JH7100

Boot User Guide

Version: V1

Date: 2021-09-30

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

e-Mail: sales@starfivetech.com (sales)

support@starfivetech.com (support)

About This Manual

Introduction

This document mainly describes the boot flow, the boot sources available for the JH7100 SoC and the Bare-metal boot examples. In order to run the examples presented in this guide, the following are required:

- Ubuntu 18.04
- VisionFive development board

Revision History

Version	Released	Revision
V1	2021-09-30	First release for VisionFive.

Tables of Content

1 Boot Sources

The GPIO is used to select the boot vector and BootLoader source and offer multiple methods to obtain the BootLoader image.

The JH7100 SoC can boot from one of the sources listed in the following table, as selected by the PAD_GPIO `[62:60]` values.

Table 1-1 PAD_GPIO Values for Boot Source Selection

<table>
<tr><th>Processor</th><th>SCFG_boot_mode</th><th>PAD_GPIO [63]</th><th>Boot Vector</th><th>PAD_GPIO [62:60]</th></tr>
<tr><td rowspan="3">U74</td><td>0x1</td><td>-</td><td>SCFG_u74_re-set_vector</td><td>-</td></tr>
<tr><td rowspan="2">0x0 (default)</td><td>0x0</td><td>0x00_2000_0000, XIP Flash</td><td rowspan="2">0x0: 1-bit quad SPI NOR flash memory
0x1: 4-bit quad SPI NOR flash memory
0x2: SDIO
(Reserved)
0x3: eMMC
(Reserved)
0x4: UART
0x5: USB
(Reserved)
0x6: chiplink
(Reserved)
0x7: SPI2AHB
(Reserved)</td></tr>
<tr><td>0x1</td><td>0x00_1840_0000, on-chip BootROM (32KB)</td></tr>
</table>

Notes:

- The boot mode and boot source selection (PAD_GPIO [63]) can be read through syscon status registers.
- Use the GPIO pad to select the vector and loader source by default.
- PAD_GPIO [63] and PAD_GPIO [62:60] can be configured to 1 or 0 via pull-up/pull-down resister, button or jumper according to board hardware design.

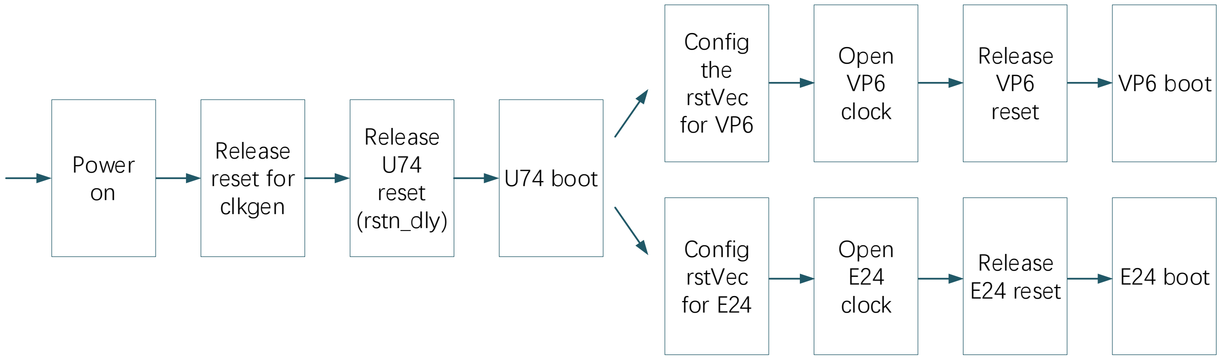


Figure 1-2 Hardware Boot Sequence

2 Boot Flow

The boot process starts when the processor is released from reset, and jumps to the reset vector address (0x1840,0000 by default), located in the BootROM address space.

