U-Boot

Boot

U-Boot and OpenSBI development and debugging guide.

uboot function and configuration

Features

The main functions of uboot are as follows:

Load boot kernel

uboot loads the kernel image from the storage medium (emmc/sd/nand/nor/ssd, etc.) to the specified location in the memory, and starts the kernel.

fastboot flash function
Use the fastboot tool to burn the image to the specified partition location.

Boot logo

During the uboot startup phase, the startup logo and boot menu are displayed.

Driver debugging

Debug device drivers based on uboot, such as mmc/spi/nand/nor/nvme and other drivers. uboot provides a shell command line to debug the functions of each driver.

The uboot driver is in the drivers/ directory.

Windows Compile

This chapter introduces the compilation and generation of uboot image files based on the uboot code environment.

Compile configuration
When compiling for the first time, or if you need to choose another solution, you need to select the compilation configuration first. Here we take k1 as an example:

cd ~/uboot-2022.10
make ARCH = riscv k1_defconfig -C ~/uboot-2022.10/
Visually change the compilation configuration:

make ARCH = riscv menuconfig

Use the keyboard "Y"/"N" to turn on/off related function configurations. After saving, it will be updated to the .config file in the uboot root directory.

Compile uboot
cd ~/uboot-2022.10

GCC_PREFIX = riscv64-unknown-linux-gnu-
make ARCH = riscv CROSS_COMPILE = ${ GCC_PREFIX } -C ~/uboot-2022.10 -j4
Compile product
~/uboot-2022.10 $ ls u-boot * -l
 u-boot
u-boot.bin   # uboot mirror
 u-boot.dtb   # dtb file
 u-boot-dtb.bin   # uboot mirror with dtb
 u-boot.itb   # Package u-boot-nodtb.bin and dtb into fit format
 u-boot-nodtb.bin
bootinfo_emmc.bin   # Used to record the spl location information when emmc starts
 bootinfo_sd.bin
bootinfo_spinand.bin
bootinfo_spinor.bin
FSBL.bin   # u -boot-spl.bin plus header information. Loaded and started by brom
 k1-x_deb1.dtb   # The dtb file of scheme deb1
 k1-x_spl.dtb   # The dtb file of spl
dts configuration
The uboot dts is configured in the directory uboot-2022.10/arch/riscv/dts/. Modify the dts of the scheme according to different schemes, such as the deb1 scheme.

~/uboot-2022.10 $ ls arch /riscv/dts/k1 * .dts -l
arch /riscv/dts/k1-x_deb1.dts
arch /riscv/dts/k1-x_deb2.dts
arch /riscv/dts/k1-x_evb .dts
arch /riscv/dts/k1-x_fpga_1x4.dts
arch /riscv/dts/k1-x_fpga_2x2.dts
arch /riscv/dts/k1-x_fpga.dts
arch /riscv/dts/k1-x_spl.dts
uboot driver development and debugging
This chapter mainly introduces the driver usage and debugging methods of uboot. By default, all drivers have been configured.

Linux Cross Compiling

Install the following packages on your Linux box.

sudo apt install build-essential ncurses-dev bison flex bc libssl-dev
sudo apt install gnulib autopoint gettext gperf libtool texinfo
sudo apt install gawk fuse libdevmapper-dev

First thing you need to do is download the RISC-V tool chain. Build it and install somewhere you can reference where it's installed.

On my machine it's installed from the root directory in /opt

I added the following lines to my .bashrc

Building a RISCV Kernel

Go to kerel.org and download a stable kernel.
Make a directorty (Not in Downloads). Remember building a kernel needs space to compile.
Once extracted set the environment

export RISCV=/opt/riscv/
export CCPREFIX=riscv64-linux-gnu-

In the /boot directory of your RISC machine there's a config file. Copy this file to the root directory you extracted the kernel to. Rename the configuation file to .config

If all was installed properly you should run a make menuconfig

When this appears on the screen it should have loaded the setting of the .config file and be the configuration of your RISC boards kernel.

Just exit and save the .config which won't change anything. The command below will run single threaded unless you have more CPU cores on your machine. Lets say you have 4 cores or your machine and 8 threads. You should compile the kernel with the -j option and the maximum number of threads so the compile will not take forever.

I'm running on an AMD Threadripper with 32 threads so I specify -j 32

make ARCH=riscv mrproper defconfig
which will make the .config

boot kernel

This section introduces the uboot startup kernel, as well as the custom configuration and startup of partitions.

After the development board is powered on, immediately press the "s" key on the keyboard to enter the uboot shell.

You can enter fastboot mode by executing fastboot 0, and send the image to the development board through fastboot stage Image on the computer. (Or other ways to download the image, such as fatload and other commands)
Execute booti to start the kernel (or bootm to start the fit format image)

#Download kernel image

```
=> fastboot -l 0x40000000 0
Starting download of 50687488 bytes
...
downloading/uploading of 50687488 bytes finished

#Computer execution command
 C: \U sers>fastboot stage Z: \k 1 \o utput \I mage
Sending 'Z:\k1\output\Image'  ( 49499 KB )            OKAY [   1.934s]
Finished. Total time : 3.358 s
```

#After the download is completed, in the uboot shell, enter CTRL+C on the keyboard to exit fastboot mode.

#Download dtb

```
=> fastboot -l 0x50000000 0
Starting download of 33261 bytes

downloading/uploading of 33261 bytes finished

#Execute the command on the computer
 C: \U sers>fastboot stage Z: \k 1 \o utput \k 1-x_deb1.dtb
Sending 'Z:\k1\output\k1-x_deb1.dtb'  ( 32 KB )       OKAY [   0.004 s]
```

Finished. Total time : 0.054s
Execute startup command

=> booti 0x40000000 - 0x50000000
Moving Image from 0x40000000 to 0x200000, end = 3d4f000
Flattened Device Tree blob at 50000000
   Booting using the fdt blob at 0x50000000
  Using Device Tree in place at 00000000500000 00, end 0000000050014896

Starting kernel...

[0.000000] Linux version 6.1.15+ ... ...
[0.000000] OF: fdt: Ignoring memory range 0x0 - 0x200000
[0.000000] Machine model: spacemit k1-x deb1 board
[0.000000] earlycon: sbi0 at I/O port 0x0 (options '')
[0.000000] printk: bootconsole [sbi0] enabled
Start the fit format image through the bootm command
Assume that partition 5 in emmc is a fat32 file system. And save the uImage.itb file inside, load and start the kernel through the following command.

