AN15119

Ethernet-to-Six-CAN FD Bridge Based on MCX E31B

Rev. 1.0 — 27 July 2026 Application note

Document information

Information	Content
Keywords	MCX E31B, Ethernet, CAN FD, SocketCAN, E2CF, eth2can, FRDM-MCXE31B, AN15119
Abstract	This application note describes a bridge that exposes six MCX E31B CAN FD channels to a Linux host as standard SocketCAN devices over a raw Ethernet E2CF link.

1 Introduction

This application note describes an Ethernet-to-six-CAN FD bridge based on the MCX E31B microcontroller. The bridge connects a Linux host to six physical CAN FD channels through a raw Layer-2 Ethernet transport named Ethernet-to-CAN FD (E2CF). On Linux, the `eth2can` kernel module exposes the gateway as standard SocketCAN network devices, `eth2can0` through `eth2can5`.

The document focuses on customer hardware bring-up, including hardware connections, firmware and driver roles, installation, basic traffic checks, performance test entry points, and troubleshooting. The implementation notes are limited to details that help users understand behavior and diagnose integration issues.

2 System overview

Figure 1 provides an overview of the MCX E31B Ethernet-to-six-CAN FD bridge and its connection to the Linux host and CAN FD networks.

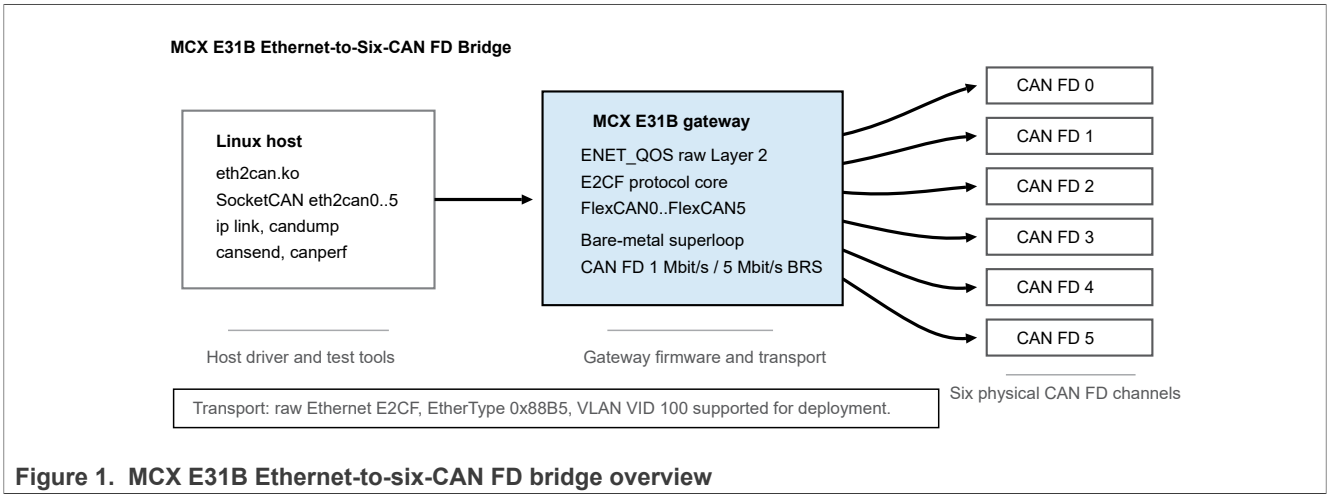

Figure 1. MCX E31B Ethernet-to-six-CAN FD bridge overview

The Linux host does not need a CAN controller for each bus. Instead, it loads `eth2can.ko`, binds the module to an Ethernet interface, and exchanges E2CF frames with the MCX E31B gateway. The gateway firmware receives and transmits CAN FD frames on `FlexCAN0` through `FlexCAN5` and moves them over raw Ethernet frames using `ENET_QOS`, without an IP stack.

Table 1. Delivery configuration

Item	Value used by this delivery
Linux CAN interface	SocketCAN devices `eth2can0` through `eth2can5`
Default CAN FD rate	Nominal 1 Mbit/s, data phase 5 Mbit/s, BRS enabled
Gateway transport	Raw Ethernet, EtherType 0x88B5; VLAN VID 100 supported for deployment
Default bring-up frame mode	Untagged E2CF frames, peer MAC learned from gateway heartbeat
Customer test tool	`linux/can_testcase/canperf` for latency and bandwidth verification

2.1 Bridge operating model

The gateway is a Layer-2 bridge for CAN FD traffic, not a CANopen, J1939, or application-protocol gateway. CAN identifiers, DLC, BRS state, payload, and error-state indicators are carried in E2CF DATA records and exposed on Linux through the standard SocketCAN API. It allows existing Linux CAN tools and applications to run without a dedicated CAN controller on the host.

