附录、Keil C 库函数

C-51 软件包的库包含标准的应用程序，每个函数都在相应的头文件（.h）中有原型声明。如果使用库函数，必须在源程序中用预编译指令定义与该函数相关的头文件（包含了该函数的原型声明）。例如：

#include

#include

如果省掉头文件，编译器则期望标准的C 参数类型，从而不能保证函数的正确执行。

1 CTYPE.H：字符函数

在CTYPE.H 头文件中包含下列一些库函数：

函数名： isalpha

原型： extern bit isalpha(char)

功能： isalpha 检查传入的字符是否在‘A’-‘Z’和‘a’-‘z’之间，如果为真返回值为1，否则为0。

函数名： isalnum

原型： extern bit isalnum(char)

功能： isalnum 检查字符是否位于‘A’-‘Z’，‘a’-‘z’或‘0’-‘9’之间，为真返回值是1，否则为0。

函数名： iscntrl

原型： extern bit iscntrl(char)

功能： iscntrl 检查字符是否位于0x00～0x1F 之间或0x7F，为真返回值是1，否则为0。

函数名： isdigit

原型： extern bit isdigit(char)

功能： isdigit 检查字符是否在‘0’-‘9’之间，为真返回值是1，否则为0。

函数名： isgraph

原型： extern bit isgraph(char)

功能： isgraph 检查变量是否为可打印字符，可打印字符的值域为0x21～0x7E。若为可打印，返回值为1，否则为0。

函数名： isprint

原型： extern bit isprint(char)

功能：除与isgraph 相同外，还接受空格字符（0X20）。

函数名： ispunct

原型： extern bit ispunct(char)

功能： ispunct 检查字符是否位为标点或空格。如果该字符是个空格或32 个标点和格式字符之一（假定使用ASCII 字符集中128 个标准字符），则返回1，否则返回0。Ispunct 对下列字符返回1：

（空格）！“$%^&()+,-./:<=>?_[‘~{}

函数名： islower

原型： extern bit islower(char)

功能： islower 检查字符变量是否位于‘a’-‘z’之间，为真返回值是1，否则为0。

函数名： isupper

原型： extern bit isupper(char)

功能： isupper 检查字符变量是否位于‘A’-‘Z’之间，为真返回值是1，否则为0。

函数名： isspace

原型： extern bit isspace(char)

功能： isspace 检查字符变量是否为下列之一：空格、制表符、回车、换行、垂直制表符和送纸。为真返回值是1，否则为0。

函数名： isxdigit

原型： extern bit isxdigit(char)

功能： isxdigit 检查字符变量是否位于‘0’-‘9’，‘A’-‘F’或‘a’-‘f’之间，为真返回值是1，否则为0。

函数名： toascii

原型： toascii(c)((c)&0x7F);

功能：该宏将任何整型值缩小到有效的ASCII 范围内,它将变量和0x7F 相与从而去掉低7 位以上所有数位。

函数名： toint

原型： extern char toint(char)

功能： toint 将ASCII 字符转换为16 进制，返回值0 到9 由ASCII 字符‘0’到‘9’得到，10 到15 由ASCII 字符‘a’-‘f’（与大小写无关）得到。

函数名： tolower

原型： extern char tolower(char)

功能： tolower 将字符转换为小写形式，如果字符变量不在‘A’-‘Z’之间，则不作转换，返回该字符。

函数名： _tolower

原型： tolower(c);(c-‘A’+‘a’)

功能：该宏将0x20 参量值逐位相或。

函数名： toupper

原型： extern char toupper(char)

功能： toupper 将字符转换为大写形式，如果字符变量不在‘a’-‘z’之间，则不作转换，返回该字符。

函数名： _toupper

原型： _toupper(c);((c)-‘a’+’A’)

功能： _toupper 宏将c 与0xDF 逐位相与。

2 STDIO.H：一般I/O 函数

C51 编译器包含字符I/O 函数，它们通过处理器的串行接口操作，为支持其它I/O机制，只需修改getkey()和putchar()函数，其它所有I/O 支持函数依赖这两个模块，不需要改动。在使用8051 串行口之前，必须将它们初始化，下例以2400波特率，12MHz 初始化串口：

