Category:Ethernet

Contents

Stable: 17.06.2020 - 15:26 / Revision: 16.01.2020 - 08:06

Stable: 17.06.2020 - 10:20 | Revision: 16.01.2020 - 08:00

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to the Linux® **ethernet** software framework.

It is recommended to first read the Ethernet overview article.

Linux® is a registered trademark of Linus Torvalds.

Pages in category "Ethernet"

The following 5 pages are in this category, out of 5 total.

- Ethernet device tree configuration
- Ethernet overview
- How to configure ethernet interface
- How to perform ping test
- How to perform ssh connection

Stable: 04.05.2021 - 08:18 / Revision: 04.05.2021 - 08:17

A checked version of this page, approved on *4 May 2021*, was based off this revision.

Contents

1 Article purpose

This article explains how to configure the Ethernet when it is assigned to the Linux®OS. In that case, it is controlled by the Ethernet framework

The configuration is performed using the device tree mechanism that provides a hardware description of the Ethernet peripheral, used by the STM32 DWMAC driver

2 DT bindings documentation

The *Ethernet* is a multifunction device.

Each function is represented by a separate binding document:

- "Generic" Ethernet device tree bindings [1]
- specific STM32 ETH device tree bindings[2]

3 DT configuration

This hardware description is a combination of the **STM32 microprocessor** device tree files (*.dtsi* extension) and **board** device tree files (*.dts* extension). See the Device tree for an explanation of the device tree file split.

3.1 DT configuration (STM32 level)

Ethernet peripheral nodes are located in stm32mp151.dtsi [3] file with a disabled status and some required properties such as:

- Physical base address and size of the device register map
- STMMAC interrupts
- stmmaceth clock and Rx, Tx clocks

This is a set of properties that may not vary for a given STM32MP device, such as: register addresses, interrupts, clocks, ...

```
ethernet0: ethernet@5800a000 {
        compatible = "st,stm32mp1-dwmac", "snps,dwmac-4.20a";
        reg = <0x5800a000 0x2000>;
        reg-names = "stmmaceth";
        interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_NONE>;
        interrupt-names = "macirq";
        clock-names = "stmmaceth",
                      "mac-clk-tx",
                      "mac-clk-rx",
                      "ethstp";
        clocks = <&rcc ETHMAC>,
                 <&rcc ETHTX>,
                 <&rcc ETHRX>,
                 <&rcc ETHSTP>;
        st,syscon = <&syscfg 0x4>;
        snps,mixed-burst;
        snps,pbl = <2>;
        snps,axi-config = <&stmmac_axi_config_0>;
        snps,tso;
        power-domains = <&pd_core>;
        status = "disabled";
};
```

The required and optional properties are fully described in the bindings files.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 Ethernet DT configuration (board level)

The device tree board file (.dts) contains all hardware configurations related to board design. The DT node (**"ethernet"**) should be updated to:

- Enable the Ethernet block by setting **status = "okay".**
- Configure the pins in use via pinctrl, through **pinctrl-0** (default pins), **pinctrl-1** (sleep pins) and **pinctrl-names**.
- Configure Ethernet interface used **phy-mode = "rgmii".**, (rmii, mii, gmii).

- Configure Ethernet max speed **max-speed = <1000>"**..

```
&ethernet0 {
        status = "okay";
        pinctrl-0 = <&ethernet0_rgmii_pins_a>;
        pinctrl-1 = <&ethernet0_rgmii_pins_sleep_a>;
        pinctrl-names = "default", "sleep";
        phy-mode = "rgmii";
        max-speed = <1000>;
        phy-handle = <&phy0>;

        mdio0 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "snps,dwmac-mdio";
                phy0: ethernet-phy@1 {
                        reg = <1>;
                };
        };
};
```

