StarFive 赛昉科技

VisionFive SBC
Technical Reference Manual

Version: V1.1

Date: 2022-02-17

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.

StarFive expressly 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.

StarFive does 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.

Shanghai StarFive Technology Co., Ltd.

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

Website: www.starfivetech.com

Email: sales@starfivetech.com (sales)

support@starfivetech.com (support)

About This Manual

Introduction

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

Revision History

Version	Released	Revision
V1	2021-12-08	The first official release.
V1.1	2022-02-17	Updated the command in step 12 in the *Method 1: Using Micro-SD Card* section.

Table of Contents

1 Required Hardware

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

- VisionFive
- Micro-SD card (16GB or more)
- PC with Linux OS
- USB to Serial Converter
- Ethernet cable
- Power adapter
- USB Type-C cable

Information:

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

2 Compiling Firmware

Alternatively, if you are interested in compiling bootloader and ddr init from source code, you can follow the guideline in this chapter.

2.1 Preparing Compilation Environment

To prepare a compilation environment, perform the following steps:

Steps:

Visit this link and download the latest version of riscv64-unknown-elf-toolchain-xxx according to your operating system.

Step 1 Unzip the downloaded toolchain by typing the following:

```
tar -xzvf <Toolchain_Name>
```

Information:

<Toolchain_Name> refers to the name of the downloaded toolchain in the previous step. For example, `riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-linux-ubuntu14.tar.gz`.

Step 2 Open .bashrc file by typing the following:

```
gedit ~/.bashrc
```

Step 3 Add the following line, save and exit:

```
export PATH=$PATH:<Unzipped_Compiler_Path>/bin
```

Information:

<Unzipped_Compiler_Path> refers to the location of the unzipped compiler. For example, `/home/yingpeng/Downloads/riscv64-unknown-elf-toolchain-xxx/`.

Example:

```
export PATH=$PATH:/home/yingpeng/Downloads/riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-linux-ubuntu14/bin
```

Step 4 Type the following to make the change effective:

```
source ~/.bashrc
```

Command Example and Result:

The following figure shows an example command and the result:

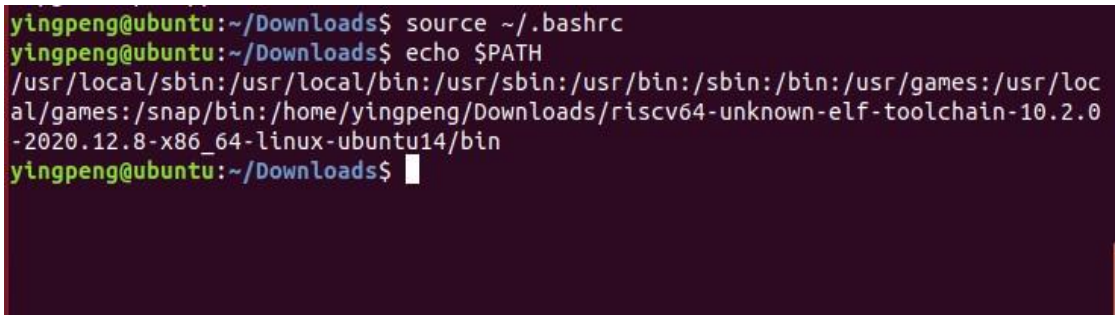

Figure 2-1 Example Output

2.2 Compiling Bootloader

To compile Bootloader, perform the following steps:

Steps:

Step 1 Clone the repo from GitHub.

```
git clone https://github.com/starfive-tech/JH7100_secondBoot.git
```

Step 2 Go into the build directory.

```
cd JH7100_secondBoot/build
```

Step 3 Compile bootloader.

```
make
```

Result:

You will see the following output if the compilation is successful.

