Serial Expansion HAT User Manual

OVERVIEW

Serial Expansion HAT for Raspberry Pi, I2C Interface, Provides 2-ch UART and 8 GPIOs

FEATURES

- Raspberry Pi connectivity, compatible with Raspberry Pi Zero/Zero W/Zero WH/2B/3B/3B+
- Onboard SC16IS752, expands 2-ch UART and 8 programmable GPIO through I2C, no extra pin required
- It is stackable up to 16 modules by setting the address jumper, that means up to 32-ch UART
- Onboard multi LEDs for indicating the UART working status
- Reserved I2C control pins, allows to work with other control boards
- Comes with development resources and manual (examples in C and python)

SPECIFICATION

- Operating voltage: 3.3V
- Expansion chip: SC16IS752
- Control interface: I2C
- Dimension: 65mm x 30mm
- Mounting hole size: 3.0mm
# CONTENT

Overview ......................................................................................................................................... 1

Features ........................................................................................................................................ 1

Specification .................................................................................................................................. 1

Hardware ....................................................................................................................................... 4

Pinout ........................................................................................................................................... 5

LED ............................................................................................................................................... 5

I2C device address setting ............................................................................................................. 6

How to use ................................................................................................................................... 7

Download Demo codes ................................................................................................................ 7

Libraries Installation (Require network) .......................................................................................... 8

Enable I2C Interface ...................................................................................................................... 8

Demo codes ................................................................................................................................ 11

C/GPIO/ ........................................................................................................................................ 11

Files ............................................................................................................................................ 12

Codes Analysis ............................................................................................................................ 12

C/UART/ ....................................................................................................................................... 13

Files ........................................................................................................................................... 14

Send code ..................................................................................................................................... 15

Receive code ................................................................................................................................. 16

python .......................................................................................................................................... 17

Files ........................................................................................................................................... 17
Receive code ................................................................. 17

Send code ......................................................................... 18
HARDWARE

The SC16IS752 is an I2C-bus/SPI bus interface to a dual-channel high performance UART. It also provides the application with 8 additional programmable I/O pins. This module uses I2C interface by default, and its device address is hardware configurable by A0 and A1.
## PINOUT

<table>
<thead>
<tr>
<th>PIN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3V3</td>
<td>3.3V</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>TXDA</td>
<td>Transmit end of Channel A</td>
</tr>
<tr>
<td>RXDA</td>
<td>Receive end of Channel A</td>
</tr>
<tr>
<td>RTS A</td>
<td>Request to send of Channel A</td>
</tr>
<tr>
<td>CTSA</td>
<td>Clear to send of Channel A</td>
</tr>
<tr>
<td>TXDB</td>
<td>Transmit end of Channel B</td>
</tr>
<tr>
<td>RXDB</td>
<td>Receive end of Channel B</td>
</tr>
<tr>
<td>RTS B</td>
<td>Request to send of Channel B</td>
</tr>
<tr>
<td>CTS B</td>
<td>Clear to send of Channel B</td>
</tr>
</tbody>
</table>

## LED

PWR: Power indicator

TXDA: Channel A transmit indicator

RXDA: Channel A receive indicator

TXDB: Channel B transmit indicator

RXDB: Channel B receive indicator
I2C DEVICE ADDRESS SETTING

I2C device address can be configured by changing status of A0 and A1, that is welding OR resister to them according to this table:

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>SC161572/SC1615762 PC address (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>VDD</td>
<td>0x90 (1001 000X)</td>
</tr>
<tr>
<td>VDD</td>
<td>VSS</td>
<td>0x92 (1001 001X)</td>
</tr>
<tr>
<td>VDD</td>
<td>SCL</td>
<td>0x94 (1001 010X)</td>
</tr>
<tr>
<td>VDD</td>
<td>SDA</td>
<td>0x96 (1001 011X)</td>
</tr>
<tr>
<td>VSS</td>
<td>VDD</td>
<td>0x98 (1001 100X)</td>
</tr>
<tr>
<td>VSS</td>
<td>VSS</td>
<td>0x9A (1001 101X)</td>
</tr>
<tr>
<td>VSS</td>
<td>SCL</td>
<td>0x9C (1001 110X)</td>
</tr>
<tr>
<td>VSS</td>
<td>SDA</td>
<td>0x9E (1001 111X)</td>
</tr>
<tr>
<td>SCL</td>
<td>VDD</td>
<td>0xA0 (1010 000X)</td>
</tr>
<tr>
<td>SCL</td>
<td>VSS</td>
<td>0xA2 (1010 001X)</td>
</tr>
<tr>
<td>SCL</td>
<td>SCL</td>
<td>0xA4 (1010 010X)</td>
</tr>
<tr>
<td>SCL</td>
<td>SDA</td>
<td>0xA6 (1010 011X)</td>
</tr>
<tr>
<td>SDA</td>
<td>VDD</td>
<td>0xA8 (1010 100X)</td>
</tr>
<tr>
<td>SDA</td>
<td>VSS</td>
<td>0xAA (1010 101X)</td>
</tr>
<tr>
<td>SDA</td>
<td>SCL</td>
<td>0xAC (1010 110X)</td>
</tr>
<tr>
<td>SDA</td>
<td>SDA</td>
<td>0xAE (1010 111X)</td>
</tr>
</tbody>
</table>

Table 32: SC161572/SC1615762 address map

[1] X = logic 0 for write cycle; X = logic 1 for read cycle.

For details, please refer to datasheet: Page39

The I2C address in table are 8bits, however, the actual address is 7bits, you need to right-shift one bit to get the actual I2C address. For example, if you connect A1 and A0 to Vdd, the address of module is 0x90 according to the table, to get the actual address you need to right-shift the data from 1001 000X to 100 1000, that is 0x48.

【Note】This module A0 and A1 are default welded to 3.3V, with I2C address 0x48
HOW TO USE

DOWNLOAD DEMO CODES

Visit WaveShare Wiki, search with key words “Serial Expansion HAT”, open it and download demo codes:

Extract and copy folders to Raspberry Pi.
LIBRARIES INSTALLATION (REQUIRE NETWORK)

1  Install wiringPi

   1.1 Open Terminal of Raspbian(Ctrl+T), clone wiringPi from github

   ```
   git clone git://git.drogon.net/wiringPi
   ```

   1.2 Install

   ```
   cd wiringPi
   ./build
   ```

2  Install python libraries

   2.1 Install python-dev

   ```
   sudo apt-get install python-dev
   ```

   2.2 Install RPi.GPIO

   ```
   sudo apt-get install python-rpi.gpio
   ```

   2.3 Install smbus, which is I2C interfaces library

   ```
   sudo apt-get install python-smbus
   ```

   2.4 Install spidev, which is SPI interfaces library

   ```
   sudo apt-get install python-spidev
   ```

ENABLE I2C INTERFACE

1  Execute command: `sudo raspi-config`
2  Choose: Interfacing Options->I2C->Yes

3  Append this line to end of /boot/config.txt file: sudo nano /boot/config.txt

dtoverlay=sc16is752-i2c,int_pin=24,addr=0x48

# addr is different according to status of A0/A1, default 0X48

4  reboot

  sudo reboot
After rebooting, you can execute command: `ls /dev` to check if SC16IS752 has been enabled to kernel.
DEMO CODES

We provide demo codes for this module, based on C codes and python.

C/GPIO/

There are 8 GPIO which are expanded. You can connect LEDs to these GPIOs when testing.

1. Open the folder of the demo code

   cd Serial_Expansion_HAT_code/c/gpio/

   #change the path of you didn’t put the code in /home/pi

2. Compile and run the code

   make

   sudo ./main

3. After running the demo code, code will light on/off 8 LEDs one by one.
FILES

```
main.c: main function

SC16IS752GPIO.c(.h): functions control IO

Makefile: Codes compilation
```

CODES ANALYSIS

Functions in SC16IS752GPIO.c:

```
int GPIOExport(int Pin): Export GPIO

int GPIOUnexport(int Pin): Unexport GPIO

int GPIODirection(int Pin, int Dir): Set direction of GPIO

int GPIORead(int Pin): Read value of GPIO

int GPIOWrite(int Pin, int value): Write value to GPIO
```

Files used by these functions are all in /sys/class/gpio, and according to use guide, the GPIO generated by SC16IS752 is coded 504. So, the codes of 8 GPIO are from 504 to 511.

