XRADIO PSRAM使用指导

Revision 1.0

Sept 11, 2019

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	2019-9-11	创建

Table 1- 1 Revision History

Contents

1 简介

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 PSRAM 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 PSRAM 的功能，第二个是 ld 文件是确定在编译链接的时候 PSRAM 的内容分布，第三个是 cfg 文件是确定 PSRAM 的 bin 文件会被初始化到 PSRAM 空间位置。

文件	配置	说明
localconfig.mk	__CONFIG_PSRAM	文件在工程 gcc 目录下，修改里面开启__CONFIG_PSRAM ：= y
project.ld	MEMORY、section	参考 linker_script/appos.ld 文件，添加 PSRAM 对应的 memory 和 section 两个字段，section 里面有 text/data/bss。对于应用而言，除了是中断调用的函数和对性能有高要求的代码外，其他所有的代码都是应该放到 PSRAM 中。
image.cfg	section	参考 image_cfg/image.cfg 文件，添加 PSRAM 对应的 section 段，该字段内容一般不会被修改

2.2 初始化流程

当在工程配置开启 PSRAM 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 PSRAM 相关的文件都进行编译；如果开启了 PSRAM 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 PSRAM 的相关初始化，PSRAM 整个初始化的流程主要分为三大步骤：PSRAM_CTRL 驱动初始化、PSRAM_CHIP 初始化和 PSRAM 内容初始化。

PSRAM_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。

PSRAM_CHIP 初始化：该步骤主要是负责对 PSRAM 类型的适配和 PSRAM 内部寄存器配置初始化等。

PSRAM 内容初始化：该步骤主要是 PSRAM 的内容进行初始化，把 app_psram.bin 拷贝到指定的位置，然后进行 BSS 段的清零初始化和 Cache 的相关配置。

2.3 动态分配

PSRAM 动态分配调用在使用中的操作必须在 platform_init_level0()初始化完后和需要包含头文件 psram.h，其路径为 "driver/chip/psram/psram.h"。

- 分配 PSRAM 内存：调用 void *psram_malloc(size_t xWantedSize);
- 释放 PSRAM 内存：调用 void psram_free(void *pv);
- 追加 PSRAM 内存：调用 void *psram_realloc(void *pv, size_t xWantedSize);
- 申请后并清零 PSRAM 内存：调用 void *psram_calloc(size_t xNmemb, size_t xMembSize);

上面的 4 和函数接口都是跟 GNU 的接口定义是一模一样的，使用可以参考 GNU。

目前 SDK 部分模块已经实现宏开关控制模块内部的动态内存分配是使用 SRAM 还是 PSRAM，有以下的模块宏开关，在 config.mk 文件中定义：

__CONFIG_MBUF_HEAP_MODE ?= 1
__CONFIG_MBEDTLS_HEAP_MODE ?= 1
__CONFIG_HTTPC_HEAP_MODE ?= 1
__CONFIG_MQTT_HEAP_MODE ?= 1
__CONFIG_NOPOLL_HEAP_MODE ?= 1
__CONFIG_WPA_HEAP_MODE ?= 1
__CONFIG_UMAC_HEAP_MODE ?= 1
__CONFIG_LMAC_HEAP_MODE ?= 1
__CONFIG_CEDARX_HEAP_MODE ?= 1
__CONFIG_AUDIO_HEAP_MODE ?= 1
__CONFIG_CODEC_HEAP_MODE ?= 1

以上的宏是当 PSRAM 打开的条件下，默认打开的。

3 相关问题

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 PSRAM 段中，PSRAM 段有 psram_data、psram_text、psram_bss 。书写的格式主要有：

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

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.*)

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 PSRAM 的符号属性有__psram_text、__psram_rodata、__psram_data、__psram_bss 四种。其中__psram_text、__psram_rodata 最后链接到 psram_text 段中，__psram_data 会链接到 psram_data 段中，__psram_bss 会链接到 psram_bss 段中。

3.2 地址相关

3.2.1 静态地址

在 LD 文件中链接过程就可以确认部分在 PSRAM，例如放在 PSRAM 中的代码段/只读数据段/bss 段大小都是可以确定的。

第一种方式：通过反汇编查看。

a.工程 gcc 目录下：make objdump， 生成 objdump 文件，打开 objdump 文件

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .xip          000b7464  00400000  00400000  00030000  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .psram_text   000000c0  01400000  01400000  000e8000  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .psram_data   000ceb10  014000c0  014000c0  000e80c0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .psram_bss    00000000  014cebd0  014cebd0  001b6bd0  2**0
                  CONTENTS
```

第二种方式：通过 image 中 app_psram.bin 的大小是否为 0，大概可以知道 PSRAM 中是否存在数据，如果是非零证明已经按照 ld 形式链接。

3.2.2 动态地址

调用 psram_malloc 内存后，可以通过打印返回值看申请到的内存是否为 PSRAM 的内存地址空间。

3.3 Cache 配置

PSRAM 搭配 cache 的使用， 开 PSRAM 默认 cache 配置是 32KB icache + 8KB dcache， 添加或修改工程 gcc/localconfig.mk

a.提高 PSRAM data 的随机操作性能--->提高 dcache 的配置， export __CONFIG_CACHE_POLICY := 0x14

b.提高 PSRAM text 的执行性能--->提高 icache 的配置，export __CONFIG_CACHE_POLICY := 0x41

c.均匀 PSRAM data 的随机操作性能和 text 的执行性能，export __CONFIG_CACHE_POLICY := 0x22