PICC 库函数说明

PICC 库函数

本章将详细列出 PICC 编译器的库函数。每个函数均从函数名开始，然后按照以下几个标题给出详细解释。

提要：函数的 C 语言定义以及定义函数的头文件。

描述：对函数及其目的进行叙述性描述。

例程：给出一个能说明该函数的应用例子。

数据类型：列出函数中使用的一些特殊的数据类型（如结构体等）的C语言定义。这些数据类型的定义包含在提要标题下列出的头文件中。

参阅：给出相关联的函数。

返回值：如果函数有返回值，则在本标题下将给出返回值的类型和性质，同时还包括错误返回的信息。

1 ABS 函数

1. 提 要

```
#include <stdlib.h>
int abs (int j)
```

2. 描 述

abs( )函数返回变量 j 的绝对值。

3. 例 程

```
#include <stdio.h>
#include <stdlib.h>
void
main (void)
{
int a = -5；
printf("The absolute value of %d is %d\n"，a，abs(a))；
}
```

4. 返回值

j 的绝对值。

2 ACOS函数

1. 提 要

```
#include <math.h>
double acos (double f)
```

2. 描 述

acos( )函数是cos( ) 的反函数。函数参数在[-1，1]区间内，返回值是一个用弧度表示的角度，而且该返回值的余弦值等于函数参数。

3. 例 程

```
#include <math.h>
#include <stdio.h>
```

```
/*以度为单位，打印[-1，1]区间内的反余弦值*/
void
main (void)
{
float i，a；
for(i = -1.0，i < 1.0；i += 0.1) {
a = acos(i)*180.0/3.141592；
printf("acos(%f) = %f degrees\n"，i，a)；
}
}
```

4．参　阅

sin( )，cos( )，tan( )，asin( )，atan( )，atan2( )

5．返回值

返回值是一个用弧度表示的角度，区间是[0，π]。如果函数参数超出区间[-1，1]，则返回值将为 0。

3　ASCTIME函数

1．提　要

```
#include <time.h>
char * asctime (struct tm * t)
```

2．描　述

asctime( )函数通过指针 t 从上struct tm结构体中获得时间，返回描述当前日期和时间的26个字符串，其格式如下：

Sun Sep 16 01:03:52 1973\n\0

值得注意的是,在字符串的末尾有换行符。字符串中的每个字长是固定的。以下例程得到当前时间，通过localtime( )函数将其转换成一个struct tm指针，最后转换成ASCII码并打印出来。其中，time( )函数需要用户提供（详情请参阅time( )函数）。

3．例　程

```
#include <stdio.h>
#include <time.h>
void
main (void)
{
time_t clock；
struct tm * tp；
time(&clock)；
tp = localtime(&clock)；
printf("%s"，asctime(tp))；
}
```

4．参　阅

ctime( )，gmtime( )，localtime( )，time( )

5. 返回值

指向字符串的指针。

注意：由于编译器不提供time( )例行程序，故在本例程中它需要由用户提供。详情请参照time( )函数。

6. 数据类型

```
struct tm {
        int tm_sec;
        int tm_min;
        int tm_hour;
        int tm_mday;
        int tm_mon;
        int tm_year;
        int tm_wday;
        int tm_yday;
        int tm_isdst;
};
```

4 ASIN函数

1. 提 要

```
#include <math.h>
double asin (double f)
```

2. 描 述

asin( )函数是sin( )的反函数。它的函数参数在[-1，1]区间内，返回一个用弧度表示的角度值，而且这个返回值的正弦等于函数参数。

3. 例 程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
float i，a;
for(i = -1.0；i < 1.0 ；i += 0.1) {
a = asin(i)*180.0/3.141592;
printf("asin(%f) = %f degrees\n"，i，a);
}
}
```

4. 参 阅

sin( )，cos( )，tan( )，acos( )，atan( )，atan2( )

5. 返回值

本函数返回一个用弧度表示的角度值，其区间为[-π/2，π/2]。如果函数参数的值超出区间[-1，1]，则函数返回值将为0。

5 ATAN函数

1. 提 要

#include <math.h>

double atan (double x)

2. 描 述

函数返回参数的反正切值。也就是说，本函数将返回一个在区间[−π/2，π/2]的角度e，而且有tan(e)=x（x为函数参数）。

3. 例 程

```
#include <stdio.h>
#include <math.h>
void
main (void)
{
printf("%f\n"，atan(1.5))；
}
```

4. 参 阅

sin( )，cos( )，tan( )，asin( )，acos( )，atan2( )

5. 返回值

返回函数参数的反正切值。

6 ATAN2函数

1. 提 要

#include <math.h>

double atan2 (double y，double x)

2. 描 述

本函数返回y/x的反正切值，并由两个函数参数的符号来决定返回值的象限。

3. 例 程

```
#include <stdio.h>
#include <math.h>
void
main (void)
{
printf("%f\n"，atan2(1.5，1))；
}
```

4. 参 阅

sin( )，cos( )，tan( )，asin( )，acos( )，atan( )

5. 返回值

返回y/x的反正切值（用弧度表示），区间为[−π，π]。如果y和x均为0，将出现定义域错误，并返回0。

7 ATOF函数

1. 提　要

#include <stdlib.h>

double atof (const char * s)

2. 描　述

atof( )函数将扫描由函数参数传递过来的字符串，并跳过字符串开头的空格。然后将一个数的ASCII表达式转换成双精度数。这个数可以用十进制数、浮点数或者科学记数法表示。

3. 例　程

```
#include <stdlib.h>
#include <stdio.h>
void
main (void)
{
char buf[80];
double i;
gets(buf);
i = atof(buf);
printf("Read %s: converted to %f\n", buf, i);
}
```

4. 参　阅

atoi( )，atol( )

5. 返回值

本函数返回一个双精度浮点数。如果字符串中没有发现任何数字，则返回0.0。

8　ATOI函数

1. 提　要

#include <stdlib.h>

int atoi (const char * s)

2. 描　述

atoi( )函数扫描传递过来的字符串，跳过开头的空格并读取其符号;然后将一个十进制数的ASCII表达式转换成整数。

3. 例　程

```
#include <stdlib.h>
#include <stdio.h>
void
main (void)
{
char buf[80];
int i;
gets(buf);
i = atoi(buf);
```

```
printf("Read %s: converted to %d\n"，buf，i)；
}
```

4. 参　阅

xtoi( )，atof( )，atol( )

5. 返回值

返回一个有符号的整数。如果在字符串中没有发现任何数字，则返回0。

9　ATOL函数

1．提　要

