JH-7110 DevKit WiFi and Bluetooth Developing and Porting Guide

Version: 1.0

Data: 2023/06/02

Doc ID: JH7110-DGEN-002

Legal Statements

Important legal notice before reading this documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to 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 arising 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 not reproduce the information contained herein, in whole or in part, without the written permission of StarFive.

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: sales@starfivetech.com
- Support: support@starfivetech.com

Contents

List of Tables..4

List of Figures..5

Legal Statements..ii

Preface..vi

1. WiFi Development Instructions..7

1.1. Introduction..7

1.1.1. WiFi Work Mode..7

1.1.2. WiFi Software Architecture..7

1.2. Porting Instruction..10

1.2.1. WiFi Driver Construction..10

1.2.2. Kernel Menu Configuration..12

1.2.3. Setting Custom MAC Address..16

1.2.4. Install Dongle Host Driver..17

1.3. WiFi Operating Instructions..17

1.3.1. Station Mode Operation..17

1.3.2. SoftAP Mode Operation..21

2. Bluetooth Development Instructions..24

2.1. Introduction..24

2.1.1. Bluetooth Software Architecture Overview..24

2.2. Bluetooth Installation..25

2.2.1. Dip Switch Settings..25

2.2.2. Enable Bluetooth Function Of Linux Kernel..26

2.2.3. Enable Bluetooth..27

2.3. Bluetooth Mac Address Configuration..28

List of Tables

Table 0-1 Revision History.. vi

List of Figures

Figure 1-1 Broadcom SDIO WLAN Dongle Concept.......... 9
Figure 1-2 Device Drivers.......... 12
Figure 1-3 Network device support.......... 13
Figure 1-4 Wireless LAN.......... 13
Figure 1-5 Broadcom FullMAC wireless cards support.......... 14
Figure 1-6 Enable Chip Interface.......... 15
Figure 1-7 Interrupt type.......... 15
Figure 1-8 CFG80211-Wireless configuration API.......... 16
Figure 1-9 Example Output.......... 19
Figure 1-10 Wlan0 IP.......... 20
Figure 1-11 Example Output.......... 22
Figure 2-1 BlueZ Overview Diagram.......... 25
Figure 2-2 Dip Switch Setting Locations.......... 26

Preface

About this guide and technical support information.

About this document

This document mainly provides the SDK developers with the programing basics and debugging know-how for the WiFi and Bluetooth of the StarFive next generation SoC platform - JH-7110.

Audience

This document mainly serves the WiFi and Bluetooth relevant driver developers. If you are developing other modules, place a request to your sales or support consultant for our complete documentation set on JH-7110.

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.0	2023/07/27	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.

1. WiFi Development Instructions

1.1. Introduction

WiFi is a wireless local area network technology. The working principle of WiFi technology is data transmission through wireless signals. Devices are connected to wireless networks through WiFi chips, and then connected to the internet through routers. When the device needs to transmit data, it will send the data to the router, which then sends the data to the target device to achieve data transmission.

This chapter displays the following two parts:

- WiFi Work Mode *(on page)*
- WiFi Software Architecture *(on page 7)*

1.1.1. WiFi Work Mode

On VisionFive 2, the WiFi module generally has the following two work modes: Station mode and AP mode.

- Station mode: The endpoint connected to the wireless network. This is the most common work mode, most network adapters work in this mode.
- AP mode: The wireless access point, also called hotspot. In this mode, your device works as a wireless router.

1.1.2. WiFi Software Architecture

1.1.2.1. WiFi Dongle Basic Concept

The WLAN software package contains the dongle host driver for the host, a downloadable binary image for WLAN dongle, and management utilities.

The wireless driver runs on the WLAN dongle. The SDIO host controller passes IEEE 802.3 packets, and the necessary control packets, back and forth over the SDIO bus. A special Broadcom Device Class protocol is used to encapsulate control packets on a separate logical control channel and to add packet information to the data channel.

The advantage of using the dongle concept is that the wireless driver is executed externally from a host device, which means the host device does not have to use CPU or memory resources in order to execute the wireless driver's functionality. The use of the dongle provides the following benefits to the host:

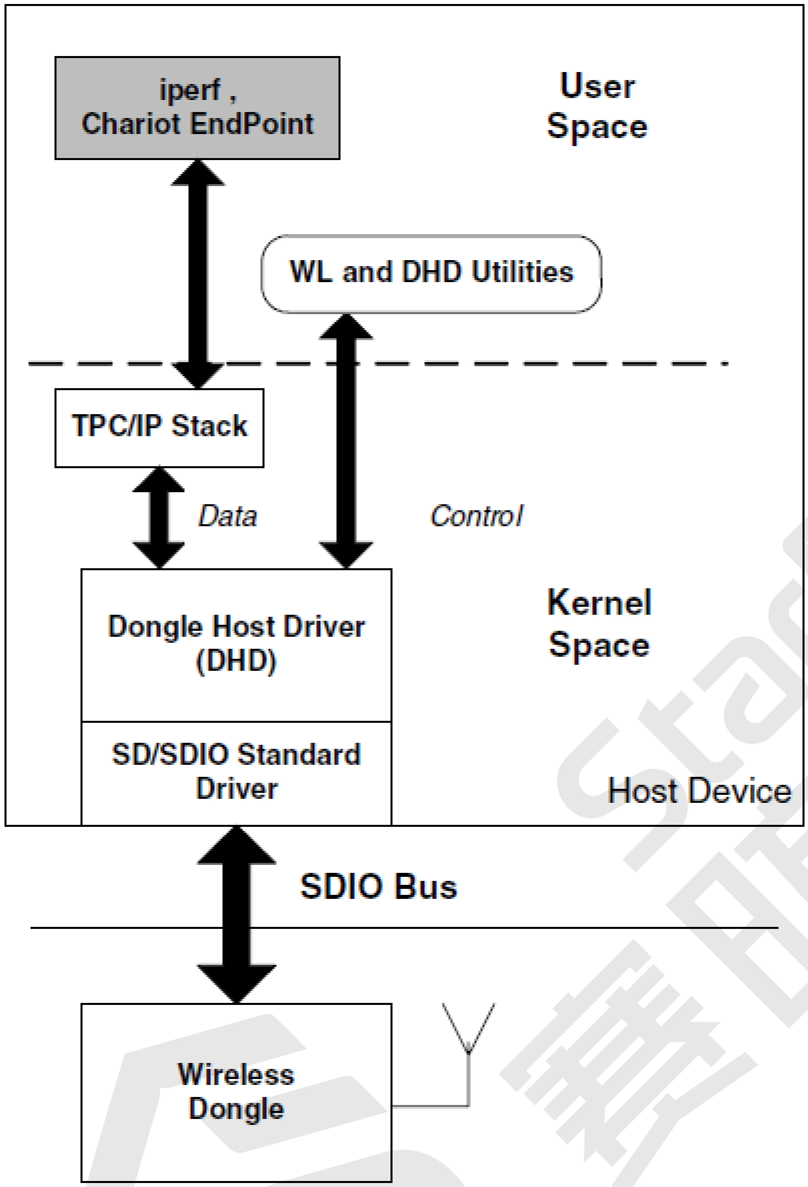
- Power savings
- A reduction in driver size and complexity
- Processor offloading for activities such as checksum calculation and Address Resolution Protocol (ARP) execution

1.1.2.2. WiFi Dongle Overview

The Dongle Host Driver (DHD) is the executable module that provides encapsulated communication between the host device and the Ampak module over the SDIO bus. The dongle software architecture is based on DHD, which is a host-based driver used to provide a communication channel with the dongle device firmware.

The following is the digram of the Broadcom SDIO WLAN Dongle concept.

Figure 1-1 Broadcom SDIO WLAN Dongle Concept



Tip:

Chaiot EndPoint is one of the top network business and performance testing software, capable of simulating numerous commercial applications for testing.

1.1.2.3. WiFi Software Package

The WiFi software package of JH-7110 DevKit contains following files:

- Dongle host driver (`bcmdhd`): WLAN adapter host driver.
- Dongle device firmware (`fw_bcm4345c5_ag.bin`): WLAN adapter device firmware.
- NVRAM (`nvram_ap6256.txt`): Ap6256 WiFi configuration file.

