AN14172

Using SmartDMA for Graphic on MCX N Series MCU

Rev. 2.0 — 6 May 2024 Application note

Document information

Information	Content
Keywords	AN14172, MCX, MCU, SmartDMA, Graphic lib, FRDM-MCXN947, FRDM-MCXN236
Abstract	This application note introduces the application of SmartDMA on the graphic.

1 Introduction

This application note introduces the application of SmartDMA on the graphic. In addition to the general DMA function, it also supports data format processing.

All MCX N series MCUs include a SmartDMA coprocessor, which can effectively reduce the load on the ARM core and perform flexible data conversions.

2 Graphic lib support

The graphic library supports the following data processing:

- Common DMA
- Endian Swap
- Reverse order
- RGB565 to RGB888
- ARGB to RGB
- ARGB to RGB, then swap endian
- ARGB to RGB, then swap endian and reverse

Here, A in ARGB is Alpha (transparency).

3 Advantages over traditional DMA

DMA mainly supports data transfer between memory and peripherals, between peripherals and peripherals, and between memory and memory. In addition to accessing all peripherals and memory, SmartDMA can execute instruction code, mathematical operations, data flipping, shifting, judgment, and so on. So, SmartDMA is more flexible than DMA.

On the MCX N series of MCUs, FlexIO can be used to drive the LCD screen. However, sometimes the data is not as expected and need slight adjustments. If traditional DMA is used, it is difficult to preprocess the data and requires the ARM core to handle it, which takes more time and load. SmartDMA can be used for preprocessing and then transmitting the processed data to FlexIO.

4 Function description

SmartDMA can achieve many functions. It can be used as a common DMA to transfer data. It can also implement data format processing, such as flipping bytes, flipping bit order, removing part of the data, and so on.

4.1 Common DMA

SmartDMA can access all peripherals and storage. It has the functions of common DMA, such as data transfer from peripheral to peripheral, memory to memory, peripheral to memory, and memory to peripheral.

Because it can execute programming instructions, its functionality can be more flexible, and its parameters can be complete.

In this application note, the demo used is the simplest common DMA function. SmartDMA moves memory data to the FlexIO peripheral data register and FlexIO outputs data to the LCD.

Currently, the functionality of a common DMA operation is relatively simple. It only needs app code to provide SmartDMA with the data address and data length to be transferred. SmartDMA moves this data into the FlexIO

data register automatically. Once the FlexIO data register requires data, it automatically sends a request to SmartDMA.

4.2 Endian Swap

Endian swap represents byte order swapping. Here specifically refers to the exchange of high and low bytes in 16-bit data.

For example:

The input data:

Data in byte [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F]

After operation of SmartDMA, the output data:

Data in byte [1, 0, 3, 2, 5, 4, 7, 6, 9, 8, B, A, D, C, F, E, 11, 10, 13, 12, 15, 14, 17, 16, 19, 18, 1B, 1A, 1D, 1C, 1F, 1E]

In the MCX N series MCU, FlexIO has 8 data registers. Each register has 4 bytes. 8 data registers need 32 bytes of data. Therefore, SmartDMA can process 32 bytes of data to send to FlexIO each time.

4.3 Reverse order

This function refers to the reversal of byte order. In other words, when providing 32 bytes of data to SmartDMA, SmartDMA reverses the data and places the result data into 8 FlexIO data registers.

For example:

The input data:

Data in byte [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F]

After operation of SmartDMA, the output data:

Data in byte [1F, 1E, 1D, 1C, 1B, 1A, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, F, E, D, C, B, A, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

4.4 RGB565 to RGB888

To save RAM space, the image data can be stored by using the RGB565 pixel format. However, some LCD screen modules use the RGB888 interface. The SmartDMA can implement the conversion from RGB565 to RGB888 format. The result of SmartDMA conversion can be directly sent to the data register of FlexIO, which does not expand the occupation of memory space.

