双椒派开发板系统安装与更新指南

(V1.1)

2023 年 4 月 29

修改历史

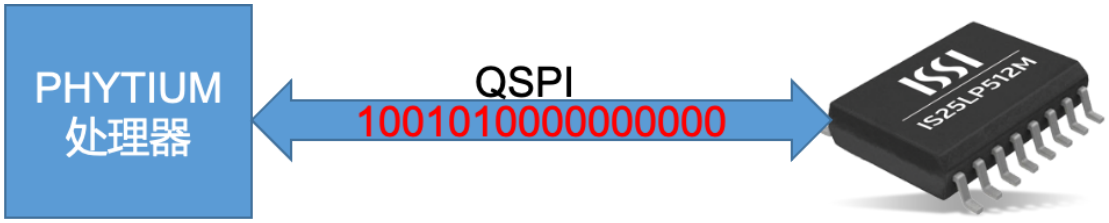
版本	作者	修改	备注
V1.0		内容创建	
V1.1		格式重排	

目 录

1 系统安装介绍

1.1 系统上电启动过程

1、处理器完成正常的上电复位之后，会从 0 地址获取指令并执行，飞腾处理器固件程序通过 QSPI 接口，从 spi norflash 芯片中读取程序。

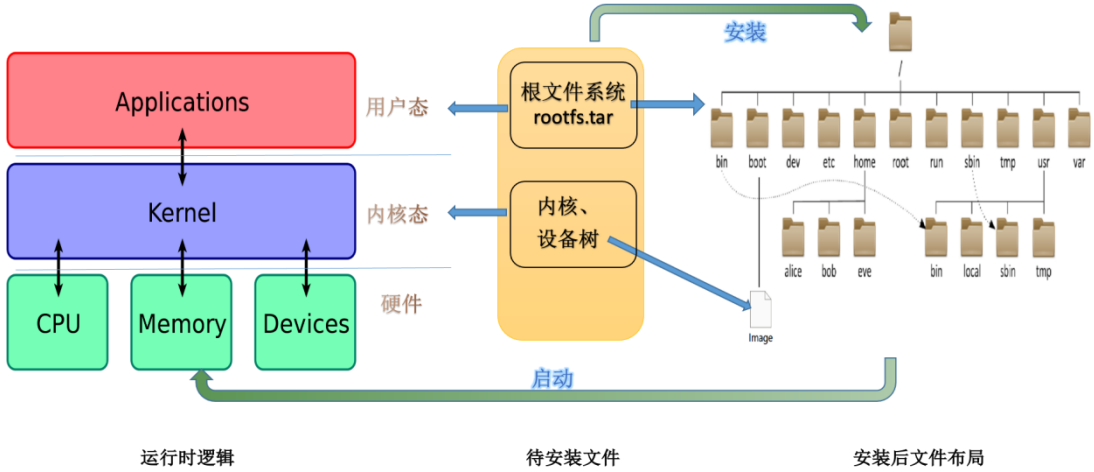


2、固件程序是处理器上电运行的第一个软件程序，完成处理器基本的硬件初始化配置，然后从存储介质加载操作系统镜像，并引导操作系统启动运行。

飞腾的固件分为飞腾处理器基础固件（Processor Base Firmware， PBF）和系统固件（System Firmware，SFW）两部分。

- 基础固件层（PBF）： 针对飞腾处理器开发的基础固件，对飞腾处理器进行基本的初始化工作，由飞腾公司开发和发布，运行在 EL3。
- 系统固件层（U-BOOT/UEFI）：负责部分外设驱动，操作系统引导，提供固件层的操作平台，运行在 Non-Secure EL2。根据应用领域不同，UEFI 多用于个人电脑、笔记本或服务器，U-Boot 通常应用于嵌入式领域。

