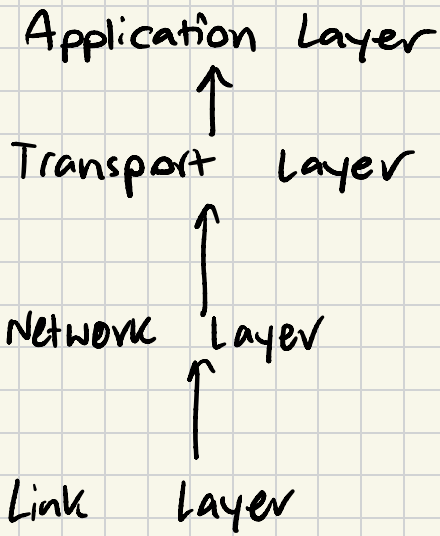
Commonly Accepted 4 Layer Model



Link Layer:

- Main task is to get packets across a single link
- Responsible for connecting the host to its local network and moving data across a single link
- Examples include
  - ↳ cellular connection, 3G, 4G, 5G
  - ↳ wifi
  - ↳ ethernet

- The link layer deals with the hardware and therefore has an address which is assigned during the manufacturing
  - ↳ This is a 48-bit serial number also known as the MAC address

  Eg.

  0f:2a:a2:1e:a1:19

Network Layer

- The main task of the Network Layer is to get packets across the Network, from source host to destination host

- This layer reads the header in each packet to see the destination of the packet and consults the routing tables in the router to see where to send the packet

- This layer uses services of the link layer to send the packets. It does not garuntee that the packets will arrive in the correct order or even arrive at all
  E.g. IP ⇒ internet protocol

Transport Layer

- The main task of the transport layer is to re-order packets to provide reliability on top the unreliable network Layer. This layer is also responsible for retransmission and handling of congestion (TCP only)

- Essentially the main task of the Network Layer is concerned with routing the right packets to the right destination host. The transport layer gets data to the right application running on the host.

- Two examples of Transport Layer protocols are TCP and UDP

  TCP: Transmission control protocol
  UDP: User Datagram protocol

Application Layer

- The Link Layer, network layer and Transportation Layer work together to reliably move data between two hosts across a network. Applications are then built to take advantage of this capability

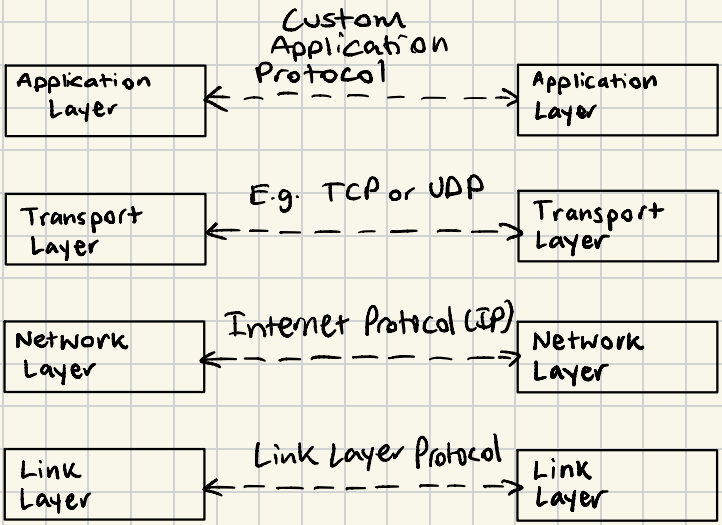
E.g.
- ↳ Skype
- ↳ WWW (World Wide Web)

