3030"为记录类型，其值"00H"表示数据记录；

第 9-40 字节共 32 个字节，分别为“3735 3831 3630 3738 3041 4432 3932 3734 3038 3131 3235 4332 3932 3734 3038 3131”，这是数据值，每两个字节对应一个字节的机器码，分别为：75 81 60 78 0A D2 92 74 08 11 25 C2 92 74 08 11；

第 41、42 字节“3231”为校验和，其值为十六进制数“21H”；

最后两个字节（0D0A）为回车和换行。

列表文件也是比较重要的文件，因为在列表文件中，可以清楚地看到每条指令对应的机器码，虽然没有必要记住每个指令的机器码，但了解每个指令的机器周期数和对应的机器码字节数是很有必要的。如最小系统实验的列表文件如下：

```
                    1       ; 实验一，最小系统实验
                    2       ; 实验目的：了解复位和时钟电路的工作原理
                    3
0000                4               ORG 0000H
0000                5   MAIN:
0000    758160      6                MOV SP,#60H      ; set  stack pointer
0003    780A        7   START:      MOV R0,#10
0005                8   LOOP1:
0005    D292        9               SETB P1.2
0007    7408        10              MOV A,#8         ; delay 500 ms
0009    1125        11              ACALL DLY
000B    C292        12              CLR P1.2
000D    7408        13              MOV A,#8
000F    1125        14              ACALL DLY
0011D8F2            15              DJNZ R0,LOOP1
0013    7805        16              MOV R0,#5
0015                17  LOOP2:
0015    D292        18               SETB P1.2
0017    743C        19              MOV A,#60        ; delay 2 seconds
```

```
0019    1125      20              ACALL DLY
001B    C292      21              CLR   P1.2
001D    743C      22              MOV A,#60          ; delay 2 seconds
001F    1125      23              ACALL DLY
0021    D8F2      24              DJNZ R0,LOOP2
0023    80DE      25              SJMP START
                  26
0025    F530      27  DLY:   MOV 30H,A
0027    7F64      28  DEL0: MOV R7,#100        ;on   6MHz，delay 50ms
029     7E7D      29  DEL1: MOV R6,#125
002B    DEFE      30  DEL2: DJNZ R6,DEL2
002D    DFFA      31        DJNZ R7,DEL1
002F    D530F5    32        DJNZ 30H,DEL0
0032    22        33        RET
                  34
                  35        END                  ; program end
```

生成的 HEX 格式目标文件用“记事本”软件就可以轻松查看。如图 2-1 所示：

图 2-1 HEX 文件的查看

第三章 详细设计

3.1 编程工具的选择及其特点

微软的 Visual Studio 6.0 集成了 Visual C++、Visual Basic、Visual FoxPro 等编程工具。其中 Visual C++ （简称 VC）是具有面向对象、功能丰富和可视化等诸多优点。VC 兼有高级和低级语言的双重性，与目前流行的 Visual Basic、Delphi 等众多开发工具相比，其功能更为强大，灵活，执行效率高。Visual C++采用面向对象的思想分析设计和开发程序，容易解决大型复杂问题，所开发软件维护起来比较容易，开发效率高，对象可重新透明定位到信息平台甚至跨网络定位，可使软件开发者摆脱全部设计性的活动，过渡到集成性组装式的工作[17]。VC 可以方便的实现一些底层的调用，如直接对 I/O 地址操作等，VC 在多线程、网络通信、分布应用方面，有着不可比拟的优势，用其开发的软件，执行效率高,故适合开发一些针对 Windows 系统级的软件项目，也多用于单片机，工业控制等软件开发，因此，开发高效、灵活的文件操作程序和优秀的基于通信的程序方面一般都选用 VC[18]。

Visual C++ 编程必须完成两件事：做界面和写代码。做界面，目的是从 Visual C++ 提供的控件中挑选出合适的来组成自己的程序与用户交互的界面。写代码，目的是把各控件联系起来，完成特定的功能。界面与代码相结合才能组成完整程序。集成开发环境的主要功能模块如图 3-1 所示。

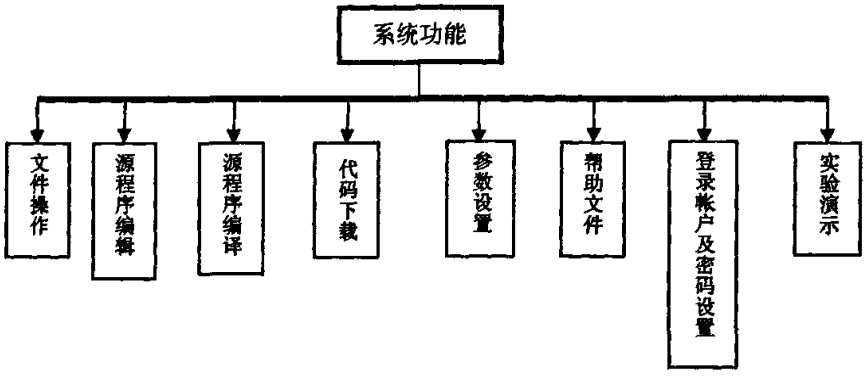

图 3-1 系统总体设计图

3.2 编辑器模块的设计

3.2.1 界面设计

为能使单片机汇编语言的初学者在较短时间内掌握好这门语言，其开发环境应具有良好的人机交互功能，做到直观、易操作。编辑框界面的设计主要有下列几部分：

1. 文件操作：新建、打开、保存、关闭、打印和退出等；
2. 文本编辑：剪切、复制、粘贴、查找和撤消等；
3. 编译：编译连接、装载单片机程序；
4. 实验演示（包含典型的 24 个单片机试验）和参数设置。
5. 系统登陆设置

3.2.2 具体操作

具体操作：在 VC 中新建一个项目（如 Yx），需要说明的是在 Projects 中要选择 MFC AppWizard[exe]，在 MFC AppWizard 中选择 Multiple documents，在 Base class 中要选择 CEditView 类，MFC 将自动生成的 CAboutDlg 、CMainFrame、CChildFrame、CYxApp、CYxDoc 和 CYxView 类，然后在 Resource 中实现上述菜单的设计，并在 classwizard 中添加各自的函数，并编辑其相应的代码[19]。

其中，如“文件”菜单中的“新建”等常见功能菜单的设计，VC都提供了具体的代码模块，因此，在Resource中实现上述菜单功能的时候，只需在相应的菜单上按“Enter”键，在弹出的“Menu Item Properties”对话框（如图3-2所示）的“ID”一栏中选择“ID_FILE_NEW”就可以了[20]。另外，新建、打开、保存、关闭、打印、退出、剪切、复制、粘贴、查找和撤消等基本功能都可以通过这种方式来完成。由此可见，对于一些基本的界面实现，VC++6.0开发环境确实为我们提供了一个轻松快捷的途径。

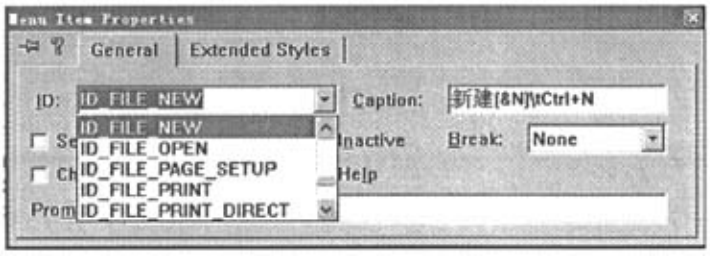

图3-2 菜单项属性对话框

3.3 编译器的设计

3.3.1 项目分解

该IDE中编译器的任务是对输入的汇编源程序进行扫描，从中找出所有词法和语法的错误并给以反馈，然后生成目标代码文件和提供查看信息的列表文件。简单的说，就是将输入的*.asm 源程序文件，转换为*.hex目标代码文件和*.lst列表文件输出。其框架图如图3-3所示：

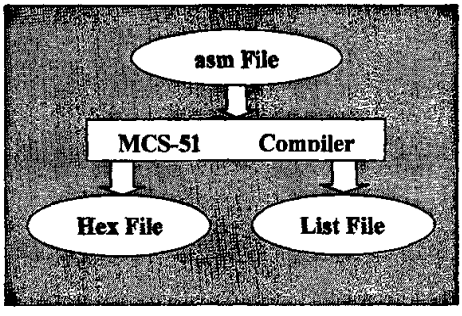

图3-3 编译器总体框架

一个编译器的整个工作流程是划分成一个个阶段进行的，每个阶段都将源程序的一种表示形式转换成另一种表示形式，各个阶段进行的操作在逻辑上是紧密联结在一起的。这几个阶段是：词法分析、语法分析、语义分析、目标代码生成。另外两个重要的工作符号表的管理和出错处理贯穿以上的所有阶段，编译过程中源程序的各种信息被保留在不同的表格里。编译时自始至终涉及到符号表格的构造、查找和更新[21]。如果编译过程中发现源程序有错误，编译程序应报告错误的性质和错误发生的地点，并且将错误所造成的影响限制在尽可能小的范围内，使得源程序的其余部分能继续被编译下去，这些工作就是出错处理。编译器整个工作流程如图3-4所示：

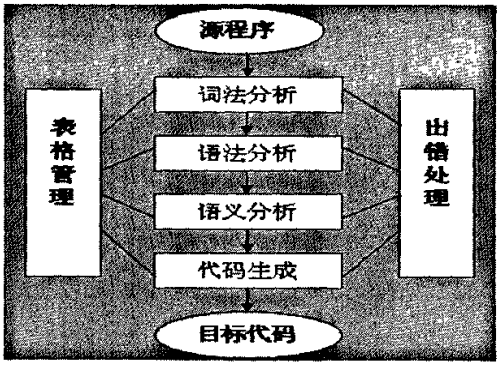

图3-4 编译器的工作流程

词法分析阶段的主要任务是自左向右扫描源程序的字符串，并识别出一个个的单词符号。这里所谓的单词是指逻辑上紧密相连的按词的组成规则结合起来的一组字符，它们具有一定的意义。把每个单词的 ASCII 码序列替换成所谓的机内表示—Token 形式，这时还需要检查词法错误。

语法分析的主要任务是检查源程序的形式语法错误，每当发现错误时将输出有关信息。扫描对象可能是源程序的 ASCII 码序列，也可能是词法分析后的 Token 序列。前者情形，词法分析程序作为语法分析程序的子程序[22]。

语义分析的任务是收集符号信息，登记在符号表格里，对汇编伪指令的语义进行解释，完成相应的动作。扫描的对象通常是语法分析后的结果。此时源程序的 Token 序列已经变换成没有错误的符合语法的 Token 流。汇编伪指令的功能是改变编译器的状态并将一些必要的信息加入到目标文件中去。

另外ORG段的选择，地址计数器的计数，指令长度的计算等都在这个阶段中完成。语义分析阶段同样进行着对错误的处理。

代码生成阶段的任务是根据汇编助记符的各种寻址方式决定它的目标代码。这时候的Token流在形式上已经比较一致，符号信息都已经登记在各种表格里。目标代码生成之后，还要依据*.hex文件的格式把目标代码写入文件，最后还要产生列表文件*.lst,为用户提供源码与目标码的对照[23]。

3.3.2 详细设计

面向对象的程序设计的好处在于使得计算机解决问题的形式更符合人的思维活

动和概念，便于程序的模块化、数据抽象、信息隐蔽、增加软件的可扩充性和可重用性，并能控制维护软件的复杂性。VC++在面向对象程序设计方面的确具有较大的优势，易于实现。

（1）总体模块设计

本文采用两遍扫描方式，在两遍扫描方式中，第一遍扫描主要是建立符号表和确定源程序每一行的偏移地址，同时检查汇编程序是否有错误。第二遍扫描是利用第一遍扫描所生成的符号表，产生可供微处理器执行的机器代码文件，并产生一个列表文件。第一遍扫描完成源程序的语法检查、建立符号表、提取数据和产生错误信息等功能[24]。其主要功能是提取数据段和建立符号表。第一遍扫描中3个最重要的数据结构是符号表、伪操作表和机器指令表。

