StarFive 赛昉科技

StarFive 40-Pin GPIO Header User Guide

Version: V1.2

Date: 2022/05/05

Doc ID: VisionFive-UGEN-001-V1.2

Legal Statements

Important legal notice before reading our documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to the product development. Shanghai StarFive Technology Co., Ltd. (hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

StarFive does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

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
V1	2021-12-08	The first official release.
V1.1	2021-12-27	• Updated the command and improved the description in the *Generating dtb* section. • Added a note in the *GitHub Repository* section.
V1.2	2022-05-05	Updated the download link for the 2inch LCD Module Example.

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

List of Figures

1. Overview

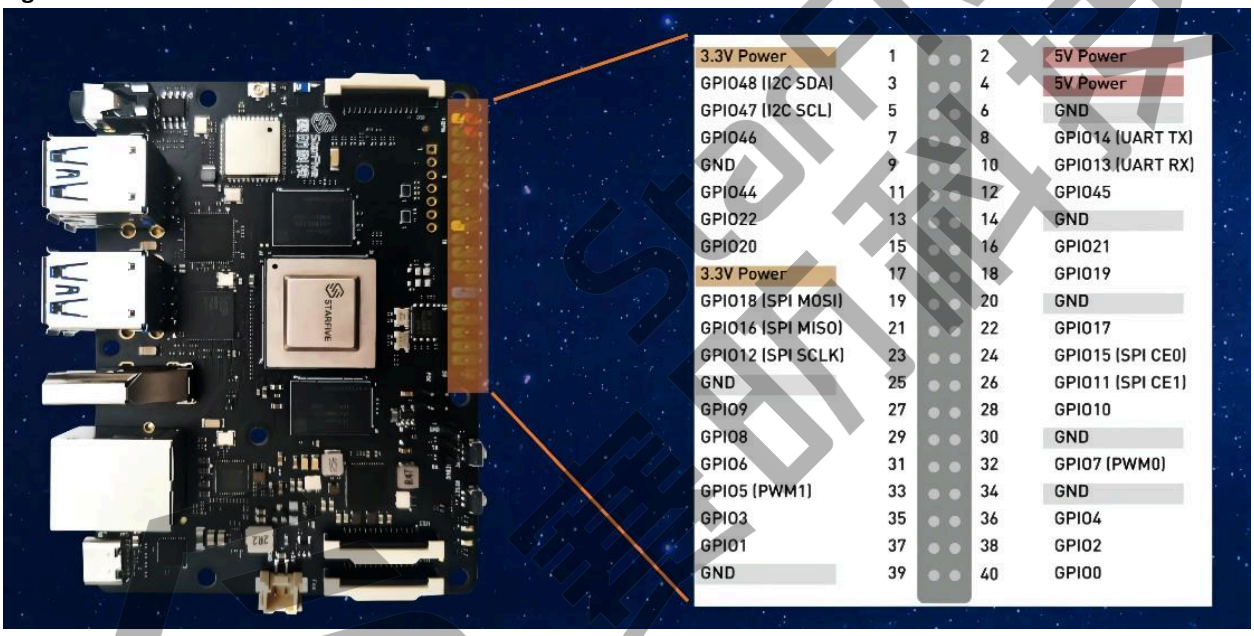
The 40-pin header allows StarFive single board computers, including both StarLight and VisionFive, to interface with a variety of external components, which enables developers to create their projects. This document is intended to:

- 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 14)*, I2C Operations *(on page 15)*, SPI Operations *(on page 19)*, PWM Operations *(on page 24)*, and UART Operations *(on page 26)* 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. The VisionFive board is taken as an example:

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 Pinout

dts	sys	Pin Name	Num	Num	Pin Name	sys	dts
		3.3V Power	1	2	5V Power		
i2c1	i2c-1	GPIO48 (I2C SDA)	3	4	5V Power		
i2c1	i2c-1	GPIO47 (I2C SCL)	5	6	GND		
	494	GPIO46	7	8	GPIO14 (UART TX)	ttyS0	uart3
		GND	9	10	GPIO13 (UART RX)	ttyS0	uart3
	492	GPIO44	11	12	GPIO45	PWM2	
	470	GPIO22	13	14	GND		
	468	GPIO20	15	16	GPIO21	469	
		3.3V Power	17	18	GPIO19	467	
spi2	spidev0.0	GPIO18 (SPI MOSI)	19	20	GND		
spi2	spidev0.0	GPIO16 (SPI MISO)	21	22	GPIO17	465	
spi2	spidev0.0	GPIO12 (SPI SCLK)	23	24	GPIO15 (SPI CE0)	spidev0.0	spi2
		GND	25	26	GPIO11 (SPI CE1)	spidev0.0	spi2
	457	GPIO9	27	28	GPIO10	458	
	456	GPIO8	29	30	GND		
	454	GPIO6	31	32	GPIO7 (PWM0)	PWM0	
	PWM1	GPIO5 (PWM1)	33	34	GND		
	451	GPIO3	35	36	GPIO4	452	
	449	GPIO1	37	38	GPIO2	450	
		GND	39	40	GPIO0	448	

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	StarFive single board computer	The following boards are applicable: • StarLight • VisionFive
General	M	• 16 GB (or more) micro-SD card • micro-SD card reader • Computer (Windows/MAC/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 Fedora OS into a micro-SD card.
GPIO/ PWM	O	An oscilloscope	The oscilloscope is used to verify the GPIO voltage or to measure the corresponding pin and check the PWM period and duty cycle.
I2C	O	• Sense Hat (B) • Dupont Line	-
SPI	O	• ADXL345 Module • Dupont Line	-
SPI LCD	O	• 2inch LCD Module • Dupont Line	-
UART	O	• 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 Fedora 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	visionfive
dts File under Linux Repo	• jh7100-common.dtsi • jh7100.dtsi	-
Uboot	Uboot	JH7100_upstream
OpenSBI	OpenSBI	master
Fedora image (Alpha version)	Fedora Image	-

3.2.2. Flashing Fedora 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, refer to *Flashing Fedora OS to a Micro-SD Card* section in VisionFive 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:

<Configuration_File>: Both `starfive_jh7100_fedora_defconfig` and `visionfive_defconfig` are applicable.

The following is the example command:

```
make starfive_jh7100_fedora_defconfig ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv dtbs
```

Different boards use different dtb files. The following table describes the relationship:

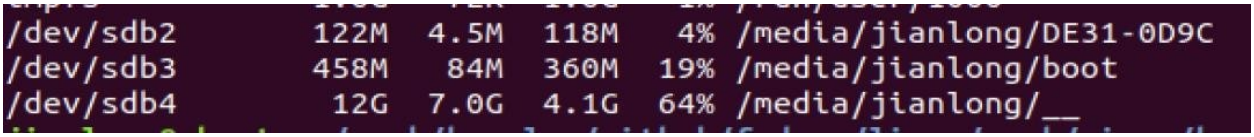
Table 3-3 dtb Files

Board	File
StarLight	`/linux/arch/riscv/boot/dts/starfive/jh7100-beaglev-starlight.dtb`
VisionFive	`/linux/arch/riscv/boot/dts/starfive/jh7100-starfive-visionfive-v1.dtb`

3.2.4. Replacing dtb

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

Figure 3-1 Identified Directories



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:

```
sudo cp arch/riscv/boot/dts/starfive/<dtb_file> <Mount_Directory>/__boot/dtbs/<Kernel_Version>/starfive
```

Note:

- *<dtb_file>* refers to the dtb file name. Different boards use different dtb files. For more information, see Table 3-3 : dtb Files *(on page 11)* table in this document.
- *<Mount_Directory>* refers to the actual mount directory. For example, `/media/jianlong`.
- *<Kernel_Version>* refers to the kernel version number. For example, `5.14.0+`.

Example Command:

```
sudo cp
 arch/riscv/boot/dts/starfive/jh7100-starfive-visionfive-v1.dtb /media/jianlong/__boot/dtbs/5.14.0+/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>/boot/dtbs/
   ```

   Tip:

   - *<dtb_file>* refers to the dtb file name. Different boards use different dtb files. For more information, see Table 3-3 : dtb Files *(on page 11)* table in this document.
   - *<Mount_Directory>* refers to the actual mount directory. For example, `/media/jianlong`.

