StarFive 赛昉科技

VisionFive 2 SDK Quick Start Guide

Version: 1.1

Date: 2022/12/27

Doc ID: VisionFive2-QSGEN-002

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 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 arising 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

Website: http://www.starfivetech.com

Email:

- Sales: sales@starfivetech.com
- Support: support@starfivetech.com

Contents

List of Tables

List of Figures

Preface

About this guide and technical support information.

About this document

This document mainly provides the SDK developers with a quick start reference when they configure VisionFive 2, the first high-performance RISC-V single board computer (SBC) with an integrated GPU. VisionFive 2 is equipped with the StarFive next generation SoC platform - JH7110

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.0	2022/12/13	The first official release.
1.1	2022/12/27	• Added a new method in Updating SPL and U-Boot *(on page 17)*. • Added example output figures in Recovering the Boot-loader *(on page 19)*.

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.

1. Introduction

StarFive provides Software Development Kit (SDK) to build U-Boot SPL, U-Boot, and a flattened image tree (FIT) image with an OpenSBI binary, Linux kernel, device tree, ramdisk, and rootdisk for VisionFive 2. It also builds a complete RISC-V cross-compile toolchain for VisionFive 2.

Follow the procedures to use the SDK:

2. Prerequisites

Before using the SDK, ensure you perform the following steps:

1. Install an Operating System (OS) on your PC.

   Tip:
   The recommended OS is Ubuntu 16.04/18.04/20.04.

2. Perform the following commands to update all packages:

```
$sudo apt update
$sudo apt upgrade
```

3. Perform the following command to install the required additional packages:

```
$ sudo apt-get install build-essential g++ git autoconf automake autotools-dev texinfo
bison xxd curl flex gawk gdisk gperf libgmp-dev libmpfr-dev libmpc-dev libz-dev libssl-
dev libncurses-dev libtool patchutils python screen texinfo unzip zlib1g-dev libyaml-
dev wget cpio bc dosfstools mtools device-tree-compiler libglib2.0-dev libpixman-1-dev kpartx
```

4. Execute the following commands to install additional packages for Git LFS:

```
$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
$ sudo apt-get install git-lfs
```

3. Download the SDK

Perform the following steps to download the SDK from the StarFive official GitHub repository.

1. Check out the SDK repository (for example, branch **JH7110_VisionFive2_devel**) and check out all of the linked sub-modules by executing the following commands:

```
$ git clone git@github.com:starfive-tech/VisionFive2.git
$ cd VisionFive2
$ git checkout JH7110_VisionFive2_devel
$ git submodule update --init --recursive
```

Note:
This will take some time and requires around 5 GB of disk space. Some modules may fail because certain dependencies don't have the best git hosting. The only solution is to wait and try again later (or ask someone for a copy of that source repository).

2. (Optional) For users who build the release tag version, step 1 *(on page 9)* is enough. For developers who need to switch the 4 submodules (namely, `buildroot`, `u-boot`, `linux`, and `opensbi`) to correct branch, you can run the following commands. You can refer to the `.gitmodule` file for the correct branch information.

```
$ cd buildroot && git checkout --track origin/<buildroot_branch> && cd ..
$ cd u-boot && git checkout --track origin/<u-boot_branch> && cd ..
$ cd linux && git checkout --track origin/<linux_branch> && cd ..
$ cd opensbi && git checkout <opensbi_branch> && cd ..
```

The following are the example commands:

```
$ cd buildroot && git checkout --track origin/JH7110_VisionFive2_devel && cd ..
$ cd u-boot && git checkout --track origin/JH7110_VisionFive2_devel && cd ..
$ cd linux && git checkout --track origin/JH7110_VisionFive2_devel && cd ..
$ cd opensbi && git checkout master && cd ..
```

4. Build Instructions

This section provides steps to build instructions after updating the sub-modules as described in Download the SDK *(on page 9)*. This procedure is quick building for the initramfs image, `image.fit`, which could be translated to the board through TFTP and could run on the board.

1. Run the following command to build the toolchain, `u-boot-spl.bin.normal.out`, `visionfive2_fw_payload.img`, and `image.fit`.

