第 3 章　C51 程序设计基础

本章结合 MCS-51 单片机的特点对 C51 的一些基本知识进行阐述，重点讲述与单片机软件编程密切相关的内容，而对于标准 C 语言的一般内容只做简单介绍。在学习本章内容之前，希望读者能够先行学习 C 语言程序设计的基础知识。

3.1　C51 概述

早期的单片机系统主要采用汇编语言编写程序，但是汇编语言程序的可读性和可移植性都比较差，而且采用汇编语言编写单片机应用程序，开发周期长，调试和查错也比较困难。为了提高编写单片机应用程序的效率、改善程序的可读性和可移植性，采用高级编程语言是很好的选择。

C 语言是一种通用的计算机程序设计语言，它既具有一般高级语言的特点，又能直接对计算机的硬件进行操作，表达和运算能力也比较强。Keil C51 是一种专为 MCS-51 单片机设计的 C 语言编译器，支持用符合 ANSI 标准的 C 语言进行程序设计。同时，考虑到 MCS-51 单片机的特殊性，人们对 C 语言进行了扩展，称之为 C51 语言。

3.1.1　C51 程序结构概述

C51 程序结构如下：

```
#include <reg51.h>                //预处理命令
//全局变量定义
//函数声明
char fun( )                       //功能函数定义
{
……                                //功能函数体
}
void 函数名( ) interrupt x        //中断服务函数定义
{
……                                //中断服务函数体
}
void main( )                      //主函数
{
    //局部变量定义
    //单片机寄存器初始化函数
    while(1)
    {
```

```
    ……                          //主函数体
    }
    }
```

注意事项：

（1）**一个 C51 源程序必须包括一个 main 函数**，也可以包括若干个其他函数。main 函数可以调用别的功能函数，但不允许其他功能函数调用 main 函数。main 函数是主函数，是程序的入口，不论 main 函数处在程序的哪个位置，程序都从 main 函数开始执行，执行到 main 函数结束则结束。一般将 main 函数放在程序尾。

（2）“#include<xxx.h>”语句表示包含库函数。库函数是 C51 在库文件中已定义的函数，其函数说明在相关的头文件中。用户编程时只要用 include 预处理命令包含相关头文件，就可在程序中直接调用。

（3）用户自定义函数是用户自己定义、自己调用的函数。

（4）全局变量在程序的所有地方都可以赋值和读出，包括中断服务函数、主函数，因此编写单片机程序要善于使用全局变量。

（5）如果使用中断、定时器、串行口等功能，则必须对单片机特殊功能寄存器进行初始化。

（6）**主程序必须采用闭环结构**，即一个闭环循环，采用 while(1){…}（类似汇编语言例程中的 LOOP: …… LJMP　LOOP）实现，表示单片机的 main 函数中的部分代码是循环执行的。在实际应用中，单片机程序先对寄存器进行一次初始化操作，然后使用循环语句（比较常用的是“while(1)”语句）循环执行其循环代码部分。后面的例程由于部分主程序没有实际功能，本书的例题中一般用一条“while(1)”语句（类似汇编语言例程中的 LJMP　$）来代替循环执行代码。

（7）注释部分可以采用“//注释内容”或“/* 注释内容 */”的形式。

（8）程序中必须采用英文标点。

3.1.2　C51 对标准 C 语言的扩展

为了更好地支持 MCS-51 单片机系统的开发，C51 对标准 C（ANSI C）语言进行了扩展，不仅完全支持 C 语言的标准指令，还进行了诸多优化。C51 中的关键字除 ANSI C 标准的 32 个关键字之外，还根据 MCS-51 单片机的特点扩展了相关的关键字。表 3-1 按用途列出了 Keil C51 编译器扩展的关键字。

表 3-1　Keil C51 编译器扩展的关键字

关键字	用途	说明
_at_	地址定位	为变量进行存储器绝对空间地址定位
alien	函数特性声明	用于声明与 PL/M51 兼容的函数
small	存储器模式	指定使用内部数据存储器空间
compact	存储器模式	指定使用“分页寻址”的外部数据存储器空间
large	存储器模式	指定使用外部数据存储器空间
code	存储器类型声明	声明程序存储器空间

