AN15047

i.MX 95 Dual-Ethernet on M7 and A55

Rev. 1.0 — 19 May 2026 **Application note**

Document information

Information	Content
Keywords	AN15047, i.MX 95, Ethernet
Abstract	This application note explains how to enable and use the independent Ethernet interfaces on different cores simultaneously - one assigned to the Cortex-M7 cluster and another to the Cortex-A55 cluster.

1 Introduction

This application note explains how to enable and use the independent Ethernet interfaces on different cores simultaneously - one assigned to the Cortex-M7 cluster and another to the Cortex-A55 cluster. It explains the architectural considerations, hardware partitioning, and configuration steps required to achieve concurrent operation, while maintaining the isolation between the real-time and application domains.

2 Prerequisites

This chapter describes the hardware and software prerequisites.

2.1 Hardware

- NXP i.MX 95 19x19 LPDDR5 EVK development kit
- 1x USB-Type C cable for the debug port (DBG port)
- 1x USB-Type C cable for the serial download port (USB1 port)
- 2x Ethernet cables

2.2 Software

- Host PC running Ubuntu 22.04 or later.
- UUU tool to write the Linux image to the EVK.
- Yocto BSP LF-6.12.49-2.2.0. The documentation is in this ZIP file.
- MCUXpresso SDK v.25.12.00.

3 Ethernet architecture

The Ethernet and Network Controller (NETC) domain is the dedicated networking subsystem in the i.MX 95 architecture that provides the Ethernet and Time-Sensitive Networking (TSN) functionality. It contains the NETC complex (three Ethernet Controllers (ENETC) and a PCI Express Root Complex Integrated Endpoint (iEP-RC)), an internal Ethernet Serializer/Deserializer (SerDes) PHY for the ENETC2, the audio support blocks (SAI/MQS), and a dedicated Trusted Resource Domain Controller (TRDC) instance for access isolation to the NETC functions inside the NETC complex.

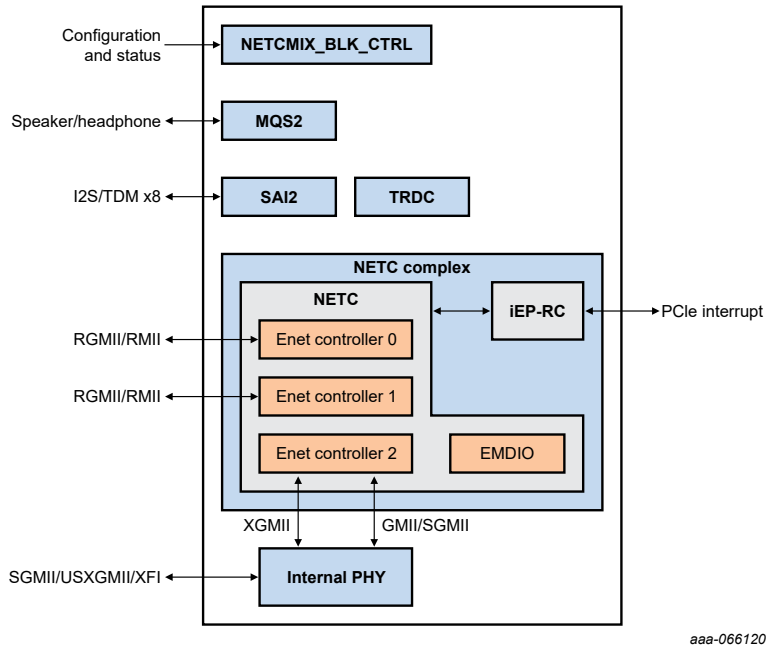

Figure 1. NETC domain architecture

The NETC is exposed to the system software as a PCIe integrated endpoint via the integrated Endpoint Root Complex (iEP-RC). Through this PCIe interface, the software accesses the NETC configuration registers using the PCIe Enhanced Configuration Access Mechanism (ECAM). The ECAM is a standardized method for mapping a PCIe device configuration space directly into the Memory-Mapped I/O (MMIO) address space of the CPU. The NETC ECAM window is 2 MB wide and is organized as two PCIe buses, 1 MB each. Each bus can address up to 32 devices with up to 8 functions each, each function occupying 4 kB of the ECAM space. A function location in the ECAM can be configured using the RID Assignment Register (RIDAR), whose PBUS field selects the bus and whose RID field defines the function requester ID. The effective ECAM address can be calculated as PBUS << 20 + RID << 12.