=> fatls mmc 2:5
sdh@d4281000: 74 clk wait timeout (100)
 50896911 uImage.itb
    4671 env_k1-x.txt

2 file (s) , 0 dir (s)

=> fatload mmc 2:5 0x40000000 uImage.itb
50896911 bytes read in 339 ms (143.2 MiB/s)
=> bootm 0x40000000
Loading kernel from FIT Image at 40000000 ...
 Boot from fit configuration k1_deb1
  Using 'conf_2' configuration
  Trying 'kernel' kernel subimage
    Description: Vanilla Linux kernel
    Type: Kernel Image
    Compression: uncompressed
    Data Start: 0x400000e8
    Data Size: 50687488 Bytes = 48.3 MiB
    Architecture: RISC-V
    OS: Linux
    Load Address: 0x01400000
    Entry Point: 0x01400000
  Verifying Hash Integrity .. . OK
Loading fdt from FIT Image at 40000000 ...
   Using 'conf_2' configuration

```
  Trying 'fdt_2' fdt subimage
    Description: Flattened Device Tree blob for k1_deb1
    Type: Flat Device Tree
    Compression: uncompressed
    Data Start: 0x43067c90
    Data Size: 68940 Bytes = 67.3 KiB
    Architecture: RISC-V
    Load Address: 0x28000000
  Verifying Hash Integrity ... OK
  Loading fdt from 0x43067c90 to 0x28000000
  Booting using the fdt blob at 0x28000000
  Loading Kernel Image
  Using Device Tree in place at 0000000028000000, end 000 0000028013d4b
```

Starting kernel...

```
[     0.000000] Linux version 6.1.15+ ... ...
[     0.000000] OF: fdt: Ignoring memory range 0x0 - 0x1400000
[     0.000000] Machine model: spacemit k1-x deb1 board
[     0.000000] earlycon: sbi0 at I/O port 0x0 ( options '' )
[     0.000000] printk: bootconsole [ sbi0] enabled
```

env

This chapter describes how to configure the env to be loaded from the specified storage medium during the uboot startup phase.

Execute make menuconfig and enter Environment.

Currently supported optional media are mmc and mtd devices (mtd devices include spinor and spinand).

The offset address of env needs to be determined according to the configuration of the partition table. For details, please refer to the partition table configuration in the flash startup settings chapter. The default is 0x80000.

```
( 0x80000 ) Environment address       #spinor’s env offset address
( 0x80000 ) Environment offset        #mmc device’s env offset address
```

mmc

Both emmc and sd cards use the mmc driver, and the dev numbers are 2 and 0 respectively.

config configuration

Execute make menuconfig, enter Device Drivers--->MMC Host controller Support, and

enable the following configuration

dts configuration
//uboot-2022.10/arch/riscv/dts/k1-x.dtsi
sdhci0 : sdh @ d4280000 { compatible = "spacemit,k1-x-sdhci" ; reg = < 0x0 0xd4280000 0x0 0x200 > ; interrupt - parent = <& intc > ; interrupts = < 99 > ; resets = <& reset RESET_SDH_AXI > , <& reset RESET_SDH0 > ; reset - names = "sdh_axi" , "sdh0" ; clocks = <& ccu CLK_SDH0 > , <& ccu CLK_SDH_AXI > ; clock - names = "sdh-io" , "sdh-core" ; status = "disabled" ; };

sdhci2 : sdh @ d4281000 {
compatible = "spacemit,k1-x-sdhci" ; reg = < 0x0 0xd4281000 0x0 0x200 > ; interrupt - parent = <& intc > ; interrupts = < 101 > ; resets = <& reset RESET_SDH_AXI > , <& reset RESET_SDH2 > ; reset - names = "sdh_axi" , "sdh2" ; clocks = <& ccu CLK_SDH2 > , <& ccu CLK_SDH_AXI > ; clock - names = "sdh-io" , "sdh-core" ; status = "disabled" ; };

//uboot-2022.10/arch/riscv/dts/k1-x_deb1.dts
& sdhci0 {
pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_mmc1 & gpio80_pmx_func0 > ; bus - width = < 4 > ; cd - gpios = <& gpio 80 0 > ; cd - inverted ; cap - sd - highspeed ; sdh - phy - module = < 0 > ; status = "okay" ; };

/* eMMC */
& sdhci2 {
bus - width = < 8 > ; non - removable ; mmc - hs400 - 1 _8v ; mmc - hs400 - enhanced - strobe ; sdh - phy - module = < 1 > ; status = "okay" ; };

Debugging verification
uboot shell provides command line debugging mmc driver, you need to enable the compilation configuration item CONFIG_CMD_MMC

=> **mmc list**
sdh@d4280000: 0 (SD)
sdh@d4281000: 2 (eMMC)
=> mmc dev 2 #switch to emmc
switch to partitions #0, OK
mmc2 (part 0) is current device

#read 0x1000 blk_cnt at offset 0 to memory 0x40000000
=> mmc read 0x40000000 0 0x1000

MMC read : dev # 2, block # 0, count 4096 ... 4096 blocks read: OK

#Write from memory address 0x40000000 to 0x1000 blk_cnt to 0 offset

=> mmc write 0x40000000 0 0x1000

MMC write: dev # 2, block # 0, count 4096 ... 4096 blocks written: OK

#For other usage, please refer to mmc -h
Common interfaces
Refer to the interface in cmd/mmc.c

nvme

The nvme driver is mainly used to debug SSD hard drives.

config configuration
Execute make menuconfig, enter Device Driver, and enable the following configuration:

dts configuration
//uboot-2022.10/arch/riscv/dts/k1-x.dtsi
pcie1_rc : pcie @ ca400000 { compatible = "k1x,dwc-pcie" ; reg = < 0x0 0xca400000 0x0 0x00001000 > , /* dbi */ < 0x0 0xca700000 0x0 0x0001ff24 > , /* atu registers */ < 0x0 0x90000000 0x0 0x00100000 > , /* config space */ < 0x0 0xd4282bd4 0x0 0x00000008 > , /*k1x soc config addr*/ < 0x0 0xc0 c20000 0x0 0x00001000 > , /* phy ahb * / < 0x0 0xc0c10000 0x0 0x00001000 > , /* phy addr */ < 0x0 0xd4282bcc 0x0 0x00000008 > , /* conf0 addr */ < 0x0 0xc0b10000 0x0 0x00001000 > ; /* phy0 addr */ reg - names = "dbi" , "atu " , "config" , "k1x_conf" , "phy_ahb" , "phy_addr" , "conf0_addr" , "phy0_addr" ;

k1x , pcie - port = < 1 > ;
clocks = <& ccu CLK_PCIE1 > ; clock - names = "pcie-clk" ; resets = <& reset RESET_PCIE1 > ; reset - names = "pcie-reset" ;
bus - range = < 0x00 0xff > ;
max - link - speed = < 2 > ; num - lanes = < 2 > ; num - viewport = < 8 > ; device_type = "pci" ; #address-cells = <3>; #size-cells = <2>; ranges = < 0x01000000 0x0 0x90100000 0 0x90100000 0x0 0x100000 > , < 0x02000000 0x0 0x90200000 0 0x90200000 0x0 0x0fe00000 > ;

interrupts = < 142 > , < 146 > ;
interrupt - parent = <& intc > ; #interrupt-cells = <1>; interrupt - map - mask = < 0 0 0 0x7 > ; interrupt - map = < 0000 0 0 1 & pcie1_intc 1 > , /* int_a */ < 0000 0 0 2 & pcie1_intc 2 > , /* int_b */ < 0000 0 0 3 & pcie1_intc 3 > , /* int_c */ < 0000 0 0 4 & pcie1_intc 4 > ; /* int_d */ linux , pci - domain = < 1 > ; status = "disabled" ; pcie1_intc: interrupt - controller @ 0 { interrupt - controller ; reg = < 0 0 0 0 0 > ; #address-cells = <0 >; #interrupt-cells = <1>; }; };