   ```
   make -j$(nproc)
   ```

   Result:

   The following files will be generated under `work/` directory:

   ```
   work/
   ├── visionfive2_fw_payload.img
   ├── image.fit
   ├── initramfs.cpio.gz
   ├── u-boot-spl.bin.normal.out
   ├── linux/arch/riscv/boot
       ├── dts
       │   └── starfive
       │       ├── jh7110-visionfive-v2-ac108.dtb
       │       ├── jh7110-visionfive-v2.dtb
       │       ├── jh7110-visionfive-v2-wm8960.dtb
       │       ├── vf2-overlay
       │       │   └── vf2-overlay-uart3-i2c.dtbo
       └── Image.gz
   ```

   The final build tree will consume about 16 GB of disk space.

2. Copy the previously-generated files to the TFTP server workspace path.

5. Configuring or Building Buildroot, U-Boot, Linux Kernel, and BusyBox

Use the following commands to configure and build Buildroot, U-Boot, Linux kernel and BusyBox.

Configuring buildroot

Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross complication.

Use the following commands to configue Buildroot on your board.

```
$ make buildroot_initramfs-menuconfig # initramfs menuconfig
$ make buildroot_rootfs-menuconfig # rootfs menuconfig
```

Configuring U-boot

Universal Boot Loader (U-Boot) is an open source, primary boot loader used in embedded devices.

Use the following command to configure U-Boot on your board.

```
$ make uboot-menuconfig # uboot menuconfig
```

Configuring Linux Kernel

The Linux kernel is the main component of a Linux *Operating System (OS)* and is the core interface between a computer's hardware and its processes.

Use the following command to build the Linux kernel.

```
$ make linux-menuconfig # Kernel menuconfig
```

Configuring BusyBox

BusyBox is a convenient tool-set containing stripped-down versions of many Linux utilities.

Use the following commands to build the BusyBox menu configuration.

- To build menu configuration for BusyBox with the Initramfs file system:

  ```
  $ make -C ./work/buildroot_initramfs/ O=./work/buildroot_initramfs busybox-menuconfig
  ```

- To build menu configuration for BusyBox with the Rootfs file system:

  ```
  $ make -C ./work/buildroot_rootfs/ O=./work/buildroot_rootfs busybox-menuconfig
  ```

Building Linux Kernel, BusyBox, and FFmpeg

If you want to build single package or module, perform the following commands according to your needs:

- To build Linux Kernel:

  ```
  $ make vmlinux
  ```

- To build BusyBox:

  ```
  make -C ./work/buildroot_rootfs/ O=./work/buildroot_rootfs busybox-rebuild # build
  busybox package
  ```

- To build FFmpeg package:

  ```
  $ make -C ./work/buildroot_rootfs/ O=./work/buildroot_rootfs ffmpeg-rebuild
  ```

6. Run SDK on VisionFive 2

Perform the following steps to run SDK on VisionFive 2:

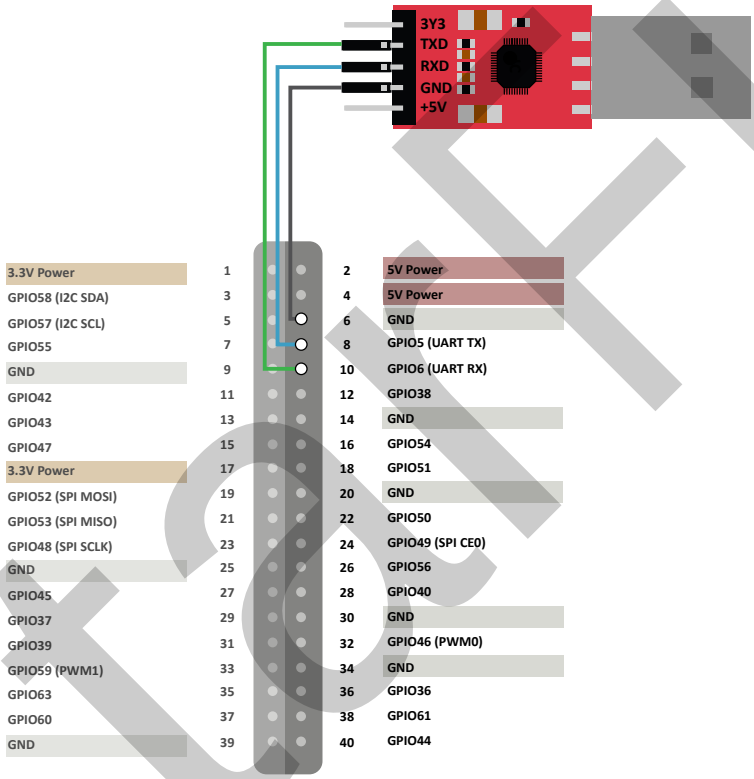
1. Connect your VisionFive 2 to network as described in Connect VisionFive 2 to Network *(on page 12)*.
2. Boot VisionFive 2 by performing one of the following methods:
   - Run the Default DTB with image.fit *(on page 14)*
   - Run the Other DTB Files with the Image.gz and initramfs.cpio.gz *(on page 14)*