   Example Command:

   ```
   sudo cp arch/riscv/boot/dts/starfive/jh7100-starfive-visionfive-v1.dtb /media/jianlong/__boot/dtbs/
   ```

2. Enter SD card mount directory:

   ```
   cd <Mount_Direcotry>/__boot
   ```

   Tip:

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

3. Update `grub.cfg`:

   ```
   sudo gedit grub.cfg
   ```

4. Add the following command lines, save and exit:

   ```
   menuentry 'MY Fedora vmlinux-5.14.0+' {
   linux /vmlinuz-5.14.0+ ro root=UUID=f852f7f6-aa4e-4404-8ea9-439568b767a1 rhgb console=tty0
    console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0
   LANG=en_US.UTF-8
   ```

```
devicetree /dtbs/<dtb_File>
initrd /initramfs-<Kernel_Version>.img
}
```

Tip:

In these command lines:

- *<dtb_file>* refers to the dtb file name. Different boards use different dtb files. For more information, see Table 3-3 : dtb Files *(on page 11)* table in this document.
- *<Kernel_Version>*: the kernel version number. For example, `5.14.0+`.
- *MY Fedora vmlinux-5.14.0*: Configurable menu item name.

5. When the grub menu occurs, select the menu item set in the previous step, for example, **MY Fedora vmlinux-5.14.0+**, during startup.

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 GPIO0:

   ```
   cd /sys/class/gpio
   echo 448 > export
   ```

2. Locate to the GPIO0 directory:

   ```
   cd gpio448
   ```

   Note:
   In this command, `448` represents the sys number of the pin. For more information, see GPIO Pinout *(on page 9)*.

3. Configure the direction of GPIO0 as in:

   ```
   echo in > direction
   ```

4. Alternatively, configure the direction of GPIO0 as out:

   ```
   echo out > direction
   ```

5. Configure the voltage level of GPIO0 as high:

   ```
   echo 1 > value
   ```

   Tip:
   You can use an oscilloscope to check the voltage level.

6. Configure the voltage level of GPIO0 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 GPIO0, and check the voltage level of GPIO0:

   ```
   cat value
   ```

8. Connect the **GND** pin with the GPIO0, and check the voltage level of GPIO0:

   ```
   cat value
   ```

5. I2C Operations

This chapter describes how to configure and debug I2C GPIO.

5.1. Configuring I2C GPIO

4 channels of I2C bus are supported: i2c0, i2c1, i2c2, and i2c3.

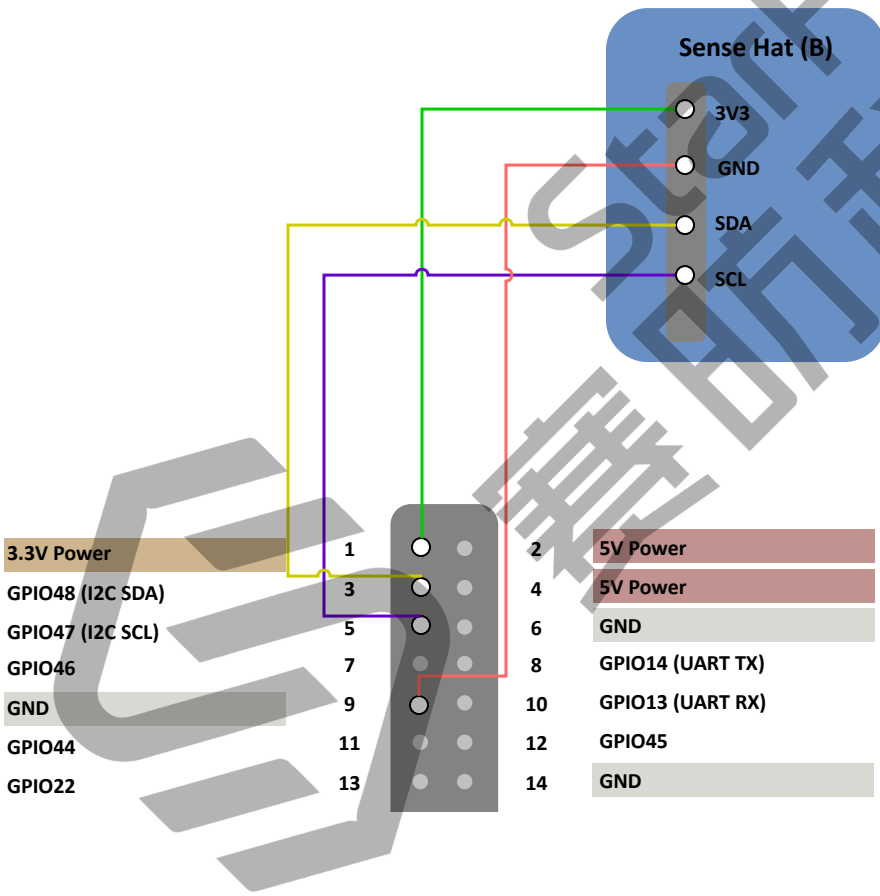
Perform the following to configure I2C:

- Hardware Setup *(on page 15)*
- Configuring dts File *(on page 15)*

5.1.1. Hardware Setup

Connect the Sense Hat (B) to the header as the following:

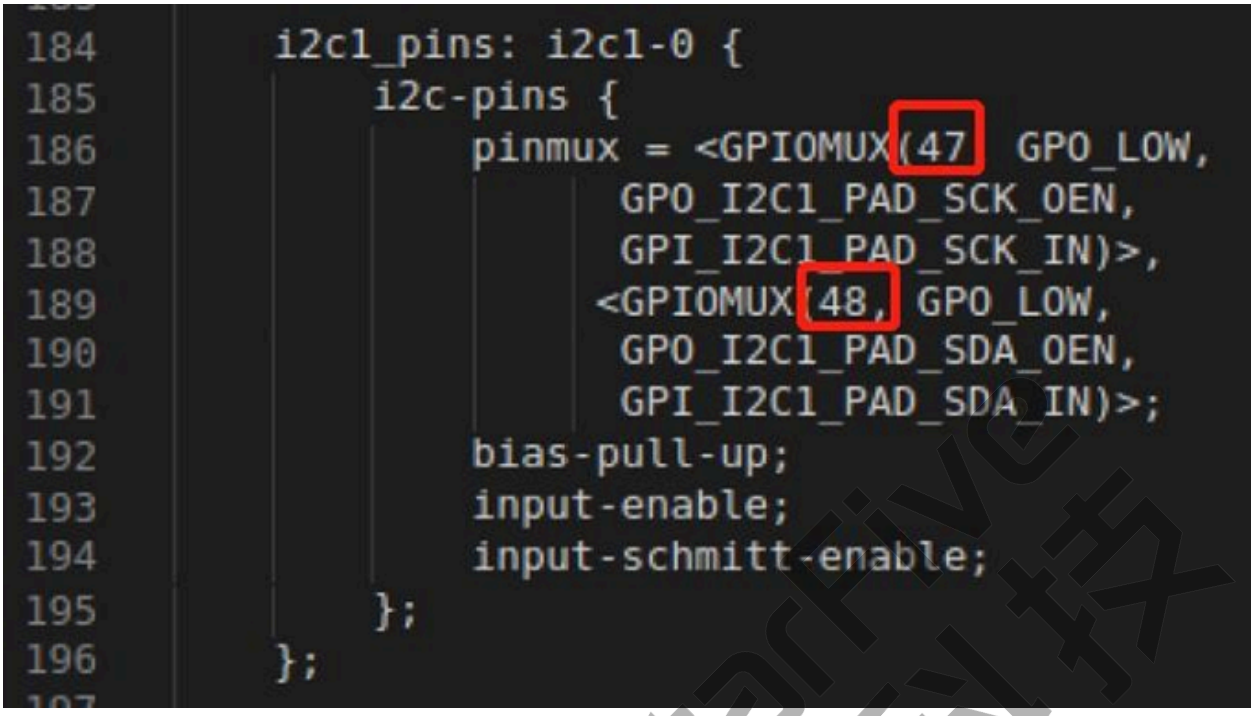
Figure 5-1 Connect the Sense Hat (B) to the Header



5.1.2. Configuring dts File

Modify the file content of jh7100-common.dtsi 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