Each E2CF channel maps one-to-one to a FlexCAN instance and one SocketCAN network device. Per-channel transmit order is preserved by using one active FlexCAN transmit message buffer and a 16-slot host echo window. Ordering between different CAN channels is intentionally independent, matching the behavior of six separate physical CAN controllers.

Table 2 summarizes the customer-visible behavior of the bridge during normal operation.

Table 2. Customer-visible bridge behavior

Bridge aspect	Behavior visible to the customer
Host model	Six SocketCAN devices, `eth2can0` through `eth2can5`, on one Ethernet interface
Transport model	Raw Ethernet frames with EtherType 0x88B5; VLAN VID 100 can be enabled for deployment
CAN model	CAN FD BRS by default at 1 Mbit/s nominal and 5 Mbit/s data phase
Ordering model	Per-channel ordering is preserved; channels run independently
Acceptance model	Use `RESULT` and `EVIDENCE` lines from `canperf` plus clean driver/gateway counters

3 Hardware setup

The FRDM-MCXE31B board connects one RMII Ethernet PHY and six external CAN FD transceivers. Use CAN SIC-grade or otherwise application-appropriate CAN FD transceivers for the external CAN buses. Each physical CAN bus requires correct transceiver power, a common ground reference, correct CANH/CANL polarity, and 120 ohm termination at both ends of the bus.

Table 3. MCX E31B signal assignments

Function	MCX E31B signals used in this project
CAN0	FLEXCAN_0_TX PTA7 pin 100; FLEXCAN_0_RX PTA6 pin 102
CAN1	FLEXCAN_1_TX PTA11 pin 160; FLEXCAN_1_RX PTA12 pin 159
CAN2	FLEXCAN_2_TX PTE24 pin 157; FLEXCAN_2_RX PTE25 pin 158
CAN3	FLEXCAN_3_TX PTC28 pin 96; FLEXCAN_3_RX PTC29 pin 99
CAN4	FLEXCAN_4_TX PTC30 pin 101; FLEXCAN_4_RX PTC31 pin 103
CAN5	FLEXCAN_5_TX PTC27 pin 93; FLEXCAN_5_RX PTC26 pin 91
Ethernet RMII	MDIO PTB4, MDC PTB5, TXD0 PTC2, TXD1 PTD7, TX_EN PTD12, RX_DV PTC17, RXD0 PTC1, RXD1 PTC0, REF_CLK PTD11, PHY reset PTC3
Debug UART	LPUART_5_RX PTE3 pin 27; LPUART_5_TX PTE14 pin 26

The default loopback harness used by the repository test flow connects three independent CAN FD buses.

```
eth2can0 <-> eth2can4
eth2can1 <-> eth2can2
eth2can3 <-> eth2can5
```

Table 4. LED status indications

LED	GPIO	State	Meaning
SYS	PTC16	Blinking about 1 Hz	Firmware superloop is running
SYS	PTC16	On	Firmware stopped in a fault or fatal path
NET	PTB22	Off	PHY link is down

Table 4. LED status indications*...continued*

LED	GPIO	State	Meaning
NET	PTB22	Blinking	PHY link is up, but the E2CF peer heartbeat is not ready
NET	PTB22	On	Ethernet and E2CF link are ready
CAN	PTC14	Blinking	CAN RX or TX traffic is present
CAN	PTC14	On	At least one active CAN channel is error-passive or bus-off

3.1 Hardware integration checklist

For customer hardware bring-up, verify the board at the physical layer before investigating protocol behavior. The E2CF link can be healthy while a single CAN bus still fails because of termination, polarity, transceiver supply, or ground-reference issues.

Table 5. Hardware integration checklist

Area	Check points before software debugging
Ethernet PHY	RMII clock, MDIO/MDC access, PHY reset pin, link partner, and selected Linux --ifname
CAN transceivers	Transceiver power, common ground, CANH/CANL polarity, standby pins, and 120 ohm termination at both bus ends
Default harness	Use the validated pairs 0<->4, 1<->2, and 3<->5 before moving to customer wiring
VLAN path	For switched deployments, pass EtherType 0x88B5 and VLAN VID 100 without filtering
LEDs and UART	Use SYS/NET/CAN LED states and the debug UART as the first fault-localization signals

4 Software architecture

This section describes the firmware, Linux driver, E2CF protocol, and traffic-handling mechanisms used by the Ethernet-to-Six-CAN FD bridge.

