

AN13933

Porting PROFINET for i.MX 8M Mini Cortex-M

Rev. 0 — 28 April 2023

Application note

Document Information

Information	Content
Keywords	i.MX 8M Mini, PROFINET, Cortex-M
Abstract	This document describes how to run the RT-Labs PROFINET device stack on i.MX 8M Mini Cortex-M, using FreeRTOS and lwIP.



1 Introduction

This document describes how to run the [RT-Labs PROFINET device stack](#) on i.MX 8M Mini Cortex-M, using [FreeRTOS](#) and [lwIP](#).

FreeRTOS is a real-time operating system kernel optimized for embedded systems.

Lightweight TCP/IP (lwIP) is an open source TCP/IP stack. The main purpose of this stack implementation is to reduce resource usage, while still having a full-scale TCP. This makes lwIP suitable for use in embedded systems, which have limited resources.

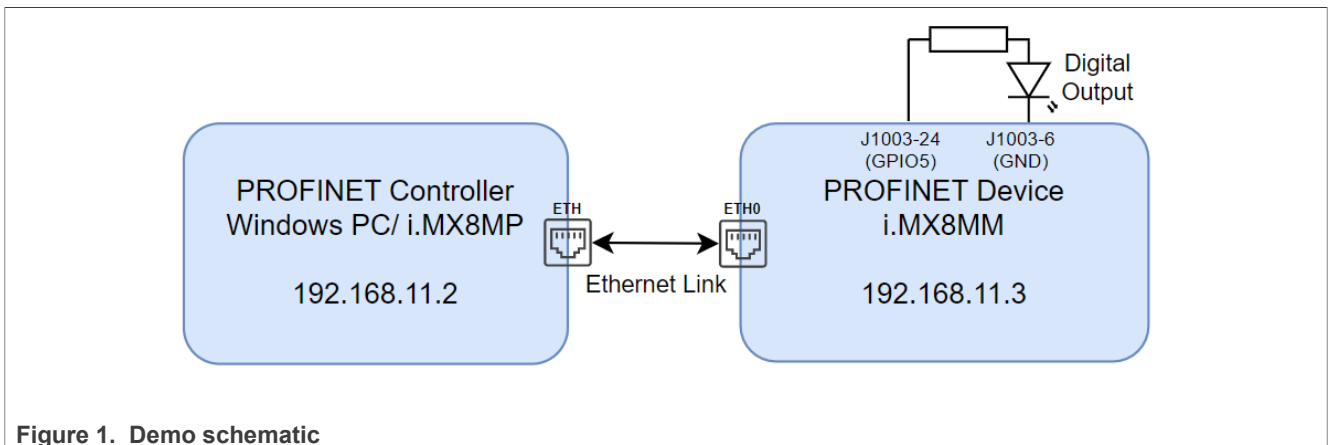
PROFINET is an industrial communication protocol that works over Ethernet. It can exchange data between controllers (typically PLCs) and devices, such as actuators and sensors. The PROFINET protocol uses simultaneously up to three channels of communication:

- Standard TCP/IP (NRT): used for non-time critical data; suitable for parameters and configurations (100 ms).
- Real Time (PROFINET RT): TCP/IP layers are bypassed to reduce the latency (1-10 ms); suitable for I/O applications, including motion control and high-performance requirements.
- Isochronous Real Time (PROFINET IRT): delivers high precision synchronization due to signal prioritization and scheduling (< 1 ms).

Note: The RT-Labs PROFINET stack does not support IRT.

This demo runs on a PROFINET Device (i.MX 8M Mini EVK). For testing, a PROFINET Controller is needed for which we are using a CODESYS Soft PLC. The CODESYS runtime can run either on Windows OS, or Linux OS. This document describes how to simulate a PLC on a Windows PC, or on an NXP i.MX 8M Plus EVK.

The PLC sends Ethernet packets to control the PROFINET Device through an Ethernet link.



To test the PROFINET Device stack functionality, the [RT-Labs sample application](#) is used, but with limited features. In this use case, the example uses only one LED connected to the IO-Device, which is controlled by the IO-Controller (PLC). The LED flashes at 1 second interval.

1.1 Hardware setup and equipment

A Windows PC is required for building the CODESYS runtime (either the controller is the Windows PC or the i.MX 8M Plus EVK).

- Development kit: **NXP i.MX 8MM EVK LPDDR4**, this is where the RT-Labs PROFINET Library runs on the Cortex-M4.
- **Option A:** One PC running Windows 10, which is the controller.

- **Option B:** Development kit: **NXP i.MX 8MP EVK LPDDR4**, which is the controller. The type of the board is not critical. Any board running Linux OS should work as long as it is supported by CODESYS (Optional).
- 2x Micro SD card: SanDisk Ultra 32 GB Micro SDHC I Class 10 is used for the current experiment (the second one is optional).
- 2x Micro-USB cable for debug port (the second one is optional).
- Ethernet cable (straight/crossover).
- 1x Managed switch (Optional), which is used for capturing the Ethernet packets between the i.MX 8M Plus and the i.MX 8M Mini for debugging.
- 1x red LED + 1x 470 ohm resistor.

1.2 Software environment

The PROFINET controller running CODESYS-implemented Soft PLC can either run on a Windows PC or on an i.MX 8M Plus EVK. The i.MX 8M Plus EVK software setup is described in the second section below.

1.2.1 Creating and flashing the i.MX 8M Mini client software image

A host PC running a recent version of Ubuntu, is assumed.

1. Install the [Real Time Edge Software 2.4.0](#) environment.
2. Build the **Real-time Edge Image** (using Yocto environment). For more details on how to do it, see [Section 5.5](#) from the [Real-Time Edge Yocto Project User's Guide](#).
3. Write the `nxp-image-real-time-edge-imx8mm-lpddr4-evk.wic.bz2` complete image (this can be found from the `<yocto_build_directory>/tmp/deploy/images/imx8mm-lpddr4-evk/` directory) on an SD card.

Note: Check your card reader partition and replace `sd<x>` with your corresponding partition.

```
bzcat nxp-image-real-time-edge-imx8mm-lpddr4-evk.wic.bz2 | sudo dd of=/dev/sd<x>
bs=1M status=progress
```

1.2.2 Retrieving and flashing the i.MX 8M Plus controller software image (optional, needed only if the CODESYS soft PLC is running on an i.MX 8M Plus EVK)

A host PC running Windows 10 is assumed. This is needed for the CODESYS environment.

1. [Real-Time Edge Image v2.4](#) can be directly downloaded from the NXP website.
2. Write the precompiled `nxp-image-real-time-edge-imx8mp-lpddr4-evk.wic` image to the SD card. [Win32DiskImager](#) can be used.

2 Building the PROFINET stack and the sample application for i.MX 8M Mini

2.1 Prerequisites for the host Ubuntu PC

1. Install CMake.

```
sudo apt-get install cmake
# Check the version >= 3.0.x
cmake --version
```

2. Install the GCC Arm embedded toolchain.

Note: It is recommended to install the 10.3 version, because currently the latest 12.2 version does not work properly.

```
mkdir ~/gcc_compiler
cd ~/gcc_compiler
wget -v https://developer.arm.com/-/media/Files/downloads/gnu-
rm/10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2
tar -xf gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2
```

Create a system environment variable and name it `ARMGCC_DIR`. The value of this variable should point to the Arm GCC Embedded toolchain installation path. For this example, the path is `~/gcc_compiler/gcc-arm-none-eabi-10.3-2021.10`. Add the following line to the `~/ .bashrc` file.

```
export ARMGCC_DIR=~/gcc_compiler/gcc-arm-none-eabi-10.3-2021.10
```

3. Download MCUXpresso SDK.

Download the MCUXpresso SDK, a package designed to simplify and accelerate application development with Arm Cortex-M-based devices.