//uboot-2022.10/arch/riscv/dts/k1-x_deb1.dts
& pcie1_rc { pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_pcie1_3 > ; status = "okay" ; };

Debugging verification

You need to enable the compilation configuration CONFIG_CMD_NVME. The debugging method is as follows:

```
=> nvme scan
=> nvme detail
Blk device 0: Optional Admin Command Support:
        Namespace Management/Attachment: no
        Firmware Commit/Image download: yes
         Format NVM: yes
         Security Send/Receive: yes
 Blk device 0: Optional NVM Command Support:
        Reservation : yes
         Save/Select field in the Set/Get features: yes
         Write Zeroes: yes
         Dataset Management: yes
         Write Uncorrectable: yes
 Blk device 0: Format NVM Attributes:
        Support Cryptographic Erase: No
        Support erase a particular namespace: Yes
        Support format a particular namespace: Yes
Blk device 0: LBA Format Support:
        LBA Foramt 0 Support: ( current )
                 Metadata Size: 0
                LBA Data Size: 512
                Relative Performance: Good
Blk device 0: End-to-End DataProtect Capabilities:
        As last eight bytes: No
        As first eight bytes: No
        Support Type3: No
        Support Type2: No
        Support Type1: No
Blk device 0: Metadata capabilities:
        As part of a separate buffer: No
        As part of an extended data LBA: No
=> nvme read /write addr blk_off blk_cnt
Common interfaces
Refer to the code interface in cmd/nvme.c

net
```

config configuration
Execute make menuconfig to enable the following configuration:
dts configuration
//uboot-2022.10/arch/riscv/dts/k1-x.dtsi
eth0 : ethernet @ cac80000 { compatible = "spacemit,k1x-emac" ; reg = < 0x00000000 0xCAC80000 0x00000000 0x00000420 > ; ctrl - reg = < 0x3e 4 > ; dline - reg = < 0x3e8 > ; clocks = <& ccu CLK_EMAC0_BUS > ; clock - names = "emac-clk" ; resets = <& reset RESET_EMAC0 > ; reset - names = "emac-reset" ; status = "disabled" ; };
//uboot-2022.10/arch/riscv/dts/k1-x_deb1.dts
& eth0 { status = "okay" ; pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_gmac0 > ;

    phy - reset - pin = < 110 > ;

    clk_tuning_enable ;
clk - tuning - by - delayline ; tx - phase = < 90 > ; rx - phase = < 73 > ;
    phy - mode = "rgmii" ;
phy - addr = < 1 > ; phy - handle = <& rgmii > ;

    ref - clock - from - phy ;

    mdio {
#address-cells = <0x1>; #size-cells = <0x0>; rgmii: phy @ 0 { compatible = "ethernet-phy-id001c.c916" ; device_type = "ethernet-phy" ; reg = < 0x1 > ; }; }; };

Debugging verification

You need to enable the compilation configuration CONFIG_CMD_NET first, connect the network cable to the network port of the development board, and prepare the tftp server (the method of setting up the tftp server can be found online, and will not be introduced here)

=> dhcp #After executing dhcp, if the address is returned, it means that it is connected to the network server. In other cases, the connection fails
 ethernet@cac80000 Waiting for PHY auto negotiation to complete... done
 emac_adjust_link link :1 speed:1000 duplex:full
BOOTP broadcast 1
BOOTP broadcast 2
BOOTP broadcast 3
BOOTP broadcast 4
BOOTP broadcast 5
BOOTP broadcast 6
BOOTP broadcast 7
DHCP client bound to address 10.0.92.130 (7982 ms)

=> tftpboot 0x40000000 site11/uImage.itb

ethernet@cac80000 Waiting for PHY auto negotiation to complete... done
 emac_adjust_link link :1 speed:1000 duplex:full
Using ethernet@cac80000 device
TFTP from server 10.0.92.134 ; our IP address is 10.0.92.130
Filename 'site11/uImage.itb' .
 Load address: 0x40000000
Loading: ################################# #################################
########   1.1 MiB/s done Bytes transferred = 66900963 (3fcd3e3 hex) =>

#Start kernel
=> bootm 0x40000000
Common interfaces
Refer to the code interface in cmd/net.c

spi
spi only leads to one hardware interface, so it only supports nand or nor flash.

config configuration
Execute make menuconfig, enter Device Drivers, and enable the following configurations

dts configuration
//k1-x.dtsi
/ dts - v1 / ;

/ {
compatible = "spacemit,k1x" , "riscv" ; #address-cells = <2>; #size-cells = <2>;
        soc: soc {
compatible = "simple-bus" ; #address-cells = <2>; #size-cells = <2>; ranges ;
                qspi: spi @ d420c000 {
compatible = "spacemit,k1x-qspi" ; #address-cells = <1>; #size-cells = <0>; reg = <
0x0 0xd420c000 0x0 0x1000 > , < 0x0 0xb8000000 0x0 0xd00000 > ; reg - names =
"qspi-base" , "qspi-mmap" ; qspi - sfa1ad = < 0xa00000 > ; qspi - sfa2ad = < 0xb00000
> ; qspi - sfb1ad = < 0xc00000 > ; qspi - sfb2ad = < 0xd00000 > ; clocks = <& ccu
CLK_QSPI > , <& ccu CLK_QSPI_BUS > ; clock - names = "qspi_clk" , "qspi_bus_clk"
; resets = <& reset RESET_QSPI > , <& reset RESET_QSPI_BUS > ; reset - names =
"qspi_reset" , "qspi_bus_reset" ; qspi - pmuap - reg = < 0xd4282860 > ; spi - max -
frequency = < 26500000 > ; qspi - id = < 4 > ; status = "disabled" ; }; }; };
//k1-x_deb1.dts

& qspi {
status = "okay" ; pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_qspi > ; };

Debugging verification
Enable uboot shell command sspi configuration, CONFIG_CMD_SPI,

Debug command:

sspi - h

"SPI utility command" ,
"[<bus>:]<cs>[.<mode>][@<freq>] <bit_len> <dout> - Send and receive bits \n " "<bus> - Identifies the SPI bus \n " "<cs> - Identifies the chip select \n " "<mode> - Identifies the SPI mode to use \n " "<freq> - Identifies the SPI bus frequency in Hz \n " "<bit_len> - Number of bits to send (base 10) \n " "<dout> - Hexadecimal string that gets sent"

Common interfaces

Refer to the interface in cmd/spi.c

nand
The nand driver is based on spi, so you need to enable the spi driver function first.

config configuration
Execute make menuconfig and enter Device Drivers --->MTD Support

If you need to add a new nand flash, you can add the jedec id of the nand flash according to the supported manufacturer driver.

