文档编号：MBOOTELFTOSBUG
第 7 版，2020 年 2 月

elftosb 用户指南

作者：恩智浦半导体

目录

1 概述

elftosb 工具可创建二进制输出文件，包含用户应用程序镜像和一系列引导加载程序命令。输出文件称为“安全二进制”或 SB 文件。这些文件的扩展名通常为.sb。该工具使用输入命令文件来控制输出文件中的引导加载程序命令序列。该命令文件称为“启动描述符文件”或简称为 BD 文件。

elftosb 是一个命令行驱动的工具，可以单独构建，以在 Windows®操作系统、Linux®操作系统和 Apple Mac® 操作系统上运行。目前，Mac 操作系统中的 elftosb 工具只能支持 i.MX 器件的非安全启动镜像，因为 Mac 操作系统不支持代码签名工具(CST)。MCU 引导加载程序软件包内含全部三种目标操作系统的可执行文件。

本文档从命令行参数、输入命令文件(.bd)结构和输出(.sb)文件内容的角度介绍 elftosb 的用法。下面的功能框图概述了 elftosb 操作。Elftosb 实用工具使用三种输入（输入文件(elf/srec/binary)、密钥文件和 BD 文件）来处理 BD 文件的内容，以生成输出 SB 文件。

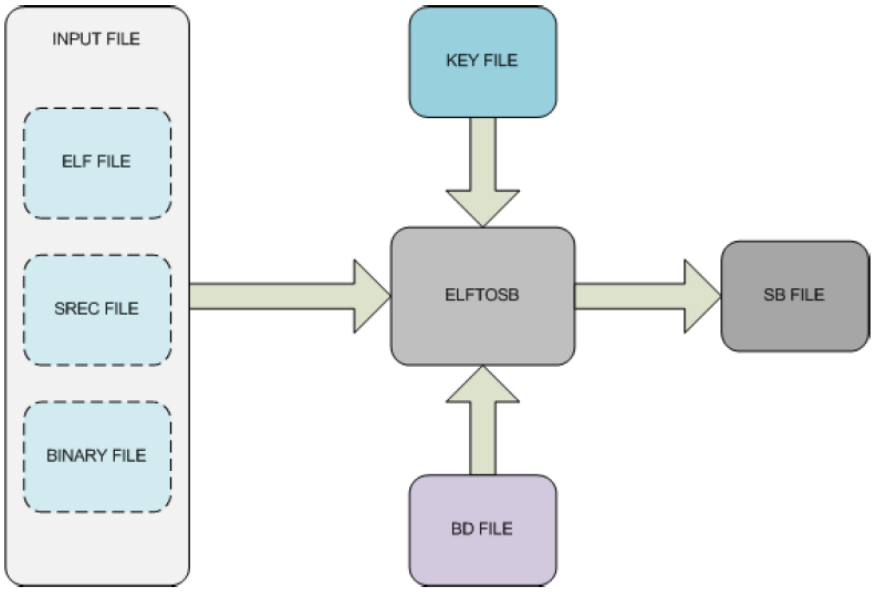

图 1. elftosb 功能框图

2 命令行界面

elftosb 提供一组命令行选项，如下表所列。表中仅列出了与本文档所述内容直接相关的选项。

表 1. 命令行选项

选项	说明
-p PATH, --search-path PATH	在搜索路径列表的末尾添加一条路径
-f CHIP, --chip-family CHIP	选择输出启动镜像格式。如需生成 Kinetis 器件的启动镜像，则指定为“kinetis”
-c FILE, --command FILE	指定要使用的命令文件。这是必选项
-o FILE, --output FILE	设置输出文件路径。这是必选项
-P VERS, --product VERS	设置产品版本
-C VERS, --component VERS	设置器件版本
-k FILE, --key FILE	添加密钥文件并启用加密
-z, --zero-key	添加一个全零密钥并启用加密
-D NAME=INT, --define NAME=INT	覆盖或设置常量值
-O OPTION=VALUE, --option NAME=VALUE	设置全局选项值
-V, --verbose	打印详细的输出
-q, --quiet	只打印警告和错误
-d, --debug	启用调试输出
-v, --version	显示工具版本并打印支持的器件系列列表

表格接下页……

表 1. 命令行选项（续）

选项	说明
-?, --help	显示用法信息
-K/--keygen <option>	根据选项值<128|256>（默认值为<128>）生成 AES-128 或 AES-256 密钥文件
-n/--number <int>	每个文件生成的密钥个数（默认值=1）（仅在指定-K 时有效）
-x/--extract/--sbtool	提取指定段
-i/--index <int>	要提取的段索引（默认值=没有段索引）（仅在指定-x 时有效）
-b/--binary	以二进制形式提取段数据。仅在指定-x 时有效。警告：如果指定了-b，则隐式启用-q
-s/--pkey <file>	签名私钥的路径
-S/--cert <file>	签名证书文件的路径。第一个证书将是自签名的根密钥证书
-R/--root-key-cert <file>	用于验证其他证书的根密钥证书文件的路径。只允许 4 个根密钥证书，其他证书都将忽略。其中一个证书必须与通过-S/--cert arg 传递的第一个证书相匹配
-h/--hash-of-hashes <file>	根密钥哈希值的输出路径。如果未提供参数，则该工具会默认在工作目录中创建 hash.bin
-J/--image-conf <file>	使用此 json 镜像配置文件生成主启动镜像（仅适用于 kinetisk3、k32w0x、lpc55xx 和 rt6xx 系列）
-T/--tzm-conf <file>	使用此 json trust zone 配置文件生成 trust zone 二进制配置文件（仅适用于 lpc55xx 和 rt6xx 系列）

设置命令文件和输出文件路径所需的两个命令行选项为

-c FILE, --command FILE

-o FILE, --output FILE

必须定义这些选项。

-f 或--chip-family 选项决定 elftosb 实用工具将采用哪种输出(.sb)文件格式。如需生成 Kinetis 器件的启动镜像，则指定为“kinetis”。比较芯片系列名称时，将忽略大小写。

默认情况下，输出启动镜像未加密。要加密启动镜像，需提供一个或多个密钥。使用-z 选项添加全零密钥。如果尚未烧写硬件密钥，则这是芯片中硬件密钥的默认状态。

一个非常有用的选项是-D 或--define。该选项用于设置或覆盖常量值。该选项的参数是一个标识符和一个整数值，以等号隔开。常量名标识符可以是命令文件中允许的任何常量名。其值可以是命令文件中允许的任何整数值，但多字符整数文字除外。

在生成输出启动镜像之前，所有通过-D 或--define 选项设置的常量都在 elftosb 内部的表达式名称空间中进行设置。这些特定常量将覆盖所有在命令文件中指定的同名常量。这使用户可以在命令文件中设置一个常量默认值，并在每次 elftosb 调用时覆盖它。

-O 或--option 选项与-D 类似，让用户可以使用命令行设置或覆盖全局选项设置。其参数值同样是一个选项名称，并且值由等号分隔。该值可以是命令文件中允许的任何整数或字符串值，但多字符文字除外。

如需提取段内容，请使用-x/--extract/--sbtool 选项。可以选择通过-i/--index 选项传递所需段的索引。段索引打印在输出中的“Section table”标头下。-x 选项可以将段内容的十六进制转储与“Sections”标头下的输出一起打印。如果在命令行中传递了附加选项-b/--binary，则该段的二进制内容将回显到 stdout，从而使用户可以轻松地将数据重定向到文件。在该模式下，不会生成其他输出。在所有情况下，段内容都将解密后显示。

如需以第 4 章“elftosb 密钥文件格式”中所述的格式生成随机 AES-128 或 AES-256 密钥文件，请使用-K/--keygen <128|256>选项。

将-s/--pkey <file>、-S/--cert <file>、-R/--root-key-cert <file>和-h/--hash-of-hashes <file>开关与 SB2 文件生成相连接。用法说明参见附录 C。

如需为 kinetis k3、k32w0x、lpc55xx、lpc55s1x、rt5xx 和 rt6xx 系列生成主启动镜像，请使用-J/--image-conf <file>开关。更多详细信息，请参见附录 D。

使用-T/--tzm-conf <file>开关为 lpc55xx、lpc55s1x，rt5xx 和 rt6xx 系列创建 TrustZone-M 预设二进制配置。用法说明参见附录 E。

elftosb 工具的命令行用法如下：

```
elftosb [-?|--help] [-v|--version] [-f|--chip-family <family>]
            [-c|--command <file>] [-o|--output <file>]
            [-P|--product <version>] [-C|--component <version>]
            [-k|--key <file>] [-z|--zero-key] [-D|--define <const>]
            [-O|--option <option>] [-K|--keygen <option>] [-n|--number <int>]
            [-x|--extract] [-x|--sbtool] [-i|--index <int>] [-b|--binary]
            [-d|--debug] [-q|--quiet] [-V|--verbose]

            [-s|--pkey <file>]

            [-S|--cert <file>]

            [-R|--root-key-cert <file>]

            [-h|--hash-of-hashes <file>]

            [-J|--image-conf <file>]

            [-T|--tzm-conf <file>]
            [-p|--search-path <path>]
            files...
```

注

- 选项（无论使用短格式还是长格式）和任何值之间都必须有一个空格
- 选项后面列出的所有参数都是供 extern()语法使用的位置源文件（参见第 3.1.1.3 节“源”）。

3 命令文件

命令文件是前 128 个字符使用 ASCII 格式的任何编码形式（包括 UTF-8）的文本文件。任何行尾格式都可以，还支持 Unix、DOS 和 Mac 操作系统的行尾，甚至可以接受混合式行尾。

命令（启动描述符）文件的标准扩展名为*.bd*。

elftosb 命令文件的功能类似于链接器命令文件。它根据输入文件来描述输出(sb)文件。elftosb 命令文件支持 ELF、S-record 和二进制输入文件。命令文件既可以显式声明输入文件的路径，也可以由用户在命令行中提供路径。这一特性使命令文件具有通用性并可重复使用。

命令文件声明多个源文件，为每个源文件分配易于引用的唯一名称。每个源可以显式调用其文件路径，也可以由用户在命令行中提供路径。当用户在命令行中输入路径时，该路径会指向一个文件，该文件可在每次调用 elftosb 时进行更改。

然后命令文件将定义输出(.sb)文件中所需的段。其中每一段都提供了一个操作序列（例如加载和调用）的定义，该操作将引用源文件的内容或位于命令文件中的常量值。这些操作将被映射到引导加载程序命令。

3.1 块

命令文件由不同的块组成：选项、常量、源、keyblob 和段。所有块均为可选项，如有需要，每种类型的块可以不止一个。唯一的规则是所有*段*块都必须位于所有其他块类型之后。块语法如下所示。

示例 1.基本块语法

```
# define the options block
options
{
     # content goes here
}
```

3.1.1 块语法

块在命令文件中分为两组。第一组是配置块：选项、常量、源和 keyblob。所有配置块类型都是可选项，但是一个有效的命令文件至少需要有一个*源*块。

段定义块位于配置块之后。命令文件中可以包含一个以上的段块。它们在命令文件中的字典顺序决定了输出启动镜像中各节的逻辑顺序。

3.1.1.1 选项

*选项*块包含零个或多个名称/值对以及选项设置，通过这些选项设置为全局选项赋值，供 elftosb 用来控制输出文件的生成。

选项块中的每个条目都采用以下形式：

```
option_def ::= IDENT '=' const_expr
  ;
const_expr ::= bool_expr
  | STRING_LITERAL
  ;
```

在块内部，每个选项定义都必须以分号结尾。选项的值可以是字符串、整数或布尔表达式。可接受的值取决于特定选项。

选项名称由 elftosb 实用工具预定义，不能在命令文件中用于任何其他目的。但是，某个源可能会与其中一个选项同名。下表列出了可用选项的完整列表。

表 2. elftosb 的选项名称

选项名称	适用于	说明
alignment	段	按 2 的幂数取整对齐作为启动镜像段的开始
cleartext	段	整数布尔值。使段保持未加密状态，即使节在加密镜像中亦是如此
componentVersion	启动镜像	版本字符串为“xxx.yyy.zzz”
driveTag	启动镜像	用于设置镜像标头的驱动标签字段的整数值
flags	启动镜像	用于整个镜像范围标志的整数值
productVersion	启动镜像	版本字符串为“xxx.yyy.zzz”
secinfoClear	GHS ELF 源文件	“default”、“ignore”、“rom”或“c”，其中“default”等于“c”
sectionFlags	段	用于为启动镜像段设置标志或与隐式标志一起使用的整数值
toolset	ELF 源文件	“GHS”、“GCC”或“ADS”
secureBinaryVersion	启动镜像	用于指定 SB2 文件的版本。预期值：“2.0”和“2.1”。如果未指定，则使用“2.0”作为默认值

两个版本选项用于设置默认的产品和组件版本号。可以使用命令行覆盖这两个版本值。

flags 选项用于在启动镜像文件的标头中设置标志字段。有关此字段的可能值，请参见介绍启动镜像格式的附录。除在启动镜像节标头中设置标志字段外，这也同样适用于 *flagsoption* 节。

3.1.1.2 常量

与选项块类似，常量块包含零个或多个常量定义语句的序列，以分号结尾。每个常量定义语句就是一个名称/值的赋值。

常量定义语法如下所示：

```
constant_def ::= IDENT '=' bool_expr
  ;
```

当在另一个表达式中使用常量时，常量值将保留整数字长。

在前面常量块中定义的常量可用于其后常量的定义，如以下示例所示。

示例 2：常量块

```
# this is an example constants block
constants {
    ocram_start = 0;
    ocram_size = 256K;
    ocram_end = ocram_start + ocram_size -- 1;
}
```

3.1.1.3 源

源块用来列出输入文件并为其分配标识符，以便在命令文件的其他部分中引用。源块中的每个语句都包含一个赋值运算符（“=”字符），其左边是源名称标识符，右边是源路径值。每个源定义都以分号结尾。

源值的语法取决于源定义类型。两种类型分别为显式路径和由外部提供的路径。使用显式路径的源以带引号的字符串文字形式列出文件路径。

外部源使用整数表达式从命令行中选择一个位置参数。它允许用户通过更改命令行参数来更改输入文件。

源定义的语法如下：

```
source_def ::= IDENT '=' source_value ( '(' source_attr_list? ')' )?
  ;
source_value ::= STRING_LITERAL
  | 'extern' '(' int_const_expr ')'
  ;
source_attr_list
  ::= source_attr ( ',' source_attr )*
  ;
source_attr ::= IDENT '=' const_expr
  ;
```

源定义也可以是在定义末尾括号中的源属性列表。这些属性与选项块中的选项相同。但是，只有几个选项适用于源。有关选项的完整列表，请参见表 2。

3.1.1.4 Keyblob

keyblob 块必须在命令文件中的段块类型之前定义。keyblob 块必须在 keywrap 语句中被引用才有用。keyblob 块的语法如下所示。

```
keyblob_block    ::= 'keyblob' '(' int_const_expr ')' keyblob_contents
                 ;
keyblob_contents
                 ::= '{' ( '(' keyblob_options_list ')' )* '}'
                 ;
keyblob_options_list
                 ::= keyblob_option ( ',' keyblob_option )*
                 ;
keyblob_option   ::= ( IDENT '=' const_expr )?
                 ;
```

如果选项列表为空，则只分配但不填充相应的 keyblob 条目。支持的 keyblob 选项标识符包括：

- start：加密区域的起始地址
- end：加密区域的结束地址
- key：区域的 AES-128 计数器模式加密密钥
- counter：区域的初始计数器值
- noByteSwap：用于在 elftosb 中基于所用闪存生成数据的选项。1——FLEX-SPI，0——QUAD-SPI（默认值：0）

Keyblob 块

```
keyblob (0)
{
 (
      start=0x68001000,
      end=0x68001fff,
      key="00112233445566778899AABBCCDDEEFF",
      counter="0011223344556677",
      noByteSwap = 0
  )
  ()
  ()
  ()
}
```

注

底层硬件必须支持 keyblob 块中出现的区域地址。例如，可能不支持别名地址。有关详细信息，请参见相应芯片的参考手册。

3.1.1.5 段

*段*块必须在命令文件中的其他块类型之后定义。每个段块直接对应于在输出(.sb)文件中创建的段。*段*块的语法具有段的唯一标识符值和特定于该段的选项，如下所示。

第 3.12 节“语句”中详细介绍了非终端语句。

```
section_block ::= 'section' '(' int_const_expr section_options? ')' section_contents
  ;
section_options ::= ';' section_option_list
  ;
section_option_list
  ::= source_option ( ',' source_option )*
  ;
source_option ::= IDENT '=' const_expr
  ;
section_contents
  ::= '{' statement* '}'
  | '<=' SOURCE_NAME ';'
  ;
```

如下例所示，段内容有两种形式。

第一种是包含在括号内的一系列语句。可以使用括号内的一系列引导加载程序命令来创建可启动段。第 3.12 节“语句”中详细介绍了可启动段中的语句语法。

第二种是创建一个任意的数据段。所列源文件的原始二进制内容被复制到输出文件的该段。没有预定义的格式或数据段。数据段可用于保存资源文件或用于虚拟内存分页的后备存储。

为 MCU ROM 创建的输出*(.sb)*文件必须以可启动段开始。ROM 在此可启动段的末尾停止处理工作。将忽略其他可启动段和数据段。

示例 4：两种段块

```
# create a bootable section
section (32)
```

```
{
     # statements...
}
# create a data section
section (64) <= my_source_file;
```

对于该段而言，出现在括号中的段标识符数字必须是唯一的。如果两个段的标识符相同，则会报告错误。

如需设置仅适用于单个段的选项，则在段的唯一标识符后插入选项，以分号隔开。在上述*段*语法中，选项使用 section_options 非终端描述。如果有多个选项，则选项之间用逗号分隔，如*选项*块中所示。

如需查看适用于输出文件中各段的选项，请参见表 2。

alignment

此选项将 2 的整数幂作为其值。在输出*(.sb)*文件中，确保对具有特殊对齐方式的段的第一个字节的偏移可以被对齐值整除。将忽略小于等于 16 的对齐，因为这是输出(.sb)文件的加密块大小所保证的最小对齐。请注意，对齐的是段本身，而非该段的启动标签。任何为了对齐段而插入的填充都由“nop”引导加载程序命令组成。

cleartext

将此选项设置为布尔值。接受关键字“yes”、“no”、“true”和“false”，以及任何取值为零或非零的整数表达式。默认值为 false。如果输出文件已加密且 cleartext 选项为 true，则应用 cleartext 选项的段将保持未加密状态。但是目前 ROM 不支持加密文件中的未加密可启动段。此选项对数据段最有用。

对于所有选项，均可使用选项块进行全局设置，而不必对每个段进行单独设置。用户也可以设置全局默认值，并使用特定于段的选项覆盖该值。例如，将默认段对齐设置为 2 K，然后将一个特定段对齐设为 4 K。

段在输出(.sb)文件中的创建顺序始终依照它们在命令文件中出现的顺序。此外，命令文件中定义的第一个可启动段将成为引导加载程序在检查输出(.sb)文件头之后首先开始处理的段。

3.2 词汇元素

本小节介绍命令文件中的各种文本组件及其语法与用法。阅读以下各小节时，如需了解命令文件中令牌的用法，请参照下表。

表 3. 令牌值示例

令牌	说明
10000	整数文字。
0x200	值为 512 的整数。
256K	值为 262144 的整数。
0b001001	值为 9 的整数。
'q'	字节大小的整数，值为 0x71 或 113。
'dude'	字大小的整数，值为 0x64756465 或 1685415013。
"this is a test"	字符串文字。
$.text	匹配“.text”的段名称。

表格接下页……

表 3. 令牌值示例（续）

令牌	说明
$*	匹配所有段的段名称。
$*.bss	匹配所有.bss 段的另一个段名称，例如“.sdram.bss”。
appElfFile:main	显式源文件的符号引用。
:printMessage	使用默认源文件的符号引用。
{{ 01 02 03 0b }}	一个四字节长的二进制对象。

3.3 空格

在整个命令文件中，将忽略空格字符、制表符、换行符或回车符形式的空格，字符串内的空格除外。允许任何形式的行尾。

3.4 关键字

下表列出了 elftosb 命令文件中使用的所有关键字。这些标识符不能用作源文件或常量名称。有一些关键字预留给未来功能，目前尚未使用。

表 4.命令文件关键字

call	**no**
constants	options
extern	raw
false	section
filters	sources
from	switch
jump	true
load	yes
mode	if
else	defined
info	warning
error	sizeof
qspi	unsecure
ifr	jump_sp
enable	keyblob
start	end
key	counter

表格接下页……

表 4.命令文件关键字（续）

keywrap	reset
all	encrypt

3.5 备注

单行注释可在行中的任何位置使用井号（“#”）或两个斜杠（“//”）引入，直到行尾。

多行注释的方式与 ANSI C 中的方式相同。

它们以“/*”开始，以“*/”结束。

此外，与 ANSI C 一样，不支持多行注释嵌套。

3.6 标识符

标识符用于表示选项名称、常量和源名称。它们遵循与 ANSI C 类似的标识符规则。它们允许以下划线或字母字符开头，可以包含任意数量的下划线和字母数字字符。

3.7 整数

整数文字采用以下三种支持制式之一：二进制、十进制或十六进制。十进制整数无前缀。十六进制整数必须以“0x”为前缀，二进制整数必须以“0b”为前缀。

整数文字后可以跟一个公制计量乘数字符：“K”、“M”或“G”。最后一位数字和乘数之间允许使用空格字符。使用的是二进制乘数值，而不是标准公制计量乘数。这意味着“K”表示整数乘以 1024，“M”表示乘以 1048576，“G”则表示乘以 1073741824。不可使用小写的“k”、“m”或“g”。

命令文件中的所有整数值都是无符号的，并且具有相应的大小。支持的整数大小包括字节（8 位）、半字（16 位）和字（32 位）。默认整数文字均为字大小的值。如需对字大小进行更改，可在表达式中使用“字长”运算符。

整数常量也可以使用单引号内的字符序列来创建。允许包含一个、两个或四个字符序列。它们分别对应于大小为字节、半字和字的整数。例如，“oh”等于一个值为十六进制 0x6f68（对应 ASCII 中字符“o”和“h”的值）或十进制 28520 的半字。

有几个关键字保留用作布尔值的内置整数常量。它们是“yes”、“no”、“true”和“false”。关键字“yes”和“true”的计算结果为 1，而关键字“no”和“false”的计算结果为 0。这些关键字可以用在任何接受整数值的地方，包括命令行。

3.7.1 整数表达式

整数表达式可以用在等式中需要整数常量值的任何地方。这些表达式（大多数）遵循标准 C 语言的表达式，但也有一些例外。下表列出了可用的运算符。

表 5. 整数表达式运算符

运算符	说明
+	加
-	减

表格接下页……

表 5. 整数表达式运算符（续）

运算符	说明
*	乘
/	除
%	取模
&	按位与
|	按位或
A	按位异或
<<	逻辑左移
>>	逻辑右移
.	设置整数大小
sizeof()	获取常量或符号的大小

除了上表中列出的运算符外，还支持单目加和单目减。

3.7.1.1 运算符优先级

下表按优先级分组列出了表达式运算符。表中第一行的优先级最低，最后一行的优先级最高。

表 6. 按升序排列的运算符优先级

运算符	说明
1	按位或
A	按位异或
&	按位与
<< >>	逻辑左移、逻辑右移
+ -	加、减
* / %	乘、除、取模
.	字大小
unary + -	单目加和单目减

3.7.1.2 字长运算符

整数大小运算符（“.”）由一个句点后跟字符“w”、“h”或“b”之一组成。这些字符区分大小写。在句点和下一个字符之间允许使用空格。此运算符更改其左边表达式的字长。“w”字符将大小设置为 32 位字，“h”设置为 16 位字，“b”则设置为 8 位字。

对于任何给定的二进制运算，得到的字长为两个操作数中的最大字长。因此，将字节大小的整数乘以半字大小的整数得到半字。实际运算始终以 32 位字的形式执行，必要时结果将被截断。

3.7.1.3 sizeof 运算符

sizeof 运算符用于检查符号或常量的大小。运算符的语法是关键字“sizeof”后跟括号内的符号引用或常量标识符（与 ANSI C 不同）。大小始终是 32 位值。

3.7.1.4 常量引用

除了整数文字外，表达式也可以使用常量名称引用在常量块中定义的常量。常量名称是标准标识符。在表达式中放置常量名称相当于插入该常量的整数值。尽管源与常量共享相同的名称空间，但是它们不能在整数表达式中使用。

3.7.1.5 符号引用

就像常量一样，符号引用也可以用在整数表达式中。符号引用的数值为符号在 ELF 文件中的值，是一个 32 位值。通常，符号的值是其地址，尽管某些特殊符号可以是其他值。如果源文件中不存在引用的符号，则符号引用的值为 0。

3.7.2 布尔表达式

在为 if 和 else-if 语句（参见第 3.12.6 节“If-Else”）定义常量、选项或条件时，使用布尔表达式。布尔表达式不能作为 load 语句的源或目标。

表 7.布尔表达式运算符

运算符	说明
&&	布尔和
ll	布尔或
<	小于
>	大于
< =	小于等于
> =	大于等于
==	等于
!=	不等于
exists(src_file)	源文件是否存在？
defined(const)	是一个常量吗？

在布尔表达式中可以使用许多新的运算符。除了上述运算符之外，还支持单目非运算符（或字符“!”）。所有这些运算符的计算结果为 0 或 1。像在 ANSI C 中一样，值 0 表示 false，任何非零值表示 true。

有两个类似函数的运算符可在布尔表达式中使用。第一个是“exists()”，如果括号内的命名源文件存在于磁盘上并已成功打开，则返回 true。在 exists 运算符内放入未在源块中定义的源名称将导致语法错误。

第二个特殊运算符是“defined()”。它使用括号内的常量名称。如果在启动描述符文件或命令行中已为命名常量赋值，则运算符的值为 true。

&&和||双目运算符是短路运算符。这意味着，如果左边操作数等于一个使右边操作数无关紧要（因为表达式不管怎样都具有相同的最终值）的值，则不会计算右边的操作数。这在诸如“if defined(const) && const > 10...”这样的表达式中特别有用。此时，仅在定义了常量“const”时才会计算右边的大于表达式。如果始终计算右边表达式且“const”未定义，则将报告错误。

3.8 字符串

所有字符串文字都包含在双引号字符中。它们不能超过行尾。不支持 C 语言将反斜杠用作转义序列的语法，所以反斜杠字符可以用于文件路径中。因此，不能在字符串中间插入双引号、换行符或其他特殊字符。

3.9 段名称

ELF 文件的命名段使用段名称文字来选择。这些特殊文字从美元符号字符（“$”）开始，一直到不允许出现在段名称中的第一个字符为止。该名称是一个标准的 glob 型表达式，可以匹配任意数量的 ELF 段。可接受的字符包括字母数字、下划线、句点、星号、问号、破折号、插入符号和方括号。其中许多字符只能作为 glob 表达式的一部分使用。

支持的 glob 子表达式如下：

表达式	说明
*	匹配任何字符，在一行中零个或多个。
?	匹配任何单个字符。
[set]	匹配集合中的任何字符。
[^set]	匹配不在集合中的任何字符。

在上面的列表中，[set]为单个字符与范围的任意组合。此范围由通过连字符隔开的两个字符构成：[a-z]包含匹配从“a”到“z”的所有字符。

在 load 语句的段列表中使用时，以波浪号（“~”）字符为前缀的段名称用于反转匹配的 ELF 段的集合。

3.10 符号引用

ELF 格式的源文件中嵌入了一个符号表。符号引用用于通过其名称来引用 ELF 文件中的特定符号。当用于整数表达式中时，符号引用的符号值即为其地址。

符号引用的语法包括一个可选的源文件名，后跟冒号和符号名称。符号名称不放在引号内，用作常规标识符。

如果冒号前无源文件，则该符号来自用“from”语句指定的默认源文件。如果符号引用不在“from”语句的上下文中，则必须提供源文件名。

3.11 二进制对象

二进制对象值（称为“blob”）是构成对象的十六进制字节序列。使用双大括号作为一个 blob 的开始和结束。每两个十六进制字符构成 blob 中的一个字节，并忽略所有空格。十六进制字符不区分大小写。非十六进制字符是非法的，不能用于 blob 中。

3.12 语句

可启动段块中的每个语句都描述了引导加载程序在处理输出*(.sb)*文件时执行的“操作”。各语句对应于在输出文件中创建的至少一个或多个启动命令。elftosb 将解译这些语句，并将其转换为输出文件中的启动命令。

下面的源块中包含 3 个语句。对于下述所有内联示例，假设有以下定义：

```
sources
{
     myElfFile = "app.elf";
     mySRecFile = "utility.s37";
     myBinFile = "data.bin";
}
```

上述代码在启动描述符文件(.bd)中时，由 elftosb 实用工具处理。elftosb 实用工具在同一文件夹中搜索 app.elf、utility.s37 和 data.bin 文件，然后在输出文件中将这些文件引用为 myElfFile、mySRecFile 和 myBinFile。

