



XR806 XIP 方案 使用指南

版本号：1.0

发布时间：2020-11-09

版本历史

版本	日期	责任人	版本描述
1.0	2020-11-09	AWA 0498	创建文档。

目录

版本历史.....	i
目录.....	ii
1 前言.....	1
1.1 文档简介.....	1
1.2 目标读者.....	1
1.3 适用范围.....	1
1.4 文档约定.....	1
1.4.1 标志说明.....	1
1.4.2 地址与数据描述方法约定.....	1
1.4.3 数值单位约定.....	2
2 概述.....	3
2.1 背景说明.....	3
2.2 规格特性.....	3
2.3 文件位置.....	3
3 应用说明.....	5
3.1 应用简述.....	5
3.2 配置说明.....	5
3.3 链接文件格式说明.....	6
3.4 地址确认.....	7
4 常见问题.....	9
4.1 编译报内存不足错误.....	9
4.2 开机异常.....	9
4.3 新加入 XIP 或 PSRAM 后运行挂死.....	9
4.4 性能偏低.....	9
附录 A： 术语表.....	11

1 前言

1.1 文档简介

本文档介绍了 XR806 平台上 XIP 方案的使用方法及其注意事项。

1.2 目标读者

使用 XR806 SDK 的开发人员。

1.3 适用范围

此文档适用于 XR806 SDK，支持 XR806 系列芯片产品。

1.4 文档约定

1.4.1 标志说明

本文档采用各种醒目的标志来表示在操作过程中应该特别注意的地方，这些标志的含义如下：

标识	说明
 警告	该标志后的说明应给予格外关注，如果不遵守，可能会导致人员受伤或死亡。
 注意	提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。
 说明	为准确理解文中指令、正确实施操作而提供的补充或强调信息。
 窍门	一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

符号	例子	说明
0x	0x0200, 0x79	地址或数据以 16 进制表示。
0b	0b010, 0b00 000 111	数据采用二进制表示(寄存器描述除外)。
x	00X, XX1	数据描述中，x 代表 0 或 1。 例如，00X 代表 000 或 001；XX1 代表 001, 011, 101 或 111。

1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

类型	符号	对应数值
数据容量（如 NAND 容量）	1 K	1024
	1 M	1 048 576
	1 G	1 073 741 824
频率，数据速率等	1 k	1000
	1 M	1 000 000
	1 G	1 000 000 000

2 概述

2.1 背景说明

XR806 是一颗高集成度无线应用 MCU，通常会内部 SiP 或外挂一片 Flash，Flash 除了用于较大体积的数据的存放，还可直接在 Flash 闪存内执行（XIP, eXecute In Place）指令和读取只读型数据。Flash 的读取数据没有 SRAM 快，性能有一定损失，但因为 Flash 便宜且容量大，节约了 SRAM 的消耗降低了成本。

2.2 规格特性

除 Flash 外，芯片还支持的可运行代码的介质有：SRAM、PSRAM、ROM。四者的对比如下：

表 2-1 XR806 可运行代码介质列表

存储介质	属性	执行中断	速度	支持型号
ROM	RX	Y	快，与 CPU 同频	ALL
SRAM	RWX	Y	快，与 CPU 同频	ALL
Flash	RX	N	中，小于 96MHz，有 Cache 缓存	ALL
PSRAM	RWX	N	中，小于 120MHz，有 Cache 缓存	XR806BM2I

属性 R 表示可读，W 表示可写，X 表示可执行。

ROM 区域中存放的是固化的代码，不能修改；XIP 和 PSRAM 功能需要先初始化后才能使用，内置 Flash 或 PSRAM 的容量见《XR806_Product_Brief.pdf》。



注意

XR806 中，Flash 和 PSRAM 共用一个 Controller，在 Flash 进行擦写时，Controller 需要切换为 SBUS 模式，此时不能执行 Flash 和 PSRAM 中的代码，也不能读写 PSRAM 中的数据。

2.3 文件位置

本技术方案涉及到的主要文件位置如下。

```

├── project
│   ├── linker_script
│   │   ├── gcc
│   │   └── appos.ld
│   ├── image_cfg
│   └── image.cfg
    
```



