Linux SPL-PUB
开发指南

版本号: 1.1
发布日期: 2024.8.31

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.11.19	AWA1835	添加初始化版本
1.1	2024.8.31	AWA2256	修正 normal boot0、fes boot0 概述及 normal boot0 流程

目　录

1 前言

1.1 编写目的

介绍 spl-pub 中开源部分接口的介绍，为二次开发提供基础。

1.2 适用范围

brandy-2.0 平台

1.3 相关人员

spl-pub 驱动的维护、应用开发人员等。

1.4 术语、定义、缩略语

2 编译方法介绍

2.1 快速编译 spl-pub

在 longan/brandy/brandy-2.0/目录下，执行./build.sh -p 平台名称。可以快速完成整个 boot 编译动作。这个平台名称是指，sun50iw10p1/sun8iw18p1/等等，下面以 {CHIP} 代替平台。

```
./build.sh -o spl-pub -p {CHIP} //快速编译spl-pub
```

2.2 编译 boot0/fes/sboot

cd longan/brandy/brandy-2.0/spl-pub 进入 spl-pub 目录，需设置平台和要编译的模块参数。

以 {CHIP}=sun50iw10p1 为例，编译 nand/emmc 的方法如下：

1）编译 boot0

```
make distclean
make p={CHIP}
make boot0
```

2）编译 fes

```
make distclean
make p={CHIP}
make fes
```

3）编译 sboot

```
make distclean
make p={CHIP}
make sboot
```

3 spl-pub 描述

spl-pub 是 allwinner 开源的 boot0 代码。其中包含 normal boot0，secure boot，fes boot0。

3.1 normal boot0 概述

- normal boot0 简称 nboot，是非安全状态下运行的 boot0 代码。
- sunxi 的 normal boot0 是主要做的任务如下：
  1. 初始化 cpu，外设时钟。
  2. 设定 cpu，sys，dram 的初始电压。
  3. 初始化 uart，dram，flash。
  3. 加载 img 到 dram，并跳转到下一个运行环境。

3.2 Secure boot0 概述

- Secure boot0 简称 sboot，是安全状态下运行的 boot0 代码。
- sunxi 的 Secure boot0 是主要做的任务如下：
  1. 初始化 cpu，外设时钟。
  2. 设定 cpu，sys，dram 的初始电压。
  3. 初始化 uart，dram，flash。
  4. 加载 img 到 dram，并按配置对 img 验签，防篡改。
  5. 判断版本，防回滚。
  6. 验签有效后，跳转到下一个运行环境。

3.3 fes boot0 概述

- fes boot0 简称 fes，是烧录模式下使用的 boot0 代码。
- sunxi 的 fes boot0 是主要做的任务如下：
  1. 初始化 cpu，外设时钟。
  2. 设定 cpu，sys，dram 的初始电压。
  3. 初始化 uart，dram，flash。
  4. 跳转回 brom 执行 fel。

4 spl-pub 目录结构

.
├── arch
├── board
├── fes
├── include
├── Makefile
├── mk
├── nboot
└── sboot

4.1 arch

放置架构相关的文件。

4.2 board

放置板级相关的文件，不开源部分的库文件。

4.3 fes

放置烧录用 boot0 源码。

4.4 include

放置相关头文件。

4.5 Makefile

顶层 Makefile

4.6 mk

放置 Makefile 配置文件。

4.7 nboot

放置 normal boot0 源码。

4.8 sboot

放置 sboot boot0 源码。

5 spl-pub 流程介绍

- 本章只介绍函数调用链，函数具体函数可以看函数接口介绍。

5.1 normal boot0 流程

5.1.1 a100,a133,b810,t509 主流程调用链：

——>sunxi_serial_init
——>print_commit_log
    ——>sunxi_set_printf_debug_mode
——>sunxi_board_init
    ——>sunxi_board_pll_init
    ——>axp_init
    ——>set_pll_voltage
    ——>set_sys_voltage_ext
    ——>sunxi_dram_handle
——>rtc_probe_fel_flag
——>check_update_key
——>init_DRAM
——>get_uart_input
——>mmu_enable
——>malloc_init
——>sunxi_board_late_init
    ——>sunxi_nsi_init
——>load_package
——>load_image
——>update_uboot_info
——>mmu_disable
——>boot0_jmp_xxx