The boot flow is a multi-stage process. Each stage is responsible for loading the next stage. The typical boot flow is illustrated in the following figure:

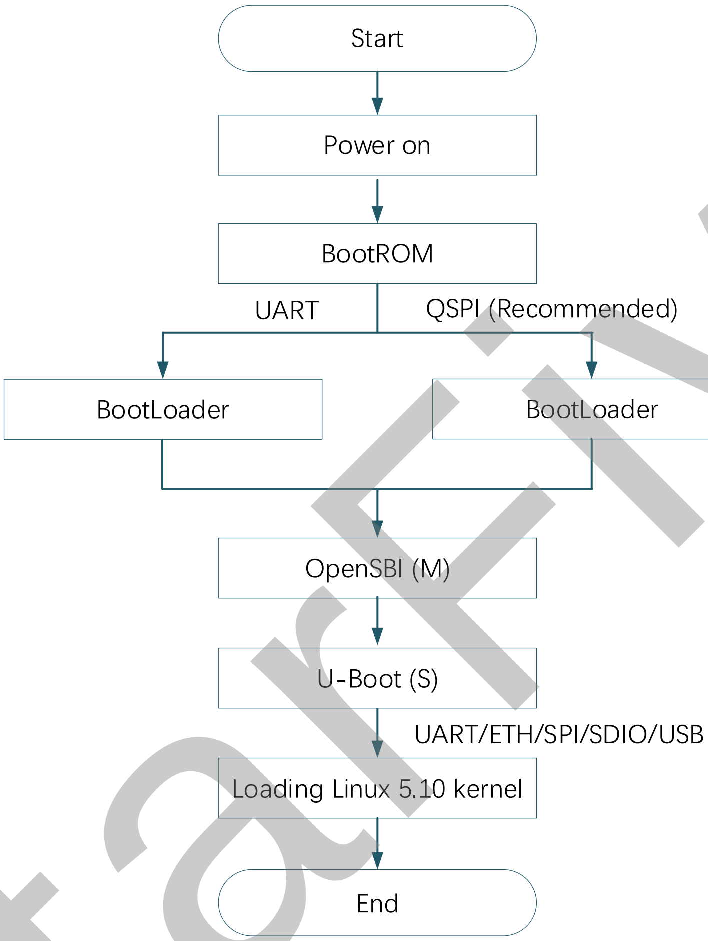


Figure 2-1 Typical Boot Flow

2.2 BootROM

The BootROM is located in on-chip ROM, and the storage address is 0x1840,0000, which cannot be dynamically updated. After power-on, each HART jumps to 0x1800,0000 (located in RAM) by default and starts to execute BootROM.

The main function of the BootROM is to select the boot source and execute it. According to different hardware jumpers on the chip, only UART and QSPI sources are supported currently.

Table 2-1 Boot Source Description

Source	Description
UART	Enter a simple command line. Load a limited size binary into the on-chip RAM and execute it. This mode is mainly used for firmware update.
QSPI	Automatically load the 32K Bootloader to 0x1800,0000 (located in RAM) from address 0 of NOR Flash and jump to it.

Limitation:

The file loaded from NOR Flash cannot exceed 32KB.

2.3 BootLoader

The BootROM limits the size of data read from NOR Flash. The BootLoader reads DDRInit from 0x10000 in NOR Flash to 0x1808,0000 (located in RAM), and then jump to it for execution.

The DDRInit will initialize the DDR, then read `fw_payload.bin` (OpenSBI+Uboot, the file header contains file size information) from 0x40000 in NOR Flash to 0x8000,0000 (located in DDR), and then jump to it to execute the OpenSBI.

The normal output information is illustrated in the following figure.

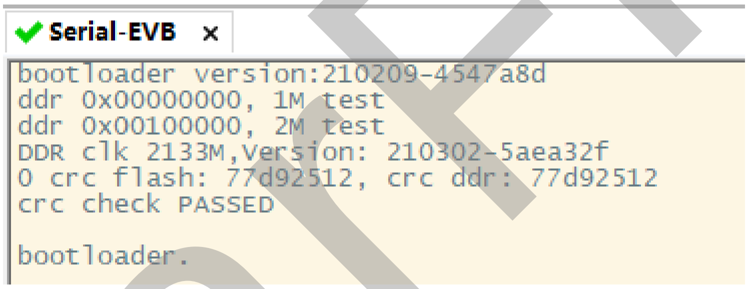


Figure 2-2 BootLoader Output Example

2.4 OpenSBI

The binary of OpenSBI is packaged with the binary compiled by U-Boot in the way of payload to generate the final `fw_payload.bin`. The main functions of OpenSBI are:

- Provide basic system calls for Linux
- Switch the mode from M mode to S mode
- Jump to 0x8002,0000 (located in DDR) to execute U-Boot.

