



ZS110A-SPI-XIP 介绍

发布 **1.0.0**

2019 年 03 月 04 日

目录

1	文档介绍	1
1.1	文档目的	1
1.2	术语说明	1
1.3	参考文档	1
1.4	版本历史	2
2	SPI-XIP 模式	3
2.1	SPI-XIP 模式	3
2.2	SPI-XIP 开发的改动点	5
3	XIP 模式-RAM 空间	6
4	XIP 模式-FLASH 空间	7
5	XIP 模式-固件	8
5.1	keil 工程	8
5.2	firmware_xip.xml	11
5.3	upgrade_xip.ini	12
5.4	build_xip.bat	12
6	XIP 模式-OTA	13
6.1	分区的起始地址	13
6.2	分区表的起始地址	13
6.3	APK 端分区表的处理	13

1.1 文档目的

介绍 ZS110A SPI-XIP 运行模式的原理和相关改动。

1.2 术语说明

表 1.1: 术语说明

术语	说明
ZEPHYR	为所有资源受限设备，构建了针对低功耗、小型内存微处理器设备而进行优化的物联网嵌入式小型、可扩展的实时操作系统（RTOS）
ZS110A	基于 ZEPHYR 建立的，用于在 ACTIONS ATB110X SOC 上开发 BLE 应用的 SDK
XIP	eXecute In Place，指不需要将代码加载到 RAM 而是直接在存储介质上运行代码

1.3 参考文档

- <http://docs.zephyrproject.org/>

1.4 版本历史

表 1.2: 版本历史

日期	版本	注释	作者
2018-12-17	1.0	初始版本	ZS110A 项目组

2.1 SPI-XIP 模式

对于逻辑很复杂的应用，RAM 已经不够了，此时需要启用 XIP 模式，ram 全部用于存储数据。

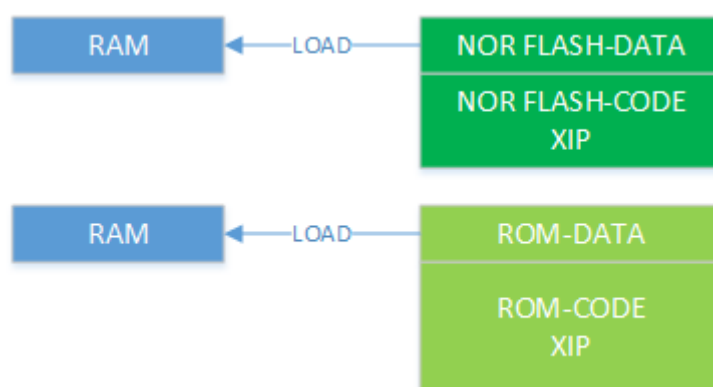


图 2.1: xip 模式

XIP 模式是通过硬件 SPI-CACHE 模块实现的。该模块支持的 feature 如下：

- 支持代码直接运行在 SPI norflash 上（只有 SPI0 控制器支持）
- 对 code 和 rodata 进行缓存
- 代码运行空间有 16MBytes: 0x01000000~0x01ffffff
- 代码存储空间有 4MBytes: 0x0~0x3ffff

