DUSUN DSGW-210 IoT Edge Computer Gateway User Guide

Contents

DUSUN DSGW-210 IoT Edge Computer Gateway

Product Information

Hangzhou Roombanker Technology Co., Ltd. presents the IoT Edge Computer Gateway Model Name: DSGW-210. This product is designed to serve as an IoT gateway between devices and the cloud. The gateway provides a secure and reliable connection to the cloud, making it easy to manage and control devices remotely.

Introduction

This Quick Start Guide explains the basics: how to connect and set up your target on the network; how to install the SDK; and how to build the firmware images.
The Linux Software Developer's Kit (SDK) is an embedded hardware and software suite that enables Linux developers to create applications on Dusun's DSGW-210 gateway.
Base on the 4.4 Linux kernel, and leveraging existing open source software, the SDK simplifies the process of adding custom applications. Device drivers, GNU toolchain, Pre defined configuration profiles, and sample applications are all in included.

Gateway Information

The DSGW-210 IoT Edge Computer Gateway is equipped with an ARM Cortex-A53 quad-core processor, 1GB DDR3 RAM, and 8GB eMMC flash memory. It also has a built-in Wi-Fi module, two Ethernet ports, and a USB 2.0 port for external devices.

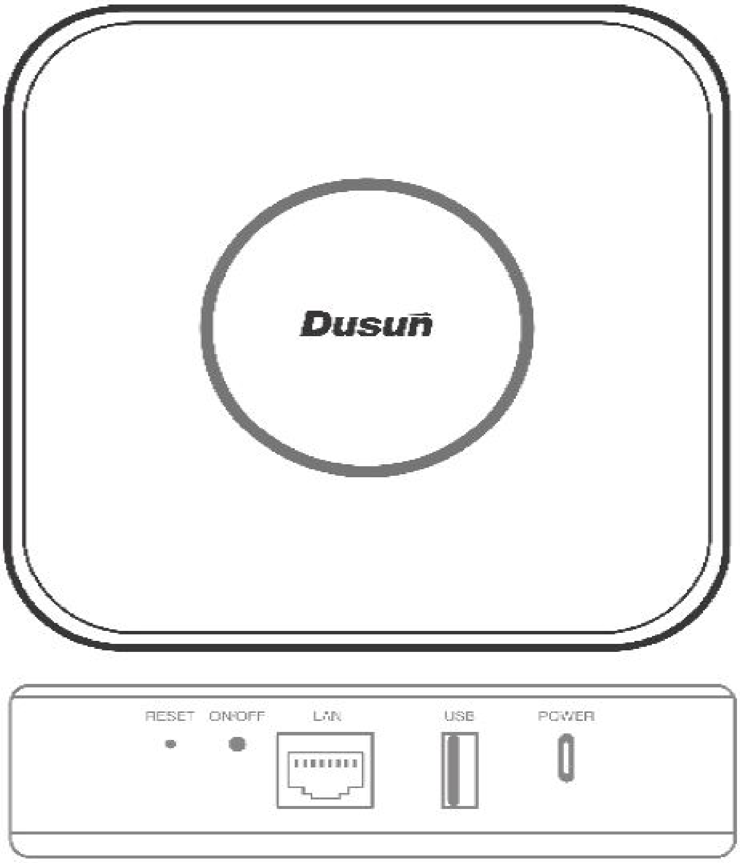
Basic Information

The gateway supports various protocols such as MQTT, CoAP, and HTTP. It also features a web-based management interface that allows users to configure and manage the gateway remotely.

- **SOC:** RK3328
  - Quad-core ARM Cortex-A53
  - Mali-450MP2 GPU
- **Power Supply:** DC-5V

- **LTE module:** BG96 (LET CAT-1)
- **Wi-Fi module:** 6221A (Wi-Fi chip: RTL8821CS)
- **Zigbee:** EFR32MG1B232F256GM32
- **Z-wave:** ZGM130S037HGN
- **Bluetooth:** EFR32BG21A020F768IM32
- **eMMC:** 8GB
- **SDRAM:** 2BG

Interface
The DSGW-210 IoT Edge Computer Gateway has the following interfaces:



- 2 Ethernet ports
- 1 USB 2.0 port
- Built-in Wi-Fi module

Target Setup

The DSGW-210 IoT Edge Computer Gateway can be set up as a target device for IoT development projects. This section describes how to connect the gateway into your host computer and network.

Connecting a gateway – Power

1. Make sure that the power adapter is 5V/3A.
2. Select the appropriate power plug adaptor for your geographical location. Insert it into the slot on the Universal Power Supply; then plug the power supply into an outlet.
3. Connect the output plug of the power supply to the gateway

Connecting a gateway – USB port

1. Connect one end of the USB cable to the USB port on the laptop or desktop
2. Connect the other end of USB cable to the USB port on the gateway.

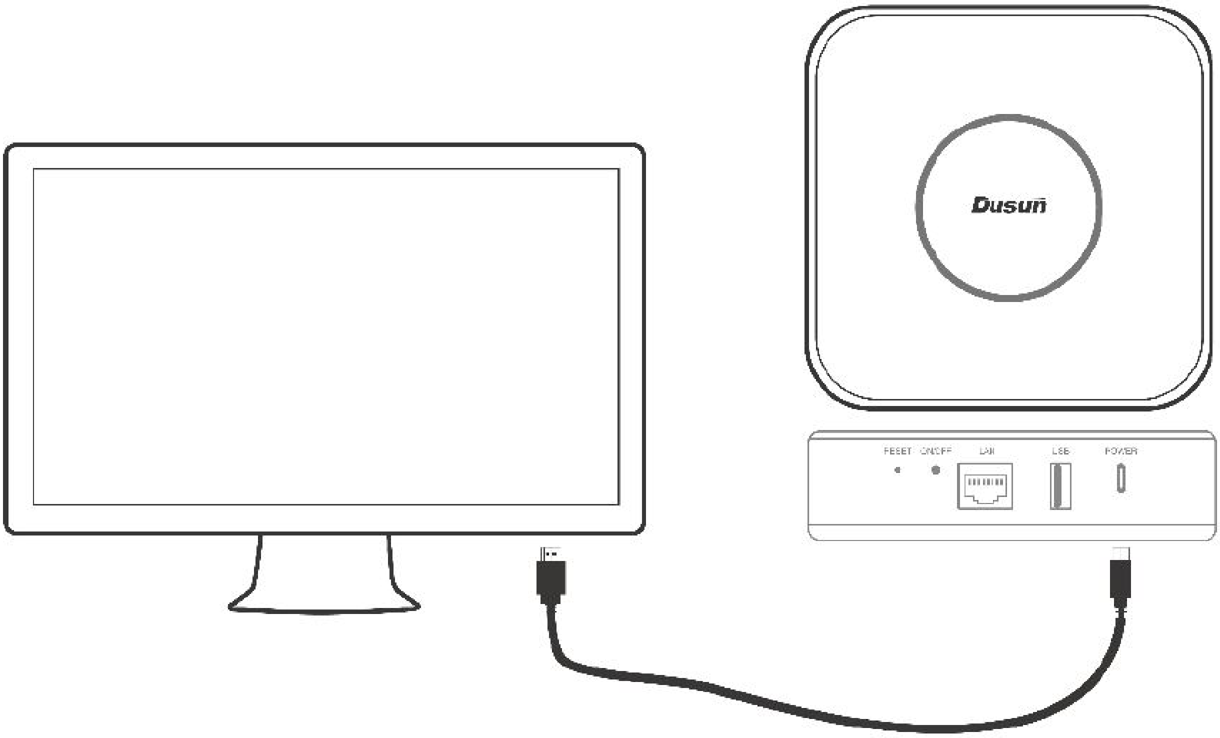
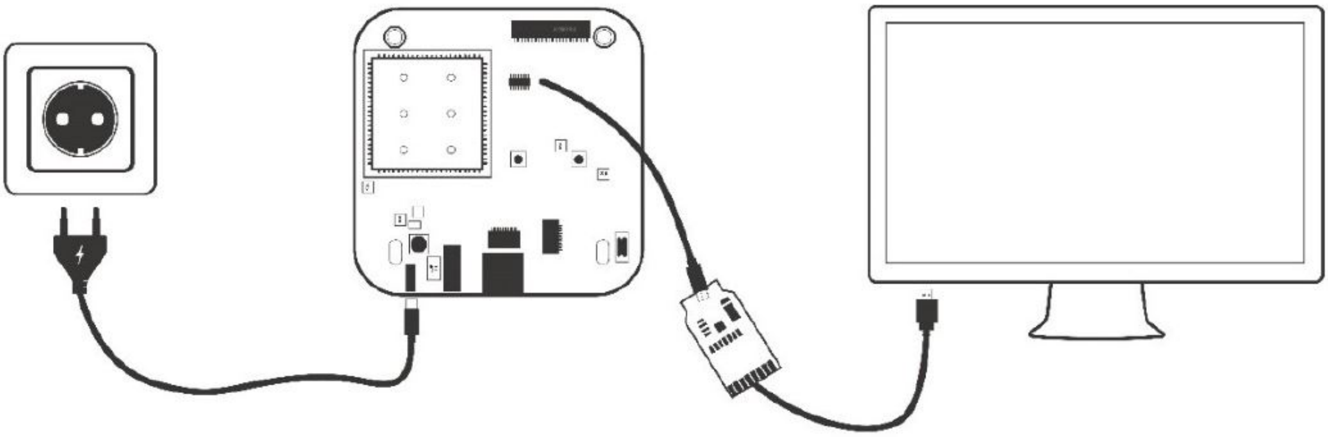


Figure3-1. Connecting a gateway via USB port

Connecting a PCBA board – Serial Port
If you want to debug the gateway, you can open the shell, Connect the PC to the PCBA board via Serial to USB tool.
PIN in board for serial connection: TP1100: RX TP1101: TX



Compile the Environment to Build

To start building IoT applications for the DSGW-210 IoT Edge Computer Gateway, you need to set up the development environment by following these steps:

Please use ubuntu 18.04 .iso image to setup your build environment. You can use a virtual machine or a physical

PC to install ubuntu 18.04.

- **Virtual Machine**
  It is recommended that novice users use virtual machines, install ubuntu 18.04 to the virtual machine, and leave enough disk space (at least 100G) for the virtual machine.
- **Ubuntu PC Compile the Environment to**
  The use of physical machine compilation users can use a ubuntu PC.

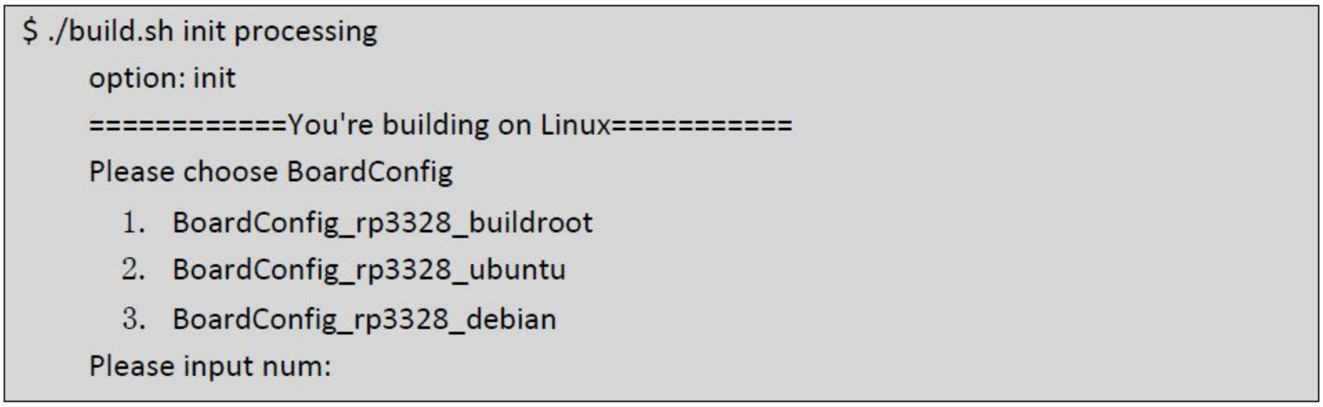
SDK Acquisition and Preparation

1. **Download the source code from the Dusun FTP**
   The source package name will be 3328-linux-*.tar.gz, get it from Dusun FTP.
2. **Code Compression Package Check**
   The next step can be taken only after generating the MD5 value of the source compression package and comparing the MD5 value of the MD5 .txt text to confirm that the MD5 value is the same, and if the MD5 value is not the same, the energy code pack is damaged, please download it again.
   $ md5sum rk3328-linux-*.tar.gz
3. **The Source Compression Package is Unzipped**
   Copy the source code to the corresponding directory and unzip the source code compression package.
   - $ sudo -i
   - $ mkdir workdir
   - $ cd workdir
   - $ tar -zxvf /path/to/rk3328-linux-*.tar.gz
   - $ cd rk3328-linux

Code Compilation

Getting started, global Compilation

1. Initialize Compilation Environment Variables (select file system)
   You can build buildroot, ubuntu or debian rootfs image. Select it in “./build.sh init”.



```
$ ./build.sh init processing
      option: init
      ===========You're building on Linux==========
      Please choose BoardConfig
         1.  BoardConfig_rp3328_buildroot
         2.  BoardConfig_rp3328_ubuntu
         3.  BoardConfig_rp3328_debian
      Please input num:
```

   We strongly recommend you to build and run the system with buildroot rootfs to get familiar with the hardware and build environment, when you get started. After you tried buildroot system, you can try ubuntu and debian system.