5.2 Secure boot0 流程

5.2.1 a100,a133,b810,t509 主流程调用链：

——>sunxi_serial_init
——>print_commit_log
——>sunxi_set_printf_debug_mode
——>sunxi_board_init
——>sunxi_board_pll_init
——>axp_init
——>set_pll_voltage
——>set_sys_voltage_ext
——>sunxi_dram_handle
——>rtc_probe_fel_flag
——>check_update_key
——>toc0_config->enable_jtag
——>boot_set_gpio
——>sid_disable_jtag
——>init_DRAM
——>get_uart_input
——>mmu_enable
——>malloc_init
——>sunxi_board_late_init
——>sunxi_nsi_init
——>sunxi_key_provision
——>sunxi_flash_init
——>toc1_init
——>toc1_verify_and_run

5.3 fes boot0 流程

5.3.1 a100,a133,b810,t509 主流程调用链：

——>sunxi_serial_init
——>sunxi_board_init
——>sunxi_board_pll_init
——>axp_init
——>set_pll_voltage
——>set_sys_voltage_ext

——>sunxi_dram_handle
——>init_DRAM
——>note_dram_log
——>return dram_size

6 增加编译 xxx.c 的方法

6.1 依赖原有 Makefile 进行编译的办法

- 可以使用如：nboot/main/路径下的 Makefile 将自己的 xxx.c 添加进来

```
machine:longan/brandy/brandy-2.0/spl-pub$git diff
diff --git a/nboot/main/Makefile b/nboot/main/Makefile
index 7114f22..1dbeaf2 100644
--- a/nboot/main/Makefile
+++ b/nboot/main/Makefile
@@ -6,6 +6,8 @@ LIB    := $(obj)libmain.o
 MAIN  += boot0_main.o
 HEAD  += boot0_head.o

+COBJS-y += xxx.c
+
 SRCS  := $(MAIN:.o=.c) $(COBJS:.o=.c) $(HEAD:.o=.c)
 OBJS  := $(addprefix $(obj),$(COBJS) $(COBJS-y) $(SOBJS))
 HEAD  := $(addprefix $(obj),$(HEAD))
machine:longan/brandy/brandy-2.0/spl-pub$
```

6.2 新建新的 Makefile 进行编译的办法

- 此 Makefile 必须保证所在目录只有一个 Makefile 文件。
- 必须以一定的格式进行组织。规则如下：

```
include $(TOPDIR)/mk/config.mk  #引用必要的配置文件
...
LIB    := libxxx.o    #最终生成的静态库文件
...
COBJS-y += xxx.o      #编译跟Makefile同级的.c文件
...
LIBS-y += xxx/child_libxxx.o      #编译xxx目录下的child_libxxx.o静态库文件,xxx为Makefile的子目录
...
LIBS := $(addprefix $(TOPDIR)/,$(sort $(LIBS-y)))  #给LIBS-y增加路径$(TOPDIR)/
SRCS   := $(COBJS-y:.o=.c)          #将COBJS-y中的.o替换为.c
OBJS   := $(addprefix $(obj),$(COBJS-y))     #给COBJS-y增加$$(obj)的前缀
OBJS   += $(LIBS)              #将LIBS加入OBJS
all:   $(obj).depend $(LIB)            #默认的编译规则和依赖

$(LIB): $(OBJS)
   $(call cmd_link_o_target, $(OBJS))    #生成具体的LIB

$(LIBS): depend
```

```
    $(MAKE) -C $(dir $(subst $(OBJTREE),,$@))  #生成子目录下的LIB
########################################################################

depend: $(obj).depend              #将depend声明为伪指令
.PHONY: depend

# defines $(obj).depend target
include $(TOPDIR)/mk/rules.mk  #引用必要的规则文件
```

- 将新建 Makefile 嵌套到上级 Makefile。上级 Makefile 分别有 nboot，sboot，fes 下的，请根据需求增加到对应的 Makefile，比如只需要在烧录时使用则只需要添加到 fes 的 Makefile 即可。
- 例如生成的静态库为 libmytest.o，路径为 spl-pub/mytest，只需要在 nboot 生效，则添加的方法如下：

```
AwExdroid68:longan/brandy/brandy-2.0/spl-pub$ git diff
diff --git a/nboot/Makefile b/nboot/Makefile
index 3a3ef07..94b4cfb 100644
--- a/nboot/Makefile
+++ b/nboot/Makefile
@@ -61,6 +61,7 @@ obj := $(TOPDIR)/nboot/

 LIBS-y  += arch/$(ARCH)/cpu/$(CPU)/libarch.o
 LIBS-y += nboot/main/libmain.o
+LIBS-y += mytest/libmytest.o
 LIBS := $(addprefix $(TOPDIR)/,$(sort $(LIBS-y)))

 A-LIBS-$(CFG_SUNXI_SDMMC)  := $(TOPDIR)/board/$(PLATFORM)/lib$(PLATFORM)_sdcard.a
```