- 代码运行空间和存储空间的映射关系可以动态配置。

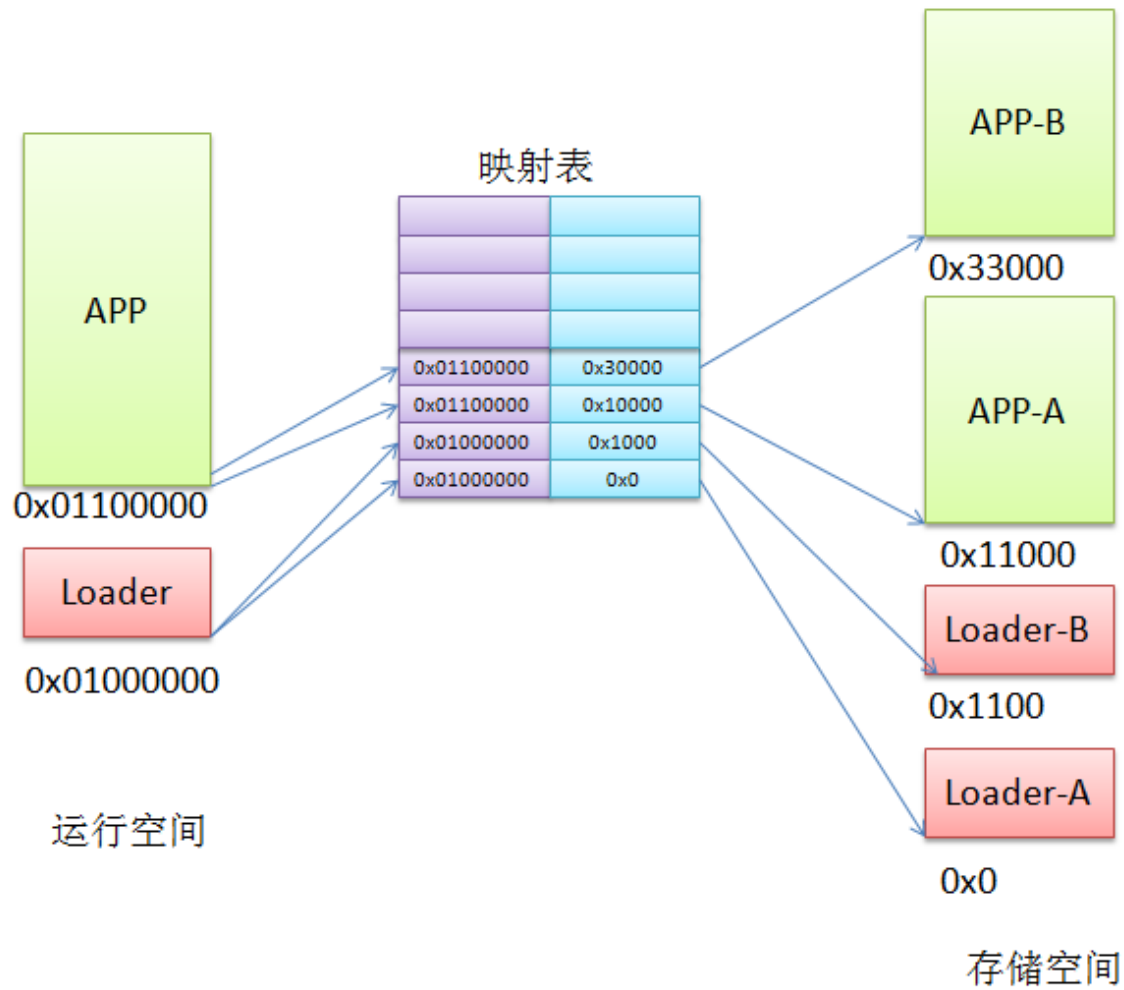


图 2.2: 代码运行空间和存储空间的映射关系

- 代码存储空间的数据需要增加 crc（每 32bytes 增加 2Bytes 的 crc 值）。

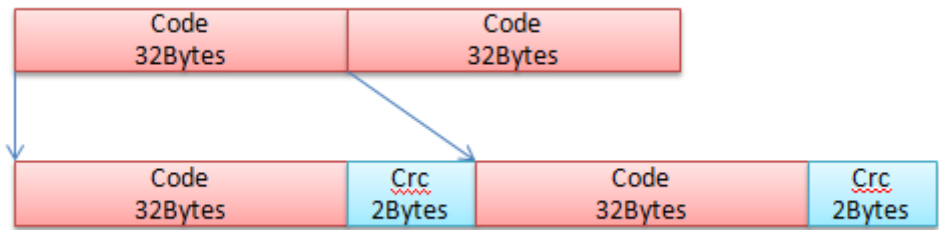


图 2.3: 分区增加 crc 值

注解: XIP 执行代码的分区 (例如: loader、rcu) 才需要插入 crc。普通的存储数据的分区 (nvram) 不需要插 crc。

2.2 SPI-XIP 开发的改动点

- RAM 空间
- FLASH 空间
- FW
- OTA

XIP 模式-RAM 空间

XIP 模式下，RAM 的高 8KBytes 将被硬件 SPI-CACHE 使用，软件只能使用低 40KBytes。

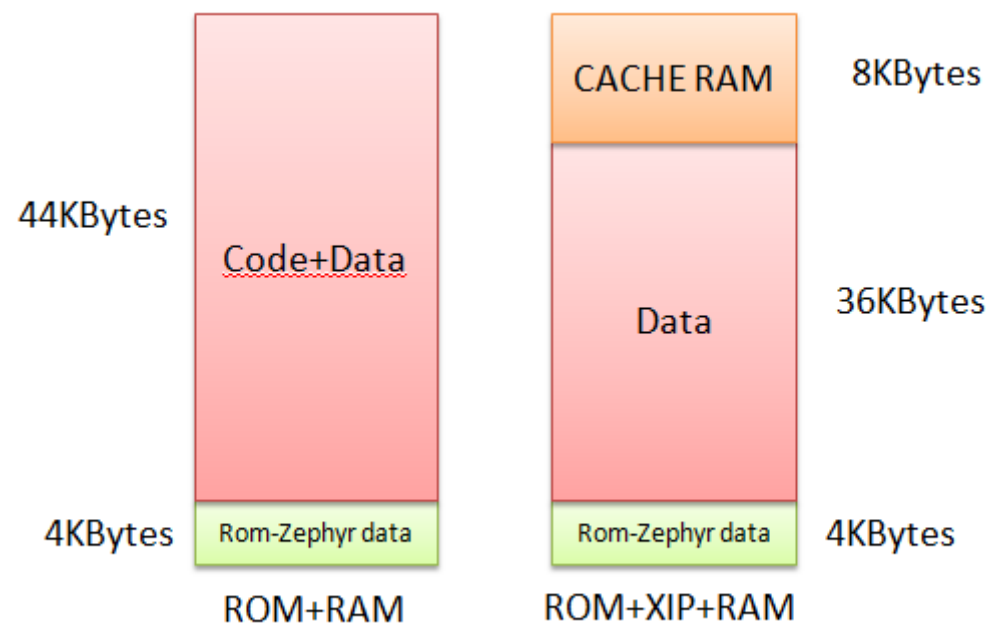


图 3.1: ram 空间

XIP 模式-FLASH 空间

以 rcu sample 为例，XIP 模式下 flash 空间划分如下：

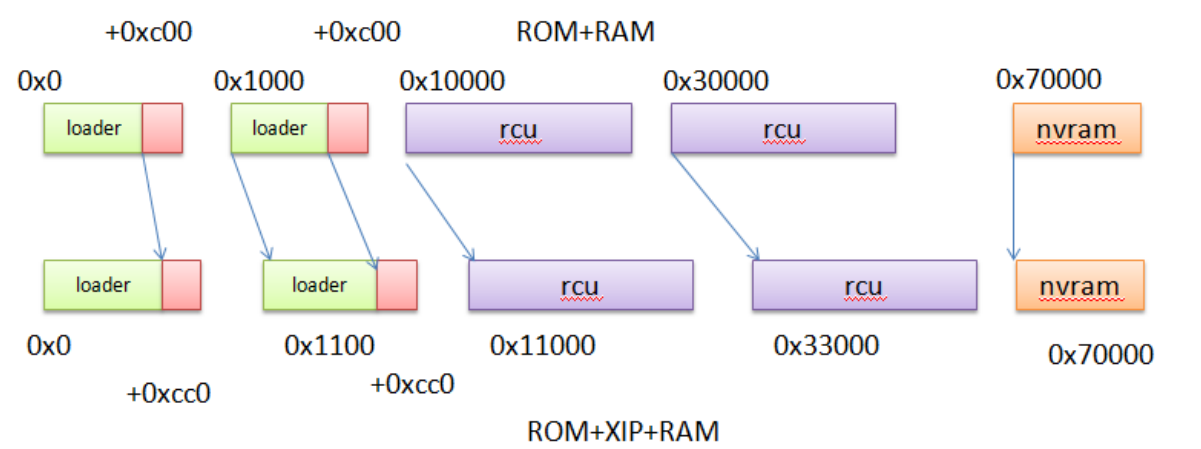