2. **Prepare the Root File System base**

   This section is for building ubuntu or debian file system. If you want to build the buildroot file system, skip this section.

   Compile Ubuntu

   Download the root file system compression package ubuntu.tar.gz The Root file system compresses the package directory: Unzip the compression package

   ```
   $ tar -zxvf ubuntu.tar.gz // you get ubuntu.img
   ```

   Copy the root file system to the specified path

   ```
   $ cd workdir/rk3328-linux
   $ mkdir ubuntu
   $ cp /path/to/ubuntu.img ./ubuntu/
   ```

   Compile Debian

   Download the root file system compression package debian.tar.gz Unzip the compression package

   ```
   $ tar -zxvf debian.tar.gz // you get linaro-rootfs.img
   ```

   Copy the root file system to the specified path

   ```
   $ cd workdir/rk3328-linux
   $ mkdir debian
   $ cp ./linaro-rootfs.img ./debian/
   ```

3. **Start Compiling**

   ```
   $ ./build.sh
   ```

   Build a complete directory of firmware files: rockdev/update.img and other separate images, update.img includes all firmware for full upgrade.

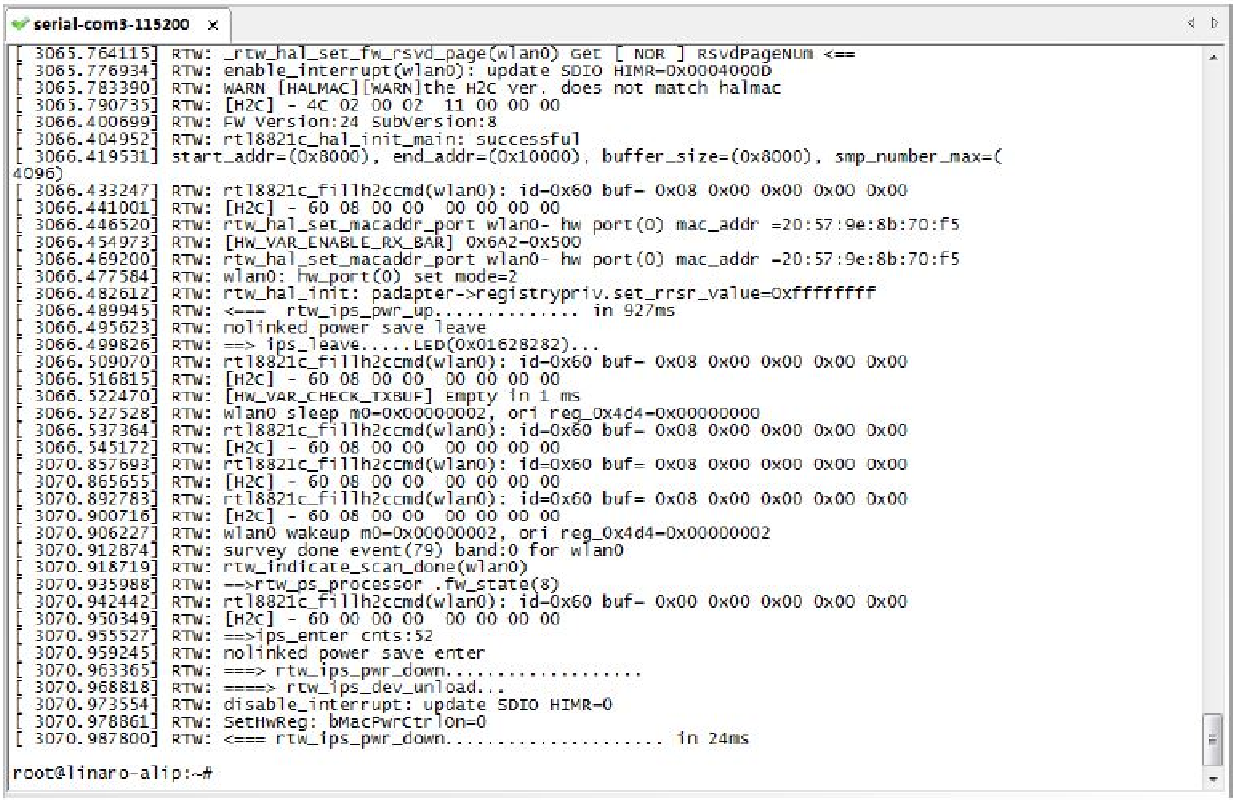
4. **Run The Image on the board**

   Connect the RK3328 board serial port to the PC via a USB to UART Bridge. Use Putty or other Terminal software as your console tool,

   SERIAL CONSOLE SETTINGS:

   - 115200/8N1
   - Baud: 115200
   - Data Bits: 8
   - Parity Bit: No
   - Stop Bit: 1

     Power UP the board, you can see the boot log on console:



```
[ 3065.764115] RTW: _rtw_hal_set_fw_rsvd_page(wlan0) Get [ NOR ] RsvdPageNum <==
[ 3065.776934] RTW: enable_interrupt(wlan0): update SDIO HIMR=0x0004000D
[ 3065.783390] RTW: WARN [HALMAC][WARN]the H2C ver. does not match halmac
[ 3065.790735] RTW: [H2C] - 4c 02 00 02  11 00 00 00
[ 3066.400699] RTW: FW Version:24 Subversion:8
[ 3066.404952] RTW: rtl8821c_hal_init_main: successful
[ 3066.419531] start_addr=(0x8000), end_addr=(0x10000), buffer_size=(0x8000), smp_number_max=(
4096)
[ 3066.433247] RTW: rtl8821c_fillh2ccmd(wlan0): id=0x60 buf= 0x08 0x00 0x00 0x00 0x00
[ 3066.441001] RTW: [H2C] - 60 08 00 00  00 00 00 00
[ 3066.446520] RTW: rtw_hal_set_macaddr_port wlan0- hw port(0) mac_addr =20:57:9e:8b:70:f5
[ 3066.454973] RTW: [HW_VAR_ENABLE_RX_BAR] 0x6A2=0x500
[ 3066.469200] RTW: rtw_hal_set_macaddr_port wlan0- hw port(0) mac_addr =20:57:9e:8b:70:f5
[ 3066.477584] RTW: wlan0: hw_port(0) set mode=2
[ 3066.482612] RTW: rtw_hal_init: padapter->registrypriv.set_rrsr_value=0xffffffff
[ 3066.489945] RTW: <===  rtw_ips_pwr_up.............. in 927ms
[ 3066.495623] RTW: nolinked power save leave
[ 3066.499826] RTW: ==> ips_leave.....LED(0x01628282)...
[ 3066.509070] RTW: rtl8821c_fillh2ccmd(wlan0): id=0x60 buf= 0x08 0x00 0x00 0x00 0x00
[ 3066.516815] RTW: [H2C] - 60 08 00 00  00 00 00 00
[ 3066.522470] RTW: [HW_VAR_CHECK_TXBUF] Empty in 1 ms
[ 3066.527528] RTW: wlan0 sleep m0=0x00000002, ori reg_0x4d4=0x00000000
[ 3066.537364] RTW: rtl8821c_fillh2ccmd(wlan0): id=0x60 buf= 0x08 0x00 0x00 0x00 0x00
[ 3066.545172] RTW: [H2C] - 60 08 00 00  00 00 00 00
[ 3070.857693] RTW: rtl8821c_fillh2ccmd(wlan0): id=0x60 buf= 0x08 0x00 0x00 0x00 0x00
[ 3070.865655] RTW: [H2C] - 60 08 00 00  00 00 00 00
[ 3070.892783] RTW: rtl8821c_fillh2ccmd(wlan0): id=0x60 buf= 0x08 0x00 0x00 0x00 0x00
[ 3070.900716] RTW: [H2C] - 60 08 00 00  00 00 00 00
[ 3070.906227] RTW: wlan0 wakeup m0=0x00000002, ori reg_0x4d4=0x00000002
[ 3070.912874] RTW: survey done event(79) band:0 for wlan0
[ 3070.918719] RTW: rtw_indicate_scan_done(wlan0)
[ 3070.935988] RTW: -->rtw_ps_processor .fw_state(8)
[ 3070.942442] RTW: rtl8821c_fillh2ccmd(wlan0): id=0x60 buf= 0x00 0x00 0x00 0x00 0x00
[ 3070.950349] RTW: [H2C] - 60 00 00 00  00 00 00 00
[ 3070.955527] RTW: ==>ips_enter cnts:52
[ 3070.959245] RTW: nolinked power save enter
[ 3070.963365] RTW: ===> rtw_ips_pwr_down...................
[ 3070.968818] RTW: ====> rtw_ips_dev_unload...
[ 3070.973554] RTW: disable_interrupt: update SDIO HIMR=0
[ 3070.978861] RTW: SetHwReg: bMacPwrCtrlOn=0
[ 3070.987800] RTW: <=== rtw_ips_pwr_down..................... in 24ms

root@linaro-alip:~#
```

Compiled Each Image Part Separately

1. **The build system and the image structure**
   The update.img is composed of several parts. Main parts are uboot.img, boot.img, recovery.img, rootfs.img. uboot.img contains bootloader uboot boot.img contains the device tree .dtb image, Linux kernel image recovery.img: The system can boot up to recovery mode, recovery.img is the rootfs used in recovery mode. rootfs.img: The normal rootfs image. In normal mode, system boot and mount this rootfs image. You may need to build the images separately, especially when you focus on single module (e.g. uboot or kernel driver) development. Then you can build only that part of image and update that partition in flash.
2. **Build Uboot only**
   $ ./build.sh uboot
3. **Build Linux Kernel Only**
   $ ./build.sh kernel
4. **Build Recovery File System Only**
   $ ./build.sh recovery
5. **Build File System Only**
   $ ./build.sh rootfs
6. **Final Image Packaging**
   $ ./build.sh updateimg

This command making rockdev/*.img scatter firmware packaging builds in the directory update.img

More about buildroot system

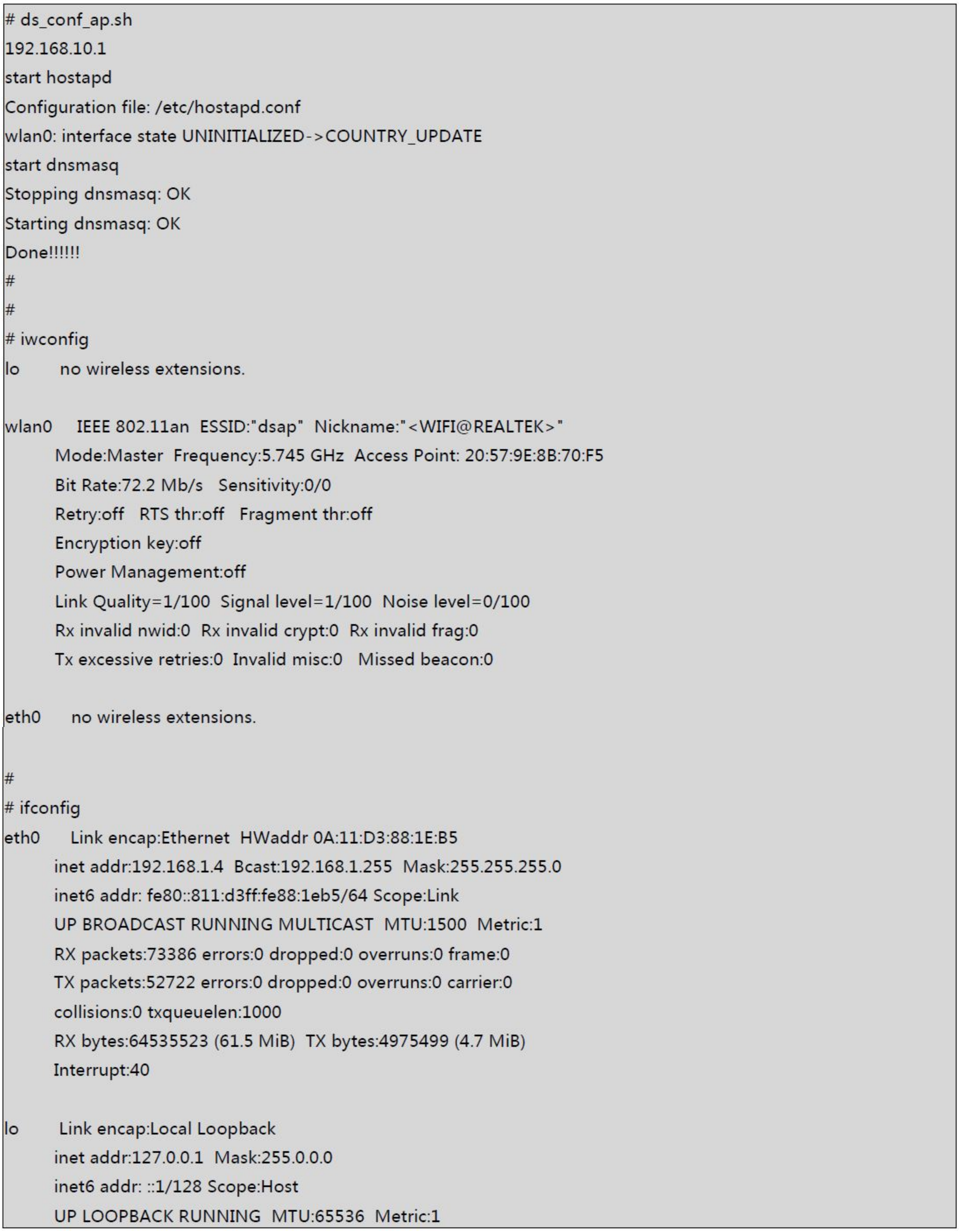
If you use buildroot rootfs, some Dusun test scripts/tools are already installed in the final buildroot rootfs. You can refer to buildroot/dusun_rootfs/add_ds_rootfs.sh

Test hardware components

The following testing are done under the buildroot system.