7 函数接口重定义的方法

有时我们需要在修改主流程/子流程，实现添加自己模块的目的，但有时候添加的模块与 allwinner 提供的模块有冲突。此时我们可以使用重定义的方式将 allwinner 的接口覆盖掉。完成自己模块的添加。

7.1 方法实现

- 只需要去对应的 common.mk 文件中添加或修改 CFG_WEAK_SYMBOL 即可，路径：board/$(platform)/common.mk, 格式如下：

```
CFG_WEAK_SYMBOL = -W sunxi_board_init\
      -W rtc_probe_fel_flag
```

- 将 sunxi_board_init 和 rtc_probe_fel_flag 定义为弱函数。

7.2 定义自己的接口

- 可以在 main.c 中添加自己定义的函数，exp:

```
int sunxi_board_init(void)
{
   printf("xxxx\n");
   return 0;
}
```

8 函数接口介绍

8.1 fes_main.c 调用类

8.1.1 void sunxi_serial_init(int uart_port, void *gpio_cfg, int gpio_max)

- 作用：初始化 uart
- 参数：
  - uart_port：uart 端口号
    - uart_port 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_port
  - gpio_cfg：gpio 配置指针
    - gpio_cfg 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_tx/rx
  - gpio_max：连续初始化 gpio 最大数量
- 返回：
  - 空
- 注意：
  - 内部接口，不建议重定义，需要更换 uart 口直接修改 sysconfig 即可。

8.1.2 int sunxi_board_init(void)

- 作用：初始化板级信息包括 clk，pmu，并设置 cpu 电，系统电。
- 参数：
  - 无
- 返回：
  - 0: success
  - other: fail
- 注意：
  - 内部接口，没有关闭或重构 pmu 需求的同学，建议不重定义该接口。
  - 需要重定义时请参看 “spl-pub 流程介绍” 章节将需要重构的地方改写即可。

8.1.3 int init_DRAM(int type, dram_para_t *buff)

- 作用：初始化 DRAM
- 参数：
  - type：ddr 类型，现已废弃
  - buff：ddr 参数，参数源头由 sysconfig 中的 dram_para 而来
- 返回：
  - other: dram 大小
  - 0: fail
- 注意：
  - 内部接口，禁止重定义。修改参数到 sysconfig->dram_para 即可。

8.1.4 static void note_dram_log(int dram_init_flag)

- 作用：更新 dram 信息，该信息将会给 DebugView 使用。
- 参数：
  - dram_init_flag：1——>success，0——>fail。
- 返回：
  - 无
- 注意：
  - 内部接口，不建议重定义。

8.2 boot0_main.c 调用类

8.2.1 void sunxi_serial_init(int uart_port, void *gpio_cfg, int gpio_max)

- 作用：初始化 uart
- 参数：
  - uart_port：uart 端口号
    - uart_port 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_port
  - gpio_cfg：gpio 配置指针
    - gpio_cfg 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_tx/rx

  - gpio_max：连续初始化 gpio 最大数量

- 返回：
  - 空
- 注意：
  - 内部接口不建议重定义，需要更换 uart 口直接修改 sysconfig 即可。

8.2.2 int sunxi_board_init(void)

- 作用：初始化板级信息包括 clk，pmu，并设置 cpu 电，系统电。
- 参数：
  - 无
- 返回：
  - 0: success
  - other: fail
- 注意：
  - 内部接口，没有关闭或重构 pmu 需求的同学，建议不重定义该接口。
  - 需要重定义时请参看 “spl-pub 流程介绍” 章节将需要重构的地方改写即可。

8.2.3 u32 rtc_probe_fel_flag(void)

- 作用：读取 rtc_buff，判断 fel_flag
- 参数：
  - 无
- 返回：
  - other: 有 fel_flag
  - 0: 无 fel_flag
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.2.4 void rtc_clear_fel_flag(void)

- 作用：清除 fel_flag
- 参数：
  - 无
- 返回：
  - 无
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.2.5 int check_update_key(u16 *key_input)

- 作用：check 组合按键，并获取按键值。
- 参数：
  - key_input：输出参数指针保存在组合键值
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，重定义可能影响组合键功能。

8.2.6 int init_DRAM(int type, dram_para_t *buff)

- 作用：初始化 DRAM
- 参数：
  - type：ddr 类型，现已废弃
  - buff：ddr 参数，参数源头由 sysconfig 中的 dram_para 而来
- 返回：
  - other: dram 大小
  - 0: fail
- 注意：
  - 内部接口，禁止重定义。修改参数到 sysconfig->dram_para 即可。

8.2.7 char get_uart_input(void)

