AN14699

Firmware Update via SPI Flash Using Secondary Bootloader

Rev. 1.0 — 13 June 2025 Application note

Document information

Information	Content
Keywords	AN14699, FRDM-K32L2A4S, firmware update, secondary bootloader, SPI flash, LPSPI, dual image update, CRC validation, flash protection, SRecord, MT25QL256ABA
Abstract	This application note describes a method for updating firmware via external SPI flash using a secondary bootloader on the FRDM-K32L2A4S development board. It extends the dual image update concept to support external SPI flash memory accessed through the LPSPI interface.

1 Introduction

Dual image update is an important feature for advanced bootloaders. It assures that at least one image is bootable and works properly at any time. If any accident happens, the bootloader detects the anomaly and uses the previous image as a bootable image.

On the other hand, if the firmware is bigger than half of the ROM, dual image update is impossible. This document expands the dual image update introduced in *Firmware Update Using Secondary Bootloader* (document AN13947) to support external SPI flash via LPSPI, using FRDM-K32L2A4S.

2 Memory allocation

Begin by defining the memory addresses allocated for the secondary bootloader and the application image. Figure 1 shows the memory allocation to update firmware safely by external flash.

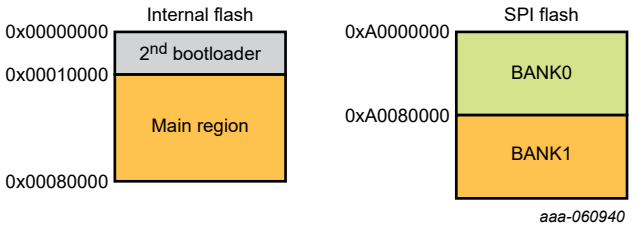

Figure 1. Memory allocation

The K32L2A41 has an internal 512 KB flash. Internal flash is used to store the second bootloader and a single bank for application image. Application image always runs in the internal flash because LPSPI does not support execute-in-place (XIP).

Let us assume that LPSPI helps access SPI flash (1 MB). The SPI flash contains two banks for storing application images. A new application image is programmed into one of these two banks.

MCUXpresso IDE customizes the memory map easily, as shown in Figure 2.

Note: *Also, customize the dimage.h accordingly.*

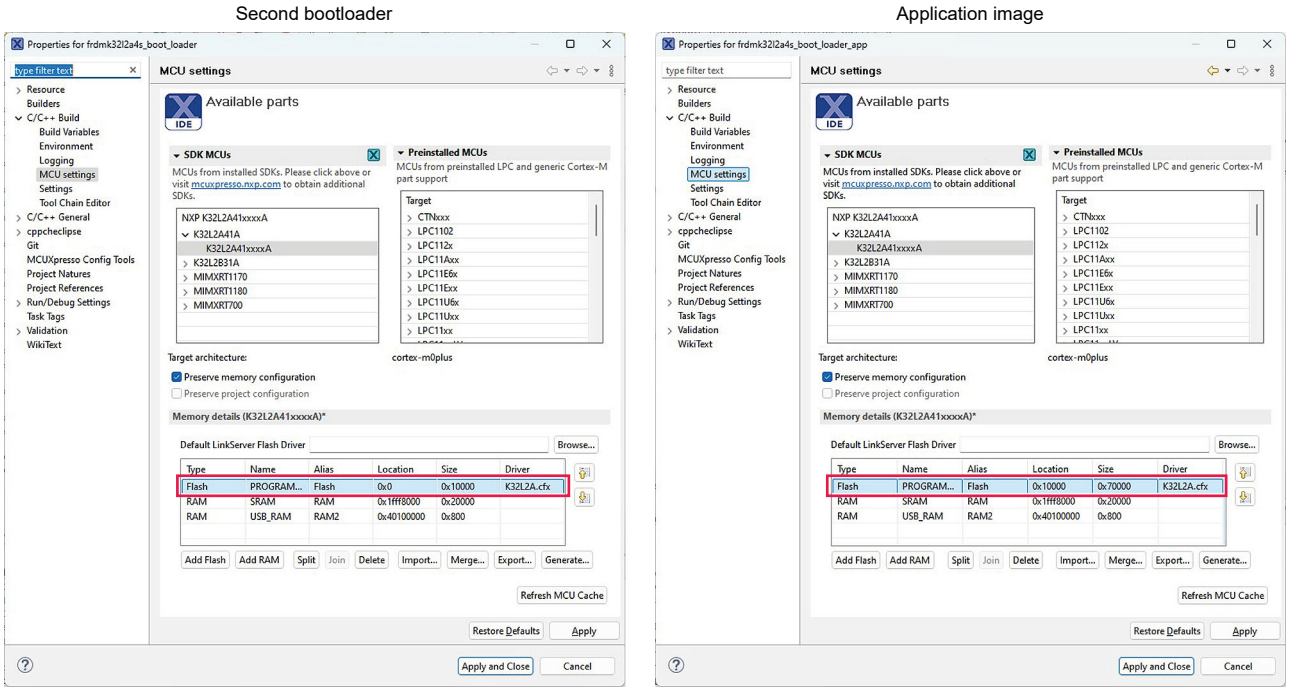

Figure 2. Memory map defined by MCUXpresso IDE

3 Boot sequence

To load and run the latest application image with the second bootloader, perform the steps below:

1. Scan the internal flash for a CRC-valid image.
2. Scan the SPI flash for a CRC-valid image.
3. Load the latest application image into the internal flash.
4. Boot the image in the internal flash.

Figure 3 shows a typical case the product life cycle. Ver.0 is programmed in the internal flash and later updated via external flash.

