

AN14188

Ethernet PHY Configuration For Win10 IoT Enterprise

Rev. 1 — 27 May 2024

Application note

Document information

Information	Content
Keywords	Ethernet PHY, ENET_QOS, ENET, Windows 10 IoT Enterprise BSP, Debug, Windows driver, ACPI table, U-boot, Windbg.
Abstract	The document describes the steps needed to configure and debug the Ethernet PHY on Windows 10 IoT BSP on i.MX SoC. It shows configuration in U-Boot, EFI, and in Windows driver on examples of two PHY chips Realtek RTL8211 and TI DP83867.



1 Introduction

This document describes the steps needed to configure and debug the Ethernet PHY on Windows 10 IoT BSP on i.MX SoC. Windows 10 IoT BSP supports two Ethernet IP blocks, ENET and ENET_QOS. Both can be found in i.MX 8M Plus and i.MX 93. ENET can be found in i.MX 8M, i.MX 8MM, i.MX 8MN, and i.MX 8QXP. Places of concern are the following:

- U-Boot
- EFI
- Windows driver

Configuration is shown on examples of two PHY chips.

- Realtek RTL8211
- TI DP83867

To understand this document, knowledge of these documents is required:

[i.MX Windows 10 IoT Quick Start Guide](#)

[i.MX Windows 10 IoT User's Guide](#)

[i.MX Windows 10 IoT Release Notes](#)

2 Configuration examples

Configuration of PHY is shown on the examples for two PHY.

2.1 U-Boot ENET and ENET_QOS PHY configuration

In U-Boot, support for several PHYs has been implemented.

According to the [Openwrt forum](#), U-Boot (just like Linux) selects the PHY driver by traversing a list of available drivers and using the driver that first matches a part of the PHY ID (for example, PHY Identifier Registers #1 and #2 in the datasheet). There is no principal difference in configuring the Ethernet PHY in the U-Boot for the ENET or ENET_QOS for this reason.

To compile U-Boot with support for specific PHY, its config feature is added to the U-Boot config file. Example 1 shows how a configuration for i.MX 8M Plus can look like. It is in file `uboot-imx/configs/imx8mp_evk_nt_uuu_defconfig`.

Example 1

```
CONFIG_PHY_REALTEK=y
CONFIG_PHY_ATHEROS=y
CONFIG_PHY_TI_DP83867=y
```

All available PHYs in U-Boot can be found in file `uboot-imx/drivers/net/phy/Makefile`:

Example 2

```
obj-$(CONFIG_BITBANGMII) += miiphybb.o
obj-$(CONFIG_B53_SWITCH) += b53.o
...
obj-$(CONFIG_PHY_ATHEROS) += atheros.o
...
obj-$(CONFIG_PHY_REALTEK) += realtek.o
...
```

```
obj-$(CONFIG_PHY_TI_DP83867) += dp83867.o
...
```

Configuration of PHY reset is in file `uboot-imx/arch/arm/dts/imx8mp-evk-u-boot.dtsi`.

Example 3

```
&ethphy0 {
    reset-gpios = <&gpio4 22 1>;
    reset-assert-us = <15000>;
    reset-deassert-us = <100000>;
};

&fec {
    phy-reset-gpios = <&gpio4 2 1>;
    phy-reset-duration = <15>;
    phy-reset-post-delay = <100>;
};
```

For more information on changing PHY in U-Boot, see [NXP Community](#).

2.2 Ethernet MAC (ENET) PHY configuration

This section provides details about Ethernet MAC (ENET) PHY configuration.

2.2.1 EFI ENET PHY Configuration

EFI configures pads and clocks for RGMII and the ACPI table provides information used by the Windows ENET driver.

2.2.1.1 EFI ENET PHY pads and pin routing

Initialization of pads and clocks for RGMII interface implements the function:

- VOID EnetInit(VOID)

The function is implemented in the file `iMX8BoardInit.c` for each platform, for example: `/mu_platform_nxp/NXP/MX8M_PLUS_EVK/Library/iMX8BoardLib/iMX8BoardInit.c`.