1. **Test Wi-Fi as AP**

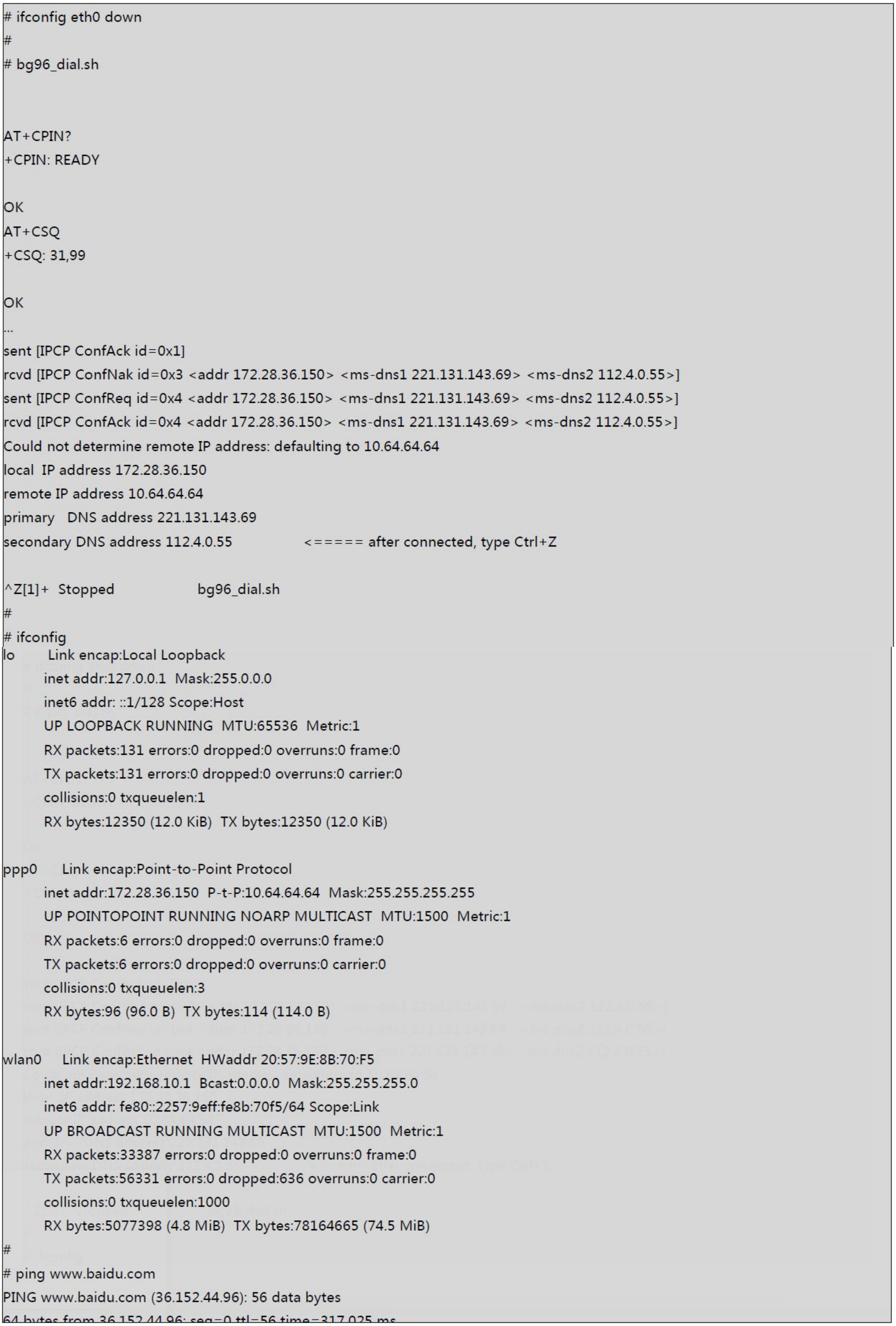
   The “ds_conf_ap.sh” script is for setting up Wi-Fi AP, SSID is “dsap”, password is “12345678”.



```
# ds_conf_ap.sh
192.168.10.1
start hostapd
Configuration file: /etc/hostapd.conf
wlan0: interface state UNINITIALIZED->COUNTRY_UPDATE
start dnsmasq
Stopping dnsmasq: OK
Starting dnsmasq: OK
Done!!!!!
#
#
# iwconfig
lo        no wireless extensions.

wlan0     IEEE 802.11an  ESSID:"dsap"  Nickname:"<WIFI@REALTEK>"
          Mode:Master  Frequency:5.745 GHz  Access Point: 20:57:9E:8B:70:F5
          Bit Rate:72.2 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=1/100  Signal level=1/100  Noise level=0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

eth0      no wireless extensions.

#
# ifconfig
eth0      Link encap:Ethernet  HWaddr 0A:11:D3:88:1E:B5
          inet addr:192.168.1.4  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::811:d3ff:fe88:1eb5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:73386 errors:0 dropped:0 overruns:0 frame:0
          TX packets:52722 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:64535523 (61.5 MiB)  TX bytes:4975499 (4.7 MiB)
          Interrupt:40

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
```

2. **Test BG96**

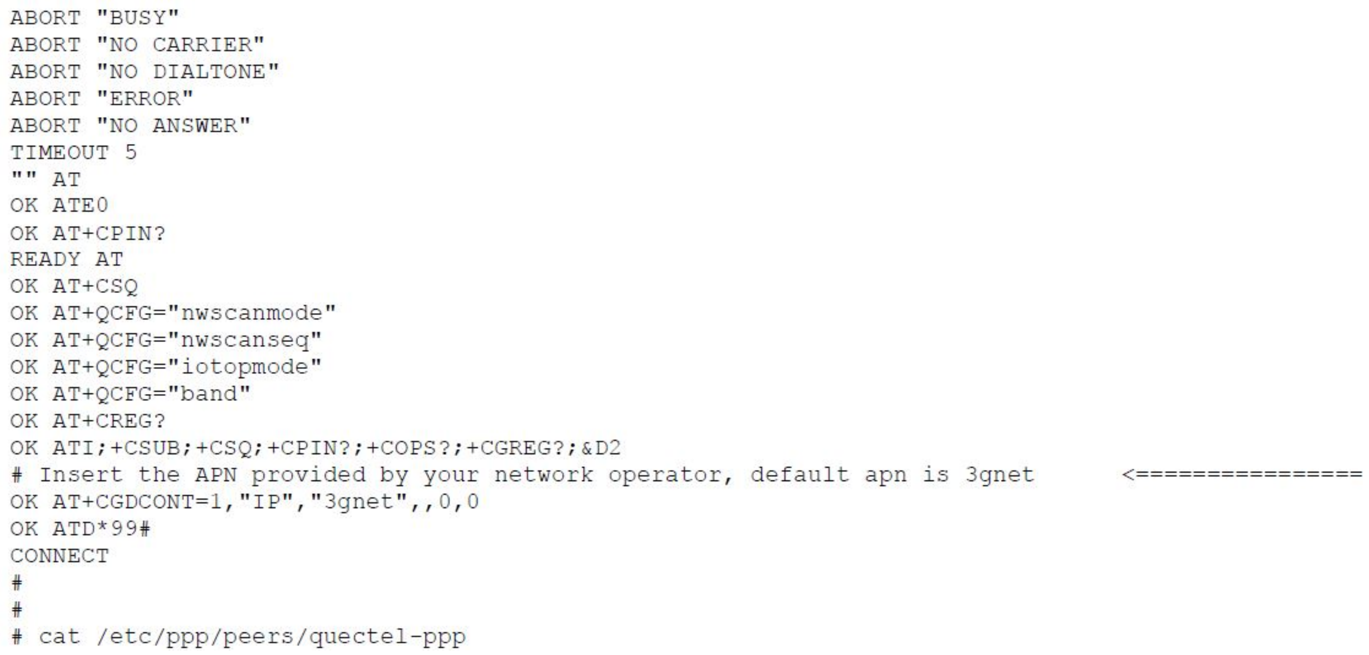
   bg96_dial.sh is used for BG96 dial.



```
# ifconfig eth0 down
#
# bg96_dial.sh


AT+CPIN?
+CPIN: READY

OK
AT+CSQ
+CSQ: 31,99

OK
...
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfNak id=0x3 <addr 172.28.36.150> <ms-dns1 221.131.143.69> <ms-dns2 112.4.0.55>]
sent [IPCP ConfReq id=0x4 <addr 172.28.36.150> <ms-dns1 221.131.143.69> <ms-dns2 112.4.0.55>]
rcvd [IPCP ConfAck id=0x4 <addr 172.28.36.150> <ms-dns1 221.131.143.69> <ms-dns2 112.4.0.55>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 172.28.36.150
remote IP address 10.64.64.64
primary   DNS address 221.131.143.69
secondary DNS address 112.4.0.55              <===== after connected, type Ctrl+Z

^Z[1]+  Stopped                  bg96_dial.sh
#
# ifconfig
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:131 errors:0 dropped:0 overruns:0 frame:0
          TX packets:131 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:12350 (12.0 KiB)  TX bytes:12350 (12.0 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:172.28.36.150  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:96 (96.0 B)  TX bytes:114 (114.0 B)

wlan0     Link encap:Ethernet  HWaddr 20:57:9E:8B:70:F5
          inet addr:192.168.10.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::2257:9eff:fe8b:70f5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:33387 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56331 errors:0 dropped:636 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5077398 (4.8 MiB)  TX bytes:78164665 (74.5 MiB)
#
# ping www.baidu.com
PING www.baidu.com (36.152.44.96): 56 data bytes
64 bytes from 36.152.44.96: seq=0 ttl=56 time=317.025 ms
```

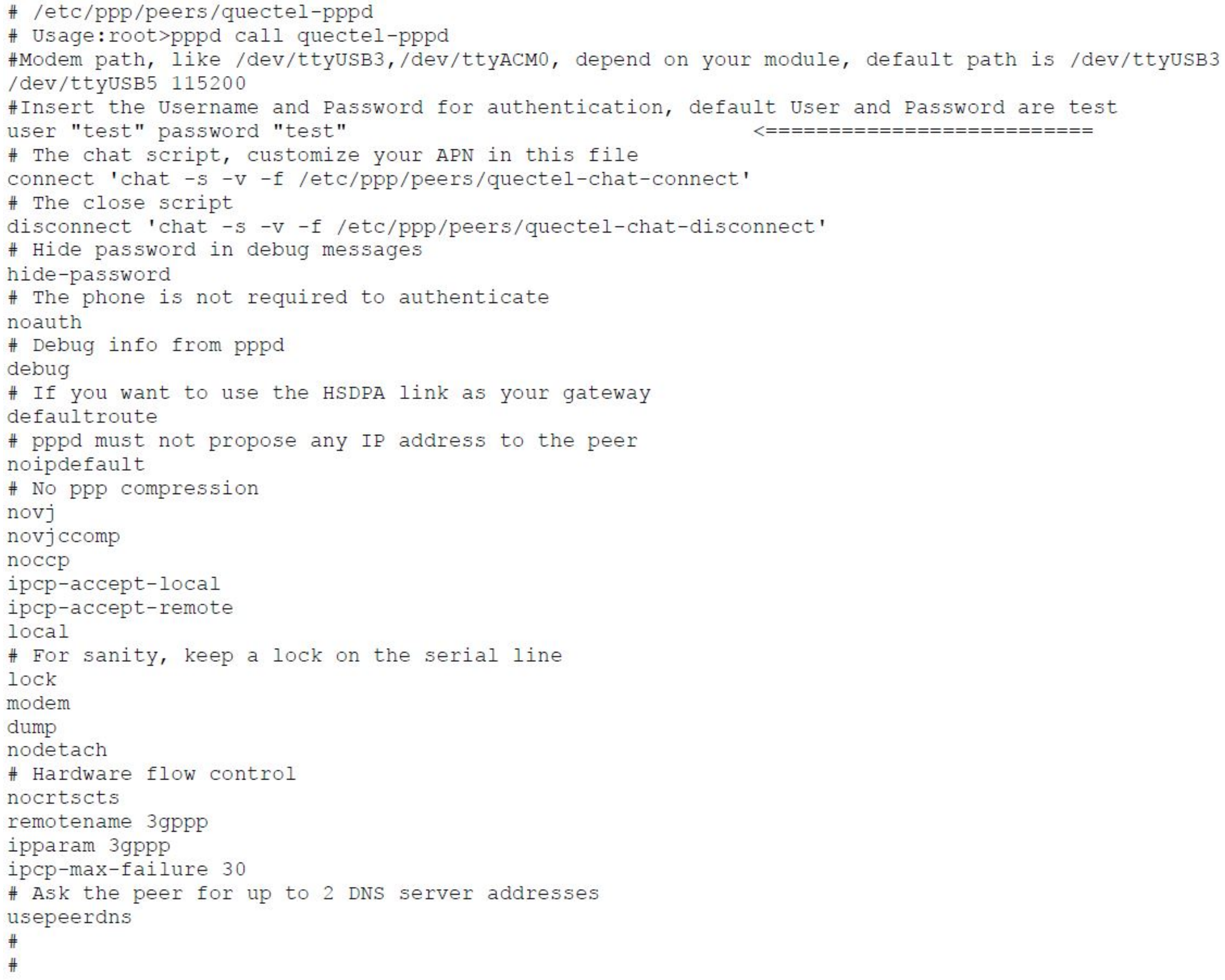
You need to configure APN, username/password for BG96, in quectel-chat-connect and quectel-ppp file. Before you run the test.