续表

关键字	用途	说明
data	存储器类型声明	指定直接寻址的内部数据存储器
bdata	存储器类型声明	指定可位寻址的内部数据存储器
idata	存储器类型声明	指定间接寻址的内部数据存储器
pdata	存储器类型声明	指定“分页寻址”的外部数据存储器
xdata	存储器类型声明	指定外部数据存储器
bit	位变量声明	声明位变量
sbit	位变量声明	声明一个可位寻址变量
sfr	特殊功能寄存器声明	声明一个 8 位的特殊功能寄存器
sfr16	特殊功能寄存器声明	声明一个 16 位的特殊功能寄存器
interrupt	中断服务函数声明	定义一个中断服务函数
using	工作寄存器组定义	定义工作寄存器组 R0～R7
priority	多任务优先声明	规定 RTX 或 RTX51 Tiny 的任务优先级
reentrant	再入函数声明	定义一个再入函数
task	任务声明	定义实时多任务函数

3.2　C51 的编译和编译预处理

3.2.1　编译

C51 编译器的作用是将 C 语言源程序翻译成 MCS-51 单片机的可执行代码。在 Keil μVision4 集成开发环境中，可以通过“Project”菜单的“Options for Target”选项中的“C51”标签页来设置各种控制命令，通过“Project”菜单的“Build Target”选项可以很方便地按所设置命令对 C51 源程序进行编译和链接（详见 9.2 节）。

编译命令分为源控制命令、列表控制命令和目标控制命令三大类。源控制命令用于宏定义及确定需要进行编译的文件名。列表控制命令用于规定编译后所产生列表文件的格式及是否生成某些特殊内容，所生成的列表文件扩展名为“.LST”。目标控制命令用于控制编译之后生成目标文件的格式和内容，例如，控制目标文件的优化级别、使用不同的编译模式来规定变量的存储器空间、是否在所生成的目标文件中加入符号调试信息等，所生成的目标文件扩展名为“.OBJ”。在编译命令中，目标控制命令最多，作用最大，使用最频繁。

C51 编译器还可以生成扩展名为“.I”和“.SRC”的输出文件。扩展名为“.I”的输出文件中包含由预处理器命令展开的源文件，对所有宏都进行展开，同时删除了所有注释。扩展名为“.SRC”的输出文件为从 C51 源程序产生的汇编语言代码文件，可以再用 A51 汇编器进行汇编。默认状态下输出文件与源程序文件同名，但扩展名不同。

C51 编译器在对源程序进行编译时将自动查错，并在编译完成之后输出 0～3 级错误提示，0 级表示没有错误和警告，1 级表示仅有警告，2 级表示同时存在错误和可能的警告，3 级表示存在致命错误。用户可根据错误级别决定是否需要修改源程序文件，在 Keil μVision4 环境中将鼠标指针指向输出窗口中的某个错误提示信息并双击，光标会自动跳到编辑窗口中发生错误的源程序行，这对于判断错误原因及修改源程序十分方便。

3.2.2 编译预处理

C 语言与其他高级程序设计语言的主要区别就是对程序的编译预处理功能，编译预处理器是 C 语言编译器的一个组成部分。C 语言中，预处理命令可以在很大程度上为 C 语言本身提供许多功能和实现符号等方面的扩充，增强了 C 语言的灵活性和方便性。编写程序时把预处理命令加在需要的地方，它只在程序编译时起作用，且通常是按行进行处理的，因此又称为编译控制行。

C 语言的预处理命令类似于汇编语言中的伪指令。编译器在对程序进行编译之前，先对程序中的编译控制行进行预处理，再将预处理的结果与整个 C 语言源程序一起进行编译，产生目标代码。

1. 宏定义

宏定义命令为#define。它的作用是用一个标识符替换一个字符串，而这个字符串既可以是常数、表达式、其他字符串等，又可以是带参数的宏。宏定义的简单形式是符号常量定义（不带参数的宏定义），复杂形式是带参数的宏定义。