```
#include <stdlib.h>
long atol (const char * s)
```

2．描　述

atol( )函数扫描传递过来的字符串，并跳过字符串开头的空格；然后将十进制数的ASCII
表达式转换成长整型。

3．例　程

```
#include <stdlib.h>
#include <stdio.h>
void
main (void)
{
char buf[80]；
long i；
gets(buf)；
i = atol(buf)；
printf("Read %s: converted to %ld\n"，buf，i)；
}
```

4．参　阅

atoi( )，atof( )

5．返回值

返回一个长整型数。如果字符串中没有发现任何数字，返回值为0。

10　CEIL函数

1．提　要

```
#include <math.h>
double ceil (double f)
```

2．描　述

本函数对函数参数 f 取整，取整后的返回值为大于或等于 f 的最小整数。

3．例　程

```
#include <stdio.h>
#include <math.h>
```

```
void
main (void)
{
double j;
scanf("%lf", &j);
printf("The ceiling of %lf is %lf\n", j, ceil(j));
}
```

11 COS函数

1. 提 要

```
#include <math.h>
double cos (double f)
```

2. 描 述

本函数将计算函数参数的余弦值。其中，函数参数用弧度表示。余弦值通过多项式级数近似值展开式算得。

3. 例 程

```
#include <math.h>
#include <stdio.h>
#define C 3.141592/180.0
void
main (void)
{
double i;
for(i = 0; i <= 180.0; i += 10)
printf("sin(%3.0f) = %f, cos = %f\n", i, sin(i*C), cos(i*C));
}
```

4. 参 阅

sin( )，tan( )，asin( )，acos( )，atan( )，atan2( )

5. 返回值

返回一个双精度数，区间为[−1，1]。

12 COSH、SINH、TANH函数

1. 提 要

```
#include <math.h>
double cosh (double f)
double sinh (double f)
double tanh (double f)
```

2. 描 述

这些函数都是cos( )，sin( )和tan( )的双曲函数。

3. 例 程

```
#include <stdio.h>
```

```
#include <math.h>
void
main (void)
{
printf("%f\n"，cosh(1.5))；
printf("%f\n"，sinh(1.5))；
printf("%f\n"，tanh(1.5))；
}
```

4. 返回值

cosh( )函数返回双曲余弦值，sinh( )函数返回双曲正弦值，tanh( )函数返回双曲正切值。

13 CTIME函数

1. 提 要

```
#include <time.h>
char * ctime (time_t * t)
```

2. 描 述

ctime( )函数将函数参数所指的时间转换成字符串，其结构与asctime( )函数所描述的相同，并且精确到秒。以下例程将打印出当前的时间和日期。

3. 例 程

```
#include <stdio.h>
#include <time.h>
void
main (void)
{
time_t clock；
time(&clock)；
printf("%s"，ctime(&clock))；
}
```

4. 参 阅

gmtime( )，localtime( )，asctime( )，time( )

5. 返回值

本函数返回一个指向该字符串的指针。

注意：由于编译器不会提供time( )程序，故它需要由用户给定。详情请参阅time( )函数。

6. 数据类型

```
typedef long time_t
```

14 DI、EI函数

1. 提 要

```
#include <pic.h>
void ei(void)
```

void di(void)

2. 描　述

ei( )和di( )函数分别实现全局中断使能和中断屏蔽，其定义在pic.h头文件中。它们将被扩展为一条内嵌的汇编指令，分别对中断使能位进行置位和清零。

以下例程将说明ei( )函数和di( )函数在访问一个长整型变量时的应用。由于中断服务程序将修改该变量，所以如果访问该变量不按照本例程的结构编程，一旦在访问变量值的连续字期间出现中断，则函数getticks( ) 将返回错误的值。

3. 例　程

```
#include <pic.h>
long count;
void interrupt tick(void)
{
count++;
}
long getticks(void)
{
long val;  /*在访问count变量前禁止中断，保证访问的连续性*/
di( );
val = count;
ei( );
return val;
}
```

15　DIV函数

1. 提　要

```
#include <stdlib.h>
div_t div (int numer，int demon)
```

2. 描　述

div( )函数实现分子除以分母，得到商和余数。

3. 例　程

```
#include <stdlib.h>
#include <stdio.h>
void
main (void)
{
div_t x;
x = div(12345，66);
printf("quotient = %d，remainder = %d\n"，x.quot，x.rem);
}
```

4. 返回值

返回一个包括商和余数的结构体div_t。.

5. 数据类型

```
typedef struct
{
int quot;
int rem;
} div_t;
```

16 EEPROM_READ、EEPROM_WRITE函数

1. 提 要

```
#include <pic.h>
unsigned char eeprom_read (unsigned char addr);
void eeprom_write (unsigned char addr，unsigned char value);
```

2. 描 述

这些函数允许访问片内EEPROM（如果片内有EEPROM）。EEPROM不是可直接寻址的寄存器空间，当需要访问EEPROM时，就需要将一些特定的字节序列加载到EEPROM控制寄存器中。写EEPROM是一个缓慢的过程。故eeprom_write( )函数在写入下一个数据前，会通过查询恰当的寄存器来确保前一个数据已经写入完毕。另外，读EEPROM可以在一个指令周期内完成，所以没有必要查询读操作是否完成。

3. 例 程

```
#include <pic.h>
void
main (void)
{
unsigned char data;
unsigned char address;
address = 0x10;
data = eeprom_read(address);
}
```

注意：如果调用eeprom_write( )函数后需即刻调用eeprom_read( )函数，则必须查询EEPROM寄存器以确保写入完毕。全局中断使能位（GIE）在eeprom_write( )程序中重新恢复（写EEPROM时需要关闭总中断）。而且，本函数不会清EEIF标志位。

17 EVAL_POLY函数

1. 提 要

```
#include <math.h>
double eval_poly (double x，const double * d，int n)
```

2. 描 述

eval_poly( )函数将求解一个多项式的值。这个多项式的系数分别包含在x和数组d中，例如：

$$y = x*x*d2 + x*d1 + d0$$

该多项式的阶数由参数n传递过来。

3. 例　程

```
#include <stdio.h>
#include <math.h>
void
main (void)
{
double x，y；
double d[3] = {1.1，3.5，2.7}；
x = 2.2；
y = eval_poly(x，d，2)；
printf("The polynomial evaluated at %f is %f\n"，x，y)；
}
```

4. 返回值

本函数返回一个双精度数，该数是自变量x对应的多项式值。

18　EXP函数

1. 提　要