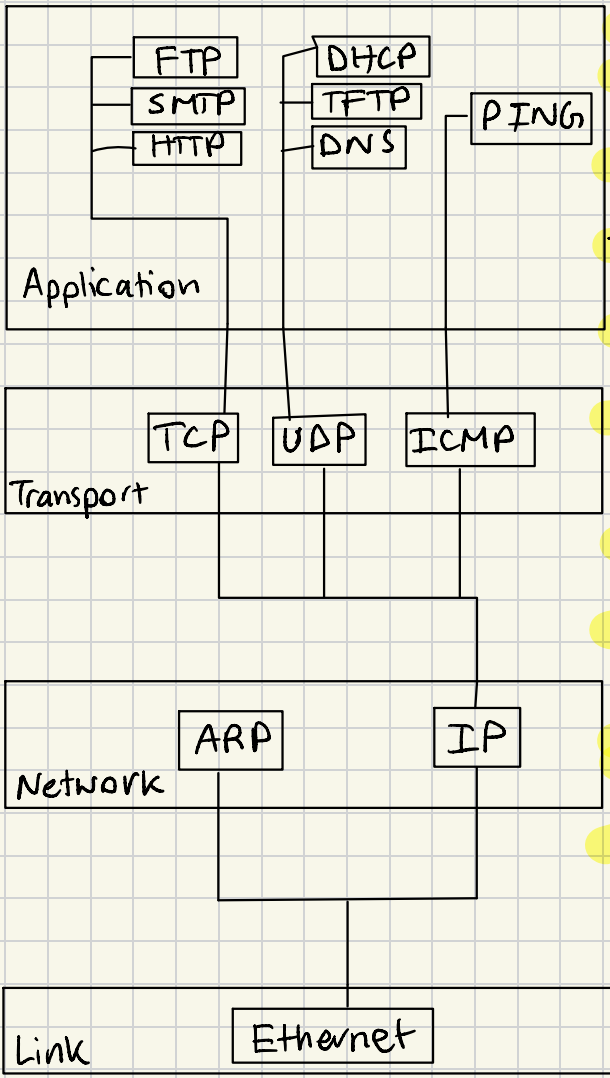
- Examples of the application layer protocols include: MQTT, HTTPS, DNS, FTP

- The Applications do not know and do not care how data gets from one host to another

LAYER INTERACTIONS

- Each layer talks to the corresponding layer on the other host using a protocol





FTP: File Transfer Protocol

SMTP: Simple Mail Transfer protocol

HTTP: Hypertext transfer protocol

TFTP: Trivial File transfer protocol

DNS: Domain Name System

PING: Utility for testing reachability of host

TCP: Transmission Control Protocol

UDP: User datagram Protocol

ARP: File Address Resolution Protocol

IP: Internet protocol

TCP/IP PROTOCOLS

Internet Protocol

- The protocol provides communication between hosts of different kinds of networks. For instance, hosts may have different link layer implementation, one may be on Wifi and another may be on Ethernet.
- It is connectionless and packet delivery is unreliable because there is no guarantee that the packets will be delivered

IP Address

- This is different from the MAC Address which is assigned during the manufacturing of the hardware.
- This is a unique 32-bit number for each host on a network.
- There are two versions: IPv4 and IPv6
- The IPv4 addresses consists of 4 numbers separated by dots, Eg. 192.168.000.50
- Each of the numbers can only be from 0 to 255, essentially 8-bit
- IPv6 addresses are longer, Eg. 2001.0ad8.62a1.0032.1000.7a1e.0250.5344

IPv4

- The IP address can be broken into two parts
- The first part is the Network Number and the second part is the Host Identifier

Eg. 192.168.000.50

↳ Network Number: 192.168
Host Identifier : 000.50

Netmask Address

- Netmask's are used to identify which part of the IP address is the network number and which part is the Host number
- This is done by a logical bitwise-AND of the IP address and the netmask

Eg. of a Netmask: 255.255.255.0

Default Gateway Address

- This is the private IP address assigned to the router

Eg. Some Routers use: 192.168.0.1

UDP (User Datagram Protocol)

- This is the minimum service over the IP Protocol
- This protocol is connectionless
- UDP is often used by applications that need multicast or broadcast delivery
- UDP packets may be delivered incorrectly or many may not be delivered at all. I.E. No guarantee of transmission
- UDP is optomized for sending packets to their destination as fast as possible. Eg. VoIP or streaming Movies

VoIP = Voice over Internet Protocol
- uses UDP is faster. VoIP is time-sensative and a slight loss of packets is generally preferable to the delay that would be incurred to ensure reliable packet delivery, which is what TCP offers.
- doesnt use TCP because TCP's error-checking and recovery features can introduce latency and jitter, which are detrimental to real-time voice communication