4.1 MCX E31B firmware

The firmware is a bare-metal Cortex-M7 application. It does not use an RTOS or lwIP. CAN and Ethernet traffic is handled by interrupt paths. The main loop performs bounded housekeeping, such as aggregation flush checks, heartbeat, event, time, statistics messages, PHY link polling, LED updates, CAN error polling, and deferred debug-log drain.

The firmware start sequence consists of `BOARD_InitHardware`, status LED initialization, gateway time initialization, debug logging, `SysTick` configuration, raw Ethernet initialization, CAN hardware initialization, E2CF core initialization, and entry into superloop.

Table 6 lists the key firmware resources and configuration settings used by the gateway.

Table 6. Gateway firmware configuration

Firmware resource	Configuration
FlexCAN instances	Six instances, FlexCAN0 through FlexCAN5
FlexCAN PE clock	AIPS_PLAT_CLK at 80 MHz for the active gateway configuration
CAN receive resources	CAN0/CAN1/CAN2 use 8 RX Message Buffers (MBs); CAN3/CAN4/CAN5 use 4 RX MBs
CAN transmit resources	One active TX message buffer per channel, preserving per-channel order

Table 6. Gateway firmware configuration*...continued*

Firmware resource	Configuration
Software TX FIFO	16 entries per channel, matching the E2CF window depth
ENET_QOS resources	16 RX descriptors, 64 TX descriptors, 8 TX staging buffers
Interrupt priorities	CAN and EMAC use priority 1; `SysTick` uses priority 3

4.2 Linux driver

The Linux driver module is named eth2can. It binds to one lower Ethernet interface and creates six SocketCAN network devices. The module defaults to `ifname=eth0`, `vid=-1` for untagged bring-up, and an empty peer parameter so the driver can learn the MCX E31B peer MAC from heartbeat frames.

Each SocketCAN TX skb is validated, assigned a free echo slot, encoded as an E2CF DATA frame, and sent on the lower Ethernet interface. The echo slot is released only when the MCU returns a TXC record after the CAN frame completes on the CAN bus. If the 16-slot per-channel window fills, the driver stops that channel queue until TXC records free slots.

4.3 E2CF protocol

Figure 2 shows the E2CF data, completion, and management paths used for communication between the Linux host and the gateway.

