Ethernet overview

Ethernet overview

Stable: 31.01.2020 - 14:01 / Revision: 31.01.2020 - 13:51

Template:ArticleMainWriter Template:ArticleApprovedVersion

SUMMARY

This article gives information about the Linux® Ethernet framework, provides its composition and explains how to configure and use it.

Contents

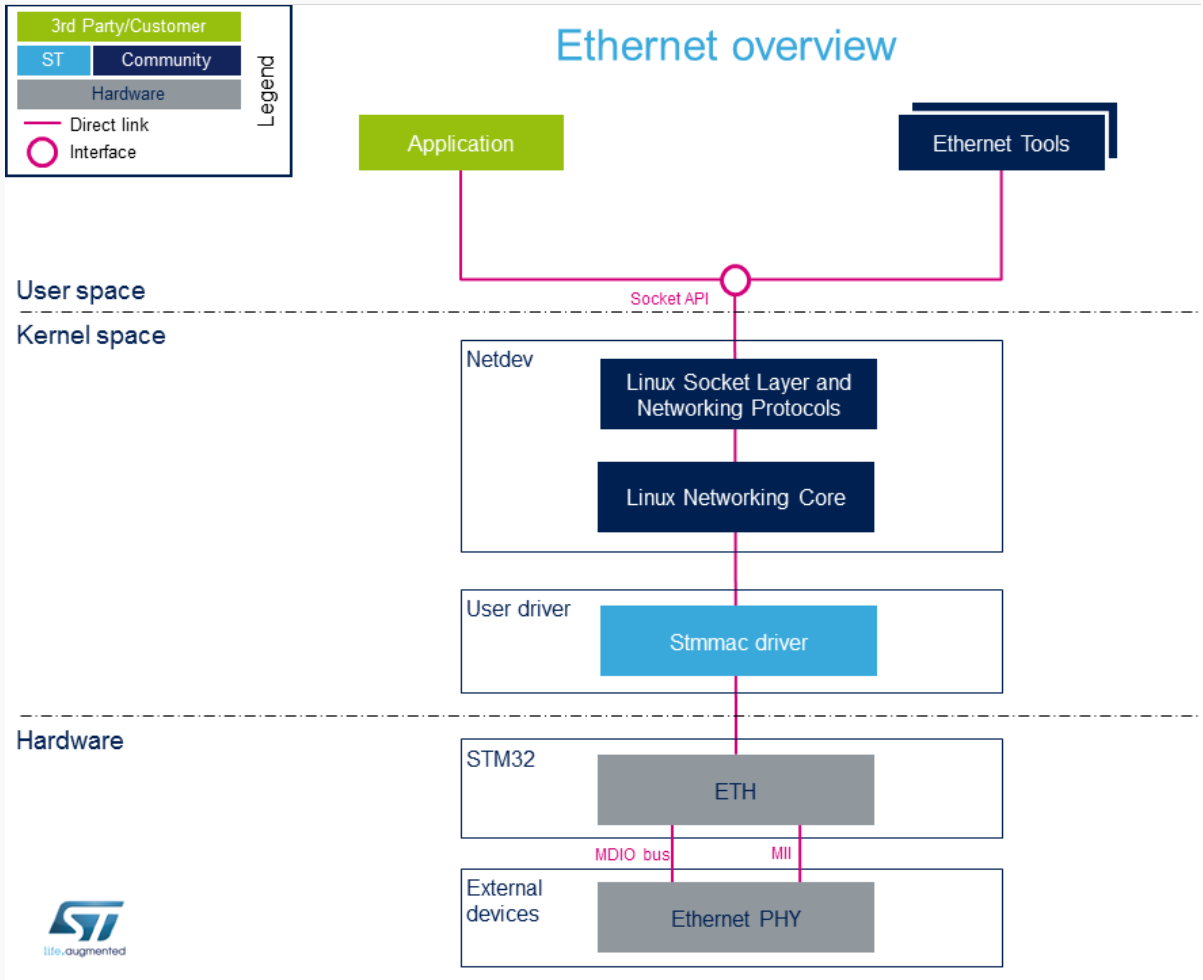
1 Framework purpose

Ethernet is a way of connecting devices together in a local area network or LAN. An Ethernet protocol is used to transmit packets of data containing any sort of information. Any two devices that are connected to the network can exchange information through an Ethernet connection. Ethernet provides a fast, efficient, and direct connection to a router.

Ethernet can be used in many different use cases, as mentioned in How to use Ethernet section:

- How to perform remote connection SSH
- How to perform ping test PING

2 System overview



Ethernet Overview

2.1 Component description

From User space to hardware

- **Application** (User space)

There are a lot of applications using ethernet: Internet Browser, Streaming applications, FTP applications etc..
The main interface that is used between an application and the Networking protocols is a socket [1]

- **Ethernet tools** (User space)

A set of utilities is available to manage and maintain networks: ethtool, ping, route, ifconfig etc..