1.2. Porting Instruction

This chapter describes the porting instruction in the following parts:

- WiFi Driver Construction *(on page 10)*
- Kernel Menu Configuration *(on page 12)*
- Setting Custom MAC Address *(on page 16)*
- Install Dongle Host Driver *(on page 17)*

1.2.1. WiFi Driver Construction

Follow the steps below to port the WiFi driver:

1. Place the bcmdhd driver code under /linux/driver/net/wireless directory to build the bcmdhd driver.
2. Add the following content to /linux/driver/net/wireless/Makefile:

```
obj-$(CONFIG_BCMDHD) += bcmdhd/
```

3. Add the following content to /linux/driver/net/wireless/Kconfig:

```
source "drivers/net/wireless/bcmdhd/Kconfig"
```

4. Add following content under the SDIO node of DTS:

```
&sdio1 {
    ...
    max-frequency = <50000000>;
    address-cells = <1>;
    size-cells = <0>;
    bus-width = <4>;
    cap-sd-highspeed;
    gpio_wl_reg_on= <&ext_gpio 1 GPIO_ACTIVE_LOW>; // Power up/down
 internal regulators used by WiFi section
    ...
    status = "okay";

    brcmf: bcmdhd_wlan {
        compatible = "bcmdhd_wlan";
        gpio_wl_host_wake = <&gpioa 3 GPIO_ATIVE_HIGH>;//WL_HOST_WAKE
    };
};
```

The following code block displays the SDIO node reference:

```
&sdio1 {
max-frequency = <25000000>;
card-detect-delay = <300>;
post-power-on-delay-ms = <200>;
#address-cells = <1>;
    #size-cells = <0>;
    bus-width = <4>;
    cap-power-off-card;
    supports-sdio;
    ignore-pm-notify;
    keep-power-in-suspend;
    cap-sdio-irq;
    gpio_wl_reg_on = <&ext_gpio 1 GPIO_ACTIVE_LOW>;
no-sd;
no-mmc;
    non-removable;
    pinctrl-names = "default";
    pinctrl-0 = <&sdcard1_pins>;
    status = "okay";

    brcmf: bcmdhd_wlan {
        compatible = "bcmdhd_wlan";
        gpio_wl_host_wake = <&gpioa 3 GPIO_ACTIVE_HIGH>;
    };
};
```

5. Add control over `gpio_wl_reg_on` in mmc driver detection function:

```
static int dw_mci_starfive_probe(struct platform_device *pdev)
{
 …
gpio_wl_reg_on = of_get_named_gpio(pdev->dev.of_node,
 "gpio_wl_reg_on", 0);
 if (gpio_wl_reg_on >= 0) {
  ret = gpio_request(gpio_wl_reg_on, "WL_REG_ON");
  if (ret < 0) {
   dev_err(&pdev->dev, "gpio_request(%d) for WL_REG_ON failed %d\n",
     gpio_wl_reg_on, ret);
   gpio_wl_reg_on = -1;
   return -EINVAL;
  }
  ret = gpio_direction_output(gpio_wl_reg_on, 0);
  if (ret) {
   dev_err(&pdev->dev, "WL_REG_ON didn't output high\n");
   return -EIO;
  }
  mdelay(10);
  ret = gpio_direction_output(gpio_wl_reg_on, 1);
  if (ret) {
   dev_err(&pdev->dev, "WL_REG_ON didn't output high\n");
   return -EIO;
  }
  mdelay(10);
 }
 …
```

6. Place the WiFi firmware and configuration file to the specified directory: According to the configuration of the 5th step in Kernel Menu Configuration *(on page 12)*, place the WiFi firmware (`fw_bcm4345c5_ag.bin`) and configuration file (`nvram_ap6256.txt`) to Firmware path and NVRAM path.

1.2.2. Kernel Menu Configuration

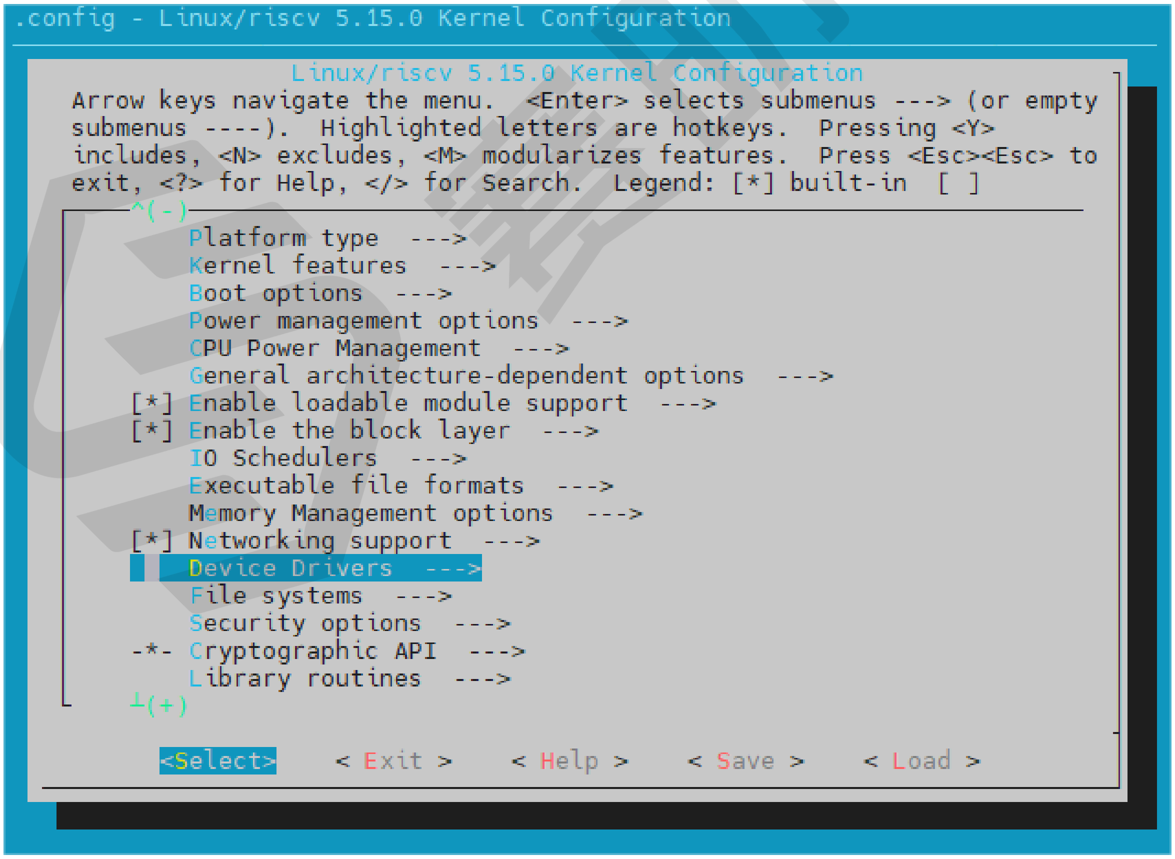
Follow the steps below to enter the kernel menu to enable the kernel configuration for WiFi.

1. Under the root directory of `devkits`, type the following command to enter the kernel menu configuration GUI.

```
make linux-menuconfig
```