3.3 DT configuration examples

The example below shows how to configure and enable an Ethernet instance at board level:

```
&ethernet0 {
   status = "okay";                                /* enable ethernet0 */
   pinctrl-0 = <&ethernet0_rmii_pins_a>;           /* configure pinctrl modes for ethernet0
*/
   pinctrl-1 = <&ethernet0_rmii_pins_sleep_a>;  /* configure ethernet0_rmii_pins_sleep_a
as sleep pinctrl configuration for ethernet0 */
   pinctrl-names = "default", "sleep";
   phy-mode = "rmii";                              /* configure ethernet phy mode for
ethernet0 */
   max-speed = <100>;                              /* configure ethernet max speed for
ethernet0 */
   phy-handle = <&phy0>;

   mdio0 {
       #address-cells = <1>;
       #size-cells = <0>;
       compatible = "snps,dwmac-mdio";
       phy0: ethernet-phy@1 {
           reg = <1>;                              /* configure ethernet phy @ for ethernet0
*/
       };
   };
};
```

How to configure Ethernet for :

3.3.1 RMII with Crystal on PHY (Reference clock (standard RMII clock name) is provided by a Phy Crystal)

```
   ethernet0: ethernet@5800a000 {
       compatible = "st,stm32mp1-dwmac", "snps,dwmac-4.20a";
       reg = <0x5800a000 0x2000>;
       reg-names = "stmmaceth";
       interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>,
                             <&intc GIC_SPI 62 IRQ_TYPE_LEVEL_HIGH>,
```

```
                                <&exti 70 1>;
        interrupt-names = "macirq",
                          "eth_wake_irq",
                          "stm32_pwr_wakeup";
        clock-names = "stmmaceth",
                      "mac-clk-tx",
                      "mac-clk-rx",
                      "ethstp";
        clocks = <&rcc ETHMAC>,
                 <&rcc ETHTX>,
                 <&rcc ETHRX>,
                 <&rcc ETHSTP>;
        st,syscon = <&syscfg 0x4>;
        snps,mixed-burst;
        snps,pbl = <2>;
        snps,en-tx-lpi-clockgating;
        snps,axi-config = <&stmmac_axi_config_0>;
        snps,tso;
        power-domains = <&pd_core>;
        status = "disabled";
};
```

```
    &ethernet0 {
        status = "okay";
        pinctrl-0 = <&ethernet0_rmii_pins_a>;
        pinctrl-1 = <&ethernet0_rmii_pins_sleep_a>;
        pinctrl-names = "default", "sleep";
        phy-mode = "rmii";
        max-speed = <100>;
        phy-handle = <&phy0>;
        mdio0 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "snps,dwmac-mdio";
                phy0: ethernet-phy@0 {
                        reg = <0>;
                };
        };
    };
```

3.3.2 RMII with 25MHz on ETH_CLK (no PHY Crystal), REF_CLK from PHY (Reference clock (standard RMII clock name) is provided by a RCC SoC internal clock)

```
    ethernet0: ethernet@5800a000 {
        compatible = "st,stm32mp1-dwmac", "snps,dwmac-4.20a";
        reg = <0x5800a000 0x2000>;
        reg-names = "stmmaceth";
        interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>,
                              <&intc GIC_SPI 62 IRQ_TYPE_LEVEL_HIGH>,
                              <&exti 70 1>;
        interrupt-names = "macirq",
                          "eth_wake_irq",
                          "stm32_pwr_wakeup";
        clock-names = "stmmaceth",
                      "eth-ck",
                      "mac-clk-tx",
                      "mac-clk-rx",
                      "ethstp";
        clocks = <&rcc ETHMAC>,
                 <&rcc ETHCK_K>,
                 <&rcc ETHTX>,
                 <&rcc ETHRX>,
```

```
                <&rcc ETHSTP>;
        st,syscon = <&syscfg 0x4>;
        snps,mixed-burst;
        snps,pbl = <2>;
        snps,en-tx-lpi-clockgating;
        snps,axi-config = <&stmmac_axi_config_0>;
        snps,tso;
        power-domains = <&pd_core>;
        status = "disabled";
};
```

```
   &ethernet0 {
        status = "okay";
        pinctrl-0 = <&ethernet0_rmii_pins_a>;
        pinctrl-1 = <&ethernet0_rmii_pins_sleep_a>;
        pinctrl-names = "default", "sleep";
        phy-mode = "rmii";
        max-speed = <100>;
        phy-handle = <&phy0>;
        mdio0 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "snps,dwmac-mdio";
                phy0: ethernet-phy@0 {
                        reg = <0>;
                };
        };
   };
```

