

JH7100 SoC Boot User Guide

01 (2021-6-7)



Shanghai StarFive Technology Co., Ltd.

All Rights Reserved

PROPRIETARY NOTICE

Copyright © Shanghai StarFive Technology Co., Ltd., 2018-2025. All rights reserved.

Information in this document is provided "as is," with all faults.

Shanghai StarFive Technology Co., Ltd., 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.

Shanghai StarFive Technology Co., Ltd., 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., reserves the right to make changes without further notice to any products herein.

All material appearing in this document is protected by copyright and is the property of Shanghai StarFive Technology Co., Ltd. You may not copy, reproduce, distribute, publish, display, perform, modify, create derivative works, transmit, or in any way exploit any such content, nor may you distribute any part of this content over any network, including a local area network, sell or offer it for sale, or use such content to construct any kind of database. Copying or storing any content except as provided above is expressly prohibited without prior written permission of the Shanghai StarFive Technology Co, Ltd (hereinafter "StarFive").

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.

Prerequisite

In order to run the examples presented in this guide, the following are required:

- Ubuntu 18.04
- BeagleV™ - StarLight development board

Revision History

Version	Released	Change Description
01	2021-6-7	The first official release.

Tables of Content

About This Manual	ii
1 Boot Sources	1
2 Boot Flow	2
2.1 BootROM.....	2
2.2 BootLoader	3
2.3 OpenSBI.....	3
2.4 U-Boot	4

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

Processor	SCFG_boot_mode	PAD_GPIO [63]	Boot Vector	PAD_GPIO [62:60]
U74	0x1	-	SCFG_u74_reset_vector	-
	0x0 (default)	0x0	0x00_2000_0000, XIP Flash	<ul style="list-style-type: none"> • 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)
		0x1	0x00_1840_0000, on-chip BootROM (32KB)	

Note:

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

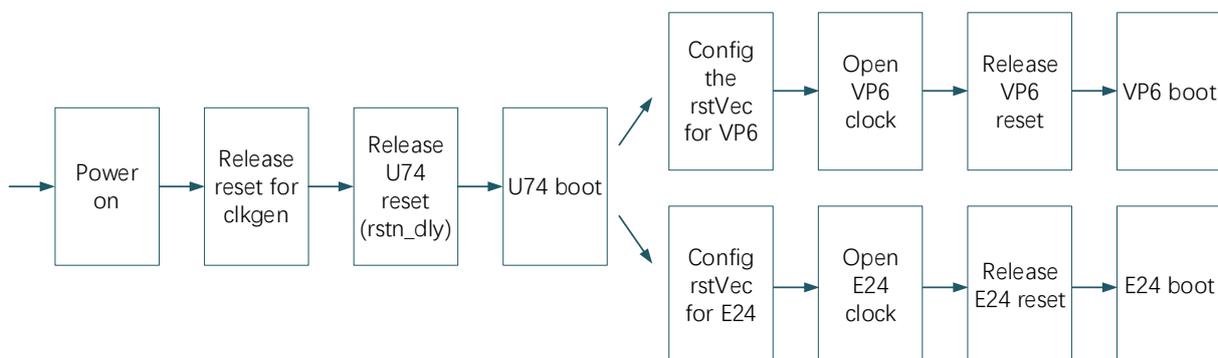


Figure 1-2 Hardware Boot Sequence

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.

