ARM Cortex-M底层技术（八）KEIL MDK 分散加载-2-语法 - weixin_39118482的博客

本文链接：https://blog.csdn.net/weixin_39118482/article/details/79983692
KEIL MDK 分散加载的结构-2-语法

语法、枯燥的、烦人的语法……，但是有些时候木有办法，我本来也不想写这些东西，但确实绕不过去，我认为把它当成一种工具比较合适，了解大概结构以及基本的语法，一些细节没必要记的那么清楚，遇到问题的时候知道去哪里查就好了，这部分东西来自Keil的帮助文档，帮助文档中内容更加丰富也更加复杂，我节选出了一部分，了解这些基本可以搞定绝大多数应用，节选了部分关键内容，供参考。

1、加载域描述

加载域

load_region_name (base_address | ("+" offset)) [attribute_list] [max_size]
{
    execution_region_description+

}

其中：

load_region_name： 加载域名称。

base_address： 指定要在其中链接加载域中对象的地址。 base_address 必须是字对齐的。

+offset： 描述一个基址，该基址超出前一个加载区末尾 offset 个字节。offset 的值必须能被 4 整除。

如果这是第一个加载域，则 +offset表示基址从零后面的 offset 个字节开始。

attribute_list： 指定加载域内容的属性：

ABSOLUTE 绝对地址： 域的加载地址由基址指示符指定。

ALIGN alignment : 将加载域的对齐约束从 4 增加到 alignment。alignment 必须为 2 的正数幂。如果加载域同样具有 base_address，则必须为 alignment 对齐。 如果加载域具有 +offset，则链接器将计算得到的域基址与 alignment 边界对齐。

NOCOMPRESS : 缺省情况下，将启用 RW 数据压缩。 使用 NOCOMPRESS 关键字可以指定不能在最终映像中压缩加载域的内容。

OVERLAY : 使用 OVERLAY 关键字可以使多个加载域位于同一个地址上。 ARM 工具没有提供重叠机制。 若要在同一地址使用多个加载域，必须提供自己的重叠区管理器。

PI : 此区与位置无关。

RELOC : 此区可重定位。

max_size : 指定加载域的最大大小。 这是在进行任何解压缩或零初始化之前的加载区大小。 如果指定可选的 max_size 值，并且为域分配的字节数多于 max_size，则 armlink 会生成错误。

execution_region_description : 指定执行域的名称、地址和内容。

2、执行域描述

执行域

```
exec_region_name (base_address | "+" offset) [attribute_list] [max_size | length]
{
    input_section_description*

}
```

exec_region_name : 命名执行区。

base_address : 指定要在其中链接执行区中对象的地址。 base_address 必须是字对齐的。

+offset : 描述一个基址，该基址超出前一个执行区末尾 offset 个字节。
offset 的值必须能被 4 整除。
如果前面没有执行区（即，这是加载区中的第一个执行区），则 +offset 表示基址从该执行区所在的加载区的基址后面 offset 个字节开始。

attribute_list : 它指定执行区内容的属性：

ABSOLUTE : 绝对地址。 区执行地址是由基址指示符指定的。

ALIGN alignment : 将执行区的对齐约束从 4 增加到 alignment。
alignment 必须为 2 的正数幂。如果执行区具有 base_address，则必须为 alignment 对齐。 如果执行区具有 +offset，则链接器将计算得到的区基址与 alignment 边界对齐。

EMPTY : 在执行区中保留一个给定长度的空白内存块，
通常供堆或堆栈使用。 不能将任何节放置在具有 EMPTY 属性的区中。

FILL : 创建一个链接器生成的区，其中包含一个值。
如果指定 FILL，则必须为其赋值，例如： FILL 0xFFFFFFFF。FILL 属性会替代以下组合： EMPTY ZEROPAD PADVALUE。
在某些情况下（例如仿真），长时间的零循环是比较合适的。

FIXED : 固定地址。 链接器尝试通过插入填充使执行地址等于加载地址。 这会使区成为根区。 如果这

无法完成，则链接器会生成错误。

NOCOMPRESS : 缺省情况下，将启用 RW 数据压缩。 使用 NOCOMPRESS 关键字可以指定不能在最终映像中压缩执行区中的 RW 数据。

OVERLAY : 用于具有重叠的地址范围的节。 将为具有 OVERLAY 属性且基址偏移为 +0 的连续执行区指定相同的基址。

PADVALUE : 定义任何填充的值。 如果指定 PADVALUE，则必须为其赋值，例如：
EXEC 0x10000 PADVALUE 0xFFFFFFFF EMPTY ZEROPAD 0x2000
这会创建一个大小为 0x2000 且充满0xFFFFFFFF 的区。

