君正®

Halley5 uboot 开发手册

Author：系统软件部

Version: 1.0
Date: 2020.12.7

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

君正®

Halley5 uboot开发手册

Copyright © Ingenic Semiconductor Co. Ltd 2020. All rights reserved.

Release history

Date	Revision	Change
2020.12.7	1.0	First release

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co., Ltd.

Ingenic Headquarters, East Bldg. 14, Courtyard #10
Xibeiwang East Road, Haidian District, Beijing, China,
Tel: 86-10-56345000
Fax:86-10-56345001
Http: //www.ingenic.com

北京君正集成电路股份有限公司

地址:北京市海淀区东北西路中关村软件园二期君正总部大楼
电话: 86-10-56345000
传真: 86-10-56345001
http: //www.ingenic.com

目 录

1 u-boot 简介 1

1.1 u-boot 功能 1
1.2 u-boot 引导流程 1
1.3 u-boot 源码目录介绍 1

2 u-boot 配置 3

2.1 boards.cfg 文件说明 3
2.1.1 默认配置 3
2.1.2 自定义配置 4
2.2 halley5.h 配置文件说明 4
2.2.1 系统时钟配置 4
2.2.2 DDR 类型配置 5
2.2.3 SFC 驱动配置 6
2.2.4 MSC 驱动配置 9
2.2.5 GMAC 网络驱动配置 11
2.2.6 u-boot built-in 命令配置 13
2.2.7 内核传参说明 14

3 u-boot 编译 15

3.1 工程顶层目录编译 15
3.1.1 编译 SFC Nand 启动镜像 15
3.1.2 编译 SFC Nor 启动镜像 15
3.1.3 编译 SD 卡启动镜像 16
3.2 u-boot 源码目录编译 16

4 u-boot 应用与内核接口 18

4.1 u-boot 常用命令 18
4.2 u-boot 内核传参说明 20
4.2.1 u-boot 加载内核挂载网络文件系统 20
4.2.2 u-boot 加载内核挂载 jffs2 文件系统 21
4.2.3 u-boot 加载内核挂载 ext4 文件系统 21
4.2.4 u-boot 加载内核挂载 ubifs 文件系统 21
4.3 u-boot 加载内核镜像 22
4.3.1 uboot 通过 tftp 加载内核镜像 22
4.3.2 uboot 通过 fastboot 命令加载内核镜像 22
4.3.3 u-boot 通过串口加载内核镜像 23

5 Ingenic tools 介绍 25

5.1 DDR 参数相关 25
5.2 MBR/GPT 分区表 25

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

5.3 nand/nor flash 相关 25
5.4 其他 26
6 参考文档 27

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

关 键 词	U-Boot 配置 编译 常用命令 启动参数
摘　　要	本文档适用于君正公司推出的 halley5 开发板，目的在于帮助开发者熟悉开发板配套的 u-boot 工程支持的配置、编译方法、常用的命令以及与 linux 内核的接口等
缩 略 语	
参考资料	

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

1 u-boot 简介

1.1 u-boot 功能

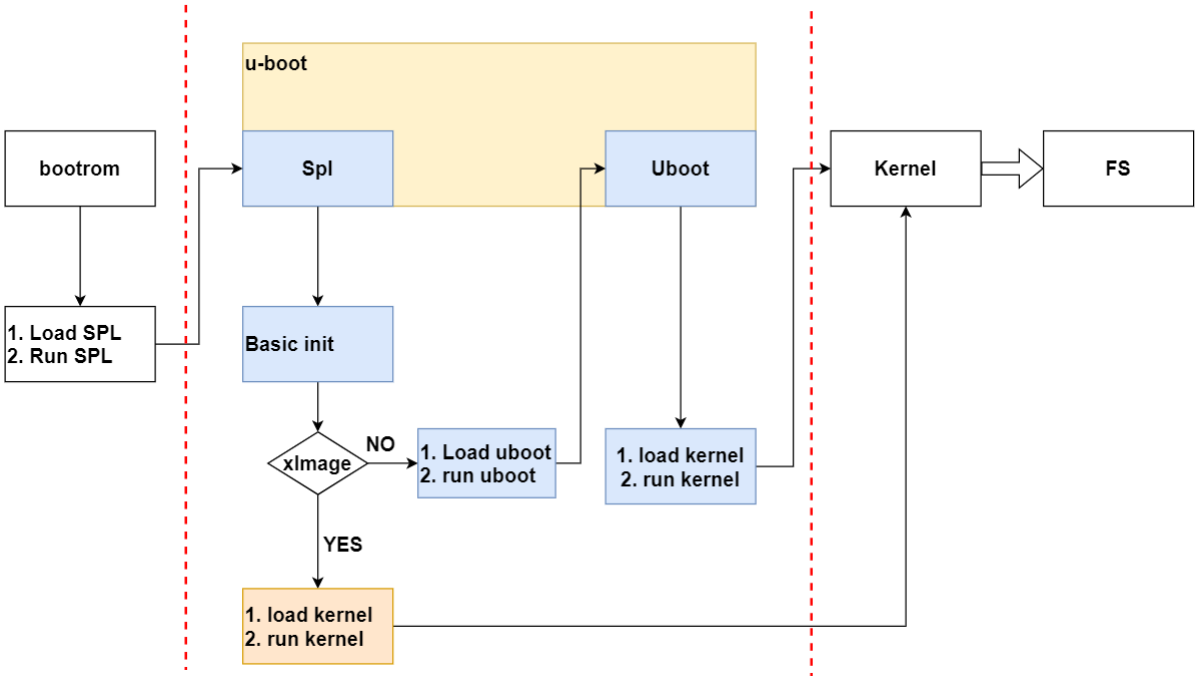
U-Boot 是为嵌入式平台提供的开放源代码的引导程序，可引导包括 linux 内核在内的多种嵌入式操作系统，支持诸如 mips、arm、powerpc 等多种处理器架构，它提供串行口、以太网等多种下载方式，提供 NOR 和 NAND 闪存和环境变量管理等功能，支持网络协议栈、JFFS2/EXT2/FAT 文件系统，同时还支持多种设备驱动如 MMC/SD 卡、USB 设备、LCD 驱动等。

注意:文中所提的 Nor 没有特殊说明，均指 SPI Nor Flash

Nand 没有特殊说明，均指 SPI Nand Flash

1.2 u-boot 引导流程

Halley5 平台 u-boot 引导流程如下图所示：



从上图可看到，分为两种引导方式：

1. bootrom ---> spl ---> u-boot ---> kernel ---> rootfs，该方式需要 u-boot 引导 kernel
2. bootrom ---> spl ---> kernel ---> rootfs，该方式不需要 u-boot，由 spl 直接引导 kernel

1.3 u-boot 源码目录介绍

api	存放 uboot 提供的接口函数
arch	体系结构相关文件
board	板级相关文件
common	通用的代码，以各种命令为主
disk	磁盘分区相关代码
doc	介绍文档