# cat /etc/ppp/peers/quectel-chat-connect



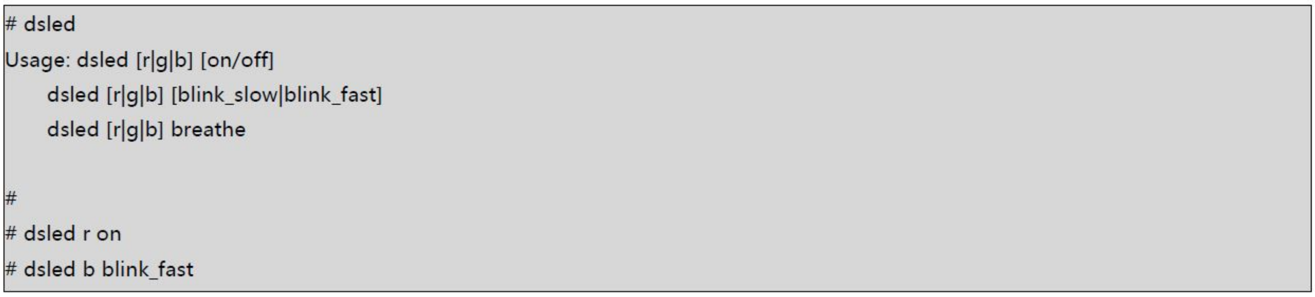
```
ABORT "BUSY"
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
TIMEOUT 5
"" AT
OK ATE0
OK AT+CPIN?
READY AT
OK AT+CSQ
OK AT+QCFG="nwscanmode"
OK AT+QCFG="nwscanseq"
OK AT+QCFG="iotopmode"
OK AT+QCFG="band"
OK AT+CREG?
OK ATI;+CSUB;+CSQ;+CPIN?;+COPS?;+CGREG?;&D2
# Insert the APN provided by your network operator, default apn is 3gnet      <================
OK AT+CGDCONT=1,"IP","3gnet",,0,0
OK ATD*99#
CONNECT
#
#
# cat /etc/ppp/peers/quectel-ppp
```

# cat /etc/ppp/peers/quectel-ppp



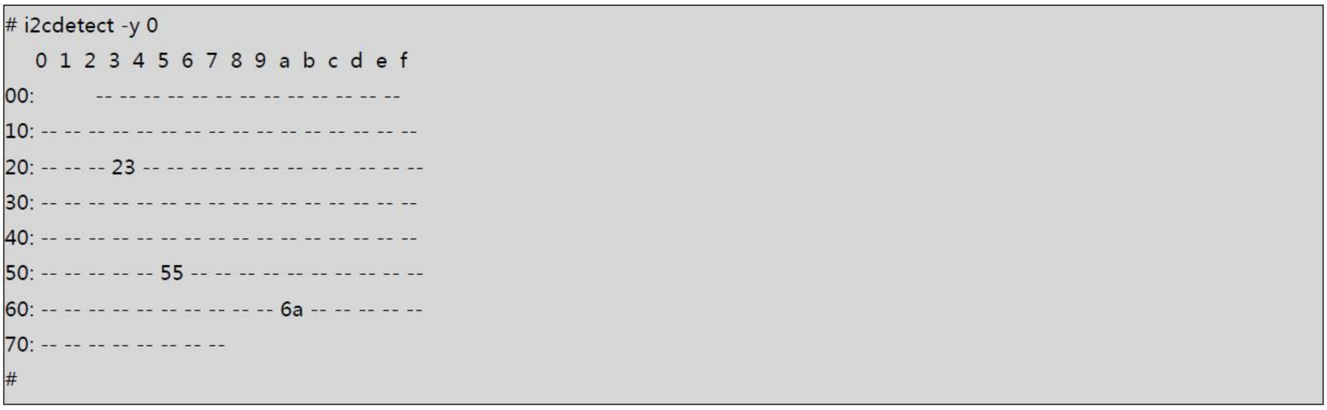
```
# /etc/ppp/peers/quectel-pppd
# Usage:root>pppd call quectel-pppd
#Modem path, like /dev/ttyUSB3,/dev/ttyACM0, depend on your module, default path is /dev/ttyUSB3
/dev/ttyUSB5 115200
#Insert the Username and Password for authentication, default User and Password are test
user "test" password "test"                             <==========================
# The chat script, customize your APN in this file
connect 'chat -s -v -f /etc/ppp/peers/quectel-chat-connect'
# The close script
disconnect 'chat -s -v -f /etc/ppp/peers/quectel-chat-disconnect'
# Hide password in debug messages
hide-password
# The phone is not required to authenticate
noauth
# Debug info from pppd
debug
# If you want to use the HSDPA link as your gateway
defaultroute
# pppd must not propose any IP address to the peer
noipdefault
# No ppp compression
novj
novjccomp
noccp
ipcp-accept-local
ipcp-accept-remote
local
# For sanity, keep a lock on the serial line
lock
modem
dump
nodetach
# Hardware flow control
nocrtscts
remotename 3gppp
ipparam 3gppp
ipcp-max-failure 30
# Ask the peer for up to 2 DNS server addresses
usepeerdns
#
#
```

- **Test LED**



```
# dsled
Usage: dsled [r|g|b] [on/off]
     dsled [r|g|b] [blink_slow|blink_fast]
     dsled [r|g|b] breathe


#
# dsled r on
# dsled b blink_fast
```

- **Test I2C**

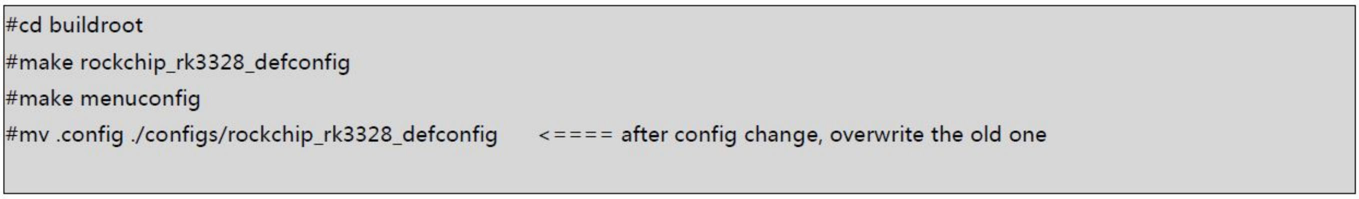


```
# i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- 23 -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- 55 -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- 6a -- -- -- -- --
70: -- -- -- -- -- -- -- --
#
```

Actually LED controlled is I2C interface.

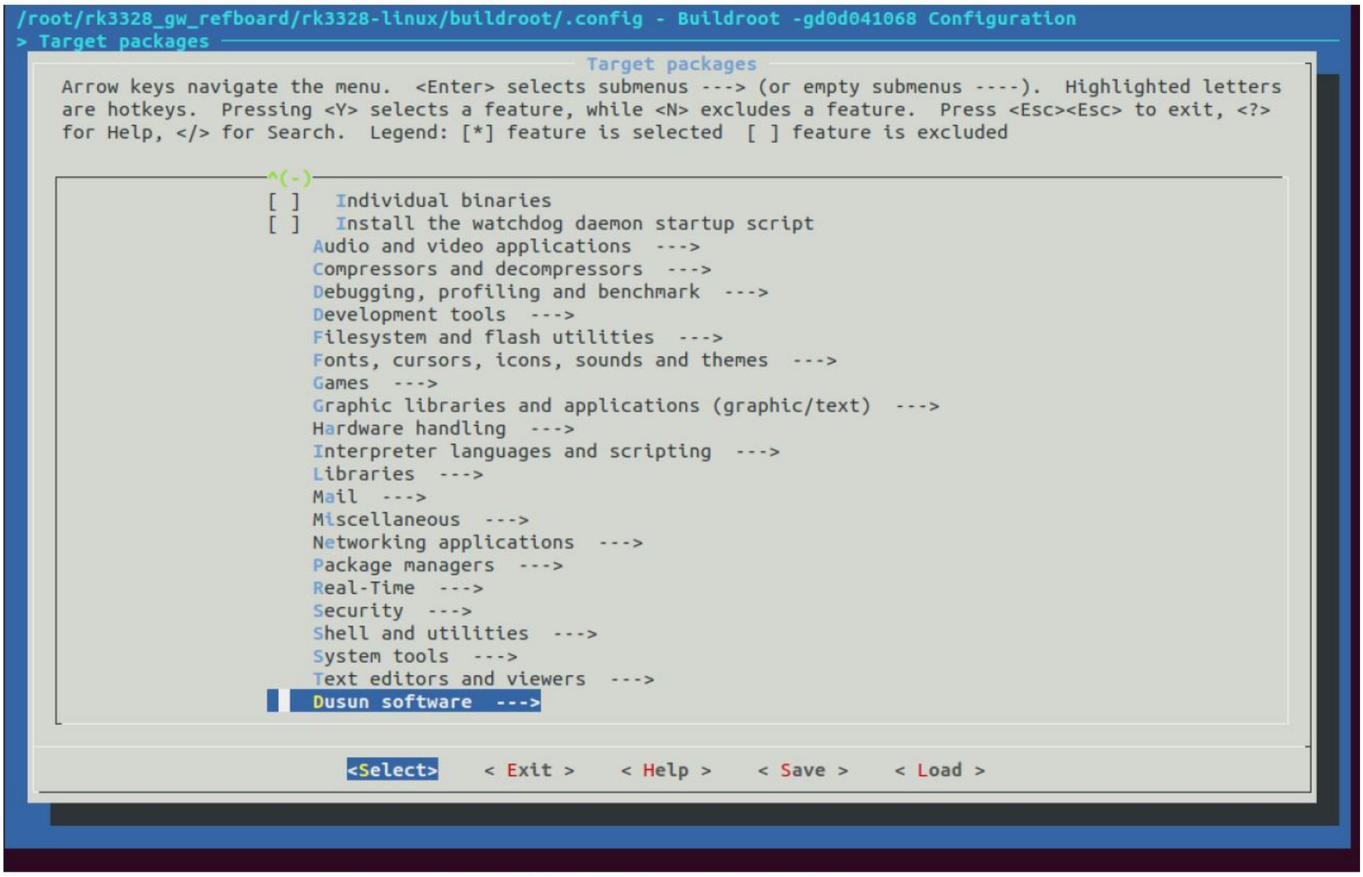
How to make menuconfig in buildroot

Normal mode buildroot rootfs config file: buildroot/configs/rockchip_rk3328_defconfig Recovery mode buildroot rootfs config file: buildroot/configs/rockchip_rk3328_recovery_defconfig



```
#cd buildroot
#make rockchip_rk3328_defconfig
#make menuconfig
#mv .config ./configs/rockchip_rk3328_defconfig     <==== after config change, overwrite the old one
```

If you want to change buildroot configure, here are the steps:

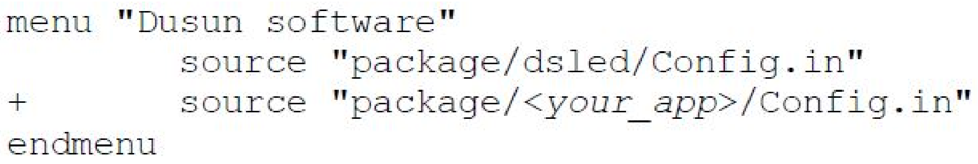


```
/root/rk3328_gw_refboard/rk3328-linux/buildroot/.config - Buildroot -gd0d041068 Configuration
> Target packages ─────────────────────────────────────────────────────────────────────
                                     Target packages
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters
   are hotkeys.  Pressing <Y> selects a feature, while <N> excludes a feature.  Press <Esc><Esc> to exit, <?>
   for Help, </> for Search.  Legend: [*] feature is selected  [ ] feature is excluded

                        ^(-)
                        [ ]   Individual binaries
                        [ ]   Install the watchdog daemon startup script
                            Audio and video applications  --->
                            Compressors and decompressors  --->
                            Debugging, profiling and benchmark  --->
                            Development tools  --->
                            Filesystem and flash utilities  --->
                            Fonts, cursors, icons, sounds and themes  --->
                            Games  --->
                            Graphic libraries and applications (graphic/text)  --->
                            Hardware handling  --->
                            Interpreter languages and scripting  --->
                            Libraries  --->
                            Mail  --->
                            Miscellaneous  --->
                            Networking applications  --->
                            Package managers  --->
                            Real-Time  --->
                            Security  --->
                            Shell and utilities  --->
                            System tools  --->
                            Text editors and viewers  --->
                            Dusun software  --->

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

Figure3-1.buildroot make menuconfig

How to add an application in buildroot source tree

1. Make directory buildroot/dusun_package/<your_app>
2. Put APP source code files and Makefile to buildroot/dusun_package/< your_app > your_app.h your_app.c Makefile
3. Make directory buildroot/package/< your_app > Config.in your_app.mk
4. Add Config.in sourcing in buildroot/package/Config.in