The normal output information is illustrated in the following figure.

```
OpenSBI v0.9
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name             : StarFive VIC7100
Platform Features         : timer,mfdeleg
Platform HART Count       : 2
Firmware Base             : 0x80000000
Firmware Size             : 92 KB
Runtime SBI Version       : 0.2

Domain0 Name              : root
Domain0 Boot HART         : 1
Domain0 HARTs             : 0*,1*
Domain0 Region00          : 0x0000000080000000-0x000000008001ffff ()
Domain0 Region01          : 0x0000000000000000-0xffffffffffffffff (R,W,X)
Domain0 Next Address      : 0x0000000080020000
Domain0 Next Arg1         : 0x0000000088000000
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes

Boot HART ID              : 1
Boot HART Domain          : root
Boot HART ISA             : rv64imafdcsux
Boot HART Features        : scounteren,mcounteren
Boot HART PMP Count       : 16
Boot HART PMP Granularity : 4096
Boot HART PMP Address Bits: 36
Boot HART MHPM Count      : 0
Boot HART MHPM Count      : 0
Boot HART MIDELEG         : 0x0000000000000222
Boot HART MEDELEG         : 0x000000000000b109


U-Boot 2021.01-gcdbfbf0c-dirty (Apr 16 2021 - 06:58:12 +0000)

CPU:   rv64imafdc
Model: sifive,freedom-u74-arty
DRAM:  8 GiB
MMC:   VIC DWMMC0: 0
In:    serial
Out:   serial
Err:   serial
Model: sifive,freedom-u74-arty
Net:   Could not get PHY for dwmac.10020000: addr 0
Board Net Initialization Failed
dwmac.10020000
StarFive #
StarFive #
StarFive #
StarFive #
```

Figure 2-3 OpenSBI Output Example

2.5 U-Boot

U-Boot runs at 0x8002,0000 and works in S mode. It contains basic file system and commonly used peripheral drivers (such as GMAC, UART, QSPI, USB, SDIO etc.). U-Boot can load the kernel image through ETH, UART, QSPI, SDIO or USB.

The following example describes how to load Linux 5.10 kernel image from SDIO.

Notes:

- The example assumes the installation of Ubuntu 18.04.
- Press Enter to confirm the operation or for the next command.

Step 1 SD card partition.

