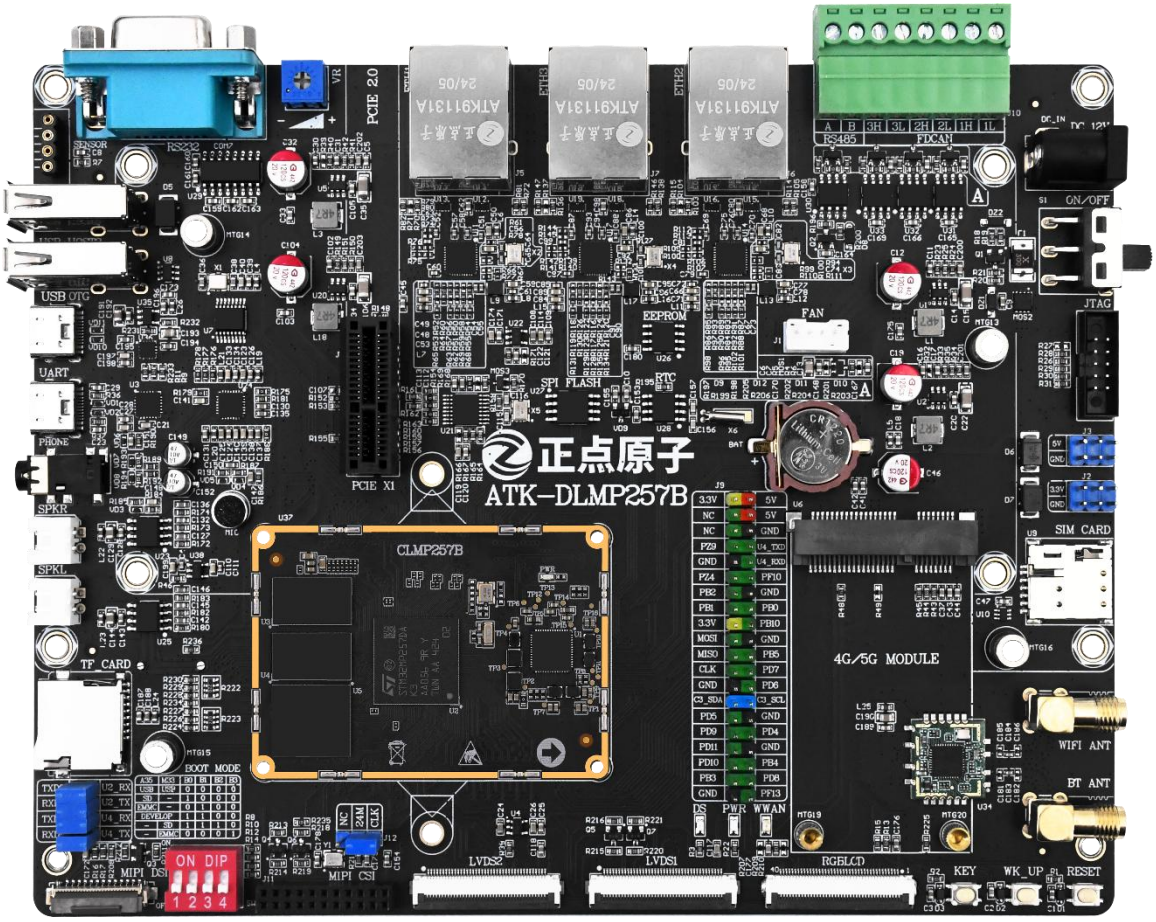
ATK-DLMP257B

Ethernet Switch Development Guide

V1.0



ALIENTEK

1. Shopping:

TMALL: https://zhengdianyuanzi.tmall.com

TAOBAO: https://openedv.taobao.com

2. Download

Address: http://www.openedv.com/docs/index.html

3. FAE

Website : www.alientek.com

Forum : http://www.openedv.com/forum.php

Videos : www.yuanzige.com

Fax : +86 - 20 - 36773971

Phone : +86 - 20 - 38271790

Disclaimer

The product specifications and instructions mentioned in this document are for reference only and subject to update without prior notice; Unless otherwise agreed, this document is intended as a product guide only, and none of the representations made herein constitutes a warranty of any kind.

In order to get the latest version of product information, please regularly visit the download center or contact the customer service of Taobao ALIENTEK flagship store. Thank you for your tolerance and support.

Revision History:

Version	Version Update Notes	Responsible person	Proofreading	Date
V1.0	release officially	ALIENTEK	ALIENTEK	2025.04.01

Catalogue

Chapter 1. MP257 Ethernet Switch (ETHSW)

1.1 What is an Ethernet switch

An Ethernet switch is a hardware device used to connect devices on a local area network (LAN) and forward packets between them. It operates in the data link layer (layer 2) of the OSI model and uses the MAC address to determine the destination of each packet. Ethernet switches can improve network performance by reducing collisions and congestion, and they can also provide security features such as VLAN and port-based access control. Ethernet switches are more efficient than Ethernet Bridges in terms of CPU usage. Since the hardware Ethernet switch is responsible for data forwarding, filtering, and prioritization, it reduces the workload on the CPU. As a result, the CPU can focus on other tasks, which improves the network performance.

1.2 Main features of MP257 ETHSW

Ethernet switch, referred to as ETHSW. Here are the features of the MP257 Ethernet switch:

- Three-port Gigabit Ethernet switch: two RMII/RGMII external ports, the third port is connected to the ETH 1 Ethernet controller
- Two real-time clocks
- Time synchronization (IEEE 802.1AS)
  - IEEE 1588 v2 specification for clock synchronization in networks.
  - Supports the timing requirements of scheduled TSN networks
- Time-aware scheduler for scheduling traffic (IEEE802.1Qbv)
  - Provides guaranteed latency for time-critical traffic on standard Ethernet
- Frame preemption (IEEE 802.1Qbu)
  - Allows optimal bandwidth utilization for unscheduled background traffic sent in parallel with scheduled traffic
- Redundancy (frame replication and reliability elimination) (IEEE 802.1CB)
- Provides Ethernet subsystem-wide time synchronization for the Ethernet controller external PTP bus

1.3 MP257 ETHSW instructions

The block diagram of MP257 Ethernet switch is as follows:

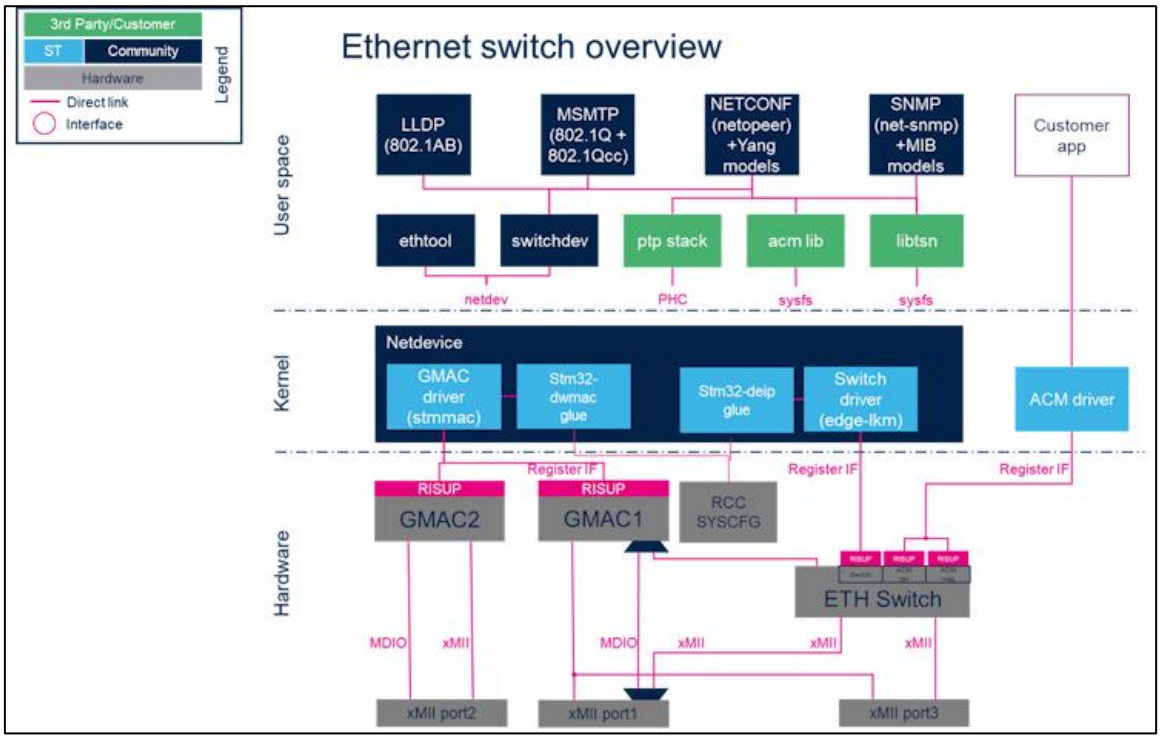


Figure 1.3-1 Block diagram of MP257 ETHSW

This diagram is an overview of an Ethernet Switch, showing the Hardware layer, the Kernel layer, the User Space, and how they interact. Here's a detailed analysis:

1. Legend

The legend explains the meaning of the different colors and symbols:

- Green: 3rd Party/Customer
- Blue (ST Community) : represents components provided by ST
- Gray (Hardware) : Indicates hardware
- Pink line (Direct link) : indicates a direct connection
- Pink circle (Interface) : Represents an interface

2.User Space

The user space contains several software components that are responsible for managing and configuring Ethernet switches. The main components include:

- LLDP (802.1AB) : link-layer discovery protocol for exchanging device information.
- MSMTP (802.1Q + 802.1Qcc) : Supports VLAN (802.1Q) and TSN resource management (802.1Qcc).
- NETCONF (netopeer) + Yang models: A protocol for network device management that supports the YANG model.
- SNMP (net-snmp) + MIB models: Management Information Base (MIB) models based on SNMP protocol.

In addition, there are several userspace tools and libraries:

- ethtool: Used to configure and query network device status.
- switchdev: A Linux kernel framework for managing hardware switches.
- ptp stack: Time Synchronization Protocol (PTP) related components.
- acm lib: A library for advanced switch management.

- libtsn: A library that supports Time-sensitive networking (TSN) functionality.

These tools and libraries interact with the kernel layer via netdev, PHC, and sysfs.

3. Kernel

The kernel layer contains the network device (Netdevice) and several drivers. The main components include:

- GMAC driver (stmmac) : For handling GMAC (Gigabit Media Access Controller).
- STM32-DWMAC glue: GMAC driver glue code provided by ST for adaptation to Stm32 hardware.
- STM32-DEIP glue: For switch glue layer drivers on STM32.
- Switch driver (edge-lkm) : The core driver of the switch.
- ACM driver: Advanced switch management driver (to interact with ACM applications in userspace).

These components communicate through netdev (Network Device Interface).

4. Hardware layer

The hardware layer is the actual STM32 Ethernet switch and its interface, including:

- GMAC (Gigabit Ethernet Controller)
  - GMAC1

Connect the ETH Switch.

xMII port1 is connected through MDIO (Managing Data input/output) and xMII (Media Independent Interface).

Register IF configuration via RCC SYSCFG.

  - GMAC2

Connect to xMII port2.

Ethernet PHY is connected via MDIO and xMII.

- Ethernet Switch (ETH Switch)

Connect GMAC1 (Main Gigabit Ethernet controller).

Provides multiple xMII ports (xMII port1, port2, port3).

The Switch Port is managed by RISUP (Register Interface for Switch User Port).

GMAC1, GMAC2, and RCC SYSCFG are connected via Register IF.

5. Interaction flow

- Hardware initialization

GMAC1 and GMAC2 connect switches and initialize registers through xMII.

The ETH Switch connects and config multiple xMII ports.

- Kernel driver loading

The GMAC driver (stmmac) handles the GMAC controller.

Stm32-dwmac glue and STM32-DEIP glue are adapted to STM32 hardware.

The Switch driver (edge-lkm) handles the switch.

- User space interaction

ethtool and switchdev access kernel devices via netdev.

The ptp stack (Precision Time Protocol) is synchronized through the PHC (kernel hardware clock).

acm lib, libtsn access switch parameters through sysfs.

- Senior management

LLDP, MSMTP, NETCONF, SNMP are used to manage and monitor switches.

Customer app uses ACM driver for custom configuration.

Chapter 2. Software configuration

ATK-DLMP257B development board is written with the factory system by default, and the factory system has been configured with relevant software configuration. Only relevant configuration instructions are made here, and customers do not need to modify.

2.1 Kernel Configuration

Factory kernel source code use can refer to the [ALIENTEK] ATK-DLMP257B Factory System Source Code Use Guide V1.0, here will not repeat. Go to the factory kernel source directory, enable the toolchain, and open the factory kernel configuration:

```
source /opt/st/stm32mp2/5.0.3-snapshot/environment-setup-cortexa35-ostl-linux
make stm32mp257_atk_defconfig
make menuconfig
```

The configuration is as follows:

```
[*] Networking support --->
Networking options --->
[*] QoS and/or fair queueing --->
<M> Credit Based Shaper (CBS)
<M> Time Aware Priority (taprio) Scheduler
<M> Multi-queue priority scheduler (MQPRIO)
[*] Actions
<M> Traffic Policing
<M> Generic actions
<M> Redirecting and Mirroring
<M> SKB Editing
<M> Vlan manipulation
<M> Frame gate entry list control tc action
```

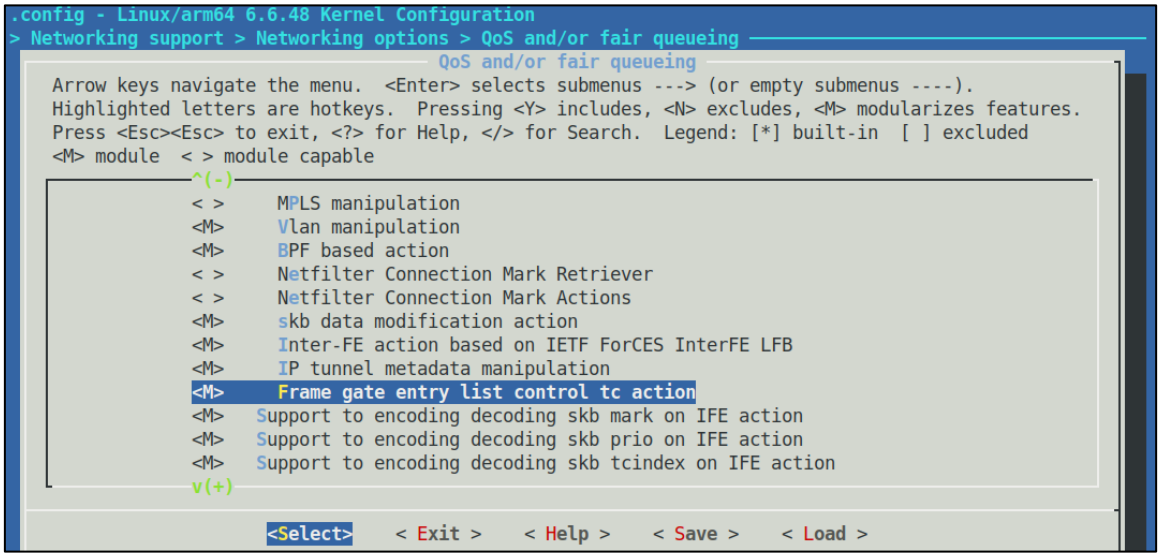


Figure 2.1-1 Kernel configuration

1.Kernel options parsing

- Networking support: Enables networking support for the Linux kernel.
- Networking options: Access the configuration of networking options.
- QoS and/or fair queueing: Used to enable QoS mechanisms and fair queueing scheduling algorithms.

2. Key TSN-related kernel options

(1) Credit Based Shaper (CBS)

Modular (M) : CBS is compiled as a module (CBS.ko).

Role:

- Credit score scheduling algorithm defined by IEEE 802.1Qav standard.
- Limit traffic and ensure bandwidth for time-sensitive traffic by allocating credit points.
- Suitable for traffic shaping and TSN traffic management.

(2) Time Aware Priority (taprio) Scheduler

taprio (Time-Aware Priority Scheduler) :

- IEEE 802.1Qbv time-aware scheduler.
- Based on time slice scheduling method, strict sending window can be allocated for different priority traffic.
- Ensure that critical data frames can be sent within a specific time window.

(3) Multi-queue priority scheduler (MQPRIO)

MQPRIO (Multi-queue Priority Scheduling) :

- Supports eight priority queues defined by 802.1Q.
- Provides a refined queue management mechanism to ensure that different types of packets can be scheduled correctly according to priority.
- Multi-queue configuration for TSN switches and GMAC. .

3. Other qos-related Traffic Control (TC) options

Actions are part of the Linux Traffic Control (TC) subsystem and support various packet handling functions.

(1) Traffic Policing

Role:

- Limit the packet transmission rate to prevent unexpected traffic from entering the network.
- Suitable for bandwidth management and QoS control.

(2) Generic actions

Role:

- Allows performing custom behavior during packet forwarding, such as changing headers, adjusting priorities, etc.

