AN14005

I3C New Features vs I2C on i.MX93

Rev. 1 — 24 August 2023 **Application note**

Document Information

Information	Content
Keywords	AN14005, i.MX 93 EVK, I3C, I2C
Abstract	This application note describes how to use various I3C functions in Linux on i.MX 93 EVK, such as performing Dynamic Address assignment (DAA), validating hot-join functionality, basic communication tasks, and IBI handling.

1 Introduction

Improved Inter-Integrated Circuit (I3C) is intended to improve upon the features of the I2C interface, while preserving backward compatibility.

The new features of the MIPI I3C interface over I2C includes:

- Faster communication speed (up to 12.5 MHz) and optional HDR mode.
- Dynamic Addressing (DAA) and priority arbitration.
- In-Band Interrupt (IBI) eliminates the need of sideband interrupt signals.
- Bus management CCC commands.
- Hot-join on to I3C bus, allowing targets to join the bus after controller initialization.
- Compatible with legacy I2C, but clock stretching is not supported.
- Lower power consumption since it works in push-pull mode most of the time.

This application note describes how to use various I3C functions in Linux on i.MX 93 EVK, such as performing Dynamic Address assignment (DAA), validating hot-join functionality, basic communication tasks, and IBI handling.

Note: *The terminology in this application note is updated to align with MIPI I3C Specification v1.1.1.*

Table 1. Updated terminology

Updated term	Deprecated term
Controller	Master
Target	Slave

This document provides an introduction to features provided by the i.MX 93 I3C controller. It introduces the I3C subsystem API provided by Linux and shows how it is used in the LSM6DSOX driver. Topics such as device initialization, DAA, hot-join functionality, and reading of sensor data are covered. It provides an example of IBI handling using registered callbacks with the Linux I3C API.

The software package is tested on the i.MX93 EVK with an on-board LSM6DSOX 6-axis IMU connected through an I3C bus. The Linux release used in this document is LF5.15.71_2.2.0.

2 I3C support in Linux

2.1 I3C controller

Two instances of I3C controllers are present in the i.MX 93 application processor (I3C1 in AONMIX and I3C2 in WAKEUPMIX). Both support all required and most optional features of the MIPI Alliance Specification of I3C, v1.0 and v1.1, except for ternary data rates (HDR-TSP and HDR-TSL) and peer-to-peer messaging. It also provides compatibility to an extended I2C 10-bit address. For more details, see the i.MX 93 Applications Processor Reference Manual (document IMX93RM).

The I3C controller in i.MX 93 is supported by the "svc-i3c-master" driver in the Linux kernel. It supports CCC command, hot-join, DAA, IBI handling, and data read/write. It emulates an I2C bus device for an I2C compatible device.

2.2 Linux I3C device

I3C device drivers are implemented using the Linux I3C device driver API. In general, the driver implements the "i3c_driver" structure with all required fields, such as "device driver", "probe", "remove", and "id_table",

containing all the 48-bit provisional IDs of devices that this driver is compatible with. This is used to match the driver during the hot-join phase.

LSM6DSOX is a 6-axis IMU supporting the I3C bus. A Linux driver utilizing the IIO framework is provided to support this device. See STMicroelectronics/st-mems-android-linux-drivers-iio for details. This application note enables this driver and patches it to illustrate the IBI function.

Note: *An older version of the LSM6DSX series driver is included in Linux kernel release LF5.15.71. This application note disables the in-tree driver and adds the new version.*

3 Board preparation

The LSM6DSOX IMU on the i.MX 93 EVK works in the I2C/I3C coexistence mode by default. Rework is needed to switch to the I3C-only mode. The I3C-associated level shifter must be disabled. The DTB file must be replaced to verify the hot-join feature.