Note: Both [Git](#) and [West](#) need to be installed to download the MCUXpresso SDK.

After the installation of Git and West, execute the following commands to achieve the whole SDK delivery at revision `MCUX_2.12.0` and place it in a folder named `mcuxsdk-2.12.0`.

```
west init -m https://github.com/NXPmicro/mcux-sdk --mr MCUX_2.12.0
  mcuxsdk-2.12.0
cd mcuxsdk-2.12.0
west update
```

2.2 Disabling the Ethernet driver from U-Boot and Linux kernel

To use the Ethernet on Cortex-M, Cortex-M must have exclusive access to the peripheral. Therefore, Ethernet access should be disabled from U-Boot and Linux kernel.

2.2.1 Disabling the Ethernet driver from U-Boot

1. Add at the end of the U-Boot device tree the following lines:

Location: `<yocto_build_directory>/tmp/work/imx8mm_lpddr4_evk-poky-linux/u-boot-imx/<specified_git_folder>/git/arch/arm/dts/imx8mm-evk.dts`

```
&fec1 {
    status = "disabled";
};
```

2. Recompile the U-Boot.

```
bitbake -f -c compile u-boot-imx
bitbake u-boot-imx imx-boot
```

3. Copy the new U-Boot image on the i.MX 8M Mini SD card.

```
dd if=imx-boot-imx8mm-lpddr4-evk-sd.bin-flash_evk of=/dev/sd<x> bs=1k seek=33
  conv=fsync
```

Note:

Storage location may vary. Adjust the `sd<x>` parameter to point to the SD card location.

2.2.2 Disabling the Ethernet driver from the Linux kernel

1. Add the following lines to the end of the kernel device tree:

Location: <yocto_build_directory>/tmp/work-shared/imx8mm-lpddr4-evk/kernel-source/arch/arm64/boot/dts/freescale/imx8mm-evk-rpmsg.dts

```
&fec1 {
    status = "disabled";
};
```

2. Recompile the kernel.

```
bitbake -f -c compile virtual/kernel
```

3. Copy the new device tree and kernel image to the SD boot partition (first (FAT) partition).

Note: The built image is located in: <yocto_build_directory>/tmp/work/imx8mm_lpddr4_evk-poky-linux/linux-imx/<git_version>/build/arch/arm64/boot/dts/freescale

```
sudo mount /dev/sd<x>1 /mnt
cp imx8mm-evk-rpmsg.dtb /mnt
cp Image /mnt
umount /mnt
```

Note: Storage location may vary. Adjust the mounted partition accordingly.

4. Check again the new image by booting the board. You should have no access to the Ethernet interface in Linux OS.

2.3 Importing the PROFINET stack and the sample application

For convenience, patches have been prepared. The patches for PROFINET can be found [here](#) and the patches for lwIP can be found [here](#).

1. Download the lwIP stack and place it in the ~/mcuxsdk-2.12.0/middleware directory:

```
cd ~/mcuxsdk-2.12.0/middleware
git clone https://github.com/lwip-tcpip/lwip.git
cd lwip
git checkout 239918ccc173cb2c2a62f41a40fd893f57faf1d6
```

Note: Checkout brings the exact version on which the patch was developed.

2. Download the imx8m_lwip_port.patch patch and apply it to lwIP, which fetches the lwIP port support for i.MX 8M.

```
cd lwip
wget https://raw.githubusercontent.com/nxp-imx-support/lwip_demo/master/imx8m_lwip_port.patch
git apply --whitespace=nowarn imx8m_lwip_port.patch
```

3. Download the RT-Labs PROFINET device stack and place it in the ~/mcuxsdk-2.12.0/middleware directory:

```
cd ~/mcuxsdk-2.12.0/middleware
git clone --recurse-submodules https://github.com/rtlabs-com/p-net.git
cd p-net
git checkout 05929f096122412a863a5fe09f9354e2f983e2dd
```

Note: Checkout brings the exact version on which the patch was developed.

Download the imx8m_pnet_port.patch patch and apply it to p-net, which fetches the PROFINET port support for i.MX 8M.

```
cd p-net
wget https://raw.githubusercontent.com/nxp-imx-support/profinet_demo/master/imx8m_pnet_port.patch
git apply --whitespace=nowarn imx8m_pnet_port.patch
```

- Download the `imx8mm_pnet_examples.patch` patch and apply it to the examples directory, which fetches the PROFINET sample application for i.MX 8M Mini.

```
cd ~/mcuxsdk-2.12.0/examples
wget https://raw.githubusercontent.com/nxp-imx-support/profinet_demo/master/imx8mm_pnet_examples.patch
git apply --whitespace=nowarn imx8mm_pnet_examples.patch
```

- The sample application can be found in: `~/mcuxsdk-2.12.0/examples/evkmimx8mm/profinet_examples`.

2.4 Application structure

2.4.1 Examples directory

The `profinet_examples/profinet_blink/freertos` directory contains the following files.

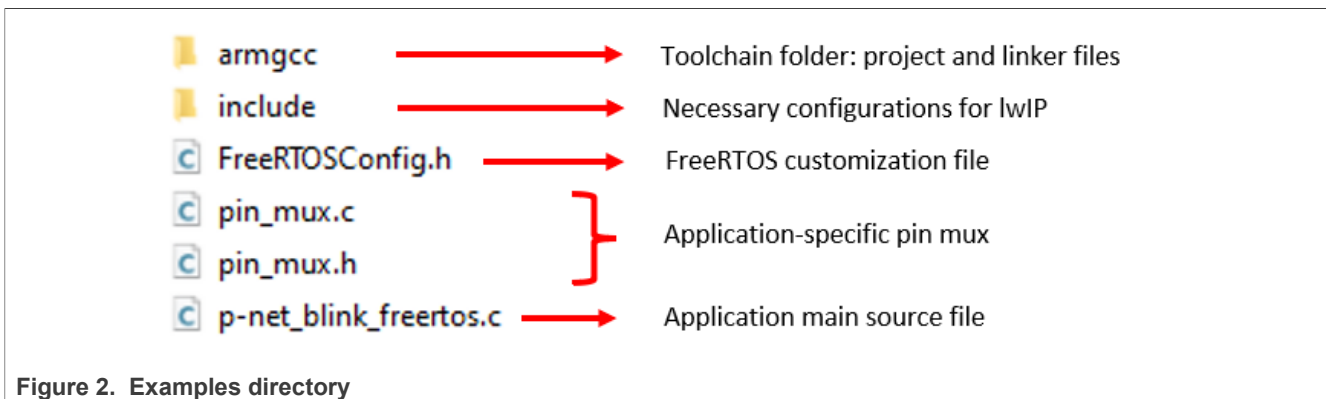


Figure 2. Examples directory

- The `include` directory contains the `lwipopts.h` file and the `lwip_hooks.h` file.
 - The `lwipopts.h` file is used to overwrite the default configuration of lwIP, located in `lwip/src/include/lwip/opt.h`, where the application-specific options are defined.
 - The `lwip_hooks.h` file defines the hook function used by the lwIP API to manage the PROFINET packets.
- The `armgcc` folder is the build directory that contains the project and linker file (`MIMX8MM6xxxxx_cm4_ddr_ram.ld`):
 - The `CMakeLists.txt` file is the file that the `cmake` uses to generate the `Makefile` automatically.