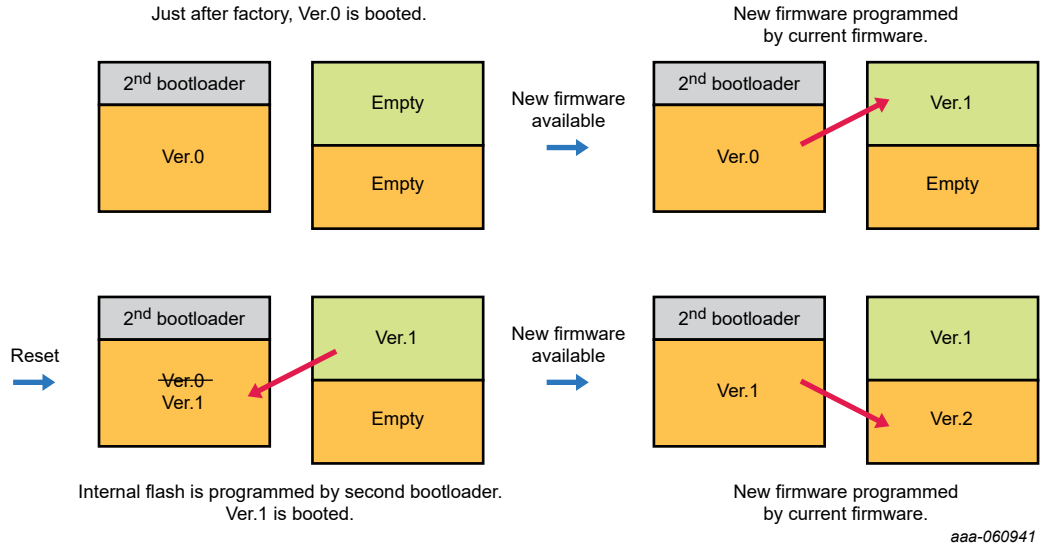

Figure 3. Typical case of a product life cycle

Important: *The two banks are used interchangeably to avoid bricking.*

Figure 4 shows the case when the internal flash has an invalid image, for example, MCU powered off during programming.

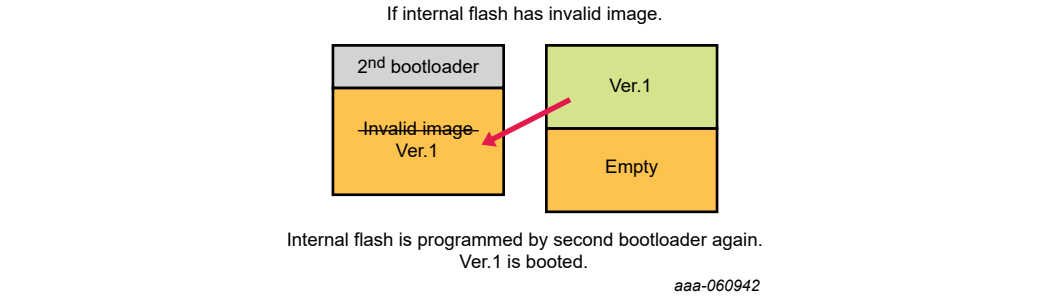

Figure 4. Boot sequence with invalid internal flash image

On the other hand, Figure 5 shows the case when the SPI flash has an invalid image, for example, a burst error during programming. In both cases, the system successfully loads and boots the valid image using the previously described procedure.

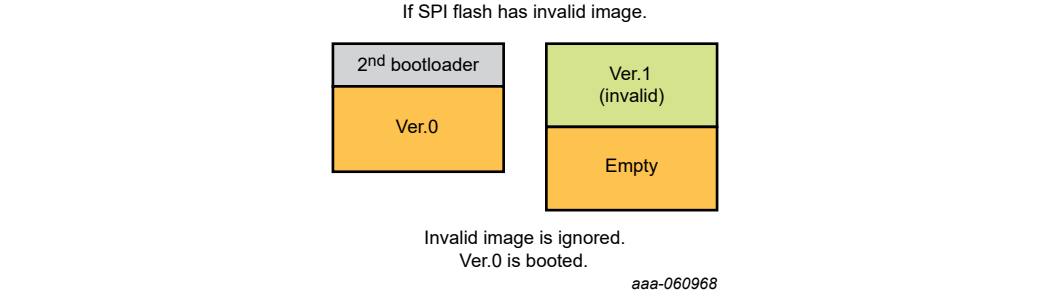

Figure 5. Boot sequence with invalid SPI flash image

4 Application image specification

Application image layout must follow a specific rule for the second bootloader to detect the image. This section explains image layout, image header structure, and how to embed CRC into the image.

4.1 Image layout

Figure 6 shows the application image layout, which is similar to the layout described in *Firmware Update Using Secondary Bootloader* (document AN13497). Application image must have an image header address at 0x28, which points to the image header.