Example 4. Pins mux setting in `iMX8BoardInit.c`

```
VOID EnetInit(VOID)
{
    // ENET1/2 MDIO bus (both ENETs share one MDIO bus connected to the ENET1
    // controller)
    IOMUXC_SW_MUX_CTL_PAD_SAI1_RXD2 = IOMUXC_MUX_ALT4; // ENET1_MDC -> PAD_SAI1_RXD2
    IOMUXC_SW_MUX_CTL_PAD_SAI1_RXD3 = IOMUXC_MUX_ALT4; // ENET1_MDIO ->
    ENET1_MDIO_SELECT_INPUT
    ...
}
```

2.2.1.2 ENET ACPI table configuration

For the Enet driver type of PHY and register values are set in the file `Dsdt-Enet.asl`, for example: `mu_platform_nxp/NXP/MX93_11X11_EVK/AcpiTables/Dsdt-Enet.asl`.

Available commands for registry setting:

- MII_REG_WR – write

- MII_REG_RMW – read, modify, write

2.2.1.3 RTL8211 ACPI table setting

Example 5. Setting RTL8211 for Enet in ACPI table Dsdt-Enet.asl

```

Name (_DSD, Package () {
    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
    Package () { // RTL8211FDI-VD-CG
        Package (2) {"MDIOBusController_InputClk_kHz", 266000},
        Package (2) {"PhyAddress", 0x00},
        Package (2) {"PhyInterafceType", 0x00}, // RGMII, default
value
        Package (2) {"PhyMaxMDIOBusClock_kHz", 15000},
        Package (2) {"PhyMinSTAHoldTime_ns", 10},
        Package (2) {"PhyDisablePreamble", 0},
        Package (2) {"ConfigCmds", Package () {
            MII_REG_WR (0x1F, 0x0d08), // Select page
            MII_REG_RMW(0x11, 0x0000, 0x0100), // Enable Tx-delay
            MII_REG_RMW(0x15, 0x0000, 0x0008), // Enable Rx-delay
            MII_REG_WR (0x1F, 0x0d04), // Select page
            MII_REG_WR (0x10, 0x617F), // Set green LED for
Link, yellow LED for Active
            MII_REG_WR (0x1F, 0x0000), // Set default page
            ENET_MII_END}}
        }
    })

```

2.2.1.4 DP83867 table ACPI setting

Example 6. Setting DP83867 for Enet in ACPI table Dsdt-Enet.asl

```

Name (_DSD, Package () {
    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
    Package () { // RTL8211FDI-VD-CG
        Package (2) {"MDIOBusController_InputClk_kHz", 266000},
        Package (2) {"PhyAddress", 0x00},
        Package (2) {"PhyInterafceType", 0x00}, // RGMII, default
value
        Package (2) {"PhyMaxMDIOBusClock_kHz", 15000},
        Package (2) {"PhyMinSTAHoldTime_ns", 10},
        Package (2) {"PhyDisablePreamble", 0},
        Package (2) {"ConfigCmds", Package () {
            MII_REG_RMW(0x1F, 0x0000, 0x8000), // 3 Global Software
Reset 3 Global Software Reset 3 Global Software Reset
(CTRLCTRL)
            MII_REG_RMW(0x32, 0x0000, 0x0003), // Enable Shift mode for
both Rx/Tx (RGMIICTL)
            MII_REG_WR (0x86, 0x0077), // 2.0ns for Tx/Rx-delay
(RGMIIDCTL)
            MII_REG_RMW(0x1F, 0x0000, 0x4000), // 3 Global Software
Reset 3 Global Software Reset 3 Global Software Restart
            MII_REG_WR (0x18, 0x5032), // 1000BT, Link,
Receive, Transmit
            ENET_MII_END}}
        }
    })

```

```
})
```

2.2.2 Windows driver

The ENET Windows driver reads all PHY registry settings from the ACPI table as shown above, therefore it does not need to be changed when different PHY is used.

2.3 Ethernet quality of service (ENET_QOS) PHY configuration

This section provides details about the Ethernet QOS (ENET_QOS) PHY configuration.

2.3.1 EFI ENET_QOS PHY configuration

EFI configures pads and clocks for RGMII and the ACPI table provides information used by the Windows ENET_QOS driver but not PHY registry settings.