图 4.1: flash 空间

以 rcu sample 为例介绍固件的相关改动。

5.1 keil 工程

1. 新建 project target: spi_xip。

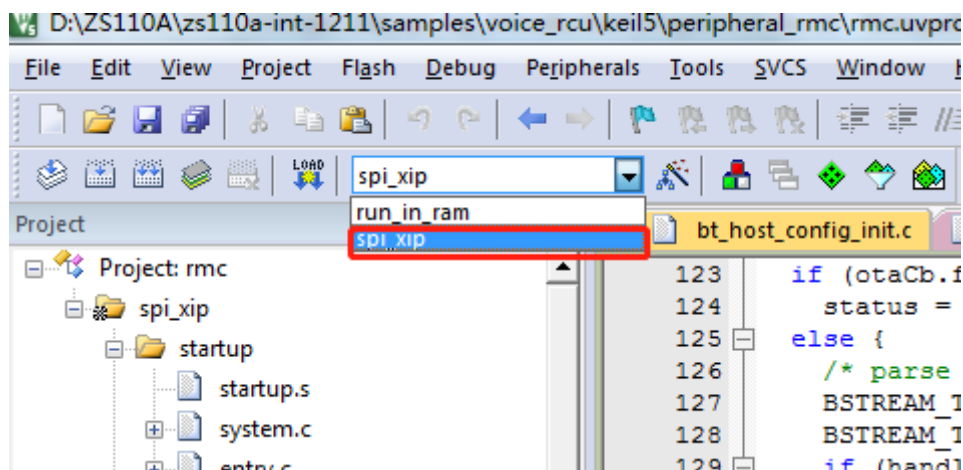


图 5.1: 新建 project target

2. 通过宏定义 CONFIG_SPI0_XIP 来区分代码

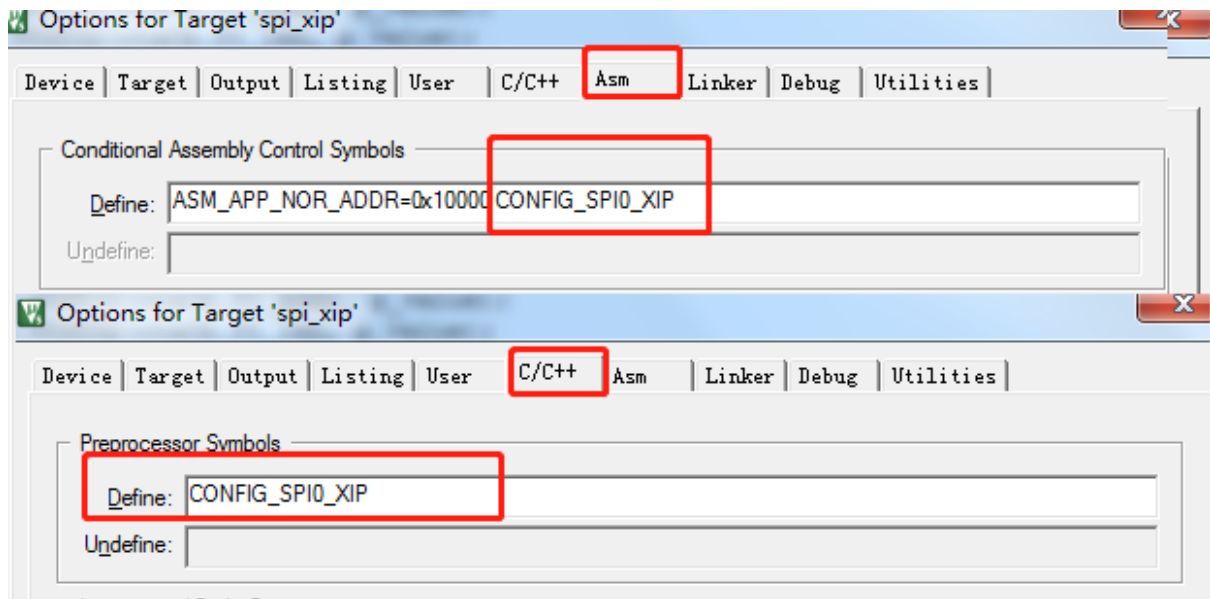


图 5.2: keil 工程增加宏定义 CONFIG_SPI0_XIP

3. link.sct

XIP 模式运行在 0x01000000~0x01ffffff 这 16MBytes 空间。

在遥控器方案中, loader 被链接到 0x01000000 地址段, rcu 被链接到 0x01100000 地址段。

```
; load region size_region
LR_IROM 0x01100000
{
    ; load address = execution address
    ER_IROM 0x01100000
    {
        *.o (RESET, +First)
        *(InRoot$$Sections)
        .ANY (+RO)
    }

    RAM_RETENTION 0x20001000
    {
        .ANY (.ramfunc)
        .ANY (.ram_retention_code)
        .ANY (.ram_retention_data)
    }

    ; RW data
    RW_IRAM_DEVICE_PRE_KERNEL_1 +0
    {
        *.o(.init_PRE_KERNEL_1*)
    }
    RW_IRAM_DEVICE_PRE_KERNEL_2 +0
    {
        *.o(.init_PRE_KERNEL_2*)
    }
    RW_IRAM_DEVICE_POST_KERNEL +0
    {
        *.o(.init_POST_KERNEL*)
    }
    RW_IRAM_DEVICE_APPLICATION +0
    {
        *.o(.init_APPLICATION*)
    }
}
```