TCP → Transmission Control Protocol

- Provides Reliability
- Is a connection oriented transport protocol
- TCP is optimized for accurate delivery, not fast delivery
- TCP can detect errors or lost data and can trigger retransmission until the data is recieved, complete without errors

STM32H7 ETHERNET MAC

Features

- Fully compliant with the IEEE-802.3 standard
- Embeds its own dedicated DMA controller for automatic data flow control and high-speed transfers between the dedicated SRAM and the descriptors
- Supports LAN communications through an industry standard MII (Medium-independant interface) or a RMII (Reduced medium-independant interface) physical data transfers
- Supports Half-Duplex communication and Full-Duplex modes of communication
  - ↳ CSMA (Carrier Sense Multiple Access) and CD (Collision Detection) are both used in Half-Duplex communication

↳ CSMA/CD are used in Half-Duplex communication because in Half-Duplex, only 1 channel is used and data can only be either sent or recieved at a time but not simultaneously. CSMA/CD helps in managing access to the shared medium by detecting collisions and initiating retransmissions.

↳ CSMA/CD is generally not needed in full duplex communication. This is because in full duplex, two separate channels are used. Full duplex means data can be sent and received simultaneously on seperate logical channels, thus making the chance of collision virtually zero.

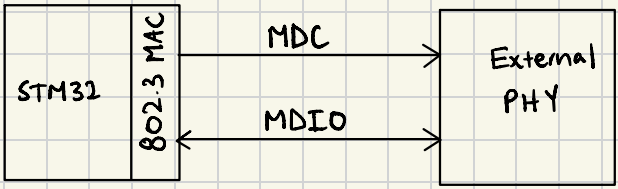
- Supports 10 and 100 Mbit/s rates
- Internal FIFO's to buffer transmit and receive frames, the transmit and receive FIFO are both 2Kb

- CSMA/CD → Carrier sense multiple access with collision detection
- MAC → Media Access control → This implements the data Link layer as defined by the IEEE-802.3 Ethernet standard
- MII → Media Independant Interface
- RMII → Reduced Media Independant Interface

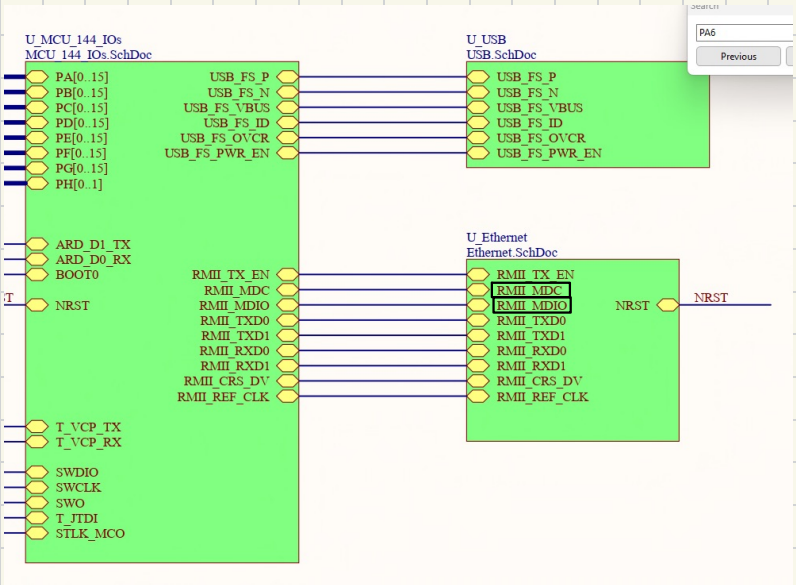
PHY Control Interface

- The external PHY is controlled by the stm32H7 ETH peripheral through the Station Management Agent (SMA) which uses the Station Management Interface (SMI) that allows read and write access to the PHY internal registers

- This interface supports the MDIO protocol on a pair of wires



- Schematics (NUCLEO-H743ZI2)



Question 1: Is the Station Management Agent (SMA) part of the Ethernet peripheral or its own peripheral?