- 作用：获取 uart 输入值
- 参数：
  - 无
- 返回：
  - 键盘按下的键值
- 注意：
  - 无

8.2.8 int sunxi_set_printf_debug_mode(u8 debug_level)

- 作用：设置 debug 打印等级
- 参数：
  - debug_level：0-> 只打印强制和错误打印，other-> 无打印限制
- 返回：
  - debug_level>8：fail
  - other：success
- 注意：
  - 无

8.2.9 __weak void mmu_enable(u32 dram_size)

- 作用：使能 mmu
- 参数：
  - dram_size：dram 的大小
- 返回：
  - 无
- 注意：
  - 无

8.2.10 uint8_t sunxi_board_late_init(void)

- 作用：处理需要依赖 dram 初始化后的板级初始化
- 参数：
  - 无
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，没有新增流程的同学，建议不重定义该接口。

8.2.11 int load_package(void)

- 作用：初始化 flash 并将 boot_package，load 到指定内存地址。
- 参数：
  - 无
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，禁止重定义，不然会影响启动。

8.2.12 int load_image(phys_addr_t uboot_base, phys_addr_t optee_base, phys_addr_t monitor_base, phys_addr_t rtos_base, phys_addr_t *opensbi_base)

- 作用：将 boot_package 解包，并搬运到各 item 的运行地址。
- 参数：
  - uboot_base：输出参数得到 uboot 基地址
  - optee_base：输出参数得到 optee 基地址
  - monitor_base：输出参数得到 monitor 基地址
  - rtos_base：输出参数得到 rtos 基地址
  - opensbi_base：输出参数得到 opensbi 基地址

- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，禁止重定义，不然会影响启动。

8.2.13 static void update_uboot_info(phys_addr_t uboot_base, phys_addr_t optee_base,

```
phys_addr_t monitor_base, phys_addr_t rtos_base, u32 dram_size,
u16 pmu_type, u16 uart_input, u16 key_input)
```

- 作用：更新一些参数，为跳转到下一运行环境做准备。
- 参数：
  - uboot_base：uboot 基地址
  - optee_base：optee 基地址
  - monitor_base：monitor 基地址
  - rtos_base：rtos 基地址
  - dram_size：dram 大小
  - pmu_type：电源芯片类型
  - uart_input：uart 输入的值
  - key_input：组合键输入的值
- 返回：
  - 无
- 注意：
  - 内部接口，禁止重定义，不然会影响启动。

8.2.14 __weak void mmu_disable(void)

- 作用：禁用 mmu
- 参数：
  - 无
- 返回：
  - 无
- 注意：
  - 无

8.2.15 boot0_jmp_xxx(phys_addr_t xxx_base)

- 作用：jump 到对应的输入的运行地址。
- 参数：
  - uboot_base：uboot 基地址
  - optee_base：optee 基地址
  - monitor_base：monitor 基地址
  - rtos_base：rtos 基地址
  - opensbi_base：opensbi 基地址
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，禁止重定义，不然会影响启动。

8.3 sboot_main.c 调用类

8.3.1 void sunxi_serial_init(int uart_port, void *gpio_cfg, int gpio_max)

- 作用：初始化 uart
- 参数：
  - uart_port：uart 端口号
    - uart_port 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_port
  - gpio_cfg：gpio 配置指针
    - gpio_cfg 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_tx/rx
  - gpio_max：连续初始化 gpio 最大数量
- 返回：
  - 空
- 注意：
  - 内部接口不建议重定义，需要更换 uart 口直接修改 sysconfig 即可。

8.3.2 static void print_commit_log(void)

- 作用：打印 commit，设置打印等级
- 参数：
  - 无
- 返回：
  - 无
- 注意：
  - 无

8.3.3 int sunxi_board_init(void)

- 作用：初始化板级信息包括 clk，pmu，并设置 cpu 电，系统电。
- 参数：
  - 无
- 返回：
  - 0: success
  - other: fail
- 注意：
  - 内部接口，没有关闭或重构 pmu 需求的同学，建议不重定义该接口。
  - 需要重定义时请参看 “spl-pub 流程介绍” 章节将需要重构的地方改写即可。

8.3.4 u32 rtc_probe_fel_flag(void)

- 作用：读取 rtc_buff，判断 fel_flag
- 参数：
  - 无
- 返回：
  - other: 有 fel_flag
  - 0: 无 fel_flag
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.3.5 void rtc_clear_fel_flag(void)

- 作用：清除 fel_flag
- 参数：
  - 无
- 返回：
  - 无
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.3.6 int check_update_key(u16 *key_input)