6.1. Connect VisionFive 2 to Network

This section provides steps to connect VisionFive 2 to network and enter U-Boot terminal.

1. Connect the jumper wires between the USB-to-Serial converter and the Debug pins of VisionFive 2 40-pin GPIO header. The following figure is an example:

   Figure 6-1 Connecting to the Debug Pins of VisionFive 2 40-pin GPIO Header



2. Configure the serial port baud rate settings to `115200` bps.
3. Connect VisionFive 2 to the network cable and power cord.
4. Turn on VisionFive 2 and you will see the start-up information as follows:

```
U-Boot SPL 2021.10 (Oct 31 2022 - 12:11:37 +0800)
DDR version: dc2e84f0.
Trying to boot from SPI

OpenSBI v1.0
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
```

```
        |_|

Platform Name             : StarFive VisionFive V2
Platform Features         : medeleg
Platform HART Count       : 5
Platform IPI Device       : aclint-mswi
Platform Timer Device     : aclint-mtimer @ 4000000Hz
Platform Console Device   : uart8250
Platform HSM Device       : ---
Platform Reboot Device    : ---
Platform Shutdown Device  : ---
Firmware Base             : 0x40000000
Firmware Size             : 360 KB
Runtime SBI Version       : 0.3

Domain0 Name              : root
Domain0 Boot HART         : 3
Domain0 HARTs             : 0*,1*,2*,3*,4*
Domain0 Region00          : 0x0000000002000000-0x000000000200ffff (I)
Domain0 Region01          : 0x0000000040000000-0x000000004007ffff ()
Domain0 Region02          : 0x0000000000000000-0xffffffffffffffff (R,W,X)
Domain0 Next Address      : 0x0000000040200000
Domain0 Next Arg1         : 0x0000000042200000
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes

Boot HART ID              : 3
Boot HART Domain          : root
Boot HART Priv Version    : v1.11
Boot HART Base ISA        : rv64imafdcbx
Boot HART ISA Extensions  : none
Boot HART PMP Count       : 8
Boot HART PMP Granularity : 4096
Boot HART PMP Address Bits: 34
Boot HART MHPM Count      : 2
Boot HART MIDELEG         : 0x0000000000000222
Boot HART MEDELEG         : 0x000000000000b109


U-Boot 2021.10 (Oct 31 2022 - 12:11:37 +0800), Build: jenkins-VF2_515_Branch_SDK_Release-10

CPU:   rv64imacu
Model: StarFive VisionFive V2
DRAM:  8 GiB
MMC:   sdio0@16010000: 0, sdio1@16020000: 1
Loading Environment from SPIFlash... SF: Detected gd25lq128 with page size 256 Bytes, erase size 4 KiB,
 total 16 MiB
*** Warning - bad CRC, using default environment

StarFive EEPROM format v2

--------EEPROM INFO--------
Vendor : StarFive Technology Co., Ltd.
Product full SN: VF7110A1-2243-D008E000-00000001
data version: 0x2
PCB revision: 0xa1
BOM revision: A
Ethernet MAC0 address: 6c:cf:39:00:14:5b
Ethernet MAC1 address: 6c:cf:39:00:14:5c
--------EEPROM INFO--------

In:    serial@10000000
Out:   serial@10000000
Err:   serial@10000000
Model: StarFive VisionFive V2
Net:   eth0: ethernet@16030000, eth1: ethernet@16040000
switch to partitions #0, OK
mmc1 is current device
found device 1
bootmode flash device 1
Failed to load 'uEnv.txt'
Can't set block device
```

```
Hit any key to stop autoboot:  0
StarFive #
```

5. Press any key to stop and enter U-Boot terminal.

6.2. Boot VisionFive 2

Choose one of the following methods to boot the board according the DTB file you want to use.