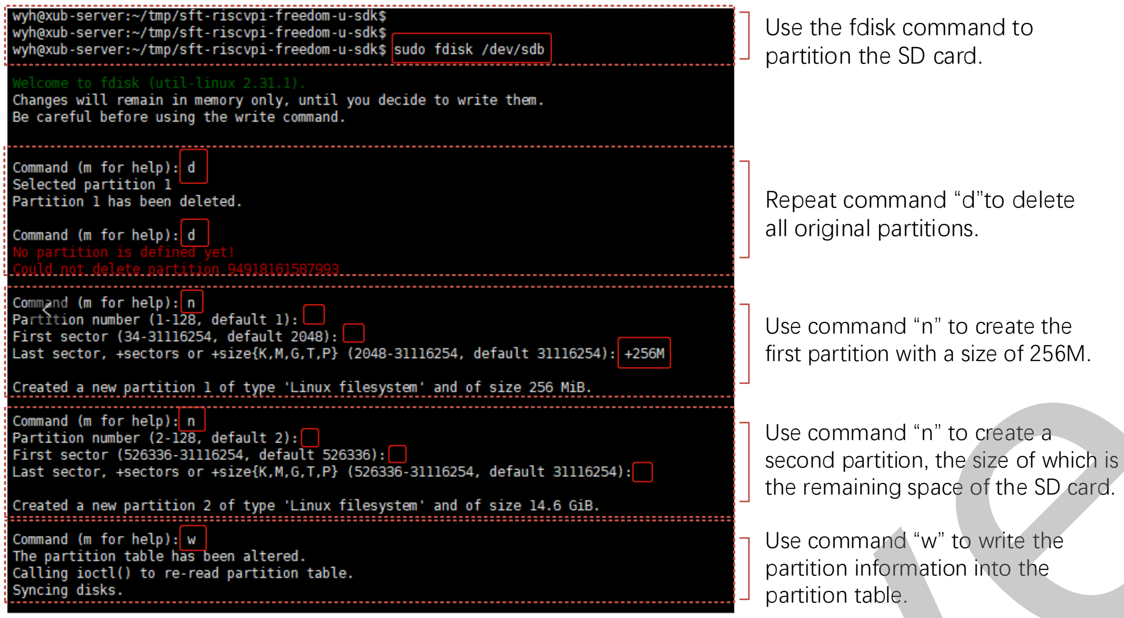


Figure 2-4 SD Card Partition Command Explanation

Step 2 Format the partition.

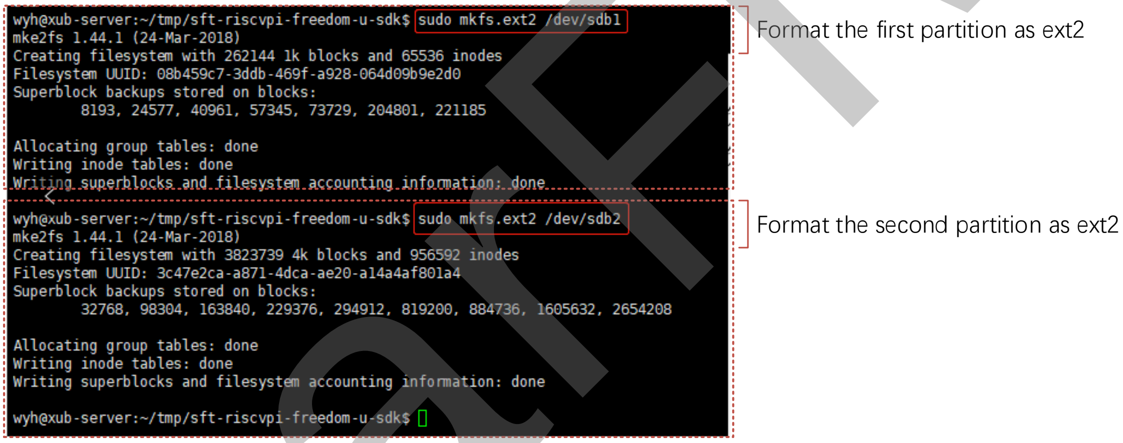


Figure 2-5 Format the Partition

Step 3 Generate image.fit from Freelight U SDK, please refer to the detailed guidelines in the link https://github.com/starfive-tech/freelight-u-sdk.

Step 4 Copy the boot file.

