StarFive 赛昉科技

Using VisionFive UART to Read GPS Data

with C

Application Note

Version: 1.1

Date: 2021/12/29

Doc ID: VisionFive-ANEN-003-1.1

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 application note provides steps to use VisionFive's UART to read GPS data through an example program with C.

Revision History

Table 0-1 Revision History

Version	Released	Revision
V1.0	2021-12-15	Preliminary release.
V1.1	2021-12-29	• Updated the Makefile content format. • Updated the Makefile description. • Added description for the `rsync` command. • Added description for `<User_Name>`.

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 steps to use VisionFive's UART to read GPS data through an example program with C.

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.
UART Demo	M	• 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.

Note:
*: M: Mandatory, O: Optional

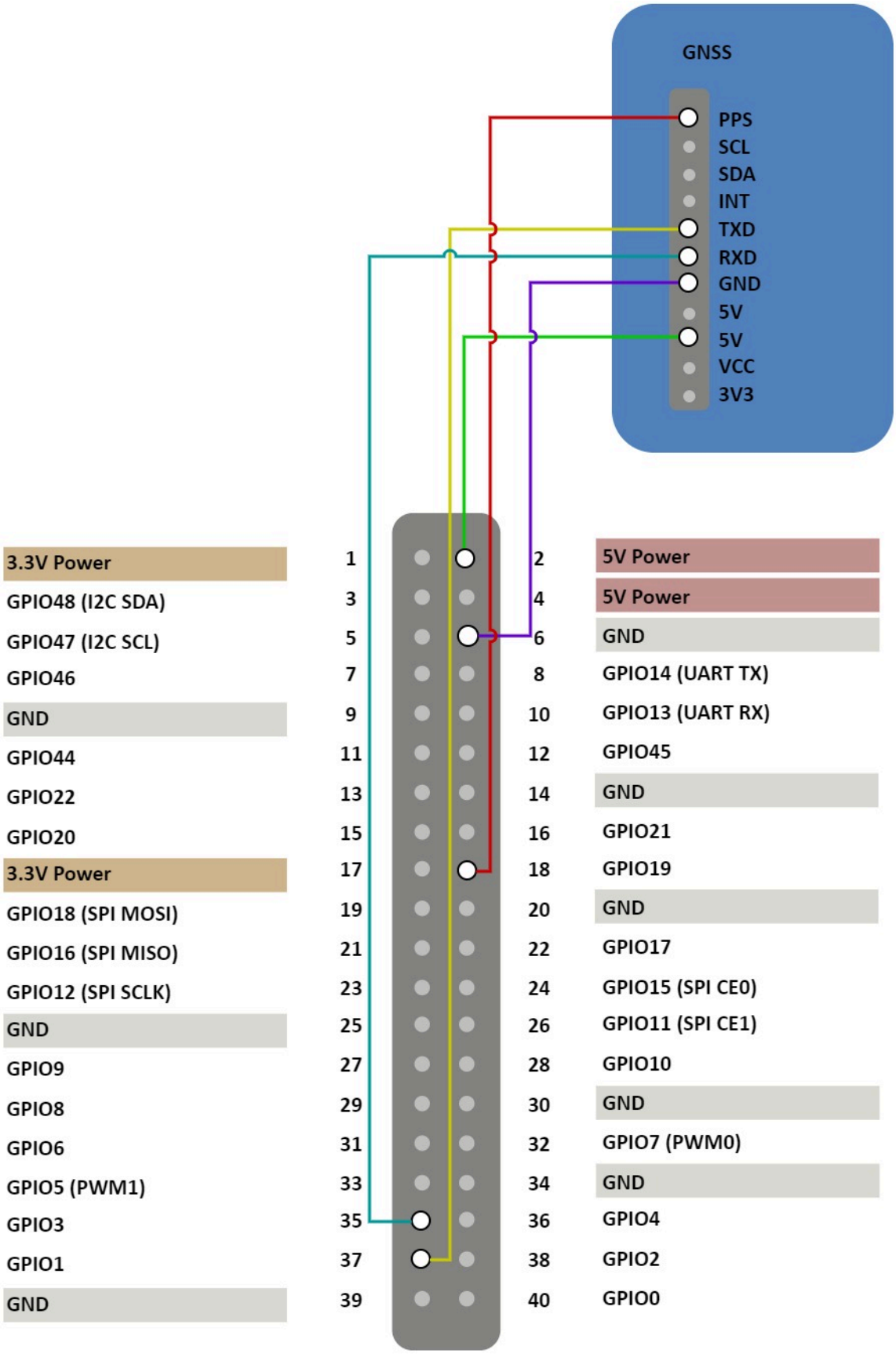
2.1.1. Hardware Setup

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

Table 2-2 Connect GNSS HAT to the 40-Pin Header

GNSS HAT	Pin Number
PPS	18
TXD	37
RXD	35
GND	6
5V	2

Figure 2-1 Connect GNSS HAT to the 40-Pin Header



2.2. Preparing Software

Software Environment

- PC: Ubuntu 20.04
- RISC-V Platform: Linux 5.16.0

Preparing Toolchain

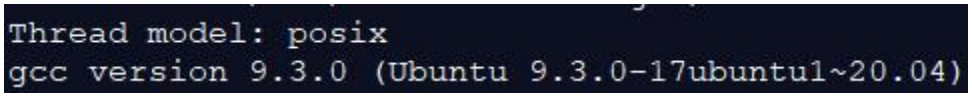
Install the tool to compile. The following is an example to install:

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

Tip:

- 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 2-2 Example Output



Preparing UART GPIO

Make sure the following procedures are performed:

1. 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*.
2. Configure the GPIO pin as UART by setting the dts file as described in the *Configuring UART GPIO* section in the *StarFive 40-Pin GPIO Header User Guide*.

Note:

You can configure the unoccupied pins as UART. The following is an example table for the mapping:

Table 2-3 UART and Pin Name Mapping

UART	GPIO (Pin Name)
Uart1	- GPIO3 - GPIO1
Uart2	- GPIO2

UART	GPIO (Pin Name)
	- GPIO0

3. Execute the following:

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

3. Running Demo Codes

To run the demo code, perform the following:

1. Create `test-gps` file under `app` directory to save test file.
2. Save the following source code as `test-gps.c` under `test-gps` directory:

```
/*************************************************************************

 /
*************************************************************************
* Function Name:
* Description:   This is a test about GPS Receiver
* return value:
*    Data        version     Author      Application Name
*
 ------------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu      test gpio
 ************************************************************************
*/

 // This is a test about GPS receiver
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include <fcntl.h>      // open() close()
#include <unistd.h>     // read() write()

#include <termios.h>    // Set baud rate
#include <getopt.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <sys/types.h>

#define FALSE    0
#define TRUE     1
#define WORDLEN  32

struct serial_config
{
        unsigned char serial_dev[WORDLEN];
        unsigned int  serial_speed;
        unsigned char databits;
        unsigned char stopbits;
        unsigned char parity;
};
```

```
struct serial_config serial;

int name_arr[] =
 {230400, 115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 300,};


#define FUNC_RUN        0
#define FUNC_NOT_RUN    1
#define ORG_GPS      1
#define SEL_GPGGA     2
#define SEL_GPGLL     3
#define SEL_GPGSA     4
#define SEL_GPGSV     5
#define SEL_GPRMC     6
#define SEL_GPVTG     7
#define FUNC_QUIT     8

/*****************************************************************
* Function Name: StarF_Read_GPS_datas
* Description:   Read GPS data
* Return value:
*    Data        version     Author   Application Name
*
 ----------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu        Test GPS
 ****************************************************************
*/

int StarF_Read_GPS_datas(int fd, char *rcv_buf)
{
 int retval;
        fd_set rfds;
        struct timeval tv;

        int ret,pos;

        tv.tv_sec = 1;
        tv.tv_usec = 0;

        pos = 0;         // Point to receive buf

        while (1)
        {
                FD_ZERO(&rfds);
                FD_SET(fd, &rfds);

                retval = select(fd+1 , &rfds, NULL, NULL, &tv);

                if (retval == -1)
                {
```

```
                        perror("select()");
                        break;
                }
                else if (retval)
                {// Determine whether data exists
                        ret = read(fd, rcv_buf+pos, 2048);
                        pos += ret;
                        if (rcv_buf[pos-2] == '\r' && rcv_buf[pos-1]
 == '\n')
                        {
                                FD_ZERO(&rfds);
                                FD_SET(fd, &rfds);

                                retval = select(fd+1 , &rfds, NULL,
 NULL, &tv);

                                if (!retval) break;// If no data
 exists, break
                        }
                }
                else
                {
                        printf("No data\n");
                        break;
                }
        }

        return 1;
} // end read_GPS_datas

/*****************************************************************
* Function Name: StarF_Read_GPS_datas
* Description:   Show all receive signal
* return value:
*    Data        version     Author   Application Name
*
 ----------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu        Test GPS
 ****************************************************************
*/

void StarF_GPS_original_signal(int fd)
{
 char rcv_buf[2048];

 while (1)
 {
  bzero(rcv_buf,sizeof(rcv_buf));
  {
   if (StarF_Read_GPS_datas(fd,rcv_buf))
```

```
   {
    printf("%s",rcv_buf);
   }
  }
 }
} // end GPS_original_sign

/*********************************************************************
* Function Name: StarF_init_ttyS
* Description:   Init serial port
* return value:
*    Data        version     Author   Application Name
*
 ---------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu        Test GPS
 *********************************************************************
*/

void StarF_init_ttyS(int fd, int i)
{
 struct termios newtio;