2.4.2 Middleware directory

The port support for PROFINET is directly included in the RT-Labs PROFINET repository:

- `src/ports/iMX8M/` includes the application layer files for network configuration.
- In the `cmake/` directory, the `iMX8M.cmake` file is used during image build.
- The `osal/` directory includes the files for the FreeRTOS abstraction layer.

2.5 Building the sample application and preparing the SD card

The example is written and run from DRAM.

- Change the directory to the example application project directory.

```
cd ~/mcuxsdk-2.12.0/examples/evkmimx8mm/profinet_examples/profinet_blink/freertos/armgcc
```

2. Run the build script on the command line to perform the build. The output is shown as follows:

```
./build_ddr_release.sh
-- TOOLCHAIN_DIR: /home/user/gcc_compiler/gcc-arm-none-eabi-10.3-2021.10
-- BUILD_TYPE: release
-- The ASM compiler identification is GNU
-- Found assembler: /home/user/gcc_compiler/gcc-arm-none-eabi-10.3-2021.10/
bin/arm-none-eabi-gcc

-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/mcuxsdk-2.12.0/examples/
evkmimx8mm/profinet_examples/profinet_blink/freertos/armgcc
Scanning dependencies of target pn_dev
[ 0%] Building C object CMakeFiles/pn_dev.dir/home/user/mcuxsdk-2.12.0/
middleware/p-net/samples/pn_dev/sampleapp_common.obj
[ 1%] Building C object CMakeFiles/pn_dev.dir/home/user/mcuxsdk-2.12.0/
middleware/p-net/samples/pn_dev/app_utils.obj
[ 1%] Building C object CMakeFiles/pn_dev.dir/home/user/mcuxsdk-2.12.0/
middleware/p-net/samples/pn_dev/app_log.obj
[ 1%] Building C object CMakeFiles/pn_dev.dir/home/user/mcuxsdk-2.12.0/
middleware/p-net/samples/pn_dev/app_data.obj
[ 2%] Building C object CMakeFiles/pn_dev.dir/home/user/mcuxsdk-2.12.0/
middleware/p-net/samples/pn_dev/app_gsdml.obj

< -- skipping lines -- >

[100%] Building C object CMakeFiles/p-net_blink_freertos.elf.dir/
home/user/mcuxsdk-2.12.0/core/components/serial_manager/
fsl_component_serial_port_uart.obj
[100%] Linking C executable ddr_release/p-net_blink_freertos.elf
Memory region      Used Size  Region Size  %age Used
  m_interrupts:      576 B      576 B      100.00%
    m_text:        170076 B    2096576 B      8.11%
    m_data:        1144264 B      2 MB      54.56%
    m_data2:         61824 B     12 MB      0.49%
[100%] Built target p-net_blink_freertos.elf
```

This script compiles the project and creates the `ddr_release` directory, which includes the `.bin` and the `.elf` files.

3. Copy the `p-net_blink_freertos.bin` file to the SD card on the first (FAT) partition (similar to Image copy in [Section 2.2.2](#)).

2.6 Setting up the U-Boot variables

Connect the i.MX 8M Mini platform to the host Ubuntu PC through the USB cable between the DEBUG USB-UART connector and the PC USB connector. The Ubuntu OS finds the serial devices automatically.

Find the TTY device with the name `/dev/ttyUSB*` to determine your debug port. One port is for the debug messages from the Cortex-A53, and the other is for the Cortex-M4. The port number is allocated randomly, so opening both is beneficial for development.

Open the serial device in your preferred serial terminal emulator (for example, PuTTY), set the speed to 115200 bps, 8 data bits, 1 stop bit (115200, 8N1), no parity.

1. Boot the board and stop the execution in U-Boot.
2. Create a U-Boot variable that stores the commands for loading and running the application.

```
u-boot=> setenv pnetddr "fatload mmc 1:1 0x80000000 p-net_blink_freertos.bin;
dcache flush; bootaux 0x80000000"
```

```
u-boot=> saveenv
```

3 Implementing the controller

The Soft PLC controller can be implemented either on a Windows-based PC, or on an i.MX 8M Plus EVK with a Linux distribution.

Download and install the CODESYS software on the Windows PC.

Download the Windows-based software ([CODESYS Development System V3](#)). For this demo, CODESYS Development System V3.5.19 is used.

3.1 Option A: Running the PLC controller on a Windows-based PC

The following configuration steps are necessary for standard-compliant behavior in Windows OS.

1. Install the [WinPCap](#) software.
2. Set a static IP address for the Ethernet interface.
Go to **Change Adapter Options**, select the Ethernet interface, double-click **Internet Protocol Version 4 (TCP/IPv4)** and set the static IP address to 192.168.11.2.

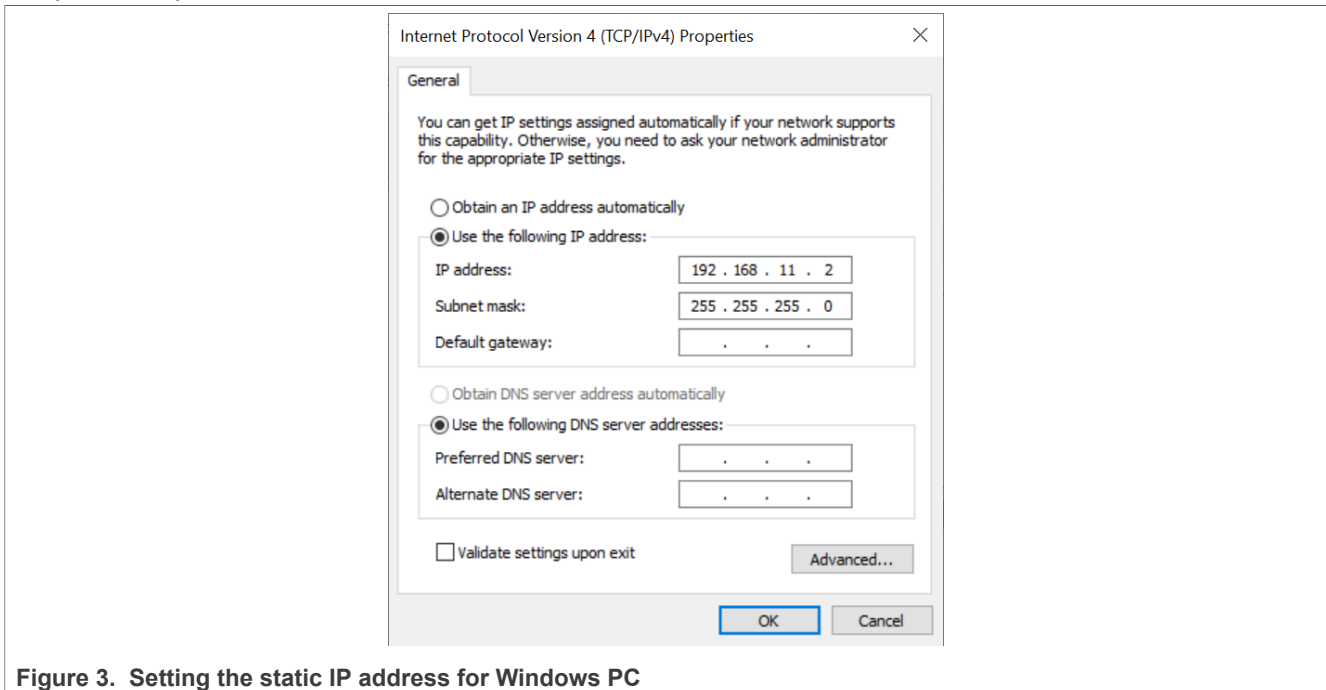






Figure 3. Setting the static IP address for Windows PC

3. Disable the firewall.
Because the Ethernet interface is public, the firewall blocks the Ethernet packets to be sent to the PROFINET device. Disable the firewall, or configure the network to private.
4. Install the CODESYS runtime.
In CODESYS, install the **CODESYS Control Win 64** (3.5.18.20 version is used) or **CODESYS Control Win** (depending on your architecture) by using the menu **Tools - CODESYS Installer**. After a successful installation, CODESYS Control Win V3 runs as a service on your computer.