The iEP-RC also supports event reporting, initializations through the Integrated Endpoint Register Block (IERB), and low-level reset and control operations through the Privileged Register Block (PRB). Within this architecture, the ENETC0, ENETC1, ENETC2, EMDIO, and other NETC functions are exposed as independent functions that can be placed in the ECAM space by software, during the initialization of the IERB.

Each Ethernet controller supports multiple Station Interfaces (SIs) - one physical SI and multiple virtual SIs.

Each ENETC interface can be connected to an external PHY configurable through a standard MDIO interface (Figure 3). There is only one external MDIO controller, which communicates to all the external PHYs through the same pair of pins. To allow independent, isolated access to the MDIO by each ENETC interface, a set of MDIO registers is provided in each ENETC register space. The access to the PHYs is automatically multiplexed when the correct PHY address is programmed for each ENETC. The PHY address must be programmed at the initialization time in the NETC IERB registers.

The resource isolation between the cores is enforced by the TRDC. All the NETC transactions on the internal AXI bus carry the Function Auxiliary Information (FAUX) on the sideband, including a Local Domain Identifier (LDID) that is programmable per function through each function FAUXR register. This LDID is used by the TRDC to enforce isolation.

4 Approach

The goal of this application is to demonstrate the assignment of the ENETC0 to the Cortex-M7 domain and ENETC2 to the Cortex-A55 domain.

Because the System Manager programs the TRDC, we reconfigure it to assign the resources to cores:

- For Cortex-M7: the NETC block-control, NETC common registers (IERB, PRB), PCIe bus 0, ENETC0 and its pins, and the LDID for the ENETC0 PF.
- For Cortex-A55: the ENETC2 and its pins, PCIe bus 1, MDIO, and the LDID for the ENETC2 PF.

To keep things simple and to minimize code changes, all resources used by the Cortex-M7 core are on PCIe bus 0 and all resources used by the Cortex-A55 core are on PCIe bus 1. Compared to the default configuration, ENETC0 remains assigned to PBUS 0, while ENETC2 is moved to PBUS 1 by programming E2RIDAR[PBUS].

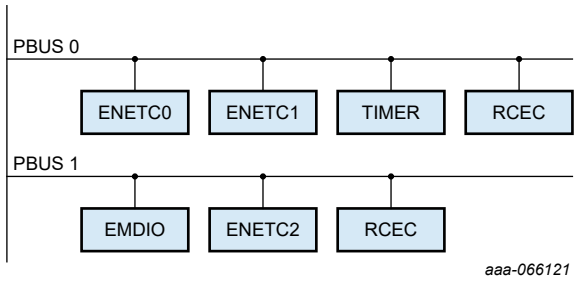

Figure 2. Allocation of the NETC resources to the PCIe buses

Because the Cortex-M7 software starts first, it performs the early NETC domain initialization. It configures the NETC IERB, including the correct LDID for ENETC2.

The MDIO ownership model is then configured to avoid cross-core contention. Because the external MDIO controller is centralized, it must remain owned by only one software environment. In this implementation, the Cortex-A55 partition retains the shared central EMDIO path for the ENETC2 PHY management, while the Cortex-M7 partition uses the ENETC0 MAC port EMDIO access path for its own PHY management. Since the Cortex-M7 application starts first, it must also configure the MDIO-related pin mutex before accessing the PHY from the ENETC0 side.