2. Enter the **Device Drivers** menu option.

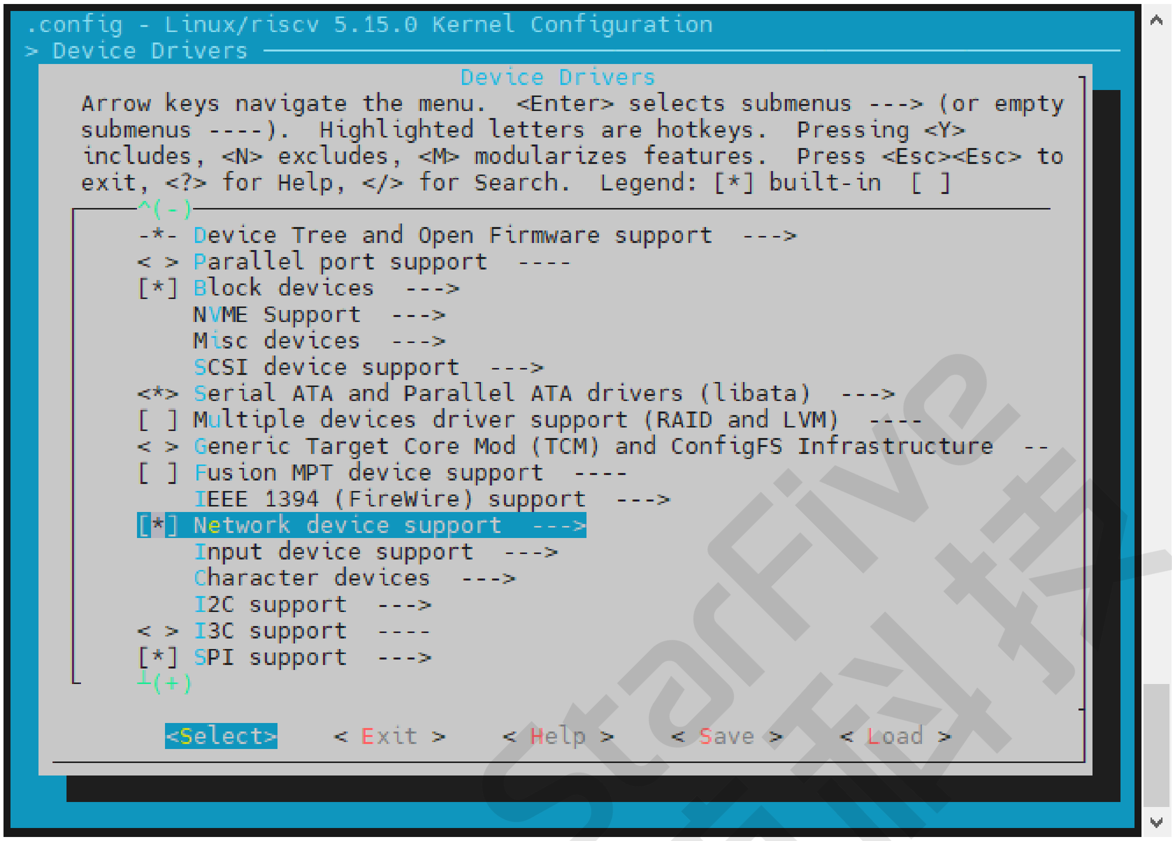
Figure 1-2 Device Drivers



```
.config - Linux/riscv 5.15.0 Kernel Configuration
                    Linux/riscv 5.15.0 Kernel Configuration
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
   submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
   includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
   exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
        ^(-)
            Platform type  --->
            Kernel features  --->
            Boot options  --->
            Power management options  --->
            CPU Power Management  --->
            General architecture-dependent options  --->
        [*] Enable loadable module support  --->
        [*] Enable the block layer  --->
            IO Schedulers  --->
            Executable file formats  --->
            Memory Management options  --->
        [*] Networking support  --->
            Device Drivers  --->
            File systems  --->
            Security options  --->
        -*- Cryptographic API  --->
            Library routines  --->
        ┴(+)

         <Select>    < Exit >    < Help >    < Save >    < Load >
```

3. Enter the **Network device support** menu option.

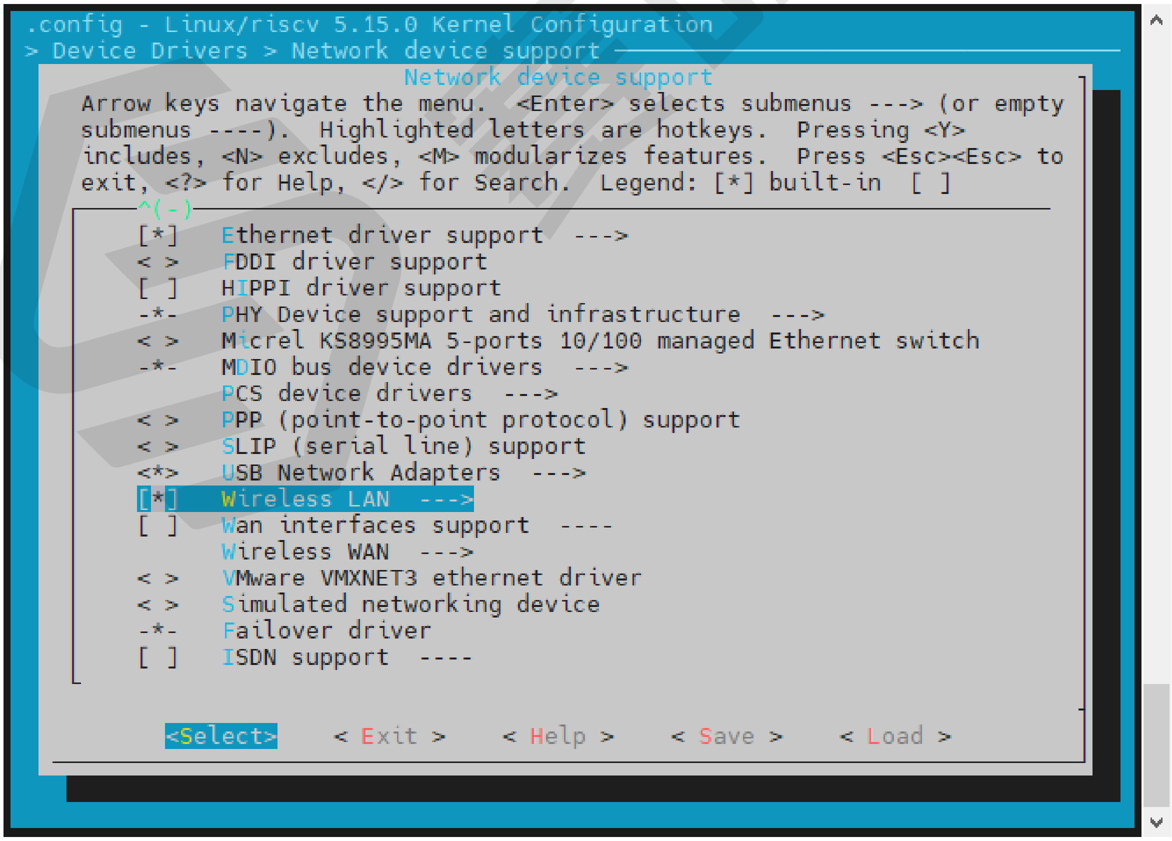
Figure 1-3 Network device support



```
.config - Linux/riscv 5.15.0 Kernel Configuration
> Device Drivers ─────────────────────────────────────────────
                         Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
    ^(-)
        -*- Device Tree and Open Firmware support  --->
        < > Parallel port support  ----
        [*] Block devices  --->
            NVME Support  --->
            Misc devices  --->
            SCSI device support  --->
        <*> Serial ATA and Parallel ATA drivers (libata)  --->
        [ ] Multiple devices driver support (RAID and LVM)  ----
        < > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  --
        [ ] Fusion MPT device support  ----
            IEEE 1394 (FireWire) support  --->
        [*] Network device support  --->
            Input device support  --->
            Character devices  --->
            I2C support  --->
        < > I3C support  ----
        [*] SPI support  --->
        └(+)

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

4. Enter **Wireless LAN** menu option.

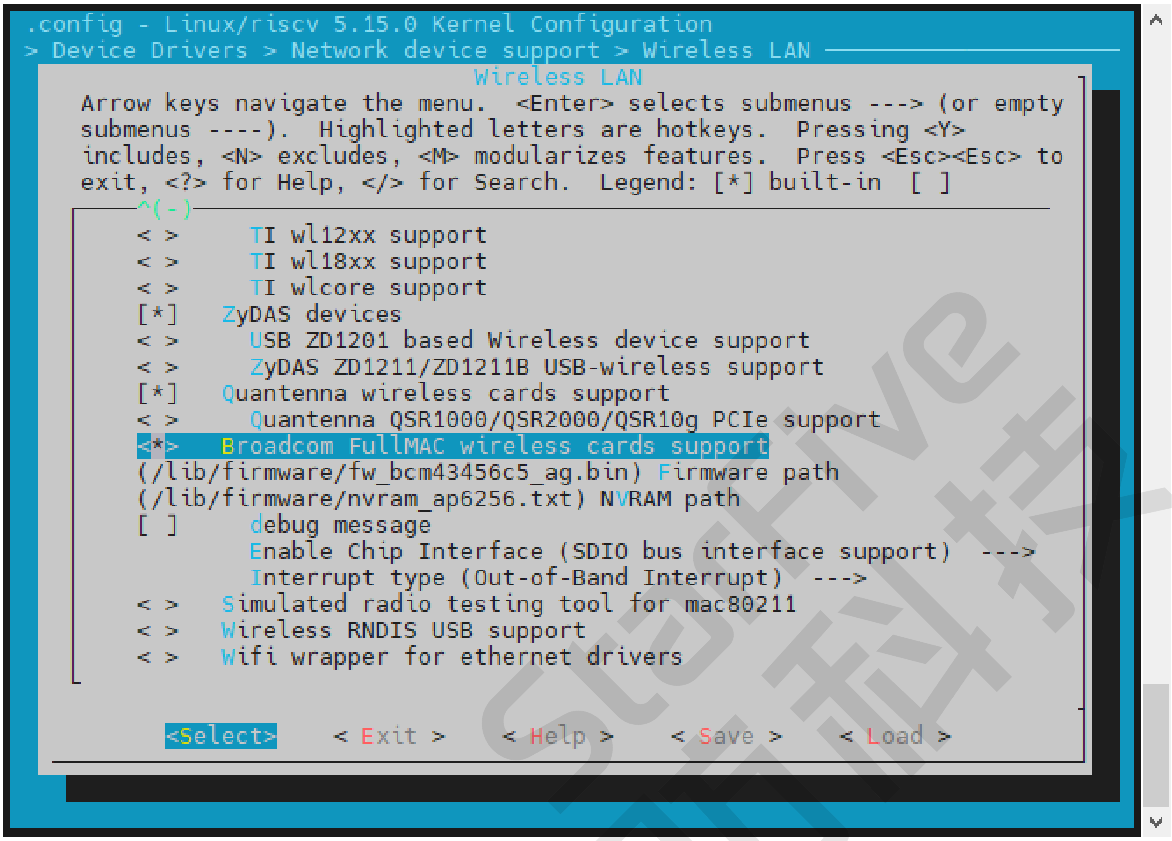
Figure 1-4 Wireless LAN



