



StarFive
赛昉科技

VisionFive 2 40-Pin GPIO Header User Guide

Version: 1.0

Date: 2022/12/21

Doc ID: VisionFive2-UGEN-001

Legal Statements

Important legal notice before reading our documentation.

PROPRIETARY NOTICE

Copyright © Shanghai StarFive Technology Co., Ltd., 2018-2022. All rights reserved.

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.

All material appearing in this document is protected by copyright and is the property of StarFive. You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. StarFive authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services.

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

Preface

About this guide and technical support information.

About this document

This document is intended to:

- introduce the 40-pin header.
- provide instructions to configure and debug GPIO, I2C, SPI, PWM, and UART.
- provide peripheral examples to use the 40-pin GPIO header.






Revision History

Table 0-1 Revision History

Version	Released	Revision
1.0	2022/12/21	The first official release.

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.....	6
List of Figures.....	7
Legal Statements.....	ii
Preface.....	iii
1. Overview.....	8
1.1. 40-Pin Header Definition.....	8
2. GPIO Pinout	9
3. Preparation.....	10
3.1. Preparing Hardware.....	10
3.2. Preparing Software.....	11
3.2.1. GitHub Repository.....	11
3.2.2. Flashing Debian OS to a Micro-SD Card.....	11
3.2.3. Generating DTB.....	11
3.2.4. Replacing DTB.....	12
4. GPIO Operations.....	15
4.1. Configuring GPIO.....	15
5. I2C Operations.....	16
5.1. Configuring I2C GPIO.....	16
5.1.1. Hardware Setup.....	16
5.1.2. Configuring dts File.....	17
5.2. Debugging I2C GPIO.....	17
6. SPI Operations.....	20
6.1. Configuring SPI GPIO.....	20
6.1.1. Modify Pins.....	20
6.2. Debugging SPI GPIO.....	20
6.2.1. Loopback Test.....	21
6.2.2. Testing SPI with ADXL345 Module.....	22
7. PWM Operations.....	25
7.1. Configuring PWM GPIO.....	25
7.1.1. Modify Pin.....	25
7.1.2. PWM and Pin Name Mapping.....	25
7.2. Debugging PWM GPIO.....	25
8. UART Operations.....	27
8.1. Configuring UART GPIO.....	27
8.1.1. Modifying dts.....	27
8.2. Debugging UART GPIO.....	28
8.2.1. Hardware Setup.....	28
8.2.2. Debugging UART Send and Receive Functions.....	29
9. Peripheral Examples.....	33
9.1. Sense Hat (B) Example.....	33
9.1.1. Hardware Setup.....	33
9.1.2. Running Example with Sense Hat (B).....	34
9.2. 2.4inch LCD Module Example.....	35

9.2.1. Hardware Setup.....	35
9.2.2. Executing Example.....	37

StarFive

List of Tables

Table 0-1 Revision History.....	iii
Table 2-1 GPIO Assignments.....	9
Table 3-1 Hardware Preparation.....	10
Table 3-2 GitHub Repository Addresses.....	11
Table 5-1 Connect Sense Hat (B) to the 40-Pin Header.....	16
Table 7-1 PWM and Pin Name Mapping.....	25
Table 8-1 UART and DEV Mapping.....	28
Table 9-1 Connect Sense Hat (B) to the 40-Pin Header.....	33
Table 9-2 Connect 2.4inch LCD with 40-pin Header.....	35

List of Figures

Figure 1-1 40-Pin Definition.....	8
Figure 3-1 Identified Directories.....	12
Figure 5-1 Connect Sense Hat (B) to the 40-Pin GPIO Header.....	16
Figure 5-2 Example File Content.....	17
Figure 5-3 Example Output.....	17
Figure 5-4 Example Output.....	18
Figure 5-5 Example Output.....	18
Figure 5-6 Example Output.....	19
Figure 6-1 Modify Pins.....	20
Figure 6-2 Connect Pin 19 with 21.....	21
Figure 6-3 Example Output.....	21
Figure 6-4 Example Output.....	22
Figure 6-5 Connect ADXL345 Module to the Header.....	23
Figure 6-6 Example Output.....	24
Figure 7-1 Example File Content.....	25
Figure 8-1 Example Configuration.....	27
Figure 8-2 Example Configuration.....	27
Figure 8-3 Example Configuration.....	28
Figure 8-4 Connect the Converter to the Header.....	29
Figure 8-5 Example Configuration.....	29
Figure 8-7 Example Output.....	30
Figure 8-8 Example Configuration.....	30
Figure 8-10 Example Command and Output.....	31
Figure 8-11 Example Output.....	31
Figure 8-12 Test UART Send.....	32
Figure 8-13 Test UART Receive.....	32
Figure 9-1 Connect Sense Hat (B) to the 40-Pin Header.....	33
Figure 9-2 Connect Sense Hat (B) to the 40-Pin Header.....	34
Figure 9-3 Connect 2.4inch LCD with 40-Pin Header.....	36
Figure 9-5 Example Output.....	37

1. Overview

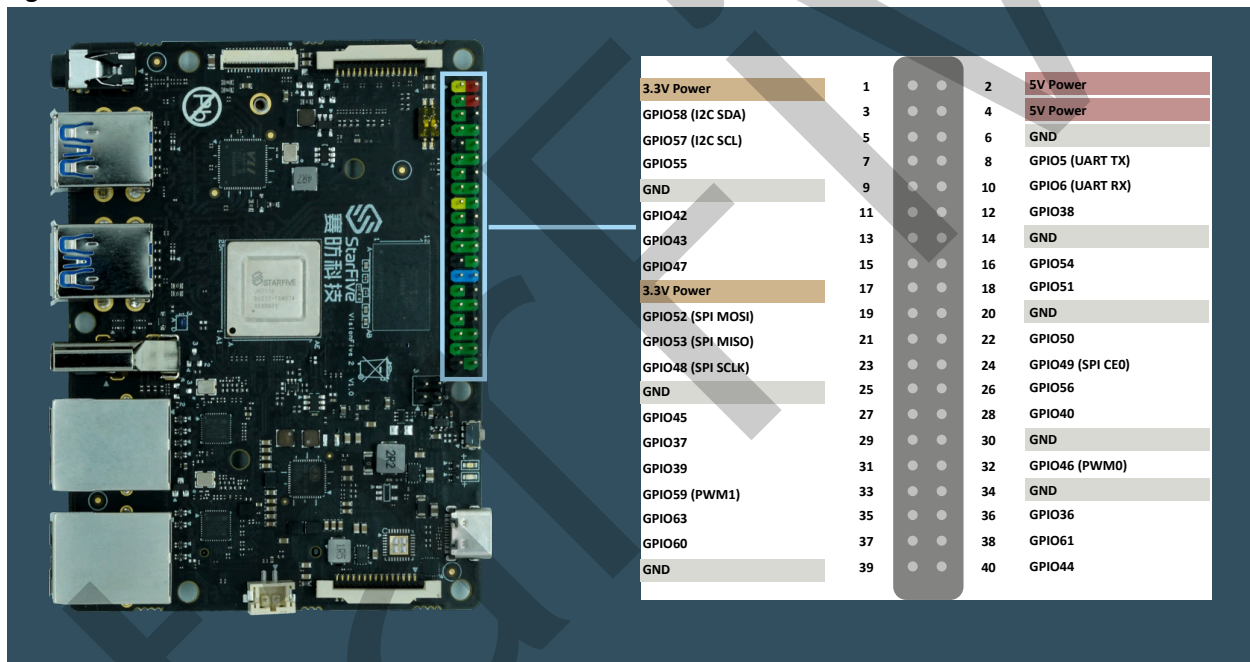
The 40-pin header allows VisionFive 2 single board computers to interface with a variety of external components, which enables developers to create their projects. This document is intended to:

- introduce the 40-pin header as described in this chapter.
- provide instructions to configure and debug GPIO, I2C, SPI, PWM, and UART, as described in [GPIO Operations \(on page 15\)](#), [I2C Operations \(on page 16\)](#), [SPI Operations \(on page 20\)](#), [PWM Operations \(on page 25\)](#), and [UART Operations \(on page 27\)](#) chapters.
- provide peripheral examples to use the 40-pin header, as described in the [Peripheral Examples \(on page 33\)](#) chapter.