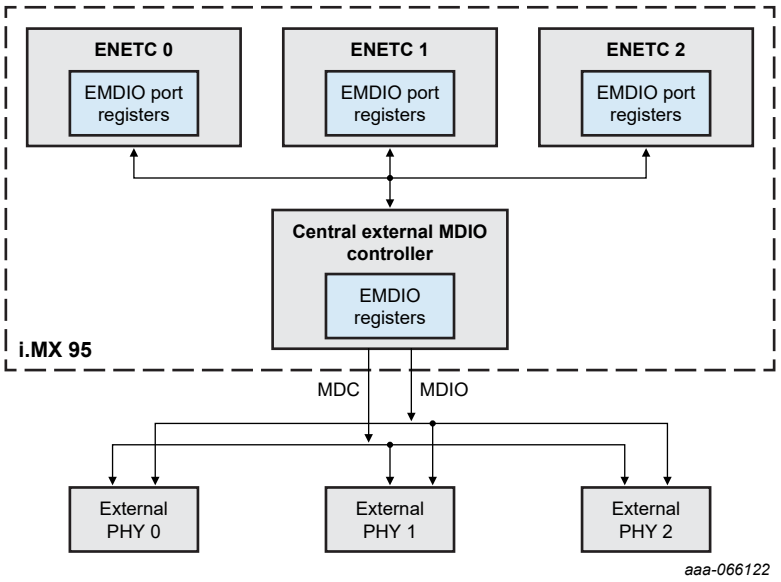

Figure 3. i.MX 95 MDIO architecture

In the U-Boot, remove the initialization of the NETC block and interfaces.

Update the Linux kernel so that:

1. It does not perform the global NETC initialization (done by the Cortex-M7 core).
2. It does not probe the ENETC0.
3. It enumerates the PCIe devices only on PCIe bus 1.

5 Implementation

1. System Manager:
   a. In `devices/MIMX95/configtool/netcmix.cfg`, divide the ECAM address space into two regions: ECAM1 mapped to PBUS0 and ECAM2 mapped to PBUS1. This follows the specifications from the Memory Map (Section 3.3, Table 14) and NETC Device Memory Map Layout (Section 106.6, Table 839).

```
# original definition
NETC_ECAM:           PD_NETC, MBC_E0=1.0-7
# modified split definition
NETC_ECAM1:          PD_NETC, MBC_E0=1.0-3
NETC_ECAM2:          PD_NETC, MBC_E0=1.4-7
```

   b. In `configs/mx95evk.cfg`, assign the NETC domain and NETC0 and its pins to Cortex-M7 core, keeping the MDIO and NETC2 at the Cortex-A55 core. Assign the ECAM1 space to the Cortex-M7 core and the ECAM2 to the Cortex-A55 core.

Table 1. Configuration for both cores

```
#=========================================#
# M7 EENV                                 #
#=========================================#
…
# Resources
…
BLK_CTRL_NETCMIX    OWNER
NETC                OWNER, test
NETC0               OWNER
NETC_ECAM1          OWNER
NETC_IERB           OWNER
NETC_LDID3          OWNER, kpa=0, sid=0x22
NETC_PRB            OWNER

# Pins
…
PIN_ENET1_MDC       OWNER
PIN_ENET1_MDIO      OWNER
PIN_ENET1_RD0       OWNER
PIN_ENET1_RD1       OWNER
PIN_ENET1_RD2       OWNER
PIN_ENET1_RD3       OWNER
PIN_ENET1_RX_CTL    OWNER
PIN_ENET1_RXC       OWNER
PIN_ENET1_TD0       OWNER
PIN_ENET1_TD1       OWNER
PIN_ENET1_TD2       OWNER
PIN_ENET1_TD3       OWNER
PIN_ENET1_TX_CTL    OWNER
PIN_ENET1_TXC       OWNER
```

```
#=========================================#
# A55 non-secure EENV                     #
#=========================================#
…
# Resources
…
NETC2               OWNER
NETC_ECAM2          OWNER
NETC_EMDIO0         OWNER
NETC_LDID5          OWNER, kpa=0, sid=0x24
NETC_TIMER0         OWNER

# Pins
…
PIN_ENET2_MDC       OWNER
PIN_ENET2_MDIO      OWNER
PIN_ENET2_RD0       OWNER
PIN_ENET2_RD1       OWNER
PIN_ENET2_RD2       OWNER
PIN_ENET2_RD3       OWNER
PIN_ENET2_RX_CTL    OWNER
PIN_ENET2_RXC       OWNER
PIN_ENET2_TD0       OWNER
PIN_ENET2_TD1       OWNER
PIN_ENET2_TD2       OWNER
PIN_ENET2_TD3       OWNER
PIN_ENET2_TX_CTL    OWNER
PIN_ENET2_TXC       OWNER
```

