AN14726

eDMA Linked Channels with Scatter/Gather Functionality

Rev. 1.0 — 25 July 2025 Application note

Document information

Information	Content
Keywords	AN14726, Enhanced Direct Memory Access, eDMA, Scatter/Gather, Channel linking, TCD, Transfer Control Descriptor, Software triggering
Abstract	This document introduces two important features of enhanced Direct Memory Access (eDMA) on the i.MX RT1180: Scatter/Gather and Channel Linking. The behavior of these features is described using a simple use case.

1 Introduction of eDMA

The Enhanced Direct Memory Access (eDMA) is a powerful feature designed to offload memory transfer tasks from the CPU. Therefore, improving the performance and efficiency, especially in high-throughput applications.

On i.MX RT118x SoCs, there are two eDMAs: eDMA3 and eDMA4. eDMA3 is connected to the AHB32 bus, whereas eDMA4 is connected to the AXI bus. eDMA3 and eDMA4 have the same functionality, but differ in some bit fields. However, this use case and the code are applicable to both.

2 TCD

TCD is a data structure used by the eDMA engine and it defines channel configuration. It defines all the parameters needed for a single DMA transfer or chain of transfers, such as the size of read/write transaction, source/destination address, and address offsets.

TCD contains status flags which indicate the state of the particular channel or indicate errors in the channel, such as destination bus error, and source bus error. This supports high configurability for each separate channel so that it can be used for every specific use case.

The channel can be activated by any one of these three methods:

- Explicit initiation by software (Writing start bit)
- Initiation by channel to channel linking (The channel triggers another channel upon Major/Minor loop completion)
- Initiation by hardware request (FIFO full signal from LPSPI)

The TCD data structure is 32 bytes long and it is stored in eDMAx local memory. When the channel is triggered, this structure is copied from eDMAx local memory to the eDMA engine and then it controls the transaction.

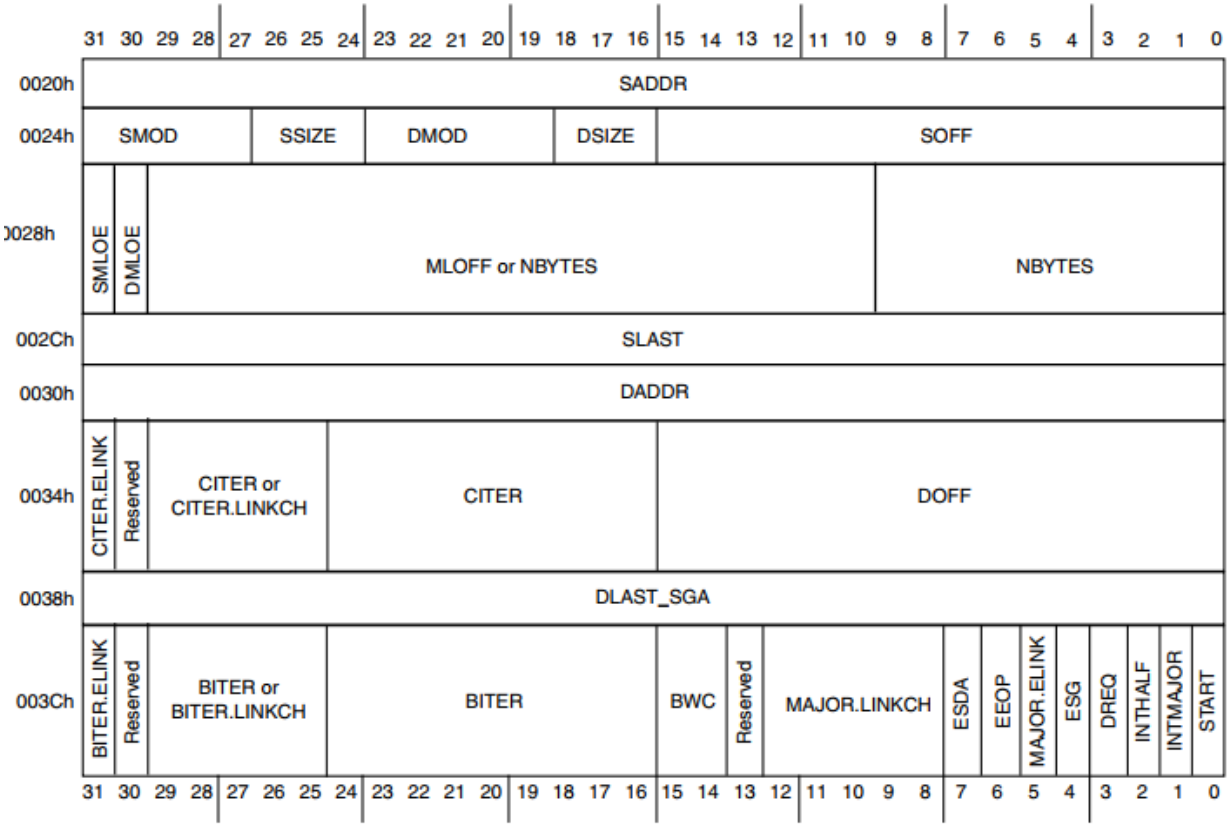

Figure 1. TCD structure

3 Scatter/Gather feature

Scatter/Gather is the process of automatically loading a new TCD into a channel from memory. It allows a DMA channel to use multiple TCDs. It in turn enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When Scatter/Gather is enabled and the channel has finished its major loop, a new TCD is fetched from the system memory and loaded into that channel's descriptor location in the eDMA programmer's model. Therefore, replacing the current descriptor.