 tcgetattr(fd, &newtio);

 bzero(&newtio, sizeof(newtio));

 switch (i)
 {
  case 230400 : newtio.c_cflag = (B230400 | CS8 | CLOCAL | CREAD);
    break;
  case 115200 : newtio.c_cflag = (B115200 | CS8 | CLOCAL | CREAD);
                                 break;
  case  57600 : newtio.c_cflag = (B57600 | CS8 | CLOCAL | CREAD);
                                 break;
  case  38400 : newtio.c_cflag = (B38400 | CS8 | CLOCAL | CREAD);
                                 break;
  case  19200 : newtio.c_cflag = (B19200 | CS8 | CLOCAL | CREAD);
                                 break;
  case   9600 : newtio.c_cflag = (B9600 | CS8 | CLOCAL | CREAD);
                                 break;
  case   4800 : newtio.c_cflag = (B4800 | CS8 | CLOCAL | CREAD);
                                 break;
  case   2400 : newtio.c_cflag = (B2400 | CS8 | CLOCAL | CREAD);
                                 break;
  case   1200 : newtio.c_cflag = (B1200 | CS8 | CLOCAL | CREAD);
                                 break;
  case    300 : newtio.c_cflag = (B300 | CS8 | CLOCAL | CREAD);
                                 break;
  default:
   break;
 }
```

```
 newtio.c_lflag &= ~(ECHO | ICANON);

 newtio.c_iflag = IGNPAR;
 newtio.c_oflag = 0;
 newtio.c_oflag &= ~(OPOST);

 newtio.c_cc[VTIME]    = 5;
 newtio.c_cc[VMIN]     = 0;
 tcflush(fd, TCIFLUSH);
 tcsetattr(fd,TCSANOW,&newtio);

}//end init_ttyS

/*****************************************************************
* Function Name: StarF_print_usage
* Description:
* return value:
*    Data        version     Author   Application Name
*
 ----------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu        Test GPS
 ****************************************************************
*/

void StarF_print_usage(FILE *stream, int exit_code)
{
    fprintf(stream,
            "\t-h  --help     Display this usage information.\n"
            "\t-d  --device   The device ttyS[0-3] or ttyEXT[0-3]\n"
            "\t-b  --baudrate Set the baud rate you can select\n"
         "\t    [230400, 115200, 57600, 38400, 19200, 9600, 4800, 2400,
 1200, 300]\n");
    exit(exit_code);
}

int StarF_set_speed(int speed)
{
 int i;
 int re_speed;
 for (i = 0; i < sizeof(name_arr)/sizeof(int); i++)
 {
  if(speed == name_arr[i])
  {
   re_speed = speed;
          break;
  }
 }

 if(i == 10)
```

```
 {
  printf("\tSorry, please set the correct baud rate!\n\n");
  StarF_print_usage(stderr, 1);
 }
 return re_speed;
}

 /
*********************************************************************
* Function Name: main
* Description:
* return value:
*    Data        version     Author   Application Name
*
 ---------------------------------------------------------------------
* 2021/12/08        V1.0    zheng.xu        Test GPS
 *********************************************************************
*/

int main(int argc, char *argv[])
{
 int fd;
 int opt;
 int speed_i;
 const char *dev = NULL;
 const char *short_options= "hd:b:";
 struct option long_options[] =
 {
  {"help", 0, NULL, 'h'},
  {"device", 1, NULL, 'd'},
  {"baudrate", 1, NULL, 'b'},
  {0, 0, 0, 0},
 };

 while ((opt = getopt_long(argc, argv, short_options, long_options,
 NULL)) != -1)
 {
  switch (opt)
  {
  case 'h':
   StarF_print_usage(stdout, 0);
  case 'd':
   dev = optarg;
   break;
  case 'b':
                        serial.serial_speed = atoi(optarg);
                        break;
                case '?':
                        StarF_print_usage(stderr, 1);
                default:
```

```
                        StarF_print_usage(stdout, 0);
 }
}

        if (dev == NULL)
        {
                printf("\tYou must enter to open the device node\n\n");
                StarF_print_usage(stderr, 1);
                exit (0);
        }

strcpy(serial.serial_dev, dev);

// Open serial port
        fd = open(serial.serial_dev, O_RDONLY);

if (fd == -1)
        {
                printf("open device %s error\n", serial.serial_dev);
        }

speed_i = StarF_set_speed(serial.serial_speed);

printf("\t\tctrl + c to exit!\n");
   StarF_init_ttyS(fd, speed_i);                // Init device
StarF_GPS_original_signal(fd);


        if (close(fd)!=0)
{
 printf("close device %s error", (serial.serial_dev));
}
return 0;

} // end main
```