注意

- 以 SDK 包根目录为文件树或文件路径的根目录。
- 各自工程的编译出的.o 文件以及使用的.a 文件也需要在 ld 文件中指定存放的位置，此处未列出。
- ld 和 cfg 文件使用公共文件为示例进行介绍，若各工程指定自己的 ld 和 cfg 文件，涉及文件还包括工程指定的文件。

3 应用说明

3.1 应用简述

在工程配置开启 XIP 后就会影响到 SDK 的编译、链接和系统启动流程，当通过虚拟地址访问 XIP 时，Flash Controller 会将虚拟地址翻译为在 Flash 中的实际地址，并发出读 cmd 和 address，完成数据的读取。

默认读 Flash 时，也会采用将要读的 Flash 地址映射为一段 XIP 虚拟地址的方式进行读取，最大限度的减小 Flash 切换为 SBUS 模式不可访问的时间，若要关闭此功能，需要注销掉 FLASH_XIP_OPT_READ 的宏定义。



说明

XIP 功能基于 FLASH 实现，因此需要 Flash Controller 而不能使用 SPI 访问 Flash，在 board_config.c 中 g_flash_cfg 数组定义时请选择.type = FLASH_DRV_FLASHC。

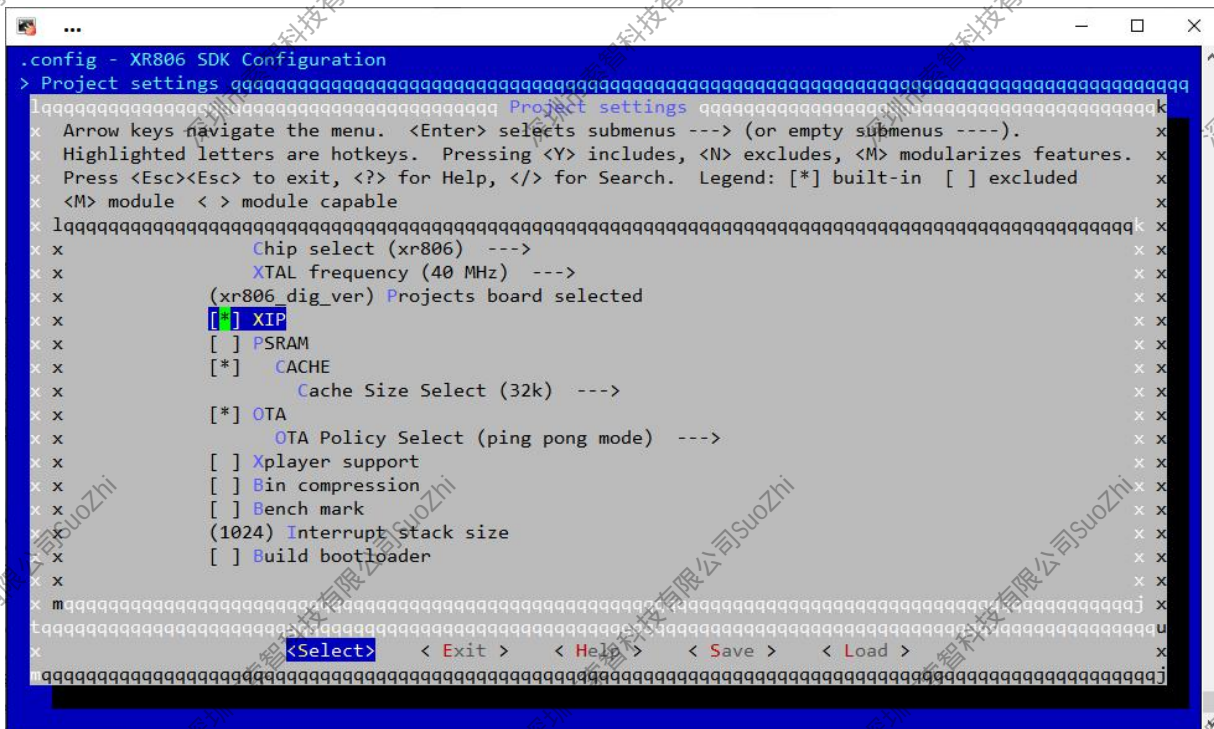
XIP 技术方案已经内嵌到 XR806 SDK 中，通过以下步骤即可启用：

1. 完成工程配置，详细可参考 3.2 节。
2. 设置相应的 Id 文件，详细可参考 3.3 节。

3.2 配置说明

打开 XIP 功能，按如下配置：执行 make menuconfig，进入 Project settings，选中 XIP，同样打开 PSRAM 功能，只需选中 PSRAM。