+ update stm32mp15-pinctrl.dtsi [4] to add ETHCK pin in ethernet0_rmii_pins_* node:
For example:

```
 <STM32_PINMUX('G', 8, AF2)>, /* ETH_RMII_ETHCK */
```

+ Need also to update TFA devicetree to generate 25Mhz clock (from PLL4P or PLL3Q):
for example if PLL4P in ed1 board: update fdts/stm32mp15xx-edx.dtsi

```
st,pkcs = <
 CLK_CKPER_HSE
 CLK_FMC_ACLK
 CLK_QSPI_ACLK
 - CLK_ETH_DISABLED
 + CLK_ETH_PLL4P
...
 /* VCO = 600.0 MHz => P = 25, Q = 50, R = 50 */
   pll4: st,pll@3 {
     compatible = "st,stm32mp1-pll";
     reg = <3>;
     cfg = < 1 49 23 11 11 PQR(1,1,1) >;
 };
```

3.3.3 RMII with 50MHz on ETH_CLK (no PHY Crystal), internal REF_CLK from RCC (Reference clock (standard RMII clock name) is provided by a RCC SoC internal clock)

```
   ethernet0: ethernet@5800a000 {
        compatible = "st,stm32mp1-dwmac", "snps,dwmac-4.20a";
        reg = <0x5800a000 0x2000>;
```

```
        reg-names = "stmmaceth";
        interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>,
                              <&intc GIC_SPI 62 IRQ_TYPE_LEVEL_HIGH>,
                              <&exti 70 1>;
        interrupt-names = "macirq",
                          "eth_wake_irq",
                          "stm32_pwr_wakeup";
        clock-names = "stmmaceth",
                      "eth-ck",
                      "mac-clk-tx",
                      "mac-clk-rx",
                      "ethstp";
        clocks = <&rcc ETHMAC>,
                 <&rcc ETHCK_K>,
                 <&rcc ETHTX>,
                 <&rcc ETHRX>,
                 <&rcc ETHSTP>;
        st,syscon = <&syscfg 0x4>;
        snps,mixed-burst;
        snps,pbl = <2>;
        snps,en-tx-lpi-clockgating;
        st,eth_ref_clk_sel;                /* In case of U-Boot */
       or
        st,eth-ref-clk-sel;                /* In case of Linux Kernel */
        snps,axi-config = <&stmmac_axi_config_0>;
        snps,tso;
        power-domains = <&pd_core>;
        status = "disabled";
};
```

```
   &ethernet0 {
       status = "okay";
       pinctrl-0 = <&ethernet0_rmii_pins_a>;
       pinctrl-1 = <&ethernet0_rmii_pins_sleep_a>;
       pinctrl-names = "default", "sleep";
       phy-mode = "rmii";
       max-speed = <100>;
       phy-handle = <&phy0>;
       mdio0 {
               #address-cells = <1>;
               #size-cells = <0>;
               compatible = "snps,dwmac-mdio";
               phy0: ethernet-phy@0 {
                       reg = <0>;
               };
       };
   };
```

+ update stm32mp15-pinctrl.dtsi to add ETHCK pin in ethernet0_rmii_pins_* node:
For example:

```
 <STM32_PINMUX('G', 8, AF2)>, /* ETH_RMII_ETHCK */
```

+ Need also to update TFA to generate 50Mhz clock (from PLL4P or PLL3Q):
for example if PLL4P in ed1 board: update fdts/stm32mp15xx-edx.dtsi

```
st,pkcs = <
 CLK_CKPER_HSE
 CLK_FMC_ACLK
 CLK_QSPI_ACLK
```

```
 - CLK_ETH_DISABLED
 + CLK_ETH_PLL4P
...
 /* VCO = 508.0 MHz => P = 50, Q = 60, R = 60 */
   pll4: st,pll@3 {
     compatible = "st,stm32mp1-pll";
     reg = <3>;
    cfg = < 1 49 11 9 9 PQR(1,1,1) >;
  };
```