```
#include <math.h>
double exp (double f)
```

2. 描　述

exp( )函数返回参数的指数函数值，即e^f（f为函数参数）。

3. 例　程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
double f；
for(f = 0.0；f <= 5；f += 1.0)
printf("e to %1.0f = %f\n"，f，exp(f))；
}
```

4. 参　阅

log( )，log10( )，pow( )

19　FABS函数

1. 提　要

```
#include <math.h>
double fabs (double f)
```

2. 描　述

本函数返回双精度函数参数的绝对值。

3. 例　程

```
#include <stdio.h>
#include <math.h>
void
main (void)
{
printf("%f %f\n"，fabs(1.5)，fabs(-1.5))；
}
```

4. 参　阅

abs( )

20　FLOOR函数

1. 提　要

```
#include <math.h>
double floor (double f)
```

2. 描　述

本函数对函数参数取整，取整后的返回值不大于函数参数f。

3. 例　程

```
#include <stdio.h>
#include <math.h>
void
main (void)
{
printf("%f\n"，floor( 1.5 ))；
printf("%f\n"，floor( -1.5))；
}
```

21　FREXP函数

1. 提　要

```
#include <math.h>
double frexp (double f，int * p)
```

2. 描　述

frexp( )函数将一个浮点数分解成规格化小数和2的整数次幂两部分，整数幂部分存于指针 p 所指的 int 单元中。本函数的返回值x或在区间（0.5，1.0）内，或为 0；而且有 $f=x\times 2^{p}$。如果f为0，则分解出来的两部分均为0。

3. 例　程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
```

```
double f;
int i;
f = frexp(23456.34，&i);
printf("23456.34 = %f * 2^%d\n"，f，i);
}
```

4. 参　阅

ldexp( )

22　GET_CAL_DATA函数

1. 提　要

```
#include <pic.h>
double get_cal_data (const unsigned char * code_ptr)
```

2. 描　述

本函数从PIC 14000标定空间返回一个32位的浮点标定数据。只有利用这个函数才能访问KREF、KBG、BHTHERM和KTC单元（32位浮点参数）。由于FOSC和TWDT均是一个字节长度，故可以直接访问它们。

3. 例　程

```
#include <pic.h>
void
main (void)
{
double x;
unsigned char y;
x = get_cal_data(KREF)；/*获得参考斜率（slope reference ratio）*/
y =TWDT；/*获得WDT溢出时间*/
}
```

4. 返回值

返回定标参数值。

注意：本函数仅用于PIC 14000

23　GMTIME函数

1. 提　要

```
#include <time.h>
struct tm * gmtime (time_t * t)
```

2. 描　述

本函数把指针 t 所指的时间分解,并且存于结构体中，精确度为秒。其中，t 所指的时间必须自1970年1月1日0时0分0秒起。本函数所用的结构体被定义在time.h文件中，可参照本节“数据类型”部分。

3. 例　程

```
#include <stdio.h>
#include <time.h>
```

```
void
main (void)
{
time_t clock；
struct tm * tp；
time(&clock)；
tp = gmtime(&clock)；
printf("It’s %d in London\n"，tp->tm_year+1900)；
}
```

4. 参　阅

ctime( )，asctime( )，time( )，localtime( )

5. 返回值

返回tm类型的结构体。

注意：由于编译器不会提供time( )程序，故它需要由用户给定。详情请参阅time( )函数。

6. 数据类型

```
typedef long time_t；
struct tm {
int tm_sec；
int tm_min；
int tm_hour；
int tm_mday；
int tm_mon；
int tm_year；
int tm_wday；
int tm_yday；
int tm_isdst；
}；
```

24　ISALNUM，ISALPHA，ISDIGIT，ISLOWER 等函数

1. 提　要

```
#include <ctype.h>
int isalnum (char c)
int isalpha (char c)
int isascii (char c)
int iscntrl (char c)
int isdigit (char c)
int islower (char c)
int isprint (char c)
int isgraph (char c)
int ispunct (char c)
int isspace (char c)
```

int isupper (char c)

int isxdigit(char c)

2. 描　述

以上函数都被定义在ctype.h文件中。它们将测试给定的字符，看该字符是否为已知的几组字符中的成员。

isalnum (c)	c在0～9、a～z或者A～Z范围内；
isalpha (c)	c在A～Z或a～z范围内；
isascii (c)	c为7位ASCII字符；
iscntrl (c)	c为控制字符；
isdigit (c)	c为十进制阿拉伯数字；
islower (c)	c在a～z范围内；
isprint (c)	c为打印字符；
isgraph (c)	c为非空格可打印字符；
ispunct (c)	c不是字母数字混合的；
isspace (c)	c是空格键、TAB键或换行符；
isupper (c)	c在A～Z范围内；
isxdigit (c)	c在0～9、a～f或A～F范围内。

3．例　程

```
#include <ctype.h>
#include <stdio.h>
void
main (void)
{
char buf[80];
int i;
gets(buf);
i = 0;
while(isalnum(buf[i]))
i++;
buf[i] = 0;
printf("'%s' is the word\n", buf);
}
```

4．参　阅

toupper( )，tolower( )，toascii( )

25　KBHIT函数

1. 提　要

```
#include <conio.h>
bit kbhit (void)
```

2. 描　述

如果键盘上的字符被按下，函数返回1；否则返回0。通常，该字符可通过getch( )函

数读取。

3. 例　程

```
#include <conio.h>
void
main (void)
{
int i;
while(!kbhit()) {
cputs("I'm waiting..");
for(i = 0 ;  i != 1000 ;  i++)
continue;
}
}
```

4. 参　阅

getch( )，getche( )

5. 返回值

如果有键被按下，函数将返回1；否则返回0。此外，返回值为1位。

注意：程序的主体需由用户实现，其主要框架可以从sources目录下直接获得。

26　LDEXP函数

1. 提　要

```
#include <math.h>
double ldexp (double f，int i)
```

2. 描　述

ldexp( )函数是frexp( )的反函数。它先进行浮点数 f 的指数部分与整数 i 的求和运算，然后返回合成结果。

3. 例　程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
double f;
f = ldexp(1.0，10);
printf("1.0 * 2^10 = %f\n"，f);
}
```

4. 参　阅

frexp( )

5. 返回值

本函数返回浮点数 f 指数部分加上整数 i 后得到的新浮点数。

27 LDIV函数

1. 提 要

#include <stdlib.h>

ldiv_t ldiv (long number，long denom)