2. For the Cortex-M7 core, start from the MCUXSDK `lwip_ping` example:
   a. Configure the ENETC0 to use the EMDIO port MAC registers for PHY communication.

```
mdioConfig.mdio.type = kNETC_ExternalMdio;  // use EMDIO port MAC
 registers
NETC_IERB->L0BCR = NETC_IERB_L0BCR_MDIO_PHYAD_PRTAD(0x1);  // set the
 ENETC0 PHY address
```

   b. Set the LDID for ENETC2 and move the ENETC2 to PCIe bus 1.

```
NETC_IERB->CFG_ENETC_INST[2].EFAUXR = 4u;
```

```
NETC_IERB->CFG_ENETC_INST[2].ERIDAR = NETC_IERB_ERIDAR_PBUS(1u) |
 NETC_IERB_ERIDAR_RID(0x80u);
```

3. U-Boot:
   a. In `board/freescale/imx95_evk/imx95_evk.c`, remove the NETC initialization. Comment out `netc_init()` and `netc_regulator_enable()` in `board_init()`.

```
#if IS_ENABLED(CONFIG_TARGET_IMX95_19X19_EVK)
-       netc_regulator_enable("regulator-m2-pwr", true);
+       //netc_regulator_enable("regulator-m2-pwr", true);
#endif
-       netc_init();
+       //netc_init();
```

4. Linux:
   a. In the kernel device tree, remove `netc_blk_ctrl` and `netcmix_blk_ctrl` and move the ENETC2 under PCIe bus 1. When moving the ENETC2 from PCIe bus 0 to PCIe bus 1, the ENETC Requester ID in the `msi-map`, `iommu-map`, and ENETC2 register definition nodes must match the value configured through the IERB registers.

Table 2. Definitions

Original definition of NETC domain	Modified definition

`netc_blk_ctrl: netc-blk-ctrl@4cde0000 {`
`  compatible = "nxp,imx95-netc-blk-ctrl";`
`  ...`
 	
`  pcie_4ca00000: pcie@4ca00000 {`
`  compatible = "pci-host-ecam-generic";`
`  reg = <0x0 0x4ca00000 0x0 0x100000>;`
`...`
`  bus-range = <0x0 0x0>;`
`  msi-map = <0x0 &its 0x60 0x1>,  //ENETC0 PF`
`  <0x10 &its 0x61 0x1>, //ENETC0 VF0`
`  <0x20 &its 0x62 0x1>, //ENETC0 VF1`
`  <0x40 &its 0x63 0x1>, //ENETC1 PF`
`  **<0x80 &its 0x64 0x1>, //ENETC2 PF**`
`  **<0x90 &its 0x65 0x1>, //ENETC2 VF0**`
`  **<0xa0 &its 0x66 0x1>, //ENETC2 VF1**`
`  <0xc0 &its 0x67 0x1>; //NETC Timer`
`  iommu-map = <0x0 &smmu 0x20 0x1>,`
`  <0x10 &smmu 0x21 0x1>,`
`  <0x20 &smmu 0x22 0x1>,`
`  <0x40 &smmu 0x23 0x1>,`
`  **<0x80 &smmu 0x24 0x1>,**`
`  **<0x90 &smmu 0x25 0x1>,**`
`  **<0xa0 &smmu 0x26 0x1>,**`
`  <0xc0 &smmu 0x27 0x1>;`
`  ...`
`  enetc_port0: ethernet@0,0 {`
`  compatible = "fsl,imx95-enetc";`
`  reg = <0x000000 0 0 0 0>;`
`  ...`
`};`
 	