3.3.4 RGMII with Crystal on PHY, CLK125 from PHY (Reference clock (standard RGMII clock name) is provided by a Phy Crystal)

```
    ethernet0: ethernet@5800a000 {
        compatible = "st,stm32mp1-dwmac", "snps,dwmac-4.20a";
        reg = <0x5800a000 0x2000>;
        reg-names = "stmmaceth";
        interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>,
                              <&intc GIC_SPI 62 IRQ_TYPE_LEVEL_HIGH>,
                              <&exti 70 1>;
        interrupt-names = "macirq",
                          "eth_wake_irq",
                          "stm32_pwr_wakeup";
        clock-names = "stmmaceth",
                      "mac-clk-tx",
                      "mac-clk-rx",
                      "ethstp";
        clocks = <&rcc ETHMAC>,
                 <&rcc ETHTX>,
                 <&rcc ETHRX>,
                 <&rcc ETHSTP>;
        st,syscon = <&syscfg 0x4>;
        snps,mixed-burst;
        snps,pbl = <2>;
        snps,en-tx-lpi-clockgating;
        snps,axi-config = <&stmmac_axi_config_0>;
        snps,tso;
        power-domains = <&pd_core>;
        status = "disabled";
    };
```

```
    &ethernet0 {
        status = "okay";
        pinctrl-0 = <&ethernet0_rgmii_pins_a>;
        pinctrl-1 = <&ethernet0_rgmii_pins_sleep_a>;
        pinctrl-names = "default", "sleep";
        phy-mode = "rgmii";
        max-speed = <1000>;
        phy-handle = <&phy0>;
        mdio0 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "snps,dwmac-mdio";
                phy0: ethernet-phy@0 {
                        reg = <0>;
                };
            };
    };
```

3.3.5 RGMII with 25MHz on ETH_CLK (no PHY Crystal), CLK125 from PHY (Reference clock (standard RGMII clock name) is provided by a RCC SoC internal clock)

```
    ethernet0: ethernet@5800a000 {
        compatible = "st,stm32mp1-dwmac", "snps,dwmac-4.20a";
        reg = <0x5800a000 0x2000>;
        reg-names = "stmmaceth";
        interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>,
                              <&intc GIC_SPI 62 IRQ_TYPE_LEVEL_HIGH>,
                              <&exti 70 1>;
        interrupt-names = "macirq",
                          "eth_wake_irq",
                          "stm32_pwr_wakeup";
        clock-names = "stmmaceth",
                      "eth-ck",
                      "mac-clk-tx",
                      "mac-clk-rx",
                      "ethstp";
        clocks = <&rcc ETHMAC>,
                 <&rcc ETHCK_K>,
                 <&rcc ETHTX>,
                 <&rcc ETHRX>,
                 <&rcc ETHSTP>;
        st,syscon = <&syscfg 0x4>;
        snps,mixed-burst;
        snps,pbl = <2>;
        snps,en-tx-lpi-clockgating;
        snps,axi-config = <&stmmac_axi_config_0>;
        snps,tso;
        power-domains = <&pd_core>;
        status = "disabled";
};
```

```
    &ethernet0 {
        status = "okay";
        pinctrl-0 = <&ethernet0_rgmii_pins_a>;
        pinctrl-1 = <&ethernet0_rgmii_pins_sleep_a>;
        pinctrl-names = "default", "sleep";
        phy-mode = "rgmii";
        max-speed = <1000>;
        phy-handle = <&phy0>;
        mdio0 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "snps,dwmac-mdio";
                phy0: ethernet-phy@0 {
                        reg = <0>;
                };
        };
    };
```

+ update stm32mp15-pinctrl.dtsi to add ETHCK pin in ethernet0_rgmii_pins_* node:
For example:

```
 <STM32_PINMUX('G', 8, AF2)>, /* ETH_RGMII_ETHCK */
```

+ Need also to update TFA to generate 25Mhz clock (from PLL4P or PLL3Q):
for example if PLL4P in ed1 board: update fdts/stm32mp15xx-edx.dtsi

```
st,pkcs = <
 CLK_CKPER_HSE
 CLK_FMC_ACLK
 CLK_QSPI_ACLK
 - CLK_ETH_DISABLED
 + CLK_ETH_PLL4P
...
 /* VCO = 600.0 MHz => P = 25, Q = 50, R = 50 */
   pll4: st,pll@3 {
     compatible = "st,stm32mp1-pll";
     reg = <3>;
     cfg = < 1 49 23 11 11 PQR(1,1,1) >;
 };
```