2. 描 述

ldiv( )函数实现分子除以分母，得到商和余数。商的符号与精确商的符号一致，绝对值是一个小于精确商绝对值的最大整数。

Ldiv( )函数与div( )函数类似；不同点在于，前者的函数参数和返回值（结构体ldiv_t）的成员都是长整型数据。

3. 例 程

```
#include <stdlib.h>
#include <stdio.h>
void
main (void)
{
ldiv_t lt;
lt = ldiv(1234567，12345);
printf("Quotient = %ld，remainder = %ld\n"，lt.quot，lt.rem);
}
```

4. 参 阅

div( )

5. 返回值

返回值是结构体ldiv_t。

6. 数据结构

```
typedef struct {
long quot；  /*商*/
long rem；  /*余数*/
} ldiv_t;
```

28 LOCALTIME函数

1. 提 要

#include <time.h>

struct tm * localtime (time_t * t)

2. 描 述

本函数把指针t所指的时间分解并且存于结构体中，精确度为秒。其中，t所指的时间必须自1970年1月1日0时0分0秒起，所用的结构体被定义在time.h文件中。localtime( )函数需要考虑全局整型变量time_zone中的内容，因为它包含有本地时区位于格林威治以西的时区数值。由于在MS-DOS环境下无法预先确定这个值，所以，在缺省的条件下，localtime( )函数的返回值将与gmtime( )的相同。

3. 例 程

#include <stdio.h>

```
#include <time.h>
char * wday[] = {
"Sunday"，"Monday"，"Tuesday"，"Wednesday"，
"Thursday"，"Friday"，"Saturday"
};
void
main (void)
{
time_t clock；
struct tm * tp；
time(&clock)；
tp = localtime(&clock)；
printf("Today is %s\n"，wday[tp->tm_wday])；
}
```

4. 参　阅

ctime( )，asctime( )，time( )

5. 返回值

本函数返回tm结构体型数据。

注意：由于编译器不会提供time( )程序，故它需要由用户给定。详情请参阅time( )函数。

6. 数据结构

```
typedef long time_t；
struct tm {
int tm_sec；
int tm_min；
int tm_hour；
int tm_mday；
int tm_mon；
int tm_year；
int tm_wday；
int tm_yday；
int tm_isdst；
};
```

29　LOG、LOG10函数

1. 提　要

```
#include <math.h>
double log (double f)
double log10 (double f)
```

2. 描　述

log( )函数返回f的自然对数值。log10( )函数返回f以10为底的对数值。

3. 例　程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
double f;
for(f = 1.0； f <= 10.0； f += 1.0)
printf("log(%1.0f) = %f\n"， f， log(f))；
}
```

4. 参　阅

exp( )，pow( )

5. 返回值

如果函数参数为负，返回值为0。

30　MEMCHR函数

1. 提　要

```
#include <string.h>
/*  初级和中级系列单片机  */
const void * memchr (const void * block，int val，size_t length)
/*  高级系列单片机*/
void * memchr (const void * block，int val，size_t length)
```

2. 描　述

memchr( )函数与strchr( )函数在功能上类似;但前者没有在字符串中寻找null（空）中止字符的功能。memchr( )函数实现在一段规定了长度的内存区域中寻找特定的字节。它的函数参数包括指向被寻内存区域的指针、被寻字节的值和被寻内存区域的长度。函数将返回一个指针，该指针指向被寻内存区域中被寻字节首次出现的单元。

3. 例　程

```
#include <string.h>
#include <stdio.h>
unsigned int ary[ ] = {1，5，0x6789，0x23}；
void
main (void)
{
char * cp；
cp = memchr(ary，0x89，sizeof ary)；
if(!cp)
printf("not found\n")；
else
printf("Found at offset %u\n"，cp - (char *)ary)；
}
```

4. 参　阅

strchr( )

5. 返回值

函数返回指针。该指针指向被寻内存区域中被寻字节首次出现的单元；否则返回NULL。

31 MEMCMP函数

1. 提 要

#include <string.h>

int memcmp (const void * s1, const void * s2, size_t n)

2. 描 述

memcmp()函数的功能是比较两块长度为 n 的内存中变量的大小，类似strncmp()函数返回一个有符号数。与strncmp()函数不同的是，memcmp()函数没有空格结束符。ASCII码字符顺序被用来比较；但如果内存块中包含非ASCII码字符，则返回值不确定。测试是否相等总是可靠的。

3. 例 程

```
#include <stdio.h>
#include <string.h>
void
main (void)
{
int buf[10], cow[10], i;
buf[0] = 1;
buf[2] = 4;
cow[0] = 1;
cow[2] = 5;
buf[1] = 3;
cow[1] = 3;
i = memcmp(buf, cow, 3*sizeof(int));
if(i < 0)
printf("less than\n");
else if(i > 0)
printf("Greater than\n");
else
printf("Equal\n");
}
```

4. 参 阅

strncpy()， strncmp()， strchr()， memset()， memchr()

5. 返回值

当内存块变量 s1 分别小于、等于或大于内存块 s2 变量时，函数返回值分别为-1，0或 1。

32 MEMCPY函数

1. 提 要

```
#include <string.h>
/* 低级或中级系列单片机 */
void * memcpy (void * d, const void * s, size_t n)
/* 高级系列单片机 */
far void * memcpy (far void * d, const void * s, size_t n)
```

2. 描 述

memcpy()函数的功能是，将指针s指向的、内存开始的 n 个字节复制到指针d指向的、内存开始的单元。复制重叠区的结果不确定。与strcpy()函数不同的是，memcpy()复制的是一定数量的字节，而不是复制所有结束符前的数据。

3. 例 程

```
#include <string.h>
#include <stdio.h>
void
main (void)
{
char buf[80];
memset(buf, 0, sizeof buf);
memcpy(buf, "a partial string", 10);
printf("buf = '%s'\n", buf);
}
```

4. 参 阅

strncpy()， strncmp()， strchr()， memset()

5. 返回值

memcpy()函数返回值为函数的第一个参数。

33 MEMMOVE函数

1. 提 要

```
#include <string.h>
/* 低级或中级系列单片机 */
void * memmove (void * s1, const void * s2, size_t n)
/* 高级系列单片机 */
far void * memmove (far void * s1, const void * s2, size_t n)
```

2. 描 述

memmove()函数与memcpy()函数相似，但memmove()函数能对重叠区进行准确的复制。也就是说，它可以适当向前或向后，正确地从一个块复制到另一个块，并将它覆盖。

3. 参 阅

strncpy(), strncmp(), strchr(), memcpy()

4. 返回值

memmove()函数同样返回它的第一个参数。

34 MEMSET函数

1. 提 要