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:

- GPIO48 (I2C SDA)
- GPIO47 (I2C SCL)

5.2. Debugging I2C GPIO

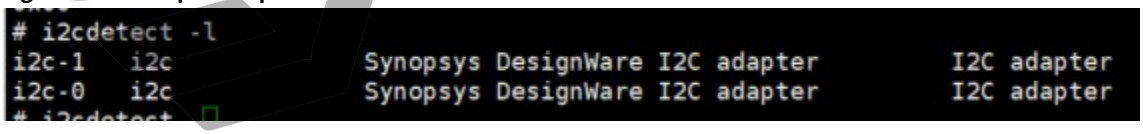
Perform the following steps to debug I2C:

1. Execute the following command to scan the bus:

   ```
   i2cdetect -l
   ```

   Result:

   Figure 5-3 Example Output

2. Execute the following command to detect the device:

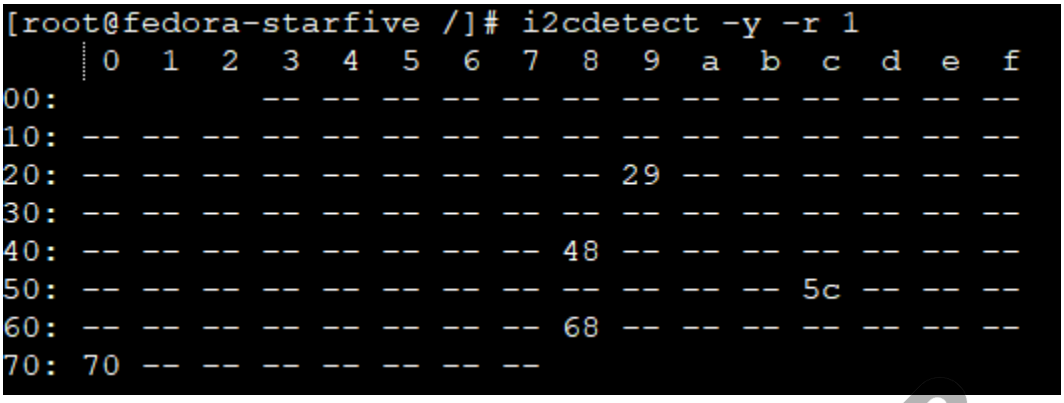
   ```
   i2cdetect -y -r 1
   ```

   Tip:

   1 is the I2C bus number.

   Result:

Figure 5-4 Example Output



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

3. Execute the following command to read register content:

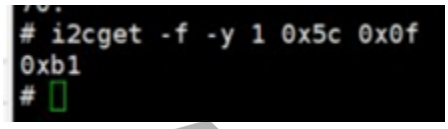
```
i2cget -f -y 1 0x5c 0x0f
```

Tip:

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

Result:

Figure 5-5 Example Output



The register content is 0xb1 in this output.

4. Execute the following command to write register data:

```
i2cset -y 1 0x5c 0x11 0x10
```

Tip:

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

5. Execute the following to read all register values:

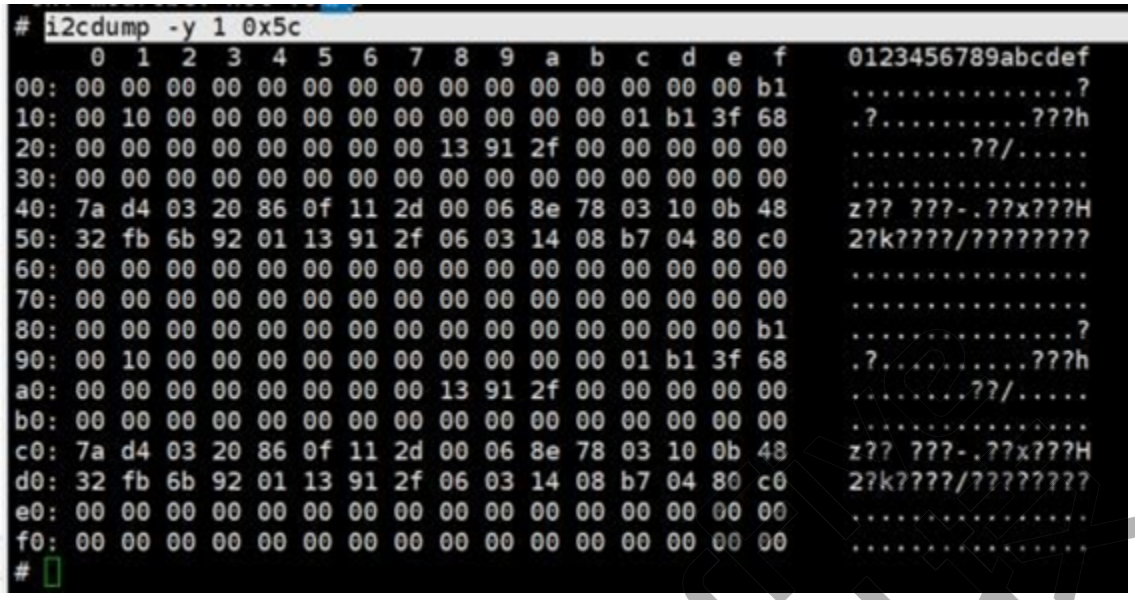
```
i2cdump -y 1 0x5c
```

Tip:

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

Result:

Figure 5-6 Example Output



6. SPI Operations

This chapter describes how to configure and debug SPI GPIO.

6.1. Configuring SPI GPIO

The configuration file, `jh7100-common.dtsi`, is under `/linux/arch/riscv/boot/dts/starfive`.

2 channels of SPI bus are supported: spi2 and spi3.

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 `jh7100-common.dtsi`:

Figure 6-1 Modify Pins

```
284     spi2_pins: spi2-0 {
285         mosi-pin {
286             pinmux = <GPIOMUX(18, GPO_SPI2_PAD_TXD,
287                   GPO_ENABLE, GPI_NONE)>;
288             bias-disable;
289             input-disable;
290             input-schmitt-disable;
291         };
292         miso-pin {
293             pinmux = <GPIOMUX(16, GPO_LOW, GPO_DISABLE,
294                   GPI_SPI2_PAD_RXD)>;
295             bias-pull-up;
296             input-enable;
297             input-schmitt-enable;
298         };
299         sck-pin {
300             pinmux = <GPIOMUX(12, GPO_SPI2_PAD_SCK_OUT,
301                   GPO_ENABLE, GPI_NONE)>;
302             bias-disable;
303             input-disable;
304             input-schmitt-disable;
305         };
306         ss-pins {
307             pinmux = <GPIOMUX(15, GPO_SPI2_PAD_SS_0_N,
308                   GPO_ENABLE, GPI_NONE)>,
309                  <GPIOMUX(11, GPO_SPI2_PAD_SS_1_N,
310                   GPO_ENABLE, GPI_NONE)>;
311             bias-disable;
312             input-disable;
313             input-schmitt-disable;
314         };
315     };
```

6.2. Debugging SPI GPIO

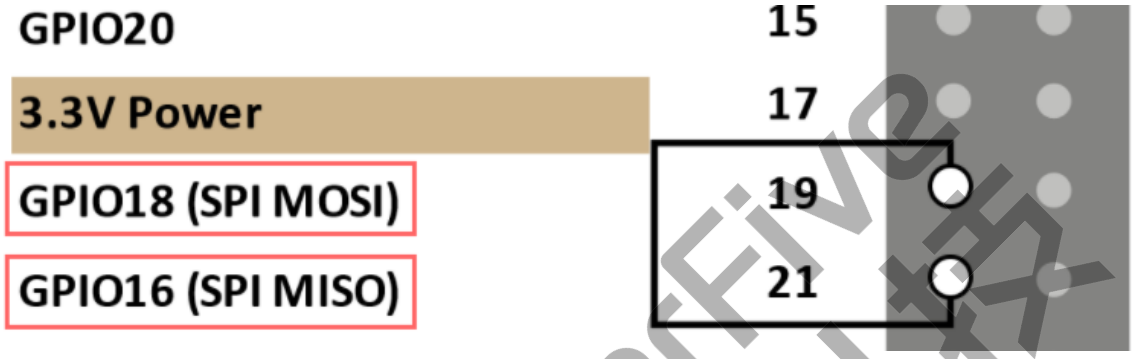
This section provides steps for loopback test and testing SPI with the ADXL345 module.