```
menu "Dusun software"
        source "package/dsled/Config.in"
+       source "package/<your_app>/Config.in"
endmenu
```

5. Make menuconfig to select your APP, and save the configure file as 5.2.
6. “./build.sh rootfs” to rebuild rootfs Please refer to buildroot/dusun_package/dsled/, it’s a useful example.

Switch to ubuntu or debian system

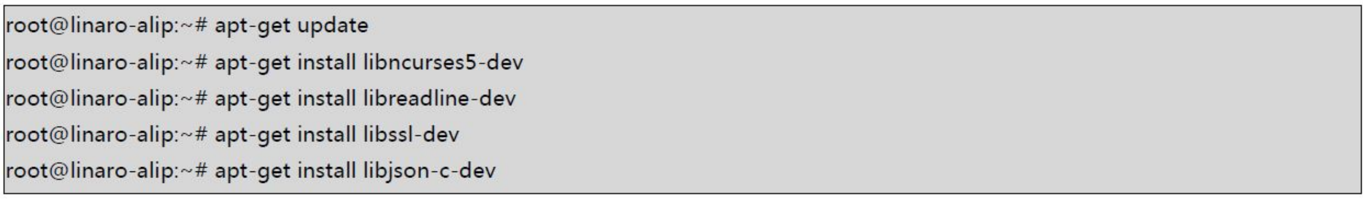
If you have built a buildroot system image, and want to switch to ubuntu or debian image. You don’t need to clean the make and do a clean rebuild. Just do the following steps:

1. “./build.sh init” to select ubuntu or debian
2. “./build.sh rootfs” to rebuild ubuntu or debian rootfs
3. “./build.sh” to build a the final update.img

Be careful, the dusun tools and scripts are default copied to buildroot rootfs, not to ubuntu or debian rootfs. If you want to copy them to ubuntu or debian rootfs, you can modify the buildroot/dusun_rootfs/add_ds_rootfs.sh. For the APPs, you can copy the code to the board and build it on the target board ubuntu or debian system, since it has gcc and other toolchains.

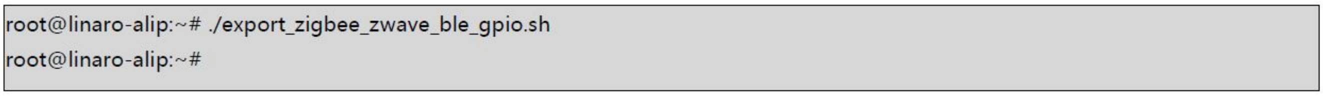
Wireless development (Zigbee, Z-Wave, BLE, LoRaWAN)

Please build a debian system to do the following steps. The code will be compiled on the board, not on host.



```
root@linaro-alip:~# apt-get update
root@linaro-alip:~# apt-get install libncurses5-dev
root@linaro-alip:~# apt-get install libreadline-dev
root@linaro-alip:~# apt-get install libssl-dev
root@linaro-alip:~# apt-get install libjson-c-dev
```

1. Prepare some library on the board
2. scp SDK “buildroot/dusun_rootfs/target_scripts/export_zigbee_zwave_ble_gpio.sh” from host to board, under /root

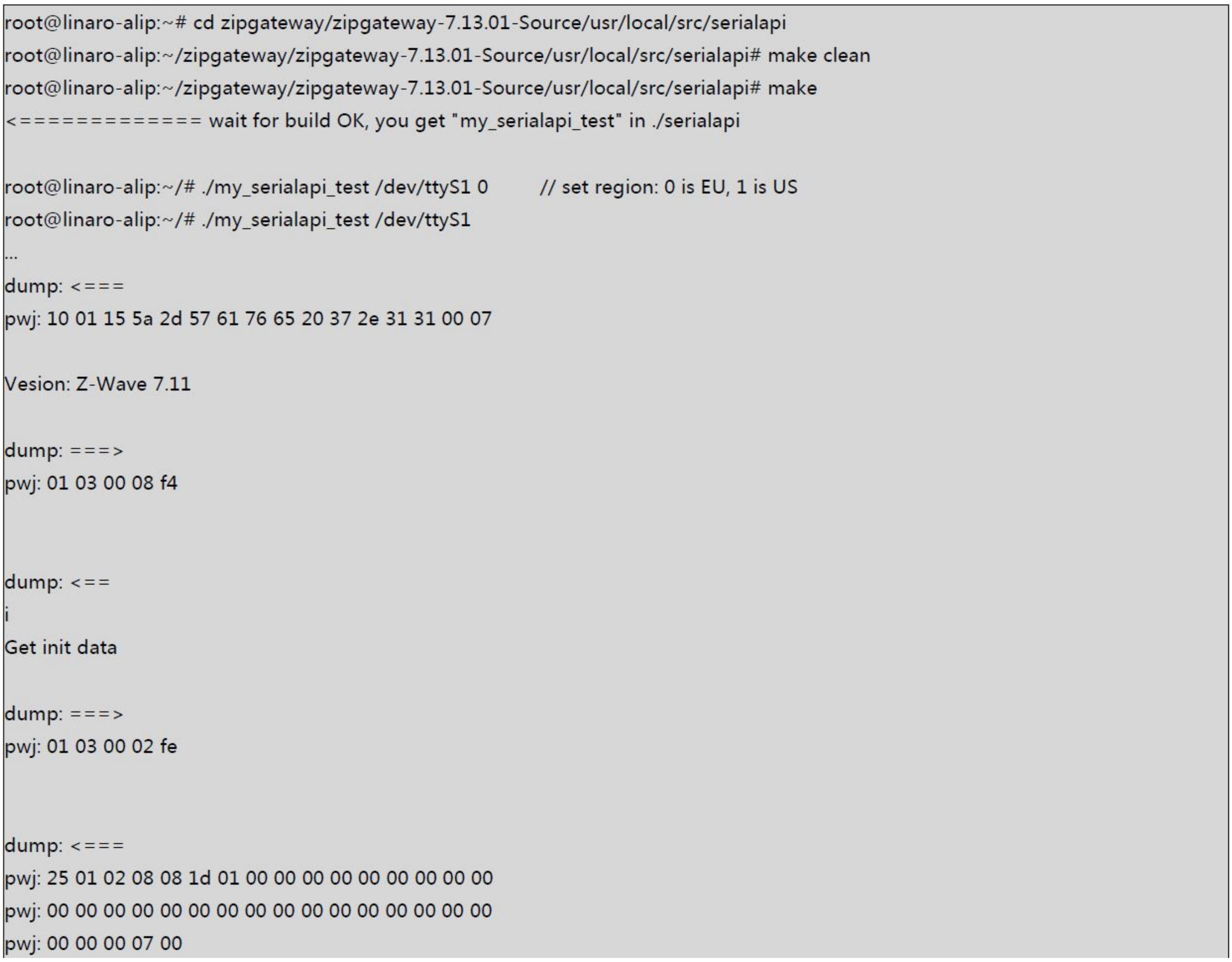


```
root@linaro-alip:~# ./export_zigbee_zwave_ble_gpio.sh
root@linaro-alip:~#
```

3. Power on wireless modules on board.

Zigbee

Zigbee interface is /dev/ttyUSB0. Download “Z3GatewayHost_EFR32MG12P433F1024GM48.tar.gz” from Dusun FTP, and copy it to board, under /root.

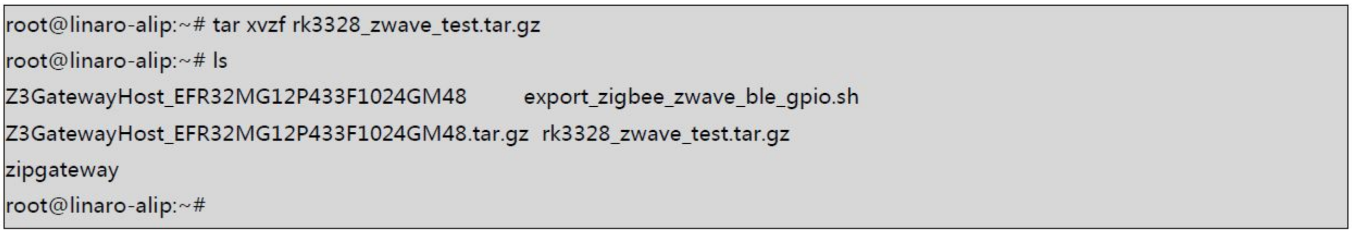


```
root@linaro-alip:~# cd zipgateway/zipgateway-7.13.01-Source/usr/local/src/serialapi
root@linaro-alip:~/zipgateway/zipgateway-7.13.01-Source/usr/local/src/serialapi# make clean
root@linaro-alip:~/zipgateway/zipgateway-7.13.01-Source/usr/local/src/serialapi# make
<============== wait for build OK, you get "my_serialapi_test" in ./serialapi

root@linaro-alip:~/# ./my_serialapi_test /dev/ttyS1 0        // set region: 0 is EU, 1 is US
root@linaro-alip:~/# ./my_serialapi_test /dev/ttyS1
...
dump: <===
pwj: 10 01 15 5a 2d 57 61 76 65 20 37 2e 31 31 00 07

Vesion: Z-Wave 7.11

dump: ===>
pwj: 01 03 00 08 f4


dump: <==
i
Get init data

dump: ===>
pwj: 01 03 00 02 fe


dump: <===
pwj: 25 01 02 08 08 1d 01 00 00 00 00 00 00 00 00 00
pwj: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
pwj: 00 00 00 07 00
```

Then build Z3Gateway and run. For more information about Z3Gateway, please visit https://docs.silabs.com/ for more information.

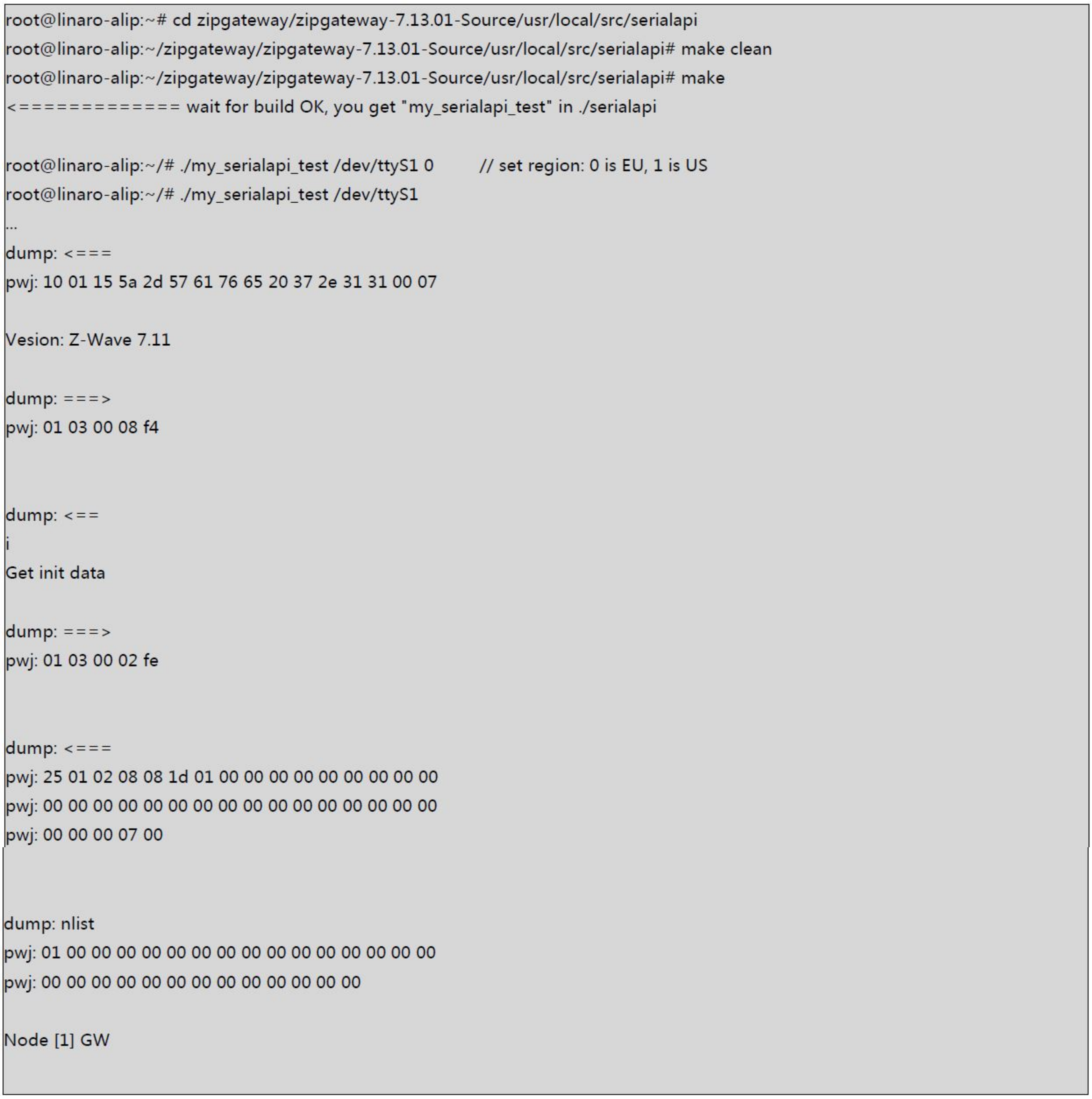
Z-Wave

Z-Wave interface is /dev/ttyS1. Download ” rk3328_zwave_test.tar.gz ” from Dusun FTP, and copy it to board, under /root.



