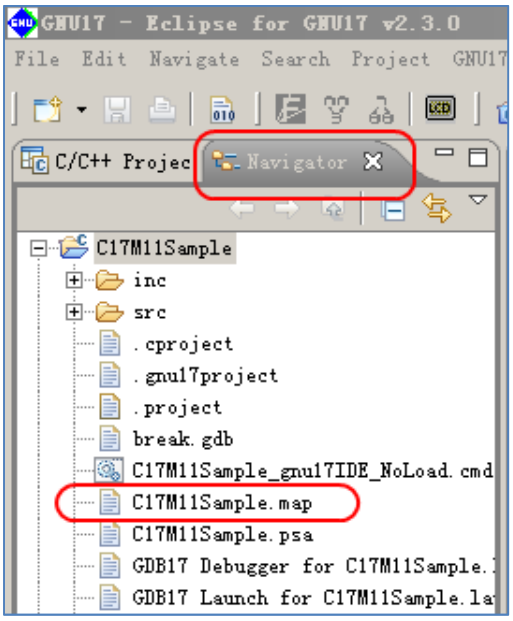
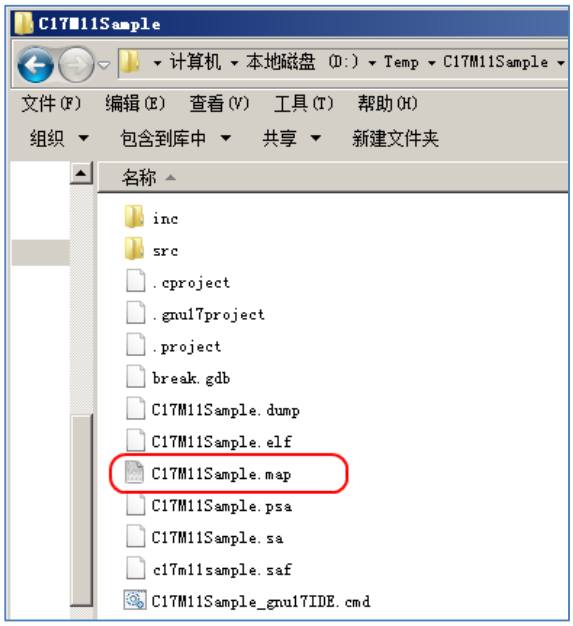
GNU17 项目空间占用情况查看方法

存储空间占用信息

GNU17 项目的存储空间占用信息保存在 map 文件中。

可以在 GNU17 开发环境中，在项目导航窗口中选择 Navigator 列表方式，找到“项目名.map”文件打开；也可以在项目所在目录下，找到 map 文件，用其它任何文本编辑器打开。如下图：





ROM 占用（即 Flash 存储区）空间

ROM 占用总空间：

在 map 文件中找到“__END_data_lma”文字，以及前面对应的十六进制地址信息，如下图：

```
192 src/t16_ch1.o(.rodata)
193 C:/EPSON/GNU17/lib/24bit/libstdio.a(.rodata)
194 C:/EPSON/GNU17/lib/24bit/libc.a(.rodata)
195 C:/EPSON/GNU17/lib/24bit/libgcc.a(.rodata)
196 C:/EPSON/GNU17/lib/24bit/libc.a(.rodata)
197                 0x00009138                __END_rodata=.
198                 0x00009138                  START_data_lma=__END_rodata
199                 0x0000915e                __END_data_lma=(__END_rodata+(__END_data-__START_data))
200LOAD src/boot.o
201LOAD src/clg.o
202LOAD src/init.o
```

将“__END_data_lma”地址减去 ROM 起始地址 0x8000，得到的数据即为 ROM 占用

空间。在上图的例子中，ROM 占用空间为：

0x915e-0x8000=0x115E　　即 4446 字节

ROM 空间占用详情：

有三类数据会保存在 ROM 空间中，这三类数据有各自不同的属性，分别是：

数据	种类	属性
1	指令	.text
2	常量	.rodata
3	变量的初值	.data

根据定位的需求不同，某些数据还应该要定位在指定的位置，例如，向量表必须定位在 ROM 起始地址，即 0x8000。因此，实际的 ROM 占用情况应该如下（默认配置）：

段名称	属性	种类	地址	功能
.vector	.rodata	常量	0x8000	常量类型的向量表数组
.text	.text	指令	跟在.vector 之后	代码编译后的指令
.rodata	.rodata	常量	跟在.text 之后	代码中使用的常量
.data_lma	.data	变量初值	跟在.rodata 之后	全局或静态变量的初值存储区

这些数据段分别占用了不同大小的 ROM 空间（长度可能为 0），可以在 map 文件中查看，如下列各图：

.vector 的地址和长度

```
80 .vector          0x00008000        0x80
81                  0x00008000                __START_vector=.
82  src/boot.o(.rodata)
83  .rodata         0x00008000        0x80 src/boot.o
```

