StarFive 赛昉科技

VisionFive 2 Single Board Computer Software Technical Reference Manual

Version: 1.2

Date: 2023/05/17

Doc ID: VisionFive2-TRMEN-001

Legal Statements

Important legal notice before reading this documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to the product development. Shanghai StarFive Technology Co., Ltd.(hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFiveexpressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

StarFivedoes not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

Contact Us

Address: Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: http://www.starfivetech.com http://www.starfivetech.com

Email: sales@starfivetech.com(sales) , support@starfivetech.com(support)

Preface

About this guide and technical support information.

About this document

This document mainly describes how to compile firmware, U-Boot, Linux Kernel and make file systems.

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.0	2022/12/26	The first official release.
1.1	2023/03/01	Updated steps in Creating SPL File *(on page 11)*.
1.2	2023/05/17	Updated steps in Adding New File *(on page 15)*.

Notes and notices

The following notes and notices might appear in this guide:

- **Tip:**
  Suggests how to apply the information in a topic or step.
- **Note:**
  Explains a special case or expands on an important point.
- **Important:**
  Points out critical information concerning a topic or step.
- **CAUTION:**
  Indicates that an action or step can cause loss of data, security problems, or performance issues.
- **Warning:**
  Indicates that an action or step can result in physical harm or cause damage to hardware.

Contents

List of Tables

List of Figures

1. Required Hardware

Make sure that the following hardware are prepared for the operation described in this manual:

- VisionFive 2
- Micro SD card (32 GB or more)
- PC with Linux/Windows/Mac OS
- USB to Serial Converter
- Ethernet cable
- Power adapter
- USB Type-C Cable

Note:
In this guide, Ubuntu 18.04 LTS is installed on the host PC.

2. Making General System

This chapter describes how to make a general system.

It contains the following sections:

2.1. Compiling U-boot and Kernel

This chapter describes how to compile the U-Boot and kernel.

It contains the following sections:

2.1.1. Set Up Compilation Environment

You can follow the steps below to set up your cross-compile.

1. Execute the following commands to install the `riscv64-linux-gnu-gcc` compiler from Ubuntu packages.

   ```
   sudo apt update
   sudo apt upgrade
   sudo apt install gcc-riscv64-linux-gnu
   ```

2. Execute the following command to check the version of the `riscv64-linux-gnu-gcc`.

   ```
   riscv64-linux-gnu-gcc -v
   ```

   The output will be as follows:

   Result:

   Figure 2-1 Example Output

   ```
   ryan@ubuntu:~$ riscv64-linux-gnu-gcc -v
   Using built-in specs.
   COLLECT_GCC=riscv64-linux-gnu-gcc
   COLLECT_LTO_WRAPPER=/usr/lib/gcc-cross/riscv64-linux-gnu/7/lto-wrapper
   Target: riscv64-linux-gnu
   Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --with-
   bugurl=file:///usr/share/doc/gcc-7/README.Bugs --enable-languages=c,c++,d,fortran,objc,obj-c
   ++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-7 --enable-shared --enable-
   linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --li
   bdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx-debug --
   enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disa
   ble-libitm --disable-libsanitizer --disable-libquadmath --disable-libquadmath-support --enab
   le-plugin --with-system-zlib --enable-multiarch --disable-werror --disable-multilib --with-a
   rch=rv64imafdc --with-abi=lp64d --enable-checking=release --build=x86_64-linux-gnu --host=x8
   6_64-linux-gnu --target=riscv64-linux-gnu --program-prefix=riscv64-linux-gnu- --includedir=/
   usr/riscv64-linux-gnu/include
   Thread model: posix
   gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
   ```

2.1.2. Compiling the U-Boot

Follow the steps below to compile the U-Boot for VisionFive 2.

1. Locate to your desired directory to store the U-Boot files. For example, the home directory.
   Example:

   ```
   cd ~ # home directory
   ```

2. Download the source code for U-Boot compilation.

   ```
   git clone https://github.com/starfive-tech/u-boot.git
   ```

3. Switch to the code branch by executing the following command:

   ```
   cd u-boot
   git checkout -b JH7110_VisionFive2_devel origin/JH7110_VisionFive2_devel
   git pull
   ```

4. Type the following to compile U-Boot under the U-Boot directory.

   ```
   make <Configuration_File> ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
   make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
   ```

   Tip:

   Configuration_File: For VisionFive 2, the file is `starfive_visionfive2_defconfig`.

   Result:

   There will be these 3 files generated after compilation inside the `u-boot` directory:

   - `u-boot.bin`
   - `arch/riscv/dts/starfive_visionfive2.dtb`
   - `spl/u-boot-spl.bin`

Figure 2-2 Example Output - u-boot.bin

Figure 2-3 Example Output - visionfive2.dtb

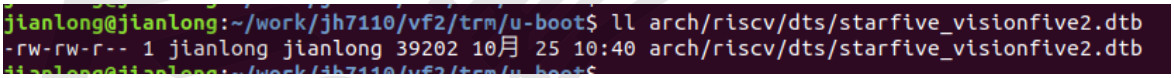


Figure 2-4 Example Output - u-boot-spl.bin

```
jianlong@jianlong:~/work/jh7110/vf2/trm/u-boot/spl$ ll
total 2800
drwxrwxr-x 13 jianlong jianlong    4096 10月 25 10:40 ./
drwxrwxr-x 26 jianlong jianlong    4096 10月 25 10:40 ../
drwxrwxr-x  3 jianlong jianlong    4096 10月 25 10:40 arch/
drwxrwxr-x  3 jianlong jianlong    4096 10月 25 10:40 board/
drwxrwxr-x  2 jianlong jianlong    4096 10月 25 10:40 cmd/
drwxrwxr-x  4 jianlong jianlong    4096 10月 25 10:40 common/
drwxrwxr-x  2 jianlong jianlong    4096 10月 25 10:40 disk/
drwxrwxr-x 16 jianlong jianlong    4096 10月 25 10:40 drivers/
drwxrwxr-x  2 jianlong jianlong    4096 10月 25 10:40 dts/
drwxrwxr-x  2 jianlong jianlong    4096 10月 25 10:40 env/
drwxrwxr-x  2 jianlong jianlong    4096 10月 25 10:40 fs/
drwxrwxr-x  3 jianlong jianlong    4096 10月 25 10:40 include/
drwxrwxr-x  3 jianlong jianlong    4096 10月 25 10:40 lib/
-rw-rw-r--  1 jianlong jianlong   15689 10月 25 10:40 u-boot.cfg
-rwxrwxr-x  1 jianlong jianlong 2030360 10月 25 10:40 u-boot-spl*
-rwxrwxr-x  1 jianlong jianlong  127400 10月 25 10:40 u-boot-spl.bin*
-rw-rw-r--  1 jianlong jianlong      73 10月 25 10:40 .u-boot-spl.bin.cmd
-rw-rw-r--  1 jianlong jianlong     610 10月 25 10:40 .u-boot-spl.cmd
-rw-rw-r--  1 jianlong jianlong    1076 10月 25 10:40 u-boot-spl.lds
-rw-rw-r--  1 jianlong jianlong    5143 10月 25 10:40 .u-boot-spl.lds.cmd
-rw-rw-r--  1 jianlong jianlong  393501 10月 25 10:40 u-boot-spl.map
-rwxrwxr-x  1 jianlong jianlong  127400 10月 25 10:40 u-boot-spl-nodtb.bin*
-rw-rw-r--  1 jianlong jianlong     111 10月 25 10:40 .u-boot-spl-nodtb.bin.cmd
-rw-rw-r--  1 jianlong jianlong   74215 10月 25 10:40 u-boot-spl.sym
-rw-rw-r--  1 jianlong jianlong      91 10月 25 10:40 .u-boot-spl.sym.cmd
```

Tip:
Both `starfive_visionfive2.dtb` and `u-boot.bin` will be used later for OpenSBI compilation.

Tip:
`u-boot-spl.bin` will be used later for creating SPL file.

2.1.3. Compiling OpenSBI

OpenSBI stands for Open-source Supervisor Binary Interface and it is an open-source implementation of the RISC-V Supervisor Binary Interface. It is a RISC-V-specific runtime service provider and it is typically used in boot stage following ROM and LOADER. A typical boot flow is as follows:

Figure 2-5 Typical Boot Flow

Follow the steps below to compile OpenSBI for VisionFive 2.