1.1. 40-Pin Header Definition

The following figure shows the location of the 40-pin header on VisionFive 2.

Figure 1-1 40-Pin Definition



2. GPIO Pinout

The following table describes the GPIO pinout, the map, and the explanation of what each pin can do.

Table 2-1 GPIO Assignments

Sys	dtb	GPIO Num	Pin Name	Pin Num	Pin Num	Pin Name	GPIO Num	dtb	Sys
		N/A	+3.3V	1	2	+5V	N/A		
i2c-0	i2c0	58	GPIO58 (I2C SDA)	3	4	+5V	N/A		
i2c-0	i2c0	57	GPIO57 (I2C SCL)	5	6	GND	N/A		
55		55	GPIO55	7		GPIO5 (UART TX)	5	uart0	ttyS0
		N/A	GND	9	10	GPIO6 (UART RX)	6	uart0	ttyS0
42		42	GPIO42	11	12	GPIO38	38		38
43		43	GPIO43	13	14	GND	N/A		
47		47	GPIO47	15	16	GPIO54	54		54
		N/A	+3.3V	17	18	GPIO51	51		51
spidev1.0	spi0	52	GPIO52 (SPI MOSI)	19	20	GND	N/A		
spidev1.0	spi0	53	GPIO53 (SPI MISO)	21	22	GPIO50	50		50
spidev1.0	spi0	48	GPIO48 (SPI SCLK)	23	24	GPIO49 (SPI CE0)	49	spi0	spidev1.0
		N/A	GND	25	26	GPIO56	56		56
45		45	GPIO45	27	28	GPIO40	40		40
37		37	GPIO37	29	30	GND	N/A		
39		39	GPIO39	31	32	GPIO46 (PWM0)	49		pwm0
pwm1		59	GPIO59 (PWM1)	33	34	GND	N/A		
63		63	GPIO63	35	36	GPIO36	36		36
60		60	GPIO60	37	38	GPIO61	61		61
		N/A	GND	39	40	GPIO44	44		44



Note:

- The *dtb* column shows the name of the node in the DTBI file (`jh7110-visionfive-v2.dtsi`). You can find the associated node by simply searching the name.
- The *Sys* column shows the pin number used when exporting the GPIO pin under the `/sys/class/gpio`.

3. Preparation

Before configuring and debugging the GPIOs, you need to prepare the follows:

3.1. Preparing Hardware

The following table describes hardware items to be prepared if you want to configure, debug, and test this 40-pin header by following this guide:

Table 3-1 Hardware Preparation

Type	M/O*	Item	Notes
General	M	VisionFive 2 single board computer	-
General	M	<ul style="list-style-type: none">• 32 GB (or more) micro-SD card• micro-SD card reader• Computer (Windows/macOS/Linux)• USB to serial converter (3.3 V I/O)• Ethernet cable• Power adapter (5 V / 3 A)• USB Type-C Cable	These items are used for flashing Debian OS into a micro-SD card.
GPIO	O	An oscilloscope	The oscilloscope is used to verify the GPIO voltage.
I2C	O	<ul style="list-style-type: none">• Sense Hat (B)• Dupont Line	-
SPI	O	<ul style="list-style-type: none">• ADXL345 Module• Dupont Line	-
PWM	O	An oscilloscope	The oscilloscope is used to measure the corresponding pin and check the PWM period and duty cycle.
SPI LCD	O	<ul style="list-style-type: none">• 2.4inch LCD Module• Dupont Line	-
UART	O	<ul style="list-style-type: none">• GNSS HAT• Dupont Line	This is a GNSS HAT based on MAX-7Q, which supports positioning systems including GPS, GLONASS, QZSS, and SBAS. It features accurate and fast positioning with minor drifting, low power consumption, outstanding ability for anti-spoofing and anti-jamming, and so on. For detailed specifications, refer to MAX-7Q GNSS HAT .

3.2. Preparing Software

Before configuring the 40-pin header, the Debian OS needs to be flashed into the Micro-SD card, and the DTB files need to be compiled and replaced. The following procedures are provided:

3.2.1. GitHub Repository

The following table describes the GitHub Repository addresses:



Note:

Make sure you have switched to the corresponding branch.

Table 3-2 GitHub Repository Addresses

Type	Repository	Branch
Linux	Linux	JH7110_VisionFive2_devel
dts File under Linux Repo	<ul style="list-style-type: none"> • jh7110-common.dtsi • jh7110.dtsi 	-
Uboot	Uboot	JH7110_VisionFive2_devel
OpenSBI	OpenSBI	master
Debian	Debian	-

3.2.2. Flashing Debian OS to a Micro-SD Card

Two methods are provided to flash images. One is for Mac/Linux, the other is for Windows. For detailed instructions on flashing Debian OS to a Micro-SD card, refer to *Flashing OS to a Micro-SD Card* section in [VisionFive 2 Single Board Computer Quick Start Guide](#).

3.2.3. Generating DTB

To compile the device tree sources (.dtsi files) into device tree blobs (.dtb files) using the device tree compiler (DTC), execute the following command under the root directory of Linux:

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



Tip:

To install **crossbuild-essential-riscv64** package, execute the following command:

```
sudo apt-get install crossbuild-essential-riscv64
```



Tip:

<Configuration_File>: Both `starfive_jh7110_defconfig` and `starfive_visionfive2_defconfig` are applicable.

The following is the example command:

```
make starfive_jh7110_defconfig ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv dtbs
```

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.

3.2.4. Replacing DTB

The Micro-SD card used to flash the images identifies the following directories:

Figure 3-1 Identified Directories

```
/dev/sdb3    30G   21G   8.1G   73%  /media/<UserName>/root
```

3.2.4.1. Method 1: Directly Replacing DTB File

Execute the following command under the root directory of Linux to replace the DTB file:

**Note:**

- `<DTB_File>`:

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.

- `<Mount_Directory>`: The actual mount directory. For example, `/media/<UserName>`.
- `<Kernel_Version>`: The Linux Kernel version. For example, `5.15.0-starfive`.

Example Command:

```
sudo cp arch/riscv/boot/dts/starfive/jh7110-visionfive-v2.dtb  
/media/<UserName>/root/usr/lib/linux-image-5.15.0-starfive/starfive/
```

3.2.4.2. Method 2: Adding Startup Item

To replace dtb file by adding a startup item, perform the following:

1. Execute the following commands under the root directory of Linux:

```
sudo cp arch/riscv/boot/dts/starfive/<DTB_File> <Mount_Direcotry>/  
root/usr/lib/linux-image-<Kernel_Version>/starfive/
```

**Tip:**

- `<DTB_File>`:

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.

- **<Mount_Directory>**: The actual mount directory. For example, `/media/<UserName>`.
- **<Kernel_Version>**: The Linux Kernel version. For example, `5.15.0-starfive`.

Example Command:

```
sudo cp
arch/riscv/boot/dts/starfive/jh7100-starfive-visionfive-v2.dtb /
media/<UserName>/root/usr/lib/linux-image-5.15.0-starfive/starfive/
```

2. Enter SD card mount directory:

```
cd <Mount_Directory>/__boot
```

**Tip:**

<Mount_Directory> refers to the actual mount directory. For example, `/media/UserName`.

3. Open the extlinux.conf file:

```
sudo gedit extlinux.conf
```

4. Added the following lines, save and exit:

```
label l1
    menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive (customized)
    linux /boot/vmlinuz-<Kernel_Version>
    initrd /boot/initrd.img-<Kernel_Version>
    fdt /usr/lib/linux-image-<Kernel_Version>/starfive/<DTB_File>
    append root=/dev/mmcblkp3 rw console=tty0 console=ttyS0,115200 earlycon rootwait
    stmmaceth=chain_mode:1 selinux=0

label l1r
    menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive (customized)(rescue target)
    linux /boot/vmlinuz-<Kernel_Version>
    initrd /boot/initrd.img-<Kernel_Version>
    fdt /usr/lib/linux-image-<Kernel_Version>/starfive/uart_test.dtb
    append root=/dev/mmcblkp3 rw console=tty0 console=ttyS0,115200 earlycon rootwait
    stmmaceth=chain_mode:1 selinux=0 single
```

**Tip:**

- **<DTB_File>**:

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.

- **<Kernel_Version>**: The Linux Kernel version. For example, `5.15.0-starfive`.
- `Debian GNU/Linux bookworm/sid 5.15.0-starfive (customized)`

: The configurable menu item name.

