LOONGSON

龙芯2K300软件用户手册

2024年8月

手册版本：

龙芯中科技术股份有限公司

V0.

版权声明

免责声明

本文档仅提供阶段性信息，所含内容可根据产品的实际情况随时更新，恕不另行通知。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

龙芯中科技术股份有限公司

Loongson Technology Corporation Limited

地址：北京市海淀区中关村环保科技示范园龙芯产业园 2 号楼

Building No.2, Loongson Industrial Park,
Zhongguancun Environmental Protection Park, Haidian District, Beijing

电话(Tel)：010-
62546668

传真(Fax)：010-
62600826

阅读指南

《龙芯2K300用户手册》主要介绍龙芯 2K300LA相关软件及源码的使用、编译。帮助用户了解和快速搭建应用环境。所涉及包括：u-boot源码编译、Linux内核源码编译、文件系统制作、交叉工具链、EJTAG软件使用。

目录

1. 文档范围

本文是龙芯2K300的用户手册，和该手册一起的包括：u-boot源码、Linux内核源码、文件系统、交叉工具链、EJTAG软件、龙芯2K300LA处理器用户手册。本文主要说明上述软件及源码的使用、编译。

本文档涉及的龙芯软件开发环境均在linux环境下进行，且均为交叉编译环境说明，本手册使用主机CPU架构为：x86_64；操作系统为：ubuntu 22.04。

2. 搭建编译环境

解压交叉编译工具：

```
1| tar -xvf CLFS-loongarch64-8.1-x86_64-cross-tools-gcc-glibc.tar.xz
```

将解压后交叉编译工具移动到“/sur/local/”目录下，并将交叉编译工具文件夹重命名为“loongson-gnu-toolchain-13.2”：

```
1| sudo mv cross-tools /usr/local/loongson-gnu-toolchain-13.2
```

将交叉编译工具路径添加到“PATH”环境变量中：

```
# 将下面命令添加到“~/.bashrc”文件最后
1| export PATH=${PATH}:/usr/local/loongson-gnu-toolchain-13.2/bin
# 执行下面命令使修改后的“~/.bashrc”文件生效
2| source ~/.bashrc
```

执行“loongarch64-unknown-linux-gnu-gcc --version”命令，输出如下信息证明交叉编译工具安装成功：

```
1| loongarch64-unknown-linux-gnu-gcc --version
loongarch64-unknown-linux-gnu-gcc (GCC) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is
NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
```

3. u-boot

3.1 编译u-boot

1） 安装编译依赖

```
1| sudo apt install bison flex libssl-dev libncursesw5-dev
```

2） u-boot源码解压

将源码压缩包拷贝到工作目录，使用下述命令对源码进行解压。

```
1| tar -zxvf u-boot-la.tar.gz
```

注：如果使用的工作环境是虚拟机，请不要直接在共享文件夹下进行解压。

3） 修改配置文件

u-boot编译配置文件为“configs/loongson_2k0300_defconfig”需要将该文件的“CONFIG_SPI_FLASH_XTX”配置使能：

```
CONFIG_SPI_FLASH_XTX=y
```

4） 手动编译

```
1| make CROSS_COMPILE=loongarch64-unknown-linux-gnu- loongson_2k0300_defconfig
2| make CROSS_COMPILE=loongarch64-unknown-linux-gnu-
```

- loongson_2k0300_defconfig：2k300的编译配置文件；
- CROSS_COMPILE：用于指定编译工具前缀；
- loongarch64-unknown-linux-gnu-：交叉编译工具前缀名；

5） 脚本编译

编译脚本如下，在u-boot源码目录下执行：

```
# !/bin/bash

make CROSS_COMPILE=loongarch64-unknown-linux-gnu- loongson_2k0300_defconfig
make CROSS_COMPILE=loongarch64-unknown-linux-gnu-
```

编译完成后在u-boot源码目录下会生成多种bin文件，其中u-boot-spl-gz.bin是要烧录的文件。

3.2 u-boot烧录

linux下烧写器程序为ejtag-debug.tar.gz，直接解压即可使用。将需要烧录的u-boot-spl-gz.bin放入ejtag目录。进入调试器目录输入如下命令：

```
1| sudo ./la_dbg_tool_usb -t
2| source configs/config.ls2k300
3| loop -1 stop # 先执行该命令再上电
4| `Ctrl+c`    # 停止 stop
5| set         # csrc-ebase 值应为:0x900000001c001000
6| program_cachelock ./u-boot-spl-gz.bin
```