1. Locate to your desired directory to store the OpenSBI files. For example, the home directory.

   ```
   cd ~ # home directory
   ```

2. Download the source code for OpenSBI compilation.

   ```
   git clone https://github.com/starfive-tech/opensbi.git
   ```

3. Inside `opensbi` directory, type the following to compile openSBI.

```
cd opensbi
make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- PLATFORM=generic
 FW_PAYLOAD_PATH={U-BOOT_PATH}/u-boot.bin
 FW_FDT_PATH={U-BOOT_PATH}/arch/riscv/dts/starfive_visionfive2.dtb FW_TEXT_START=0x40000000
```

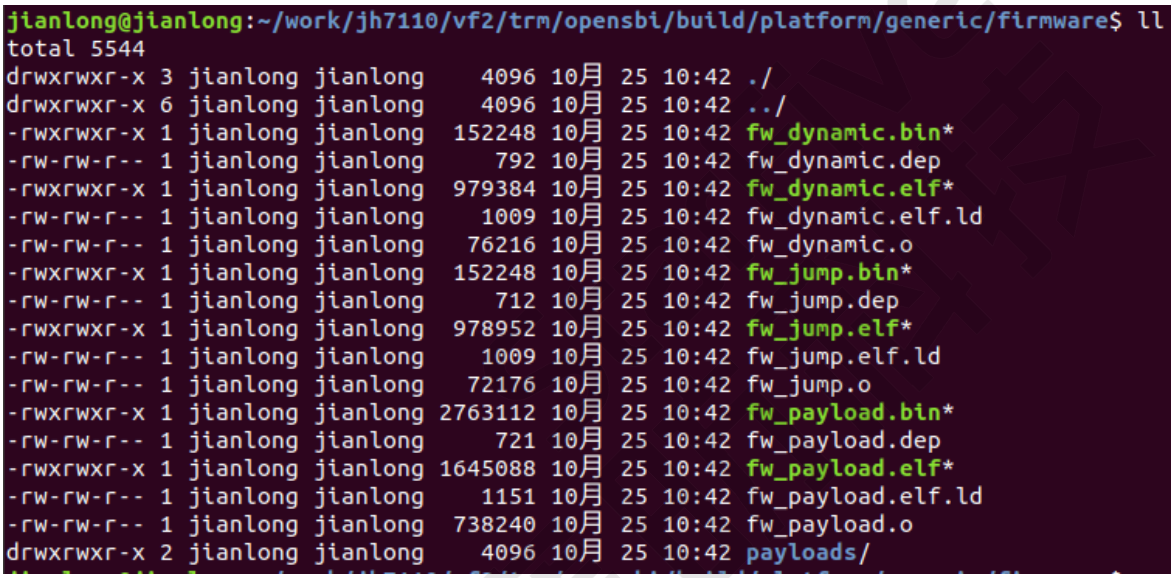
Tip:

Modify the *{U-BOOT_PATH}* to the path of U-Boot from before.

Result:

After compilation, the file `fw_payload.bin` will be generated in the directory `opensbi/build/platform/generic/firmware` and the size is larger than 2M.

Figure 2-6 Example Output



2.1.4. Creating SPL File

Follow the steps below to create the SPL file for VisionFive 2.

1. Locate to your desired directory to store the tools files. For example, the home directory.

   Example:

   ```
   cd ~ # home directory
   ```

2. Download the source code for U-Boot compilation.

   ```
   git clone https://github.com/starfive-tech/Tools
   ```

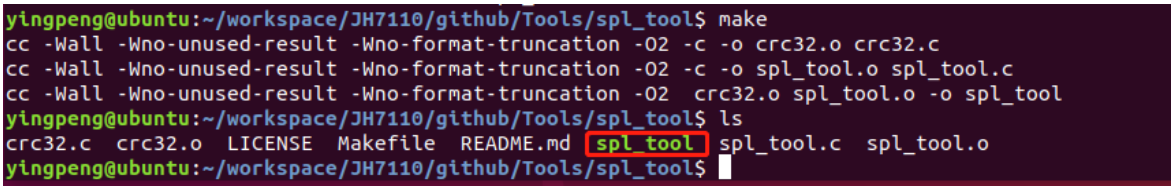
3. Switch to the code branch by executing the following command:

   ```
   cd Tools
   git checkout master
   git pull
   ```

4. Type the following to generate SPL tool under the `spl_tool` directory.

   ```
   cd spl_tool/
   make
   ```

Figure 2-7 Example Output



5. Type the following to generate SPL file:

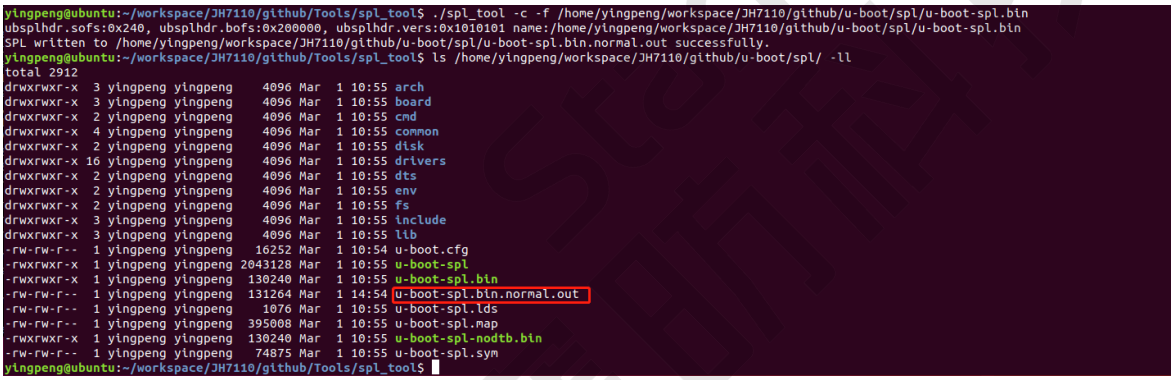
```
./spl_tool -c -f {U-BOOT_PATH}/spl/u-boot-spl.bin
```

Tip:
Modify the *{U-BOOT_PATH}* to the path of u-boot from before.

Result:

You will see a new file named `u-boot-spl.bin.normal.out` generated under `{U-BOOT_PATH}/spl`. Refer to *Updating SPL and U-Boot* section in *VisionFive 2 Single Board Computer Quick Start Guide* to flash `u-boot-spl.bin.normal.out`.

Figure 2-8 Example Output



2.1.5. Creating fw_payload File

Follow the steps below to create the `fw_payload` for VisionFive 2.

1. Locate to the tools directory, which git clone from before.

```
cd Tools/uboot_its
```

2. Copy the output file `fw_payload.bin` from the OpenSBI compilation to the tools path:

```
cp {OPENSBI_PATH}/build/platform/generic/firmware/fw_payload.bin ./
```

Note:
Modify the *{OPENSBI_PATH}* to the path of OpenSBI before executing.

3. Type the following to create `fw_payload` file under the `uboot_its` directory.

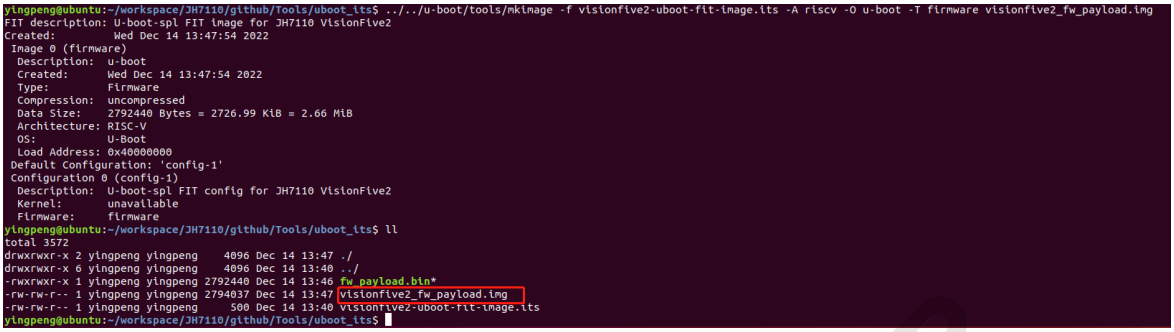
```
{U-BOOT_PATH}/tools/mkimage -f visionfive2-uboot-fit-image.its -A riscv -O u-boot -T firmware
 visionfive2_fw_payload.img
```

Note:
Remove the line break when copying this command from PDF.

Result:

You will see a new file named `visionfive2_fw_payload.img` generated. Refer to *Updating SPL and U-Boot* section in *VisionFive 2 Single Board Computer Quick Start Guide* to flash `visionfive2_fw_payload.img`.

Figure 2-9 Example Output



2.2. Compiling Linux Kernel

Follow the following steps to compile Linux Kernel for VisionFive 2.