4.5 ARGB to RGB

ARGB is RGB data with an alpha component. Some original image data is in ARGB format, but the display screen may not support this format. In this case, it is necessary to remove the alpha value from each pixel data and then send it to the display interface. This implementation requires processing each pixel, and SmartDMA can easily and efficiently remove the alpha value. Additionally, SmartDMA can access the data register of FlexIO and send the converted data to the FlexIO data register directly.

SmartDMA can combine the functions of ARGB to RGB, swap endian, and reverse for use. The result is placed in the data register of FlexIO.

The flexibility and independence of SmartDMA can be verified in these functions.

5 Software description

The task execution instructions of SmartDMA are encapsulated in an array. Open some API functions for users to use. The functions used in this application note have been involved in the SDK of MCX N series MCU.

5.1 SDK example introduction

In the SDK of MCX N series MCU, there is a sample called `lvgl_demo_widgets_bm`. This example is used to demonstrate the LVGL widget demo.

The example uses FlexIO to emulate the MCU8080 interface, driving a 3.5-inch LCD screen. The SmartDMA serves as the function of a common DMA and is responsible for transferring data to FlexIO data registers.

5.2 SmartDMA function array

The SmartDMA display API can be found in the file `fsl_smartdma_mcxn.h`. As below code snippet:

```
/*!
* @brief The API index when using s_smartdmaDisplayFirmware.
*/
enum _smartdma_display_api
{
    kSMARTDMA_FlexIO_DMA_Endian_Swap = 0U,
    kSMARTDMA_FlexIO_DMA_Reverse32,
    kSMARTDMA_FlexIO_DMA,
    kSMARTDMA_FlexIO_DMA_Reverse,  /*!< Send data to FlexIO with reverse order.
 */
    kSMARTDMA_RGB565To888,  /*!< Convert RGB565 to RGB888 and save to output
 memory, use parameter
    smartdma_rgb565_rgb888_param_t. */
    kSMARTDMA_FlexIO_DMA_RGB565To888,  /*!< Convert RGB565 to RGB888 and send to
 FlexIO, use parameter
    smartdma_flexio_mculcd_param_t. */
    kSMARTDMA_FlexIO_DMA_ARGB2RGB,  /*!< Convert ARGB to RGB and send to FlexIO,
 use parameter
    smartdma_flexio_mculcd_param_t. */
    kSMARTDMA_FlexIO_DMA_ARGB2RGB_Endian_Swap, /*!< Convert ARGB to RGB, then
 swap endian, and send to FlexIO, use
    parameter smartdma_flexio_mculcd_param_t. */
    kSMARTDMA_FlexIO_DMA_ARGB2RGB_Endian_Swap_Reverse, /*!< Convert ARGB to RGB,
 then swap endian and reverse, and send
    to FlexIO, use parameter smartdma_flexio_mculcd_param_t. */
};
```

In the `fsl_smartdma_mcxn.c` file, there is an array called `s_smartdmaDisplayFirmware`, which contains the implementation of SmartDMA functions. The purpose of encapsulating the SmartDMA functions into an array is to reduce the SmartDMA research cost for users. It also allows them to directly use the module functions implemented, enabling faster implementation of application functions.

5.3 SmartDMA initialization

The functions described in Table 1 implement the initialization of SmartDMA.

Table 1. SmartDMA initialization

Routine	Description
`SMARTDMA_InitWithoutFirmware`	Initialize the SmartDMA
`SMARTDMA_InstallFirmware`	Install the firmware
`SMARTDMA_InstallCallback`	Install the complete callback function
`SMARTDMA_Boot`	Boot the SMARTDMA to run the program
`SMARTDMA_Deinit`	De-initialize the SMARTDMA
`SMARTDMA_Reset`	Reset the SMARTDMA
`SMARTDMA_HandleIRQ`	SMARTDMA IRQ
`FLEXIO_MCULCD_SMARTDMA_Callback`	SMARTDMA interrupt callback

5.3.1 Init SmartDMA

To enable SmartDMA, perform the following operations:

1. Clear reset of SmartDMA.
2. Set FlexIO IRQ as the SmartDMA trigger input.
3. Enable the clock for SmartDMA.
4. Enable the IRQ for SmartDMA.