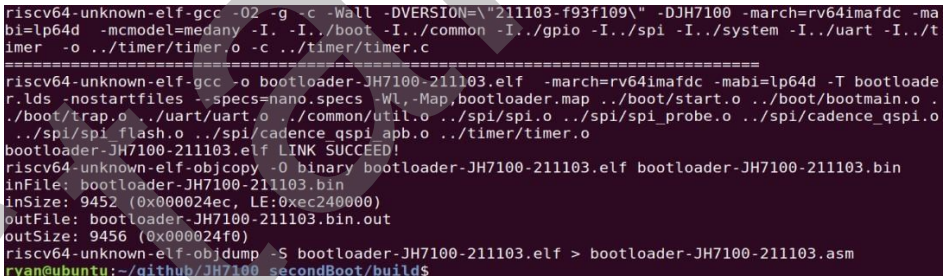

Figure 2-2 Example Output

2.2.2 Compiling ddr init

To compile ddr init, perform the following:

Steps:

Step 1 Clone the repo from GitHub.

```
git clone https://github.com/starfive-tech/JH7100_ddrinit.git
```

Step 2 Go into build directory.

```
cd JH7100_ddrinit/build
```

Step 3 Compile ddr init.

```
make
```

Result:

You will see the following output if the compilation is successful.

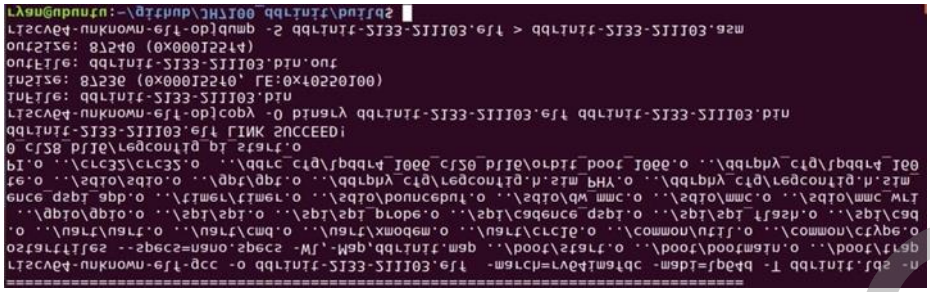

Figure 2-3 Example Output

Information:

Refer to Appendix B: Updating Firmware and u-boot section in VisionFive_Single_Board_Computer_Quick_Start_Guide to flash bootloader and ddr init.

3 Making General System

3.1 Compiling u-boot and Kernel

3.1.1 Set Up Compilation Environment

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

Steps:

Step 1 Install the riscv64-linux-gnu-gcc compiler from Ubuntu packages.

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

Step 2 Check the version of the riscv64-linux-gnu-gcc.

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

Result:

The output will be as follows:

```
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)
```

Figure 3-1 Example Output

3.1.2 Compiling the u-boot

Follow the steps below to compile the u-boot for VisionFive.

Steps:

Step 1 Locate to your desired directory to store the u-boot files. For example, the home directory.

Example:

```
cd ~ # home directory
```

Step 2 Download the source code for u-boot compilation.

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

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

```
cd u-boot
git checkout -b JH7100_upstream origin/JH7100_upstream
git pull
```

Step 4 Type the following to compile u-boot under the u-boot directory.

```
make <Configuration_File> ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
make u-boot.bin u-boot.dtb ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
```

Information:

`Configuration_File:`

- For VisionFive, the file is `starfive_jh7100_visionfive_smode_defconfig`.
- For Starlight, the file is `starfive_jh7100_starlight_smode_defconfig`.

Result:

There will be these 2 files generated after compilation inside the u-boot directory: `u-boot.bin` and `u-boot.dtb`.

```
-rwxrwxr-x  1 ryan ryan 7224016 Nov  9 18:14 u-boot
-rw-rw-r--  1 ryan ryan  921147 Nov  9 18:14 u-boot.bin
-rw-rw-r--  1 ryan ryan      47 Nov  9 18:14 .u-boot.bin.cmd
-rw-rw-r--  1 ryan ryan   15585 Nov  9 18:14 u-boot.cfg
-rw-rw-r--  1 ryan ryan     953 Nov  9 18:14 .u-boot.cmd
-rw-rw-r--  1 ryan ryan   41643 Nov  9 18:14 u-boot.dtb
-rw-rw-r--  1 ryan ryan  921147 Nov  9 18:14 u-boot-dtb.bin
```

Figure 3-2 Example Output

Information:

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

3.1.3 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 3-3 Typical Boot Flow

Follow the steps below to compile OpenSBI for VisionFive.