(3) Redirecting and Mirroring

Role:

- Redirecting: Forwarding packets from one interface to another.
- Mirroring: Copying packets and sending them to another port (for traffic monitoring).

(4) SKB Editing

Role:

- Allows you to modify fields in the packet, such as MAC address, IP address, etc.

(5) VLAN manipulation

Role:

- Support VLAN tag insertion, deletion, modification, for VLAN network environment.

(6) Frame gate entry list control (tc action)

Role:

- Cooperated with IEEE 802.1Qbv time-aware scheduling, used to control the entry time window of data frames.
- Used to precisely schedule data flows in TSN networks.

2.2 Device tree configuration

2.2.1 &switch0 node

2 gb version core board, for example, Ethernet peripheral node is located in the arch/arm64 / boot/DTS/st/stm32mp257d - atk - DDR - 2 gb. DTS, open the file, find switch0 nodes:

```
&switch0 {
2   status = "okay";
3   pinctrl-0 = <&eth1_rgmii_pins_a>, <&eth3_rgmii_pins_a>;
4   pinctrl-1 = <&eth1_rgmii_sleep_pins_a>, <&eth3_rgmii_sleep_pins_a>;
5   pinctrl-names = "default", "sleep";
6   phy-mode = "rgmii";
7   st,ethsw-internal-125;
8 };
```

Line 1, &switch0: defines the Ethernet Switch (ETH Switch) device node.

Line 2, status = "okay" : Enables the device so that it takes effect in the Linux kernel.

Line 3, pinctrl-0: Pin configuration in default state.

eth1_rgmii_pins_a and eth3_rgmii_pins_a represent the pin configuration of ports 1 and 3 in RGMII mode.

Line 4, pinctrl-1: Pin configuration in sleep mode.

eth1_rgmii_sleep_pins_a and eth3_rgmii_sleep_pins_a represent the pin Settings of the port in low-power mode.

Line 5, pinctrl-names: defines two pin states:

- "default" : pinctrl-0 is used in normal operation mode.
- "sleep" : pinctrl-1 is used when the device goes into sleep mode.

Line 6, PHY-mode = "rgmii" : Set PHY (Ethernet physical layer) to adopt RGMII mode (Reduced Gigabit Media-Independent Interface). RGMII is mainly used in Gigabit Ethernet (1000 Mbps) and reduces the number of pins compared to GMII mode. TSN switches support RGMII and RMII modes, but RGMII is selected here, which means: the Ethernet PHY is connected to the MAC (Ethernet Controller) through the RGMII interface. Full duplex 1000Mbps transmission is supported.

Line 7, st,ethsw-internal-125: enables the internal 125MHz clock of the switch. This option is suitable for RGMII devices that use an internal clock source to ensure that the RGMII clock is working properly.

2.2.2 The eth1 and eth2 nodes

```
&eth1 {
  status = "okay";
  pinctrl-0 = <&eth1_mdio_pins_a>;
  pinctrl-names = "default";
  phy-mode = "rgmii";
  st,eth-clk-sel;
  snps,ext-systime;
     fixed_link: fixed-link {
      speed = <1000>;
      full-duplex;
     };

     mdio1 {
      #address-cells = <1>;
          #size-cells = <0>;
      compatible = "snps,dwmac-mdio";
      phy1_eth1: ethernet-phy@4 {
             compatible = "ethernet-phy-id4f51.e91b",
                    "ethernet-phy-ieee802.3-c22";
             reg = <4>;
      };
      phy2_eth1: ethernet-phy@5 {
             compatible = "ethernet-phy-id4f51.e91b",
                    "ethernet-phy-ieee802.3-c22";
             reset-gpios = <&gpiof 3 GPIO_ACTIVE_LOW>;
             reset-assert-us = <10000>;
             reset-deassert-us = <80000>;
             reg = <5>;
      };
     };
};

&eth2 {
 status = "okay";
 pinctrl-0 = <&eth2_rgmii_pins_a>;
 pinctrl-1 = <&eth2_rgmii_sleep_pins_a>;
 pinctrl-names = "default", "sleep";
 phy-mode = "rgmii-id";
```

```
39  max-speed = <1000>;
40  phy-handle = <&phy1_eth2>;
41  st,eth-ptp-from-rcc;
42
43  mdio1 {
44      #address-cells = <1>;
45      #size-cells = <0>;
46      compatible = "snps,dwmac-mdio";
47      phy1_eth2: ethernet-phy@1 {
48          compatible = "ethernet-phy-id4f51.e91b";
49          reset-gpios = <&gpiof 4 GPIO_ACTIVE_LOW>;
50          reset-assert-us = <10000>;
51          reset-deassert-us = <80000>;
52          reg = <1>;
53      };
54  };
55 };
```

Line 1, &eth1: refers to the eth1 device node in the device tree, which represents Ethernet Controller 1.

Line 2, status = "okay" : Enable the eth1 device.

Line 3, pinctrl-0: Specifies eth1 to use eth1_mdio_pins_a as the MDIO pin configuration.

Line 4, pinctrl-names = "default" : pinctrl-0 pin configuration is used by default.

Line 5, PHy-mode = "rgmii" : Set the PHY connection mode of the eth1 Ethernet to RGMII (Gigabit Media Independent Interface).

Line 6, st,eth-clk-sel: ST private attribute, used to select Ethernet clock source.

Line 7, snps,ext-systime: Synopsys design-specific attributes that indicate synchronization using external system time.

Line 8, Fixed-link: The fixed link is configured, that is, the eth1 is directly connected and does not rely on the PHY.

Line 9, speed = <1000> : The link rate is 1000 Mbps (gigabit).

Line 10, full-duplex: Full-duplex mode.

Line 13, mdio1: Defines the MDIO bus for managing PHY devices.

Line 14, #address-cells = <1> : 1 cell is used for the PHY address.

Line 15, #size-cells = <0> : No additional address space is allocated.

Line 16, compatible = "snps, DWMAC-mdio ": This MDIO controller is compatible with Synopsys DWMAC MDIO drivers.

Line 17, phy1_eth1: first PHY device, address 4.

Lines 18-19, compatible:

- "ethernet-phy-id4f51.e91b" : PHY chip ID.
- "ethernet-phy-ieee802.3-c22" : complies with IEEE 802.3 standard.

Line 20, reg = <4> : The address of this PHY on the MDIO bus is 4.

Lines 22 to 24, phy2_eth1: The second PHY device, address 5, also conforms to the IEEE 802.3 standard.

Line 25, reset-gpios = <&gpiof 3 GPIO_ACTIVE_LOW> : PHY2 reset GPIO connected to gpiof 3, low reset.

Line 26, reset-assert-us = <10000> : reset lasts 10ms.

Line 27, reset-deassert-us = <80000> : Wait 80ms after releasing the reset.

Line 28, reg = <5> : The address of this PHY on the MDIO bus is 5.

Line 33, &eth2: refers to the eth2 device node in the device tree, which represents Ethernet Controller 2.

On line 34, status = "okay" : Enable the eth2 device.

Line 35, pinctrl-0: Default RGMII pin configuration.

Line 36, pinctrl-1: Sleep-mode RGMII pin configuration.

Line 37, pinctrl-names = "default", "sleep" : pinctrl for normal and sleep mode, respectively.

Line 38, phy-mode = "rgmii-id" : The PHY connection mode of the eth2 Ethernet is RGMII-ID (with internal delay).

On line 39, max-speed = <1000> : The maximum speed is 1000 Mbps.

Line 40, phy-handle = <&phy1_eth2> : The PHY of the device is connected to phy1_eth2.

Line 41, st,eth-ptp-from-rcc: The PTP (Precision Time Protocol) clock of this device comes from the RCC (clock controller).

Line 43, mdio1: Defines the MDIO bus, which is used to manage PHY devices.

Line 44, #address-cells = <1> : 1 cell is used for the PHY address.

Line 45, #size-cells = <0> : No additional address space is allocated.

Line 46, compatible = "snps, DWMAC-mdio ": This MDIO controller is compatible with Synopsys DWMAC MDIO drivers.

Line 47, phy1_eth2: first PHY device, address 1.

Line 48, compatible = "ethernet-phy-id4f51.e91b" : The PHY device is compatible with chips with ID 4f51.e91b.

Line 49, reset-gpios = <&gpiof 4 GPIO_ACTIVE_LOW> : Reset the PHY's GPIO connection to gpiof 4, low reset.

Line 50, reset-assert-us = <10000> : reset lasts 10ms.

Line 51, reset-deassert-us = <80000> : Wait 80ms after releasing the reset.

Line 52, reg = <1> : The address of this PHY on the MDIO bus is 1.