5. When you see the U-Boot startup item, select the menu item set in the previous step, for example, **Debian GNU/Linux bookworm/sid 5.15.0-starfive (customized)**.



Tip:

Multiple startup items can be added according to the actual number of DTB files.

4. GPIO Operations

This section provides commands to configure GPIO:

4.1. Configuring GPIO

1. To configure GPIO, perform the following:

Execute the following command to configure GPIO44:

```
cd /sys/class/gpio  
echo 44 > export
```

2. Locate to the GPIO44 directory:

```
cd gpio44
```



Note:

In this command, 44 represents the Sys number of the pin. For more information, see [GPIO Pinout \(on page 9\)](#).

3. Configure the direction of GPIO44 as in:

```
echo in > direction
```

4. Alternatively, configure the direction of GPIO44 as out:

```
echo out > direction
```

5. Configure the voltage level of GPIO44 as high:

```
echo 1 > value
```



Tip:

You can use an oscilloscope to check the voltage level.

6. Configure the voltage level of GPIO44 as low:

```
echo 0 > value
```



Tip:

You can use an oscilloscope to check the voltage level.

7. Connect the **3.3V Power** pin with the GPIO44, and check the voltage level of GPIO44:

```
cat value
```

8. Connect the **GND** pin with the GPIO44, and check the voltage level of GPIO44:

```
cat value
```

5. I2C Operations

This chapter describes how to configure and debug I2C GPIO.

5.1. Configuring I2C GPIO

Perform the following procedures to configure I2C:

- [Hardware Setup \(on page 16\)](#)
- [Configuring dts File \(on page 17\)](#)

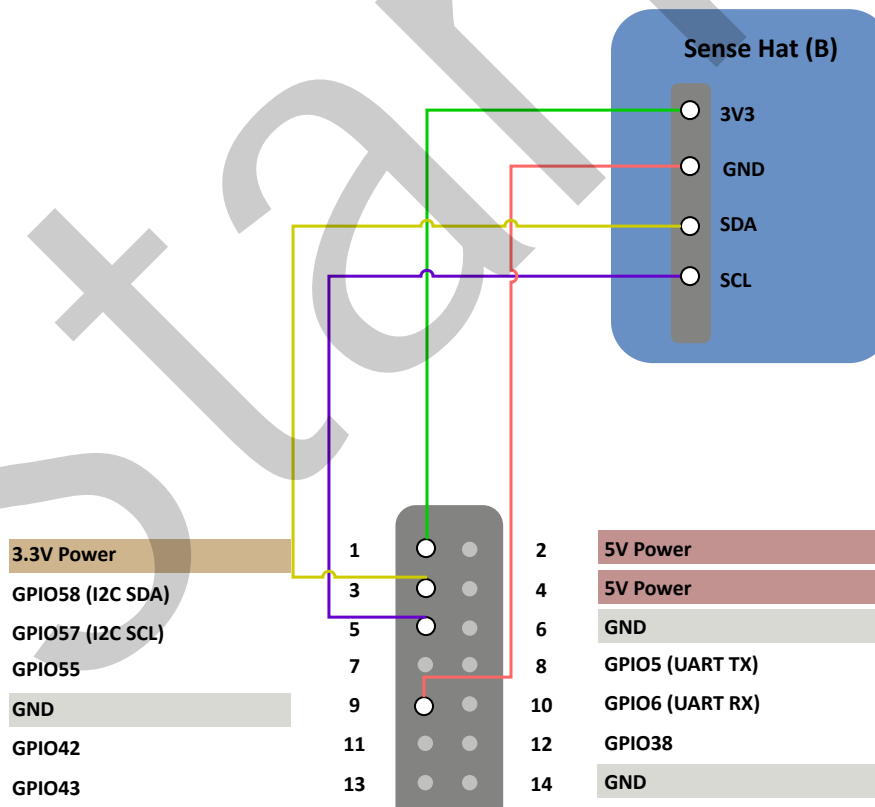
5.1.1. Hardware Setup

The following table and figure describe how to connect Sense HAT to the 40-pin header:

Table 5-1 Connect Sense Hat (B) to the 40-Pin Header

Sense HAT (B)	40-Pin GPIO Header	
	Pin Number	Pin Name
3V3	1	3.3V Power
GND	9	GND
SDA	3	GPIO58 (I2C SDA)
SCL	5	GPIO57 (I2C SCL)

Figure 5-1 Connect Sense Hat (B) to the 40-Pin GPIO Header



5.1.2. Configuring dts File

7 channels of I2C bus are supported: i2c0 to i2c6.

The DTSI file, `jh7110-visionfive-v2.dtsi`, is under `/linux/arch/riscv/boot/dts/starfive`.

The following is the default setting. You can configure the unoccupied GPIOs as required.

Figure 5-2 Example File Content

```

81     i2c0_pins: i2c0-pins {
82         i2c0-pins-scl {
83             sf,pins = <PAD_GPIO57>;
84             sf,pinmux = <PAD_GPIO57_FUNC_SEL 0>;
85             sf,pin-ioconfig = <IO(GPIO_IE(1) | (GPIO_PU(1)))>;
86             sf,pin-gpio-dout = <GPO_LOW>;
87             sf,pin-gpio-doen = <OEN_I2C0_IC_CLK_OE>;
88             sf,pin-gpio-din = <GPI_I2C0_IC_CLK_IN_A>;
89         };
90
91         i2c0-pins-sda {
92             sf,pins = <PAD_GPIO58>;
93             sf,pinmux = <PAD_GPIO58_FUNC_SEL 0>;
94             sf,pin-ioconfig = <IO(GPIO_IE(1) | (GPIO_PU(1)))>;
95             sf,pin-gpio-dout = <GPO_LOW>;
96             sf,pin-gpio-doen = <OEN_I2C0_IC_DATA_OE>;
97             sf,pin-gpio-din = <GPI_I2C0_IC_DATA_IN_A>;
98         };
99     };
100

```



Note:

The I2C GPIO pin number is the number indicated in the **Pin Name**. For more details about the GPIO Pin Name, see the [GPIO Pinout \(on page 9\)](#) in this document. The pin names of the I2C GPIO are listed as follows:

- GPIO58 (I2C SDA)
- GPIO57 (I2C SCL)

5.2. Debugging I2C GPIO

Perform the following steps to debug I2C:

1. Execute the following command to scan the bus:

```
i2cdetect -l
```

Result:

Figure 5-3 Example Output

```

root@starfive:~# i2cdetect -l
i2c-0  i2c          Synopsys DesignWare I2C adapter      I2C adapter
i2c-2  i2c          Synopsys DesignWare I2C adapter      I2C adapter
i2c-5  i2c          Synopsys DesignWare I2C adapter      I2C adapter
i2c-6  i2c          Synopsys DesignWare I2C adapter      I2C adapter
i2c-7  i2c          Inno HDMI                      I2C adapter

```

2. Execute the following command to detect the device:

```
i2cdetect -y -r 0
```

**Tip:**

0 is the I2C bus number.

Result:**Figure 5-4 Example Output**

```
root@starfive:~# i2cdetect -y -r 0
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 29 -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- 48 -- -- -- -- --
50: -- -- -- -- -- -- -- -- 5c -- -- --
60: -- -- -- -- -- -- 68 -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```

In this figure, the detected devices are 0x29, 0x48, 0x5c, 0x68, and 0x70.

- Execute the following command to read register content:

```
i2cget -f -y 0 0x5c 0x0f
```

**Tip:**

- 0: I2C bus number
- 0x5c: I2C device address
- 0x0f: Memory address

Result:**Figure 5-5 Example Output**

```
root@starfive:~# i2cget -f -y 0 0x5c 0x0f
0xb1
```

The register content is 0xb1 in this output.

- Execute the following command to write register data:

```
i2cset -y 0 0x5c 0x11 0x10
```

**Tip:**

- 0: I2C bus number.
- 0x5c: I2C device address.
- 0x11: Memory address.
- 0x10: The content to be written in the register.

- Execute the following to read all register values:

```
i2cdump -y 0 0x5c
```

**Tip:**

- 0: I2C bus number
- 0x5c: I2C device address

Result:**Figure 5-6 Example Output**

```

root@starfive:~# i2cdump -y 0 0x5c
No size specified (using byte-data access)
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  0123456789abcdef
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1 .....?
10: 00 10 00 00 00 00 00 00 00 00 00 00 01 b1 68 6e .?.....??hn
20: 00 00 00 00 00 00 00 02 71 8f 2f 00 00 00 00 00 .....?q?/.....
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
40: 72 e4 03 20 8a 0f 05 49 00 06 27 7b 8b 11 0b 44 r?? ???I.?'{???D
50: 42 fd 7b 0e 00 71 8f 2f 06 03 15 0a b9 04 80 c0 B?{?.q?/????????
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b1 .....?
90: 00 10 00 00 00 00 00 00 00 00 00 00 01 b1 68 6e .?.....??hn
a0: 00 00 00 00 00 00 00 00 71 8f 2f 00 00 00 00 00 .....q?/.....
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
c0: 72 e4 03 20 8a 0f 05 49 00 06 27 7b 8b 11 0b 44 r?? ???I.?'{???D
d0: 42 fd 7b 0e 00 71 8f 2f 06 03 15 0a b9 04 80 c0 B?{?.q?/????????
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

6. SPI Operations

This chapter describes how to configure and debug SPI GPIO.