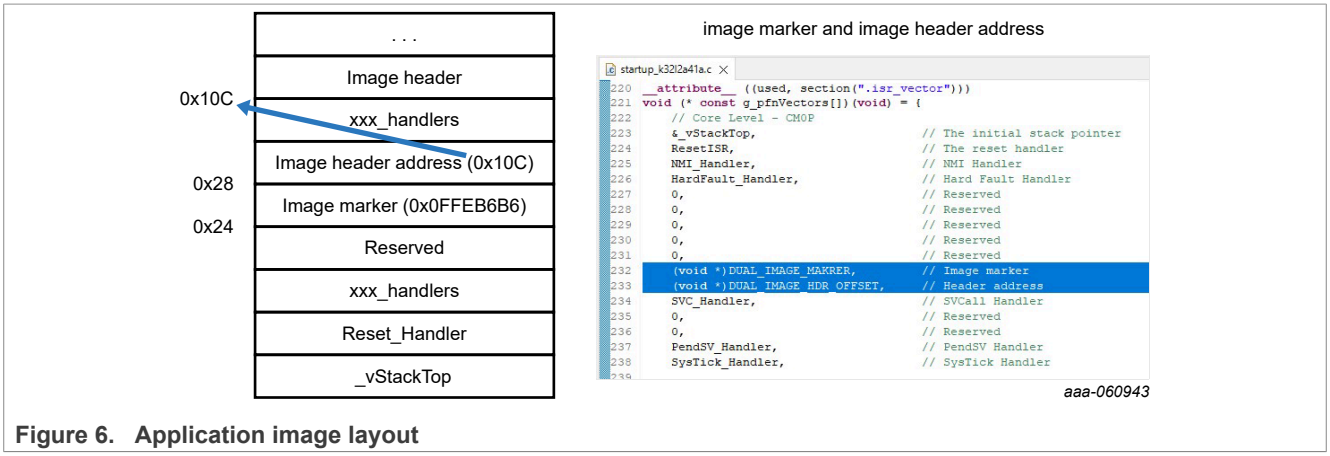

Figure 6. Application image layout

4.2 Image header structure

Table 1 shows the image header structure.

Table 1. Image header structure

Offset	Description
0x00	Header maker set to 0xFEEDA5A5
0x04	Image type (NORMAL = 0 or NO_CRC = 1)
0x08	Reserved
0x0C	Image length: The length must represent the actual value. If the CRC value field falls within the length, the total must be 4.
0x10	CRC value
0x14	Version

The image header is defined at the end of the vector table, as shown in the following snippet. SRecord embeds the image length and CRC value after built. NO_CRC can be used to bypass CRC check when the debugger is used.

```
__attribute__ ((used, section(".isr_vector")))
void (* const g_pfnVectors[])(void) = {
    // Core Level - CM0P
    &_vStackTop,                        // The initial stack pointer
    ResetISR,                           // The reset handler
….
    CMP1_IRQHandler,                    // 65: CMP1 interrupt (INTMUX source IRQ17)
    RTC_IRQHandler,                     // 66: RTC Alarm interrupt (INTMUX source IRQ18)
```

```
    (void *)HEADER_BLOCK_MARKER,        // Image header marker should always be set to
 0xFEEDA5A5
    (void *)0,                          // Image check type, with or without optional CRC
    0,                                  // Reserved word; must be 0
    0,                                  // Image length or the length of image CRC check
 should be done.
    0,                                  // CRC value
    (void *)0                           // Image version for multi-image support
}; /* End of g_pfnVectors */
```

4.3 CRC calculation by SRecord

SRecord is used to embed CRC (polynomial: 0xedb88320) and image length into the final image. In the sample project, "srec_cat @../srec/crc_cmd.txt" is automatically executed after built as shown in Figure 7. Ignore the warning, as the second bootloader disregards the hole (CRC value field position) in the same way that SRecord does.

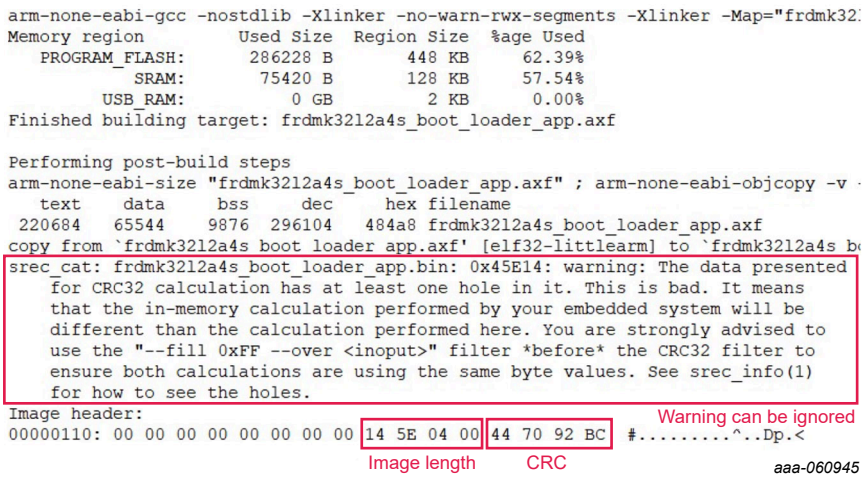

Figure 7. CRC calculation by SRecord in the post-build step

5 Memory specification

This section describes some important notation about internal flash and SPI flash in a hardware perspective.

5.1 Internal flash

The flash memory module includes a memory controller that executes commands to modify flash memory contents.

5.1.1 Parallel operation

Some operations cannot be executed simultaneously because the memories share certain hardware resources. Only the operations marked "OK" in Table 2 are permitted to run simultaneously on the program flash memories.

On the other hand, an interrupt is required to read external flash during programming the internal flash.

Therefore, interrupt must be disabled during flash operation to avoid vector fetch as shown in Figure 8, or you encounter HardFault.

Table 2. Allowed simultaneous memory operations

		Program Flash 0			Program Flash 1		
		Read	Program	Sector erase	Read	Program	Sector erase
Program Flash 0	Read	-			OK	OK	OK
	Program		-		OK		
	Sector Erase			-			
Program Flash 1	Read		OK	OK	-		
	Program	OK				-	
	Sector Erase	OK					-