- Run the Default DTB with image.fit *(on page 14)*
- Run the Other DTB Files with the Image.gz and initramfs.cpio.gz *(on page 14)*

Different boards use different dtb files:

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

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

6.2.1. Run the Default DTB with `image.fit`

This section provides steps to transfer `image.fit` through TFTP and run `image.fit` with the default DTB: `jh7110-visionfivev2.dtb`.

1. Run the following command to set environment parameters:

```
setenv bootfile vmlinuz; setenv fdtcontroladdr 0xffffffffffffffff; setenv fileaddr a0000000; setenv
 ipaddr 192.168.xxx.xxx; setenv serverip 192.168.xxx.xxx;
```

2. Upload the image file to ddr:

```
tftpboot ${fileaddr} ${serverip}:image.fit;
```

3. Load and execute the file by running:

```
bootm start ${fileaddr};bootm loados ${fileaddr};run chipa_set_linux;booti 0x40200000
 0x46100000:${filesize} 0x46000000
```

4. Login with the following credentials:

```
buildroot login:root
Password: starfive
```

Result:

The launch is successful!

6.2.2. Run the Other DTB Files with the `Image.gz` and `initramfs.cpio.gz`

If you want to load the other DTBs, for example, `jh7110-visionfive-v2-wm8960.dtb`, follow the steps below.

1. Set the environment parameter:

```
setenv bootfile vmlinuz; setenv fdtcontroladdr 0xffffffffffffffff; setenv fileaddr a0000000; setenv
 ipaddr 192.168.xxx.xxx; setenv serverip 192.168.xxx.xxx;
setenv kernel_comp_addr_r 0xb0000000;setenv kernel_comp_size 0x10000000;
```

2. Upload files to DDR:

```
tftpboot ${fdt_addr_r} jh7110-visionfive-v2-wm8960.dtb;
tftpboot ${kernel_addr_r} Image.gz;
```

```
tftpboot ${ramdisk_addr_r} initramfs.cpio.gz;
run chipa_set_linux;
```

3. Load and execute:

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

4. Login with the following credentials:

```
buildroot login:root
Password: starfive
```

Result:

The launch is successful!

7. Appendix

7.1. Generating Booting TF Card

If you don't use a local TFTP server, then you may want to make the TF card target.

Important:
The operation will overwrite all existing data on the target TF card.

The default size is 16 GB. The GPT Partition Table for the TF card is recommended.

Perform the following steps to generate the booting SD card:

1. Execute the following commands to generate `sdcard.img` file.

```
$ make -j$(nproc)
$ make buildroot_rootfs -j$(nproc)
$ make img
```

Result:

The output file `work/sdcard.img` will be generated.

Tip:
The image file can be burned into a TF card by:

- executing the following `dd` command:

```
$ sudo dd if=work/sdcard.img of=/dev/sdX bs=4096
$ sync
```

- or using `rpi-imager` or `balenaEtcher` tool.

2. (Optional) Extend the partition if needed. The following methods are both applicable:

- Option 1: On the Ubuntu host:

a. Install the package by running the following command on the Ubuntu host:

```
$ sudo apt install cloud-guest-utils e2fsprogs
```

b. Insert the TF card to the Ubuntu host.

c. Execute the following command to extend partition.

Note:
`/dev/sdX` is the TF card device name. Change this variable according the actual situation.

```
$ sudo growpart /dev/sdX 4  # extend partition 4
$ sudo e2fsck -f /dev/sdX4
$ sudo resize2fs /dev/sdX4  # extend filesystem
$ sudo fsck.ext4 /dev/sdX4
```