5.3.2 Install SmartDMA firmware

The function module of SmartDMA must be placed at a fixed memory address to work fine. In this application, it must be placed at 0x04000000, as described below:

```
/*! @brief The firmware used for display. */
extern const uint8_t s_smartdmaDisplayFirmware[];
/*! @brief The s_smartdmaDisplayFirmware firmware memory address. */
#define SMARTDMA_DISPLAY_MEM_ADDR 0x04000000U
/*! @brief Size of s_smartdmaDisplayFirmware */
#define SMARTDMA_DISPLAY_FIRMWARE_SIZE (s_smartdmaDisplayFirmwareSize)
```

The process of installing SmartDMA firmware is essentially copying the code array of SmartDMA function modules to a specified RAM address, as described below:

```
SMARTDMA_InstallFirmware(SMARTDMA_DISPLAY_MEM_ADDR, s_smartdmaDisplayFirmware,
 SMARTDMA_DISPLAY_FIRMWARE_SIZE);
```

5.3.3 SmartDMA callback routine

SmartDMA can actively trigger an interruption in the ARM core, such as after the end of data transfer.

SmartDMA has a related interrupt number (`SMARTDMA_IRQHandler`) in the ARM vector table. In the configuration phase of SmartDMA, a callback function can be installed, as described below:

```
SMARTDMA_InstallCallback(FLEXIO_MCULCD_SMARTDMA_Callback, handle);
```

In the callback function, the ARM core can configure the FlexIO to allow the task to continue.

5.3.4 Boot SmartDMA API

In the application, define a structure to set parameters related to SmartDMA. These parameters include the address of the data buffer, the length of data transfer, and the address of SmartDMA stack space. The most important thing is to find an API that must be executed from the SmartDMA function block code. See the below code.

```
handle->smartdmaApi = (uint8_t)kSMARTDMA_FlexIO_DMA;
handle->smartdmaParam.p_buffer = (uint32_t *)(xfer->dataAddrOrSameValue +
 part1Len);
handle->smartdmaParam.buffersize = part2Len;
handle->smartdmaParam.smartdma_stack = handle->smartdmaStack;
SMARTDMA_Reset();
SMARTDMA_Boot(handle->smartdmaApi, &(handle->smartdmaParam), 0);
```

The process of boot is to give the address of the corresponding API to the program counter of SmartDMA, and then it begins to execute the function block.

6 Demo based on FRDM-MCXN947 introduction

Download the latest SDK for MCX N MCU. Open the path of example `lvgl_demo_widgets_bm`. The root path is:

`\boards\frdmmcxn947\lvgl_examples\lvgl_demo_widgets_bm\cm33_core0`

This project primarily demonstrates the functionality of LVGL widgets. The display driver is implemented by using FlexIO to emulate the MCU8080 interface. SmartDMA assists FlexIO in transferring data from RAM to the data register.

Figure 1 shows the IAR project.