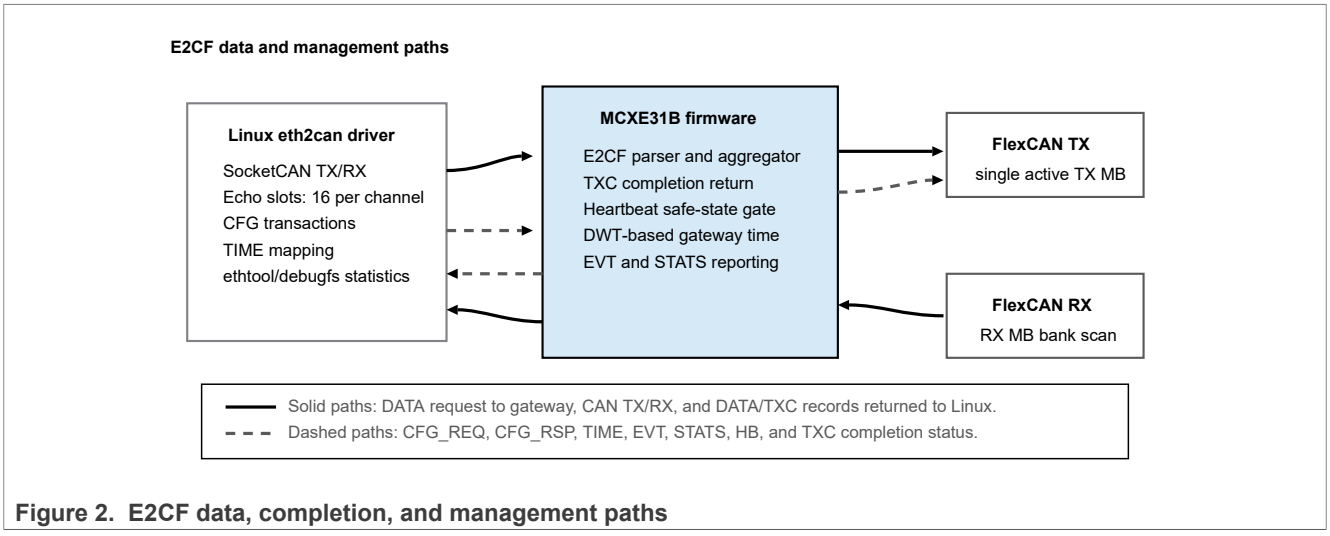

Figure 2. E2CF data, completion, and management paths

E2CF is a little-endian-packed, raw Ethernet protocol shared by the MCU firmware and the Linux driver. The two copies of `e2cf_proto.h` in `source/` and `linux/src/` are byte-identical and define the wire format. DATA carries CAN frames or transmit requests. TXC returns transmit completion status. EVT, CFG_REQ, CFG_RSP, TIME, HB, and STATS messages provide management, recovery, and diagnostics.

Table 7. E2CF protocol parameters

Protocol item	Value
Version	1
EtherType	0x88B5
VLAN	VID 100 supported for deployment; untagged frames allowed during bring-up
PCP usage	PCP 6 for DATA/TXC/EVT/TIME/HB; PCP 2 for CFG_REQ/CFG_RSP/STATS

Table 7. E2CF protocol parameters*...continued*

Protocol item	Value
Window depth	16 transmit slots per channel
MCU-to-Linux aggregation	Records are sent when full or when the 50 µs aggregation timer expires
Heartbeat	100 ms period; MCU enters safe state after 500 ms without Linux heartbeat
Timebase	Gateway monotonic time from the Cortex-M7 DWT cycle counter, reported through TIME messages

4.4 Traffic handling and recovery

The Linux driver applies normal SocketCAN backpressure when the 16-slot transmit window for a channel is full. The queue is restarted when TXC records arrive from the gateway. A TXC record is returned after CAN bus completion, so this mechanism covers the Linux driver, Ethernet transport, firmware queueing, and FlexCAN transmission path.

MCU-to-Linux DATA and TXC records are aggregated by record count or by the 50 µs aggregation timer. Aggregation reduces Ethernet frame overhead while preserving record sequence information. The Linux driver also tracks receive sequence gaps and distinguishes confirmed loss from in-window reordering before reporting dirty counters.

Table 8. Bridge operational states and diagnostic evidence

Condition	Expected bridge behavior	Primary evidence
Heartbeat present	The gateway peer is learned and the SocketCAN carrier remains on.	NET LED on, dmesg gateway-alive messages, EtherType 0x88B5 HB frames
Heartbeat timeout	The Linux carrier is turned off, and the gateway enters a safe state for host-driven traffic.	500 ms timeout path, NET LED not ready
CAN bus-off or error-passive	The channel state is reported through EVT/STATS messages and SocketCAN state handling.	CAN LED on, ethtool/debugfs counters, dmesg
Gateway reboot	The driver detects the reboot and schedules channel recovery instead of silently continuing with a stale state.	Reboot recovery log and renewed CFG sequence
Window pressure	The SocketCAN TX queue is stopped until TXC frees echo slots.	No fixed performance threshold; inspect counters and `canperf` evidence

5 Bring-up procedure

Follow the steps below to bring up the Ethernet-to-six-CAN FD bridge and verify basic communication between the Linux host and the MCX E31B gateway.

1. Build or verify the MCX E31B gateway firmware. If the board is not preprogrammed, open `e2cf_mcxe31.uvprojx` in Keil MDK, build the firmware, and program the device using the standard MCX E31B SWD programming flow.
2. Connect the Linux host Ethernet interface to the gateway Ethernet port and power the FRDM-MCXE31B board.
3. Install and load the Linux driver from the root repository.

```
sh linux/scripts/install_driver.sh
```

4. Check that six SocketCAN devices exist and are configured for CAN FD 1 Mbit/s / 5 Mbit/s operation.

```
ip -details link show type can
```

5. Run a basic receive/transmit check on one default harness pair.

```
candump eth2can4 &
cansend eth2can0 123##3001122334455667788
```

If the frame is received successfully, the Linux driver, raw Ethernet link, gateway firmware, CAN transceivers, and one physical CAN FD bus are all functioning for basic traffic.

5.1 Bring-up acceptance checkpoints

Use a staged bring-up sequence. Do not start with a bandwidth stress run until the Ethernet link, heartbeat, SocketCAN configuration, and at least one physical CAN pair have passed a basic traffic check.