- Option 2: Run the `fdisk` and `resize2fs` commands on the VisionFive 2 board:

```
# fdisk /dev/mmcblk1
Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
This disk is currently in use - repartitioning is probably a bad idea.
It's recommended to umount all file systems, and swapoff all swap
partitions on this disk.
Command (m for help): d
Partition number (1-4, default 4): 4
```

```
Partition 4 has been deleted.
Command (m for help): n
Partition number (4-128, default 4): 4
First sector (614400-62333918, default 614400):
): t sector, +/-sectors or +/-size{K,M,G,T,P} (614400-62333918, default 62333918)
Created a new partition 4 of type 'Linux filesystem' and of size 29.4 GiB.
Partition #4 contains a ext4 signature.
Do you want to remove the signature? [Y]es/[N]o: N
Command (m for help): w
The partition table has been altered.
Syncing disks.

# resize2fs /dev/mmcblk1p4
resize2fs 1.46.4 (18-Aug-2021)
Filesystem at /d[
111.756178] EXT4-fs (mmcblk1p4): resizing filesystem from 512000
to 30859756 blocks
ev/mmcblk1p4 is [
111.765203] EXT4-fs (mmcblk1p4): resizing filesystem from 512000
to 30859265 blocks
mounted on /; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 118
[ 112.141953] random: crng init done
[ 112.145369] random: 7 urandom warning(s) missed due to ratelimiting
[ 115.474184] EXT4-fs (mmcblk1p4): resized filesystem to 30859265
The filesystem on /dev/mmcblk1p4 is now 30859756 (1k) blocks long.
```

7.2. Using DTB Overlay Dynamically

The system support loading DTB overlay dynamically when the board is running.

```
mount -t configfs none /sys/kernel/config
mkdir -p /sys/kernel/config/device-tree/overlays/dtoverlay
cd <the dtoverlay.dtbo path>
cat vf2-overlay-uart3-i2c.dtbo > /sys/kernel/config/device-tree/overlays/dtoverlay/dtbo
```

Besides, you could use the following command to remove the DTBO (Device Tree Blob for Overlay) feature:

```
rmdir /sys/kernel/config/device-tree/overlays/dtoverlay
```

7.3. Updating SPL and U-Boot

To update SPL and U-Boot for VisionFive 2, two methods are provided:

Note:
For instructions to create SPL and `fw_payload` (U-Boot) files, refer to *Creating SPL File* and *Creating fw_payload File* sections in the *VisionFive 2 Single Board Computer Software Technical Reference Manual*.

1. Through the `tftpboot` command as described in Through tftpboot Command *(on page 17)*.
2. Through the `flashcp` command as described in Through flashcp Command *(on page 18)*.

Note:
Method 2 only supports versions equal to or later than `VF2_v2.5.0`.

Through `tftpboot` Command

To update SPL and U-Boot through the `tftpboot` command, perform the following steps:

1. Prepare the TFTP server. The following is an example command for Ubuntu distribution.

   ```
   sudo apt install tftpd-hpa
   ```

2. Power on VisionFive 2 and wait until it enters the U-Boot command line interface.

3. Configure the environment variables by executing:

```
setenv ipaddr 192.168.120.222;setenv serverip 192.168.120.99
```

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

5. Initialize SPI flash:

```
sf probe
```

6. Update SPL binary:

```
tftpboot 0xa0000000 ${serverip}:u-boot-spl.bin.normal.out
sf update 0xa0000000 0x0 $filesize
```

7. Update U-Boot binary:

```
tftpboot 0xa0000000 ${serverip}:visionfive2_fw_payload.img
sf update 0xa0000000 0x100000 $filesize
```

Through `flashcp` Command

To update SPL and U-Boot through the `flashcp` command, perform the following steps:

Note:
This method only supports versions equal to or later than `VF2_v2.5.0`.

1. Install the `mtd-utils` package by executing the following command:

```
apt install mtd-utils
```

2. Transfer the latest `u-boot-spl.bin.normal.out` and `visionfive2_fw_payload.img` files to Debian system through SCP.

3. Execute the following command to check the MTD partition:

```
cat /proc/mtd
```

Example Output:

You will see the partition information in the QSPI flash:

```
dev:    size   erasesize  name
mtd0: 00020000 00001000 "spl"
mtd1: 00300000 00001000 "uboot"
mtd2: 00100000 00001000 "data"
```

4. Update the SPL and U-Boot binaries according to different partitions:

   - Example command to update SPL:

     ```
     flashcp -v u-boot-spl.bin.normal.out /dev/mtd0
     ```

   - Example command to update U-Boot:

     ```
     flashcp -v visionfive2_fw_payload.img  /dev/mtd1
     ```

Example Command and Output:

```
# flashcp -v u-boot-spl.bin.normal.out /dev/mtd0
Erasing blocks: 32/32 (100%)
Writing data: 124k/124k (100%)
Verifying data: 124k/124k (100%)

# flashcp -v visionfive2_fw_payload.img  /dev/mtd1
Erasing blocks: 682/682 (100%)
Writing data: 2727k/2727k (100%)
Verifying data: 2727k/2727k (100%)
```

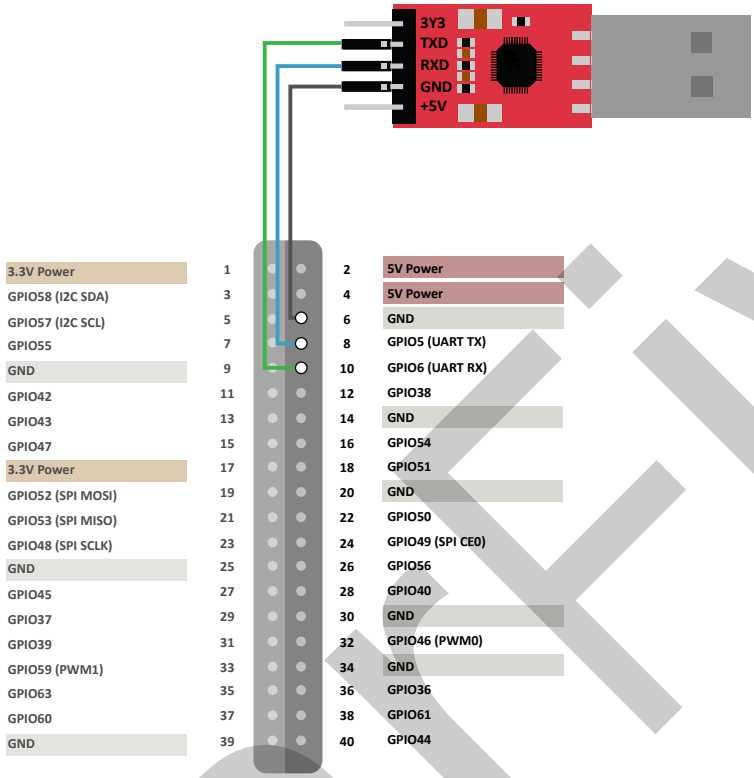
5. Restart the system to make the updates take effect.

7.4. Recovering the Bootloader

The SPL and U-Boot are stored inside the SPI flash of your board. There may be situations where you accidentally empty the flash or if the flash is damaged on your board. In these situations, it's better to recover the bootloader.

1. Connect the jumper wires between the USB-to-Serial converter and the Debug pins of VisionFive 2 40-pin GPIO header. The following figure is an example:

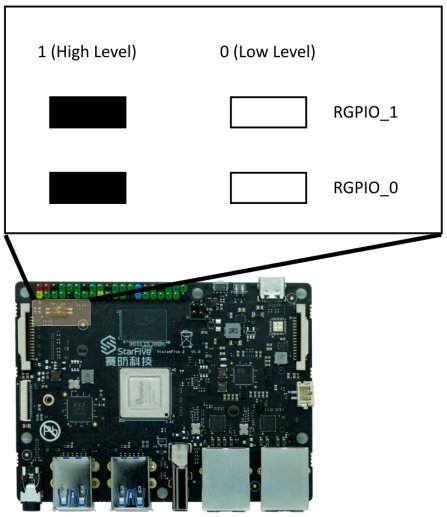
Figure 7-1 Connecting to the Debug Pins of VisionFive 2 40-pin GPIO Header



2. Before you recover the bootloader, double check the boot mode jumpers (Switch_2) on your board has already been switched to UART mode (RGPIO_1,RGPIO_0: 1,1).

Tip:
The following figure shows the boot mode settings. For more information, refer to Boot Mode Settings *(on page 23)*.