Note:

*If there are errors regarding missing libraries, choose **PLC logic** -> **Application** -> **Library Manager** on the left menu. CODESYS should automatically detect if there are any missing libraries. Click **Download missing libraries** under the **Library Manager** tab to download any missing libraries.*

5. Start the CODESYS Control Win V3.

In the system bar, right-click the  or  symbol and click **Start PLC** on the menu. After a successful start, the CODESYS Control Win V3 symbol in the system bar changes to  or .

6. Create a project in CODESYS.

From the menu, create a project by selecting **Standard Project**. From the **Device** drop list, select **CODESYS Control Win (64)** and select to program in **Structured Text (ST)**.

7. Establish a connection to CODESYS Control.

Double-click **Device (<variant>)**, and from the **Communication Settings** tab, select **Scan Network**.

Double-click on the scanned device to establish the connection. Echo messages can be sent to check the connection.

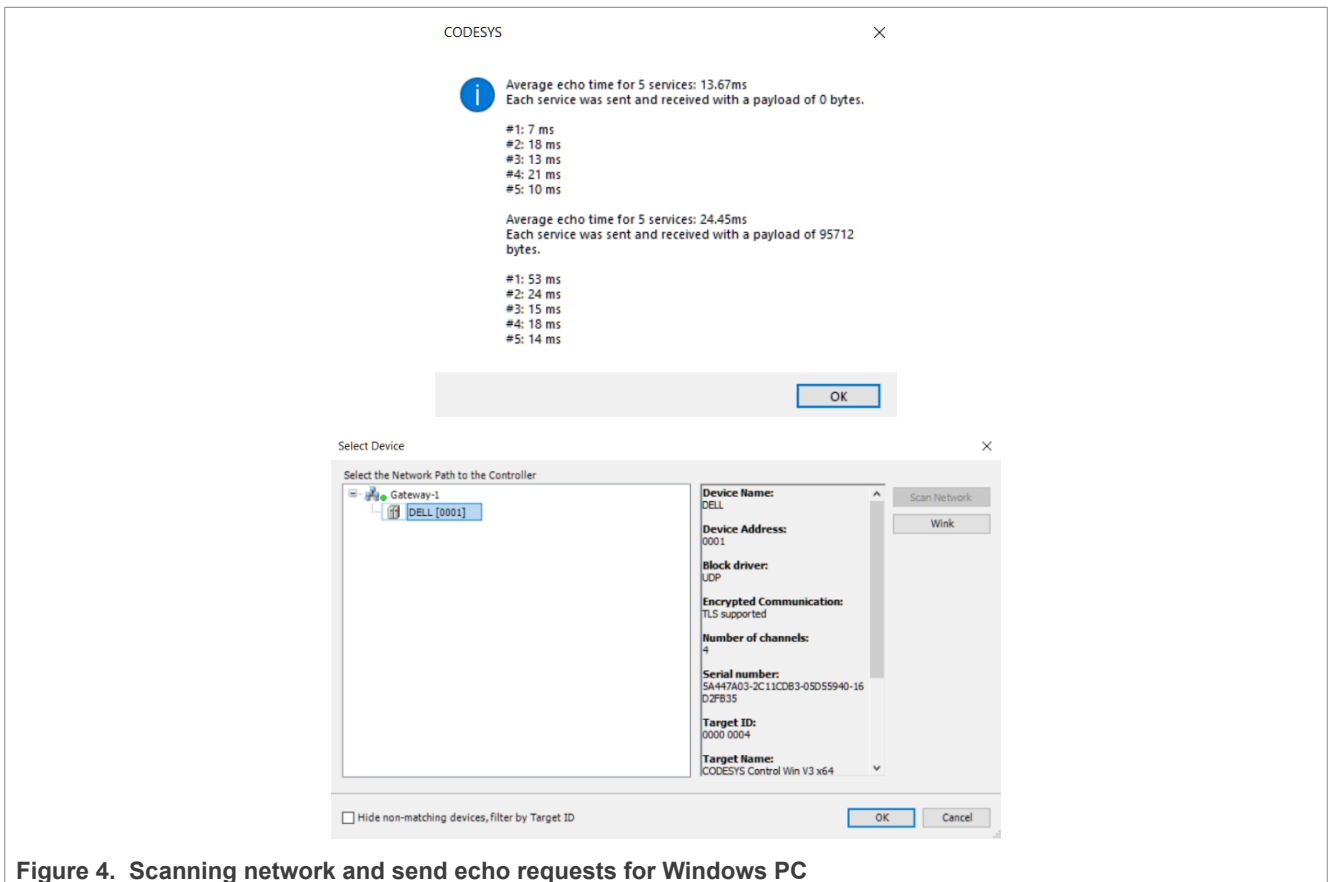


Figure 4. Scanning network and send echo requests for Windows PC

Note: If the device is not found, on the **Communication** tab, change the Filter from **Target system ID** to **None**.

CODESYS tries to authenticate itself on this target system and get the communication channel.

8. Implement the CODESYS application according to [Section 3.3](#).

3.2 Option B: Run the PLC controller on i.MX 8M Plus

Connect the i.MX 8M Plus board to the Windows PC through an Ethernet cable. Use the ENET0 (ETH1) port of the board. Set the static IP address of the host Windows PC to: 192.168.11.4/24.

Connect the i.MX 8M Plus board to the Windows PC through a USB cable between the DEBUG USB-UART connector and the PC USB connector.

Note: You may install the **USB-to-Serial driver (FTDI or CP)** corresponding to the chip you have on the EVK.

Open the Device Manager, find the USB serial port in Ports (COM). Assume that the ports are COM2, COM3, COM4, and COM5. Three of them are for debug messages from the Cortex-A53, and the other one is for the Cortex-M4. The port number is allocated randomly, so opening all ports is beneficial for development.

Open the serial device in your preferred serial terminal emulator (for example, PuTTY), set the speed to 115200 bps, 8 data bits, 1 stop bit (115200, 8N1), and no parity, and then power on the board. Wait for the board to boot and log in with root.

On the i.MX 8M Plus:

1. Set the static IP address by creating the `20-static.network` configuration file in the `/etc/systemd/network` directory.

```
[Match]
Name=eth1
KernelCommandLine=!nfsroot
[Network]
DNS=8.8.8.8
Address=192.168.11.2/24
```

Note: `eth1` is the software interface of the ENET0 hardware port.

Reboot the board and check if the ETH1 has the assigned IP address (192.168.11.2).

```
root@imx8mp-lpddr4-evk:~# ip addr
```

2. Add the symbolic link to `ifconfig`, `route`, `ldconfig`, and `start-stop-daemon`.

```
root@imx8mp-lpddr4-evk:~# ln -s /bin/ifconfig /sbin/ifconfig
root@imx8mp-lpddr4-evk:~# ln -s /bin/route /sbin/route
root@imx8mp-lpddr4-evk:~# ln -s /usr/sbin/start-stop-daemon /usr/bin/start-stop-daemon
root@imx8mp-lpddr4-evk:~# ln -s /sbin/ldconfig /bin/ldconfig
```

3. Set the correct system date and time.

Example:

```
root@imx8mp-lpddr4-evk:~# date -s "18 MARCH 2023 18:58:00"
```

On the Windows host PC:

1. Install the CODESYS runtime.

In CODESYS, use the menu **Tools - CODESYS Installer** to install the **CODESYS Code Generator ARM64** (version 4.0.0.0 is used) and **CODESYS Control for Linux ARM64 SL** (version 4.7.0.0 is used).