图 5.3: keil 工程 sct 文件

4. download 配置

- 务必勾选 “Do not Erase”
- Programming Algorithm 中的 Address Range 请根据 link.sct 实际情况填写。

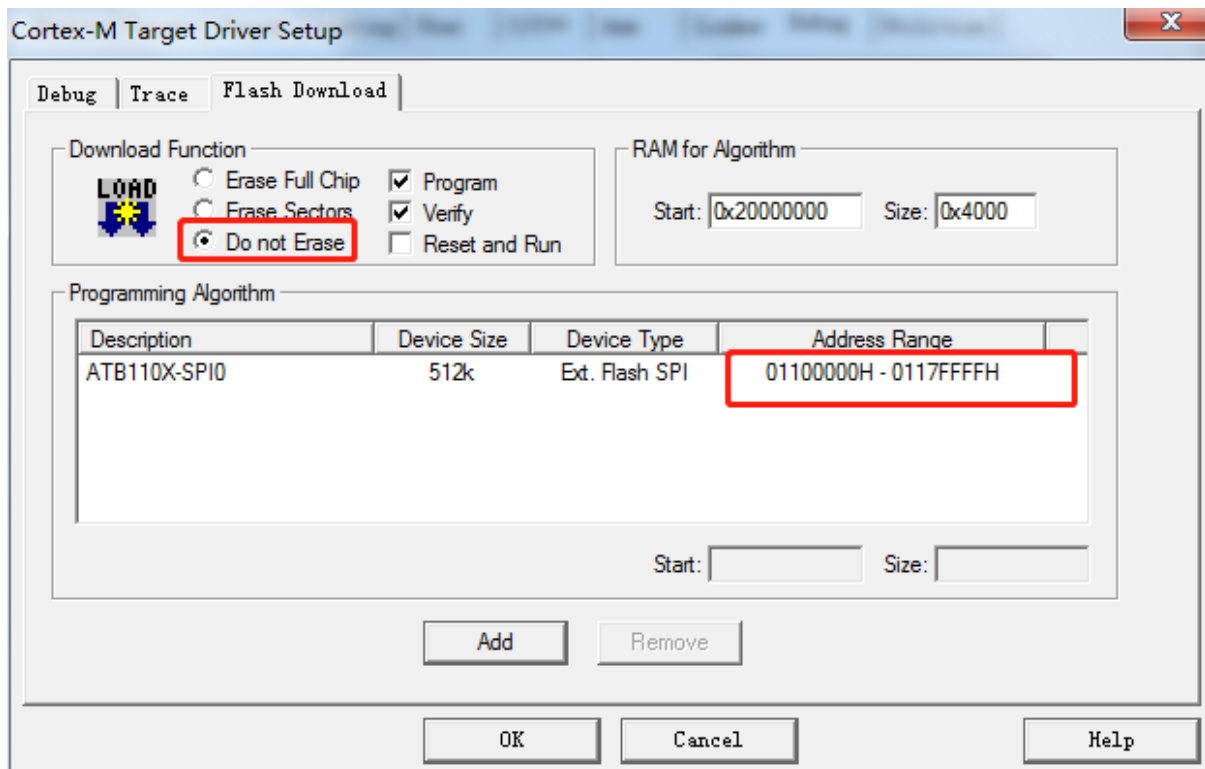


图 5.4: keil download 配置

5.2 firmware_xip.xml

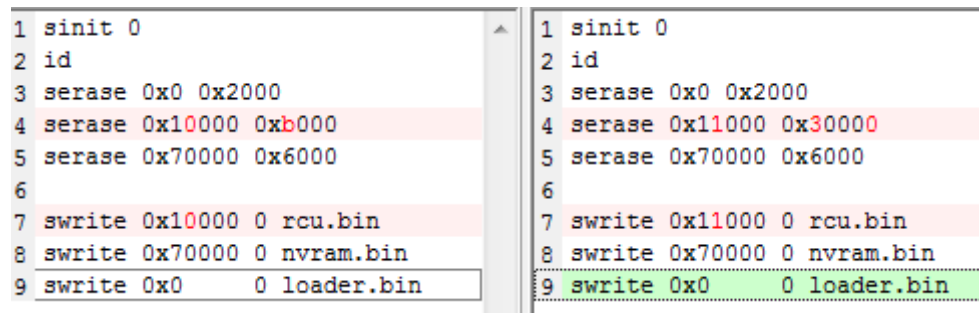
XIP 执行代码的分区需要设置 enable_crc 字段为 true

<pre> <firmware> <description>Firmware layout for ZS110A</description> <disk_size>0x80000</disk_size> <partitions> <partition> <type>BOOT</type> <name>fw0_boot</name> <file>loader.bin</file> <address>0x0</address> <enable_crc>false</enable_crc> <enable_randomize>false</enable_randomize> <enable_ota>true</enable_ota> <enable_raw>true</enable_raw> <enable_dfu>true</enable_dfu> <fw_id>0</fw_id> </partition> <partition> <type>BOOT</type> <name>fw1_boot</name> <file>loader.bin</file> <address>0x1000</address> <enable_crc>false</enable_crc> <enable_randomize>false</enable_randomize> <enable_ota>true</enable_ota> <enable_raw>true</enable_raw> <enable_dfu>false</enable_dfu> <fw_id>1</fw_id> </partition> </partitions> </pre>	<pre> 2 <firmware> 3 <description>Firmware layout for ZS110A</description> 4 <disk_size>0x80000</disk_size> 5 6 <partitions> 7 <partition> 8 <type>BOOT</type> 9 <name>fw0_boot</name> 10 <file>loader.bin</file> 11 <address>0x0</address> 12 <enable_crc>true</enable_crc> 13 <enable_randomize>false</enable_randomize> 14 <enable_ota>true</enable_ota> 15 <enable_raw>true</enable_raw> 16 <enable_dfu>true</enable_dfu> 17 <fw_id>0</fw_id> 18 </partition> 19 20 <partition> 21 <type>BOOT</type> 22 <name>fw1_boot</name> 23 <file>loader.bin</file> 24 <address>0x1000</address> 25 <enable_crc>true</enable_crc> 26 <enable_randomize>false</enable_randomize> 27 <enable_ota>true</enable_ota> 28 <enable_raw>true</enable_raw> 29 <enable_dfu>false</enable_dfu> 30 <fw_id>1</fw_id> 31 </partition> </pre>