1.2 操作系统的安装与启动



如上图所示，操作系统安装的目的是将编译好的内核及根文件系统放入可以启动的存储介质中。而启动的目的是将存储介质上的内核与根文件系统进行加载运行。

2 系统更新指南

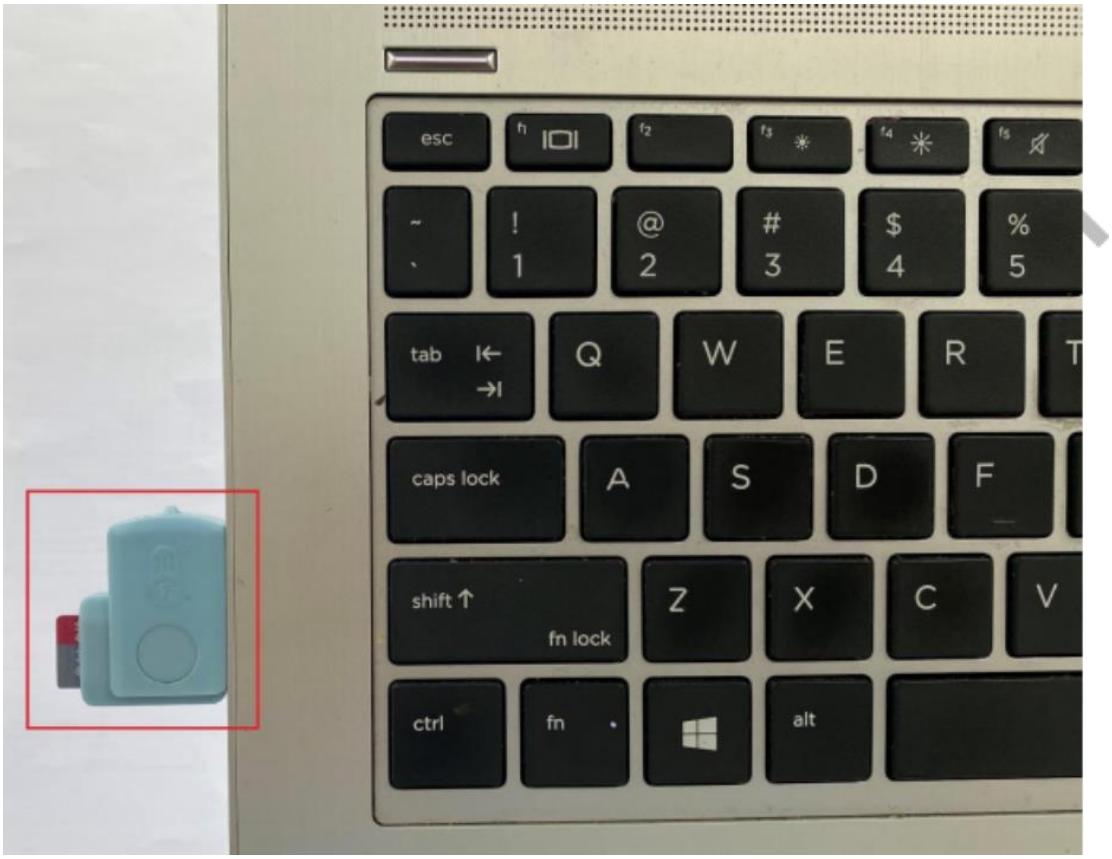
本节我们来介绍如何将编译好的、用于启动的 Linux 内核及根文件系统装载入选定测存储介质。

启动存储介质可以选择：U 盘、Micro SD 卡、NVME 硬盘、SATA 硬盘，这里以 Micro SD 卡为例。

2.1 准备启动介质

1、插入

通过 USB 读卡器，将 Micro SD 卡连接到 Ubuntu PC 主机



2、分区

1）使用 df 命令获得 Micro SD 卡自动挂载设备名称

```
linux@ubuntu:~$ df
```

2）使用 fdisk 命令分区

```
sudo fdisk /dev/sd[X]
```

例如：分为两个区

- /dev/sdc1 为 Micro SD 卡 存放内核分区 大小为 500M
- /dev/sdc2 为 Micro SD 卡 rootfs 分区，不小于 10G

注意：Micro SD 卡的自动挂载目录（既/dev/sdc）会发生变化，请根据实际情况进行操作， 以免造成数据丢失

3、格式化

```
linux@ubuntu:~$ sudo mkfs.ext4 /dev/sdc1
```

```
linux@ubuntu:~$ sudo mkfs.ext4 /dev/sdc2
```

注意：用于装入 rootfs 的分区需用 ext 文件系统格式

2.2 装入 Linux 内核镜像和设备树镜像

拷贝新的 Linux 内核镜像和设备树镜像到 Micro SD 卡 第一 分区

```
linux@ubuntu:~$ sudo mount /dev/sdc1 /mnt
linux@ubuntu:~$ sudo cp Image /mnt
linux@ubuntu:~$ sync
linux@ubuntu:~$ sudo umount /mnt
```