The Station Management Agent (SMA) is generally considered a part of the Ethernet MAC (Media Access Control) peripheral. It is responsible for managing the PHY (Physical Layer) settings and status by communicating with the PHY's registers. It is not usually its own separate peripheral but is integrated into the Ethernet MAC.

Question 2: Does the SMA include a 2-wire Station Management Interface (MII Management Interface, MIM) that uses the MDIO protocol?

Yes, the SMA often includes a 2-wire Station Management Interface that uses the MDIO (Management Data Input/Output) protocol. This interface typically consists of two wires:

1. MDIO (Management Data Input/Output) - For data
2. MDC (Management Data Clock) - For clock signal

These wires are used to communicate with the PHY layer to read or write configuration registers.

Question 3: Are MDIO and MDC part of the RMII protocol?

RMII (Reduced Media-Independent Interface) is a standard for connecting the MAC to the PHY, and it does include MDIO and MDC lines for management. So, in this context, MDIO and MDC are part of the RMII interface, but they serve a different purpose than the data and clock lines used for packet data transfer. They are specifically for management tasks like configuration or status checking of the PHY layer.

Summary:

1. SMA: Integrated into the Ethernet MAC, not a separate peripheral.
2. 2-wire MIM: Uses MDIO and MDC wires for PHY management.
3. MDIO/MDC in RMII: They are part of the RMII interface but are used for management tasks, not data transfer.

I hope this clears up your confusion. Would you like to know more about any of these points?

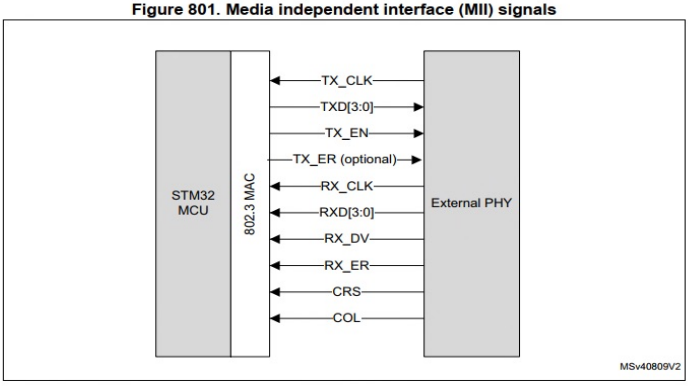
MII Interface

58.6.2 Media independent interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 801: Media independent interface (MII) signals*.

Figure 801. Media independent interface (MII) signals



- TX_CLK: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- TXD[3:0]: transmit data.

  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TX_ER (optional): required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- RXD[3:0]: receive data

  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.
- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.
- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.
- CRS: carrier sense.

  This signal is asserted by the PHY when either the transmit or receive medium is non idle. It is deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don't care for the MAC sublayer.
- COL: collision detection signal

  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode, the state of this signal is don't care for the MAC sublayer.

RMII Interface – less lines ⇒ saves GPIO pins

58.6.3 Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU. According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

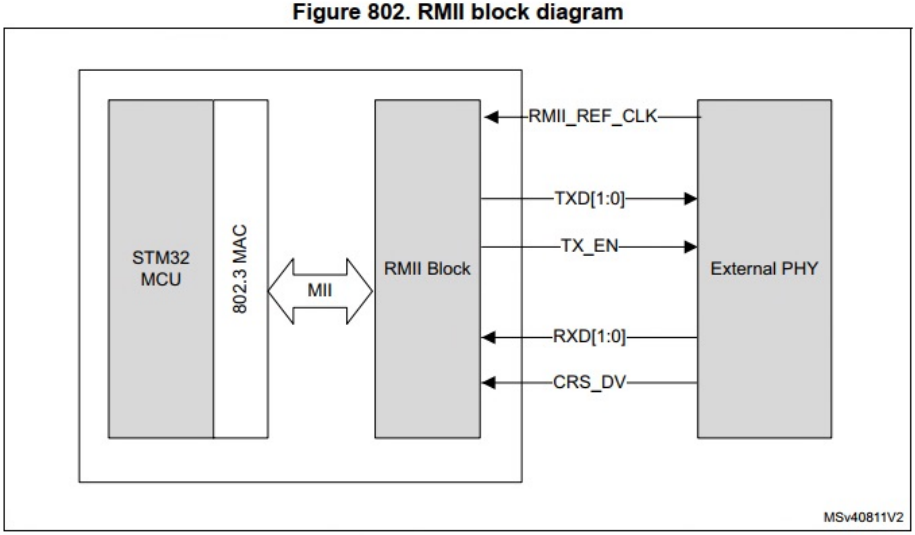
Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps operation.
- Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock references externally.