Figure 7-2 Boot Mode Setting (UART)

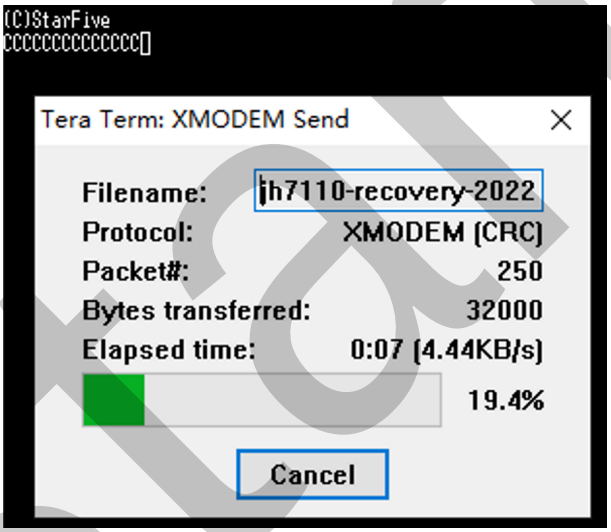


3. Configure the serial port baud rate settings to `115200` bps.
4. Power up, you will see an output like this:

```
CCCCCCCCCCCCCCCCCCCC
```

5. Transfer the recovery binary (`jh7110-recovery-20221205.bin`) by XMODEM. The recovery binary is located at: https://github.com/starfive-tech/Tools/tree/master/recovery.

Figure 7-3 Example Output

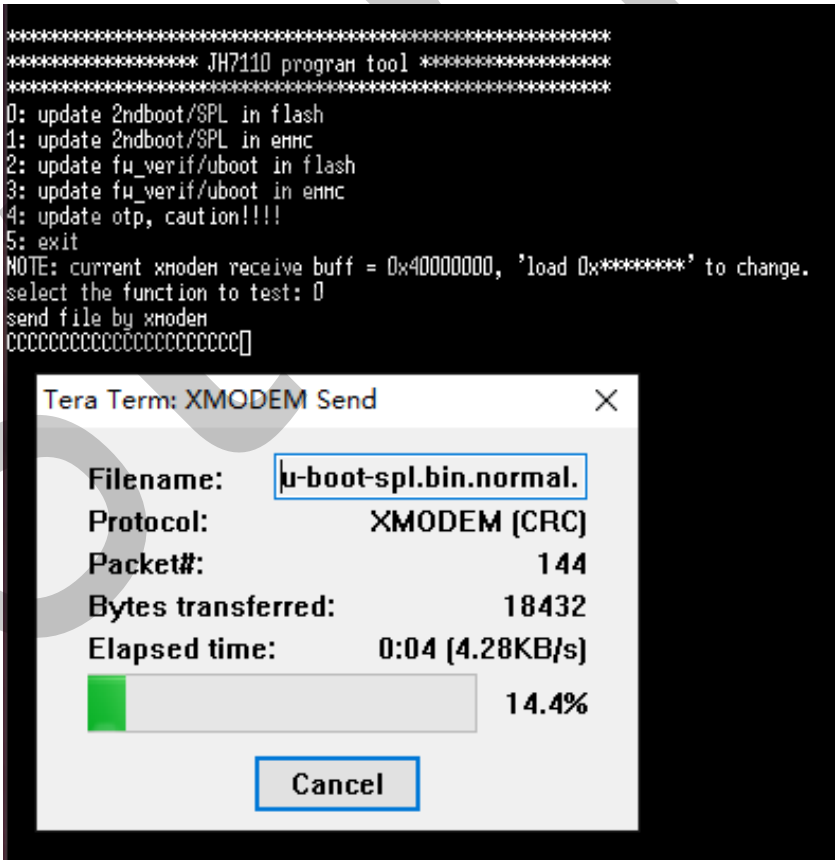


```
(C)StarFive
CCCCCCCCCCCC
JH7110 secondboot version: 221205-74596a9
CPU freq: 1250MHz
idcode: 0x1860C8
CSD:0xd00f0032 0x8f5903ff 0xffffffef 0x8a404023
mmc_send_ext_csd err 0
Device: EMMC
Manufacturer ID: 45
OEM: 100
Name: DG4D3
Tran Speed: 25000000
Rd Block Len: 512
MMC version 4.0
High Capacity: Yes
Capacity: 29.1 GiB
Bus Width: 8-bit
Erase Group Size: 0x80000
ddr 0x00000000, 4M test
ddr 0x00400000, 8M test
DDR clk 2133M, size 8GB

************************************************************
****************** JH7110 program tool ******************
************************************************************
0: update 2ndboot/SPL in flash
1: update 2ndboot/SPL in emmc
2: update fw_verif/uboot in flash
3: update fw_verif/uboot in emmc
4: update otp, caution!!!!
5: exit
NOTE: current xmodem receive buff = 0x40000000, 'load 0x********' to change.
select the function to test:
```