Steps:

Step 2 Locate to your desired directory to store the OpenSBI files. For example, the home directory.

Example:

```
cd ~ # home directory
```

Step 3 Download the source code for OpenSBI compilation.

```
git clone https://github.com/riscv/opensbi.git
```

Step 4 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}/u-boot.dtb
```

Information:

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.

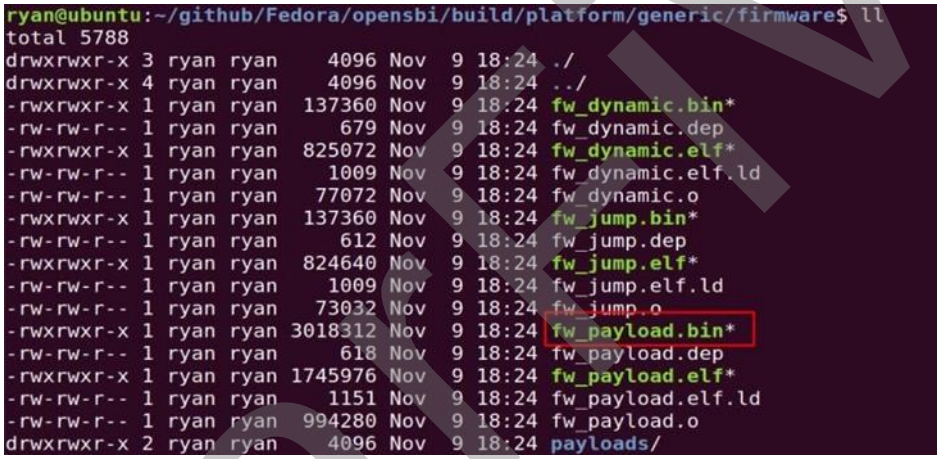

Figure 3-4 Example Output

Step 5 Navigate to the directory containing fw_payload.bin.

```
cd opensbi/build/platform/generic/firmware
```

Step 6 Copy the file fw_payload.bin to a different location.

```
cp fw_payload.bin ~/Desktop/payload/
```

Step 7 Navigate to the location where `fw_payload.bin` is copied and execute the following to install an image conversion tool.

```
cd ~ /Desktop/payload/

sudo apt install subversion

svn export https://github.com/starfive-tech/freelight-u-sdk.git/branches/starfive/fsz.sh
```

Information:

Here is the source code.

Step 8 Change the user rights of the tool.

```
chmod 777 fsz.sh
```

Step 9 Convert the file from fw_payload.bin to fw_payload.bin.out.

```
./fsz.sh fw_payload.bin fw_payload.bin.out
```

```
ryan@ubuntu:~/Desktop/payload$ ./fsz.sh fw_payload.bin fw_payload.bin.out
inFile: fw_payload.bin
inSize: 3018312 (0x002e0e48, LE:0x480e2e00)
outFile: fw_payload.bin.out
outSize: 3018316 (0x002e0e4c)
```

Figure 3-5 Example Output

> **Information:**
>
> You will see a new file named `fw_payload.bin.out` generated. Refer to Appendix B: Updating Firmware and u-boot section in VisionFive_Single_Board_Computer_Quick_Start_Guide to flash u-boot.

3.1.4 Compiling Linux Kernel

Follow the following steps to compile Linux Kernel for VisionFive.

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

Example:

```
cd ~ # home directory
```

Step 2 Download the source code for Linux Kernel.

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

Step 3 Type the following to set the default configuration settings for compiling Linux Kernel.

```
cd linux
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv visionfive_defconfig
```

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

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

Step 5 Compile the Linux Kernel.

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

> **Information:**
>
> 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`.