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

drivers	驱动程序相关代码
examples	示例程序
fs	文件系统代码，支持常见的嵌入式文件系统
include	头文件
lib	通用库文件
nand_spl	nand flash 启动相关代码
net	网络相关代码
post	Power On Self Test，上电自检程序
spl	spl 相关代码
test	测试代码
tools	辅助工具，用于编译和检查 uboot 目标文件

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

2 u-boot 配置

当前 Halley5 平台，决定 u-boot 配置的文件主要有：

1. u-boot/boards.cfg
2. u-boot/include/configs/halley5.h

两个文件共同作用决定最终生成的烧录镜像的配置。

2.1 boards.cfg 文件说明

boards.cfg 是 u-boot 中最基础的配置文件，用于区分不同厂家的不同板级的配置。该文件中每一行表示一个具体的板级配置，格式均相同，具体如下表所示：

Target	ARCH	CPU	Board name	Vendor	SoC	Options
目标名称	架构名称	处理器名称	板级名称	厂商名称	Soc 名称	附加选项

其中每一个配置项说明如下：

1. Target：作为该种配置的一个标志；

2. ARCH、CPU、Board name、Vendor、SoC：主要用于 Makefile 编译时选择代码目录，比如 u-boot/include/configs 目录下有许多配置文件，选择 halley5.h 便是通过 Board name 确定的；

3. Options：每一种配置之间的主要区别，u-boot 会为附加选项中的每个字段前面加上“CONFIG_”前缀并导出为全局可用的宏定义，这些宏定义会进一步决定其他的配置，目前主要是 halley5.h 中的配置。

2.1.1 默认配置

Halley5 共支持四种默认编译配置，如下表所示：

halley5_uImage_msc2	**该配置支持从 sd 卡启动 linux 内核镜像 uImage** CONFIG_SPL_JZMMC_SUPPORT：spl 中的 mmc 相关配置 CONFIG_ENV_IS_IN_MMC：环境变量存储相关配置 CONFIG_GPT_CREATOR：分区相关配置 CONFIG_JZ_MMC_MSC2：msc 控制器相关配置
halley5_uImage_sfc_nand	**该配置支持从 nand flash 启动 linux 内核镜像 uImage** CONFIG_SPL_SFC_NAND：sfc nand flash 相关配置 CONFIG_MTD_SFCNAND：sfc nand flash 驱动编译配置
halley5_uImage_sfc_nor	**该配置支持从 nor flash 启动 linux 内核镜像 uImage** CONFIG_SPL_SFC_NOR：sfc nor flash 相关配置 CONFIG_ENV_IS_IN_SFC：环境变量存储相关配置 CONFIG_MTD_SFCNOR：sfc nor flash 驱动编译相关配置
halley5_xImage_sfc_nand	**该配置支持从 nand flash 启动 linux 内核镜像 xImage** CONFIG_SPL_SFC_NAND：sfc nand flash 相关配置 CONFIG_MTD_SFCNAND：sfc nand flash 驱动编译相关配置 CONFIG_SPL_OS_BOOT：spl 直接启动 linux 内核镜像相关配置 CONFIG_OTA_VERSION30：ota 升级相关配置

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

2.1.2 自定义配置

可根据实际需要，在每种板级配置中增删相应的字段或者新添加一个新的板级配置。

下面以添加一个新的板级配置 halley5_uImage_msc0 为例进行介绍：

1. 在 boards.cfg 文件中新起一行，按照该文件的格式要求，先确定第一个配置项为 halley5_uImage_msc0，第二到第六个配置项可参考现存的板级配置，一般都是比较固定的；

2. 主要是附加选项的配置，如果对 Halley5 平台的 u-boot 整体配置不是很了解的话，可以先参考现存配置，比如这里可以参考 halley5_uImage_msc2，将 JZ_MMC_MSC2 改为 JZ_MMC_MSC0，这样会选择 MSC0 相关的配置，其他的保持一致即可；

3. 如果想更加细致的修改附加选项，建议先整体阅读一遍该文档，尤其是第二章关于配置的介绍，在了解了 Halley5 平台的 u-boot 整体配置后，再进行修改会更加游刃有余。

2.2 halley5.h 配置文件说明

halley5.h 为 u-boot 中针对 halley5 平台核心的配置文件，该配置文件主要包含以下内容：

- 系统时钟配置
- Cache 大小配置
- DDR 配置
- SFC/MSC/GMAC/LCD 等驱动相关配置
- u-boot built-in 命令

2.2.1 系统时钟配置

2.2.1.1 默认配置

当前系统时钟配置如下：

```
#define CONFIG_SYS_APLL_FREQ 1200000000 /*If APLL not use mast be set 0*/
#define CONFIG_SYS_MPLL_FREQ 1500000000 /*If MPLL not use mast be set 0*/
#define CONFIG_SYS_EPLL_FREQ 300000000 /*If MPLL not use mast be set 0*/
#define CONFIG_CPU_SEL_PLL APLL
#define CONFIG_DDR_SEL_PLL MPLL
#define CONFIG_SYS_CPU_FREQ 1200000000
#define CONFIG_SYS_MEM_FREQ 750000000

#define CONFIG_SYS_AHB0_FREQ 300000000
#define CONFIG_SYS_AHB2_FREQ 300000000 /*APB = AHB2/2*/
```

2.2.1.2 自定义配置

1. PLL 锁相环频率配置，可根据需要进行配置

```
#define CONFIG_SYS_APLL_FREQ 1200000000 /*If APLL not use mast be set 0*/
#define CONFIG_SYS_MPLL_FREQ 1500000000 /*If MPLL not use mast be set 0*/
#define CONFIG_SYS_EPLL_FREQ 300000000 /*If MPLL not use mast be set 0*/
```

2. CPU、DDR 的 PLL 选择，建议使用默认配置

```
#define CONFIG_CPU_SEL_PLL APLL
#define CONFIG_DDR_SEL_PLL MPLL
```

3. CPU、DDR 频率配置，其中 CPU 的频率需要与选择的 PLL 频率成倍数关系，并且最高支持

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

1.5GHz，同时 DDR 的频率也需要与选择的 PLL 频率成倍数关系

```
#define CONFIG_SYS_CPU_FREQ 1200000000
#define CONFIG_SYS_MEM_FREQ 750000000
```

4. AHB 总线频率配置

```
#define CONFIG_SYS_AHB0_FREQ 300000000
#define CONFIG_SYS_AHB2_FREQ 300000000 /*APB = AHB2/2*/
```

2.2.2 DDR 类型配置

2.2.2.1 默认配置

当前默认 DDR 参数如下：

```
#define CONFIG_DDR_INNOPHY
#define CONFIG_DDR_DLL_OFF
#define CONFIG_DDR_PARAMS_CREATOR
#define CONFIG_DDR_HOST_CC
#define CONFIG_DDR_CS0 1 /* 1-connected, 0-disconnected */
#define CONFIG_DDR_CS1 0 /* 1-connected, 0-disconnected */
#define CONFIG_DDR_DW32 0 /* 1-32bit-width, 0-16bit-width */
/*#define CONFIG_DDR3_TSD34096M1333C9_E*/

#define CONFIG_DDR_PHY_IMPEDANCE 40
#define CONFIG_DDR_PHY_ODT_IMPEDANCE 120
/* #define CONFIG_FPGA_TEST */
/*#define CONFIG_DDR_AUTO_REFRESH_TEST*/

#define CONFIG_DDR_AUTO_SELF_REFRESH
#define CONFIG_DDR_AUTO_SELF_REFRESH_CNT 257
```