3.1 Board rework

By default, the LSM6DSOX works in the I2C/I3C coexistence mode, which is incompatible with the implementation of the Linux driver. The Linux driver probes the I3C device in high-speed (12.5 Mbit/s) communication, where LXM6DSOX expects low-speed (under 400 kbit/s) I2C communication in the beginning. To force LSM6DSOX to run in I3C-only mode, add a 10-kΩ pull-up resistor to solder pad R1010.

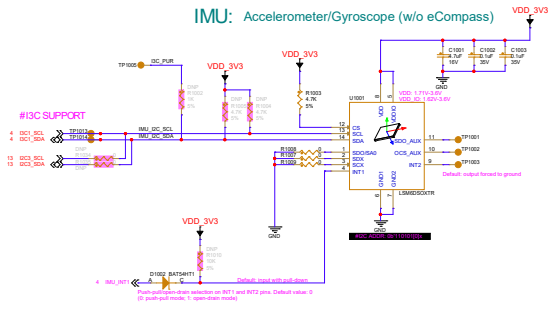

Figure 1. Schematics of LSM6DSOX IMU

3.2 Image build

Build the i.MX93 Yocto image from source following the instruction in the i.MX Yocto Project User Guide (document IMXLXYOCTOUG):

```
$ mkdir imx-yocto-bsp
$ cd imx-yocto-bsp
$ repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-kirkstone -m
 imx-5.15.71-2.2.0.xml
$ repo sync
$ MACHINE=imx93evk DISTRO=fsl-imx-xwayland source ./imx-setup-release.sh -b
 b_imx93-i3c-an
$ bitbake imx-image-core
```

Wait until the build is complete. Find the "imx-boot and imx-image-core-imx93evk.wic.zst" image file in the "tmp/deploy/images/imx93evk" folder.

3.3 Boot on A1 silicon

This section is only required for the i.MX 93 A1 silicon. LF5.15.71 BSP supports the A0 silicon by default. To boot on the A1 silicon, the "imx-boot" image must be updated. Download the i.MX 93 EVK 6.1.36-2.1.0 release image from the NXP web site (https://www.nxp.com/design/software/embedded-software/i-mx-software/embedded-linux-for-i-mx-applications-processors:IMXLINUX). Take the "imx-boot-imx93-11x11-lpddr4x-evk-sd.bin-flash_singleboot" image file and use it as the "imx-boot" image in the following sections.

3.4 DTB replacement

Follow the instruction in the i.MX Linux User Guide (document IMXLUG) to flash the boot media with UUU:

```
$ uuu -b sd_all imx-boot imx-image-core-imx93evk.wic.zst
```

The release image does not enable I3C by default, but includes a DTB with I3C enabled for replacement. Stop the boot in U-Boot during the boot and enter the U-Boot console. Replace the DTB file by running the following command:

```
$ u-boot=> setenv fdtfile imx93-11x11-evk-i3c.dtb
$ u-boot=> saveenv
```

Note: *This should be done only once, and U-Boot saves the environment to use the I3C DTB file by default.*

3.5 Disabling level shifter

This step is optional on new boards. Boards earlier than SCH-51961 REV B (including SCH-51961 REV B) require to disable the I2C level shifter, or the I3C probe might fail. To do this, run the following command in the U-Boot console:

```
$ u-boot=> i2c dev 1
$ u-boot=> i2c mw 0x25 0x0a 0x0 1
$ u-boot=> boot
```

Note: *If required, this must be done on every boot.*

3.6 Verify LSM6DSOX working

Check the IIO device in "sysfs" to verify that the LSM6DSOX works:

```
$ ls /sys/bus/iio/devices
iio:device0 iio:device1 iio:device2
```

`iio:device1` represents the gyroscope of LSM6DSOX and `iio:device2` is the accelerator. This can be verified by printing their names, respectively:

```
$ cat /sys/bus/iio/devices/iio\:device1/name
lsm6dso_gyro
$ cat /sys/bus/iio/devices/iio\:device2/name
lsm6dso_accel
```