SCON=0x52

TMOD=0x20

TR1=1

TH1=0Xf3

其它工作模式和波特率等细节问题可以从8051 用户手册中得到。

函数名： _getkey

原型： extern char _getkey();

功能： _getkey()从8051 串口读入一个字符，然后等待字符输入，这个函数是改变整个输入端口机制应作修改的唯一一个函数。

函数名： getchar

原型： extern char _getchar();

功能： getchar()使用_getkey 从串口读入字符，除了读入的字符马上传给putchar 函数以作响应外，与_getkey 相同。

函数名： gets

原型： extern char *gets(char *s, int n);

功能：该函数通过getchar 从控制台设备读入一个字符送入由‘s’指向的数据组。考虑到ANSI 标准的建议，限制每次调用时能读入的最大字符数，函数提供了一个字符计数器‘n’，在所有情况下，当检测到换行符时，放弃字符输入。

函数名： ungetchar

原型： extern char ungetchar(char);

功能： ungetchar 将输入字符推回输入缓冲区，因此下次gets 或getchar 可用该字符。ungetchar 成功时返回‘char’，失败时返回EOF，不可能用ungetchar 处理多个字符。

函数名： _ungetchar

原型： extern char _ungetchar(char);

功能： _ungetchar 将传入字符送回输入缓冲区并将其值返回给调用者，下次使用getkey时可获得该字符，写回多个字符是不可能的。

函数名： putchar

原型： extern putchar(char);

功能： putchar 通过8051 串口输出‘char’，和函数getkey 一样，putchar是改变整个输出机制所需修改的唯一一个函数。

函数名： printf

原型： extern int printf(const char*, …);

功能： printf 以一定格式通过8051 串口输出数值和串，返回值为实际输出的字符数，参量可以是指针、字符或数值，第一个参量是格式串指针。

注：允许作为printf 参量的总字节数由C51 库限制，因为8051 结构上存贮空间有限，在SMALL 和COMPACT 模式下，最大可传递15 个字节的参数（即5 个指针，或1 个指针和3 个长字节），在LARGE 模式下，至多可传递40 个字节的参数。格式控制串包含下列域（方括号内的域是可能的）：

%[flags][width][.precision]type“width”域定义了参量欲显示的字符数，它必须是一个十进制数，如果实际显示的字符数小于“width”，输出左端补以空格，如果“width”域以0 开始，则左端补0。

“flag”域用来定义下面选项：

Falg　意义

-　输出左齐

+　输出值如果是有符号数值，则加上+/-号

‘ ‘(空格)

输出值如果为正则左边补以空格显示

#

如果它与0，x 或X 联用，则在输出前加上字符0、0x，0X。当与值类型g、G、f、e、E 联用时，‘#’使输出数产生一个十进制小数点。

b,B

它们与格式类型d、i、o、u、x、X 联用，这样参量类型被接受为‘[unsigned]char’，如：%bu，%bd 或%bx。

L,L

它们与格式类型d、i、o、u、x、X 联用，这样参量类型被接受为‘[unsigned]long’，如：%lu，%ld 或%lx。

*

下一个参量不作输出。

“type“域定义参量如下类型：

字符

类型

输出格式

d

int

有符号十进制数（16 位）

U

int

无符号十进制数

o

int

无符号八进制数

X，x

int

无符号十六进制数

f

float

[-]dddd.dddd 形式的浮点数

e，E

float

[-]d.ddddE[sign]dd 形式的浮点数

g，G

float

e 或f 形式浮点数，看哪一种输出形式好。

c

char

字符

s

pointer

指向一个带结束符号的串

p

pointer

带存贮器指示符和偏移的指针。M:aaaa。

M:=C(ode),D(ata),I(data),P(data) aaaa:指针偏移值。

例子：

printf(“Int-Val%d,Char-Val%bd,Long-Val%d”,I,c,l);

printf(“String%s,Character%c”,array,character);

printf(“Pointer%p”,&array[10]);