- 作用：check 组合按键，并获取按键值。
- 参数：
  - key_input：输出参数指针保存在组合键值
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，重定义可能影响组合键功能。

8.3.7 int boot_set_gpio(void *user_gpio_list, u32 group_count_max, int set_gpio)

- 作用：初始化 GPIO
- 参数：
  - user_gpio_list：gpio 结构体指针。
  - group_count_max：最大连续设置 gpio。
  - set_gpio：根据 gpio 结构体设置 gpio 复用。0——> 禁止，other——> 设置。
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，重定义可能影响 gpio 初始化。

8.3.8 void sid_disable_jtag(void)

- 作用：将 Efuse 中 jtag 禁止位使能
- 参数：
  - 无
- 返回：
  - 无
- 注意：
  - 内部接口，一次调用永久生效，重定义无意义。

8.3.9 int init_DRAM(int type, dram_para_t *buff)

- 作用：初始化 DRAM
- 参数：
  - type：ddr 类型，现已废弃
  - buff：ddr 参数，参数源头由 sysconfig 中的 dram_para 而来
- 返回：
  - other: dram 大小
  - 0: fail
- 注意：
  - 内部接口，禁止重定义。修改参数到 sysconfig->dram_para 即可。

8.3.10 char get_uart_input(void)

- 作用：获取 uart 输入值
- 参数：
  - 无
- 返回：
  - 键盘按下的键值
- 注意：
  - 无

8.3.11 int sunxi_set_printf_debug_mode(u8 debug_level)

- 作用：设置 debug 打印等级
- 参数：
  - debug_level：0-> 只打印强制和错误打印，other-> 无打印限制
- 返回：
  - debug_level>8：fail
  - other：success
- 注意：
  - 无

8.3.12 __weak void mmu_enable(u32 dram_size)

- 作用：使能 mmu
- 参数：
  - dram_size：dram 的大小
- 返回：
  - 无
- 注意：
  - 无

8.3.13 __s32 malloc_init(__u32 pHeapHead, __u32 nHeapSize)

- 作用：初始化 malloc 设置堆池的起始和大小
- 参数：
  - pHeapHead：堆基地址
  - nHeapSize：堆大小
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 设置的堆池区域是 dram 的一部分，必须保证在 dram_init 后在调用。

8.3.14 uint8_t sunxi_board_late_init(void)

- 作用：处理需要依赖 dram 初始化后的板级初始化
- 参数：
  - 无
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，没有新增流程的同学，建议不重定义该接口。

8.3.15 int sunxi_flash_init(int boot_type)

- 作用：初始化 flash，并将 toc1 包，load 到指定内存地址
- 参数：
  - boot_type：flash 类型，参数禁止修改，否则可能影响 flash 初始化。
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，禁止重定义。否则可能影响 flash 初始化。

8.3.16 int toc1_init(void)

- 作用：初始化 toc1 包，检查版本，防止版本回滚。
- 参数：
  - 无
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，禁止重定义。否则可能影响 toc1 包下一步的解析。

8.3.17 int toc1_verify_and_run(u32 dram_size, u16 pmu_type, u16 uart_input, u16 key_input)

- 作用：校验根证书在内的所有证书，并做验签操作，更新参数，jmp 到下一个运行环境。
- 参数：
  - dram_size：dram 大小
  - pmu_type：电源芯片类型
  - uart_input：uart 输入的值
  - key_input：组合键输入的值
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，禁止重定义。

8.4 通用接口类

8.4.1 u32 get_sys_ticks(void)

- 作用：滴答函数间隔为 1ms
- 参数：
  - 无
- 返回：
  - 返回滴答的值
- 注意：
  - 内部接口，不建议重定义。

8.4.2 __weak void udelay(unsigned long us)

- 作用：微秒延时函数
- 参数：
  - us：延时 us 微秒
- 返回：

- 无

- 注意：
  - 无

8.4.3 __weak void mdelay(unsigned long ms)

- 作用：毫秒秒延时函数
- 参数：
  - ms：延时 ms 毫秒
- 返回：
  - 无
- 注意：
  - 无

8.4.4 __s32 malloc_init(__u32 pHeapHead, __u32 nHeapSize)

- 作用：初始化 malloc 设置堆池的起始和大小
- 参数：
  - pHeapHead：堆基地址
  - nHeapSize：堆大小
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 设置的堆池区域是 dram 的一部分，必须保证在 dram_init 后在调用。

8.4.5 void *malloc(__u32 num_bytes)

- 作用：申请一块 buf
- 参数：
  - num_bytes：buf 大小
- 返回：

  - 申请 buf 首地址

- 注意：
  - 必须要在 malloc_init 被调用之后调用。

8.4.6 void free(void *p)