```
#include <string.h>
      /* 低级和中级系列的单片机 */
void * memset (void * s, int c, size_t n)
      /* 高级系列单片机 */
far void * memset (far void * s, int c, size_t n)
```

2. 描 述

memset()函数将字节c存储到指针s指向的，内存开始的n个内存字节。

3. 例 程

```
#include <string.h>
#include <stdio.h>
void
main (void)
{
char abuf[20];
strcpy(abuf, "This is a string");
memset(abuf, 'x', 5);
printf("buf = '%s'\n", abuf);
}
```

4. 参 阅

strncpy(), strncmp(), strchr(), memcpy(), memchr()

35 MODF函数

1. 提 要

```
#include <math.h>
double modf (double value, double * iptr)
```

2. 描 述

modf()函数将参数value分为整数和小数2部分，每部分都和value的符号相同。例如，-3.17将被分为整数部分（-3）和小数部分（-0.17）。其中整数部分以双精度数据类型存储在指针iptr指向的单元中。

3. 例 程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
double i_val, f_val;
f_val = modf( -3.17, &i_val);
}
```

4. 返回值

函数返回值为value的带符号小数部分。

36 PERSIST_CHECK, PERSIST_VALIDATE函数

1. 提 要

```
#include <sys.h>
int persist_check (int flag)
void persist_validate (void)
```

2. 描 述

persist_check()函数要用到非可变（non-volatile）的RAM变量，这些变量在定义时被加上限定词persistent。在测试NVRAM（非可变RAM）区域时，先调用persist_validate()函数，并用到一个存储在隐藏变量中的虚拟数据，且由persist_validate()函数计算得到一个测试结果。如果虚拟数据和测试结果都正确，则返回值为真（非零）。如果都不正确，则返回零。在这种情况下，函数返回零并且重新检测NVRAM区域（通过调用persist_validate()函数）。函数被执行的条件是标志变量变量不为0。persist_validate()函数应在每次转换为永久变量之后调用。它将重新建立虚拟数据和计算测试结果。

3. 例 程

```
#include <sys.h>
#include <stdio.h>
persistent long reset_count;
void
main (void)
{
if(!persist_check(1))
printf("Reset count invalid - zeroed\n");
else
printf("Reset number %ld\n", reset_count);
reset_count++; /* update count */
persist_validate(); /* and checksum */
for(;;)
continue; /* sleep until next reset */
}
```

4. 返回值

如果NVRAM区域无效，则返回值为假（零）；如果NVRAM区域有效，则返回值为真（非零）。

37 POW函数

1. 提 要

```
#include <math.h>
double pow (double f, double p)
```

2. 描 述

pow()函数表示第一个参数f的p次幂。

3. 例　程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
double f;
for(f = 1.0 ; f <= 10.0 ; f += 1.0)
printf("pow(2, %1.0f) = %f\n", f, pow(2, f));
}
```

4. 参　阅

log(), log10(), exp()

5. 返回值

返回值为 f 的 p 次幂。

38　PRINTF函数

1. 提　要

```
#include <stdio.h>
unsigned char printf (const char * fmt, ...)
```

2. 描　述

printf()函数是一个格式输出子程序，其运行的基础是标准输出（staout）。它有对应的程序形成字符缓冲区（sprintf()函数）。printf()函数以格式字符串、一系列0及其它作为参数。格式字符串都转换为一定的格式，每一规格化都用来输出变量表。

转换格式的形式为%m.nc。其中％表示格式，m表示选择的字符宽度，n表示选择的精度，c为一个字母表示规格类型。字符宽度和精度只适于中级和高级系列单片机，并且精度只对格式％s有效。

如果指针变量为十进制常数，例如格式为%*d时，则一个整型数将从表中被取出，提供给指针变量。对低级系列单片机而言，有下列转换格式：

o x X u d　即分别为整型格式——八进制、十六进制、十六进制、十进制和十进制。其中d为有符号十进制数，其它为无符号。其精度值为被输出数的总的位数，也可以强制在前面加0。例如%8.4x将产生一8位十六进制数，其中前4位为0，后为十六进制数。X输出的十六进制数中，字母为A～F，x输出的十六进制中字母为a～f。当格式发生变化时，八进制格式前要加0，十六进制格式的前面要加0x或0X。

S　打印一个字符串——函数参数值被认为是字符型指针。最多从字符串中取n个字符打印，字符宽度为m。

C　函数参数被认为一个单字节字符并可自由打印。

任何其它有格式规定的字符将被输出。那么％％将产生一个百分号。

对中级和高级系列单片机而言，转换格式在低级系列单片机的基础上再加上：

l　长整型格式——在整型格式前加上关键字母l即表示长整型变量。

f　浮点格式——总的宽度为m，小数点后的位数为n。如果n没有写出，则默认为6。如果精度为0，则小数点被省略，除非精度已预先定义。

3. 例　程

printf("Total = %4d%%", 23)

输出为'Total = 23%'

printf("Size is %lx" , size)

这里size为长整型十六进制变量。注意当使用%s时，精度只对中级和高级系列单片机有效。

printf("Name = %.8s", "a1234567890")

输出为 'Name = a1234567'

字符变量宽度只对中级和高级系列单片机有效。

printf("xx%*d", 3, 4)

输出为 'xx 4'

```
/* vprintf 例程 */
#include <stdio.h>
int
error (char * s, ...)
{
va_list ap;
va_start(ap, s);
printf("Error: ");
vprintf(s, ap);
putchar('\n');
va_end(ap);
}
void
main (void)
{
int i;
i = 3;
error("testing 1 2 %d", i);
}
```

参见sprintf()函数。

4. 返回值

printf()将返回的字符值写到标准输出口。注意返回值为字符型，而不是整形。

注意：printf函数的部分特征只对中级和高级系列单片机有效。详见描述部分。输出浮点数要求浮点数不大于最大长整型变量。为了使用长整型变量或浮点数格式必须将适当的函数库包含进来。参见有关PICC –L的描述以及有关HPDPIC长整型格式在printf的菜单选项。

39　RAND函数

1. 提　要