除“from”和“if-else”语句外，所有语句都必须以分号结尾。

3.12.1 加载(Load)

load 语句用于将数据存储到内存中。该加载命令包括引导加载程序中使用的数据加载、模式填充和 word poke 命令。语句的语法可能很简单，但解译可能非常复杂。换句话说，简短的 load 语句会生成大量的启动命令序列，反之亦然。elftosb 实用工具将 load 语句转换为引导加载程序命令。

加载命令也用于写入闪存。当加载到闪存中时，在加载操作之前必须先擦除待加载区域。有关详细信息，请参见擦除命令。

load 语句的示例代码为：

```
load_stmt ::= 'load' load_data ( '>' load_target )?
  ;
load_data ::= const_expr
  | SOURCE_NAME
  | section_list ( 'from' SOURCE_NAME )?
  ;
section_list ::= section_ref ( ',' section_ref )*
  ;
section_ref ::= ( '~' )? SECTION_NAME
  ;
load_target ::= '.'
  | address_or_range
  ;
address_or_range
  ::= int_const_expr
  | int_const_expr '..' int_const_expr
  ;
```

如上述代码所示，所有 load 语句均以“load”关键字开头。每个 load 语句均由一个数据/数据源和一个目标位置组成。源始终是必需的。目标可以是隐式的，在此示例中，它基于源本身。并非所有源和目标类型的组合都被允许。

在上述代码中，源由 load_data 非终端表示。源可以是整数值、字符串文字、源文件或源文件的一个或多个命名段。这些源会按照源类型产生一个或多个数据段。数据源，即相应的段，可能有一个与之相关的物理存储位置，也可能没有。此存储位置是在内存中默认放置数据的地址范围。

例如，ELF 文件的某一段将链接到某个地址并具有某个长度。两者结合构成该段的自然地址和大小。例如，二进制文件的内容具有自然大小，但没有地址。

load 语句的目标决定了源加载到内存中的地址以及加载长度。对于某些具有自然位置的源类型，目标是可选项，可以不包含在语句中。如需将目标列出，则将其放在源数据和“>”符号之后。如果目标是隐式的（与源相同），则在“>”符号后使用一个圆点（句点）。目标的值是一个地址或地址范围。如果目标只是一个地址，则没有与之相关的长度。在这种情况下，加载长度取决于源数据本身。对 ELF 文件中符号的引用也可以用作加载目标。它们等同于从符号起始地址到其结束地址的地址范围。

如果目标只是一个地址，则将整个数据源加载到该地址。即使源具有自然地址也是如此。例如，这允许用户将 ELF 段加载到与其链接地址不同的地址。

当目标为地址范围或等同于地址范围的符号时，源会被定位并可能被截断。加载地址为目标范围的起始地址。这与只有一个目标地址的工作方式相同。如果数据源的自然大小等于或小于目标范围的大小（结束地址减起始地址），则将加载整个源。在这种情况下，不会修改目标中的剩余字节。一旦源的自然大小大于目标范围，加载源时会将源按目标地址范围的大小截断。

由多个段组成的数据源（例如包含多个段的 ELF 文件）必须加载到其自然位置。这是因为在目标地址中只能指定一个地址或范围，将每段加载到同一地址没有意义。

load 语句的最常见形式是按名称加载源文件。这可能会产生完全不同的数据源，具体取决于源文件类型。每种数据源类型的具体特性如下所述。

ELF 文件——使用整个 ELF 文件作为数据源将加载文件中的所有段。并非所有段都被加载；仅考虑类型为 SHT_PROGBITS 或 SHT_NOBITS 的那些段。ELF 文件中的所有段都具有自然位置和大小。

```
# these two loads are completely equivalent
load myElfFile;
load myElfFile > .;
```

S-record 文件——文件的内容被转换为内存镜像，在该镜像中，通过组合各个加载命令可以找到连续的数据区域。加载段从每个连续区域开始创建。这些段具有自然地址。

```
load mySRecFile;
```

二进制文件——文件的全部内容构成一个没有自然地址的加载段。但是，二进制文件具有一个自然长度。

```
// load an entire binary file to an address
load myBinFile > 0x70000000;

// load part of a binary file
load myBinFile > 0x70000000..0x70001000;
```

二进制对象——除了数据以内联方式在启动描述符文件中列出外，这几乎就像一个二进制文件。同样，原始二进制数据没有自然地址，但具有一个自然长度。

```
// load an eight byte blob
load {{ ff 2e 90 07 77 5f 1d 20 }} > 0xa0000000;
```

ELF 段列表——如果用户只想加载 ELF 文件的某些段，则支持使用 glob 表达式选择 ELF 段的语法。有关段名称的更多信息，请参见第 3.9 节，段名称。数据源语法是一个或多个段名称的列表，后跟“from”关键字和 ELF 文件的源名称。如果 load 语句在“from”语句内，则可以省略“from”关键字和后面的源名称。以下示例演示了该语法：

加载块示例

```
// inclusive section name
load $.text from myElfFile;

// exclusive section name
load ~$.mytext from myElfFile;

// example load inside a from statement
from myElfFile
{
    load $.text.*, ~$.text.sdram;
}
```

ELF 文件的所有段都具有自然位置和大小，并且这些段中的代码仍会位于该位置，因此不能使用显式加载目标。实际上，elftosb 实用工具仅允许选择单一 ELF 段的语句使用显式目标，因为将多个段加载到同一目标地址没有意义。另一方面，将单个段重新定位到内存中的新地址会很有用。

跟随“load”关键字之后以逗号分隔的 ELF 段名称表达式的实际列表会对所选的 ELF 段逐一过滤。如果列表中的每个段名称前面都有波浪号字符（“~”），在这种情况下将反转匹配段集合。例如，段名称“~$.sdram.*”将匹配每个不以“.sdram”开头的段。

查看上述代码块中的第三个加载命令。第一个段名称“$.text.*”匹配每个以“.text.”开头的 ELF 段。列表中的第二个名称 (~.text.sdram) 匹配上一个段名称匹配的所有 ELF 段，但名为“.text.sdram”的除外。如果源文件包含“.text.ocram”、“.text.sdram”、“.bss”和“.data”，则仅选择“.text.ocram”。

整数值——整数值是一个独特的加载数据类型。该值用作填充内存区域的模式。整数源没有自然地址，但确实具有自然长度。

```
# pattern fill
load 0x55.b > 0x2000..0x3000;

# load two bytes at an address
load 0x1122.h > 0xf00;
```

如果将整数值加载到单个地址，则加载填充的字节数与整数值的长度相同。上述示例中的第二个 load 语句将两个字节加载到 0xf00，因为整数值为半字大小。

如果将整数加载到一个地址范围，则仅填充该范围内包含的那些字节。即使整数值的大小大于地址范围的长度，也是如此。

字符串文字——使用字符串文字作为加载数据源与加载二进制文件非常相似。一种用例是在内存中填充一个缓冲区，该缓冲区包含要显示给用户或通过串行端口打印的消息。设置缓冲区后，用户可以使用 call 语句调用打印例程。

```
# load a string at the address of a symbol
load "hello world!" > myElfFile:szMessage;
```

3.12.1.1 Load IFR

加载命令的 IFR 选项用于指定应将数据源中的数据编程到目标位置中所指示的闪存 IFR 索引。

命令文件

其语法如下：

```
load_ifr_stmt   ::=    'load ifr' int_const_expr '>' int_const_expr
                ;
```

load IFR 语句有两种形式，一种是编程到 4 字节 IFR 位置，另一种是编程到 8 字节 IFR 位置。

4 字节 load IFR 语句

```
section (0)
{
    load ifr 0x1234567 > 0x30;
}
```

8 字节 load IFR 语句

```
section (0)
{
    load ifr {{11 22 33 44 55 66 77 88}} > 0x40;
}
```

3.12.2 调用(Call)

call 语句用于插入引导加载程序命令，该命令从一个加载到内存中的文件执行一个函数。函数调用的类型由语句的介绍性关键字决定。

这些语句的语法如下所示：

```
call_stmt ::= call_type call_target call_arg?
  ;
call_type ::= 'call'
  | 'jump'
  ;
call_target ::= SOURCE_NAME
  | symbol_ref
  | int_const_expr
  ;
call_arg ::= '(' int_const_expr? ')'
  ;
```

与 load 语句一样，call 语句以特殊关键字开头。但它不是单个关键字，而是有两种可能。关键字将选择该语句生成哪种特定的启动命令，具体取决于输出启动镜像的格式。通常，“call”命令应该返回引导加载程序，而“jump”命令则不能返回引导加载程序。对于启动镜像，“call”生成 ROM_CALL_CMD，“jump”生成 ROM_JUMP_CMD。有关这些命令的详细信息（如期望的函数原型），请参见启动镜像格式设计文档。

在介绍性的关键字之后是调用目标，支持三种形式，具有各自的语法。所有形式的目标都可归结为一个内存地址。下文将详细介绍这三种不同的形式。

源文件——如果使用源文件名作为调用目标，则 call 语句将使用该源文件的入口点作为目标地址。这意味着源文件必须具有一个入口点。如果使用了不支持入口点或没有设置入口点的源文件，则会报告错误。

```
# call the entry point
call myElfFile;

# same here
jump mySRecFile;
```

```
# this produces an error because binary files
# do not have an entry point
call myBinFile;
```

整数表达式——最直接的调用目标是使用整数表达式。该表达式直接计算由 call 或 jump 启动命令调用的函数的地址。

```
# jump to a fixed address
jump 0xffff0000;
```

符号——虽然它只是整数表达式的另一种形式，但必须指出的是，对 ELF 文件中符号的引用可以用作调用目标。支持显式和隐式两种源文件形式。隐式形式使用封闭式 from 语句提供的源文件。在 from 语句之外使用隐式格式是错误的。列出源文件中不存在的符号或使用非 ELF 类型的源文件也是错误的。

```
# call a function by name and pass it an arg
call myElfFile:initSDRAM (32);

# this is the implicit form of symbol usage
from myElfFile
{
    call :reboot();
}

# this is an error because Srecords do not have symbols
jump mySRecFile:anEntryPoint();
```

注意，提供符号的文件实际上不一定要由同一个命令文件加载。它只是用来查找一个地址，无论该函数是否实际存在于该位置。

call 语句的最后一部分是可选的参数值。它是一个用括号括起来的整数表达式。该表达式确定将什么值作为第一个参数传递给调用或跳转启动命令。如果语句中不含表达式，则参数值默认为零。使用空括号等同于不包含括号。

3.12.3 来源(From)

from 语句的语法最简单，它更像是一个块，而不是一个真正的语句。它本身不产生任何启动命令。相反，使用 from 语句让你可以使用其中所含更简单的语句形式。

from 语句的简单语法遵循以下形式：

```
from_stmt ::= 'from' SOURCE_NAME '{' statement* '}'
;
```

from 语句由“from”关键字、源标识符和用大括号括起来的一系列语句组成。右括号后没有终止分号。左右括号之间允许使用任何类型的语句，但更多 from 语句除外，因为不支持嵌套使用。

某些形式的 load 和 call 语句使用隐式源文件。from 语句所做的就是为其中的语句设置该隐式源文件。这样可以使命令文件更简洁、更容易阅读。

from 语句

```
# name our input file
sources
{
```

```
    example = extern(0);
}

# create a section
section (0)
{
    from example
    {
        # load from example and call a function inside it
        load $.ocram.*;
        call :_start;
    }
}
```

上述示例演示了如何使用 from 语句。from 语句中的 load 和 call 语句并没有显式列出任何源。应该从哪个文件加载命名段？“_start”符号位于哪个文件中？from 语句为这些语句提供隐含的源文件。

load 语句加载示例源中名称以“.ocram”开头的所有段。call 语句会生成一个调用启动命令，指向示例源文件中“_start”符号的地址。

3.12.4 擦除(Erase)

erase 语句插入一个引导加载程序命令，以擦除闪存。

erase 语句的语法为：

```
erase_stmt ::= 'erase' address_or_range
  |       'erase' 'all'
  ;
```

erase 语句有两种形式。最简单的形式（擦除全部）创建一个擦除所有可用闪存的命令。该命令的实际效果取决于引导加载程序的运行时设置以及引导加载程序是否驻留在闪存、ROM 或 RAM 中。

erase 语句的第二种形式接受地址或地址范围作为参数。它将擦除与该地址或范围相交的闪存扇区。如需擦除单个扇区，请提供该扇区内的一个地址。

erase 语句

```
sources
{
    example = extern(0);
}

# create a section
section (0)
{
    erase all;
    load example;
}
```

3.12.5 打印(Print)

print 语句实际上包含三种非常相似的语句，用于为用户打印不同类别的消息。print 语句的三种类型是信息、警告和错误。所有 print 语句均以与其类型相对应的关键字开头，语法如下所示：

```
print_stmt ::= 'info' STRING
  | 'warning' STRING
  | 'error' STRING
  ;
```

info 语句直接将消息打印到标准输出。除非调用程序启用了静默输出功能，否则该消息总是可见。warning 语句的作用与 info 语句基本相同，除了它在消息之前加上了“ warning:”（警告：）。此外，该消息始终可见。最后，error 语句会立即停止 elftosb 的执行，并打印以“error:”为前缀的消息。

print 语句

```
sources
{
    # give the ELF file a name
    afile = "file.elf";
}

constants
{
    # create a constant that is the size of a symbol
    bufsize = sizeof(afile:_my_buf);
}

# create a section
section (0)
{
    if bufsize < 128
    {
        # elftosb stops after this is printed
        error "Buffer size $(bufsize) is too small!";
    }
    else
    {
        info "Buffer size $(bufsize) is acceptable";
    }
    /* ...more... */
}
```

这三种print语句均支持使用类似Unix Shell变量替换的语法来替换常量值和源文件路径。将常量名称或源文件名称置于括号中并以“$”符号作为前缀，则在将消息打印到标准输出之前，会插入相应的值。

对于常量替换，对常量值的格式化有一定的限制。在括号内，将格式化选项放在冒号之前，作为常量名称的前缀。支持的两种格式选项为字符“d”和“x”，一次只能使用其中一个。“d”字符将常量格式化为十进制，“x”字符将常量格式化为十六进制。例如，“$(x:floop)”将常量“floop”格式化为十六进制。

3.12.6 If-Else

为了能够轻松创建可重复使用的启动描述符文件，elftosb 提供 if-else 语句。这些语句的工作方式与您使用的任何其他语言的 if 语句一样。您可以根据需要嵌套多个的 if-else 语句。最后的 else 分支是可选的，可以去除。

其语法如下：

```
if_stmt ::= 'if' bool_expr '{' statement* '}' else_stmt?
;
else_stmt ::= 'else' '{' statement* '}'
| 'else' if_stmt
;
```

在语法上与 ANSI C 有所不同。“if”关键字后的布尔表达式不需要用括号括起来。另外，始终需要将 if 和 else 分支语句用大括号括起来。

if-else 语句中允许使用所有类型的语句，包括 from 语句。反之亦然，if-else 语句也可以放在 from 语句内。

3.12.7 Erase QuadSPI all 语句

erase QuadSPI all 语句将擦除整个外部 QuadSPI 闪存。

其语法为：

```
erase_qspi_stmt ::=      'erase' 'qspi' 'all'
                ;
```

Erase QuadSPI All 语句

```
section (0)
{
    erase unsecure all;
}
```

3.12.8 Erase Unsecure All 语句

erase unsecure all 语句将擦除整个内部闪存并禁用闪存安全性。

其语法为：

```
unsecure_stmt    ::=     'erase' 'unsecure' 'all'
                 ;
```

Erase Unsecure All 语句

```
section (0)
{
    erase unsecure all;
}
```

3.12.9 Enable QuadSPI 语句

enable QuadSPI 语句使用先前加载到 RAM 的参数块来初始化外部 QuadSPI 闪存。

其语法为：

```
enable_stmt     ::=    'enable' 'qspi' int_const_expr
                ;
```

Enable QuadSPI 语句

```
section (0)
{
    # Load quadspi config block bin file to RAM, use it to enable QSPI.
    load myBinFile > 0x20001000;
    enable qspi 0x20001000;
}
```

3.12.10 Reset 语句

reset 语句生成一个引导加载程序复位命令，用于复位目标器件。引导加载程序将忽略复位命令之后 SB 文件中的所有其他命令。

其语法为：

```
reset_stmt     ::=     'reset'
               ;
```

Reset 语句

```
section (0)
{
    reset;
}
```

3.12.11 带堆栈指针的跳转语句

带堆栈指针的跳转语句生成一个引导加载程序跳转命令，该命令在跳转之前会设置堆栈指针。引导加载程序将忽略跳转命令之后 SB 文件中的所有其他命令。第一个参数是堆栈指针的值。第二个参数是跳转地址。第三个（可选）参数是要跳转到的函数的参数。

其语法如下。常规 elftosb 文档中介绍了 call_target 和 call_arg 元素。

```
jump_sp_stmt    ::=   'jump_sp' sp_arg call_target call_arg?
                ;
sp_arg   ::=    int_const_expr
                ;
```

带堆栈指针的跳转语句

```
section (0)
{
    jump_sp 0x20000e00 0x1000 (0x5a5a5a5a);
}
```

3.13 常见用法示例

elftosb 的最常见用法是简单加载单个 ELF 文件并跳转到其入口点，该入口点几乎始终是由 C 运行时库定义的_start 符号。

基本可重用启动描述符文件

```
// Define one input file that will be the first file listed
// on the command line. The file can be either an ELF file
// or an S-record file.
sources
{
    inputFile = extern(0);
}

// create a section
section (0)
{
    load inputFile; // load all sections
    call inputFile; // jump to entry point
}
```

4 elftosb 密钥文件格式

通过-k/--key 命令行开关提供给 elftosb 的密钥文件的格式非常简单。密钥文件的每一行都包含一个密钥，它是一个由 32/64 个十六进制字符组成的的连续字符串，总共包含 128/256 位密钥数据。在一个密钥文件中可能会出现多个密钥。每个密钥单独一行。行尾格式不重要。

示例 16.带有两个 128 位密钥的密钥文件

```
3F3CFBC001F399991035C3C6C7065924
1BA3CD4030FC4376B4AA8CB5E932432E
```

示例 17.带有一个 256 位密钥的密钥文件

```
AAB5CCFB687D378C93821E8793337EA8F98B48A0B596F36CDD169347322E8C87
```

密钥文件的内容为明文。

5 附录 A：命令文件语法

命令文件格式的语法使用扩展巴克斯范式(EBNF)显示如下。

```
command_file ::= pre_section_block* section_def*
  ;

pre_section_block
  :: options_block
  | constants_block
  | sources_block
  ;

options_block ::= 'options' '{' option_def* '}'
  ;
```

```
option_def ::= IDENT '=' const_expr ';'
  ;

constants_block
  ::= 'constants' '{' constant_def* '}'
  ;

constant_def ::= IDENT '=' int_const_expr ';'
  ;

sources_block ::= sources '{' source_def* '}'
  ;

source_def ::= IDENT '=' source_value ( '(' source_attr_list? ')' )? ';'
  ;

source_value ::= STRING_LITERAL
  | 'extern' '(' int_const_expr ')'
  ;

source_attr_list
  ::= option_def ( ',' option_def )*
   ;

section_block ::= 'section' '(' int_const_expr section_options? ')'
              section_contents
   ;

keyblob_block ::= 'keyblob' '(' int_const_expr ')' keyblob_contents
  ;

keyblob_contents
  ::= '{' ( '(' keyblob_options_list ')' )* '}'
  ;

keyblob_options_list
  ::= keyblob_option ( ',' keyblob_option )*
  ;

keyblob_option ::= ( IDENT '=' const_expr )?
  ;

section_options
  ::= ';' source_attr_list?

section_contents
  ::= '{' statement* '}'
  | '<=' SOURCE_NAME ';'
  ;

statement ::= basic_stmt ';'
  | from_stmt
  | if_stmt
  ;

basic_stmt ::= load_stmt
  | call_stmt
  | mode_stmt
  | message_stmt
  ;
```

```
load_stmt ::= 'load' load_data ( '>' load_target )?
  ;

load_data ::= int_const_expr
  | STRING_LITERAL
  | SOURCE_NAME
  | section_list ( 'from' SOURCE_NAME )?
  ;

section_list ::= section_ref ( ',' section_ref )*
  ;

section_ref ::= ( '~' )? SECTION_NAME
  ;

load_target ::= '.'
  | address_or_range
  ;

address_or_range
  ::= int_const_expr
  | int_const_expr '..' int_const_expr
  ;

symbol_ref ::= SOURCE_NAME? ':' IDENT
  ;

load_ifr_stmt ::= 'load ifr' int_const_expr '>' int_const_expr
  ;

call_stmt ::= call_type call_target call_arg?
  ;

call_type ::= 'call'
  | 'jump'
  ;

call_target ::= SOURCE_NAME
  | symbol_ref
  | int_const_expr
  ;

call_arg ::= '(' int_const_expr? ')'
  ;

jump_sp_stmt ::= 'jump_sp' sp_arg call_target call_arg?
  ;

sp_arg ::= int_const_expr
  ;

from_stmt ::= 'from' SOURCE_NAME '{' in_from_stmt* '}'
  ;

in_from_stmt ::= basic_stmt ';'
  | if_stmt
  ;

mode_stmt ::= 'mode' int_const_expr
```

```
  ;

message_stmt ::= message_type STRING_LITERAL
  ;

message_type ::= 'info'
  | 'warning'
  | 'error'
  ;

if_stmt ::= 'if' bool_expr '{' statement* '}' else_stmt?
  ;

else_stmt ::= 'else' '{' statement* '}'
  | 'else' if_stmt
  ;

encrypt_stmt ::= 'encrypt' '(' int_const_expr ')' encrypt_stmt_list
  ;

encrypt_stmt_list
  ::= '{' ( statement )* '}'
  ;

erase_qspi_stmt ::= 'erase' 'qspi' 'all'
  ;

unsecure_stmt ::= 'erase' 'unsecure' 'all'
  ;

enable_stmt ::= 'enable' 'qspi' int_const_expr
  ;

reset_stmt ::= 'reset'
  ;

const_expr ::= bool_expr
  | STRING_LITERAL
  ;

int_const_expr ::= expr
  ;

bool_expr ::= int_const_expr
  | bool_expr '<' bool_expr
  | bool_expr '<=' bool_expr
  | bool_expr '>' bool_expr
  | bool_expr '>=' bool_expr
  | bool_expr '==' bool_expr
  | bool_expr '!=' bool_expr
  | bool_expr '&&' bool_expr
  | bool_expr '||' bool_expr
  | '!' bool_expr
  | IDENT '(' SOURCE_NAME ')'
  | '(' bool_expr ')'
  | 'defined' '(' IDENT ')'
  ;

expr ::= INT_LITERAL
  | IDENT
```

```
  | symbol_ref
  | expr '+' expr
  | expr '-' expr
  | expr '*' expr
  | expr '/' expr
  | expr '%' expr
  | expr '<<' expr
  | expr '>>' expr
  | expr '&' expr
  | expr '|' expr
  | expr '^' expr
  | unary_expr
  | expr '.' INT_SIZE
  | '(' expr ')'
  | 'sizeof' '(' symbol_ref ')'
  | 'sizeof' '(' IDENT ')'
  ;

unary_expr ::= '+' expr
  | '-' expr
  ;
```

6 附录 B：SB 启动镜像文件格式

6.1 术语

AES-128——加密块和密钥大小为 128 位的 Rijndael 密码。

Block cipher（块加密）——适用于 N={64, 128, ...}位块的加密算法。

CBC——加密块链接，一种使用密文块之间反馈的密码模式。

CBC-MAC——使用块加密计算的消息认证码。

Cipher block（加密块）——运行块加密操作的最小数据量。

Ciphertext（密文）——加密数据。

DEK——数据加密密钥，用于加密大量启动镜像的一次性会话密钥。

ECB——电子密码本，一种在密文块之间无反馈的加密模式。

Hash（哈希）——摘要计算算法。

KEK——密钥加密密钥，用于加密会话密钥或 DEK。

MAC——消息认证码。提供完整性和身份验证检查。

Message digest（消息摘要）——使用哈希算法从数据计算出的唯一值。仅提供完整性检查（除非已加密）。

Plaintext（明文）——未加密的数据。

Rijndael——美国政府选择的取代 DES 的块密码。读作“rain-dahl”。

Session key（会话密钥）——加密时生成的加密密钥。只能用一次。

SHA-1——可产生 160 位消息摘要的哈希算法。

6.2 简介

整个启动镜像格式是根据 AES-128 的要求构建的，其最小块大小为 128 位或 16 个字节。AES-128 是用于加密启动镜像的对称块加密。使用其块大小作为整个镜像的基本单位，可以更轻松地进行加密。

为了在一个镜像中支持多个可执行文件，该格式设置了段概念。每个段都可以包含一个独立的可启动镜像，也可以作为更大的段序列的一部分。提供了一个启动命令，可用于指示引导加载程序在运行时从另一个段继续。

此格式的许多功能并非对所有应用程序或读数据方法都有用。例如，段表仅在可以随机访问启动镜像时才有用，而启动标签则在从流媒体启动时最有用。其目标是为镜像提供更多功能，不论其访问方式如何。

6.3 基本类型

在本文档中，使用了几种基本的 C 类型来表示加密块、密钥和其他重要元素。这些类型的定义如下所示。

```
//! An AES-128 cipher block is 16 bytes.
typedef uint8_t cipher_block_t[16];

//! An AES-128 key is 128 bits, or 16 bytes.
typedef uint8_t aes128_key_t[16];

//! A SHA-1 digest is 160 bits, or 20 bytes.
typedef uint8_t sha1_digest_t[20];

//! Unique identifier for a section.
typedef uint32_t section_id_t;
```

6.4 启动镜像格式

启动镜像格式包含五个不同的区域。首先，有一个明文标头，其中包含有关镜像的基本信息。随后是一个段表，也是明文。它描述了镜像中各个不同的段。对于加密镜像，接着是用于支持多个客户密钥的密钥字典。然后，每个段都有其数据，以引导加载程序使用的标签为前缀。最后，镜像的末尾是对整个镜像的校验。下图显示了启动镜像的基本布局。

该镜像格式设计用于从以下流媒体读取：不支持随机访问，同时需要尽量减少要缓存的数据。但是，该格式也包含一些功能，当可以随机访问镜像时，这些功能会非常有用。例如，镜像使用从整个镜像其余部分计算出的验证代码作为结尾。这对于 ROM 并不是特别有用，但是可以由驻留在主机的实用工具用来在使用启动镜像之前对其进行核实和身份验证。

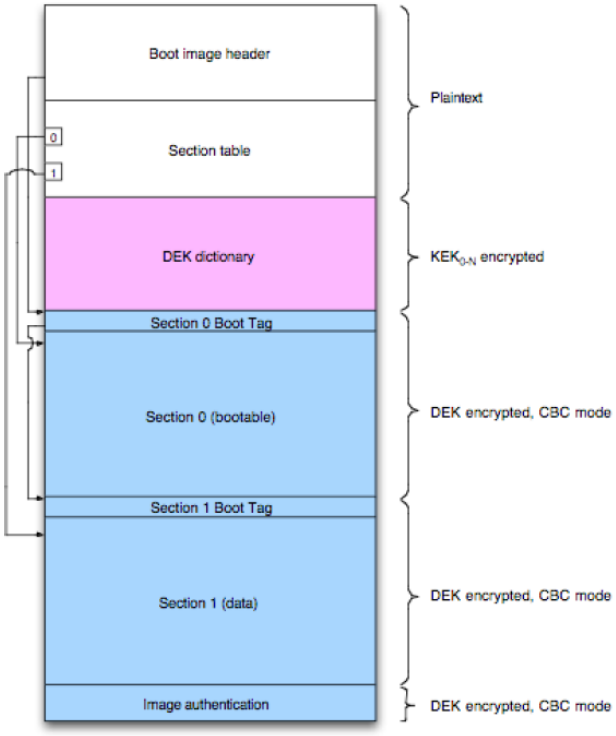

图 2. 启动镜像区域

格式的基本单位大小是 AES-128 加密块的大小，即 16 个字节。文件中的每个区域始终从加密块边界开始。镜像中的每个字段均以小端字节顺序进行格式化。

6.4.1 镜像标头

启动镜像的标头始终未加密。它提供了有关整个镜像的必备信息，以及指向镜像中其他区域的一些有用的指针。

镜像标头的大小始终是加密块的取整值。填充结构所需的任何填充字节始终设置为随机值。如果标头可以完全填满所占用的最后一个加密块，则无需填充。后面紧接着段表字典。

镜像标头的 C 结构定义为：

```
struct boot_image_header_t
{
    union
    {
        sha1_digest_t m_digest;
        struct
        {
            cipher_block_t m_iv;
            uint8_t m_extra[4];
        };
    };
    uint8_t m_signature[4];
    uint8_t m_majorVersion;
    uint8_t m_minorVersion;
    uint16_t m_flags;
    uint32_t m_imageBlocks;
```

```
    uint32_t m_firstBootTagBlock;
    section_id_t m_firstBootableSectionID;
    uint16_t m_keyCount;
    uint16_t m_keyDictionaryBlock;
    uint16_t m_headerBlocks;
    uint16_t m_sectionCount;
    uint16_t m_sectionHeaderSize;
    uint8_t m_padding0[2];
    uint8_t m_signature2[4];
    uint64_t m_timestamp;
    version_t m_productVersion;
    version_t m_componentVersion;
    uint16_t m_driveTag;
    uint8_t m_padding1[6];
};
```