符号表主要负责标号，字符名称（变量）的记录，并且判断他们的合法性，主要操作有：向符合表中添加标号或变量，记录他们的值，类型，是否定义，是否已经使用过等一些信息，查找标号或者变量等[25]。为此在扫描时要确定指令的长度，对伪指令作适当的处理，以便确定符号对应的地址与名字的值。第一遍扫描过程中可能出现的错误有两种，一是多重符号定义，另一种是指令操作码不合法。多重符号定义是指重定义一个已经定义过的符号。而指令操作码不合法也包括伪指令操作码不合法。第一编编译过后形成 *.lst1文件，它中间会有一些行没有完全编译好，因为有些地址不能确定，要在第二遍编译才能回填，第二遍编译在CSeccomp()类中实现。但第一遍扫描结束之后符号表就建立起来了，为第二遍扫描提供了基础。第二遍扫描的中心任务是产生汇编源程序的目标代码，这个过程所用到的数据结构是第一遍扫描产生的符号表以及指令的操作码表。第二遍扫描和第一遍扫描一样也是对源程序进行逐行扫描，一开始也是将地址计数器清零。标号域忽略不计，操作符再次被测试以确定是机器指令还是伪指令。如果是伪指令，若是END则程序结束，否则进行相应伪操作的数据处理；如果是机器指令，则查找机器指令表以确定其操作码，再根据指令类别生成最终的指令目标代码并修改地址计数器值。所有各类语句都将生成相应的目标代码，对于非法语句给出错误信息。在具体编程中，CAnalyse::ProcessOneLine(CString &sLine)的主要功能是根据读到源程序每行的第一个词，语法分析该行语句属于哪一种句型，做出相应的处理，即调用相应的处理函数（模块）。

另外，在处理模块中，最重要的是指令语句的处理。下列代码是指令语句的实现过程，首先分析指令语句的操作数（函数Analyse（）），接着转换操作数的值并且为该指令语句形成“字符串标志”（用于匹配111条指令语句中的某一条），然后根据形成的标志，定位该指令语句属于111条语句中的哪一条，最后，根据位置调用相应的处理函数进行处理。

```
int COpcodeOp::Process(CString sLine, CString sOpcode)
{
    /*  功能：进行操作码处理
    入口参数：sLine，表示需要处理的一行指令语句；
              sOpcode，表示这一行指令语句的指令码
    返回值：  正确则返回OP_SUC；
              错误则返回错误代码
    返回的错误代码如下：
    OP_OPDATA_WRONG表示操作数错误；      OP_EVAL_WRONG表示操作数赋值错误
    ERROR_LANGUAGE_WRONG表示语法错误；ERROR_DATA_OVER表示数据越范围
    */
    int ret;
    ret=Analyse(sLine);
    if(OP_ANALYSE_SUC!=ret)
        return OP_OPDATA_WRONG;
    ret=EvalParAndCreLab();
    if(OP_EVAL_SUC!=ret)
        return OP_EVAL_WRONG;
    int pos=Locate(sOpcode);
    ret=ProcessOpcode(pos);
    return ret;
}
```

（2）指令系统模块设计

MCS-51共有111条可执行指令和8条伪指令，这个类的对象存储了用户使用的计算机对应的指令系统的信息，如指令的序号、操作码和操作数对应的Token 码、所需操作数的个数，机器指令占用的长度。MCS-51的111条指令描述如表3-1所示：

表3-1 MCS-51指令描述表

序号	Token 码				字节数	操作数个数
	操作码	操作数1	操作数2	操作数3		
0	ACALL	addr11			2	1
1	ADD	A	Rn		1	2
2	ADD	A	direct		1	2
…	…	…	…	…	…	…
18	CJNZ	A	direct	rel	3	3
…	…	…	…	…	…	…
108	XRL	A	#data		1	2
109	XRL	direct	A		1	2
110	XRL	direct	#data		2	2

为了在编译时能够判断所输入语句是否正确，需要将输入的语句与上述111条指令进行匹配，因此必须先建立了一个指令表和操作码表，将上述指令存储到计算机中。

在对指令表（包含0-110条指令）的扫描中，如果只比较第一个操作数就能够得到正确的匹配结果，那么再继续比较下去就是多余的，因此，在这里引入一个SetValue方法，定义的变量“size”表示该条指令所占的字节数，“lab”表示第一操作数标识符号，“lab2”表示第二操作数标识符号，“flag”是一个整型变量，用来说明是否有必要进行“lab2”的比较，“0”代表“否”，“1”代表“是”。代码实现如下：

```
class COpcodeTabData
{
private:
 int m_nNum;          //序号
 CString m_sLab;      //第一标识符号
```

```
CString m_sLab2;    //第二标识符号
int m_nByteSize;    //字节数
int m_nFlag;
public:
COpcodeTabData()
{
    m_nNum=0;
    m_nByteSize=0;
    m_nFlag=0;
}
void SetValue(int size,CString lab,CString lab2="",int flag=0)
{   //flag 用来说明是否有必要进行lab2的比较，0：没必要；1：有必要。
    m_nFlag=flag;
    m_sLab=lab;
    m_sLab2=lab2;
    m_nByteSize=size;
}
void SetNum(int num)
{ m_nNum=num;
}
CString GetLab()
{ return m_sLab;
}
CString GetLab2()
{ return m_sLab2;
}
int GetByteSize()
{ return m_nByteSize;
```

```
}
int GetNum()
{ return m_nNum;
}
int GetFlag()
{ return m_nFlag;
}
};
```

如下是存储111条指令的代码实现：

```
class COpcodeTable
{
private:
 COpcodeTabData m_Data[111];
public:
 COpcodeTable()
 {
     for(int i=0;i<111;i++)
         m_Data[i].SetNum(i+1);
     m_Data[0].SetValue(2,"_AD");
     m_Data[1].SetValue(1,"_A_Rn");
     m_Data[2].SetValue(2,"_A_AD");
     m_Data[3].SetValue(1,"_A_INRi");
                                  ...
                                  ...
     m_Data[109].SetValue(2,"_AD_A");
     m_Data[110].SetValue(3,"_AD_D");
 }
```

在111条单片机汇编语言指令中，总共有44个操作码，因此，在查找一条指令的

时候，只要先判定出操作码，即可以只在该操作数相关的几条记录中查找，而不用与111条指令中的每一条都相匹配，这样就大大减少了查找时间，极大的提高了效率。这样需要建立一个操作码表。在操作码表（包含0-43个操作码）中，变量“m_sName”表示该操作码的字符串表示，“m_nNum”表示该操作码在操作码表中的序号，变量“m_nTypeCnts”表示该操作码所包含的具有相同操作码的语句的个数，变量“m_nStPosInInsfile”表示该操作码在111条指令表中的开始存放位置。此时又引入了一个“SetValue”方法，与上个指令表中的“SetValue”不同，它包含“sName”、“nNum”、“nTypeCnts”和“nStPos”四个变量，分别对应于“m_sName”、“m_nNum”、“m_nTypeCnts”和“m_nStPosInInsfile”。代码实现如下：

```
class Opcode : public CObject
{//类COpcode 的数据格式
public:
 CString m_sName;//该操作码的字符串表示
 INT m_nNum;//该操作码在操作码表中的序号
 INT m_nTypeCnts;//该操作码所包含的具有相同操作码的语句的个数
 INT m_nStPosInInsfile;//该操作码在111条指令文件符号表中的开始存放位置
 Opcode()
 {
     m_nNum=0;
     m_nTypeCnts=0;
     m_nStPosInInsfile=0;
 }
 void SetValue(CString sName,int nNum,int nTypeCnts,int nStPos)
 {
     m_sName=sName;
     m_nNum=nNum;
     m_nTypeCnts=nTypeCnts;
     m_nStPosInInsfile=nStPos;
```

```
}
};
class COpcode : public CObject
{//此类保存文件keyword.txt中的关键字信息
public:
 int GetOpcodeNum(CString sOpcode);
 BOOL IsIn(CString str);
 void Display();
 BOOL Init();
 void SetValue();
 COpcode();
 virtual ~COpcode();
 Opcode m_Data[CNTS_KEYWORDS];//to save the messages of 44 keywords
private:
 BOOL m_bIsFileOpend;
 CStdioFile m_sfile;
};
```

44个操作码的存储表的代码实现如下：

```
COpcode::COpcode()
{
 //0--9
 m_Data[0].SetValue("ACALL",1,1,1);
 m_Data[1].SetValue("ADD",2,4,2);
 m_Data[2].SetValue("ADDC",3,4,6 );
                                    …
                                    …
 m_Data[42].SetValue("XCHD",43,1,105 );
 m_Data[43].SetValue("XRL",44,6,106 );
```

```
}
```

通过扫描输入的代码，根据上面描述的方法在111条指令表中查找sLab，从bBegin开始在Total范围内找，找到返回它的序号，找不到返回0。其代码实现如下：

```
int SearchLab(CString sLab,int nBegin,int nTotal)
{
    /*****************************************
    功能：在表中找sLab，从bBegin开始在Total范围内找，找到返回它的序号，找
不到返回0
    入口参数：sLab：COpcodeOp::m_sLab，查找对象
              nBegin:查找开始位置，
              nTotal：查找的范围
    *****************************************/
    int nEnd=nTotal+nBegin-1;
    for(int i=nBegin-1;i<nEnd;i++)
    {
        if(m_Data[i].GetLab()==sLab)
            return m_Data[i].GetNum();
        if(m_Data[i].GetFlag()==1)
        {
            if(m_Data[i].GetLab2()==sLab)
                return m_Data[i].GetNum();
        }
    }
    return 0;
}
};
```

8条伪指令用数组m_sNRIns[n]来存储，如下所示：

```
CNRIns::CNRIns()
```

```
{
m_sNRIns[0]="ORG";
m_sNRIns[1]="END";
m_sNRIns[2]="EQU";
m_sNRIns[3]="DATA";
m_sNRIns[4]="DB";
m_sNRIns[5]="DW";
m_sNRIns[6]="DS";
m_sNRIns[7]="BIT";
m_sNRIns[8]="";//保留
}
CNRIns::~CNRIns()
{
}
BOOL CNRIns::IsIn(CString sNRIns)
{
for(int i=0;i<CNTS_NRINS;i++)
 if(m_sNRIns[i]= =sNRIns)
 return TRUE;
return FALSE;
}
```

（3）词法、语法分析模块设计

MCS-51单片机汇编语言程序是由若干条MCS-51指令行组成的，MCS-51指令行的格式为：

[标号：] MCS-51操作码 [操作数1]，[操作数2]，[操作数3] [；注释]

正确的汇编伪指令语句格式如下：

[标号：或字符串] MCS-51伪代码 表达式列表 [；注释]

①标号位于一条语句的开头，是可选项。它可用来表示程序的转移地址，同时也方便程序的调试。MCS-51的标号是由大写英文字母开头的字母和数字组成，长度为1至8个字符。在标号中的字符超过8个字符时，编译程序会自动舍去超过部分的字符。

②MCS-51操作码是任一条语句不可缺少的必选项，编译程序根据这一字段生成目标代码。操作码可以是指令的保留字，也可以是伪指令和宏指令的助记符，用于指示计算机进行何种操作。

③操作数1至3是可选项，它的作用依赖于不同的MCS-51 指令助记符。操作数可以是数字、工作寄存器、特殊功能寄存器、标号地址和$符。数字可以使用十进制、十六进制、八进制或二进制数。其中十进制数以字符"D"为后缀，十六进制数以字符"H"为后缀，八进制数以字符"O"为后缀，二进制数以字符"B"为后缀，省略后缀时默认为十进制数。立即数的前面需要冠以符号"#"[26]。