函数名： sprintf

原型： extern int sprintf(char *s，const char*，…);

功能： sprintf 与printf 相似，但输出不显示在控制台上，而是通过一个指针S，送入可寻址的缓冲区。

注：sprintf 允许输出的参量总字节数与printf 完全相同。

函数名： puts

原型： extern int puts(const char*，…);

功能： puts 将串‘s’和换行符写入控制台设备，错误时返回EOF，否则返回一非负数。

函数名： scanf

原型： extern int scanf(const char*，…);

功能： scanf 在格式串控制下，利用getcha 函数由控制台读入数据，每遇到一个值（符号格式串规定），就将它按顺序赋给每个参量，注意每个参量必须都是指针。scanf返回它所发现并转换的输入项数。若遇到错误返回EOF。格式串包括：

l 空格、制表符等，这些空白字符被忽略。
l 字符，除需匹配的“%”（格式控制字符）外。
l 转换指定字符“%”，后随几个可选字符；赋值抑制符“*”，一个指定最大域宽的数。
注：scanf 参量允许的总字节数与printf 相同，格式控制串可包括下列域（方括号内是可选的）：
%[flags][width]type
格式串总是以百分号开始，每个域包含一个或多个字符或数。“width”域定义了参量可接受的字符数，“width”必须是一个正十进制数。如果实际输入字符数量小于“width”，则不会进行填充。‘flag’域用来定义下面选项：
Flag
意义
*
输入被忽略
b，h
它们用作格式类型d，i，o，u 和x 的前缀，用这些变量可定义参量是字符指针还是无符号字符指针。如%bu，%bd，%bx。
L
它们被作格式类型d，i，o，u 和x 的前缀，使用这个前缀可定义参量是长指针还是无符号字长指针。如%lu，%ld，%lx。
“type”域定义参量为如下类型：
描述符
类型
输入格式
d
ptr to int
有符号十进制数（16 位）
i
ptr to int
如C 中记号一样，整型值
u
ptr to int
无符号十进制数
o
ptr to int
无符号八进制数
x
ptr to int
无符号十六进制数

f，e，g

ptr to float

浮点数

c

ptr to char

一个字符

s

ptr to string

一个字串

例子：

scanf(“%d%bd%ld”,&i,&c,&l);

scanf(“%f”,&f);

scanf(“%3s,%c”,&string[0],&character);

函数名： sscanf

原型： extern int sscanf(const *s,const char*, …);

功能： sscanf 与scanf 方式相似,但串输入不是通过控制台，而是通过另一个以空结束的指针。

注：sscanf 参量允许的总字节数由C-51 库限制，这是因为8051 处理器结构内存的限制，在SMALL和COMPACT 模式，最大允许15 字节参数（即至多5 个指针，或2 个指针，2 个长整型或1 个字符型）的传递。在LARGE 模式下，最大允许传送40 个字节的参数。

3 STRING.H：串函数

串函数通常将指针串作输入值。一个串就包括2 个或多个字符。串结以空字符表示。在函数memcmp，memcpy，memchr，memccpy，memmove 和memset 中，串长度由调用者明确规定，使这些函数可工作在任何模式下。

函数名： memchr

原型： extern void *memchr(void *s1， char val，int len);

功能： memchr 顺序搜索s1 中的len 个字符找出字符val，成功时返回s1 中指向val 的指针，失败时返回NULL。

函数名： memcmp

原型： extern char memcmp(void *s1， void *s2，int len);

功能： memcmp 逐个字符比较串s1 和s2 的前len 个字符。相等时返回0，如果串s1 大于或小于s2，则相应返回一个正数或负数。

函数名： memcpy

原型： extern void *memcpy(void *dest， void *src，int len);

功能： memcpy 由src 所指内存中拷贝len 个字符到dest 中，返回批向dest 中的最后一个字符的指针。如果src 和dest 发生交迭，则结果是不可预测的。

函数名： memccpy

原型： extern void *memccpy(void *dest， void *src，char val，int len);

