AN12604

Implement Second Bootloader on i.MX RT10xx Series

Rev. 1 — 19 September 2023 Application note

Document Information

Information	Content
Keywords	i.MX RT1050, i.MX RT1020, i.MX RT1060, XIP flash, second bootloader.
Abstract	This document intends to introduce one example to illustrate how to implement the second bootloader on i.MX RT1050, i.MX RT1020, and i.MX RT1060 part based on XIP flash.

1 Introduction

Although i.MX RT series already supports ROM boot by UART or USB to upgrade the firmware, but still there is a customized requirement to upgrade firmware in On-The-Air(OTA) or other use case. It requires to implement the second bootloader instead of the ROM bootloader. This document intends to introduce one example to illustrate how to implement the second bootloader on i.MX RT1050, i.MX RT1020, and i.MX RT1060 part based on XIP flash.

Note: *This application note was written using legacy tools and flow. The MCUXpresso Secure Provisioning (SEC) Tool and SPSDK are the latest tools. The information in this application note describing the secure flow within the chip still applies, but we recommend using the latest tools (MCUXpresso SEC or SPSDK) instead of following the steps shown here. For any questions, please contact your local support.*

2 Overview

TheSecond bootloader manages the upgrade of the firmware. Generally it includes three parts, flash operation, communication, and security management. Second bootloader may be XIP(eXecute-In-Place) or NON-XIP depending upon customer application case and memory size on the system. As for NON-XIP, it means to run a bootloader in internal memory or external memory; it must consume memory space for bootloader that is a limitation for the case with limited memory space.

This document introduces one common example to implement the second bootloader based on XIP. It describes how to implement flash operation, communicate with the host device by UART and USB. It also presents an example to show how to implement security management, which program the encrypted image and boot with BEE configuration to implement on-the-fly.

See the system diagram in Figure 1 .

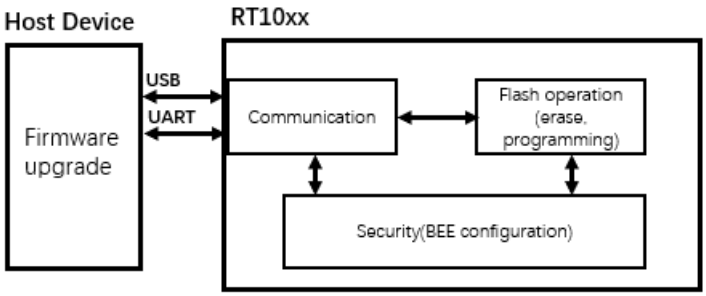

Figure 1. System diagram

3 Second bootloader implement

Here is an example to introduce how to implement a XIP second bootloader in i.MX RT1050. Refer to IMXRT1050 EVKB Board Hardware User's Guide to rework board boot by QSPI flash.

For software architecture, it contains three sessions, flash operation, communication, and BEE configuration. The boot flow is shown in Figure 2.