④注释用于注解指令或程序的含义，易于编写和阅读程序，是可选部分。

定义一个CAnalyse类来进行词法、语法分析。首先打开*.asm源文件，并从源文件的第一行开始扫描，先通过判断读入一行代码，创建一个SLine,让当前行指针指向它，定义GetWordType(CString &sWord)方法来识别和获取词语类型，并在yx.cpp中定义的两个变量"ne"、"nb"分别代表读入单词的头指针和尾指针，开始时将"nb"的值赋值给"ne"，然后通过循环逐个判断，得出"ne"对应的真正位置，从而判定该词语。然后通过返回值判断读入的词语(sWord)的类型（标号，字符名称，指令，伪指令）。文件retvalue.h详细描述代表上述类型的整数返回值。这个过程中如果词语有错,则返回错误代码，否则判断该行是否空行；如果是空行，则读下一行；最后判断文件是否遇到END，如果没遇上，则重复以上动作读入下一行代码。

字符名称由大写英文字母开头的字母和数字串组成，长度为1-16个字符，如果超过则截取为16个；可以用下划线"_"。方法IsIdentifier(CString &str)实现对字符名称的识别。标号和字符名称一样由大写英文字母开头的字母和数字串组成，长度为1-16个字符，如果超过则截取为16个；可以用下划线"_"。方法IsLabel(CString &str)实现对标号的识别。先判断str是否为标号，同时还要去掉标号最后的冒号(:)。方法NRIns_Proc1(CString &sLine)实现对伪指令码的处理。其入口参数是"sLine"，表示需要处理的一行语句，其返回值为"SUCCESS"时代表正确;否则返回错误代码。另外，需要有一个方法NRIns_Proc2(CString &sLine,CString sNRIns)，专门处理伪指令码

db,dw,ds等，定义了方法Opcode_Proc(CString &sLine,CString sOpcode) 处理111条指令中的操作码，通过与指令表中111条指令的匹配来判断和识别操作码。语法分析的任务是判断每一个Line中是否符合语法规则，以及汇编指令和汇编伪代码后面所带的参数的个数是否合法。如果有语法错误，报告错误并退出。

部分核心代码如下：

```
int CAnalyse::GetWordType(CString &sWord)
{
  if(g_Opcode.IsIn(sWord))
        return IS_OPCODE;
    if(g_NRIns.IsIn(sWord))
        return IS_NRINS;
    if(IsLabel(sWord))
        return IS_LABEL;
    if(IsIdentifier(sWord))
        return IS_IDENTIFIER;
    return IS_NO_WORD;
}
int CAnalyse::Identi_Proc(CString &sLine)
{
    CString sWork,sIden;
    sIden=g_InsLine.GetPart1();
    //向符号表中加入该字符名称
    while(nb<sLine.GetLength() && (sLine[nb]==' ' || sLine[nb]==9))//sLine[nb]==9是TAB键
        nb++;
    ne=nb;                  //开始读入这个词
    while(ne<sLine.GetLength() && sLine[ne]!=' ' && sLine[ne]!=9 && sLine[ne]!=';')
    {   ne++;               //继续读入词语
    }
```

```
        sWork=sLine.Mid(nb,ne-nb);
        nb=ne;                    //另一个词开始
        g_InsLine.SetPart2(sWork);
        int nWordType=GetWordType(sWork);
        if(IS_NRINS!=nWordType)
        {
            return ERROR_LANGUAGE_WRONG;
        }
        else
        {
            int ret=NRIns_Proc1(sLine);
            if(SUCCESS!=ret)
                return ret;
        }
        return SUCCESS;
    }
```

通过对 Token 流的两次扫描，汇编伪指令 Token 序列和汇编指令 Token 序列各自进行解释。并把所有的ORG段定义、变量声明、符号常量、局部变量和标号定义等登记在符号表格中。

（4）符号表管理模块设计

对应符号表管理模块，定义了CSymbolTable 类。这是一个专门操控标号、变量、常量的管理器和计算器，所有的ORG段定义、变量声明、符号常量、局部变量和标号定义等都要登记在符号表格中。

符号表一个链表中的每一个元素都是一个结构体，每个结构体都包含很多域[27]。如果遇到一个词语，一旦判断出它是一个变量，就要描述出它的名字、类型、值等，然后都存入如符号表。对于标号，需要存储它所对应的地址，如表3-2所示。伪指令存储为一个值。

表3-2 标号述表

序号	标号名	对应地址
1	START	0000H
2	CINT0	0003H
3	CINT1	0013H
4	LOOP	00F9H
5	DALAY	015EH
…	…	…

符号表模块的部分核心代码：

```
class CSymbolTable : public CObject
{
public:
    void AddDefLine(int nLine);
    int AddUseLine(int nLine);
    CSymbolTable();
    virtual ~CSymbolTable();
    CArray <int,int> m_arrUseLine;  //使用该符号的行数
    int m_nDefLine;                 //声明该符号的行数
    CString m_sName;/               //标号，字符名称，操作码，伪指令的名字
    INT m_nType;                    //词语的类型：标号，字符名称
    BOOL m_blsDefined;              //该词是否已定义
    DHValue m_dhBeginAddr;          //该词，往往是标号，的开始地址
    DHValue m_dhEndAddr;            //该词的目的地址
    INT m_nValueType;               //值的类型：数据或地址,寄存器
    BOOL m_bUsed;          //该标号是否使用过（出现过），在DATA伪指令中会使用
    DHValue m_dhValue;              //该词代表的值
    DHValue *m_pdhValue;            //当该词代表的值是数表时，该指针指向数组
```

};

（5）出错处理模块设计

对应出错处理模块，定义了CErrorOp类和ErrorData类来处理。

发现的错误必须由用户对源程序进行改正后再重新编译，而警告则是可以忽略或编译器已经帮助用户改正了其错误。错误信息和警告信息包括：出错的汇编源文件的文件名，出错行的行号，具体的出错信息等。

常出现的错误信息和警告信息格式如表3-3和3-4：

表3-3 错误信息描述表

错误类型	详细信息
ERROR_LANGUAGE_WRONG	SYNTAX ERROR
ERROR_NO_SYMBOL	the symbol not in symbol table
ERROR_DATA_OVER	ERROR DATA OVER
NRINS_NOT_NRINS	the keyword is not pseudoinstruction
ERROR_DATA_ERROR	illegal character in numeric constant
SYM_SYMBOL_REDEFINE	attemp to define an already defined symbol
ERROR_LABEL_UNDEFINE	UNDEFINED SYMBOL
ERROR_MISSEND	miss END statement
ERROR_SYSTEM	SYSTEM ERROR
ERROR_OPENC51FILE_FAIL	open file *.asm failure
ERROR_IDENTIFIER_UNDEFINE	symbol undefined

表3-4 警告信息描述表

警告类型	详细信息
FileNeedEndWarn	File need 'END' to finish
DataBeExtentWarn	Data have been extended to 16-bit
DataBeTrun8Warn	Data have been truncated to 8-bit
DataBeTrun16Warn	Data have been truncated to 16-bit
DataBeTrun11Warn	Data have been truncated to 11-bit

核心代码如下：

```
int CErrorOp::AddError(int nLine,int err,CString sSource,CString sMsg)
{   /* 功能：增加错误
```

```
        入口参数：nLine:出错的行数，如果为-1：为编译系统错误。
        返回值：返回错误在链表中的position  */
    m_bErrOccur=TRUE;
    ErrorData *p=new ErrorData;
    p->SetValue(nLine,m_nSerial,err,sSource,sMsg);
    p->CreateErrorMsg();
    g_OutErrMsgToOutBar(p->m_sErrMsg);
    AddSerial();
    POSITION pos=pError.AddHead(p);
    return (int)pos;
}
void CErrorOp::AddSerial()
{
    m_nSerial++;
}
void CErrorOp::Delete()
{
    pError.RemoveAll();
}
void CErrorOp::Init()
{
    m_nSerial=1;
    m_bErrOccur=FALSE;
    Delete();
}
```

处理操作数错误的具体代码如下：

```
class ErrorData :public CObject
{
```

```
public:
    int m_nNum;
    int m_nLine;
    int m_nErrorNum;
    CString m_sMsg;//一些错误的提示信息
    CString m_sErrMsg;//最终形成的错误信息行
    CString m_sSource;
    ErrorData()
    {
        m_nNum=0;
        m_nLine=0;
        m_nErrorNum=0;
    }
    void Init()
    {
        m_nNum=0;
        m_nLine=0;
        m_nErrorNum=0;
        m_sMsg="";
        m_sSource="";
    }
```

（6）HEX 文件的操作模块和目标代码生成模块设计

由于一般编译软件产生的用于写入芯片的文件都是HEX格式的文件。本文第2章已经简单介绍了HEX文件格式。HEX文件属于文本文件，可以方便地用记事本查看。

HEX目标代码文件是由一条一条不同的记录联合而成的，每条记录都有各自的记录起始标志、记录长度、数据装入的地址、记录类型数据值和记录校验和。而记录内容则根据记录类型的不同而相异甚远。记录本身构成一个实体，用CHexRrecord类来

表示。记录的格式如下表3-5所示：

表3-5 目标代码记录描述表

标志	记录长度	首地址	记录类型	数据值	校验和
:	10	0000	00	75 81 60 78 0A D2 92 74 08 11 25 C2 92 74 08 11	21
:	10	0010	00	25 D8 F2 78 05 D2 92 74 3C 11 25 C2 92 74 3C 11	15
:	10	0020	00	25 D8 F2 80 DE F5 30 7F 64 7E 7D DE FE DF FA D5	F6
:	03	0030	00	30 F5 22	86
:	00	0000	01		FF

CHexRrecord类的声明和定义如下：

```
class CHexRecord
{
public:
    void CreateHexLastRecord();
    void Init();
    void CreateHexRecord(DWORD dwStAddr);
    CString m_sHexLine;
    CString m_sData;
    UCHAR m_bParity;
    UCHAR m_bByteSize;
    UCHAR m_bFlag;
    CString m_sFlag;
    DHValue m_dhStAddr;
    CHexRecord()
    {
        m_bParity=0;
        m_bByteSize=0;
        m_bFlag=0;
    }
};
```

在ChexRecord类中定义一个方法CreateHexRecord(DWORD dwStAddr)，它主要是用来生成一条Hex格式的记录。代码如下：

```
void CHexRecord::CreateHexRecord(DWORD dwStAddr)
{
    UCHAR cc=0,ct=0;
    CString sByteSize,sWork,sCheck;
    m_bByteSize=m_sData.GetLength()/2;
    g_nConvertDecToHex(m_bByteSize,sByteSize);
    m_dhStAddr.EvalueSelf(dwStAddr);
    m_dhStAddr.Make16Bits();
    m_sHexLine=sByteSize+m_dhStAddr.sHex+"00"+m_sData;
    for(int i=0;i<m_sHexLine.GetLength()/2;i++)
    {
        sWork=m_sHexLine.Mid(2*i,2);
        ct=(BYTE)g_dwConvertStrToDWORD(sWork,DATA_H);
        cc+=ct;
    }
    cc=cc%0x100;
    cc=(UCHAR)~cc;
    cc++;
    sCheck.Format("%X",cc);
    while(sCheck.GetLength()<2)
    sCheck.Insert(0,'0');
    m_sHexLine.Insert(0,':');
    m_sHexLine+=sCheck+"\n";
}
```