- 作用：释放一块 buf
- 参数：
  - p：释放 buf 的首地址
- 返回：
  - 无
- 注意：
  - 必须要在 malloc_init 被调用之后调用。

8.4.7 void ndump(u8 *buf, int count)

- 作用：打印内存块数据
- 参数：
  - buf：内存块首地址
  - count：打印大小
- 返回：
  - 无
- 注意：
  - 无

8.5 GPIO 接口类

8.5.1 int boot_set_gpio(void *user_gpio_list, u32 group_count_max, int set_gpio)

- 作用：初始化 GPIO
- 参数：

- user_gpio_list：gpio 结构体指针。
- group_count_max：最大连续设置 gpio。
- set_gpio：根据 gpio 结构体设置 gpio 复用。0——> 禁止，other——> 设置。

- 返回：
  - 0：success
  - other：fail

- 注意：
  - 内部接口，重定义可能影响 gpio 初始化。

8.5.2 void sunxi_gpio_set_cfgpin(u32 pin, u32 val);

- 作用：设置 GPIO 复用属性
- 参数：
  - pin：gpio_pin 的偏移值。
    - #define SUNXI_GPA(_nr) (SUNXI_GPIO_A_START + (_nr))
    - #define SUNXI_GPB(_nr) (SUNXI_GPIO_B_START + (_nr))
    - ……
    - #define SUNXI_GPN(_nr) (SUNXI_GPIO_N_START + (_nr))
    - _nr：gpio 号。
  - val：复用的值。
- 返回：
  - 无
- 注意：
  - exp：设置 PGIOB15 复用为输出——>sunxi_gpio_set_cfgpin(SUNXI_GPB(15), 1);

8.5.3 int sunxi_gpio_get_cfgpin(u32 pin);

- 作用：得到 GPIO 复用属性
- 参数：
  - pin：gpio_pin 的偏移值。
    - #define SUNXI_GPA(_nr) (SUNXI_GPIO_A_START + (_nr))
    - #define SUNXI_GPB(_nr) (SUNXI_GPIO_B_START + (_nr))
    - ……
    - #define SUNXI_GPN(_nr) (SUNXI_GPIO_N_START + (_nr))
    - _nr：gpio 号。

- 返回：
  - 复用的值
- 注意：
  - exp：读取 GPIOB15 的复用属性——>sunxi_gpio_get_cfgpin(SUNXI_GPB(15));

8.5.4 int sunxi_gpio_set_drv(u32 pin, u32 val);

- 作用：设置 GPIO 驱动能力
- 参数：
  - pin：gpio_pin 的偏移值。
    - #define SUNXI_GPA(_nr) (SUNXI_GPIO_A_START + (_nr))
    - #define SUNXI_GPB(_nr) (SUNXI_GPIO_B_START + (_nr))
    - ……
    - #define SUNXI_GPN(_nr) (SUNXI_GPIO_N_START + (_nr))
    - _nr：gpio 号。
  - val：驱动能力的值。
- 返回：
  - 0：success
  - other：fail
- 注意：
  - exp：设置 PGIOB15 驱动能力为 level2——>sunxi_gpio_set_drv(SUNXI_GPB(15), 2);

8.5.5 int sunxi_gpio_set_pull(u32 pin, u32 val);

- 作用：设置 GPIO 内部上下拉属性
- 参数：
  - pin：gpio_pin 的偏移值。
    - #define SUNXI_GPA(_nr) (SUNXI_GPIO_A_START + (_nr))
    - #define SUNXI_GPB(_nr) (SUNXI_GPIO_B_START + (_nr))
    - ……
    - #define SUNXI_GPN(_nr) (SUNXI_GPIO_N_START + (_nr))
    - _nr：gpio 号。
  - val：上下拉属性。
- 返回：
  - 0：success

- other：fail

- 注意：
  - exp：设置 PGIOB15 为上拉——>sunxi_gpio_set_pull(SUNXI_GPB(15), 1);

8.5.6 #define PIO_REG_DATA(n)

- 作用：获取 GPIO(n) Data Register 的值
- 参数：
  - n：gpio_port 的偏移值。
    - n = 1：GPIOA
    - n = 2：GPIOB
    - ……
    - n = 14：GPIOB
- 返回：
  - GPIO(n) data 的值
- 注意：
  - exp：获取 PGIOB，data 的值——>PIO_REG_DATA(2);

8.5.7 #define PIO_ONE_PIN_DATA(n, i)