4 Channel linking feature

Channel linking or chaining is a mechanism where one channel sets the TCDn_CSR[START] bit of another channel or itself. Therefore, initiating a service request for that channel. When properly enabled, the eDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop or one iteration of the major loop. The TCDn_CITER[ELINK] field determines whether a minor loop link is requested. When the TCDn_CITER[ELINK] field is enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made.

5 Use case

This application demonstrates the use of three separate DMA channels: Channel 0 (CH0), Channel 1 (CH1), and Channel 2 (CH2).

It also enables two important eDMAx features: Channel linking (end of the Major loop), and Scatter/Gather feature.

5.1 Data sequence

5.1.1 Data buffers:

- A source buffer containing 4 words is allocated in OCRAM1:
  - Each channel has its own source buffer.
- Destination buffers containing 8 words (4 words from source buffer, 4 words offsets) are allocated in OCRAM2:
  - CH0 has only 1 destination buffer
  - Both CH1 and CH2 have 2 destination buffers (Scatter/Gather feature)

5.1.2 All the channels perform the following data transfer:

- In the Major loop, 4 words are transferred.
- Read 1 word from the source address.
- Write 1 word to the destination address.
- Add 1 word offset to the destination address.
- Repeat until 4 words are transferred (after major loop completion).

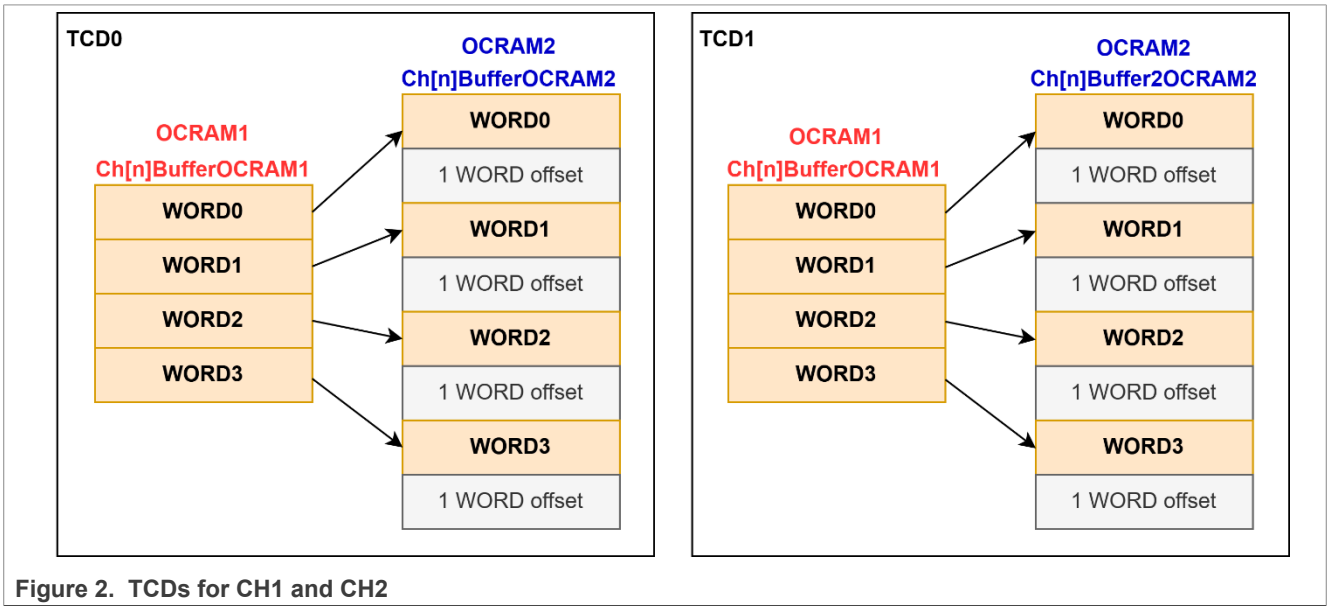

Figure 2. TCDs for CH1 and CH2

5.2 Channel configuration

Configurations of Channel 0, Channel 1, and Channel 2 are given below:

5.2.1 Channel 0

- TCD is configured to transfer data from the Ch0BufferOCRAM1 in OCRAM1 to Ch0BufferOCRAM2 in OCRAM2.
- Triggered by the software in periodic event and is set to link CH1.
- At the end of the major loop, CH0 triggers CH1.