对应目标代码生成模块，在Cseccomp类中定义了方法CreateHexFile()，用来将一条条记录整合起来形成Hex文件。

```
int CSeccomp::CreateHexFile()
```

```
{ //功能：形成HEX文件。
    DWORD dwStAddr=0;
    int nTotal=0,nNeed=16;
    int i=0;     //定位数组元素
    CString sWork;
    int nArraySize=m_LineArray.GetSize();
    while(i<nArraySize)
    {
        if((dwStAddr+nTotal)==m_LineArray[i].m_dhAddr.dwDec)
        {
            if(m_LineArray[i].m_nDataNum<=nNeed)
            {
                m_hrRecord.m_sData+=m_LineArray[i].m_sData;
                nTotal+=m_LineArray[i].m_nDataNum;
                nNeed=16-nTotal;
                i++;
            }
            else
            {
                sWork=m_LineArray[i].m_sData.Left(2*nNeed);
                m_hrRecord.m_sData+=sWork;
                m_hrRecord.CreateHexRecord(dwStAddr);
                AddHex();
                m_hrRecord.Init();
                dwStAddr+=16;
                nNeed=16;
                nTotal=0;
    sWork=m_LineArray[i].m_sData.Right(m_LineArray[i].m_nSizeData-sWork.GetLength());
```

```
                m_hrRecord.m_sData+=sWork;
                nTotal=sWork.GetLength()/2;
                nNeed=16-nTotal;
                i++;
            }
        }
        else
        {
            m_hrRecord.CreateHexRecord(dwStAddr);
            AddHex();
            m_hrRecord.Init();
            dwStAddr=m_LineArray[i].m_dhAddr.dwDec;
            nTotal=nNeed=0;
            m_hrRecord.m_sData+=m_LineArray[i].m_sData;
            nTotal+=m_LineArray[i].m_nDataNum;
            nNeed=16-nTotal;
            i++;
        }
    }
    m_hrRecord.CreateHexRecord(dwStAddr);
    AddHex();
    m_hrRecord.CreateHexLastRecord();
    AddHex();
    return 0;
}
```

（7）LST 列表文件的操作模块和产生列表文件模块设计

列表文件是纯文本文件，为用户提供源码与目标码的对照。其格式如下：

表3-6 列表文件描述表

地址	目标代码	行号	源代码
0003	7588D2	1	MOV TCON, #0D2H
…	…	…	…
0018	E582	19	MORE: MOV A, DPL
001A	04	20	INC A
001B	F582	21	MOV DPL, A
001D	50EF	22	JNC LOOP
…	…	…	…

对应产生列表文件模块，在类CFileProcess中定义了一个方法CreateListHexFile（），其功能是生成列表文件。

```
int CFileProcess::CreateListHexFile(CString sFileContent)
{/* 功能：在c51文件目录下生成list,hex文件
    返回值：成功返回 1，否则为 0；
  */
    g_OutMsgToOutBar("\r\n   creating file *.lst *.hex...");
    CString sList1Name=m_sC51Path+".LST1";
    CString sListName=m_sC51Path+".LST";
    CString sHexName=m_sC51Path+".HEX";
    int ret=m_sfList1.Open(sList1Name,CFile::modeCreate | CFile::modeWrite);
    if(0==ret)
    {
        g_ErrorOp.AddError(-1,ERROR_SYSTEM,"","open list1 file failure ");
        g_OutMsgToOutBar("\r\n   creating file *.lst *.hex failure");
        return 0;
    }
```

```
m_sfList1.WriteString(sFileContent);
m_sfList1.Close();
CSeccomp sc;
ret=sc.Process(sList1Name);
if(ret!=SECCOMP_SUC)
{
    g_ErrorOp.AddError(-1,ERROR_SYSTEM,"","second complier wrong");
    g_OutMsgToOutBar("\r\n   creating file *.lst *.hex failure \r\n");
    return 0;
}
ret=m_sfList.Open(sListName,CFile::modeCreate | CFile::modeWrite);
if(0==ret)
{
    g_ErrorOp.AddError(-1,ERROR_SYSTEM,"","open file *.lst failure ");
    g_OutMsgToOutBar("\r\n   creating file *.lst *.hex failure");
    return 0;
}
m_sfList.WriteString(sc.m_sList);
m_sfList.Close();
ret=m_sfHex.Open(sHexName,CFile::modeCreate | CFile::modeWrite);
if(0==ret)
{
    g_ErrorOp.AddError(-1,ERROR_SYSTEM,"","open file *.hex failure ");
```

```
            g_OutMsgToOutBar("\r\n   creating file *.lst *.hex failure");
            return 0;
        }
        m_sfHex.WriteString(sc.m_sHexFile);
        //m_sfList1.Close();
        //m_sfList.Close();
        m_sfHex.Close();
        g_OutMsgToOutBar("   create file *.lst *.hex success\r\n");
        return 1;
    }
```

3.4 串口通信部分

串行通信作为计算机之间常用的通信方法之一，在工业控制领域得到了广泛的应用。本文利用 Windows 提供的一系列标准 API 函数，通过具体的程序实现了将计算机上编译好的二进制文件采用串行通信方式传给单片机系统执行，主要采用了基于 Windows API 的串行通信编程技术及 MFC 文件操作。利用 Windows API 函数进行串口通信基本模式是打开串口，配置串口，对串口读写，关闭串口；文件操作与对串口操作比较相似[28]。事实上，在 VC 中就是把串口当作一种特殊文件进行操作的[29]。

3.4.1 串口通信的基本概念

（1）串行通信系统的基本模式

当需要在两台计算机之间，或者在计算机和其他智能设备之间通过串口进行通信时，这两台计算机或其智能设备就构成了串行通信的基本系统[30]。不管进行通信的双方是两台计算机还是计算机和调制解调器，串行通信的基本模式总可以抽象为由 DTE(Data Terminal Equipment)和 DCE(Data Communication Equipment)组成的通信系统，如图 3-5 所示：

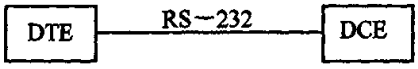

图 3-5 串行通信的基本模式

图中，DTE 叫数据终端设备，DCE 叫数据通信设备。当两台计算机之间的空间距离较近时，可以按直接用电缆将两台计算机连接起来。当通信的双方空间距离较远时，就需要在通信链路上增加功率驱动设备，如调制解调器、数据链路等。这时 DTE 为计算机，DCE 就是调制解调器或者是数据链路设备[31]。这种通讯方式可以用图 3-6 来表示。

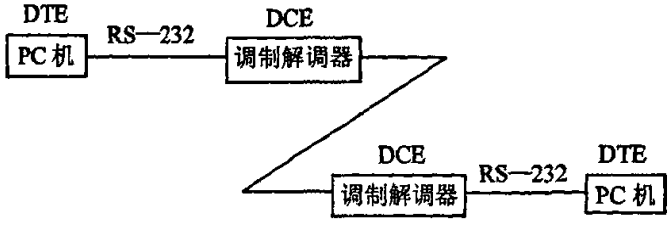

图 3-6 带调制解调器的串行通信应用

（2）RS-232 及其信号功能说明

连接 DTE 和 DCE 的串行电缆提供数据信号和控制信号的链路。数据信号包括数据发送 TD 和数据接受 RD 两条链路，控制信号包括信号的公共地、RTS、CTS、DSR、DTR 以及数据载波检测 DCD 等。为了规范串行通信接口，电子工业协会推荐了一个串行通信的工业标准 EIA RS—232。表 3-7 给出了 RS—232 标准 D 型 25 针和 9 针引脚的定义[32]。

表 3-7 串行接口引脚 RS—232 标准

25 针引脚	9 针引脚	信息方向	助记符	说明
1	5	n/a	n/a	保护地
2	3	to DCE	TD	发送数据
3	2	to DTE	RD	接收数据
4	7	to DCE	RTS	请求发送
5	8	to DTE	CTS	清除发送
6	6	to DTE	DSR	数据设备就绪
7	9	*	*	信号地
8	1	to DTE	DCD	数据载波检测
20	4	to DCE	DTR	数据终端就绪

表中，TD 和 RD 是通信双方进行数据交互的物理链路。TD 是 DTE 发送数据给 DCE 的物理链路，RD 是 DTE 接受 DCE 数据的物理链路，信号地是整个线路的公共回路。

（3）典型的串口通信连接

一个完整的 DTE—DCE 连接如图 3-7。

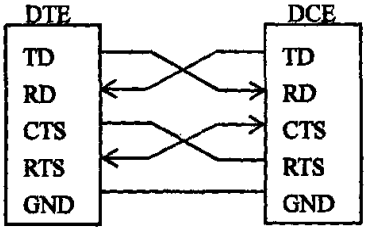

图 3-7 典型的带握手控制的通信连接

DTE 的 TD 连接到 DCE 的 RD，DTE 的 RD 连接到 DCE 的 TD，DTE 的 CTS 连接到 DCE 的 RTS，DTE 的 RTS 连接到 DCE 的 CTS，DTE 的 DSR 连接到 DCE 的 DTR，DTE 的 DTR 连接到 DCE 的 DSR。事实上，用 TD、RD 和公共地三根线就可以通信了。

3.4.2 Windows 提供的串行通信机制

在 Windows 环境下，程序员不必了解硬件实现的细节，只需知道与串行通信相关的通行控制，就可以实现计算机之间的串行通信[33]。

（1）数据的接受与发送由 Windows 管理

Windows 在其系统数据区为每一个通信端口开辟了数据输入输出缓冲区，也叫数据队列。当应用程序发送数据时，就将要发送的数据放入输出队列。Windows 后台发送中断服务程序会按照应用程序的设置将数据发送到断口。当端口上有数据到来时，Windows 后台中断服务程序会将数据接收好，并按到来的先后顺序放入接收队列。应用程序可以查询接收队列，也可以设置 Windows，当接收的字节数到达设定值时，就会调用应用程序及时从输入队列中读取数据[34]。

（2）避免应用程序对 CPU 的独占

串行通信的数据传输速度相对于计算机来说是比较慢的，如果应用程序简单地用循环查询方式，就会导致其独占系统运行时间，违反了 Windows 基本编程规则。Windows 的编程机制对串行通信程序的编写作出了相应的规定，以解决这个问题[35]。

3.4.3 Windows 串行通信 API 函数

（1）打开串口

在使用串口之前，应先用函数 CreateFile( )将其打开，返回一个标识该串口的句柄。函数原型如下：

```
HANDLE CreateFile(
LPCTSTR lpFileName,              // pointer to name of the file
DWORD dwDesiredAccess,           // access (read-write) mode
DWORD dwShareMode,               // share mode
LPSECURITY_ATTRIBUTES lpSecurityAttributes,
//pointer to security  attributes
DWORD dwCreationDisposition,     // how to create
DWORD dwFlagsAndAttributes,      // file attributes
HANDLE hTemplateFile // handle to file with attributes to copy
);
```

参数说明如下：

lpFileName 指向一字符串，指定了要打开的串口逻辑名，如"COM1"，"COM2"等，常用的 PC 机有两个串行口，尽量使用"COM2"，因为"COM1"响应更多的系统中断；

dwDesiredAccess 指定对串口的访问模式。大部分通信是双向的，而通常设置为：GENERIC_READ|GENERIC_WRITE，符号"|"用于对操作进行组合；

dwShareMode 该参数指定串口的共享模式，该参数是为文件共享提供的，串行口不能作为共享设备，只能独占访问，故数值必须置为 0，这是文件与通信设备之

间的主要差别；

lpSecurityAttributes 该参数指向一个 SECURITY_ATTRIBUTES 结构，该结构指定串口的安全描述，设为 NULL；

dwCreationDisposition 该参数指定文件创建方式，对于串口只能为 OPEN_EXISTING，表示改串口必须存在，因为 CreateFile( )只能打开存在的端口，而不能像创建新文件一样创建物理上不存在的新端口；

dwFlagsAndAttributes 该参数制定了文件属性和标志，对于文件来说，具有多种属性（只读、隐藏、系统）是可能的，但对于串行口，它只能是 0（同步传输）或者 FILE_FLAG_OVERLAPPED（异步传输）；

hTemplateFile 该参数对于串口只能设置成 NULL；

返 回 值： 成功时返回创建的句柄；否则返回 INVALID_HANDLE_VALUE；

举例：

```
HANDLE hComm;//定义句柄变量
hComm = CreateFile( "COM1", GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, |FILE_FLAG_OVERLAPPED, NULL );
if (hComm=INVALID_HANDLE_VALUE)
{
.....//打开串口的出错处理
}
```