2.2.2.2 自定义配置

X2000 芯片内置的 DDR 类型为 LPDDR3，X2000E 芯片内置的 DDR 类型为 LPDDR3，可根据实际情况选择 DDR 类型：

1. DDR3 配置，如果定义 CONFIG_DDR_TYPE_DDR3，则 DDR3 相关的配置会被打开：

```
/*#define CONFIG_DDR_TYPE_DDR3*/
#ifdef CONFIG_DDR_TYPE_DDR3
#define CONFIG_DDR3_TSD34096M1333C9_E_FPGA
#endif
```

2. LPDDR3 配置，如果定义 CONFIG_DDR_TYPE_LPDDR3，则 LPDDR3 相关配置会被打开：

```
#define CONFIG_DDR_TYPE_LPDDR3
#ifdef CONFIG_DDR_TYPE_LPDDR3
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
/* #define CONFIG_LPDDR3_MT52L256M32D1PF_FPGA*/
/* #define CONFIG_LPDDR3_AD310032C_AB_FPGA */
#define CONFIG_LPDDR3_W63AH6NKB_BI
#define CONFIG_LPDDR3_W63AH6NKB_BI_MEM_FREQ 750000000
#endif
```

3. LPDDR2 配置，如果定义 CONFIG_DDR_TYPE_LPDDR2，则 LPDDR2 相关配置会被打开：

```
/*#define CONFIG_DDR_TYPE_LPDDR2*/
#ifdef CONFIG_DDR_TYPE_LPDDR2
/*#define CONFIG_LPDDR2_FMT4D32UAB_25LI_FPGA*/
/* #define CONFIG_LPDDR2_AD210032F_AB_FPGA */
#define CONFIG_LPDDR2_W97BV6MK
#define CONFIG_LPDDR2_W97BV6MK_MEM_FREQ 500000000
#endif
```

2.2.3 SFC 驱动配置

2.2.3.1 驱动文件

位置： drivers/mtd/devices/jz_sfc_v2

```
├── jz_sfc_common.c
├── jz_sfc_common.h
├── jz_sfc_nand.c
├── jz_sfc_nor.c
├── jz_sfc_ops.c
├── Makefile
└── nand_device
    ├── ato_nand.c
    ├── dosilicon_nand.c
    ├── foresee_nand.c
    ├── gd_nand.c
    ├── mxic_nand.c
    ├── nand_common.c
    ├── nand_common.h
    ├── xtx_mid0b_nand.c
    ├── xtx_nand.c
    └── zetta_nand.c
```

2.2.3.2 配置说明

1. sfc 控制器 gpio 引脚功能配置

```
/* sfc gpio */
/* #define CONFIG_JZ_SFC_PD_4BIT */
/* #define CONFIG_JZ_SFC_PD_8BIT */
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
/* #define CONFIG_JZ_SFC_PD_8BIT_PULL */
#define CONFIG_JZ_SFC_PE
```

2. sfc nand flash 配置，如果定义 CONFIG_SPL_SFC_NAND，则 nand 相关的配置会被打开：

```
/* sfc nand config */
#ifdef CONFIG_SPL_SFC_NAND
#define CONFIG_SFC_NAND_RATE 400000000 /* value <= 400000000(sfc 100Mhz)*/
#define CONFIG_SFC_QUAD
#define CONFIG_SPI_SPL_CHECK
#define CONFIG_SPIFLASH_PART_OFFSET 0x5800
#define CONFIG_SPI_NAND_BPP (2048 +64) /*Bytes Per Page*/
#define CONFIG_SPI_NAND_PPB (64) /*Page Per Block*/
#define CONFIG_JZ_SFC
#define CONFIG_CMD_SFCNAND
#define CONFIG_CMD_NAND
#define CONFIG_SYS_MAX_NAND_DEVICE 1
#define CONFIG_SYS_NAND_BASE 0xb3441000
#define CONFIG_SYS_MAXARGS 16
/**#define CONFIG_NAND_BUILTIN_PARAMS*/

/* sfc nand env config */
#define CONFIG_MTD_DEVICE
#define CONFIG_CMD_SAVEENV /* saveenv */
#define CONFIG_CMD_UBI
#define CONFIG_CMD_UBIFS
#define CONFIG_CMD_MTDPARTS
#define CONFIG_MTD_PARTITIONS
#define MTDIDS_DEFAULT "nand0:nand"
#define MTDPARTS_DEFAULT
    "mtdparts=nand:1M(boot),8M(kernel),40M(rootfs),-(data)"
#define CONFIG_SYS_NAND_BLOCK_SIZE (128 * 1024)
#endif
```

3. sfc nor flash 配置，如果定义 CONFIG_SPL_SFC_NOR，则 nor 相关的配置会被打开：

```
/* sfc nor config */
#ifdef CONFIG_SPL_SFC_NOR
#define CONFIG_JZ_SFC
#define CONFIG_CMD_SFC_NOR
#define CONFIG_JZ_SFC_NOR
#define CONFIG_SPI_SPL_CHECK
#define CONFIG_SFC_NOR_RATE 400000000 /* value <= 400000000(sfc 100Mhz)*/
#define CONFIG_SFC_QUAD
#define CONFIG_SPIFLASH_PART_OFFSET 0x5800
#define CONFIG_SPI_NORFLASH_PART_OFFSET 0x5874
#define CONFIG_NOR_MAJOR_VERSION_NUMBER 1
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
#define CONFIG_NOR_MINOR_VERSION_NUMBER 0
#define CONFIG_NOR_REVERSION_NUMBER 0
#define CONFIG_NOR_VERSION (CONFIG_NOR_MAJOR_VERSION_NUMBER |
    (CONFIG_NOR_MINOR_VERSION_NUMBER << 8) |
    (CONFIG_NOR_REVERSION_NUMBER <<16))
/*#define CONFIG_NOR_BUILTIN_PARAMS*/
#endif
```

2.2.3.3 默认配置

1. 使用 nand flash 启动，在 boards.cfg 文件的 halley5_uImage_sfc_nand 配置和 halley5_xImage_sfc_nand 配置中导出了 CONFIG_SPL_SFC_NAND。

2. 使用 nor flash 启动，在 boards.cfg 文件的 halley5_uImage_sfc_nor 配置中导出了 CONFIG_SPL_SFC_NOR。

2.2.3.4 自定义配置

1. sfc 控制器 gpio 引脚，可根据硬件连接情况进行选择

```
/* sfc gpio */
/* #define CONFIG_JZ_SFC_PD_4BIT */
/* #define CONFIG_JZ_SFC_PD_8BIT */
/* #define CONFIG_JZ_SFC_PD_8BIT_PULL */
#define CONFIG_JZ_SFC_PE
```