4 I3C sensor demo

4.1 Update IMU device driver

Clone the Linux kernel source tree and checkout "L5.15.71". Create a new branch called “imx93-i3c-an”:

```
$ git clone https://github.com/nxp-imx/linux-imx
$ cd linux-imx
$ git checkout lf-5.15.71-2.2.0
$ cd linux-imx
$ git switch -c imx93-i3c-an
```

The built-in driver for LSM6DSOX is outdated. Pull and add a new driver from the STmems repository:

```
$ git remote add stmems_iio_github \
https://github.com/STMicroelectronics/st-mems-android-linux-drivers-iio.git
$ git fetch stmems_iio_github
$ git branch patch-branch ad72495db29ebdee00d973f158197e6b8c5d9b2c
(or use a shorter hash: $ git branch patch-branch ad72495db29ebdee)
$ git merge --allow-unrelated-histories \
imx93-i3c-an \
patch-branch
$ git am stm_iio_patches/5.15.y/*-stm-*.patch
```

Enable the new driver in KConfig:

```
$ make menuconfig
```

In “Device Drivers ---> Industrial I/O support ---> Inertial measurement units”, remove the selection of “ST_LSM6DSx driver for STM 6-axis IMU MEMS sensors”.

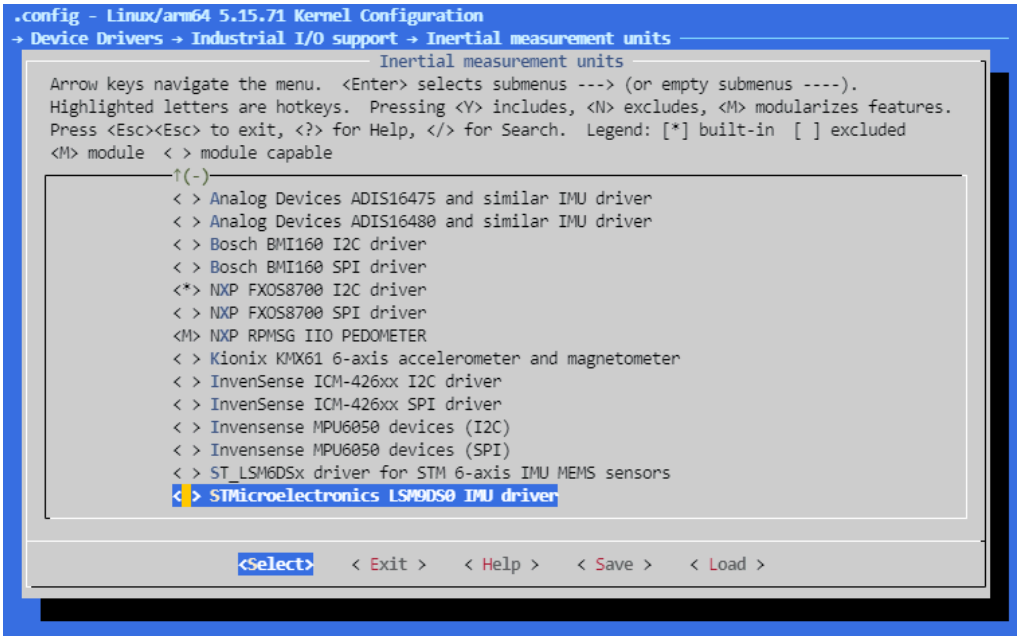

Figure 2. Disabling in-tree LSM6DSO driver

In “Device Drivers ---> Industrial I/O support ---> STM MEMS Device Drivers ---> Inertial measurement units”, select the “STMicroelectronics LSM6DSOX sensor”. Enable “Enable async hw timestamp read” as well.

