AN14543

Connect Barcode Scanner with MCX N Series USB Host Port

Rev. 1.0 — 22 January 2025 Application note

Document information

Information	Content
Keywords	AN14543, MCX N series, barcode scanner, USB host, FRDM-MCXN947
Abstract	This application note describes how to build a USB host port connected with a USB barcode scanner using the FRDM-MCXN947 board for demonstration.

1 Introduction

This application note describes how to build a USB host port connected with a USB barcode scanner using the FRDM-MCXN947 board for demonstration. NXP's MCX N series devices feature a high-speed (HS) USB port capable of reaching transmission speeds up to 480 Mbit/s and compatible with Full-speed mode. This speed is sufficient for transmitting barcode result data.

To ensure ease of use and compatibility with other devices, the examples use a USB host keyboard port for communication and a serial terminal to display the barcode scanner result.

2 USB barcode scanner

A barcode scanner, also called a barcode reader or price scanner, is a handheld input device that captures, reads, and decodes 1D or 2D barcode information. Usually, a barcode scanner connects with a computer or POS system with an RS-232 cable or a USB cable. This document introduces the USB cable-based barcode scanner that connects with the MCU USB host port.

This document uses the Newland's NLS-HR11 handheld 1D barcode scanner (HR11 Aringa) shown in Figure 1.

aaa-059084

Figure 1. USB barcode scanner

Most of the USB barcode scanners are based on the USB keyboard protocol. The user can update source code to support different types of barcode scanners with the help of this example.

3 USB host keyboard driver

NXP's MCX SDK provide host keyboard driver and example. The path is under the following SDK pack's folder:

`boards\frdmmcxn947\usb_examples\usb_host_hid_mouse_keyboard`

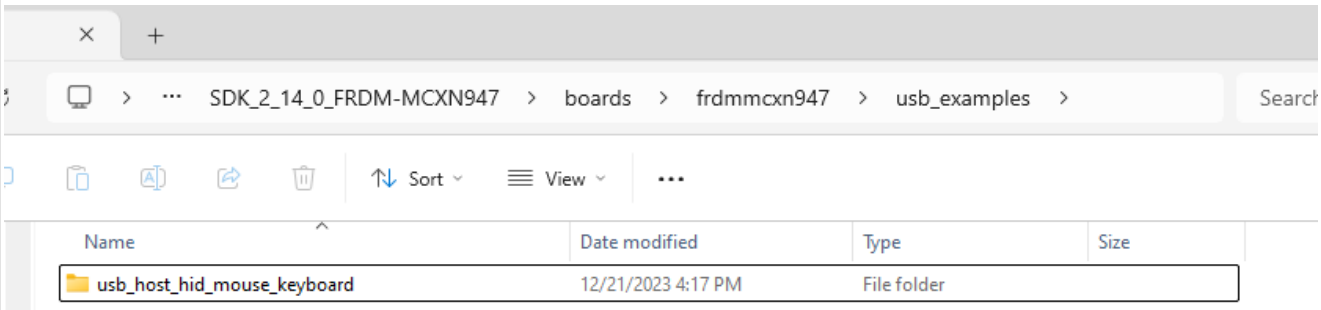

Figure 2. USB host keyboard example in SDK

For more information about USB, see USB 2.0 for Kinetis MCUs.

4 Demo implementation

This section introduces the hardware setup and how to modify the code to support the USB bar code scanner.

4.1 Hardware requirements

Figure 3 shows the connection overview for this reference design. The hardware requirements are as follows:

- One USB barcode scanner
- USB Type-C to USB Type-A converter dongle
- FRDM-MCXN947 board
- PC with serial terminal tool software installed

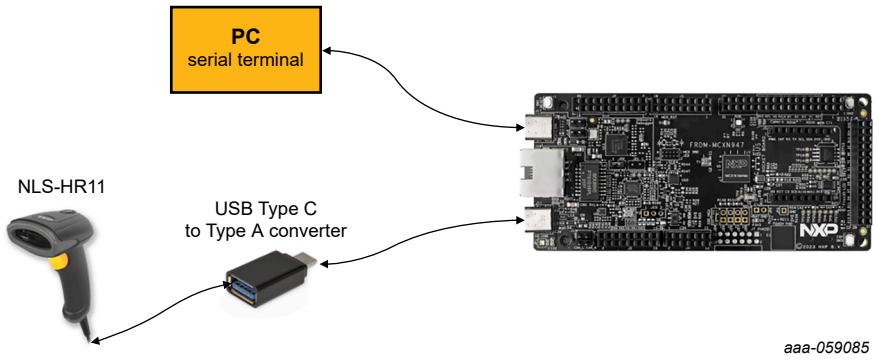

Figure 3. FRDM-MCXN947 connection with PC and barcode scanner

4.2 Example based on SDK

SDK provided `host_hid_mouse_keyboard_bm` example can be a good start.

