Title: TG7100B Flash分区说明

1. 分区表用途

分区表信息存储于各个应用方案的config.yaml配置文件中，用于定义各个分区的存储地址、加载地址以及升级相关的参数。SDK中提供的系统工具product，将根据分区表配置文件生成相应系统升级包。用户设置完分区表信息后，可通过该系统工具生成系统升级包，完成系统的固件升级。

2. 分区表示例

每个应用方案的分区表不尽相同，SDK中的分区表目录为board/tg7100b/configs/config.yaml，该文件为文本格式，可打开查看分区情况。

示例：

```
mtb_version: 4
chip: tg7100b
diff:
  digest_type: SHA1
  signature_type: RSA1024
  fota_version: 0
  ram_buf: 50      #DEC     KB    ( max ram need)
  flash_buf: 16     #DEC    KB    ( buffer size)
  flash_sector: 4096 #DEC   byte ( flash sector)
  diff_mode: 010    #BIN
  double_control: 1
flash:
  base_address: 0x11000000
  run_base_address: 0x1FFF3800 # base address of Storage
  sector: 4096       # Bytes
  size: 524288      # $(sector count) * sector
partitions:
  # BootROM引导表，定义该引导表的Flash存储地址和大小
  - { name: bomtb,  address: 0x11002100, size: 0x001000 }
  # 工厂参数区，记录出厂配置
  - { name: FCDS,   address: 0x11004000, size: 0x001000 }
  # BootLoader引导表，定义该引导表的Flash存储地址和大小
  - { name: imtb,   address: 0x11005000, size: 0x002000 }
  # 用户数据存储区信息，  定义该分区的Flash存储地址和大小，升级属性为全量升级、不需要验签
  - { name: kv,     address: 0x11007000, size: 0x002000 }
  # BootLoader镜像信息，定义该镜像的Flash存储地址和大小，SRAM加载地址
  - { name: boot,   address: 0x11009000, size: 0x005000, load_addr: 0x2000A800}
  # jumptable信息，定义jumptable的Flash存储地址和大小，SRAM加载地址，升级属性为全量升级、需要验签
  - { name: jumptb, address: 0x1100E000, size: 0x001000, load_addr: 0x1FFF0800,
update: FULL, verify: true }
  # 应用镜像信息（需要加载至SRAM运行的部分），定义应用镜像的Flash存储地址和大小，SRAM加载地址，升级属性为全量升级、需要验签
  - { name: prim,   address: 0x1100F000, size: 0x012000, load_addr: 0x1fff4800,
update: FULL, verify: true }
  # 升级备份区信息，  定义该分区的Flash存储地址和大小
  - { name: misc,   address: 0x11021000, size: 0x039000 }
```

```
  # 应用镜像信息（直接在XIP Flash运行的部分），定义应用镜像的Flash存储地址和大小，升级属性为全量升级、需要验签
  - { name: xprim,  address: 0x1105A000, size: 0x026000, update: FULL, verify: true }
```

字段	说明
mtb_version	MTB版本号。目前支持的版本为4，不允许用户修改。
chip	芯片类型
flash	Flash相关参数
partitions	分区表相关参数

flash字段：

字段	是否必选	说明
base_address	是	Flash基地址
run_base_address	否	SRAM基地址
sector	是	Flash扇区大小，单位：Byte
size	是	Flash大小，单位：Byte

partitions字段：

字段	是否必选	说明
name	是	分区名字，最长8个字节
address	是	该分区存储于Flash上的起始地址
load_addr	否	该分区在SRAM中的加载地址，如果未定义该字段，则默认使用`address`为加载地址；如果定义了该字段，则将程序加载至RAM上运行
size	是	该分区大小，单位：Byte
update	否	该分区升级类型[`DIFF`，`FULL`]，默认值为`DIFF`；DIFF为差分升级，FULL为全量升级。目前只支持全量升级。
verify	否	是否要对该分区进行签名验证
file	否	指定该分区对应的镜像文件名字，不填则表示文件名与分区名相同

分区	说明
bomtb	BootROM引导表，定义BootLoader镜像的大小，存储地址以及运行地址
FCDS	工厂参数区，记录出厂配置，如设备MAC地址、出厂参数等
imtb	BootLoader引导表，定义应用镜像的大小，存储地址以及运行地址
boot	BootLoader镜像，该部分镜像需要加载至SRAM上运行。目前不支持升级。
jumptb	ROM程序跳转表，需要加载至SRAM上运行
prim	应用镜像，该部分镜像需要加载至SRAM上运行
xprim	应用镜像，该部分镜像存放并运行在XIP Flash上
misc	升级备份区
kv	用户数据存储区，用于存储用户数据

> bomtb\imtb\boot\jumptb分区的调整涉及程序引导流程，不建议用户修改。
>
> prim\xprim\misc\kv 可以根据实际情况进行调整。

3. 分区调整FAQ

- 当分区需要调整时，需要注意的是：
  分区调整需要根据sector大小来调整，目前TG7100B上的Flash是4K一个sector。
- 分区表config.yaml中定义的prim分区用来指定在RAM中运行的代码的存放位置和大小。

  参考gcc_eflash.ld文件，可以看到目前LR_IRAM4大小为0x1E000，这是预留的实际可用RAM大小；参考分区表文件config.yaml中，可以看到prim大小定义为0x12000。
- 分区表config.yaml中定义的xprim分区用来指定存放至Flash的代码的位置和大小.