Note: If there are errors regarding missing libraries, select **PLC logic** → **Application** → **Library Manager** from the menu on the left. CODESYS should automatically detect if there are any missing libraries. Click **Download missing libraries** under the **Library Manager** tab to download any missing libraries.

When it is completed, there should be a new entry **Update Linux ARM64** available in the **Tools** menu.
2. Scan the network to find the i.MX 8M Plus board and install the CODESYS runtime onto it.

From the menu on the top, select **Tools - Update Linux ARM64**, and then click **Scan** to find the IP address of the i.MX 8M Plus. In this use case, it is 192.168.11.2. Click **Install** for the CODESYS runtime package.
3. Create a new project in CODESYS.

From the menu, create a project by selecting **Standard Project**. From the **Device** drop list, select **CODESYS Control for Linux ARM64 SL** and select to program in **Structured Text (ST)**.

To connect to i.MX 8M Plus, double-click the **Device (<variant>)** entry from the menu on the left. Then click the **Scan Network** tab from the **Communication Settings** tab, and then select the **imx8mpevk**. Use the **Device** and **Send Echo Service** tabs to verify the communication.

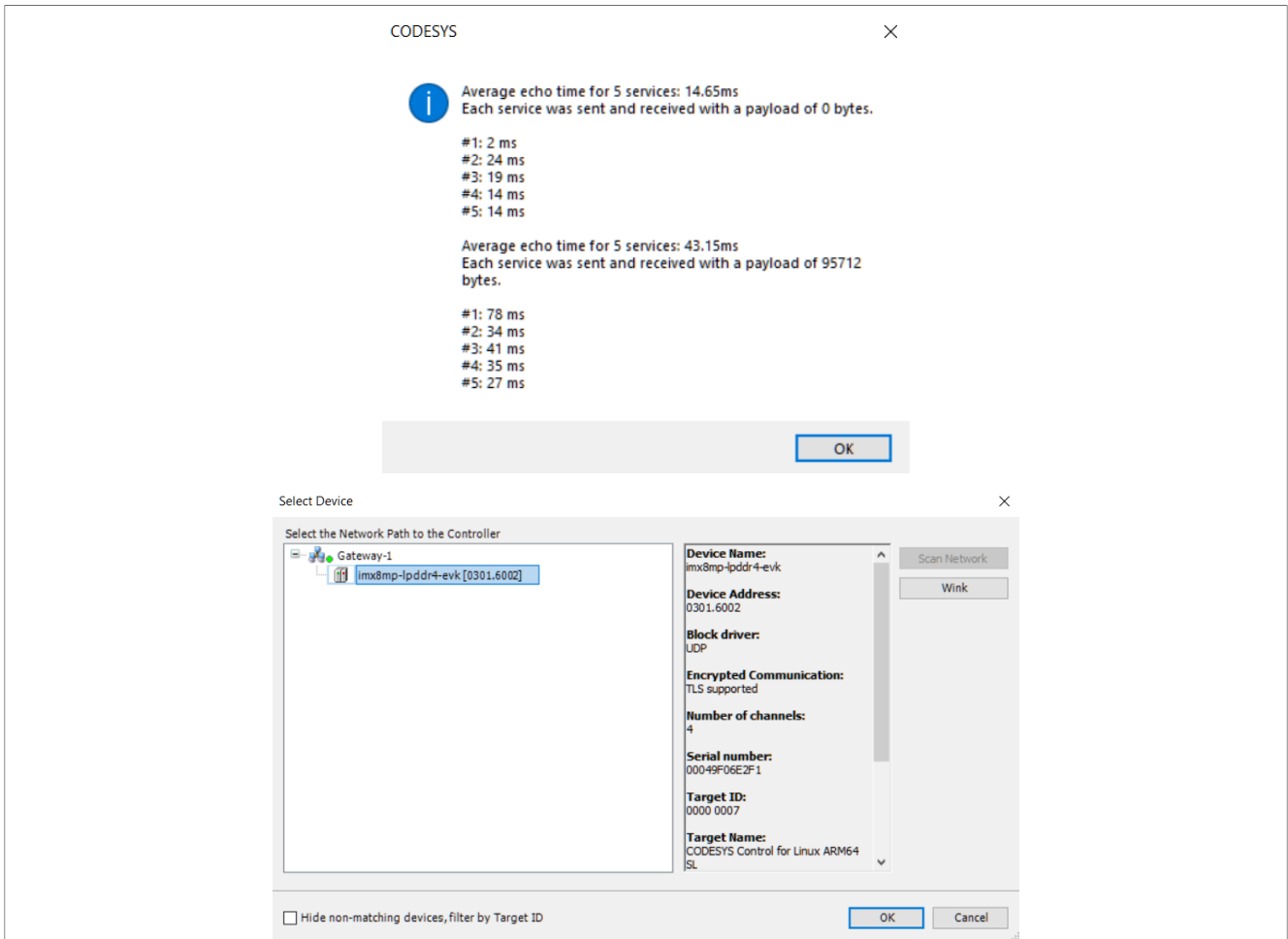


Figure 5. Scanning network and send echo requests for i.MX 8M Plus

Note:

- If the device is not detected, try to restart the CODESYS Gateway.
 - If the board is connected to the Windows PC after CODESYS is started, the board will not be detected. Restart the CODESYS software.
4. Implement the CODESYS application according to [Section 3.3](#).
After the CODESYS Soft PLC starts, the Ethernet cable between the i.MX 8M Plus and the PC can be disconnected.
 5. Start and enable the CODESYS services on i.MX 8M Plus.
When the board is rebooted, the application does not start automatically. Connect to the serial console and start the services on the i.MX 8M Plus board.

```
root@imx8mp-lpddr4-evk:~# systemctl start codesysedge.service
root@imx8mp-lpddr4-evk:~# systemctl start codesyscontrol.service
```

3.3 Implementing the CODESYS application

To implement the CODESYS application, perform the following steps:

1. In the CODESYS menu **Tools**, select **Device Repository**, click **Install**, and then select the [GSDML](#) file.

Add the following devices:

- Right-click **Device (<variant>)** from the left panel and select **Add Device - Fieldbuses -Ethernet Adapter - Ethernet**.
 - Right-click **Ethernet** and select **Add Device - Profinet IO - Profinet IO Master - PN-Controller**.
 - Right-click **PN_Controller** and select **Add Device - Profinet IO Slave - I/O - P-Net Sample - P-Net multi-module sample app**.
 - Right-click **P_Net_multi_module_sample_app** and select **Add Device - Profinet IO Module - DIO 8xLogicLevel**.
2. Double-click the **Ethernet** node in the left menu and select the network interface with the 192.168.11.2 IP address. The IP address is automatically updated.