6.1. Configuring SPI GPIO

The DTSI file, `jh7110-visionfive-v2.dtsi`, is under `/linux/arch/riscv/boot/dts/starfive`.

7 channels of SPI bus are supported: `spi0` to `spi6`.

6.1.1. Modify Pins

The configured SPI GPIO number is the number indicated in the Pin Name. For more details about the GPIO Pin Name, see the [GPIO Pinout \(on page 9\)](#) in this document. You can configure the unoccupied pins. The following are the default settings in the `jh7110-visionfive-v2.dtsi`:

Figure 6-1 Modify Pins

```
187     ssp0_pins: ssp0-pins {
188         ssp0-pins_tx {
189             sf,pins = <PAD_GPIO52>;
190             sf,pinmux = <PAD_GPIO52_FUNC_SEL 0>;
191             sf,pin-ioconfig = <IO(GPIO_IE(1))>;
192             sf,pin-gpio-dout = <GPO_SPI0_SSPTXD>;
193             sf,pin-gpio-doen = <OEN_LOW>;
194         };
195
196         ssp0-pins_rx {
197             sf,pins = <PAD_GPIO53>;
198             sf,pinmux = <PAD_GPIO53_FUNC_SEL 0>;
199             sf,pin-ioconfig = <IO(GPIO_IE(1))>;
200             sf,pin-gpio-doen = <OEN_HIGH>;
201             sf,pin-gpio-din = <GPI_SPI0_SSPRXD>;
202         };
203
204         ssp0-pins_clk {
205             sf,pins = <PAD_GPIO48>;
206             sf,pinmux = <PAD_GPIO48_FUNC_SEL 0>;
207             sf,pin-ioconfig = <IO(GPIO_IE(1))>;
208             sf,pin-gpio-dout = <GPO_SPI0_SSPCLKOUT>;
209             sf,pin-gpio-doen = <OEN_LOW>;
210         };
211
212         ssp0-pins_cs {
213             sf,pins = <PAD_GPIO49>;
214             sf,pinmux = <PAD_GPIO49_FUNC_SEL 0>;
215             sf,pin-ioconfig = <IO(GPIO_IE(1))>;
216             sf,pin-gpio-dout = <GPO_SPI0_SSPFSSOUT>;
217             sf,pin-gpio-doen = <OEN_LOW>;
218         };
219     };
```

6.2. Debugging SPI GPIO

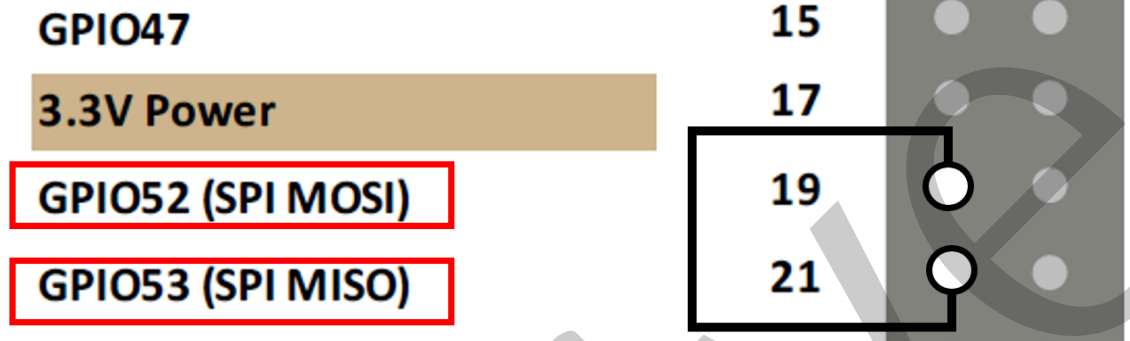
This section provides steps for loopback test and testing SPI with the ADXL345 module.

6.2.1. Loopback Test

The following steps are provided for the loopback test:

1. Wiring: Connect Pin19 with Pin 21 as the following:

Figure 6-2 Connect Pin 19 with 21



2. Locate to the following path for the test tool, `spidev_test.c`:

```
cd /linux/tools/spi
```

3. Execute the following command under the test tool directory:

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

Result:

The output file is `spidev_test` in the same directory.

4. Upload `spidev_test` to VisionFive 2, and change the execution permission by executing the following:

```
chmod +x spidev_test
```

5. Confirm the SPI device.

```
ls /dev/spidev*
```

Result:

Figure 6-3 Example Output

```
root@starfive:~# ls /dev/spi*
/dev/spidev1.0
root@starfive:~#
```

In this output, `spidev1.0` is the device name.

6. Execute the following command to perform the test:

```
./spidev_test -D /dev/spidev1.0 -v -p string_to_send
```



Tip:

`spidev1.0` is the device name got from the previous step.

Result:

Figure 6-4 Example Output

```
root@starfive:~/spi# ./spidev_test -D /dev/spidev1.0 -v -p string_to_send
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 kHz)
TX | 73 74 72 69 6E 67 5F 74 6F 5F 73 65 6E 64 |string_to_send|
RX | 73 74 72 69 6E 67 5F 74 6F 5F 73 65 6E 64 |string_to_send|
root@starfive:~/spi#
```

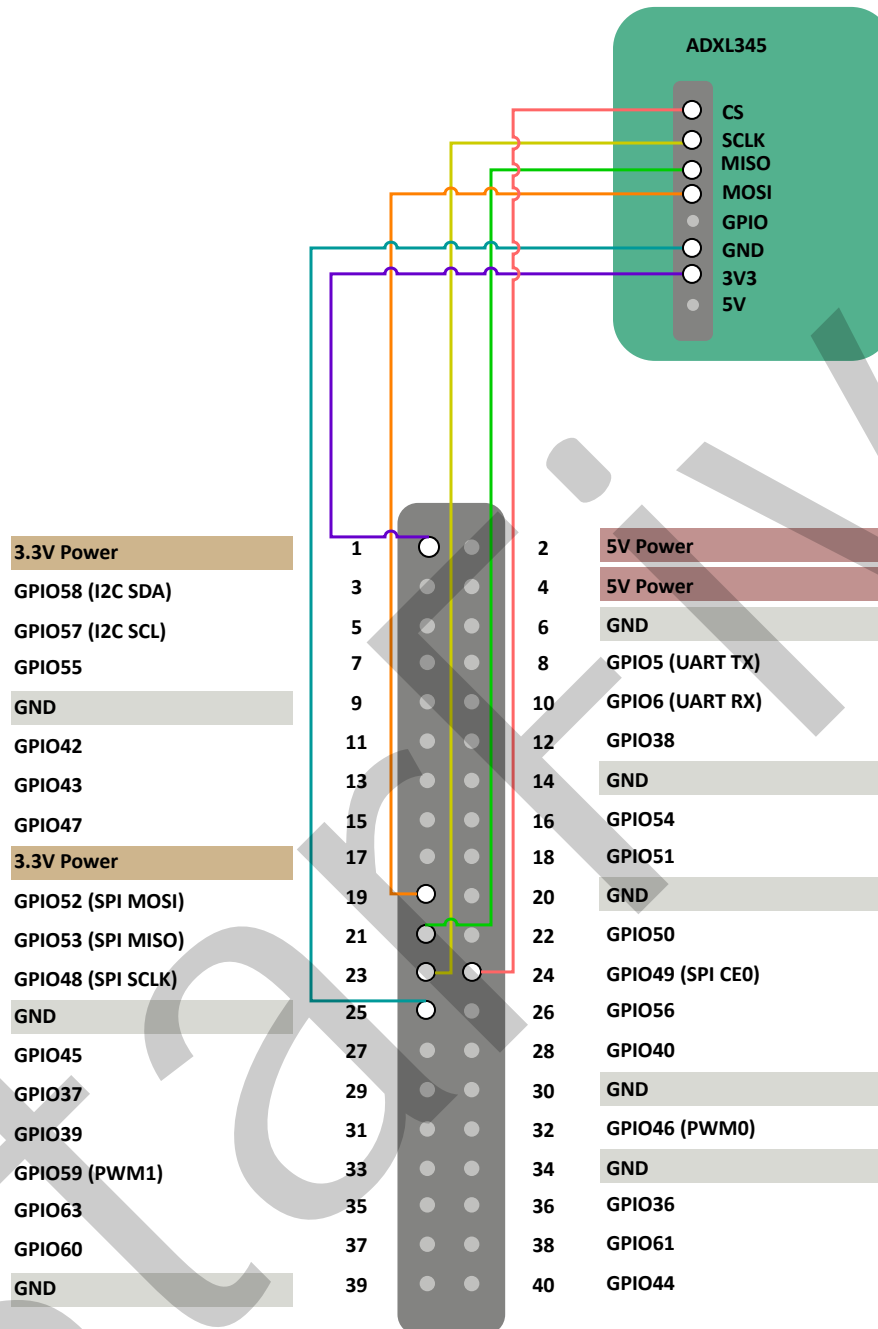
In this figure, the **TX** and **RX** output are the same, which indicates the test is successful.