3.3.6 RGMII with Crystal on PHY, no 125Mhz from PHY

```
   ethernet0: ethernet@5800a000 {
       compatible = "st,stm32mp1-dwmac", "snps,dwmac-4.20a";
       reg = <0x5800a000 0x2000>;
       reg-names = "stmmaceth";
       interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>,
                             <&intc GIC_SPI 62 IRQ_TYPE_LEVEL_HIGH>,
                             <&exti 70 1>;
       interrupt-names = "macirq",
                         "eth_wake_irq",
                         "stm32_pwr_wakeup";
       clock-names = "stmmaceth",
                     "eth-ck",
                     "mac-clk-tx",
                     "mac-clk-rx",
                     "ethstp";
       clocks = <&rcc ETHMAC>,
                <&rcc ETHCK_K>,
                <&rcc ETHTX>,
                <&rcc ETHRX>,
                <&rcc ETHSTP>;
       st,syscon = <&syscfg 0x4>;
       snps,mixed-burst;
       snps,pbl = <2>;
       snps,en-tx-lpi-clockgating;
       st,eth_clk_sel;                /* In case of U-Boot */
      or
       st,eth-clk-sel;                /* In case of Linux Kernel */
       snps,axi-config = <&stmmac_axi_config_0>;
       snps,tso;
       power-domains = <&pd_core>;
       status = "disabled";
};
```

```
   &ethernet0 {
       status = "okay";
       pinctrl-0 = <&ethernet0_rgmii_pins_a>;
       pinctrl-1 = <&ethernet0_rgmii_pins_sleep_a>;
       pinctrl-names = "default", "sleep";
       phy-mode = "rgmii";
       max-speed = <1000>;
       phy-handle = <&phy0>;
       mdio0 {
               #address-cells = <1>;
               #size-cells = <0>;
```

```
                compatible = "snps,dwmac-mdio";
                phy0: ethernet-phy@0 {
                        reg = <0>;
                };
        };
    };
```

+ update stm32mp15-pinctrl.dtsi to delete CLK125 pin (also no need of ETHCK pin) in ethernet0_rgmii_pins_* node:

+ Need also to update TFA to generate 125Mhz clock (from PLL4P or PLL3Q):
for example if PLL4P in ed1 board: update fdts/stm32mp15xx-edx.dtsi

```
st,pkcs = <
 CLK_CKPER_HSE
 CLK_FMC_ACLK
 CLK_QSPI_ACLK
 - CLK_ETH_DISABLED
 + CLK_ETH_PLL4P
...
 /* VCO = 750.0 MHz => P = 125, Q = 62.5, R = 62.5 */
 pll4: st,pll@3 {
     compatible = "st,stm32mp1-pll";
     reg = <3>;
     cfg = < 3 124 5 11 11 PQR(1,1,1) >;
 };
```

3.4 DT configuration to reset Phy with GPIO

The example below shows how to configure GPIO to reset the Ethernet PHY.
This chapter is applicable for custom board where the Ethernet PHY reset signal is connected to a GPIO.
IMPORTANT, for STMicroelectronics boards it is not possible to use this configuration because there is no dedicated GPIO to reset PHY.

For kernel update see "reset-gpios" in Documentation/devicetree/bindings/net/ethernet-phy.yaml[5]

```
&ethernet0 {
     status = "okay";
     pinctrl-0 = <&ethernet0_rgmii_pins_a>;
     pinctrl-1 = <&ethernet0_rgmii_pins_sleep_a>;
     pinctrl-names = "default", "sleep";
     phy-mode = "rgmii";
     max-speed = <1000>;
     phy-handle = <&phy0>;
     mdio0 {
             #address-cells = <1>;
             #size-cells = <0>;
             compatible = "snps,dwmac-mdio";
             phy0: ethernet-phy@0 {
                     reg = <0>;
                     reset-gpios = <&gpioa 11 GPIO_ACTIVE_LOW>;
                     reset-assert-us = <1000>;
                     reset-deassert-us = <2000>;
             };
     };
};
```