RMII block diagram

Figure 802: RMII block diagram shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

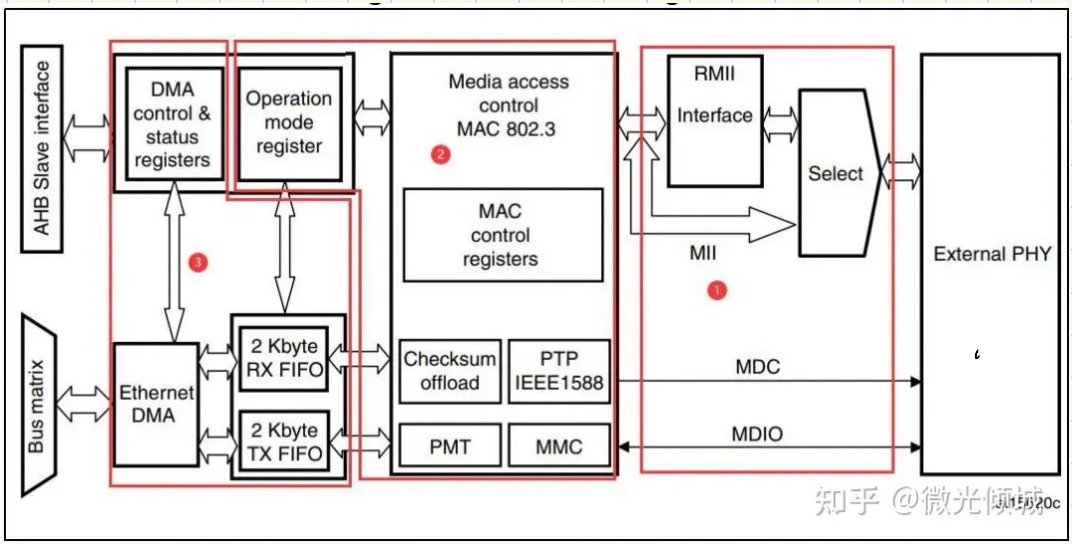
Figure 802. RMII block diagram



- RMII_REF_CLK: continuous 50 MHz reference clock input
- TXD[1:0]: transmit data
- TX_EN: transmit data enable.
  When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- RXD[1:0]: receive data
- CRS_DV: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.

ANALYZING THE ETHERNET BLOCK

Dedicated DMA



RMII

- We can choose to use MII or RMII. Mode can be selected using a single bit in the SYSCFG_PMC register

AHB

- The AHB Master Interface controls Data Transfers while the AHB Slave Interface accesses Control and Status Registers

- The AHB clock frequency must be atleast 25MHz when Ethernet is used.

FIFO

- The Transmit FIFO (Tx FIFO) buffers read data from system memory by the DMA before transmission by the MAC core

- The Receive FIFO (Rx FIFO) stores the ethernet frames received from the line until they are transferred to system memory by the DMA

SMA

- The Ethernet peripheral also includes a Station Management Agent to communicate with the external PHY.

- A set of configuration parameters permit the user to select the desired mode and features for the MAC and the DMA controller.