（1）不带参数的宏定义。不带参数的宏定义又称符号常量定义，一般格式为：

```
#define 标识符 常量表达式
```

其中，“标识符”是所定义的宏符号名（也称宏名）。它的作用是在程序中使用指定的标识符来代替所指定的常量表达式。例如，#define NaN 0xFFFFFFFF 就是用 NaN 这个符号来代替常数 0xFFFFFFFF。使用了这个宏定义之后，程序中就不必每次都写出常数 0xFFFFFFFF，而可以用符号 NaN 来代替了。

编译时，编译器会自动将程序中所有的符号名 NaN 都替换成常数 0xFFFFFFFF。这使得在 C 语言源程序中可以用一个简单的符号名来替换一个很长的字符串，还可以使用一些有一定意义的标识符，以提高程序的可读性。通常程序中的所有符号定义都集中放在程序开始处，这样便于检查和修改，从而提高程序的可靠性。如果需要修改程序中的某个常量，则不必修改整个程序，只要修改一下相应的符号常量定义即可。

实际使用宏定义时，通常将宏符号名用大写字母表示，以区别于其他的变量名。宏定义不是 C 语言语句，因此在宏定义行的末尾不要加分号。例如：

```
#define     PAI     3.1415926
#define     R       3.0
```

（2）带参数的宏定义。带参数的宏定义与符号常量定义的不同之处在于，对于源程序中出现的宏符号名，不仅进行字符串替换，还进行参数替换。带参数的宏定义的一般格式为：

```
#define  宏符号名(参数表) 表达式
```

其中，表达式内包含了括号中所指定的参数，这些参数称为形式参数（简称形参），在以后的程序中它们将被实际参数替换。程序中所有带实际参数表的宏符号名用指定的表达式来替换，同时用参数表中的实际参数替换表达式中对应的形式参数。

带参数的宏定义常用来代表一些简短的表达式，以代替函数调用，从而提高程序的执行效率。例如：

```
#define  MIN(x,y)  (((x)<(y))? (x):(y))
```

定义了一个带参数的宏 MIN(x,y)，以后在程序中就可以用这个宏而不用函数 min( )。语句“m=MIN(u,v);”经宏展开后成为“m=((u)<(v)?(u):(v));”。

2. 文件包含

文件包含是将另一个指定的文件内容包含进来。文件包含命令的一般格式为：

```
#include  <文件名> 或
#include  "文件名"
```

文件包含命令#include 的功能是用指定文件的全部内容替换该预处理行，当采用<文件名>格式时，在头文件目录中查找指定文件；当采用"文件名"格式时，先在当前目录中查找指定文件，若没找到，再到头文件目录中查找。

为了适应模块化编程的需要，可以将组成 C 语言程序的各个功能函数分散到多个程序文件中，分别由若干人员完成，最后用#include 命令将它们嵌入一个总的程序文件中。也可以将一些常用的符号常量、带参数的宏及构造类型的变量等定义在一个独立的文件中，当某个程序需要时再将其包含进来，这样做可以减少重复劳动，提高程序的编制效率。由于某种原因需要修改其内容时，只需对相应的包含文件做修改，而不必对使用它们的各个程序文件都做修改，这样有利于程序的维护和更新。

文件包含命令#include 通常放在 C 语言程序的开头，被包含文件的类型通常为以“.h”为后缀的头文件和以“.c”为后缀的源程序文件，它们既可以是系统提供的，也可以是自己编写的。需要注意的是，一个文件包含命令只能指定一个被包含文件，如果程序中需要包含多个文件，则需要使用多个文件包含命令。

C51 在预处理上的功能与 C 语言相同。

3.3　C51 的基本语法

3.3.1　常量

常量又称为标量，它的值在程序执行过程中不能改变。常量的数据类型有整型、浮点型、字符型和字符串型等。

1. 整型常量

整型常量就是整型常数，可表示为以下几种形式。

十进制整数：如 1234、−5678、0 等。

十六进制整数：ANSI C 标准规定十六进制数据以 0x 开头，数字为 0～9、a～f。如 0x123 表示十六进制数，相当于十进制数 291。又如，−0x1A 表示十六进制数，相当于十进制数−26。