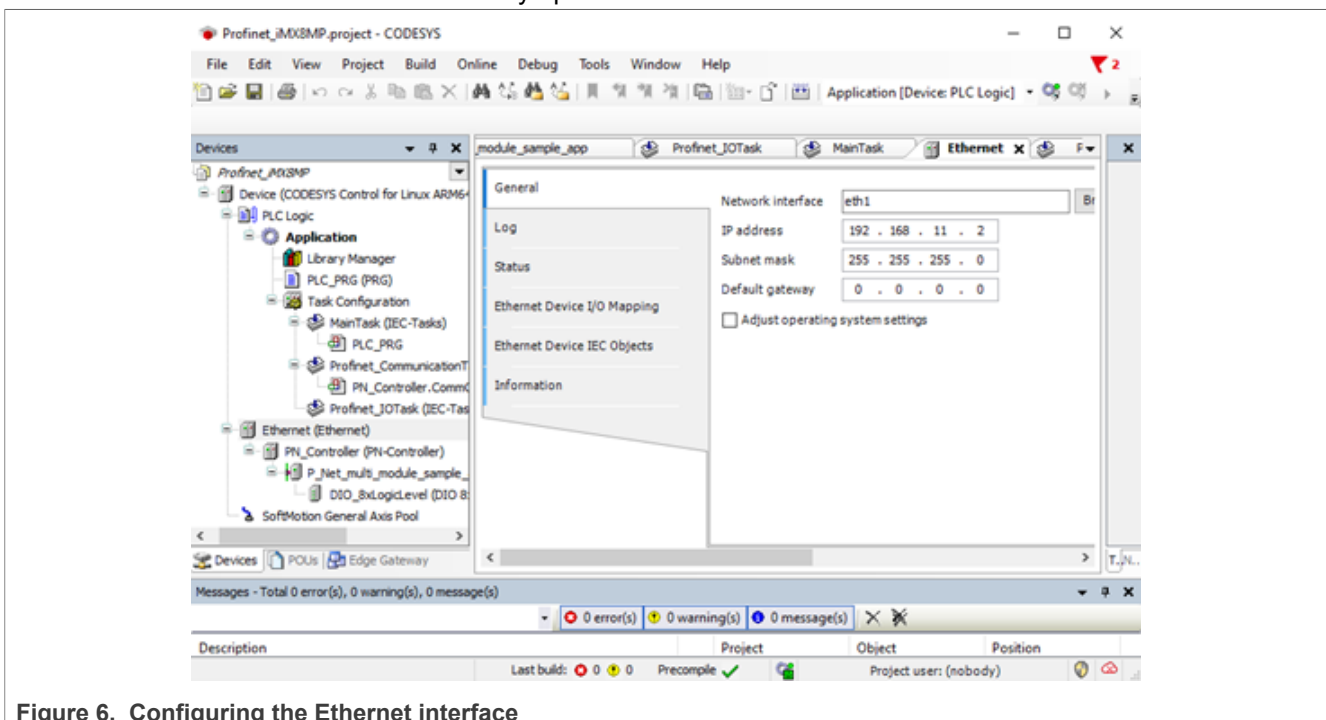


Figure 6. Configuring the Ethernet interface

3. Double-click the **PN_controller** node in the left menu and set the **First IP** and **Last IP** to both the existing IP address of the IO-device. In this use case, it is 192.168.11.3.

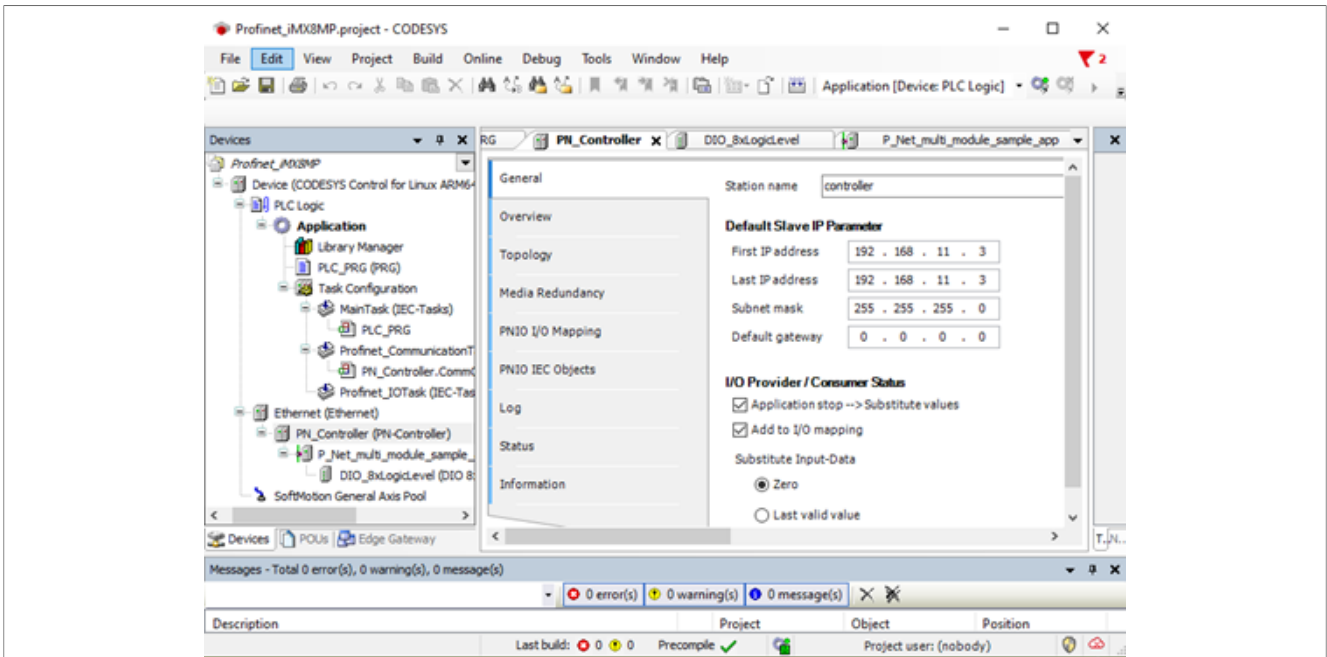


Figure 7. Configuring the PROFINET Controller

4. Double-click the **P_Net_multi_module_sample_app** node in the left menu and set the IP address to the existing address of the IO-device. In this use case, it is 192.168.11.3.