For U-Boot same syntax of kernel, except that "reset-assert-us" and "reset-deassert-us" properties which are not managed (values of this properties are hardcoded in driver (udelay(2)), so you can change these value in function eqos_start_resets_stm32 of file dwc_eth_qos.c[6]

4 How to configure Ethernet using CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/net/stmmac.txt
- Documentation/devicetree/bindings/net/stm32-dwmac.txt
- arch/arm/boot/dts/stm32mp151.dtsi , STM32MP151 device tree file
- arch/arm/boot/dts/stm32mp15-pinctrl.dtsi , STM32MP15 pinctrl device tree file
- https://www.kernel.org/doc/Documentation/devicetree/bindings/net/ethernet-phy.yaml, More information
- https://github.com/u-boot/u-boot/blob/master/drivers/net/dwc_eth_qos.c, More information

Operating System

Device Tree

Ethernet

Generic Interrupt Controller

Serial Peripheral Interface

Reset and Clock Control

High Speed External oscillator (STM32 clock source)

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Stable: 19.10.2020 - 12:02 / Revision: 19.10.2020 - 12:01

A quality version of this page, approved on *19 October 2020*, was based off this revision.

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

Contents

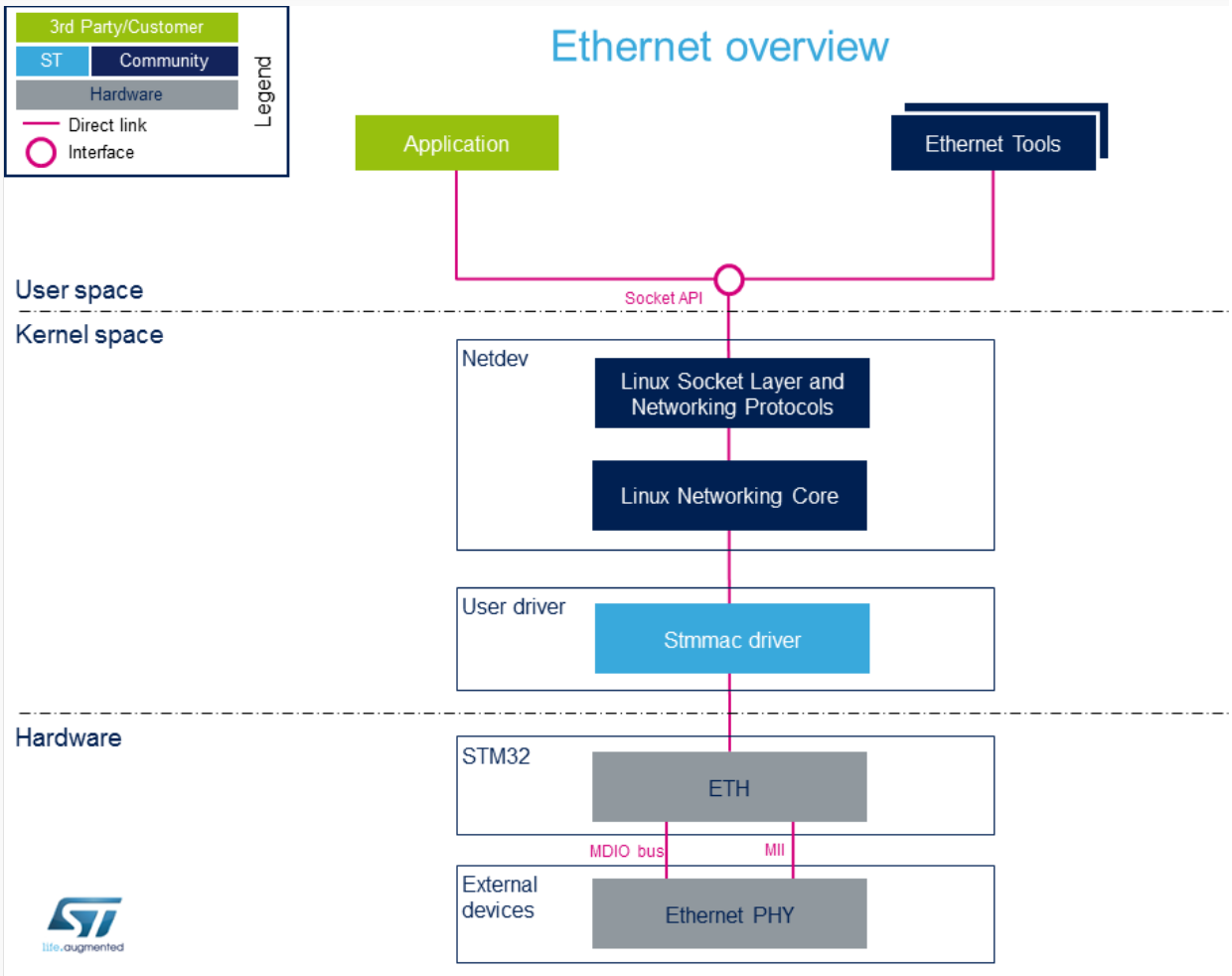
1 Framework purpose

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[6].

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
supported
[   12.481051] stm32-dwmac 5800a000.ethernet eth0: registered PTP clock
[   14.951370] stm32-dwmac 5800a000.ethernet eth0: Link is Up - 1Gbps/Full - flow control
rx/tx
```

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[7] 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
initialization failed
[   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
gating/mux configured with syscfg)
```

6 Source code location

The source files are located inside the Linux kernel.

- **Ethernet driver**: dwmac-stm32.c[8]

7 References

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

Ethernet

Application programming interface

Dynamic Host Configuration Protocol (See https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol for more details)

Device Tree

System File System (See https://en.wikipedia.org/wiki/Sysfs for more details)

Receive

Direct Memory Access

media access control address (https://en.wikipedia.org/wiki/MAC_address)

Transmit

Stable: 13.10.2020 - 14:08 / Revision: 13.10.2020 - 14:07

A quality version of this page, approved on *13 October 2020*, was based off this revision.

Contents

1 Purpose