长整数：在数字后面加一个字母 L 就构成了长整数，如 2048L、0123L、0xff00L 等。

2. 浮点型常量

浮点型常量有十进制数表示形式和指数表示形式两种。十进制数表示形式又称定点表示形式，由数字和小数点组成。如 0.3141、.3141、314.1、3141.及 0.0 都是十进制数表示形式

的浮点型常量。在这种表示形式中，如果整数或小数部分为 0，可以省略不写，但必须有小数点。指数表示形式为：

```
[±]数字[.数字]e[±]数字
```

其中，[]中的内容为可选项，根据具体情况可有可无，但其余部分必须有。如 123e4、5e6、-7.0e-8 等都是合法的指数形式的浮点型常量；而 e9、5e4.3 和 e 都是不合法的表示形式。

3. 字符型常量

字符型常量是单引号内的字符，如'a'，'b'等。对于不可显示的控制字符，可以在该字符前面加一个反斜杠“\”组成转义字符。转义字符可以完成一些特殊功能和输出时的格式控制。

4. 字符串型常量

字符串型常量由双引号" "内的字符组成，如"ABCD"、"$1234"等都是字符串型常量。C 语言将字符串型常量作为一个字符类型数组来处理，在存储字符串型常量时要在字符串的尾部加一个转义字符“\0”作为该字符串型常量的结束符。需要注意的是，字符串型常量首尾的双引号是界限符，当需要表示双引号字符串时，可用双引号转义字符“\"”来表示。

3.3.2 变量

变量是一种在程序执行过程中其值能不断变化的量。在使用一个变量之前，必须要进行定义，用一个标识符作为变量名并指出它的数据类型和存储类型，以便编译系统为它分配相应的存储单元。在 C51 中对变量进行定义的格式如下：

```
[存储种类] 数据类型 [存储类型] 变量名;
```

其中“存储种类”和“存储类型”是可选项。变量的存储种类有四种：自动（auto）、外部（extern）、静态（static）和寄存器（register）。其中，存储种类为静态和外部的变量存放在静态存储区，存储种类为自动的变量存放在动态存储区，存储种类为寄存器的变量则直接送入寄存器。定义一个变量时如果忽略存储种类选项，则该变量将为自动（auto）变量。定义一个变量时除需要说明其数据类型之外，Keil C51 编译器还允许说明变量的存储类型。接下来对这些概念进行详细解释。

1. 标准 C 语言数据类型

数据是单片机操作的对象，是具有一定格式的数字或数值，数据的不同格式就称为数据类型。标准 C 语言支持的基本数据类型，如表 3-2 所示。

表 3-2 标准 C 语言支持的基本数据类型

数据类型	位数	字节数	值域
signed char	8	1	-128~+127，有符号字符变量
unsigned char	8	1	0~255，无符号字符变量
signed int	16	2	-32768~+32767，有符号整型数
unsigned int	16	2	0~65535，无符号整型数
signed long	32	4	-2147483648~+2147483647，有符号长整型数

续表

数据类型	位数	字节数	值域
unsigned long	32	4	0~+4294967695，无符号长整型数
float	32	4	±1.175494E-38~±3.402823E+38，浮点数（精度：6~7 位）
double	64	8	±4.940656458412465E−324~±1.797693134862316E+308，浮点数（精度：15~16 位）

2. C51 扩展的数据类型

针对 MCS-51 单片机的硬件特点，C51 在标准 C 语言的基础上，扩展了 4 种数据类型，主要针对单片机片内数据存储区，如表 3-3 所示。需要注意的是，对扩展的数据类型的数据不能采用指针进行存取操作。

表 3-3　C51 扩展的数据类型

数据类型	位数	字节数	值域
bit	1	—	0 或 1
sfr	8	1	0~255
sfr16	16	2	0~65535
sbit	1	—	0 或 1

（1）位变量 bit。bit 直接用于定义位变量，编译器会对其分配地址。位变量分配在内部 RAM 的 20H~2FH 单元相应的位区域，位地址范围是 00~7FH，共 128 个。用 bit 定义位变量的值可以是 1，也可以是 0。定义方法如下：

```
bit 位变量;
```