（2）配置串口（串口的初始化）

串口打开成功后，接下来可以配置串口通信参数如波特率、数据位数、停止位、校验位等[36]。修改这些参数是要和设备控制快 DCB（Device Control Block)打交道，DCB 有近 30 个数据成员，是一个很复杂的数据结构，全部弄清楚它们的含义相当费时。而对于采用 3 线方式的串行通信来说，DCB 结构中的绝大多数参数可以不予考虑，因为只要设置好波特率、数据位、停止位、校验位等几个关键参数就行。初始化工作还包括，包括设置输出/接收缓冲区大小、超时控制等。

设置时用到的函数为：

```
BOOL GetCommState(
HANDLE hFile,     // handle to communications device
LPDCB lpDCB       // pointer to device-control block structure
);
```

取出系统当前设备控制快信息，根据通信需要修改 DCB 的相关数据成员的值；

- 函数

```
BOOL SetCommState(
HANDLE hFile,     // handle to communications device
LPDCB lpDCB       // pointer to device-control block structure
);
```

实现用修改过的 DCB 重新设置串口，SetCommState()函数重新初始化所有硬件和控制设置，但不清空 I/O 缓冲区[37]；

- 函数

```
BOOL PurgeComm(
HANDLE hFile,     // handle to communications resource
DWORD dwFlags     // action to perform
);
```

清空串口指定的输入输出缓冲区的内容；

- 函数

```
BOOL SetupComm(
HANDLE hFile,          // handle to communications device
DWORD dwInQueue,       // size of input buffer
DWORD dwOutQueue       // size of output buffer
);
```

设置输入缓冲区和输入缓冲区的大小；

- 函数

```
BOOL SetCommTimeouts(
HANDLE hFile,                    // handle to comm device
```

```
LPCOMMTIMEOUTS lpCommTimeouts //pointer to commtimesout  structure
);
```

设置读写操作超时，如果不设置该参数，通信端口将使用由驱动程序提供的缺省值或以前通信程序设置的值，一个不合适的超时设置可能造成读、写操作过于频繁或永不结束[38]。设置如下：

COMMTIMESOUT

```
typedef struct _COMMTIMEOUTS {
DWORD ReadIntervalTimeout;
DWORD ReadTotalTimeoutMultiplier;
DWORD ReadTotalTimeoutConstant;
DWORD WriteTotalTimeoutMultiplier;
DWORD WriteTotalTimeoutConstant;
} COMMTIMEOUTS，*LPCOMMTIMEOUTS;
```

若读操作总超时乘数 ReadTotalTimeoutMultiplier 和读操作总超时常数 ReadTotalTimeoutConstant 都为 0，表明读操作与总超时无关;如果读操作间隔超时 ReadIntervalTimeout 和读操作总超时乘数设为 MAXWORD，而读操作总超时常数的值小于 MAXWORD 且大于 0，则调用 ReadFile()函数时，如果输入缓冲区有数据则立即返回，否则等待只有收到数据后才返回，如果在 ReadTotalTimeoutConstant 参数指定的时间内还没有收到数据，则 ReadFile()函数超时返回。在多任务程序设计中通常将 ReadIntervalTimeout 设置成 MAXWORD，而将 ReadTotalTimeoutMultiplier 和 ReadTotalTimeoutConstant 设置为 0，这样读操作可以立即返回。

- 函数

```
BOOL GetCommTimeouts(
HANDLE hFile,                    // handle to comm device
LPCOMMTIMEOUTS lpCommTimeouts   // pointer to comm time-outs structure
);
```

获得当前超时设置；

通过下面的程序来说明串行通信参数的设置方法。

```
HANDLE  hCom;
```

```
DWORD  dwError;
DCB dcb;
COMMTIMESOUT timesout;
hCom=CreateFile("COM2"，//串口名
GENERIC_READ|GENERIC_WRITE，//允许读和写
0，  //独占方式
NULL，
OPEN_EXISTING，////打开而非创建
FILE_FLAG_OVERLAPPED，//重叠方式
NULL );
if (hCom=INVALID_HANDLE_VALUE)
{
DwError=GetLastError();
.....//处理错误
}
SetupComm(hCom，2048，2048)//输入输出缓冲区的大小均为 2048 字节
GetCommTimesOut(hCom，&timesout);//获得当前超时设置
CommTimeOuts. ReadIntervalTimeout=0×FFFFFFFF;
CommTimeOuts. ReadTotalTimeoutMultiplier = 0;
CommTimeOuts. ReadTotalTimeoutConstant = 0;
CommTimeOuts. WriteTotalTimeoutMultiplier = 0;
CommTimeOuts. WriteTotalTimeoutConstant = 5000;
SetCommTimeouts(m_hIDComDev，  &CommTimeOuts);//设置超时
GetCommState(hCom，&dcb);
dcb.BaudRate =9600; //设置波特率 9600
dcb.ByteSize =8;//8 位数据位
dcb.Parity = NOPARITY;//无奇偶校验
```

```
dcb.StopBits = ONESTOPBIT ;//1 为停止位
dcb.fBinary = TRUE ;//二进制传输
SetCommState(hCom,  &dcb ) ; //串口参数配置
```

（3）读串口状态

- 函数

```
BOOL ClearCommError(
HANDLE hFile,        // handle to communications device
LPDWORD lpErrors,  // pointer to variable to receive error codes
LPCOMSTAT lpStat   // pointer to buffer for communications status
);
```

用来获得串口状态，LPCOMSTAT 结构中的 cbInQue 和 cbOutQue 分别用来记录串口的输入和输出缓冲区中的字节数，在查询方式读取串口中经常用到这个函数[39]。

（4）读写串口

串口打开后可以对串口进行读写操作

- 读串行口的函数原型：

```
BOOL ReadFile(
HANDLE hFile,      // handle of file to read
LPVOID lpBuffer,    // pointer to buffer that receives data
DWORD nNumberOfBytesToRead,    // number of bytes to read
LPDWORD lpNumberOfBytesRead,  // pointer to number of bytes read
LPOVERLAPPED lpOverlapped      // pointer to structure for data
);
```

参数说明如下：

hFile　　该参数是由 CreateFile( )返回的柄；

lPBuffer　　该参数是读取的数据缓冲区指针；

nNumberOfBytesToRead 该参数是要读取的字节数；

lpNumberOfBytesRead　该参数为指针，指向代表实际读取的字节数的变量；

LpOverlapped　该参数是指向一个可重叠 I/O（异步通信）的数据结指针构。如果设为 NULL，则 ReadFile( )工作在同步；如果指向一个重叠结构，则工作在异步方式；

- 写串行口的函数原型如下：

```
BOOL WriteFile(
HANDLE hFile,                    // handle to file to write to
LPCVOID lpBuffer,                // pointer to data to write to file
DWORD nNumberOfBytesToWrite,     // number of bytes to write
LPDWORD lpNumberOfBytesWritten,  // pointer to number of bytes written
LPOVERLAPPED lpOverlapped        // pointer to structure for overlapped I/O
);
```

参数说明可参考 ReadFile( )中的参数说明。

（5）关闭串口

串行口是非共享资源，所以打开串口后，一定要关闭串口。关闭串口函数较简单。函数原型为：BOOL CloseHandle(HANDLE hObject);其中 hObject 参数为 File( )返回的端口句柄。返回值非 0，则调用成功。

3.4.4 串口通信的设计实现

本文使用 VC++ 6.0 来实现计算机和单片机之间的通信。先从 PC 机上读取已被编译过的目标代码文件（.HEX 格式）进行数据处理，然后通过串口和单片机进行通信。实现串行通信的方法有三种。方法一：使用 VC++提供的串行通信控件 MSComm；方法二：在单线程中实现自定义的串口通信类；方法三： 多线程下实现串行通信[40]。本设计中采用第二种方法来实现对串行通信的编程，因为利用 Windows API 提供的通信函数可以编写出可移植性更强的串行通信程序。在 VC++ 6.0 建立 MFC

AppWizard(exe)项目 comm，在生成类 CCommView 类编辑代码，主要有：

- 读取 HEX 文件一条记录，将该记录转化成记录类（CRecord 类）；
- 读取二进制文件并将文本数据类型转化成传输所要求的字节类型；
- 通过同步方式、采用查询方式读串口、依据传输协议（图 4.3.1）与串行口进行数据传输；
- 打开文件对话框显示，传输数据的无模式对话框显示，DCB 块设置对话框显示等；

程序基本轮廓如下图 3-8：

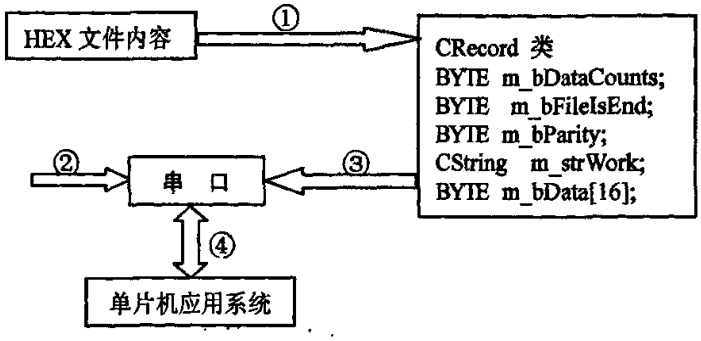

图 3-8 程序轮廓图

注释：

① 从 HEX 文件中读取一行记录，将该记录转换成记录（CRecord）类对象；

② 对串口的一些操作，包括打开串口、配置串口、关闭串口；

③ 按照传输协议（图 4.3.1）实现应用程序和串口的通信；

④ 串口和单片机之间通信。

（1）读取并处理文件记录

程序通过以下步骤读取并处理文件中的记录：

- 使用 CFileDialog 类生成打开文件对话框；
- 声明 CStdioFile 文件类对象，对选中的文件进行操作，当程序结束时，应调用 CStdioFile ::Close( )关闭文件；
- 使用 CStdioFile ::ReadString(CString& rString)读取 HEX 文件一条记录（即一行），将其放入 rString 中，再执行则读取下一条记录；当该函数返回值为 FALSE，

则文件结束；

- CStdioFile::Seek(LONG lOff， UINT nFrom)进行文件指针定位，参数 lOff 应设置成一条记录长度 rString.GetLength( )+2，可以灵活地读取文件中的任何记录；
- 对 rString 处理，提取 HEX 记录的相关信息，放入类 CRecord 的对象中。

（2）程序通过串口和单片机通信

本文利用 Windows 串行通信 API 函数，按通信协议采用同步、查询式读串口方式与单片机进行通信[41]。查询式读串口方式利用函数 ClearCommError()读取串口状态，向串口中写数据同样使用该函数读取串口状态，通过 comstat.cbInQue 和 comstat.cbOutQue(comstat 为 COMSTAT 结构)的值决定是否读写串口。程序的流程图如下所示：