图 3-1 打开 XIP 功能



选中 XIP 后，即可在 XIP 中存放相应的代码；选中 PSRAM 后，即可在 PSRAM 中存放代码和数据，剩余空间还可以做 heap 使用。

3.3 链接文件格式说明

如何将函数及数据分布在 XIP 和 PSRAM，请修改工程使用的 ld 文件，参考 project\linker_script\gcc\appos.ld：

SECTIONS

```
#if (defined(CONFIG_XIP))
```

```
.xip :
```

```
{
```

```
    . = ALIGN(16);
```

```
    __xip_start__ = .;
```

```
/* MUST not put IRQ handler/callback in .xip section */
```

```
*libmp3.a: (.text .text.* .rodata .rodata.*)
```

```
*libaac.a: (.text .text.* .rodata .rodata.*)
```

```
...
```

```
*command.o: (.text .text.* .rodata .rodata.*)
```

```
*(.xip_text* .xip_rodata*)
```

```
    . = ALIGN(16);
```

```
__xip_end__ = .;
}> FLASH
#endif /* CONFIG_XIP */
```

将 libmp3.a、libaac.a 中的代码和只读数据放入 XIP 中，PSRAM 段使用也类似，不同的是 PSRAM 还可以放入 data 和 bss 段，用户可参考此写法在 XIP 和 PSRAM 加入自己的代码和数据。使能 XIP 和 PSRAM 时，会产生相应的 bin 文件，若采用自己的打包文件，打包时，不要将 bin 文件漏打包，可参考 project\image_cfg\image.cfg。



注意

不可以将中断中访问的资源放在 XIP 或 PSRAM 中，包括：中断处理代码、中断处理函数中调用的函数、中断回调函数、以及访问到的字符串常量、data、bss 等数据。否则在 Flash 擦写期间，XIP 和 PSRAM 不能访问，此时若发生中断，将造成系统挂死。另外在 XIP 和 PSRAM 未初始化完成时，也不能访问，在此之前的运行代码和访问的数据等要在 ROM 或 SRAM 中。

将中断回调函数放入 SRAM 的方式是在函数前加“__sram_test”，示例如下：

```
__sram_test
void *dma_malloc( size_t size , uint32_t flag )
{
    return malloc(size);
}
```

3.4 地址确认

当开启 XIP 后添加了新函数，可以通过 map 文件或者 objdump 文件来确认是否达到预期效果：

- 查看 map 文件：在 out 目录下会在编译的时候自动生成 xxx.map 文件，可以查看目标函数/变量保存的区域。
- 查看 objdump 文件：使用 make objdump，在 out 目录下，产生 xxx.objdump 文件，找到对应的函数/变量的地址，判断是否在指定的链接脚本的 XIP 或 PSRAM 区域中。

图 3-2 确认打开 XIP 功能

Sections:						
Idx	Name	Size	VMA	LMA	File off	Algn
0	.xip	00089ff0	00400000	00400000	00010000	2**4
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
1	.text	00003a20	00201000	00201000	00001000	2**3
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
2	.ARM.exidx	00000008	00204a20	00204a20	00004a20	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
3	.data	00000e54	00204a28	00204a28	00004a28	2**3
	CONTENTS, ALLOC, LOAD, DATA					
4	.bss	00012bf8	00205880	00205880	0000587c	2**3
	ALLOC					

xip 段地址在 0x400000 开始，当前 xip 段的大小为 0x89ff0。

图 3-3 确认函数在 XIP 中

```
int snd_pcm_read(Snd_Card_Num card_num, void *data, uint32_t count)
{
    4000d4: e92d 4ff8 stmdb sp!, {r3, r4, r5, r6, r7, r8, r9, sl,
    fp, lr}
    4000d8: 4682 mov sl, r0
    4000da: 4614 mov r4, r2
    struct cap_priv *cpriv;
    struct pcm_priv *audio_pcm_priv;
    uint32_t half_buf_size, read_remain, hw_read;

    /* Check parms to be valid */
    if(!data || !count){
```