1. Locate to your desired directory to store the Linux Kernel files. For example, the home directory.

   Example:

   ```
   cd ~ # home directory
   ```

2. Download the source code for Linux Kernel.

   ```
   git clone https://github.com/starfive-tech/linux.git
   ```

3. Switch to the code branch by executing the following commands:

   ```
   cd linux
   git checkout -b JH7110_VisionFive2_devel origin/JH7110_VisionFive2_devel
   git pull
   ```

4. Type the following to set the default configuration settings for compiling Linux Kernel.

   ```
   make <Configuration_File> CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
   ```

   Tip:
   <Configuration_File>: For VisionFive 2, the file is `starfive_visionfive2_defconfig`.

5. Type the following to set additional configuration settings for compiling Linux Kernel.

   ```
   make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv menuconfig
   ```

6. Compile the Linux Kernel.

   ```
   make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv -jx
   ```

   Note:
   Here you need to change the *-jx* value according to the number of cores in your CPU. If your CPU has 8 cores, change this to `-j8`. This process will take some time and therefore please wait patiently.

   Result:

   The kernel image will be generated inside the directory `linux/arch/riscv/boot` as `Image.gz`.

Figure 2-10 Example Output

```
jianlong@jianlong:~/work/jh7110/vf2/trm/linux/arch/riscv/boot$ ll
total 21964
drwxrwxr-x  3 jianlong jianlong     4096 10月 26 11:03 ./
drwxrwxr-x 10 jianlong jianlong     4096 10月 26 11:01 ../
drwxrwxr-x  6 jianlong jianlong     4096 10月 26 11:00 dts/
-rw-rw-r--  1 jianlong jianlong       83 10月 26 11:00 .gitignore
-rwxrwxr-x  1 jianlong jianlong 22016512 10月 26 11:03 Image*
-rw-rw-r--  1 jianlong jianlong      151 10月 26 11:03 .Image.cmd
-rw-rw-r--  1 jianlong jianlong  7744843 10月 26 11:03 Image.gz
-rw-rw-r--  1 jianlong jianlong      101 10月 26 11:03 .Image.gz.cmd
-rw-rw-r--  1 jianlong jianlong     1561 10月 26 11:00 install.sh
-rw-rw-r--  1 jianlong jianlong      206 10月 26 11:00 loader.lds.S
-rw-rw-r--  1 jianlong jianlong      143 10月 26 11:00 loader.S
-rw-rw-r--  1 jianlong jianlong     1612 10月 26 11:00 Makefile
jianlong@jianlong:~/work/jh7110/vf2/trm/linux/arch/riscv/boot$
```

The dtb files will be generated inside the directory `linux/arch/riscv/boot/dts/starfive`

Figure 2-11 Generated dtb Files

```
jianlong@jianlong:~/work/jh7110/vf2/trm/linux/arch/riscv/boot/dts/starfive$ ll *.dtb
-rw-rw-r-- 1 jianlong jianlong 64849 10月 26 11:01 jh7110-evb-can-pdm-pwmdac.dtb
-rw-rw-r-- 1 jianlong jianlong 64498 10月 26 11:01 jh7110-evb.dtb
-rw-rw-r-- 1 jianlong jianlong 64249 10月 26 11:01 jh7110-evb-dvp-rgb2hdmi.dtb
-rw-rw-r-- 1 jianlong jianlong 64713 10月 26 11:01 jh7110-evb-i2s-ac108.dtb
-rw-rw-r-- 1 jianlong jianlong 65144 10月 26 11:01 jh7110-evb-pcie-i2s-sd.dtb
-rw-rw-r-- 1 jianlong jianlong 64369 10月 26 11:01 jh7110-evb-spi-uart2.dtb
-rw-rw-r-- 1 jianlong jianlong 64405 10月 26 11:01 jh7110-evb-uart1-rgb2hdmi.dtb
-rw-rw-r-- 1 jianlong jianlong 64907 10月 26 11:01 jh7110-evb-uart4-emmc-spdif.dtb
-rw-rw-r-- 1 jianlong jianlong 65005 10月 26 11:01 jh7110-evb-uart5-pwm-i2c-tdm.dtb
-rw-rw-r-- 1 jianlong jianlong 64353 10月 26 11:01 jh7110-evb-usbdevice.dtb
-rw-rw-r-- 1 jianlong jianlong 63510 10月 26 11:01 jh7110-fpga.dtb
-rw-rw-r-- 1 jianlong jianlong 47299 10月 26 11:01 jh7110-visionfive-v2-A10.dtb
-rw-rw-r-- 1 jianlong jianlong 47491 10月 26 11:01 jh7110-visionfive-v2-A11.dtb
-rw-rw-r-- 1 jianlong jianlong 48381 10月 26 11:01 jh7110-visionfive-v2-ac108.dtb
-rw-rw-r-- 1 jianlong jianlong 47743 10月 26 11:01 jh7110-visionfive-v2.dtb
-rw-rw-r-- 1 jianlong jianlong 48252 10月 26 11:01 jh7110-visionfive-v2-wm8960.dtb
```

The `Image.gz` and `.dtb` files will be used later in this guide when we try to move rootfs, dtb and kernel to VisionFive 2.

Different boards use different dtb files:

- `jh7110-visionfive-v2.dtb`: for Version 1.2A and 1.3B board.
- `jh7110-visionfive-v2-ac108.dtb`: for version 1.2A and 1.3B board with ac108 codec.
- `jh7110-visionfive-wm8960.dtb`: for Version 1.2A and 1.3B board with wm8960 codec.

Tip:
You can refer to the silk print on the board for version information.

2.3. Updating Kernel

This chapter describes how to update kernel.

It contains the following sections:

2.3.1. Obtaining OS Version (Debian OS)

Steps:

1. Visit this link to download the latest operating system.
2. Flash the latest operating system to the Micro-SD card. For details, see *Flashing OS to a Micro-SD Card* section in *VisionFive 2 Single Board Computer Quick Start Guide*.

2.3.2. Adding New File

To add new file, perform the following steps:

Note:
If there is no update, no need to add the new file.

1. Insert the micro-SD card to the PC with Ubuntu system, and execute the following command to check the SD card partition:

   ```
   sudo fdisk -l
   ```

   Example Output:

   ```
   Device      Start     End Sectors  Size Type
   /dev/sdc1    4096    8191    4096    2M unknown
   /dev/sdc2    8192   16383    8192    4M unknown
   /dev/sdc3   16384  221183  204800  100M EFI System
   /dev/sdc4  221184 4503518 4282335    2G Linux filesystem
   ```

   In this output, the `/dev/sdc3` partition is the SD card partition.

2. Mount the SD card partition under the `mnt` file path by executing:

   ```
   sudo mount /dev/sdc3 /mnt
   ```

3. Compile the file under the `linux` directory with the following command:

   ```
   make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv INSTALL_PATH=<ROOTFS_PATH> zinstall -<jx>
   ```

   Tip:

   - *<ROOTFS_PATH>*: This is a user-defined directory where the vmliunz files will be generated.
   - *<jx>*: It refers to the number of cores in your CPU. If your CPU has 8 cores, change this to `-j8`.

   Figure 2-12 Example Command and Output

   ```
   atlas@atlas-VirtualBox:~/vf2/referenceTemp/linux$ make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv INSTALL_PATH
   =~/Desktop/compiled/ zinstall -j6
   sh ./arch/riscv/boot/install.sh 5.15.0 \
   arch/riscv/boot/Image.gz System.map "/home/atlas/Desktop/compiled/"
   ```

   Result:

   vmliunz files will be generated under the ROOTFS_PATH.

4. View the files generated under `${ROOTFS_PATH}`. The following is an example:

   Figure 2-13 Example

   ```
   atlas@atlas-VirtualBox:~/Desktop/compiled$ ls
   config-5.15.0  System.map-5.15.0  vmlinuz-5.15.0
   ```

5. Add the new file:

   a. View the Micro-SD card information.