~/uboot-2022.10 $ ls drivers/mtd/nand/spi/ * .c -l
drivers/mtd/nand/spi/core.c
drivers/mtd/nand/spi/gigadevice.c
drivers/mtd/nand/spi/ macronix.c
drivers/mtd/nand/spi/micron.c
drivers/mtd/nand/spi/other.c
drivers/mtd/nand/spi/toshiba.c
drivers/mtd/nand/spi/winbond.c
For example, add new flash in gigadevice

//uboot-2022.10/drivers/mtd/nand/spi/gigadevice.c
static const struct spinand_info gigadevice_spinand_table [] = { SPINAND_INFO ("GD5F1GQ4UExxG" , 0xd1 , NAND_MEMORG (1 , 2048 , 128 , 64 , 1024 , 1 , 1 , 0 _status)) , SPINAND_INFO (" GD5F1GQ5UExxG " , 0x51 , NAND_MEMORG (1 , 2048 , 128 , 64 , 1024 , 1 , 1 , 1), NAND_ECCREQ (4 , 512), SPINAND_INFO_OP_VARIANTS (& gd5f1gq5_read_cache_variants , & write_cache_variants , & update_cache_variants), 0 , SPINAND_ECCINFO (& gd5fxgqxxexxg_ooblayout , gd5fxgq5xexxg_ecc_get_status)), };

If it is another brand of nand flash, you can refer to the gigadevice driver to re-

implement it.

dts configuration
The nand driver is hung under the spi driver, so dts needs to be configured under the spi node.

```
 & qspi  {
status = "okay" ; pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_qspi > ;

     spi - nand @ 0  {
compatible = "spi-nand" ; reg = < 0 > ; spi - tx - bus - width = < 1 > ; spi - rx - bus - width = < 1 > ; spi - max - frequency = < 6250000 > ; u - boot , dm - spl ; status = "okay" ; }; };
```

Debugging verification

The nand driver can be debugged based on the mtd command

```
=> mtd
mtd - MTD utils

Usage:
mtd - generic operations on memory technology devices

mtd list
mtd read [ .raw][.oob] <name> <addr> [ <off> [ <size>]]
mtd dump[.raw][.oob] <name>         [ <off> [ <size>] ]
mtd write[.raw][.oob][.dontskipff] <name> <addr> [ <off> [ <size>]]
mtd erase[.dontskipbad] <name>          [ <off> [ <size>]]

Specific functions:
mtd bad <name>

With:
        <name>: NAND partition/chip name ( or corresponding DM device name or OF path )
         <addr>: user address from/to which data will be retrieved/stored
        <off>: offset in <name> in bytes ( default : start of the part )
* must be block-aligned for erase * must be page-aligned otherwise          <size>: length of the operation in bytes ( default: the entire device ) * must be a multiple of a block for erase * must be a multiple of a page otherwise ( special case : default is a page with dump )
The .dontskipff option forces writing empty pages, don 't use it if unsure.

=> mtd list
```

[RESET]spacemit_reset_set assert=1, id=77
[RESET]spacemit_reset_set assert=1, id=78
clk qspi_bus_clk already disabled
clk qspi_clk already disabled
ccu_mix_set_rate of qspi_clk timeout
[RESET]spacemit_reset_set assert=0, id=77
[RESET]spacemit_reset_set assert=0, id=78
SF: Detected w25q32 with page size 256 Bytes, erase size 64 KiB, total 4 MiB
Could not find a valid device for spi-nand
List of MTD devices:
* nor0
  - device: flash@ 0
  - parent: spi@d420c000
  - driver: jedec_spi_nor
  - path: /soc/spi@d420c000/flash@0
  - type: NOR flash
  - block size: 0x10000 bytes
  - min I/O: 0x1 bytes
  - 0x000000000000-0x000000400000 : "nor0 "
        - 0x0000000a0000-0x000000100000 : "opensbi"
        - 0x000000100000-0x000000300000 : "uboot"
=> mtd read/write partname addr off size
Common interfaces
Refer to the code interface of cmd/mtd.c

nor
The nor driver is based on the spi driver, so you need to enable the spi driver function first.

config configuration
Execute make menuconfig, enter Device Drivers --->MTD Support --->SPI Flash Support, and turn on the following configuration (enabled by default). The example takes nor flash with winbond enabled as an example.

Add a new spi nor flash:

For the nor flash of the supported manufacturer in the above figure, you can directly enable the corresponding compilation configuration, such as the flash of the gigadevice manufacturer.
The jedec id list of spi flash is maintained in uboot-2022.10/drivers/mtd/spi/spi-nor-ids.c. If there is no specific nor flash jedec id in the list, you can add the jedec id to the list yourself (jedec id is the manufacturer code corresponding to spi flash, you can find the manufac keyword according to the nor flash datasheet, such as winbond is 0xfe)
dts configuration
The nor driver relies on the spi driver interface. Please refer to the spi sub-chapter for the spi driver. You need to add the dts node, as follows:

//k1/uboot-2022.10/arch/riscv/dts/k1-x_deb1.dts
& qspi { status = "okay" ; pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_qspi > ;

flash @ 0 {
compatible = "jedec,spi-nor" ; reg = < 0 > ; spi - max - frequency = < 26500000 > ;
m25p , fast - read ; broken - flash - reset ; status = "okay" ; }; };

Debugging verification
It can be debugged based on the mtd/sf command of the uboot command line. The compilation configuration needs to enable CONFIG_CMD_MTD=y, CONFIG_CMD_SF

Read and write nor flash based on mtd command:

```
=> mtd list
List of MTD devices:
* nor0
  - device: flash@0
  - parent: spi@d420c000
  - driver: jedec_spi_nor
  - path: /soc/spi@d420c000/flash@0
  - type : NOR flash
  - block size: 0x1000 bytes
  - min I/O: 0x1 bytes
  - 0x000000000000-0x000000400000 : "nor0"
          - 0x0000000a0000-0x0000000e0000 : "opensbi"
          - 0x000000100000-0x000000200000 : "uboot "
=>

=> mtd
mtd - MTD utils

Usage:
mtd - generic operations on memory technology devices

mtd list
mtd read [ .raw][.oob] <name> <addr> [ <off> [ <size>]]
mtd dump[.raw][.oob] <name>         [ <off> [ <size>] ]
mtd write[.raw][.oob][.dontskipff] <name> <addr> [ <off> [ <size>]]
mtd erase[.dontskipbad] <name>         [ <off> [ <size>]]

Specific functions:
mtd bad <name>

With:
        <name>: NAND partition/chip name ( or corresponding DM device name or OF
```

path)
<addr>: user address from/to which data will be retrieved/stored
<off>: offset in <name> in bytes (default : start of the part)
* must be block-aligned for erase * must be page-aligned otherwise <size>: length of the operation in bytes (default: the entire device) * must be a multiple of a block for erase * must be a multiple of a page otherwise (special case : default is a page with dump)

The .dontskipff option forces writing empty pages, don 't use it if unsure.

