N32H7xx系列软件TCM使用指南

简介

此文档的目的在于让使用者能够快速熟悉N32H7xx系列微控制器（MCU）的TCM使用方法，用于指导用户更容易的了解和使用 TCM。

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

目录

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

1. TCMSRAM 存储器特性

TCMSRAM 存储器由 SRAM 存储器构建，包括以下类型：作为 M7 紧密耦合存储器使用的 TCM（ITCM、D0TCM、D1TCM），以及作为芯片上通用 SRAM 使用的 AXISRAM（AXISRAM2/3）。TCM 存储器也可通过 Cortex-M7 的 AHBS 接口，借助 MDMA 以 DMA 方式访问。TCMSRAM 存储器总共有 1024KB 的 SRAM 可用。这是 64 位数据宽度的存储器，分为八个 bank，每个 bank 128KB。它支持 ECC（错误检查和纠正），并且 ECC 校验数据与普通数据并行存储。ECC 监视器 1（ECCMON1）控制并报告 TCMSRAM 和系统 AXI SRAM 的 ECC 状态。

用户可以以 128KB 为粒度配置 ITCM、DTCM 的大小，其余的内存空间将自动分配给两个 AXI SRAM2/3。请注意，DTCM 分为两部分：D0TCM 分配在 64 位 DTCM 数据总线的低 32 位部分，而 D1TCM 分配在高 32 位部分。

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

2. TCMSRAM 内存映射

表 2-1 展示了代码和 RAM 空间的内存映射。基于 TCMSRAM 的配置，内存区域相应地进行分配。

注意：AXI SRAM 1/2/3 的地址始终是连续的，任何 ITCM/DTCM 的分配将从 AXI SRAM3 结束地址开始减少分配，请参考表 3-1。

表 2-1 TCMSRAM 和相关模块的内存映射表

Region	Start Address	End Address	Size	Cortex M7	Cortex M4
Peripheral (Register access)	0x5111_8000	0x5111_BFFF	16KB	OTPC	
	0x5110_4000	0x5110_4FFF	4KB	ECCMON1	
	0x5004_B000	0x5004_BFFF	4KB	TCMSRAMC	
RAM (WBWA)	0x240A_0000	0x2411_FFFF	512KB	AXI SRAM 3	
	0x2402_0000	0x2409_FFFF	512KB	AXI SRAM 2	
	0x2400_0000	0x2409_FFFF	128KB	AXI SRAM 1	
	0x2000_0000	0x200F_FFFF	1MB	DTCM	Reserved
Code (WT)	0x0000_0000	0x000F_FFFF	1MB	ITCM	INIT VTOR REMAP

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

3. TCM 控制器

TCM 控制器将 TCM（64 位 ITCM 总线或两个 32 位总线：D0TCM 和 D1TCM）接口转换为 SRAM 接口。TCM 接口与 Cortex-M7 内核同步运行，并且频率相同。TCM 控制器负责分配 SRAM bank 给 ITCM、D0TCM、D1TCM、AXISRAM2 和 AXISRAM3，并确保它们之间没有 bank 冲突。根据 ITCMS 和 DTCMS 设置，它计算每种内存类型的大小并分配相应的 SRAM bank。

任何对 TCM 内存的越界地址访问，包括错误的大小设置，都会返回错误响应。在这种情况下，内存内容不会改变，读取的数据不确定。

该控制器支持 ECC 功能，具有单比特错误纠正和双比特错误检测（SECDED），并具有错误注入功能。ECC 功能的使能信号来自 ECC 监视器。如果相应接口的 ECC 功能被启用，当出现 ECC 错误时，TCM 控制器会向 ECC 监视器输出 ECC 错误类型、地址、数据和相应的 ECC 代码。

注意：有关更多 ECC 处理的详细信息，请参考《EN_UM_N32H7xx Series User Manual.pdf》的 2.4 ECCMON 章节。

3.1. TCMSRAM 大小配置

TCMSRAM 内存大小可以通过 OTP 配置在系统启动时定义，如果配置正确，TCM_CTRL 的状态位（CSA）将被设置。用户必须在访问 TCMSRAM 内存之前检查该状态位。当 CSA＝L 时，对 TCM 和 AXI SRAM2/3 的任何操作都会导致不可预测的行为。

3.1.1. OTP 配置

OTP 使用 TCM_SZ_CFG_OTP 设置进行大小配置的详细信息如下所示。

表 3-1 OTP 配置表(N32H76x 或者 N32H78x)