`enetc_port1: ethernet@8,0 {`
`  compatible = "fsl,imx95-enetc";`
`  reg = <0x004000 0 0 0 0>;`
`  ...`
`};`
 	
`**enetc_port2: ethernet@10,0 {**`
`  **compatible = "fsl,imx95-enetc";**`
`  **reg = <0x008000 0 0 0 0>;**`
`  ...`
`**};**`
 	
`netc_timer: ethernet@18,0 {`
`  compatible = "fsl,imx95-netc-timer";`
`  reg = <0x00c000 0 0 0 0>;`
`  ...`
`};`
 	
`rcec@1,0 {`
`  reg = <0x000800 0 0 0 0>;`
`  interrupts = <1>;`
`};`
`  };`
	`/delete-node/ &netcmix_blk_ctrl;`
	`/delete-node/ &netc_blk_ctrl;`
	
	`pcie_4cb00000: pcie@4cb00000 {`
	`  compatible = "pci-host-ecam-generic";`
	`  reg = <0x0 0x4cb00000 0x0 0x100000>;`
	`  ...`
	`  msi-map = **<0x180 &its 0x64 0x1>, //ENETC2 PF**`
	`  **<0x190 &its 0x65 0x1>, //ENETC2 VF0**`
	`  **<0x1a0 &its 0x66 0x1>; //ENETC2 VF1**`
	`  iommu-map = **<0x180 &smmu 0x24 0x1>,**`
	`  **<0x190 &smmu 0x25 0x1>,**`
	`  **<0x1a0 &smmu 0x26 0x1>;**`
	`  bus-range = <0x1 0x1>;`
	`  ...`
	
	`  netc_emdio: mdio@0,0 {`
	`  compatible = "fsl,imx95-netc-emdio";`
	`  reg = <0x010000 0 0 0 0>;`
	`  ...`
	`  };`
	
	`  **enetc_port2: ethernet@10,0 {**`
	`  **compatible = "fsl,imx95-enetc";**`
	`  **reg = <0x018000 0 0 0 0>;**`
	`...`
	`  **};**`
	
	`  rcec@1,0 {`
	`reg = <0x010800 0 0 0 0>;`
	`interrupts = <1>;`
	`  };`
	`};`

Table 2. Definitions

Original definition of NETC domain	Modified definition

pcie_4cb00000: pcie@4cb00000 {
compatible = "pci-host-ecam-generic";
reg = <0x0 0x4cb00000 0x0 0x100000>;
...
 	
netc_emdio: mdio@0,0 {
compatible = "fsl,imx95-netc-emdio";
reg = <0x010000 0 0 0 0>;
...
};
 	
rcec@1,0 {
reg = <0x010800 0 0 0 0>;
interrupts = <1>;
};
};
};	

6 Testing

This chapter describes the testing.

6.1 Setting up MCUXpresso SDK and building the Cortex-M7 application

1. Set up the MCUXpresso SDK v.25.12.00 environment following the MCUXSDK getting started guide.
2. Initialize the workspace and update the repositories with the i.MX 95-specific code.

```
cd ~/
west init -m https://github.com/nxp-mcuxpresso/mcuxsdk-manifests.git
 mcuxpresso-sdk --mr v25.12.00
cd ~/mcuxpresso-sdk
```

3. Apply the patch for the `lwip_ping` demo. The patch is in the `meta-imx-dual-eth-m7-a55` layer, downloaded for step 2 in Section 6.2 in the MCUXSDK getting started guide.

```
cd ~/mcuxsdk/examples/
git apply 0001-imx95-lwip_ping-setup-ENETC2-for-Linux.patch
```

4. Build the application.

```
cd lwip_examples/lwip_ping/freertos
west build  -b imx95lpd5evk19 . -Dcore_id=cm7
```

6.2 Setting up Yocto and building the Linux image

1. Follow the steps described in Section 3.4 from the i.MX Yocto project user guide to prepare your Yocto environment. We will further assume that the BSP directory is `~/imx-yocto-bsp`.
2. Clone the `meta-imx-dual-eth-m7-a55` layer.

```
cd ~/imx-yocto-bsp/sources
git clone -b lf-6.12.49-2.2.0_25.12 https://github.com/nxp-imx-support/meta-
imx-dual-eth-m7-a55
```