4. Linux

4.1 编译Linux

目前久久派有两个版本分别为：“久久派wifi版”和“久久派tf卡版”。不同版本编译方法略有不同，编译之前先确定板卡是哪个版本，两个版本差异如图 2 所示：

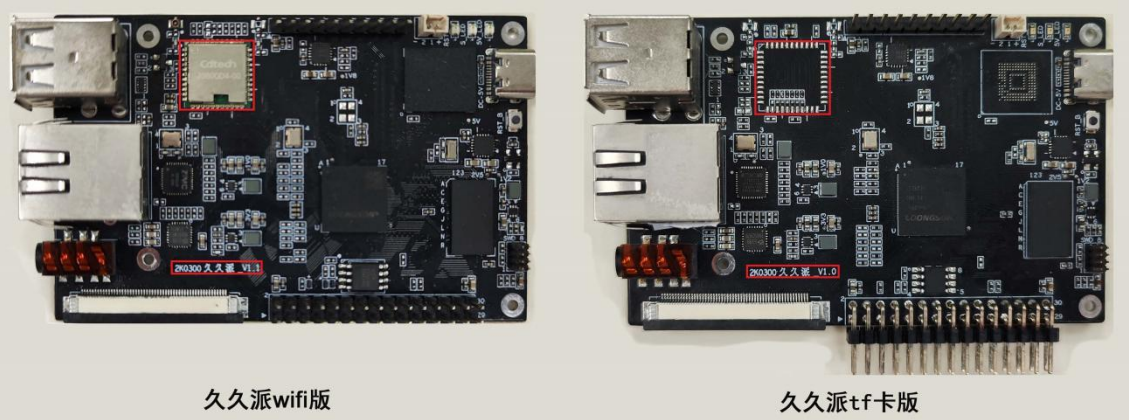

图 2 久久派差异

1） 安装编译依赖

如果搭建u-boot编译环境时已经安装依赖，则此步骤可跳过。

```
1| sudo apt install bison flex libssl-dev libncursesw5-dev
```

2） u-boot源码解压

将源码压缩包拷贝到工作目录，使用下述命令对源码进行解压。

```
1| tar -zxvf linux-6.9.tar.gz
```

注：如果使用的工作环境是虚拟机，请不要直接在共享文件夹下进行解压。

3） 手动编译

久久派wifi版：

```
1| make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu- ls2k0300_99_pai_wifi_defconfig
2| make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu-
```

久久派tf卡版：

```
1| make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu- ls2k0300_99_pai_tfcard_defconfig
2| make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu-
```

- ARCH：用于指定编译架构；
- CROSS_COMPILE：用于指定编译工具前缀；
- loongarch64-unknown-linux-gnu-：交叉编译工具前缀名；
- ls2k0300_99_pai_wifi_defconfig：久久派wifi版编译配置文件；
- ls2k0300_99_pai_tfcard_defconfig：久久派tf卡版编译配置文件；

4） 脚本编译

编译脚本如下，在linux-6.9源码目录下执行：

久久派wifi版：

```
# !/bin/bash

make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu- ls2k0300_99_pai_wifi_defconfig
make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu-
```

久久派tf卡版：

```
# !/bin/bash
```

```
make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu-
ls2k0300_99_pai_tfcard_defconfig
make ARCH=loongarch CROSS_COMPILE=loongarch64-unknown-linux-gnu-
```

编译完成后会生成“arch/loongarch/boot/vmlinux.bin”文件，该文件就是后面烧录要用到的二进制文件。同时源码目录下还会生成System.map文件，该文件记录了所有符号的运行地址，后文中制作uImage镜像需要的内核入口地址可以从该文件获取。

5. buildroot

5.1 编译buildroot

1） 解压buildroot.tar.gz文件

```
1| tar -zxvf buildroot.tar.gz
```

2） 设置编译配置文件

```
1| cd buildroot
2| cp ./config_systemd .config
```

3） 配置交叉编译工具链

打开图形化配置界面：

```
1| make ARCH=loongarch64 menuconfig
```

使用方向选中“Toolchain”选项，回车进入Toolchain配置界面：

```
    Toolchain type (External toolchain)  --->
    *** Toolchain External Options ***
    Toolchain (Custom toolchain)  --->
    Toolchain origin (Pre-installed toolchain)  --->
(/opt/loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-rc1.3-1) Toolchain path
($(ARCH)-linux-gnu) Toolchain prefix
    External toolchain gcc version (8.x)  --->
    External toolchain kernel headers series (4.19.x)  --->
    External toolchain C library (glibc)  --->
```