No.	TCM_SZ_CFG_OTP [5:0]	ITCM Size(KB)	DTCM Size(KB)	AXISRAM Size(KB)
1	6'h00	1024	0	128
2	6'h01	896	128	128
3	6'h02	768	256	128

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

4	6'h03	640	384	128
5	6'h04	512	512	128
6	6'h05	384	640	128
7	6'h06	256	768	128
8	6'h07	128	896	128
9	6'h08	0	1024	128
10	6'h09	896	0	256
11	6'h0A	768	128	256
12	6'h0B	640	256	256
13	6'h0C	512	384	256
14	6'h0D	384	512	256
15	6'h0E	256	640	256
16	6'h0F	128	768	256
17	6'h10	0	896	256
18	6'h11	768	0	384
19	6'h12	640	128	384
20	6'h13	512	256	384
21	6'h14	384	384	384
22	6'h15	256	512	384

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

23	6'h16	128	640	384
24	6'h17	0	768	384
25	6'h18	640	0	512
26	6'h19	512	128	512
27	6'h1A	384	256	512
28	6'h1B	256	384	512
29	6'h1C	128	512	512
30	6'h1D	0	640	512
31	6'h1E	512	0	640
32	6'h1F	384	128	640
33	6'h20	256	256	640
34	6'h21	128	384	640
35	6'h22	0	512	640
36	6'h23	384	0	768
37	6'h24	256	128	768
38	6'h25	128	256	768
39	6'h26	0	384	768
40	6'h27	256	0	896
41	6'h28	128	128	896

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

42	6'h29	0	256	896
43	6'h2A	128	0	1024
44	6'h2B	0	128	1024
45	6'h2c~3F	0	0	1152

表 3-2 OTP 配置表(N32H73x)

No.	TCM_SZ_CFG_OTP [5:0]	ITCM Size(KB)	DTCM Size(KB)	AXISRAM Size(KB)
1	6'h1E	512	0	256
2	6'h1F	384	128	256
3	6'h20	256	256	256
4	6'h21	128	384	256
5	6'h22	0	512	256
6	6'h23	384	0	384
7	6'h24	256	128	384
8	6'h25	128	256	384
9	6'h26	0	384	384
10	6'h27	256	0	512
11	6'h28	128	128	512
12	6'h29	0	256	512

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

13	6'h2A	128	0	640
14	6'h2B	0	128	640
15	6'h2c~3F	0	0	768

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

4. 分散加载文件

分散加载文件（Scatter Loading File）是嵌入式系统开发中用于描述程序内存布局的配置文件，尤其在 ARM 架构的微控制器开发中广泛应用。它通过定义代码、数据、堆栈等在内存中的具体位置和大小，帮助开发者优化内存使用，确保程序在编译和链接时能够正确分配到指定的内存区域。

4.1. Map 文件

了解分散加载文件前先了解一下 map 文件，map 文件是链接器生成的用于描述程序内存布局的详细报告文件。

4.1.1. Image component sizes

Image component sizes 部分提供了程序映像（Image）的各个组成部分的详细大小信息，其中 Code (inc. data)、RO Data、RW Data、ZI Data 和 Debug 是常见的字段，它们分别代表了程序的不同部分在内存中的分布情况。

图 4-1 map_映像组件大小 1

```
==============================================================================

Image component sizes

      Code (inc. data)   RO Data    RW Data    ZI Data      Debug   Object Name

         232         18          0          4          0       1727   log.o
         618        304          0          4          0       4074   main.o
         132         22          0          0          0       4583   misc.o
         912         26          0          0          0      11536   n32h76x_78x_eccmon.o
         240          0          0          0          0      23618   n32h76x_78x_gpio.o
         272          4          0          0          0       5061   n32h76x_78x_it.o
        1616        140          0          0          0     116149   n32h76x_78x_rcc.o
         408         52          0          0          0      12354   n32h76x_78x_usart.o
          36          8       1000          0       8192       1016   startup_n32h76x.o
         116         18          0          0          0     876017   system_n32h76x_78x.o

    ------------------------------------------------------------------------------
        4592        592       1032          8       8192    1056135   Object Totals
           0          0         32          0          0          0   (incl. Generated)
          10          0          0          0          0          0   (incl. Padding)

    ------------------------------------------------------------------------------
```