```
root@linaro-alip:~# tar xvzf rk3328_zwave_test.tar.gz
root@linaro-alip:~# ls
Z3GatewayHost_EFR32MG12P433F1024GM48        export_zigbee_zwave_ble_gpio.sh
Z3GatewayHost_EFR32MG12P433F1024GM48.tar.gz  rk3328_zwave_test.tar.gz
zipgateway
root@linaro-alip:~#
```

Unzip it and you can get ./zipgateway



```
root@linaro-alip:~# cd zipgateway/zipgateway-7.13.01-Source/usr/local/src/serialapi
root@linaro-alip:~/zipgateway/zipgateway-7.13.01-Source/usr/local/src/serialapi# make clean
root@linaro-alip:~/zipgateway/zipgateway-7.13.01-Source/usr/local/src/serialapi# make
<============= wait for build OK, you get "my_serialapi_test" in ./serialapi

root@linaro-alip:~/# ./my_serialapi_test /dev/ttyS1 0        // set region: 0 is EU, 1 is US
root@linaro-alip:~/# ./my_serialapi_test /dev/ttyS1
...
dump: <===
pwj: 10 01 15 5a 2d 57 61 76 65 20 37 2e 31 31 00 07

Vesion: Z-Wave 7.11

dump: ===>
pwj: 01 03 00 08 f4



dump: <==
i
Get init data

dump: ===>
pwj: 01 03 00 02 fe



dump: <===
pwj: 25 01 02 08 08 1d 01 00 00 00 00 00 00 00 00 00
pwj: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
pwj: 00 00 00 07 00



dump: nlist
pwj: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
pwj: 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Node [1] GW
```

Now build a zwave simple test tool and run: In “my_serialapi_test”, press ‘a’ to include zwave device, ‘r’ to exclude device, ‘d’ to reset to default, ‘i’ to get devices list and ‘q’ to quit. The Zipgateway is siliabs software, “my_serialapi_test” is just a very simple tool. For more information about Zipgateway, please visit https://docs.silabs.com/ for more information.

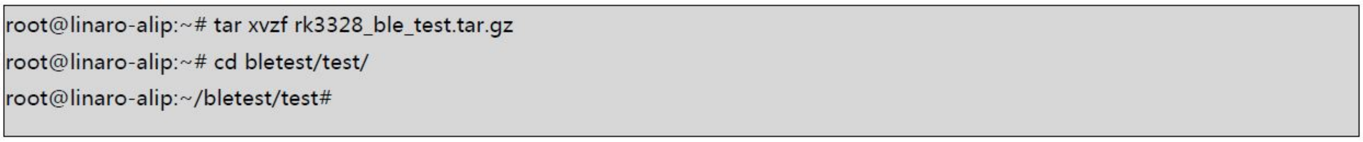
Z-Wave region

If for default Dusun built, Z-Wave frequency can be configured in /etc/config/dusun/zwave/region Default is 0x00: EU

0x01 – US	0x02 – ANZ	0x03 – HK	0x04 – Malaysia
0x05 – India	0x06 – Israel	0x07 – Russia	0x08 – China
0x20 – Japan	0x21 – Korea		

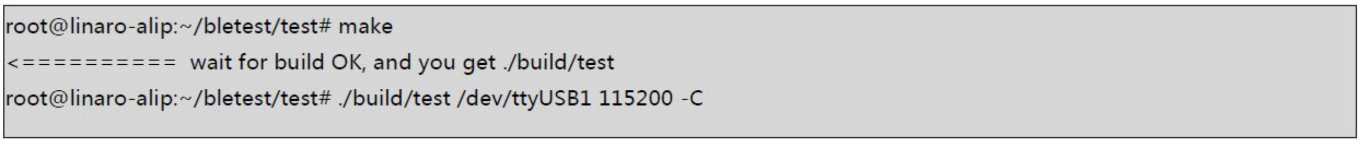
BLE

BLE interface is /dev/ttyUSB1. Download “rk3328_ble_test.tar.gz” from Dusun FTP, and copy it to board, under /root.



```
root@linaro-alip:~# tar xvzf rk3328_ble_test.tar.gz
root@linaro-alip:~# cd bletest/test/
root@linaro-alip:~/bletest/test#
```

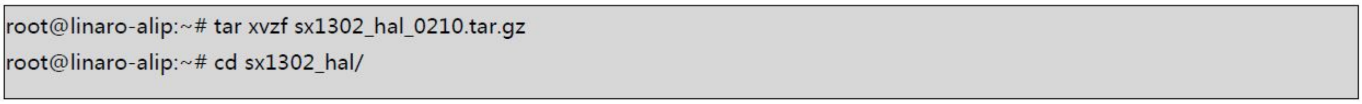
Unzip it and you can get ./bletest build ble test tool and run: More information about the BLE test tool, please visit https://docs.silabs.com/ for more information.



```
root@linaro-alip:~/bletest/test# make
<==========  wait for build OK, and you get ./build/test
root@linaro-alip:~/bletest/test# ./build/test /dev/ttyUSB1 115200 -C
```

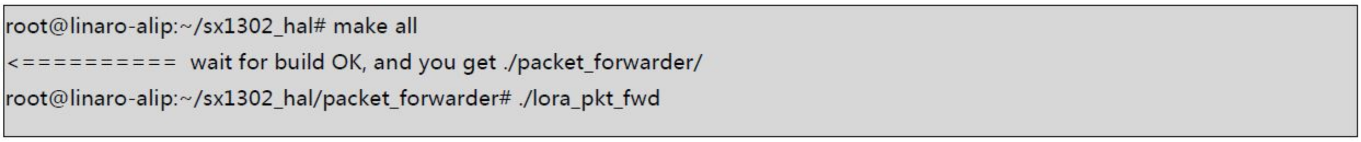
LoRaWAN

Choose the correct interface for LoRaWAN, for example /dev/spidev32766.0. The configuration file for it is in ./sx1302_hal/packet_forwarder/global_conf.json. Download “sx1302_hal_0210.tar.gz” from Dusun FTP, and copy it to board, under /root.



```
root@linaro-alip:~# tar xvzf sx1302_hal_0210.tar.gz
root@linaro-alip:~# cd sx1302_hal/
```

Untar it and you can get ./sx1302_hal build LoRaWAN sample code sx1302_hal and run: More information about the LoRaWAN code, please visit https://www.semtech.com/products/wireless-rf/lora-core/sx1302 for more information.



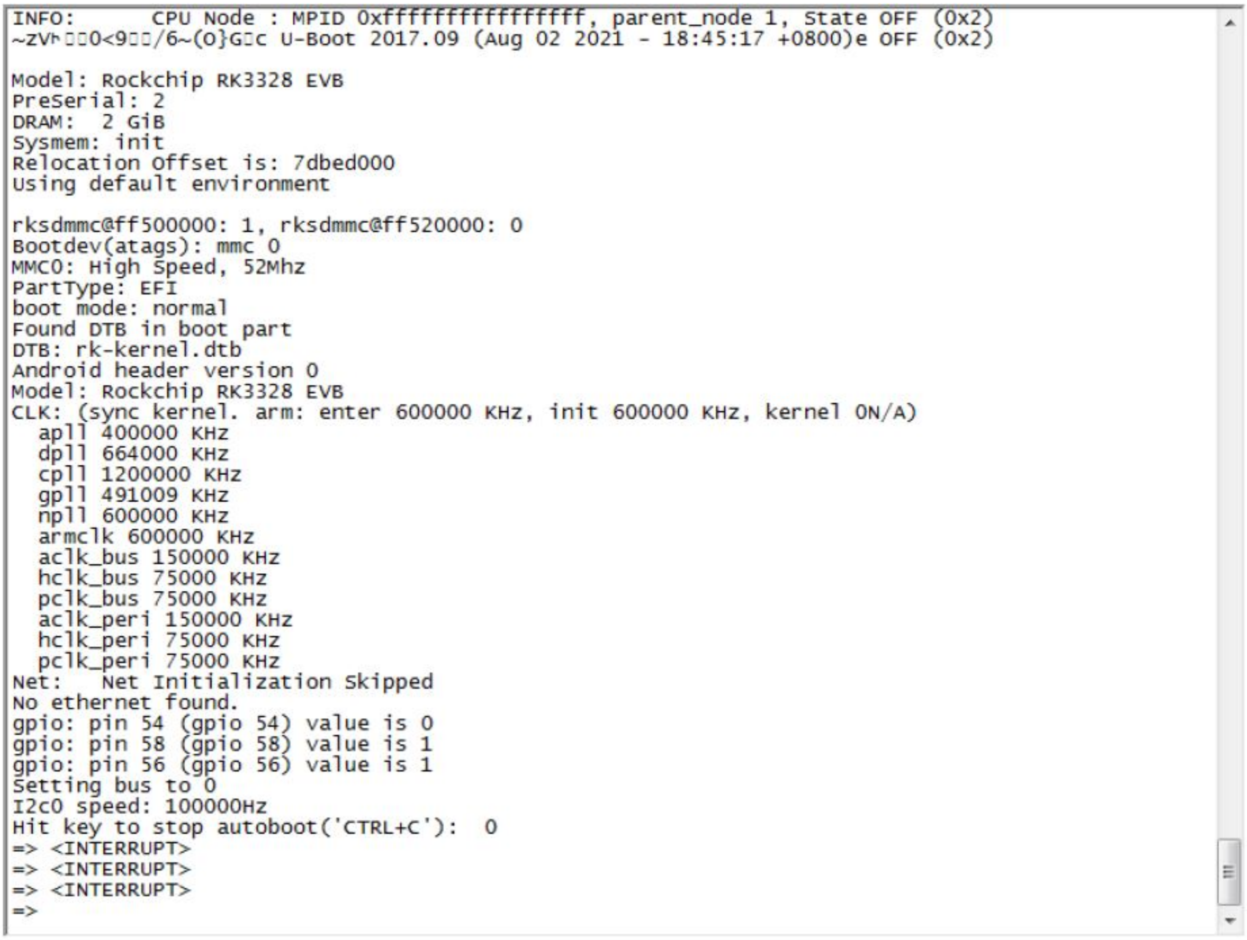
```
root@linaro-alip:~/sx1302_hal# make all
<==========  wait for build OK, and you get ./packet_forwarder/
root@linaro-alip:~/sx1302_hal/packet_forwarder# ./lora_pkt_fwd
```

Image Upgrade

1. **Upgrade Tool**

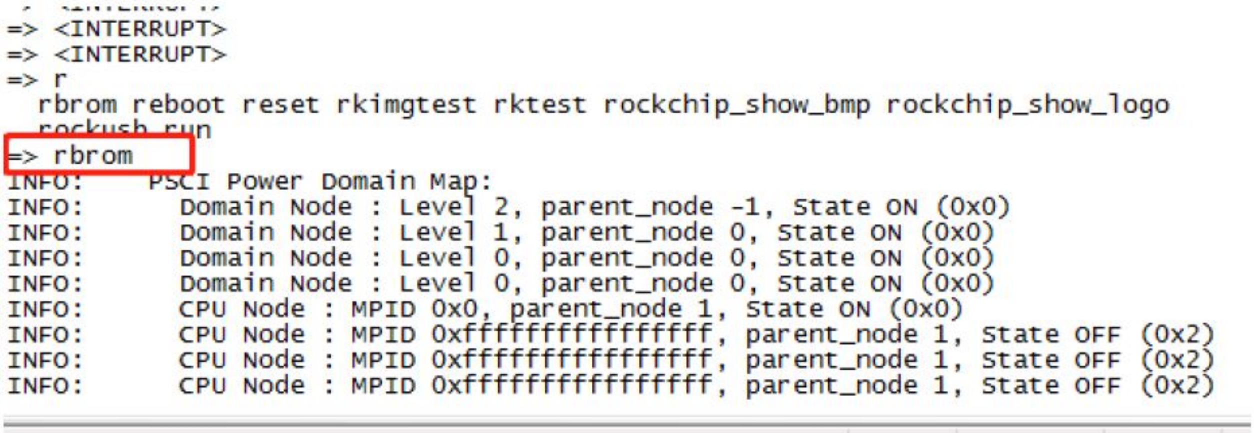
   Upgrade tool AndroidTool_Release_v2.69

2. **Go into Upgrade Mode**
   1. Connect the OTG port to the burning computer USB port, it’s also act as 5V power supply
   2. Press “Ctrl+C” when uboot is booting up, to enter uboot:



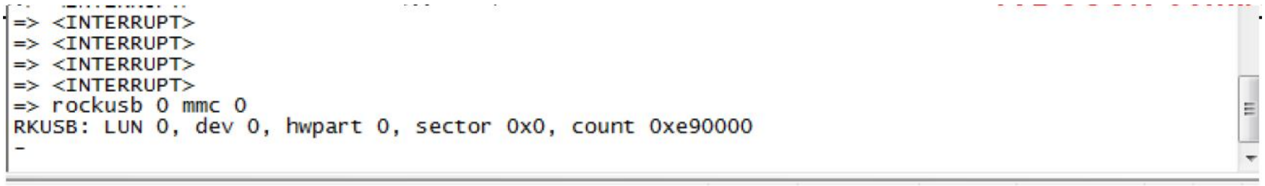
```
INFO:      CPU Node : MPID 0xffffffffffffffff, parent_node 1, State OFF (0x2)
~zVh□□0<9□□/6~(O}G□c U-Boot 2017.09 (Aug 02 2021 - 18:45:17 +0800)e OFF (0x2)