对Toolchain作如下修改：

- Toolchain path：修改为“/usr/local/loongson-gnu-toolchain-13.2”。

- Toolchain prefix：修改为“$(ARCH)-unknown-linux-gnu”。
- External toolchain gcc version：修改为“13.x”。
- External toolchain kernel headers series：修改为“6.5.x”。
- Toolchain has RPC support：改为“n”。

修改完成后如下所示：

```
    Toolchain type (External toolchain)  --->
    *** Toolchain External Options ***
    Toolchain (Custom toolchain)  --->
    Toolchain origin (Pre-installed toolchain)  --->
(/usr/local/loongson-gnu-toolchain-13.2) Toolchain path
($(ARCH)-unknown-linux-gnu) Toolchain prefix
    External toolchain gcc version (13.x)  --->
    External toolchain kernel headers series (6.5.x)  --->
    External toolchain C library (glibc)  --->
[*] Toolchain has SSP support?
[*]   Toolchain has SSP strong support?
[ ] Toolchain has RPC support?
[*] Toolchain has C++ support?
[ ] Toolchain has D support?
```

保存并退出

4） 编译文件系统

主机在联网状态下执行一下命令进行编译：

```
1| make
```

编译完成后，在buildroot源码的output/images/目录下会生成文件系统镜像文件。

```
1| ls output/images
rootfs.cpio  rootfs.tar  rootfs.tar.gz
```

6. 启动与烧录

6.1 准备工作

6.1.1 制作uImage

1） 安装u-boot-tools

```
1| sudo apt install u-boot-tools
```

2） 制作vmlinux.bin.lzma

```
1| cd linux-6.9
2| cp arch/loongarch/boot/vmlinux.bin  ./
3| lzma -k vmlinux.bin
```

3） 编写镜像描述文件multi.its

```
/*
 * U-Boot uImage source file with multiple kernels and ramdisks blobs
 */

/dts-v1/;

/ {
        description = "Various kernels and ramdisks blobs";
        #address-cells = <2>;

        images {
                kernel-1 {
                        description = "vmlinux";
                        data = /incbin/("vmlinux.bin.lzma");
                        type = "kernel";
                        arch = "loongarch";
                        os = "linux";
                        compression = "lzma";
                        load = <0x90000000 0x00200000>;
                        entry = <0x90000000 0x01317000>;
                };
        };

        configurations {
                default = "config-1";

                config-1 {
                        description = "vanilla-2.6.23 configuration";
                        kernel = "kernel-1";
                        loadables = "kernel-1";
                };

        };
};
```

● Images>kernel-1>data：指定vmlinux.bin.lzma文件路径（本手册镜

像制作路径与vmlinux.bin.lzma在同一路径）。

- Images>kernel-1>entry：指定内核入口地址，入口地址可在“linux-6.9/System.map”文件中搜索“kernel_entry”关键字查看，如下所示：

```
1| cat System.map | grep kernel_entry
9000000001317000 T kernel_entry
9000000001317114 T kernel_entry_end
```

4） 生成uImage

```
mkimage -f multi.its uImage
```

执行完成后，会在当前目录下生成uImage镜像文件。

6.1.2 制作根文件系统U盘

1） 格式化U盘

将U盘格式化为FAT或ext4文件系统（推荐格式化为ext4文件系统）。

2） 根文件系统解压至U盘

```
1| cd buildroot/output/images/
2| sudo tar -xvf rootfs.tar -C /media/user/U-disk/
3| sudo umount /media/user/U-disk
```

6.2 TFTP启动

注：该启动方式主机需搭建TFTP服务器，并将uImage文件拷贝至TFTP服务路径下。

1） 插入根文件系统U盘

将制作好的根文件系统U盘，插入如图 3所示的USB A口：

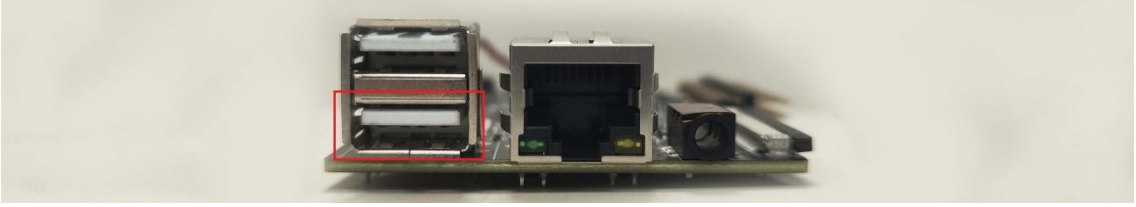

图 3 USB接口

2) 启动板卡

将久久派UART0和网口连接至主机并启动板卡。

3) 配置环境变量

进入u-boot命令行后进行如下配置：

a. 配置IP地址

```
1| setenv ipaddr 192.168.0.20
```

b. 配置TFTP服务器IP地址（演示环境的主机IP为：192.168.0.30）

```
2| setenv serverip 192.168.0.30
```