Code (inc. data)：表示程序的代码段大小，包括（Code）程序的(inc. data)实际执行代码和嵌入在代码中的常量数据（如立即数、跳转表等），通常存储在 Flash 中。

RO Data：表示只读数据（Read-Only Data），即程序中定义的常量数据及 const 修饰的全局变量、字符串常量，这些数据在程序运行期间不会被修改，因此可以存储在 Flash 中。

RW Data：表示已初始化的读写数据（Read-Write Data），包括已初始化的全局变量和静态变量。这些数据在程序启动时会被加载到 RAM 中，并且在程序运行期间可以被修改。RW Data 的初始值会存储在 Flash 中，程序启动时会被拷贝到 RAM 中

ZI Data：表示零初始化数据（Zero-Initialized Data），包括未初始化的全局变量和静态变量，

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

这些变量在程序启动时会被初始化为零，不占用 Flash 空间，仅在 RAM 中分配。

Debug：表示调试信息占用的空间，通常包含调试符号、行号信息、变量名等调试相关的数据，不会影响程序的运行，仅用于调试和开发阶段，不会加载到目标设备中，仅存在于调试文件中。

图 4-2 map_映像组件大小 2

```
==============================================================================

    Total RO  Size (Code + RO Data)                 5924 (   5.79kB)
    Total RW  Size (RW Data + ZI Data)              8208 (   8.02kB)
    Total ROM Size (Code + RO Data + RW Data)       5936 (   5.80kB)

==============================================================================
```

在 Image component sizes 最后展现总的程序在 Flash（ROM）和 RAM 中的内存占用情况， Total RO Size、Total RW Size 和 Total ROM Size 是重要的内存占用统计信息。

Total RO Size (Code + RO Data)：只读（Read-Only）部分的总大小， Total RO Size = Code + RO Data，存储在 Flash 中。

Total RW Size (RW Data + ZI Data)：读写（Read-Write）部分的总大小，Total RW Size = RW Data + ZI Data，存储在 RAM 中。

Total ROM Size (Code + RO Data + RW Data)：程序在 Flash 中占用的总大小， Total ROM Size = Code + RO Data + RW Data，存储在 Flash 中。如果工程编译后有生成 bin 文件，这个 bin 文件大小就等于 Total ROM Size 。RW Data 的初始值存储在 Flash 中，程序启动时会被拷贝到 RAM，所以 RW 占 FLASH 和 RAM 的空间。