PADVALUE : 必须为一个字大小。将忽略加载区中 的 PADVALUE 属性。

PI : 此区仅包含与位置无关的节。

SORTTYPE : 指定执行区的排序算法。

UNINIT : 用于创建包含未初始化的数据或内存映射的I/O 的执行区。

ZEROPAD : 零初始化的节作为零填充块写入 ELF 文件，因此，运行时无需使用零进行填充。
只有根执行区能够使用 ZEROPAD 属性进行零初始化。 如果将 ZEROPAD 属性用于非根执行区，则会生成警告并忽略该属性。
在某些情况下（例如仿真），长时间的零循环是比较合适的。

max_size : 对于标记为 EMPTY 或 FILL 的执行区，max_size 值会解释为区的长度。 在其他情况下，max_size 值会解释为执行区的最大大小。

[–]length : 只能与 EMPTY 一起使用，以表示在内存中向下增长的堆栈。 如果指定的长度为负值，则将 base_address 作为区结束地址。

input_section_description指定输入节的内容。

3、输入节描述

输入节：

input_section_description ::=
module_select_pattern
["(" input_section_selector ("," input_section_selector)* ")"]
input_section_selector ::=

("+" input_section_attr | input_section_pattern | input_symbol_pattern)

module_select_pattern:

一个由文字文本构成的模式。 通配符 * 与零个或零个以上的字符相匹配；? 与任何单个字符相匹配。
匹配不区分大小写，即使是在区分文件命名大小写的主机上。
可以使用 *.o 匹配所有对象； 而使用 * 匹配所有对象文件和库。
当 module_select_pattern 与以下内容之一相匹配时，则表示输入节与模块选择器模式相匹配：

- 包含节的对象文件的名称。
- 库成员名称（不带前导路径名）。
- 从中提取节的库的完整名称（包括路径名）。 如果名称包含空格，则可以使用通配符简化搜索。 例如，使用 *libname.lib 匹配 C:\lib dir\libname.lib。

通过使用特殊模块选择器模式 .ANY，您可以将输入节分配给执行区，而无需考虑其父模块。 可以使用 .ANY 以任意 分配方式填充执行区。
使用 .ANYnum 可指定特定的优先级顺序，其中 num是一个大于零的正整数后缀。 使用最大整数指定的优先级最高。

input_section_attr

属性选择器与输入节属性相匹配。 每个 input_section_attr 的前面都有一个 +。
如果指定一个模式以匹配输入节名称，名称前面必须有一个 + 号。
可以省略紧靠 + 号前面的任何逗号。
选择器不区分大小写。 可以识别以下选择器：

- RO-CODE
- RO-DATA
- RO，同时选择 RO-CODE 和 RO-DATA
- RW-DATA
- RW-CODE
- RW，同时选择 RW-CODE 和 RW-DATA
- ZI
- ENTRY，即，包含 ENTRY 点的节。

可以识别以下同义词：

- CODE 表示 RO-CODE
- CONST 表示 RO-DATA
- TEXT 表示 RO
- DATA 表示 RW
- BSS 表示 ZI。

可以识别以下伪属性：

- FIRST
- LAST

以下属性选择器模式描述了节在执行区中的放置顺序：

第一个和最后一个节
如果节位置顺序很重要（例如，如果特定输入节必须是区中的第一个输入节，而包含校验和的输入节必须是最后一个输入节），则可以使用 FIRST 和 LAST 标记执行区中的第一个节和最后一个节。

input_section_selector 列表中只能有一个 FIRST 或 LAST 属性，且该属性必须在单个 input_section_attr 的后面。 例如：
*(section, +FIRST)

此模式是正确的。

*(+FIRST, section)
此模式不正确，会生成错误消息。特殊模块选择器
通过使用特殊模块选择器模式 .ANY，您可以将输入节分配给执行区，而无需考虑其父模块。 可以使用一个或多个 .ANY 模式以任意 分配方式填充执行区。 在大多数情况下，使用单个 .ANY 等效于使用 * 模块选择器。修改的选择器
一个分散加载文件中不能包含两个 * 选择器。 但是，可以使用两个修改的选择器（如 *A 和 *B），也可将 .ANY 选择器与 * 模块选择器配合使用。 * 模块选择器的优先级比 .ANY 高。 如果删除了文件中包含 * 选择器的部分，.ANY 选择器将变为活动状态。未分配的节
在解析所有其他（非 .ANY）输入节描述后，才会解析具有 .ANY 模块选择器模式的输入节描述。 所有未分配给执行区的节将分配给 .ANY 区。
如果存在多个 .ANY 模式，链接器将使用未分配给执行区的最大节，并将该节分配给具有足够可用空间的最明确的 .ANY 执行区。 当 armlink 进行此项选择时，它将 .ANY(.text) 视为比 .ANY(+RO) 更明确。
如果几个执行区具有同等的明确性，则将该节分配给可用剩余空间最大的执行区。例如：

- 如果两个执行区具有同等的明确性，其中一个执行区的大小限制为 0x2000，另一个执行区没有限制，则将所有节分配给第二个没有限制的 .ANY 区。
- 如果两个执行区具有同等的明确性，其中一个执行区的大小限制为 0x2000，另一个执行区的大小限制

为 0x3000，则将要放置的第一批节分配给第二个大
小限制为 0x3000 的 .ANY 区，直至第二个 .ANY 的剩余

> 大小减少到 0x2000。 此后，将在两个 .ANY 执行区之间交替分配节。