```
yingpeng@ubuntu:~/Desktop/github/linux/arch/riscv/boot$ ll
total 17628
drwxrwxr-x  3 yingpeng yingpeng     4096 Nov 22 16:18 ./
drwxrwxr-x 11 yingpeng yingpeng     4096 Nov 22 16:15 ../
drwxrwxr-x  6 yingpeng yingpeng     4096 Nov 18 18:52 dts/
-rw-rw-r--  1 yingpeng yingpeng       83 Nov 18 18:52 .gitignore
-rwxrwxr-x  1 yingpeng yingpeng 19723776 Nov 22 16:18 Image*
-rw-rw-r--  1 yingpeng yingpeng      151 Nov 22 16:18 .Image.cmd
-rw-rw-r--  1 yingpeng yingpeng  6353268 Nov 22 16:18 Image.gz
-rw-rw-r--  1 yingpeng yingpeng      101 Nov 22 16:18 .Image.gz.cmd
-rw-rw-r--  1 yingpeng yingpeng     1561 Nov 18 18:52 install.sh
-rw-rw-r--  1 yingpeng yingpeng      206 Nov 18 18:52 loader.lds.S
-rw-rw-r--  1 yingpeng yingpeng      143 Nov 18 18:52 loader.S
-rw-rw-r--  1 yingpeng yingpeng     1612 Nov 18 18:52 Makefile
```

Figure 3-6 Example Output

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

```
yingpeng@ubuntu:~/Desktop/github/linux/arch/riscv/boot/dts/starfive$ ls
jh7100-beaglev-starlight-a1.dtb  jh7100.dtsi
jh7100-beaglev-starlight-a1.dts  jh7100-starfive-visionfive-v1.dtb
jh7100-beaglev-starlight.dtb     jh7100-starfive-visionfive-v1.dts
jh7100-beaglev-starlight.dts     Makefile
jh7100-common.dtsi
```

Figure 3-7 Generated dtb Files

The Image.gz and .dtb files will be used later in this guide when we try to move rootfs, dtb and kernel to VisionFive. Different boards use different dtb files, and for the detailed information, refer to the dtb Files table in StarFive_40-Pin_GPIO_Header_User_Guide.

3.2 Updating Kernel and Modules

3.2.1 Obtaining OS Version（e.g.: Fedora OS）

Steps:

Step 1 Visit StarFive's official GitHub to download the latest operating system.

Step 2 Flash the latest operating system to the Micro-SD card. For details, see Flash Fedora OS to Micro-SD Card section in VisionFive_Single_Board_Computer_Quick_Start_Guide.

3.2.2 Adding New File

To add new file, perform the following steps:

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

Step 1 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>
```

> **Information**:
>
> - *<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.

Example Command and Output:

```
yingpeng@ubuntu:~/Desktop/github/linux$ make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
INSTALL_PATH=~/Desktop/github/boot zinstall -j4
sh ./arch/riscv/boot/install.sh 5.16.0-rc2-visionfive-g6e924cb10a60 \
arch/riscv/boot/Image.gz System.map "/home/yingpeng/Desktop/github/boot"
yingpeng@ubuntu:~/Desktop/github/linux$
```

Figure 3-8 Example Command and Output

Result:

vmliunz files will be generated under the ROOTFS_PATH.

Step 2 (Optional) Compile the file under the linux path with the following command:

```
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv INSTALL_MOD_PATH=<ROOTFS_PATH> modules_install -jx
```

Information:

If you need to add new modules, this step is required.

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

Example Command and Output:

```
yingpeng@ubuntu:~/Desktop/github/linux$ make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
INSTALL_MOD_PATH=~/Desktop/github/boot modules_install -j4
  DEPMOD  /home/yingpeng/Desktop/github/boot/lib/modules/5.16.0-rc2-visionfive-g6e924cb10
a60
yingpeng@ubuntu:~/Desktop/github/linux$
```

Figure 3-9 Example Command and Output

Result:

The module files will be generated under the ROOTFS_PATH.