2. nand flash 配置

a. sfc 控制器速率，注意最终的速率为该宏定义 4 分频之后的结果

```
#define CONFIG_SFC_NAND_RATE 400000000 /* value <= 400000000(sfc 100Mhz)*/
```

b. 数据传输线宽，定义该宏定义为 4 线，否则为 1 线

```
#define CONFIG_SFC_QUAD
```

c. nand flash 分区参数存储在 nand flash 中的偏移

```
#define CONFIG_SPIFLASH_PART_OFFSET 0x5800
```

d. nand flash 页大小和块大小，根据具体的 nand flash 型号确定

```
#define CONFIG_SPI_NAND_BPP (2048 +64) /*Bytes Per Page*/
#define CONFIG_SPI_NAND_PPB (64) /*Page Per Block*/
```

e. 是否使用内置分区参数，定义则使用，否则不使用

```
/*#define CONFIG_NAND_BUILTIN_PARAMS*/
```

3. nor flash 配置

a. SFC 控制器时钟速率，注意最终的速率为该宏定义 4 分频之后的结果

```
#define CONFIG_SFC_NOR_RATE 400000000 /* value <= 400000000(sfc 100Mhz)*/
```

b. 数据传输线宽，定义该宏定义为 4 线，否则为 1 线

```
#define CONFIG_SFC_QUAD
```

c. nor flash 分区参数存储在 nor flash 中的偏移

```
#define CONFIG_SPIFLASH_PART_OFFSET 0x5800
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

d. nor flash 硬件参数存储在 nor flash 中的偏移

```
#define CONFIG_SPI_NORFLASH_PART_OFFSET 0x5874
```

e. 是否使用内置分区参数，定义则使用，否则不使用

```
/*#define CONFIG_NOR_BUILTIN_PARAMS*/
```

2.2.4 MSC 驱动配置

2.2.4.1 驱动文件

位置： drivers/mmc

```
├── jz_sdhci.c
├── jz_sdhci_regs.h
└── sdhci.c
```

2.2.4.2 配置说明

1. MSC0 配置，如果定义 CONFIG_JZ_MMC_MSC0，则 MSC0 相关的配置会被打开：

```
#ifdef CONFIG_JZ_MMC_MSC0
#define CONFIG_GENERIC_MMC
#define CONFIG_MMC
#define CONFIG_MMC_SPL_PARAMS
#define CONFIG_SDHCI
#define CONFIG_JZ_SDHCI
/*#define CONFIG_MMC_SDMA*/

/* MSC Command configuration */
#define CONFIG_CMD_MMC
#define CONFIG_SPL_JZ_MSC_BUS_8BIT

#define CONFIG_JZ_MMC_MSC0_PD //set gpio
/*#define CONFIG_MMC_TRACE // only for DEBUG*/
/*#define CONFIG_SDHCI_TRACE // only for DEBUG*/
#endif
```

2. MSC1 配置，如果定义 CONFIG_JZ_MMC_MSC1，则 MSC1 相关的配置会被打开：

```
#ifdef CONFIG_JZ_MMC_MSC1
#define CONFIG_GENERIC_MMC
#define CONFIG_MMC
#define CONFIG_SDHCI
#define CONFIG_MMC_SPL_PARAMS
#define CONFIG_JZ_SDHCI
/*#define CONFIG_MMC_SDMA*/
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
/* MSC Command configuration */
#define CONFIG_CMD_MMC

#define CONFIG_JZ_MMC_MSC1_PE //set gpio
/*#define CONFIG_MMC_TRACE // only for DEBUG*/
/*#define CONFIG_SDHCI_TRACE // only for DEBUG*/
#endif
```

3. MSC2 配置，如果定义 CONFIG_JZ_MMC_MSC2，则 MSC2 相关的配置会被打开：

```
#ifdef CONFIG_JZ_MMC_MSC2
#define CONFIG_GENERIC_MMC
#define CONFIG_MMC
#define CONFIG_SDHCI
#define CONFIG_MMC_SPL_PARAMS
#define CONFIG_JZ_SDHCI
/*#define CONFIG_MMC_SDMA*/

/* MSC Command configuration */
#define CONFIG_CMD_MMC
#define CONFIG_SPL_JZ_MSC_BUS_4BIT

#define CONFIG_JZ_MMC_MSC2_PE //set gpio
/*#define CONFIG_MMC_TRACE // only for DEBUG*/
/*#define CONFIG_SDHCI_TRACE // only for DEBUG*/
#endif
```

2.2.4.3 默认配置

使用 sd 卡启动，在 boards.cfg 文件的 halley5_uImage_msc2 配置中导出了 CONFIG_JZ_MMC_MSC2，即当前默认使用 MSC2 配置。

2.2.4.4 自定义配置

MSC0、MSC1、MSC2 每种配置中，均可以自定义如下配置：

1. 数据传输位数，MSC0 最多支持 8 位，MSC1 最多支持 4 位，MSC2 最多支持 4 位

```
#define CONFIG_SPL_JZ_MSC_BUS_4BIT
```

```
#define CONFIG_SPL_JZ_MSC_BUS_8BIT
```

2. 是否打开 DEBUG 信息

```
/*#define CONFIG_MMC_TRACE // only for DEBUG*/
/*#define CONFIG_SDHCI_TRACE // only for DEBUG*/
```

2.2.4.5 分区表介绍

sd 卡启动的 u-boot 镜像中包含有分区表的信息，在编译 u-boot 镜像的最后，有如下输出信息：

```
cat tools/ingenic-tools/mbr-gpt.bin u-boot-with-spl.bin > u-boot-with-spl-mbr-gpt.bin
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

可以看到，在最终生成的 u-boot 镜像前面加入了 mbr-gpt.bin，该文件为按照 GPT 分区标准生成的分区数据，用于将 u-boot 镜像烧录到 sd 卡后，可直接在 PC 端识别出分区信息，而不必再利用分区工具对 sd 卡进行分区操作。

该分区数据 mbr-gpt.bin 中包括了具体的分区信息，下面介绍分区信息，关于 MBR 与 GPT 分区标准相关内容，可参考 5.2MBR/GPT 分区表。

在 u-boot/board/ingenic/halley5/partitions.tab 中存放有具体的分区信息：

```
property:
    disk_size = 4096m
    gpt_header_lba = 512
    custom_signature = 0

partition:
   #name      =  start,    size,  fstype
   xboot      =     0m,      3m,
   boot       =     3m,      8m,  EMPTY
   recovery   =    12m,     16m,  EMPTY
   pretest    =    28m,     16m,  EMPTY
   reserved   =    44m,     52m,  EMPTY
   misc       =    96m,      4m,  EMPTY
   cache      =   100m,    100m,  LINUX_FS
   system     =   200m,   1800m,  LINUX_FS
   data       =  2000m,   2048m,  LINUX_FS

#fstype could be: LINUX_FS, FAT_FS, EMPTY
```

当我们把镜像烧录到 sd 卡后，在 PC 端使用 fdisk 命令可看到如下分区信息：

```
设备         Start  末尾  扇区  Size 类型
/dev/sdc1     6144   22527   16384    8M Microsoft basic data
/dev/sdc2    24576   57343   32768   16M Microsoft basic data
/dev/sdc3    57344   90111   32768   16M Microsoft basic data
/dev/sdc4    90112  196607  106496   52M Microsoft basic data
/dev/sdc5   196608  204799    8192    4M Microsoft basic data
/dev/sdc6   204800  409599  204800  100M Microsoft basic data
/dev/sdc7   409600 4095999 3686400  1.8G Microsoft basic data
/dev/sdc8  4096000 8290303 4194304    2G Microsoft basic data
```