下表描述了 boot_image_header_t 的各字段。为 m_flags 字段定义的标志如第二个表所示。

表 8. 镜像标头字段

字段	说明
m_digest	此前所有标头字段的 SHA-1 摘要。前 16 个字节（总共 20 个字节）还用作 CBC 加密区域的初始化向量。
m_signature	值始终为“STMP”。
m_majorVersion	启动镜像格式的主要版本，当前为 1。
m_minorVersion	启动镜像格式的次要版本，当前为 1 或 2。
m_flags	与整个镜像相关的标志。
m_imageBlocks	整个镜像的大小（以块为单位）。
m_firstBootTagBlock	从文件开头到包含第一个启动标签的加密块的偏移量。
m_firstBootableSectionID	启动开始处的段的唯一标识符。
m_keyCount	DEK 字典中的条目数。
m_keyDictionaryBlock	从镜像的开头开始，DEK 字典的起始块编号。
m_headerBlocks	整个镜像标头的大小，以块为单位。
m_sectionCount	段数。
m_sectionHeaderSize	段标头的大小，以块为单位。
m_padding0	填充两个字节，用于将 m_signature2 与字边界对齐。设置为随机值。
m_signature2	始终设置为“sgtl”。这个第二签名仅存在于次要版本大于或等于 1 的文件中。
m_timestamp	创建镜像时的时间戳（以微妙为单位），大小为 1-1-2000 00:00。
m_productVersion	产品版本。

表格接下页……

表 8. 镜像标头字段（续）

字段	说明
m_componentVersion	器件版本。
m_driveTag	包含该镜像的磁盘驱动器或分区的标识符。
m_padding1	填充八个字节，用于填充加密块。设置为随机值。

表 9. 启动镜像字段

常量	位	说明
ROM_DISPLAY_PROGRESS	0	打开已执行命令的进度报告。
ROM_VERBOSE_PROGRESS	1	打印已执行命令其他信息相关报告。仅适用于同时启用了 ROM_DISPLAY_PROGRESS 的情况。

m_majorVersion 和 m_minorVersion 字段描述启动镜像格式的版本，而不是 ROM 的版本（与以前的启动镜像格式一样）。主要版本字段当前为 1。每当更改此字段时，该格式就不再向后兼容先前版本，并且需要新的引导加载程序。对于任何向后兼容引导加载程序先前版本的格式更改，次要版本字段均应递增。例如，由于存在 m_headerBlocks 字段，因此在镜像标头的末尾添加新字段将向后兼容。在这种情况下，仅应将 m_minorVersion 递增。但是，如果重新排序镜像标头字段，则当前的引导加载程序将无法再读取镜像，并且 m_majorVersion 字段必须递增。有关版本的更多详细信息，请参见本文档末尾的文件格式版本表。

如果 m_keyCount 的值为零，则启动镜像完全未加密。如果字典中至少有一个密钥，则始终对镜像进行加密。

标头的 SHA-1 摘要提供了对未加密镜像的基本完整性检查。它不提供任何额外的安全性，由于它会在标头发生任何更改时直接随之发生更新，因此它不提供任何额外的安全性。

在文件的其余部分，每当使用 CBC 模式对某些内容加密时，m_digest 字段的前 16 个字节都将用作初始化向量。该摘要足够随机，因为所有启动镜像的标头均不同。m_ timestamp 字段除了其名义上的用途外，还可确保每个创建的启动镜像的明文标头都不同。除了提高标头摘要的随机性之外，这一点也很重要，因为标头是使用客户的密钥进行验证。

m_keyDictionaryBiock 字段还用于帮助启动 ROM 加快对标头的处理。这个值可以从其他的标头字段中计算出来，但预先计算该值可以减少 ROM 代码对标头字段的跟踪。

m_productVersion 和 m_componentversion 字段包含描述启动镜像中固件的版本值。这些字段使用以下 C 结构定义：

```
struct version_t
{
    uint16_t m_major;
    uint16_t m_pad0;
    uint16_t m_minor;
    uint16_t m_pad1;
    uint16_t m_revision;
    uint16_t m_pad2;
};
```

在 version_t 结构的每个主要、次要和修订版本字段中，版本号均采用右对齐的 BCD 格式。两个版本的默认值均为 999.999.999。

m_padding0 和 m_padding1 字段用于对齐其他字段，并将结构大小四舍五入为一个偶数加密块。这些字节在创建镜像时将设置为随机值，以增加标头的“白度”，用于加密目的。

6.4.2 段表

该段表实质上是启动镜像中每个段的起始块和长度的索引。它还包含仅适用于该段的标志。

该表始终未加密，并且紧跟在明文镜像标头之后和 DEK 字典之前（如果存在字典）。

段表的 C 类型定义及其条目如下：

```
struct section_header_t
{
    section_id_t m_identifier;
    uint32_t m_offset;
    uint32_t m_length;
    uint32_t m_flags;
};
struct section_table_t
{
    section_info_t m_sections[1];
};
```

下表描述了 section_header_t 的各字段。为 section_header_t 的 m_flags 字段定义的标志如第二个表所示。

表 10. 段标头字段

字段	说明
m_identifier	该段的 32 位唯一标识符。
m_offset	该段数据的起始加密块，从镜像的开头开始。
m_length	段数据的长度（以加密块为单位）。
m_flags	适用于整段的标志。

表 11. 段标志

常量	位	说明
ROM_SECTION_BOOTABLE	0	该段为可启动段，并包含一系列启动加载程序命令。
ROM_SECTION_CLEARTEXT	1	该段未加密。仅适用于启动镜像的其余部分已加密的情况。

段表中每个条目的长度来自镜像标头的 m_sectionHeaderSize 字段。条目长度始终是整数个加密块，必要时需进行填充。表中所有条目的长度均相同。在文件格式版本 1 中，段表条目的长度为单个加密块，无填充。

段的总数（以及段表中的条目数）在镜像标头的 m_sectionCount 字段中提供。对于有效的可启动镜像来说，该值应始终至少为 1。如果为 0，则镜像不包含启动命令，并且被视为无效。另外，必须至少有一个设置了 ROM_SECTION_BOOTABLE 标志的段，镜像才能有效。

段表的大小是 (header.m_sectionCount * header.m_sectionHeaderSize) 加密块或 (header.m_sectionCount * header.m_sectionHeaderSize * 16)字节。

6.4.3 DEK 字典

密钥字典始终跟随加密镜像中下一个加密块中的镜像标头。未加密镜像没有 DEK 字典。

其目的是允许单一启动镜像与任意数量的客户密钥一起使用。这是通过每次生成启动镜像时生成一个新密钥，即数据加密密钥(DEK)来实现的。除此字典外，镜像的其余部分均使用此 DEK 加密。通过使用每个要支持的客户密钥来加密 DEK，将该字典用于以安全的方式从任何给定的客户密钥映射到 DEK。因此，如果没有有效的客户密钥，DEK 将永远无法使用。

字典中的每个条目都由两部分数据组成：消息认证码(MAC)和加密的 DEK 本身。MAC 充当校验码（可搜索的已知值）。否则，将无法区分有效的解密 DEK 和垃圾数据。

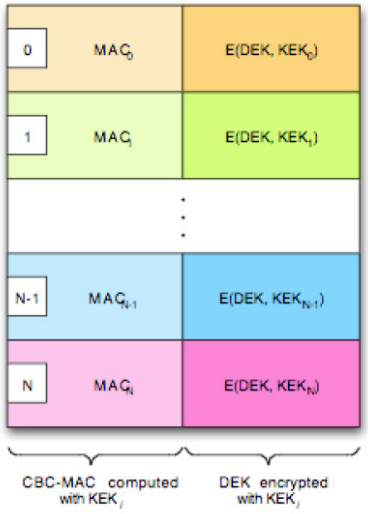

图 3. DEK 字典

MAC 是通过一种称为 CBC-MAC 的技术生成的。启动镜像的标头和段表始终都是明文，在 CBC 模式下使用给定字典条目的 KEK 进行加密。该加密的初始化向量始终为零。在此过程中，仅保留最后一个加密块。验证代码是最后一个加密块。

DEK 字典的 C 类型定义如下：

```
struct dek_dictionary_entry_t
{
    cipher_block_t m_mac;
    aes128_key_t m_dek;
};
struct dek_dictionary_t
{
    dek_dictionary_entry_t m_entries[1];
};
```

每个条目中的 m_dek 字段在 CBC 模式下使用来自镜像标头的 IV，利用 KEK 进行加密。不对保存在 m_mac 字段中的 CBC-MAC 结果进行加密。这没有必要，因为它是由安全 OTP 密钥生成的。

字典中的条目数由镜像标头中的 m_keyCount 字段确定。字典大小始终为 header.m_keyCount * 2 个加密块或 header.m_keyCount * 32 个字节。如果 m_keyCount 为零，那么 DEK 字典在镜像中不占用任何加密块，并且整个镜像未加密。

对字典大小的唯一实际限制是启动时间。字典条目越多，启动设备所需的时间就越长。至少搜索 DEK 的算法应为 O(n)。

6.4.4 段启动标签

在每个段数据区域之前，都有一个特殊的标记加密块，用于描述下一段。这些标签称为启动标签，因为启动 ROM 使用它们来搜索段，无需在内存中维护整个段表的副本或从存储器中重新读取镜像的某些段。启动标签始终与段数据区域配对——两者缺一不可。另一种方式是将启动标签看作段内容的本地段标头。

启动标签的实际结构是 ROM_TAG_CMD 引导加载程序命令的结构。启动标签重复使用启动命令结构可以简化 ROM 代码。标签命令包含与之配对的段数据区域的段表条目中某些字段的复制项。其中最重要的是段标识符和段长度（以块为单位）。

由于各段之间不允许填充，因此该段的长度实际上指向下一个启动标签。这样，启动 ROM 可以通过比较标识符并遵循启动标签形成的链表来轻松搜索段数据区域。镜像中的最后一个启动标签始终设置配有 ROM_LAST_TAG 标志，以帮助 ROM 知道何时停止搜索。

6.4.5 段数据区域

段数据区域有两种类型。第一种是包含一系列启动命令的可启动区域。第二种是任何非启动区域，其中可能包含未被启动 ROM 处理的任意数据。这些区域可能包含客户应用需使用的资源或其他数据。

可启动区域的内容就是逐一排序的一系列引导加载程序命令。可启动段必须始终以 ROM_TAG_CMD 引导加载程序命令开头。有关引导加载程序命令的结构和各命令的详细信息，请参见第 9 节。

为 MCU ROM 创建的 SB 文件必须以可启动段开始。ROM 在此可启动段的末尾停止处理工作。将忽略其他可启动段和数据段。

段数据区域的顺序必须与它们在段表中显示的顺序相同。也就是说，在启动镜像中，段编号为 1 的数据区域必须位于段编号为 0 的数据区域之后。另外，在段数据区域之前或之后也不得插入填充块，尽管格式通过使用加密块指针隐式支持这一操作。这些限制旨在简化 ROM 对启动镜像的处理。

6.4.6 镜像验证代码

每个启动镜像都以验证代码结尾，该验证代码是根据镜像的整个内容计算得出的（当然不含验证代码）。此代码是使用 CBC 模式利用 DEK 加密的 SHA-1 摘要。验证代码在镜像中占用两个加密块，并在 SHA-1 摘要的最后一个字之后添加三个字长的填充（因为 SHA-1 摘要为 160 位，加密块为 128 位）。填充字节设置为随机值。

摘要从以下组件中计算得出，计算顺序为：明文标头、明文段表、DEK 字典、明文段内容。

哈希算法本身并不提供验证，仅提供完整性检查。但是，如果摘要使用了密钥加密，则可将其用于提供验证。

在未加密的启动镜像中，镜像验证代码当然也是未加密的。该代码不再提供验证，但仍提供对整个镜像的完整性检查。

验证代码始终从加密块编号(header.m_imageBlocks-2)开始。

6.5 加密细节

6.5.1 加密过程

加密过程仅发生在 elftosb 实用工具中，因为它会将 ELF 或 S-record 二进制文件转换为启动镜像。以下序列显示了 elftosb 加密镜像的步骤。

1. 构建明文镜像标头
    a. 生成 IV
    b. 对镜像标头进行 SHA-1 计算
2. 生成明文段表
3. 生成 DEK
4. 对于每个 KEK：
    a. 读取 KEK 密钥文件
    b. 使用 IV=0，对明文镜像标头进行 CBC-MAC 计算
    c. 在 CBC 模式下使用来自标头的 IV，利用 KEK 加密 DEK
    d. 将未加密的 CBC-MAC 和加密的 DEK 合并为字典条目
5. 对于每个段：
    a. 生成 ROM_TAG_CMD 作为此段的启动标签
    b. 使用 CBC 模式和来自标头的 IV，加密启动标签
    c. 生成明文段的内容
    d. 使用 CBC 模式和来自标头中的 IV，加密段内容
6. 计算镜像的 SHA-1 摘要
7. 使用 CBC 节点和来自标头的 IV，加密镜像摘要

6.5.2 解密过程

解密过程在 ROM 中进行。此外，还有一个主机实用工具，也可出于测试目的对启动镜像进行解密。

1. 读取镜像标头的第一个加密块。第一个加密块中的 m_keyCount 字段指示镜像是否已加密。如果镜像已加密，则 m_keyCount 将为非零值。
2. 读取镜像标头后，使用客户密钥对其进行 CBC-MAC 计算。
3. 对于 DEK 字典中的每个条目：
    a. m_mac 字段与计算出的 CBC-MAC 是否匹配？如果不匹配，跳转至下一个条目。
    b. 如果 m_mac 字段匹配，则使用客户密钥解密 DEK 并退出循环。
4. 对于每个段表和要读取的所有段数据区域：
    a. 在 CBC 模式下使用来自标头的 IV，利用 DEK 解密该区域。

6.5.3 启动命令

镜像中的可启动段包含一系列启动命令以及这些命令所需的任何数据。命令从第一条开始按线性顺序处理。每个启动命令占用一个加密块，加上与该命令相关的数据所需的所有加密块。启动命令的 C 结构定义如下：

```
struct boot_command_t
{
    uint8_t m_checksum;
    uint8_t m_tag;
    uint16_t m_flags;
    uint32_t m_address;
    uint32_t m_count;
    uint32_t m_data;
};
```

选择本章节中介绍的命令可以在构建启动镜像时，使用最少的命令类型获得最大的灵活性。在大多数情况下，boot_command_t 的各个字段在每个命令中均有不同的确切含义，请参见下文的介绍。

因为 m_checksum 字段在每个命令中都采用相同的计算方式，所以有必要特别提一下。该字段提供了一种低成本且简便的方法来验证加密块是否包含有效的引导加载程序命令。虽然 8 位肯定不足以可靠地防御损坏或有意篡改，但总比没有好。

校验和的计算方法如下：

```
boot_command_t bootCommand;
uint8_t * bytes = reinterpret_cast<uint8_t *>(&bootCommand);
uint8_t checksum = 0x5a;
int i;

// Unroll this loop for better optimization.
for (i = 1; i < sizeof(bootCommand); ++i)
{
    checksum += bytes[i];
}
```

注意，仅对每个启动命令的 boot_command_t 结构字节 1 到 15 计算校验和。换句话说，命令后的任何其他加密数据块均不包括在校验和中。另请注意，初始校验和值为 0x5a，而不是 0。这是为了防止全零命令的校验和也为零。

每个启动命令的 m_tag 字段都包含一个唯一的字节值，用来标识结构所描述的命令。启动命令标签值的列表如下所示。

表 12. 启动命令标签值

命令标签值	命令标签助记符
0x00	ROM_NOP_CMD
0x01	ROM_TAG_CMD
0x02	ROM_LOAD_CMD
0x03	ROM_FILL_CMD
0x04	ROM_JUMP_CMD

表格接下页……

表 12. 启动命令标签值（续）

命令标签值	命令标签助记符
0x05	ROM_CALL_CMD
0x06	保留
0x07	ROM_ERASE_CMD
0x08	ROM_RESET_CMD
0x09	ROM_MEM_ENABLE_CMD
0x10	ROM_PROG_CMD

任何与上表列出的值不匹配的 m_tag 值均无效。如果遇到这种情况，引导加载程序将停止并报告错误。

ROM_NOP_CMD

ROM_NOP_CMD 为无操作命令。引导加载程序会直接跳过它。除 m_tag 字段以外，引导加载程序会忽略所有其他字段，并且可包含任何值。但是，在记录这些字段的其他用途之前，它们应包含下表中显示的值。

表 13. 无操作命令字段

字段	说明
m_checksum	简单校验和，当所有其他字段均为零时，其值为 0x5a。
m_tag	0x00 或 ROM_NOP_CMD
m_flags	0
m_address	0
m_count	0
m_data	0

任何与上表列出的值不匹配的 m_tag 值均无效。如果遇到这种情况，引导加载程序将停止并报告错误。

ROM_TAG_CMD

ROM_TAG_CMD 用作描述段或本地段标头的一种“关键帧”。它包含段表中段条目的大多数字段。

该命令不应当出现在可启动段的命令流中，如果出现了，引导加载程序会直接忽略该命令。该命令定义旨在描述启动标签加密块的结构。启动标签使用的结构与启动命令完全相同，以使引导加载程序更易于执行。

表 14. 提示标签命令字段

字段	说明
m_checksum	boot_command_t 其他字段的简单校验和。
m_tag	0x01 或 ROM_TAG_CMD
m_flags	位 0：ROM_LAST_TAG

表格接下页……

表 14. 提示标签命令字段（续）

字段	说明
m_address	段标头中的 m_tag 字段。
m_count	该段数据所占用的加密块数。这也是直到下一个启动标签（最后一个除外）为止的加密块的数量。
m_data	段标头中的 m_flags 字段。

ROM_LOAD_CMD

该命令后跟任意数量的加密块，其中包含要加载到内存中的数据，从 boot_command_t 的 m_address 字段指定的位置开始。m_count 字段包含要加载到内存中该位置的字节数。

表 15. 加载命令字段

字段	说明
m_checksum	boot_command_t 其他字段的简单校验和。
m_tag	**0x02 或** ROM_LOAD_CMD
m_flags	**位 0：**保留
m_address	存储数据的内存地址。
m_count	要加载的字节数。这也是该命令之后数据加密块中的有效字节数。
m_data	对要加载的数据进行 CRC-32。

该命令后的加密块数为(m count + 15) / 16。这意味着最后一个数据加密块中最多可以填充 15 个字节。填充字节始终使用随机数据进行填充。有关如何为数据大小为 18 字节的加载命令安排加密块的示例，请参见下图。

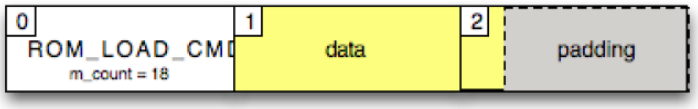

图 4. 加载注释加密块

对 m_address 或 m_count 字段的对齐方式无任何限制。由 ROM 实现决定如何以最佳方式优化数据加载。因此，无法保证数据写入存储器的顺序。

m_data 字段包含对命令标头块后面的数据计算得出的 CRC-32 值。最后一个数据加密块中的任何填充字节都包含在 CRC-32 计算中。

ROM_FILL_CMD

该引导加载程序命令用于以位模式填充内存区域。填充模式始终是完整的 32 位宽，但完全支持字节对齐的填充长度和目标地址。

表 16. 填充命令字段

字段	说明
m checksum	boot_command_t 其他字段的简单校验和。

表格接下页……

表 16. 填充命令字段（续）

字段	说明
m_tag	0x03 或 ROM_FILL_CMD
m_flags	始终为 0。
m_address	写入填充模式的起始存储器地址。
m_count	要填充的字节数。
m_data	填充模式。无论模式大小如何，始终复制整个字。

无论填充模式实际大小如何，都必须填满整个 m_data 字段。一个字节宽的模式必须在整个 m_data 中复制四次，对于半字模式则必须复制两次。

填充时，将对模式进行调整，以使最高有效字节与要填充的第一个字节对齐。下图演示了该命令的执行情况。

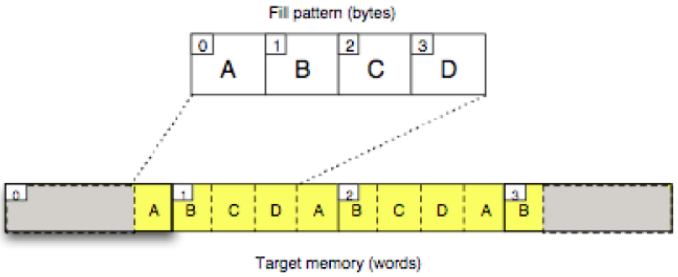

图 5. 填充模式对齐

注意，该命令可确保在任何参差不齐的未对齐边缘之间使用字写入操作。这样就可以将填充命令用作 word poke 操作，以写入寄存器。

ROM_JUMP_CMD

当引导加载程序遇到此命令时，将停止引导加载，CPU 控制权将转移到位于 m_address 的函数中。将 m_data 的内容作为单个参数传递给函数。执行此命令后，ROM 不会重新获得对 CPU 的控制权。

表 17. 跳转命令字段

字段	说明
m_checksum	boot_command_t 其他字段的简单校验和。
m_tag	0x04 或 ROM_JUMP_CMD
m_flags	位 0：保留。
m_address	为 PC 设置的地址。
m_count	如果 m_flags 的位 1 置位，则为初始堆栈指针，否则为 0。
m_data	传递给 R0 中入口点的参数。

由 ROM_JUMP_CMD 执行的函数原型如下。

```
void jump_function( uint32_t arg );
```

注意无效结果。如果函数确实返回，则引导加载程序将失败，并显示 ERROR_ROM_LDR_JUMP_RETURNED 错误。

如果 m_flags 的位 1 置位，则 m_count 字段包含在执行跳转之前要设置的初始堆栈指针寄存器值。

ROM_CALL_CMD

与 ROM_JUMP_CMD 一样，ROM_CALL_CMD 也调用驻留在 m 地址的函数，并将值 m_data 作为其参数传递。这两个命令之间第一个也是最重要的区别是语义上的区别（由 ROM_CALL_CMD 调用的函数应当放弃控制权，并返回到 ROM 中，以便继续进行引导加载）。另外，此命令在函数原型中添加了第二个可选参数。第二个参数可以与函数的返回值结合使用，以告诉引导加载程序跳转到当前启动镜像中的另一段，或准备一个全新的启动镜像。

表 18. 调用命令字段

字段	说明
m_checksum	boot_command_t 其他字段的简单校验和。
m_tag	0x05 或 ROM_CALL_CMD
m_flags	位 0：保留。
m_address	函数要调用的地址。
m_count	0
m_data	要传递给 R0 中函数的参数。

由 ROM_CALL_CMD 执行的函数的完整原型如下：

```
int call_function( uint32_t arg, uint32_t * resultId );
```

将 m_data 字段的值在第一个参数中传递给函数。第二个参数是一个指向字的指针，函数可以修改该字以返回段或镜像 ID。

返回值确定在 call_function()返回时所发生的情况，以及是否检查了*resultId。可能的返回值如下表所示。

表 19. 调用命令返回值

返回值	操作
< 0	负值表示错误。
0=SUCCESS	成功。继续执行当前段中的命令。
1 = ROM_BOOT_SECTION_ID	切换到 ID 为*resultId 的段。
2=ROM_BOOT_IMAGE_ID	重新启动引导加载程序，期待新的启动镜像。当再次调用其初始化函数时，会将*resultId 值传递给驱动程序。
> 2	忽略，与 SUCCESS 相同。

这两个正值返回代码具有特殊含义。如果函数返回 ROM_BOOT_SECTION_ID，则引导加载程序开始搜索当前镜像的某一段，该段的 ID 等于通过 resultId 返回的值。该段必须紧跟在镜像的当前段后，否则将找不到该段，因为引导加载程序仅在影像中向前搜索。如果未找到具有唯一匹配标识符的段，则启动失败并显示错误。

如果函数返回 ROM_BOOT_IMAGE_ID，则引导加载程序准备开始读取全新的启动镜像文件，并通过再次调用其初始化函数将该信号发送给当前的启动驱动程序。通过 resultId 返回的值是一个启动镜像的 ID；镜像 ID 对于每个启动驱动程序都有特定的含义，并非所有启动驱动程序都支持切换到新的镜像文件。当使用不支持此功能的驱动程序切换启动镜像时，未定义其行为。

只有当返回值是ROM_BOOT_SECTION_ID或ROM_BOOT_IMAGE_ID时，引导加载程序恢复执行时才会检查resultId所指向的值。由于这一点以及 ARM® ABI 的工作方式，对那些不希望返回 ROM_BOOT_SECTION_ID 或 ROM_BOOT_IMAGE_ID 的函数，可将其原型缩短为以下形式：

```
int call_function_short( uint32_t arg );
```

ROM_ERASE_CMD

擦除命令仅适用于具有内部闪存阵列的器件（如 Kinetis 器件）。它对整个闪存阵列或命令字段中指定的内存范围执行闪存擦除命令。

表 20. 擦除命令字段

字段	说明
m_checksum	boot_command_t 其他字段的简单校验和。
m_tag	0x07 或 ROM_ERASE_CMD
m_flags	请参见下表。
m_address	要擦除的闪存的起始地址。
m_count	要擦除的闪存的字节数。结束地址为m_address + m_count-1。
m_data	0

表 21. 擦除命令标志位

位	标志	说明
0	ROM_ERASE_ALL_MASK	如果置位，则擦除所有闪存，而不是仅擦除指定范围。如果清 0，则使用 m_address 和 m_count 字段来确定要擦除的闪存范围。
1	ROM_ERASE_ALL_UNSECURE_MASK	如果置位，则擦除所有闪存，并将闪存的安全状态设置为禁用(erase-all- unsecure)。
11:8	0x00kLdrMemoryCtrl_InternalFlash 0x01 kLdrMemoryCtrl_QSPI0	存储器控制器ID。值0x0（默认）表示内部闪存。值0x01表示支持QSPI0的设备上的外部QSPI0。如果设置为0x01，则将忽略位1 (ROM_ERASE_ALL_UNSECURE_MASK)。

由 m_fiags 字段的位 0 决定是擦除整个闪存阵列，还是仅擦除一个子集。如果位 0 置位，则该命令将擦除整个闪存。在这种情况下，将忽略 m_address 和 m_count 字段。

如果 m_flags 的位 0 清零，则要擦除的闪存范围由 m_address 和 m_count 命令字段指定。由于只能以整个扇区为单位擦除闪存，因此将擦除所有与地址范围相交的闪存扇区。即使地址范围不以对齐的扇区边界开始或结束，亦是如此。

如果 m_fiags 字段的位 1 置位，则在擦除闪存后，会将闪存安全状态设置为“禁用”。有关闪存 erase-all-unsecure 命令的详细信息，请参见特定芯片的参考手册。

位 11:8 指示要擦除的闪存设备的存储器控制器 ID。值 0x0（默认）表示内部闪存。值 0x01 表示支持 QSPI0 的设备上的外部 QSPI0。

ROM_RESET_CMD

目标被重置。

表 22. 复位命令字段

字段	说明
m_checksum	简单校验和，当所有其他字段均为零时，其值为 0x5a。
m_tag	0x08 或 ROM_RESET_CMD
m_flags	0
m_address	0
m_count	0
m_data	0

ROM_MEM_ENABLE_CMD

启用（配置）外部存储器。m_flags 字段位 11:8 指示存储器控制器 ID。m_address 字段包含先前写入配置块的 RAM 中的地址，m_count 字段则包含配置块的大小。配置块的格式取决于内存空间。

注意，此命令实际上并未将配置块写入外部介质，而只是使用配置块来配置接口。

表 23. 内存启用命令字段

字段	说明
m_checksum	简单校验和，当所有其他字段均为零时，其值为 0x5a。
m_tag	0x09 或 ROM_MEM_ENABLE_CMD
m_flags	参见存储器控制器 ID 表。
m_address	RAM 中现有配置块的地址。
m_count	配置块的大小。
m_data	0

表 24. 存储器启用命令标志位

位	值	说明
11:8	0x01 kLdrMemoryCtrl_QSPI0	存储器控制器 ID。值 0x01 表示支持 QSPI0 的设备上的外部 QSPI0。不支持其他值。

ROM_PROG_CMD

写入 program-once 永久位。m_fiags 字段的位 11:8 包含内存空间 ID（仅支持 kLdrMemorySpace_iFR0）。m_flags 字段的位 1 表示 8 字节写操作（如果置位）或 4 字节写操作（如果清零）。m_address 字段包含 IFR 索引。m_count 字段包含要编程的前四个字节。m_data 字段（可选）包含要写入的下 4 个字节（如果标志字段的位 1 置位）。

表 25. 编程命令字段

字段	说明
m_checksum	简单校验和，当所有其他字段均为零时，其值为 0x5a。
m_tag	0x0a 或 ROM_PROG_CMD
m_flags	请参见“编程命令标志位”表。
m_address	IFR 索引。
m_count	要编程的前四个字节。
m_data	要编程的后四个字节（如果 m_flags 的位 1 置位）。