2.2.3 switch0 node

Switch0 node from the arch/arm64 / boot/DTS/st/stm32mp257 dtsi file, the file cannot be modified, the content is as follows:

```
&rifsc {
2   switch0: ttt-sw@4c000000 {
3       #address-cells  = <1>;
4       #size-cells     = <1>;
5       compatible = "st,stm32-deip";
6       clock-names = "ethsw-bus-clk", "ethsw-clk",
7                 "ethswacmcfg-bus-clk", "ethswacmmsg-bus-clk";
8       clocks = <&rcc CK_BUS_ETHSW>,
9           <&rcc CK_KER_ETHSW>,
```

```
10          <&rcc CK_BUS_ETHSWACMCFG>,
11          <&rcc CK_BUS_ETHSWACMMSG>;
12      st,syscon = <&syscfg 0x3800>;
13      ranges = <0x4c000000 0x4c000000 0x2000000>,
14          <0x4b000000 0x4b000000 0xc0000>;
15      access-controllers = <&rifsc 70>;
16      power-domains = <&CLUSTER_PD>;
17      status = "disabled";
18
19      deip_sw0: deip-sw@4c000000 {
20          compatible =  "ttt,deip-sw";
21          reg = <0x4c000000 0x2000000>;
22          interrupts = <GIC_SPI 250 IRQ_TYPE_LEVEL_HIGH>;
23      };
24
25      acm@4b000000 {
26          compatible = "ttt,acm-4.0";
27          reg = <0x4b000000 0x00400>,
28              <0x4b010000 0x10000>,
29              <0x4b030000 0x10000>,
30              <0x4b050000 0x10000>,
31              <0x4b060000 0x20000>,
32              <0x4b080000 0x40000>;
33          reg-names = "CommonRegister",
34                  "Bypass1",
35                  "Bypass0",
36                  "Redundancy",
37                  "Scheduler",
38                  "Messagebuffer";
39          buffers = <32>;
40          ptp_worker = <&deip_sw0>;
41      };
42  };
43 };
```

Line 2, switch0: defines the Time-Sensitive Networking Switch (TSN), alias ttt-sw, @4c000000: indicates that the register base address of this device is 0x4C000000.

Line 3, #address-cells = <1> : The subdevice address occupies 1 32-bit cell.

Line 4, #size-cells = <1> : The child device size occupies 1 32-bit cell.

Line 5, compatible: declares that the device is compatible with st, STM32-DEIP, meaning that the switch uses the STM32 DEIP switch driver.

On lines 6-11, clock-names: Defines four clock signals:

- ethsw-bus-clk: Switch bus clock.
- ethsw-clk: switch core clock.

- ethswacmcfg-bus-clk: ACM configuration bus clock.
- ethswacmmsg-bus-clk: ACM message transfer bus clock.
- clocks: &rcc stands for clock controller; the next four clocks are provided by the rcc.

On line 12, st,syscon binds syscfg (system configuration register) at offset 0x3800.

Lines 13 through 14 define the memory mapping range of the device:

- 0x4C000000: Switch (DEIP), size 0x2000000 (32MB).
- 0x4B000000: ACM-related, size 0xC0000 (768KB).

Line 15, the access control is managed by the rifsc device with ID 70.

In line 16, Power-Domains binds the CLUSTER_PD power domain, indicating that the power of the device is managed by this domain.

Line 17, status = "disabled" : The switch is disabled by default and needs to be enabled on system initialization.

Line 19, deip_sw0: Defines the DEIP switch subdevice.

Line 20, compatible = "ttt,deip-sw" : The device is compatible with ttt,deip-sw.

On line 21, reg = <0x4c000000 0x2000000>; Device base address: 0x4C000000. Size: 0x2000000 (32MB)

On line 22, interrupts = <GIC_SPI 250 IRQ_TYPE_LEVEL_HIGH>; The device uses GIC (Universal Interrupt Controller), interrupt number 250, high level trigger.

Line 25, acm@4b000000: Defines ACM (Accelerated Computation module).

Line 26, compatible = "ttt,acm-4.0"; : Compatible with ttt,acm-4.0 device drivers.

On lines 27 through 32, reg: defines the addresses and sizes of the different registers:

- 0x4b000000 0x00400: general purpose register.
- 0x4b010000 0x10000: Bypass mode 1.
- 0x4b030000 0x10000: Bypass mode 0.
- 0x4b050000 0x10000: Redundancy management.
- 0x4b060000 0x20000: Scheduler
- 0x4b080000 0x40000: Message buffer.

Lines 33 through 38, reg-names: This gives names to registers in reg so that the driver can call them.

On line 39, the number of device buffers is set to 32.

On line 40, ptp_worker is bound to deip_sw0 (DEIP switch), indicating that PTP time synchronization is handled by the switch.

2.3 Ethernet Switch Configuration

The initialization order during the load and initialization sequence is:

1, GMAC ETH1 driver.

2, STM32-DEIP glue program, used for switch glue layer driver on STM32.

3. edge-lkml program, switch core driver.

These files describe the process of detecting and initializing hardware devices during system boot and are located in the /etc/modprobe.d/edgx_sw_modprobe.conf file of the ATK-DLMP257B development board.

2.3.1 edgx_sw_core.conf

During the detection of the switch driver (edge-lkm), the interface of the internal port connection of the switch is given as a parameter in the file, and the development board execs the following command:

```
cat /etc/modprobe.d/edgx_sw_core.conf
```

```
root@ATK-DLMP257:~# cat /etc/modprobe.d/edgx_sw_core.conf
options edgx_pfm_lkm netif="end1:0"
root@ATK-DLMP257:~#
```

Figure 2.3-1 Check out edgx_sw_core.conf

Here the internal port of the switch is connected to the end1 interface.

2.3.2 st-tsn.service

You can list all available interfaces using the ifconfig command:

```
ifconfig
```

```
root@ATK-DLMP257:~# ifconfig
br0       Link encap:Ethernet  HWaddr 16:4B:54:2E:02:16
          inet6 addr: fe80::144b:54ff:fe2e:216/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1419 (1.3 KiB)

end0      Link encap:Ethernet  HWaddr EA:EB:A1:B3:78:1F
          inet addr:192.168.6.215  Bcast:192.168.6.255  Mask:255.255.255.0
          inet6 addr: fe80::e8eb:a1ff:feb3:781f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:584 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:46293 (45.2 KiB)  TX bytes:9873 (9.6 KiB)
          Interrupt:68 Base address:0x8000

end1      Link encap:Ethernet  HWaddr 12:81:E1:CE:06:6E
          inet6 addr: fe80::1081:e1ff:fece:66e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:417 errors:0 dropped:0 overruns:0 frame:0
          TX packets:148 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:42622 (41.6 KiB)  TX bytes:22875 (22.3 KiB)
          Interrupt:66 Base address:0x8000

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:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7249 (7.0 KiB)  TX bytes:7249 (7.0 KiB)

sw0ep     Link encap:Ethernet  HWaddr 12:81:E1:CE:06:6E
          inet addr:192.168.0.10  Bcast:0.0.0.0  Mask:255.255.255.255
          inet6 addr: fe80::1081:e1ff:fece:66e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:417 errors:0 dropped:0 overruns:0 frame:0
          TX packets:148 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:44290 (43.2 KiB)  TX bytes:23467 (22.9 KiB)

sw0p1     Link encap:Ethernet  HWaddr 2E:94:65:84:86:00
          inet6 addr: fe80::2c94:65ff:fe84:8600/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:148 errors:0 dropped:4 overruns:0 frame:0
          TX packets:417 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23467 (22.9 KiB)  TX bytes:44290 (43.2 KiB)

sw0p2     Link encap:Ethernet  HWaddr 2E:94:65:84:86:01
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:379 (379.0 B)

sw0p3     Link encap:Ethernet  HWaddr 2E:94:65:84:86:02
          inet6 addr: fe80::2c94:65ff:fe84:8602/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:563 errors:1 dropped:5 overruns:0 frame:1
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:56225 (54.9 KiB)  TX bytes:7244 (7.0 KiB)
```

Figure 2.3-2 ifconfig

These interfaces are set up during boot, thanks to the script /usr/sbin/ttt-ip-init-systemd.sh, which is launched from the /lib/systemd/system/st-tsn.service service.

```
cat /lib/systemd/system/st-tsn.service
```

```
root@ATK-DLMP257:~# cat /lib/systemd/system/st-tsn.service
[Unit]
Description=TSN service
After=network.target systemd-networkd.service

[Service]
#Type=forking
Type=oneshot
RemainAfterExit=yes
ExecStartPre=/bin/bash -c 'modprobe stm32_deip;modprobe edgx_pfm_lkm'
ExecStart=/usr/sbin/ttt-ip-init-systemd.sh start
ExecStop=/usr/sbin/ttt-ip-init-systemd.sh stop

[Install]
WantedBy=multi-user.target
```

Figure 2.3-3 st-tsn.service content

The Unit part

- Description=TSN service

This service is used for Time-Sensitive Networking (TSN).