```
.config - Linux/riscv 5.15.0 Kernel Configuration
> Device Drivers > Network device support ────────────────────
                     Network device support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
    ^(-)
        [*]   Ethernet driver support  --->
        < >   FDDI driver support
        [ ]   HIPPI driver support
        -*-   PHY Device support and infrastructure  --->
        < >   Micrel KS8995MA 5-ports 10/100 managed Ethernet switch
        -*-   MDIO bus device drivers  --->
              PCS device drivers  --->
        < >   PPP (point-to-point protocol) support
        < >   SLIP (serial line) support
        <*>   USB Network Adapters  --->
        [*]   Wireless LAN  --->
        [ ]   Wan interfaces support  ----
              Wireless WAN  --->
        < >   VMware VMXNET3 ethernet driver
        < >   Simulated networking device
        -*-   Failover driver
        [ ]   ISDN support  ----

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

5. Choose **Broadcom FullMAC wireless cards support** option.

Figure 1-5 Broadcom FullMAC wireless cards support



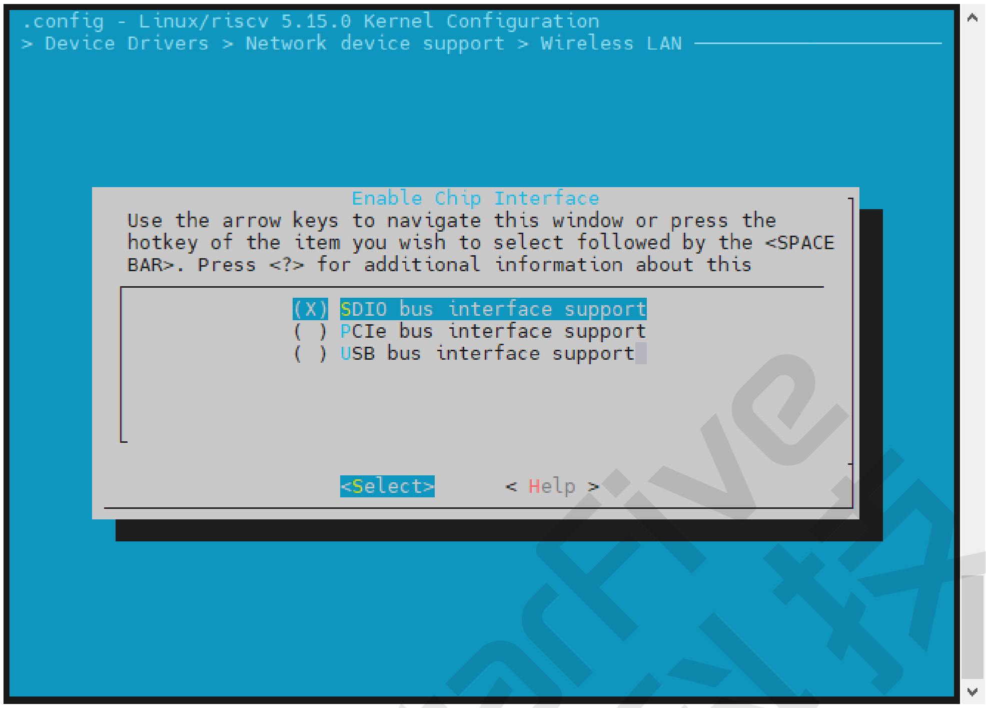
```
 .config - Linux/riscv 5.15.0 Kernel Configuration
 > Device Drivers > Network device support > Wireless LAN ──────────────────────
                               Wireless LAN
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
   submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
   includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
   exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
  ┌────^(-)─────────────────────────────────────────────────────────────┐
  │    < >      TI wl12xx support
  │    < >      TI wl18xx support
  │    < >      TI wlcore support
  │    [*]    ZyDAS devices
  │    < >      USB ZD1201 based Wireless device support
  │    < >      ZyDAS ZD1211/ZD1211B USB-wireless support
  │    [*]    Quantenna wireless cards support
  │    < >      Quantenna QSR1000/QSR2000/QSR10g PCIe support
  │    <*>    Broadcom FullMAC wireless cards support
  │    (/lib/firmware/fw_bcm43456c5_ag.bin) Firmware path
  │    (/lib/firmware/nvram_ap6256.txt) NVRAM path
  │    [ ]      debug message
  │             Enable Chip Interface (SDIO bus interface support)  --->
  │             Interrupt type (Out-of-Band Interrupt)  --->
  │    < >    Simulated radio testing tool for mac80211
  │    < >    Wireless RNDIS USB support
  │    < >    Wifi wrapper for ethernet drivers
  └

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

Note:

Pressing **Y** to compile it into kernel while pressing **M** to compile as module. The following lists are the description of some options in the menu.

- **Firmware path** is the firmware storage path while the **NVRAM path** is the NVRAM configuration file path, which can be selected according to needs.
- **debug message**: Choose this option to add debugging information for loading drivers in the kernel.
- **Enable Chip Interface**: This is used to select the WiFi interface. In this example, SDIO is used as the WiFi interface. The following figure shows the options under **Enable Chip Interface**:

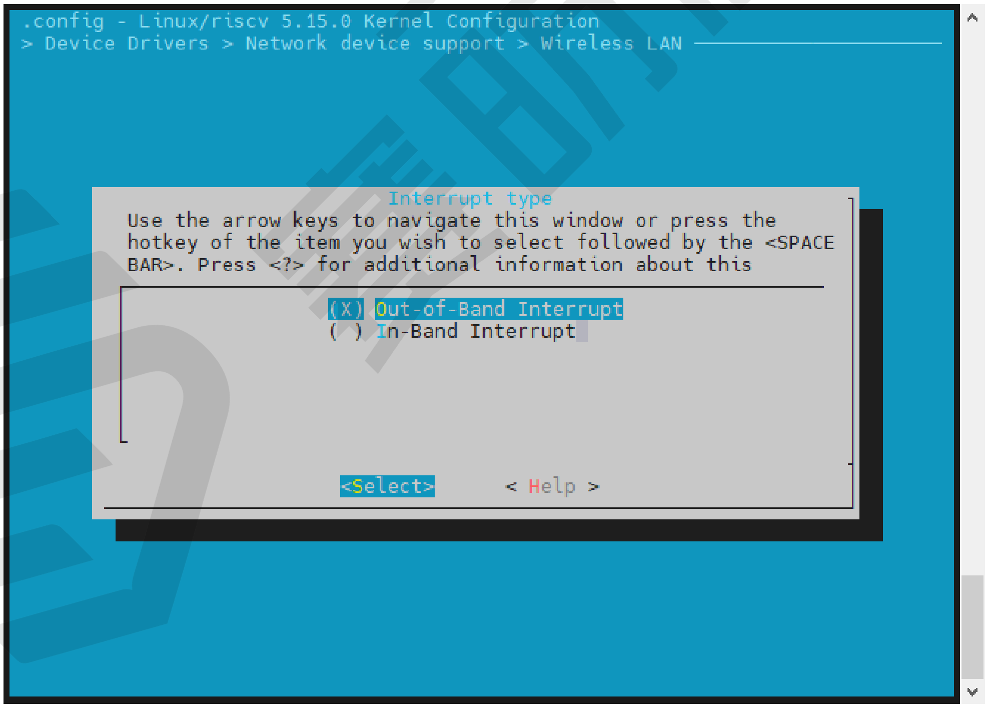
Figure 1-6 Enable Chip Interface