该分区信息与上述 partitions.tab 文件中的内容完全一致。

2.2.5 GMAC 网络驱动配置

2.2.5.1 驱动文件

位置： drivers/net

```
├── jz_gmac_v12.c
├── SynopGMAC_Dev.c
└── SynopGMAC_Dev.h
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

2.2.5.2 配置说明

1. ip 地址相关

```
/* DEBUG ETHERNET */
#define CONFIG_SERVERIP 192.168.4.13
#define CONFIG_IPADDR 192.168.4.145
#define CONFIG_GATEWAYIP 192.168.4.1
#define CONFIG_NETMASK 255.255.255.0
#define CONFIG_ETHADDR 00:11:22:33:44:55
```

2. 网口工作模式选择，需要按照实际连接的网络 phy 芯片进行选择

```
#define GMAC_PHY_MII 0
#define GMAC_PHY_RMII 4
#define GMAC_PHY_GMII 0
#define GMAC_PHY_RGMII 1

#ifdef CONFIG_RGMII
#define CONFIG_NET_GMAC_PHY_MODE GMAC_PHY_RGMII
#else
#define CONFIG_NET_GMAC_PHY_MODE GMAC_PHY_RMII
#endif
```

3. gmac 控制器选择

```
#ifdef CONFIG_GMAC1
#define CONFIG_GAMAC_MODE_CTRL_ADDR 0xb00000e8
#define JZ_GMAC_BASE GMAC1_BASE
#define CONFIG_GMAC_CRLT_PORT GPIO_PORT_B
#define CONFIG_GMAC_CRLT_PORT_PINS (0xffff << 8)
#define CONFIG_GMAC_CRTL_PORT_INIT_FUNC GPIO_FUNC_3
#define CONFIG_GMAC_PHY_RESET GPIO_PB(16)
#define CONFIG_GMAC_TX_CLK_DELAY 0x3f
#define CONFIG_GMAC_RX_CLK_DELAY 0
#endif
#ifdef CONFIG_GMAC0
#define CONFIG_GAMAC_MODE_CTRL_ADDR 0xb00000e4
#define JZ_GMAC_BASE GMAC0_BASE
#define CONFIG_GMAC_CRLT_PORT GPIO_PORT_C
#define CONFIG_GMAC_CRLT_PORT_PINS (0x7fff << 1)
#define CONFIG_GMAC_CRTL_PORT_INIT_FUNC GPIO_FUNC_1
#define CONFIG_GMAC_PHY_RESET GPIO_PC(22)
#define CONFIG_GMAC_TX_CLK_DELAY 0x3f
#define CONFIG_GMAC_RX_CLK_DELAY 0
#endif

#define CONFIG_GMAC_CRTL_PORT_SET_FUNC GPIO_INPUT
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
#define CONFIG_GMAC_PHY_RESET_ENLEVEL 0
```

2.2.5.3 默认配置

1. 当前默认选择 gmac1 控制器：

```
/* Select GMAC Controller */
/*#define CONFIG_GMAC0*/
#define CONFIG_GMAC1
```

2. 当前默认选择 RGMII 网口工作模式：

```
/* Select GMAC Interface mode */
#define CONFIG_RGMII
```

2.2.5.4 自定义配置

1.当前 u-boot 只同时支持一个 gmac 控制器工作，根据实际硬件配置，控制器可选择 gmac0 或者 gmac1。

2.根据实际速度需要，工作模式可选择 RGMII 或者 RMII，其中 RGMII 支持 10/100/1000 兆的总线接口速度，RMII 支持 10 兆和 100 兆的总线接口速度。

2.2.6 u-boot built-in 命令配置

2.2.6.1 默认配置

当前默认支持如下命令：

```
/**
* Command configuration.
*/
#define CONFIG_CMD_BOOTD /* bootd */
#define CONFIG_CMD_CONSOLE /* coninfo */
#define CONFIG_CMD_DHCP /* DHCP support */
#define CONFIG_CMD_ECHO /* echo arguments */
#define CONFIG_CMD_EXT4 /* ext4 support */
#define CONFIG_CMD_FAT /* FAT support */
#define CONFIG_USE_XYZMODEM /* xyzModem */
#define CONFIG_CMD_LOAD /* serial load support */
#define CONFIG_CMD_LOADB /* loadb */
#define CONFIG_CMD_LOADS /* loads */
#define CONFIG_CMD_MEMORY /* md mm nm mw cp cmp crc base loop mtest */
#define CONFIG_CMD_MISC /* Misc functions like sleep etc*/
#define CONFIG_CMD_NET /* networking support */
#define CONFIG_CMD_PING
#define CONFIG_CMD_RUN /* run command in env variable */
#define CONFIG_CMD_SETGETDCR /* DCR support on 4xx */
#define CONFIG_CMD_SOURCE /* "source" command support */
#define CONFIG_CMD_GETTIME
#define CONFIG_CMD_GPIO
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
#define CONFIG_CMD_EXT2
#define CONFIG_CMD_EXT4
#define CONFIG_CMD_FAT
#define CONFIG_EFI_PARTITION
#define CONFIG_SOFT_BURNER

#define CONFIG_CMD_DDR_TEST /* DDR Test Command */
```

2.2.6.2 自定义配置

下面以新建一个简单的 hello 命令为例介绍如何在 u-boot 中自定义命令。

1. 在 common 目录下，新建一个文件 cmd_hello.c，输入如下代码：

```
#include<command.h>
#include<common.h>

static int do_hello(cmd_tbl_t *cmdtp, int flag, int argc, char * const argv[])
{
    printf("hello world!\n");
    return 0;
}

U_BOOT_CMD(
    hello,   1,   0,   do_hello,
    "short description. This is a string.\n",
    "long description. This is a string.\n"
);
```

上述代码中，U_BOOT_CMD 为 u-boot 新建命令使用的宏定义，一共有 6 个参数，具体介绍如下：

a. 第一个参数：该命令的名称
b. 第二个参数：该命令的参数的个数
c. 第三个参数：按下 enter 键后是否重复执行该命令
d. 第四个参数：该命令对应的回调函数
e. 第五个参数：该命令的短帮助信息
f. 第六个参数：该命令的长帮助信息

2. 在 common/Makefile 中添加编译选项：

```
COBJS-y += cmd_hello.o
```

3. 重新编译 u-boot 即可

2.2.7 内核传参说明

参考 u-boot 内核传参说明

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

3 u-boot 编译

u-boot 的编译分为两种，两种的特点如下：

编译方式	特点
工程顶层目录编译	依赖配合君正 SDK 板级配置。适合基于君正 SDK 的开发方式
u-boot 源码目录编译	直接按照 u-boot 的标准编译方式进行，适合需要自定义 u-boot 的配置，脱离君正 SDK 的开发方式。

3.1 工程顶层目录编译

3.1.1 编译 SFC Nand 启动镜像

该启动镜像对应 halley5_uImage_sfc_nand 配置。

1. 在工程顶层目录下执行如下命令，导入编译 halley5 平台所需的环境变量：