6.2.1. Loopback Test

The following steps are provided for the loopback test:

1. Wiring: Connect GPIO18 with GPIO16 as the following:

   Figure 6-2 Connect Pin 18 with 16

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 the board, for example, Starlight, 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@fedora-starfive /]# ls /dev/spidev*
   /dev/spidev0.0
   [root@fedora-starfive /]#
   ```

   In this output, `spidev0.0` is the device name.

6. Execute the following command to perform the test:

   ```
   ./spidev_test -D /dev/spidev0.0 -v -p string_to_send
   ```

   Tip:

   `spidev0.0` is the device name got from the previous step.

   Result:

Figure 6-4 Example Output

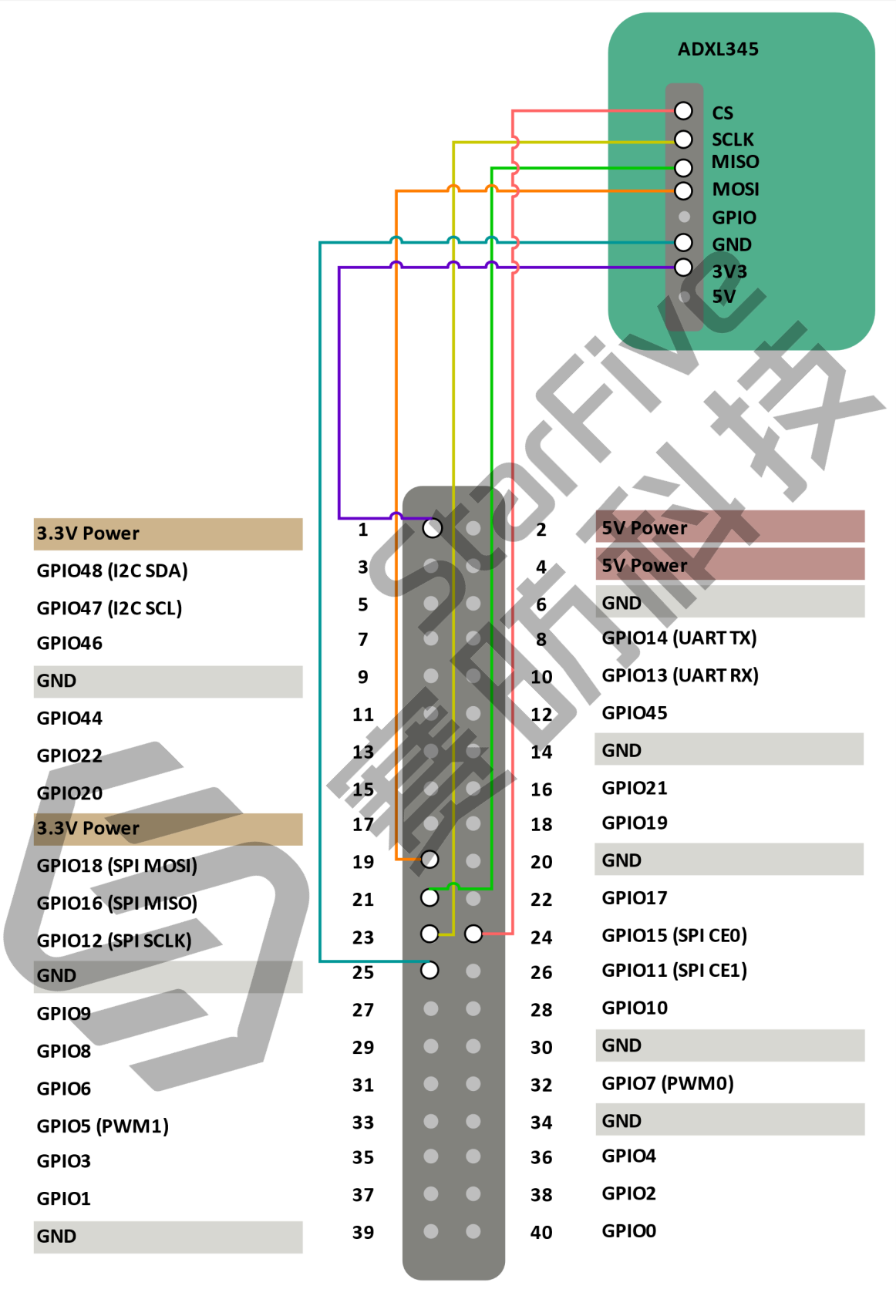
In this figure, the highlighted part indicates the test is successful.

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.

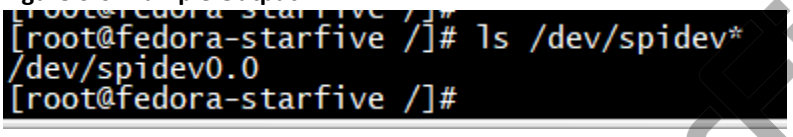
4. Upload `spidev_test` to the board, for example, StarLight, and change the execution permission by executing the following:

```
chmod +x spidev_test
```

5. Confirm the SPI device.

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

Figure 6-6 Example Output



In this output, `spidev0.0` is the device name.

6. Execute the following to read the device ID:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x80\\x00
```

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

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\xec\\x00\\x00\\x00\\x00\\x00\\x00
```

8. Execute the following to read:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x9e\\x00
```

9. Execute the following to write:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x1e\\xaa
```

10. Execute the following to read the verification:

```
./spidev_test -H -O -D /dev/spidev0.0 -v -p \\x9e\\x00
```

7. PWM Operations

This chapter describes how to configure and debug PWM GPIO:

7.1. Configuring PWM GPIO

The configuration file, `jh7100-common.dtsi`, is located 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

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	GPIO7
PWM1	GPIO5
PWM2	GPIO45

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.

8. UART Operations

This chapter describes how to configure and debug UART GPIO:

8.1. Configuring UART GPIO

The configuration file, `jh7100-common.dtsi`, is located under:`/linux/arch/riscv/boot/dts/starfive`.

4 channels of UART are supported at the most:

- Uart3 is for Debug.
- Uart0 is for Bluetooth.
- Uart1 and Uart2 can be used.

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

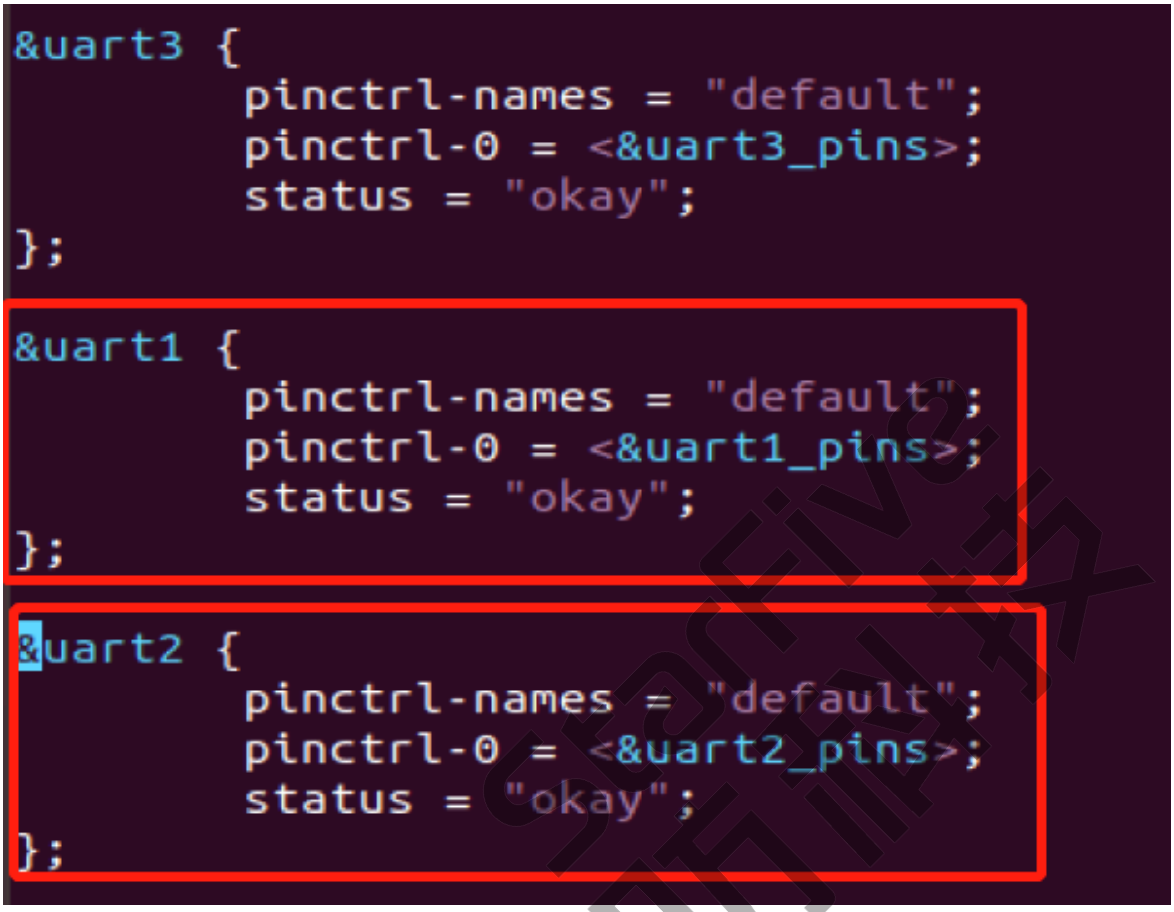
```
/ {
        aliases {
                mshc0 = &sdio0;
                mshc1 = &sdio1;
                serial0 = &uart3;
                serial1 = &uart0;
                serial2 = &uart1;
                serial3 = &uart2;
        };