```
.config - Linux/riscv 5.15.0 Kernel Configuration
> Device Drivers > Network device support > Wireless LAN

                         Enable Chip Interface
   Use the arrow keys to navigate this window or press the
   hotkey of the item you wish to select followed by the <SPACE
   BAR>. Press <?> for additional information about this

                 (X) SDIO bus interface support
                 ( ) PCIe bus interface support
                 ( ) USB bus interface support

                    <Select>      < Help >
```

- **Interrupt type**: This is used to choose the wake interrupt.

Figure 1-7 Interrupt type



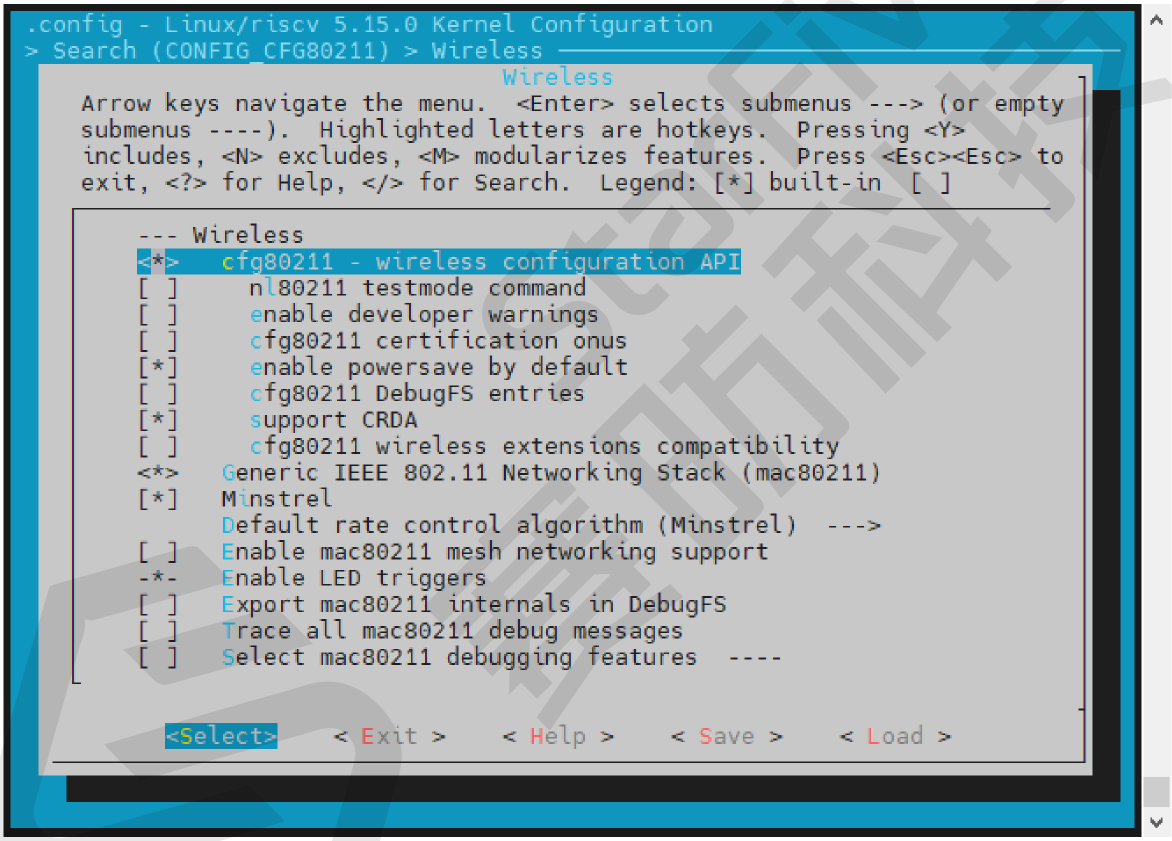
```
.config - Linux/riscv 5.15.0 Kernel Configuration
> Device Drivers > Network device support > Wireless LAN

                            Interrupt type
   Use the arrow keys to navigate this window or press the
   hotkey of the item you wish to select followed by the <SPACE
   BAR>. Press <?> for additional information about this

                    (X) Out-of-Band Interrupt
                    ( ) In-Band Interrupt

                    <Select>      < Help >
```

- **Out-of-Band Interrupt**: The BCM chip defines an Out of Band (OOB) interrupt, also known as `WL_ HOST_ WAKE` pin, and you can remap the interrupt signal to this pin.
- **In-Band Interrupt**: In the SDIO protocol, the DATA1 line of the SDIO card can be used as an interrupt line, called In-Band Interrupt.

6. Enable `CFG80211` and remove `CONFIG_BRCMFMAC`:

```
CONFIG_CFG80211=y
# CONFIG_BRCMFMAC is not set
```

Figure 1-8 CFG80211-Wireless configuration API



```
.config - Linux/riscv 5.15.0 Kernel Configuration
> Search (CONFIG_CFG80211) > Wireless ──────────────────────────────────────
                              Wireless
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

    --- Wireless
    <*>   cfg80211 - wireless configuration API
    [ ]     nl80211 testmode command
    [ ]     enable developer warnings
    [ ]     cfg80211 certification onus
    [*]     enable powersave by default
    [ ]     cfg80211 DebugFS entries
    [*]     support CRDA
    [ ]     cfg80211 wireless extensions compatibility
    <*>   Generic IEEE 802.11 Networking Stack (mac80211)
    [*]   Minstrel
          Default rate control algorithm (Minstrel)  --->
    [ ]   Enable mac80211 mesh networking support
    -*-   Enable LED triggers
    [ ]   Export mac80211 internals in DebugFS
    [ ]   Trace all mac80211 debug messages
    [ ]   Select mac80211 debugging features  ----

     <Select>    < Exit >    < Help >    < Save >    < Load >
```

7. Save your change before you exit the kernel configuration dialog.

1.2.3. Setting Custom MAC Address

If you would like to configure WiFi MAC address, following the steps to modify driver to get it work.

1. Add `-DGET_CUSTOM_MAC_ENABLE` in driver Makefile.
2. Modify the `dhd_wlan_get_mac_addr` function in `dhd_gpio.c` to read your MAC address where located in your system.
3. Then the WiFi driver will update the firmware MAC address during initialization.

1.2.4. Install Dongle Host Driver

The following provides two methods to install WiFi Dongle host driver:

- To compile the Dongle host driver into kernel, execute the following command:

```
ifconfig wlan0 up
```

- To compile the Dongle host driver as a module, execute the following command:

```
insmod /lib/module/bcmdhd.ko
ifconfig wlan0 up
```

1.3. WiFi Operating Instructions

This chapter describes the WiFi operation instruction in the following two parts:

- Station Mode Operation *(on page 17)*
- SoftAP Mode Operation *(on page 21)*

1.3.1. Station Mode Operation

1.3.1.1. Add WPA Supplicant Configuration File

Follow the steps below to create the `wpa_supplicant.conf` file:

1. Manually creating the `wpa_supplicant.conf` configuration file or use `/etc/wpa_supplicant.conf`.
2. Add the following content to the configuration file:

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
```

Note:
This allows `wpa_supplicant` to overwrite configuration files after modifying the configuration. For example, add new network statement blocks through `wpa_cli` tool, write configuration to `wpa_gui`, change password, etc.