此函数的地址在 0x4000d4，属于 XIP 段。

4 常见问题

4.1 编译报内存不足错误

当加入一个库文件或加入太多代码，而又没放入到 XIP 时，会编译报内存不足的错误：...

arm-none-eabi/bin/ld: region `RAM' overflowed by 321120 bytes。此时要参考使用说明章节，把新加入的代码放入 XIP 或 PSRAM 中（若开启 PSRAM 功能，也可以将数据放入 PSRAM 段中）。

4.2 开机异常

当没有开启 xip 的时候可以正常启动，而开启 xip 后启动失败，参考解决步骤有以下：

- 检查 Flash 线数配置是否正确：当使用 2 线正常，而 4 线异常时，检查硬件，确保原理图上 Flash 用的是 4 线，检查使用的 Flash 的 Datasheet，确保 Flash 支持 4 线模式；
- 检查 Flash 配置频率：当使用 48M 正常而使用 64M 或 96M 的时候会出现异常，可能是板子布线走线长度不一致或太长导致，也可能是 Flash 不支持高频率导致。
- 检查 image：在 image.cfg 里面有 app_xip.bin 的存在，如果没有请参考 project\image_cfg\image.cfg 来设计 image 的配置文件，确保 Flash 中已经存放了 xip 或 psram 段。
- 检查是否在 XIP 初始化之前使用的函数或变量是否在 XIP 中：添加的 xip 的函数/变量必须在 platform_xip_init()后使用，添加的 psram 的函数/变量必须在 platform_init_level0()后调用。

4.3 新加入 XIP 或 PSRAM 后运行挂死

未改动前可长时间运行，新加入后，长时间运行系统挂死，无任何打印。可能是中断代码和中断中访问的数据放入了 XIP 或 PSRAM 中导致的。检查新加入的代码是否有中断回调函数，中断回调函数是否放入了 XIP 或 PSRAM 中，以及中断回调函数中访问的数据是否在 PSRAM 中，例如 DMA 回调函数，GPIO 中断回调函数等。

4.4 性能偏低

使用 XIP 后，性能不达预期，CPU 占用率过高，可能是由于 XIP 中代码过大，Cache 效率偏低导致，可尝试以下几方面进行优化：

- 增大 Cache size：按如下配置，执行 make menuconfig，进入 Project settings，选中 CACHE，在 Cache Size Select 选择 32K（只支持 8/16/32K，Size 选择越大，Cache 性能越高）；
- 提高 Flash 的访问速率：修改对应工程下面的 board_config.c 文件中的 g_flash_cfg.clk，一般为 24M/48M/64M/96M，在能稳定运行的范围内尽量选择高频率。
- 提高 Flash 的线数，修改对应工程下面的 board_config.c 文件中的 g_flash_cfg.mode，一般为 FLASH_READ_DUAL_O_MODE 或 FLASH_READ_QUAD_O_MODE，注意：若 Flash 与 SWD 复用了数据线，则需要关闭 SWD 功能。

- 当提高了 xip 的性能后都无法解决，可能是新模块/函数有性能要求，尝试把新添加的模块/函数/变量中的关键函数指定为 nonxip，可能会有一定的改善。

附录 A： 术语表

表 A-1 术语表

S		
SiP	System-in-Package	系统单封装
X		
XIP	eXecute In Place	在 Flash 中执行代码

著作权声明

版权所有©2020 广州芯之联科技有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由广州芯之联科技有限公司（“芯之联”）拥有并保留一切权利。

本文档是芯之联的原创作品和版权财产，未经芯之联书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明



（不完全列举）均为广州芯之联科技有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与广州芯之联科技有限公司（“芯之联”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，芯之联概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。芯之联尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，芯之联概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予芯之联的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。芯之联不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。芯之联不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。