StarFive
赛昉科技

Testing VisionFive GPIO

Application Note

For C

Version: 1.1

Date: 2021/12/27

Doc ID: VisionFive-ANEN-001-1.1

Legal Statements

Important legal notice before reading our documentation.

PROPRIETARY NOTICE

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 application note provides two methods to test VisionFive GPIO:

Revision History

Table 0-1 Revision History

Version	Released	Revision
V1.0	2021-12-15	Preliminary release.
V1.1	2021-12-27	In the *Running Demo Codes* section: • Added description for the `app` directory. • Added description for the `rsync` command. • Added description for `<User_Name>`. • Fixed a typo.

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. Introduction

This application note provides two methods to test VisionFive GPIO:

- Test with command lines.
- Test with demo code.

2. Preparation

Before executing the demo program, make sure you prepare the following:

2.1. Preparing Hardware

Prepare the following hardware items before running the demo code:

Table 2-1 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	M	An oscilloscope	The oscilloscope is used to verify the GPIO voltage.

Note:
*: M: Mandatory, O: Optional

2.2. Preparing Software

- Software Environment:
  - PC: Ubuntu 20.04
  - RISC-V Platform: Linux 5.16.0
- Flash Fedora OS into a Micro-SD card and compile and replace dtb files as described in the *Preparing Software* section in *StarFive 40-Pin GPIO Header User Guide*.

3. Testing GPIO with Command Lines

Test the GPIO0 as described in the *Configuring GPIO* section in the *StarFive 40-Pin GPIO Header User Guide*.

4. Running Demo Codes

To run the demo code, perform the following:

4.1. Compiling the Source Code

To compile the source code, perform the following:

1. Save the following source code for C language as `test-gpio.c` to your desired directory under Ubuntu:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/stat.h>


#define GPIO44  492
#define GPIO22  470

#define GPIO_KEY1     GPIO44
#define FISCV_GPIO    GPIO22
#define MAX_BUF       128                    //Define array size
#define StarF_Gpio_Dir "/sys/class/gpio"    //GPIO control paths
/**********************************************************************
* Function Name: StarF_gpio_export
* Description:   Set the pin number
* return value:  0 Success; Others: fail
*     Data        version     Author    Application Name
*
 ----------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu    test gpio
 **********************************************************************
*/
int StarF_gpio_export(unsigned int gpio)
{
    int fd, len;
    char buf[MAX_BUF];
 // /sys/class/gpio/export
    fd = open( "/sys/class/gpio/export", O_WRONLY);
    if (fd < 0) {
        perror("gpio/export");
        return fd;
    }
    len = snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, len);
    close(fd);

    return 0;
}

/***********************************************************************
* Function Name: StarF_gpio_unexport
* Description:   Cancel the GPIO pin number
* return value:  0 Success; Others: fail
*     Data         version     Author    Application Name
*
 ----------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu    test gpio
 **********************************************************************
*/
int StarF_gpio_unexport(unsigned int gpio)
{
    int fd, len;
    char buf[MAX_BUF];
 // /sys/class/gpio/unexport
    fd = open("/sys/class/gpio/unexport", O_WRONLY);
    if (fd < 0) {
        perror("gpio/export");
        return fd;
    }

    len = snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, len);
    close(fd);
    return 0;
}

/***********************************************************************
* Function Name: StarF_gpio_set_dir
* Description:   Set GPIO pin I/O
* return value:  0 Success; Others: fail
*     Data         version     Author    Application Name
*
 ----------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu    test gpio
 **********************************************************************
*/

int StarF_gpio_set_dir(unsigned int gpio, unsigned int out_flag)
{
    int fd, len;
    char buf[MAX_BUF];
 // /sys/class/gpio/gpioN/direction
    len = snprintf(buf, sizeof(buf),
 StarF_Gpio_Dir "/gpio%d/direction", gpio);

    fd = open(buf, O_WRONLY);
    if (fd < 0) {
        perror(buf);
        return fd;
    }

    if (out_flag)                       //'1' set to output
        write(fd, "out", 4);
    else                                //'0' set input
        write(fd, "in", 3);