- After=network.target systemd-networkd.service

Make sure that the network services (network.target and systemd-networkd.service) are started before starting the TSN service to ensure that the dependent network components are ready.

Service part

- #Type=forking

The service runs as a backprocess, which means that a script started by ExecStart will itself fork a child process and exit. The factory system is commented out by default.

- Type=oneshot is a type of a service unit (.service file) that executes a command once and exits without remaining running.
- RemainAfterExit=yes for oneshot, but if ttt-ip-init-systemd.sh is a continuously running process, it is recommended to change it to Type= forking
- ExecStartPre
  - modprobe stm32_deip: Loads the stm32_deip kernel module, possibly associated with the ST's TSN switch (ETHSW).
  - modprobe edgx_pfm_lkm: Loads the edgx_pfm_lkm kernel module, possibly switch management or TSN related modules.
- ExecStart

Execute /usr/sbin/ttt-ip-init-systemd.sh start, which starts the TSN-related services.

- ExecStop

stop the TSN service (/usr/sbin/ttt-ip-init-systemd.sh stop).

The Install section

- WantedBy=multi-user.target

This service is automatically started when the system enters the multi-user runtime level and is typically used in network or server mode for embedded devices.

Summary: The st-tsn.service is used to:

Ensure that the TSN-related kernel modules (stm32_deip and edgx_pfm_lkm) are loaded.

Execute the ttt-ip-init-systemd.sh script to initialize the TSN switching functionality.

Target is automatically started at the multi-user.target runtime level to ensure that the TSN switching function is available immediately after the device is started.

2.3.3 ttt-ip-init-systemd.sh

You can open /usr/sbin/ttt-ip-init-systemd.sh for yourself, but here are some important things:

(1) Get the MAC address

```
REF_ETH_INTERFACE=end1
get_mac() {
    read MAC </sys/class/net/$REF_ETH_INTERFACE/address
    echo "[INFO]: Mac Address of $REF_ETH_INTERFACE: $MAC"
}
```

Read the MAC address of end1 (ETH1), which is subsequently used for the MAC address configuration of sw0ep port.

(2) Detect the SoC device path

```
get_soc_path() {
    devicetree_path=$(ls -1 -d /sys/devices/platform/* | grep "/soc" | head -n 1)
    if [ -d "$devicetree_path" ];
    then
        SOC_PATH=$devicetree_path
    else
        echo "[ERROR]: /sys/devices/platform/soc* is not available"
        exit 1
    fi
}
```

Find the SoC device path by ls-1 -d /sys/devices/platform/* and store it in the SOC_PATH variable. This path is used to access the sysfs configuration of the TSN switch (ETHSW) device.

(3) Wait for the sysfs device to be created

```
wait_sysfs() {
    path=$1
    for i in $(seq 0 5)
    do
        if [ ! -e "$path" ]; then
            break;
        else
            sleep 0.5s
        fi
    done
}
```

Polling is performed up to five times (0.5 seconds each) to wait for the sysfs device file to appear and prevent further configuration before the device has been fully initialized.

(4) Configure the PHY of the TSN switch (ETHSW)

```
st_configure() {
    get_soc_path
    wait_sysfs $SOC_PATH/$IP_REF_NAME/net/sw0p3/phy/mdiobus
    if [ -e $SOC_PATH/$IP_REF_NAME/net/sw0p3/phy/mdiobus ]; then
        echo -n stmmac-1:04 > $SOC_PATH/$IP_REF_NAME/net/sw0p3/phy/mdiobus
```

```
            echo -n stmmac-1:05 > $SOC_PATH/$IP_REF_NAME/net/sw0p2/phy/mdiobus
        else
            echo "[ERROR]: $SOC_PATH/$IP_REF_NAME/net/sw0p3/phy/mdiobus not available"
            exit 1
        fi

        echo 170 > /sys/class/net/sw0p2/phy/delay1000tx_min
        echo 200 > /sys/class/net/sw0p2/phy/delay1000tx_max
        echo 170 > /sys/class/net/sw0p3/phy/delay1000tx_min
        echo 200 > /sys/class/net/sw0p3/phy/delay1000tx_max
        echo 520 > /sys/class/net/sw0p2/phy/delay1000rx_min
        echo 570 > /sys/class/net/sw0p2/phy/delay1000rx_max
        echo 520 > /sys/class/net/sw0p3/phy/delay1000rx_min
        echo 570 > /sys/class/net/sw0p3/phy/delay1000rx_max
    }
```

- PHY address configuration:

sw0p3 (switch port to which ETH1 is connected) corresponds to stmmac-1:04

sw0p2 (switch port to which ETH3 is connected) corresponds to stmmac-1:05

- PHY delay tuning:

The delay1000tx_min/max and delay1000rx_min/max of sw0p2 and sw0p3 are configured to optimize the TSN accuracy.

(5) Configuring the bridge

```
    set_interfaces_up() {
        get_mac
        ip link set dev sw0ep address $MAC
        ip link set dev sw0ep up
        ip addr add 192.168.0.10 dev sw0ep
        ip route add 192.168.0.0/24 dev sw0ep

        sleep 1

        ip link add name br0 type bridge
        ip link set dev br0 up
        ip link set dev sw0p1 master br0 up
        ip link set dev sw0p2 master br0 up
        ip link set dev sw0p3 master br0 up
        ip link set dev br0 up
        ip link set dev sw0ep up
    }
```

- Configure sw0ep (switch management interface)

Setting MAC address

Set static IP (192.168.0.10)

- Create the bridge br0

Bind ports sw0p1, sw0p2 and sw0p3

The bridge br0 is used for data forwarding

(6).Start the daemon

```
start_daemons() {
        systemctl stop systemd-timesyncd
        systemctl stop ntpd

        ip link set br0 type bridge stp_state 1
        mstpctl addbridge br0
        mstpctl setforcevers br0 mstp
        mstpctl setvid2fid br0 0:1

        systemctl start lldpd &
        systemctl start deptp &
        /usr/share/netopeer2-server/netopeer2-server-service start &
}
```

Stop the NTP service

Start mstpctl (STP Spanning Tree protocol) to ensure the stability of the bridge

Start lldpd (Link Layer Discovery Protocol)

Start deptp (possibly TSN-related precision time protocol)

Start netopeer2-server (NETCONF server)

(7) Stop the service

```
stop_daemons() {
        mstpctl delbridge br0
        systemctl stop lldpd &
        systemctl stop deptp &
        /usr/share/netopeer2-server/netopeer2-server-service stop
}
```

Stop mstpctl, lldpd, deptp, netopeer2-server

(8) Main entrance

```
case "$1" in
        start)
                start
                ;;
        stop)
                stop
              ;;
        restart)
                stop
                start
                ;;
        restore)
```

```
        /usr/share/netopeer2-server/netopeer2-server-service restore
        ;;
esac
exit 0
```

Support start, stop, restart, restore operations

Here's a quick summary of what this script does:

- Configure STM32MP2 switch (ETHSW) and related PHY
- Initialize the TSN bridge (br0) and port (sw0p) *
- Assign IP (192.168.0.10) and configure routing
- Start TSN related processes (deptp, lldpd, netopeer2-server)
- Support the start/stop/restart/restore management services

Chapter 3. Ethernet switch configuration method

ATK-DLMP257B development board factory system default support Ethernet switch function, use method refer to this section.

3.1 Network topology

The following is the Ethernet switch network topology of the ST, which can also be referenced by the ATK-DLMP257B development board.