      ```
      df -h
      ```

Figure 2-14 Example SD Card Information

b. Copy the kernel file to the Micro-SD card. Please operate under the path of *${ROOTFS_PATH}*.

```
sudo cp vmlinuz-5.15.0 /media/<User_Name>/root/boot/ && sync
```

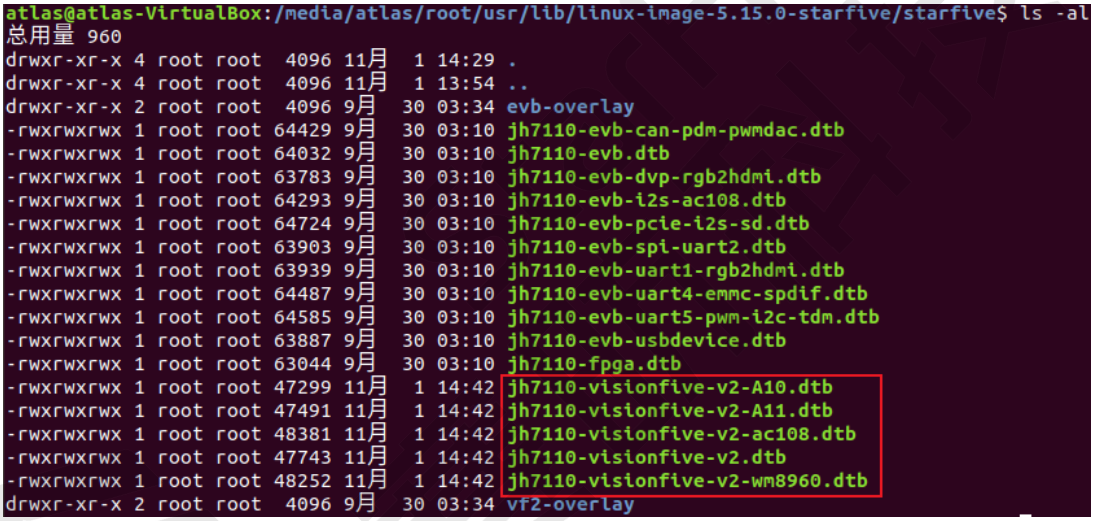
Tip:

<User_Name> is your username, for example, `atlas`.

c. Copy the modules file to the Micro-SD card. Please operate under the path of `linux/arch/riscv/boot/dts/starfive`.

```
sudo cp jh7110-visionfive-v2.dtb jh7110-visionfive-v2-A11.dtb jh7110-visionfive-v2-ac108.dtb
 jh7110-visionfive-v2-wm8960.dtb /media/<User_Name>/root/usr/lib/linux-image-5.15.0-
starfive/starfive/ && sync
```

Figure 2-15 dtb File List



6. Perform the following step to update the `extlinux.conf` file:

```
cd /mnt/extlinux
sudo vim extlinux.conf
```

7. Add the following command lines, save and exit:

```
label <Label >
        menu label <Menu_Label>
        linux /boot/<Newly_Complied_Kernel_file>
        initrd /boot/initrd.img-5.15.0-starfive
        fdtdir /usr/lib/linux-image-5.15.0-starfive/
        append  root=/dev/mmcblk1p3 rw console=tty0 console=ttyS0,115200 earlycon rootwait
 stmmaceth=chain_mode:1 selinux=0
```

Tip:

In these commands:

- *<Label>*: The label of startup item. For example, `l1`.
- *<Menu_Label>*: The configuration name displayed on the menu. For example, `myDebian_atlas GNU/Linux-5.15.0-starfive`.
- *<Newly_Complied_Kernel_file>*: The vmlinuz file name that is newly compiled in the previous steps. For example, `vmlinuz-5.15.0` or `Image.gz`.

Example: The following are the example commands:

```
label l1
menu label myDebian_atlas GNU/Linux-5.15.0-starfive
linux /boot/vmlinuz-5.15.0
initrd /boot/initrd.img-5.15.0-starfive
fdtdir /usr/lib/linux-image-5.15.0-starfive/
append  root=/dev/mmcblk1p3 rw console=tty0 console=ttyS0,115200 earlycon rootwait
 stmmaceth=chain_mode:1 selinux=0
```

8. (Optional) Load different dtb files:

   a. Execute the following commands to edit `uEnv.txt`:

   ```
   cd /mnt
   sudo vim uEnv.txt
   ```

   b. Modify the **fdtfile** parameter, save and exit:

   ```
   fdtfile=starfive/<dtb_File_Name>
   ```

   Note:

   - *<User_Name>*: Your user name. For example, `atlas`.
   - *<dtb_File_Name>*: The dtb files are located in `boot/usr/lib/linux-image-5.15.0-starfive/starfive/` as shown below:

     Different boards use different dtb files:

     - `jh7110-visionfive-v2.dtb`: for Version 1.2A and 1.3B board.
     - `jh7110-visionfive-v2-ac108.dtb`: for version 1.2A and 1.3B board with ac108 codec.
     - `jh7110-visionfive-wm8960.dtb`: for Version 1.2A and 1.3B board with wm8960 codec.

     Tip:
     You can refer to the silk print on the board for version information.

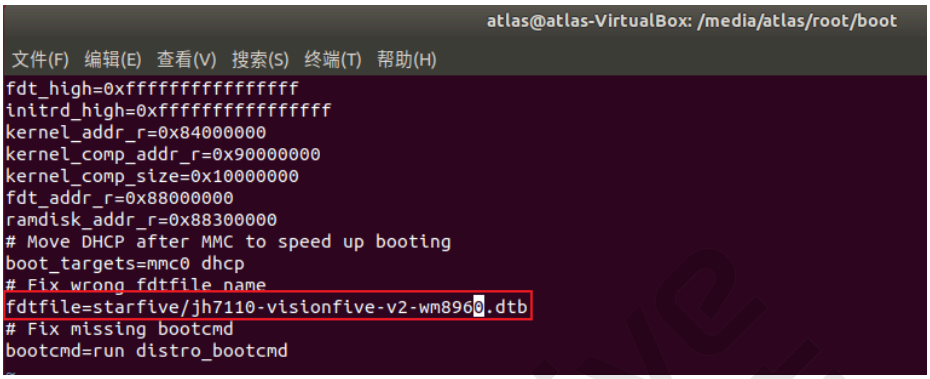
Figure 2-16 dtb File List

```
atlas@atlas-VirtualBox:/media/atlas/root/usr/lib/linux-image-5.15.0-starfive/starfive$ ls -al
总用量 960
drwxr-xr-x 4 root root  4096 11月  1 14:29 .
drwxr-xr-x 4 root root  4096 11月  1 13:54 ..
drwxr-xr-x 2 root root  4096 9月  30 03:34 evb-overlay
-rwxrwxrwx 1 root root 64429 9月  30 03:10 jh7110-evb-can-pdm-pwmdac.dtb
-rwxrwxrwx 1 root root 64032 9月  30 03:10 jh7110-evb.dtb
-rwxrwxrwx 1 root root 63783 9月  30 03:10 jh7110-evb-dvp-rgb2hdmi.dtb
-rwxrwxrwx 1 root root 64293 9月  30 03:10 jh7110-evb-i2s-ac108.dtb
-rwxrwxrwx 1 root root 64724 9月  30 03:10 jh7110-evb-pcie-i2s-sd.dtb
-rwxrwxrwx 1 root root 63903 9月  30 03:10 jh7110-evb-spi-uart2.dtb
-rwxrwxrwx 1 root root 63939 9月  30 03:10 jh7110-evb-uart1-rgb2hdmi.dtb
-rwxrwxrwx 1 root root 64487 9月  30 03:10 jh7110-evb-uart4-emmc-spdif.dtb
-rwxrwxrwx 1 root root 64585 9月  30 03:10 jh7110-evb-uart5-pwm-i2c-tdm.dtb
-rwxrwxrwx 1 root root 63887 9月  30 03:10 jh7110-evb-usbdevice.dtb
-rwxrwxrwx 1 root root 63044 9月  30 03:10 jh7110-fpga.dtb
-rwxrwxrwx 1 root root 47299 11月  1 14:42 jh7110-visionfive-v2-A10.dtb
-rwxrwxrwx 1 root root 47491 11月  1 14:42 jh7110-visionfive-v2-A11.dtb
-rwxrwxrwx 1 root root 48381 11月  1 14:42 jh7110-visionfive-v2-ac108.dtb
-rwxrwxrwx 1 root root 47743 11月  1 14:42 jh7110-visionfive-v2.dtb
-rwxrwxrwx 1 root root 48252 11月  1 14:42 jh7110-visionfive-v2-wm8960.dtb
drwxr-xr-x 2 root root  4096 9月  30 03:34 vf2-overlay
```

Example:

The following is the example of editing `uEnv.txt` to load the `jh7110-visionfive-v2-wm8960.dtb`:

Figure 2-17 Example uEnv.txt Content



9. Unmount the `/mnt` directory:

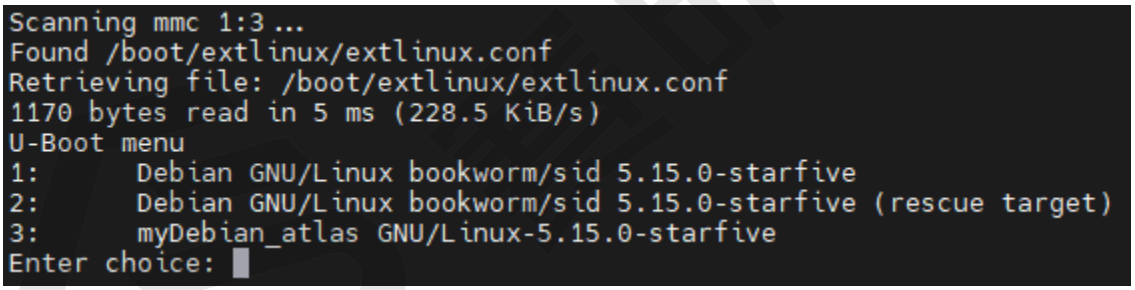
```
sudo umount /mnt
```

Verification:

The following steps are provided to verify if the configuration is successful:

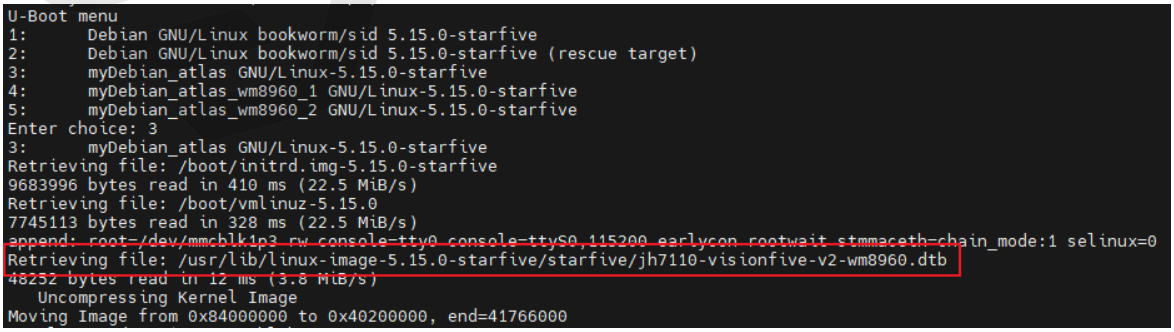
1. Pull out the card from the PC and insert it into the VisionFive 2 board. The system will start normally after power-on.
2. You can find the defined configuration item, for example, **myDebian_atlas GNU/Linux-5.15.0-starfive**, on the menu, as shown below:

   Figure 2-18 Example Interface

   Also, you can see the loaded dtb file, `jh7110-visionfive-v2-wm8960.dtb` on the startup interface.

   Figure 2-19 Startup Interface

3. After the system starts successfully, you can see the version of the new vmlinuz file:

```
root@starfive:~# cat /proc/version
Linux= version 5.15.0 (atlas@atlas-VirtualBox) (riscv64-linux-ginu-gcc
 (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0, GNU ld (GNU Binutils for Ubuntu) 2.30) #1 SMP Mon Oct 31 15:12:31
 CST 2022
root@starfive:~#
```

3. Making BusyBox System

This section describes how to make BusyBox system.

It contains the following sections:

3.1. Making File System

Follow the following steps to make the file system.

1. Create the directory structure.

```
mkdir rootfs
cd rootfs
mkdir dev usr bin sbin lib etc proc tmp sys var root mnt
```

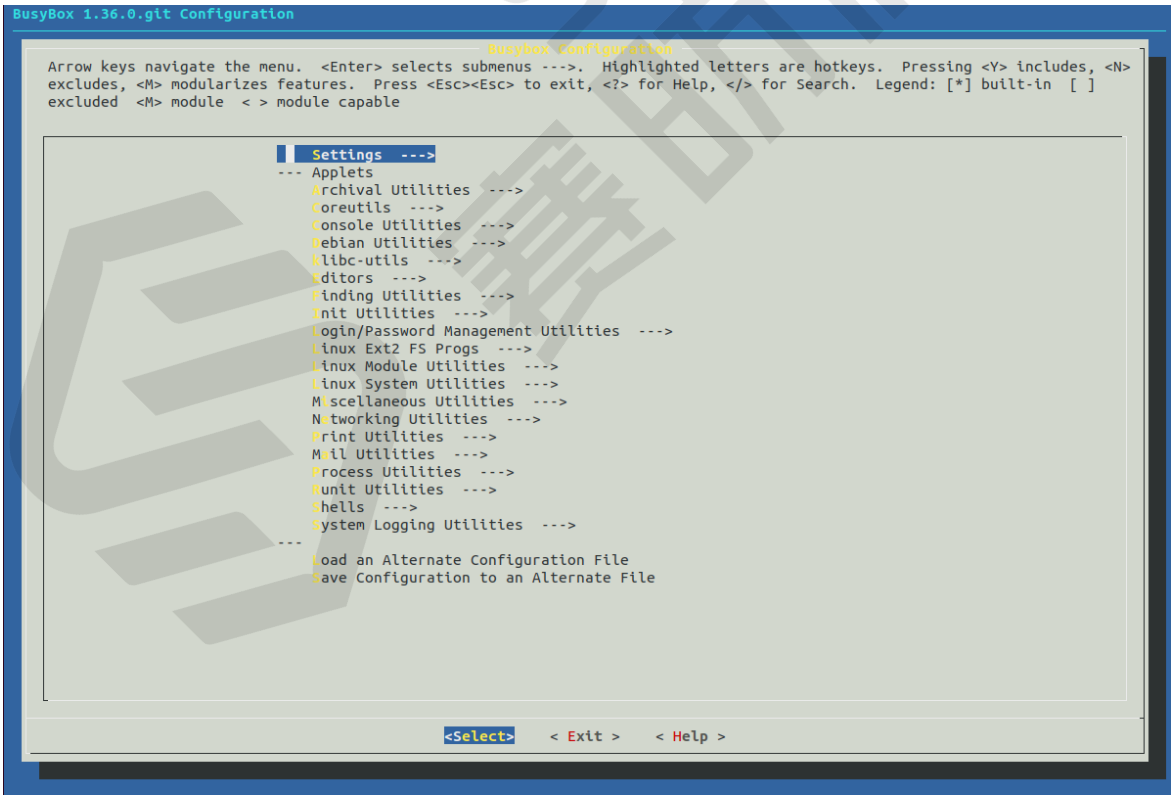
2. Download the BusyBox source code outside the rootfs directory.

```
git clone https://git.busybox.net/busybox
```

3. Navigate to the extracted location and enter BusyBox configuration.

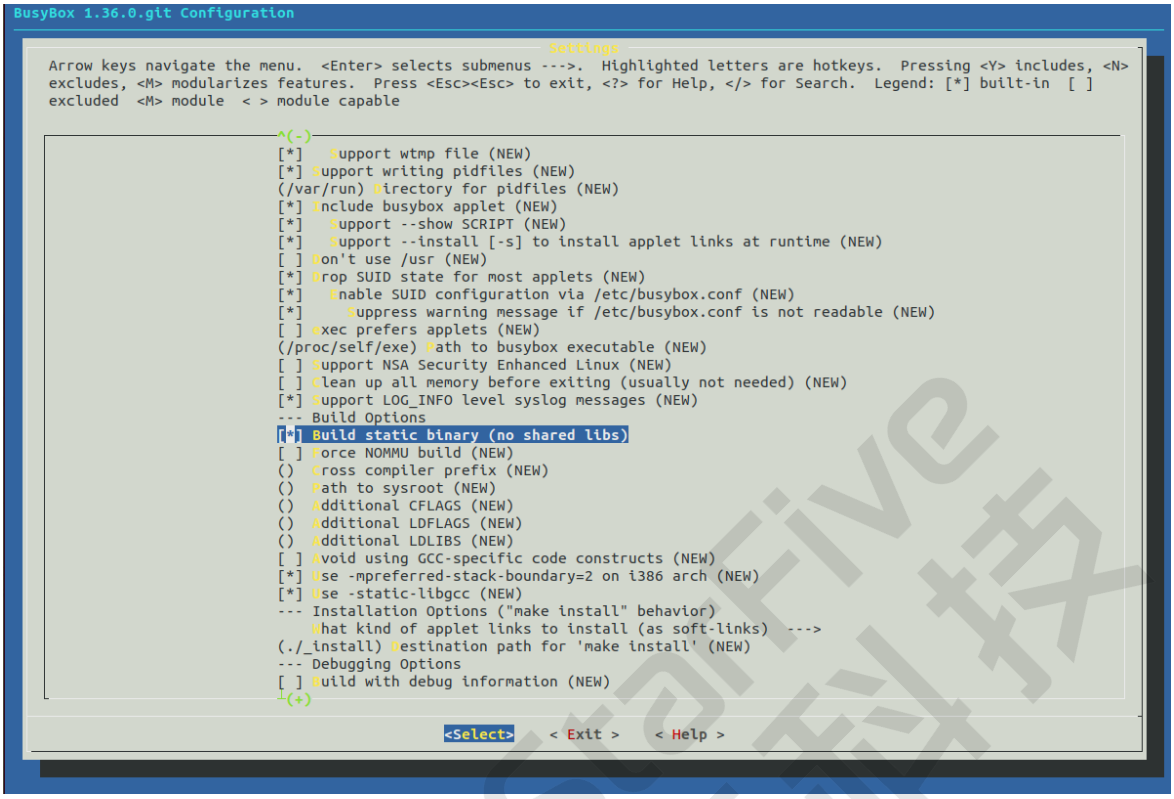
```
cd busybox
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv menuconfig
```