        chosen {
                stdout-path = "serial0:115200n8";
        };

        cpus {
                timebase-frequency = <6250000>;
        };
```

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

Figure 8-2 Example Configuration

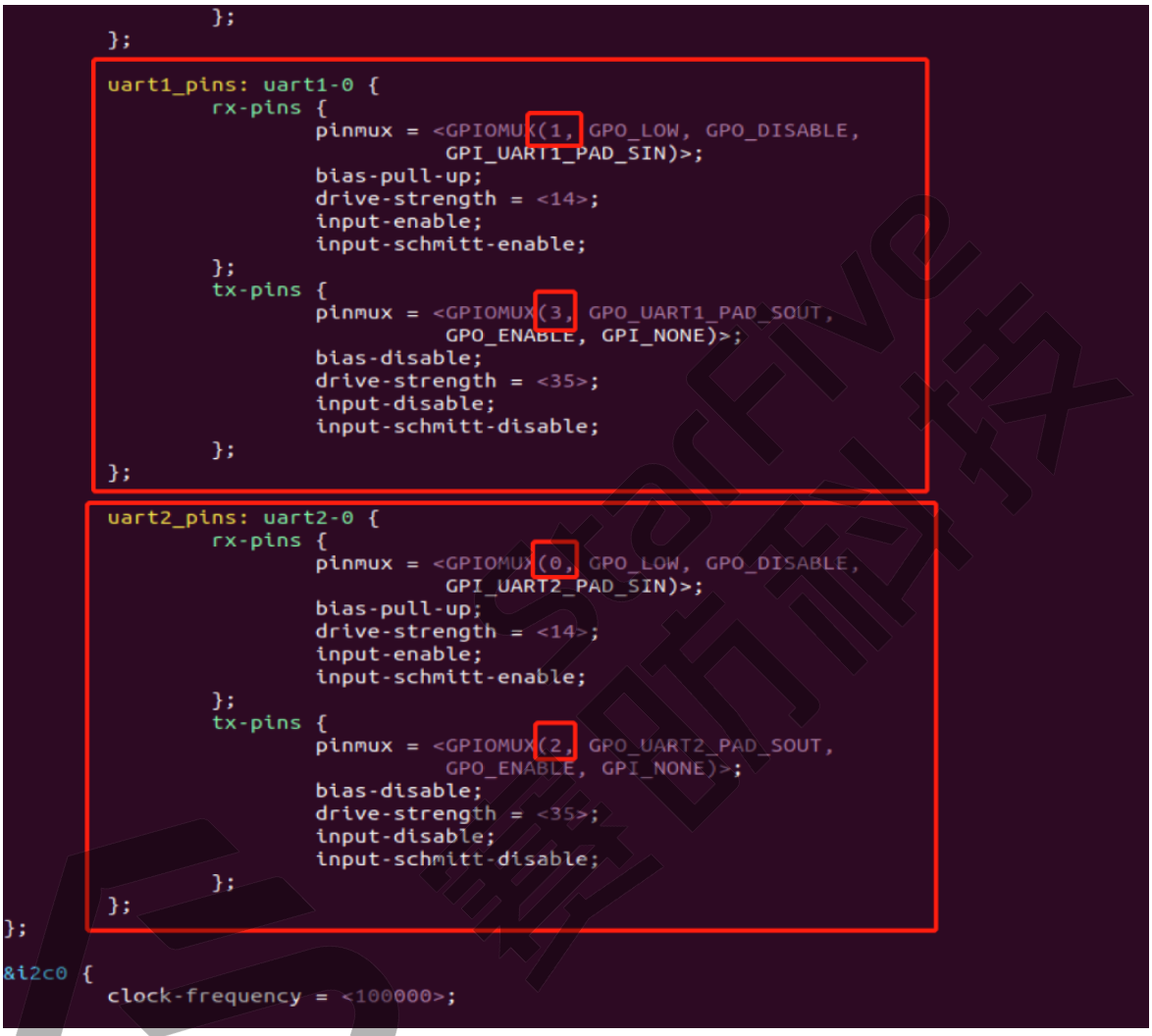


3. Add uart1_pins or uart2_pins node on the &gpio node:

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



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/ttyS2
UART2	/dev/ttyS3

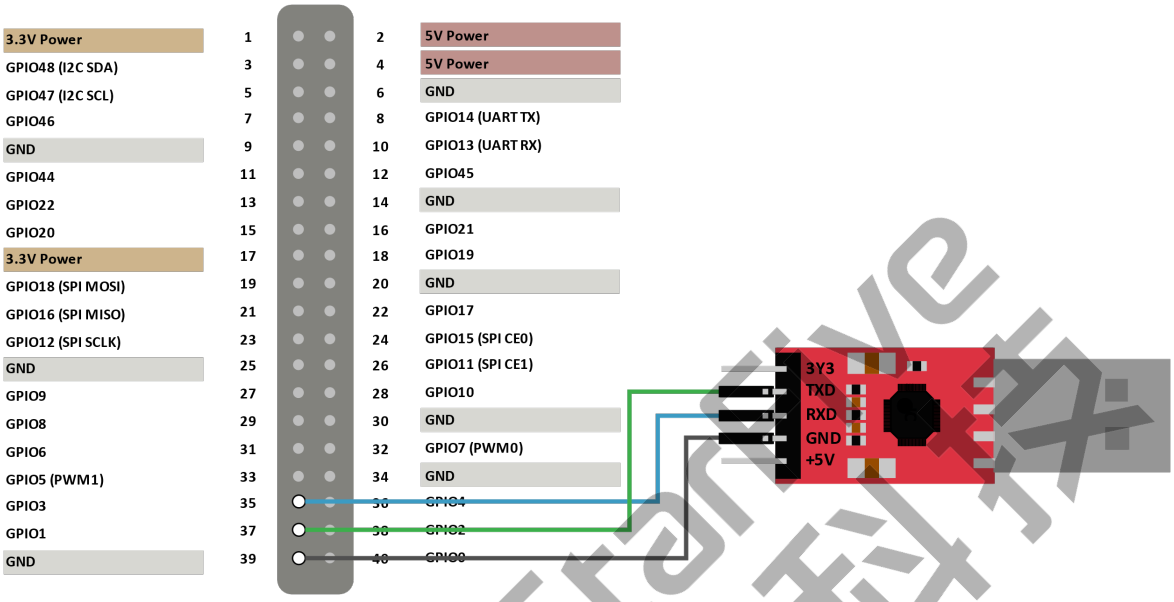
8.2. Debugging UART GPIO

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 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).

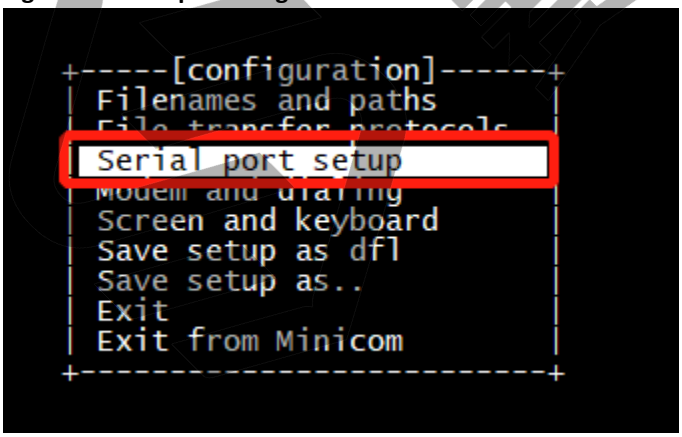
8.2.2. Debugging UART Send and Receive Functions

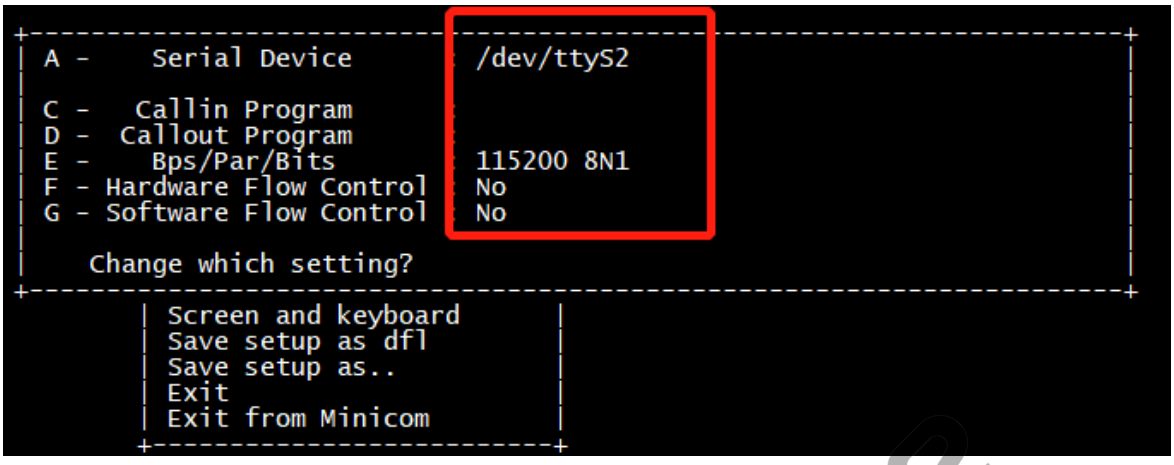
1. Configure Visionfive minicom:

```
sudo minicom -s
```

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

Figure 8-5 Example Configuration

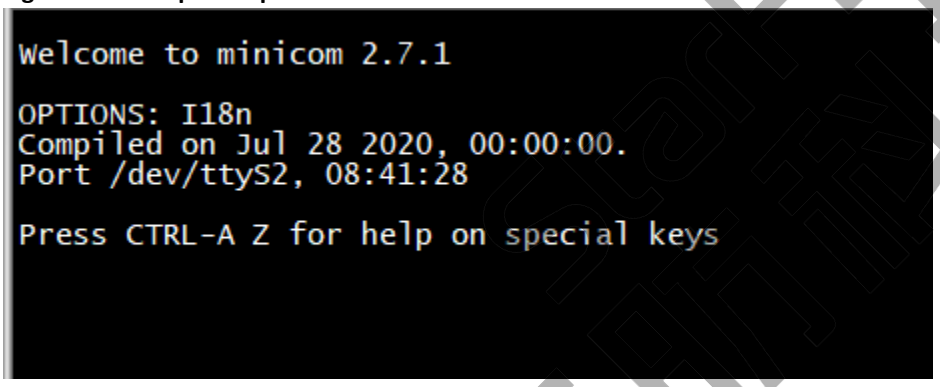




3. Start VisionFive minicom by typing the following command on the PC:

```
minicom -o -D /dev/ttyS2
```

Figure 8-7 Example Output

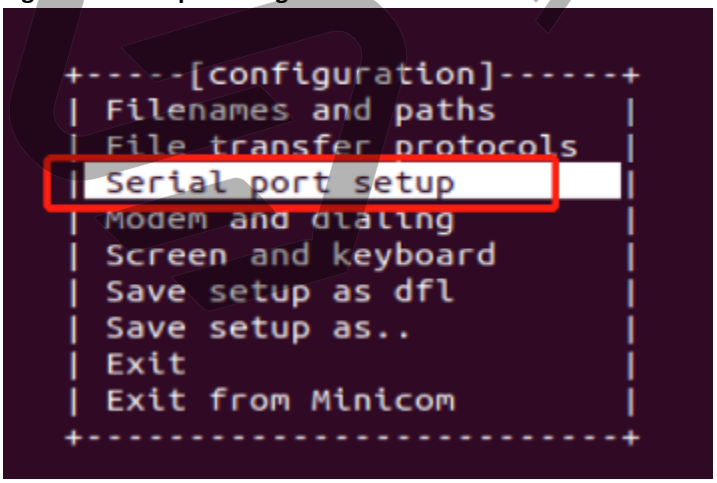


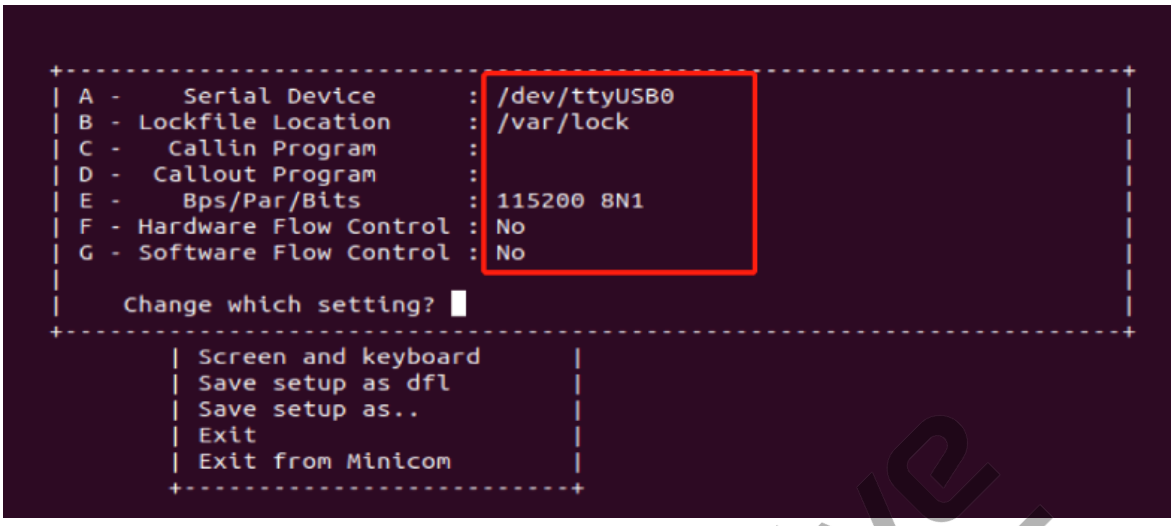
4. Configure Ubuntu minicom by typing the following:

```
sudo minicom -s
```

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

Figure 8-8 Example Configuration





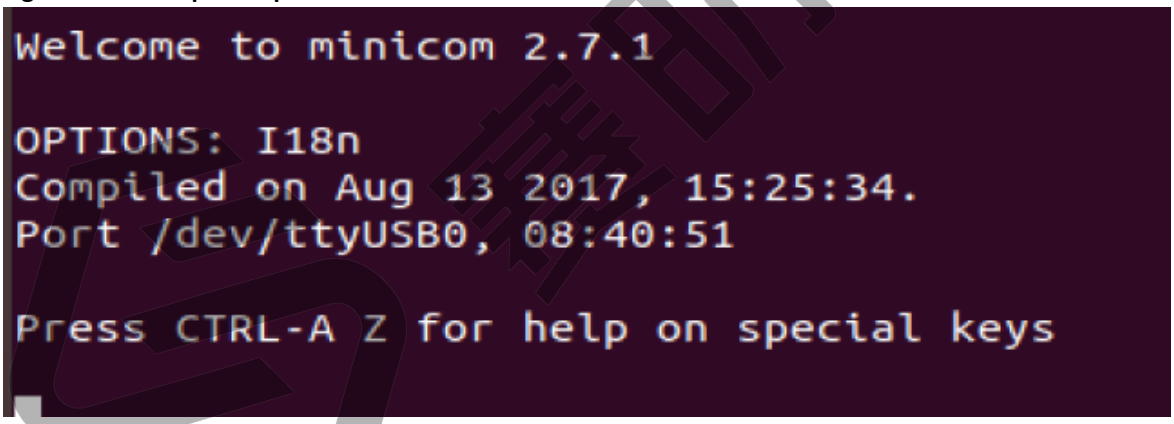
Note:

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

Figure 8-10 Example Command and Output

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

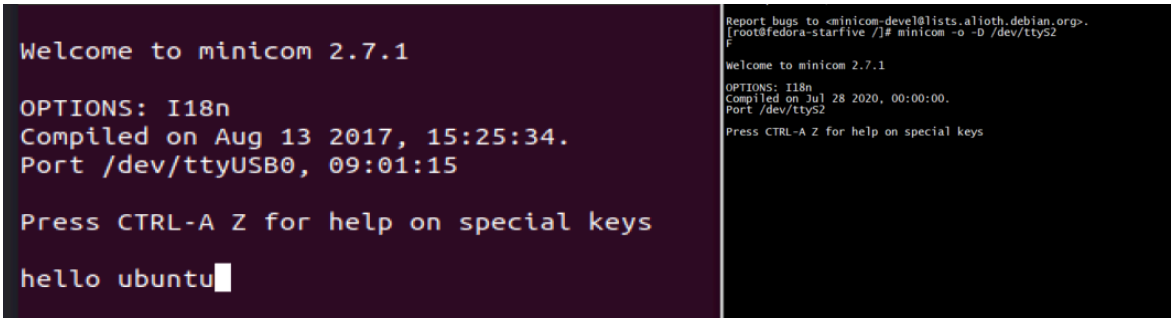
Figure 8-11 Example Output



Test UART Send:

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

Figure 8-12 Test UART Send

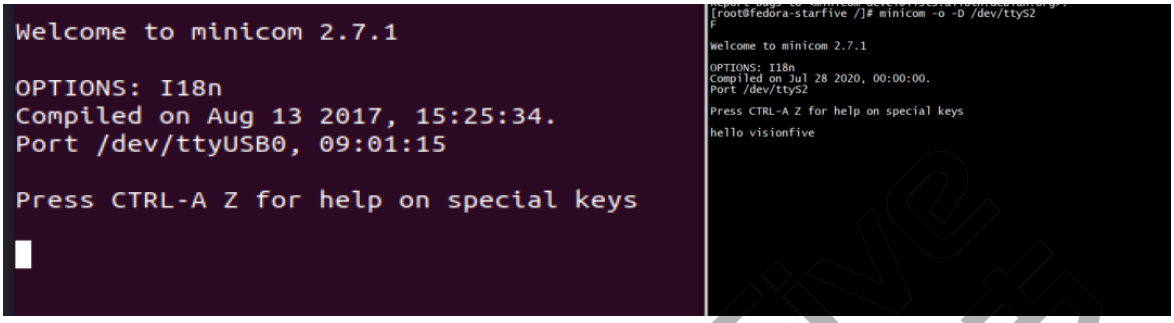


- Figure on the Left: Ubuntu minicom interface
- Figure on the Right: VisionFive minicom interface

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 minicom:

Figure 8-13 Test UART Receive:



- Figure on the Left: Ubuntu minicom interface
- Figure on the Right: VisionFive 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).

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

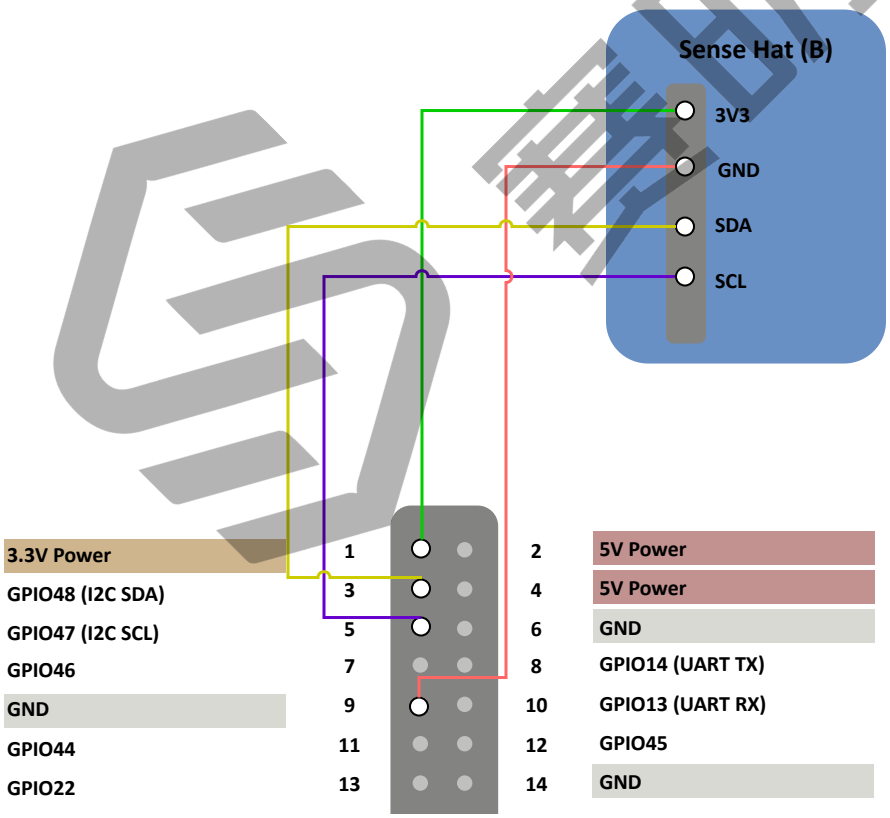
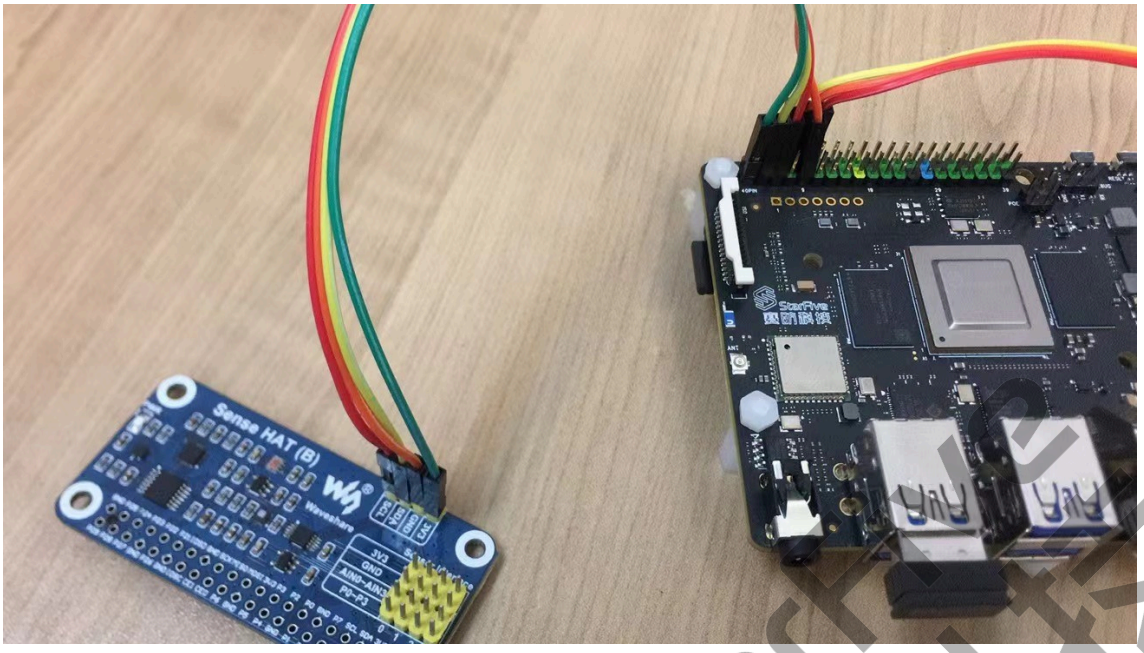


Figure 9-2 Connect Sense Hat (B) to the 40-Pin Header



9.1.2. Examples

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 codes 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@fedora-starfive riscv]# ./shtc3
   SHTC3 Sensor Test Program ...
   Fopen : /dev/i2c-1
   Temperature = 75.61°C , Humidity = 68.55
   Temperature = 27.40°C , Humidity = 68.54
   Temperature = 27.40°C , Humidity = 68.55
   ```

```
Temperature = 27.40°C , Humidity = 68.54
Temperature = 27.39°C , Humidity = 68.54
```