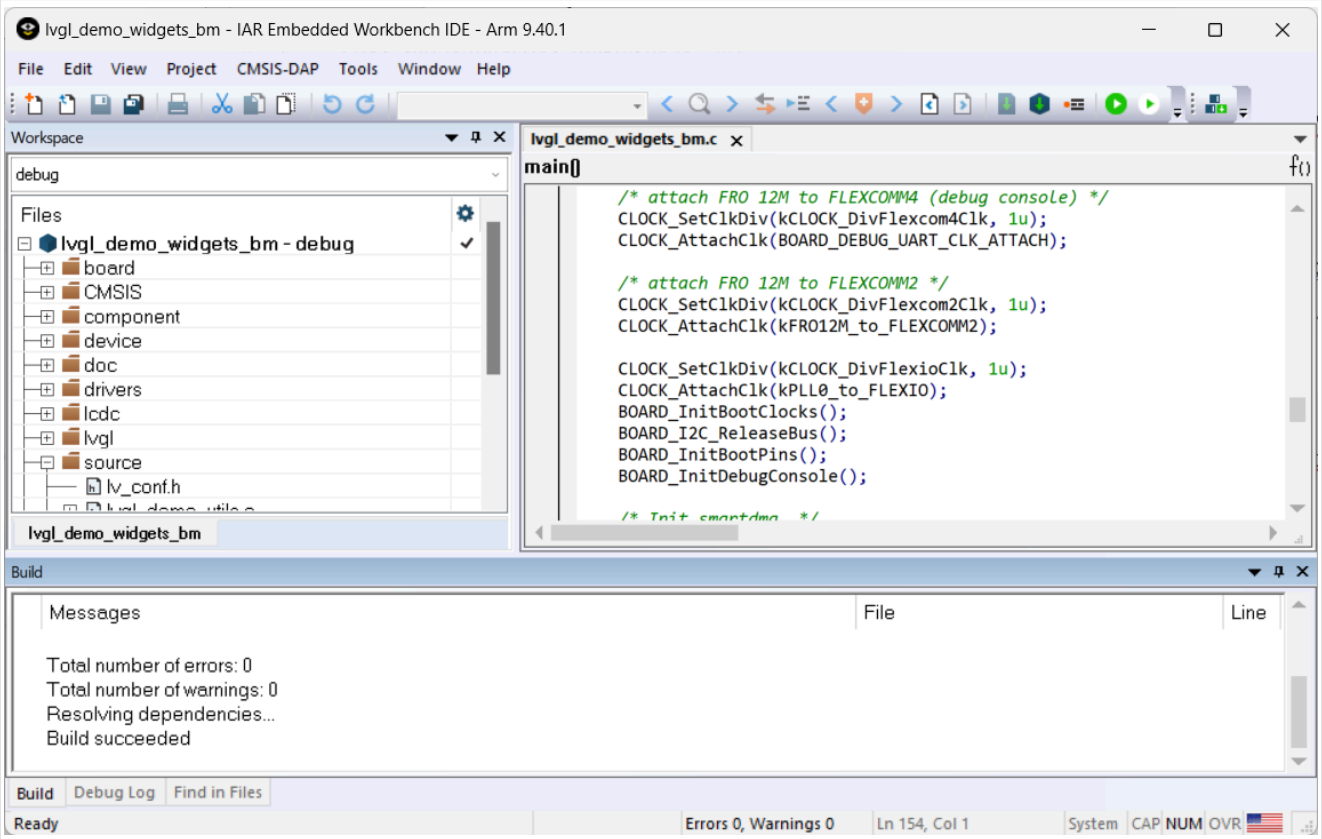

Figure 1. IAR project

To show the common DMA function of SmartDMA, perform the following steps:

1. Connect the USB cable to the computer and FRDM-MCXN947 port J17.
2. Compile and download the code.
3. Press the Reset button, and the code starts to run.
4. The operation of the LVGL widget demo displays on the screen.

