# 寫好C語言，宏定義很重要！

C語言與CPP編程 2022-02-23 08:45

收錄於話題
#c語言 98 #C/C++ 164



> 來源：玩轉嵌入式

寫好C語言，漂亮的宏定義很重要！使用宏定義可以防止出錯，提高可移植性，可讀性，方便性等。下面列舉了一些成熟軟件中常用的宏定義。

**1. 防止一個頭文件被重複包含**

```
# ifndef  COMDEF_H
# define  COMDEF_H
//頭文件內容
# endif
```

**2. 重新定義一些類型，防止由於各種平台和編譯器的不同，而產生的類型字節數差異，方便移植。**

```
typedef  unsigned  char  boolean; /* 布爾值類型。*/
typedef  unsigned  long  int  uint32; /* 無符號 32 位值 */
typedef  unsigned  short  uint16; /* 無符號 16 位值 */
typedef  unsigned  char  uint8; /* 無符號 8 位值 */
typedef  signed  long  int  int32; /* 帶符號的 32 位值 */
typedef  signed  short  int16; /* 有符號 16 位值 */
typedef 有符號的 char  int8; /* 有符號 8 位值 */
```

不建議使用：

```
個  typedef 無符號 字符 字節； /* 無符號 8 位值類型。*/
  typedef 無符號 短 字； /* Unsinged 16 位值類型。*/
  typedef  unsigned  long  dword; /* 無符號 32 位值類型。*/
  typedef  unsigned  char  uint1; /* 無符號 8 位值類型。*/
  typedef  unsigned  short  uint2; /* 無符號 16 位值類型。*/
  typedef  unsigned  long  uint4; /* 無符號 32 位值類型。*/
typedef 有 符號 字符 int1； /* 有符號的 8 位值類型。*/
  typedef 有 符號 的 short  int2; /* 有符號 16 位值類型。*/
  typedef  long  int  int4; /* 帶符號的 32 位值類型。*/
   typedef有 符號 的 long  sint31; /* 帶符號的 32 位值 */
   typedef  signed  short  sint15; /* 有符號 16 位值 */
   typedef  signed  char  sint7; /* 有符號 8 位值 */
```

### 3. 得到指定地址上的一個字節或字

```
#define MEM_B( x ) ( *( (byte *) (x) ) )
#define MEM_W( x ) ( *( (word *) (x) ) )
```

### 4. 求最大值和最小值

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

### 5. 得到一个field在结构体(struct)中的偏移量

```
#define FPOS( type, field ) \
/*lint -e545 */ ( (dword) &(( type *) 0)-> field ) /*lint +e545 */
```

### 6. 得到一个结构体中field所占用的字节数

```
#define FSIZ( type, field ) sizeof( ((type *) 0)->field )
```

### 7. 按照LSB格式把两个字节转化为一个Word

```
#define FLIPW( ray ) ( (((word) (ray)[0]) * 256) + (ray)[1] )
```

### 8. 按照LSB格式把一个Word转化为两个字节

```
#define FLOPW( ray, val ) \
(ray)[0] = ((val) / 256); \
(ray)[1] = ((val) & 0xFF)
```

### 9. 得到一个变量的地址(word宽度)

```
#define B_PTR( var ) ( (byte *) (void *) &(var) )
#define W_PTR( var ) ( (word *) (void *) &(var) )
```

### 10. 得到一个字的高位和低位字节

```
#define WORD_LO(xxx) ((byte) ((word)(xxx) & 255))
#define WORD_HI(xxx) ((byte) ((word)(xxx) >> 8))
```

### 11. 返回一个比X大的最接近的8的倍数

```
#define RND8( x ) ((((x) + 7) / 8 ) * 8 )
```

### 12. 将一个字母转换为大写

```
#define UPCASE( c ) ( ((c) >= 'a' && (c) <= 'z') ? ((c) - 0x20) : (c) )
```

### 13. 判断字符是不是10进制的数字

```
#define DECCHK( c ) ((c) >= '0' && (c) <= '9')
```

### 14. 判断字符是不是16进制的数字

```
#define HEXCHK( c ) ( ((c) >= '0' && (c) <= '9') ||\
((c) >= 'A' && (c) <= 'F') ||\
((c) >= 'a' && (c) <= 'f') )
```

### 15. 防止溢出的一个方法

```
#define INC_SAT( val ) (val = ((val)+1 > (val)) ? (val)+1 : (val))
```

### 16. 返回数组元素的个数

```
#define ARR_SIZE( a ) ( sizeof( (a) ) / sizeof( (a[0]) ) )
```

### 17. 返回一个无符号数n尾的值MOD_BY_POWER_OF_TWO(X,n)=X%(2^n)

```
#define MOD_BY_POWER_OF_TWO( val, mod_by ) \
( (dword)(val) & (dword)((mod_by)-1) )
```

### 18. 对于IO空间映射在存储空间的结构，输入输出处理：

```
#define inp(port) (*((volatile byte *) (port)))
#define inpw(port) (*((volatile word *) (port)))
#define inpdw(port) (*((volatile dword *)(port)))
#define outp(port, val) (*((volatile byte *) (port)) = ((byte) (val)))
```

```
#define outpw(port, val) (*((volatile word *) (port)) = ((word) (val)))
#define outpdw(port, val) (*((volatile dword *) (port)) = ((dword) (val)))
```

### 19. 使用一些宏跟踪调试

ANSI标准说明了五个预定义的宏名，其分别是：

```
_LINE_
_FILE_
_DATE_
_TIME_
_STDC_
```

如果编译不是标准的，则可能仅支持以上宏名中的几个，或根本不支持。记住编译程序也许还提供其它预定义的宏名。

_LINE_ 及 _FILE_ 宏指令在有关 #line 的部分中已讨论，这里讨论其余的宏名。

_DATE_ 宏指令含有形式为月/日/年的串，表示源文件被翻译到代码时的日期。

源代码翻译到目标代码的时间作为串包含在 _TIME_ 中。串形式为时：分：秒。

如果实现是标准的，则宏 _STDC_ 含有十进制常量1。如果它含有任何其它数，则实现是非标准的。

可以定义宏，例如：当定义了 _DEBUG ，输出数据信息和所在文件所在行。

```
#ifdef _DEBUG
#define DEBUGMSG(msg,date) printf(msg);printf(“%d%d%d”,date,_LINE_,_FILE_)
#else
#define DEBUGMSG(msg,date)
#endif
```

**20. 宏定义防止使用时错误用小括号包含。**

例如：

```
#define ADD(a,b) (a+b)
```

用 do{}while(0) 语句包含多语句防止错误，例如：

```
#difne DO(a,b) a+b;\
a++;
```

应用时：

```
if(….)
DO(a,b); //产生错误
else
```

解决方法:

```
#define DO(a,b) do{a+b;\
a++;}while(0)
```





收錄於話題# c語言 98

下一篇 · 400 行C 代碼實現一個虛擬機

喜歡此內容的人還喜歡

C語言常用標準庫解讀

嵌入式ARM

C語言逆向之循環結構分析

嵌入式ARM

花15分鐘快速掌握C語言中的指針

STM32嵌入式開發