.text 的地址和长度

```
87 .text            0x00008080      0x10b8
88                  0x00008080                __START_text=.
89  src/boot.o(.text)
```

.rodata 的地址和长度

```
183 .rodata         0x00009138         0x0
184                 0x00009138                __START_rodata=.
185  src/clg.o(.rodata)
```

.data_lma 的地址和长度

```
197                 0x00009138                __END_rodata=.
198                 0x00009138                __START_data_lma=__END_rodata
199                 0x0000915e                __END_data_lma=(__END_rodata+(__END_data-__START_data))
```

注：.data_lma 是.data 段在 ROM 中对应的初值空间，其长度可以参考.data 段，或者根据__END_data_lma 和__START_data_lma 之差计算出来。

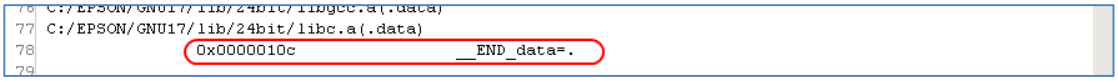
RAM 占用空间

RAM 占用空间被分为变量（全局、静态变量）和堆栈（局部变量）两部分。

变量：

在代码中声明的全局变量、静态变量，会被分配固定地址，从 RAM 的低地址空间开始分配。

在 map 文件中找到“__END_data”文字，以及前面对应的十六进制地址信息，如下图：



“__END_data”地址即为 RAM 中全局变量静态变量占用空间。在上图的例子中，RAM 占用空间为：0x10C，即 268 字节。

RAM 中全局变量和静态变量空间占用详情：

保存在 RAM 空间中的全局和静态变量，根据是否有初始值（即定义的时候就直接赋值），有两类不同的数据和属性，分别是：

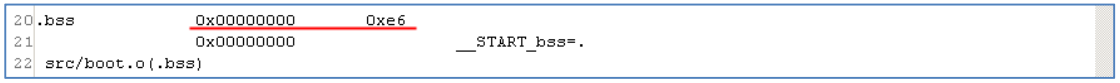
数据	种类	属性
1	无初始值	.bss
2	有初始值	.data

默认情况下，这些变量 RAM 占用情况应该如下：

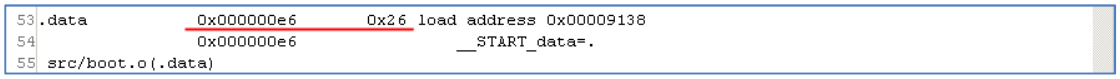
段名称	属性	种类	地址	功能
.bss	.bss	无初始值	0x0000	无初始值的全局或静态变量
.data	.data	有初始值	跟在.bss 之后	有初始值的全局或静态变量

这些数据段分别占用了不同大小的 RAM 空间（长度可能为 0），可以在 map 文件中查看，如下列各图：

.bss 的地址和长度



.data 的地址和长度



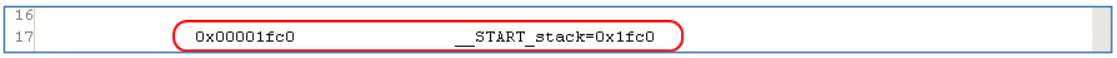
注：.bss 段中的初始值在初次上电时是随机的，通常初始化程序要将这些初始值清零；

注：.data 段在 ROM 中对应的初值空间（即上图中的 load address）是.data_lma；初始化程序需要将这些初始值从.data_lma 段复制到.data 段中。

堆栈：

这里的堆栈，实际上仅仅是指栈（stack）。

C17 的栈空间底部是在 RAM 空间的高地址处，通常是由指令设置为 RAM 总空间减去 64 字节的地址位置，如下图：



此例中，RAM 总大小是 8192 字节，因此栈底部初值需要被设置为 8192-64=8128，即 0x1FC0 地址。

函数调用、中断响应、参数传递，以及无法用通用寄存器保存的函数局部变量，都是保存在栈空间中的。入栈操作将是栈指针减小，即栈顶部是向低地址增长的。栈的最大使用范围不受硬件限制。只要栈空间不超过 RAM 空间的总范围，并且不与全局/静态变量空间（即.bss 和.data 段）重叠，栈的使用和程序运行就不受任何影响。

除非程序完整运行，否则无法从静态的代码分析或者 map 文件中获知栈空间的最大使用情况。

若要分析栈空间的最大使用情况，需要在程序运行（并且尽量覆盖到最大的栈使用程度）后，通过查看 RAM 内的数据情况（例如通过调试器查看 Memory 的内容），来判断栈顶部最远曾经到达的位置。

例如将 RAM 空间（除变量空间.bss 和.data 段外）全部初始化为 0xAA，程序运行后查看哪些 RAM 空间被修改过了，就可以知道栈可能需要的空间大小。

注意事项

以上 ROM 和 RAM 空间分配，可以在 GNU17 的项目属性窗口中，“GNU17 Linker Script Settings”配置下查看和修改；栈指针的初始化，应该在复位向量函数（boot 函数）中用汇编指令设置。