6.2.2. Testing SPI with ADXL345 Module

Perform the following steps to test SPI with the ADXL345 module:

1. Connect the ADXL345 module to the 40-pin header as the following:

Figure 6-5 Connect ADXL345 Module to the Header



2. Locate to the following path for test tool, `spidev_test.c`:

```
cd /linux/tools/spi
```

3. Execute the following command under the test tool directory:

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

Result:

The output file is `spidev_test` in the same directory.

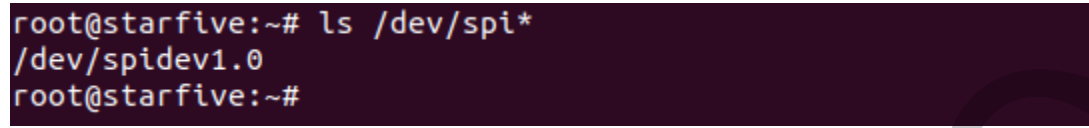
4. Upload `spidev_test` to VisionFive 2, and change the execution permission by executing the following:

```
chmod +x spidev_test
```

5. Confirm the SPI device.

```
ls /dev/spidev*
```

Figure 6-6 Example Output



```
root@starfive:~# ls /dev/spi*  
/dev/spidev1.0  
root@starfive:~#
```

In this output, `spidev1.0` is the device name.

6. Execute the following to read the device ID:

```
./spidev_test -H -O -D /dev/spidev1.0 -v -p \\x80\\x00
```

7. Execute the following to read the value for multiple registers:

```
./spidev_test -H -O -D /dev/spidev1.0 -v -p \\xec\\x00\\x00\\x00\\x00\\x00\\x00
```

8. Execute the following to read:

```
./spidev_test -H -O -D /dev/spidev1.0 -v -p \\x9e\\x00
```

9. Execute the following to write:

```
./spidev_test -H -O -D /dev/spidev1.0 -v -p \\x1e\\xaa
```

10. Execute the following to read the verification:

```
./spidev_test -H -O -D /dev/spidev1.0 -v -p \\x9e\\x00
```


7. PWM Operations

This chapter describes how to configure and debug PWM GPIO:

7.1. Configuring PWM GPIO

The DTSI file, `jh7110-visionfive-v2.dtsi`, is under `/linux/arch/riscv/boot/dts/starfive`.

8 channels of PWM are supported at the most.

7.1.1. Modify Pin

The following figure shows the example file content to modify the pin:

Figure 7-1 Example File Content

```
169     pwm_pins: pwm-pins {
170         pwm_ch0-pins {
171             sf,pins = <PAD_GPIO46>;
172             sf,pinmux = <PAD_GPIO46_FUNC_SEL 0>;
173             sf,pin-ioconfig = <IO(GPIO_IE(1))>;
174             sf,pin-gpio-dout = <GPO_PTC0_PWM_0>;
175             sf,pin-gpio-doen = <OEN_PTC0_PWM_0_OE_N>;
176         };
177
178         pwm_ch1-pins {
179             sf,pins = <PAD_GPIO59>;
180             sf,pinmux = <PAD_GPIO59_FUNC_SEL 0>;
181             sf,pin-ioconfig = <IO(GPIO_IE(1))>;
182             sf,pin-gpio-dout = <GPO_PTC0_PWM_1>;
183             sf,pin-gpio-doen = <OEN_PTC0_PWM_1_OE_N>;
184         };
185     };
186
```

The configured PWM GPIO number is the number contained in the **Pin Name**. For more details about the GPIO Pin Name, see the [GPIO Pinout \(on page 9\)](#) in this document.

7.1.2. PWM and Pin Name Mapping

The following table describes the PWM and pin name mapping:

Table 7-1 PWM and Pin Name Mapping

PWM	GPIO (Pin Name)
PWM0	GPIO46
PWM1	GPIO59

7.2. Debugging PWM GPIO

This section describes how to debug PWM GPIO:

1. Execute the following to configure the PWM channel:

```
cd /sys/class/pwm/pwmchip0
echo 0 > export
```

2. Execute the following to configure the PWM period:

```
cd pwm0  
echo 5000000 > period
```

3. Execute the following to configure the PWM duty cycle:

```
echo 1000000 > duty_cycle
```

4. Use an oscilloscope to measure the corresponding pin and check the PWM period and duty cycle.



Note:

A duty cycle is the fraction of one period in which a signal or system is active.

8. UART Operations

This chapter describes how to configure and debug UART GPIO:

8.1. Configuring UART GPIO

The DTSI file, `jh7110-visionfive-v2.dtsi`, is under `/linux/arch/riscv/boot/dts/starfive`.

6 channels of UART are supported at the most.

The configured UART GPIO number is the number contained in the **Pin Name**. For more details about the GPIO Pin Name, see the [GPIO Pinout \(on page 9\)](#) in this document.

8.1.1. Modifying dts

To modify dts file, perform the following steps:

1. Add aliases of uart1 or uart2 on the aliases node. The following is an example:

Figure 8-1 Example Configuration

```
aliases {
    spi0 = &qspi;
    gpio0 = &gpio;
    ethernet0 = &gmac0;
    ethernet1 = &gmac1;
    mmc0 = &sdio0;
    mmc1 = &sdio1;
    serial0 = &uart0;
    serial1 = &uart1;
    serial2 = &uart2;
    serial3 = &uart3;
```

2. Add uart1 or uart2 node on the dts. The following is an example:

Figure 8-2 Example Configuration

```
&uart0 {
    pinctrl-names = "default";
    pinctrl-0 = <&uart0_pins>;
    status = "okay";
};

&uart1 {
    pinctrl-names = "default";
    pinctrl-0 = <&uart1_pins>;
    status = "okay";
};

&uart2 {
    pinctrl-names = "default";
    pinctrl-0 = <&uart2_pins>;
    status = "okay";
};
```

3. Add uart1_pins or uart2_pins node on the &gpio node:



Note:

The configured UART GPIO number is the number contained in the **Pin Name**. You can configure the unoccupied pins. For more details about the GPIO Pin Name, see the [GPIO Pinout \(on page 9\)](#) in this document.