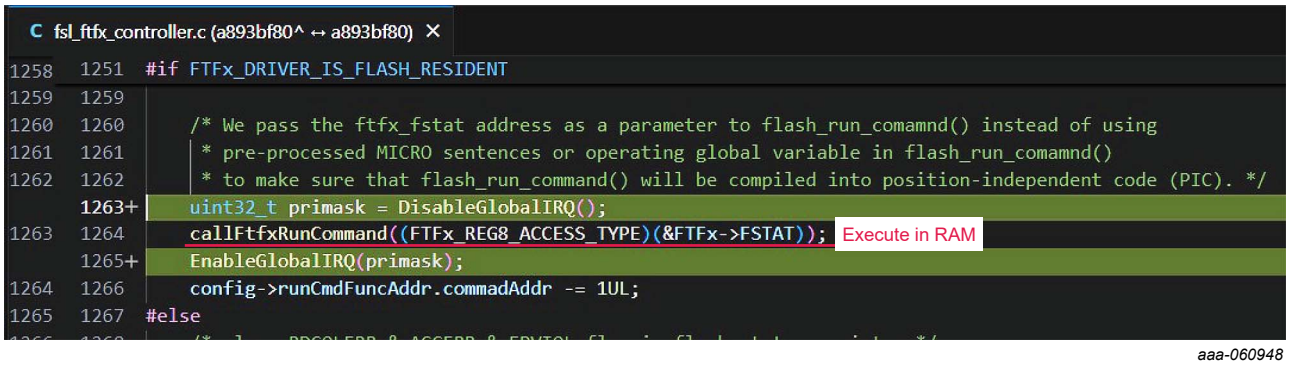

Figure 8. Mitigating HardFault: Disable interrupts during flash command execution

5.1.2 Flash protection

Individual regions within the flash memory can be protected from program and erase operations. The FPROTn registers control the protection. For 2^n program flash sizes, four registers typically protect 32 regions of the program flash memory, as shown in Figure 9.

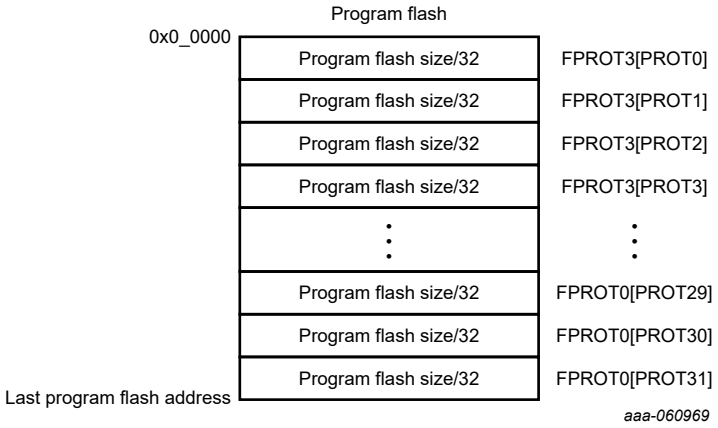

Figure 9. Flash protection regions

During the reset sequence, the FPROT registers are loaded with the contents of the program flash protection bytes in the flash configuration field as indicated in Table 3.

Table 3. Flash protection regions

Program flash protection register	Flash configuration field offset address
FPROT0	0x000B
FPROT1	0x000A
FPROT2	0x0009
FPROT3	0x0008

The `Flash_Config` in frdmk32l2a4s_boot_loader is used to protect the second bootloader as shown in Figure 10.

The `Flash_Config` is inherently protected as it resides within the first 64 KB of memory. If the flash configuration field is protected, the users cannot erase or reprogram the flash configuration field to change the protection settings. To ensure flexibility while maintaining security, enable mass erase when using flash protection. For more details, refer to *Using the Kinetis Security and Flash Protection Features* (document AN4507).

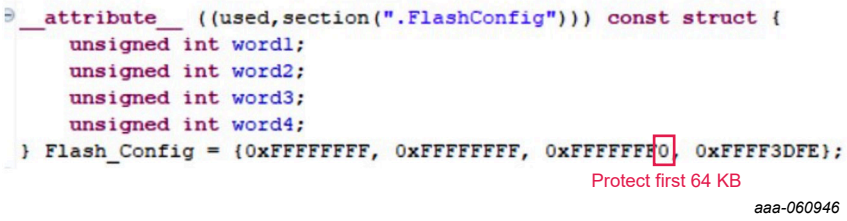

Figure 10. Flash protection by Flash_Config

The J-Link commander negates flash protection as shown in Figure 11. To streamline debugging, disable flash protection, as negating it via J-Link each time requires more time and effort.

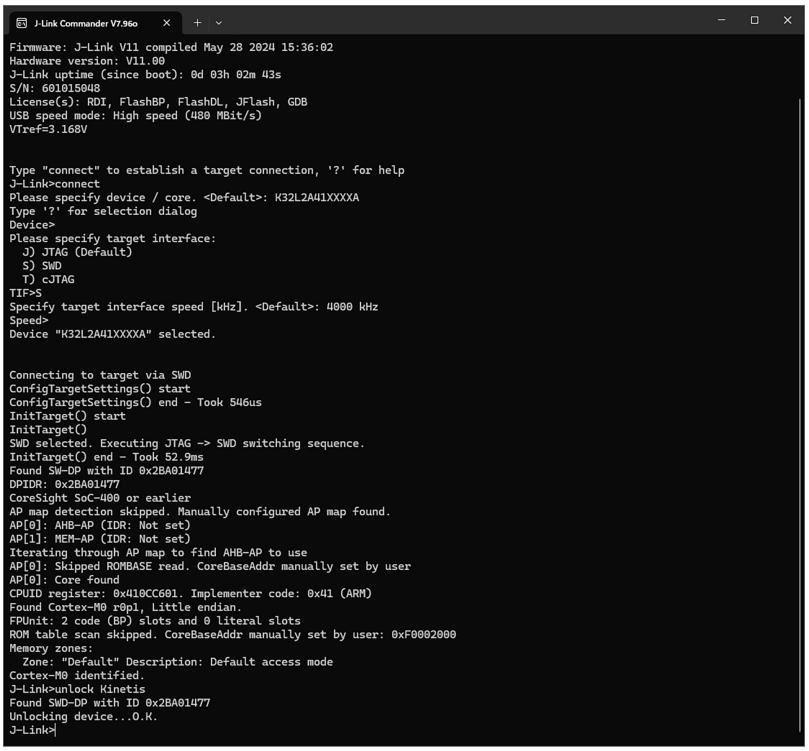

Figure 11. Disable flash protection by J-Link commander

5.2 SPI flash interface by LPSPI

LPSPI can be used as an interface of SPI flash.