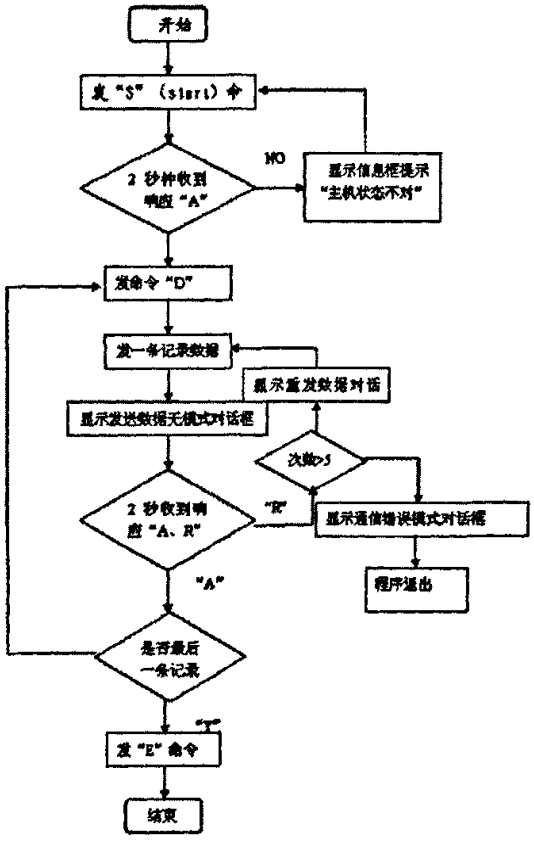

图 3-9 程序流程图

（3） 通信过程的核心代码

```
while(nRecordsTrased<m_nTotalRecords)
{if(!IsCommWriteBuffEmpty())//until outbuffer is empty，usually it is empty
        return FALSE;
    bResult=WriteFile(m_hComm，&bDataCommond，1，&dwWritten，NULL);
    if(!(bResult && dwWritten))
    {MessageBox("Write DataCommond D failure!");
        return FALSE;
    }
    nReTrasTimes=0;
    while(1)
    {TrasOneRecord();
        nRecordsTrased++;
        CTime time=CTime::GetCurrentTime();
        int nStartSeconds=time.GetSecond();
        while(TimeOuts(nStartSeconds)<=2)
        {//读取串口缓冲区状态
            bResult=ClearCommError(m_hComm，&dwError，&comstat);
            if(!bResult)
            {MessageBox("Failure ClearCommError");
                return FALSE;
            }
          if(comstat.cbInQue>0)
            {bResult=ReadFile(m_hComm，&bWaitCommond1，1，&dwRead，NULL);
                if(!bResult) //只有当串口缓冲区里存在数，即单片机响应程序 A 才执行
                {MessageBox("read commond A or R in the process of trasdata");
                    return FALSE;
                }
                if(bWaitCommond1=='A' || bWaitCommond1=='R')
```

```
                {break;
                }
            }
            bWaitCommond1='L';
        }
/*  临时改变 bWaitCommond1 的值，因为存在这样的情况：当第一次 bWaitCommond1 取得 A
    后，尽管单片机不再发 A 其值依然为 A，程序会误认为单片机的响应，在####处继续
    执行（而事实上，单片机有可能并没有响应)；改变后####处就会跳过；*/
            bResult=PurgeComm(m_hComm，PURGE_RXCLEAR);
            if(!bResult)
            {MessageBox("Failure PurgeComm");
            return FALSE;
            }
            if(bWaitCommond1=='A')              //#########//
            {break;
            }
            else
            {   nReTrasTimes++;
                nRecordsTrased--;
                m_file.Seek(-(m_rString.GetLength()+2)，CFile::current);
                if(nReTrasTimes<5)
                    MessageBox("重新发送数据!");
                else
                {MessageBox("通信错误! 将退出程序!");
                return FALSE;}
            }
    }
}
```

实现界面如下所示：

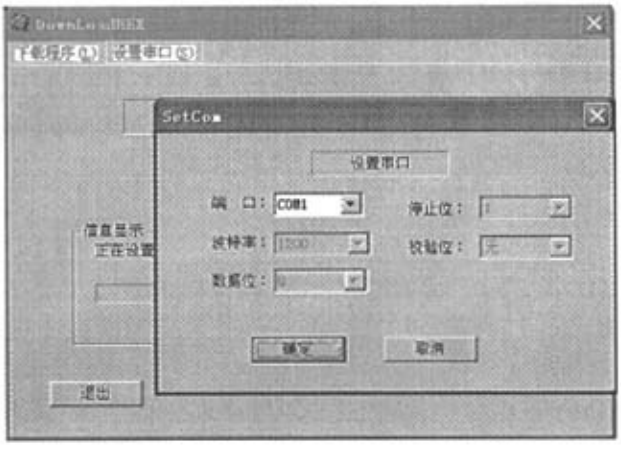

图 3-10 串口通信的实现界面

3.5 用户登录模块

用户通过本模块登录到系统，登录后本系统根据用户角色权限设置其可使用的功能及进入该系统。用户可在此模块中输入用户名和密码，点击登录按钮进入登录系统，倘若为合法用户名和密码，则成功登录系统。如下图所示：

图 3-11 登录界面

其中用户分为“user”和“administer”，其管理权限也分为用户权限和管理员权限。管理员登录后的界面要比普通用户登录后的界面多一个“用户密码管理”的菜单，可以生成、获得用户密码，也可以登录普通用户和更改管理员密码，如下图 3-12 所示：

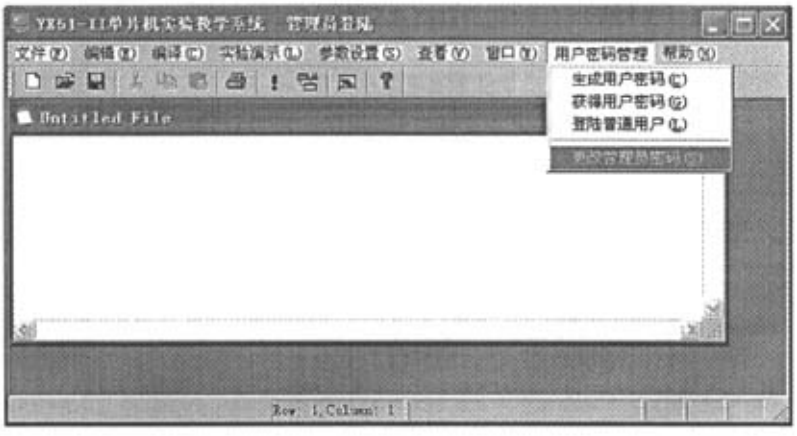

图 3-12 管理员登录后的界面

若用户登录名位非法用户或者用户密码错误，则系统提示“第 1 次输入密码错误，最大输入次数为 3 次”，用户可选择点"确定”重新登录，如下图 3-13 所示：

图 3-13 错误提示

若用户登录时点击“自动登录”，系统将自动记录下用户本次登录的帐户及密码，在下次登录自动登录。

用户设定的密码和管理员设定的密码都存放在 exp.sys 文件中，登录时只需判定用户输入的密码和 exp.sys 文件中存储的密码是否一致。此时要注意，由于密码文件和 24 个实验演示的程序代码都存放在 exp.sys 文件中，因此需要检查从 exp.sys 文件中读到的内容是否为密码区域的内容。因为密码文件以两个“F”开头，因此采用算法来判定头两个字符是否为“F”即可。

3.6 实验演示模块

将 24 个单片机实验的代码放在文件 exp.sys 中，每个实验代码分配 2K 的存储空间，因此总共需要 48K 的存储空间。在下拉菜单“硬件演示实验下载”中直接点击所选的实验，就可以轻松的地将实验程序代码直接下载到单片机上。如下图 3-14 所示：

图 3-14 24 个硬件实验演示

3.7 帮助文件模块

帮助文件主要包含三个方面，使用指南、操作手册和版本说明。

其中“操作手册”功能是系统通过调用“操作手册.chm”文件来实现的。CHM 是 Windows 下的标准帮助文件格式，也是网上比较流行的电子书格式。可以先做一个.html 格式的文件，然后通过免费制作工具 Microsoft HTML Help Workshop 可以十分方便地把 HTML 格式编译成 CHM 文件[42]。调用“操作手册.chm”文件的代码如下：

```
void CMainFrame::OnHelp2()
{ShellExecute(this->m_hWnd,"open","操作手册.chm",NULL,NULL,SW_SHOW ); }
```

使用方法十分简单，如下图 3-15 所示，在“帮助”下拉菜单中点“操作手册”。

图 3-15 帮助文件

弹出界面如图 3-16 所示：

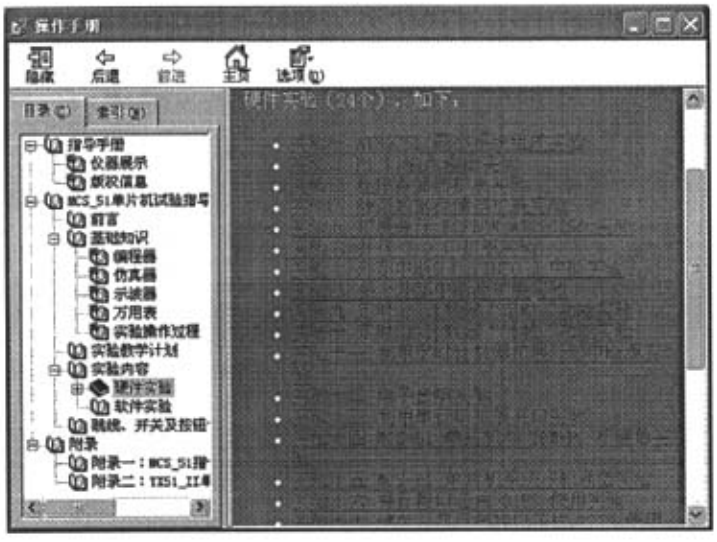

图 3-16 操作手册界面

操作手册中包含了 24 个硬件实验，查看和使用都非常简便快捷。例如，想查看电子音响实验的具体内容，则点击其中的“实验十二 电子音响实验”，即弹出其实验内容和具体说明，如图 3-17 所示：

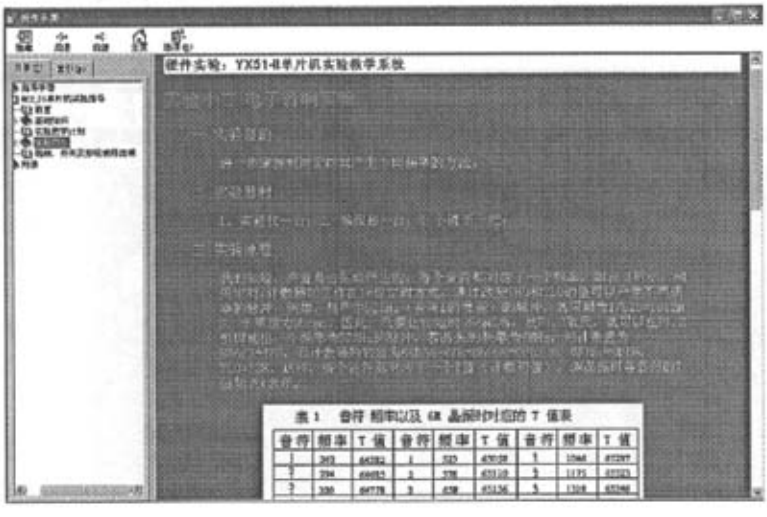

图 3-17 硬件实验十二的实验内容和说明

该实验的源程序也可以在“操作手册”中显示，界面如图 3-18 所示：

图 3-18 电子音响实验的源程序

由此可以看出，该操作手册详细介绍了与单片机实验相关的一些基础知识和实验内容。此外，该手册的附录一是 MCS_51 指令详解，附录二是 YX51_II 单片机实验教学系统元件装配图。图 3-19 给出了附录一的显示内容：

图 3-19 操作手册附录一的显示内容

第四章 该系统在高校单片机教学方面的应用

单片机因其具有成本低，体积小，功能强等优点，应用非常广泛，从航空航天、工业控制、通信工程到家用电器，无所不在。为了普及单片机技术的应用，各高等院校、专科学校甚至职业中专都相继开设了《单片机原理》课程，这是一门实践性很强的课程，学好的关键在于动手，只有亲自动手编程，做好实验才能真正提高水平。然而，本课程的实验教学比较混乱，实验内容也有些老化，一方面由于单片机技术的发展，有些内容已经过时而现在仍被作为教学内容，另一方面很多新的、非常适用的技术却被排斥在教学内容之外，也没有统一的实验教学仪器，因此，各个学校自行其事，特别是条件较差的学校甚至根本不开实验课，有些学校让学生在面板上插线进行实验，不仅浪费了大量的时间，损坏了很多元件，而且严重影响了教学效果。

为了解决上述问题，在导师的指导下研制了 YX51 单片机实验教学系统，给它定名为"YX51_II 单片机实验教学系统"。该系统的特点是立足于单片机技术的学习和实验教学，使初学者学习单片机更方便、容易。