Figure 3-1 Busybox Configuration



4. Navigate to **Settings > Build Options** and check **Build static binary (no shared libs)** by pressing **Y.**

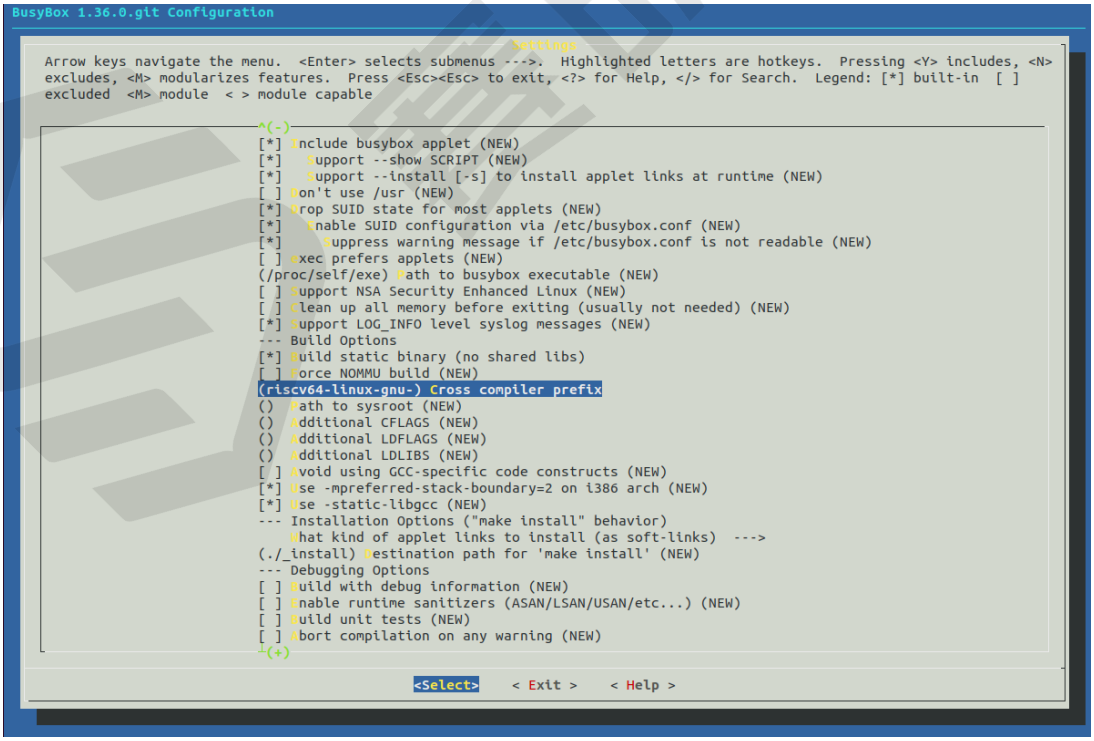
Figure 3-2 Check Build static binary (no shared libs)



5. Specify the compiler:

   a. Under **Build Options**, select **(riscv64-linux-gnu-) Cross compiler prefix**.

Figure 3-3 Select Cross Compiler Prefix



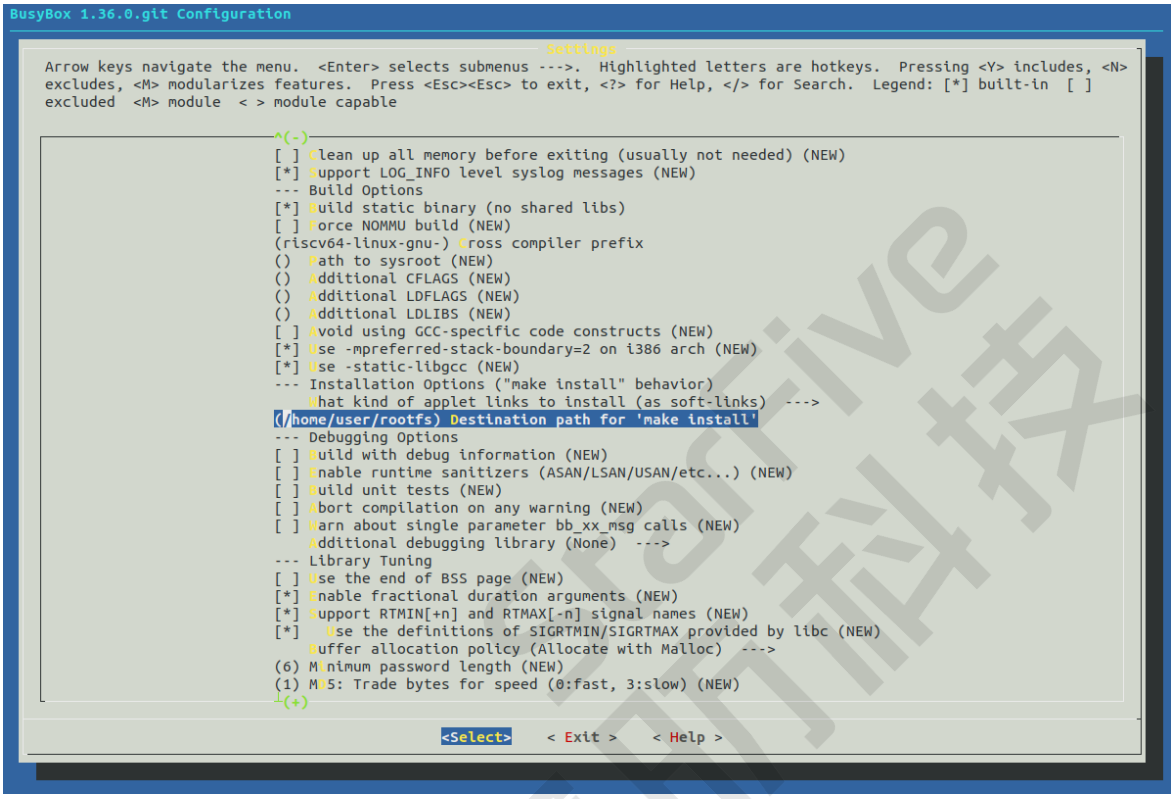
   b. Type the following command:

```
riscv64-linux-gnu-
```

6. Under **Installation Options > >Destination path for 'make install'**, change the path to the path of the rootfs file directory (this is the installation location of the compiled BusyBox).
   Example:

```
/home/user/rootfs
```

Figure 3-4 UI Example



7. Save the configuration and exit from the busybox configuration window.
8. Compile BusyBox.

```
make ARCH=riscv
```

9. Install BusyBox.

```
make install
```

10. Navigate to the `rootfs/etc` directory created before, create a file called `inittab` and open it using vim text editor.

```
cd rootfs/etc
vim inittab
```

11. Copy and paste the following content inside the `inittab` file.

```
::sysinit:/etc/init.d/rcS
::respawn:-/bin/login
::restart:/sbin/init
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
::shutdown:/sbin/swapoff -a
```

12. Create a file called profile inside `rootfs/etc` and open it using vim text editor.

```
vim profile
```

13. Copy and paste the following content inside the profile file.

```
# /etc/profile: system-wide .profile file for the Bourne shells
echo
```

```
# echo -n "Processing /etc/profile... "
# no-op
# Set search library path
# echo "Set search library path in /etc/profile"
export LD_LIBRARY_PATH=/lib:/usr/lib
# Set user path
# echo "Set user path in /etc/profile"
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH
# Set PS1
# Note: In addition to the SHELL variable, ash supports \u, \h, \W, \$, \!, \n, \w, \nnn (octal numbers
 corresponding to ASCII characters)
# And \e[xx;xxm (color effects), etc.
# Also add an extra '\' in front of it!
# echo "Set PS1 in /etc/profile"
export PS1="\\e[00;32m[$USER@\\w\\a]\\$\\e[00;34m"
# echo "Done"
```

14. Create a file called `fstab` inside `rootfs/etc` and open it using vim text editor.

```
vim fstab
```

15. Copy and paste the following content inside the `fstab` file.

```
proc /proc proc defaults 0 0
none /tmp tmpfs defaults 0 0
mdev /dev tmpfs defaults 0 0
sysfs /sys sysfs defaults 0 0
```