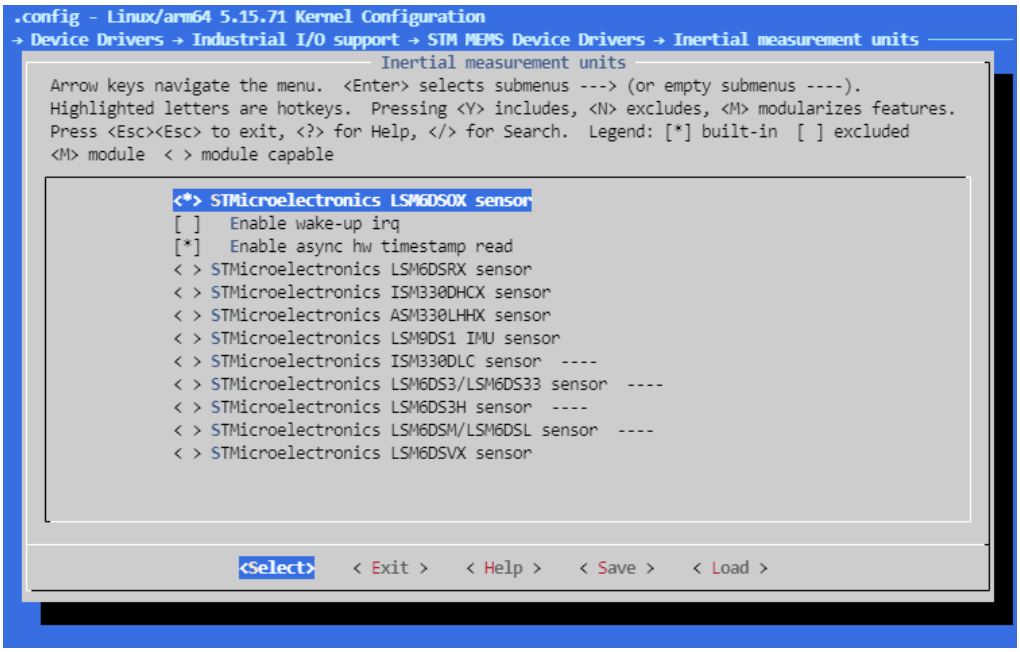

Figure 3. Enabling new LSM6DSOX driver

Build the kernel for i.MX 93. Copy the new kernel onto the boot VFAT partition, replacing the old kernel. Boot the board. Check IIO devices in "sysfs".

```
$ ls /sys/bus/iio/devices
iio:device0 iio:device2 iio:device4 iio:device6 iio:device8
iio:device1 iio:device3 iio:device5 iio:device7
```

A total of 9 devices are present. Devices 1 and 2 are still gyroscopes and accelerators, devices 3 to 8 are various “Embedded Functions” provided by the LSM6DSOX IMU sensor, such as pedometer, tilt detection, and so on. In the latter part of this AN, the tilt-detection function is used to illustrate the IBI function of the I3C bus.

4.2 Hot join and DAA

Because the LSM6DSOX sensor is built into i.MX 93 EVK base board, it is impossible to actually plug in the device after the board boot. Instead of getting device information from devicetree, the I3C device probe process has already utilized the hot-join functionality. The IMU device could be probed even if its devicetree node is deleted from devicetree.

Apply "0001-imx93-i3c-an-remove-dts-node-of-lsm6dsox.patch" to delete the lsm6dso_i3c node inside "imx93-11x11-evk-i3c.dts":

```
$ git am 0001-imx93-i3c-an-remove-dts-node-of-lsm6dsox.patch
```

Recompile the kernel and replace the new DTB file. Boot the board, then run:

```
$ dmesg | grep lsm6dsox
[    1.488957] st_lsm6dsox_i3c 0-208006c100b: supply vdd not found, using dummy
 regulator
[    1.497030] st_lsm6dsox_i3c 0-208006c100b: supply vddio not found, using
 dummy regulator
[    1.505191] st_lsm6dsox_i3c 0-208006c100b: Freq Fine 75 (ts 24925)
```

```
[    1.628324] st_lsm6dsox_i3c 0-208006c100b: mounting matrix not found: using
 identity...
[    2.841915] st_lsm6dsox_i3c 0-208006c100b: invalid file format for device
[    2.848781] st_lsm6dsox_i3c 0-208006c100b: Device probed
```

The LSM6DSOX still probed successfully.