6. Type `0` and press **Enter** on your keyboard to update SPL binary <`u-boot-spl.bin.normal.out`>.

Figure 7-5 Example Output

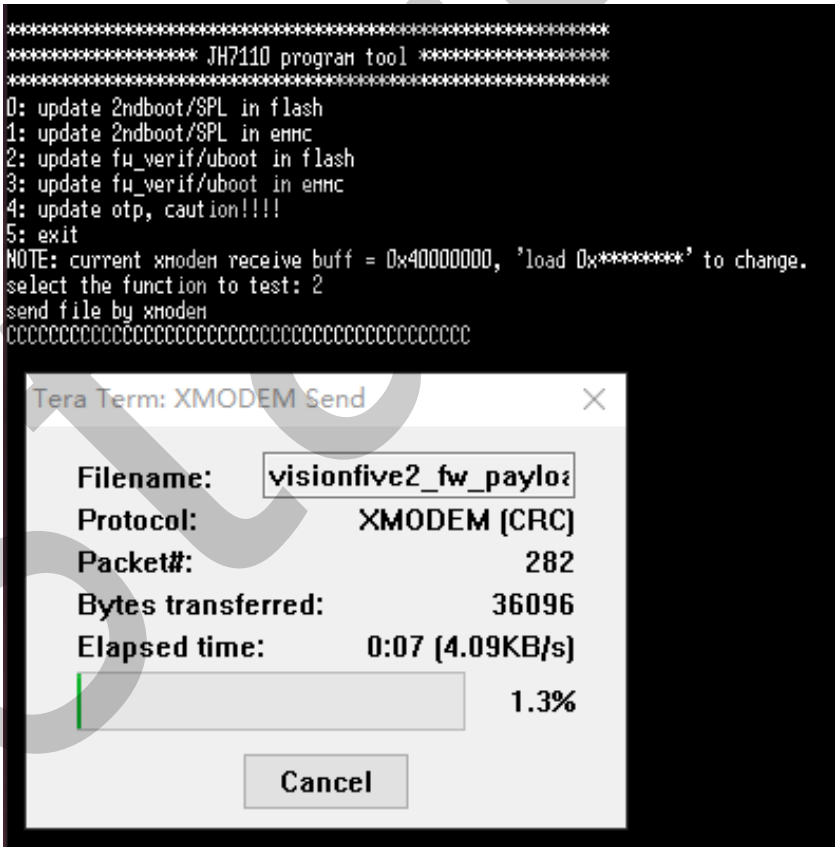


```
send file by xmodem
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCupdata first section
.
................................................................
................................................................
................................................................
................................................................
................................................................
................................................................
................................................................
.....................................................updata backup section
.
................................................................
................................................................
................................................................
................................................................
................................................................
................................................................
................................................................
.....................................................updata success

************************************************************
****************** JH7110 program tool ******************
************************************************************
0: update 2ndboot/SPL in flash
1: update 2ndboot/SPL in emmc
2: update fw_verif/uboot in flash
3: update fw_verif/uboot in emmc
4: update otp, caution!!!!
5: exit
NOTE: current xmodem receive buff = 0x40000000, 'load 0x********' to change.
select the function to test:
```

7. Type 2 and press **Enter** on your keyboard to update U-Boot binary <visionfive2_fw_payload.img>.

Figure 7-7 Example Output



8. Power off and switch jumpers back to Flash mode (RGPIO_1,RGPIO_0: 0,0).

7.5. Boot Mode Settings

VisionFive 2 provides pins to determine the boot mode before it is powered up. The following are the available boot modes and details.

Table 7-1 Boot Mode Settings

Boot Mode	RGPIO_0	RGPIO_1
1-bit QSPI Nor Flash	0	0
SDIO3.0	1	0
eMMC	0	1
UART	1	1

The following figure displays the location and the pin definitions of the boot mode settings.

Figure 7-9 Boot Mode Settings