```
$ source build/envsetup.sh
```

2. 执行 lunch 命令选择需要编译镜像的类型：

```
$ lunch
```

a. 如果是 halley5 v10 开发板，根据需要在以下配置中进行选择：

```
1. halley5.v10_nand_4.4.94-eng
2. halley5.v10_nand_4.4.94-user
3. halley5.v10_nand_4.4.94-userdebug
```

b. 如果是 halley5 v20 开发板，根据需要在以下配置中进行选择：

```
13. halley5.v20_nand_4.4.94-eng
14. halley5.v20_nand_4.4.94-user
15. halley5.v20_nand_4.4.94-userdebug
```

3. 执行 make uboot-clean 清除上次编译产生的中间产物：

```
$ make uboot-clean
```

4. 执行 make uboot 进行编译：

```
$ make uboot
```

5. 等待编译完成，在工程/out/product/halley5/image 目录下便会生成 uboot 镜像文件，该镜像即可直接进行烧录。

3.1.2 编译 SFC Nor 启动镜像

该启动镜像对应 halley5_uImage_sfc_nor 配置。

1. 在工程顶层目录下执行如下命令，导入编译 halley5 平台所需的环境变量：

```
$ source build/envsetup.sh
```

2. 执行 lunch 命令选择需要编译镜像的类型：

```
$ lunch
```

a. 如果是 halley5 v10 开发板，根据需要在以下配置中进行选择：

```
4. halley5.v10_nor_4.4.94-eng
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
5. halley5.v10_nor_4.4.94-user
6. halley5.v10_nor_4.4.94-userdebug
```

b.如果是 halley5 v20 开发板，根据需要在以下配置中进行选择：

```
16. halley5.v20_nor_4.4.94-eng
17. halley5.v20_nor_4.4.94-user
18. halley5.v20_nor_4.4.94-userdebug
```

3.执行 make uboot-clean 清除上次编译产生的中间产物：

```
$ make uboot-clean
```

4.执行 make uboot 进行编译：

```
$ make uboot
```

5.等待编译完成，在工程/out/product/halley5/image 目录下便会生成 uboot 镜像文件，该镜像即可直接进行烧录。

3.1.3 编译 SD 卡启动镜像

该启动镜像对应 halley5_uImage_msc2 配置。

1.在工程顶层目录下执行如下命令，导入编译 halley5 平台所需的环境变量：

```
$ source build/envsetup.sh
```

2.执行 lunch 命令选择需要编译镜像的类型：

```
$ lunch
```

a.如果是 halley5 v10 开发板，根据需要在以下配置中进行选择：

```
7. halley5.v10_msc_4.4.94-eng
8. halley5.v10_msc_4.4.94-user
9. halley5.v10_msc_4.4.94-userdebug
```

b.如果是 halley5 v20 开发板，根据需要在以下配置中进行选择：

```
19. halley5.v20_msc_4.4.94-eng
20. halley5.v20_msc_4.4.94-user
21. halley5.v20_msc_4.4.94-userdebug
```

3.执行 make uboot-clean 清除上次编译产生的中间产物：

```
$ make uboot-clean
```

4.执行 make uboot 进行编译：

```
$ make uboot
```

5.等待编译完成，在工程/out/product/halley5/image 目录下便会生成 uboot 镜像文件，该镜像即可直接进行烧录。

3.2 u-boot 源码目录编译

1. 在工程顶层目录下执行如下命令，导入编译 halley5 平台所需的环境变量：

```
$ source build/envsetup.sh
```

2. 执行 lunch 命令选择需要编译镜像的类型， 这里只是为了导入交叉编译工具环境，如果只编译 u-boot 可以随意选择：

```
$ lunch
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

3. 之后进入 u-boot 目录，执行 make distclean 清除上次编译产生的中间产物：

```
$ make distclean
```

执行 make xxxxxx -jN 命令进行编译，xxxxxx 为具体的板级配置名称，需要根据编译的启动镜像来决定，详细介绍请参考 2.1 boards.cfg 文件说明。当前共支持 4 种板级配置，具体如下表所示：

halley5_uImage_msc2	该配置支持从 sd 卡启动 linux 内核镜像 uImage
halley5_uImage_sfc_nand	该配置支持从 nand flash 启动 linux 内核镜像 uImage
halley5_uImage_sfc_nor	该配置支持从 nor flash 启动 linux 内核镜像 uImage
halley5_xImage_sfc_nand	该配置支持从 nand flash 启动 linux 内核镜像 xImage

比如，编译从 nand flash 启动 uImage 的启动镜像，执行如下命令：

```
$ make halley5_uImage_sfc_nand -jN
```

其中-jN 为使用多线程编译，可以加快编译速度，具体的 N 值需要根据使用的 PC 决定。

4. 编译好之后，会在 u-boot 工程顶层目录下生成 u-boot-with-spl-mbr-gpt.bin，该文件便是最终生成的镜像文件，可以直接进行烧录。

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

4 u-boot 应用与内核接口

4.1 u-boot 常用命令

进入 uboot 的命令行模式以后输入“help”或者“？”，然后按下回车即可查看当前 uboot 所支持的命令。

现以 sfc nand flash 启动为例，支持的命令如下图所示：

```
halley5# ?
?       - alias for 'help'
base    - print or set address offset
boot    - boot default, i.e., run 'bootcmd'
boota   - boot android system
bootd   - boot default, i.e., run 'bootcmd'
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
chpart  - change active partition
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
dhcp    - boot image via network using DHCP/TFTP protocol
echo    - echo args to console
env     - environment handling commands
ext2load- load binary file from a Ext2 filesystem
ext2ls  - list files in a directory (default /)
ext4load- load binary file from a Ext4 filesystem
ext4ls  - list files in a directory (default /)
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
gettime - get timer val elapsed,
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
go      - start application at address 'addr'
gpio    - input/set/clear/toggle gpio pins
help    - print command description/usage
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
md      - memory display
mm      - memory modify (auto-incrementing address)
mtdparts- define flash/nand partitions
mw      - memory write (fill)
nand    - NAND sub-system
nboot   - boot from NAND device
nm      - memory modify (constant address)
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv  - set environment variables
sfcnand - sfcnand    - SFC_NAND sub-system

sleep   - delay execution for some time
softburn- Ingenic usb soft burn
source  - run script from memory
tftpboot- boot image via network using TFTP protocol
ubi     - ubi commands
ubifsload- load file from an UBIFS filesystem
ubifsls - list files in a directory
ubifsmount- mount UBIFS volume
ubifsumount- unmount UBIFS volume
version - print monitor, compiler and linker version
```