Table 9. Bring-up acceptance checkpoints

Step	Pass condition
Driver load	`eth2can0` through `eth2can5` are registered and `dmesg` does not report module-load errors
Heartbeat	The driver observes the gateway heartbeat on the selected Ethernet interface and VLAN setting
CAN configuration	All six devices accept the CAN FD 1 Mbit/s / 5 Mbit/s configuration and transition to the up state
Basic traffic	A frame sent using `cansend` on one side of a default pair is visible with `candump` on the peer side
Latency	`canperf` latency --count 10000 reports PASS and an `EVIDENCE` line
Bandwidth	`canperf` bandwidth final confirmation reports `PASS`, `MSR`, and `counters=clean`

When a run fails, keep the final `RESULT` and `EVIDENCE` lines together with `dmesg`, `tcpdump` heartbeat capture, and `debugfs` or `ethtool` statistics. This arrangement provides enough context to isolate host, Ethernet, firmware, and CAN physical-layer issues.

6 Verification with `canperf`

The `canperf` utility is the customer-facing verification entry point. It runs over the SocketCAN devices and therefore verifies the Linux driver, Ethernet transport, MCX E31B firmware, and physical CAN FD buses together. Build it on the target Linux host:

```
cd linux/can_testcase
make
```

Run the latency test and the bidirectional maximum sustainable bandwidth test:

```
./canperf latency --count 10000
./canperf bandwidth
```

Use the final `RESULT` line as the customer-facing conclusion. The latency command reports p50, p99, and p99.9 end-to-end latency. The bandwidth command reports MSR, the maximum sustainable rate found by the final confirmation run. This application note intentionally does not specify fixed p99 or MSR thresholds; acceptance thresholds depend on the host, wiring, bus loading, and customer system constraints.

The EVIDENCE line is intended for repeatable records. A clean run reports zero_loss=yes and counters=clean, meaning the application did not detect lost frames and the selected driver/gateway loss, overflow, reject, starvation, and send-failure counters did not increase during the confirmation window.

7 Troubleshooting

Table 10 summarizes common bring-up and operational issues together with recommended diagnostic checks.

Table 10. Troubleshooting guide

Symptom	Checks
No eth2can0 through eth2can5 devices	Check dmesg for module load errors and confirm CONFIG_CAN, CONFIG_CAN_RAW, and CONFIG_CAN_DEV are enabled in the running kernel.
Missing kernel build directory	Install headers for the exact running kernel or rerun with KDIR=/path/to/kernel/build.
No gateway heartbeat	Check cable, selected --ifname, switch/VLAN path, firmware image, and tcpdump for EtherType 0x88B5 heartbeat traffic.
CAN frame not received	Check default harness pairs, 120 ohm termination at both bus ends, CANH/CANL polarity, transceiver power, and common ground.
CAN LED on	At least one active channel is error-passive or bus-off. Check rate, wiring, termination, polarity, and transceiver supply.
counters=dirty	Use seq_lost, rx_ovf, rejects, face_starved, send_fail, and emac_rxdrop deltas to locate the failing path.

A heartbeat capture command for bring-up is:

```
sudo tcpdump -eni eth0 'ether proto 0x88b5 or (vlan and ether proto 0x88b5)'
```

8 Acronyms

Table 11 lists the acronyms used in this document.

Table 11. Acronyms

Term	Description
BRS	Bit Rate Switch; CAN FD data phase at a higher bit rate
CAN FD	Controller Area Network Flexible Data-rate
DLC	Data Length Code
DWT	Data Watchpoint and Trace
E2CF	Ethernet-to-CAN FD raw Ethernet protocol used by this gateway
EMAC	Ethernet Media Access Controller
EVT	E2CF event message for CAN state and error reporting
HB	E2CF heartbeat message for peer discovery and safe-state supervision
MB	Message Buffer
MDC	Management Data Clock
MDIO	Management Data Input/Output
MSR	Maximum sustainable rate reported by canperf bandwidth

Table 11. Acronyms...*continued*

Term	Description
PCP	Priority Code Point
PHY	Physical Layer Device
RMII	Reduced Media Independent Interface
RTOS	Real-Time Operating System
SIC	Signal Improvement Capability
SocketCAN	Linux CAN networking subsystem
SWD	Serial Wire Debug
TXC	Transmit completion message returned by the MCU after CAN bus completion
VID	VLAN Identifier
VLAN	Virtual Local Area Network

9 Revision history

Table 12 summarizes revisions to this document.

Table 12. Revision history

Document ID	Release date	Description
AN15119 v.1.0	27 July 2026	Initial public 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