Figure 8-3 Example Configuration

```
&gpio {
    uart0_pins: uart0-pins {
        uart0-pins-tx {
            starfive,pins = <PAD_GPIO5>;
            starfive,pin-ioconfig = <IO(GPIO_IE(1) | GPIO_DS(3))>;
            starfive,pin-gpio-dout = <GPO_UART0_SOUT>;
            starfive,pin-gpio-doen = <OEN_LOW>;
        };

        uart0-pins-rx {
            starfive,pins = <PAD_GPIO6>;
            starfive,pinmux = <PAD_GPIO6_FUNC_SEL 0>;
            starfive,pin-ioconfig = <IO(GPIO_IE(1) | GPIO_PU(1))>;
            starfive,pin-gpio-doen = <OEN_HIGH>;
            starfive,pin-gpio-din = <GPI_UART0_SIN>;
        };
    };

    uart1_pins: uart1-pins {
        uart1-pins-tx {
            starfive,pins = <PAD_GPIO63>;
            starfive,pinmux = <PAD_GPIO63_FUNC_SEL 0>;
            starfive,pin-ioconfig = <IO(GPIO_IE(1) | GPIO_DS(3))>;
            starfive,pin-gpio-dout = <GPO_UART1_SOUT>;
            starfive,pin-gpio-doen = <OEN_LOW>;
        };
    };
};
```

8.1.1.1. UART and DEV Mapping

The following table describes the UART and DEV mapping:

Table 8-1 UART and DEV Mapping

UART	DEV
UART1	/dev/ttyS1
UART2	/dev/ttyS2

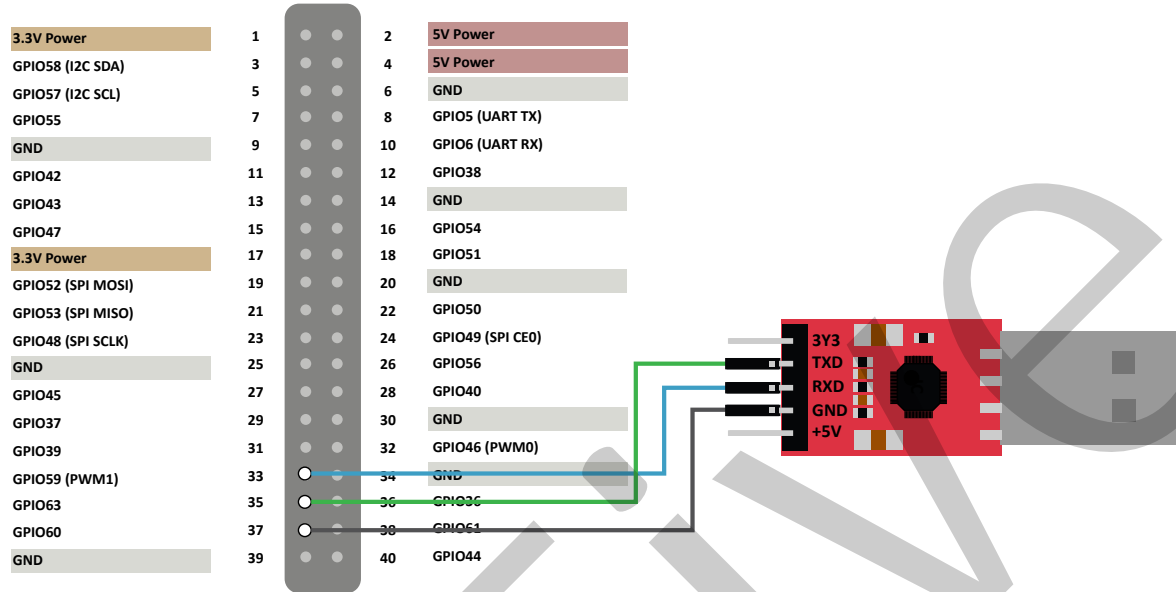
8.2. Debugging UART GPIO

8.2.1. Hardware Setup

To set up the hardware, perform the following steps:

1. Connect the jumper wires from the USB-to-Serial Converter to the 40-Pin GPIO header of the VisionFive 2 as follows.

Figure 8-4 Connect the Converter to the Header



2. Connect the other end of the USB-to-Serial Converter to your device (Windows/Mac/Linux).

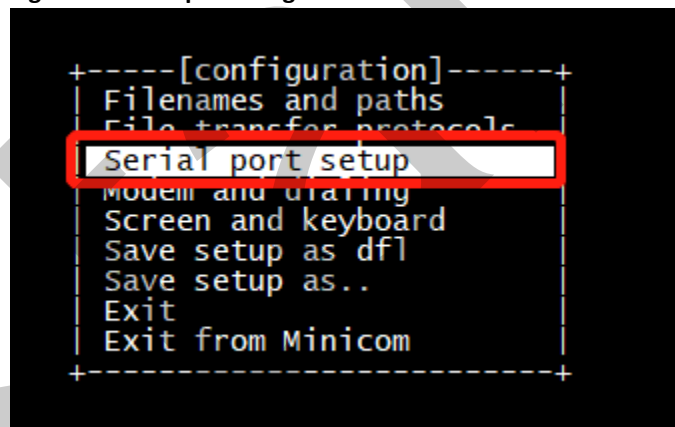
8.2.2. Debugging UART Send and Receive Functions

1. Configure VisionFive 2 Minicom:

```
sudo minicom -s
```

2. Select **Serial port setup**, and configure Minicom as follows:

Figure 8-5 Example Configuration

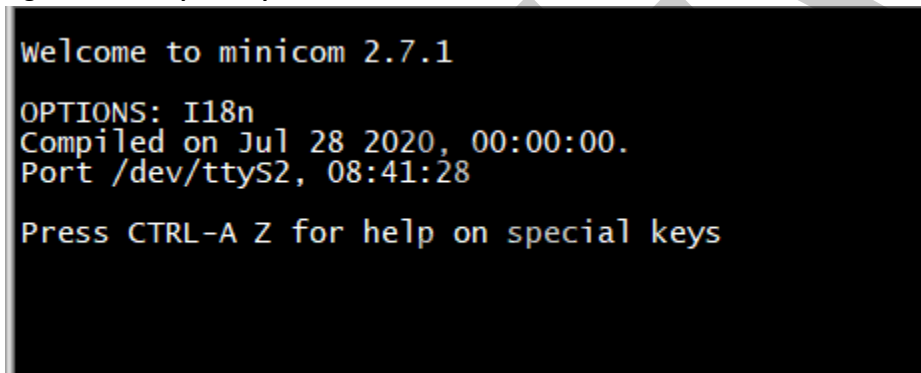




3. Start VisionFive 2 minicom by typing the following command on the PC:

```
minicom -o -D /dev/ttyS1
```

Figure 8-7 Example Output

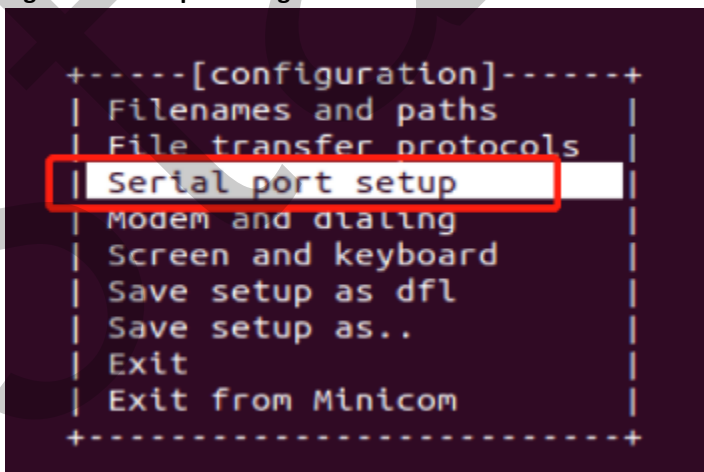


4. Configure Ubuntu minicom by typing the following:

```
sudo minicom -s
```

5. Select **Serial port setup**, and configure minicom as follows:

Figure 8-8 Example Configuration



```

+-----+
| A -   Serial Device       : /dev/ttyUSB0
| B - Lockfile Location    : /var/lock
| C -   Callin Program     :
| D - Callout Program      :
| E -   Bps/Par/Bits       : 115200 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control: No
+-----+

Change which setting? 0

+-----+
| Screen and keyboard
| Save setup as dfl
| Save setup as..
| Exit
| Exit from Minicom
+-----+

```

**Note:**

Serial Device can be detected by command `dmesg | grep tty` on Ubuntu

Figure 8-10 Example Command and Output

```

xiangyao@xiangyao-VirtualBox:~$ dmesg | grep tty
[  0.158110] printk: console [tty0] enabled
[  5.322731] ttyS2: LSR safety check engaged!
[  5.323281] ttyS2: LSR safety check engaged!
[91855.795788] usb 1-2: pl2303 converter now attached to ttyUSB0
[92154.097583] ttyS2: LSR safety check engaged!