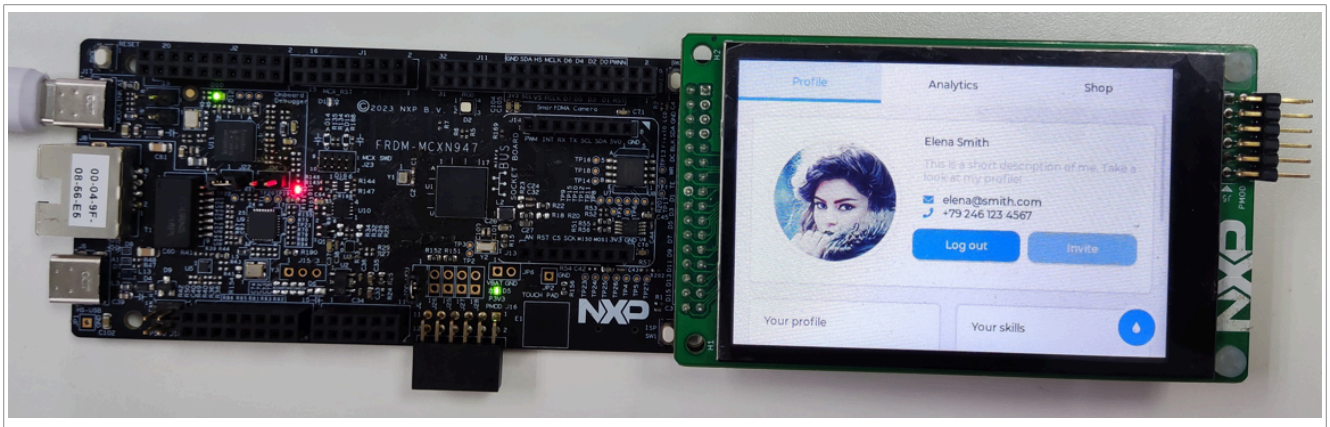

Figure 2. LVGL widget demo

7 Demo based on FRDM-MCXN236 introduction

Download the latest SDK for MCX N MCU. Open the path of `lvgl_demo_widgets_bm` example. The root path is: `\boards\frdmmcxn236\lvgl_examples\lvgl_demo_widgets_bm`.

This project primarily demonstrates the functionality of LVGL widgets. The display driver is implemented by using FlexIO to emulate the MCU8080 interface. SmartDMA assists FlexIO in transferring data from RAM to the data register.

Figure 3 shows the MCUXpresso project.