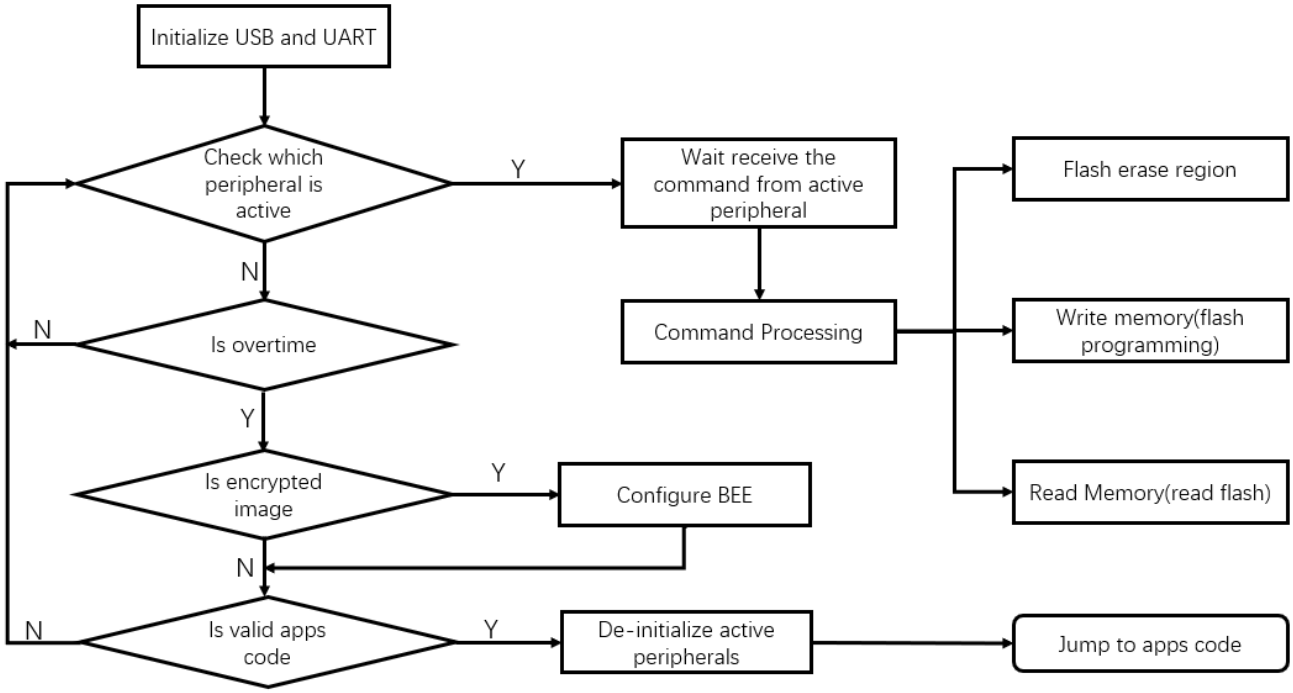

Figure 2. Boot Flow

After power-on, it first initializes the USB and LPUART1, and then checks, which peripheral is active. Meanwhile, detects if it is greater than the default TIMEOUT value(default is 10S for debug project, and 5S for release project). The user can also change this setting in the file “bootloader_config.h”, as shown in Figure 3.

```
#if DEBUG
#define BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT 10000
#else
#define BL_DEFAULT_PERIPHERAL_DETECT_TIMEOUT 5000
#endif // DEBUG

// The bootloader will check this address for the application vector tabl
#if !defined(BL_APP_VECTOR_TABLE_ADDRESS)
#define BL_APP_VECTOR_TABLE_ADDRESS (0x10000+0x60000000)
#endif
#define BL_PARAMETERS_ADDRESS         (0xF000+0x60000000)
#endif // __BOOTLOADER_CONFIG_H__

#define FLASH_VALID_START              BL_APP_VECTOR_TABLE_ADDRESS
#define FLASH_VALID_END                BL_FLEXSPI_AMBA_BASE+8*1024*1024
```

Figure 3. Bootloader default parameters

3.1 Flash operation

i.MX RT support FlexSPI to interface QSPI, Octal and hyper flash. Here we take QSPI as an example to show how to implement flash erasing and programming on XIP. As the ROM initializes FlexSPI and QSPI flash configuration on boot phase, it is not required to initialize FlexSPI once more. By default it do not support flash erasing and programming operations. It only initialize LUT to support fast read commands by ROM, so the user has to update LUT to support more commands as per customer requirements. To get stable operation, follow the below guidelines:

- Disable interrupt and prefetch buffer before doing any FlexSPI operation. To abort on-going prefetching, set the SWRESET bit in register MCR0 to enable FlexSPI software reset. For example, update LUT.
- Allocate all the code of operating FlexSPI to internal or external RAM, say updating LUT, flash erase and program API function.
- After flash erasing and programming complete, set SWRESET bit to enable FlexSPI software reset.