16. Create a file called `passwd` inside `rootfs/etc` and open it using vim text editor.

```
vim passwd
```

17. Copy and paste the following content inside the `passwd` file.

```
root:x:0:0:root:/root:/bin/sh
```

18. Create a file called `group` inside `rootfs/etc` and open it using vim text editor.

```
vim group
```

19. Copy and paste the following content inside the group file.

```
root:x:0:root
```

20. Create a file called `shadow` inside `rootfs/etc` and open it using vim text editor.

```
vim shadow
```

21. Copy and paste the following content inside the `shadow` file.

```
root:BAy5qvelNWKns:1:0:99999:7:::
```

22. Create a directory called `init.d` inside `rootfs/etc` and navigate inside it.

```
mkdir init.d
cd init.d
```

23. Create a file called `rcS` inside `rootfs/etc/init.d` and open it using vim text editor.

```
vim rcS
```

24. Copy and paste the following content inside the `rcS` file.

```
#! /bin/sh
#echo "----------mount all"
/bin/mount -a
#echo "----------Starting mdev......"
#/bin/echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
echo "*******************************************************"
echo " starfive mini RISC-V Rootfs"
```

```
echo "*******************************************************"
```

25. Navigate to the `rootfs/dev` directory created before and execute the following.

```
1 cd rootfs/dev
2 sudo mknod -m 666 console c 5 1
3 sudo mknod -m 666 null c 1 3
```

26. Create a soft link in the root directory of rootfs.

```
1 cd rootfs/
2 ln -s bin/busybox init
```

27. Modify the permissions of all files in the rootfs directory.

```
sudo chmod 777 -R *
```

28. Execute the following command in the rootfs directory to generate `rootfs.cpio.gz` (cpio file system package) in a different directory.

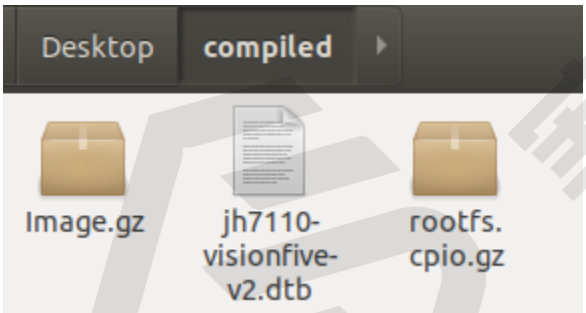
```
1 cd rootfs
2 find . | cpio -o -H newc | gzip > /home/user/Desktop/rootfs.cpio.gz
```

Note:
After you successfully run the command above, you will see a file named `rootfs.cpio.gz` on your Desktop. This directory can be any directory you want. If your CPU has 8 cores, change this to `-j8`. This process will take some time and therefore please wait patiently.

3.2. Moving Rootfs, Kernel, and dtb into VisionFive 2

Start by moving the previously compiled rootfs file system package, kernel and dtb images into a single directory.

Figure 3-5 Example Interface



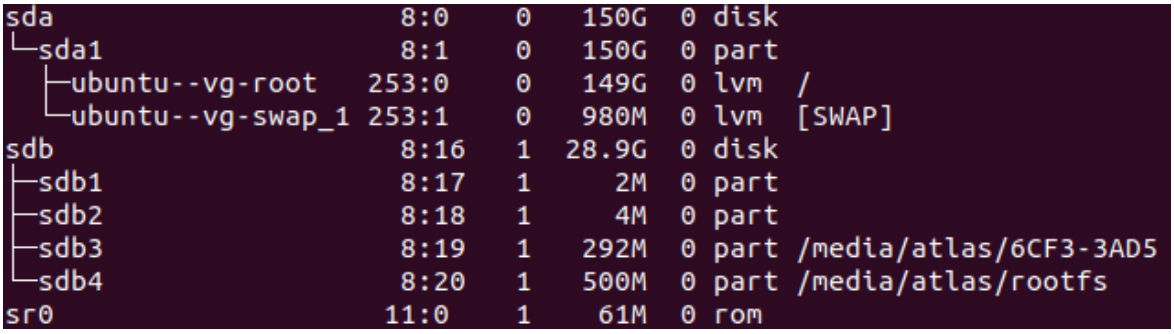
3.2.1. Method 1: Using Micro-SD Card

1. Insert a micro-SD card to the host PC.
2. Type the following to see the location of the connected micro-SD card.

```
lsblk
```

For example, it's `/dev/sdb`.

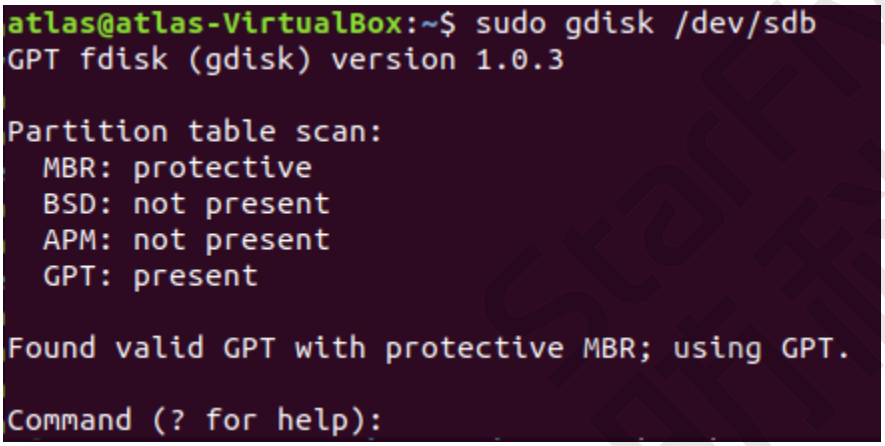
Figure 3-6 Example



3. Type the following to enter the partition configuration.

```
sudo gdisk /dev/sdb
```

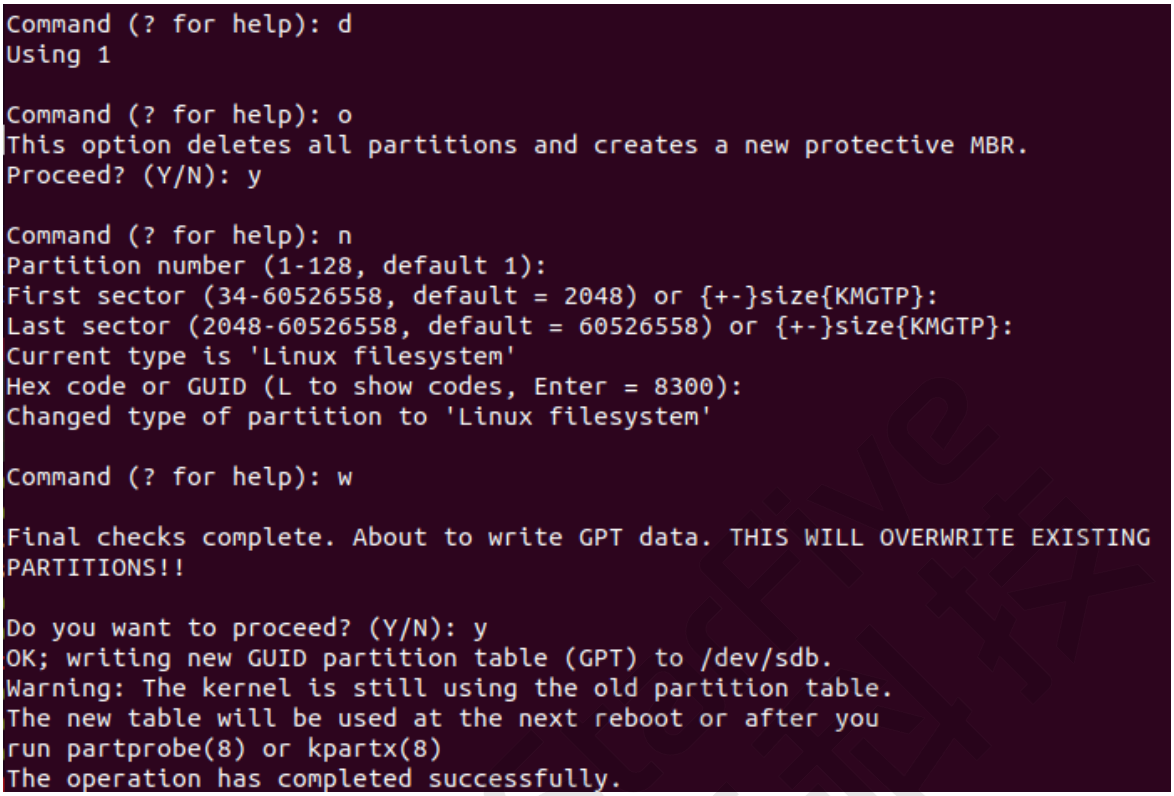
Figure 3-7 Example Output



4. Delete the original partition and then create a new partition by entering the following respectively.

```
d--->o--->n--->w--->y
```

Figure 3-8 Example Command and Output



Tip:
Press `Enter` to keep some settings to default in this configuration.