功能： memccpy 拷贝src 中len 个字符到dest 中，如果实际拷贝了len 个字符返回NULL。拷贝过程在拷贝完字符val 后停止，此时返回指向dest 中下一个元素的指针。

函数名： memmove

原型： extern void *memmove(void *dest， void *src，int len);

功能： memmove 工作方式与memcpy 相同，但拷贝区可以交迭。

函数名： memset

原型： extern void *memset(void *s， char val，int len);

功能： memset 将val 值填充指针s 中len 个单元。

函数名： strcat

原型： extern char *strcat(char *s1， char *s2);

功能： strcat 将串s2 拷贝到串s1 结尾。它假定s1 定义的地址区足以接受两个串。返回指针指向s1 串的第一字符。

函数名： strncat

原型： extern char *strncat(char *s1， char *s2，int n);

功能： strncat 拷贝串s2 中n 个字符到串s1 结尾。如果s2 比n 短，则只拷贝s2。

函数名： strcmp

原型： extern char strcmp(char *s1， char *s2);

功能： strcmp 比较串s1 和s2，如果相等返回0，如果s1s2 则返回一个正数。

函数名： strncmp

原型： extern char strncmp(char *s1， char *s2，int n);

功能： strncmp 比较串s1 和s2 中前n 个字符，返回值与strncmp 相同。

函数名： strcpy

原型： extern char *strcpy(char *s1， char *s2);

功能： strcpy 将串s2 包括结束符拷贝到s1，返回指向s1 的第一个字符的指针。

函数名： strncpy

原型： extern char *strncpy(char *s1， char *s2，int n);

功能： strncpy 与strcpy 相似，但只拷贝n 个字符。如果s2 长度小于n，则s1 串以‘0’补齐到长度n。

函数名： strlen

原型： extern int strlen(char *s1);

功能： strlen 返回串s1 字符个数（包括结束字符）。

函数名： strchr，strpos

原型： extern char *strchr(char *s1， char c);

extern int strpos（char *s1，char c）；

功能： strchr 搜索s1 串中第一个出现的‘c’字符，如果成功，返回指向该字符的别指针，搜索也包括结束符。搜索一个空字符返回指向空字符的指针而不是空指针。strpos 与strchr 相似，但它返回字符在串中的位置或-1，s1 串的第一个字符位置是0。

函数名： strrchr，strrpos

原型： extern char *strrchr(char *s1， char c);extern int strrpos（char *s1，char c）；

功能： strrchr 搜索s1 串中最后一个出现的‘c’字符，如果成功，返回指向该字符的指针，否则返回NULL。对s1 搜索也返回指向字符的指针而不是空指针。strrpos 与strrchr 相似，但它返回字符在串中的位置或-1。

函数名： strspn，strcspn，strpbrk，strrpbrk

原型： extern int strspn(char *s1， char *set);

extern int strcspn(char *s1，char *set);

extern char *strpbrk(char *s1,char *set);

extern char *strpbrk(char *s1,char *set);

功能： strspn 搜索s1 串中第一个不包含在set 中的字符，返回值是s1 中包含在set 里字符的个数。如果s1 中所有字符都包含在set 里，则返回s1 的长度（包括结束符）。如果s1 是空串，则返回0。strcspn 与strspn 类似，但它搜索的是s1 串中的第一个包含在set 里的字符。strpbrk 与strspn 很相似，但它返回指向搜索到字符的指针，而不是个数，如果未找到，则返回NULL。strrpbrk 与strpbrk 相似，但它返回s1 中指向找到的set 字集中最后一个字符的指针。

4 STDLIB.H：标准函数

函数名： atof

原型： extern double atof(char *s1);

功能： atof 将s1 串转换为浮点值并返回它。输入串必须包含与浮点值规定相符的数。C51 编译器对数据类型float 和double 相同对待。

函数名： atol

原型： extern long atol(char *s1);

功能： atol 将s1 串转换成一个长整型值并返回它。输入串必须包含与长整型值规定相符的数。

函数名： atoi

原型： extern int atoi(char *s1);