对于每一个命令，还可以执行“help 命令名称”获取该命令具体的使用说明，如下图所示：

```
halley5# help bootm
bootm - boot application image from memory

Usage:
bootm [addr [arg ...]]
    - boot application image stored in memory
        passing arguments 'arg ...'; when booting a Linux kernel,
        'arg' can be the address of an initrd image

Sub-commands to do part of the bootm sequence.  The sub-commands must be
issued in the order below (it's ok to not issue all sub-commands):
        start [addr [arg ...]]
        loados  - load OS image
        cmdline - OS specific command line processing/setup
        bdt     - OS specific bd_t processing
        prep    - OS specific prep before relocation or go
        go      - start OS
```

下面介绍一些常用的命令：

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

printenv	打印当前环境变量
setenv	设置环境变量，格式：setenv name value …，表示将 name 变量设置成 value 值；如果没有这个参数，表示删除该变量
saveenv	保存环境变量到存储设备中
sleep	延迟执行，格式：sleep N，可以延迟 N 秒钟执行
bootm	从内存的指定地址处启动内核镜像。格式：bootm addr，参数是镜像在内存中的地址
md	读内存数据，格式：md [.b, .w, .l] address [# of objects]，第一个参数表示一次读取的数据单位（b:8 位，w:16 位，l:32 位），第二个参数 address 表示要读取的内存起始地址，第三个参数表示从 address 开始读取的数据个数
nand erase	擦除 nand flash，格式：nand erase addr1 count，第一个参数是 offset，第二个参数是擦除字节数
nand write	将内存数据写入 nand flash，格式：nand write addr offset count，第一个参数是写入基地址，第二个参数是偏移地址，第三个参数是写入字节数
nand read	将 nand flash 的数据读取到内存，格式：nand read addr offset count，第一个参数是读取的 nand flash 地址，第二个参数是内存位置偏移，第三个参数是读取字节数
cp	在内存中复制数据，格式：cp source target count，第一个参数是源地址，第二个参数是目的地址，第三个参数是复制数目
cmp	比较内存中的数据，格式：cmp addr1 addr2 count，第一个参数是内存地址一，第二个参数是内存地址二，第三个是比较长度（单位是字节数除以 4，以 WORDS 为单位）
crc32	计算校验值，格式：crc32 address count [addr]，第一个参数是需校验的起始地址，第二个参数是校验的数据字节数，第三个参数是保存校验值的地址

4.2 u-boot 内核传参说明

linux 内核在启动之前，需要知道当前硬件的一些必要的参数，比如可使用的内存大小、使用的串口设备、根文件系统所在的设备及分区等。关于以上的启动参数，linux 内核已经定义好了一系列规则，负责启动它的 bootloader 需要遵守这些规则来传递启动参数。方便的是，u-boot 已经帮我们做好了大部分工作，它提供了一个环境变量 bootargs，我们只需要按照规定的格式将必要的启动参数设置给 bootargs 即可。

具体的，我们可以通过在 halley5.h 中将启动参数设置给宏定义 CONFIG_BOOTARGS。

在 halley5.h 中，根据加载内核时挂载的文件系统的不同均有对应的启动参数，它们有一些共有的内容，在 halley5.h 中定义如下：

```
#define BOOTARGS_COMMON "console=ttyS3,115200 mem=128M@0x0 "
```

具体说明如下：

1. console=ttyS3,115200 表示串口使用/dev/ttyS3 这个设备，波特率为 115200
2. mem=128M@0x0 表示可使用的内存总大小为 128MB，起始地址为 0x0

启动参数中特有的部分请见下文。

4.2.1 u-boot 加载内核挂载网络文件系统

网络文件系统一般用于开发时调试使用，当前在 halley5.h 中的参考启动参数如下：

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
#define CONFIG_BOOTARGS
    BOOTARGS_COMMON "ip=192.168.10.207:192.168.10.1:192.168.10.1:255.255.255.0 \
nfsroot=192.168.4.13:/home/nfsroot/fpga/user/pzqi/rootfs-tst rw"
```

具体说明如下：

1. ip=192.168.10.207:192.168.10.1:192.168.10.1:255.255.255.0 表示客户端（开发板）ip 地址为 192.168.10.207，服务端 ip 地址为 192.168.10.1，网关 ip 地址为 192.168.10.1，子网掩码 ip 地址为 255.255.255.0
2. nfsroot=192.168.4.13:/home/nfsroot/fpga/user/pzqi/rootfs-tst 表示服务端 ip 地址为 192.168.4.13，挂载到的服务端目录为/home/nfsroot/fpga/user/pzqi/rootfs-tst
3. rw 表示挂载的文件系统可读写

注意：如果 kernel 配置了双网络控制器，需要在 ip 项中指定启动网络文件系统的设备：

ip=192.168.10.207:192.168.10.1:192.168.10.1:255.255.255.0::eth0:off /*根据选择的网口，修改 eth0:off 或 eth1:off*/

4.2.2 u-boot 加载内核挂载 jffs2 文件系统

该启动参数对应 **halley5_uImage_sfc_nor** 配置，当前在 halley5.h 中的启动参数如下：

```
#define CONFIG_BOOTARGS
    BOOTARGS_COMMON "ip=off init=/linuxrc rootfstype=jffs2 root=/dev/mtdblock2 rw"
```

具体说明如下：

1. ip=off 表示不使用网络文件系统
2. init=/linuxrc 表示 init 进程使用的是根目录下的 linuxrc
4. rootfstype=jffs2 表示根文件系统格式为 jffs2
5. root=/dev/mtdblock2 表示存放根文件系统的设备为/dev/mtdblock2，对应着 nor flash 编号为 2 的分区
6. rw 表示挂载的文件系统可读写

4.2.3 u-boot 加载内核挂载 ext4 文件系统

该启动参数对应 **halley5_uImage_msc2** 配置，当前在 halley5.h 中的启动参数如下：

```
#define CONFIG_BOOTARGS
    BOOTARGS_COMMON " rootfstype=ext4 root=/dev/mmcblk0p7 rootdelay=3 rw"
```

具体说明如下：

1. rootfstype=ext4 表示根文件系统格式为 ext4
2. root=/dev/mmcblk0p7 表示存放根文件系统的设备为/dev/mmcblk0p7，对应着当前系统第一个 emmc 或 sd 设备的编号为 7 的分区
3. rootdelay=3 表示等待 3 秒再挂载根文件系统
4. rw 表示挂载的文件系统可读写

4.2.4 u-boot 加载内核挂载 ubifs 文件系统

该启动参数对应 **halley5_uImage_sfc_nand** 配置，当前在 halley5.h 中的启动参数如下：

```
#define CONFIG_BOOTARGS
    BOOTARGS_COMMON "ip=off init=/linuxrc ubi.mtd=2 root=ubi0:rootfs ubi.mtd=3  \
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
rootfstype=ubifs rw"
```

具体说明如下：

1. ip=off 表示不使用网络文件系统启动
2. init=/linuxrc 表示 init 进程使用的是根目录下的 linuxrc
3. ubi.mtd=2 root=ubi0:rootfs ubi.mtd=3 rootfstype=ubifs rw 表示当前 nand flash 编号为 2 和 3 的分区使用的是 ubifs 文件系统,并且根文件系统以可读写的方式挂载在了编号为 2 的分区上。

4.3 u-boot 加载内核镜像

u-boot 通过 tftp 或者 fastboot 方式加载内核镜像可省去烧录 kernel 镜像的过程，直接将内核镜像加载到内存，使用该方式在开发过程中非常便于调试使用。

4.3.1 uboot 通过 tftp 加载内核镜像

tftp 方式可通过网络将 kernel 镜像直接下载到内存，所以注意使用该方式需要 u-boot 支持网络。

1. uboot 只支持一个 gmac 控制器工作，需要在 halley5.h 中配置选择的控制器和对应控制器的接口模式：