5.2.1 Command sequence by software

To read/program/erase external SPI flash, follow the specified command sequence. For example, Figure 12 shows the read command by LPSPI. As LPSPI does not support an external memory interface by hardware, the software implements a command sequence. The command sequence must be common for most SPI flash. If needed, update fsl_lpspi_nor_flash.c/h.

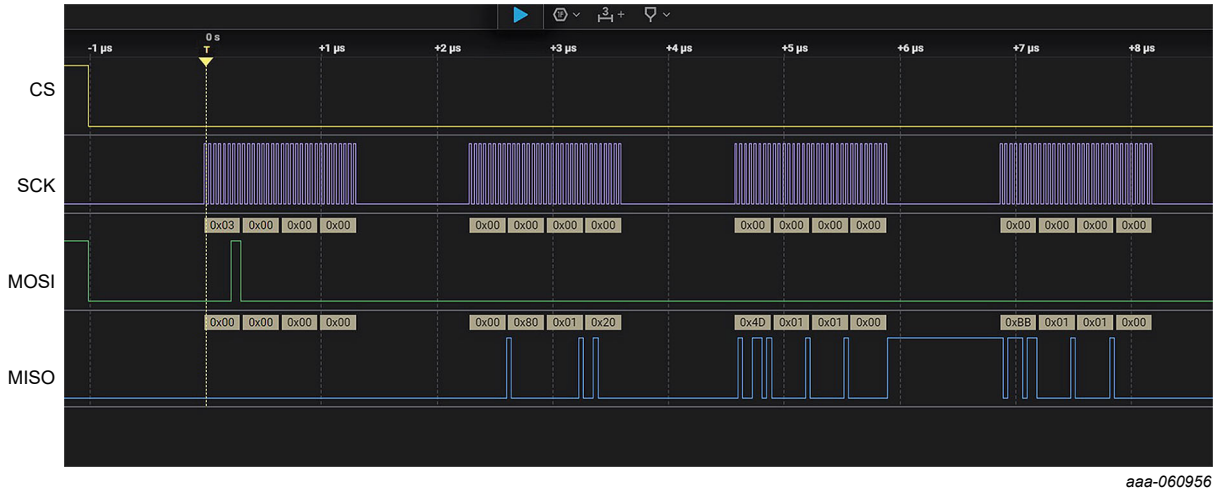

Figure 12. Read command by LPSPI

Figure 13 shows the write enable command. Due to DMA latency, the last falling edge in the 8-bit frame experiences a delay at a high baud rate. However, this delay has no impact on SPI communication.

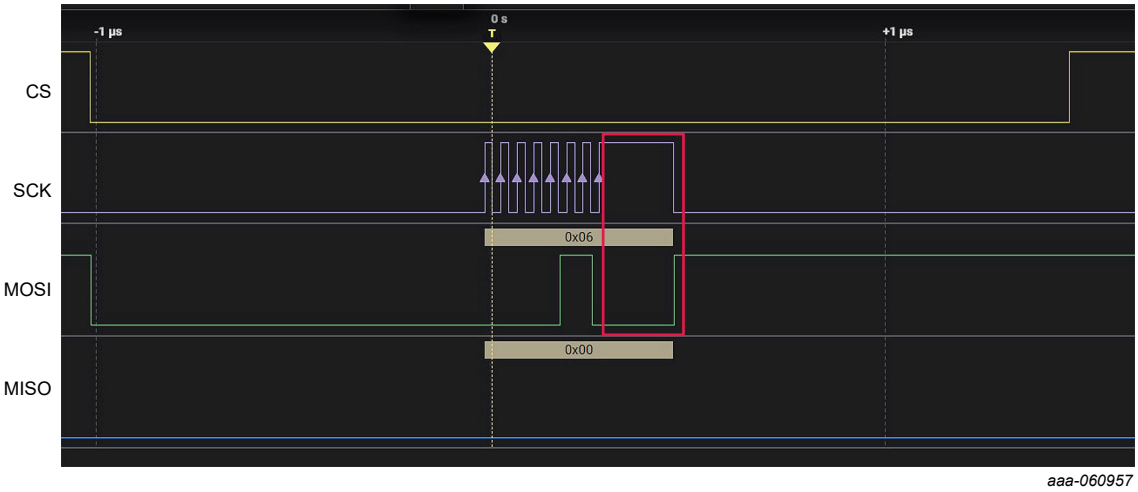

Figure 13. Write enable command by LPSPI

5.2.2 Pin assignment

To achieve a high bit rate:

- Use fast pins (PTE20, PTE21, PTE22, PTE23, PTD4, PTD5, PTD6, and PTD7)
- Use high drive pins (PTC3, PTC4, PTD4, PTD5, PTD6, and PTD7)

For more information about drive strength and slew rate, refer to *K32 L2A Microcontroller with 512 KB Flash and 128 KB SRAM* (document K32L2Ax).

6 Running the demo

Figure 14 shows the test case described in this section, which reproduces a typical product life cycle.