This article describes how to configure the Ethernet interface.
This article provides two ways to make it:

- via ifconfig: to put quickly and temporary the Ethernet interface with static IP address.
- via systemd: to change durably the Ethernet configuration with static IP address

2 Using Ifconfig tools

- ***Connect an Ethernet cable and enjoy***

At startup SSH daemon (sshd) and ifplugd daemon are automatically launched:

- sshd (useful to perform ssh, scp)
- ifplugd detect:
  - udhcpc is launched on cable detection to retrieve an IP address,

If a DHCP server is not available, one can set the Ethernet IP adress with :

```
Board $> ifconfig eth0 uuu.xxx.yyy.zzz
```

Warning

If there is some service such as *systemd-networkd*, *NetworkManager*, *Connman* , the configuration of Ethernet interface can change when the service see the change

To check if eth0 and the gateway are well configured, one can type:

```
Board $>ifconfig
Board $>route
```

In the console, a log similar to the one below should be displayed:

```
Board $> ifconfig
eth0      Link encap:Ethernet  HWaddr 00:80:E1:01:39:61
          inet addr:10.48.1.172  Bcast:10.48.3.255  Mask:255.255.252.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3038 (2.9 KiB)  TX bytes:684 (684.0 B)
          Interrupt:103
STM32MP1 # route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         lme-gw-vl802.lm 0.0.0.0         UG    10     0        0 eth0
10.48.0.0       *               255.255.252.0   U     0      0        0 eth0
```

3 Using systemd-networkd service

Systemd-networkd is a network service provided by systemd.

3.1 Ethernet interface

Systemd-networkd provides a directory on which the Ethernet interface must be configured via configuration file: /lib/systemd /network.

```
 Board $> ls /lib/systemd/network
50-wired.network            52-static.network.static    80-container-vz.network
50-wired.network.static     80-container-host0.network  99-default.link
51-wireless.network.sample  80-container-ve.network
```

ST provide some configurations file:

- 50-wired.network: basic DHCP configuration for each **ethX** ethernet interface
- 50-wired.network.static: example of basic DHCP configuration on which eth0 are exclude of list
- 52-static.network.static: example of static IP configuration for eth0

3.2 Configuration file

Example of configuration:
DHCP configuration;

```
[Match]
Name=eth*

[Network]
DHCP=ipv4
```

Static IP configuration:

```
[Match]
Name=eth0

[Network]
DNS=192.168.72.254
Address=192.168.72.2/24
Gateway=192.168.72.254
```