c. 配置加载文件名

```
3| setenv bootfile uImage
```

d. 配置启动参数（root设为U盘）

```
4| setenv bootargs root=/dev/sda1 rootdelay=5
```

注：U盘的设备文件不一定为“/dev/sda1”，请以实际U盘设备文件名为准。

e. 保存环境变量

```
5| saveenv
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI flash...done
```

4) 手动启动系统

环境变量设置完成后在命令行执行“tftpboot”即可启动系统：

```
1| tftpboot
Speed: 1000, full duplex
Using ethernet@0x16020000 device
TFTP from server 192.168.0.10; our IP address is 192.168.0.30
Filename 'uImage'.
Load address: 0x9000000003000000
... ...
Welcome to Buildroot
buildroot login:
```

5） 自动启动系统

进入u-boot命令行，将“bootcmd”环境变量配置为“tftpboot”：

```
1| setenv bootcmd tftpboot
2| saveenv
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI
flash...done
```

6.3 U盘启动

1） 拷贝uImage

将“uImage”文件拷贝至U盘的“boot”目录下。

```
1| cd linux-6.9
2| sudo cp uImage /media/user/U-disk/boot/
```

2） 手动启动系统

插入U盘并启动，进入u-boot命令行后执行如下命令：

```
1| ext4load usb 0:1 ${loadaddr} boot/uImage
2| bootm ${loadaddr}
```

3） 自动启动系统

进入u-boot命令行，对“bootcmd”环境变量做如下配置：

```
1| setenv bootcmd 'ext4load usb 0:1 ${loadaddr} boot/uImage;
bootm ${loadaddr}'
2| saveenv
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI
flash...done
```

6.4 EMMC/SD卡启动

本手册以EMMC为例介绍EMMC/SD卡启动方法。

1） 进入系统

将“uImage”和“rootfs.tar”拷贝至U盘的“boot”目录下，然后使用U

盘启动或TFTP启动进入系统命令行：

```
Welcome to Buildroot
buildroot login: root
#
```

2） 初始化EMMC

查看磁盘信息：

```
1| # fdisk -l
... ...
Disk /dev/mmcblk0: 7456 MB, 7818182656 bytes, 15269888
sectors
238592 cylinders, 4 heads, 16 sectors/track
Units: sectors of 1 * 512 = 512 bytes
... ...
```

创建分区：

```
1| fdisk /dev/mmcblk0

The number of cylinders for this disk is set to 238592.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): n          # 创建分区
Partition type
   p   primary partition (1-4)
   e   extended
p                                # 主分区模式
Partition number (1-4): 1        # 创建分区数
First sector (16-15269887, default 16):
Using default value 16
Last sector or +size{,K,M,G,T} (16-15269887, default 15269887):
Using default value 15269887
```

```
Command (m for help): wq        # 保存并退出

The partition table has been altered.
Calling ioctl() to re-read partition table
```

格式化并挂载EMMC：

```
1| mke2fs -c /dev/mmcblk0p1
2| mkdir /media/mmc
3| mount /dev/mmcblk0p1 /media/mmc
```

3) EMMC制作为根文件系统

```
1| tar -xvf /boot/rootfs.tar -C /media/mmc
2| cp /boot/uImage /media/mmc/boot/
```

4) 手动启动系统

```
1| ext4load mmc 0:1 ${loadaddr} boot/uImage
2| setenv bootargs root=/dev/mmcblk0p1 rootdelay=5
3| bootm ${loadaddr}
```

5) 自动启动系统

进入u-boot命令行，对“bootcmd”环境变量做如下配置：

```
1| setenv bootcmd 'ext4load mmc 0:1 ${loadaddr} boot/uImage;
bootm ${loadaddr}'
2| setenv bootargs root=/dev/mmcblk0p1 rootdelay=5
3| saveenv
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI
flash...done
```