注意：设备树文件与板卡型号匹配

2.3 装入文件系统

解压缩新的根文件系统到 Micro SD 卡 第二分区

```
linux@ubuntu:~$ sudo mount /dev/sdc2 /mnt
linux@ubuntu:~$ cd /mnt
linux@ubuntu:~$ tar xvf <pathtobuild>output/images/rootfs.tar
linux@ubuntu:~$ sync
linux@ubuntu:~$ sudo umount /mnt
```

注意：rootfs 文件系统必须在 Linux 下解压（因为有硬连接）

3 U-boot 启动参数配置

3.1 MMC 介质启动

```
E2000# setenv bootargs console=ttyAMA1,115200  audit=0
earlycon=pl011,0x2800d000 root=/dev/mmcblk1p2 rootdelay=3 rw;
E2000# mmc dev 1
```

```
E2000# ext4load mmc 1:1 0x90000000 e2000d-chillipi-board.dtb;
E2000# ext4load mmc 1:1 0x90100000 Image;
E2000# booti 0x90100000 - 0x90000000
```

更新 U-boot 配置自动启动

```
E2000# setenv bootargs console=ttyAMA1,115200  audit=0
earlycon=pl011,0x2800d000 root=/dev/mmcblk1p2 rootdelay=3 rw;
E2000# setenv bootcmd 'mmc dev 1; ext4load mmc 1:1 0x90000000
e2000d-chillipi-board.dtb; ext4load mmc 1:1 0x90100000 Image;booti
0x90100000 - 0x90000000;'
E2000# saveenv
```

3.2 USB 介质启动配置

```
C
E2000# setenv bootargs console=ttyAMA1,115200  audit=0
earlycon=pl011,0x2800d000 root=/dev/sda2 rootdelay=3 rw;
E2000# usb start
E2000# ext4load usb 0:1 0x90000000 e2000d-chillipi-board.dtb;
E2000# ext4load usb 0:1 0x90100000 Image;
E2000# booti 0x90100000 - 0x90000000
```

更新 U-boot 配置自动启动

```
C
E2000# setenv bootargs console=ttyAMA1,115200  audit=0
earlycon=pl011,0x2800d000 root=/dev/sda2 rootdelay=3 rw;
E2000# setenv bootcmd 'usb start; ext4load usb 0:1 0x90000000
e2000d-chillipi-board.dtb; ext4load usb 0:1 0x90100000 Image;booti
0x90100000 - 0x90000000;'
E2000# saveenv
```

4 内核与文件系统编译

4.1 基于 phytium-linux-kernel 编译 E2000 内核

1、获取内核源码

在主机 Ubuntu 环境内，从 gitee.com 获取内核代码

```
git clone https://gitee.com/phytium_embedded/phytium-linux-kernel.git
```

2、配置内核

进入内核源码目录

```
make e2000_defconfig
```

3、编译内核

```
make -jn
```

4、编译后的结果

编译时间大概是 50 分钟，编译成功后可以得到设备树文件和内核镜像

arch/arm64/boot/dts/phytium/e2000d*.dtb 为设备树镜像

arch/arm64/boot/Image 为 Linux 内核镜像

4.2 基于 phytium-linux-buildroot 编译内核及文件系统

1、安装编译依赖包

1）主机环境： X86 + Ubuntu20.04

2）检查已安装交叉编译器

3）安装依赖包

```
sudo apt install debootstrap qemu-system-common qemu-user-static binfmt-support autoconf automake libtool fuse debhelper findutils autotools-dev pkg-config libltdl-dev bison flex openssl libssl-dev git
```

2、获取 buildroot 项目代码

```
git clone https://gitee.com/phytium_embedded/phytium-linux-buildroot.git
```

3、配置

1）phytium-linux-buildroot 提供了 3 种文件系统

```
方式一：phytium Linux
$ make phytium_e2000_defconfig
方式二：Ubuntu 带桌面
$ make phytium_e2000_ubuntu_desktop_defconfig
方式三：Debian 不带桌面
$ make phytium_e2000_debian_defconfig
```

2）这里我们以编译 Debian11 为例

```
make phytium_d2000_debian_defconfig
```

3、编译

```
make -jn
```

4、编译结果

编译时间较长，可能为十几个小时甚至更久，编译时长取决于网络状况以及主机性能，编译成功后

output/images/Image Linux 内核镜像

output/images/e2000d*.dtb 为设备树镜像

output/images/rootfs.tar 为根文件系统

5 通过 TFTP 与 NFS 进行开发

在我们开发过程中有可能需要经常更改开发板的软件，最为经常更换的会是驱动及应用程序，如果进行内核开发调试的还需要频繁更换内核。如果每次都通过 TF 卡或 U 盘传输更改效率较低，我们可以采取 TFTP 方式加载内核，NFS 方式加载文件系统或者传输文件。