表 26. 编程命令标志位

位	标志/值	说明
1	ROM_PROG_8BYTE_MASK	如果置位，则写入八个字节，否则写入四个字节。
11:8	**0x04** kLdrMemorySpace_IFR0	内存空间。值 0x04 表示内部 IFR 闪存。不支持其他值。

6.6 文件格式版本

版本以 Major.minor 形式列出。

表 27. 文件格式版本

版本	说明
1.3	支持 Kinetis 特定功能。

7 附录 C：SB2 启动镜像文件生成

SB2 容器是对附录 B 中描述的 SB（安全二进制）容器的扩展。SB2 使用更新、更安全的版本取代可能较弱的旧安全算法，并增加了可选的内容数字签名可能性。

7.1 SB2.0

SB2.0 容器能够以两种形式生成：加密或加密并签名。

使用示例（加密的 SB2.0）：

e/ftosb -f lpc55xx -k "sbkek.txt"-c "commandFUe.bd"-o "output.sb2" "input.bin"

其中

-f = 系列 [kinetisk3, k32w0x, lpc55xx, rt6xx]

-k = KEK 文件(SBKEK)的路径，用于 keyblob 加密，应为十六进制格式的 AES-256 位密钥。有关格式的其他信息，请参见 elftosb 密钥文件格式

-c = 要处理的命令文件的路径。有关命令文件的其他信息，请参见命令文件。本示例中使用的基本命令文件如下所示：

```
options
    {
        flags = 0x4; // 0x8 encrypted + signed, 0x4 encrypted
        buildNumber = 0x1;
        productVersion = "1.00.00";
        componentVersion = "1.00.00";
        secureBinaryVersion = "2.0";
    }
sources
    {
        inputFile = extern(0);
    }
section (0)
    {
        load inputFile > 0x0;
    }
```

-o = 输出文件的路径

files...= 文件（通常为镜像文件）的路径，它将替换命令文件中定义的占位符，路径可以在命令文件中进行硬编码，而不作为输入插入

用例（加密+签名的 SB2.0）：

elftosb.exe -f lpc55xx -k "sbkek.txt"-c "commandFiie.bd"-o "output.sb2"-s "selfsign_privatekey_rsa2048.pem"-S "selfsign_v3.der.crt"-R "selfsign_v3.der.crt"-h "RKTH.bin" "test_output.bin"

其中

-f = 系列 [kinetisk3, k32w0x, lpc55xx, rt6xx]

-k = KEK 文件(SBKEK)的路径，用于 keyblob 加密，应为十六进制格式的 AES-256 位密钥，有关格式的更多信息参见第 4 章

c = 要处理的命令文件的路径，有关命令文件的更多信息参见第 3 章。本示例中使用的基本命令文件如下所示：

```
options {
            flags = 0x8; // 0x8 encrypted + signed, 0x4 encrypted
            buildNumber = 0x1;
            productVersion = "1.00.00";
            componentVersion = "1.00.00";
            secureBinaryVersion = "2.0";
        }
sources {
            inputFile = extern(0);
        }
section (0) {
            load inputFile > 0x0;
        }
```

-o = 输出文件的路径

-s = 用于签名的证书私钥的路径

-S = 证书链中证书的路径，证书链中的每个证书都必须按照证书链的创建顺序（最先是根证书）用新的-S 开关指定

-R = 根证书的路径，可以指定 1-4 个根证书，每个根证书必须使用新的-R 开关指定，其中一个根证书必须是由-S 开关指定的第一个证书

-h = elftosb 生成的输出二进制文件的路径和名称，该文件包含必须上传到设备寄存器的所有根证书(RKTH)的哈希

files...= 将替换命令文件中所定义占位符的文件（通常为镜像文件）路径，路径可以在命令文件中进行硬编码，而不作为输入插入

7.2 SB2.1

Sb2.1 容器只能以加密+签名一种形式生成。加密+签名的 SB2.0 和 SB2.1 之间的区别在于数字签名的位置（SB2.0：在容器的末尾，SB2.1：在容器标头之后，其中签名的标头包含下一段的 hmac 表）。这样可以在 SB2.1 容器处理的起始阶段验证数字签名，比 SB2.0 安全性更高。

用例（加密+签名的 SB2.1）：

elftosb.exe -f lpc55xx -k "sbkek.txt" -c "commandFile.bd" -o "output.sb2" -s "selfsign_privatekey_rsa2048.pem" -S "selfsign_v3.der.crt" -R "selfsign_v3.der.crt" -h "RKTH.bin" "test_output.bin"

其中

-f = 系列 [lpc55xx, lpc55s1x, rt5xx, rt6xx]

-k = KEK 文件(SBKEK)的路径，用于 keyblob 加密，应为十六进制格式的 AES-256 位密钥。有关格式的其他信息，请参见 elftosb 密钥文件格式

-c = 要处理的命令文件的路径。有关命令文件的其他信息，请参见命令文件。本示例中使用的基本命令文件如下所示：

```
options
{
        flags = 0x8; // 0x8 always for SB2.1
        buildNumber = 0x1;
        productVersion = "1.00.00";
        componentVersion = "1.00.00";
        secureBinaryVersion = "2.1";
}
sources
{
        inputFile = extern(0);
}
section (0)
{
        load inputFile > 0x0;
}
```

-o = 输出文件的路径

-s = 用于签名的证书私钥的路径

-S = 证书链中证书的路径，证书链中的每个证书都必须按照证书链的创建顺序（最先是根证书）用新的-S 开关指定

-R = 根证书的路径，可以指定 1-4 个根证书，每个根证书必须使用新的-R 开关指定，其中一个根证书必须是由-S 开关指定的第一个证书

-h = elftosb 生成的输出二进制文件的路径和名称，该文件包含必须上传到设备寄存器的所有根证书(RKTH)的哈希

files...= 替换命令文件中所定义占位符的文件（通常为镜像文件）的路径。路径可以在命令文件中进行硬编码，而不作为输入插入

8 附录 D：主启动镜像文件生成

8.1 概述

Elftosb 为 k32w0x、lpc55xx、lpc54x0xx 和 rt6xx 器件系列生成主启动镜像。镜像规范使用的是 json 镜像配置文件，elftosb 基于该文件创建输出二进制镜像文件。产生的二进制镜像可以直接使用，也可以用于进一步处理（例如，用作 SB2 镜像容器的输入）。

用例：

elftosb -f lpc55xx -J pathToJsonFile

其中

-f = 系列 [k32w0x, lpc55xx, lpc54x0xx, lpc55s1x, rt5xx, rt6xx]

-J = 要处理的 json 文件

8.2 k32w0x 的 json 镜像配置文件

json 镜像配置文件的结构：

```
{
    "family": "k32w0x",
    "inputImageFile": "C:/mkimage/images/image.bin",
    "rootCertificate0File": "C:/mkimage/keys_and_certs/rootCert0_CA_selfSign.crt",
    "chainCertificate0File0": "C:/mkimage/keys_and_certs/Cert0_CA.crt",
    "chainCertificate0File1": "C:/mkimage/keys_and_certs/Cert1_noCA.crt",
    "rootCertificate1File": "C:/mkimage/keys_and_certs/rootCert1_noCA_ selfSign.crt",
    "rootCertificate2File": "",
    "rootCertificate3File": "",
    "mainCertChainId": 1,
    "mainCertPrivateKeyFile": "C:/mkimage/keys_and_certs/rootCert1PrivateKey.pem",
    "masterBootOutputFile": "C:/mkimage/signed_images/master_boot_image.bin"
}
```

json 镜像配置文件中的字段说明：

family：["k32w0x"]

inputImageFile：包含要处理的应用程序（镜像）的 bin 文件的路径

rootCertificate0File：根证书 0 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

chainCertificate1File0：链中第一个证书的路径，应为采用 DER 编码的 X.509 v3 证书

chainCertificate1File1：链中第二个证书的路径（使用**“chainCertificate1FileN”**，其中 N 为链中的最后一个证书的编号），应为采用 DER 编码的 X.509 v3 证书，链中的所有证书必须为 CA，除了最后一个不可以为 CA

rootCertificate1File：根证书 1 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

rootCertificate2File：根证书 2 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

rootCertificate3File：根证书 3 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

mainCertChainId [0,1,2,3]：应该使用哪个根证书或证书链对镜像签名

mainCertPrivateKeyFile：用于签名的证书的私钥（不是 CA 自签名根证书或链中的最后一个证书）

masterBootOutputFile：输出认证镜像的文件路径和文件名

8.3 lpc54x0xx 的 json 镜像配置文件

json 镜像配置文件的结构：

```
{
"family": " lpc54x0xx",
"inputImageFile": "C:/mkimage/images/image.bin", "imageLinkAddress": "0x20000000",
"imageLinkAdressFromImage": false, "outputImageType": "ram", "outputImageAuthenticationType":
"Signed", "outputImageEncryptionKeyFile": "",
"preformattedSignature": false,
"useKeyStore": false,
"keyStoreFile": "",
"mainCertChainId": 1,
"rootOfTrustKeyFile": "unencrypted_rotk_key.pem",
"rootOfTrustKeyFilePassword": "",
"privateImageKey": "unencrypted_image_key.pem",
"privateImageKeyPassword": "",
"masterBootOutputFile": "C:/mkimage/signed_images/master_boot_image.bin"
}
```

json 镜像配置文件中的字段说明：

family：["lpc54x0xx"]

inputImageFile：包含要处理的应用程序（镜像）的 bin 文件的路径

imageLinkAddress：["0x00000000","0x20000000"] SRAM 中输入镜像的起始地址

imageLinkAdressFromImage：[false, true]设置为 true 时，将从镜像源加载/保留起始地址（其使用将优先于 imageLinkAddress 中的值）

outputImageExecutionTarget：["External flash (XIP)", "RAM"]

outputImageAuthenticationType：["CRC", "Signed", "Encrypted", "Encrypted + Signed", "Signed + Encrypted"]。

"CRC"只能特定用于"External flash (XIP)"的 *outputImageExecutionTarget*。

"Signed"、"Encrypted"、"Encrypted+Signed"、"Signed+Encrypted"只能特定用于"RAM"的 *outputimageExecutionTargefof*。

"Signed+Encrypted"对输出镜像先加密后签名。已经签名或加密的输入镜像将被拒绝。

"Encrypted+Signed"对输出镜像先签名后加密。已经签名或加密的输入镜像将被拒绝。

"Encrypted"对输出镜像加密。已经加密的输入镜像将被拒绝。

"Signed"对输出镜像签名。已经签名的输入镜像将被拒绝。

outputImageEncryptionKeyFile：用于"Signed"（镜像标头的 HMAC）和"Encrypted + Signed"（镜像标头的 HMAC+整个镜像加密）镜像类型的加密密钥的路径，如果 SB2 容器以后将与生成的主启动镜像一起使用，则应与 SBKEK 相同。可以通过 elftosb 和-K 开关生成，有关格式的更多信息参见第 4 章。应为采用十六进制字符串格式的 128/256 位 AES 密钥。

useKeyStore：[false, true]镜像是否包含密钥存储数据

keyStoreFile：如果使用了密钥存储(**useKeyStore** = true)，该文件将包含在镜像中。

imageKeyCertificate：镜像证书的路径，它将嵌入签名的镜像内。当 *outputImageAuthenticationType* 为["Signed", "Encrypted+Signed","Signed+Encrypted"]其中之一时，必须要有该字段

rootOfTrustKeyFile：可选的私钥，可用于对 *imageKeyCertificate* 执行额外检查。

privateImageKey：用于对镜像签名的私钥。当 *outputImageAuthenticationType* 为["Signed", "Encrypted+Signed","Signed+Encrypted"]其中之一时，必须要有该字段。

rootOfTrustKeyFilePassword：*rootOfTrustKeyFUe* 的密码（如果需要）。如果使用密码加密了 *rootOfTrustKeyFile*，必须要有该字段。

privateImageKeyPassword：*privateImageKey* 的密码（如果需要）。如果使用密码加密了 *privateImageKey*，必须要有该字段。

masterBootOutputFile：输出认证镜像的文件路径和文件名。始终需要。

8.3.1 lpc54x0xx 的镜像类型

相关详细信息，请参见 LPC540x00 用户指南。可从 nxp.com 获取。

8.4 lpc55xx、lpc55s1x、rt5xx 和 rt6xx 的 json 镜像配置文件

json 镜像配置文件的结构：

```
{
    "family": "lpc55xx",
    "inputImageFile": "C:/mkimage/images/image.bin",
    "imageLinkAddress": "0x0",
    "outputImageType": "Internal flash (XIP)",
    "outputImageAuthenticationType": "Signed",
    "outputImageEncryptionKeyFile": "",
    "enableTrustZone": false,
    "trustZonePresetFile": "C:/mkimage/tzm/tmz_preset.bin",
       "deviceKeySource": "",
    "useKeyStore": false,
    "keyStoreFile": "",
       "enableHwUserModeKeys": false,
       "imageBuildNumber": "0x0",
    "rootCertificate0File": "C:/mkimage/keys_and_certs/rootCert0_CA_selfSign.crt",
    "chainCertificate0File0": "C:/mkimage/keys_and_certs/Cert0_CA.crt",
    "chainCertificate0File1": "C:/mkimage/keys_and_certs/Cert1_noCA.crt",
    "rootCertificate1File": "C:/mkimage/keys_and_certs/rootCert1_noCA_ selfSign.crt",
    "rootCertificate2File": "",
    "rootCertificate3File": "",
    "mainCertChainId": 1,
    "mainCertPrivateKeyFile": "C:/mkimage/keys_and_certs/rootCert1PrivateKey.pem",
    "masterBootOutputFile": "C:/mkimage/signed_images/master_boot_image.bin"
}
```

json 镜像配置文件中的字段说明：

Family：["lpc55xx", "rt6xx"]

inputImageFile：包含要处理的应用程序（镜像）的 bin 文件的路径

imageLinkAddress：输入镜像的起始地址

imageLinkAdressFromImage：[false, true]将从镜像源加载/保留起始地址

outputImageExecutionTarget：["Internal flash (XIP)", "External flash (XIP)", "RAM"]基于目标系列

outputImageAuthenticationType：["CRC", "Signed", "Encrypted + Signed"]基于目标系列

outputImageEncryptionKeyFile：用于"Signed"（HMAC 密钥派生）和"Encrypted + Signed"（HMAC 密钥派生+镜像加密）镜像类型的加密密钥(USERKEY)的路径。密钥可以通过 elftosb 和-K 开关生成。应为采用十六进制字符串格式的 256 位 AES 密钥（有关格式的更多信息参见第 4 章）。

enableTrustZone：[false, true]将镜像类型设置为启用或禁用 TrustZone-M

trustZonePresetFile：如果启用 TrustZone-M (**enableTrustZone** = true)，将在镜像中包含该 TrustZone-M 预设文件，或保持""值，从而不在使用镜像中的预设配置。文件类型可以是*.bin（文件内容将直接插入镜像中）或*.json，其中 json 文件包含附录 E 中所述的 TrustZone-M 预设配置，将处理该文件，并将镜像生成过程中创建的二进制数据包含在镜像中。

deviceKeySource：["otp, "keyStore"]选项仅对 rt5xx 和 rt6xx 有效。它指定是将 PUF 密钥存储区用作密钥源，还是从 OTP 密钥派生密钥

useKeyStore：[false, true]镜像是否包含密钥存储数据

keyStoreFile：如果使用了密钥存储(**useKeyStore** = true)，此文件将包含在镜像中，如果未指定文件，则仅在镜像结构中保留值为 0 的可用空间区域，以便进一步替换为真实数据

enableHwUserModeKeys：[false, true]控制安全硬件密钥总线的标志。如果为 true，则可以从非安全应用访问硬件安全总线上的密钥，否则非安全应用将读取零

imageBuildNumber：镜像的版本号。将该值与器件中的单调计数器值进行比较，如果更低，则无法启动镜像

rootCertificate0File：根证书 0 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

chainCertificate1File0：链中第一个证书的路径，应为采用 DER 编码的 X.509 v3 证书

chainCertificate1File1：链中第二个证书的路径（使用**“chainCertificate1FileN”**，其中 N 为链中的最后一个证书的编号），应为采用 DER 编码的 X.509 v3 证书，链中的所有证书必须为 CA，除了最后一个不能为 CA

rootCertificate1File：根证书 1 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

rootCertificate2File：根证书 2 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

rootCertificate3File：根证书 3 的路径，可以不是 CA 自签名证书，也可以是 CA 自签名证书，后跟链中的其他证书，应为采用 DER 编码的 X.509 v3 证书

mainCertChainId [0,1,2,3]：应该使用哪个根证书或证书链对镜像签名

mainCertPrivateKeyFile：用于签名的证书的私钥（不是 CA 自签名根证书或链中的最后一个证书）

masterBootOutputFile：输出认证镜像的文件路径和文件名

8.4.1 lpc55xx 和 rt6xx 的镜像类型

基本上存在两种主要的镜像类型：加载到 RAM 镜像，将会上传至 RAM（通常是从远程位置），或 XIP（就地执行）镜像，将根据设备配置直接在内部或外部闪存中执行这些镜像。

加载到 RAM 镜像可以是“带 CRC 的未签名纯文本”、“签名纯文本”和“加密+签名”。带数字签名的加载到 RAM 镜像可以包含可选的密钥存储段，其中包含安全启动所需的密钥，供没有非易失性存储器的设备用来存储密钥。

XIP 镜像可以是“带 CRC 的未签名纯文本”和“签名纯文本”。

所有镜像类型都可以包含可选的 TrustZone-M 预设数据。有关 TrustZone-M 预设数据生成的更多信息参见附录 E。

以下小节将简单介绍每种镜像类型的结构。

8.4.1.1 加载到 RAM 镜像

带 CRC 校验和的未签名纯文本镜像：

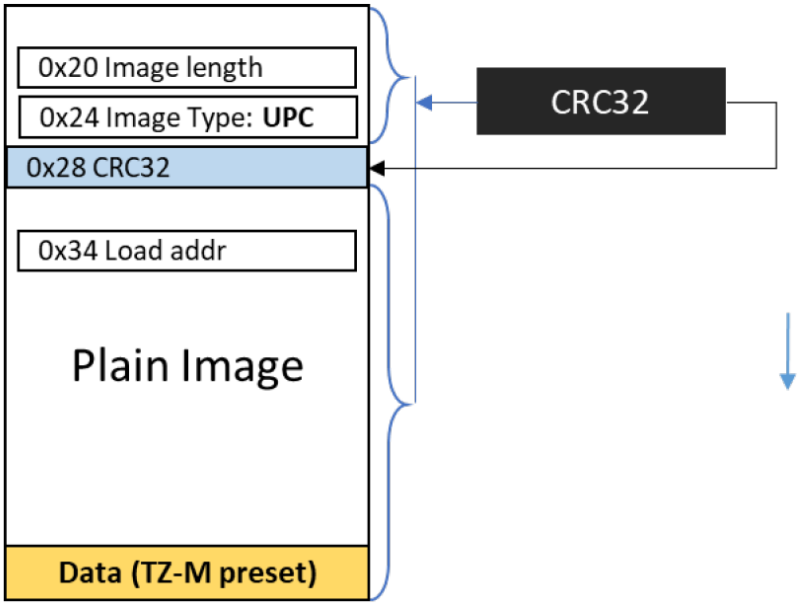

图 6. 带 CRC 校验和的未签名纯文本镜像

签名纯文本镜像：

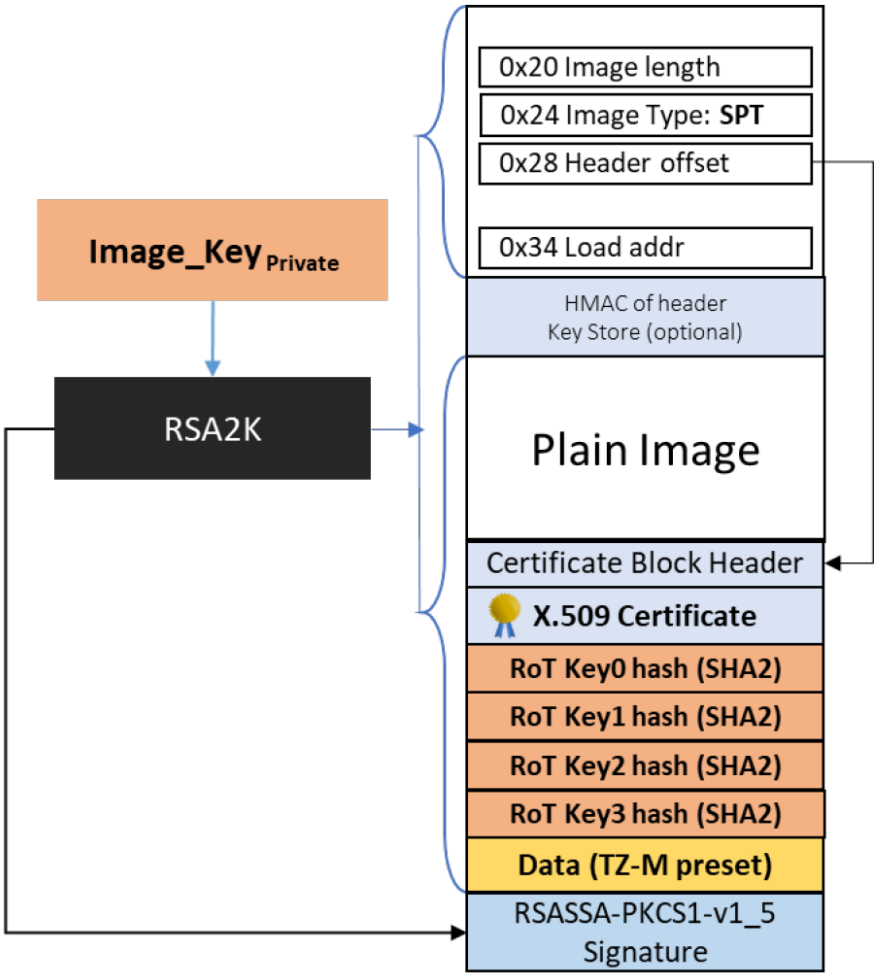

图 7. 签名纯文本镜像

加密签名镜像：

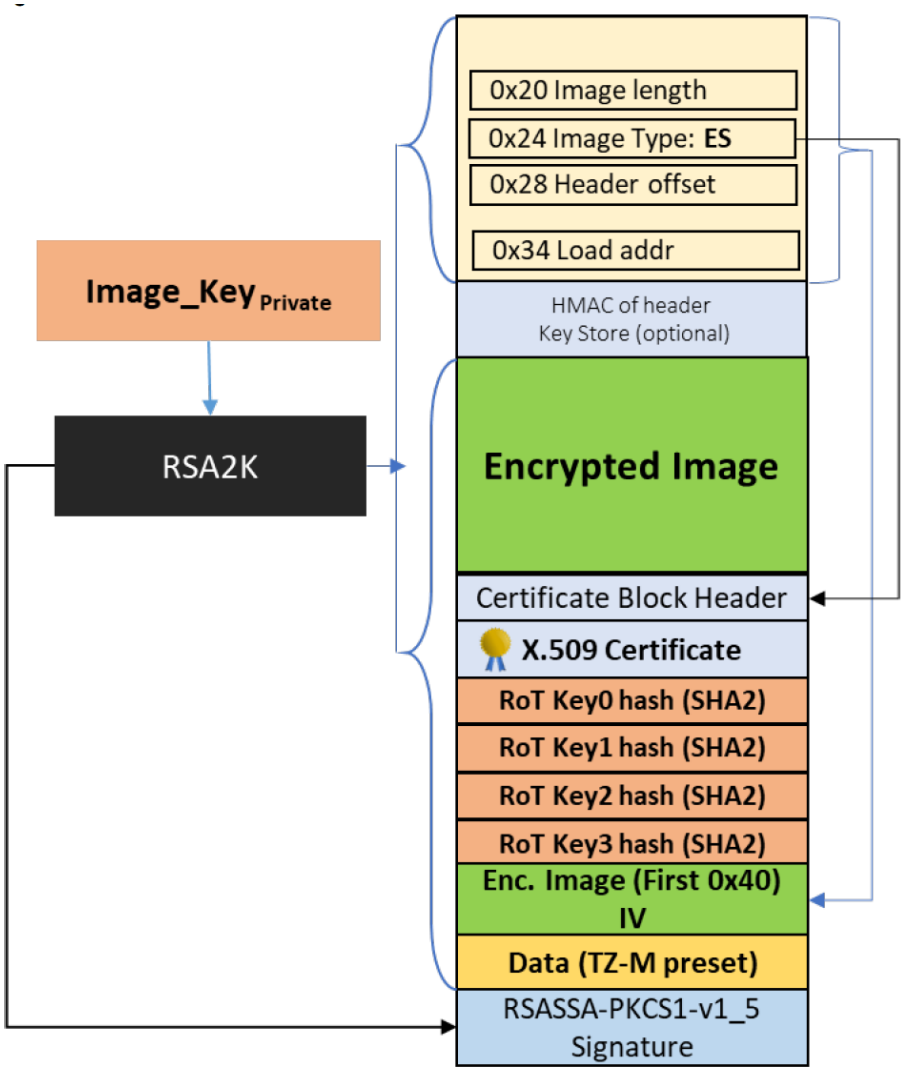

图 8. 加密签名镜像

8.4.1.2 XIP（就地执行）镜像

带 CRC 校验和的未签名纯文本镜像：

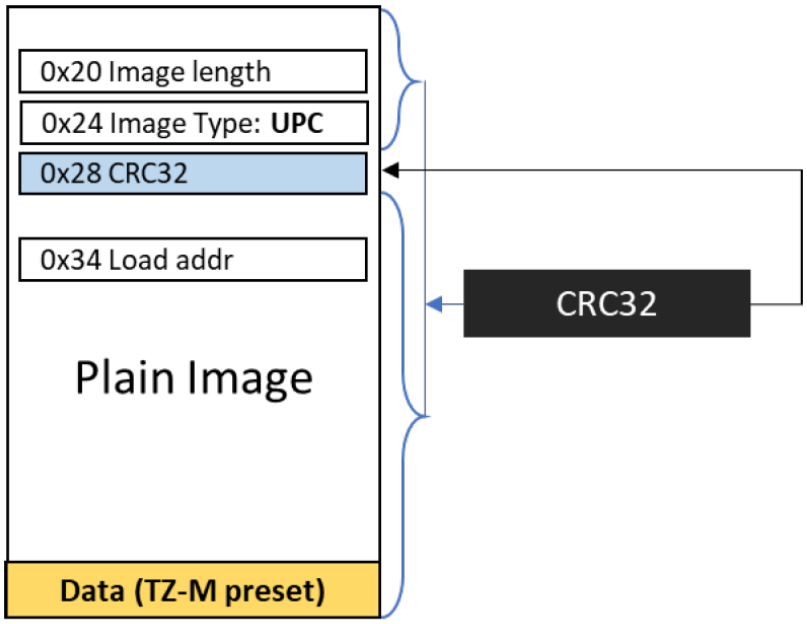

图 9. 带 CRC 校验和的未签名纯文本镜像

签名纯文本镜像：

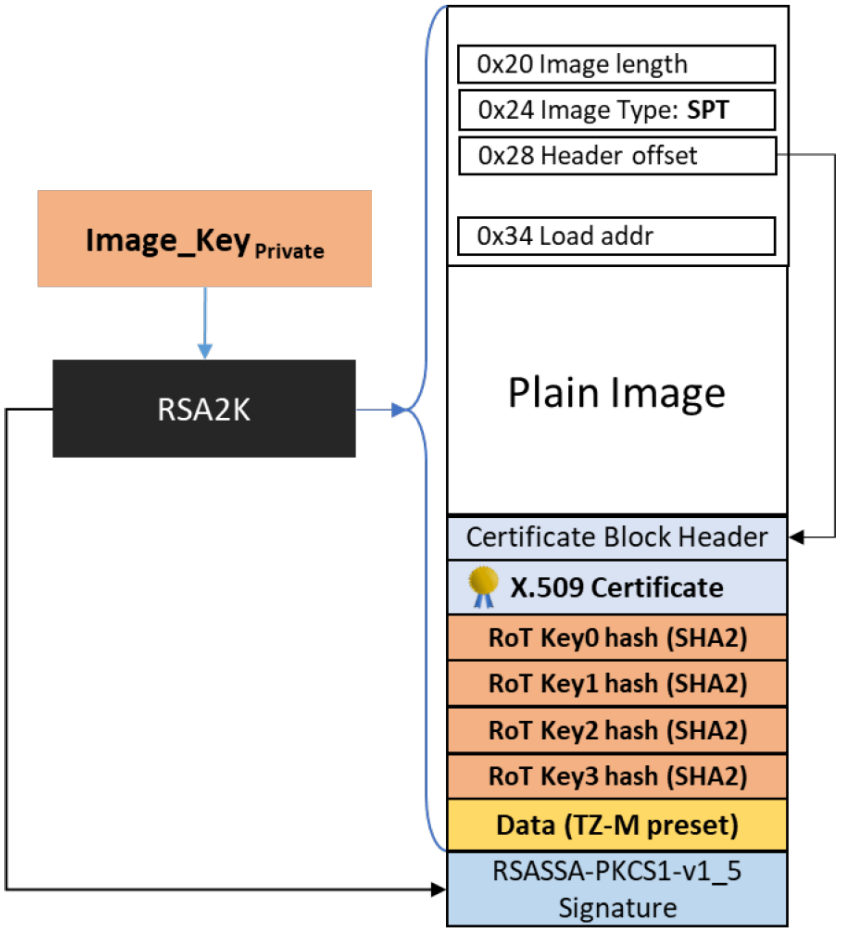

图 10. 签名纯文本镜像

9 附录 E：TrustZone-M 预设文件生成

9.1 概述

lpc55xx 和 rt6xx 器件系列支持生成 TrustZone-M 预设配置文件。二进制输出配置文件根据提供的输入 jason TrustZone-M 配置文件创建，其中包含配置规范和寄存器的值。