- 作用：获取 GPIO(n)Data Register 偏移第 i 位的值
- 参数：
  - n：gpio_port 的偏移值。
    - n = 1：GPIOA
    - n = 2：GPIOB
    - ……
    - n = 14：GPIOB
  - i：gpio_num
- 返回：
  - GPIO(n)(i) data 的值
- 注意：
  - exp：获取 PGIOB15，data 的值——>PIO_ONE_PIN_DATA(2, 15);

8.6 i2c 接口类

8.6.1 void i2c_init(u32 i2c_base, int speed, int slaveaddr)

- 作用：初始化 i2c
- 参数：
  - i2c_base：i2c 基地址
  - speed：i2c 速度
  - slaveaddr：器件地址
- 返回：
  - 无
- 注意：
  - 内部接口，不建议重定义。
  - 初始化 i2c 前，先将对应的 GPIO 复用为 i2c

8.6.2 int i2c_read(u8 chip, uint addr, int alen, u8 *buffer, int len)

- 作用：i2c 读函数
- 参数：
  - chip：器件地址
  - addr：器件寄存器地址
  - alen：器件寄存器地址位宽（1->u8，2->u16，4->u32）
  - buffer：读取器件寄存器地址的值
  - len：读取器件寄存器地址的值位宽（1->u8，2->u16，4->u32）
- 返回：
  - 无
- 注意：
  - 内部接口，不建议重定义。

8.6.3 int i2c_write(u8 chip, uint addr, int alen, u8 *buffer, int len)

- 作用：i2c 写函数

- 参数：
  - chip：器件地址
  - addr：器件寄存器地址
  - alen：器件寄存器地址位宽（1->u8，2->u16，4->u32）
  - buffer：写入器件寄存器地址的值
  - len：写入器件寄存器地址的值位宽（1->u8，2->u16，4->u32）
- 返回：
  - 无
- 注意：
  - 内部接口，不建议重定义。

8.7 POWER 接口类

8.7.1 int get_power_mode(void)

- 作用：获取 axp 工作模式
- 参数：
  - 无
- 返回：
  - axp 工作模式
- 注意：
  - 内部接口，不同 axp 芯片都可以使用该接口，不建议重定义。

8.7.2 int axp_init(u8 power_mode)

- 作用：初始化 axp 电源管理芯片
- 参数：
  - power_mode：axp 工作模式，每个板级设置会有所不同具体参看板级的 sysconfig
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 使用其他的 axp 芯片，请根据实际情况重定义该接口。
  - 使用无 axp 芯片方案，可重定义为空函数。

8.7.3 int set_pll_voltage(int set_vol)

- 作用：设置 CPU 电压
- 参数：
  - set_vol：CPU 电压值
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，使用其他的 axp 芯片，请根据实际情况重定义该接口。
  - 使用无 axp 芯片方案，可重定义为空函数。

8.7.4 int set_sys_voltage(int set_vol)

- 作用：设置 SYS 电压
- 参数：
  - set_vol：SYS 电压值
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，使用其他的 axp 芯片，请根据实际情况重定义该接口。
  - 使用无 axp 芯片方案，可重定义为空函数。

8.7.5 int set_sys_voltage_ext(char *name, int set_vol)

- 作用：设置 SYS 电压
- 参数：
  - name：子电压名，可根据原理图确定 SYS 电挂载在哪路子电压下。
  - set_vol：SYS 电压值
- 返回：
  - 0：success
  - other：fail
- 注意：

- 内部接口，使用其他的 axp 芯片，请根据实际情况重定义该接口。
- 使用无 axp 芯片方案，可重定义为空函数。

8.7.6 int set_ddr_voltage(int set_vol)

- 作用：设置 ddr 电压
- 参数：
  - set_vol：ddr 电压值
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，使用其他的 axp 芯片，必须重定义该接口，即使是硬件设计好电路也必须重定义为空函数。
  - 使用无 axp 芯片方案，可重定义为空函数。

8.7.7 int probe_power_key(void)

- 作用：获取 power 键的值
- 参数：
  - 无
- 返回：
  - 松开：0
  - 按下：1
- 注意：
  - 内部接口，使用其他的 axp 芯片，且使用组合键功能，则需要重定义该接口。
  - 使用无 axp 芯片方案，可重定义为空函数。

8.7.8 int axp_reg_write(u8 addr, u8 val)

- 作用：设置 axp 寄存器
- 参数：
  - addr：寄存器地址
  - val：写入 addr 的值

- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，使用其他的 axp 芯片，请根据实际情况重定义该接口。
  - 使用无 axp 芯片方案，可重定义为空函数。

8.7.9 int axp_reg_read(u8 addr, u8 *val)