3. Copy the resulted M7 binary into the `meta-imx-dual-eth-m7-a55` Yocto layer.

```
cp ~/mcuxpresso-sdk/mcuxsdk/examples/lwip_examples/lwip_ping/freertos/build/
lwip_ping_freertos_cm7.bin ~/imx-yocto-bsp/sources/meta-imx-dual-eth-m7-a55/
recipes-bsp/imx-mkimage/files/
```

4. Set up the build directory.

```
cd ~/imx-yocto-bsp
DISTRO=fsl-imx-wayland MACHINE=imx95-19x19-lpddr5-evk source imx-setup-
release.sh -b build
```

5. Add the `meta-imx-dual-eth-m7-a55` layer to the Yocto BBLAYERS.

```
bitbake-layers add-layer ../sources/meta-imx-dual-eth-m7-a55
```

6. Build the image.

```
bitbake imx-image-core
```

6.3 Setting up the EVK and testing the application

1. Connect the USB1 port on the EVK to your PC using a USB-Type C cable. Connect the DBG port of the EVK to your PC through a second USB Type-C cable. Connect the ENETC0 interface (J11) to your PC using an Ethernet cable. Connect the ENETC2 interface (J28) to a network with the DHCP and Internet access.
2. On your PC, open all four serial ports using your preferred serial emulator. The first port is used by the Cortex-M7 core, the third port by the Cortex-A55 core (Linux), and the fourth port by the Cortex-M33 core (OEI/System Manager).
3. To test the Cortex-M7 application, configure the host PC IP address to 192.168.0.100/24. This is the address that the `lwip_ping` application will ping.
4. Write the image into the eMMC. The resulted image is ~/`imx-yocto-bsp/build/tmp/deploy/images/imx95-19x19-lpddr5-evk/imx-image-core-imx95-19x19-lpddr5-evk.rootfs.wic.zst`. Switch the EVK into the Serial Download mode SW7[1001] and power on the board. Run the following command on your PC:

```
uuu -b emmc_all imx-image-core-imx95-19x19-lpddr5-evk.rootfs.wic.zst
```

5. Switch the EVK to the eMMC boot mode SW7[1010] and boot the board.
6. The `lwip_ping` application demonstrates a ping demo on the lwIP TCP/IP stack, which uses the ICMP protocol. The application periodically sends ICMP echo requests to a PC and processes the PC's reply. You can also ping address 192.168.0.102 from the PC. The lwIP stack on the Cortex-M7 core must reply.
7. In the Linux on the EVK, you can ping 8.8.8.8 (or a local address) to test the ENETC2 interface.

7 Abbreviations

Table 3. Abbreviations

Abbreviation	Description
AXI	Advanced Extensible Interface
BSP	Board Support Package
DHCP	Dynamic Host Configuration Protocol
eMMC	embedded Multi-Media Card
EVK	Evaluation Kit
ICMP	Internet Control Message Protocol
IP	Internet Protocol

Table 3. Abbreviations...*continued*

Abbreviation	Description
MAC	Media Access Controller
MQS	Medium Quality Sound
NETC	Network Controller
OEI	Optional Executable Image
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect express
PF	Post Filter
PHY	Physical Layer
RC	Return Code
SAI	Serial Audio Interface
SDK	Software Development Kit
SI	Signal Integrity
TCP/IP	Transmission Control Protocol/Internet Protocol
TRDC	Trusted Resource Domain Controller
TSN	Time-Sensitive Networking
USB	Universal Serial Bus

8 Note about the source code in the document

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

Copyright 2026 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.

9 Revision history

Table 4. Revision history

Document ID	Release date	Description
AN15047 v.1.0	19 May 2026	Initial release

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.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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.

Contents