1. Download the FRDM-MCXN947 package from https://mcuxpresso.nxp.com and install this package to MCUXpresso IDE. For information on selecting and installing an SDK package into the MCUXpresso, see the video tutorial Importing an SDK Package into MCUXpresso IDE.
2. Import the SDK example `host_hid_mouse_keyboard_bm` into your MCUXpresso workspace.

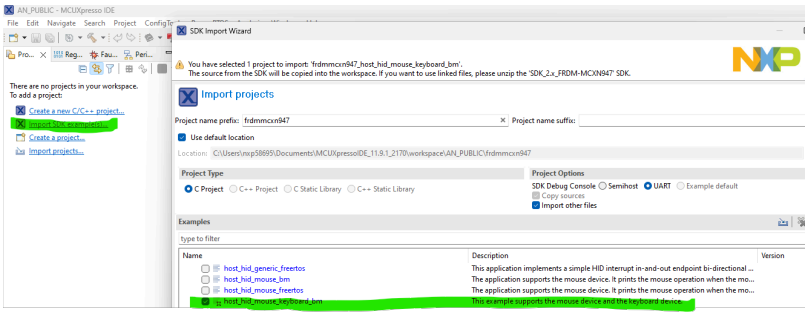

Figure 4. Import SDK example

3. Update the project name from `frdmmcxn947_host_hid_mouse_keyboard_bm` to `frdmmcxn947_host_hid_barcode_scanner`.

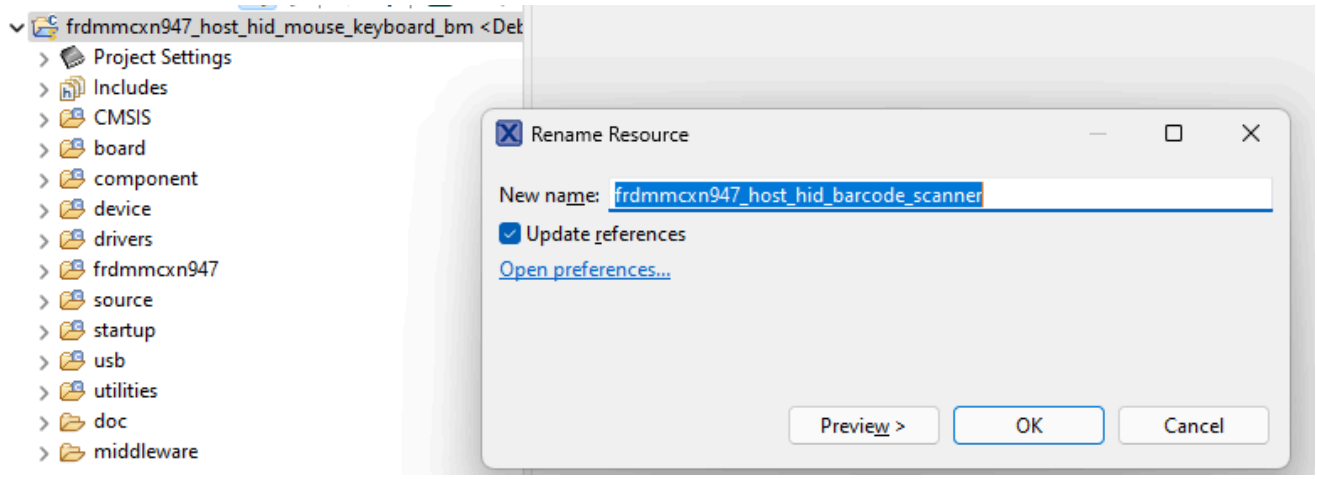