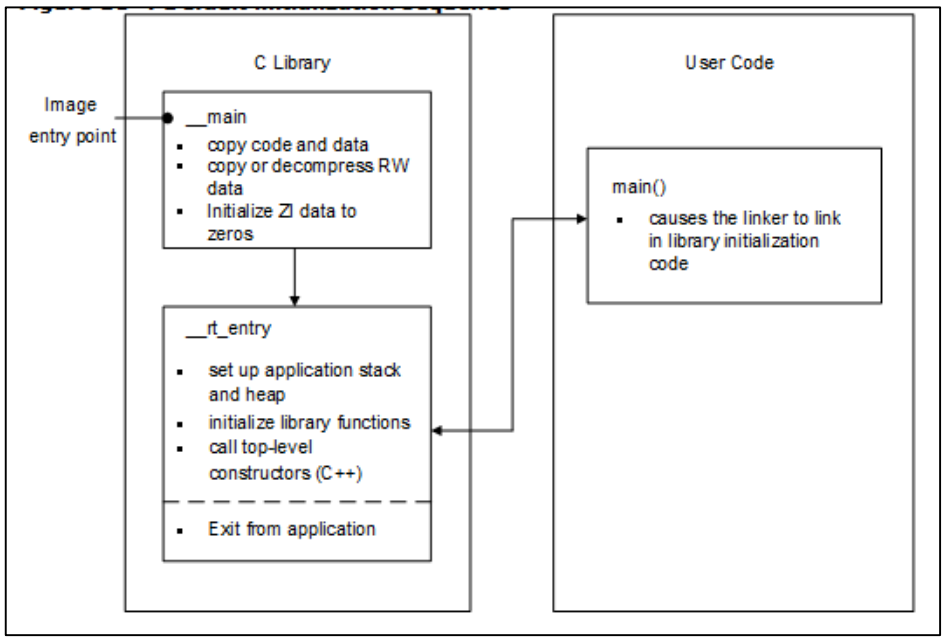
RW Data 是谁在什么时候从 Flash 拷贝到 RAM 的呢？

答案是：启动文件中__main()函数实现。

__main 和__rt_entry 是一个由编译器自动生成的函数，是程序启动过程中的关键部分。他们的主要职责是初始化 C 运行时环境，并确保程序能够正确地从启动代码跳转到用的 main() 函数。

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

图 4-3 默认初始化流程



__main 的主要功能包括：

- **初始化 RW Data：**将根据分散加载文件（Scatter Loading File）中的定义，找到 RW Data 在 Flash 中的起始地址和大小，将已初始化的全局变量和静态变量从 Flash 拷贝到 RAM。
- **清零 ZI Data：**将未初始化的全局变量和静态变量所在的 RAM 区域清零。

__rt_entry 的主要功能包括：

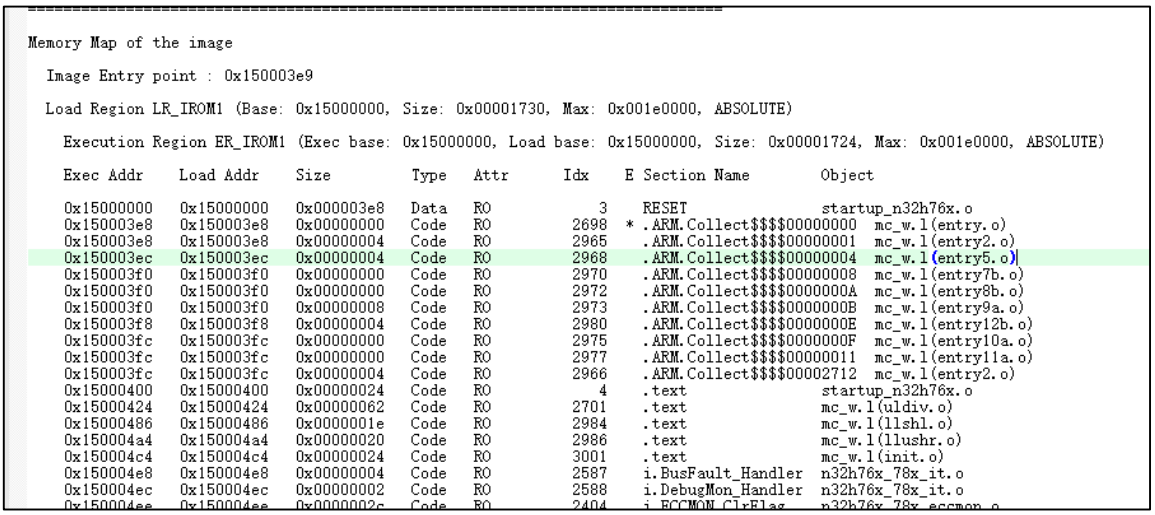
- **初始化堆和栈：**设置堆（Heap）和栈（Stack）的初始状态。
- **跳转到 main() 函数：**完成初始化后，跳转到用户的 main() 函数。

4.1.2. Memory Map of the image

Memory Map of the image 是 map 文件中的一个重要部分，它详细描述了程序映像（Image）在内存中的布局和分配情况。通过分析 Memory Map of the image，可以了解程序的各个部分（如代码、数据、堆栈等）在内存中的具体位置和大小，从而优化内存使用并确保程序的正确运行。