```
/* Select GMAC Controller */
#define CONFIG_GMAC1 /*根据实际测试，配置对应的 GMAC1、GMAC0*/
/* Select GMAC Interface mode */
#define CONFIG_RGMII /*根据实际测试，配置对应 RGMII、RMII*/
```

2. PC 服务器端配置

```
$  sudo apt-get install tftpd-hpa  #安装服务端程序
$  sudo service tftpd-hpa start    #启动服务端
$  sudo apt-get install tftp-hpa   #安装客户端程序
$  tftp 127.0.0.1                  #测试服务端是否设置成功
tftp> get test.txt
```

3. 开发板端配置

a. 进入 uboot 命令行模式，根据网络情况设置 ip 地址：

```
halley5# set ipaddr 192.168.4.145
halley5# set serverip 192.168.4.146
```

b. 将需要的 uImage 放入 PC 服务器文件夹下，执行以下命令将 uImage 下载到内存 0x80800000 位置处：

```
halley5# tftp 0x80800000 uImage
```

c. 启动 uImage：

```
halley5# bootm 0x80800000
```

4.3.2 uboot 通过 fastboot 命令加载内核镜像

fastboot 方式是使用 usb 端口把 kernel 镜像下载到内存并启动的命令，只用于调试使用,不支持 android fastboot 其他功能。

1. PC 端使用以下命令安装 fastboot 应用程序：

```
$  sudo apt-get install android-tools-fastboot
```

2. 进入 uboot 命令行模式，输入 fastboot 命令：

```
halley5# fastboot
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

3. 在 PC 端存放 kernel 镜像的路径下执行以下命令：

```
$ sudo fastboot boot uImage
```

4. 命令执行成功后，即可将 kernel 镜像下载到内存并启动。

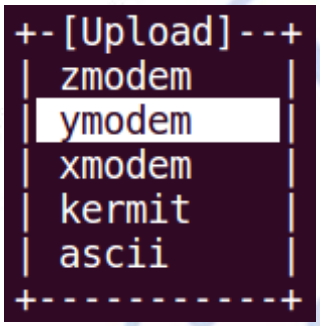
4.3.3 u-boot 通过串口加载内核镜像

串口方式是通过串口将 kernel 镜像直接下载到内存，注意该方式传输速度会较慢，请根据实际情况选择。

1. 进入 u-boot 命令行模式，执行如下命令：

```
halley5# loady 0x80800000 115200
## Ready for binary (ymodem) download to 0x80800000 at 115200 bps...
C
```

2. 先按住 ctrl 键不放，接着按住 a 键，再同时放开两个按键，最后按一下 s 键，可以看到如下图所示画面：



这里按↓键选择 ymodem 协议，然后回车；

3. 然后会出现如下图所示画面，这里可以选择需要传输的文件，↑↓键进行移动，双敲空格键进入目录，单敲空格键选择文件，回车键确认选择。

我们选择需要传输的 kernel 镜像：

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
+-------------------[Select one or more files for upload]-------------------+
|Directory: /home/user/work/x2000_release_test/image_sd                     |
| [..]                                                                      |
| kernel                                                                    |
| system.ext2                                                               |
| uboot                                                                     |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|              ( Escape to exit, Space to tag )                             |
+---------------------------------------------------------------------------+
```

4. 选择完毕并回车确认后，会出现如下进行文件的传输的画面：

```
+-----------[ymodem upload - Press CTRL-C to quit]------------+
|Sending: kernel                                              |
|Ymodem sectors/kbytes sent: 11912/1489k                      |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
+-------------------------------------------------------------+
```

5. 等待传输完毕，便可以通过 bootm 命令启动写到内存 0x80800000 位置处的 kernel 镜像：

```
$ bootm 0x80800000
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

5 Ingenic tools 介绍

Ingenic tools 是君正厂商添加在 u-boot 内的工具集合。

5.1 DDR 参数相关

源码位置：u-boot/tools/ingenic-tools/

```
├── ddr_creator_x2000
│     ├── add_chip_info.c
│     ├── ddr3_params.c
│     ├── ddr_params_creator.c
│     ├── ddr_params_creator.h
│     ├── lpddr2_params.c
│     ├── lpddr3_params.c
│     ├── Makefile
│     ├── supported_ddr_chips.c
```

这里主要用于生成相应的 DDR 参数，然后写到 u-boot 镜像的指定位置，以此进行区分。

5.2 MBR/GPT 分区表

源码位置：u-boot/tools/ingenic-tools/

```
├── gpt.bin
├── gpt_creator.c
├── gpt_tab_to_c.sh
├── mbr_creator.c
├── mbr-of-gpt.bin
├── mk-gpt-xboot.sh
```

GPT 和 MBR 均是关于硬盘分区的标准。

MBR：Master Boot Record，主分区引导记录，记录着硬盘本身的相关信息以及硬盘各个分区的大小及位置信息。

GPT：Globally Unique Identifier Partition Table，GUID 分区表，是源自 UEFI 标准的一种较新的磁盘分区表结构的标准。与 MBR 分区方案相比，GPT 提供了更加灵活的磁盘分区机制。

这里文件的作用是按照 GPT 和 MBR 的标准生成对应的分区数据。

5.3 nand/nor flash 相关

源码位置：u-boot/tools/ingenic-tools/

1. 保存有各种 nand flash 的硬件参数，用于 spl 中的 sfc 驱动使用

```
├── nand_device
```

2. 保存 nand flash 和 nor flash 的分区参数，区别于由烧录工具写入分区参数，最终会由 kernel 中的 sfc 驱动读取并生成对应的 mtd 分区，如有需要可自行更改

```
├── sfc_builtin_params
│     ├── nand_device.c
│     ├── nand_device.h
│     ├── nor_device.c
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

```
│       └── nor_device.h
```

3. 用于写入 nand flash 和 nor flash 的分区参数到 u-boot 镜像中的指定位置

```
├── sfc_nand_builtin_params.c
├── sfc_nor_builtin_params.c
```

5.4 其他

源码位置：u-boot/tools/ingenic-tools/

1. 用于生成 sfc 控制器的硬件 timing 参数

```
├── sfc_timing_params.c
```

2. 用于 sfc boot 时生成 spl 前面的头部数据，为 bootom 加载 spl 作校验用

```
├── spi_checksum.c
```

3. x2000 快速启动时使用，用于生成 spl env 参数

```
└── spl_params_fixer_x2000_v12.c
```


Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.

6 参考文档

1. u-boot/README
2. kernel-4.4.94/Documentation/kernel-parameters.txt

Copyright® 2005-2020 Ingenic Semiconductor Co., Ltd. All rights reserved.