2.3.1.1 EFI ENET_QOS PHY pads and pin routing

The initialization of pads and clocks for the RGMII interface implements the function:

- VOID EnetQosInit()

The function is implemented in the file `iMX8BoardInit.c` for each platform, for example: `/mu_platform_nxp/NXP/MX8M_PLUS_EVK/Library/iMX8BoardLib/iMX8BoardInit.c`.

Example 7. Pins mux setting for ENET_QOS in `iMX8BoardInit.c`

```
VOID EnetQosInit()
{
...
/* Tx pads */
IOMUXC_SW_MUX_CTL_PAD_ENET_TD0 = IOMUXC_MUX_ALT0;
IOMUXC_SW_PAD_CTL_PAD_ENET_TD0 = IOMUXC_SW_PAD_CTL_PAD_FSEL_MASK |
IOMUXC_SW_PAD_CTL_PAD_DSE(0x03);
```

2.3.1.2 ENET_QOS ACPI table configuration

The registry setting for ENET_QOS PHY is hardcoded in its Windows driver and must be adapted there.

2.3.2 ENET_QOS Windows Driver

The PHY register setting is hardcoded in the Windows driver for the ENET_QOS Ethernet. For RTL 8211 the setting is in the function `MII_Rtl8211fInit`. Detection of connected PHY is done by the `MII_PhySpecificInit` function. If other PHY must be detected, vendor end model switches must be extended with new PHY identification and a new function, for example, `MII_DP83867fInit` must be implemented. Code examples can be found in [Section 5](#).

3 Common issues

This section provides a list of solutions for common issues that may arise while PHYs are debugged.

3.1 MAC address missing

When the MAC address is not written in fuses, it can be set in U-Boot, ACPI or via Windows registers for development purposes.

3.1.1 Windows registers MAC address setting

With the `ipconfig /all` command, the MAC address for the Ethernet interface can be checked.

If there is an invalid physical address, for example "00-00-00-00-00-00", set the address by either registry editor or command-line command.

3.1.1.1 Entering MAC address with Regedit

1. Open the register editor and find `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-08002BE10318}\xxxx`
2. Check the folders, for example, 0000, 0001... and find your wanted interface (DriverDesc = i.MX Ethernet adapter).
3. Add the new variable `NetworkAddress` as a string with the format `xx-xx-xx-xx-xx-xx`. It has to be a locally administered address (LAA). For details, see [MAC address](#).
4. Restart the board.

3.1.1.2 Entering MAC address with REG cmd

In the Command Prompt window enter:

```
REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Class\{4d36e972-e325-11ce-bfc1-08002be10318}\0000" /V NetworkAddress /T REG_SZ /D xx-xx-xx-xx-xx-xx /F
```

Find the correct folder 0000, 0001... the same way as in the previous case, you must identify the interface for which you want to set the MAC address.

There is a batch script for setting the MAC address via the registry available at <http://lallouslab.net/2016/06/20/batchography-change-mac-address-batch-script/>.

3.1.2 U-Boot MAC address setting

In U-Boot MAC address can be set either manually in the shell (the Windows Ethernet driver will not use the MAC address from U-Boot), or the usage of a random MAC address in case of a missing setting, it can be enabled in configuration.

3.1.2.1 MAC address setting manually by U-Boot variables

Put these commands in the U-Boot shell:

```
setenv ethaddr xx:xx:xx:xx -> for ENET
setenv ethladdr xx:xx:xx:xx -> for ENET_QOS
saveenv
```

3.1.2.2 Enabling random MAC address

Put `CONFIG_NET_RANDOM_ETHADDR=y` in the board `defconfig` file.

If neither `SROM` nor the environment contain a MAC address, an error is raised. If `CONFIG_NET_RANDOM_ETHADDR` is defined, a random, locally assigned MAC is used.

Further details can be found on [NXP Community](#).