调用 elftosb 的示例：

e/ftosb -f lpc55xx -T pathToJsonFile

其中

-f = 系列 [lpc55xx, lpc55s1x, rt5xx, rt6xx]

-T = 要处理的 json 文件

9.2 Json TrustZone-M 预设配置文件

TrustZone-M 预设 json 配置文件的结构：

```
{
    "family": "rt6xx",
    "revision": "a0",
    "tzpOutputFile": "C:/Work/TrustZone/tzFile.bin",
    "trustZonePreset": {
            "Secure vector table address (vtor_addr)": "0xabcdef ",
            "Interrupt target non-secure register 0 (nvic_itns0)": "0x55",
            "Interrupt target non-secure register 1 (nvic_itns1)": "0x68254",
            .
            .
            .
            "Non-secure MPU Region 0 Limit Address Register (mpu_rlar0_ns)": "0x0",
            "SAU Region 5 Base Address Register (sau_rbar5)": "0x36",
            "SAU Region 5 Limit Address Register (sau_rlar5)": "0x2",
                        }
}
```

TrustZone-M 预设 json 配置文件中的字段说明：

"family" ["lpc55xx", "lpc55s1x", "rt5xx", "rt6xx"]

"revision"字符串值，指定目标器件的修订版本号。如果该选项缺失，将根据已知的最新 TZ-M 预设配置生成输出

"tzpOutputFile"输出文件的路径和名称

"trustZonePreset" 包含寄存器列表的 json 对象，应将其更改为 TrustZone-M 预设配置的默认值，未列出的寄存器将获得默认（复位）值。配置文件中寄存器的顺序并不重要。如果同一个寄存器出现多次，则将使用最后一个值。

"Secure vector table address (vtor_addr)"十六进制寄存器值，有效值从 0x0 到 Oxffffffff

9.3 TrustZone-M 预设寄存器

9.3.1 lpc55xx A0 的 TrustZone-M 预设寄存器

lpc55xx A0 的 Json TrustZone-M 配置文件，包含所有 TrustZone-M 预设寄存器及默认（复位）值：

```
{
 "family": "lpc55xx"
 "revision": "a0",
 "tzpOutputFile":"C:/Work/TrustZone/tzFile.bin",
 "trustZonePreset": {
                      "CM33 Secure vector table address (cm33_vtor_addr)": "0x0u",
                      "CM33 Non-secure vector table address (cm33_vtor_ns_addr)": "0x0u",
                      "CM33 Interrupt target non-secure register 0 (cm33_nvic_itns0)": "0x0u",
                      "CM33 Interrupt target non-secure register 1 (cm33_nvic_itns1)": "0x0u",
                      "MCM33 Secure vector table address (mcm33_vtor_addr)": "0x0u",
                      "MPU Control Register.(cm33_mpu_ctrl)": "0x0u",
                      "MPU Memory Attribute Indirection Register 0 (cm33_mpu_mair0)": "0x0u",
                      "MPU Memory Attribute Indirection Register 1 (cm33_mpu_mair1)": "0x0u",
                      "MPU Region 0 Base Address Register (cm33_mpu_rbar0)": "0x0u",
                      "MPU Region 0 Limit Address Register (cm33_mpu_rlar0)": "0x0u",
                      "MPU Region 1 Base Address Register (cm33_mpu_rbar1)": "0x0u",
                      "MPU Region 1 Limit Address Register (cm33_mpu_rlar1)": "0x0u",
                      "MPU Region 2 Base Address Register (cm33_mpu_rbar2)": "0x0u",
                      "MPU Region 2 Limit Address Register (cm33_mpu_rlar2)": "0x0u",
                      "MPU Region 3 Base Address Register (cm33_mpu_rbar3)": "0x0u",
                      "MPU Region 3 Limit Address Register (cm33_mpu_rlar3)": "0x0u",
                     "MPU Region 4 Base Address Register (cm33_mpu_rbar4)": "0x0u",
                      "MPU Region 4 Limit Address Register (cm33_mpu_rlar4)": "0x0u",
                      "MPU Region 5 Base Address Register (cm33_mpu_rbar5)": "0x0u",
                      "MPU Region 5 Limit Address Register (cm33_mpu_rlar5)": "0x0u",
                      "MPU Region 6 Base Address Register (cm33_mpu_rbar6)": "0x0u",
                      "MPU Region 6 Limit Address Register (cm33_mpu_rlar6)": "0x0u",
                      "MPU Region 7 Base Address Register (cm33_mpu_rbar7)": "0x0u",
                      "MPU Region 7 Limit Address Register (cm33_mpu_rlar7)": "0x0u",
                      "Non-secure MPU Control Register.(cm33_mpu_ctrl_ns)": "0x0u",
                      "Non-secure MPU Memory Attribute Indirection Register 0 (cm33_mpu_mair0_ns)":
"0x0u",
                      "Non-secure MPU Memory Attribute Indirection Register 1 (cm33_mpu_mair1_ns)":
"0x0u",
                      "Non-secure MPU Region 0 Base Address Register (cm33_mpu_rbar0_ns)": "0x0u",
                      "Non-secure MPU Region 0 Limit Address Register (cm33_mpu_rlar0_ns)": "0x0u",
                      "Non-secure MPU Region 1 Base Address Register (cm33_mpu_rbar1_ns)": "0x0u",
                      "Non-secure MPU Region 1 Limit Address Register (cm33_mpu_rlar1_ns)": "0x0u",
                      "Non-secure MPU Region 2 Base Address Register (cm33_mpu_rbar2_ns)": "0x0u",
                      "Non-secure MPU Region 2 Limit Address Register (cm33_mpu_rlar2_ns)": "0x0u",
                      "Non-secure MPU Region 3 Base Address Register (cm33_mpu_rbar3_ns)": "0x0u",
                      "Non-secure MPU Region 3 Limit Address Register (cm33_mpu_rlar3_ns)": "0x0u",
                      "Non-secure MPU Region 4 Base Address Register (cm33_mpu_rbar4_ns)": "0x0u",
                      "Non-secure MPU Region 4 Limit Address Register (cm33_mpu_rlar4_ns)": "0x0u",
                      "Non-secure MPU Region 5 Base Address Register (cm33_mpu_rbar5_ns)": "0x0u",
                      "Non-secure MPU Region 5 Limit Address Register (cm33_mpu_rlar5_ns)": "0x0u",
                      "Non-secure MPU Region 6 Base Address Register (cm33_mpu_rbar6_ns)": "0x0u",
                      "Non-secure MPU Region 6 Limit Address Register (cm33_mpu_rlar6_ns)" :"0x0u",
                      "Non-secure MPU Region 7 Base Address Register (cm33_mpu_rbar7_ns)": "0x0u",
                      "Non-secure MPU Region 7 Limit Address Register (cm33_mpu_rlar7_ns)": "0x0u",
                      "SAU Control Register.(cm33_sau_ctrl)": "0x0u",
                      "SAU Region 0 Base Address Register (cm33_sau_rbar0)": "0x0u",
                      "SAU Region 0 Limit Address Register (cm33_sau_rlar0)": "0x0u",
                      "SAU Region 1 Base Address Register (cm33_sau_rbar1)": "0x0u",
                      "SAU Region 1 Limit Address Register (cm33_sau_rlar1)": "0x0u",
                      "SAU Region 2 Base Address Register (cm33_sau_rbar2)": "0x0u",
```

```
                    "SAU Region 2 Limit Address Register (cm33_sau_rlar2)": "0x0u",
                    "SAU Region 3 Base Address Register (cm33_sau_rbar3)": "0x0u",
                    "SAU Region 3 Limit Address Register (cm33_sau_rlar3)": "0x0u",
                    "SAU Region 4 Base Address Register (cm33_sau_rbar4)": "0x0u",
                    "SAU Region 4 Limit Address Register (cm33_sau_rlar4)": "0x0u",
                    "SAU Region 5 Base Address Register (cm33_sau_rbar5)": "0x0u",
                    "SAU Region 5 Limit Address Register (cm33_sau_rlar5)": "0x0u",
                    "SAU Region 6 Base Address Register (cm33_sau_rbar6)": "0x0u",
                    "SAU Region 6 Limit Address Register (cm33_sau_rlar6)": "0x0u",
                    "SAU Region 7 Base Address Register (cm33_sau_rbar7)": "0x0u",
                    "SAU Region 7 Limit Address Register (cm33_sau_rlar7)": "0x0u",
                    "FLASH/ROM Slave Rule Register 0 (flash_rom_slave_rule)": "0x0u",
                    "FLASH Memory Rule Register 0 (flash_mem_rule0)": "0x0u",
                    "FLASH Memory Rule Register 1 (flash_mem_rule1)": "0x0u",
                    "FLASH Memory Rule Register 2 (flash_mem_rule2)": "0x0u",
                    "ROM Memory Rule Register 0 (rom_mem_rule0)": "0x0u",
                    "ROM Memory Rule Register 1 (rom_mem_rule1)": "0x0u",
                    "ROM Memory Rule Register 2 (rom_mem_rule2)": "0x0u",
                    "ROM Memory Rule Register 3 (rom_mem_rule3)": "0x0u",
                    "RAMX Slave Rule Register (ramx_slave_rule)": "0x0u",
                    "RAMX Memory Rule Register 0 (ramx_mem_rule0)": "0x0u",
                    "RAM0 Slave Rule Register (ram0_slave_rule)": "0x0u",
                    "RAM0 Memory Rule Register 0 (ram0_mem_rule0)": "0x0u",
                    "RAM0 Memory Rule Register 1 (ram0_mem_rule1)": "0x0u",
                    "RAM1 Slave Rule Register (ram1_slave_rule)": "0x0u",
                    "RAM1 Memory Rule Register 0 (ram1_mem_rule0)": "0x0u",
                    "RAM1 Memory Rule Register 1 (ram1_mem_rule1)": "0x0u",
                    "RAM2 Slave Rule Register (ram2_slave_rule)": "0x0u",
                    "RAM2 Memory Rule Register 0 (ram2_mem_rule0)": "0x0u",
                    "RAM2 Memory Rule Register 1 (ram2_mem_rule1)": "0x0u",
                    "RAM3 Slave Rule Register (ram3_slave_rule)": "0x0u",
                    "RAM3 Memory Rule Register 0 (ram3_mem_rule0)": "0x0u",
                    "RAM3 Memory Rule Register 1 (ram3_mem_rule1)": "0x0u",
                    "RAM4 Slave Rule Register (ram4_slave_rule)": "0x0u",
                    "RAM4 Memory Rule Register 0 (ram4_mem_rule0)": "0x0u",
                    "APB Bridge Group Slave Rule Register (apb_grp_slave_rule)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 0 (apb_grp0_mem_rule0)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 1 (apb_grp0_mem_rule1)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 2 (apb_grp0_mem_rule2)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 3 (apb_grp0_mem_rule3)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 0 (apb_grp1_mem_rule0)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 1 (apb_grp1_mem_rule1)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 2 (apb_grp1_mem_rule2)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 3 (apb_grp1_mem_rule3)": "0x0u",
                    "AHB Peripherals 0 Slave Rule Register 0 (ahb_periph0_slave_rule0)": "0x0u",
                    "AHB Peripherals 0 Slave Rule Register 1 (ahb_periph0_slave_rule1)": "0x0u",
                    "AHB Peripherals 1 Slave Rule Register 0 (ahb_periph1_slave_rule0)": "0x0u",
                    "AHB Peripherals 1 Slave Rule Register 1 (ahb_periph1_slave_rule1)": "0x0u",
                    "AHB Peripherals 2 Slave Rule Register 0 (ahb_periph2_slave_rule0)": "0x0u",
                    "AHB Peripherals 2 Slave Rule Register 1 (ahb_periph2_slave_rule1)": "0x0u",
                    "AHB Peripherals 2 Memory Rule Register 0 (ahb_periph2_mem_rule0)": "0x0u",
                    "HS USB Slave Rule Register 0 (usb_hs_slave_rule0)": "0x0u",
                    "HS USB Memory Rule Register 0 (usb_hs__mem_rule0)": "0x0u",
                    "Secure GPIO Register 0 (sec_gp_reg0)": "0xffffffffu",
                    "Secure GPIO Register 1 (sec_gp_reg1)": "0xffffffffu",
                    "Secure GPIO Register 2 (sec_gp_reg2)": "0xffffffffu",
                    "Secure GPIO Register 3 (sec_gp_reg3)": "0xffffffffu",
                    "Secure Interrupt Mask for CPU1 Register 0 (sec_int_reg0)": "0xffffffffu",
                    "Secure Interrupt Mask for CPU1 Register 1 (sec_int_reg1)": "0xffffffffu",
                    "Secure GPIO Lock Register (sec_gp_reg_lock)": "0x00000aaau",
```

```
                        "Master Secure Level Register (master_sec_reg)": "0x80000000u",
                        "Master Secure Level Anti-pole Register (master_sec_anti_pol_reg)":
"0xbfffffffu",
                        "CM33 Lock Control Register (cm33_lock_reg)": "0x800002aau",
                        "MCM33 Lock Control Register (mcm33_lock_reg)": "0x8000000au",
                        "Secure Control Duplicate Register (misc_ctrl_dp_reg)": "0x0000aaaau",
                        "Secure Control Register (misc_ctrl_reg)": "0x0000aaaau"
                    }
}
```

9.3.2 lpc55xx A1 的 TrustZone-M 预设寄存器

lpc55xx A1 的 Json TrustZone-M 配置文件，包含所有 TrustZone-M 预设寄存器及默认（复位）值：

```
{
  "family": "lpc55xx"
 "revision": "a1",
 "tzpOutputFile": "C:/Work/TrustZone/tzFile.bin",
 "trustZonePreset": {
                      "CM33 Secure vector table address (cm33_vtor_addr)": "0x0u",
                      "CM33 Non-secure vector table address (cm33_vtor_ns_addr)": "0x0u",
                      "CM33 Interrupt target non-secure register 0 (cm33_nvic_itns0)": "0x0u",
                      "CM33 Interrupt target non-secure register 1 (cm33_nvic_itns1)": "0x0u",
                      "MCM33 Secure vector table address (mcm33_vtor_addr)": "0x0u",
                      "MPU Control Register.(cm33_mpu_ctrl)": "0x0u",
                      "MPU Memory Attribute Indirection Register 0 (cm33_mpu_mair0)": "0x0u",
                      "MPU Memory Attribute Indirection Register 1 (cm33_mpu_mair1)": "0x0u",
                      "MPU Region 0 Base Address Register (cm33_mpu_rbar0)": "0x0u",
                      "MPU Region 0 Limit Address Register (cm33_mpu_rlar0)": "0x0u",
                      "MPU Region 1 Base Address Register (cm33_mpu_rbar1)": "0x0u",
                      "MPU Region 1 Limit Address Register (cm33_mpu_rlar1)": "0x0u",
                      "MPU Region 2 Base Address Register (cm33_mpu_rbar2)": "0x0u",
                      "MPU Region 2 Limit Address Register (cm33_mpu_rlar2)": "0x0u",
                      "MPU Region 3 Base Address Register (cm33_mpu_rbar3)": "0x0u",
                      "MPU Region 3 Limit Address Register (cm33_mpu_rlar3)": "0x0u",
                      "MPU Region 4 Base Address Register (cm33_mpu_rbar4)": "0x0u",
                      "MPU Region 4 Limit Address Register (cm33_mpu_rlar4)": "0x0u",
                      "MPU Region 5 Base Address Register (cm33_mpu_rbar5)": "0x0u",
                      "MPU Region 5 Limit Address Register (cm33_mpu_rlar5)": "0x0u",
                      "MPU Region 6 Base Address Register (cm33_mpu_rbar6)": "0x0u",
                      "MPU Region 6 Limit Address Register (cm33_mpu_rlar6)": "0x0u",
                      "MPU Region 7 Base Address Register (cm33_mpu_rbar7)": "0x0u",
                      "MPU Region 7 Limit Address Register (cm33_mpu_rlar7)": "0x0u",
                      "Non-secure MPU Control Register.(cm33_mpu_ctrl_ns)": "0x0u",
                      "Non-secure MPU Memory Attribute Indirection Register 0 (cm33_mpu_mair0_ns)":
"0x0u",
                      "Non-secure MPU Memory Attribute Indirection Register 1 (cm33_mpu_mair1_ns)":
"0x0u",
                      "Non-secure MPU Region 0 Base Address Register (cm33_mpu_rbar0_ns)": "0x0u",
                      "Non-secure MPU Region 0 Limit Address Register (cm33_mpu_rlar0_ns)": "0x0u",
                      "Non-secure MPU Region 1 Base Address Register (cm33_mpu_rbar1_ns)": "0x0u",
                      "Non-secure MPU Region 1 Limit Address Register (cm33_mpu_rlar1_ns)": "0x0u",
                      "Non-secure MPU Region 2 Base Address Register (cm33_mpu_rbar2_ns)": "0x0u",
                      "Non-secure MPU Region 2 Limit Address Register (cm33_mpu_rlar2_ns)": "0x0u",
                      "Non-secure MPU Region 3 Base Address Register (cm33_mpu_rbar3_ns)": "0x0u",
                      "Non-secure MPU Region 3 Limit Address Register (cm33_mpu_rlar3_ns)": "0x0u",
                      "Non-secure MPU Region 4 Base Address Register (cm33_mpu_rbar4_ns)": "0x0u",
                      "Non-secure MPU Region 4 Limit Address Register (cm33_mpu_rlar4_ns)": "0x0u",
                      "Non-secure MPU Region 5 Base Address Register (cm33_mpu_rbar5_ns)": "0x0u",
```

```
                    "Non-secure MPU Region 5 Limit Address Register (cm33_mpu_rlar5_ns)": "0x0u",
                    "Non-secure MPU Region 6 Base Address Register (cm33_mpu_rbar6_ns)": "0x0u",
                    "Non-secure MPU Region 6 Limit Address Register (cm33_mpu_rlar6_ns)": "0x0u",
                    "Non-secure MPU Region 7 Base Address Register (cm33_mpu_rbar7_ns)": "0x0u",
                    "Non-secure MPU Region 7 Limit Address Register (cm33_mpu_rlar7_ns)": "0x0u",
                    "SAU Control Register.(cm33_sau_ctrl)": "0x0u",
                    "SAU Region 0 Base Address Register (cm33_sau_rbar0)": "0x0u",
                    "SAU Region 0 Limit Address Register (cm33_sau_rlar0)": "0x0u",
                    "SAU Region 1 Base Address Register (cm33_sau_rbar1)": "0x0u",
                    "SAU Region 1 Limit Address Register (cm33_sau_rlar1)": "0x0u",
                    "SAU Region 2 Base Address Register (cm33_sau_rbar2)": "0x0u",
                    "SAU Region 2 Limit Address Register (cm33_sau_rlar2)": "0x0u",
                    "SAU Region 3 Base Address Register (cm33_sau_rbar3)": "0x0u",
                    "SAU Region 3 Limit Address Register (cm33_sau_rlar3)": "0x0u",
                    "SAU Region 4 Base Address Register (cm33_sau_rbar4)": "0x0u",
                    "SAU Region 4 Limit Address Register (cm33_sau_rlar4)": "0x0u",
                    "SAU Region 5 Base Address Register (cm33_sau_rbar5)": "0x0u",
                    "SAU Region 5 Limit Address Register (cm33_sau_rlar5)": "0x0u",
                    "SAU Region 6 Base Address Register (cm33_sau_rbar6)": "0x0u",
                    "SAU Region 6 Limit Address Register (cm33_sau_rlar6)": "0x0u",
                    "SAU Region 7 Base Address Register (cm33_sau_rbar7)": "0x0u",
                    "SAU Region 7 Limit Address Register (cm33_sau_rlar7)": "0x0u",
                    "FLASH/ROM Slave Rule Register 0 (flash_rom_slave_rule)": "0x0u",
                    "FLASH Memory Rule Register 0 (flash_mem_rule0)": "0x0u",
                    "FLASH Memory Rule Register 1 (flash_mem_rule1)": "0x0u",
                    "FLASH Memory Rule Register 2 (flash_mem_rule2)": "0x0u",
                    "ROM Memory Rule Register 0 (rom_mem_rule0)": "0x0u",
                    "ROM Memory Rule Register 1 (rom_mem_rule1)": "0x0u",
                    "ROM Memory Rule Register 2 (rom_mem_rule2)": "0x0u",
                    "ROM Memory Rule Register 3 (rom_mem_rule3)": "0x0u",
                    "RAMX Slave Rule Register (ramx_slave_rule)": "0x0u",
                    "RAMX Memory Rule Register 0 (ramx_mem_rule0)": "0x0u",
                    "RAM0 Slave Rule Register (ram0_slave_rule)": "0x0u",
                    "RAM0 Memory Rule Register 0 (ram0_mem_rule0)": "0x0u",
                    "RAM0 Memory Rule Register 1 (ram0_mem_rule1)": "0x0u",
                    "RAM1 Slave Rule Register (ram1_slave_rule)": "0x0u",
                    "RAM1 Memory Rule Register 0 (ram1_mem_rule0)": "0x0u",
                    "RAM1 Memory Rule Register 1 (ram1_mem_rule1)": "0x0u",
                    "RAM2 Slave Rule Register (ram2_slave_rule)": "0x0u",
                    "RAM2 Memory Rule Register 0 (ram2_mem_rule0)": "0x0u",
                    "RAM2 Memory Rule Register 1 (ram2_mem_rule1)": "0x0
                    "RAM3 Slave Rule Register (ram3_slave_rule)": "0x0u",
                    "RAM3 Memory Rule Register 0 (ram3_mem_rule0)": "0x0u",
                    "RAM3 Memory Rule Register 1 (ram3_mem_rule1)": "0x0u",
                    "RAM4 Slave Rule Register (ram4_slave_rule)": "0x0u",
                    "RAM4 Memory Rule Register 0 (ram4_mem_rule0)": "0x0u",
                    "APB Bridge Group Slave Rule Register (apb_grp_slave_rule)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 0 (apb_grp0_mem_rule0)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 1 (apb_grp0_mem_rule1)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 2 (apb_grp0_mem_rule2)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 3 (apb_grp0_mem_rule3)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 0 (apb_grp1_mem_rule0)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 1 (apb_grp1_mem_rule1)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 2 (apb_grp1_mem_rule2)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 3 (apb_grp1_mem_rule3)": "0x0u",
                    "AHB Peripherals 0 Slave Rule Register 0 (ahb_periph0_slave_rule0)": "0x0u",
                    "AHB Peripherals 0 Slave Rule Register 1 (ahb_periph0_slave_rule1)": "0x0u",
                    "AHB Peripherals 1 Slave Rule Register 0 (ahb_periph1_slave_rule0)": "0x0u",
                    "AHB Peripherals 1 Slave Rule Register 1 (ahb_periph1_slave_rule1)": "0x0u",
                    "AHB Peripherals 2 Slave Rule Register 0 (ahb_periph2_slave_rule0)": "0x0u",
```

```
                    "AHB Peripherals 2 Slave Rule Register 1 (ahb_periph2_slave_rule1)": "0x0u",
                    "AHB Peripherals 2 Memory Rule Register 0 (ahb_periph2_mem_rule0)": "0x0u",
                    "HS USB Slave Rule Register 0 (usb_hs_slave_rule0)": "0x0u",
                    "HS USB Memory Rule Register 0 (usb_hs__mem_rule0)": "0x0u",
                    "Secure GPIO Register 0 (sec_gp_reg0)": "0xffffffffu",
                    "Secure GPIO Register 1 (sec_gp_reg1)": "0xffffffffu",
                    "Secure GPIO Register 2 (sec_gp_reg2)": "0xffffffffu",
                    "Secure GPIO Register 3 (sec_gp_reg3)": "0xffffffffu",
                    "Secure Interrupt Mask for CPU1 Register 0 (sec_int_reg0)": "0xffffffffu",
                    "Secure Interrupt Mask for CPU1 Register 1 (sec_int_reg1)": "0xffffffffu",
                    "Secure GPIO Lock Register (sec_gp_reg_lock)": "0x00000aaau",
                    "Master Secure Level Register (master_sec_reg)": "0x80000000u",
                  "Master Secure Level Anti-pole Register (master_sec_anti_pol_reg)": "0xbfffffffu",
                    "CM33 Lock Control Register (cm33_lock_reg)": "0x800002aau",
                    "MCM33 Lock Control Register (mcm33_lock_reg)": "0x8000000au",
                    "Secure Control Duplicate Register (misc_ctrl_dp_reg)": "0x0000aaaau",
                    "Secure Control Register (misc_ctrl_reg)": "0x0000aaaau",
                    "Miscellaneous TZM settings (misc_tzm_settings)": "0x0u"
                  }
}
```