功能： atoi 将s1 串转换为整型数并返回它。输入串必须包含与整型数规定相符的数。

5 MATH.H：数学函数

函数名： abs，cabs，fabs，labs

原型： extern int abs(int val);

extern char cabs(char val);

extern float fabs(float val);

extern long labs(long val);

功能： abs 决定了变量val 的绝对值，如果val 为正，则不作改变返回；如果为负，则返回相反数。这四个函数除了变量和返回值的数据不一样外，它们功能相同。

函数名： exp，log，log10

原型： extern float exp(float x);

extern float log(float x);

extern float log10(float x);

功能：exp 返回以e 为底x 的幂，log 返回x 的自然数（e=2.718282），log10 返回x 以10为底的数。

函数名： sqrt

原型： extern float sqrt(float x);

功能： sqrt 返回x 的平方根。

函数名：rand，srand

原型： extern int rand(void);

extern void srand（int n）；

功能： rand 返回一个0 到32767 之间的伪随机数。srand 用来将随机数发生器初始化成一个已知（或期望）值，对rand 的相继调用将产生相同序列的随机数。

函数名： cos，sin，tan

原型： extern float cos(flaot x);

extern float sin(flaot x);

extern flaot tan(flaot x);

功能： cos 返回x 的余弦值。Sin 返回x 的正弦值。tan 返回x 的正切值，所有函数变量范围为-π/2～+π/2，变量必须在±65535 之间，否则会产生一个NaN 错误。

函数名： acos，asin，atan，atan2

原型： extern float acos(float x);

extern float asin(float x);

extern float atan(float x);

extern float atan(float y,float x);

功能： acos 返回x 的反余弦值，asin 返回x 的正弦值，atan 返回x 的反正切值，它们的值域为-π/2～+π/2。atan2 返回x/y 的反正切，其值域为-π～+π。

函数名： cosh，sinh，tanh

原型： extern float cosh(float x);

extern float sinh(float x);

extern float tanh(float x);

功能： cosh 返回x 的双曲余弦值；sinh 返回x 的双曲正弦值；tanh 返回x 的双曲正切值。

函数名： fpsave，fprestore

原型： extern void fpsave(struct FPBUF *p);

extern void fprestore (struct FPBUF *p);

功能： fpsave 保存浮点子程序的状态。fprestore 将浮点子程序的状态恢复为其原始状态，当用中断程序执行浮点运算时这两个函数是有用的。

6 ABSACC.H：绝对地址访问

函数名： CBYTE，DBYTE，PBYTE，XBYTE

原型： #define CBYTE((unsigned char *)0x50000L)

#define DBYTE((unsigned char *)0x40000L)

#define PBYTE((unsigned char *)0x30000L)

#define XBYTE((unsigned char *)0x20000L)

功能： 上述宏定义用来对8051 地址空间作绝对地址访问，因此，可以字节寻址。CBYTE寻址CODE区，DBYTE 寻址DATA 区，PBYTE 寻址XDATA 区（通过MOVX @RO 命令），XBYTE 寻址XDATA 区（通过MOVX @DPTR 命令）。

例：下列指令在外存区访问地址0x1000

xval=XBYTE[0x1000];

XBYTE[0X1000]=20;

通过使用#define 指令，用符号可定义绝对地址，如符号X10 可与XBYTE[0x1000]地址相等：

#define X10 XBYTE[0x1000]。

函数名： CWORD，DWORD，PWORD，XWORD

原型： #define CWORD((unsigned int *)0x50000L)

#define DWORD((unsigned int *)0x40000L)

#define PWORD((unsigned int *)0x30000L)

#define XWORD((unsigned int *)0x20000L)

功能：这些宏与上面相似，只是它们指定的类型为unsigned int。通过灵活的数据类型，所有地址空间都可以访问。

7 INTRINS.H：内部函数

函数名： _crol_，_irol_，_lrol_

原型： unsigned char _crol_(unsigned char val,unsigned char n);

unsigned int _irol_(unsigned int val,unsigned char n);

unsigned int _lrol_(unsigned int val,unsigned char n);

