TERMA

TEMU
Ethernet Bus Modelling

Mattias Holm

Version 1.0, 2019-12-12

Table of Contents

1. Introduction

TEMU provides support for Ethernet bus based devices. To support the development of custom MAC controllers, TEMU provides three generic models.

The `MDIOBus` model implements MDIO routing. As multiple MDIO devices can be connected to the same bus, a bus model is needed.

A `GenericPHY` model is implemented to expose the MDIO interface to the MAC models.

The `GenericPHY` model can be attached to the `EthernetLink` model. `EthernetLink` is responsible for routing EthernetFrames between registered nodes. It has two routing lists. Firstly, a list of *promiscuous* nodes that will receive all messages. Secondly, a routing map for non-promiscuous nodes.

When the `EthernetLink` model receives a frame, it forwards the frame to all the promiscous nodes. Then, it routes it to the destination MAC.

The `EthernetLink` assumes unique MACs, thus it will emit a warning in the case of a MAC address collission.

2. Connecting Devices

An ethernet link must be connected to its attached PHYs. Connection is done using the `ethernet-link-connect` command.

Connect Syntax

```
ethernet-connect link=ethlink0 phy=phy0:PHYIface
```

Disconnect Syntax

```
ethernet-disconnect link=ethlink0 phy=phy0:PHYIface
```

3. Checksums

Ethernet frames typically have a checksum that is generated and checked by hardware. To optimise the bus model, it is expected that MAC models supports opt in control on checksum generation and checking. This applies to all checksums, including Ethernet frame CRCs and IP header, TCP, UDP checksums. Since the Ethernet link is fully virtual, data cannot normally be corrupted in transit. Thus checksum checking and generation would be a waste of cycles.

There are still several usecases where one want to enable checksums:

- When viewing capture files with *Wireshark*, the tool will complain if ethernet CRCs are invalid.

- When receiving frames in a device which do not have hardware assisted CRC checking.

Thus, normally Ethernet CRC generation and checking will be disabled, while TCP/UDP/IP checksum generation (but not hardware checking) will be enabled.

4. MDIO Bus

The MDIO bus distributes MDIO control messages and supports routing of them. The MDIO bus use the same interface as an MDIO device. Thus, if only one MDIO device (e.g. GenericPHY) is available no MDIO bus instance is needed.

5. Generic PHY

The `GenericPHY` is a PHY / MII device which supports both the MDIO interface and the PHY interface for sending/receiving ethernet frames.

The `GenericPHY` device class by default enables support for BASE10, BASE100 and BASE1000 transfers. To only enable specific speed modes, the constructor accepts arguments:

- base10:1
- base100:1
- base1000:1

If any of these are set, the unset ones will be disabled.

Thus by default a PHY supports all BASE10, BASE100 and BASE1000 modes. By setting the base10 argument, only BASE10 modes will be supported. By setting base10 and base 100 arguments, only BASE10 and BASE100 will be supported.

At present it is not possible to control the support on a lower level.

6. Ethernet Link

6.1. Frame Capture

The ethernet link can be instructed to dump all traffic to a PCAPNG file.

Wireshark may flag frames as having invalid CRCs. To avoid this you can enable CRC generation in the MAC, or turn off checking in Wireshark.

To enable capture execute the enableCapture command ont he ethernet link.

Enable Capture Syntax

```
ethernet-enable-capture link=ethlink0 file="foo.pcap"
```

7. Implementing a MAC Model

TEMU comes with some bundled MAC models. In some cases it will be needed to implement additional project specific MAC models.

Consult the `eth-device` example for more info.

8. Auto Negotiation

The ethernet model supports autonegotiation for transfer speed capabilities.

The process is based on issuing an auto-negotiation request to the ethernet link model. The link will then issue autonegotiationg requests to each attached PHY, and finally call autonegotiateDone for all attached PHYs.

Each PHY will be called with the current known capabilities. It should return the same capabilities with potentially some of them cleared.

The actual final capabilities are reported with autonegotiateDone.

There, a PHY will select the highest priority common mode. Which by the standard is:

1. 40GBASE T FD
2. 25GBASE T FD
3. 10GBASE T FD
4. 5GBASE T FD
5. 2.5GBASE T FD
6. 1000BASE T FD
7. 1000BASE T HD
8. 100BASE T2 FD
9. 100BASE TX FD
10. 100BASE T2 HD
11. 100BASE T4
12. 100BASE TX HD
13. 10BASE T FD
14. 10BASE T HD

Note that TEMU does not support emulation of 2.5 GBASE and above at this moment.