The I3C controller inside i.MX 93 is set to do DAA after it is initialized. During this process, The I3C controller broadcasts the ENTDAA CCC command and I3C target devices who haven't already got the dynamic address (in this case, the LSM6DSOX IMU) drive their own 48-bit provisional ID on the I3C bus. The target device that wins the arbitration gets the dynamic address assigned by the I3C controller. In the Linux kernel, the I3C controller driver searches for a compatible driver according to the 48-bit provisional ID it just got from the bus. So even without a device tree node, the LSM6DSOX probed successfully.

The hot-join process happens when a new I3C target device joins the bus. It will drive "START", put a reserved target address of "7'b0000_010" following a "W" (write) bit on the bus as an IBI (In-Band-Interrupt, more on that later) on the bus, using "W" (write) after the "START". The I3C controller could reply with ACK and send the ENTDAA CCC command and assign a dynamic address similar to what it did after the initialization. This enables the I3C bus to act as a zero-configuration, USB-like, plug-and-play peripheral bus.

4.3 Read and write data

There are currently no user-space tools similar to "i2c-tools" to perform read/write on the I3C bus. The LSM6DSOX data can be accessed via the IIO driver. For example:

```
$ cat /sys/bus/iio/devices/iio\:device1/in_anglvel_x_raw
29
$ cat /sys/bus/iio/devices/iio\:device2/in_accel_z_raw
16644
```

This example reads a raw value of the x-axis of a gyroscope and a raw value of the Z axis of the accelerator. The LSM6DSOX driver also provides the "debugFS" symbol to access the register directly:

```
$ echo 0x0f > /sys/kernel/debug/iio/iio\:device2/direct_reg_access
$ cat /sys/kernel/debug/iio/iio\:device2/direct_reg_access
0x1c
```

This command reads the "WHO_AM_I" register of LSM6DSOX.

The low-level register operation is handled by a regmap I3C instance. The structure and API are similar to a regmap I2C instance.

4.4 IBI handling

4.4.1 Introduction to IBI

One of the most significant improvements of I3C over I2C is the ability to handle In-Band Interrupts (IBI). For I2C devices, it is typical to have a separate INT pin (sideband signals) connected to GPIO pins of the main processor. Not only does it occupy precious GPIO pads of the processor, it also adds extra complexity to the board layout.

In-band Interrupt (IBI) enables I3C targets to "notify" the I3C controller in the first place. An I3C target device starts the IBI by pulling the SDA line LOW. The I3C controller detects this transition and the SCL, forming a START condition. The I3C target device then drives its dynamic address on the bus followed by an R (read) bit. If the IBI request is accepted by the I3C controller, the I3C controller would drive the ACK and receive the mandatory data byte from the target.

Note: *If more than one target sends an IBI request at the same time, an arbitration takes place and the target with a lower dynamic address wins. Arrange the dynamic address according to the desired IBI request priority.*

4.4.2 IBI handling in Linux kernel

In the Linux kernel, the IBI request handler is registered during device initialization and called by the I3C controller driver once an IBI request arrives. The following object describes an IBI request handler function:

```
struct i3c_ibi_setup
{
    unsigned int max_payload_len;
    unsigned int num_slots;
    void (*handler)(struct i3c_device *dev,
    const struct i3c_ibi_payload *payload);
};
```

And the following API is provided to manage IBI handlers:

```
int i3c_device_request_ibi(struct i3c_device *dev,
              const struct i3c_ibi_setup *setup);
void i3c_device_free_ibi(struct i3c_device *dev);
int i3c_device_enable_ibi(struct i3c_device *dev);
int i3c_device_disable_ibi(struct i3c_device *dev);
```

These definitions are in “include/linux/i3c/device.h” under the Linux kernel tree.