```

6. Start Ubuntu minicom, you can see as follows:

Figure 8-11 Example Output

```

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB0, 08:40:51

Press CTRL-A Z for help on special keys

```

Test UART Send:

7. To test UART send function, you can input characters, such as `hello ubuntu`, on the VisionFive 2 minicom. Then you will see the character are outputted on the Ubuntu minicom as the following:

Figure 8-12 Test UART Send

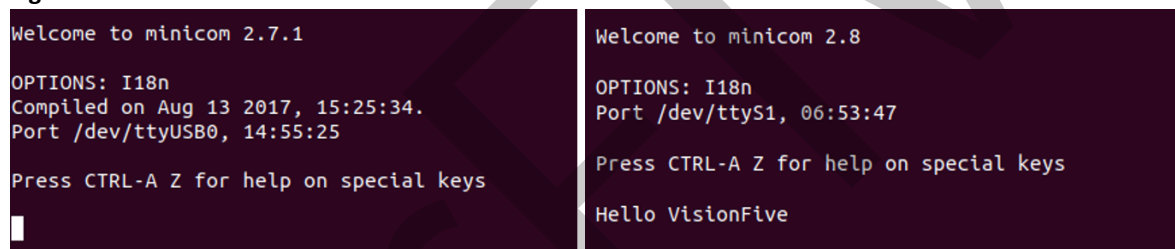


- Figure on the Left: Ubuntu minicom interface
- Figure on the Right: VisionFive 2 minicom interface

Test UART Receive:

8. To test UART receive, you can input characters, such as `hello visionfive` on the Ubuntu minicom. Then you will see the characters are outputted on the VisionFive 2 minicom:

Figure 8-13 Test UART Receive



- Figure on the Left: Ubuntu minicom interface
- Figure on the Right: VisionFive 2 minicom interface

9. Peripheral Examples

In this demo, Sense Hat (B) is used. For the detailed specifications, refer to [https://www.waveshare.com/wiki/Sense_HAT_\(B\)](https://www.waveshare.com/wiki/Sense_HAT_(B)).



Note:

The official libraries of BCM2835, Python, and wiringPi are not supported, and we use the system call instead. The examples are required to be modified.

9.1. Sense Hat (B) Example

9.1.1. Hardware Setup

The following table and figure describe how to connect Sense HAT to the 40-pin header:

Table 9-1 Connect Sense Hat (B) to the 40-Pin Header

Sense HAT (B)	Pin Number
3V3	1
GND	9
SDA	3
SCL	5

Figure 9-1 Connect Sense Hat (B) to the 40-Pin Header

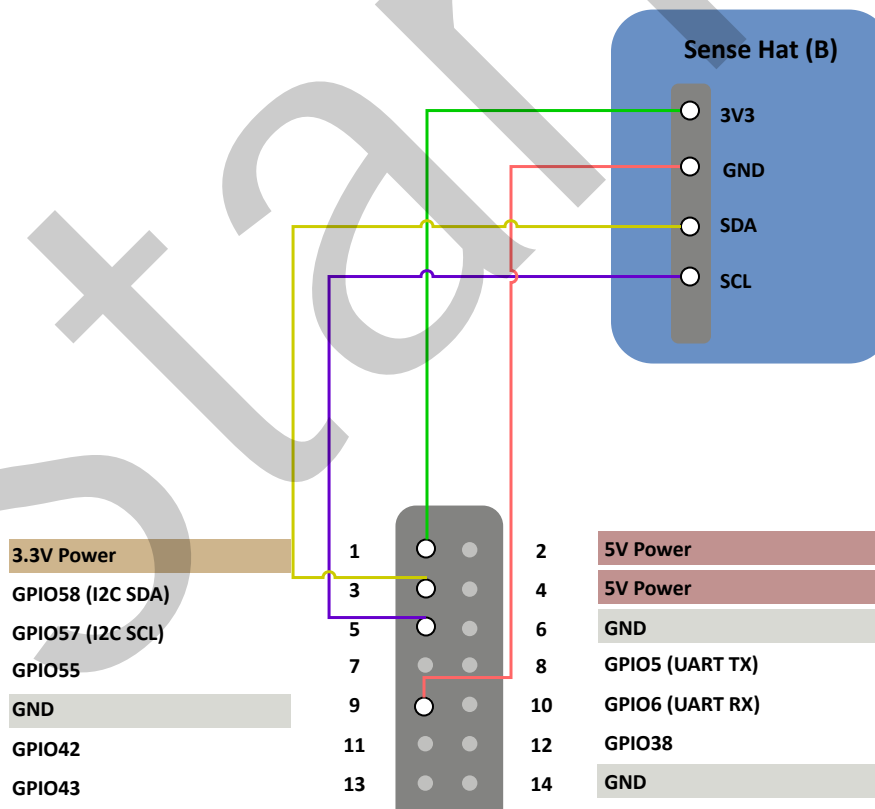
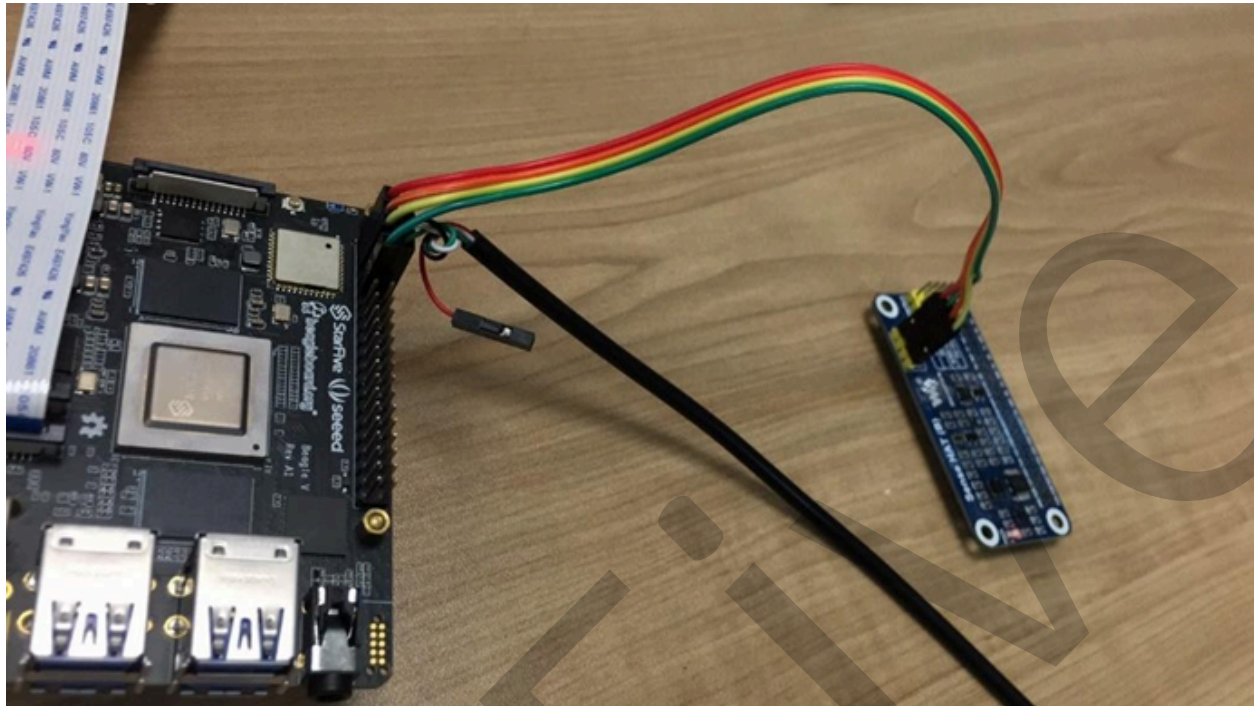


Figure 9-2 Connect Sense Hat (B) to the 40-Pin Header



9.1.2. Running Example with Sense Hat (B)

Take SHTC3 sensor as an example:

1. Download the source code from: [SHTC3_dev.c](#)
2. (Optional) Install the tool to compile. The following is an example to install:

```
sudo apt-get install gcc-riscv64-linux-gnu
```



Note:

This step can be skipped if the tool has been installed.