```
#include <stdlib.h>
int rand (void)
```

2. 描　述

rand()函数用来产生一个随机数数据。它返回一个0～32767的整数，并且这个整数在每次被调用后，以随机数据形式出现。这一运算规则将产生一个从同一起点开始的确定顺序。起点通过调用srand()函数获得。下面的例程说明了每次通过调用time()函数获得不同的起点。

3. 例　程

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
void
main (void)
{
time_t toc;
int i;
time(&toc);
srand((int)toc);
for(i = 0 ; i != 10 ; i++)
printf("%d\t", rand());
putchar('\n');
}
```

参见srand()函数。

注意：例程中需要用户自己提供time()函数，因为它不能由汇编器产生。更详细的情况参见time()函数。

40　SIN函数

1. 提　要

```
#include <math.h>
double sin (double f)
```

2. 描　述

这个函数返回参数的正弦值。

3. 例　程n

```
#include <math.h>
#include <stdio.h>
#define C 3.141592/180.0
void
main (void)
{
double i;
for(i = 0 ; i <= 180.0 ; i += 10)
printf("sin(%3.0f) = %f, cos = %f\n", i, sin(i*C), cos(i*C));
}
```

4. 参　阅

cos(), tan(), asin(), acos(), atan(), atan2()

5. 返回值

返回值为参数f的正弦值。

41　SPRINTF函数

1. 提　要

```
#include <stdio.h>
/*  中级和低级系列单片机  */
unsigned char sprintf (char *buf, const char * fmt, ...)
/*  高级系列单片机  */
unsigned char sprintf (far char *buf, const char * fmt, ...)
```

2. 描　述

sprintf()函数和 printf()函数操作基本相同; 只是输出在不同的输出终端，所有的字符被放到 buf 缓冲器。字符串带有空格结束符，buf 缓冲器中的数据被返回。

参见 printf()函数。

3. 返回值

sprintf()函数的返回值为被放入缓冲器中的数据。注意，返回值为字符型而非整型。

注意：对高级单片机而言，缓冲器是通过长指针访问的。

42　SQRT函数

1. 提　要

```
#include <math.h>
double sqrt (double f)
```

2. 描　述

sqrt()函数利用牛顿法得到参数的近似平方根。

3. 例　程

```
#include <math.h>
#include <stdio.h>
void
main (void)
{
double i;
for(i = 0 ; i <= 20.0 ; i += 1.0)
printf("square root of %.1f = %f\n", i, sqrt(i));
}
```

参见 exp()函数。

4. 返回值

返回值为参数的平方根。

注意：如果参数为负则出现错误。

43　SRAND函数

1. 提　要

```
#include <stdlib.h>
void srand (unsigned int seed)
```

2. 描　述

srand()函数是在调用 rand()函数时被用来初始化随机数据发生器的。它为 rand()函数产生不同起点虚拟数据顺序提供一个机制。在 z80 上，随机数据最好从新的寄存器获得。否则，控制台的响应时间或系统时间将充当这一数据。

3. 例　程

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
void
main (void)
{
time_t toc;
int i;
time(&toc);
srand((int)toc);
for(i = 0 ; i != 10 ; i++)
printf("%d\t", rand());
putchar('\n');
}
```

参见 rand()函数。

44　STRCAT函数

1. 提　要

```
#include <string.h>
/*  中级和低级系列单片机  */
char * strcat (char * s1, const char * s2)
/*  高级系列单片机  */
far char * strcat (far char * s1, const char * s2)
```

2. 描　述

这个函数将字符串 s2 连接到字符串 s1 的后面。新的字符串以空格作为结束符。指针型参数 s1 指向的字符数组必须保证大于结果字符串。

3. 例　程

```
#include <string.h>
#include <stdio.h>
void
main (void)
{
char buffer[256];
```

```
char * s1, * s2;
strcpy(buffer, "Start of line");
s1 = buffer;
s2 = " ... end of line";
strcat(s1, s2);
printf("Length = %d\n", strlen(buffer));
printf("string = \"%s\"\n", buffer);
}
```

4. 参　阅

strcpy(), strcmp(), strncat(), strlen()

5. 返回值

即为字符串 s1。

45　STRCHR, STRICHR函数

1. 提　要

```
#include <string.h>
/*  中级和低级系列单片机  */
const char * strchr (const char * s, int c)
const char * strichr (const char * s, int c)
/*  高级系列单片机  */
char * strchr (const char * s, int c)
char * strichr (const char * s, int c)
```

2. 描　述

strchr()函数查找字符串 s 中是否出现字符变量 c。如果找到了，则字符指针被返回；否则返回 0。Strichr()函数与 strchr()函数作用相同。

3. 例　程

```
#include <strings.h>
#include <stdio.h>
void
main (void)
{
static char temp[] = "Here it is...";
char c = 's';
if(strchr(temp, c))
printf("Character %c was found in string\n", c);
else
printf("No character was found in string");
}
```

4. 参　阅

strrchr(), strlen(), strcmp()

5. 返回值

如果找到，则返回第一个字符的指针；否则返回 0。

注意：函数对字符使用整型参数，只有低 8 位有效。

46 STRCMP, STRICMP函数

1. 提 要

```
#include <string.h>
int strcmp (const char * s1, const char * s2)
int stricmp (const char * s1, const char * s2)
```

2. 描 述

strcmp()函数用来比较 2 个字符串的大小。字符串带有空格结束符，根据字符串 s1 是否小于、等于或大于字符串 s2，返回一个有符号整数。比较是根据 ASCII 字母的顺序表进行的。Stricmp()函数功能和 strcmp()函数完全一样。

3. 例 程

```
#include <string.h>
#include <stdio.h>
void
main (void)
{
int i;
if((i = strcmp("ABC", "ABc")) < 0)
printf("ABC is less than ABc\n");
else if(i > 0)
printf("ABC is greater than ABc\n");
else
printf("ABC is equal to ABc\n");
}
```

4. 参 阅

strlen(), strncmp(), strcpy(), strcat()

5. 返回值

返回一个有符号整数.

注意：其它的 C 应用程序也可以采用不同的字母顺序表。返回值为正、零或负，也就是说不一定是-1 或 1。

47 STRCPY函数

1. 提 要

```
#include <string.h>
/* 低级和中级系列单片机 */
char * strcpy (char * s1, const char * s2)
/* 高级系列单片机 */
far char * strcpy (far char * s1, const char * s2)
```

2. 描 述

这个函数将以空格键结束的字符串 s2 拷贝到 s1 指向的字符数组。目的数组必须足够大，以容纳包括空格在内的字符串 s2。

3. 例　程