    close(fd);
    return 0;
}
/***********************************************************************
* Function Name: StarF_gpio_set_dir
* Description:   Set GPIO high & low levels
* function parameters:@GPIO   Set GPIO number for the output level
                       value   1: Set gpio output to high level;   0: Set
 gpio output to low level.
* return value:  0 Success; Others: fail
*    Data        version     Author   Application Name
*
 ----------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu    test gpio
 **********************************************************************
*/

int StarF_gpio_set_value(unsigned int gpio, unsigned int value)
{
    int fd, len;
    char buf[MAX_BUF];
 // /sys/class/gpio/gpioN/value
    len = snprintf(buf, sizeof(buf), StarF_Gpio_Dir "/gpio%d/value",
 gpio);
    fd = open(buf, O_WRONLY);
    if (fd < 0) {
        perror(buf);
        return fd;
    }

    if (value)                           //'1' output is high level
        write(fd, "1", 2);
    else                                 //'0' output is Low level
        write(fd, "0", 2);
    close(fd);
    return 0;
}

/*********************************************************************
* Function Name: StarF_gpio_get_value
* Description:   Read GPIO high & low levels
* function parameters:@GPIO   Set GPIO number for the output level
                      value   1: Set gpio output to high level;   0: Set
 gpio output to low level.
* return value:  0 Success; Others: fail
*     Data         version     Author    Application Name
*
 ----------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu    test gpio
 **********************************************************************
*/

int StarF_gpio_get_value(unsigned int gpio, unsigned int *value)
{
    int fd, len;
    char buf[MAX_BUF];
    char ch;
 // /sys/class/gpio/gpioN/value
    len = snprintf(buf, sizeof(buf), StarF_Gpio_Dir "/gpio%d/value",
 gpio);

    fd = open(buf, O_RDONLY);
    if (fd < 0) {
        perror("gpio/get-value");
        return fd;
    }

    read(fd, &ch, 1);                //Read the external input level

    if (ch != '0') {                 //'1' Input is high level
        *value = 1;
    } else {                         //'0' Input is Low level
        *value = 0;
    }

    close(fd);
    return 0;
}

/*********************************************************************
* FunctionName: main
* Description:
* function parameters:
* return value:  0 Success; Others: fail
*    Data        version     Author   Application Name
*
 ----------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu    test gpio
 **********************************************************************
*/

int main(int argc, char **argv) {
    unsigned int i;
    unsigned int value1,value2;

    printf("\t********************************************\n");
    printf("\t********  StarF_GPIO_TEST_DEMO  ***********\n");
    printf("\t********  Version date: 2021/12 ***********\n");
    printf("\t********************************************\n");
    printf("Gpio begin to init\r\n");
    StarF_gpio_export(FISCV_GPIO);                    //export gpio
 Gpio

    StarF_gpio_set_dir(FISCV_GPIO, 1);                //set as output
    printf("Gpio init ok\r\n");


    /* Confirm INIT_B Pin as High */
 while(1)
 {
  StarF_gpio_set_value(FISCV_GPIO, 1);       //output high
  printf("Gpio off\r\n");
  usleep(500000);                            //delay
  StarF_gpio_set_value(FISCV_GPIO, 0);       //output low
  printf("Gpio on\r\n");
  usleep(500000);                            //delay
    }

    StarF_gpio_unexport(FISCV_GPIO);              //unexport gpio Gpio

    return 0;
}
```

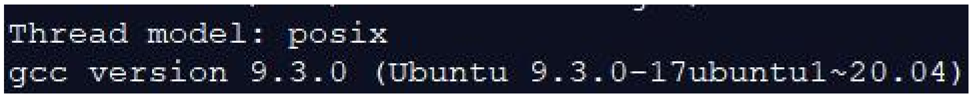
2. (Optional) Install the tool to compile. The following is an example to install:

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

Information:

- This step can be skipped if the tool has been installed.
- After successful installation, check the version by running: `linus@starfive$ riscv64-linux-gnu-gcc -v`. The following is the example output:

Figure 4-1 Example Output



```
Thread model: posix
gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)
```

Result: The executable `test-gpio` file is generated in the current directory.

3. Compile the source code by executing the following:

```
riscv64-linux-gnu-gcc -o test-gpio test-gpio.c
```

4. Execute the following to see if the compilation is successful:

```
file test-gpio
```

Result: `UCB RISC-V` in the following output indicates the compilation is successful:

```
Riscv@starfive:~/work/app$ file test-gpio
test-gpio: ELF 64-bit LSB executable, UCB
 RISC-V, version 1 (SYSV), dynamically linked,
 interpreter /lib/ld-linux-riscv64-lp64d.so.1, for GNU/Linux 4.15.0,
 BuildID[sha1]=476d5a99c84f995d03227a18285222ac25e2cd0d, not stripped
c-v2x@starfive:~/work/app$
```

4.2. Testing GPIO with Demo Codes

1. Power on the VisionFive, and check the GPIO22 voltage changes.
2. Execute the following command in Ubuntu to upload the executable file `test-gpio` to your desired directory of the board, for example, `test`:

```
rsync ./test-gpio <User_Name>@<Board_IP_Address>:/home/riscv/test
```

Information:

- `<User_Name>`: Your user name of the board. For example, riscv.
- `<Board_IP_Address>`: The board IP address. For example, 192.168.92.133.

Example:

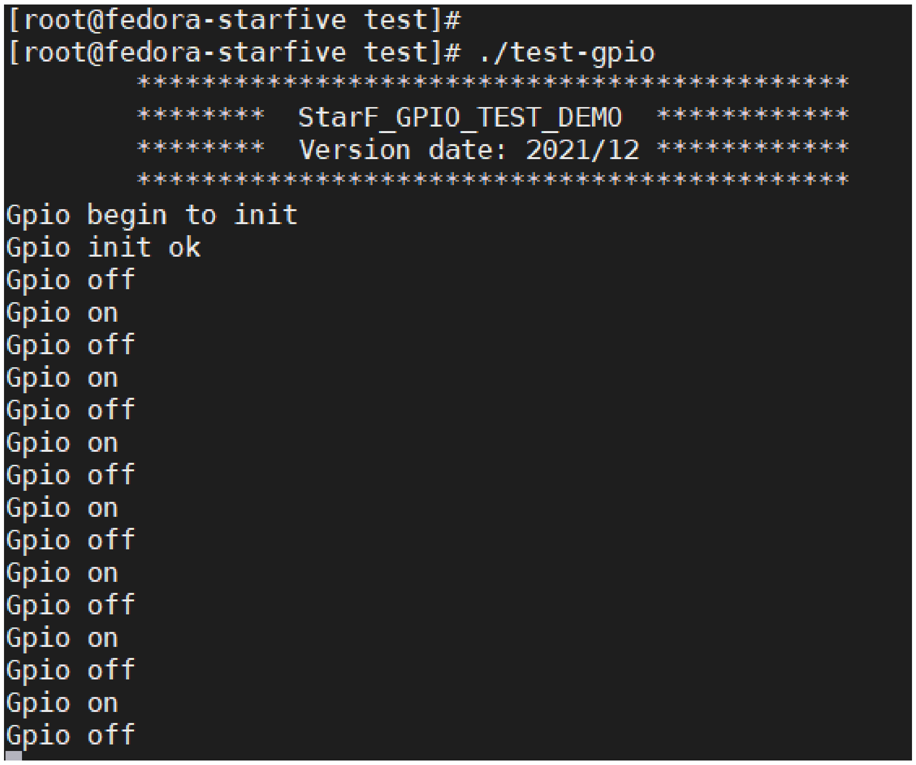
```
rsync ./test-gpio riscv@192.168.92.133:/home/riscv/test
```

3. Execute the following on VisionFive to run the demo code:

```
./test-gpio
```

The following is an example output:

Figure 4-2 Example Output



```
[root@fedora-starfive test]#
[root@fedora-starfive test]# ./test-gpio
        ********************************************
        ********  StarF_GPIO_TEST_DEMO  ************
        ********  Version date: 2021/12 ************
        ********************************************
Gpio begin to init
Gpio init ok
Gpio off
Gpio on
Gpio off
Gpio on
Gpio off
Gpio on
Gpio off
Gpio on
Gpio off
Gpio on
Gpio off
Gpio on
Gpio off
Gpio on
Gpio off
```

Tip:

- `Gpio on`: High voltage
- `Gpio off`: Low voltage