  `gcc_eflash.ld文件里的LR_IROM0地址和length需要保持与config.yaml文件中的xprim定义一致。`
  参考gcc_eflash.ld文件，可以看到LR_IROM0的首地址和size定位为

```
LR_IROM0   : ORIGIN = 0x11056000, LENGTH = 0x21000
```

  参考分区表文件config.yaml中，可以看到xprim定义为

```
{ name: xprim,  address: 0x11056000, size: 0x025000, update: FULL, verify: true
}
```

- 如果编译提示`Partition[prim]'s size[xxxxxx] is less than binary file size`，可以调整config.yaml文件中prim的大小，但同时需要考虑预留的RAM空间是否满足程序正常运行。

  修改方法如下:
  config.yaml文件中prim首地址不变，增加size;分区表中其他分区的首地址和size根据需要改动。
  例如size修改为0x13000

```
{ name: prim,   address: 0x1100D000, size: 0x013000, load_addr: 0x1FFF4800,
update: FULL, verify: true }
{ name: misc,   address: 0x11020000, size: 0x037000, update: FULL, verify:
true }
{ name: xprim,  address: 0x11057000, size: 0x021000, update: FULL, verify:
true }
{ name: kv,     address: 0x11078000, size: 0x008000, update: FULL, verify:
true }
```

- 如果编译提示LR_IRAM4 overflow，说明代码中存在大的数组变量定义，内存不够用。

  此时只能尽量避免大数组的内存占用，比如减少任务栈的大小，减少新任务的创建

- 如果编译提示LR_IROM0 overflow，需要调整xprim大小。

  修改方法如下:
  xprim首地址不变，增加size;分区表中KV的首地址和size根据需要改动。
  config.yaml中更改为

```
{ name: prim,   address: 0x1100D000, size: 0x012000, load_addr: 0x1FFF4800,
update: FULL, verify: true }
{ name: misc,   address: 0x1101F000, size: 0x037000, update: FULL, verify: true
}
{ name: xprim,  address: 0x11056000, size: 0x025000, update: FULL, verify: true
}
{ name: kv,     address: 0x1107B000, size: 0x005000, update: FULL, verify: true
}
```

  gcc_eflash.ld文件更改为

```
  LR_IROM0   : ORIGIN = 0x11056000, LENGTH = 0x25000
```

- 如果出现misc overflow，说明用来存放OTA升级包的misc分区不够用了，这时需要根据config.yaml中的各个分区的定义来调整misc分区。增加misc分区的size，随之而来的，xprim分区和KV分区的首地址和大小都要调整。
  以misc分区增加4K空间，xprim分区减少4K空间为例，修改方法为：
  config.yaml中更改为

```
{ name: prim,   address: 0x1100D000, size: 0x012000, load_addr: 0x1FFF4800,
update: FULL, verify: true }
{ name: misc,   address: 0x1101F000, size: 0x038000, update: FULL, verify: true
}
{ name: xprim,  address: 0x11057000, size: 0x024000, update: FULL, verify: true
}
{ name: kv,     address: 0x1107B000, size: 0x005000, update: FULL, verify: true
}
```

gcc_eflash.ld文件更改为

```
 LR_IROM0   : ORIGIN = 0x11057000, LENGTH = 0x24000
```

- 更改分区时还需要注意
  board_init.c中也有一部分分区的定义，当KV分区和MISC分区发生改变时，需要同步到board\tg7100b\init\board_init.c中

```
const hal_logic_partition_t hal_partitions[] =
    {
...
        [HAL_PARTITION_KV] =
            {
                .partition_owner = HAL_FLASH_EMBEDDED,
                .partition_description = "kv",  //对应config.yaml中的KV分区
                .partition_start_addr = 0x1107B000,
                .partition_length = 0x5000,
                .partition_options = PAR_OPT_READ_EN | PAR_OPT_WRITE_EN,
            },
        [HAL_PARTITION_OTA_TEMP] =
            {
                .partition_owner = HAL_FLASH_EMBEDDED,
                .partition_description = "ota", //对应config.yaml中的MISC分区
                .partition_start_addr = 0x1101F000,
                .partition_length = 0x38000,
                .partition_options = PAR_OPT_READ_EN | PAR_OPT_WRITE_EN,
            }};
```