- **Linux Socket Layer and Networking Protocols** (Kernel space)

The socket layer [2] is a uniform interface between the user process and the network protocol [3] stacks within the kernel

- **Linux Networking Core** (Kernel space)

The kernel network layer adapts the message with the transport protocol in use. The network subsystem of the Linux kernel is designed to be completely protocol-independent.

- **Stmmac Driver** (Kernel space)

This is the driver for the MAC 10/100/1000 on-chip Ethernet controllers (Synopsys IP blocks).
Documentation/networking/stmmac.txt[4]

- **ETH** (Hardware)

This is the Ethernet IP: GMAC [5]

- **Ethernet phy** (Hardware)

The Ethernet PHY is connected to a media access controller (MAC). The MAC controls the data-link-layer portion of the OSI model.
The media-independent interface (MII) defines the interface between the MAC and the PHY.
Variations of the MII are available (RGMII, GMII, RMII, MII) that provide minimal pin count and varied data rates depending on system requirements.
The MDIO bus includes two signals:
- MDC clock: driven by the MAC device to the PHY.
- MDIO data: bidirectional, it is driven by the PHY to provide register data at the end of a read operation.
The connector used by ethernet phy is RJ45.

2.2 API description

The Ethernet API is documented in the Linux Kernel https://www.kernel.org/doc/html/v4.14/networking/kapi.html

3 Configuration

3.1 Kernel configuration

The Ethernet API is activated by default in ST deliveries. Nevertheless, if a specific configuration is required, one can use Linux Menuconfig tool: Menuconfig or how to configure kernel and select:

For Network features:

```
[*] Networking support  --->
    [*] Networking options  --->
        [*] Packet socket
        [*] TCP/IP networking
          [*] IP: kernel level autoconfiguration
            [*] IP: DHCP support
            [*] IP: BOOTP support
            [*] IP: RARP support
        [*] INET: socket monitoring interface
        [*] The IPv6 protocol
        [*] DNS Resolver support
```

For Phy (Generic PHY support) :

```
[*] Device Drivers  --->
  [*] PHY Subsystem   --->
    [*] PHY Core
```

For STM32 DWMAC :

```
[*] Device Drivers  --->
    [*]   Network device support  --->
        [*]   Ethernet driver support  --->
            [*]   STMicroelectronics devices
               [*]   STMicroelectronics 10/100/1000/EQOS Ethernet driver
               [*]   STMMAC Platform bus support
                  [*]   Generic driver for DWMAC
                  [*]   STM32 DWMAC support
```

3.2 Device tree configuration

DT bindings documentation deals with all required or optional device tree properties.

Detailed DT configuration for STM32 internal peripherals: Ethernet device tree configuration.

4 How to use Ethernet

4.1 How to use the Ethernet user space interface

Please see examples based on the following use cases:

- How to configure ethernet interface: How to configure ethernet interface
- How to perform ssh connection: How to perform ssh connection
- How to perform ping test: How to perform ping test

5 How to trace and debug the framework

5.1 How to monitor

5.1.1 How to monitor with sysfs

sysfs entry can be used to browse for available descriptors and hardware capabilies.

```
Board $> /sys/kernel/debug/stmmaceth/eth0# ls
   descriptors_status  dma_cap
 root@stm32mp1://sys/kernel/debug/stmmaceth/eth0# cat descriptors_status
```

```
RX Queue 0:
Descriptor ring:
0 [0xf4e8d000]: 0xecb01842 0x0 0x0 0x81000000
1 [0xf4e8d010]: 0xecb02042 0x0 0x0 0x81000000
....
root@stm32mp1://sys/kernel/debug/stmmaceth/eth0# cat dma_cap
==============================
        DMA HW features
==============================
     10/100 Mbps: Y
     1000 Mbps: Y
     Half duplex: Y
     Hash Filter: Y
     Multiple MAC address registers: Y
     PCS (TBI/SGMII/RTBI PHY interfaces): N
     SMA (MDIO) Interface: Y
     PMT Remote wake up: Y
     PMT Magic Frame: Y
     RMON module: Y
     IEEE 1588-2002 Time Stamp: N
     IEEE 1588-2008 Advanced Time Stamp: Y
     802.3az - Energy-Efficient Ethernet (EEE): Y
     AV features: Y
     Checksum Offload in TX: Y
     IP Checksum Offload in RX: Y
     RXFIFO > 2048bytes: N
     Number of Additional RX channel: 1
     Number of Additional TX channel: 2
     Enhanced descriptors: N
```