例如：

```
bit PX;
/*定义一个称为 PX 的位变量，存放在地址单元 20H～2FH 中的某一位，其值可以是 0 或 1*/
```

（2）特殊功能寄存器 sfr 和 sfr16。特殊功能寄存器分布在片内数据存储区的 80H~FFH 单元之间，“sfr”数据类型占用一个内存单元，利用它可以直接对 MCS-51 单片机的特殊功能寄存器进行定义。“sfr16”数据类型则占用两个内存单元，利用它可以定义占两字节的特殊功能寄存器，在定义时地址选用低位地址。

定义方法如下：

```
sfr  特殊功能寄存器名=地址;
sfr16  特殊功能寄存器名=地址;
```

例如：

```
sfr     PSW = 0xD0;      //定义寄存器 PSW 的地址为 D0H
sfr     P1 = 0x90;       //定义寄存器 P1 的地址为 90H
sfr16   DPTR = 0x82;     //定义寄存器 DPTR 的低 8 位字节地址为 82H，高 8 位字节地址为 83H
```

通过 sfr 和 sfr16 对 MCS-51 单片机的特殊功能寄存器进行定义，在程序的后续语句中可以直接对特殊功能寄存器进行操作，如在程序后续的语句中可以用“P1=0xff”使 P1 的所有引脚输出高电平。

（3）特殊功能位 sbit。sbit 用于定义位变量的名字和地址。被定义的位变量是特殊功能寄存器中的可以进行位寻址的确定位，该位变量的绝对地址是确定的，且不用编译器分配。利用 sbit 定义位变量名字和地址的方法有如下三种。

① 第一种方法。

```
sbit        位变量名=位地址;
```

这种方法将位的绝对地址赋给位变量，位地址必须在 80H~FFH 之间。

例如：

```
sbit    OV = 0xD2;      //定义 OV 位的绝对地址为 D2H
sbit    CY = 0xD7;      //定义 CY 位的绝对地址为 D7H
```

② 第二种方法。

```
sbit        位变量名=特殊功能寄存器名^位位置;
```

当可寻址位位于特殊功能寄存器中时可以采用这种方法。其中，符号“^”前是特殊功能寄存器的名字，“^”后面的数字是特殊功能寄存器可寻址位在寄存器中的位置，取值必须是 0~7。

例如：

```
sfr     PSW = 0xD0;     //定义寄存器 PSW 的地址为 D0H
sbit    OV = PSW^2;     //定义 OV 位为 PSW.2
sbit    CY = PSW^7;     //定义 CY 位为 PSW.7
```

③ 第三种方法。

```
sbit        位变量名=字节地址^位位置;
```

这种方法以一个常数（字节地址）作为基地址，该常数必须在 80H~FFH 之间并能被 8 整除。“位位置”是一个 0~7 之间的常数。

例如：

```
sbit    OV = 0xD0^2;    //定义 OV 位为字节地址 D0H 的第 2 位
sbit    CY = 0xD0^7;    //定义 CY 位为字节地址 D0H 的第 7 位
```

注意：不要把 bit 与 sbit 混淆。bit 直接用于定义位变量，而 sbit 用于定义位变量的名字和地址。

3. 存储类型

定义 C51 一般变量的数据类型时，应同时提及它的存储类型（缺省则由系统默认选择）和与存储器结构的关系，因为 C51 定义的任何数据类型必须以一定的方式定位在 MCS-51 单片机的某一存储区中，否则就没有实际意义。

MCS-51 单片机有片内、片外数据存储区和片内、片外程序存储区。其中，片内数据存储区是可读写的，MCS-51 单片机（增强型）的衍生系列最多可有 256 字节的片内数据存储区，其中低 128 字节可直接寻址，高 128 字节（80H～FFH）只能间接寻址。在片内数据存储区的低 128 字节中，20H～2FH 的 16 字节可位寻址。因此，定义在片内数据存储区的数据存储类型有三种：data、idata 和 bdata。