Figure 5. Rename the project's name

4. Remove `host_mouse.c` and `host_mouse.h` files from the project.

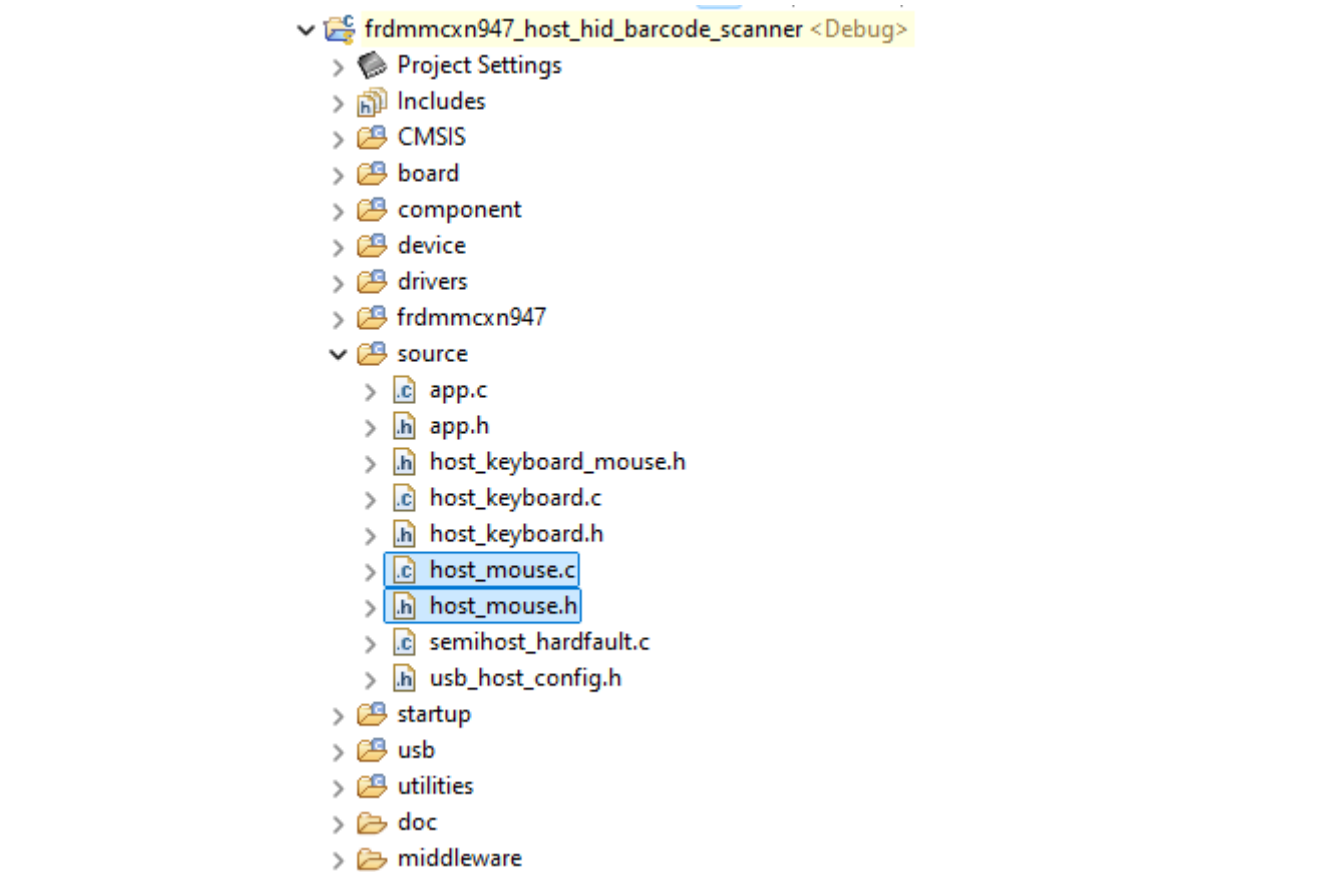

Figure 6. Remove unnecessary .c code files

5. Remove some code files from app.c, including the following:
   - Remove Line 15 `#include "host_mouse.h"`
   - Remove Line 67 and 68 `"g_HostHidMouse"`

```
/*! @brief USB host mouse instance global variable */
extern usb_host_mouse_instance_t g_HostHidMouse;
/*! @brief USB host keyboard instance global variable */
extern usb_host_keyboard_instance_t g_HostHidKeyboard;
usb_host_handle g_HostHandle;
```