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

```
yingpeng@ubuntu:~/Desktop/github/boot$ ls
config-5.16.0-rc2-visionfive-g6e924cb10a60
lib
System.map-5.16.0-rc2-visionfive-g6e924cb10a60
vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60
yingpeng@ubuntu:~/Desktop/github/boot$
```

Figure 3-10 Example

Step 3 Add the new file:

Sub-steps:

1. View the SD-card information.

```
df -h
```

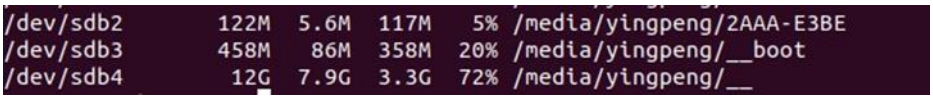

Figure 3-11 Example SD Card Information

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

```
sudo cp vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60 /media/<User_Name>/__boot/ && sync
```

Information:

<User_Name> is your username, for example, yingpeng.

3. (Optional) Copy the modules file to the Micro-SD card. Please operate under the path of ${ROOTFS_PATH}.

```
sudo cp -r lib/modules/5.16.0-rc2-visionfive-g6e924cb10a60/ /media/<User_Name>/__/lib/modules && sync
```

Information:

- If you have compiled modules in the Compile Module step, and you need to add these modules, this step is required.
- *<User_Name>* is your username, for example, yingpeng.

Step 4 Perform the following step to update the grub.cfg file:

```
cd /media/<UserName>/__boot/
sudo gedit grub.cfg
```

Information:

<User_Name> is your username, for example, yingpeng.

Step 5 Add the following command lines, save and exit:

```
menuentry '<Configuration_Item_Name_on_Menu>' {
linux /<Newly_Compiled_vmlinuz_file> ro root=UUID=59fcd098-2f22-441a-ba45-4f7185baf23f rhgb console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0 LANG=en_US.UTF-8
devicetree /dtbs/5.16.0-rc2+/starfive/<dtb_File_Name>
initrd /initramfs-5.16.0-rc2+.img
}
```

Information:

In these commands:

- *<Configuration_Item_Name_on_Menu>*: The configuration name displayed on the menu. For example, My Fedora vmlinux-5.16.0-rc2 visionfive.
- *<Newly_Compiled_vmlinuz_file>*: The vmlinuz file name that is newly compiled in the previous steps. For example, vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60.
- *<dtb_File_Name>*: Different boards use different dtb files, and for the detailed

information, refer to the *dtb Files* table in StarFive_40-Pin_GPIO_Header_User_Guide.

Example:

The following are the example commands:

```
menuentry 'My Fedora vmlinux-5.16.0-rc2 visionfive' {
linux /vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60 ro root=UUID=59fcd098-2f22-441a-ba45-4f7185baf23f rhgb console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0 LANG=en_US.UTF-8
devicetree /dtbs/5.16.0-rc2+/starfive/jh7100-starfive-vision-five-v1.dtb
initrd /initramfs-5.16.0-rc2+.img
}
```

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 board. The system will start normally after power-on.
2. You can find the defined configuration item, for example, **My Fedora vmlinux-5.16.0-rc2 visionfive**, on the menu, as shown below:

Figure 3-12 Example Interface

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

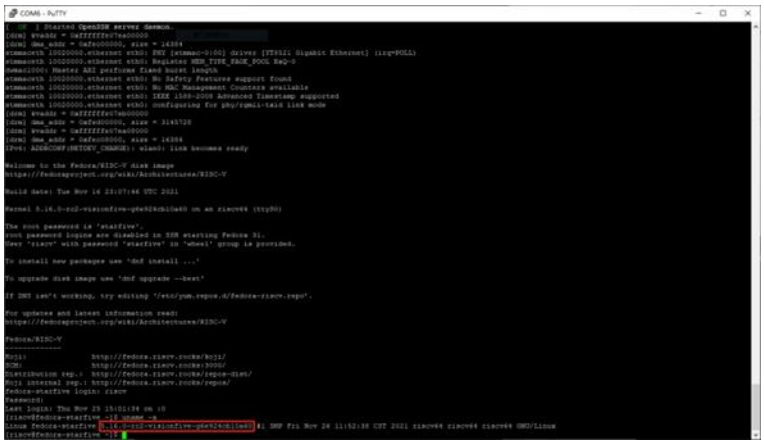

Figure 3-13 Example Output

4 Making Busybox System

4.1 Making File System

Follow the following steps to make the file system.