9.3.3 lpc55Ss1x 的 TrustZone-M 预设寄存器

lpc55Ss1x A0 的 Json TrustZone-M 配置文件，包含所有 TrustZone-M 预设寄存器及默认（复位）值：

```
{
 "family": "lpc55s1x"
 "revision": "a0",
 "tzpOutputFile": "C:/Work/TrustZone/tzFile.bin",
 "trustZonePreset": {
                    "CM33 Secure vector table address (cm33_vtor_addr)": "0x0u",
                    "CM33 Non-secure vector table address (cm33_vtor_ns_addr)": "0x0u",
                    "CM33 Interrupt target non-secure register 0 (cm33_nvic_itns0)": "0x0u",
                    "CM33 Interrupt target non-secure register 1 (cm33_nvic_itns1)": "0x0u",
                    "MPU Control Register.(cm33_mpu_ctrl)": "0x0u",
                    "MPU Memory Attribute Indirection Register 0 (cm33_mpu_mair0)": "0x0u",
                    "MPU Memory Attribute Indirection Register 1 (cm33_mpu_mair1)": "0x0u",
                    "MPU Region 0 Base Address Register (cm33_mpu_rbar0)": "0x0u",
                    "MPU Region 0 Limit Address Register (cm33_mpu_rlar0)": "0x0u",
                    "MPU Region 1 Base Address Register (cm33_mpu_rbar1)": "0x0u",
                    "MPU Region 1 Limit Address Register (cm33_mpu_rlar1)": "0x0u",
                    "MPU Region 2 Base Address Register (cm33_mpu_rbar2)": "0x0u",
                    "MPU Region 2 Limit Address Register (cm33_mpu_rlar2)": "0x0u",
                    "MPU Region 3 Base Address Register (cm33_mpu_rbar3)": "0x0u",
                    "MPU Region 3 Limit Address Register (cm33_mpu_rlar3)": "0x0u",
                    "MPU Region 4 Base Address Register (cm33_mpu_rbar4)": "0x0u",
                    "MPU Region 4 Limit Address Register (cm33_mpu_rlar4)": "0x0u",
                    "MPU Region 5 Base Address Register (cm33_mpu_rbar5)": "0x0u",
                    "MPU Region 5 Limit Address Register (cm33_mpu_rlar5)": "0x0u",
                    "MPU Region 6 Base Address Register (cm33_mpu_rbar6)": "0x0u",
                    "MPU Region 6 Limit Address Register (cm33_mpu_rlar6)": "0x0u",
                    "MPU Region 7 Base Address Register (cm33_mpu_rbar7)": "0x0u",
                    "MPU Region 7 Limit Address Register (cm33_mpu_rlar7)": "0x0u",
                    "Non-secure MPU Control Register.(cm33_mpu_ctrl_ns)": "0x0u",
                    "Non-secure MPU Memory Attribute Indirection Register 0 (cm33_mpu_mair0_ns)":
"0x0u",
                    "Non-secure MPU Memory Attribute Indirection Register 1 (cm33_mpu_mair1_ns)":
"0x0u",
                    "Non-secure MPU Region 0 Base Address Register (cm33_mpu_rbar0_ns)": "0x0u",
```

```
                    "Non-secure MPU Region 0 Limit Address Register (cm33_mpu_rlar0_ns)": "0x0u",
                    "Non-secure MPU Region 1 Base Address Register (cm33_mpu_rbar1_ns)": "0x0u",
                    "Non-secure MPU Region 1 Limit Address Register (cm33_mpu_rlar1_ns)": "0x0u",
                    "Non-secure MPU Region 2 Base Address Register (cm33_mpu_rbar2_ns)": "0x0u",
                    "Non-secure MPU Region 2 Limit Address Register (cm33_mpu_rlar2_ns)": "0x0u",
                    "Non-secure MPU Region 3 Base Address Register (cm33_mpu_rbar3_ns)": "0x0u",
                    "Non-secure MPU Region 3 Limit Address Register (cm33_mpu_rlar3_ns)": "0x0u",
                    "Non-secure MPU Region 4 Base Address Register (cm33_mpu_rbar4_ns)": "0x0u",
                    "Non-secure MPU Region 4 Limit Address Register (cm33_mpu_rlar4_ns)": "0x0u",
                    "Non-secure MPU Region 5 Base Address Register (cm33_mpu_rbar5_ns)": "0x0u",
                    "Non-secure MPU Region 5 Limit Address Register (cm33_mpu_rlar5_ns)": "0x0u",
                    "Non-secure MPU Region 6 Base Address Register (cm33_mpu_rbar6_ns)": "0x0u",
                    "Non-secure MPU Region 6 Limit Address Register (cm33_mpu_rlar6_ns)": "0x0u",
                    "Non-secure MPU Region 7 Base Address Register (cm33_mpu_rbar7_ns)": "0x0u",
                    "Non-secure MPU Region 7 Limit Address Register (cm33_mpu_rlar7_ns)": "0x0u",
                    "SAU Control Register.(cm33_sau_ctrl)": "0x0u",
                    "SAU Region 0 Base Address Register (cm33_sau_rbar0)": "0x0u",
                    "SAU Region 0 Limit Address Register (cm33_sau_rlar0)": "0x0u",
                    "SAU Region 1 Base Address Register (cm33_sau_rbar1)": "0x0u",
                    "SAU Region 1 Limit Address Register (cm33_sau_rlar1)": "0x0u",
                    "SAU Region 2 Base Address Register (cm33_sau_rbar2)": "0x0u",
                    "SAU Region 2 Limit Address Register (cm33_sau_rlar2)": "0x0u",
                    "SAU Region 3 Base Address Register (cm33_sau_rbar3)": "0x0u",
                    "SAU Region 3 Limit Address Register (cm33_sau_rlar3)": "0x0u",
                    "SAU Region 4 Base Address Register (cm33_sau_rbar4)": "0x0u",
                    "SAU Region 4 Limit Address Register (cm33_sau_rlar4)": "0x0u",
                    "SAU Region 5 Base Address Register (cm33_sau_rbar5)": "0x0u",
                    "SAU Region 5 Limit Address Register (cm33_sau_rlar5)": "0x0u",
                    "SAU Region 6 Base Address Register (cm33_sau_rbar6)": "0x0u",
                    "SAU Region 6 Limit Address Register (cm33_sau_rlar6)": "0x0u",
                    "SAU Region 7 Base Address Register (cm33_sau_rbar7)": "0x0u",
                    "SAU Region 7 Limit Address Register (cm33_sau_rlar7)": "0x0u",
                    "FLASH/ROM Slave Rule Register 0 (flash_rom_slave_rule)": "0x0u",
                    "FLASH Memory Rule Register 0 (flash_mem_rule0)": "0x0u",
                    "FLASH Memory Rule Register 1 (flash_mem_rule1)": "0x0u",
                    "FLASH Memory Rule Register 2 (flash_mem_rule2)": "0x0u",
                    "ROM Memory Rule Register 0 (rom_mem_rule0)": "0x0u",
                    "ROM Memory Rule Register 1 (rom_mem_rule1)": "0x0u",
                    "ROM Memory Rule Register 2 (rom_mem_rule2)": "0x0u",
                    "ROM Memory Rule Register 3 (rom_mem_rule3)": "0x0u",
                    "RAMX Slave Rule Register (ramx_slave_rule)": "0x0u",
                    "RAMX Memory Rule Register 0 (ramx_mem_rule0)": "0x0u",
                    "RAM0 Slave Rule Register (ram0_slave_rule)": "0x0u",
                    "RAM0 Memory Rule Register 0 (ram0_mem_rule0)": "0x0u",
                    "RAM0 Memory Rule Register 1 (ram0_mem_rule1)": "0x0u",
                    "RAM1 Slave Rule Register (ram1_slave_rule)": "0x0u",
                    "RAM1 Memory Rule Register 0 (ram1_mem_rule0)": "0x0u",
                    "RAM1 Memory Rule Register 1 (ram1_mem_rule1)": "0x0u",
                    "RAM2 Slave Rule Register (ram2_slave_rule)": "0x0u",
                    "RAM2 Memory Rule Register 0 (ram2_mem_rule0)": "0x0u",
                    "HS USB Slave Rule Register 0 (usb_hs_slave_rule0)": "0x0u",
                    "HS USB Memory Rule Register 0 (usb_hs__mem_rule0)": "0x0u",
                    "APB Bridge Group Slave Rule Register (apb_grp_slave_rule)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 0 (apb_grp0_mem_rule0)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 1 (apb_grp0_mem_rule1)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 2 (apb_grp0_mem_rule2)": "0x0u",
                    "APB Bridge Group 0 Memory Rule Register 3 (apb_grp0_mem_rule3)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 0 (apb_grp1_mem_rule0)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 1 (apb_grp1_mem_rule1)": "0x0u",
                    "APB Bridge Group 1 Memory Rule Register 2 (apb_grp1_mem_rule2)": "0x0u",
```

```
                    "APB Bridge Group 1 Memory Rule Register 3 (apb_grp1_mem_rule3)": "0x0u",
                    "AHB Peripherals 0 Slave Rule Register 0 (ahb_periph0_slave_rule0)": "0x0u",
                    "AHB Peripherals 0 Slave Rule Register 1 (ahb_periph0_slave_rule1)": "0x0u",
                    "AHB Peripherals 1 Slave Rule Register 0 (ahb_periph1_slave_rule0)": "0x0u",
                    "AHB Peripherals 1 Slave Rule Register 1 (ahb_periph1_slave_rule1)": "0x0u",
                    "AHB Peripherals 2 Slave Rule Register 0 (ahb_periph2_slave_rule0)": "0x0u",
                    "AHB Peripherals 2 Slave Rule Register 1 (ahb_periph2_slave_rule1)": "0x0u",
                    "AHB Peripherals 2 Memory Rule Register 0 (ahb_periph2_mem_rule0)": "0x0u",
                    "Secure GPIO Register 0 (sec_gp_reg0)": "0xffffffffu",
                    "Secure GPIO Register 1 (sec_gp_reg1)": "0xffffffffu",
                    "Secure GPIO Register 2 (sec_gp_reg2)": "0xffffffffu",
                    "Secure GPIO Lock Register (sec_gp_reg_lock)": "0x00000aaau",
                    "Master Secure Level Register (master_sec_reg)": "0x80000000u",
                  "Master Secure Level Anti-pole Register (master_sec_anti_pol_reg)": "0xbfffffffu",
                    "CM33 Lock Control Register (cm33_lock_reg)": "0x800002aau",
                    "Secure Control Duplicate Register (misc_ctrl_dp_reg)": "0x0000aaaau",
                    "Secure Control Register (misc_ctrl_reg)": "0x0000aaaau",
                    "Miscellaneous TZM settings (misc_tzm_settings)": "0x0u"
                  }
}
```

9.3.4 rt5xx 的 TrustZone-M 预设寄存器

rt5xx A0 的 Json TrustZone-M 配置文件，包含所有 TrustZone-M 预设寄存器及默认（复位）值：

```
{
 "family": "rt5xx",
 "revision": "a0",
 "tzpOutputFile": "C:/Work/TrustZone/tzFile.bin",
 "trustZonePreset": {
                    "Secure vector table address (vtor_addr)": "0x00000000",
                    "Non-secure vector table address (vtor_ns_addr)": "0x00000000",
                    "Interrupt target non-secure register 0 (nvic_itns0)": "0x00000000",
                    "Interrupt target non-secure register 1 (nvic_itns1)": "0x00000000",
                    "MPU Control Register (mpu_ctrl)": "0x00000000",
                    "MPU Memory Attribute Indirection Register 0 (mpu_mair0)": "0x00000000",
                    "MPU Memory Attribute Indirection Register 1 (mpu_mair1)": "0x00000000",
                    "MPU Region 0 Base Address Register (mpu_rbar0)": "0x00000000",
                    "MPU Region 0 Limit Address Register (mpu_rlar0)": "0x00000000",
                    "MPU Region 1 Base Address Register (mpu_rbar1)": "0x00000000",
                    "MPU Region 1 Limit Address Register (mpu_rlar1)": "0x00000000",
                    "MPU Region 2 Base Address Register (mpu_rbar2)": "0x00000000",
                    "MPU Region 2 Limit Address Register (mpu_rlar2)": "0x00000000",
                    "MPU Region 3 Base Address Register (mpu_rbar3)": "0x00000000",
                    "MPU Region 3 Limit Address Register (mpu_rlar3)": "0x00000000",
                    "MPU Region 4 Base Address Register (mpu_rbar4)": "0x00000000",
                    "MPU Region 4 Limit Address Register (mpu_rlar4)": "0x00000000",
                    "MPU Region 5 Base Address Register (mpu_rbar5)": "0x00000000",
                    "MPU Region 5 Limit Address Register (mpu_rlar5)": "0x00000000",
                    "MPU Region 6 Base Address Register (mpu_rbar6)": "0x00000000",
                    "MPU Region 6 Limit Address Register (mpu_rlar6)": "0x00000000",
                    "MPU Region 7 Base Address Register (mpu_rbar7)": "0x00000000",
                    "MPU Region 7 Limit Address Register (mpu_rlar7)": "0x00000000",
                    "Non-secure MPU Control Register (mpu_ctrl_ns)": "0x00000000",
                    "Non-secure MPU Memory Attribute Indirection Register 0 (mpu_mair0_ns)":
"0x00000000",
                    "Non-secure MPU Memory Attribute Indirection Register 1 (mpu_mair1_ns)":
"0x00000000",
                    "Non-secure MPU Region 0 Base Address Register (mpu_rbar0_ns)": "0x00000000",
```

```
                        "Non-secure MPU Region 0 Limit Address Register (mpu_rlar0_ns)": "0x00000000",
                        "Non-secure MPU Region 1 Base Address Register (mpu_rbar1_ns)": "0x00000000",
                        "Non-secure MPU Region 1 Limit Address Register (mpu_rlar1_ns)": "0x00000000",
                        "Non-secure MPU Region 2 Base Address Register (mpu_rbar2_ns)": "0x00000000",
                        "Non-secure MPU Region 2 Limit Address Register (mpu_rlar2_ns)": "0x00000000",
                        "Non-secure MPU Region 3 Base Address Register (mpu_rbar3_ns)": "0x00000000",
                        "Non-secure MPU Region 3 Limit Address Register (mpu_rlar3_ns)": "0x00000000",
                        "Non-secure MPU Region 4 Base Address Register (mpu_rbar4_ns)": "0x00000000",
                        "Non-secure MPU Region 4 Limit Address Register (mpu_rlar4_ns)": "0x00000000",
                        "Non-secure MPU Region 5 Base Address Register (mpu_rbar5_ns)": "0x00000000",
                        "Non-secure MPU Region 5 Limit Address Register (mpu_rlar5_ns)": "0x00000000",
                        "Non-secure MPU Region 6 Base Address Register (mpu_rbar6_ns)": "0x00000000",
                        "Non-secure MPU Region 6 Limit Address Register (mpu_rlar6_ns)": "0x00000000",
                        "Non-secure MPU Region 7 Base Address Register (mpu_rbar7_ns)": "0x00000000",
                        "Non-secure MPU Region 7 Limit Address Register (mpu_rlar7_ns)": "0x00000000",
                        "SAU Control Register (sau_ctrl)": "0x00000000",
                        "SAU Region 0 Base Address Register (sau_rbar0)": "0x00000000",
                        "SAU Region 0 Limit Address Register (sau_rlar0)": "0x00000000",
                        "SAU Region 1 Base Address Register (sau_rbar1)": "0x00000000",
                        "SAU Region 1 Limit Address Register (sau_rlar1)": "0x00000000",
                        "SAU Region 2 Base Address Register (sau_rbar2)": "0x00000000",
                        "SAU Region 2 Limit Address Register (sau_rlar2)": "0x00000000",
                        "SAU Region 3 Base Address Register (sau_rbar3)": "0x00000000",
                        "SAU Region 3 Limit Address Register (sau_rlar3)": "0x00000000",
                        "SAU Region 4 Base Address Register (sau_rbar4)": "0x00000000",
                        "SAU Region 4 Limit Address Register (sau_rlar4)": "0x00000000",
                        "SAU Region 5 Base Address Register (sau_rbar5)": "0x00000000",
                        "SAU Region 5 Limit Address Register (sau_rlar5)": "0x00000000",
                        "SAU Region 6 Base Address Register (sau_rbar6)": "0x00000000",
                        "SAU Region 6 Limit Address Register (sau_rlar6)": "0x00000000",
                        "SAU Region 7 Base Address Register (sau_rbar7)": "0x00000000",
                        "SAU Region 7 Limit Address Register (sau_rlar7)": "0x00000000",
                        "ROM Slave Rule Register 0 (bootrom0_slave_rule0)": "0x00000000",
                        "ROM Memory Rule Register 0 (bootrom0_mem_rule0)": "0x00000000",
                        "ROM Memory Rule Register 1 (bootrom0_mem_rule1)": "0x00000000",
                        "ROM Memory Rule Register 2 (bootrom0_mem_rule2)": "0x00000000",
                        "ROM Memory Rule Register 3 (bootrom0_mem_rule3)": "0x00000000",
                        "Quad/Octal SPI Slave Rule Register 0 (qospi_slave_rule0)": "0x00000000",
                        "Quad/Octal SPI 0 Memory Rule Register 0 (qospi0_mem_rule0)": "0x00000000",
                        "Quad/Octal SPI 0 Memory Rule Register 1 (qospi0_mem_rule1)": "0x00000000",
                        "Quad/Octal SPI 0 Memory Rule Register 2 (qospi0_mem_rule2)": "0x00000000",
                        "Quad/Octal SPI 0 Memory Rule Register 3 (qospi0_mem_rule3)": "0x00000000",
                        "Quad/Octal SPI 1 Memory Rule Register 0 (qospi1_mem_rule0)": "0x00000000",
                        "Quad/Octal SPI 1 Memory Rule Register 1 (qospi1_mem_rule1)": "0x00000000",
                        "Quad/Octal SPI 1 Memory Rule Register 2 (qospi1_mem_rule2)": "0x00000000",
                        "Quad/Octal SPI 1 Memory Rule Register 3 (qospi1_mem_rule3)": "0x00000000",
                        "Quad/Octal SPI 2 Memory Rule Register 0 (qospi2_mem_rule0)": "0x00000000",
                        "Quad/Octal SPI 2 Memory Rule Register 1 (qospi2_mem_rule1)": "0x00000000",
                        "Quad/Octal SPI 2 Memory Rule Register 2 (qospi2_mem_rule2)": "0x00000000",
                        "Quad/Octal SPI 2 Memory Rule Register 3 (qospi2_mem_rule3)": "0x00000000",
                        "Quad/Octal SPI 3 Memory Rule Register 0 (qospi3_mem_rule0)": "0x00000000",
                        "Quad/Octal SPI 3 Memory Rule Register 1 (qospi3_mem_rule1)": "0x00000000",
                        "Quad/Octal SPI 3 Memory Rule Register 2 (qospi3_mem_rule2)": "0x00000000",
                        "Quad/Octal SPI 3 Memory Rule Register 3 (qospi3_mem_rule3)": "0x00000000",
                        "Quad/Octal SPI 4 Memory Rule Register 0 (qospi4_mem_rule0)": "0x00000000",
                        "Quad/Octal SPI 4 Memory Rule Register 1 (qospi4_mem_rule1)": "0x00000000",
                        "Quad/Octal SPI 4 Memory Rule Register 2 (qospi4_mem_rule2)": "0x00000000",
                        "Quad/Octal SPI 4 Memory Rule Register 3 (qospi4_mem_rule3)": "0x00000000",
                        "RAM0 Slave Rule Register (ram0_slave_rule)": "0x00000000",
                        "RAM00 Memory Rule Register 0 (ram00_mem_rule0)": "0x00000000",
```

```
                    "RAM00 Memory Rule Register 1 (ram00_mem_rule1)": "0x00000000",
                    "RAM00 Memory Rule Register 2 (ram00_mem_rule2)": "0x00000000",
                    "RAM00 Memory Rule Register 3 (ram00_mem_rule3)": "0x00000000",
                    "RAM01 Memory Rule Register 0 (ram01_mem_rule0)": "0x00000000",
                    "RAM01 Memory Rule Register 1 (ram01_mem_rule1)": "0x00000000",
                    "RAM01 Memory Rule Register 2 (ram01_mem_rule2)": "0x00000000",
                    "RAM01 Memory Rule Register 3 (ram01_mem_rule3)": "0x00000000",
                    "RAM1 Slave Rule Register (ram1_slave_rule)": "0x00000000",
                    "RAM10 Memory Rule Register 0 (ram10_mem_rule0)": "0x00000000",
                    "RAM10 Memory Rule Register 1 (ram10_mem_rule1)": "0x00000000",
                    "RAM10 Memory Rule Register 2 (ram10_mem_rule2)": "0x00000000",
                    "RAM10 Memory Rule Register 3 (ram10_mem_rule3)": "0x00000000",
                    "RAM11 Memory Rule Register 0 (ram11_mem_rule0)": "0x00000000",
                    "RAM11 Memory Rule Register 1 (ram11_mem_rule1)": "0x00000000",
                    "RAM11 Memory Rule Register 2 (ram11_mem_rule2)": "0x00000000",
                    "RAM11 Memory Rule Register 3 (ram11_mem_rule3)": "0x00000000",
                    "RAM2 Slave Rule Register (ram2_slave_rule)": "0x00000000",
                    "RAM20 Memory Rule Register 0 (ram20_mem_rule0)": "0x00000000",
                    "RAM20 Memory Rule Register 1 (ram20_mem_rule1)": "0x00000000",
                    "RAM20 Memory Rule Register 2 (ram20_mem_rule2)": "0x00000000",
                    "RAM20 Memory Rule Register 3 (ram20_mem_rule3)": "0x00000000",
                    "RAM21 Memory Rule Register 0 (ram21_mem_rule0)": "0x00000000",
                    "RAM21 Memory Rule Register 1 (ram21_mem_rule1)": "0x00000000",
                    "RAM21 Memory Rule Register 2 (ram21_mem_rule2)": "0x00000000",
                    "RAM21 Memory Rule Register 3 (ram21_mem_rule3)": "0x00000000",
                    "RAM22 Memory Rule Register 0 (ram22_mem_rule0)": "0x00000000",
                    "RAM22 Memory Rule Register 1 (ram22_mem_rule1)": "0x00000000",
                    "RAM22 Memory Rule Register 2 (ram22_mem_rule2)": "0x00000000",
                    "RAM22 Memory Rule Register 3 (ram22_mem_rule3)": "0x00000000",
                    "RAM23 Memory Rule Register 0 (ram23_mem_rule0)": "0x00000000",
                    "RAM23 Memory Rule Register 1 (ram23_mem_rule1)": "0x00000000",
                    "RAM23 Memory Rule Register 2 (ram23_mem_rule2)": "0x00000000",
                    "RAM23 Memory Rule Register 3 (ram23_mem_rule3)": "0x00000000",
                    "RAM3 Slave Rule Register (ram3_slave_rule)": "0x00000000",
                    "RAM30 Memory Rule Register 0 (ram30_mem_rule0)": "0x00000000",
                    "RAM30 Memory Rule Register 1 (ram30_mem_rule1)": "0x00000000",
                    "RAM30 Memory Rule Register 2 (ram30_mem_rule2)": "0x00000000",
                    "RAM30 Memory Rule Register 3 (ram30_mem_rule3)": "0x00000000",
                    "RAM31 Memory Rule Register 0 (ram31_mem_rule0)": "0x00000000",
                    "RAM31 Memory Rule Register 1 (ram31_mem_rule1)": "0x00000000",
                    "RAM31 Memory Rule Register 2 (ram31_mem_rule2)": "0x00000000",
                    "RAM31 Memory Rule Register 3 (ram31_mem_rule3)": "0x00000000",
                    "RAM32 Memory Rule Register 0 (ram32_mem_rule0)": "0x00000000",
                    "RAM32 Memory Rule Register 1 (ram32_mem_rule1)": "0x00000000",
                    "RAM32 Memory Rule Register 2 (ram32_mem_rule2)": "0x00000000",
                    "RAM32 Memory Rule Register 3 (ram32_mem_rule3)": "0x00000000",
                    "RAM33 Memory Rule Register 0 (ram33_mem_rule0)": "0x00000000",
                    "RAM33 Memory Rule Register 1 (ram33_mem_rule1)": "0x00000000",
                    "RAM33 Memory Rule Register 2 (ram33_mem_rule2)": "0x00000000",
                    "RAM33 Memory Rule Register 3 (ram33_mem_rule3)": "0x00000000",
                    "RAM4 Slave Rule Register (ram4_slave_rule)": "0x00000000",
                    "RAM40 Memory Rule Register 0 (ram40_mem_rule0)": "0x00000000",
                    "RAM40 Memory Rule Register 1 (ram40_mem_rule1)": "0x00000000",
                    "RAM40 Memory Rule Register 2 (ram40_mem_rule2)": "0x00000000",
                    "RAM40 Memory Rule Register 3 (ram40_mem_rule3)": "0x00000000",
                    "RAM41 Memory Rule Register 0 (ram41_mem_rule0)": "0x00000000",
                    "RAM41 Memory Rule Register 1 (ram41_mem_rule1)": "0x00000000",
                    "RAM41 Memory Rule Register 2 (ram41_mem_rule2)": "0x00000000",
                    "RAM41 Memory Rule Register 3 (ram41_mem_rule3)": "0x00000000",
                    "RAM42 Memory Rule Register 0 (ram42_mem_rule0)": "0x00000000",
```

```
                "RAM42 Memory Rule Register 1 (ram42_mem_rule1)": "0x00000000",
                "RAM42 Memory Rule Register 2 (ram42_mem_rule2)": "0x00000000",
                "RAM42 Memory Rule Register 3 (ram42_mem_rule3)": "0x00000000",
                "RAM43 Memory Rule Register 0 (ram43_mem_rule0)": "0x00000000",
                "RAM43 Memory Rule Register 1 (ram43_mem_rule1)": "0x00000000",
                "RAM43 Memory Rule Register 2 (ram43_mem_rule2)": "0x00000000",
                "RAM43 Memory Rule Register 3 (ram43_mem_rule3)": "0x00000000",
                "RAM5 Slave Rule Register (ram5_slave_rule)": "0x00000000",
                "RAM50 Memory Rule Register 0 (ram50_mem_rule0)": "0x00000000",
                "RAM50 Memory Rule Register 1 (ram50_mem_rule1)": "0x00000000",
                "RAM50 Memory Rule Register 2 (ram50_mem_rule2)": "0x00000000",
                "RAM50 Memory Rule Register 3 (ram50_mem_rule3)": "0x00000000",
                "RAM51 Memory Rule Register 0 (ram51_mem_rule0)": "0x00000000",
                "RAM51 Memory Rule Register 1 (ram51_mem_rule1)": "0x00000000",
                "RAM51 Memory Rule Register 2 (ram51_mem_rule2)": "0x00000000",
                "RAM51 Memory Rule Register 3 (ram51_mem_rule3)": "0x00000000",
                "RAM52 Memory Rule Register 0 (ram52_mem_rule0)": "0x00000000",
                "RAM52 Memory Rule Register 1 (ram52_mem_rule1)": "0x00000000",
                "RAM52 Memory Rule Register 2 (ram52_mem_rule2)": "0x00000000",
                "RAM52 Memory Rule Register 3 (ram52_mem_rule3)": "0x00000000",
                "RAM53 Memory Rule Register 0 (ram53_mem_rule0)": "0x00000000",
                "RAM53 Memory Rule Register 1 (ram53_mem_rule1)": "0x00000000",
                "RAM53 Memory Rule Register 2 (ram53_mem_rule2)": "0x00000000",
                "RAM53 Memory Rule Register 3 (ram53_mem_rule3)": "0x00000000",
                "RAM6 Slave Rule Register (ram6_slave_rule)": "0x00000000",
                "RAM60 Memory Rule Register 0 (ram60_mem_rule0)": "0x00000000",
                "RAM60 Memory Rule Register 1 (ram60_mem_rule1)": "0x00000000",
                "RAM60 Memory Rule Register 2 (ram60_mem_rule2)": "0x00000000",
                "RAM60 Memory Rule Register 3 (ram60_mem_rule3)": "0x00000000",
                "RAM61 Memory Rule Register 0 (ram61_mem_rule0)": "0x00000000",
                "RAM61 Memory Rule Register 1 (ram61_mem_rule1)": "0x00000000",
                "RAM61 Memory Rule Register 2 (ram61_mem_rule2)": "0x00000000",
                "RAM61 Memory Rule Register 3 (ram61_mem_rule3)": "0x00000000",
                "RAM62 Memory Rule Register 0 (ram62_mem_rule0)": "0x00000000",
                "RAM62 Memory Rule Register 1 (ram62_mem_rule1)": "0x00000000",
                "RAM62 Memory Rule Register 2 (ram62_mem_rule2)": "0x00000000",
                "RAM62 Memory Rule Register 3 (ram62_mem_rule3)": "0x00000000",
                "RAM63 Memory Rule Register 0 (ram63_mem_rule0)": "0x00000000",
                "RAM63 Memory Rule Register 1 (ram63_mem_rule1)": "0x00000000",
                "RAM63 Memory Rule Register 2 (ram63_mem_rule2)": "0x00000000",
                "RAM63 Memory Rule Register 3 (ram63_mem_rule3)": "0x00000000",
                "RAM7 Slave Rule Register (ram7_slave_rule)": "0x00000000",
                "RAM70 Memory Rule Register 0 (ram70_mem_rule0)": "0x00000000",
                "RAM70 Memory Rule Register 1 (ram70_mem_rule1)": "0x00000000",
                "RAM70 Memory Rule Register 2 (ram70_mem_rule2)": "0x00000000",
                "RAM70 Memory Rule Register 3 (ram70_mem_rule3)": "0x00000000",
                "RAM71 Memory Rule Register 0 (ram71_mem_rule0)": "0x00000000",
                "RAM71 Memory Rule Register 1 (ram71_mem_rule1)": "0x00000000",
                "RAM71 Memory Rule Register 2 (ram71_mem_rule2)": "0x00000000",
                "RAM71 Memory Rule Register 3 (ram71_mem_rule3)": "0x00000000",
                "RAM72 Memory Rule Register 0 (ram72_mem_rule0)": "0x00000000",
                "RAM72 Memory Rule Register 1 (ram72_mem_rule1)": "0x00000000",
                "RAM72 Memory Rule Register 2 (ram72_mem_rule2)": "0x00000000",
                "RAM72 Memory Rule Register 3 (ram72_mem_rule3)": "0x00000000",
                "RAM73 Memory Rule Register 0 (ram73_mem_rule0)": "0x00000000",
                "RAM73 Memory Rule Register 1 (ram73_mem_rule1)": "0x00000000",
                "RAM73 Memory Rule Register 2 (ram73_mem_rule2)": "0x00000000",
                "RAM73 Memory Rule Register 3 (ram73_mem_rule3)": "0x00000000",
                "RAM8 Slave Rule Register (ram8_slave_rule)": "0x00000000",
                "RAM80 Memory Rule Register 0 (ram80_mem_rule0)": "0x00000000",
```

```
    "RAM80 Memory Rule Register 1 (ram80_mem_rule1)": "0x00000000",
    "RAM80 Memory Rule Register 2 (ram80_mem_rule2)": "0x00000000",
    "RAM80 Memory Rule Register 3 (ram80_mem_rule3)": "0x00000000",
    "RAM81 Memory Rule Register 0 (ram81_mem_rule0)": "0x00000000",
    "RAM81 Memory Rule Register 1 (ram81_mem_rule1)": "0x00000000",
    "RAM81 Memory Rule Register 2 (ram81_mem_rule2)": "0x00000000",
    "RAM81 Memory Rule Register 3 (ram81_mem_rule3)": "0x00000000",
    "RAM81 Memory Rule Register 0 (ram82_mem_rule0)": "0x00000000",
    "RAM81 Memory Rule Register 1 (ram82_mem_rule1)": "0x00000000",
    "RAM81 Memory Rule Register 2 (ram82_mem_rule2)": "0x00000000",
    "RAM81 Memory Rule Register 3 (ram82_mem_rule3)": "0x00000000",
    "RAM81 Memory Rule Register 0 (ram83_mem_rule0)": "0x00000000",
    "RAM81 Memory Rule Register 1 (ram83_mem_rule1)": "0x00000000",
    "RAM81 Memory Rule Register 2 (ram83_mem_rule2)": "0x00000000",
    "RAM81 Memory Rule Register 3 (ram83_mem_rule3)": "0x00000000",
    "(ezh_ram_slave_rule)": "0x00000000",
    "(ezh_ram0_slave_rule0)": "0x00000000",
    "(ezh_ram0_slave_rule1)": "0x00000000",
    "(ezh_ram0_slave_rule2)": "0x00000000",
    "(ezh_ram0_slave_rule3)": "0x00000000",
    "Quad/Octal SPI Slave Rule Register 0 (qospi1_slave_rule0)": "0x00000000",
    "Quad/Octal SPI 0 Memory Rule Register 0 (qospi10_mem_rule0)": "0x00000000",
    "Quad/Octal SPI 0 Memory Rule Register 1 (qospi10_mem_rule1)": "0x00000000",
    "Quad/Octal SPI 0 Memory Rule Register 2 (qospi10_mem_rule2)": "0x00000000",
    "Quad/Octal SPI 0 Memory Rule Register 3 (qospi10_mem_rule3)": "0x00000000",
    "Quad/Octal SPI 1 Memory Rule Register 0 (qospi11_mem_rule0)": "0x00000000",
    "Quad/Octal SPI 1 Memory Rule Register 1 (qospi11_mem_rule1)": "0x00000000",
    "Quad/Octal SPI 1 Memory Rule Register 2 (qospi11_mem_rule2)": "0x00000000",
    "Quad/Octal SPI 1 Memory Rule Register 3 (qospi11_mem_rule3)": "0x00000000",
    "Quad/Octal SPI 2 Memory Rule Register 0 (qospi12_mem_rule0)": "0x00000000",
    "Quad/Octal SPI 2 Memory Rule Register 1 (qospi12_mem_rule1)": "0x00000000",
    "Quad/Octal SPI 2 Memory Rule Register 2 (qospi12_mem_rule2)": "0x00000000",
    "Quad/Octal SPI 2 Memory Rule Register 3 (qospi12_mem_rule3)": "0x00000000",
    "Quad/Octal SPI 3 Memory Rule Register 0 (qospi13_mem_rule0)": "0x00000000",
    "Quad/Octal SPI 3 Memory Rule Register 1 (qospi13_mem_rule1)": "0x00000000",
    "Quad/Octal SPI 3 Memory Rule Register 2 (qospi13_mem_rule2)": "0x00000000",
    "Quad/Octal SPI 3 Memory Rule Register 3 (qospi13_mem_rule3)": "0x00000000",
    "Quad/Octal SPI 4 Memory Rule Register 0 (qospi14_mem_rule0)": "0x00000000",
    "Quad/Octal SPI 4 Memory Rule Register 1 (qospi14_mem_rule1)": "0x00000000",
    "Quad/Octal SPI 4 Memory Rule Register 2 (qospi14_mem_rule2)": "0x00000000",
    "Quad/Octal SPI 4 Memory Rule Register 3 (qospi14_mem_rule3)": "0x00000000",
    "APB Bridge Slave Rule Register (apb_bridge_slave_rule0)": "0x00000000",
    "APB Bridge Group 0 Memory Rule Register 0 (apb_grp0_mem_rule0)": "0x00000000",
    "APB Bridge Group 0 Memory Rule Register 1 (apb_grp0_mem_rule1)": "0x00000000",
    "APB Bridge Group 0 Memory Rule Register 2 (apb_grp0_mem_rule2)": "0x00000000",
    "APB Bridge Group 0 Memory Rule Register 3 (apb_grp0_mem_rule3)": "0x00000000",
    "APB Bridge Group 1 Memory Rule Register 0 (apb_grp1_mem_rule0)": "0x00000000",
    "APB Bridge Group 1 Memory Rule Register 1 (apb_grp1_mem_rule1)": "0x00000000",
    "APB Bridge Group 1 Memory Rule Register 2 (apb_grp1_mem_rule2)": "0x00000000",
    "APB Bridge Group 1 Memory Rule Register 3 (apb_grp1_mem_rule3)": "0x00000000",
  "AHB Peripherals 0 Slave Rule Register 0 (ahb_periph0_slave_rule0)": "0x00000000",
    "AIPS bridge 0 Memory Rule Register 0 (aips_bridge0_mem_rule0)": "0x00000000",
    "AIPS bridge 0 Memory Rule Register 1 (aips_bridge0_mem_rule1)": "0x00000000",
   "AHB Peripherals 1 Slave Rule Register (ahb_periph1_slave_rule0)": "0x00000000",
   "AHB Peripherals 1 Slave Rule Register (ahb_periph1_slave_rule1)": "0x00000000",
    "AIPS Bridge Slave Rule Register (aips_bridge_slave_rule0)": "0x00000000",
    "AIPS bridge 1 Memory Rule Register 0 (aips_bridge1_mem_rule0)": "0x00000000",
    "AIPS bridge 1 Memory Rule Register 1 (aips_bridge1_mem_rule1)": "0x00000000",
  "AHB Peripherals 2 Slave Rule Register 0 (ahb_periph2_slave_rule0)": "0x00000000",
  "AHB Peripherals 2 Slave Rule Register 0 (security_ctrl_mem_rule0)": "0x00000000",
```

```
                    "AHB Peripherals 3 Slave Rule Register 0 (ahb_periph3_slave_rule0)": "0x00000000",
                    "AHB Peripherals 3 Slave Rule Register 0 (ahb_periph3_slave_rule1)": "0x00000000",
                      "Secure GPIO Register 0 (sec_gp_reg0)": "0xffffffff",
                      "Secure GPIO Register 1 (sec_gp_reg1)": "0xffffffff",
                      "Secure GPIO Register 2 (sec_gp_reg2)": "0xffffffff",
                      "Secure GPIO Register 3 (sec_gp_reg3)": "0xffffffff",
                      "Secure GPIO Register 4 (sec_gp_reg4)": "0xffffffff",
                      "Secure GPIO Register 5 (sec_gp_reg5)": "0xffffffff",
                      "Secure GPIO Register 6 (sec_gp_reg6)": "0xffffffff",
                      "Secure GPIO Register 7 (sec_gp_reg7)": "0xffffffff",
                      "Secure GPIO Register 8 for DSP (sec_gp_reg8)": "0xffffffff",
                      "Secure GPIO Lock Register (sec_gp_reg_lock)": "0x0000aaaa",
                      "Master Secure Level Register (master_sec_reg)": "0x80000000",
                     "Master Secure Level Anti-pole Register (master_sec_anti_pol_reg)": "0xbfffffff",
                      "M33 Lock Control Register (m33_lock_reg)": "0x800002aa",
                      "Secure Control Duplicate Register (misc_ctrl_dp_reg)": "0x0000aaaa",
                      "Secure Control Register (misc_ctrl_reg)": "0x0000aaaa",
                      "Miscellaneous TZM settings (misc_tzm_settings)": "0x00000000"
                    }
}
```

