RX32x6xx 应用笔记

文档编号：AN00017

SCT 文件编写应用指导

版本：V1.0

目录

表目录

1 简介

SCT 文件（分散加载文件）用于指定 ARM 连接器在生成映像文件时如何分配 RO（Read-Only）、RW（Read-Write）、ZI（Zero Initialized）等数据的存放地址。通过 SCT 文件，开发者可以对程序的内存分布进行精细控制，以满足特定的应用需求。本文以 RX32x6xx（64KB Flash / 8KB SRAM）系列芯片为例，试编写 SCT 文件语法、格式以及注释，以便于用户参考。

2 SCT 文件语法与示例

SCT 文件使用简单的文本格式，包含加载域（Load Region）和执行域（Execution Region）的定义，以及输入节区（Input Section）的描述。

基本语法

参考 RX32S610 默认的 SCT 文件，基本语法如下：

```
LR_IROM1 0x08000000 0x00010000  {     ; load region size_region
  ER_IROM1 0x08000000 0x00010000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1 0x20000000 0x00002000  {  ; RW data
   .ANY (+RW +ZI)
  }
}
```

上述 RX32S610 默认 SCT 文件涉及到的语法内容解析：

LR_IROM1 0x08000000 0x00010000	->	Flash 起始地址和大小
ER_IROM1 0x08000000 0x00010000	->	代码和只读数据区
*.o (RESET, +First)	->	初始化代码段
*(InRoot$$Sections)	->	引导代码段
.ANY (+RO)	->	所有只读代码和数据
.ANY (+XO)	->	所有只执行代码
RW_IRAM1 0x20000000 0x00002000	->	SRAM 起始地址和大小
.ANY (+RW +ZI)	->	所有可读写的变量和零初始化数据

3 SCT 文件创建和应用

1. 在项目中新建一个文本文件，命名为.sct，例如 S610_Data.sct。
2. 编辑 SCT 文件：根据实际需求，编辑 SCT 文件内容。（*参见章节 4：SCT 文件编写*）。
3. 配置编译器选项：选择 Options for Target -> Linker，取消勾选 Use Memory Layout from Target Dialog。新勾选 Report 'might fail' Conditions as Errors。并在 Scatter File 中指定刚才创建的 SCT 文件路径。
4. 编译项目：编译项目时，编译器会根据 SCT 文件的配置生成映像文件，并按照指定的内存布局进行加载和执行。

4 SCT 文件编写

当开发者需要对 Flash 进行分区时需要改写 SCT 文件。

例如在 Flash 末尾分一片区域用于数据保存，以 RX32S610 为例，将 64KB 的 Flash 分为 63KB 和 1KB，其中 1KB 专用于数据保存，编译器在编译下载时就不会使用到这 1KB 的指定空间。

S610_Data.sct

改写后 RX32S610 的 SCT 示例如下：

```
LR_IROM1 0x08000000 0x0000FC00  {      ; load region size_region
  ER_IROM1 0x08000000 0x0000FC00  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1 0x20000000 0x00002000  {  ; RW data
   .ANY (+RW +ZI)
  }
}

LR_IROM2 0x0800FC00 0x00000400  {      ; S610 Data
  ER_IROM2 0x0800FC00 0x00000400  {  ; load address = execution address
   *.o(myArray_Info)
  }
}
```

上述 RX32S610 改动 SCT 文件涉及到的关键内容解析：

LR_IROM1 0x08000000 0x0000FC00	->	Flash 起始地址和大小 63KB
LR_IROM2 0x0800FC00 0x00000400	->	用户数据保存区起始地址和大小 1KB
*.o(myArray_Info)	->	存放“myArray_Info”用户定义字段数据

5 数据定义和保存举例

当 SCT 文件编写好后，RX32S610 的 Flash 末 1KB 的空间将用于保存数据。用户可以通过两种方式在 Flash 的 1KB 空间中保存数据。

1. 用户可自定义字段“myArray_Info”用于在地址 0x0800FC00 存放数据，数据地址自动排序。
2. 用户可以在 0x0800FC00~0x08010000 地址中指定地址存放数据

注：用户定义数据的地址和大小不可超过 SCT 分配的地址和大小。

用户自定义的数据和保存示例如下：

```
//方式 1
__attribute__((used,section("myArray_Info"))) //保持这里的字段名称与 sct 文件中一致
const uint32_t myArray1[10] =
{
  0xA1, 0xA2, 0xA3, 0xA4, 0xA5,
  0xA6, 0xA7, 0xA8, 0xA9, 0xAA
};

//方式 2
const uint32_t myArray2[10] __attribute__((at(0x0800FC00)))  =
{
  0xA111, 0xA222, 0xA333, 0xA444, 0xA555,
  0xA666, 0xA777, 0xA888, 0xA999, 0xAAAA
};
```

6 调试与验证

当用户在 SCT 文件编写完成，并定义好数据和地址后，务必进行充分的调试和验证，以确保程序能够按预期运行。可通过两种常用的方式查看数据区域：

1. 在项目编译生成的.map 文件中查看是否数据被编译进去。
2. 在 Debug 中查看分配的指定地址空间是否存在用户定义的值。

通过以上步骤和注意事项，用户可以在项目中成功编写和应用 SCT 文件，实现 Flash 分区以存放用户数据。

7 版本历史

表 7.1 版本历史

日期	版本	更改内容
2024 年 12 月 25 日	V1.0	初版