- The Station Management Agent allows the Application to access any PHY register through a two-wire clock and data line protocol

- The application can select one of the 32-registers within any PHY and send control data or recieve status information

- Both the MDC and MDIO data lines are implemented as alternate function I/O

- The MDC (Management Data Clock) is a periodic clock that provides the timing reference for the data transfer at the maximum frequency of 2.5 MHz

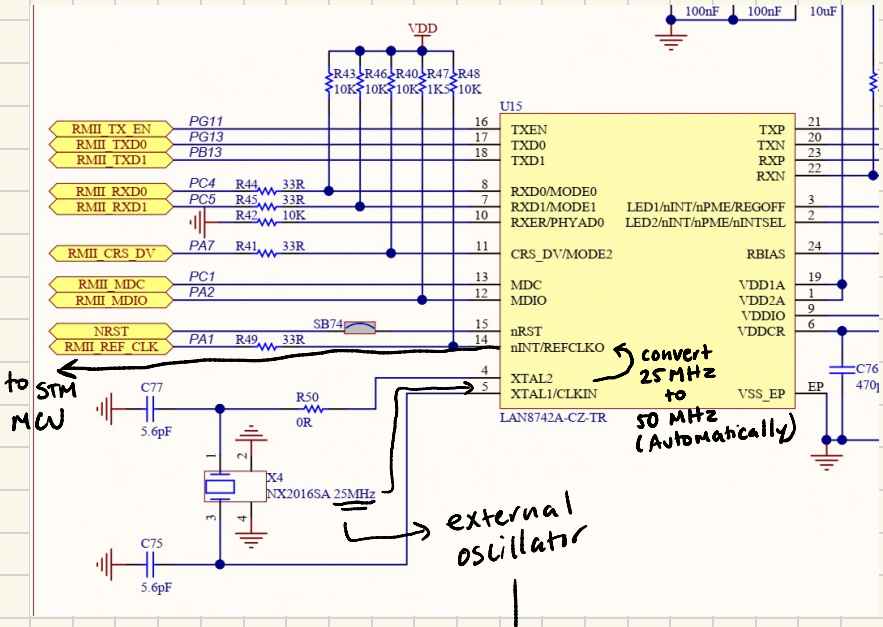
- The MDIO (Management Data Input/Output bitstream to transfer status information to/from the PHY device synchronously with the MDC clock signal

RMII Clock Source

- We can either clock the PHY from an external 50 MHz clock or use the PLL to generate the 50 MHz frequency.

↳ With respect to STM32H7432I2 NUCLEO BOARD

↳ schematics:

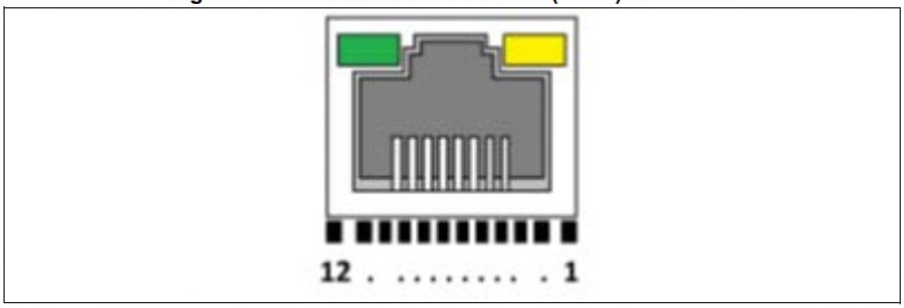


Ethernet RJ45 connector (CN14)

The STM32H7 Nucleo-144 board supports 10Mbps/100Mbps Ethernet communication with the PHY (U15) and integrated RJ45 connector (CN14). The Ethernet PHY is connected to the MCU via the RMII interface.

The X4 oscillator generates the 25 MHz clock for the PHY. The 50 MHz clock for the MCU (derived from the 25 MHz crystal oscillator) is provided by the RMII_REF_CLK of the PHY.

Figure 15. Ethernet RJ45 connector (CN14) front view



1. Green LED: Ethernet traffic
2. Amber LED: Ethernet connection

The related pinout for the Ethernet connector is listed in *Table 17*.

The clock flow can be as follows:

1) External oscillator
Oscillator (25MHz) → XTAL1 ⁉ XTAL2 pins of PHY → PHY converts 25MHz to 50 MHz → PHy sends clock signal to MCU using RMII_REF_CLK

2) PLL from STM32 MCU
MCU generates 50MHz signal via PLL → output that signal on the pin that is connected to the RMII ref clock pin of the PHY