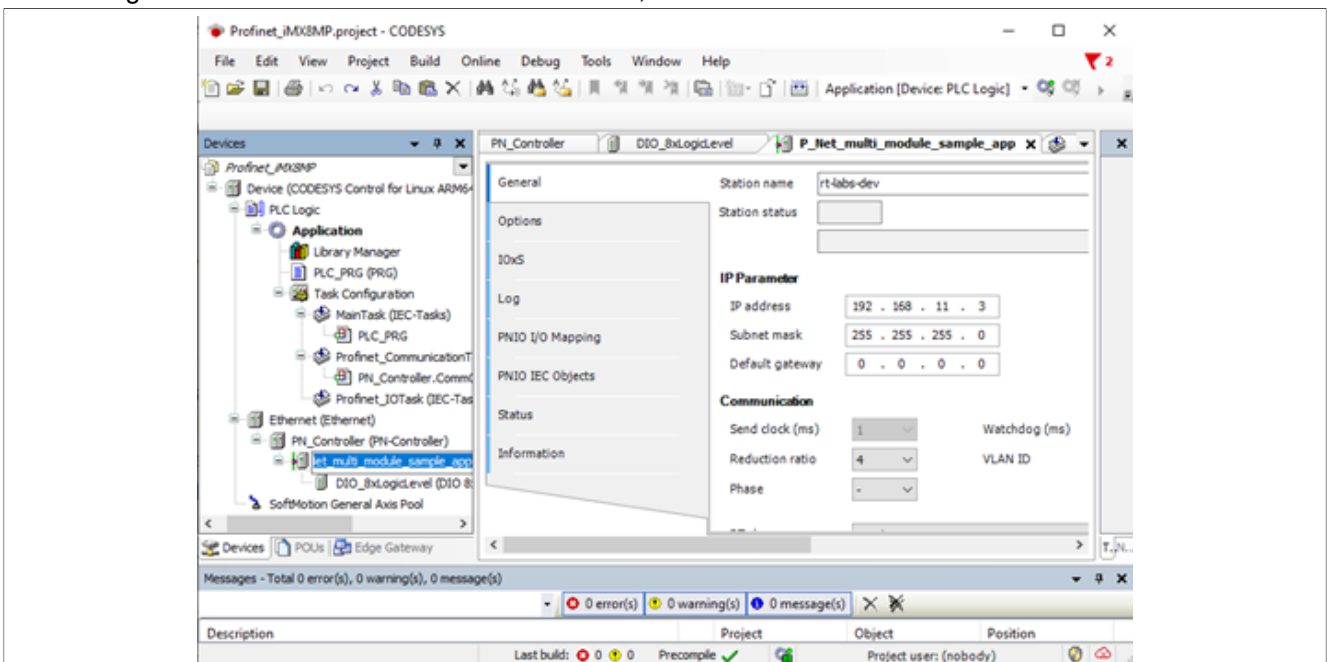


Figure 8. Setting the device IP address

5. Create the controller application by entering the following program code in **Application - MainTask - PLC_PRG**.

Variable section:

```
PROGRAM PLC_PRG
VAR
  pin_state: BOOL := FALSE;
  counter: UINT := 0;
```

```
END_VAR
```

Program section:

```
counter := counter + 1;
IF counter > 250 THEN
    counter := 0;
    pin_state := NOT pin_state;
END_IF
```

6. Right-click the **DIO_8xLogicLevel** node and select **Edit IO mapping**. Open **Output 8 bits**, double-click **Output Bit 7**, and map it to **pin_state**.
In **Application**, double-click the **MainTask** node and select **Cyclic** with 4 ms.
In **Application**, double-click the **Profinet_CommunicationTask** node and select **Cyclic** with 10 ms and **Priority 14**.
7. Transfer the controller application to the device (PC/i.MX 8M Plus) and start it.
 - Select **Build - Generate Code** from the menu on the top.
 - Transfer the application by using the **Online - Login** menu on the top, and then press **Yes** from the menu on the top.
 - Click **Debug - Start** from the menu on the top.

Note: On trial version of CODESYS, the software has to be restarted every two hours.

4 Running the application

To run the application, perform the following steps:

1. On i.MX 8M Mini, connect an LED to GPIO5_IO[13].
Connect a red LED in series with a 470 ohm resistor between the 24th pin of the J1003 expansion connector and GND (the 5th pin of the J1003 connector). The 24th pin of the J1003 connector is used with GPIO function (GPIO5_IO[13]).



Figure 9. LED connected to GPIO5_IO13

2. Connect the Windows PC or i.MX 8M Plus (ETH1) to i.MX 8M Mini (ETH0) through an Ethernet cable.

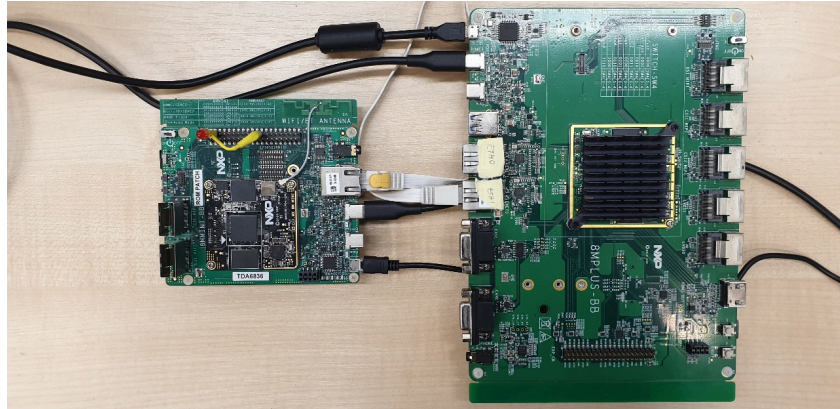


Figure 10. Hardware setup using i.MX8MP as controller

- 3. Start the Soft PLC on the Windows PC or on i.MX 8M Plus.
 - a. Boot the i.MX 8M Mini board and stop the boot in U-Boot. Run the application:

```
u-boot=> run pnetddr
```

Note: If the Linux OS kernel runs together with Cortex-M4, make sure the correct dtb file is used. This dtb file reserves resources used by Cortex-M4 and avoids the Linux kernel from configuring them. Use the following command before running the kernel:

```
u-boot=> setenv fdtfile imx8mm-evk-rpmsg.dtb
```

- b. Test the application:
The following figure shows the output on the debug serial port of the Cortex-M.

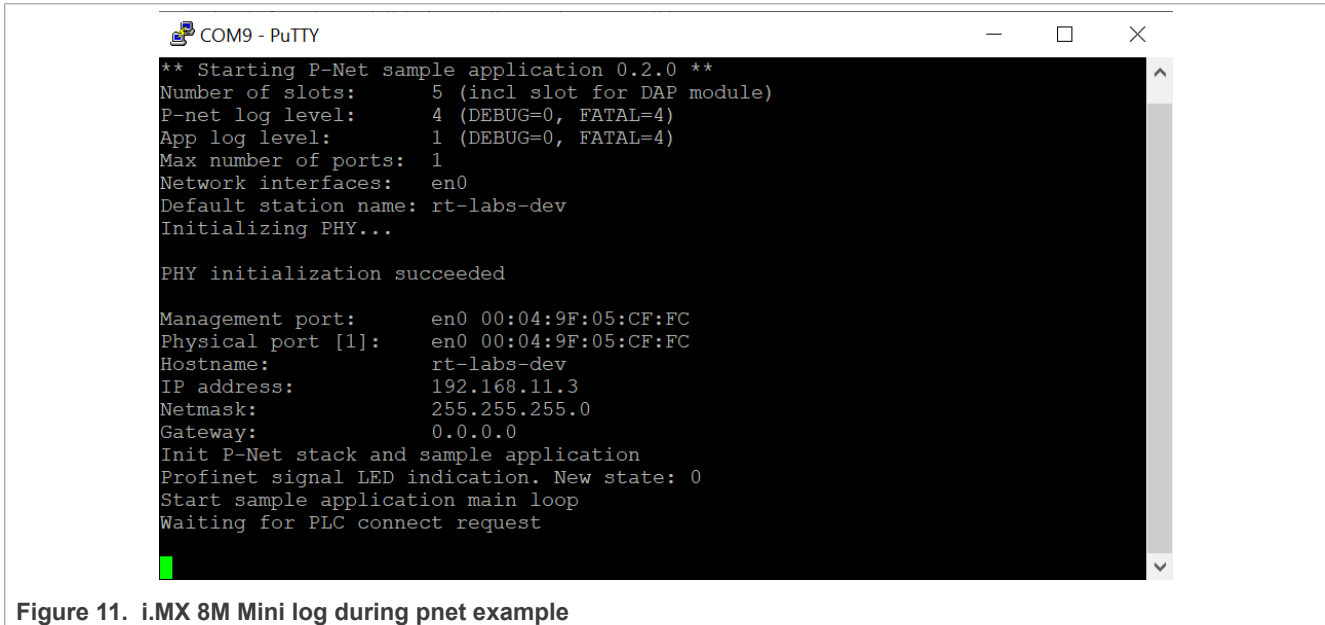


Figure 11. i.MX 8M Mini log during pnet example

On Windows PC, the Soft PLC runs successfully when all the symbols on the left of the devices are green.