=>
=> mtd read uboot 0x40000000
Reading 1048576 byte(s) at offset 0x00000000
=> mtd dump uboot 0 0x10
Reading 16 byte(s) at offset 0x00000000

Dump 16 data bytes from 0x0:
0x00000000: d0 0d fe ed 00 0d e8 95 00 00 00 38 00 0d e4 44
=>
Read and write nor flash based on sf command

=> sf
sf - SPI flash sub-system

Usage:
sf probe [[bus:]cs] [hz] [mode] - init flash device on given SPI bus
and chip select
sf read addr offset|partition len - read ` len ' bytes starting at
`offset' or from start of mtd
` partition 'to memory at `addr' sf write addr offset|partition len - write ` len ' bytes from memory at `addr' to flash at ` offset '
or to start of mtd `partition' sf erase offset|partition [ + ]len - erase ` len ' bytes from `offset' or from start of mtd ` partition '
`+len' round up ` len ' to block size sf update addr offset|partition len - erase and write `len' bytes from memory at ` addr ' to flash at `offset'
or to start of mtd ` partition ' sf protect lock/unlock sector len - protect/unprotect ' len ' bytes starting at address ' sector ' => sf probe SF: Detected w25q32 with page size 256 Bytes, erase size 4 KiB, total 4 MiB => sf read 0x40000000 0 0x10 device 0 offset 0x0, size 0x10 SF: 16 bytes @ 0x0 Read: OK =>

Common interfaces

```
include  < spi . h >
#include  <spi_flash.h>

struct  udevice  * new ,  * bus_dev ;
int  ret ;
static  struct  spi_flash  * flash ;

//bus, cs corresponds to the bus and cs numbers of spi, such as 0, 0
ret  =  spi_find_bus_and_cs ( bus ,  cs ,  & bus_dev ,  & new );
flash  =  spi_flash_probe ( bus ,  cs ,  speed ,  mode );

ret  =  spi_flash_read ( flash ,  offset ,  len ,  buf );
```

hdmi

This section mainly introduces how to turn on the hdmi driver.

config configuration

Execute make uboot_menuconfig, that is, enter Device Drivers -> Graphics support and turn on the following configuration (enabled by default).

dts configuration

```
& dpu  {
status = "okay" ; };


& hdmi  {
pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_hdmi_0 > ; status = "okay" ; };
```

boot logo

This section mainly introduces how to display the bootlogo during the uboot startup phase.

config configuration
Execute make menuconfig to open the following configuration.

First turn on hdmi support under uboot, refer to 5.12 summary.
Then open bootlogo support under uboot, enter Device Drivers -> Graphics support, and enable the following options.

env configuration
In k1-xh in the uboot-2022.10\include\configs directory, add the three env variables required for bootlogo: splashimage, splashpos, and splashfile.

//uboot-2022.10/include/configs/k1-xh
... ... #define CONFIG_EXTRA_ENV_SETTINGS \ "fdt_high=0xffffffffffffffff\0" \ "initrd_high=0xffffffffffffffff\0" \ ... . "splashimage=" __stringify (CONFIG_FASTBOOT_BUF_ADDR) " \0 " \ "splashpos=m,m \0 " \ "splashfile=k1-x.bmp \0 " \

... ...
... ... BOOTENV_DEVICE_CONFIG \ BOOTENV

#endif /* __CONFIG_H */
Among them, splashimage represents the address where the bootlogo image is loaded into the memory;

splashpos represents the position where the image is displayed, "m,m" represents the image displayed in the middle of the screen;

splashfile refers to the name of the bmp file to be displayed. This image needs to be placed in the partition where bootfs is located.

Pack the .bmp image into bootfs
Pack the bmp image to be displayed into bootfs:

Place the k1-x.bmp file in the ./buildroot-ext/board/spacemit/k1 directory. The file name

must be consistent with UBOOT_LOGO_FILE and the environment variable splashfile in buildroot-ext/board/spacemit/k1/prepare_img.sh. , such as k1-x.bmp.

After compilation and packaging, the bmp image will be packaged into bootfs.

//buildroot-ext/board/spacemit/k1/prepare_img.sh
#!/bin/bash

######################### Prepare sub-iamges and pack ####################
#$0 is this file path
#$1 is buildroot output images dir

IMGS_DIR = $1
DEVICE_DIR = $(dirname $0)

... ...

UBOOT_LOGO_FILE = " $DEVICE_DIR /k1-x.bmp"
How to modify bootlogo
Directly replace k1-x.bmp in the buildroot-ext/board/spacemit/k1/ directory, or add pictures according to the above description
boot menu
This section mainly introduces the bootmenu function of opening uboot.

config configuration
Execute make menuconfig, enter Command line interface > Boot commands, and open the following configuration

Then go to Boot options > Autoboot options and turn on the following options

env configuration
Bootdelay and bootmenu_delay need to be added to buildroot-ext/board/spacemit/k1/env_k1-x.txt, for example, bootdelay=5, bootmenu_delay=5, where 5 represents the waiting time of bootmenu, in seconds.

//buildroot-ext/board/spacemit/k1/env_k1-x.txt
bootdelay = 5

# Boot menu definitions
boot_default = echo "Current Boot Device: ${boot_device}"
flash_default = echo "Returning to Boot Menu..."
spacemit_flashing_usb = echo "recovery from usb... " ; \

spacemit_flashing usb ; spacemit_flashing_mmc = echo "recovery from mmc... " \ spacemit_flashing mmc ; spacemit_flashing_net = echo "recovery from net... " \ spacemit_flashing net ; bootmenu_delay = 5 bootmenu_0 = "-------- Boot Options --------" = run boot_default bootmenu_1 = "Boot from Nor" = run nor_boot bootmenu_2 = "Boot from Nand" = run nand_boot bootmenu_3 = "Boot from MMC" = run try_mmc bootmenu_4 = "Autoboot" = run autoboot bootmenu_5 = "Show current Boot Device" = run boot_default bootmenu_6 = "-------- Flash Options --------" = run flash_default bootmenu_7 = "recovery from usb" = run spacemit_flashing_usb bootmenu_8 = "recovery from mmc" = run spacemit_flashing_mmc bootmenu_9 = "recovery from net" = run spacemit_flashing_net

Enter bootmenu
After powering on, press and hold the Esc key on the keyboard to enter the bootmenu.

fastboot command
This section mainly introduces the fastboot commands supported by the k1-deb1 solution.

Compile configuration
Execute make menuconfig, enter Device Drivers --->Fastboot support, and enable the following compilation configuration

Fastboot relies on the USB driver and needs to enable the USB support of the USB configuration.

Enter fastboot mode

You can enter the uboot shell by pressing the "s" key after system startup, and execute fastboot 0 to enter fastboot mode.
The system's default fastboot buffer addr/size is the macro definition CONFIG_FASTBOOT_BUF_ADDR/CONFIG_FASTBOOT_BUF_SIZE

```
#Or fastboot -l 0x30000000 -s 0x10000000 0, specify buff addr/size
=> fastboot 0
k1xci_udc: phy_init
k1xci_udc probe
k1xci_udc: pullup 1
--  suspend  --
 handle setup GET_DESCRIPTOR, 0x80, 0x6 index 0x0 value 0x100 length 0x40
handle setupSET_ADDRESS , 0x0, 0x5 index 0x0 value 0x22 length 0x0
handle setup GET_DESCRIPTOR, 0x80, 0x6 index 0x0 value 0x100 length 0x12
..
```