9.2. 2inch LCD Module Example

A 2inch LCD Module is used in this example. For the detailed specifications, refer to the following: https://www.waveshare.com/wiki/2inch_LCD_Module.

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

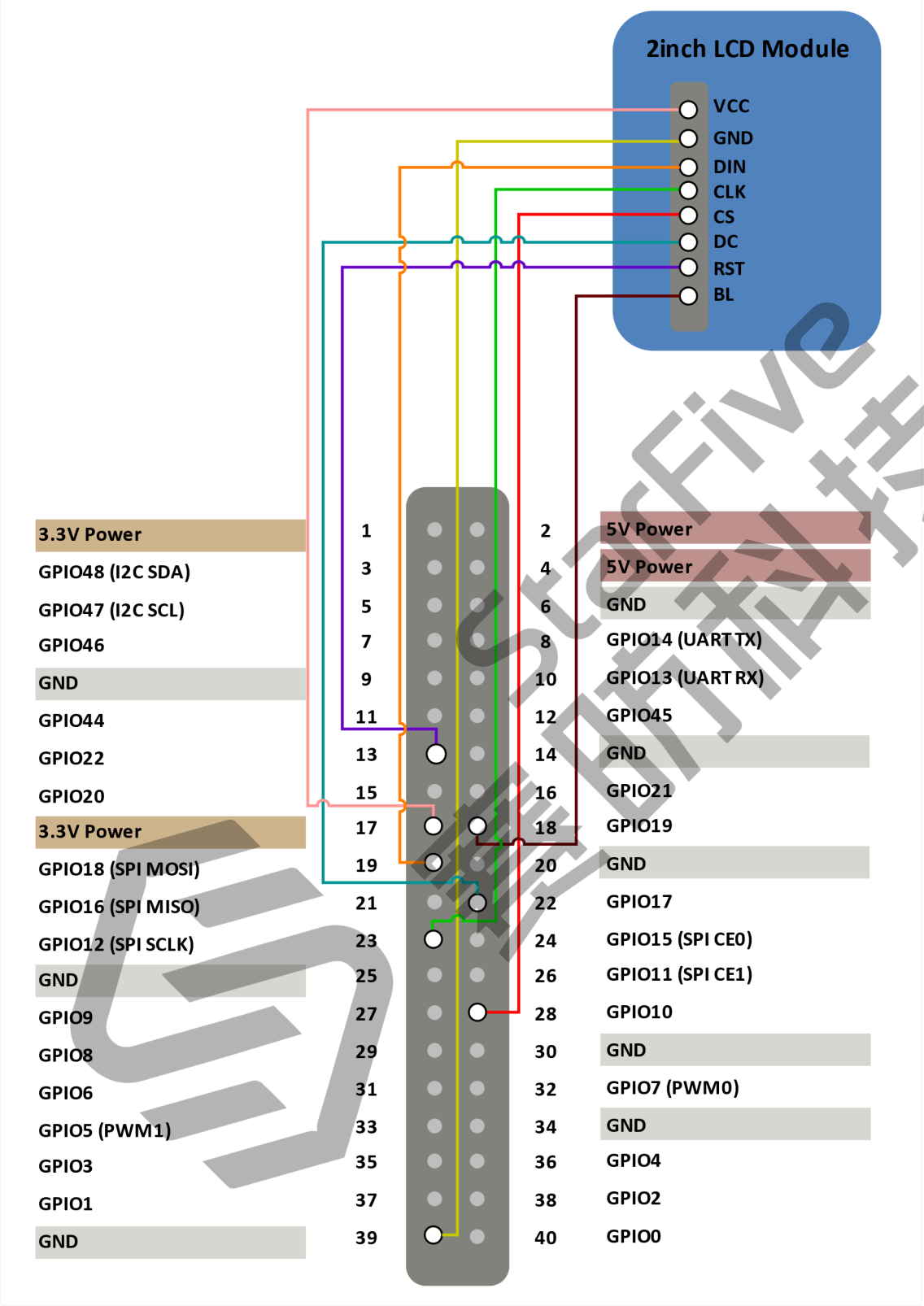
9.2.1. Hardware Setup

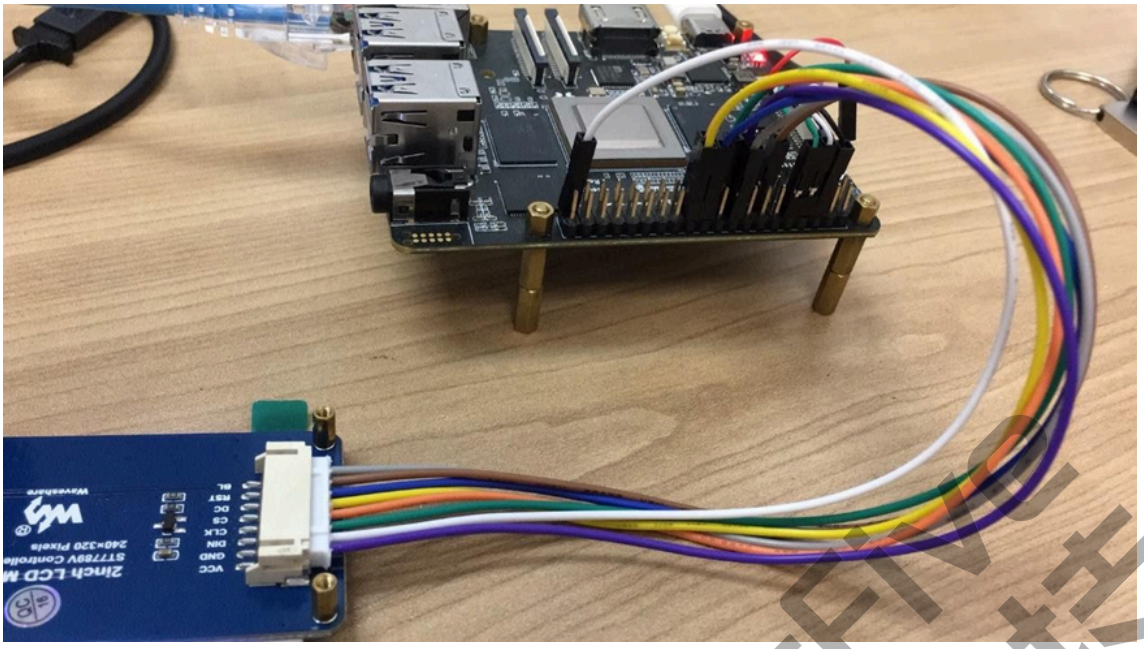
The following table and figure describe how to connect the 2inch LCD module with the 40-pin header:

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

2inch LCD Module	Pin Number
VCC	17
GND	39
DIN	19
CLK	23
CS	28
DC	22
RST	13
BL	18

Figure 9-3 Connect 2inch LCD with 40-Pin Header





9.2.1.1. Executing Example

Perform the following steps to execute the example:

1. Download the source code from visionfive.tar.gz.
2. Execute the following command to copy the code to the board. For example, VisionFive.

```
tar -xvf visionfive.tar.gz
cd visionfive/
./main 2
```

Result:

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

Figure 9-5 Example Output