XRADIO XIP使用指导

Revision 1.1

Aug 12, 2020

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Data	Summary of Changes
1.0	2019-9-12	创建
1.1	2020-8-12	修改 XIP 初始化流程说明

Table 1- 1 Revision History

Contents

1 介绍

目前 SDK 中支持的存储类型主要有四种类型：SRAM、PSRAM、ROM、XIP，可以分为两大类型，一种是 RWX 型即可读可写可执行，另外一种是 RX 型即可读可执行。

RWX 类型：SRAM、PSRAM

RX 类型：ROM、XIP

SRAM 和 ROM 都是系统一上电就可以使用的。RWX 型的 PSRAM 相对于 SRAM 使用而已，只要在 PSRAM 驱动初始化完成后，就可以跟 SRAM 一样使用。目前 SDK 支持的 PSRAM 是 32Mb，在性能上会跟 SRAM 有所不同。RX 型的 XIP 相对于 ROM 使用而言，只有在 XIP 驱动初始化完成后，XIP 才可以使用，在性能上也会比 ROM 会慢许多而且不可以存放和执行中断相关的代码和数据。四者的对比如下：

	属性	上电即用	执行中断	可配缓存	速度	支持型号
SRAM	RWX	Y	Y	N	快	ALL 系列
PSRAM	RWX	N	Y	Y	中	XR872_AT
ROM	RX	Y	Y	N	快	XR872 系列、XR808 系列
XIP	RX	N	N	Y	慢	ALL 系列

2 项目使用

2.1 工程配置

项目开启 XIP 功能使用主要有可能会修改或关心三个文件，分别是 localconfig.mk、appos.ld 和 image.cfg。第一个是 mk 文件是确定是否开启 XIP 的功能，第二个是 ld 文件是确定在编译链接的时候 XIP 的内容分布，第三个是 cfg 文件是确定 XIP 的 bin 文件会被初始化到 XIP 空间位置。

文件	配置	说明
localconfig.mk	__CONFIG_XIP	文件在工程 gcc 目录下，修改里面开启__CONFIG_XIP ：= y
project.ld	MEMORY、section	参考 linker_script/appos.ld 文件，添加 XIP 对应的 memory 和 section 两个字段，section 里面有 text/data/bss。XIP 初始化完成前和中断调用的函数相关的代码和数据不许放到 XIP 中。
image.cfg	section	参考 image_cfg/image.cfg 文件，添加 XIP 对应的 section 段，该字段内容一般不会被修改

2.2 初始化流程

XIP 的技术是基于 FLASH 特性而设计的，所有很多操作都是基于 Flash 驱动支持之上。当在工程配置开启 XIP 后就会影响到编译、链接和系统启动流程，在编译的时候会把 XIP 相关的文件都进行编译；如果开启了 XIP 属性的定义，在链接的过程中会按照属性类型链接到相关的段中；在系统初始化中会进行 XIP 的相关初始化，XIP 整个初始化的流程主要分为三大步骤：Flash_CTRL 驱动初始化、Flash_CHIP 初始化和 XIP 内容初始化。

FLASH_CTRL 驱动初始化：该步骤主要是负责对硬件进行初始化，例如供电配置、时钟的配置等。

FLASH_CHIP 初始化：该步骤主要是多 flash memory 进行初始化，例如运行速率、线数和地址模式等。

XIP 内容初始化：该步骤主要是对 flash memory 到 XIP 地址的映射配置。

2.3 性能提高

由于物理特性，其本质上是 Norflash，执行代码和读数据上的速度还是会远远低于 SRAM 的速度，但是只要 FLAH 中的数据在 Cache 有缓存，那就可以不访问 Norflash 直接从 Cache 中返回数据。

优化点一：Icache 和 Dcahe 的大小配置

在 Cache 饱和工作下，提高 Cache 的大小是会提高 PSRAM 的性能的；但是 Cache 会分为 Dcache 和 Icache，注意分配 Dcache 和 Icache 配置，到达一个性能最优的配置。