功能：_crol_，_irol_，_lrol_以位形式将val 左移n 位，该函数与8051“RLA”指令相关，上面几个函数不同于参数类型。

例：

```
#include
main()
{
unsigned int y;
y=0x00ff;
y=_irol_(y,4);
}
```

函数名： _cror_，_iror_，_lror_

原型： unsigned char _cror_(unsigned char val,unsigned char n);

unsigned int _iror_(unsigned int val,unsigned char n);

unsigned int _lror_(unsigned int val,unsigned char n);

功能：_cror_，_iror_，_lror_以位形式将val 右移n 位，该函数与8051“RRA”指令相关，上面几个函数不同于参数类型。

例：

```
#include
main()
{
unsigned int y;
y=0x0ff00;
y=_iror_(y,4);
}
```

函数名： _nop_

原型： void _nop_(void);

功能：_nop_产生一个NOP 指令,该函数可用作C 程序的时间比较。C51 编译器在_nop_函数工作期间不产生函数调用，即在程序中直接执行了NOP 指令。

例：

```
P()=1;
_nop_();
P()=0;
```

函数名： _testbit_

原型：bit _testbit_(bit x);

功能：_testbit_产生一个JBC 指令，该函数测试一个位，当置位时返回1，否则返回0。如果该位置为1，则将该位复位为0。8051 的JBC 指令即用作此目的。_testbit_只能用于可直接寻址的位；在表达式中使用是不允许的。

8 STDARG.H：变量参数表

C51 编译器允许再入函数的变量参数（记号为“…”）。头文件STDARG.H 允许处理函数的参数表,在编译时它们的长度和数据类型是未知的。为此，定义了下列宏。

宏名： va_list

功能： 指向参数的指针。

宏名： va_stat(va_list pointer,last_argument)

功能： 初始化指向参数的指针。

宏名： type va_arg(va_list pointer,type)

功能：返回类型为type 的参数。

宏名： va_end(va_list pointer)

功能： 识别表尾的哑宏。

9 SETJMP.H：全程跳转

Setjmp.h 中的函数用作正常的系列数调用和函数结束，它允许从深层函数调用中直接返回。

函数名： setjmp

原型： int setjmp(jmp_buf env);

功能： setjmp 将状态信息存入env 供函数longjmp 使用。当直接调用setjmp 时返回值是0，当由longjmp 调用时返回非零值，setjmp 只能在语句IF 或SWITCH 中调用一次。

函数名：longjmp

原型：longjmp(jmp_bufenv,intval);

功能：longjmp恢复调用setjmp时存在env中的状态。程序继续执行，似乎函数setjmp已被执行过。由setjmp返回的值是在函数longjmp中传送的值val，由setjmp调用的函数中的所有自动变量和未用易失性定义的变量的值都要改变。

10REGxxx.H：访问SFR和SFR-BIT地址

文件REG51.H，REG52.H和REG552.H允许访问8051系列的SFR和SFR-bit的地址，这些文件都包含#include指令，并定义了所需的所有SFR名以寻址8051系列的外围电路地址，对于8051系列中其它一些器件，用户可用文件编辑器容易地产生一个“.h”文件。

下例表明了对8051PORT0和PORT1的访问：

```
#include
main(){
if(p0==0x10)p1=0x50;
}
```

KeilC51库函数原型列表

1.1.CTYPE.H

bitisalnum(charc);

bitisalpha(charc);

bitiscntrl(charc);

bitisdigit(charc);

bitisgraph(charc);

bitislower(charc);

bitisprint(charc);

bitispunct(charc);

bitisspace(charc);

bitisupper(charc);

bitisxdigit(charc);

bittoascii(charc);

bittoint(charc);

chartolower(charc);

char__tolower(charc);

chartoupper(charc);

char__toupper(charc);

2.2.INTRINS.H

unsignedchar_crol_(unsignedcharc,unsignedcharb);

unsignedchar_cror_(unsignedcharc,unsignedcharb);

unsignedchar_chkfloat_(floatual);

unsignedint_irol_(unsignedinti,unsignedcharb);

unsignedint_iror_(unsignedinti,unsignedcharb);

unsignedlong_irol_(unsignedlongl,unsignedcharb);

unsignedlong_iror_(unsignedlongL,unsignedcharb);

void_nop_(void);

bit_testbit_(bitb);