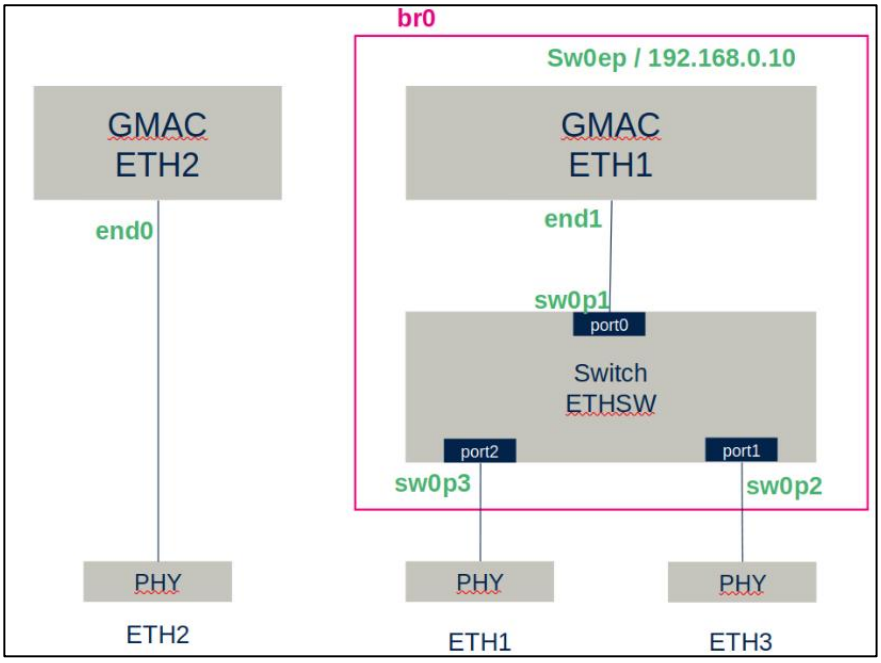


Figure 3.1-1 Ethernet switch network topology

The network architecture shows two independent Ethernets (ETH2, ETH1) as well as an integrated switch (ETHSW) with a bridge formed by br0.

1. Main components

ETH2

- GMAC (Gigabit Media Access Controller)
- Connect to the PHY
- Independent interface, no switch access

ETH1

- GMAC
- Connect to switch (ETHSW)
- Connect GMAC via switch port port0
- Share switching function with ETH3 (sw0p2) and ETH1 (sw0p3)

Switch (ETHSW)

- port0 connect to ETH1 (GMAC)
- port1 (sw0p2) connect PHY (ETH3)
- port2 (sw0p3) connect PHY (ETH1)

br0 (network bridge)

- Merge the ETH1 (GMAC) and ETHSW switch ports
- Sw0ep as the bridge IP (192.168.0.10)
- ETH2 does not join br0 and is an independent interface

2. Network function analysis

- ETH1 connects to the switch (ETHSW) and extends multiple ports (ETH1, ETH3).
- ETH2 is directly connected to the PHY, independent of the switching network.
- Switch ETHSW allows data communication between ETH1, ETH3.
- br0 is used as a bridge interface for unified management of ETH1 and ETHSW, and 192.168.0.10 is set as IP.

3. Application scenarios

- ETH2 can be used for independent network communication, such as connecting different subnetworks or management interfaces.
- The ETH1 + switch is used for intra-LAN switching, can access multiple devices, and share IP with br0.
- br0 allows access to ETH1/ETH3 ports via Sw0ep (192.168.0.10), suitable for embedded routing, gateway, or switching functions.

3.2 Basic function test of Ethernet switch

3.2.1 Hardware connectivity

The development board ETH2 is connected to device 1, the development board ETH3 is connected to device 2, and the development board ETH1 is not connected to peripherals. The author here device 1 and device 2 use MP157 development board as a test, any device that can use Ethernet function can be used. The ATK-DLMP257B development board serves as a switch routing relay.

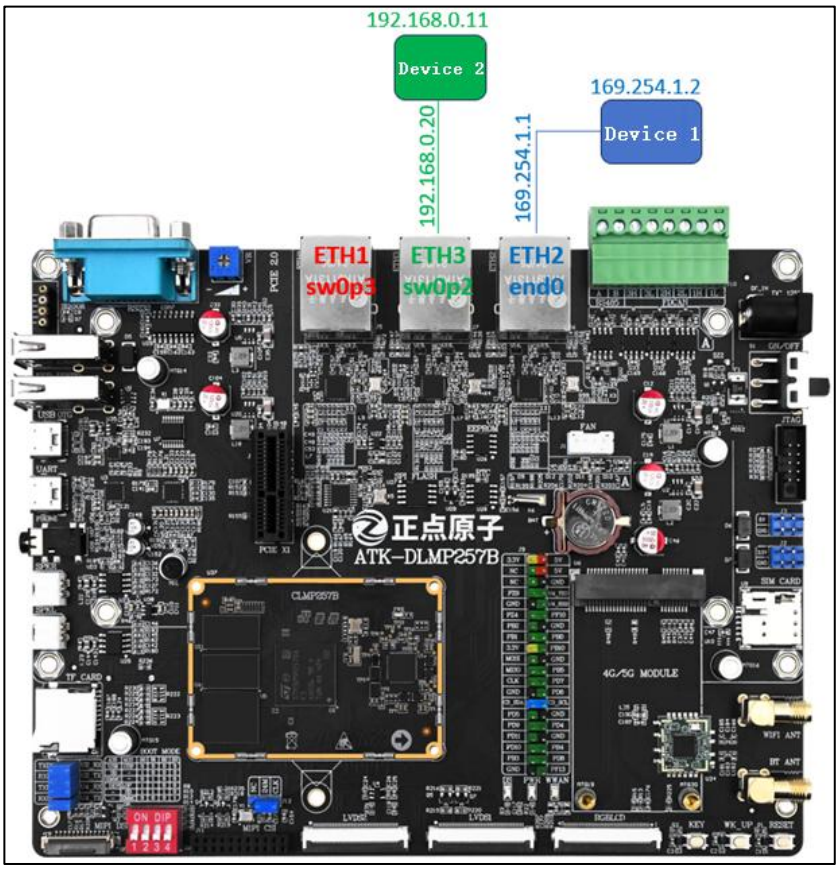


Figure 3.2-1 Hardware connection

3.2.2 Configuration of forwarding function

The ATK-DLMP257B development board is configured according to the topology structure, the device is started and the end1/sw0ep interface is set. At the same time, the IP forwarding function is enabled, so that the device can act as a router and forward network traffic.

The ATK-DLMP257 configuration is as follows:

```
ifconfig end0 169.254.1.1 up
ifconfig sw0p2 192.168.0.20 up
echo 1 > /proc/sys/net/ipv4/ip_forward
```

`echo 1 > /proc/sys/net/ipv4/ip_forward` is used temporarily to enable the IP forwarding function of the kernel. It is invalid after restarting. To use it permanently try echo"`net.ipv4.ip_forward = 1" >> /etc/sysctl.conf sysctl -p`

Configure device 1 (development board or PC) network interface IP and routing, here device 1 uses an MP157 development board, configured on the 169.254.1.x network segment:

```
ifconfig eth0 169.254.1.2 up
ip route add default via 169.254.1.1
```

Configure Device 2 (development board or PC) network IP and routing, here device 2 also uses an MP157 development board, configured on the 192.168.0.x network segment:

```
ifconfig eth0 192.168.0.11 up
ip route add default via 192.168.0.10
```

3.2.3 Connectivity verification

From device 1 to device 2, when ping 192.168.0.11 is executed, the packet is sent from device 1 to the default gateway ATK-DLMP257B (169.254.1.1), ATK-DLMP257B forwards the data packet to sw0p2 interface (192.168.0.20) according to the routing table. Device 2 receives the packet and replies, and the same is true for the reverse path.

ping 192.168.0.11

```
root@ATK-MP157:~# ping 192.168.0.11
PING 192.168.0.11 (192.168.0.11) 56(84) bytes of data.
64 bytes from 192.168.0.11: icmp_seq=1 ttl=63 time=1.78 ms
64 bytes from 192.168.0.11: icmp_seq=2 ttl=63 time=0.944 ms
64 bytes from 192.168.0.11: icmp_seq=3 ttl=63 time=0.941 ms
64 bytes from 192.168.0.11: icmp_seq=4 ttl=63 time=0.939 ms
64 bytes from 192.168.0.11: icmp_seq=5 ttl=63 time=0.910 ms
64 bytes from 192.168.0.11: icmp_seq=6 ttl=63 time=0.905 ms
64 bytes from 192.168.0.11: icmp_seq=7 ttl=63 time=0.916 ms
^C
--- 192.168.0.11 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6007ms
rtt min/avg/max/mdev = 0.905/1.047/1.778/0.298 ms
root@ATK-MP157:~#
```

Figure 3.2-2 Device 1 ping device 2 test

From device 2 to device 1, same process:

```
root@ATK-MP157:~# ping 169.254.1.2
PING 169.254.1.2 (169.254.1.2) 56(84) bytes of data.
64 bytes from 169.254.1.2: icmp_seq=1 ttl=63 time=1.54 ms
64 bytes from 169.254.1.2: icmp_seq=2 ttl=63 time=0.973 ms
64 bytes from 169.254.1.2: icmp_seq=3 ttl=63 time=0.928 ms
64 bytes from 169.254.1.2: icmp_seq=4 ttl=63 time=0.983 ms
64 bytes from 169.254.1.2: icmp_seq=5 ttl=63 time=0.968 ms
64 bytes from 169.254.1.2: icmp_seq=6 ttl=63 time=0.930 ms
64 bytes from 169.254.1.2: icmp_seq=7 ttl=63 time=0.952 ms
64 bytes from 169.254.1.2: icmp_seq=8 ttl=63 time=0.934 ms
64 bytes from 169.254.1.2: icmp_seq=9 ttl=63 time=0.893 ms
^C
--- 169.254.1.2 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8011ms
rtt min/avg/max/mdev = 0.893/1.010/1.536/0.187 ms
root@ATK-MP157:~#
```

Figure 3.2-3 Device 2 ping Device 1 test

The ATK-DLMP257B board was used as a routing relay to successfully realize the communication between two different subnets (169.254.1.0/24 and 192.168.0.0/24).