[Match]

- **MACAddress=** a whitespace-separated list of hardware addresses
- **Name=** a white-space separated list of device names or expression (e.g. eth*).
- **Host=** the machine hostname

[NETWORK]

[Network]

- **DHCP=** enables the DHCP client, value can be: yes, no, ipv4, ipv6

- **DHCPServer=** enables the DHCP server for this configuration
- **DNS=** list DNS addresses in case of static configuration (you may specify several addresses)
- **IPForward=** configures IP packet forwarding, value can be: ipv4 or ipv6
- **IPMasquerade=** configures IP masquerading for the network interface. If enabled, packets forwarded from the network interface will be appear as coming from the local host. Takes a boolean argument. Implies IPForward=ipv4. Defaults to "no". Value can be yes or no.

All parameters are described on official documentation of systemd-networkd: [1]

3.3 DHCP configuration

DHCP configuration;

```
[Match]
Name=eth*

[Network]
DHCP=ipv4
```

All Ethernet interface are configured with DHCP client address.

3.4 STATIC configuration

Static configuration;
Add a new file 52-static.network with the following content
(/lib/systemd/network/50-wired.network or /lib/systemd/network/52-static.network)

```
[Match]
Name=eth0

[Network]
DNS=192.168.72.254
Address=192.168.72.2/24
Gateway=192.168.72.254
```

Eth0 Ethernet interface are configured with static IP address.

Information

In case of static configuration or specific configuration, please take care to not configure two time your Ethernet interface
, with the static configuration example you need to exclude eth0 interface of default configuration (here 50-wired.network)

Exclude eth0 from default Ethernet configuration:
Change the file 50-wired.network (or 80-wired.network) with the following content
(/lib/systemd/network/50-wired.network or /lib/systemd/network/80-wired.network)

```
[Match]
Name=eth[1-9]

[Network]
DHCP=ipv4
```

3.5 DHCP Server configuration

DHCP server example:

```
[Match]
Name=eth0

[Network]
Address=192.168.72.1/24
DHCPServer=yes
IPForward=ipv4
IPMasquerade=yes
```

4 References

- https://www.freedesktop.org/software/systemd/man/systemd.network.html

Dynamic Host Configuration Protocol (See https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol for more details)

uniprocessor

Receive

Transmit

Stable: 03.02.2020 - 08:06 / Revision: 03.02.2020 - 07:59

A quality version of this page, approved on *3 February 2020*, was based off this revision.

1 Purpose

This article describes how to configure the Ethernet interface using the PING tool.

1.1 Perform ping test

To test the Ethernet connection, it is useful to know how to perform a “PING” between the PC and the board. The PING command exchanges a short message between the two devices to verify that the Ethernet connection is working.

To test the communication between the PC to the board controller, one can execute the PING command from the PC:

For example, to PING a board at IP address 10.48.1.172, type:

```
PC $>ping 10.48.1.172
```

If the PING is successful, a message similar to the following is displayed:

```
PC $>Pinging 10.48.1.172 with 32 bytes of data:
PC $>Reply from 10.48.1.172: bytes=32 time=39ms TTL=15
PC $>Reply from 10.48.1.172: bytes=32 time=39ms TTL=15
PC $>Reply from 10.48.1.172: bytes=32 time=39ms TTL=15
```

2 References

Stable: 19.06.2020 - 10:06 / Revision: 19.06.2020 - 10:00

A quality version of this page, approved on *19 June 2020*, was based off this revision.

1 Purpose

This article describes how to perform remote connection using the **ssh**[1] tool.

1.1 Perform ssh connection

On host PC, one can type:

If it is a first connection:

```
PC $> ssh root@10.48.1.172
The authenticity of host '10.48.1.172 (10.48.1.172)' can't be established.
ECDSA key fingerprint is a0:a2:a3:09:b4:99:b3:90:6a:d0:35:05:6e:37:d0:6e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.48.1.172' (ECDSA) to the list of known hosts.
root@(none):~#
```

Otherwise:

```
PC $>ssh root@10.48.1.172
root@(none):~#
```

2 References

- [1] ifconfig

Elliptic Curve Digital Signature Algorithm