Network Settings

There are 3 examples to add network for `wpa_supplicant.conf` file:

- Open system without encryption:

```
network={
ssid="tttb"
key_mgmt=NONE
}
```

- `Open/Shared` authentication with WEP encryption:

```
network={
ssid="tttb"
key_mgmt=NONE
auth_alg=OPEN SHARED
wep_key0=1234567890
}
```

- WPA/WPA2-PSK authentication with TKIP/AES encryption:

```
network={
ssid="tttb"
psk="12345678"
}
```

You can also use `wpa_passphrase` tool to add encrypted network automatically for your `wpa_supplicant.conf` file:

```
wpa_passphrase ssid >> wpa_supplicant.conf
password
```

1.3.1.2. Install WPA Supplicant

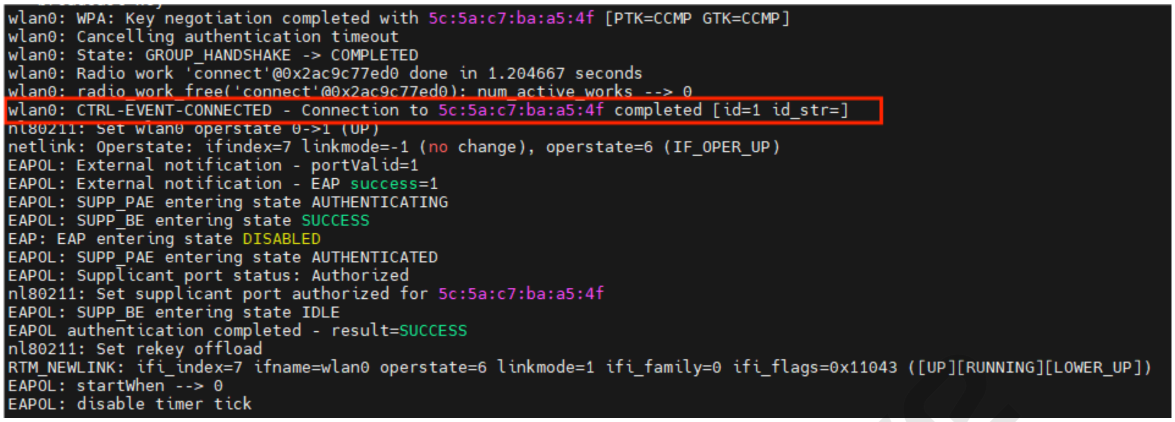
Follow the steps below to install WPA supplicant.

1. Execute the following command to connect to the network configured in `wpa_supplicant.conf`:

```
wpa_supplicant -Dnl80211 -i wlan0 -c wpa_supplicant.conf -d&
```

Example Output

Figure 1-9 Example Output



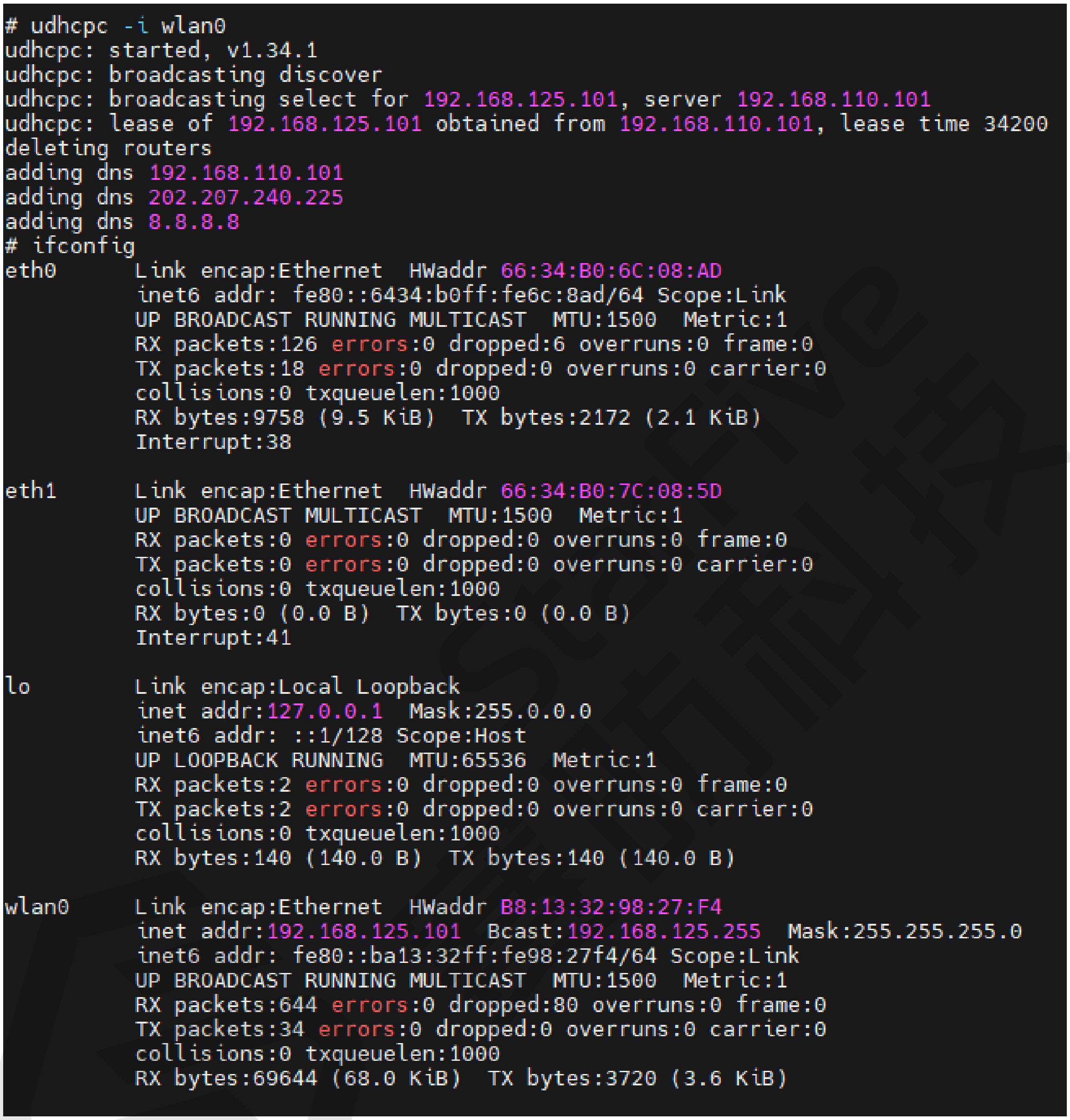
```
wlan0: WPA: Key negotiation completed with 5c:5a:c7:ba:a5:4f [PTK=CCMP GTK=CCMP]
wlan0: Cancelling authentication timeout
wlan0: State: GROUP_HANDSHAKE -> COMPLETED
wlan0: Radio work 'connect'@0x2ac9c77ed0 done in 1.204667 seconds
wlan0: radio_work_free('connect'@0x2ac9c77ed0): num_active_works --> 0
wlan0: CTRL-EVENT-CONNECTED - Connection to 5c:5a:c7:ba:a5:4f completed [id=1 id_str=]
nl80211: Set wlan0 operstate 0->1 (UP)
netlink: Operstate: ifindex=7 linkmode=-1 (no change), operstate=6 (IF_OPER_UP)
EAPOL: External notification - portValid=1
EAPOL: External notification - EAP success=1
EAPOL: SUPP_PAE entering state AUTHENTICATING
EAPOL: SUPP_BE entering state SUCCESS
EAP: EAP entering state DISABLED
EAPOL: SUPP_PAE entering state AUTHENTICATED
EAPOL: Supplicant port status: Authorized
nl80211: Set supplicant port authorized for 5c:5a:c7:ba:a5:4f
EAPOL: SUPP_BE entering state IDLE
EAPOL authentication completed - result=SUCCESS
nl80211: Set rekey offload
RTM_NEWLINK: ifi_index=7 ifname=wlan0 operstate=6 linkmode=1 ifi_family=0 ifi_flags=0x11043 ([UP][RUNNING][LOWER_UP])
EAPOL: startWhen --> 0
EAPOL: disable timer tick
```

2. After connect successfully, execute the following command to obtain the IP of **wlan0** by using the `udhcpc` tool:

```
udhcpc -i wlan0
```

Example Output

Figure 1-10 Wlan0 IP



```
# udhcpc -i wlan0
udhcpc: started, v1.34.1
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.125.101, server 192.168.110.101
udhcpc: lease of 192.168.125.101 obtained from 192.168.110.101, lease time 34200
deleting routers
adding dns 192.168.110.101
adding dns 202.207.240.225
adding dns 8.8.8.8
# ifconfig
eth0      Link encap:Ethernet  HWaddr 66:34:B0:6C:08:AD
          inet6 addr: fe80::6434:b0ff:fe6c:8ad/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:126 errors:0 dropped:6 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9758 (9.5 KiB)  TX bytes:2172 (2.1 KiB)
          Interrupt:38

eth1      Link encap:Ethernet  HWaddr 66:34:B0:7C:08:5D
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:41

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:140 (140.0 B)  TX bytes:140 (140.0 B)

wlan0     Link encap:Ethernet  HWaddr B8:13:32:98:27:F4
          inet addr:192.168.125.101  Bcast:192.168.125.255  Mask:255.255.255.0
          inet6 addr: fe80::ba13:32ff:fe98:27f4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:644 errors:0 dropped:80 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:69644 (68.0 KiB)  TX bytes:3720 (3.6 KiB)
```

1.3.1.3. `wpa_cli` Tool

`wpa_cli` is a text based front-end program used to communicate with `wpa_supplicant`. It is used to check the current state, change configuration, trigger events, and request interactive user input.