- 作用：读取 axp 寄存器
- 参数：
  - addr：寄存器地址
  - val：读取 addr 值的指针
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，使用其他的 axp 芯片，请根据实际情况重定义该接口。
  - 使用无 axp 芯片方案，可重定义为空函数。

8.8 RTC 接口类

8.8.1 void rtc_write_data(int index, u32 val)

- 作用：将值写入 rtc buff
- 参数：
  - index：rtc buff 索引，有多少个 index 请参看 spec
  - val：写入 rtc buff 的值
- 返回：
  - 无
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.8.2 u32 rtc_read_data(int index)

- 作用：根据 index 读取 rtc buff
- 参数：
  - index：rtc buff 索引，有多少个 index 请参看 spec
- 返回：
  - rtc_buff 的值
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.8.3 u32 rtc_probe_fel_flag(void)

- 作用：读取 rtc_buff，判断 fel_flag
- 参数：
  - 无
- 返回：
  - other: 有 fel_flag
  - 0: 无 fel_flag
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.8.4 void rtc_clear_fel_flag(void)

- 作用：清除 fel_flag
- 参数：
  - 无
- 返回：
  - 无
- 注意：
  - 内部接口，禁止重定义，否则可能影响 usb 烧录功能。

8.9 adc_key 接口类

8.9.1 gpadc_key 接口类

8.9.1.1 int sunxi_gpadc_init(void)

- 作用：gpadc_key 初始化
- 参数：
  - 无
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 无

8.9.1.2 int sunxi_read_gpadc(int channel)

- 作用：读取 gpadc_key 的值
- 参数：
  - channel：gpadc 通道
- 返回：
  - < 0：fail
  - other：key 值
- 注意：
  - 无

8.9.1.3 int sunxi_read_gpadc_vol(int channel)

- 作用：读取 gpadc 的电压转换值
- 参数：
  - channel：gpadc 通道
- 返回：
  - 0：success

- other：fail

- 注意：
  - 内部接口，禁止重定义，可能影响 auto_dram_para 功能。

8.9.2 lradc_key 接口类

8.9.2.1 int sunxi_key_init(void)

- 作用：lradc_key 初始化
- 参数：
  - 无
- 返回：
  - 0：success
  - other：fail
- 注意：
  - 无

8.9.2.2 int sunxi_key_read(void)

- 作用：读取 lradc_key 的值
- 参数：
  - 无
- 返回：
  - < 0：fail
  - other：key 值
- 注意：
  - 无

8.9.2.3 int check_update_key(u16 *key_input)

- 作用：check 组合按键，并获取按键值。
- 参数：
  - key_input：输出参数指针保存在组合键值

- 返回：
  - 0：success
  - other：fail
- 注意：
  - 内部接口，重定义可能影响组合键功能。

8.10 uart 接口类

8.10.1 void sunxi_serial_init(int uart_port, void *gpio_cfg, int gpio_max)

- 作用：初始化 uart
- 参数：
  - uart_port：uart 端口号
    - uart_port 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_port
  - gpio_cfg：gpio 配置指针
    - gpio_cfg 由对应板级的 sysconfig 中的 [uart_port]——>uart_debug_tx/rx
  - gpio_max：连续初始化 gpio 最大数量
- 返回：
  - 空
- 注意：
  - 内部接口不建议重定义，需要更换 uart 口直接修改 sysconfig 即可。

8.10.2 char get_uart_input(void)

- 作用：获取 uart 输入值
- 参数：
  - 无
- 返回：
  - 键盘按下的键值
- 注意：
  - 无

8.10.3 void puts(const char *s)

- 作用：输出一串字符串
- 参数：
  - 字符串 buf
- 返回：
  - 无
- 注意：
  - 无

8.10.4 int sprintf(char * buf, const char *fmt, ...)

- 作用：把格式化的数据写入 buf 缓冲区
- 参数：
  - buf：这是指向一个字符数组的指针，该数组存储了 C 字符串。
  - fmt：格式化输出字符串。
- 返回：
  - fail：负数
  - success：写入字符串总数
- 注意：
  - 无

8.10.5 int printf(const char *fmt, ...)

- 作用：把格式化的数据输出到 uart
- 参数：
  - fmt：格式化输出字符串。
- 返回：
  - fail：负数
  - success：写入字符串总数
- 注意：
  - 无

8.10.6 int sunxi_set_printf_debug_mode(u8 debug_level)

- 作用：设置 debug 打印等级
- 参数：
  - debug_level：0-> 只打印强制和错误打印，other-> 无打印限制
- 返回：
  - debug_level >8：fail
  - other：success
- 注意：
  - 无

著作权声明

商标声明

免责声明