3.1.3 ACPI MAC address setting

The Windows driver uses the `_DSM` method to obtain the MAC address from the ACPI table. The `_DSM` method uses defines `MC1X` and `MC2X` from the `Dsdt-Platform.asl` file describing where MAC bytes are stored in fuses:

```
OperationRegion(FUSE, SystemMemory, 0x30350400, 0x900) // 0x3035_0D00
Field(FUSE, AnyAcc, NoLock, Preserve)
{
  Offset(0x240),
  MC15, 8, // 0x640 NET1 MAC address bytes 5
  MC14, 8, // 0x641 NET1 MAC address bytes 4
  MC13, 8, // 0x642 NET1 MAC address bytes 3
  MC12, 8, // 0x643 NET1 MAC address bytes 2
  Offset(0x250),
  MC11, 8, // 0x650 NET1 MAC address bytes 1
  MC10, 8, // 0x651 NET1 MAC address bytes 0
  MC25, 8, // 0x652 NET2 MAC address bytes 5
  MC24, 8, // 0x653 NET2 MAC address bytes 4
  Offset(0x260),
  MC23, 8, // 0x660 NET2 MAC address bytes 3
  MC22, 8, // 0x661 NET2 MAC address bytes 2
  MC21, 8, // 0x662 NET2 MAC address bytes 1
  MC20, 8, // 0x663 NET2 MAC address bytes 0
}
```

Then `_DSM` method in `Dsdt-Enet.asl` can return those values when asked:

```
// Function 1: Return Mac Address
case (1) {
  Store (MC10, MAC0)
  Store (MC11, MAC1)
  Store (MC12, MAC2)
  Store (MC13, MAC3)
  Store (MC14, MAC4)
  Store (MC15, MAC5)
  Return (MAC)
}
```

`MC2X` values are used for a second Ethernet interface (see `Dsdt-Enet_QoS.asl`).

In case the MAC was stored in the fuses in the wrong order, it can be patched here.

3.2 Tx/Rx delay

For the ENET driver, the delay setting can be set in the ACPI table, for `ENET_QOS` it must be changed in the Windows driver code.

3.2.1 ENET TX delay setting example

Example of setting for i.MX 8M Nano in ACPI: `mu_platform_nxp/NXP/MX8M_NANO_EVK/AcpiTables/Dsdt-Enet.asl`

```
Name (_DSD, Package () {
  ToUUID ("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
```

```

Package () { // ATHEROS AR8031>
  Package (2) {"MDIOBusController_InputClk_kHz", 266000},
  Package (2) {"PhyAddress", 0x00},
  Package (2) {"PhyInterafceType", 0x00}, // RGMII, default value
  Package (2) {"PhyMaxMDIOBusClock_kHz", 15000},
  Package (2) {"PhyMinSTAHoldTime_ns", 10},
  Package (2) {"PhyDisablePreamble", 0},
  Package (2) {"ConfigCmds", Package () {
    // Enable GTX_CLK delay
    MII_WRITE_COMMAND(MII_REG_AR8031_DP_ADDR, 0x0005), // Choose SerDes Test
and System Mode Control
    MII_WRITE_COMMAND(MII_REG_AR8031_DP_RW, 0x0100), // Select 1 - RGMII Tx
Clock Delay Enable
    // Specific
    MII_WRITE_COMMAND(MII_REG_AR8031_SS, 0x000C), // Smart speed off
    ENET_MII_END}}
  }
})

```

3.2.2 Example of TX delay setting ENET_QOS in driver

Setting of TX/RX delay is located in function `MII_Rtl8211fInit`.

```

// Enable TX-delay for rgmii-id and rgmii-txid
Val = MII_Read(pAdapter, PhyAddr, 0x11);
if (pAdapter->MiiCfg.MiiInterfaceType == RGMII) {
  // RGMII config
  Val |= 0x0100;
} else {
  Val &= ~0x0100;
}
MII_Write(pAdapter, PhyAddr, 0x11, Val);
// Enable RX-delay for rgmii-id and rgmii-rxid
Val = MII_Read(pAdapter, PhyAddr, 0x15);
if (pAdapter->MiiCfg.MiiInterfaceType == RGMII) {
  // RGMII config
  Val |= 0x0008;
} else {
  Val &= ~0x0008;
}
MII_Write(pAdapter, PhyAddr, 0x15, Val);

```

4 Debugging

This section provides help on the PHY debugging on the target board.