In addition, it can be used to configure variables such as EAPOL state machine parameters and trigger events such as re-association and IEEE 802.1X logout/login.

`wpa_cli` has many functions, which can be seen through execute `wpa_cli -h` command. The following list are some example commands:

- Check the WLAN status:

```
wpa_cli -p/var/run/wpa_supplicant -iwlan0 status
```

- Scan the nearby WLAN:

```
wpa_cli -i wlan0 scan
```

- Scan the nearby WLAN and list the devices:

```
wpa_cli -i wlan0 scan_result
```

- View all accessible networks and the currently connected network:

```
wpa_cli -i wlan0 list_networks
```

- Obtain an ID that stores the WLAN structure, assuming 1:

```
wpa_cli -i wlan0 add_network
```

- Set the hotspot SSID with ID 1:

```
wpa_cli -i wlan0 set_network 1 ssid '"HO4428"'
```

- Set the password for the hotspot with ID 1:

```
wpa_cli -i wlan0 set_network 1 psk '"442701102"'
```

1.3.2. SoftAP Mode Operation

1.3.2.1. Add Hostapd Configuration File

Follow the steps to create the `hotsapd.conf` file:

1. Create the `hostapd.conf` configuration file manually or use `/etc/hostapd.conf`.
2. Add the following content to the configuration file.

AP Hotpot Settings

The following are 2 examples to add AP hotpot for your `hostapd.conf` file:

- Open system without encryption:

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=AndroidAP
channel=6
```

```
ieee80211n=1
hw_mode=g
ignore_broadcast_ssid=0
```

- WPA2-PSK authentication with AES encryption:

```
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=AndroidAP
channel=6
ieee80211n=1
hw_mode=g
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=12345678
```

1.3.2.2. Install Hostapd

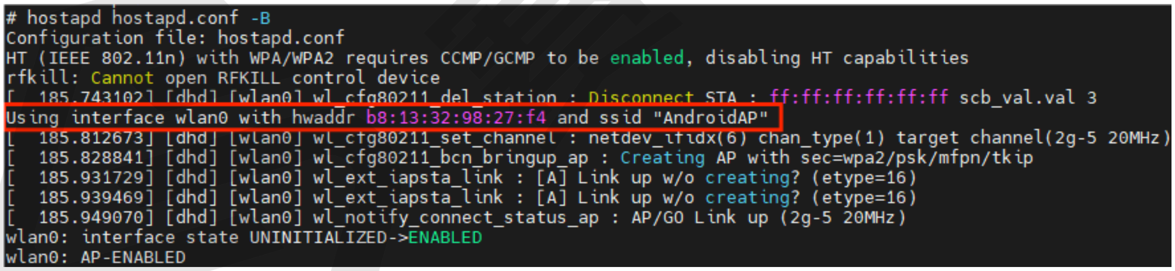
Follow the steps below to install hostapd.

1. Execute the following command to create AP hotpot configured in `hostapd.conf`:

```
hostapd hostapd.conf -B
```

Example Output

Figure 1-11 Example Output



```
# hostapd hostapd.conf -B
Configuration file: hostapd.conf
HT (IEEE 802.11n) with WPA/WPA2 requires CCMP/GCMP to be enabled, disabling HT capabilities
rfkill: Cannot open RFKILL control device
[  185.743102] [dhd] [wlan0] wl_cfg80211_del_station : Disconnect STA : ff:ff:ff:ff:ff:ff scb_val.val 3
Using interface wlan0 with hwaddr b8:13:32:98:27:f4 and ssid "AndroidAP"
[  185.812673] [dhd] [wlan0] wl_cfg80211_set_channel : netdev_ifidx(6) chan_type(1) target channel(2g-5 20MHz)
[  185.828841] [dhd] [wlan0] wl_cfg80211_bcn_bringup_ap : Creating AP with sec=wpa2/psk/mfpn/tkip
[  185.931729] [dhd] [wlan0] wl_ext_iapsta_link : [A] Link up w/o creating? (etype=16)
[  185.939469] [dhd] [wlan0] wl_ext_iapsta_link : [A] Link up w/o creating? (etype=16)
[  185.949070] [dhd] [wlan0] wl_notify_connect_status_ap : AP/GO Link up (2g-5 20MHz)
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

You can use your phone or computer to locate the hotspot, but you cannot connect to it. To solve this issue, you need to assign an IP address to this hotspot.

2. Add subnet to the main configuration file `/etc/dhcp/dhcpconf of dhcp`, for example:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
range 192.168.0.10 192.168.0.20;
option routers 192.168.0.1;
}
```

Then, execute the following command to configure the IP address for the WiFi interface:

```
ifconfig wlan0 192.168.0.1
```

3. Execute the following command to troubleshoot the DHCP server:

   ```
   dhcpd
   ```

   Now, you can connect to the hotspot through devices such as mobile phone or PC and ping it for test.

4. (Optional) If you want to access the internet through this hotspot, you need to set up IP forwarding. IP forwarding means that forwards the interface connecting the network cable on SBC to the wlan0 interface. The following is an example:

   ```
   echo 1 > /proc/sys/net/ipv4/ip_forward
   iptables -F
   iptables -P FORWARD ACCEPT
   iptables -t nat -A POSTROUTING -o ethX -j MASQUERADE
   ```

2. Bluetooth Development Instructions

2.1. Introduction

Bluetooth is a wireless technology standard whose purpose is to connect gadgets without a cable. The Bluetooth module is a tiny part of the chip in a device, which lets it wirelessly communicate with a Bluetooth module on any other devices. Generally speaking, Bluetooth is used to transfer small amounts of data while being efficient with battery usage. Among the various wireless standards (like WiFi), Bluetooth is known for maintaining a stable connection in short distances, and transferring small amounts of data without taking too much power.

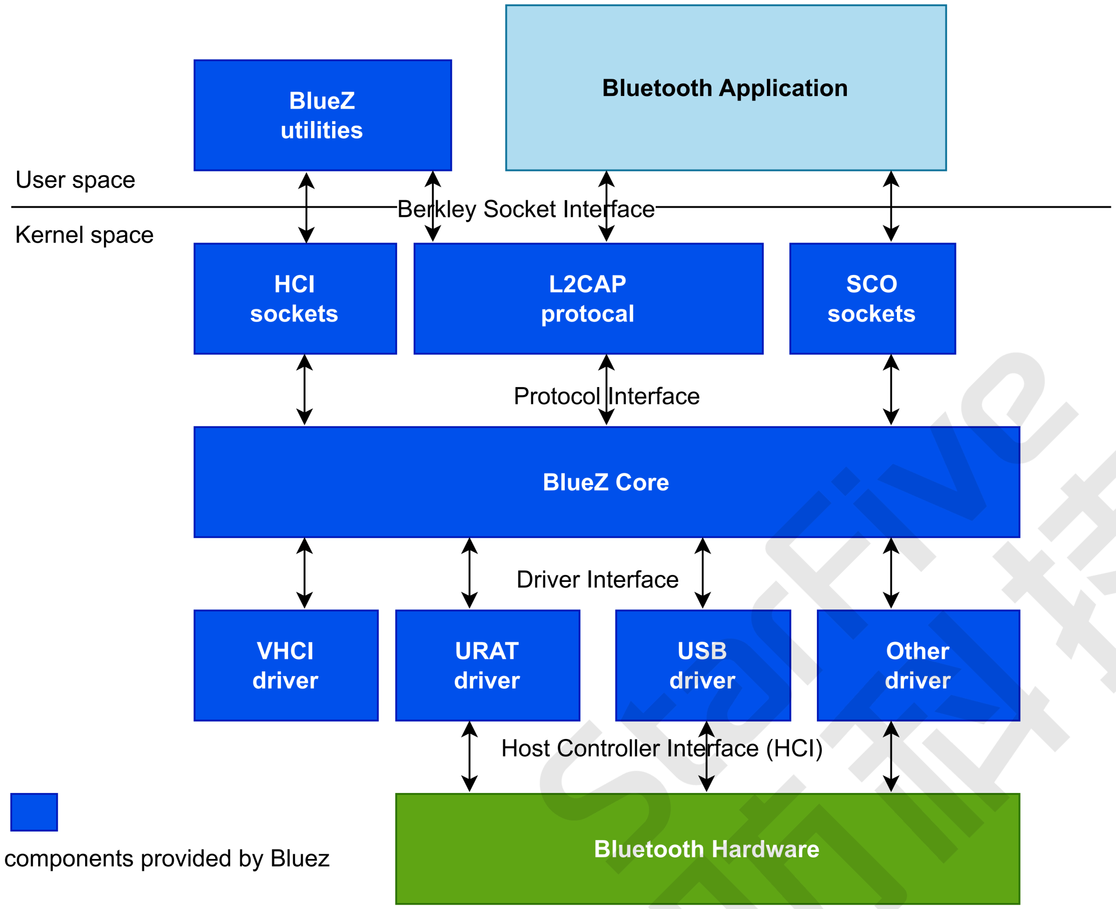
This chapter is intended to give Ampak Bluetooth module users a general guide of how to bring up the Bluetooth hci interface and customer can use bluez stack to enable full bluetooth function.

2.1.1. Bluetooth Software Architecture Overview

BlueZ is the official Linux Bluetooth stack as well as Android. It provides support for core Bluetooth layers and protocols. We use it to provide Bluetooth profiles on GB86XX and it consists of following components:

- HCI Core
- HCI UART, USB and Virtual HCI device drivers
- L2CAP protocol module
- Configuration and testing utilities

Figure 2-1 BlueZ Overview Diagram



In our case, we use UART as the **Host Controller Interface** (HCI) and AP6XXX is the **Bluetooth Hardware** in figure above.

2.2. Bluetooth Installation

2.2.1. Dip Switch Settings

Before connecting the Bluetooth mode on JH-7110 DevKit, you need to set the following dip switch:

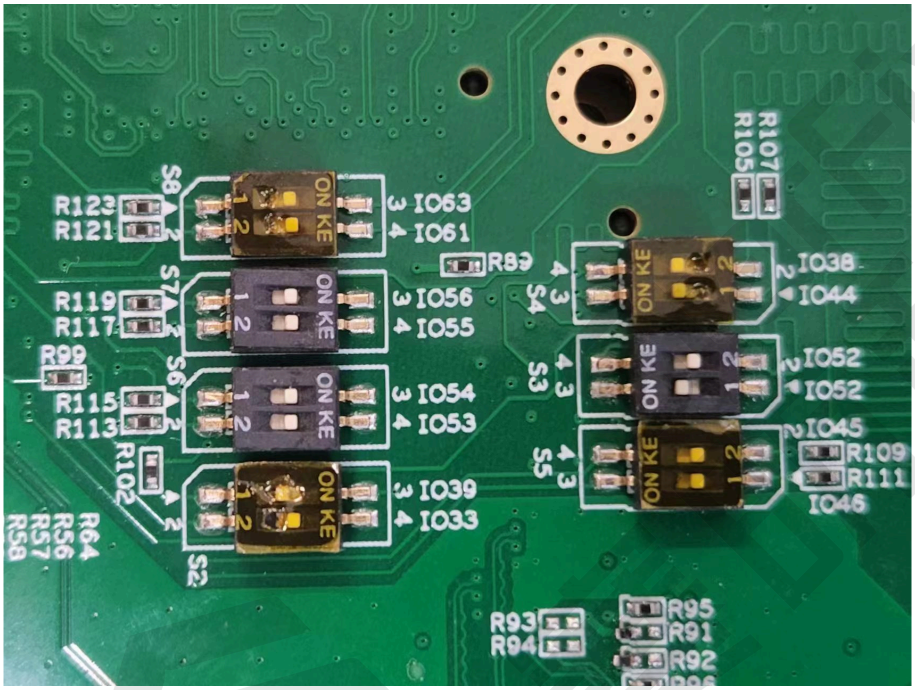
GPIO Group	GPIO	Dip Switch for Function	
		Dip Switch	Function
2	GPIO52	S3pin1: ON*	BT_EN_H
	GPIO53	S6pin2: ON	BT_UART_TXD
	GPIO54	S6pin1: ON	BT_UART_RXD
	GPIO55	S7pin2: ON	BT_UART_RTS

GPIO Group	GPIO	Dip Switch for Function	
		Dip Switch	**Function**
	GPIO56	S7pin1: ON	BT_UART_CTS

Note:
*: ON means connected, otherwise means disconnected.

The following figure displays the location and the settings of the dip switch.

Figure 2-2 Dip Switch Setting Locations



2.2.2. Enable Bluetooth Function Of Linux Kernel

Add following items into your kernel configuration to enable Bluetooth function of Linux kernel:

```
CONFIG_BT_HCIUART=y
CONFIG_BT_HCIUART_H4=y
CONFIG_BT=y
CONFIG_BT_L2CAP=y
CONFIG_BT_SCO=y
CONFIG_BT_RFCOMM=y
CONFIG_BT_RFCOMM_TTY=y
CONFIG_BT_BNEP=y
CONFIG_BT_BNEP_MC_FILTER=y
CONFIG_BT_BNEP_PROTO_FILTER=y
CONFIG_BT_HIDP=y
```

2.2.3. Enable Bluetooth

Following the steps below to enable the Bluetooth:

1. Perform the steps bellow to initialize Bluetooth:

    a. Perform the following command to bring up **hci** interface.

    ```
    #hciconfig hci0 up
    ```

2. Perform the following command to check Bluetooth device status:

    ```
    # hciconfig
    hci0: Type: BR/EDR Bus: UART
    BD Address: 43:30:B1:00:00:00 ACL MTU: 1021:8 SCO MTU: 64:1
    UP RUNNING
    RX bytes:1011 acl:0 sco:0 events:39 errors:0
    TX bytes:208 acl:0 sco:0 commands:39 errors:0
    ```

3. Perform the following to scan Bluetooth devices:

    ```
    # hcitool scan
    Scanning ...
    00:22:43:A0:A7:0A n/a
    00:10:60:56:56:7B hhhh
    00:1A:6B:85:F3:67 n/a
    00:22:43:A0:A7:48 AmUrO
    00:1F:E1:E1:A1:8F GEMTEK-8AE51F68
    ```

4. You can also enter Bluetooth interactive interface by using **Bluetoothctl** tool. **Bluetoothctl** has many functions, which can be seen through execute `help` command. The following list are some example commands:

    - Enter the tool:

      ```
      bluetoothctl
      ```

        - Check:

          ```
          default-agent
          ```

        - Register agent:

          ```
          agent on
          ```

        - Scan:

          ```
          scan on
          ```

        - Stop scan:

```
scan off
```

- View the matching devices:

```
devices
```

- Find the pair devices:

```
pair xx:xx:xx:xx:xx:xx
```

- Add trust devices:

```
trust xx:xx:xx:xx:xx:xx
```

- Connect devices:

```
connect xx:xx:xx:xx:xx:xx
```

2.3. Bluetooth Mac Address Configuration

Run the following to enable and configure Bluetooth.

```
# brcm_patchram_plus --enable_hci --no2bytes --tosleep 200000
--baudrate 115200 --patchram /lib/firmware/BCM4345C5.hcd /dev/ttyS1 &
cmd: HCI_Reset
cmd: HCI_Download_Minidriver
Sleep 200ms before downloading...
Downloaded
cmd: HCI_Reset
cmd: HCI_Write_BD_ADDR
Done setting line discpline
Device setup complete
pid : 1948

# hciconfig hci0 up

# hciconfig
hci0: Type: BR/EDR Bus: UART
 BD Address: 11:22:33:44:55:66 ACL MTU: 1021:8 SCO MTU: 64:1
 UP RUNNING
 RX bytes:1011 acl:0 sco:0 events:39 errors:0
 TX bytes:208 acl:0 sco:0 commands:39 errors:0
```