9.3.5 rt6xx A0 的 TrustZone-M 预设寄存器

rt6xx A0 的 Json TrustZone-M 配置文件，包含所有 TrustZone-M 预设寄存器及默认（复位）值：

```
{
    "family": "rt6xx",
    "revision": "a0",
    "tzpOutputFile": "C:/Work/TrustZone/tzFile.bin",
    "trustZonePreset": {
                    "Secure vector table address (vtor_addr)": "0x00000000",
        "Non-secure vector table address (vtor_ns_addr)": "0x00000000",
        "Interrupt target non-secure register 0 (nvic_itns0)": "0x00000000",
        "Interrupt target non-secure register 1 (nvic_itns1)": "0x00000000",
        "MPU Control Register (mpu_ctrl)": "0x00000000",
        "MPU Memory Attribute Indirection Register 0 (mpu_mair0)": "0x00000000",
        "MPU Memory Attribute Indirection Register 1 (mpu_mair1)": "0x00000000",
        "MPU Region 0 Base Address Register (mpu_rbar0)": "0x00000000",
        "MPU Region 0 Limit Address Register (mpu_rlar0)": "0x00000000",
        "MPU Region 1 Base Address Register (mpu_rbar1)": "0x00000000",
        "MPU Region 1 Limit Address Register (mpu_rlar1)": "0x00000000",
        "MPU Region 2 Base Address Register (mpu_rbar2)": "0x00000000",
        "MPU Region 2 Limit Address Register (mpu_rlar2)": "0x00000000",
        "MPU Region 3 Base Address Register (mpu_rbar3)": "0x00000000",
        "MPU Region 3 Limit Address Register (mpu_rlar3)": "0x00000000",
        "MPU Region 4 Base Address Register (mpu_rbar4)": "0x00000000",
        "MPU Region 4 Limit Address Register (mpu_rlar4)": "0x00000000",
        "MPU Region 5 Base Address Register (mpu_rbar5)": "0x00000000",
        "MPU Region 5 Limit Address Register (mpu_rlar5)": "0x00000000",
        "MPU Region 6 Base Address Register (mpu_rbar6)": "0x00000000",
        "MPU Region 6 Limit Address Register (mpu_rlar6)": "0x00000000",
        "MPU Region 7 Base Address Register (mpu_rbar7)": "0x00000000",
        "MPU Region 7 Limit Address Register (mpu_rlar7)": "0x00000000",
        "Non-secure MPU Control Register(mpu_ctrl_ns)": "0x00000000",
        "Non-secure MPU Memory Attribute Indirection Register 0 (mpu_mair0_ns)": "0x00000000",
        "Non-secure MPU Memory Attribute Indirection Register 1 (mpu_mair1_ns)": "0x00000000",
        "Non-secure MPU Region 0 Base Address Register (mpu_rbar0_ns)": "0x00000000",
        "Non-secure MPU Region 0 Limit Address Register (mpu_rlar0_ns)": "0x00000000",
        "Non-secure MPU Region 1 Base Address Register (mpu_rbar1_ns)": "0x00000000",
```

```
        "Non-secure MPU Region 1 Limit Address Register (mpu_rlar1_ns)": "0x00000000",
        "Non-secure MPU Region 2 Base Address Register (mpu_rbar2_ns)": "0x00000000",
        "Non-secure MPU Region 2 Limit Address Register (mpu_rlar2_ns)": "0x00000000",
        "Non-secure MPU Region 3 Base Address Register (mpu_rbar3_ns)": "0x00000000",
        "Non-secure MPU Region 3 Limit Address Register (mpu_rlar3_ns)": "0x00000000",
        "Non-secure MPU Region 4 Base Address Register (mpu_rbar4_ns)": "0x00000000",
        "Non-secure MPU Region 4 Limit Address Register (mpu_rlar4_ns)": "0x00000000",
        "Non-secure MPU Region 5 Base Address Register (mpu_rbar5_ns)": "0x00000000",
        "Non-secure MPU Region 5 Limit Address Register (mpu_rlar5_ns)": "0x00000000",
        "Non-secure MPU Region 6 Base Address Register (mpu_rbar6_ns)": "0x00000000",
        "Non-secure MPU Region 6 Limit Address Register (mpu_rlar6_ns)": "0x00000000",
        "Non-secure MPU Region 7 Base Address Register (mpu_rbar7_ns)": "0x00000000",
        "Non-secure MPU Region 7 Limit Address Register (mpu_rlar7_ns)": "0x00000000",
        "SAU Control Register.(sau_ctrl)": "0x00000000",
        "SAU Region 0 Base Address Register (sau_rbar0)": "0x00000000",
        "SAU Region 0 Limit Address Register (sau_rlar0)": "0x00000000",
        "SAU Region 1 Base Address Register (sau_rbar1)": "0x00000000",
        "SAU Region 1 Limit Address Register (sau_rlar1)": "0x00000000",
        "SAU Region 2 Base Address Register (sau_rbar2)": "0x00000000",
        "SAU Region 2 Limit Address Register (sau_rlar2)": "0x00000000",
        "SAU Region 3 Base Address Register (sau_rbar3)": "0x00000000",
        "SAU Region 3 Limit Address Register (sau_rlar3)": "0x00000000",
        "SAU Region 4 Base Address Register (sau_rbar4)": "0x00000000",
        "SAU Region 4 Limit Address Register (sau_rlar4)": "0x00000000",
        "SAU Region 5 Base Address Register (sau_rbar5)": "0x00000000",
        "SAU Region 5 Limit Address Register (sau_rlar5)": "0x00000000",
        "SAU Region 6 Base Address Register (sau_rbar6)": "0x00000000",
        "SAU Region 6 Limit Address Register (sau_rlar6)": "0x00000000",
        "SAU Region 7 Base Address Register (sau_rbar7)": "0x00000000",
        "SAU Region 7 Limit Address Register (sau_rlar7)": "0x00000000",
        "ROM Slave Rule Register 0 (bootrom0_slave_rule0)": "0x00000000",
        "ROM Memory Rule Register 0 (bootrom0_mem_rule0)": "0x00000000",
        "ROM Memory Rule Register 1 (bootrom0_mem_rule1)": "0x00000000",
        "ROM Memory Rule Register 2 (bootrom0_mem_rule2)": "0x00000000",
        "ROM Memory Rule Register 3 (bootrom0_mem_rule3)": "0x00000000",
        "Quad/Octal SPI Slave Rule Register 0 (qospi_slave_rule0)": "0x00000000",
        "Quad/Octal SPI 0 Memory Rule Register 0 (qospi0_mem_rule0)": "0x00000000",
        "Quad/Octal SPI 0 Memory Rule Register 1 (qospi0_mem_rule1)": "0x00000000",
        "Quad/Octal SPI 0 Memory Rule Register 2 (qospi0_mem_rule2)": "0x00000000",
        "Quad/Octal SPI 0 Memory Rule Register 3 (qospi0_mem_rule3)": "0x00000000",
        "Quad/Octal SPI 1 Memory Rule Register 0 (qospi1_mem_rule0)": "0x00000000",
        "Quad/Octal SPI 1 Memory Rule Register 1 (qospi1_mem_rule1)": "0x00000000",
        "Quad/Octal SPI 1 Memory Rule Register 2 (qospi1_mem_rule2)": "0x00000000",
        "Quad/Octal SPI 1 Memory Rule Register 3 (qospi1_mem_rule3)": "0x00000000",
        "Quad/Octal SPI 2 Memory Rule Register 0 (qospi2_mem_rule0)": "0x00000000",
        "Quad/Octal SPI 2 Memory Rule Register 1 (qospi2_mem_rule1)": "0x00000000",
        "Quad/Octal SPI 2 Memory Rule Register 2 (qospi2_mem_rule2)": "0x00000000",
        "Quad/Octal SPI 2 Memory Rule Register 3 (qospi2_mem_rule3)": "0x00000000",
        "Quad/Octal SPI 3 Memory Rule Register 0 (qospi3_mem_rule0)": "0x00000000",
        "Quad/Octal SPI 3 Memory Rule Register 1 (qospi3_mem_rule1)": "0x00000000",
        "Quad/Octal SPI 3 Memory Rule Register 2 (qospi3_mem_rule2)": "0x00000000",
        "Quad/Octal SPI 3 Memory Rule Register 3 (qospi3_mem_rule3)": "0x00000000",
        "RAM0 Slave Rule Register (ram0_slave_rule)": "0x00000000",
        "RAM00 Memory Rule Register 0 (ram00_mem_rule0)": "0x00000000",
        "RAM00 Memory Rule Register 1 (ram00_mem_rule1)": "0x00000000",
        "RAM00 Memory Rule Register 2 (ram00_mem_rule2)": "0x00000000",
        "RAM00 Memory Rule Register 3 (ram00_mem_rule3)": "0x00000000",
        "RAM01 Memory Rule Register 0 (ram01_mem_rule0)": "0x00000000",
        "RAM01 Memory Rule Register 1 (ram01_mem_rule1)": "0x00000000",
        "RAM01 Memory Rule Register 2 (ram01_mem_rule2)": "0x00000000",
```

```
        "RAM01 Memory Rule Register 3 (ram01_mem_rule3)": "0x00000000",
        "RAM1 Slave Rule Register (ram1_slave_rule)": "0x00000000",
        "RAM10 Memory Rule Register 0 (ram10_mem_rule0)": "0x00000000",
        "RAM10 Memory Rule Register 1 (ram10_mem_rule1)": "0x00000000",
        "RAM10 Memory Rule Register 2 (ram10_mem_rule2)": "0x00000000",
        "RAM10 Memory Rule Register 3 (ram10_mem_rule3)": "0x00000000",
        "RAM11 Memory Rule Register 0 (ram11_mem_rule0)": "0x00000000",
        "RAM11 Memory Rule Register 1 (ram11_mem_rule1)": "0x00000000",
        "RAM11 Memory Rule Register 2 (ram11_mem_rule2)": "0x00000000",
        "RAM11 Memory Rule Register 3 (ram11_mem_rule3)": "0x00000000",
        "RAM2 Slave Rule Register (ram2_slave_rule)": "0x00000000",
        "RAM20 Memory Rule Register 0 (ram20_mem_rule0)": "0x00000000",
        "RAM20 Memory Rule Register 1 (ram20_mem_rule1)": "0x00000000",
        "RAM20 Memory Rule Register 2 (ram20_mem_rule2)": "0x00000000",
        "RAM20 Memory Rule Register 3 (ram20_mem_rule3)": "0x00000000",
        "RAM21 Memory Rule Register 0 (ram21_mem_rule0)": "0x00000000",
        "RAM21 Memory Rule Register 1 (ram21_mem_rule1)": "0x00000000",
        "RAM21 Memory Rule Register 2 (ram21_mem_rule2)": "0x00000000",
        "RAM21 Memory Rule Register 3 (ram21_mem_rule3)": "0x00000000",
        "RAM22 Memory Rule Register 0 (ram22_mem_rule0)": "0x00000000",
        "RAM22 Memory Rule Register 1 (ram22_mem_rule1)": "0x00000000",
        "RAM22 Memory Rule Register 2 (ram22_mem_rule2)": "0x00000000",
        "RAM22 Memory Rule Register 3 (ram22_mem_rule3)": "0x00000000",
        "RAM23 Memory Rule Register 0 (ram23_mem_rule0)": "0x00000000",
        "RAM23 Memory Rule Register 1 (ram23_mem_rule1)": "0x00000000",
        "RAM23 Memory Rule Register 2 (ram23_mem_rule2)": "0x00000000",
        "RAM23 Memory Rule Register 3 (ram23_mem_rule3)": "0x00000000",
        "RAM3 Slave Rule Register (ram3_slave_rule)": "0x00000000",
        "RAM30 Memory Rule Register 0 (ram30_mem_rule0)": "0x00000000",
        "RAM30 Memory Rule Register 1 (ram30_mem_rule1)": "0x00000000",
        "RAM30 Memory Rule Register 2 (ram30_mem_rule2)": "0x00000000",
        "RAM30 Memory Rule Register 3 (ram30_mem_rule3)": "0x00000000",
        "RAM31 Memory Rule Register 0 (ram31_mem_rule0)": "0x00000000",
        "RAM31 Memory Rule Register 1 (ram31_mem_rule1)": "0x00000000",
        "RAM31 Memory Rule Register 2 (ram31_mem_rule2)": "0x00000000",
        "RAM31 Memory Rule Register 3 (ram31_mem_rule3)": "0x00000000",
        "RAM32 Memory Rule Register 0 (ram32_mem_rule0)": "0x00000000",
        "RAM32 Memory Rule Register 1 (ram32_mem_rule1)": "0x00000000",
        "RAM32 Memory Rule Register 2 (ram32_mem_rule2)": "0x00000000",
        "RAM32 Memory Rule Register 3 (ram32_mem_rule3)": "0x00000000",
        "RAM33 Memory Rule Register 0 (ram33_mem_rule0)": "0x00000000",
        "RAM33 Memory Rule Register 1 (ram33_mem_rule1)": "0x00000000",
        "RAM33 Memory Rule Register 2 (ram33_mem_rule2)": "0x00000000",
        "RAM33 Memory Rule Register 3 (ram33_mem_rule3)": "0x00000000",
        "RAM4 Slave Rule Register (ram4_slave_rule)": "0x00000000",
        "RAM40 Memory Rule Register 0 (ram40_mem_rule0)": "0x00000000",
        "RAM40 Memory Rule Register 1 (ram40_mem_rule1)": "0x00000000",
        "RAM40 Memory Rule Register 2 (ram40_mem_rule2)": "0x00000000",
        "RAM40 Memory Rule Register 3 (ram40_mem_rule3)": "0x00000000",
        "RAM41 Memory Rule Register 0 (ram41_mem_rule0)": "0x00000000",
        "RAM41 Memory Rule Register 1 (ram41_mem_rule1)": "0x00000000",
        "RAM41 Memory Rule Register 2 (ram41_mem_rule2)": "0x00000000",
        "RAM41 Memory Rule Register 3 (ram41_mem_rule3)": "0x00000000",
        "RAM42 Memory Rule Register 0 (ram42_mem_rule0)": "0x00000000",
        "RAM42 Memory Rule Register 1 (ram42_mem_rule1)": "0x00000000",
        "RAM42 Memory Rule Register 2 (ram42_mem_rule2)": "0x00000000",
        "RAM42 Memory Rule Register 3 (ram42_mem_rule3)": "0x00000000",
        "RAM43 Memory Rule Register 0 (ram43_mem_rule0)": "0x00000000",
        "RAM43 Memory Rule Register 1 (ram43_mem_rule1)": "0x00000000",
        "RAM43 Memory Rule Register 2 (ram43_mem_rule2)": "0x00000000",
```

```
        "RAM43 Memory Rule Register 3 (ram43_mem_rule3)": "0x00000000",
        "RAM5 Slave Rule Register (ram5_slave_rule)": "0x00000000",
        "RAM50 Memory Rule Register 0 (ram50_mem_rule0)": "0x00000000",
        "RAM50 Memory Rule Register 1 (ram50_mem_rule1)": "0x00000000",
        "RAM50 Memory Rule Register 2 (ram50_mem_rule2)": "0x00000000",
        "RAM50 Memory Rule Register 3 (ram50_mem_rule3)": "0x00000000",
        "RAM51 Memory Rule Register 0 (ram51_mem_rule0)": "0x00000000",
        "RAM51 Memory Rule Register 1 (ram51_mem_rule1)": "0x00000000",
        "RAM51 Memory Rule Register 2 (ram51_mem_rule2)": "0x00000000",
        "RAM51 Memory Rule Register 3 (ram51_mem_rule3)": "0x00000000",
        "RAM52 Memory Rule Register 0 (ram52_mem_rule0)": "0x00000000",
        "RAM52 Memory Rule Register 1 (ram52_mem_rule1)": "0x00000000",
        "RAM52 Memory Rule Register 2 (ram52_mem_rule2)": "0x00000000",
        "RAM52 Memory Rule Register 3 (ram52_mem_rule3)": "0x00000000",
        "RAM53 Memory Rule Register 0 (ram53_mem_rule0)": "0x00000000",
        "RAM53 Memory Rule Register 1 (ram53_mem_rule1)": "0x00000000",
        "RAM53 Memory Rule Register 2 (ram53_mem_rule2)": "0x00000000",
        "RAM53 Memory Rule Register 3 (ram53_mem_rule3)": "0x00000000",
        "RAM6 Slave Rule Register (ram6_slave_rule)": "0x00000000",
        "RAM60 Memory Rule Register 0 (ram60_mem_rule0)": "0x00000000",
        "RAM60 Memory Rule Register 1 (ram60_mem_rule1)": "0x00000000",
        "RAM60 Memory Rule Register 2 (ram60_mem_rule2)": "0x00000000",
        "RAM60 Memory Rule Register 3 (ram60_mem_rule3)": "0x00000000",
        "RAM61 Memory Rule Register 0 (ram61_mem_rule0)": "0x00000000",
        "RAM61 Memory Rule Register 1 (ram61_mem_rule1)": "0x00000000",
        "RAM61 Memory Rule Register 2 (ram61_mem_rule2)": "0x00000000",
        "RAM61 Memory Rule Register 3 (ram61_mem_rule3)": "0x00000000",
        "RAM62 Memory Rule Register 0 (ram62_mem_rule0)": "0x00000000",
        "RAM62 Memory Rule Register 1 (ram62_mem_rule1)": "0x00000000",
        "RAM62 Memory Rule Register 2 (ram62_mem_rule2)": "0x00000000",
        "RAM62 Memory Rule Register 3 (ram62_mem_rule3)": "0x00000000",
        "RAM63 Memory Rule Register 0 (ram63_mem_rule0)": "0x00000000",
        "RAM63 Memory Rule Register 1 (ram63_mem_rule1)": "0x00000000",
        "RAM63 Memory Rule Register 2 (ram63_mem_rule2)": "0x00000000",
        "RAM63 Memory Rule Register 3 (ram63_mem_rule3)": "0x00000000",
        "RAM7 Slave Rule Register (ram7_slave_rule)": "0x00000000",
        "RAM70 Memory Rule Register 0 (ram70_mem_rule0)": "0x00000000",
        "RAM70 Memory Rule Register 1 (ram70_mem_rule1)": "0x00000000",
        "RAM70 Memory Rule Register 2 (ram70_mem_rule2)": "0x00000000",
        "RAM70 Memory Rule Register 3 (ram70_mem_rule3)": "0x00000000",
        "RAM71 Memory Rule Register 0 (ram71_mem_rule0)": "0x00000000",
        "RAM71 Memory Rule Register 1 (ram71_mem_rule1)": "0x00000000",
        "RAM71 Memory Rule Register 2 (ram71_mem_rule2)": "0x00000000",
        "RAM71 Memory Rule Register 3 (ram71_mem_rule3)": "0x00000000",
        "RAM72 Memory Rule Register 0 (ram72_mem_rule0)": "0x00000000",
        "RAM72 Memory Rule Register 1 (ram72_mem_rule1)": "0x00000000",
        "RAM72 Memory Rule Register 2 (ram72_mem_rule2)": "0x00000000",
        "RAM72 Memory Rule Register 3 (ram72_mem_rule3)": "0x00000000",
        "RAM73 Memory Rule Register 0 (ram73_mem_rule0)": "0x00000000",
        "RAM73 Memory Rule Register 1 (ram73_mem_rule1)": "0x00000000",
        "RAM73 Memory Rule Register 2 (ram73_mem_rule2)": "0x00000000",
        "RAM73 Memory Rule Register 3 (ram73_mem_rule3)": "0x00000000",
        "RAM8 Slave Rule Register (ram8_slave_rule)": "0x00000000",
        "RAM80 Memory Rule Register 0 (ram80_mem_rule0)": "0x00000000",
        "RAM80 Memory Rule Register 1 (ram80_mem_rule1)": "0x00000000",
        "RAM80 Memory Rule Register 2 (ram80_mem_rule2)": "0x00000000",
        "RAM80 Memory Rule Register 3 (ram80_mem_rule3)": "0x00000000",
        "RAM81 Memory Rule Register 0 (ram81_mem_rule0)": "0x00000000",
        "RAM81 Memory Rule Register 1 (ram81_mem_rule1)": "0x00000000",
        "RAM81 Memory Rule Register 2 (ram81_mem_rule2)": "0x00000000",
```

```
        "RAM81 Memory Rule Register 3 (ram81_mem_rule3)": "0x00000000",
        "HiFi4 DSP Slave Rule Register (pif_hifi4_x_slave_rule0)": "0x00000000",
        "HiFi4 DSP Memory Rule Register 0 (pif_hifi4_x_mem_rule0)": "0x00000000",
        "HiFi4 DSP Memory Rule Register 1 (pif_hifi4_x_mem_rule1)": "0x00000000",
        "HiFi4 DSP Memory Rule Register 2 (pif_hifi4_x_mem_rule2)": "0x00000000",
        "HiFi4 DSP Memory Rule Register 3 (pif_hifi4_x_mem_rule3)": "0x00000000",
        "APB Bridge Slave Rule Register (apb_bridge_slave_rule0)": "0x00000000",
        "APB Bridge Group 0 Memory Rule Register 0 (apb_grp0_mem_rule0)": "0x00000000",
        "APB Bridge Group 0 Memory Rule Register 1 (apb_grp0_mem_rule1)": "0x00000000",
        "APB Bridge Group 0 Memory Rule Register 2 (apb_grp0_mem_rule2)": "0x00000000",
        "APB Bridge Group 0 Memory Rule Register 3 (apb_grp0_mem_rule3)": "0x00000000",
        "APB Bridge Group 1 Memory Rule Register 0 (apb_grp1_mem_rule0)": "0x00000000",
        "APB Bridge Group 1 Memory Rule Register 1 (apb_grp1_mem_rule1)": "0x00000000",
        "APB Bridge Group 1 Memory Rule Register 2 (apb_grp1_mem_rule2)": "0x00000000",
        "APB Bridge Group 1 Memory Rule Register 3 (apb_grp1_mem_rule3)": "0x00000000",
        "AHB Peripherals 0 Slave Rule Register 0 (ahb_periph0_slave_rule0)": "0x00000000",
        "AHB Peripherals 0 Slave Rule Register 1 (ahb_periph0_slave_rule1)": "0x00000000",
        "AIPS bridge 0 Memory Rule Register 0 (aips_bridge0_mem_rule0)": "0x00000000",
        "AIPS bridge 0 Memory Rule Register 1 (aips_bridge0_mem_rule1)": "0x00000000",
        "AHB Peripherals 1 Slave Rule Register (ahb_periph1_slave_rule0)": "0x00000000",
        "AIPS bridge 1 Slave Rule Register (aips_bridge1_slave_rule0)": "0x00000000",
        "AIPS bridge 1 Memory Rule Register 0 (aips_bridge1_mem_rule0)": "0x00000000",
        "AIPS bridge 1 Memory Rule Register 1 (aips_bridge1_mem_rule1)": "0x00000000",
        "AHB Peripherals 2 Slave Rule Register 0 (ahb_periph2_slave_rule0)": "0x00000000",
        "AHB Peripherals 3 Slave Rule Register 0 (ahb_periph3_slave_rule0)": "0x00000000",
        "Secure GPIO Register 0 (sec_gp_reg0)": "0xffffffff",
        "Secure GPIO Register 1 (sec_gp_reg1)": "0xffffffff",
        "Secure GPIO Register 2 (sec_gp_reg2)": "0xffffffff",
        "Secure GPIO Register 3 (sec_gp_reg3)": "0xffffffff",
        "Secure GPIO Register 4 (sec_gp_reg4)": "0xffffffff",
        "Secure GPIO Register 5 (sec_gp_reg5)": "0xffffffff",
        "Secure GPIO Register 6 (sec_gp_reg6)": "0xffffffff",
        "Secure GPIO Register 7 (sec_gp_reg7)": "0xffffffff",
        "Secure GPIO Register 8 for DSP(sec_gp_reg8)": "0xffffffff",
        "Secure GPIO Lock Register (sec_gp_reg_lock)": "0x00000000000aaaa",
        "Master Secure Level Register (master_sec_reg)": "0x80000000",
        "Master Secure Level Anti-pole Register (master_sec_anti_pol_reg)": "0xbfffffff",
        "M33 Lock Control Register (m33_lock_reg)": "0x800002aa",
        "Secure Control Duplicate Register (misc_ctrl_dp_reg)": "0x00000000000aaaa",
        "Secure Control Register (misc_ctrl_reg)": "0x00000000000aaaa"
    }
}
```