5.1.2 Other ways of monitoring

Ethtool is a Linux-based utility for displaying and modifying some parameters of the network interface controllers (NICs) and their device drivers.

```
Board $> ethtool eth0
      Settings for eth0:
      Supported ports: [ TP AUI BNC MII FIBRE ]
      Supported link modes:   10baseT/Half 10baseT/Full
                              100baseT/Half 100baseT/Full
                              1000baseT/Half 1000baseT/Full
      Supported pause frame use: Symmetric Receive-only
      Supports auto-negotiation: Yes
      Advertised link modes:  10baseT/Half 10baseT/Full
                              100baseT/Half 100baseT/Full
                              1000baseT/Half 1000baseT/Full
      Advertised pause frame use: No
      Advertised auto-negotiation: Yes
      Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                           100baseT/Half 100baseT/Full
                                           1000baseT/Full
      Link partner advertised pause frame use: Symmetric
      Link partner advertised auto-negotiation: Yes
      Speed: 1000Mb/s
      Duplex: Full
      Port: MII
      PHYAD: 0
      Transceiver: internal
      Auto-negotiation: on
      Supports Wake-on: ug
      Wake-on: d
      Current message level: 0x0000003f (63)
                             drv probe link timer ifdown ifup
      Link detected: yes
```

5.2 How to trace

The Ethernet Framework (and specifically the stmmac driver) prints out information and error messages in the kernel console. They are available via dmesg command:

```
Board $> dmesg | grep ethernet
[    1.454632] stm32-dwmac 5800a000.ethernet: PTP uses main clock
[    1.459010] stm32-dwmac 5800a000.ethernet: no reset control found
[    1.465199] stm32-dwmac 5800a000.ethernet: No phy clock provided...
[    1.472347] stm32-dwmac 5800a000.ethernet: User ID: 0x40, Synopsys ID: 0x42
[    1.478319] stm32-dwmac 5800a000.ethernet:   DWMAC4/5
[    1.483310] stm32-dwmac 5800a000.ethernet: DMA HW capability register supported
[    1.490564] stm32-dwmac 5800a000.ethernet: RX Checksum Offload Engine supported
[    1.497888] stm32-dwmac 5800a000.ethernet: TX Checksum insertion supported
[    1.504753] stm32-dwmac 5800a000.ethernet: Wake-Up On Lan supported
[    1.510994] stm32-dwmac 5800a000.ethernet: TSO supported
[    1.516329] stm32-dwmac 5800a000.ethernet: TSO feature enabled
[    1.522143] stm32-dwmac 5800a000.ethernet: Enable RX Mitigation via HW Watchdog Timer
[   12.356485] stm32-dwmac 5800a000.ethernet eth0: No Safety Features support found
[   12.426208] stm32-dwmac 5800a000.ethernet eth0: IEEE 1588-2008 Advanced Timestamp suppo
[   12.481051] stm32-dwmac 5800a000.ethernet eth0: registered PTP clock
[   14.951370] stm32-dwmac 5800a000.ethernet eth0: Link is Up - 1Gbps/Full - flow control
```

It is possible to modify the amount of 'debugging messages/data' returned by the Ethernet driver with ethtool. More documentation is available in Documentation/networking/netif-msg.txt[6] in kernel source folder.

Ethtool to set the message level:

```
Board $> ethtool -s eth1 msglvl [level]
```

5.3 How to debug

During Ethernet bring up, there are 2 frequent errors:

- DMA reset error:

```
[   15.650981] dwmac4_dma_reset err
[   15.652849] stm32-dwmac 5800a000.ethernet: Failed to reset the dma
[   15.659006] stm32-dwmac 5800a000.ethernet eth0: stmmac_hw_setup: DMA engine initializat
[   15.668518] stm32-dwmac 5800a000.ethernet eth0: stmmac_open: Hw setup failed
```

When this error occurs, it is linked to the DMA Software Reset (not linked to memory transfert)

Definition of the Software Reset in GMAC specification:

```
When this bit is set, the MAC and the DMA controller reset the logic
and all internal registers of the DMA, MTL, and MAC. This bit is
automatically cleared after the reset operation is complete in all
DWC_ether_qos clock domains. Before reprogramming any DWC_ether_qos
register, a value of zero should be read in this bit.
*Note*: The reset operation is complete only when all resets in all
active clock domains are de-asserted. Therefore, it is essential that
```

```
all PHY inputs clocks (applicable for the selected PHY interface) are
present for software reset completion. The time to complete the
software reset operation depends on the frequency of the slowest active clock.
Access restriction applies. Setting 1 sets. Self-cleared. Setting 0
has no effect.
```

- Ethernet clock tree error:

The GMAC IP verifies that the Ethernet clock tree is well configured. When this error occurs, it is due to the Ethernet PHY that do not detect all needed clocks (tx, rx, aclk or hclk).

To solve this issue:

```
- check that the pinctrl of each clock is well configured
- check if syscfg register is well configured (in Ethernet clock tree there are some gati
```

6 Source code location

The source files are located inside the Linux kernel.

- **Ethernet driver**: dwmac-stm32.c[7]

7 References

- â [1], Berkeley sockets
- â [2], Socket Layer
- â [3], Internet Protocol
- â https://www.kernel.org/doc/Documentation/networking/stmmac.txt, More information
- â [4], DesignWare Ethernet GMAC IP
- â [5], Documentation/networking/netif-msg.txt
- â [6],dwmac-stm32.c