---	--

图 5.5: 配置 xml 文件的 enable_crc 字段

5.3 upgrade_xip.ini

对应 flash 分区地址改动, 需要同步做修改



1	sinit 0
2	id
3	serase 0x0 0x2000
4	serase 0x10000 0xb000
5	serase 0x70000 0x6000
6	
7	swrite 0x10000 0 rcu.bin
8	swrite 0x70000 0 nvram.bin
9	swrite 0x0 0 loader.bin

1	sinit 0
2	id
3	serase 0x0 0x2000
4	serase 0x10000 0x30000
5	serase 0x70000 0x6000
6	
7	swrite 0x10000 0 rcu.bin
8	swrite 0x70000 0 nvram.bin
9	swrite 0x0 0 loader.bin

图 5.6: 配置 upgrade 文件中的分区地址

5.4 build_xip.bat

双击 build_xip.bat 批处理后自动生成固件和 ota 包。

以 rcu sample 为例子介绍 XIP 模式下 OTA 的相关改动（主要要 APK 端的改动）

6.1 分区的起始地址

firmware_xip.xml 中 enable_crc 字段配置为 true 的分区的起始地址需要转换为插入 crc 之后的地址。

$$\text{partition_offset} = (\text{address} / 32) * 34$$

6.2 分区表的起始地址

1. loader A 的分区表的地址 $= (0xc00 / 32) * 34 = 0xcc0$
2. loader B 的分区表的地址 $= ((0x1000 + 0xc00) / 32) * 34 = 0x1dc0$

6.3 APK 端分区表的处理

1. 小机端返回给 APK 端的分区表是带有 crc 值的，APK 端处理分区表时需要跳过 crc 值。
2. APK 端修改分区表之后还需要更新 crc 值。
3. APK 端发命令擦除 loader 分区时需注意：
 - loader B 分区跨了 2 个 4KBytes。

- 为了实现方便, 插入了 crc 值的 Loader A 分区 (包括分区表) 必须 <4KBytes。

List of Figures

2.1	xip 模式	3
2.2	代码运行空间和存储空间的映射关系	4
2.3	分区增加 crc 值	4
3.1	ram 空间	6
4.1	flash 空间	7
5.1	新建 project target	8
5.2	keil 工程增加宏定义 CONFIG_SPI0_XIP	9
5.3	keil 工程 sct 文件	10
5.4	keil download 配置	11
5.5	配置 xml 文件的 enable_crc 字段	11
5.6	配置 upgrade 文件中的分区地址	12

List of Tables

1.1	术语说明	1
1.2	版本历史	2