**目录**

[1文件类型 2](#_Toc17162)

[2 prog 文件 2](#_Toc5965)

[3 format 文件 2](#_Toc4427)

[3.1变量命名规则 3](#_Toc879)

[3.2关键字使用规则 4](#_Toc32604)

[3.3xmemalloc label()详细说明 5](#_Toc31864)

[4 dat文件 6](#_Toc15016)

# 1文件类型

YC 汇编所使用的文件主要有三类：\*.prog 、\*.format 、\*.dat

# 2 prog 文件

\*.prog 文件是 YC 汇编的程序文件，其中存放 YC 汇编可执行的代码部分。相当于 C 语言中的\*.c 文件。

# 3 format 文件

\*.format 文件是YC汇编的头文件，其中存放YC汇编的宏定义、变量声明部分。相当于C语言中的\*.h 文件。

常量宏定义在\*.format 文件中。定义时常数数值在前，宏名在前，用一个空格隔开，宏名前可用\t 对  
齐。例如：  
37 LMP\_VERSION\_REQ //常量的值为 37

枚举定义在\*.format 文件中。定义时常数数值在前，枚举名在前，用一个空格隔开，枚举名前可用\t对齐。由于枚举与变量并不书写在同一个区域，所以枚举在定义时应用注释描述与枚举相关的变量名。例如：

/\*mem\_hf\_call\_state\*/  
(  
0 HF\_CALL\_STATE\_INACTIVE  
1 HF\_CALL\_STATE\_SETUP  
2 HF\_CALL\_STATE\_ACTIVE  
)

变量定义在\*.format 文件中。变量定义时应注意变量的长度和作用时间。变量定义时使用关键字指示变量的作用时间，并包含在圆括号中。定义时变量长度数值在前，变量名在后，用一个空格隔开，长度数值前可用\t 对齐，。

## 3.1变量命名规则

变量需要以mem开头，例如：1 mem\_xxxx

若不以mem开头，e命令会出现e命令无法识别变量的情况。

## 3.2关键字使用规则

|  |  |  |
| --- | --- | --- |
| 关键字 | 地址范围 | 变量特性 |
| memalloc() | 0x0 ~0x1fff | 1. 芯片上电以及lpm唤醒后为随机值（lpm掉电）； 2. lpm唤醒后不会再次走.dat的初始化流程，所以即使memalloc的变量在.dat中申请了初始，lpm唤醒后，仍然为随机值。 |
| xmemalloc() | 0x4000~0x4fff | 1. lpm唤醒后值会保持（lpm保电） 2. xmemalloc有两种申请方式：   xmemalloc()  新增的关键字中申请的变量会跟在已经存在的变量的后面  xmemalloc label()  label可以有多个，主要用途是可以用来区分不同的项目，label名可以自定义，每一个label中的变量，会从最后一个xmemalloc()变量的后面开始递增。  注意：   1. 、不同的label会相互覆盖。   例如，xmemalloc label1(  1 mem\_label1  )  xmemalloc label2(  1 mem\_label2  )  以上mem\_label1和mem\_label2被分配为同一个地址。   1. 、xmemalloc()申请的变量一定是在xmemalloc label()之前的，所以不要直接使用xmemalloc()来申请变量，否则会改变rom中已经存在的xmemalloc label()中申请变量的地址。 |
| omemalloc() | 0x4000~0x4fff | 1、lpm唤醒后值会保持（lpm保电）  2、会从最后一个xmemalloc()变量的后面开始递增，每一个omemalloc申请的变量地址会相互覆盖，即不同的omemalloc申请变量的起始地址相同。 |

## 3.3xmemalloc label()详细说明

xmemalloc label()中的label用来做变量分配管理。

规则：

1. xmemalloc/memalloc后面会跟一个label，label以下划线作为分隔符，下划线增加，一次只增加一个下划线，申请变量的地址就会**累加**。如：AA和AA\_BB两个label，则认为AA\_BB是在AA的基础上增加了“\_BB”得到的新label。
2. 如果label之间不符合依次累加**一个下划线**的规则，则不满足规则label申请出来的变量会**相互覆盖**。

以上1,2两点的设计初衷：

1、为了在同一个项目里面把不同模块的变量申请在一个连续的区域，且体现出层次关系（非必须也没必要区分）。

2 、芯片的可用内存地址是有限的，如果不同应用的地址都是一味的累加，很快内存就用完了。所以不同的应用不同的label，就可以让没有关联的应用，可以复用内存地址。

*通过举例说明，假设有三个label：AA\_BB,AA\_BB\_CC,AA\_EE*

变量申请示例：

xmemalloc AA\_BB(

1 mem\_AA\_BB

)

xmemalloc AA\_BB\_CC(

1 mem\_AA\_BB\_CC

)

xmemalloc AA\_EE(

1 mem\_AA\_EE

)

解析：

1、mem\_AA\_BB\_CC会在mem\_AA\_BB的后面累加。即

**AA\_BB\_CC中申请的变量，会在AA\_BB中的最后一个变量为基地址累加。**

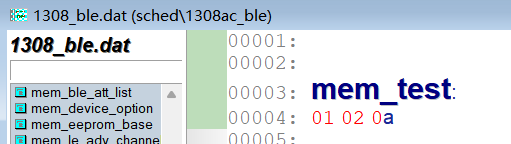
2、mem\_AA\_BB和mem\_AA\_EE的地址是一样的，即相互覆盖的关系。因为AA\_EE和AA\_BB不满足依次累加**一个下划线**的规则。

# 4 dat文件

\*.dat 文件是 YC 汇编的初始化文件，其中存放了 YC 汇编在执行前自动初始化内存的信息。

\*.dat初始化文件的存储格式：**16进制，小端模式。**

例如：



上图中对应的内存为：

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | mem\_test | mem\_test+1 | mem\_test+2 |
| 值 | 0x01 | 0x02 | 0x0a |