After the device boots to bianbu os, send the adb reboot bootloader command and the system will reboot into fastboot mode.
Supported fastboot commands
For fastboot environment configuration on the computer side, please refer to the computer environment installation chapter.

```
#fastboot native protocol command
 fastboot devices              #Display available devices
 fastboot reboot               #Restart device
 fastboot getvar [ version/product/serialno/max-download-size]
fastboot flash partname image #Burn the image image to the partname partition
 fastboot erase partname       #Erase partname partition
 fastboot stage file           #Download file file to memory buff addr

#OEM manufacturer-defined commands and functions
 fastboot getvar [ mtd-size/blk-size] #Get the mtd/blk device size, if not, return NULL
 fastboot oem read part              #Read the data in part to buff addr
 fastboot get_staged file       # Upload the data and name it file. Depends on OEM read part command
File system
fat
=> fat
  fatinfo fatload fatls fatmkdir fatrm fatsize fatwrite
=> fatls mmc 2:5
50896911 uImage.itb
    4671 env_k1-x.txt

2 file ( s ) , 0 dir ( s )

=> fatload mmc 2:5 0x40000000 uImage.itb #load uImage.itb to 0x40000000
 50896911 bytes read in 339 ms ( 143.2 MiB/s )
```

=>
ext4
Similar to fat command

=> ext4
  ext4load ext4ls ext4size
=> ext4load
ext4load - load binary file from an Ext4 filesystem

Usage:
ext4load <interface> [<dev[:part]> [addr [filename [bytes [pos]]]]]
    - load binary file 'filename' from 'dev' on 'interface'
       to address 'addr' from ext4 filesystem
=>
Other shell commands
Common commands, commonly used tools

printenv   - print environment variables
md - memory display
mw - memory write (fill)
 fdt - flattened device tree utility commands

help   - print command description/usage
fd
The fdt command is mainly used to print dts content, such as the dtb file currently loaded after uboot starts.

=> fdt
fdt - flattened device tree utility commands

Usage:
fdt addr [-c] [-q] <addr> [<size>] - Set the [control] fdt location to <addr>
fdt move <fdt> <newaddr> <length> - Copy the fdt to <addr > and make it active
fdt resize [<extrasize>] - Resize fdt to size + padding to 4k addr + some optional <extrasize> if needed
fdt print <path> [<prop>] - Recursive print starting at <path>
fdt list <path> [<prop>] - Print one level starting at <path>
fdt get value <var> <path> <prop> [<index>] - Get <property> and store in <var>
                                In case of stringlist property , use optional <index>
                                to select string within the stringlist. Default is 0.
fdt get name <var> <path> <index> - Get name of node <index> and store in <var>
fdt get addr <var> <path> <prop> - Get start address of <property> and store in < var>
fdt get size <var> <path> [<prop>] - Get size of [<property>] or num nodes and store in <var>

```
fdt set     <path> <prop> [ <val>] - Set < property> [ to <val>]
fdt mknode <path> <node> - Create a new node after <path>
fdt rm      <path> [ <prop>] - Delete the node or <property>
fdt header [ get <var> <member>] - Display header info
                                   get - get header member <member> and store it in <var>
fdt bootcpu <id> -                    Set boot cpuid fdt memory
 <addr> <size> - Add/Update memory node
fdt rsvmem print - Show current mem reserves
fdt rsvmem add <addr> <size> - Add a mem reserve
fdt rsvmem delete <index> - Delete a mem reserves
fdt chosen [ <start> <size>] - Add/update the /chosen branch in the tree
                                    <start>/<size> - initrd start addr/size
NOTE: Dereference aliases by omitting the leading '/' , eg fdt print ethernet0.
=>

=> fdt addr $fdtcontroladdr
=> fdt print
/ {
        compatible =  "spacemit,k1x" , "riscv" ;
#address-cells = <0x00000002>; #size-cells = <0x00000002>;         model = "spacemit
k1-x deb1 board" ; ... ...         memory@0 {                 device_type = "memory" ;
reg = <0x00000000 0x00000000 0x00000000 0x80000000> ; } ;         chosen {
bootargs = "earlycon=sbi console=ttyS0,115200 debug loglevel=8,initcall_debug= 1
rdinit=/init.tmp" ;                 stdout-path = "serial0:115200n8" ; } ; } ;

=> fdt print /chosen
chosen {
        bootargs =  "earlycon=sbi console=ttyS0,115200 debug
loglevel=8,initcall_debug=1 rdinit=/init.tmp" ;
        stdout-path =  "serial0:115200n8" ;
} ;
=>
shell command
uboot supports shell-style commands, such as if/fi, echo, etc.
```

```
=>  if test  ${ boot_device }  = nand ;  then echo  "nand boot" ;  else echo  "not nand
boot" ; fi
 not nand boot
=>  printenv boot_device
boot_device = nor
=>  if test  ${ boot_device }  = nor ;  then echo  "nor boot" ;  else echo  "not nor boot" ;
fi
 nor boot
=>
```

opensbi function and configuration

opensbi compile

```
cd ~/opensbi/

#Note that the compilation tool chain here needs to be provided by spacemit. If it is not
provided by spacemit, it may cause compilation exception
GCC_PREFIX = riscv64-unknown-linux-gnu-

CROSS_COMPILE = ${ GCC_PREFIX }  PLATFORM = generic \
PLATFORM_DEFCONFIG = k1-x_deb1_defconfig \
PLATFORM_RISCV_ISA = rv64gc \
FW_TEXT_START = 0x0   \
 make
```

The generated compiled file is as follows

```
~/opensbi $ ls build/platform/generic/firmware/ -l
 fw_dynamic.bin      #The dynamic mirror jump will pass the configuration parameter
 fw_dynamic.elf
fw_dynamic.elf.dep
fw_dynamic.itb      #Package fw_dynamic.bin into fit format
 fw_dynamic.its
fw_dynamic.o
fw_jump.bin        #jump image only jumps
 fw_payload.bin     #payload image will include uboot image
```

opensbi function configuration

You can turn on or off certain functions by executing menuconfig

```
make PLATFORM = generic PLATFORM_DEFCONFIG = k1-x_deb1_defconfig
menuconfig
```

Flash the opensbi image

For devices that have been started normally, the opensbi image can be burned separately based on fastboot

```
#The device enters uboot through adb reboot bootloader or long press "s" on the
keyboard after powering on.
```

#Enter fastboot 0 in uboot shell.
#After entering fastboot mode, you can burn the image through the fastboot tool.

#Tool side command

#Burn opensbi image
fastboot flash opensbi ~/opensbi/fw_dynamic.itb
Startup settings
This section introduces how to configure different storage media. For all storage media startup, the startup process after powering on the board is as shown in the figure below:

According to the boot pin select of the hardware, the software will select the drive medium corresponding to the storage medium to start the next-level image. The specific boot pin select is introduced in the flashing chapter.

After K1 is powered on, it will first try boot from sd, and then load the uboot and opensbi images from storage media such as emmc, nand, or nor according to the boot pin select.

The following sections will introduce the configuration of the two stages of spl and uboot, including partition table configuration, menuconfig configuration, dts, etc.

For partition table configuration, mtd storage media (nand/nor flash) will be expressed in the form of partition_2M.json and other capacities, and blk devices will be named with partition_universal.json. The partition table is stored in buildroot-ext/board/spacemit/k1/flash_config/partition_xxx.json,

Notice:

The partition table is written in json format, and the definitions of all partitions are in the partitions list.
The name/size of each partition is required. offset/image is optional. If there is no offset, the sum of the sizes of all previous partitions is used as the offset. If there is no image, the partition will not be burned but the partition will be retained.
The bootinfo partition is required and cannot be changed. The main function is to guide brom to load the FSBL.bin image.
fsbl is a required option, the offset address is specified by bootinfo, and the fsbl_1 backup partition can be added. FSBL.bin consists of u-boot-spl.bin plus address information and other header information.
The env partition will be related to uboot's env loading.
sd startup configuration
By default, it will try boot from sd, and then try other media after failure. You need to ensure that uboot has enabled mmc driver configuration. For specific mmc driver

configuration, please refer to the mmc chapter of uboot driver development and debugging.