3. Execute the following to compile:

```
riscv64-linux-gnu-gcc SHTC3_dev.c -o shtc3
```

Result:

The output file is `shtc3` in the same directory.

4. Copy the executable code from the `shtc3` file to the board, and change the execution permission by executing the following command:

```
chmod +x shtc3
```

5. Execute the following command to run:

```
./shtc3
```

Result:

The following output indicates the execution is successful:

```
root@starfive:~# ./shtc3

SHTC3 Sensor Test Program ...
Fopen : /dev/i2c-0
```

```

Temperature = 52.20°C , Humidity = 55.32%
Temperature = 23.81°C , Humidity = 55.29%
Temperature = 23.79°C , Humidity = 55.30%
Temperature = 23.82°C , Humidity = 55.29%
Temperature = 23.81°C , Humidity = 55.29%
Temperature = 23.82°C , Humidity = 55.30%

```

9.2. 2.4inch LCD Module Example

A 2.4inch LCD Module is used in this example. For the detailed specifications, refer to the following: https://www.waveshare.com/wiki/2.4inch_LCD_Module.



Note:

The official examples are required to be modified for this demo.

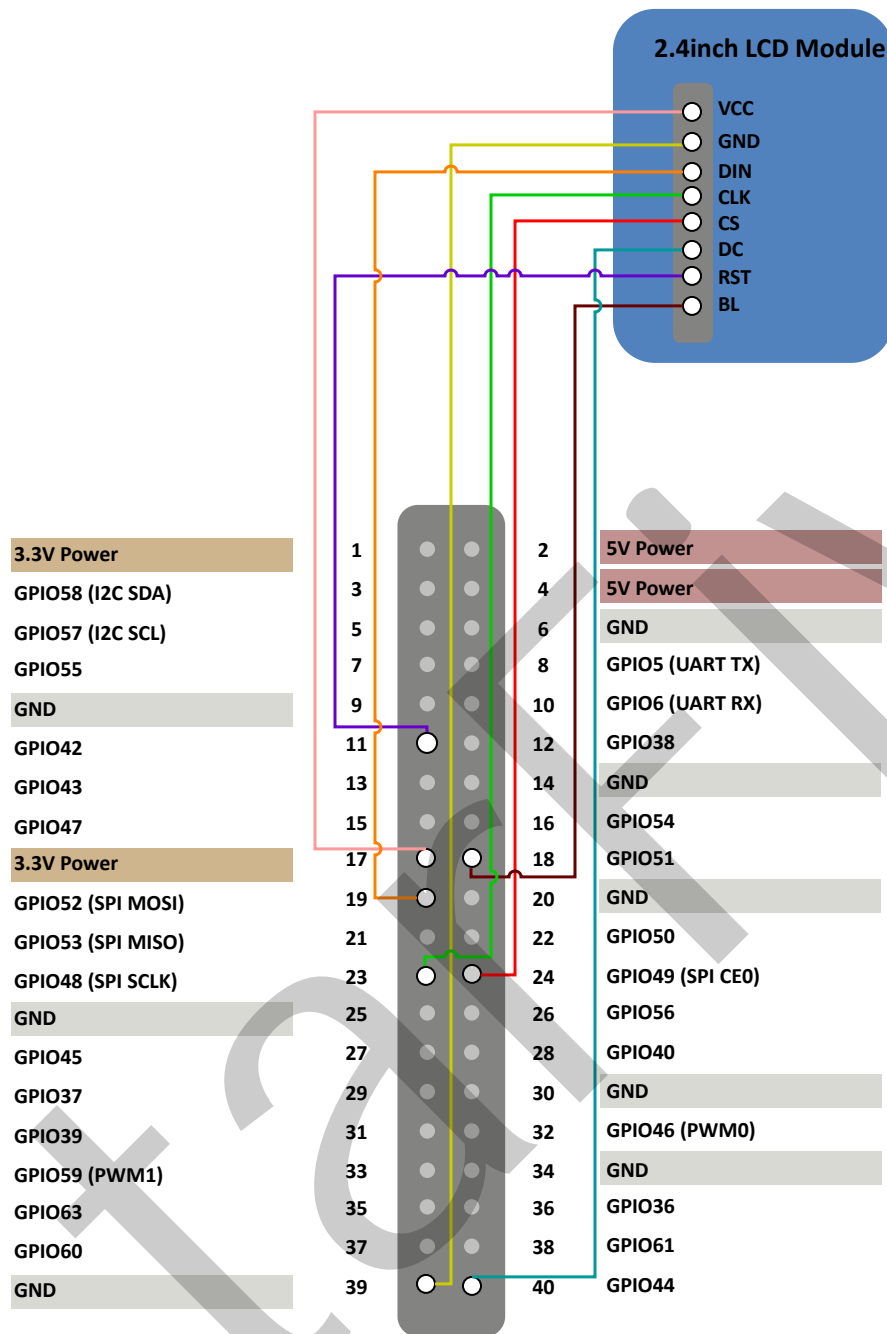
9.2.1. Hardware Setup

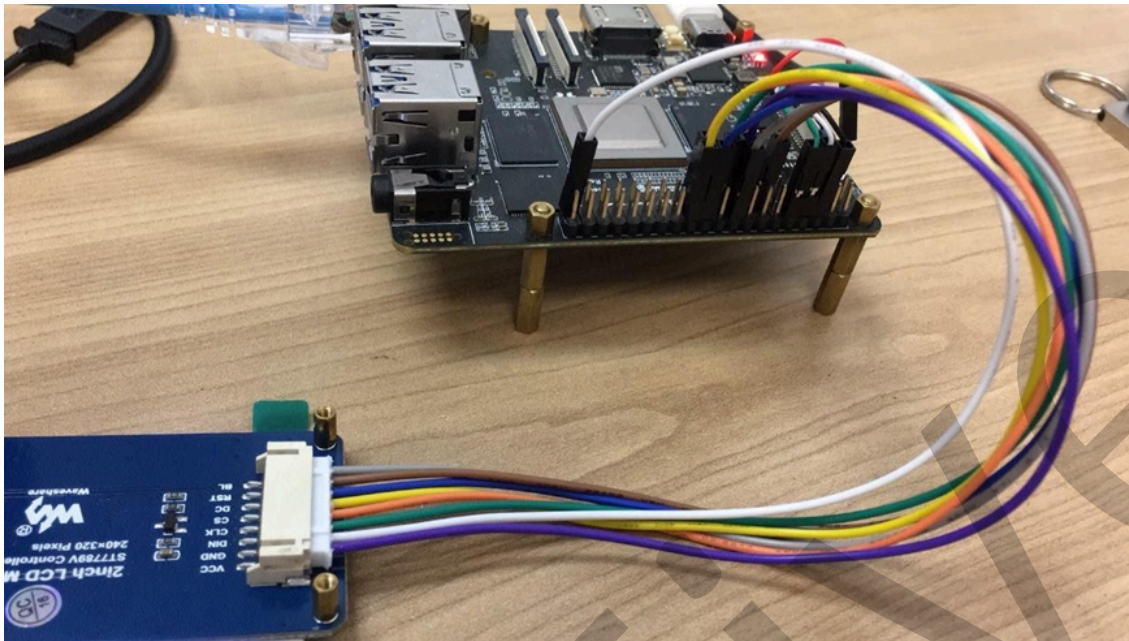
The following table and figure describe how to connect the 2.4inch LCD module with the 40-pin header:

Table 9-2 Connect 2.4inch LCD with 40-pin Header

2.4inch LCD Module	Pin Number
VCC	17
GND	39
DIN	19
CLK	23
CS	24
DC	40
RST	11
BL	18

Figure 9-3 Connect 2.4inch LCD with 40-Pin Header





9.2.2. Executing Example

Perform the following steps to execute the example:

1. Download the source code from [LCD Demo](#).
2. Execute the following command to copy the code to the board. For example, VisionFive 2.

```
tar -xvf LCD_Demo.tar.gz
cd visionfive2/
./main 2.4
```

Result:

The following two figures will be displayed in turn. One is the photo of VisionFive 2, the other is the official example figure.

Figure 9-5 Example Output