4.4.3 Running the IBI demo

This demo utilizes the “Tilt Detection EmbFunction” of LSM6DSOX. The IMU generates an IBI request if the overall direction of the board changes. For example, rotating the board from horizontal position to vertical position (just as you would normally do to change the display orientation on a smart tablet) triggers an IBI.

IBI is not supported by the stock LSM6DSOX driver, which takes the IRQ number as a parameter during probe. This is common when implementing traditional GPIO-based sideband interrupt. The I3C IBI does not use an IRQ number, but binds to an I3C device object to identify an interrupt request. Therefore, a special hack is used in this application note, forcing IRQ = 255 as an identifier inside the LSM6DSOX driver if IBI is enabled. This is just a temporary approach for demo purpose.

It is worthy to mention that the IBI handler is called in the lower half of the I3C controller IRQ. The upper half is not available for IBI interrupts. If your application requires special handling in the upper half of the interrupt, IBI is not suitable.

Apply `0002-imx93-i3c-an-enable-ibi-support-for-lsm6dsox.patch` on top of the source tree.

```
$ git am 0002-imx93-i3c-an-enable-ibi-support-for-lsm6dsox.patch
```

Compile the kernel and copy it to the board. Connect it to the serial console and boot the system. Run the following command to verify IBI is enabled:

```
$ dmesg | grep lsm6dsox
[    1.490292] st_lsm6dsox_i3c 0-208006c100b: of_node = 00000000, ibi_enable = 0
[    1.497491] st_lsm6dsox_i3c 0-208006c100b: supply vdd not found, using dummy
 regulator
[    1.505492] st_lsm6dsox_i3c 0-208006c100b: supply vddio not found, using
 dummy regulator
[    1.513649] st_lsm6dsox_i3c 0-208006c100b: Freq Fine 75 (ts 24925)
[    1.636061] st_lsm6dsox_i3c 0-208006c100b: get int reg ibi
```

```
[    1.641614] st_lsm6dsox_i3c 0-208006c100b: mounting matrix not found: using
 identity...
[    2.854092] st_lsm6dsox_i3c 0-208006c100b: invalid file format for device
[    2.861189] st_lsm6dsox_i3c 0-208006c100b: probe enable ibi
[    2.866773] st_lsm6dsox_i3c 0-208006c100b: probe ibi enabled
```

Rotate the board and a message shows up in the console indicating that a tilt event is detected.

The result is as follows:

```
[  123.513944] st_lsm6dsox_i3c 0-208006c100b: tilt detection
[  123.529684] st_lsm6dsox_i3c 0-208006c100b: tilt detection
[  124.317020] st_lsm6dsox_i3c 0-208006c100b: tilt detection
[  125.312073] st_lsm6dsox_i3c 0-208006c100b: tilt detection
```

5 References

- i.MX 93 Applications Processor Reference Manual (document IMX93RM)
- Building an I3C Sensor Network Using LPC55S3x (document AN13577)
- MIPI I3C (Sensor Specification) Informational Whitepaper
- MIPI Alliance Specification for I3C

6 Conclusion

This application note shows the advantages and provides a demo of using I3C in the Linux kernel. The I3C controller on the i.MX 93 application processor can connect to various I3C compatible devices with a ready-to-use Linux driver.

The I3C is superior to I2C in various ways. A higher clock speed and optional HDR modes cover the increasing bandwidth demand of inter-chip communication. The dynamic address assignment and CCC commands provide extra bus-management features. IBI eliminates the need of sideband interrupt pins and the hot-join feature provides USB-like plug-and-play experience.

7 Note about the source code in the document

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

Copyright 2023 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.

8 Revision history

Table 2 summarizes the changes done to this document.

Table 2. Revision history

Revision number	Release date	Description
1	24 August 2023	Initial external release

9 Legal information

9.1 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.

9.2 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.

9.3 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.

i.MX — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

Contents

Date of release: 24 August 2023
Document identifier: AN14005