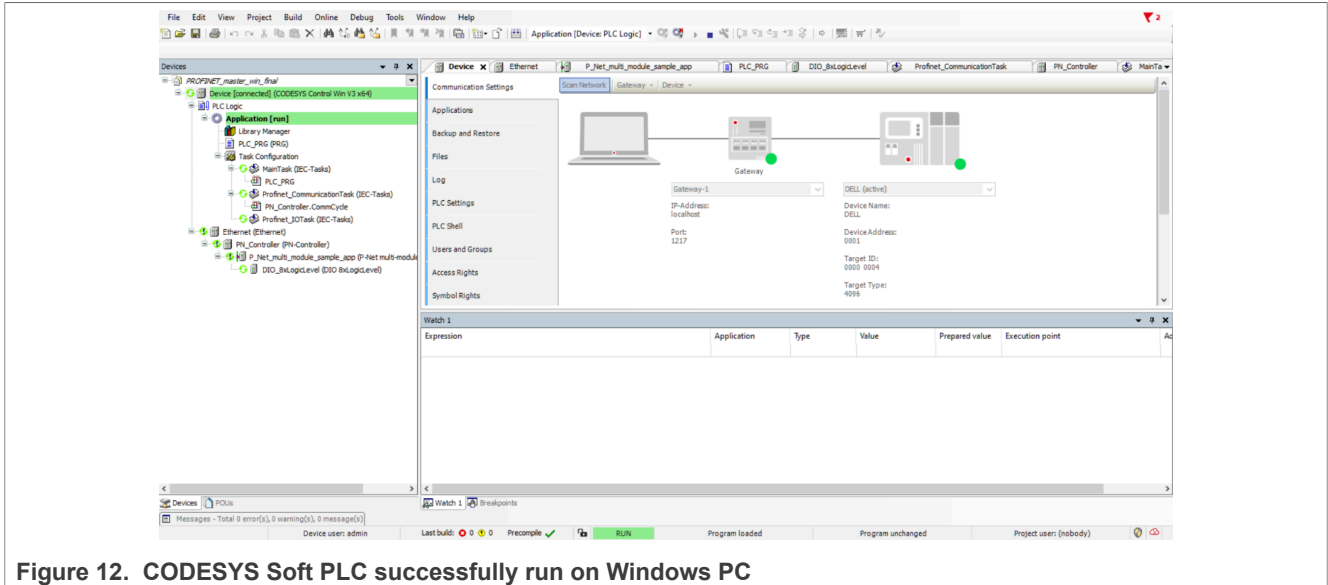


Figure 12. CODESYS Soft PLC successfully run on Windows PC

After the connection between the two boards is established, the LED should be flashing.

5 Debugging

5.1 Option A: Windows PC is used as a PLC Controller

Wireshark can be used directly on the PC where CODESYS Soft PLC runs. Wireshark captures all the Ethernet packets on the configured Ethernet interface.

5.2 Option B: i.MX 8M Plus is used as a PLC Controller

The resulting Ethernet traffic can be studied by connecting the two boards and a host PC with Wireshark to a managed switch. Configure the mirroring/monitoring function for the port connected to the PC.

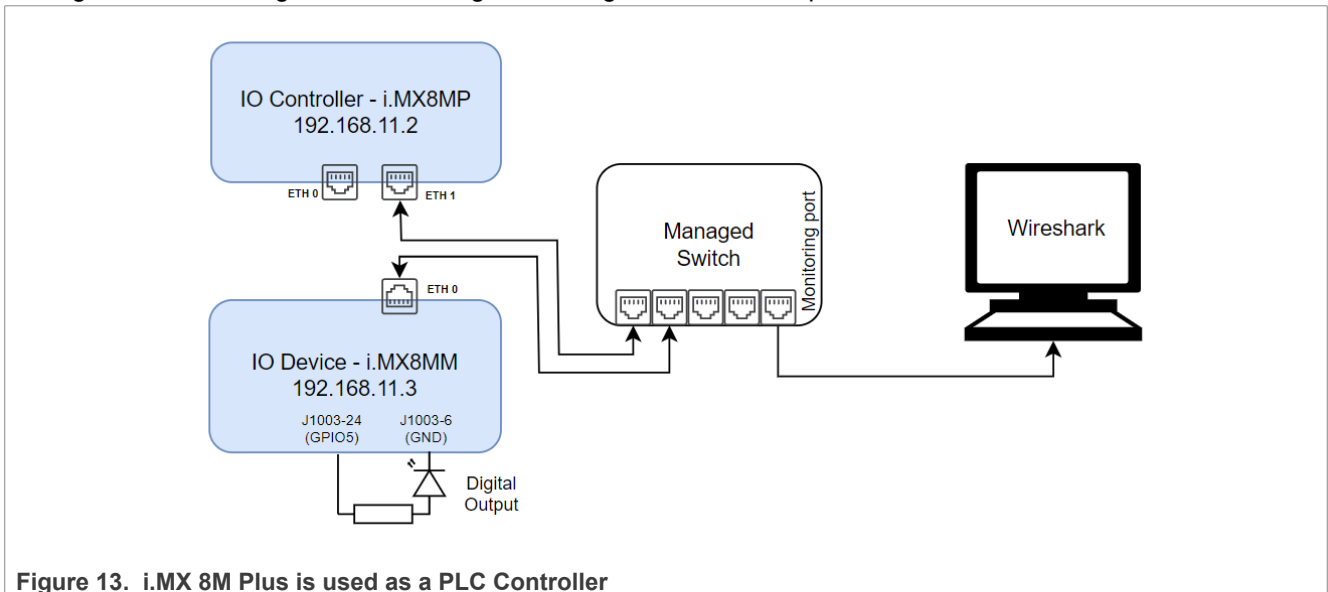


Figure 13. i.MX 8M Plus is used as a PLC Controller

6 Acronyms

Table 1. Acronyms

Acronym	Description
CODESYS	Controller Development System
IRT	Isochronous Real-Time
LwIP	Lightweight IP
NRT	Non-Real-Time
PLC	Programmable Logic Controller
PROFINET	Process Field Network
RT	Real-Time
RTOS	Real-Time Operating System

7 Revision history

Table 2. Revision history

Revision number	Date	Substantive changes
0	04/2023	Initial release

8 Legal information

8.1 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.

8.2 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 <http://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 data sheet 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.

8.3 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.

Contents

1	Introduction	2
1.1	Hardware setup and equipment	2
1.2	Software environment	3
1.2.1	Creating and flashing the i.MX 8M Mini client software image	3
1.2.2	Retrieving and flashing the i.MX 8M Plus controller software image (optional, needed only if the CODESYS soft PLC is running on an i.MX 8M Plus EVK)	3
2	Building the PROFINET stack and the sample application for i.MX 8M Mini	3
2.1	Prerequisites for the host Ubuntu PC	3
2.2	Disabling the Ethernet driver from U-Boot and Linux kernel	4
2.2.1	Disabling the Ethernet driver from U-Boot	4
2.2.2	Disabling the Ethernet driver from the Linux kernel	4
2.3	Importing the PROFINET stack and the sample application	5
2.4	Application structure	6
2.4.1	Examples directory	6
2.4.2	Middleware directory	6
2.5	Building the sample application and preparing the SD card	6
2.6	Setting up the U-Boot variables	7
3	Implementing the controller	8
3.1	Option A: Running the PLC controller on a Windows-based PC	8
3.2	Option B: Run the PLC controller on i.MX 8M Plus	9
3.3	Implementing the CODESYS application	11
4	Running the application	14
5	Debugging	16
5.1	Option A: Windows PC is used as a PLC Controller	16
5.2	Option B: i.MX 8M Plus is used as a PLC Controller	16
6	Acronyms	17
7	Revision history	17
8	Legal information	18

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