All the flash API functions can be found in file “flexspi_nor_flash_ops.c”, and for i.MX RT1060, also can call the ROM API to implement flash operation.

ROM provides the entry address(0x0020001c) for all API tree, it must initialize the API tree. An example is shown below:

g_bootloaderTree = (bootloader_api_entry_t *)*(uint32_t *)0x0020001c;

A snippet for the API entry is as below:

```
typedef struct
{
    const uint32_t version;                                 //!< Bootloader version number
    const char *copyright;                                  //!< Bootloader Copyright
    void (*runBootloader)(void *arg);                       //!< Function to start the bootloader executing
    const uint32_t *reserved0;                              //!< Reserved
    const flexspi_nor_driver_interface_t *flexSpiNorDriver; //!< FlexSPI NOR Flash API
    const uint32_t *reserved1;                              //!< Reserved
    const clock_driver_interface_t *clockDriver;
    const rtwdog_driver_interface_t *rtwdogDriver;
    const wdog_driver_interface_t *wdogDriver;
    const uint32_t *reserved2;
} bootloader_api_entry_t;
```

Figure 4. ROM API Entry

The user can call an appropriate API by this API entry. For details, refer to the attached example of “rt1060_rom_api.zip”.

3.2 Communication

This bootloader currently supports UART and USB for boot, default support LPUART1, and band rate is 115200 bps, available to use host tool “blhost.exe” to get image upgrade, to simplify the operation, remove some commands and supported commands listed below:

- get_property
- write memory
- flash erase region
- read memory

For detailed protocols introduction, refer to doc MCUX Flashloader Reference Manual and Kinetis blhost User's Guide.

Boot pins are as below:

Table 1. Boot Pins

Peripheral	Instance	IO Func	ALT	PAD
LPUART	1	lpuart1.TX	2	GPIO_AD_B0_12
		lpuart1.RX	2	GPIO_AD_B0_13
USB	1	USB_OTG1_DP	-	USB_OTG1_DP
		USB_OTG1_DN	-	USB_OTG1_DN

User may change the UART instance and pin mux as below.

- Navigate the file “peripherals_MIMXRT1052.C” to change the instance number.

```
//! @brief Peripheral array for MIMXRT1051.
const peripheral_descriptor_t g_peripherals[] = {
#if BL_CONFIG_LPUART_1
    // LPUART1
    {.typeMask = kPeripheralType_UART,
     .instance = 1,
     .pinmuxConfig = uart_pinmux_config,
     .controlInterface = &g_lpuartControlInterface,
     .byteInterface = &g_lpuartByteInterface,
     .packetInterface = &g_framingPacketInterface },
#endif // BL_CONFIG_LPUART_1
```

For example, available to change the LPUART instance to 2 or others.

Here “.pinmuxconfig” specify the pin mux configuration function for LPUART. To change pin mux settings, navigate this function to modify it.

3.3 Security management

The second bootloader support to upgrade the encrypted or raw image. To encrypt image by default it uses SW_GP2 as key, and decrypted on-the-fly by configure BEE region1, and reserve BEE region0 for second bootloader, available to select OTPMK/SNVS as bootloader key, or use the same key(SW_GP2) for second bootloader. The image layout is shown in Figure 5.