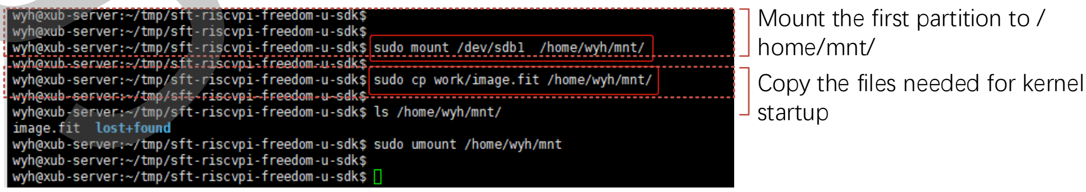
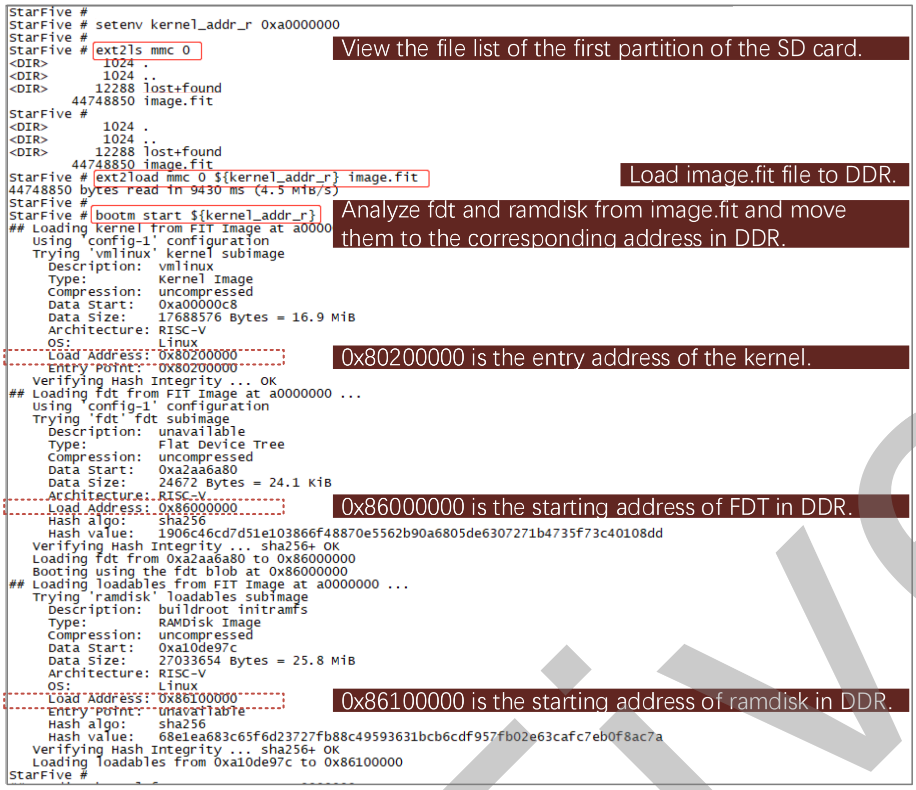


Figure 2-6 Copying the Boot File

Step 5 Load the kernel (Linux 5.10 as an example).