Steps:

Step 1 Create the directory structure.

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

Step 2 Download the busybox source code outside the rootfs directory.

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

Step 3 Navigate to the extracted location and enter busybox configuration.

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

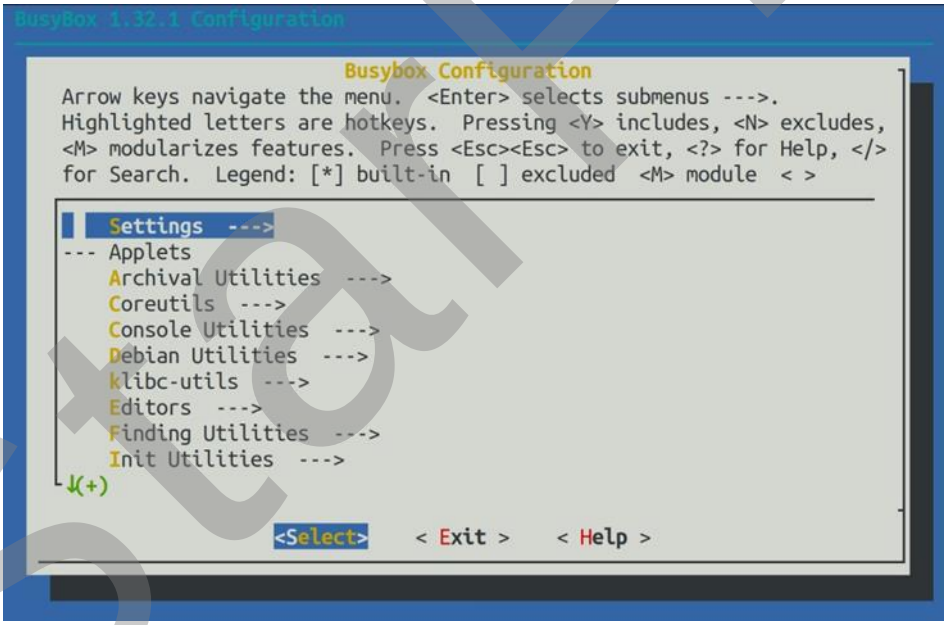

Figure 4-1 Busybox Configuration

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

Step 5 Under Build Options, select cross compiler prefix and type the following to specify the compiler.

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

Step 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
```

Step 7 Exit from the busybox configuration window and save the configuration.

```
Compile busybox.
make ARCH=riscv
```

Step 8 Install busybox.

```
make install
```

Step 9 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
```

Step 10 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
```

Step 11 Create a file called profile inside rootfs/etc and open it using vim text editor.

```
vim profile
```

Step 12 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"
```

Step 13 Create a file called fstab inside rootfs/etc and open it using vim text editor.

```
vim fstab
```

Step 14 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
```

Step 15 Create a file called passwd inside rootfs/etc and open it using vim text editor.

```
vim passwd
```

Step 16 Copy and paste the following content inside the passwd file.

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

Step 17 Create a file called group inside rootfs/etc and open it using vim text editor.

```
vim group
```

Step 18 Copy and paste the following content inside the group file.

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

Step 19 Create a file called shadow inside rootfs/etc and open it using vim text editor.

```
vim shadow
```

Step 20 Copy and paste the following content inside the shadow file.

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

Step 21 Create a directory called init.d inside rootfs/etc and navigate inside it.

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

Step 22 Create a file called rcS inside rootfs/etc/init.d and open it using vim text editor.

```
vim rcS
```

Step 23 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
"*******************************************************"
```

Step 24 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
```

Step 25 Create a soft link in the root directory of rootfs.

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

Step 26 Modify the permissions of all files in the rootfs directory.

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

Step 27 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
```

> **Information:**
>
> 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.