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

图 4-4 映像的存储映射



```
Memory Map of the image

  Image Entry point : 0x150003e9

  Load Region LR_IROM1 (Base: 0x15000000, Size: 0x00001730, Max: 0x001e0000, ABSOLUTE)

    Execution Region ER_IROM1 (Exec base: 0x15000000, Load base: 0x15000000, Size: 0x00001724, Max: 0x001e0000, ABSOLUTE)

    Exec Addr    Load Addr    Size         Type   Attr      Idx    E Section Name        Object

    0x15000000   0x15000000   0x000003e8   Data   RO            3    RESET               startup_n32h76x.o
    0x150003e8   0x150003e8   0x00000000   Code   RO         2698  * .ARM.Collect$$$$00000000  mc_w.l(entry.o)
    0x150003e8   0x150003e8   0x00000004   Code   RO         2965    .ARM.Collect$$$$00000001  mc_w.l(entry2.o)
    0x150003ec   0x150003ec   0x00000004   Code   RO         2968    .ARM.Collect$$$$00000004  mc_w.l(entry5.o)
    0x150003f0   0x150003f0   0x00000000   Code   RO         2970    .ARM.Collect$$$$00000008  mc_w.l(entry7b.o)
    0x150003f0   0x150003f0   0x00000000   Code   RO         2972    .ARM.Collect$$$$0000000A  mc_w.l(entry8b.o)
    0x150003f0   0x150003f0   0x00000008   Code   RO         2973    .ARM.Collect$$$$0000000B  mc_w.l(entry9a.o)
    0x150003f8   0x150003f8   0x00000004   Code   RO         2980    .ARM.Collect$$$$0000000E  mc_w.l(entry12b.o)
    0x150003fc   0x150003fc   0x00000000   Code   RO         2975    .ARM.Collect$$$$0000000F  mc_w.l(entry10a.o)
    0x150003fc   0x150003fc   0x00000000   Code   RO         2977    .ARM.Collect$$$$00000011  mc_w.l(entry11a.o)
    0x150003fc   0x150003fc   0x00000004   Code   RO         2966    .ARM.Collect$$$$00002712  mc_w.l(entry2.o)
    0x15000400   0x15000400   0x00000024   Code   RO            4    .text               startup_n32h76x.o
    0x15000424   0x15000424   0x00000062   Code   RO         2701    .text               mc_w.l(uldiv.o)
    0x15000486   0x15000486   0x0000001e   Code   RO         2984    .text               mc_w.l(llshl.o)
    0x150004a4   0x150004a4   0x00000020   Code   RO         2986    .text               mc_w.l(llushr.o)
    0x150004c4   0x150004c4   0x00000024   Code   RO         3001    .text               mc_w.l(init.o)
    0x150004e8   0x150004e8   0x00000004   Code   RO         2587    i.BusFault_Handler  n32h76x_78x_it.o
    0x150004ec   0x150004ec   0x00000002   Code   RO         2588    i.DebugMon_Handler  n32h76x_78x_it.o
    0x150004ee   0x150004ee   0x0000002c   Code   RO         2404    i.ECCMON_ClrFlag    n32h76x_78x_eccmon.o
```

Load Region (LR_IROM1):加载区域，表示程序在加载时的内存布局,可以清楚的了解到加载域的基地址、已占用大小、最大可使用大小和属性。

Execution Region (ER_IROM1): 执行区域，表示程序在运行时的内存布局。执行域的基地址、已占用大小、最大可使用大小和属性。

执行区域与加载区域的基地址相同，表示代码和数据在加载和运行时位于同一位置,如果执行区域与加载区域的基地址不同，就需要__main 从加载地址拷贝到执行地址。

4.2. 分散加载文件的作用

分散加载文件的主要作用是为链接器提供详细的内存布局信息，确保程序在编译和链接时能够正确分配到指定的内存区域。具体来说，分散加载文件可以：

定义内存区域的起始地址和大小：通过指定内存区域的起始地址和大小，确保程序的不同部分能够正确加载到指定的内存区域。

指定代码段、数据段、堆栈段等在不同内存区域中的分布：通过分散加载文件，可以将代码、数据、堆栈等部分分配到不同的内存区域，优化内存使用。

支持多块内存区域的配置：适用于复杂的嵌入式系统，支持多块内存区域的配置，确保系统资源的合理分配。

优化内存使用：通过合理配置分散加载文件，开发者可以精确控制程序的内存布局，避免内存浪费。

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

4.3. 分散加载文件的结构

分散加载文件通常由多个部分组成，每个部分描述了内存区域和加载区域的配置。以下是一个典型的分散加载文件的结构：

```
LR_ROM1 0x15000000 0x1E0000   {      ; 加载区域定义
  ER_ROM1 0x15000000 0x1E0000   {   ; 执行区域定义
    *.o (RESET, +First)                ; 复位向量表
    *(InRoot$$Sections)                ; 标准库的初始化代码
    .ANY (+RO)                         ; 只读数据（代码和常量）
  }
  RW_RAM1 0x24000000 0x00020000   {   ; 读写数据区域
    .ANY (+RW +ZI)                     ; 读写数据和零初始化数据
  }
}
```