Functions in main.c:

```
GPIO_Init(): Initialize 8 GPIO.

GPIO_Exit(): Unexport GPIOs
```
C/UART/

To test UART demo code, you need to use two Serial Expansion HAT and two Raspberry Pi, one is set as receiver and another is sender. You can also connect with serial module to PC if you have no other Raspberry Pi and Expansion HAT. Connect RXB of Serial Expansion HAT to TXB/TX of other module, and TXB of Serial Expansion HAT to RXB/RX of another module.

1. Open the folder of code

```bash
cd Serial_Expansion_HAT_code/c/uart/
#change the path of you didn’t put the code in /home/pi
```

2. Two codes, receiver and sender, choose the one you want to run

```bash
cd receive/
# cd send/
```

3. Compile and run the code
make

sudo ./uart_receive

# sudo ./uart_send

4. After running the code. Receiver: keep receiving and print data received; Sender:

    send characters

FILES

```
pit@raspberrypi:/Serial_Expansion_HAT_code/c/uart $ tree
  .  |
  |  receive  
  |  Makefile
  |  uart_receive
  |  uart_receive.c
  |  
  |  send
  |  Makefile
  |  uart_send
  |  uart_send.c

2 directories, 6 files
```

/receive:

    Makefile: Code compilation, execute command make to compile project

    uart_receive.c: Receive function

    uart_receive: Executable file, generated after command make

/send:

    Makefile: Code compilation, execute command make to compile project

    uart_send.c: Send function

    uart_send: Executable file, generated after command make
SEND CODE

1. Initialize wiringPi

```c
if(wiringPiSetupGpio() < 0) { //use BCM2835 Pin number table
    printf("set wiringPi lib failed !!! \n");
    return 1;
} else {
    printf("set wiringPi lib success !!! \n");
}
```

2. Open serial

```c
int fd;

    if((fd = serialOpen (UART_DEV1, 115200)) < 0) {
        printf("serial err\n");
        return -1;
    }
```

3. Clear buffer

```c
serialFlush(fd);
serialPrintf(fd,"\r");
```

4. Send a string

```c
char *buf = "abcdefgh";
serialPuts(fd, buf);
```

5. Close serial

```c
serialClose(fd);
```
RECEIVE CODE

1. Initialize wiringPi
   
   ```c
   if(wiringPiSetupGpio() < 0) { //use BCM2835 Pin number table
       printf("set wiringPi lib failed  !!! \n\n");
       return 1;
   } else {
       printf("set wiringPi lib success  !!! \n\n");
   }
   ```

2. Open serial
   
   ```c
   if((fd = serialOpen (UART_DEV2, 115200)) < 0) {
       printf("serial err\n");
       return -1;
   }
   
   #define UART_DEV1    
   #define UART_DEV2    
   ```

3. Receive and print
   
   ```c
   for (;;) {
       putchar(serialGetchar(fd));
   }
   ```
PYTHON

1. Open the folder of code
   ```
   cd Serial_Expansion_HAT_code/pythont/
   # change the path if you didn’t put the code in /home/pi
   ```

2. Compile and run
   ```
   make
   sudo python receive.py
   # or sudo python send.py
   ```

FILES

```bash
pi@raspberrypi:~/Serial_Expansion_HAT_code/python $ ls
receive.py send.py
```

【Note】 Only serial codes provided

RECEIVE CODE

1. Open serial
   ```
   ser = serial.Serial("/dev/ttySC1",115200,timeout=1)
   ```

2. Clear buffer
   ```
   ser.flushInput()
   ```

3. Read data
   ```
   ser.inWaiting()
   ```

4. Read specified bytes of data
   ```
   ser.read(ser.inWaiting())
   ```
SEND CODE

1. Open serial

   ```
   ser = serial.Serial("/dev/ttySC1", 115200, timeout=1)
   ```

2. Define data sent

   ```
   ser = serial.Serial("/dev/ttySC1", 115200, timeout=1)
   ```

3. Write the data to serial

   ```
   ser.write(command)
   ```