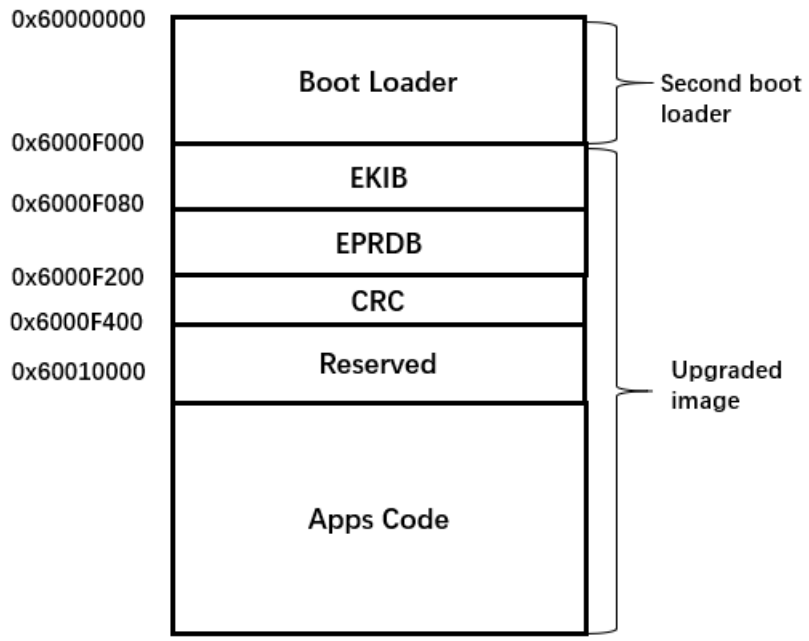

Figure 5. Layout View

It reserves the address space from 0x60000000 to 0x6000f000 for the second bootloader, and the address space from 0x6000f000 to 0x6000f400 is for configuration parameters of the encrypted image. This is similar with ROM EKIB and EPRDB, EKIB encrypted by SW_GP2 and EPRDB encrypted by KIB with AES128 CBC mode.

After detect timeout, the second bootloader tries to check if the image is valid or not, if the image is encrypted, try to decrypt EKIB and EPRDB, and then configure BEE region1 with a specified setting by PRDB, and reserve BEE region 0 for bootloader encryption.

Note: *Customer must burn the SW_GP2 and set fuse "BEE_KEY1_SEL" to "0b11", this is to enable BEE region 1 key from SW_GP2.*

4 Generate encrypted image

As the second bootloader supports to program the encrypted image and boot up, so it needs one tool to convert a plain image to an encrypted image. A simple diagram and image layout is below:

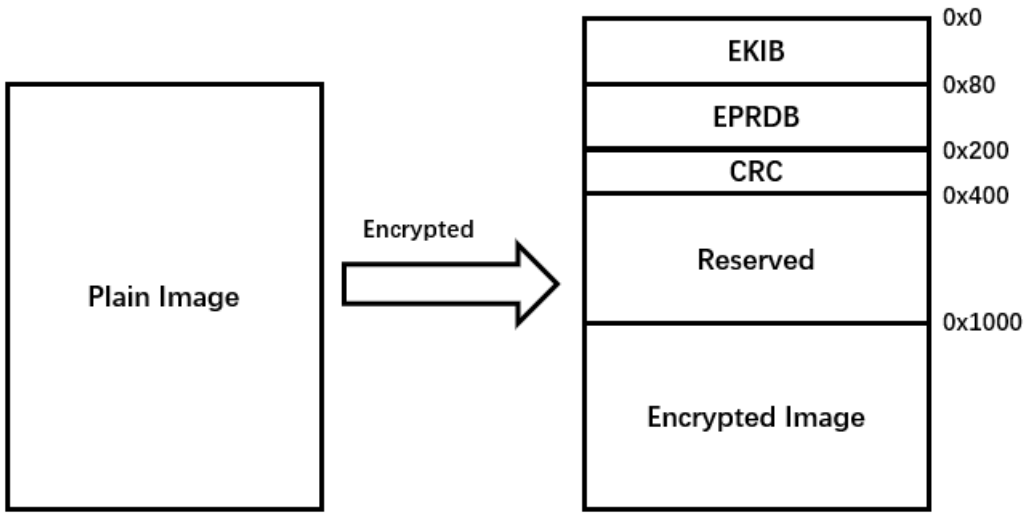

Figure 6. Image Encrypted

The description of the encrypted image layout is as below:

Table 2. Encryption image parameters layout

Offset	Description
0x0000 - 0x003	Tag, must equal to 0xEA5CA5A5
0x4 - 0x23	Encrypted Key Info Block(EKIB)
0x24 – 0x7f	Reserved
0x80 - 0x17F	Encrypted Protection Region Descriptor Block(EPRDB)
0x180 - 0xFFF	Reserved
0x200 – 0x400 CRC	Save image CRC value for check
0x400 - 0xFFF	Reserved

Key Info Block(KIB) consists of 128-bit AES Key and 128-bit Initial Vector.

Table 3. Key Info Block

Offset	Field	Description
0x00 - 0x0f	AES-128 KEY	128-bit AES KEY used for EPRDB decryption
0x10 - 0x1f	Initial Vector	128-bit Initial Vector used for EPRDB decryption

Protection Region Descriptor Block (PRDB) is a 128 byte crypto block using AES-ECB mode and the AES Key provisioned in eFUSE. The plaintext of PRDB is shown in Table 4.

Table 4. Protection Region Descriptor Block

Offset	Field	Description
0x000 - 0x003	tagl	Must equal to 0x5F474154 (ASCII: "TAG_")
0x004 - 0x007	tagh	Must equal to 0x52444845 (ASCII: "EHDR")

Table 4. Protection Region Descriptor Block*...continued*

<table>
<tr><th>Offset</th><th>Field</th><th>Description</th></tr>
<tr><td>0x008 - 0x00b</td><td>Version</td><td>[31:24] 'V'
[23:16] Major version
[15:08] Minor version
[07:00] Bug fix

Must equal to 0x56010000 for i.MX RT Family</td></tr>
<tr><td>0x00c - 0x00f</td><td>Reversed</td><td></td></tr>
<tr><td>0x010-0x04f</td><td>encrypt_region_info</td><td>Information for encrypted region

<table>
<tr><th>Offset</th><th>Field</th><th>Description</th></tr>
<tr><td>0x00 - 0x03</td><td>Start</td><td>The absolute start address of the encrypted region, must be 4 KB aligned</td></tr>
<tr><td>0x04 - 0x07</td><td>End</td><td>The absolute end address of the encrypted region, must be 4 KB aligned</td></tr>
<tr><td>0x08 - 0x0b</td><td>Mode</td><td>AES encryption mode
0 - AES-ECB (128)
1 - AES-CTR (128), it is the recommend mode for Encrypted XIP</td></tr>
<tr><td>0x0c - 0x0f</td><td>lock_option</td><td>Region Lock options
0 - All regions are unlocked
3 - BEE regions 1 are locked.</td></tr>
<tr><td>0x10 - 0x1f</td><td>Counter</td><td>AES-CTR Counter, Valid only if mode is 1
Only the upper 96 bits are configurable, the lower 32 bits must be 0.
Note:
During encryption and decryption, the nonce/is

<table>
<tr><td>127:32</td><td>31:28</td><td>27:0</td></tr>
<tr><td>Counter [127:32]</td><td>0</td><td>FAC Region Start [31:4]</td></tr>
</table>
</td></tr>
<tr><td>0x20-0x3f</td><td>Reserved</td><td>Reserved for future use.</td></tr>
</table>
</td></tr>
<tr><td>0x050-0xFF</td><td>Reserved</td><td>Reserved for future use</td></tr>
</table>

CRC block is to check the code integrity of programming to flash, the second bootloader makes CRC check before jumping to user code, if failed, will stay in the boot load image.