4.3.1. 加载域（Load Region）

加载域定义了程序在加载时的内存布局。通常，加载域对应于 Flash 或其他非易失性存储器。加载域的起始地址和大小根据硬件配置指定。

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

load_region_name：为本加载域的名称，名称可以按照用户意愿自己定义；

加载域的起始地址可以选择下面两种格式：

base_address：表示本加载域中的对象在链接时的起始地址，地址必须是字对齐的；

+offset：表示本加载域中的对象在链接时的起始地址是在前一个加载域的结束地址后偏移量 offset 字节处。本加载域是第一个加载域，则它的起始地址即为 offset，offset 的值必须能被 4 整除。

attribute_list：指定本加载域内容的属性，包含以下几种，默认加载域的属性是 ABSOLUTE。

- ABSOLUTE：绝对地址；

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

- PI：与位置无关；
- RELOC：可重定位；
- OVERLAY：覆盖；

NOCOMPRESS：不能进行压缩。

max_size：指定本加载域的最大尺寸。如果本加载域的实际尺寸超过了该值， 链接器将报告错误，默认取值为 0xFFFFFFFF；

execution_region_description：表示运行域，后面有个+号，表示其可以有一个或者多个运行时域，关于运行域的介绍请看后面。

关于加载域更多详情参考：Keil help/Arm Compiler 5 User's Guides/Linker User Guide /Scatter File Syntax/Load region descriptions

4.3.2. 执行域（Execution Region）

执行域定义了程序在运行时的内存布局。执行域可以分布在不同的内存块中，如内部 RAM、外部 RAM 等。每个执行域可以包含多个代码段和数据段。

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

}
```

exec_region_name：为为本执行域的名称，名称可以按照用户意愿自己定义；

执行域的起始地址可以选择下面两种格式：

base_address：表示本执行域中的对象在链接时的起始地址，地址必须是字对齐的；

+offset：表示本执行域中的对象在链接时的起始地址是在前一个执行域的结束地址后偏移量 offset 字节处，offset 的值必须能被 4 整除。

attribute_list：指定本执行域内容的属性：

- ABSOLUTE：绝对地址；
- PI：与位置无关；
- RELOC：可重定位；
- OVERLAY：覆盖；

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

FIXED：固定地址。加载地址和执行地址都是由基址指示符指定的，基址指示符必须是绝对基址，或者偏移为 0。

ALIGN alignment：alignment=4、8、16…可将执行域的对齐约束从 4 增加到 alignment。alignment 必须为 2 的正数幂。如果执行域具有 base_address，则它必须为 alignment 对齐。如果执行区具有 offset，则链接器将计算的区基址与 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。
- ZEROPAD：零初始化的段作为零填充块写入 ELF 文件，因此，运行时无需使用零进行填充；
- PADVALUE：定义任何填充的值。如果指定 PADVALUE，则必须为其赋值；
- NOCOMPRESS：不能进行压缩；
- UNINIT：未初始化的数据。

max_size：指定本执行域的最大大小。如果本执行域的实际大小超过了该值，链接器将报告错误，默认取值为 0xFFFFFFFF；

length：如果指定的长度为负值，则将 base_address 作为区结束地址。它通常与 EMPTY 一起使用，以表示在内存中变小的堆栈。

input_section_description：指定输入段的内容。

关于执行域更多详情请参考 Keil help/Arm Compiler 5 User's Guides/Linker User Guide/ Scatter File Syntax/Execution region descriptions

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

4.3.3. 输入段（Input Section）

段是分散加载文件中的最小单位，用于描述代码或数据的具体分配。

```
module_select_pattern [ "(" input_section_selector ( "," input_section_selector )* ")" ]
( " + " input_section_attr| input_section_pattern| input_symbol_pattern| section_properties)
```

module_select_pattern：支持使用通配符“*”与“?”。其中符号“*”匹配零个或多个字符，符号“？”匹配任何单个字符。进行匹配时所有字符不区分大小写。

当 module_select_pattern 与以下内容之一相匹配时，输入段将与模块选择器模式相匹配：

- 包含段和目标文件的名称；
- 库成员名称（不带前导路径名）；
- 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。

input_section_attr：属性选择器。每个 input_section_attr 的前面有一个“+”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 点的段。
- 可以识别以下同义词：
- CODE 表示 RO-CODE；
- CONST 表示 RO-DATA；
- TEXT 表示 RO；

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

- DATA 表示 RW；
- BSS 表示 ZI。

可以识别以下伪属性：

- FIRST；
- LAST。

使用 FIRST 和 LAST 去标记第一和最后一个段，放置顺序是很重要的，例如，特定输入部分必须位于第一个段和校验和必须在最后一个段。

可以使用一个或多个.ANY 模式以任意分配方式填充运行时域。在大多数情况下，使用单个.ANY 等效于使用*模块选择器。

关于输入段更多详情请参考 Keil help/Arm Compiler 5 User's Guides/Linker User Guide/ Scatter File Syntax/Input section descriptions

4.4. 分散加载应用实例

4.4.1. 普通的分散加载配置

假设只使用 N32H7xx 的 flash 和 AXISRAM，

Flash 基址：0x15000000，大小：0x1E0000 Byte；

RAM 基址：0x24000000，大小：0x20000 Byte。

分散加载文件配置如下：