Partition table configuration
The partition table is saved in buildroot-ext/board/spacemit/k1/partition_universal.json. Among them, bootinfo is not used as an explicit partition and is used to save information related to boot media.

{
"version" : "1.0" , "format" : "gpt" , "partitions" : [{ "name" : "bootinfo" , "offset" : "0" , "size" : "80B" , "image" : "factory/bootinfo_sd.bin" }, { "name" : "fsbl" , "offset" : "256K" , "size" : "256K" , "image" : "factory/FSBL.bin" }, { " name" : "env" , "offset" : "512K" , "size" : "128K" }, { "name" : "opensbi" , "offset" : "1M" , "size" : "1M" , " image" : "opensbi.itb" }, { "name" : "uboot" , "offset" : "2M" , "size" : "2M" , "image" : "u-boot.itb" }, { " name" : "bootfs" , "offset" : "4M" , "size" : "128M" , "image" : "bootfs.img" }, { "name" : "rootfs" , "size" : "-" }] }

spl configuration
Execute make uboot_menuconfig and select SPL configuration options

Turn on MMC Raw mode: by partition, Support MMC (keyboard input Y means to select, N means to cancel)

Partition to use to load U-Boot from indicates which partition to load the next-level boot image from, which should be determined based on the partition table.

The K1 solution loads opensbi and uboot independently by default, and loads image files from the opensbi and uboot partitions of mmc respectively.

Notice:

If you support opensbi/uboot to be loaded and started separately, you need to enable Support loading from mtd device. The first partition must be opensbi, followed by uboot. spl will first load the image according to the partition table. If the loading fails, it will load the image according to the partition number.

The location of the partition name and partition number must be consistent with the partition table.

spl-dts configuration
The dts of spl needs to enable the mmc driver, as shown in the following dts configuration

//uboot-2022.10/arch/riscv/dts/k1-x_spl.dts
/* eMMC */ sdh @ d4281000 { bus - width = < 8 > ; non - removable ; mmc - hs400 - 1

_8v ; mmc - hs400 - enhanced - strobe ; sdh - phy - module = < 1 > ; status = "okay" ; u - boot , dm - spl ; };

emmc startup configuration
Both emmc and sd use the mmc driver. The uboot code process will select emmc or sd according to the boot pin select started.

The startup configuration is the same as that of sd.

nor+ssd startup configuration
For nor media boot, k1 will provide nor(u-boot-spl/uboot/opensbi)+ssd(bootfs/rootfs) or pure nor(u-boot-spl/uboot/opensbi/kernel) boot. The following will introduce the nor+ssd startup scheme configuration.

Please ensure that nor, spi, nvme and other driver configurations are turned on normally.

Partition table configuration
nor partition table, such as buildroot-ext/board/spacemit/k1/partition_4M.json. For nand/nor devices, the partition table is named after partiton_xM.json, and needs to be renamed according to the actual flash capacity, otherwise the corresponding partition table will not be found when flashing.

Nor flash partition table modification:

1. The partition starting address and size are aligned to 64KB by default (corresponding to erasesize of 64KB).

2. If the starting address and size need to be changed to 4KB alignment, you need to enable uboot's compilation configuration
"CONFIG_SPI_FLASH_USE_4K_SECTORS"

//buildroot-ext/board/spacemit/k 1 /partition_ 4 M.json
{
"version" : "1.0" , "format" : "mtd" , "partitions" : [{ "name" : "bootinfo" , " offset" : "0" , "size" : "64K" , "image" : "factory/bootinfo_spinor.bin" }, { "name" : "fsbl" , "offset" : "64K" , "size" : " 256K" , "image" : "factory/FSBL.bin" }, { "name" : "env" , "offset" : "512K" , "size" : "128K" , "image" : "env.bin" }, { "name" : "opensbi" , "offset" : "640K" ,

"size" : "384K" , "image" : "opensbi.itb" }, { "name" : "uboot" , "offset" : "1M" , "size" : "1M" , "image" : "u-boot.itb" }] }

ssd partition table. For the partition table of the blk device, it is partition_universal.json. At this time, bootinfo, fsbl, env, opensbi, uboot and other partitions and the data in them will not affect normal startup.

//buildroot-ext/board/spacemit/k 1 /partition_universal.json
{
"version" : "1.0" , "format" : "gpt" , "partitions" : [{ "name" : "bootinfo" , "offset" : "0" , "size" : "80B" , "image" : "factory/bootinfo_sd.bin" }, { "name" : "fsbl" , "offset" : "256K" , "size" : "256K" , "image" : "factory/FSBL.bin" }, { "name" : "env" , "offset" : "512K" , "size" : "128K" }, { "name" : "opensbi" , " offset" : "1M" , "size" : "1M" , "image" : "opensbi.itb" }, { "name" : "uboot" , "offset" : "2M" , "size" : "2M" , "image" : "u-boot.itb"

}, { "name" : "bootfs" , "offset" : "4M" , "size" : "128M" , "image" : "bootfs.img" }, { "name" : "rootfs" , "size" : "-" }] }

spl configuration
Execute make uboot_menuconfig and select SPL configuration options

Turn on "Support MTD drivers", "Support SPI DM drivers in SPL", "Support SPI drivers", "Support SPI flash drivers", "Support for SPI flash MTD drivers in SPL", "Support loading from mtd device",

"Partition name to use to load U-Boot from" is consistent with the partition name in the partition table.

If you enable opensbi/uboot independent mirroring. Then you need to turn on "Second partition to use to load U-Boot from", and you must keep the opensbi/uboot partition in the order of first/second.

For mtd devices, env needs to be enabled to ensure that the mtd partition information can be obtained from env after spl is started.

Execute make uboot_menuconfig, select Environment, and select env loading to enable spi. The env offset address here needs to be consistent with the env partition of the partition table, such as 0x80000.

The spi flash driver needs to adapt to the spi flash corresponding to the manufacturer's model on the hardware. Select the corresponding manufacturer manufacturer ID on menuconfig

Execute make uboot_menuconfig and select Device Drivers--->MTD Support--->SPI Flash Support

According to the spi flash manufacturer of the hardware, select the corresponding driver.