3.3.STDIO.H

chargetchar(void);

char_getkey(void);

char*gets(char*string,intlen);

intprintf(constchar*fmtstr[,argument]…);

charputchar(charc);

intputs(constchar*string);

intscanf(constchar*fmtstr.[,argument]…);

intsprintf(char*buffer,constchar*fmtstr[;argument]);

intsscanf(char*buffer,constchar*fmtstr[,argument]);

charungetchar(charc);

voidvprintf(constchar*fmtstr,char*argptr);

voidvsprintf(char*buffer,constchar*fmtstr,char*argptr);

4.4.STDLIB.H

floatatof(void*string);

intatoi(void*string);

longatol(void*string);

void*calloc(unsignedintnum,unsignedintlen);

voidfree(voidxdata*p);

voidinit_mempool(void*data*p,unsignedintsize);

void*malloc(unsignedintsize);

intrand(void);

void*realloc(voidxdata*p,unsignedintsize);

voidsrand(intseed);

5.5.STRING.H

void*memccpy(void*dest,void*src,charc,intlen);

void*memchr(void*buf,charc,intlen);

charmemcmp(void*buf1,void*buf2,intlen);

void*memcopy(void*dest,void*SRC,intlen);

void*memmove(void*dest,void*src,intlen);

void*memset(void*buf,charc,intlen);

char*strcat(char*dest,char*src);

char*strchr(constchar*string,charc);

charstrcmp(char*string1,char*string2);
char*strcpy(char*dest,char*src);
intstrcspn(char*src,char*set);
intstrlen(char*src);
char*strncat(char8dest,char*src,intlen);
charstrncmp(char*string1,char*string2,intlen);
charstrncpy(char*dest,char*src,intlen);
char*strpbrk(char*string,char*set);
intstrpos(constchar*string,charc);
char*strrchr(constchar*string,charc);
char*strrpbrk(char*string,char*set);
intstrrpos(constchar*string,charc);
intstrspn(char*string,char*set);

C51强大功能及其高效率的重要体现之一在于其丰富的可直接调用的库函数，多使用库函数使程序代码简单，结构清晰，易于调试和维护，下面介绍C51的库函数系统。

本征库函数(intrinsicroutines)和非本征证库函数

C51提供的本征函数是指编译时直接将固定的代码插入当前行，而不是用ACALL和LCALL语句来实现，这样就大大提供了函数访问的效率，而非本征函数则必须由ACALL及LCALL调用。

C51的本征库函数只有9个，数目虽少，但都非常有用，列如下：
crol,_cror_：将char型变量循环向左(右)移动指定位数后返回
iror,_irol_：将int型变量循环向左(右)移动指定位数后返回
lrol,_lror_：将long型变量循环向左(右)移动指定位数后返回
nop：相当于插入NOP
testbit：相当于JBCbitvar测试该位变量并跳转同时清除。
chkfloat：测试并返回源点数状态。
使用时，必须包含#inclucle<intrins.h>一行。
如不说明，下面谈到的库函数均指非本征库函数。

1.专用寄存器include文件

例如8031、8051均为REG51.h其中包括了所有8051的SFR及其位定义，一般系统都必须包括本文件。
2.绝对地址include文件absacc.h
该文件中实际只定义了几个宏，以确定各存储空间的绝对地址。
3.动态内存分配函数，位于stdlib.h中
4.缓冲区处理函数位于“string.h”中
其中包括拷贝比较移动等函数如：

memccpymemchrmemcmpmemcpymemmovememset
这样很方便地对缓冲区进行处理。
5.输入输出流函数，位于“stdio.h”中
流函数通8051的串口或用户定义的I/O口读写数据，缺省为8051串口，如要修改，比如改为LCD显示，可修改lib目录中的getkey.c及putchar.c源文件，然后在库中替换它们即可。__