```
LR_ROM1 0x15000000 0x1E0000  {      ; 加载区域定义
  ER_ROM1 0x15000000 0x1E0000  {  ; 执行区域定义
    *.o (RESET, +First)                ; 复位向量表
    *(InRoot$$Sections)                ; 标准库的初始化代码
    .ANY (+RO)                         ; 只读数据（代码和常量）
  }
  RW_RAM1 0x24000000 0x20000  {    ; 读写数据区域
    .ANY (+RW +ZI)                     ; 读写数据和零初始化数据
  }
}
```


国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

4.4.2. 多块 RAM 的分散加载文件配置

假设需要使用 N32H7xx 的 flash、AXISRAM 和 AHBSRAM，

Flash 基址：0x15000000，大小：0x1E0000 Byte；

AXIRAM 基址：0x24000000，大小：0x20000 Byte。

AHBRAM 基址：0x30000000，大小：0x20000 Byte。

分散加载文件配置如下：

```
LR_ROM1 0x15000000 0x1E0000  {      ; 加载区域定义
  ER_ROM1 0x15000000 0x1E0000  {  ; 执行区域定义
       *.o (RESET, +First)              ; 复位向量表
       *(InRoot$$Sections)              ; 标准库的初始化代码
       .ANY (+RO)                       ; 只读数据（代码和常量）
  }
  RW_RAM1 0x24000000 0x20000  {   ; 读写数据区域 1
       .ANY (+RW +ZI)                   ; 读写数据和零初始化数据
  }
  RW_RAM2 0x30000000 0x20000  {   ; 读写数据区域 2
       .ANY (+RW +ZI)                   ; 读写数据和零初始化数据
  }
}
```

4.4.3. 将某个函数/变量/数组指定到 SRAM 中执行

基于 5.2 的场景，需要给将一个函数/变量/数组指定到 AXIRAM 中，有 2 种方法：

- **__attribute__((setion(“mySectionCode”)))**

首先在分手加载文件中定义 section,分散加载文件配置如下：

```
LR_ROM1 0x15000000 0x1E0000  {      ; 加载区域定义
  ER_ROM1 0x15000000 0x1E0000  {  ; 执行区域定义
    *.o (RESET, +First)             ; 复位向量表
    *(InRoot$$Sections)             ; 标准库的初始化代码
```


国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

```
    .ANY (+RO)                          ; 只读数据（代码和常量）
  }
  RW_RAM1 0x24000000 0x20000   {    ; 读写数据区域 1
    .ANY (+RW +ZI)                      ; 读写数据和零初始化数据
    *(mySectionCode)                        ; mySectionCode 段定义
    *(mySectionData)                        ; mySectionData 段定义
  }
  RW_RAM2 0x30000000 0x20000   {    ; 读写数据区域 2
    .ANY (+RW +ZI)                      ; 读写数据和零初始化数据
  }
}
```

C 语言中将函数指定到 mySectionCode，代码如下：

```
__attribute__((section("mySectionCode "))) void myFunction(void) { }
```

将变量指定到 mySectionData，代码如下：

```
int myVariable __attribute__((section("mySectionData"))) = 42;
```

将数组指定到 mySectionData，代码如下：

```
uint8_t myTable[256] __attribute__((section("mySectionData "))) = {};
```

注：在使用 keil 编译器版本 6，变量/函数不能使用同一个 section 名，而编译器版本 5 可以。

__attribute__((section(".ARM.__at_0x24000000")))

不用在分散加载文件中定义 section，直接设置函数/变量/数组到指定地址，C 代码实现如下：

```
int myVariable __attribute__((section(".ARM.__at_0x24004000"))) = 100;

uint8_t myTable[256] __attribute__((section(".ARM.__at_0x24008000")))={};