Model: Rockchip RK3328 EVB
PreSerial: 2
DRAM:  2 GiB
Sysmem: init
Relocation Offset is: 7dbed000
Using default environment

rksdmmc@ff500000: 1, rksdmmc@ff520000: 0
Bootdev(atags): mmc 0
MMC0: High Speed, 52Mhz
PartType: EFI
boot mode: normal
Found DTB in boot part
DTB: rk-kernel.dtb
Android header version 0
Model: Rockchip RK3328 EVB
CLK: (sync kernel. arm: enter 600000 KHz, init 600000 KHz, kernel 0N/A)
  apll 400000 KHz
  dpll 664000 KHz
  cpll 1200000 KHz
  gpll 491009 KHz
  npll 600000 KHz
  armclk 600000 KHz
  aclk_bus 150000 KHz
  hclk_bus 75000 KHz
  pclk_bus 75000 KHz
  aclk_peri 150000 KHz
  hclk_peri 75000 KHz
  pclk_peri 75000 KHz
Net:   Net Initialization Skipped
No ethernet found.
gpio: pin 54 (gpio 54) value is 0
gpio: pin 58 (gpio 58) value is 1
gpio: pin 56 (gpio 56) value is 1
Setting bus to 0
I2c0 speed: 100000Hz
Hit key to stop autoboot('CTRL+C'):  0
=> <INTERRUPT>
=> <INTERRUPT>
=> <INTERRUPT>
=>
```

3. uboot “rbrom” comand to reboot the board into maskrom mode, for a complete “update.img” upgrade.

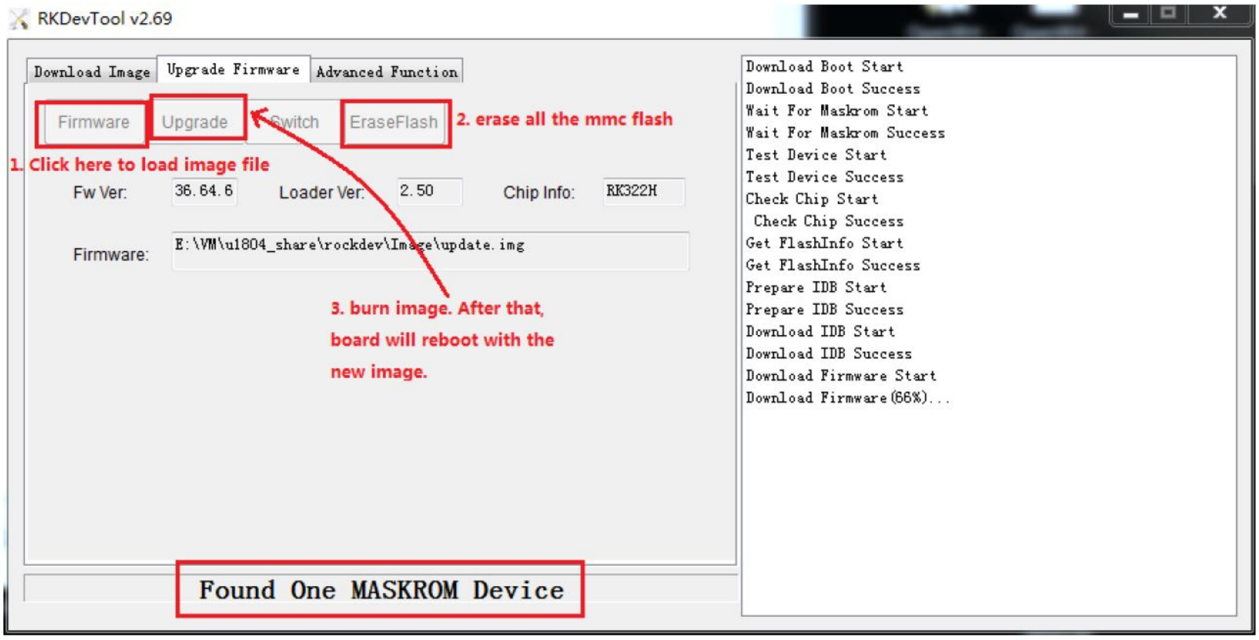


```
=> <INTERRUPT>
=> <INTERRUPT>
=> r
  rbrom reboot reset rkimgtest rktest rockchip_show_bmp rockchip_show_logo
  rockusb run
=> rbrom
INFO:     PSCI Power Domain Map:
INFO:       Domain Node : Level 2, parent_node -1, State ON (0x0)
INFO:       Domain Node : Level 1, parent_node 0, State ON (0x0)
INFO:       Domain Node : Level 0, parent_node 0, State ON (0x0)
INFO:       Domain Node : Level 0, parent_node 0, State ON (0x0)
INFO:       CPU Node : MPID 0x0, parent_node 1, State ON (0x0)
INFO:       CPU Node : MPID 0xffffffffffffffff, parent_node 1, State OFF (0x2)
INFO:       CPU Node : MPID 0xffffffffffffffff, parent_node 1, State OFF (0x2)
INFO:       CPU Node : MPID 0xffffffffffffffff, parent_node 1, State OFF (0x2)
```

4. “rockusb 0 mmc 0” command to reboot board to loader mode, for a partial firmware upgrade or a complete “update.img” upgrade.



```
=> <INTERRUPT>
=> <INTERRUPT>
=> <INTERRUPT>
=> <INTERRUPT>
=> rockusb 0 mmc 0
RKUSB: LUN 0, dev 0, hwpart 0, sector 0x0, count 0xe90000
-
```



1. Click here to load image file

2. erase all the mmc flash

3. burn image. After that, board will reboot with the new image.

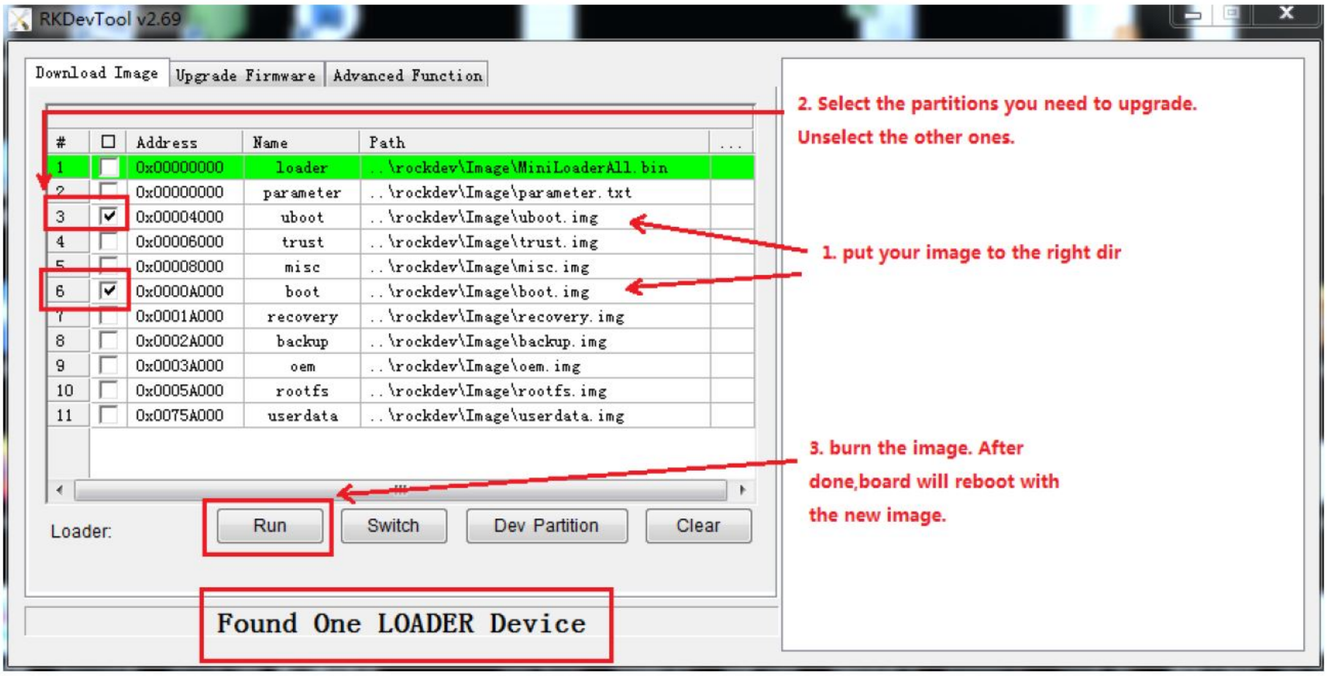
Fw Ver: 36.64.6 Loader Ver: 2.50 Chip Info: RK322H

Firmware: E:\VM\u1804_share\rockdev\Image\update.img