3. Execute the following to create Makefile:

```
touch Makefile
```

4. Copy the following to the Makefile, save and exit:

```
EXEC  = test-gps
OBJS    = test-gps.o

#CROSS  = riscv64-unknown-elf-
CROSS   = riscv64-linux-gnu-
CC  = $(CROSS)gcc
STRIP  = $(CROSS)strip
CFLAGS  = -O2
```

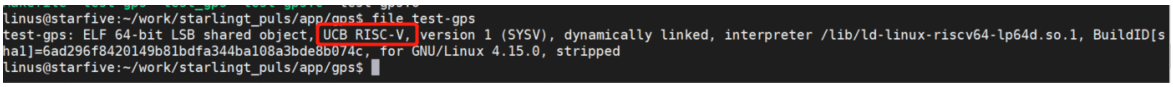
```
all:  clean $(EXEC)

$(EXEC):$(OBJS)
                $(CC) $(CFLAGS) -o $@ $(OBJS)
                $(STRIP) $@

clean:
                -rm -f $(EXEC) *.o
```

5. Execute `make` to generate the executable `test-gps` file.
6. Execute `file` command to check if it is a RISC-V file. The following is an example output:

Figure 3-1 Example Output



7. Execute the following command in Ubuntu to upload the executable file `test-gps` to your desired directory of the board, for example, `test`:

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

Note:

- `<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-gps riscv@192.168.92.133:/home/riscv/test
```

8. Execute the following command on VisionFive to run the demo codes:

```
./test-gps -d /dev/ttyUSB1 -b 9600
```

Result:

The following output indicates the execution is successful:

```
[root@fedora-starfive test]# ./test-gps -d /dev/ttyUSB1 -b 9600
                ctrl + c to exit!
$GPGSV,2,1,06,03,26,272,47,04,23,313,48,16,40,224,42,22,,,43*41
$GPGSV,2,2,06,26,,,49,41,,,38*78
$GPVTG,,T,,M,,N,,K,N*2C
$GPGSA,A,1,,,,,,,,,,,,,,*1E
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSV,2,1,06,03,26,272,47,04,23,313,48,16,40,224,42,22,,,43*41
$GPGSV,2,2,06,26,,,49,41,,,38*78
$GPVTG,,T,,M,,N,,K,N*2C
$GPGSA,A,1,,,,,,,,,,,,,,*1E
$GPGGA,,,,,,0,,,,,,,,*66
$GPRMC,,V,,,,,,,,,,N*53
$GPGSV,2,1,06,03,26,272,47,04,23,313,48,16,40,224,42,22,,,42*40
$GPGSV,2,2,06,26,,,49,41,,,38*78
$GPGGA,075306.00,3958.446141,N,11618.981715,E,1,03,4.5,78.4,M,-7.0,M,,*41
$GPVTG,0.0,T,5.9,M,0.0,N,0.0,K,A*2F
$GPRMC,075306.00,A,3958.446141,N,11618.981715,E,0.0,0.0,260402,5.9,W,A*27
$GPGSA,A,2,03,04,16,,,,,,,,,,4.6,4.5,1.0*31
$GPGSV,2,1,06,03,26,272,47,04,23,313,48,16,40,224,42,22,,,42*40
$GPGSV,2,2,06,26,,,49,41,,,37*77
$GPGGA,075307.00,3958.446301,N,11618.982243,E,1,03,4.5,79.4,M,-7.0,M,,*42
$GPVTG,0.0,T,5.9,M,0.0,N,0.0,K,A*2F
$GPRMC,075307.00,A,3958.446301,N,11618.982243,E,0.0,0.0,260402,5.9,W,A*25
$GPGSA,A,2,03,04,16,,,,,,,,,,4.6,4.5,1.0*31
$GPGSV,2,1,06,03,26,272,47,04,23,313,48,16,40,224,41,22,,,43*42
$GPGSV,2,2,06,26,,,49,41,,,38*78
$GPGGA,075308.00,3958.446523,N,11618.983069,E,1,03,4.5,82.3,M,-7.0,M,,*43
$GPVTG,0.0,T,5.9,M,0.0,N,0.0,K,A*2F
```

Refer to http://aprs.gids.nl/nmea/ for how to analyze the GPS data.