4.1 How to start kernel debugging on target board

Serial debug must be used when the Ethernet is not yet working. To start kernel debugging over serial port:

1. Enable kernel debug on your target/development board by these commands in the elevated cmd window:

```

bcdedit /debug on
bcdedit /dbgsettings serial debugport:3 baudrate:921600

```

Use the port number appropriate for your board design. The baudrate must match the value of `CONFIG_BAUDRATE` in the U-Boot defconfig file.

2. Start WinDBG by typing this command in the elevated cmd window on the development PC:

```
"C:\Program Files (x86)\Windows Kits\10\Debuggers\x64\windbg.exe" -k
com:port=COM3,baud=921600
```

4.2 How to show the debug messages in WinDbg

To see debug messages from the Ethernet driver in the WinDbg window, they must be uncommented in the driver source code and the WinDbg debug print filter must be set up.

4.2.1 Enable debug messages In Windows driver

For enabling debug messages:

1. Open the iMXPlatform project.
2. Open the file `imxnetmini->header files->mp_dbg.h`. Enable/disable desired log output by un-comment defines, for example, `//#define DBG_MDIO_DEV`

Example 8. Suggested candidates for PHY debug

```
// ENET PHY device-specific macros - uncomment next line for message printing
//#define DBG_PHY_DEV
// MDIO bus-specific macros - uncomment next line for message printing
//#define DBG_MDIO_BUS
// MDIO device-specific macros - uncomment next line for message printing
//#define DBG_MDIO_DEV
// MDIO device command-specific macros - uncomment next line for message
printing
//#define DBG_MDIO_DEV_CMD
```

4.2.2 Enable debug messages in WinDbg

Debug messages can be enabled for the current debug session by putting the command in WinDbg:

```
ed nt!Kd_IHVDRIVER_Mask 0xFFFFFFFF
```

Or it can be set permanently in Windows registers by this command:

```
REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Debug Print
Filter" /v IHVDRIVER /t REG_DWORD /d 0xFFFFFFFF
```

5 Code examples

Windows driver function extended with TI DP83867 detection

```
NTSTATUS MII_PhySpecificInit(PMP_ADAPTER pAdapter)
{
    NTSTATUS Status = STATUS_SUCCESS;
    switch (pAdapter->ENETDev_PHYDevice.PhyVendor) {
        case REALTEK:
            switch (pAdapter->ENETDev_PHYDevice.PhyModel) {
                case RTL8211F:
                case RTL8211F_VD_CG:
                    DBG_PHY_DEV_PRINT_INFO("Detected Realtek RTL8211F");
                    MII_Rtl8211fInit(pAdapter);
                    break;
            }
    }
}
```

```

        default:
            DBG_PHY_DEV_PRINT_WARNING("Unknown Realtek PHY Model: 0x
            %02X", pAdapter->ENETDev_PhyDevice.PhyModel);
            break;
    }
    break;
    case TEXAS_INSTRUMENTS:
        switch (pAdapter->ENETDev_PhyDevice.PhyModel) {
            case DP83867:
                DBG_PHY_DEV_PRINT_INFO("Detected TI DP83867");
                MII_DP83867fInit(pAdapter);
                break;
            default:
                DBG_PHY_DEV_PRINT_WARNING("Unknown TI PHY Model: 0x%02X",
                pAdapter->ENETDev_PhyDevice.PhyModel);
                break;
        }
        break;
    default:
        DBG_PHY_DEV_PRINT_WARNING("Unknown PHY vendor: 0x%02X", pAdapter-
        >ENETDev_PhyDevice.PhyVendor);
        break;
    }
    return Status;
}

```

Windows driver init function for RTL8211

```

NTSTATUS MII_Rtl8211fInit(PMP_ADAPTER pAdapter)
{
    NTSTATUS Status = STATUS_SUCCESS;
    UINT16 Val;
    UINT8 PhyAddr = pAdapter->MiiCfg.PhyAddr;
    // Select Page 0x0d08*/
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0d08);
    // Enable TX-delay for rgmii-id and rgmii-txid
    Val = MII_Read(pAdapter, PhyAddr, 0x11);
    if (pAdapter->MiiCfg.MiiInterfaceType == RGMII) {
        // RGMII config
        Val |= 0x0100;
    } else {
        Val &= ~0x0100;
    }
    MII_Write(pAdapter, PhyAddr, 0x11, Val);
    // Enable RX-delay for rgmii-id and rgmii-rxid
    Val = MII_Read(pAdapter, PhyAddr, 0x15);
    if (pAdapter->MiiCfg.MiiInterfaceType == RGMII) {
        // RGMII config
        Val |= 0x0008;
    } else {
        Val &= ~0x0008;
    }
    MII_Write(pAdapter, PhyAddr, 0x15, Val);
    // Restore to default page 0
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0000);
    // Set green LED for Link, yellow LED for Active
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0D04);
    MII_Write(pAdapter, PhyAddr, 0x10, 0x617F);
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0000);
}