Figure 7. Remove certain code files

6. To remove the mouse function, use the `USB_HostEvent()` code instead of the original content as below:

```
/*!
 * @brief USB isr function.
 */

static usb_status_t USB_HostEvent(usb_device_handle deviceHandle,
                                  usb_host_configuration_handle configurationHandle,
                                  uint32_t eventCode)
{
    usb_status_t status_keyboard;
    usb_status_t status = kStatus_USB_Success;

    switch (eventCode & 0x0000FFFFU)
    {
        case kUSB_HostEventAttach:
         status_keyboard = USB_HostHidKeyboardEvent(deviceHandle, configurationHandle,
 eventCode);
            if (status_keyboard == kStatus_USB_NotSupported)
            {
                status = kStatus_USB_NotSupported;
            }
            break;

        case kUSB_HostEventNotSupported:
            usb_echo("device not supported.\r\n");
            break;

        case kUSB_HostEventEnumerationDone:
         status_keyboard = USB_HostHidKeyboardEvent(deviceHandle, configurationHandle,
 eventCode);
            if (status_keyboard != kStatus_USB_Success)
            {
                status = kStatus_USB_Error;
            }
            break;

        case kUSB_HostEventDetach:
         status_keyboard = USB_HostHidKeyboardEvent(deviceHandle, configurationHandle,
 eventCode);
            if (status_keyboard != kStatus_USB_Success)
            {
                status = kStatus_USB_Error;
            }
            break;

        case kUSB_HostEventEnumerationFail:
            usb_echo("enumeration failed\r\n");
            break;

        default:
            break;
    }
    return status;
}
```

7. Then, remove the `USB_HostHidMouseTask(&g_HostHidMouse);` in the main function, as shown in Figure 8.

```
245
246 int main(void)
247 {
248     BOARD_InitBootPins();
249     BOARD_PowerMode_OD();
250     BOARD_InitBootClocks();
251     BOARD_InitDebugConsole();
252     CLOCK_SetupExtClocking(BOARD_XTAL0_CLK_HZ);
253
254     USB_HostApplicationInit();
255
256     while (1)
257     {
258         USB_HostTaskFn(g_HostHandle);
259         USB_HostHidKeyboardTask(&g_HostHidKeyboard);
260         USB_HostHidMouseTask(&g_HostHidMouse);
261     }
262 }
263
```

Figure 8. Remove USB_HostHidMouseTask

8. To support NLS-HR11 `SetReport` commands, update the `USB_HostHidControlCallback()` and `USB_HostHidKeyboardTask()`. The NLS-HR11 supports SetReport progress instead of the SDK code's SetProtocol.
9. Update the `kUSB_HostHidRunWaitSetProtocol` to `kUSB_HostHidRunWaitSetReport` and update `kUSB_HostHidRunSetProtocolDone` to `kUSB_HostHidRunSetReportDone` in the `host_keyboard_mouse.h` file.
10. Update the `kUSB_HostHidRunWaitSetProtocol` to `kUSB_HostHidRunWaitSetReport` and update `kUSB_HostHidRunSetProtocolDone` to `kUSB_HostHidRunSetReportDone` in `USB_HostHidControlCallback()` in the `host_keyboard.c` file as follows:

```
/*Update the following*/
else if (keyboardInstance->runWaitState == kUSB_HostHidRunWaitSetProtocol) /*
 hid set protocol done */
{
keyboardInstance->runState = kUSB_HostHidRunSetProtocolDone;
}
/*To*/
else if (keyboardInstance->runWaitState == kUSB_HostHidRunWaitSetReport) /*
 hid set report done */
{
keyboardInstance->runState = kUSB_HostHidRunSetReportDone;
}
```

11. For `USB_HostHidKeyboardTask()` support NLS-HR11 commands, first add below parameter in `USB_HostHidKeyboardTask()` function:

```
uint8_t reportLEDStatus[2];
```

12. Change `case kUSB_HostHidRunGetReportDescriptorDone:` and update the codes as below:

```
case kUSB_HostHidRunGetReportDescriptorDone: /* 4. hid set output report */
            keyboardInstance->runWaitState = kUSB_HostHidRunWaitSetReport;
            keyboardInstance->runState     = kUSB_HostHidRunIdle;
            /* third: set output report */
```

```
            /*                                      (Not used)        Kana
Compse          Scroll Lock     Caps Loc      Num lock*/
            reportLEDStatus[0] = (0x00 << 5) | (0x00 << 4) | (0x00 << 3) | (0x01 << 2)
| (0x00 << 1) | (0x01<<0);
            if (USB_HostHidSetReport(keyboardInstance->classHandle,
                                     0x00,          // reportId
                                     0x02,          // reportType : Output Report
                                     reportLEDStatus,
                                     1,
                                     USB_HostHidControlCallback,
                                     keyboardInstance) != kStatus_USB_Success)
            {
                usb_echo("error in USB_HostHidSetProtocol\r\n");
            }

            break;
```

13. Update `case kUSB_HostHidRunSetProtocolDone:` in `USB_HostHidControlCallback()` to `case kUSB_HostHidRunSetReportDone:`.
14. Finally, compile the project and download it to FRDM-MCXA276:
    - Ensure that the FRDM-MCXA276 board connects with the USB barcode scanner with J6 and MCU-Link's USB J17 connects to the PC.
    - Ensure that the PC is running a serial terminal application with correct VCOM link settings, as shown in Figure 9.