优化点二：提高 Flash 的频率

在 SDK 中默认是使用 48Mhz 的频率的，可以尝试提高到 64Mhz 和 96Mhz。

优化点三：提高 Flash 的线数

在 SDK 中默认是使用 2 线的，可以修改为 4 线模式。

优化点四：LD 文件的排布提高 Cache 命中率

尽量把密切相关的代码和数据在链接文件排布放在一起，减少 Cache 内容被替换。

3 XIP 常见问题和调试

3.1 链接相关

控制链接过程有两个方式，第一种是修改链接脚本，第二种是修改符号链接属性。

3.1.1 链接脚本

链接脚本可以参考 linker_script/appos.ld 文件，普通的符号段属性主要是 data、BSS 或 COMMON、text。根据符号属性添加到对应的 XIP 段中，XIP 段有 xip_text,对于可执行代码或者只读数据都可以直接放这里 。书写的格式主要有：

a. 直接指定.o 文件：*hal_xxx.o (.text *.text.* .rodata *.rodata.*)

b. 指定某个.a 文件下的.o 文件可以加上其.a 的名字：*libxxx.a:hal_xxx.o (.text *.text.* .rodata *.rodata.*)

3.1.2 链接属性

符号链接属性可以在符号定义的时候进行确定，SDK 中自定义 XIP 的符号属性有__xip_text、__xip_rodata 四种。__xip_text、__xip_rodata 最后链接到 xip_text 段中，__noxip_text、__noxip_rodata 就肯定不会链接到 XIP 里面，例如一些跟中断相关的就肯定不能放这里。

3.2 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
1：查看 map 文件，在工程的 gcc 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

2：查看 objdump 文件，在工程的 gcc 目录下，使用 make objdump 产生 xxx.dump 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 FLASH 区域中。

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .xip          0001e438  00400000  00400000  00010000  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .psram_text   00000000  01400000  01400000  0003027c  2**0
                  CONTENTS
  2 .psram_data   0000027c  01400000  01400000  00030000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .psram_bss    0000264c  0140027c  0140027c  0003027c  2**2
                  ALLOC
  4 .text         00008784  00201000  00201000  00001000  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  5 .ARM.exidx    00000008  00209784  00209784  00009784  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .data         00000220  0020978c  0020978c  0000978c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  7 .bss          00000930  002099ac  002099ac  000099ac  2**2
                  ALLOC
  8 .ram_table    00000400  0020a2dc  002099ac  0000a2dc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  9 .ARM.attributes 0000002e  00000000  00000000  0003027c  2**0
                  CONTENTS, READONLY
 10 .comment      00000070  00000000  00000000  000302aa  2**0
                  CONTENTS, READONLY
```

3.3 开机失败

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：

1. 确定 Flash 线数，当使用 4 线 64M 和 4 线 96M 的时候会出现 flash 异常，证明该 flash 型号不支持或者板子布线不对齐。

2. 先确保开启 xip 后，在 image.cfg 里面有 app_xip.bin 的存在，如果没有请参考 image_xip.cfg 来设计 image 的配置文件，确保已经存放了 xip 段到 flash 中。

3. 确保新添加的函数/变量是否保存到 xip 中或者非 xip 中。

4. 新添加到 xip 的函数/变量必须在 platform_xip_init()调用后使用。

3.4 卡顿现象

当没有开启 xip 的时候可以正常使用，而开启 xip 后出现卡顿，应该是出现 xip 性能问题，参考解决步骤有以下：

1. 提高 xip 的线数，需要关闭 SWD 功能，修改对应工程下面的 board_cfg.c 文件中的 g_flash_cfg.mode，一般为 FLASH_READ_DUAL_O_MODE / FLASH_READ_QUAD_O_MODE。

2. 提高 xip 的操作速率，修改对应工程下面的 board_cfg.c 文件中的 g_flash_cfg.clk，一般为 24M/48M/64M/96M，需要注意的是当使用 4 线 64M 和 4 线 96M 的时候会出现 flash 异常，证明该 flash 型号不支持或者板子布线不对齐。

3. 提高 cache 的大小配置

4. 当提高了 xip 的性能后都无法解决卡顿的问题，那么应该是新模块/函数有性能要求，尝试把新添加的模

块/函数/变量指定为 nonxip 的看是否会有改善。