4.2 Moving Rootfs, Kernel, and dtb into VisionFive

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

Figure 4-2 Example Interface

<dtb_File_Name>: Different boards use different dtb files, and for the detailed information, refer to the dtb Files table in StarFive_40-Pin_GPIO_Header_User_Guide.

Method 1: Using Micro-SD Card

Steps:

Step 1 Insert a micro-SD card to the host PC.

Step 2 Type the following to see the location of the connected micro-SD card.

```
lsblk
```

For example, it's /dev/sdb.

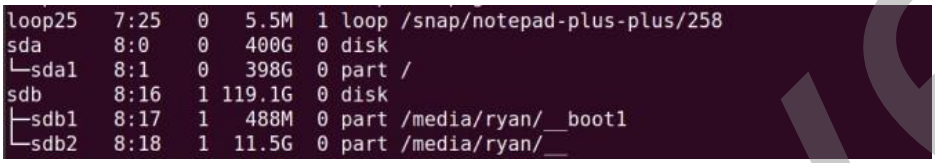

Figure 4-3 Example

Step 3 Type the following to enter the partition configuration.

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

Example Output:

Figure 4-4 Example Output

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

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

Information:

Press **Enter** to keep some settings to default in this configuration.

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

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

Step 6 Remove the micro-SD card from PC and plug again to mount it.

Step 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.

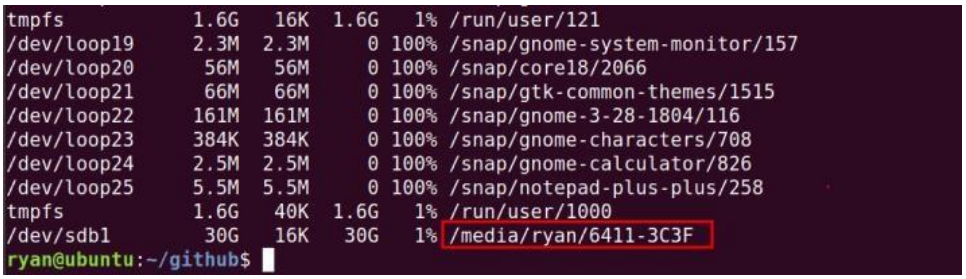

Figure 4-5 Example Output

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

Example:

```
cd Desktop/compiled
```

Step 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
```

Information:

<Mount_Location>: is the mount location as shown above.

<dtb_File_Name>: Different boards use different dtb files, and for the detailed information, refer to the dtb Files table in StarFive_40-Pin_GPIO_Header_User_Guide.

Example:

The following are the example commands:

```
sudo cp Image.gz /media/user/6411-3C3F/
sudo cp rootfs.cpio.gz /media/user/6411-3C3F/
sudo cp jh7100-starfive-visionfive-v1.dtb /media/user/6411-3C3F/
sync
```

Step 10 Remove the micro-SD card from PC, insert into VisionFive and turn it on.

Step 11 Open minicom while USB to Serial Adapter is connected between VisionFive and PC, and wait until the board enters u-boot mode. You will see the following output when it is in u-boot mode.

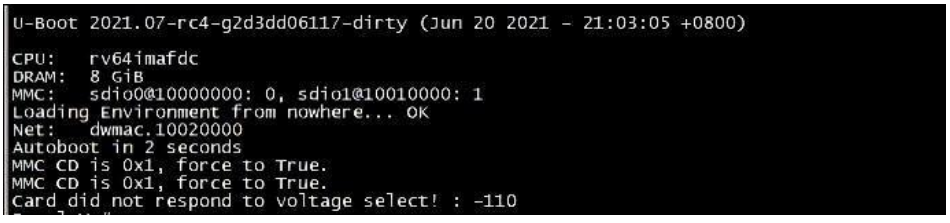

Figure 4-6 Example Output

Step 12 Enter the following commands.