5. Format the micro-SD card and create the file system.

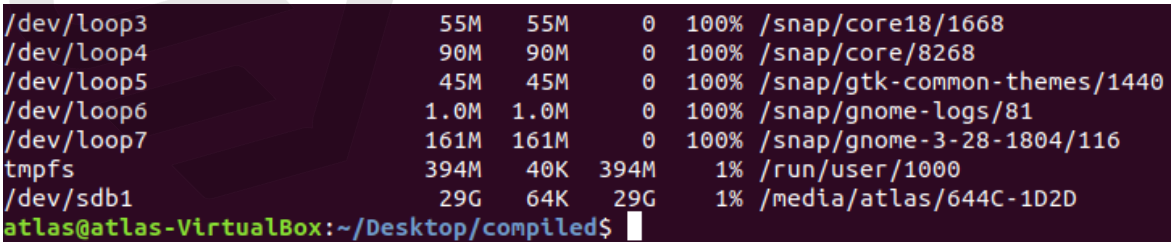
```
sudo mkfs.vfat /dev/sdb1
```

6. Remove the micro-SD card from PC and plug again to mount it.
7. Enter the following to check whether it gets mounted.

```
df -h
```

You will see an output as follows and take a note of the mount location.

Figure 3-9 Example Output



8. Navigate to the directory containing the 3 images as before.

```
cd Desktop/compiled
```

9. Copy the files to the micro-SD card by typing the following.

```
sudo cp Image.gz <Mount_Location>
sudo cp rootfs.cpio.gz <Mount_Location>
sudo cp <dtb_File_Name> <Mount_Location>
sync
```

Note:

- *<Mount_Location>*: the mount location as shown above.
- *<dtb_File_Name>*: the DTB file for VisionFive 2.

  Different boards use different dtb files:

  - `jh7110-visionfive-v2.dtb`: for Version 1.2A and 1.3B board.
  - `jh7110-visionfive-v2-ac108.dtb`: for version 1.2A and 1.3B board with ac108 codec.
  - `jh7110-visionfive-wm8960.dtb`: for Version 1.2A and 1.3B board with wm8960 codec.

Tip:
You can refer to the silk print on the board for version information.

Example:

The following are the example commands:

```
sudo cp Image.gz /media/user/644C-1D2D/
sudo cp rootfs.cpio.gz /media/user/644C-1D2D/
sudo cp jh7110-visionfive-v2.dtb /media/user/644C-1D2D/
sync
```

10. Remove the micro-SD card from PC, insert into VisionFive 2 and turn it on.
11. Open minicom while USB to Serial Adapter is connected between VisionFive 2 and PC, and wait until the board enters **U-Boot** mode. You will see the following output when it is in U-Boot mode.

Figure 3-10 Example Output

```
U-Boot 2021.10-00044-g135126c47b-dirty (Oct 28 2022 - 16:36:03 +0800)

CPU:   rv64imacu
Model: StarFive VisionFive V2
DRAM:  8 GiB
MMC:   sdio0@16010000: 0, sdio1@16020000: 1
```

12. Enter the following commands.

```
setenv kernel_comp_addr_r 0xb0000000;setenv kernel_comp_size 0x10000000;
fatls mmc 1:1
fatload mmc 1:1 ${kernel_addr_r} Image.gz
fatload mmc 1:1 ${fdt_addr_r} jh7110-visionfive-v2.dtb
fatload mmc 1:1 ${ramdisk_addr_r} rootfs.cpio.gz
run chipa_set_linux;
booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Figure 3-11 Example Command and Output

```
StarFive # setenv kernel_comp_addr_r 0xb0000000;setenv kernel_comp_size 0x10000000;
StarFive # fatls mmc 1:1
            System Volume Information/
  7745113   Image.gz
    47743   jh7110-visionfive-v2.dtb
  1211720   rootfs.cpio.gz

3 file(s), 1 dir(s)

StarFive # fatload mmc 1:1 ${kernel_addr_r} Image.gz
7745113 bytes read in 330 ms (22.4 MiB/s)
StarFive # fatload mmc 1:1 ${fdt_addr_r} jh7110-visionfive-v2.dtb
47743 bytes read in 4 ms (11.4 MiB/s)
StarFive # fatload mmc 1:1 ${ramdisk_addr_r} rootfs.cpio.gz; run chipa_set_linux;
1211720 bytes read in 54 ms (21.4 MiB/s)
StarFive # booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
   Uncompressing Kernel Image
## Flattened Device Tree blob at 46000000
   Booting using the fdt blob at 0x46000000
   Using Device Tree in place at 0000000046000000, end 000000004600ea7e

Starting kernel ...
```

13. Log in by typing the following credentials.
    - Username: root
    - Password: starfive

3.2.2. Method 2: Using Ethernet Cable

1. Connect an Ethernet Cable from the RJ45 port of VisionFive 2 to a router, connect serial adapter cable and power on the board.

   Note:
   Make sure the host PC is also connected to the same router using Ethernet or Wi-Fi.

2. Open **minicom** and wait until the board enters **U-Boot** mode. You will see the following output when it is in U-Boot mode.

   Figure 3-12 Example Output

```
U-Boot 2021.07-rc4-g2d3dd06117-dirty (Jun 20 2021 - 21:03:05 +0800)

CPU:   rv64imafdc
DRAM:  8 GiB
MMC:   sdio0@10000000: 0, sdio1@10010000: 1
Loading Environment from nowhere... OK
Net:   dwmac.10020000
Autoboot in 2 seconds
MMC CD is 0x1, force to True.
MMC CD is 0x1, force to True.
Card did not respond to voltage select! : -110
```

3. Enter the following commands to set U-Boot environment variables.

```
setenv serverip 192.168.125.142;setenv ipaddr 192.168.125.200;setenv hostname starfive;setenv netdev
 eth0;setenv kernel_comp_addr_r 0xb0000000;setenv kernel_comp_size 0x10000000; setenv bootargs
 console=ttyS0,115200 earlycon=sbi root=/dev/ram0 stmmaceth=chain_mode:1 loglevel=8
```

   Note:
   Generally, the default IP of a router is 192.168.120.1. In this case, use the server IP as the IP assigned by the DHCP server of the router and use the VisionFive 2 IP as 192.168.120.xxx. However, if your router IP is different (e.g.: 192.168.2.1), the server and VisionFive 2 should follow the IP format of 192.168.2.xxx.

4. Check the connectivity by pinging the host PC from VisionFive 2.

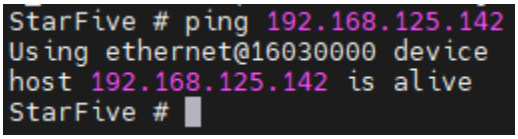
Example:

```
ping 192.168.120.12
```

Result:

If you see the following output, the host PC and VisionFive 2 have established communication on the same network.

Figure 3-13 Example Output



5. Install a TFTP server on the Host PC.

```
sudo apt-get update
sudo apt install tftpd-hpa
```

6. Check the status of the server.

```
sudo systemctl status tftpd-hpa
```

7. Execute the following to enter the TFTP server configuration.

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

8. Configure the TFTP server as follows.

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/home/user/Desktop/compiled"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

Note:
The `TFTP_DIRECTORY` is the directory that we created before with all the 3 images (`Image.gz`, `jh7110-visionfive-v2.dtb`, `rootfs.cpio.gz`).

9. Restart the TFTP server.

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

10. Type the following inside the U-Boot mode of VisionFive 2 to download the files from the TFTP server of the host PC and start the kernel.

```
tftpboot ${fdt_addr_r} <dtb_File_Name>;tftpboot ${kernel_addr_r} Image.gz;tftpboot ${ramdisk_addr_r}
 rootfs.cpio.gz;run chipa_set_linux;booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Note:

Example:

The following command is an example for VisionFive 2:

```
tftpboot ${fdt_addr_r} jh7110-visionfive-v2.dtb;tftpboot ${kernel_addr_r} Image.gz;tftpboot
 ${ramdisk_addr_r} rootfs.cpio.gz;run chipa_set_linux;booti ${kernel_addr_r}
 ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Result:

```
starfive mini RISC-V Rootfs
```

11. Log in with the following credentials.
    - Username: root
    - Password: starfive