If it doesn't exist in the driver, you can add it directly in the code. flash_name can be customized, usually the name of the hardware flash, 0x1f4501 is the jedecid of the flash, and other parameters can be added according to the flash of the hardware.

```
//uboot-2022.10/drivers/mtd/spi/spi-nor-ids.c
const  struct  flash_info  spi_nor_ids []  =  {
{ INFO ( "flash_name" , 0x1f4501 , 0 , 64 * 1024 , 16 , SECT_4K ) },
```

blk device configuration
Execute make uboot_menuconfig and select Device Drivers --->Fastboot support

Select Support blk device. The supported blk devices are ssd/emmc. For example, ssd corresponds to nvme, and emmc corresponds to mmc.

spl-dts configuration
//uboot-2022.10/arch/riscv/dts/k1-x_spl.dts
spi @ d420c000 { status = "okay" ; pinctrl - names = "default" ; pinctrl - 0 = <& pinctrl_qspi > ; u - boot , dm - spl ;

spi - max - frequency = < 15140000 > ;
flash @ 0 { compatible = "jedec,spi-nor" ; reg = < 0 > ; spi - max - frequency = < 26500000 > ;

m25p , fast - read ;
broken - flash - reset ; u - boot , dm - spl ; status = "okay" ; }; };

nand startup configuration
For nand media boot, K1 will provide nand(u-boot-spl/uboot/opensbi)+ssd(bootfs/rootfs) or pure nand(u-boot-spl/uboot/opensbi/kernel) boot. The pure nand startup scheme configuration will be introduced below.

Please ensure that nand and spi drivers are configured normally.

Partition table configuration
For example, for nand with a capacity of 256MB, the partition table is partition_256M.json

//buildroot-ext/board/spacemit/k 1 /partition_ 256 M.json
{
"version" : "1.0" , "format" : "mtd" , "partitions" : [{ "name" : "bootinfo" , " offset" : "0" , "size" : "128K" , "image" : "factory/bootinfo_spinand.bin" }, { "name" : "fsbl" , "offset" : "256K" , "size" : " 256K" , "image" : "factory/FSBL.bin" }, { "name" : "env" , "offset" :

"512K" , "size" : "128K" }, { "name" : "opensbi" , "offset" : "640K" , "size" : "384K" , "image" : "opensbi.itb" }, { "name" : "uboot" , "offset" : "1M" , "size" : " 1M" , "image" : "u-boot.itb" }, { "name" : "user" , "offset" : "2M" , "size" : "-" , "volume_images" : { "bootfs" : "bootfs.img" , "rootfs" : "rootfs.img" } }] }

spl configuration
Execute make uboot_menuconfig and select SPL configuration options

turn on

Support MTD drivers

Support SPI DM drivers in SPL

Support SPI drivers

Use standard NAND driver

Support simple NAND drivers in SPL

Support loading from mtd device,

Partition name to use to load U-Boot from is consistent with the partition name in the partition table.

If you enable opensbi/uboot independent mirroring. Then you need to enable the Second partition to use to load U-Boot from, and the order of opensbi/uboot must be maintained.

For mtd devices, env needs to be enabled to ensure that the mtd partition information can be obtained from env after spl is started.

Execute make uboot_menuconfig, select Environment, and select env loading to enable spi. The env offset address here needs to be consistent with the env partition of the partition table, such as 0x80000.

The nand flash driver needs to adapt to the spi flash on the hardware corresponding to the manufacturer's model. The currently supported nand flash drivers are shown below. If there is no corresponding driver, you can add the manufacturer's jedecid in the other.c driver.

~/ uboot - 2022.10 $ ls drivers / mtd / nand / spi / * .c
uboot-2022.10/drivers/mtd/nand/spi/core.c
uboot-2022.10/drivers/mtd/nand/spi/micron.c
uboot-2022.10/drivers/mtd/nand/spi/winbond.c
uboot-2022.10/drivers/mtd/nand/spi/gigadevice.c
uboot-2022.10/drivers/mtd/nand/spi/other.c
uboot-2022.10/drivers /mtd/nand/spi/macronix.c
uboot-2022.10/drivers/mtd/nand/spi/toshiba.c

//uboot-2022.10/drivers/mtd/nand/spi/other.c
static int other_spinand_detect(struct spinand_device *spinand)
{
    u8 *id = spinand->id.data;
    int ret = 0;

    /*
     * dosilicon nand flash
     */
if (id [1] == 0xe5) ret = spinand_match_and_init (spinand , dosilicon_spinand_table
, ARRAY_SIZE (dosilicon_spinand_table), id [2]);

    /*FORESEE nand flash*/
if (id [1] == 0xcd) ret = spinand_match_and_init (spinand , foresee_spinand_table ,
ARRAY_SIZE (foresee_spinand_table), id [2]); if (ret) return ret ;

    return 1 ;
}
spl-dts configuration
//uboot-2022.10/arch/riscv/dts/k1-x_spl.dts
spi @ d420c000 { status = "okay" ; pinctrl - names = "default" ; pinctrl - 0 = <&
pinctrl_qspi > ; u - boot , dm - spl ;

            spi - max - frequency = < 15140000 > ;
spi - nand @ 0 { compatible = "spi-nand" ; reg = < 0 > ; spi - tx - bus - width = < 1 > ;
spi - rx - bus - width = < 1 > ; spi - max - frequency = < 6250000 > ; u - boot , dm - spl ;
status = "okay" ; }; };

Jump to a line...
Jump
report
Report successful
We will feedback the results to you via in-site message within 2 working days!
Please fill in the reason for the report carefully and describe it in as much detail as possible.
Report type
Please select report type
Report reason
Please explain the reason for reporting
Cancel
send
Miscarriage of justice appeal
There may be content here that is inappropriate for display and the page will not be displayed. You can self-check and modify through the relevant editing functions.

If you confirm that the content does not involve inappropriate language/pure advertising diversion/violence/vulgar pornography/infringement/piracy/false/worthless content or content that violates relevant national laws and regulations, you can click Submit to lodge an appeal and we will handle it for you as soon as possible.

Cancel
submit
Ma Jiancang AI Assistant
try more
Code interpretation
Find faults in code
Code optimization
1
https://gitee.com/bianbu-linux/docs-en.git
git@gitee.com:bianbu-linux/docs-en.git
bianbu-linux
docs-en
docs-en
main
Gitee - Git-based code hosting and R&D collaboration platform

Git encyclopedia
Git command learning
CopyCat code clone detection
APP and plug-in download
Gitee Reward
Gitee Cover Character

GVP project
Gitee Blog
Gitee Charity Plan
Gitee continuous integration
OpenAPI
Help documentation
Online self-service
Change log
about Us
join us
terms of use
suggestions
Partner
Technical exchange QQ group
Technical exchange QQ group

WeChat service account
WeChat service account

client#oschina.cn
Enterprise version online use: 400-606-0201
Professional version private deployment:
13670252304
13352947997
Open Atom Open Source Foundation Open Atom Open Source Foundation
Cooperative code hosting platform
Illegal and harmful information reporting center Illegal and harmful information reporting center
Guangdong ICP No. 12009483
Simplified / Traditional / English
Click here to find more help
search help
Please enter product name or question
Git command online learning How to import GitHub repository in Gitee
Basic operations of Git warehouse
Function comparison between Enterprise Edition and Community Edition
SSH public key settings
How to handle code conflicts
The warehouse is too large, how to reduce it?
How to retrieve deleted warehouse data
Gitee Product Quota Description
GitHub warehouse quickly imports Gitee and synchronizes updates
What is Release?
Automatically publish PHP projects to packagist.org
Comment
Warehouse report

back to the top
Login prompt
This operation requires logging in to a Gitee account, please log in first before operating.
log in immediately
No account, please register