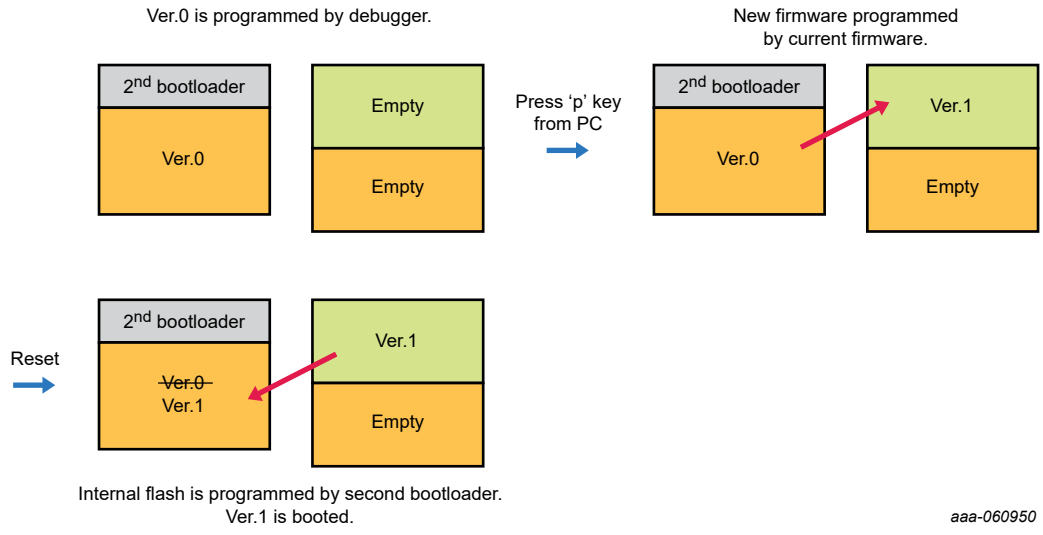

Figure 14. Test case

6.1 Overview

The demo is composed of the following two projects:

- Bootloader
- Application

The proprietary files for each projects are as follows:

- boot_loader: source/, dimage/
- boot_loader_app: source/, dimage/, startup/

Nothing is special except for them.

The test environment is as follows:

- FRDM-K32L2A4S
- MCUXpresso IDE v24.12.148
- MCUXpresso SDK v24.12.00
- Pmod SF3 (SPI NOR flash, MT25QL256ABA)
- SRecord 1.65
  Note: *Do not forget to download and paste bin directory under frdmk32l2a4s_boot_loader_app/srec as shown in Figure 15.*

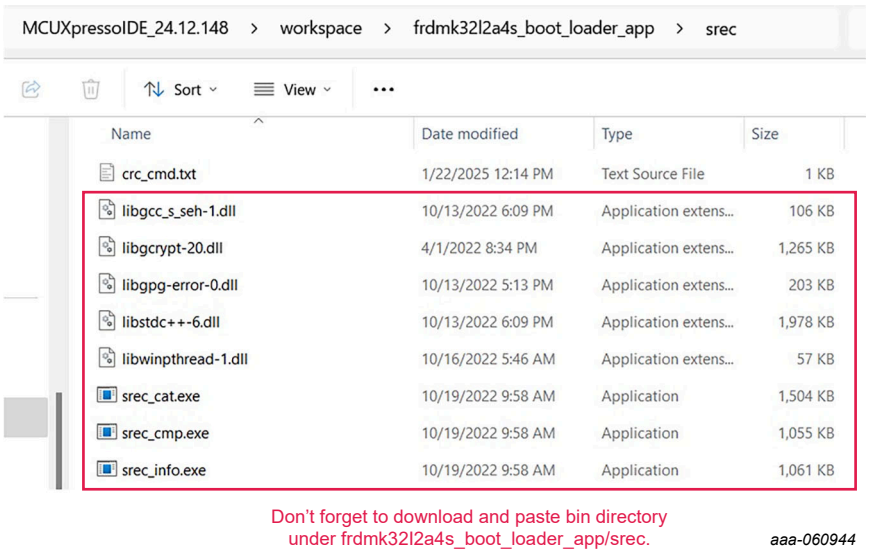

Figure 15. Download and paste bin of SRecord 1.65

6.2 Hardware setup

For hardware setup, perform the following steps:

1. Connect LPSPI master and Pmod SF3, as listed in Table 4.
2. Connect the board and PC via USB, as shown in Figure 16.

Table 4. Pin assignment

LPSPI master		SPI flash	
Function	**Location**	**Function**	**Location**
MISO	J4-5	MISO	J1-3
MOSI	J4-3	MOSI	J1-2

Table 4. Pin assignment*...continued*

LPSPI master		SPI flash	
Function	**Location**	**Function**	**Location**
SCK	J4-1	SCK	J1-4
PCS0	J4-7	PCS0	J1-1
GND	J3-14	GND	J1-5, J1-11
VCC	J3-4	VCC	J1-6, J1-12
VCC	J3-8	WP	J1-9
VCC	J3-8	HLD	J1-10