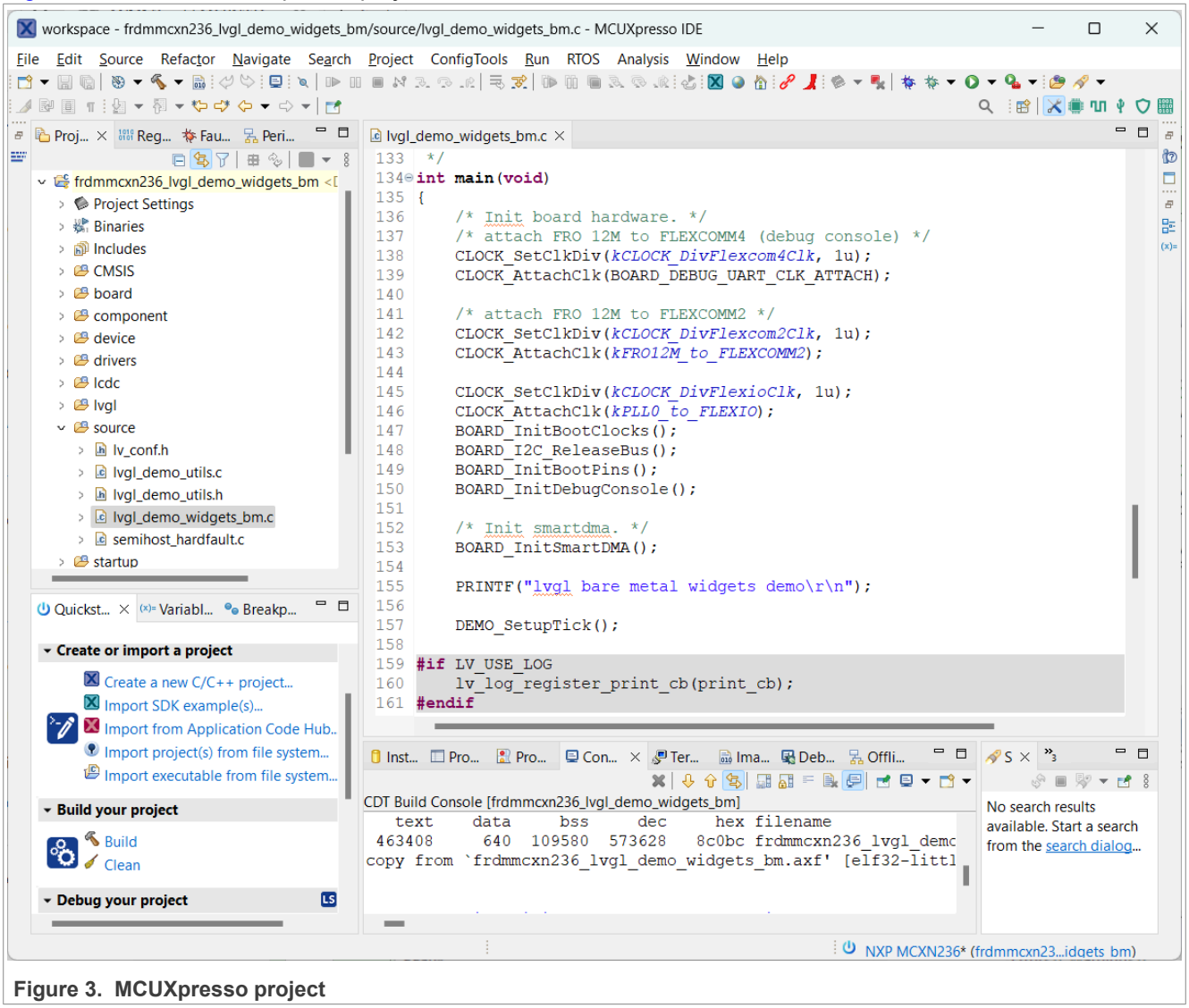

Figure 3. MCUXpresso project

To show the common DMA function of SmartDMA, perform the following steps:

1. Connect the USB cable to the computer and FRDM-MCXN236 port J10.
2. Compile and download the code.
3. Press the Reset button, and the code starts to run.
4. The operation of the LVGL widget demo displays on the screen.

Figure 4 shows the demo result.

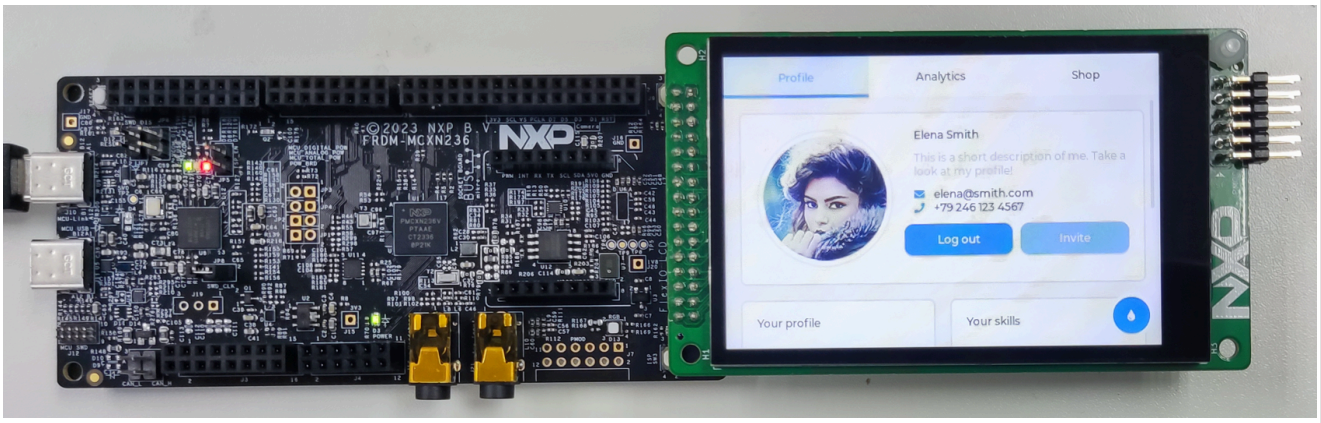

Figure 4. Demo result

8 Note about the source code in the document

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

Copyright 2024 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.

9 Revision history

Table 2 summarizes the revisions done to this document.

Table 2. Revision history

Document ID	Release date	Description
AN14172 v.2.0	6 May 2024	• Updated Section 6 • Added Section 7
AN14172 v.1.0	20 January 2024	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.

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

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

Date of release: 6 May 2024
Document identifier: AN14172