```
setenv kernel_comp_addr_r 0x90000000;setenv kernel_comp_size 0x10000000;setenv kernel_addr_r 0x84000000;setenv fdt_addr_r 0x88000000;setenv ramdisk_addr_r 0x88300000

fatls mmc 0:1

fatload mmc 0:1 ${kernel_addr_r} Image.gz

fatload mmc 0:1 ${fdt_addr_r} jh7100-starfive-visionfive-v1.dtb

fatload mmc 0:1 ${ramdisk_addr_r} rootfs.cpio.gz

booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Step 13 Log in by typing the following credentials.

- **Username**: root
- **Password**: starfive

Method 2: Using Ethernet Cable

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

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

Open minicom and wait until the board enters u-boot mode. You will see the following output when it is in u-boot mode.

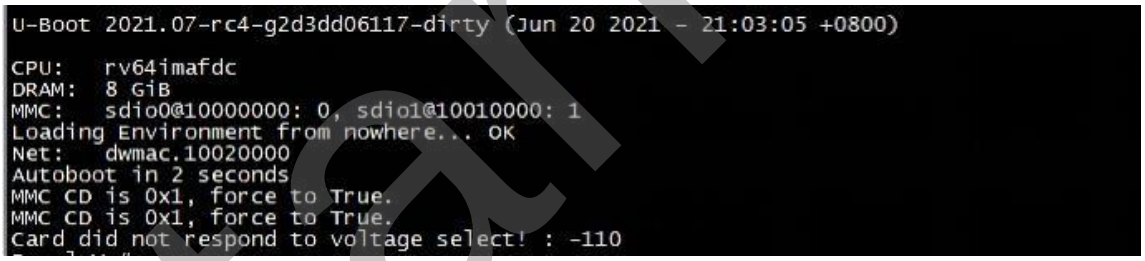

Figure 4-7 Example Output

Step 14 Enter the following commands to set u-boot environment variables.

```
setenv serverip 192.168.120.12;setenv ipaddr 192.168.120.200;setenv hostname starfive;setenv netdev eth0;setenv kernel_comp_addr_r 0x90000000;setenv kernel_comp_size 0x10000000; setenv bootargs console=ttyS0,115200 earlycon=sbi root=/dev/ram0 stmmaceth=chain_mode:1 loglevel=8
```

> **Information**:
>
> 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 IP as 192.168.120.xxx. However, if your router IP is different (e.g.: 192.168.2.1), the server and VisionFive should follow the IP format of 192.168.2.xxx.

Step 15 Check the connectivity by pinging the host PC from VisionFive.

Example:

```
ping 192.168.120.12
```

Result:

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

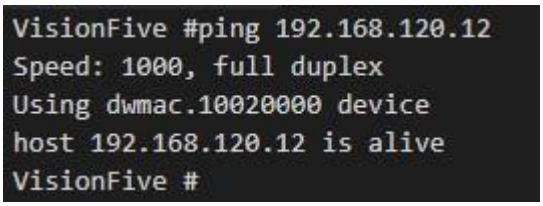

Figure 4-8 Example Output

Step 16 Install a tftp server on the Host PC.

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

Step 17 Check the status of the server.

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

Step 18 Execute the following to enter the tftp server configuration.

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

Step 19 Configure the tftp server as follows.

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

Information:

The TFTP_DIRECTORY is the directory that we created before with all the 3 images (Image.gz, jh7100-starfive-visionfive-v1.dtb, rootfs.cpio.gz).

Step 20 Restart the tftp server.

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

Step 21 Type the following inside the u-boot mode of VisionFive 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;booti ${ker-
nel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

<dtb_File_Name>: Different boards use different dtb files, and for the detailed information, refer to the dtb Files table in StarFive_40-Pin_GPIO_Header_User_Guide.

Example:

The following command is an example for VisionFive:

```
tftpboot ${fdt_addr_r} jh7100-starfive-visionfive-
v1.dtb;tftpboot ${kernel_addr_r} Image.gz;tftpboot
```

```
${ramdisk_addr_r}     rootfs.cpio.gz;booti     ${kernel_addr_r}
${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Step 22 Log in with the following credentials.

- **Username**: root
- **Password**: starfive