```
Download Boot Start
Download Boot Success
Wait For Maskrom Start
Wait For Maskrom Success
Test Device Start
Test Device Success
Check Chip Start
 Check Chip Success
Get FlashInfo Start
Get FlashInfo Success
Prepare IDB Start
Prepare IDB Success
Download IDB Start
Download IDB Success
Download Firmware Start
Download Firmware(66%)...
```

Found One MASKROM Device

3. The Entire Package of Firmware “update.img” Upgrade
4. Upgrade the Firmware Separately



#	□	Address	Name	Path	...
1	□	0x00000000	loader	..\rockdev\Image\MiniLoaderAll.bin	
2	□	0x00000000	parameter	..\rockdev\Image\parameter.txt	
3	☑	0x00004000	uboot	..\rockdev\Image\uboot.img	
4	□	0x00006000	trust	..\rockdev\Image\trust.img	
5	□	0x00008000	misc	..\rockdev\Image\misc.img	
6	☑	0x0000A000	boot	..\rockdev\Image\boot.img	
7	□	0x0001A000	recovery	..\rockdev\Image\recovery.img	
8	□	0x0002A000	backup	..\rockdev\Image\backup.img	
9	□	0x0003A000	oem	..\rockdev\Image\oem.img	
10	□	0x0005A000	rootfs	..\rockdev\Image\rootfs.img	
11	□	0x0075A000	userdata	..\rockdev\Image\userdata.img	

2. Select the partitions you need to upgrade. Unselect the other ones.

1. put your image to the right dir

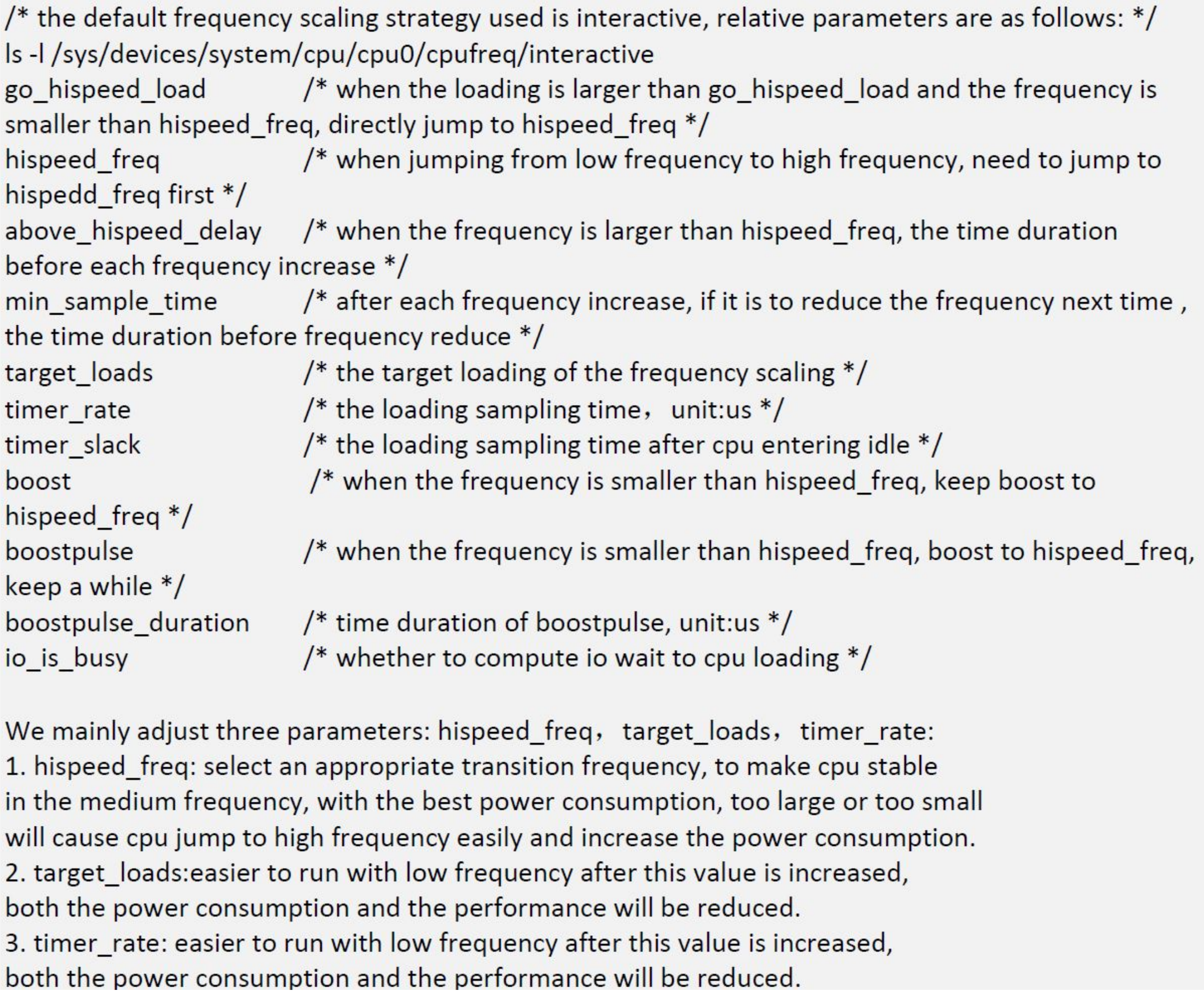
3. burn the image. After done,board will reboot with the new image.

Found One LOADER Device

Power management configuration

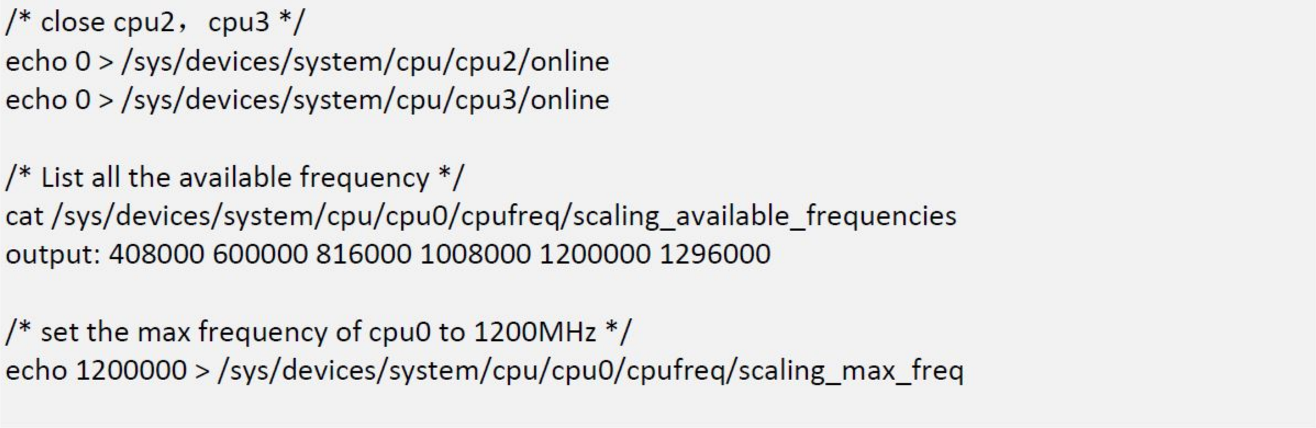
The battery management chip Dusun used is BQ25895 Methods to optimize CPU power consumption is listed,

- Adjust cpufreq parameter.



```
/* the default frequency scaling strategy used is interactive, relative parameters are as follows: */
ls -l /sys/devices/system/cpu/cpu0/cpufreq/interactive
go_hispeed_load          /* when the loading is larger than go_hispeed_load and the frequency is
smaller than hispeed_freq, directly jump to hispeed_freq */
hispeed_freq             /* when jumping from low frequency to high frequency, need to jump to
hispedd_freq first */
above_hispeed_delay      /* when the frequency is larger than hispeed_freq, the time duration
before each frequency increase */
min_sample_time          /* after each frequency increase, if it is to reduce the frequency next time ,
the time duration before frequency reduce */
target_loads             /* the target loading of the frequency scaling */
timer_rate               /* the loading sampling time， unit:us */
timer_slack              /* the loading sampling time after cpu entering idle */
boost                     /* when the frequency is smaller than hispeed_freq, keep boost to
hispeed_freq */
boostpulse               /* when the frequency is smaller than hispeed_freq, boost to hispeed_freq,
keep a while */
boostpulse_duration      /* time duration of boostpulse, unit:us */
io_is_busy               /* whether to compute io wait to cpu loading */

We mainly adjust three parameters: hispeed_freq， target_loads， timer_rate:
1. hispeed_freq: select an appropriate transition frequency, to make cpu stable
in the medium frequency, with the best power consumption, too large or too small
will cause cpu jump to high frequency easily and increase the power consumption.
2. target_loads:easier to run with low frequency after this value is increased,
both the power consumption and the performance will be reduced.
3. timer_rate: easier to run with low frequency after this value is increased,
both the power consumption and the performance will be reduced.
```

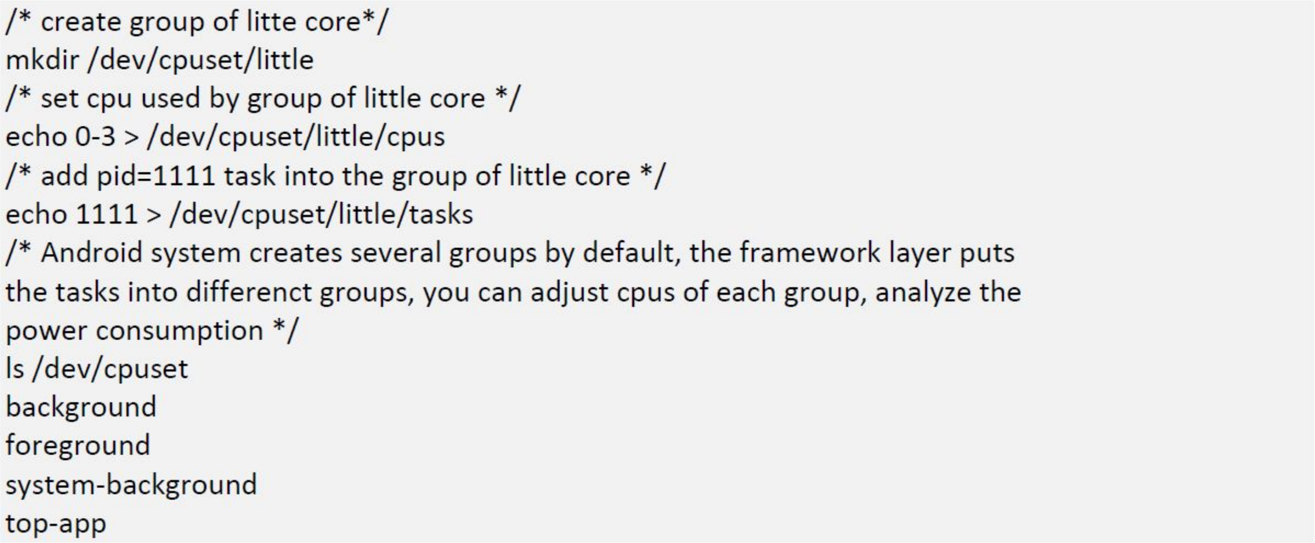
- Close some cpu, limit the highest frequency of cpu



```
/* close cpu2， cpu3 */
echo 0 > /sys/devices/system/cpu/cpu2/online
echo 0 > /sys/devices/system/cpu/cpu3/online

/* List all the available frequency */
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
output: 408000 600000 816000 1008000 1200000 1296000

/* set the max frequency of cpu0 to 1200MHz */
echo 1200000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
```

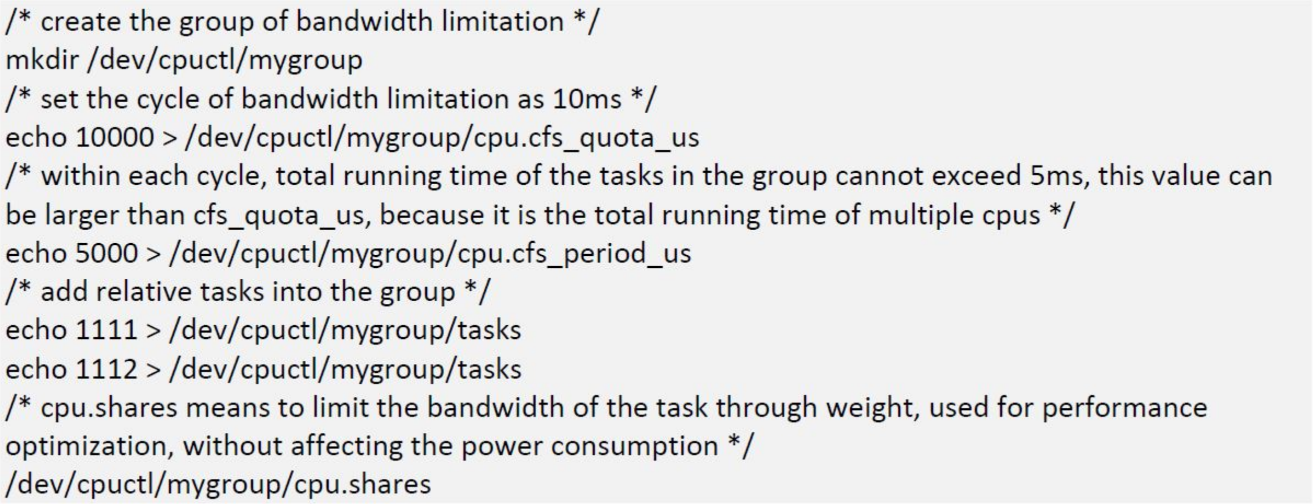
- SoC with ARM Big-Little architecture can bind the tasks with high loading to little cores through CPUSET since the energy efficiency of the little core is better.

  Note: SoC with SMP architecture can also bind the tasks to some cpu so that other cpus can enter low power consumption mode, but maybe it will make cpu easy to run with high frequency, which will increase the power consumption.



```
/* create group of litte core*/
mkdir /dev/cpuset/little
/* set cpu used by group of little core */
echo 0-3 > /dev/cpuset/little/cpus
/* add pid=1111 task into the group of little core */
echo 1111 > /dev/cpuset/little/tasks
/* Android system creates several groups by default, the framework layer puts
the tasks into differenct groups, you can adjust cpus of each group, analyze the
power consumption */
ls /dev/cpuset
background
foreground
system-background
top-app
```

- Limit the cpu bandwidth of the tasks with high loading through CPUCTL (need to enable the macro CONFIG_CFS_BANDWIDTH).



```
/* create the group of bandwidth limitation */
mkdir /dev/cpuctl/mygroup
/* set the cycle of bandwidth limitation as 10ms */
echo 10000 > /dev/cpuctl/mygroup/cpu.cfs_quota_us
/* within each cycle, total running time of the tasks in the group cannot exceed 5ms, this value can
be larger than cfs_quota_us, because it is the total running time of multiple cpus */
echo 5000 > /dev/cpuctl/mygroup/cpu.cfs_period_us
/* add relative tasks into the group */
echo 1111 > /dev/cpuctl/mygroup/tasks
echo 1112 > /dev/cpuctl/mygroup/tasks
/* cpu.shares means to limit the bandwidth of the task through weight, used for performance
optimization, without affecting the power consumption */
/dev/cpuctl/mygroup/cpu.shares
```

Floor 8, building A, Wantong center, Hangzhou 310004, china
Tel: 86-571-86769027/8 8810480
Website: www.dusuniot.com
www.dusunremotes.com
www.dusunlock.com

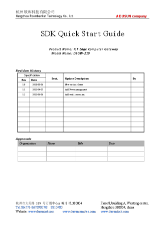
Revision History

Specification		Sect.	Update Description	By
Rev	Date			
1.0	2021-08-06		New version release	
1.1	2022-04-05		Add Power management	
1.2	2022-06-06		Add serial connection	

Approvals

Organization	***Name***	***Title***	***Date***

Documents / Resources

	DUSUN DSGW-210 IoT Edge Computer Gateway [pdf] User Guide DSGW-210 IoT Edge Computer Gateway, DSGW-210, IoT Edge Computer Gateway, Computer Gateway, Gateway

References

-
- Dusun IoT: IoT Gateway Hardware Suppplier & Solutions Vendor- DusunIoT
- Dusunremotes | Custom Intelligent Remote Control Manufacturer
- Software Developer Docs - Silicon Labs
- SX1302 LoRa Core Digital Baseband Chip | Semtech