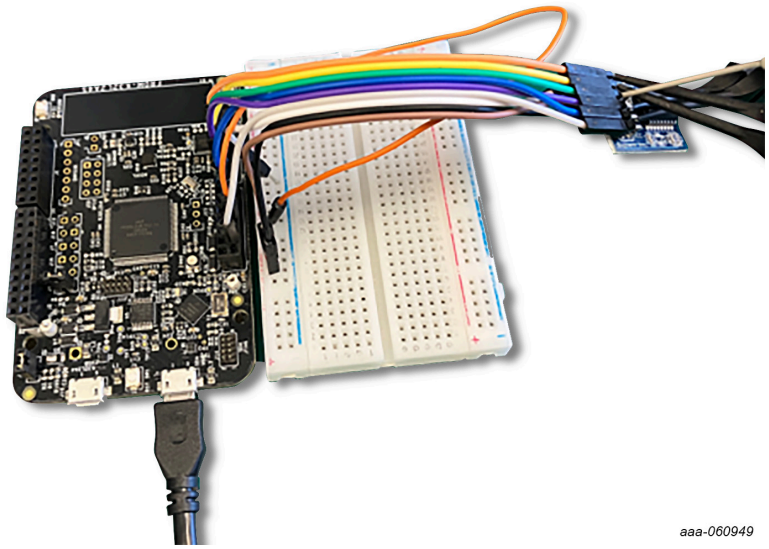

Figure 16. Hardware setup

6.3 Running the project

1. Open boot_loader and boot_loader_app projects from file system, as shown in Figure 17.

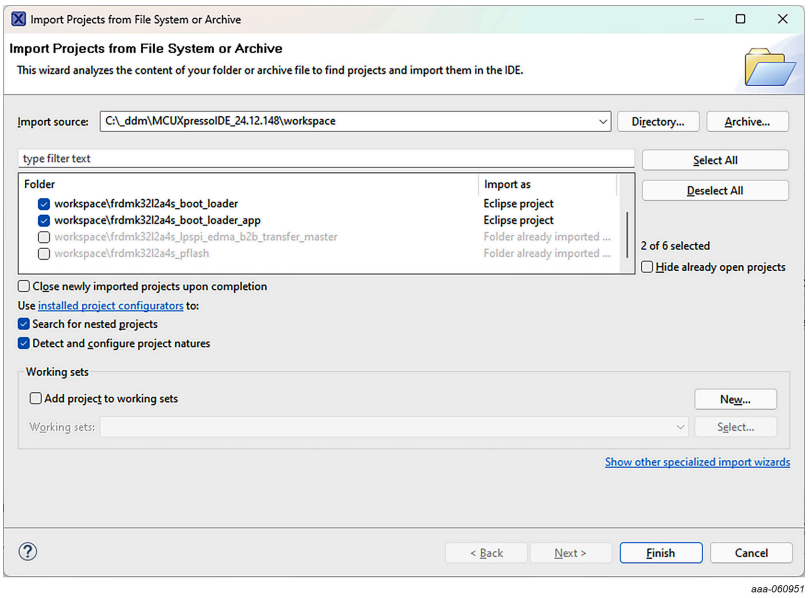

Figure 17. Open projects from file system

2. Mass erase the internal flash by GUI Flash Tool as shown in Figure 18.

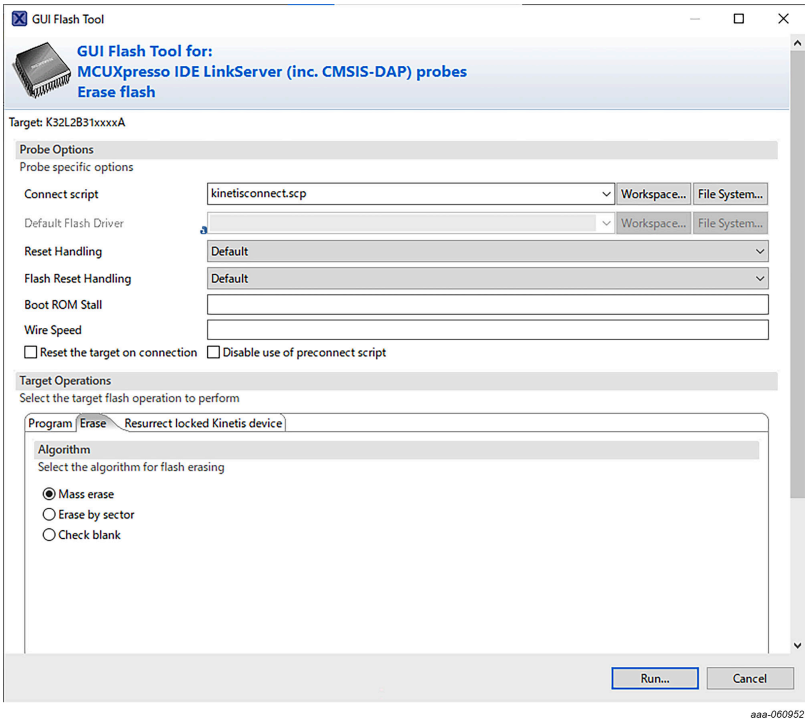

Figure 18. Mass erase the internal flash by a GUI Flash Tool

3. Program the second bootloader:
   Start debugging frdmk32l2a4s_boot_loader. If the red LED blinks, that means no firmware is found in the internal flash or the SPI flash. Defining a macro DIMAGE_DEBUG, displays the second bootloader's log.
4. Build frdmk32l2a4s_boot_loader_app Ver.1:
   a. Comment out `#define INCLUDE_NEWER_FIRMWARE` in main.c. Set `.version=1` in startup_k32l2a41a.c.

b. Build the project. An object file containing the image (Ver.1) is generated in app_bin.

5. Build frdmk32l2a4s_boot_loader_app Ver.0:
   The Ver.1 image is embedded into the Ver.0 image for debug purpose.
   a. Comment in `#define INCLUDE_NEWER_FIRMWARE` in main.c. Set `.version = 0` in startup_k32l2a41a.c.
   b. Build the project. Binary image (Ver.0) is built in Debug or Release. This image is programmed into the internal flash.

   By recursively following this step, images can be generated containing versions 0, 1, 2, 3, and so on.
6. Program Ver.0 image into the internal flash:
   a. Program frdmk32l2a4s_boot_loader_app_crc.bin in the internal flash by GUI Flash Tool as shown in Figure 19.