```

```

    return Status;
}

```

Windows driver init function for TI DP83867

```

NTSTATUS MII_DP83867fInit(PMP_ADAPTER pAdapter)
{
    NTSTATUS Status = STATUS_SUCCESS;
    UINT16 Val;
    UINT8 PhyAddr = pAdapter->MiiCfg.PhyAddr;
    // Select Page 0x0d08*/
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0d08);
    // Enable TX-delay for rgmii-id and rgmii-txid
    Val = MII_Read(pAdapter, PhyAddr, 0x11);
    if (pAdapter->MiiCfg.MiiInterfaceType == RGMII) {
        // RGMII config
        Val |= 0x0100;
    } else {
        Val &= ~0x0100;
    }
    MII_Write(pAdapter, PhyAddr, 0x11, Val);
    // Enable RX-delay for rgmii-id and rgmii-rxid
    Val = MII_Read(pAdapter, PhyAddr, 0x15);
    if (pAdapter->MiiCfg.MiiInterfaceType == RGMII) {
        // RGMII config
        Val |= 0x0008;
    } else {
        Val &= ~0x0008;
    }
    MII_Write(pAdapter, PhyAddr, 0x15, Val);
    // Restore to default page 0
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0000);
    // Set green LED for Link, yellow LED for Active
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0D04);
    MII_Write(pAdapter, PhyAddr, 0x10, 0x617F);
    MII_Write(pAdapter, PhyAddr, 0x1F, 0x0000);
    return Status;
}

```

6 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7 Revision history

Table 1. Revision history

Document ID	Release date	Description
AN14188 v.1.0	27 May 2024	Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <https://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μ Vision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Freescale — is a trademark of NXP B.V.

i.MX — is a trademark of NXP B.V.

Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

Contents

1	Introduction	2
2	Configuration examples	2
2.1	U-Boot ENET and ENET_QOS PHY configuration	2
2.2	Ethernet MAC (ENET) PHY configuration	3
2.2.1	EFI ENET PHY Configuration	3
2.2.1.1	EFI ENET PHY pads and pin routing	3
2.2.1.2	ENET ACPI table configuration	3
2.2.1.3	RTL8211 ACPI table setting	4
2.2.1.4	DP83867 table ACPI setting	4
2.2.2	Windows driver	5
2.3	Ethernet quality of service (ENET_QOS) PHY configuration	5
2.3.1	EFI ENET_QOS PHY configuration	5
2.3.1.1	EFI ENET_QOS PHY pads and pin routing	5
2.3.1.2	ENET_QOS ACPI table configuration	5
2.3.2	ENET_QOS Windows Driver	5
3	Common issues	5
3.1	MAC address missing	6
3.1.1	Windows registers MAC address setting	6
3.1.1.1	Entering MAC address with Regedit	6
3.1.1.2	Entering MAC address with REG cmd	6
3.1.2	U-Boot MAC address setting	6
3.1.2.1	MAC address setting manually by U-Boot variables	6
3.1.2.2	Enabling random MAC address	6
3.1.3	ACPI MAC address setting	7
3.2	Tx/Rx delay	7
3.2.1	ENET TX delay setting example	7
3.2.2	Example of TX delay setting ENET_QOS in driver	8
4	Debugging	8
4.1	How to start kernel debugging on target board	8
4.2	How to show the debug messages in WinDbg	9
4.2.1	Enable debug messages In Windows driver	9
4.2.2	Enable debug messages in WinDbg	9
5	Code examples	9
6	Note about the source code in the document	11
7	Revision history	12
	Legal information	13

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.