9.3.6 rt6xx B0 的 TrustZone-M 预设寄存器

rt6xx B0 的 Json TrustZone-M 配置文件，包含所有 TrustZone-M 预设寄存器及默认（复位）值：

```
{
 "family": "rt6xx",
 "revision": "b0",
 "tzpOutputFile": "C:/Work/TrustZone/tzFile.bin",
 "trustZonePreset": {
                "Secure vector table address (vtor_addr)": "0x00000000",
                "Non-secure vector table address (vtor_ns_addr)": "0x00000000",
                "Interrupt target non-secure register 0 (nvic_itns0)": "0x00000000",
                "Interrupt target non-secure register 1 (nvic_itns1)": "0x00000000",
                "MPU Control Register (mpu_ctrl)": "0x00000000",
                "MPU Memory Attribute Indirection Register 0 (mpu_mair0)": "0x00000000",
                "MPU Memory Attribute Indirection Register 1 (mpu_mair1)": "0x00000000",
```

```
                "MPU Region 0 Base Address Register (mpu_rbar0)": "0x00000000",
                "MPU Region 0 Limit Address Register (mpu_rlar0)": "0x00000000",
                "MPU Region 1 Base Address Register (mpu_rbar1)": "0x00000000",
                "MPU Region 1 Limit Address Register (mpu_rlar1)": "0x00000000",
                "MPU Region 2 Base Address Register (mpu_rbar2)": "0x00000000",
                "MPU Region 2 Limit Address Register (mpu_rlar2)": "0x00000000",
                "MPU Region 3 Base Address Register (mpu_rbar3)": "0x00000000",
                "MPU Region 3 Limit Address Register (mpu_rlar3)": "0x00000000",
                "MPU Region 4 Base Address Register (mpu_rbar4)": "0x00000000",
                "MPU Region 4 Limit Address Register (mpu_rlar4)": "0x00000000",
                "MPU Region 5 Base Address Register (mpu_rbar5)": "0x00000000",
                "MPU Region 5 Limit Address Register (mpu_rlar5)": "0x00000000",
                "MPU Region 6 Base Address Register (mpu_rbar6)": "0x00000000",
                "MPU Region 6 Limit Address Register (mpu_rlar6)": "0x00000000",
                "MPU Region 7 Base Address Register (mpu_rbar7)": "0x00000000",
                "MPU Region 7 Limit Address Register (mpu_rlar7)": "0x00000000",
                "Non-secure MPU Control Register. (mpu_ctrl_ns)": "0x00000000",
               "Non-secure MPU Memory Attribute Indirection Register 0 (mpu_mair0_ns)": "0x00000000",
               "Non-secure MPU Memory Attribute Indirection Register 1 (mpu_mair1_ns)": "0x00000000",
                "Non-secure MPU Region 0 Base Address Register (mpu_rbar0_ns)": "0x00000000",
                "Non-secure MPU Region 0 Limit Address Register (mpu_rlar0_ns)": "0x00000000",
                "Non-secure MPU Region 1 Base Address Register (mpu_rbar1_ns)": "0x00000000",
                "Non-secure MPU Region 1 Limit Address Register (mpu_rlar1_ns)": "0x00000000",
                "Non-secure MPU Region 2 Base Address Register (mpu_rbar2_ns)": "0x00000000",
                "Non-secure MPU Region 2 Limit Address Register (mpu_rlar2_ns)": "0x00000000",
                "Non-secure MPU Region 3 Base Address Register (mpu_rbar3_ns)": "0x00000000",
                "Non-secure MPU Region 3 Limit Address Register (mpu_rlar3_ns)": "0x00000000",
                "Non-secure MPU Region 4 Base Address Register (mpu_rbar4_ns)": "0x00000000",
                "Non-secure MPU Region 4 Limit Address Register (mpu_rlar4_ns)": "0x00000000",
                "Non-secure MPU Region 5 Base Address Register (mpu_rbar5_ns)": "0x00000000",
                "Non-secure MPU Region 5 Limit Address Register (mpu_rlar5_ns)": "0x00000000",
                "Non-secure MPU Region 6 Base Address Register (mpu_rbar6_ns)": "0x00000000",
                "Non-secure MPU Region 6 Limit Address Register (mpu_rlar6_ns)": "0x00000000",
                "Non-secure MPU Region 7 Base Address Register (mpu_rbar7_ns)": "0x00000000",
                "Non-secure MPU Region 7 Limit Address Register (mpu_rlar7_ns)": "0x00000000",
                "SAU Control Register (sau_ctrl)": "0x00000000",
                "SAU Region 0 Base Address Register (sau_rbar0)": "0x00000000",
                "SAU Region 0 Limit Address Register (sau_rlar0)": "0x00000000",
                "SAU Region 1 Base Address Register (sau_rbar1)": "0x00000000",
                "SAU Region 1 Limit Address Register (sau_rlar1)": "0x00000000",
                "SAU Region 2 Base Address Register (sau_rbar2)": "0x00000000",
                "SAU Region 2 Limit Address Register (sau_rlar2)": "0x00000000",
                "SAU Region 3 Base Address Register (sau_rbar3)": "0x00000000",
                        "SAU Region 3 Limit Address Register (sau_rlar3)": "0x00000000",
                   "SAU Region 4 Base Address Register (sau_rbar4)": "0x00000000",
                   "SAU Region 4 Limit Address Register (sau_rlar4)": "0x00000000",
                   "SAU Region 5 Base Address Register (sau_rbar5)": "0x00000000",
                   "SAU Region 5 Limit Address Register (sau_rlar5)": "0x00000000",
                   "SAU Region 6 Base Address Register (sau_rbar6)": "0x00000000",
                   "SAU Region 6 Limit Address Register (sau_rlar6)": "0x00000000",
                   "SAU Region 7 Base Address Register (sau_rbar7)": "0x00000000",
                   "SAU Region 7 Limit Address Register (sau_rlar7)": "0x00000000",
                   "ROM Slave Rule Register 0 (bootrom0_slave_rule0)": "0x00000000",
                   "ROM Memory Rule Register 0 (bootrom0_mem_rule0)": "0x00000000",
                   "ROM Memory Rule Register 1 (bootrom0_mem_rule1)": "0x00000000",
                   "ROM Memory Rule Register 2 (bootrom0_mem_rule2)": "0x00000000",
                   "ROM Memory Rule Register 3 (bootrom0_mem_rule3)": "0x00000000",
                   "Quad/Octal SPI Slave Rule Register 0 (qspi_slave_rule0)": "0x00000000",
                   "Quad/Octal SPI 0 Memory Rule Register 0 (qspi0_mem_rule0)": "0x00000000",
                   "Quad/Octal SPI 0 Memory Rule Register 1 (qspi0_mem_rule1)": "0x00000000",
```

```
                    "Quad/Octal SPI 0 Memory Rule Register 2 (qspi0_mem_rule2)": "0x00000000",
                    "Quad/Octal SPI 0 Memory Rule Register 3 (qspi0_mem_rule3)": "0x00000000",
                    "Quad/Octal SPI 1 Memory Rule Register 0 (qspi1_mem_rule0)": "0x00000000",
                    "Quad/Octal SPI 1 Memory Rule Register 1 (qspi1_mem_rule1)": "0x00000000",
                    "Quad/Octal SPI 1 Memory Rule Register 2 (qspi1_mem_rule2)": "0x00000000",
                    "Quad/Octal SPI 1 Memory Rule Register 3 (qspi1_mem_rule3)": "0x00000000",
                    "Quad/Octal SPI 2 Memory Rule Register 0 (qspi2_mem_rule0)": "0x00000000",
                    "Quad/Octal SPI 2 Memory Rule Register 1 (qspi2_mem_rule1)": "0x00000000",
                    "Quad/Octal SPI 2 Memory Rule Register 2 (qspi2_mem_rule2)": "0x00000000",
                    "Quad/Octal SPI 2 Memory Rule Register 3 (qspi2_mem_rule3)": "0x00000000",
                    "Quad/Octal SPI 3 Memory Rule Register 0 (qspi3_mem_rule0)": "0x00000000",
                    "Quad/Octal SPI 3 Memory Rule Register 1 (qspi3_mem_rule1)": "0x00000000",
                    "Quad/Octal SPI 3 Memory Rule Register 2 (qspi3_mem_rule2)": "0x00000000",
                    "Quad/Octal SPI 3 Memory Rule Register 3 (qspi3_mem_rule3)": "0x00000000",
                    "Quad/Octal SPI 4 Memory Rule Register 0 (qspi4_mem_rule0)": "0x00000000",
                    "Quad/Octal SPI 4 Memory Rule Register 1 (qspi4_mem_rule1)": "0x00000000",
                    "Quad/Octal SPI 4 Memory Rule Register 2 (qspi4_mem_rule2)": "0x00000000",
                    "Quad/Octal SPI 4 Memory Rule Register 3 (qspi4_mem_rule3)": "0x00000000",
                    "RAM0 Slave Rule Register (ram0_slave_rule)": "0x00000000",
                    "RAM00 Memory Rule Register 0 (ram00_mem_rule0)": "0x00000000",
                    "RAM00 Memory Rule Register 1 (ram00_mem_rule1)": "0x00000000",
                    "RAM00 Memory Rule Register 2 (ram00_mem_rule2)": "0x00000000",
                    "RAM00 Memory Rule Register 3 (ram00_mem_rule3)": "0x00000000",
                    "RAM01 Memory Rule Register 0 (ram01_mem_rule0)": "0x00000000",
                    "RAM01 Memory Rule Register 1 (ram01_mem_rule1)": "0x00000000",
                    "RAM01 Memory Rule Register 2 (ram01_mem_rule2)": "0x00000000",
                    "RAM01 Memory Rule Register 3 (ram01_mem_rule3)": "0x00000000",
                    "RAM1 Slave Rule Register (ram1_slave_rule)": "0x00000000",
                    "RAM10 Memory Rule Register 0 (ram10_mem_rule0)": "0x00000000",
                            "RAM10 Memory Rule Register 1 (ram10_mem_rule1)": "0x00000000",
                            "RAM10 Memory Rule Register 2 (ram10_mem_rule2)": "0x00000000",
                            "RAM10 Memory Rule Register 3 (ram10_mem_rule3)": "0x00000000",
                            "RAM11 Memory Rule Register 0 (ram11_mem_rule0)": "0x00000000",
                            "RAM11 Memory Rule Register 1 (ram11_mem_rule1)": "0x00000000",
                            "RAM11 Memory Rule Register 2 (ram11_mem_rule2)": "0x00000000",
                            "RAM11 Memory Rule Register 3 (ram11_mem_rule3)": "0x00000000",
                            "RAM2 Slave Rule Register (ram2_slave_rule)": "0x00000000",
                            "RAM20 Memory Rule Register 0 (ram20_mem_rule0)": "0x00000000",
                            "RAM20 Memory Rule Register 1 (ram20_mem_rule1)": "0x00000000",
                            "RAM20 Memory Rule Register 2 (ram20_mem_rule2)": "0x00000000",
                            "RAM20 Memory Rule Register 3 (ram20_mem_rule3)": "0x00000000",
                            "RAM21 Memory Rule Register 0 (ram21_mem_rule0)": "0x00000000",
                            "RAM21 Memory Rule Register 1 (ram21_mem_rule1)": "0x00000000",
                            "RAM21 Memory Rule Register 2 (ram21_mem_rule2)": "0x00000000",
                            "RAM21 Memory Rule Register 3 (ram21_mem_rule3)": "0x00000000",
                            "RAM22 Memory Rule Register 0 (ram22_mem_rule0)": "0x00000000",
                            "RAM22 Memory Rule Register 1 (ram22_mem_rule1)": "0x00000000",
                            "RAM22 Memory Rule Register 2 (ram22_mem_rule2)": "0x00000000",
                            "RAM22 Memory Rule Register 3 (ram22_mem_rule3)": "0x00000000",
                            "RAM23 Memory Rule Register 0 (ram23_mem_rule0)": "0x00000000",
                            "RAM23 Memory Rule Register 1 (ram23_mem_rule1)": "0x00000000",
                            "RAM23 Memory Rule Register 2 (ram23_mem_rule2)": "0x00000000",
                            "RAM23 Memory Rule Register 3 (ram23_mem_rule3)": "0x00000000",
                            "RAM3 Slave Rule Register (ram3_slave_rule)": "0x00000000",
                            "RAM30 Memory Rule Register 0 (ram30_mem_rule0)": "0x00000000",
                            "RAM30 Memory Rule Register 1 (ram30_mem_rule1)": "0x00000000",
                            "RAM30 Memory Rule Register 2 (ram30_mem_rule2)": "0x00000000",
                            "RAM30 Memory Rule Register 3 (ram30_mem_rule3)": "0x00000000",
                            "RAM31 Memory Rule Register 0 (ram31_mem_rule0)": "0x00000000",
                            "RAM31 Memory Rule Register 1 (ram31_mem_rule1)": "0x00000000",
```

```
                    "RAM31 Memory Rule Register 2 (ram31_mem_rule2)": "0x00000000",
                    "RAM31 Memory Rule Register 3 (ram31_mem_rule3)": "0x00000000",
                    "RAM32 Memory Rule Register 0 (ram32_mem_rule0)": "0x00000000",
                    "RAM32 Memory Rule Register 1 (ram32_mem_rule1)": "0x00000000",
                    "RAM32 Memory Rule Register 2 (ram32_mem_rule2)": "0x00000000",
                    "RAM32 Memory Rule Register 3 (ram32_mem_rule3)": "0x00000000",
                    "RAM33 Memory Rule Register 0 (ram33_mem_rule0)": "0x00000000",
                    "RAM33 Memory Rule Register 1 (ram33_mem_rule1)": "0x00000000",
                    "RAM33 Memory Rule Register 2 (ram33_mem_rule2)": "0x00000000",
                    "RAM33 Memory Rule Register 3 (ram33_mem_rule3)": "0x00000000",
                    "RAM4 Slave Rule Register (ram4_slave_rule)": "0x00000000",
                    "RAM40 Memory Rule Register 0 (ram40_mem_rule0)": "0x00000000",
                    "RAM40 Memory Rule Register 1 (ram40_mem_rule1)": "0x00000000",
                    "RAM40 Memory Rule Register 2 (ram40_mem_rule2)": "0x00000000",
                    "RAM40 Memory Rule Register 3 (ram40_mem_rule3)": "0x00000000",
                    "RAM41 Memory Rule Register 0 (ram41_mem_rule0)": "0x00000000",
                    "RAM41 Memory Rule Register 1 (ram41_mem_rule1)": "0x00000000",
                    "RAM41 Memory Rule Register 2 (ram41_mem_rule2)": "0x00000000",
                    "RAM41 Memory Rule Register 3 (ram41_mem_rule3)": "0x00000000",
                    "RAM42 Memory Rule Register 0 (ram42_mem_rule0)": "0x00000000",
                    "RAM42 Memory Rule Register 1 (ram42_mem_rule1)": "0x00000000",
                    "RAM42 Memory Rule Register 2 (ram42_mem_rule2)": "0x00000000",
                    "RAM42 Memory Rule Register 3 (ram42_mem_rule3)": "0x00000000",
                    "RAM43 Memory Rule Register 0 (ram43_mem_rule0)": "0x00000000",
                    "RAM43 Memory Rule Register 1 (ram43_mem_rule1)": "0x00000000",
                    "RAM43 Memory Rule Register 2 (ram43_mem_rule2)": "0x00000000",
                    "RAM43 Memory Rule Register 3 (ram43_mem_rule3)": "0x00000000",
                    "RAM5 Slave Rule Register (ram5_slave_rule)": "0x00000000",
                    "RAM50 Memory Rule Register 0 (ram50_mem_rule0)": "0x00000000",
                    "RAM50 Memory Rule Register 1 (ram50_mem_rule1)": "0x00000000",
                    "RAM50 Memory Rule Register 2 (ram50_mem_rule2)": "0x00000000",
                    "RAM50 Memory Rule Register 3 (ram50_mem_rule3)": "0x00000000",
                    "RAM51 Memory Rule Register 0 (ram51_mem_rule0)": "0x00000000",
                    "RAM51 Memory Rule Register 1 (ram51_mem_rule1)": "0x00000000",
                    "RAM51 Memory Rule Register 2 (ram51_mem_rule2)": "0x00000000",
                    "RAM51 Memory Rule Register 3 (ram51_mem_rule3)": "0x00000000",
                    "RAM52 Memory Rule Register 0 (ram52_mem_rule0)": "0x00000000",
                    "RAM52 Memory Rule Register 1 (ram52_mem_rule1)": "0x00000000",
                    "RAM52 Memory Rule Register 2 (ram52_mem_rule2)": "0x00000000",
                    "RAM52 Memory Rule Register 3 (ram52_mem_rule3)": "0x00000000",
                    "RAM53 Memory Rule Register 0 (ram53_mem_rule0)": "0x00000000",
                    "RAM53 Memory Rule Register 1 (ram53_mem_rule1)": "0x00000000",
                    "RAM53 Memory Rule Register 2 (ram53_mem_rule2)": "0x00000000",
                    "RAM53 Memory Rule Register 3 (ram53_mem_rule3)": "0x00000000",
                    "RAM6 Slave Rule Register (ram6_slave_rule)": "0x00000000",
                    "RAM60 Memory Rule Register 0 (ram60_mem_rule0)": "0x00000000",
                    "RAM60 Memory Rule Register 1 (ram60_mem_rule1)": "0x00000000",
                    "RAM60 Memory Rule Register 2 (ram60_mem_rule2)": "0x00000000",
                    "RAM60 Memory Rule Register 3 (ram60_mem_rule3)": "0x00000000",
                    "RAM61 Memory Rule Register 0 (ram61_mem_rule0)": "0x00000000",
                    "RAM61 Memory Rule Register 1 (ram61_mem_rule1)": "0x00000000",
                    "RAM61 Memory Rule Register 2 (ram61_mem_rule2)": "0x00000000",
                    "RAM61 Memory Rule Register 3 (ram61_mem_rule3)": "0x00000000",
                    "RAM62 Memory Rule Register 0 (ram62_mem_rule0)": "0x00000000",
                    "RAM62 Memory Rule Register 1 (ram62_mem_rule1)": "0x00000000",
                    "RAM62 Memory Rule Register 2 (ram62_mem_rule2)": "0x00000000",
                    "RAM62 Memory Rule Register 3 (ram62_mem_rule3)": "0x00000000",
                    "RAM63 Memory Rule Register 0 (ram63_mem_rule0)": "0x00000000",
                    "RAM63 Memory Rule Register 1 (ram63_mem_rule1)": "0x00000000",
                    "RAM63 Memory Rule Register 2 (ram63_mem_rule2)": "0x00000000",
```

```
                            "RAM63 Memory Rule Register 3 (ram63_mem_rule3)": "0x00000000",
                            "RAM7 Slave Rule Register (ram7_slave_rule)": "0x00000000",
                            "RAM70 Memory Rule Register 0 (ram70_mem_rule0)": "0x00000000",
                            "RAM70 Memory Rule Register 1 (ram70_mem_rule1)": "0x00000000",
                            "RAM70 Memory Rule Register 2 (ram70_mem_rule2)": "0x00000000",
                            "RAM70 Memory Rule Register 3 (ram70_mem_rule3)": "0x00000000",
                            "RAM71 Memory Rule Register 0 (ram71_mem_rule0)": "0x00000000",
                            "RAM71 Memory Rule Register 1 (ram71_mem_rule1)": "0x00000000",
                            "RAM71 Memory Rule Register 2 (ram71_mem_rule2)": "0x00000000",
                            "RAM71 Memory Rule Register 3 (ram71_mem_rule3)": "0x00000000",
                            "RAM72 Memory Rule Register 0 (ram72_mem_rule0)": "0x00000000",
                            "RAM72 Memory Rule Register 1 (ram72_mem_rule1)": "0x00000000",
                            "RAM72 Memory Rule Register 2 (ram72_mem_rule2)": "0x00000000",
                            "RAM72 Memory Rule Register 3 (ram72_mem_rule3)": "0x00000000",
                            "RAM73 Memory Rule Register 0 (ram73_mem_rule0)": "0x00000000",
                            "RAM73 Memory Rule Register 1 (ram73_mem_rule1)": "0x00000000",
                            "RAM73 Memory Rule Register 2 (ram73_mem_rule2)": "0x00000000",
                            "RAM73 Memory Rule Register 3 (ram73_mem_rule3)": "0x00000000",
                            "RAM8 Slave Rule Register (ram8_slave_rule)": "0x00000000",
                            "RAM80 Memory Rule Register 0 (ram80_mem_rule0)": "0x00000000",
                            "RAM80 Memory Rule Register 1 (ram80_mem_rule1)": "0x00000000",
                            "RAM80 Memory Rule Register 2 (ram80_mem_rule2)": "0x00000000",
                            "RAM80 Memory Rule Register 3 (ram80_mem_rule3)": "0x00000000",
                            "RAM81 Memory Rule Register 0 (ram81_mem_rule0)": "0x00000000",
                            "RAM81 Memory Rule Register 1 (ram81_mem_rule1)": "0x00000000",
                            "RAM81 Memory Rule Register 2 (ram81_mem_rule2)": "0x00000000",
                            "RAM81 Memory Rule Register 3 (ram81_mem_rule3)": "0x00000000",
                            "RAM82 Memory Rule Register 0 (ram82_mem_rule0)": "0x00000000",
                            "RAM82 Memory Rule Register 1 (ram82_mem_rule1)": "0x00000000",
                            "RAM82 Memory Rule Register 2 (ram82_mem_rule2)": "0x00000000",
                            "RAM82 Memory Rule Register 3 (ram82_mem_rule3)": "0x00000000",
                            "RAM83 Memory Rule Register 0 (ram83_mem_rule0)": "0x00000000",
                            "RAM83 Memory Rule Register 1 (ram83_mem_rule1)": "0x00000000",
                            "RAM83 Memory Rule Register 2 (ram83_mem_rule2)": "0x00000000",
                            "RAM83 Memory Rule Register 3 (ram83_mem_rule3)": "0x00000000",
                            "HiFi4 DSP Slave Rule Register (pif_hifi4_x_slave_rule0)": "0x00000000",
                            "HiFi4 DSP Memory Rule Register 0 (pif_hifi4_x_mem_rule0)": "0x00000000",
                            "HiFi4 DSP Memory Rule Register 1 (pif_hifi4_x_mem_rule1)": "0x00000000",
                            "HiFi4 DSP Memory Rule Register 2 (pif_hifi4_x_mem_rule2)": "0x00000000",
                            "HiFi4 DSP Memory Rule Register 3 (pif_hifi4_x_mem_rule3)": "0x00000000",
                            "APB Bridge Slave Rule Register (apb_bridge_slave_rule)": "0x00000000",
                        "APB Bridge Group 0 Memory Rule Register 0 (apb_grp0_mem_rule0)": "0x00000000",
                        "APB Bridge Group 0 Memory Rule Register 1 (apb_grp0_mem_rule1)": "0x00000000",
                        "APB Bridge Group 0 Memory Rule Register 2 (apb_grp0_mem_rule2)": "0x00000000",
                        "APB Bridge Group 0 Memory Rule Register 3 (apb_grp0_mem_rule3)": "0x00000000",
                        "APB Bridge Group 1 Memory Rule Register 0 (apb_grp1_mem_rule0)": "0x00000000",
                        "APB Bridge Group 1 Memory Rule Register 1 (apb_grp1_mem_rule1)": "0x00000000",
                        "APB Bridge Group 1 Memory Rule Register 2 (apb_grp1_mem_rule2)": "0x00000000",
                        "APB Bridge Group 1 Memory Rule Register 3 (apb_grp1_mem_rule3)": "0x00000000",
                        "AHB Peripherals 0 Slave Rule Register (ahb_periph0_slave_rule)": "0x00000000",
                         "AIPS bridge 0 Memory Rule Register (aips_bridge0_slave_rule)": "0x00000000",
                        "AHB Peripherals 1 Slave Rule Register (ahb_periph1_slave_rule)": "0x00000000",
                          "AIPS bridge 1 Slave Rule Register (aips_bridge1_slave_rule)": "0x00000000",
                         "AIPS bridge 1 Memory Rule Register 0 (aips_bridge1_mem_rule0)": "0x00000000",
                         "AIPS bridge 1 Memory Rule Register 1 (aips_bridge1_mem_rule1)": "0x00000000",
                         "AIPS bridge 1 Memory Rule Register 2 (aips_bridge1_mem_rule2)": "0x00000000",
                         "AIPS bridge 1 Memory Rule Register 3 (aips_bridge1_mem_rule3)": "0x00000000",
                        "AHB Peripherals 2 Slave Rule Register (ahb_periph2_slave_rule)": "0x00000000",
                            "AHB Peripherals 2 Security Memory Rule Register
(ahb_periph2_security_mem_rule)": "0x00000000",
```

```
                    "AHB Peripherals 3 Slave Rule Register (ahb_periph3_slave_rule)": "0x00000000",
                        "Secure GPIO Mask Register 0 (sec_gpio_mask0)": "0xffffffff",
                        "Secure GPIO Mask Register 1 (sec_gpio_mask1)": "0xffffffff",
                        "Secure GPIO Mask Register 2 (sec_gpio_mask2)": "0xffffffff",
                        "Secure GPIO Mask Register 3 (sec_gpio_mask3)": "0xffffffff",
                        "Secure GPIO Mask Register 4 (sec_gpio_mask4)": "0xffffffff",
                        "Secure GPIO Mask Register 5 (sec_gpio_mask5)": "0xffffffff",
                        "Secure GPIO Mask Register 6 (sec_gpio_mask6)": "0xffffffff",
                        "Secure GPIO Mask Register 7 (sec_gpio_mask7)": "0xffffffff",
                        "Secure DSP Interrupt Mask Register (sec_hifi4_int_mask)": "0xffffffff",
                        "Secure GPIO Mask Lock Register (sec_gpio_mask_lock)": "0x0000aaaa",
                        "Master Secure Level Register (master_sec_reg)": "0x80000000",
                        "Master Secure Level Anti-pole Register (master_sec_anti_pol_reg)":
"0xbfffffff",
                        "M33 Lock Control Register (m33_lock_reg)": "0x800002aa",
                        "Secure Control Duplicate Register (misc_ctrl_dp_reg)": "0x0000aaaa",
                        "Secure Control Register (misc_ctrl_reg)": "0x0000aaaa",
                        "Miscellaneous TZM settings (misc_tzm_settings)": "0x00000000"
}
}
```


10 修订记录

下表包含本用户指南所做更改的历史记录。

表 28. 修订记录

修订版本号	日期	重要变化
0	2015 年 9 月	初版
1	2016 年 4 月	Kinetis 引导加载程序版本 v2.0
2	2018 年 5 月	MCU 引导加载程序版本 v2.5.0
3	2018 年 9 月	MCU 引导加载程序版本 v2.6.0
4	2018 年 10 月	LPC55S69 更新
5	2018 年 11 月	MCU 引导加载程序版本 v2.7.0
6	2019 年 1 月	LPC54x0xx 更新
7	2020 年 2 月	LPC55S1x 更新

如何联系我们

主页：

nxp.com

网络支持：

nxp.com/support

欲了解更多信息，请访问：http://www.nxp.com

欲咨询销售办事处地址，请发送电子邮件至：salesaddresses@nxp.com

发布日期：2020 年 2 月

文档编号：MBOOTELFTOSBUG