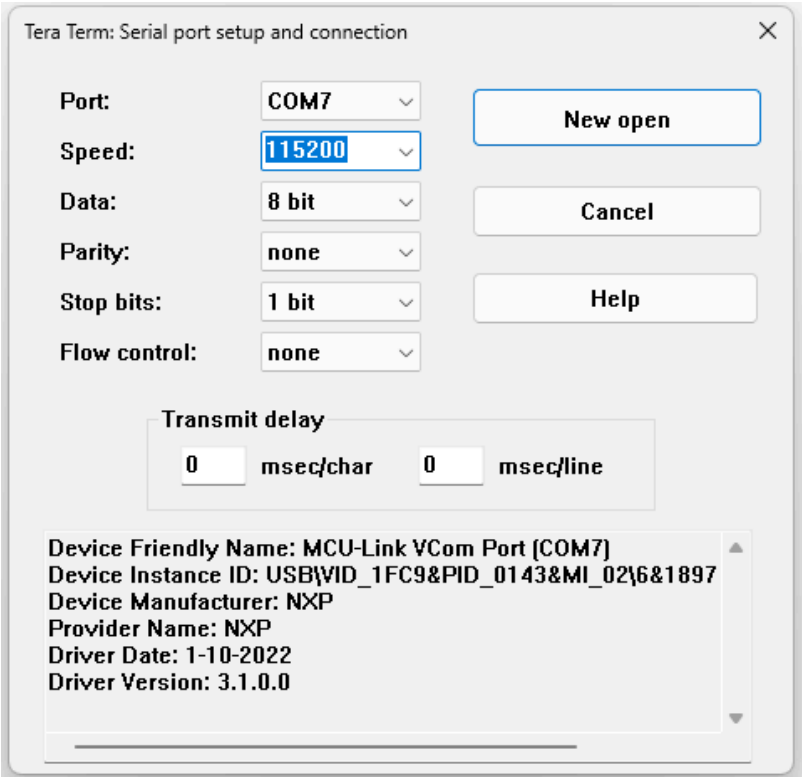

Figure 9. Serial Terminal settings

15. Run the code fully and click the Reset button.
16. Then, use the barcode scanner to scan a barcode, as shown in Figure 10.

Figure 10. Use a barcode scanner to scan the code

The barcode information prints on the Terminal screen, as shown in Figure 11.

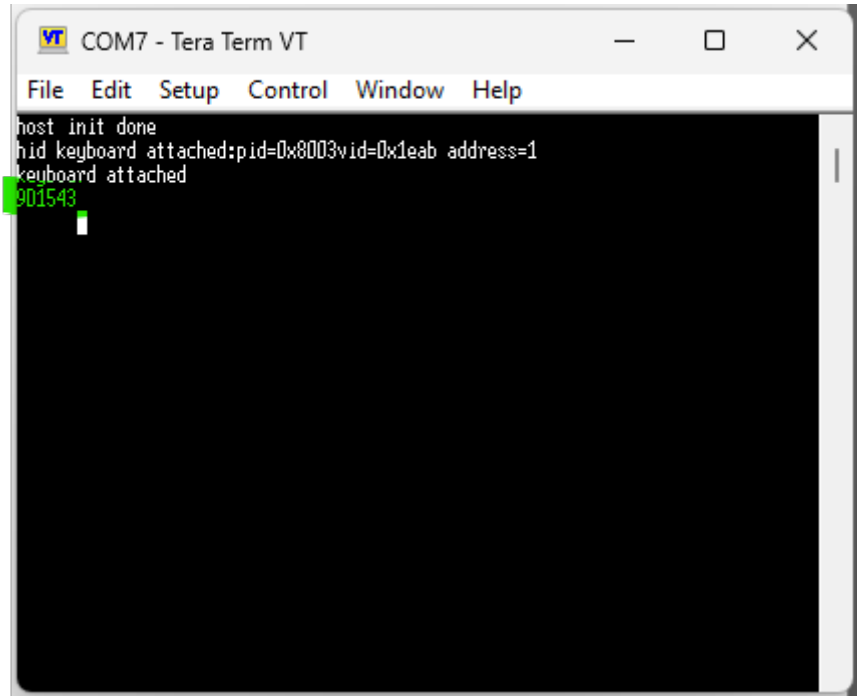

Figure 11. Terminal information

5 Note about the source code in the document

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

Copyright 2025 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.

6 Revision history

Table 1 summarizes revisions to this document.

Table 1. Revision history

Document ID	Release date	Description
AN14543 v.1.0	22 January 2025	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.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

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.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

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