5.1 TFTP 简介

TFTP（Trivial File Transfer Protocol，简单文件传输协议），是一个基于 UDP 协议实现的用于在客户机和服务器之间进行简单文件传输的协议，适合于开销不大、不复杂的应用场合。TFTP 协议专门为小文件传输而设计，只能从服务器上获取文件，或者向服务器写入文件，不能列出目录，也不能进行认证。

5.2 TFTP 服务器搭建

这里，我们以 X86 PC 主机机可以作为 TFTP 服务端，开发板作为客户端。安装过程我们以 Ubuntu20.04 为例：

1、安装

```
sudo apt-get install tftp-hpa tftpd-hpa xinetd
```

2、配置

```
sudo vi /etc/default/tftpd-hpa
```

注意下列变量配置

TFTP_USERNAME="tftp"

TFTP_DIRECTORY="/tftpboot"

TFTP_ADDRESS="0.0.0.0:69"

TFTP_OPTIONS="-l -c -s"

3、创建 tftp 的路径

需要 tftp 传输的文件需要至于此路径，路径由配置文件 TFTP_DIRECTORY 定义，此路径要保证 777 权限。

```
sudo mkdir /tftpboot
sudo chmod -R 777
```

4、重启 tftp 服务并设置为开机自启动

```
sudo systemctl restart tftpd-hpa
sudo systemctl enable tftpd-hpa
```

```
或者
sudo systemctl stop tftpd-hpa
sudo systemctl start tftpd-hpa
sudo systemctl enable tftpd-hpa
```

5.3 测试 TFTP 传输文件

首先在/tftpboot 目录生成测试文件'ok' 然后通过 tftp 协议将其下传

```
touch /tftpboot
linux@ubuntu:~$ tftp localhost
tftp> get ok
tftp> quit
linux@ubuntu:~$ ls ok
ok
```

5.4 NFS 简介

网络文件系统，英文 Network File System(NFS)，是由 SUN 公司研制的 UNIX 表示层协议(presentation layer protocol)，能使用者访问网络上别处的文件系统就像使用自己的计算机一样。NFS 是基于 UDP/IP 协议的应用，它是当前主流异构平台共享文件系统之一。与前述协议类似，NFS 也分为服务端及客户端。

5.5 NFS 服务搭建

安装服务端

```
sudo apt-get install nfs-kernel-server
```

5.7 配置 NFS

```
sudo vim /etc/exports
```

在文件的最后添加想要共享的目录

/user/nfsroot *(rw,sync,no_root_squash,insecure)

5.8 重新加载

```
sudo exportfs -rv
```

5.6 测试 NFS

开发板如果没有获取 IP 地址则需要执行下述命令获取 IP 地址

```
sudo dhclient
```

在开发板安装 nfs 客户端

```
sudo apt-get install nfs-common
```

挂载 NFS 网络文件系统，注意此处 IP 地址需替换为服务端实际 IP 地址

```
sudo mount -t nfs 192.168.1.10:/user/nfsroot /mnt
ls /mnt
```

6 开发板通过 TFTP + NFS 方式从网络启动

首先按照上述章节配置好主机的 TFTP 服务，及 NFS 服务。将内核文件及设备树文件拷贝至主机的/tftpboot 目录下，将跟文件系统解压至主机的 NFS 共享目录的子目录下，方法过程参考 2.2 及 2.3 节。

进入 U-boot，执行下述命令配置配置 TFTP、NFS 启动

```
setenv bootargs 'noinitrd console=ttyAMA1,115200 root=/dev/nfs rw
nfsroot=10.10.70.186:/nfsroot/phy-root/,proto=tcp,nfsvers=3
ip=10.10.70.191:10.10.70.191:10.10.70.1:255.255.255.0::eth0:off'
setenv serverip 192.168.1.10; （服务器 IP，按照实际替代）
setenv ipaddr 192.168.1.12;（目标板 IP，按照实际替代）
setenv bootcmd 'tftpboot 0x90100000 Image;tftpboot 0x90000000 e2000q-chillipi.dtb;booti
0x90100000 - 0x90000000;'
printenv (检查设置)
Saveenv(保存设置)
```

配置完成后目标板重上电，完成从网络加载内核与文件系统。