> **Notes:**
>
> The addresses 0x80200000, 0x86100000 and 0x86000000 have been specified when compiling and generating `image.fit` and cannot be modified.



```
StarFive #
StarFive # setenv kernel_addr_r 0xa0000000
StarFive #
StarFive # ext2ls mmc 0
<DIR>          1024 .
<DIR>          1024 ..
<DIR>         12288 lost+found
           44748850 image.fit
StarFive #
<DIR>          1024 .
<DIR>          1024 ..
<DIR>         12288 lost+found
           44748850 image.fit
StarFive # ext2load mmc 0 ${kernel_addr_r} image.fit
44748850 bytes read in 9430 ms (4.5 MiB/s)
StarFive #
StarFive # bootm start ${kernel_addr_r}
## Loading kernel from FIT Image at a0000
   Using 'config-1' configuration
   Trying 'vmlinux' kernel subimage
     Description:  vmlinux
     Type:         Kernel Image
     Compression:  uncompressed
     Data Start:   0xa00000c8
     Data Size:    17688576 Bytes = 16.9 MiB
     Architecture: RISC-V
     OS:           Linux
     Load Address: 0x80200000
     Entry Point:  0x80200000
   Verifying Hash Integrity ... OK
## Loading fdt from FIT Image at a0000000 ...
   Using 'config-1' configuration
   Trying 'fdt' fdt subimage
     Description:  unavailable
     Type:         Flat Device Tree
     Compression:  uncompressed
     Data Start:   0xa2aa6a80
     Data Size:    24672 Bytes = 24.1 KiB
     Architecture: RISC-V
     Load Address: 0x86000000
     Hash algo:    sha256
     Hash value:   1906c46cd7d51e103866f48870e5562b90a6805de6307271b4735f73c40108dd
   Verifying Hash Integrity ... sha256+ OK
   Loading fdt from 0xa2aa6a80 to 0x86000000
   Booting using the fdt blob at 0x86000000
## Loading loadables from FIT Image at a0000000 ...
   Trying 'ramdisk' loadables subimage
     Description:  buildroot initramfs
     Type:         RAMDisk Image
     Compression:  uncompressed
     Data Start:   0xa10de97c
     Data Size:    27033654 Bytes = 25.8 MiB
     Architecture: RISC-V
     OS:           Linux
     Load Address: 0x86100000
     Entry Point:  unavailable
     Hash algo:    sha256
     Hash value:   68e1ea683c65f6d23727fb88c49593631bcb6cdf957fb02e63cafc7eb0f8ac7a
   Verifying Hash Integrity ... sha256+ OK
   Loading loadables from 0xa10de97c to 0x86100000
StarFive #
```


```
StarFive # bootm loados ${kernel_addr_r}
   Loading Kernel Image
StarFive #
Trying to execute a command out of order
bootm - boot application image from memory

Usage:
bootm [addr [arg ...]]
    - boot application image stored in memory
        passing arguments 'arg ...'; when booting a Linux kernel,
        'arg' can be the address of an initrd image
        When booting a Linux kernel which requires a flat device-tree
        a third argument is required which is the address of the
        device-tree blob. To boot that kernel without an initrd image,
        use a '-' for the second argument. If you do not pass a third
        a bd_info struct will be passed instead

For the new multi component uImage format (FIT) addresses
        must be extended to include component or configuration unit name:
        addr:<subimg_uname> - direct component image specification
        addr#<conf_uname>   - configuration specification
        Use iminfo command to get the list of existing component
        images and configurations.

Sub-commands to do part of the bootm sequence.  The sub-commands must be
issued in the order below (it's ok to not issue all sub-commands):
        start [addr [arg ...]]
        loados  - load OS image
        ramdisk - relocate initrd, set env initrd_start/initrd_end
        fdt     - relocate flat device tree
        cmdline - OS specific command line processing/setup
        bdt     - OS specific bd_info processing
        prep    - OS specific prep before relocation or go
        go      - start OS
StarFive # booti 0x80200000 0x86100000:${filesize} 0x86000000
## Flattened Device Tree blob at 86000000
   Booting using the fdt blob at 0x86000000
   Using Device Tree in place at 0000000086000000, end 000000008600905f

Starting kernel ...

[    0.000000] Linux version 5.10.6-ga4e958526177-dirty (clivia@ubuntu) (riscv64-buildroot-linux-gnu-gcc.br_real (
 3 03:38:06 PDT 2021
[    0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[    0.000000] efi: UEFI not found.
[    0.000000] Initial ramdisk at: 0x(____ptrval____) (44752896 bytes)
[    0.000000] Reserved memory: created CMA memory pool at 0x00000000a0000000, size 640 MiB
[    0.000000] OF: reserved mem: initialized node linux,cma, compatible id shared-dma-pool
[    0.000000] Reserved memory: created DMA memory pool at 0x00000000f9000000, size 16 MiB
[    0.000000] OF: reserved mem: initialized node framebuffer@f9000000, compatible id shared-dma-pool
[    0.000000] Reserved memory: created DMA memory pool at 0x00000000fb000000, size 32 MiB
[    0.000000] OF: reserved mem: initialized node framebuffer@fb000000, compatible id shared-dma-pool
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x00000000ffffffff]
[    0.000000]   Normal   [mem 0x0000000100000000-0x000000027fffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x00000000f8ffffff]
[    0.000000]   node   0: [mem 0x00000000fa000000-0x00000000faffffff]
[    0.000000]   node   0: [mem 0x00000000fd000000-0x000000027fffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x000000027fffffff]
[    0.000000] On node 0 totalpages: 2084352
[    0.000000]   DMA32 zone: 7161 pages used for memmap
[    0.000000]   DMA32 zone: 0 pages reserved
[    0.000000]   DMA32 zone: 511488 pages, LIFO batch:63
[    0.000000]   Normal zone: 21504 pages used for memmap
[    0.000000]   Normal zone: 1572864 pages, LIFO batch:63
[    0.000000] software IO TLB: mapped [mem 0x00000000cc000000-0x00000000d0000000] (64MB)
[    0.000000] SBI specification v0.2 detected
[illegible]
```


Figure 2-7 Loading the Kernel