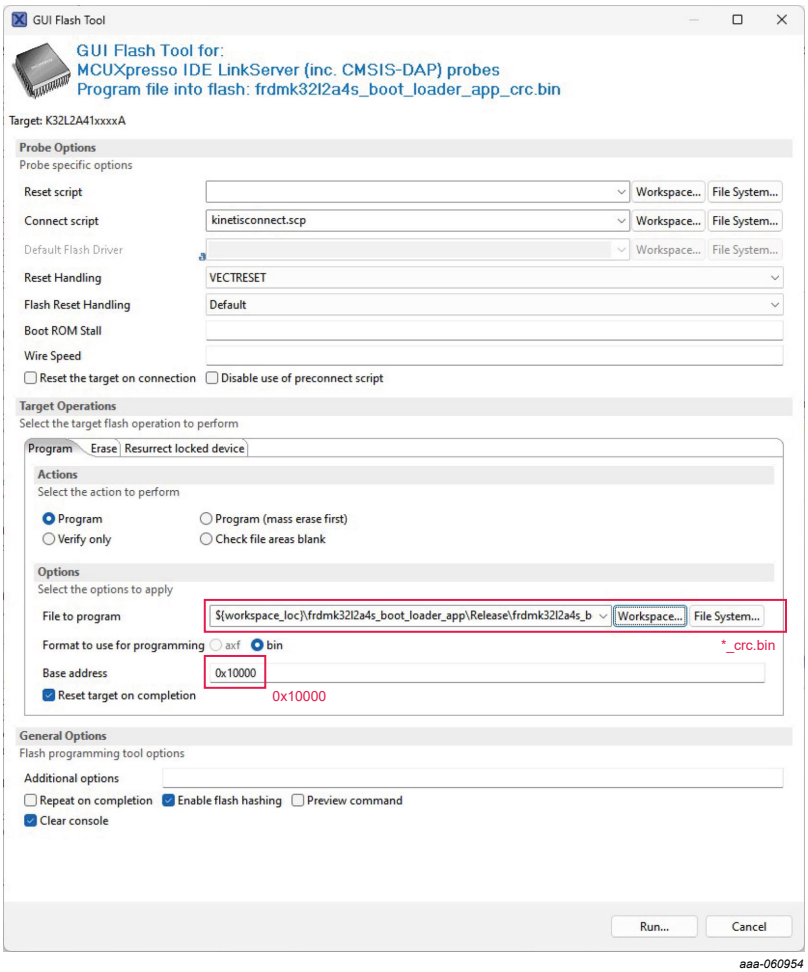

Figure 19. Programming frdmk32l2a4s_boot_loader_app_crc.bin in the internal flash

b. After reset, "Ver.0 booted" is displayed in the console as shown in Figure 20.

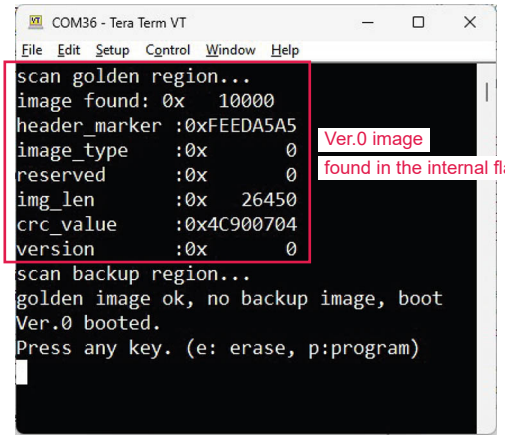

Figure 20. After reset, "Ver.0 booted" is displayed

7. Update the firmware:
   a. To program Ver.1 into the SPI flash, press the 'p' key.
   Note: *If you want to erase the image in the SPI flash while debugging, press 'e' key to erase both BANK0 and BANK1 in the SPI flash.*
   b. After reset, "Ver.1 booted" is displayed in the console as shown in Figure 21.

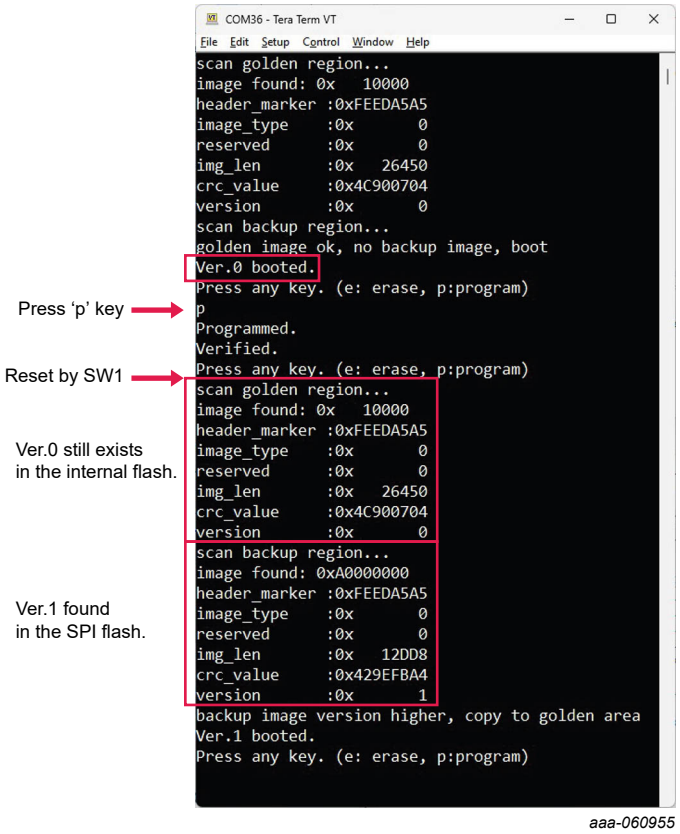

Figure 21. Firmware updated successfully

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 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.

8 Revision history

Table 5 summarizes the revisions to this document.

Table 5. Revision history

Document ID	Release date	Description
AN14699 v.1.0	13 June 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.

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.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

Kinetis — is a trademark of NXP B.V.

Contents