```
#include <string.h>
#include <stdio.h>
void
main (void)
{
char buffer[256];
char * s1, * s2;
strcpy(buffer, "Start of line");
s1 = buffer;
s2 = " ... end of line";
strcat(s1, s2);
printf("Length = %d\n", strlen(buffer));
printf("string = \"%s\"\n", buffer);
}
```

4. 参　阅

strncpy(), strlen(), strcat(), strlen()

5. 返回值

目的数组被返回。

48　STRCSPN函数

1. 提　要

```
#include <string.h>
size_t strcspn (const char * s1, const char * s2)
```

2. 描　述

strcspn()函数用于取得在字符串常数 s1 中有而字符串常数 s2 中没有的字符的长度。

3. 例　程

```
#include <stdio.h>
#include <string.h>
void
main (void)
{
static char set[] = "xyz";
printf("%d\n", strcspn( "abcdevwxyz", set));
printf("%d\n", strcspn( "xxxbcadefs", set));
printf("%d\n", strcspn( "1234567890", set));
}
```

参见 strspn()函数。

4. 返回值

为剩余部分的长度。

49 STRLEN函数

1. 提 要

```
#include <string.h>
size_t strlen (const char * s)
```

2. 描 述

strlen()用来测量字符串 s1 的长度，不包括空格结束符。

3. 例 程

```
#include <string.h>
#include <stdio.h>
void
main (void)
{
char buffer[256];
char * s1, * s2;
strcpy(buffer, "Start of line");
s1 = buffer;
s2 = " ... end of line";
strcat(s1, s2);
printf("Length = %d\n", strlen(buffer));
printf("string = \"%s\"\n", buffer);
}
```

4. 返回值

即为不包括结束符在内的字符长度。

50 STRNCAT函数

1. 提 要

```
#include <string.h>
/* 低级和中级系列单片机 */
char * strncat (char * s1, const char * s2, size_t n)
/* 高级系列单片机 */
far char * strncat (far char * s1, const char * s2, size_t n)
```

2. 描 述

函数将字符串 s2 连接到字符串 s1 的尾端。最多只有 n 个字符被拷贝，结果包含空格结束符。指针 s1 指向的字符数组应足够大, 以容纳结果字符串。

3. 例 程

```
#include <string.h>
#include <stdio.h>
void
main (void)
```

```
{
char buffer[256];
char * s1, * s2;
strcpy(buffer, "Start of line");
s1 = buffer;
s2 = " ... end of line";
strncat(s1, s2, 5);
printf("Length = %d\n", strlen(buffer));
printf("string = \"%s\"\n", buffer);
}
```

4. 参　阅

strcpy(), strcmp(), strcat(), strlen()

5. 返回值

即为字符串 s1。

51　STRNCMP, STRNICMP函数

1. 提　要

```
#include <string.h>
int strncmp (const char * s1, const char * s2, size_t n)
int strnicmp (const char * s1, const char * s2, size_t n)
```

2. 描　述

strncmp()函数用来比较两个带有空格结束符的字符串的大小，最多比较n个字符。根据字符串s1是否小于、等于或大于字符串s2，返回一个有符号数。比较是根据ASCII字母顺序进行的。Strnicmp()函数与之相同。

3. 例　程

```
#include <stdio.h>
#include <string.h>
void
main (void)
{
int i;
i = strcmp("abcxyz", "abcxyz");
if(i == 0)
printf("Both strings are equal\n");
else if(i > 0)
printf("String 2 less than string 1\n");
else
printf("String 2 is greater than string 1\n");
}
```

4. 参　阅

strlen(), strcmp(), strcpy(), strcat()

5. 返回值

有符号整数。

注意：其它C应用函数可以采用不同的字母顺序。返回值为负、零或正，并不一定是-1或1。

52 STRNCPY函数

1. 提 要

```
#include <string.h>
/* 低级和中级系列单片机 */
char * strncpy (char * s1, const char * s2, size_t n)
/* 高级系列单片机 */
far char * strncpy (far char * s1, const char * s2, size_t n)
```

2. 描 述

这个函数将带结束符的字符串s2拷贝到字符指针s1指向的字符数组。最多有n个字符被拷贝。如果s2的长度大于n，则结果中不包含结束符。目的数组必须足够大，以容纳包括结束符在内的新字符串。

3. 例 程

```
#include <string.h>
#include <stdio.h>
void
main (void)
{
char buffer[256];
char * s1, * s2;
strncpy(buffer, "Start of line", 6);
s1 = buffer;
s2 = " ... end of line";
strcat(s1, s2);
printf("Length = %d\n", strlen(buffer));
printf("string = \"%s\"\n", buffer);
}
```

4. 参 阅

strcpy(), strcat(), strlen(), strcmp()

5. 返回值

指针s1指向的目的缓冲区。

53 STRPBRK函数

1. 提 要

```
#include <string.h>
/* 低级和中级系列单片机 */
const char * strpbrk (const char * s1, const char * s2)
```

```
/* 高级系列单片机 */
char * strpbrk (const char * s1, const char * s2)
```

2. 描　述

strpbrk()函数查找字符串 s1 是否包含字符串 s2 的字符。如果包含，则返回被找到的第一个字符的指针；否则不返回任何值。

3. 例　程

```
#include <stdio.h>
#include <string.h>
void
main (void)
{
char * str = "This is a string.";
while(str != NULL) {
printf( "%s\n", str );
str = strpbrk( str+1, "aeiou" );
}
}
```

4. 返回值

第一个匹配的字符，否则返回值为空。

54　STRRCHR, STRRICHR函数

1. 提　要

```
#include <string.h>
/* 中级和低级系列单片机 */
const char * strrchr (char * s, int c)
const char * strrichr (char * s, int c)
/* 高级系列单片机 */
char * strrchr (char * s, int c)
char * strrichr (char * s, int c)
```

2. 描　述

strrchr()函数和 strchr()函数相似；但它从字符串的尾端开始查找，也就是说，其返回值为字符 c 最后一次在字符串中出现时的指针。如果没出现，则返回值为空。strrichr()函数和 strrchr()函数完全一样。

3. 例　程

```
#include <stdio.h>
#include <string.h>
void
main (void)
{
char * str = "This is a string.";
while(str != NULL) {
```

```
printf( "%s\n", str );
str = strrchr( str+1, 's');
}
}
```

4. 参　阅

strchr(), strlen(), strcmp(), strcpy(), strcat()

5. 返回值

字符指针，或者返回值为空。

55　STRSPN函数

1. 提　要

```
#include <string.h>
size_t strspn (const char * s1, const char * s2)
```

2. 描　述

strspn()函数返回字符串s1中包含的、完全由字符串s2组成的字符的长度。

3. 例　程