3.3 Dual-network segment and multi-device test

3.3.1 Hardware connection

Development board ETH2 is connected to device 1, development board ETH3 is connected to device 2, development board ETH1 is connected to device 3, ETH3 and ETH2 need to be in the same network segment.

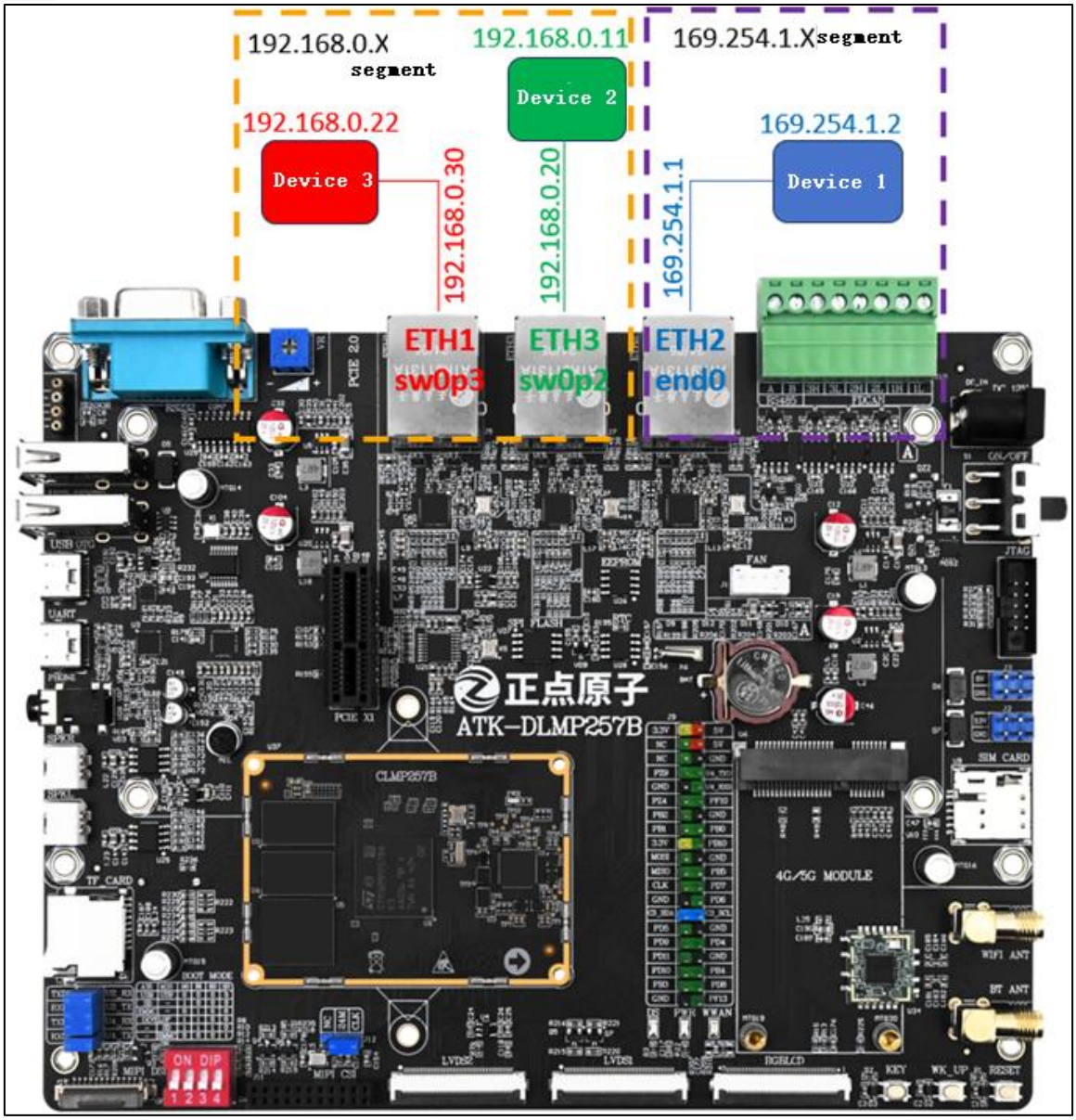


Figure 3.3-1 Hardware connection

3.3.2 Configuration of forwarding function

The ATK-DLMP257B development board is configured according to the topology structure, the device is started and the end1/sw0ep interface is set. At the same time, the IP forwarding function is enabled, so that the device can act as a router and forward network traffic.

The ATK-DLMP257 configuration is as follows:

```
ifconfig end0 169.254.1.1 up
ifconfig sw0p2 192.168.0.20 up
ifconfig sw0p3 192.168.0.30 up
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Configure device 1 (development board or PC) network interface IP and routing, here device 1 uses an MP157 development board:

```
ifconfig eth0 169.254.1.2 up
```

```
ip route add default via 169.254.1.1
```

Configure device 2 (development board or PC) network interface IP and routing, here device 2 also uses an MP157 development board:

```
ifconfig eth0 192.168.0.11 up
ip route add default via 192.168.0.10
```

Configure device 3 (development board or PC) network interface IP and routing, here device 3 also uses an MP157 development board:

```
ifconfig eth0 192.168.0.22 up
ip route add default via 192.168.0.10
```

3.3.3 Connectivity testing

From device 1 to device 2:

```
ping 192.168.0.11
```

From device 1 to device 3:

```
ping 192.168.0.22
```

```
root@ATK-MP157:~# ping 192.168.0.11
PING 192.168.0.11 (192.168.0.11) 56(84) bytes of data.
64 bytes from 192.168.0.11: icmp_seq=1 ttl=63 time=1.83 ms
64 bytes from 192.168.0.11: icmp_seq=2 ttl=63 time=0.954 ms
^C
--- 192.168.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.954/1.391/1.829/0.437 ms
root@ATK-MP157:~# ping 192.168.0.22
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
64 bytes from 192.168.0.22: icmp_seq=1 ttl=63 time=1.44 ms
64 bytes from 192.168.0.22: icmp_seq=2 ttl=63 time=0.874 ms
^C
--- 192.168.0.22 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.874/1.156/1.438/0.282 ms
```

Figure 3.3-2 Device 1 ping device 2 and device 3 test

From device 2 to device 1:

```
ping 169.254.1.2
```

From device 2 to device 3:

```
ping 192.168.0.22
```

```
root@ATK-MP157:~# ping 169.254.1.2
PING 169.254.1.2 (169.254.1.2) 56(84) bytes of data.
64 bytes from 169.254.1.2: icmp_seq=1 ttl=63 time=0.877 ms
64 bytes from 169.254.1.2: icmp_seq=2 ttl=63 time=0.904 ms
^C
--- 169.254.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.877/0.890/0.904/0.013 ms
root@ATK-MP157:~# ping 192.168.0.22
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
64 bytes from 192.168.0.22: icmp_seq=1 ttl=64 time=1.04 ms
64 bytes from 192.168.0.22: icmp_seq=2 ttl=64 time=0.482 ms
^C
--- 192.168.0.22 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.482/0.758/1.035/0.276 ms
```

Figure 3.3-3 Device 2 ping Device 1 and device 3 test

From device 3 to device 1:

```
ping 169.254.1.2
```

From device 3 to device 2:

```
ping 192.168.0.11
```

```
root@ATK-MP157:~# ping 169.254.1.2
PING 169.254.1.2 (169.254.1.2) 56(84) bytes of data.
64 bytes from 169.254.1.2: icmp_seq=1 ttl=63 time=0.916 ms
64 bytes from 169.254.1.2: icmp_seq=2 ttl=63 time=0.926 ms
^C
--- 169.254.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.916/0.921/0.926/0.005 ms
root@ATK-MP157:~# ping 192.168.0.11
PING 192.168.0.11 (192.168.0.11) 56(84) bytes of data.
64 bytes from 192.168.0.11: icmp_seq=1 ttl=64 time=0.513 ms
64 bytes from 192.168.0.11: icmp_seq=2 ttl=64 time=0.393 ms
^C
--- 192.168.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1052ms
rtt min/avg/max/mdev = 0.393/0.453/0.513/0.060 ms
```

Figure 3.3-4 Device 3 ping Device 1 and device 2 test

3.4 Test of Ethernet switch connecting to Internet

3.4.1 Hardware connectivity

The ATK-DLMP257B development board end0 (ETH2) is connected to the Internet, and the devices two and three are connected to the sw0p2\sw0p3 (ETH3\ETH1) of the ATK-DLMP257B development board, so that the devices two and three should ping the ATK-DLMP257B development board. And device two or three should be able to connect to the external network.

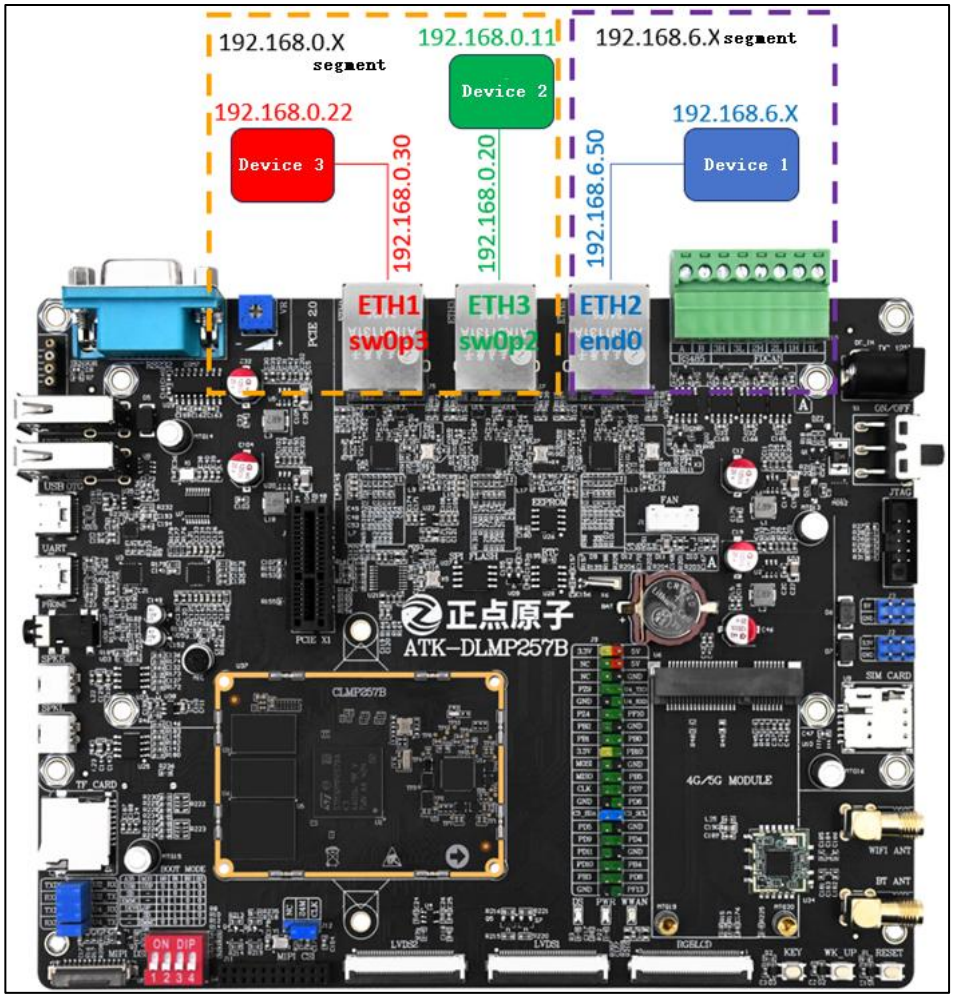


Figure 3.4-1 Hardware connection

3.4.2 Using iptables for NAT translation

On the basis of the previous expansion, device 2/3 can ping ATK-DLMP257B development board, but cannot connect to external network. At this point, the ATK-DLMP257B development board needs to be set as follows to use iptables for NAT conversion:

```
iptables -t nat -A POSTROUTING -o end0 -j MASQUERADE
```

Make sure the NAT rules work:

```
iptables -t nat -L -n -v
```

You should see the following instructions:

```
Chain POSTROUTING (policy ACCEPT 2 packets, 516 bytes)
 pkts bytes target     prot opt in     out     source               destination
    8  1297 MASQUERADE  0    --  *      end0    0.0.0.0/0            0.0.0.0/0
```

Make sure iptables rules are not lost:

```
iptables-save > /etc/iptables.rules
```

```
root@ATK-DLMP257:~# ifconfig end0 169.254.1.1 up
root@ATK-DLMP257:~# ifconfig sw0p2 192.168.0.20 up
root@ATK-DLMP257:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@ATK-DLMP257:~# iptables -t nat -A POSTROUTING -o end0 -j MASQUERADE
root@ATK-DLMP257:~# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 63 packets, 4764 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 2 packets, 512 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain POSTROUTING (policy ACCEPT 1 packets, 257 bytes)
 pkts bytes target     prot opt in     out     source               destination

    1   255 MASQUERADE  0    --  *      end0    0.0.0.0/0            0.0.0.0/0

root@ATK-DLMP257:~# iptables-save > /etc/iptables.rules
```

Figure 3.4-2 Use iptables for NAT translation

3.4.3 Setup and test of networking equipment

Device 2\3 manually set dns:

```
echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

At this point, devices 2 and 3 can connect to the external network:

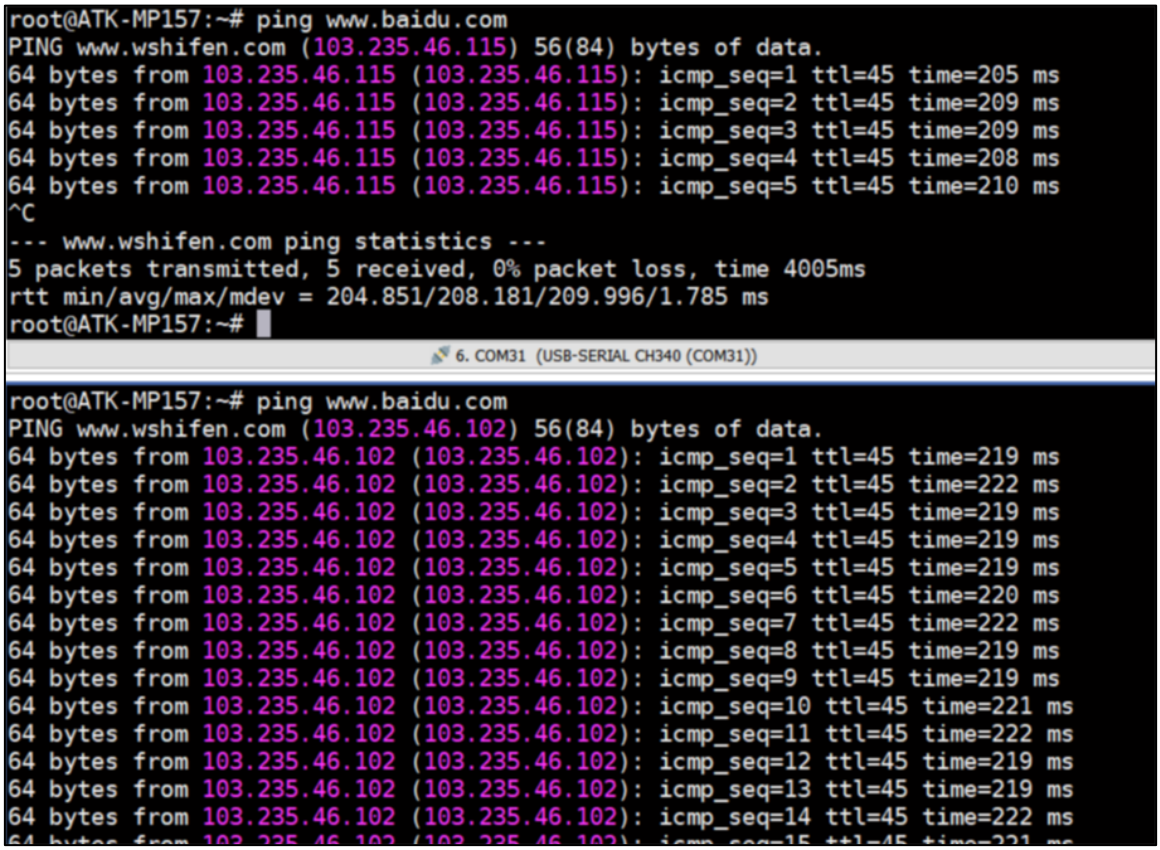


Figure 3.4-3 Device 2\3 networking test