其中的实验演示模块面向单片机技术的开发，共集成了 24 个精选的单片机硬件实验。其中，每个实验讨论一个专题。首先，讲清实验目的和实验原理，然后，突出编程和使用。全部实验大致可分为三类，第一类实验是为了满足教学要求的基础实验；第二类实验为单片机开发中非常有实用价值的技术实验；第三类实验是单片机技术领域新发展的技术实验。第二、三类实验旨在提高学生的单片机技术水平和开发能力。通过本系统也可以进行各种软件实验。本实验教学系统，软、硬件完成透明，学生可以在其上进行很多综合型、设计型实验，有利于培养学生的创新能力。开发者花了很多时间，查阅了大量的资料，为实验者提供了一套非常珍贵的、有保存价值的资料。本系统既可作为各高等院校单片机原理课程实验教学仪器的配套使用软件，也可以为自学单片机技术的各类人员提供最有效的帮助。无论是在校学生，还是工程技术人员以及单片机爱好者只要能踏踏实实地将本系统的所有实验作一遍，并掌握每个实验所讨论的技术，定能胜任单片机技术开发的各种工作。

该系统采用特殊的技术，在 MCS-51 单片机上实现了代码直接下载的功能，在计算机上调试好的程序代码既可以通过编程器写入单片机或 EPROM 中，也可以直接通过计算机的串口直接下载到应用系统中。系统运行时，程序可存放在单片机内部，也

可存放在单片机外部的 EPROM 或 Flash 存储器中，非常方便灵活。

其安装和使用方法非常简单，本软件是绿色软件，无需安装，无任何插件。只需要将“YX51-II_绿色版”文件夹复制到你所使用的电脑上，然后双击文件夹中的可执行文件“YX51.exe”，（如图 4-1 所示）就能轻松使用该软件。

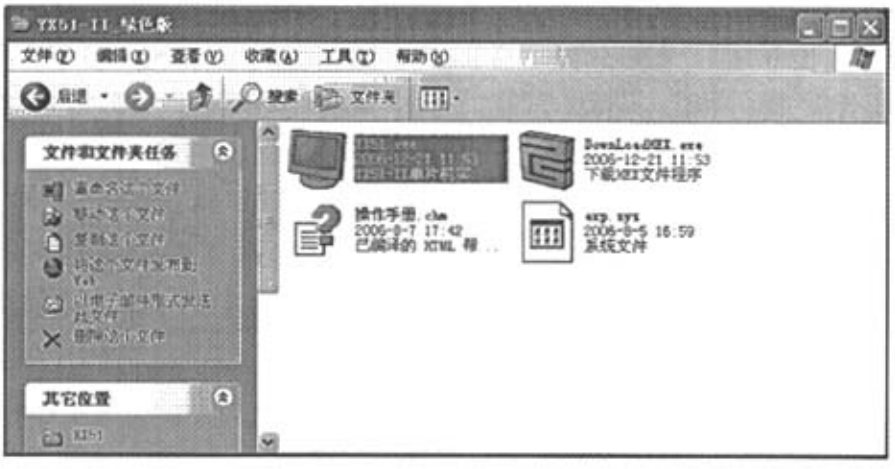

图 4-1 安装文件

该系统与硬件实验仪（如图 4-2 所示）相连时只需要用一根串口线，就能将在计算机上编译好的程序或所选的 24 个实验的程序通过串口直接下载到实验仪，完全省去了以往要使用编程器的繁琐步骤。

图 4-2（a）

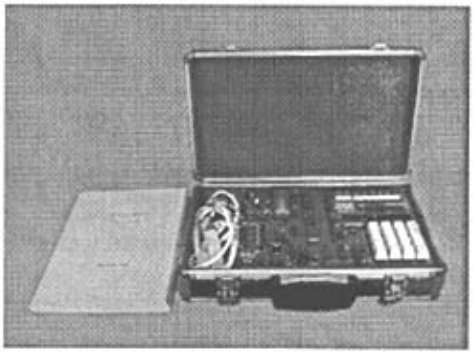

图 4-2（b）

第五章 结束语

5.1 总结

通过两年的努力，终于实现了 MCS-51 汇编语言程序的 IDE。它能支持 MCS-51 单片机的所有汇编指令及其寻址方式，支持所有的汇编伪指令，能正确产生 HEX 文件和 LST 文件。同时它能报告汇编过程中发现的各种错误，包括词法、语法、语义等的错误，并显示出错行的行号和错误类型等以便用户改正。

该系统将编辑、编译源程序和程序下载、帮助信息（源程序、指令表等）都集成在统一的开发环境中，功能齐全，界面友好，而且还提供了 24 个硬件综合性和设计性实验演示。此外，该系统实现了代码直接下载的功能，在计算机上调试好的程序可以脱离编程器，直接通过计算机的串口下载到单片机实验设备上，实验方便灵活。

其主要模块的设计主要有以下几个部分：文件操作设计；编辑器的设计；编译器的设计；参数设置设计；用户登录模块设计；帮助文件模块设计；代码下载功能的设计。其中文件操作实现对文件的读、写、打印、关闭和保存等功能。编辑器实现对源文件的编辑，如查找、剪切、复制等功能。帮助文件显示单片机相关概念、指令附录、指导手册和每个实验的具体信息等。编译器可实现对 MCS-51 汇编语言的编译功能，其主要目标是将单片机汇编语言源程序文件翻译成写入单片机的 Intel HEX 文件和为用户提供源码与目标码对照的列表（list）文件。整个编译过程按照词法分析、语法分析、语义分析、目标代码生成等几个阶段进行，通过两遍扫描实现对 MCS-51 汇编语言的翻译，并生成写入单片机的目标文件、为用户提供源码与目标码对照的列表文件的信息以及编译过程中出现的错误反馈信息。代码下载功能实现将生成的目标代码（Intel Hex 格式）通过串口直接下载到单片机的功能，从而省去了以往必须要使用编程器才能下载的烦琐步骤及相关实验设备。

该系统的安装和使用方法非常简单，无任何插件，不需修改注册表，是一个绿色软件。使用时只需要将“YX51-II_绿色版”文件夹复制到所使用的电脑上，然后双击文件夹中的可执行文件“YX51.exe”（如图 4-1 所示），就能轻松使用该软件。

5.2 展望

软件开发并不一定难，难就难在如何开发出有用的软件。软件的设计出发点可以概括为 3W(Who, What, Why)。即为谁设计、要解决哪些用户的问题、为什么要解决这些用户的问题。对企业而言，设计是为了生存。设计之源在于用户。满足用户需求，便于用户使用，同时又使得制作工艺尽可能简单，这就是设计之本。

集成开发环境的研制是个系统工程，需要多方面的知识。如软件工程、算法、数据结构、面向对象程序设计、Windows API、数据库、人工智能、编译原理、单片机原理等等，独立完成这样复杂的一个系统设计是有难度的，也不可能做到尽善尽美，该系统仍需继续完善。例如源代码自动纠错及函数索引功能等。此外，有一部分冗余代码留存其中，需要不断去发现、修改。最后，IDE 本身仍在不断改进和发展，同样，调试手段也是在发展，都需要逐步引入 IDE 的设计之中。

嵌入式软件 IDE 是一个比较专业的系统，需要将开发人员的开发经验集成到其中，不断完善，才能成为一个成熟可靠方便的开发环境。该系统显然还有一定差距。笔者希望能够在不断修改之中将它渐渐完善起来，至少能够成为一个具有参考价值的嵌入式 IDE 的例子。

参考文献

[1]朱德新，谢丹夏，程旭. 面向嵌入式系统的优化编译器生成器设计[J]. 小型微型计算机系统，2002，32(1)：1-3.

[2]占志彪. 用VB开发汇编集成编译器[J]. 温州师范学院学报，2004，25(5)：79-81.

[3]MIPS Teshnologies group.MIPS RO Processor Core MIPS32 4Kp-Embedded[M].Mountain View：MIPS Technologies Inc，1999：76-90.

[4]韩中元，杨泽雪. 汇编语言集成开发环境与智能上机辅导系统的开发[J]. 黑龙江工程学院学报，2005，19(1)：34-36.

[5]郑阿奇，丁有和，郑进等. Visual C++实用教程（第2版）[M]. 北京：电子工业出版社，2003：1-22.

[6]童长武，王博，贾春丽，魏居作. Visual C++计算机语言函数应用[M]. 北京：科学出版社，2000：440-453，531-558.

[7]南建辉，熊鸣等. MCS-51单片机原理及应用实例[M]. 北京：清华大学出版社，2004：72-163.

[8]钱春英. C51编译器在单片机系统开发中的若干问题[J]. 无锡商业职业技术学院学报，2003，3(3)：8-10.

[9]严俊康. MCS-51单片机汇编语言中的伪指令[J]. 沙洲职业工学院学报，2000，3(1)：7-11.

[10]Microsoft Corporation. MSDN Library July 2000.

[11]黄海林，范东睿，许彤等. 嵌入式处理器在片调试功能的设计与实现[J]. 计算机辅助设计与图形学学报，2006，18(7)：1005-1010.

[12]何立民. 单片机应用技术选编(1-8)[M]. 北京：清华大学出版社，1997：28-139.

[13]周红波. 嵌入式系统软件开发环境中调试器的设计[J]. 微计算机信息，2006，22（5-2）：60-61.

[14]Maek C Paulk. The Capability Maturity Model[M]. MA. USA Addison-Wesley，2000：78-146.

[15]陈火旺，刘春林，刘越. 程序设计语言---编译原理[M]. 北京：国防工业出版社，2005：309-324.

[16]王静，张磊. PDA设备的串行通信程序设计[J]. 新乡师范高等专科学校学报，2002，16(2)：25-27.

[17]陈宏刚，林斌，刘小宁等. 软件开发的科学与艺术[M]. 北京：电子工业出版社，2002：101-156.

[18]吕军，杨琦，罗建军，刘陆放. Visual C++与面向对象程序设计教程[M]. 北京：高等教育出版社，2004：160-212.

[19]胡昌林，李永军，卢海星，陶贵明. VC编程中的消息机制及关键函数[J]. 微计算机信息，2006，22(5-3)：250-251.

[20]凌贤伍，吴永礼. Visual C++6.0学习教程[M]. 北京：北京大学出版社，1999：306-340.

[21]沈洁萍，章红，陈勇. 汇编语言集成编译环境的开发--几种实时捕获控制台程序标准输出的方法[J]. 微计算机信息，2004，20(6)：120-121.

[22]Ravi Sethi. Programming Languages：concepts and constants[M]. Addison-Wesley Publishing Company，1996：111-113.

[23]杨光，冉峰. 单片机汇编器的设计与实现[J]. 微计算机应用，2005，26(2)：231-233.

[24]习华勇，石祥钟. 汇编语言集成开发环境的设计与实现[J]. 华北水利水电学院学报，1996，17(2)：20-23.

[25]李光宇，李延新，袁爱进. 面向嵌入式系统的编译器设计及实现[J]. 微计算机信息，2006，22(10-2)：175-177.

[26]胡汉才. 单片机原理及接口技术[M]. 北京：清华大学出版社，2004：127-138.

[27]严蔚敏，吴伟民. 数据结构（C语言版）[M]. 北京：清华大学出版社，2002：165-178.

[28]James Lconger.MicroSoft Windows API使用大全[M]. 北京：北京科海培训中心，2001：303-310.

[29]赵强，宫辉力，赵文吉，邓伟. 基于VC++环境的ComGIS 开发初探[J]. 计算机工程与应用，2004，40(7)：214-216.

[30]Peter W. Goftan. 精通串行通信[M]. 北京：电子工业出版社，2001：54-59.

[31]Douglas Boling.Windows CE程序设计[M]. 北京：北京大学出版社，2002：187-190.

[32]惠鸿忠. 利用VC++6.0实现上位机与PLC的串行通信[J]. 微计算机信息，2006，22(6-1)：55-57.