Table 5. CRC Block

Offset	Field	Description
0x00 - 0x04	Tag	Must equal to 0xccea5a5a
0x04 - 0x07	CRC value	Final CRC results

Table 5. CRC Block*...continued*

Offset	Field	Description
0x08 – 0x0b	Byte count	Number of bytes for CRC processed

The attached zip file with the document provides one tool “image_generate.exe”. To encrypt the image, detailed arguments introduction is described as below:

Table 6. Arguments View

ifile	Input file path	
ofile	Output file path	
base_addr	base address of input file	
region_key	hex string	
region_arg	[start, length, permission],…, [start, length, permission]	start, length, and permission are integer variables
region_lock	0/1/2	0/1/2
use_zero_key	0/1	0 - KIB and Nonce/Counter[1/2/3] are generated randomly 1 - KIB and Nonce/Counter[1,2,3] = 0
kib_key	hex string	
kib_iv	hex string	
region_iv	hex string	

Below is example to generate the encrypted image by the "image_generate.exe" tool:

```
image_generate.exe ifile=iled_blinky.bin ofile=iled_blinky_encrypt.bin
base_addr=0x60010000 region_key=00112233445566778899aabbccddeeff region_arg=1,
[0x60010000,0xF000,0]
```

5 Run bootloader

To program the encrypted image to flash, the second bootloader must first fuse SW_GP2 to i.MXRT10xx(part RT1020/RT1050/RT1060), and fuse “BEE_KEY1_SEL” to “0b11” to enable SW_GP2 as BEE region1 key selection.

Get the attached package, and then build the second bootloader to program the code to flash by IDE or other tools. Attached the code support a program to flash by IAR directly.

To update an image, follow the below steps:

1. Prepare one app image for upgrade, it must modify the linker file to get the interrupt vector address to be 0x60010000. Refer to the attached linker file, below is snippet linker file.

```
define symbol m_interrupts_start        = 0x60010000;
define symbol m_interrupts_end          = 0x600103FF;

define symbol m_text_start              = 0x60010400;
define symbol m_text_end                = 0x63FFFFFF;

define symbol m_data_start              = 0x20000000;
define symbol m_data_end                = 0x2001FFFF;
```

Also, remove the XIP information by " XIP_BOOT_HEADER_ENABLE = 0".

2. Program the second bootloader to flash by IDE or other tools.
3. Generate the encrypted image by the attached tool " image_generate.exe " with the command below:
   image_generate.exe ifile=iled_blinky.bin ofile=iled_blinky_encrypt.bin base_addr=0x60010000 region_key=00112233445566778899aabbccddeeff region_arg=1,[0x60010000,0xF000,0]
   This is optional, the user can also directly program the plain image to flash.
4. Plug in the USB cable, and run commands in sequence.
   USB command as below:
   - blhost.exe -u -- get-property 1
   - blhost.exe -u -- flash-erase-region 0x6000f000 0x10000
   - blhost.exe -u -- write-memory 0x6000f000 iled_blinky_encrypt.bin

   UART command as below:
   - blhost.exe -p COM7,115200 -- get-property 1
   - blhost.exe -p COM7,115200 -- flash-erase-region 0x6000f000 0x10000
   - blhost.exe -p COM7,115200 -- write-memory 0x6000f000 iled_blinky_encrypt.bin

   With the above commands, it can erase the specified flash space and program the image file to flash.
   Note: *For encrypted image, the start address of the image is 0x6000f000, for raw image, the start address is 0x60010000.*
   Also the user can check flash contents by the read-memory command. For detailed information on how to use blhost, refer to "blhost User's Guide.pdf".
5. After successfully write image to flash, reset MCU, and then it works, see the LED blinking, and print the message on the serial terminal.

6 Conclusion

This document introduces one way to implement the second bootloader based on XIP flash, which is helpful to implement the second bootloader based customized applications.

7 Revision history

Table 7 summarizes the revisions to this document.

Revision history

Revision number	Release date	Description
1	19 September 2023	The document is updated to correspond to the latest guidelines, Section 1 is updated.
0	06/2020	Initial public release

8 Legal information

8.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

8.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

i.MX — is a trademark of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 19 September 2023
Document identifier: AN12604