__attribute__(( section(".ARM.__at_0x15008000"))) void myFunction(void) {        }
```


国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

4.4.4. 部分程序拷贝到 ITCM 中执行的分散加载文件配置

假设需要使用 N32H7xx 的 flash、AXISRAM 和 ITCM，由于 ITCM 的执行速度比 flash 快，希望用户代码在 ITCM 中执行。

Flash 基址：0x15000000，大小：0x1E0000 Byte；

AXIRAM 基址：0x24000000，大小：0x20000 Byte。

ITCM 基址：0x00000000，大小：0x20000 Byte。

必须放在 Flash 里的程序有如下几个：

复位向量表（Reset Vector Table）：是系统启动时首先执行的部分

标准库的初始化代码:包含__main(), __rt_entry(),从 FLASH 拷贝程序到 ITCM 是由__main 执行的所以__main 及__main 之前执行的代码肯定也只能在 flash 里执行.

startup_n32h76x.o 文件: __main 和 systemInit 都是在 Reset_Handler 中调用的，都是在启动文件中。

system_n32h7xx.o 文件:进入__main 之前还执行了 systemInit()。

分散加载文件配置如下：

```
LR_ROM1 0x15000000 0x1E0000  {      ; 加载区域定义
  ER_ROM1 0x15000000 0x1E0000  {  ; 执行区域定义
    *.o (RESET, +First)                    ; 复位向量表
    *(InRoot$$Sections)                    ; 标准库的初始化代码，包含__main,__rt_entry
    startup_n32h76x.o (+RO)                ; 启动文件只读数据（代码和常量）
    system_n32h7xx.o (+RO)                 ; 系统文件只读数据（代码和常量）
  }
  ER_ITCM 0x00000000 0x20000  {    ; ITCM 执行区域
    .ANY (+RO)                             ; 只读数据（代码和常量）
  }
  RW_RAM1 0x24000000 0x20000  {    ; 读写数据区域 1
    .ANY (+RW +ZI)                         ; 读写数据和零初始化数据
  }
}
```

注意：*栈不能放在 DTCM.*

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

版本历史

版本	日期	备注
V1.0.0	2025.12.5	初始版本
V1.0.1	2025-08-21	1，修改页眉的logo
V1.1.0	2026-01-05	1，修改页眉的logo 2，将N32H76x_78x型号修改为N32H7xx 3，优化OPT配置表，区分N32H76x/78x与N32H73x的区分

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057

声明

国民技术股份有限公司（下称“国民技术”）对此文档拥有专属产权。依据中华人民共和国的法律、条约以及世界其他法域相适用的管辖，此文档及其中描述的国民技术产品（下称“产品”）为公司所有。

国民技术在此并未授予专利权、著作权、商标权或其他任何知识产权许可。所提到或引用的第三方名称或品牌（如有）仅用作区别之目的。

国民技术保留随时变更、订正、增强、修改和改良此文档的权利，恕不另行通知。请使用人在下单购买前联系国民技术获取此文档的最新版本。

国民技术竭力提供准确可信的资讯，但即便如此，并不推定国民技术对此文档准确性和可靠性承担责任。

使用此文档信息以及生成产品时，使用者应当进行合理的设计、编程并测试其功能性和安全性，国民技术不对任何因使用此文档或本产品而产生的任何直接、间接、意外、特殊、惩罚性或衍生性损害结果承担责任。

国民技术对于产品在系统或设备中的应用效果没有任何故意或保证，如有任何应用在其发生操作不当或故障情况下，有可能致使人员伤亡、人身伤害或严重财产损失，则此类应用被视为“不安全使用”。

不安全使用包括但不限于：外科手术设备、原子能控制仪器、飞机或宇宙飞船仪器、所有类型的安全装置以及其他旨在支持或维持生命的应用。

所有不安全使用的风险应由使用人承担，同时使用人应使国民技术免于因为这类不安全使用而导致被诉、支付费用、发生损害或承担责任时的赔偿。

对于此文档和产品的任何明示、默示之保证，包括但不限于适销性、特定用途适用性和不侵权的保证责任，国民技术可在法律允许范围内进行免责。

未经明确许可，任何人不得以任何理由对此文档的全部或部分进行使用、复制、修改、抄录和传播。

国民技术股份有限公司 NSING TECHNOLOGIES INC.
地址：深圳市南山区高新北区宝深路109号国民技术大厦
电话：+86-755-86309900 传真：+86-755-86169100
网址：https://www.nsingtech.com 邮编：518057