```
#include <stdio.h>
#include <string.h>
void
main (void)
{
printf("%d\n", strspn("This is a string", "This"));
printf("%d\n", strspn("This is a string", "this"));
}
```

参见strcspn()函数。

4. 返回值

部分长度。

56　STRSTR, STRISTR函数

1. 提　要

```
#include <string.h>
/*中级和低级系列单片机 */
const char * strstr (const char * s1, const char * s2)
const char * stristr (const char * s1, const char * s2)
/* 高级系列单片机 */
char * strstr (const char * s1, const char * s2)
char * stristr (const char * s1, const char * s2)
```

2. 描　述

strstr()函数返回字符数组 s1 中第一次出现字符数组 s2 的指针位置。stristr()与之一样。

3. 例　程

```
#include <stdio.h>
```

```
#include <string.h>
void
main (void)
{
printf("%d\n", strstr("This is a string", "str"));
}
```

4. 返回值

字符指针。如果每有字符串被找到，则返回为空。

57 STRTOK函数

1. 提 要

```
#include <string.h>
/*中级和低级系列单片机 */
char * strtok (char * s1, const char * s2)
/*高级系列单片机 */
far char * strtok (far char * s1, const char * s2)
```

2. 描 述

多次调用 strtok()函数可以将字符串 s1 分为几个独立的部分。s1 中包含 0 或者其它一些包含在字符串 s2 中的字符。这个调用返回一个指向分隔符的第一个字符的指针。如果不存在分隔符则返回为空。分隔符将被空格所覆盖，从而使目前的分隔标记不再起作用。

调用 strtok()函数之后，应使指针 s1 为空。这样，将从后向前查找，又返回分隔符中第一个字符的指针。如果没找到，则返回为空。

3. 例 程

```
#include <stdio.h>
#include <string.h>
void
main (void)
{
char * ptr;
char * buf = "This is a string of words.";
char * sep_tok = ".,?! ";
ptr = strtok(buf, sep_tok);
while(ptr != NULL) {
printf("%s\n", ptr);
ptr = strtok(NULL, sep_tok);
}
}
```

4. 返回值

分隔符中的第一个字符的指针，或者返回为空。

注意：每次调用函数时, 分隔字符串 s2 可以不一样。

58 TAN函数

1. 提 要

```
#include <math.h>
double tan (double f)
```

2. 描 述

tan()函数用来计算参数 f 的正切值。

3. 例 程

```
#include <math.h>
#include <stdio.h>
#define C 3.141592/180.0
void
main (void)
{
double i;
for(i = 0 ; i <= 180.0 ; i += 10)
printf("tan(%3.0f) = %f\n", i, tan(i*C));
}
```

4. 参 阅

sin(), cos(), asin(), acos(), atan(), atan2()

5. 返回值

f 的正切值。

59 TIME函数

1. 提 要

```
#include <time.h>
time_t time (time_t * t)
```

2. 描 述

函数需要目标系统提供当前时间，函数没有给出。这个函数需由用户实现。在运行时，函数以秒为单位返回当前时间。当前时间从 1970 年 1 月 1 日 0 点 0 分 0 秒开始有效。如果参数 t 不为空，那么这个值同样被保存到 t 所指的内存单元。

3. 例 程

```
#include <stdio.h>
#include <time.h>
void
main (void)
{
time_t clock;
time(&clock);
printf("%s", ctime(&clock));
}
```

4. 参 阅

ctime(), gmtime(), localtime(), asctime()

返回值：被执行的函数将返回从 1970 年 1 月 1 日 0 点 0 分 0 秒开始的精确到秒的当前时间。

注意：time()函数没有被提供，用户必须采用前面提到的规范来执行这一程序。

60 TOLOWER, TOUPPER, TOASCII函数

1. 提 要

```
#include <ctype.h>
char toupper (int c)
char tolower (int c)
char toascii (int c)
```

2. 描 述

toupper()函数将小写字母转换为大写字母；而 tolower()则与之相反；toascii()用来保证得到一个 0～0177 之间的结果。如果参数不为字母表中的字母，则 toupper()函数和 tolower()函数都返回它们原来的参数值。

3. 例 程

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void
main (void)
{
char * array1 = "aBcDE";
int i;
for(i=0;i < strlen(array1); ++i) {
printf("%c", tolower(array1[i]));
}
printf("\n");
}
```

4. 参 阅

islower(), isupper(), isascii()等。

61 VA_START, VA_ARG, VA_END函数

1. 提 要

```
#include <stdarg.h>
void va_start (va_list ap, parmN)
type va_arg (ap, type)
void va_end (va_list ap)
```

2. 描 述

这些函数的宏提供一个方便的路径来传递函数参数。函数定义时，函数参数用省略号替代。这里函数参数的个数及类型在汇编时并不知道。

函数最右端的参数（用parmN表示），在宏汇编中起着非常重要的作用，因为它是得到更多参数的开始。函数可以取不同数量的参数，不同类型的va_list（变量表）应事先定义，然后激活带有名为parmN的一系列参数的宏va_start()。将变量初始化，从而允许调用宏va_arg()，得到其它的参数。

每次调用va_arg()需有两个参数；一个在前面已定义，另一个参数也需要说明其类型。注意所有的参数都将被自动加宽为整型、无符号整型和双精度型。例如，如果一个参数为字符型，则字符型自动转换为整型，函数调用的形式相当于va_arg(ap,int)。

下面用例子说明带有一个整型变量和一些其它变量作为参数的函数。在这个例子中，函数将得到一个字符型指针；但要注意编译器并不知道，程序员对参数正确性负责。

3．例　程

```
#include <stdio.h>
#include <stdarg.h>
void
pf (int a, ...)
{
va_list ap;
va_start(ap, a);
while(a--)
puts(va_arg(ap, char *));
va_end(ap);
}
void
main (void)
{
pf(3, "Line 1", "line 2", "line 3");
}
```

62　XTOI函数

1．提　要

```
#include <stdlib.h>
unsigned xtoi (const char * s)
```

2．描　述

xtoi()函数扫描参数中的字符串。它跳过前面的空格，读到符号后，将用ASCII码表示的十六进制数转换为整型。

3．例　程

```
#include <stdlib.h>
#include <stdio.h>
void
main (void)
{
char buf[80];
```

```
int i;
gets(buf);
i = xtoi(buf);
printf("Read %s: converted to %x\n", buf, i);
}
```

参见atoi()函数。

4．返回值

有符号整数。如果字符串中不包含数，则返回零。