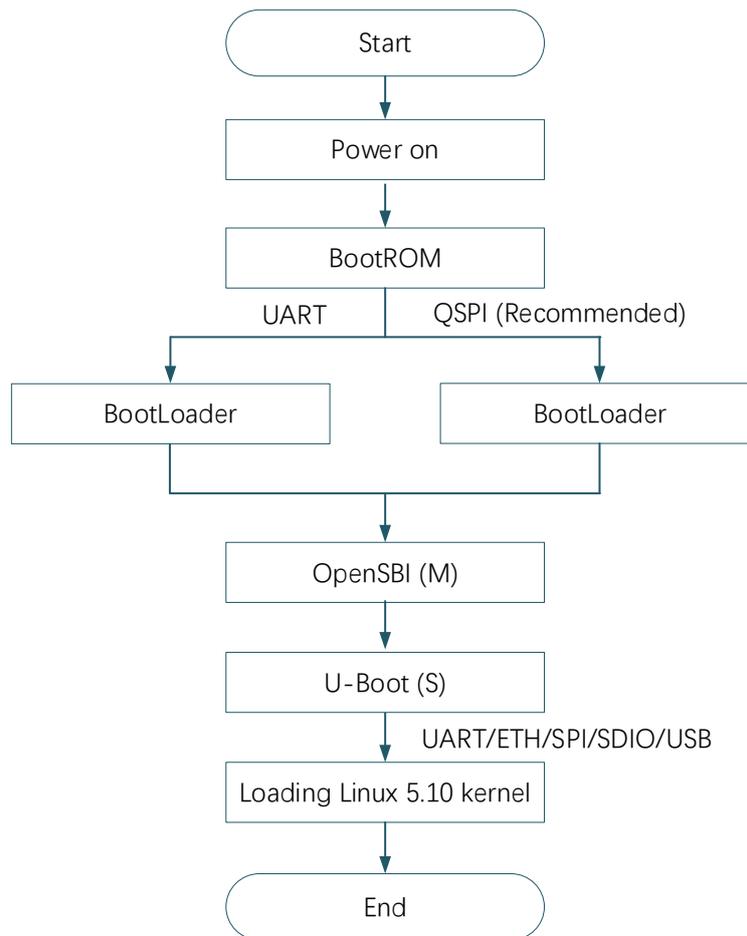


Figure 2-1 Typical Boot Flow

2.1 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

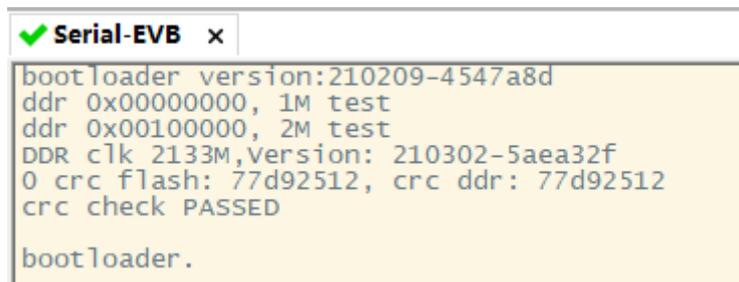
The file loaded from NOR Flash cannot exceed 32KB.

2.2 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.



```

Serial-EVB x
bootloader version:210209-4547a8d
ddr 0x00000000, 1M test
ddr 0x00100000, 2M test
DDR clk 2133M,Version: 210302-5aea32f
0 crc flash: 77d92512, crc ddr: 77d92512
crc check PASSED
bootloader.

```

Figure 2-2 BootLoader Output Example

2.3 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

  O p e n S B I

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   : 0x0000000008000000-0x000000008001ffff (C)
Domain0 Region01   : 0x0000000000000000-0xffffffffffffffff (R,w,x)
Domain0 Next Address : 0x00000000080020000
Domain0 Next Arg1   : 0x00000000088000000
Domain0 Next Mode   : S-mode
Domain0 SysReset    : yes

Boot HART ID       : 1
Boot HART Domain   : root
Boot HART ISA      : rv64imafdcsub
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  : 0x00000000000000222
Boot HART MEDELEG  : 0x00000000000000b109

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

2.4 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.



The example assumes the installation of Ubuntu 18.04.

Press Enter to confirm the operation or for the next command.

1. SD card partition.

```
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$ sudo fdisk /dev/sdb
Welcome to fdisk (util-linux 2.31.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

Command (m for help): d
No partition is defined yet!
Could not delete partition 24978161587903

Command (m for help): n
Partition number (1-128, default 1):
First sector (34-31116254, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-31116254, default 31116254): +256M

Created a new partition 1 of type 'Linux filesystem' and of size 256 MiB.

Command (m for help): n
Partition number (2-128, default 2):
First sector (526336-31116254, default 526336):
Last sector, +sectors or +size{K,M,G,T,P} (526336-31116254, default 31116254):

Created a new partition 2 of type 'Linux filesystem' and of size 14.6 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Use the fdisk command to partition the SD card.

Repeat command "d" to delete all original partitions.

Use command "n" to create the first partition with a size of 256M.

Use command "n" to create a second partition, the size of which is the remaining space of the SD card.

Use command "w" to write the partition information into the partition table.

Figure 2-4 SD Card Partition Command Explanation

2. Format the partition.

```
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$ sudo mkfs.ext2 /dev/sdb1
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 262144 1k blocks and 65536 inodes
Filesystem UUID: 08b459c7-3ddb-469f-a928-064d09b9e2d0
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729, 204801, 221185

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$ sudo mkfs.ext2 /dev/sdb2
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 3823739 4k blocks and 956592 inodes
Filesystem UUID: 3c47e2ca-a871-4dca-ae20-a14a4af801a4
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$
```

Format the first partition as ext2

Format the second partition as ext2

Figure 2-5 Format the Partition

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>.

4. Copy the boot file.

```
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$ sudo mount /dev/sdb1 /home/wyh/mnt/
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$ sudo cp work/image.fit /home/wyh/mnt/
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$ ls /home/wyh/mnt/
image.fit  lost+found
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$ sudo umount /home/wyh/mnt
wyh@xub-server:~/tmp/sft-riscvpi-freedom-u-sdk$
```

Mount the first partition to /home/mnt/

Copy the files needed for kernel startup

Figure 2-6 Copying the Boot File

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

i 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> 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 a0000000
Using 'config-1' configuration
Trying 'vmlinux' kernel subimage
Description: vmlinux
Type: Kernel Image
Compression: uncompressed
Data start: 0xa0000c8
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' 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: 68e1ea683c65f6d23727fb88c49593631bcb6cdf957fb02e63caf7eb0f8ac7a
Verifying Hash Integrity ... sha256+ OK
Loading loadables from 0xa10de97c to 0x86100000
StarFive #
    
```

View the file list of the first partition of the SD card.

Load image.fit file to DDR.

Analyze fdt and ramdisk from image.fit and move them to the corresponding address in DDR.

0x80200000 is the entry address of the kernel.

0x86000000 is the starting address of FDT in DDR.

0x86100000 is the starting address of ramdisk in DDR.

```

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_name> - direct component image specification
addr#<conf_name> - 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-g4e958526177-dirty (clivia@ubuntu) (riscv64-buildroot-linux-gnu-gcc.br_real (
3 09:38:06 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 0x00000000fd000000-0x00000000fdffffff]
0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x000000027fffffff]
0.000000] on node 0 total pages: 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
0.000000] set_timer_list: no timer found
    
```

Load the kernel to the corresponding address in the DDR.

Boot the kernel.

Figure 2-7 Loading the Kernel