[33]匡万，毛幼菊. 单片机与PC机串行通信的研究[J]. 微计算机信息，1998，14(4)：24-26.

[34]蔡白兴，徐光佑. 人工智能及其应用(第二版)[M]. 北京：清华大学出版社，1996：12-67.

[35]陈立定. Windows98下串行通信编程技术及其应用[J]. 软件时空，2000，23(5)：12-16.

[36]张杜勇，陈芳琼. API for Windows2000/XP详解[M]. 北京：清华大学出版社，2003：52-107.

[37]Walter Savitch. Problem solving with VC++： the object of programming[M]. Addison -Weslety Publishing Commpany，1995：331-335.

[38]Brian W.Kernighan anf Dennis M.Ritchie. The C Programming Language[M]. Prentice Hall，PTR，1998：179-180.

[39]龚建伟，熊光明. Visual C++/Turbo C串口通信编程实践[M]. 北京：电子工业出版社，2004：116-118.

[40]Michael R. Sweet. Serial Programming Guide for POSIX Operatring Systems[M]. Addison-Wesley Publishing Company，1999：156-159.

[41]赵琳，吴亚君. 基于Visual C++的PC机与单片机串行通信的实现[J]. 沈阳电力高等专科学校学报，2004，6(2)：19-20.

[42]郑振涛. 用FreeCHM免费制作CHM电子书[J]. 电脑应用文萃，2005，(07)：67-68.

致 谢

光阴荏苒，日月如梭，转眼已度过了三年的研究生学习生活。这是我人生新的转折。在这三年的学习研究过程中，我深受导师毋茂盛教授的指导和教诲，他循循善诱的教导和不拘一格的思路给予我无尽的启迪。毋老师不仅在论文内容上给与我指导，还培养了我对科研的浓厚兴趣，为我以后进一步的深入学习提供了方向和方法。导师严谨务实的治学态度，身体力行的工作作风，都给我留下了深刻的印象，使我终生受用。在此，谨向毋老师致以最真诚的感谢！

同时，我还要感谢徐久成教授、王晓东教授、郑延滨教授、孙玉强教授、闫林副教授、刘春红老师等各位领导和老师的关心和指导！

此外，感谢所有对我的论文提供帮助的同学，特别是张百成同学！

最后，感谢河南师范大学计算机与信息技术学院的培养！

攻读学位期间的科研成果

参与完成一项河南省科技攻关项目（项目编号为0524220074），通过河南省科技厅科技成果鉴定，鉴定证书编号为豫科鉴字（2006）第163，科技成果登记号为9412006Y1237。

共发表四篇论文：

[1]闫娟，毋玉芝. 径向基函数神经网络曲面数据破损修补[J]. 微计算机信息，2007，23(3-1)：240-241.

[2]闫娟. 图像检索技术的研究和发展[J]. 新疆石油教育学院学报，2006，(5)：152-153.

[3]崔金玲，闫娟. 基于SNMP的校园网络性能管理系统的实现[J]. 河南师范大学学报，2007，35(1)：59-61.

[4]田喜平，李立新，闫娟. 基于补偿度和合作度的重复谈判研究[J]. 计算机工程与设计，2007，28(2)：455-458.

MCS-51单片机汇编语言程序IDE设计与实现

作者： 闫娟
学位授予单位： 河南师范大学

相似文献(10条)

1.学位论文 沈显庆 单片机系统仿真—生成用户硬件电路和汇编语言程序的故障诊断 2004

该文针对大中专院校开设的《单片机原理与接口技术》课程实验教学环节中,单片机硬件仿真器容易损坏及维护问题,研制开发了单片机系统仿真软件.它将完全取代单片机硬件仿真器和单片机汇编语言仿真软件,取而代之的是以PC机为平台,基于VB的可视化界面、事件驱动原理,在VB的环境下,实现对单片机系统软件的模拟和硬件电路的仿真.该课题包括两部分,第一部分是通过人机对话的形式输入用户对硬件电路的要求,自动生成所需硬件电路;第二部分是根据硬件和用户所编写的汇编语言程序,仿真系统(硬件和汇编语言程序)的运行过程.该文重点介绍第一部分即采用专家系统配置单片机系统硬件电路.该文首先叙述了离散事件系统的仿真原理,详细论述了离散事件系统的仿真策略,为单片机系统仿真提供理论基础.其后,该文重点阐述了该课题所采用的基于面向对象技术专家系统的基本理论以及知识的获取、知识库的建立及推理策略.并在此基础上建立了基于故障诊断专家系统,实现对用户编写的汇编语言程序的故障诊断.最后,论述了整个系统的程序设计思想和实现方法,编写了实现课题要求的软件及调试方法.并对单片机系统响应外部事件(中断)提出了设想.

2.期刊论文 庞(山严)英.曹海建.PANG Yan-ying.CAO Hai-jian 单片机汇编语言程序开发平台的设计 -青岛大学学报（工程技术版）2005,20(3)

为了提高单片机汇编语言程序的编程效率,缩短开发周期,从平台模式的观点出发,阐述了单片机汇编语言程序开发平台的构成要素、功能和特点以及设计思路,并给出数学运算平台的基本程序框图.已经设计开发的<PIC单片机数学运算程序平台>,经实际试用证明,该平台方便了程序开发者编写运算程序.

3.期刊论文 张云鹤.刘玲.ZHANG Yun-he.LIU Ling 基于Visual C++6.0的反汇编语言程序的设计与研究 -仪器仪表用户2006,13(1)

本文介绍了一种基于Visual C++6.0的反汇编语言程序的方法,以Motorola M68HC11系列单片机为例,介绍了反汇编语言的设计原理,并且介绍了设计步骤和方法.

4.学位论文 刘文秀 单片机系统仿真—对用户的软、硬件系统运行过程仿真 2004

单片机系统仿真软件是面向各大中专院校的师生而研制开发的.它完全依靠软件手段来开发单片机的应用软件,无需任何单片机硬件设备.该仿真软件能在通用计算机上实现对单片机的硬件模拟、指令模拟和运行状态的模拟,集编程、修改、运行等功能于一体.单片机系统仿真软件对大专院校的单片机教学起到很重要的辅助作用.使用单片机系统仿真软件不仅可以充分利用现有丰富的通用机资源提高实验效率,而且,只需对软件开发系统的部分程序稍作修改,便可升级为新的单片机软件开发系统以满足教学和工程设计的要求.该课题包括两部分:第一部分是利用人工智能专家系统知识,自动生成用户所需的硬件电路;第二部分是根据硬件和用户所编写的汇编语言程序,仿真系统(硬件和汇编语言程序)的运行过程,实现对固定电路的仿真.该文重点介绍第二部分即对单片机应用系统硬、软件两部分的仿真.该论文首先从离散事件系统仿真原理出发,通过四种仿真策略的比较,确定采用进程交互法实现仿真钟的管理.并以此确定整个程序的时间管理.然后,该文介绍了单片机系统仿真软件的功能.它是以Windows为操作平台,利用VB6.0的可视化界面、事件驱动原理、各种控件及数据库编程知识,开发的仿真软件.经过编程及调试,基本上实现了主要的单片机实验的仿真.其中包括数据存储器、程序存储器、键盘的输入、数码管的输出、数/模转换器、模/数转换器、并行口、串行口等等的仿真.最后,在软件工程的指导下,介绍了软件的结构和设计过程;论述了整个系统的程序设计思想和实现方法,编写了实现课题要求的软件及调试方法.

5.期刊论文 徐爱钧.彭秀华 单片机高级语言C51与汇编语言ASM51的通用接口 -微计算机应用2000,21(1)

在单片机应用系统中采用高级语言编程时注意与汇编语言程序正确接口,可极大地提高编程效率.本文讨论高级语言C51编译器对目标程序的段管理,与汇编语言程序接口时的参数传递方法,给出了通用接口程序设计实例.

6.学位论文 刘明 单片机智能控制医用设备的研究 2006

随着生活水平的提高,人们对医疗灭菌消毒更加关注了.而中小诊所的消毒灭菌设备简陋,自动化程度低,消毒质量难以控制.已经不能很好的满足实际需要了。 单片机进入中国20余年以来以其体积小、功能强、扩展灵活、使用方便等特点，逐渐应用到各行业的工程实际当中。目前出现的一款较新的PIC单片机,以其内部集成模块多,易于开发,抗干扰能力强等特点,逐渐在中小型控制系统中得到了广泛的应用。模糊控制是一种新型的智能控制方法。其最突出的优点是无需建立控制对象的数学模型，尤其适用于非线性、时变、滞后系统的控制。基于以上考虑,以及对各种灭菌方法的综合比较，参考国内外的发展动态。 本文将重点研究了基于PIC16F877单片机模糊控制的温度压力蒸汽灭菌器系统。 本论文主要工作和研究内容如下： 通过比较确定以PIC16F877单片机为核心的、以模糊控制为控制策略的蒸汽灭菌系统； 对温度压力的测控电路进行了设计； 在MPLAB开发环境下完成了整个控制系统的程序编制； 利用MATLAB对模糊控制系统进行仿真,结合仿真结果优化参数, 使得模糊控制器具有动态响应快，超调量小，稳定性好的特点;然后在虚拟试验室(PROTEUS)中搭建整个系统的硬件电路,将预先编制好的汇编语言程序写入(虚拟)单片机,对设计系统的软硬件进行整体仿真，并给出了仿真结果图及相应的分析。

7.学位论文 漆念华 基于单片机的对讲系统的研究开发 2004

随着社会经济的发展，科学的进步，人们对居住空间提出了新的要求，既环境的安全性。应这种需求，市场上各种住宅楼宇安全设备应运而生，其中最广泛采用的，也最被广大用户所接受的设备即楼宇对讲系统。 在本文中，以楼宇对讲系统为研究对象，设计了一种新型的简线制对讲系统。该系统技术解决方案可推广应用于工业、医疗等领域的呼叫对讲系统中。 此系统以单片机为核心，通过采用专用解码器及合理主机分机电路设计，实现了楼宇对讲系统功能2线通讯完成，同时，较以前多线制与5总线制系统实现了主机、分机电路简化、成本降低、可靠性提高。主要的研究成果可归纳为以下几点：1)对本文的楼宇对讲系统主机与分机通讯，采用了一种新的运行方式：数据线与语音线合用，主机通过与专用解码器通讯实现分机选通控制，然后通过主机检测电路对分机状态检测，实现通话、开锁控制。设计出了新的相关电路，以满足这种运行要求。2)采用专用解码器实现楼层分线功能，同时通过单片机对VD5026数据传送进行仿真，实现与VD5027进行串行通讯。从而简化了电路，提高了通讯可靠性。3)编写了系统主控单片机、显示控制单片机的汇编语言程序，及定义了相关通讯协议。

8.期刊论文 高晓红.GAO Xiao-hong I2C总线在单片机串行扩展中的应用 -吉首大学学报（自然科学版）2008,29(1)

介绍I2C总线的结构、工作原理、数据传输方式和优点,对于不带PC总线接口的单片机.利用其常规的I/O口来模拟I2C总线接口,消除了串行扩展的局限性,并设计了数字温度传感器DS1624与不带I2C总线接口的单片机AT89C52的硬件系统,给出了软件实现的流程图及汇编语言程序.

9.期刊论文 何桥 单片机键盘"一键多用"的实现 -电子与电脑2007(4)

本文介绍了在单片机的键盘上实现"一键多用"的方法和技巧.并给出了程序流程图和单片机的汇编语言程序.

10.期刊论文 熊益铭.Xiong Yiming C语言在单片机串口通信中的应用 -电子科技2005(8)

介绍了单片机串行通讯基本知识,及如何利用MAX32 32扩展单片机配置接口.在此基础上,突破汇编语言程序的可读性和可移植性差的缺陷,用C语言编写了一套接口程序,可根据需要稍加改动,就可以适应各种实际的应用.