5.2.2 Channel 1

- TCD0 is configured to transfer data from the Ch1BufferOCRAM1 in OCRAM1 to Ch1Buffer1OCRAM2 in OCRAM2.
- TCD1 is configured to transfer data from the Ch1BufferOCRAM1 in OCRAM1 to Ch1Buffer2OCRAM2 in OCRAM2.
- Channel 1 has a Scatter/Gather feature enabled. At the end of the Major loop, the new TCD is loaded from memory to eDMAx local TCD memory.
- Then it triggers CH2 as the Channel linking feature is enabled for this channel.

5.2.3 Channel 2

- TCD0 is configured to transfer data from the Ch2BufferOCRAM1 in OCRAM1 to Ch1Buffer1OCRAM2 in OCRAM2.
- TCD1 is configured to transfer data from the Ch2BufferOCRAM1 in OCRAM1 to Ch2Buffer2OCRAM2 in OCRAM2.
- Channel 2 has a Scatter/Gather feature enabled. At the end of the Major loop, the new TCD is loaded from memory to eDMAx local TCD memory.
- The channel linking for this channel is not enabled. So, at the end of the major loop, the whole sequence is done. The channel CH0 is waiting for software trigger which starts the sequence again.

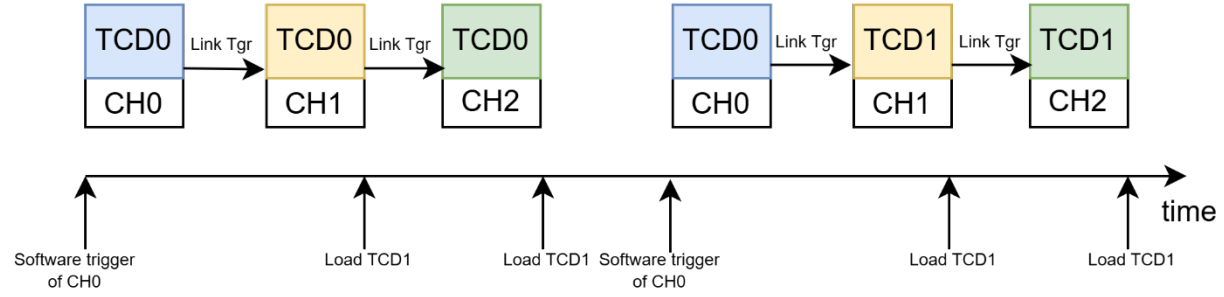

Figure 3. Trigger sequence

5.3 Configuration of the eDMAx

To configure the eDMAx, perform the following steps:

1. Create the buffers for data and place them in to the corresponding memories as given below:

```
/* OCRAMs buffers */
// Buffers for Channel 0
uint32_t Ch0BufferOCRAM1[4]__attribute__((section(".OCRAM1")));
uint32_t Ch0BufferOCRAM2[8] __attribute__((section(".OCRAM2")));
// Buffers for Channel 1
uint32_t Ch1BufferOCRAM1[4] __attribute__((section(".OCRAM1")));
uint32_t Ch1Buffer1OCRAM2[8] __attribute__((section(".OCRAM2")));
uint32_t Ch1Buffer2OCRAM2[8] __attribute__((section(".OCRAM2")));
```

```
// Buffers for Channel 2
uint32_t Ch2BufferOCRAM1[4]__attribute__((section(".OCRAM1")));
uint32_t Ch2Buffer1OCRAM2[8]__attribute__((section(".OCRAM2")));
uint32_t Ch2Buffer2OCRAM2[8]__attribute__((section(".OCRAM2")));
```

2. Create Arrays for TCD0, 1 for CH1 and CH2.
   These arrays are stored in OCRAM1 memory and after major loop completion of CH1/2 are copied into eDMAx internal TCD memory.
   Note: *Ensure that the TCD address is 32 bytes aligned.*

```
edma_tcd_t Ch1TCDArr[2] __attribute__((section(".OCRAM1"))) __attribute__
 ((aligned (32)));
edma_tcd_t Ch2TCDArr[2]__attribute__((section(".OCRAM1"))) __attribute__
 ((aligned (32)));
```

3. Go to settings of the channels and enable linking globally for eDMAx:

```
DMA4->MP_CSR |= DMA_MP_CSR_GCLC_MASK;
```

4. Set the following addresses:

```
// Set source address - address of Ch0BufferOCRAM1
DMA4->TCD[0].SADDR = &Ch0BufferOCRAM1;
// Source offset - after each read of source add this offset to read address
DMA4->TCD[0].SOFF = 0x4;
// set read transaction size to 4 bytes, set store transaction to 4 bytes
DMA4->TCD[0].ATTR = DMA_TCD_ATTR_DMOD(0x0) | DMA_TCD_ATTR_SMOD(0x0) |
 DMA_TCD_ATTR_SSIZE(0x2) | DMA_TCD_ATTR_DSIZE(0x2);
// Set number of bytes to be transferred each minor loop, for this case 16
 bytes
DMA4->TCD[0].NBYTES_MLOFFYES = DMA_TCD_NBYTES_MLOFFNO_NBYTES(16); // 4 bytes
 per transaction
// Set source address offset after end of the major loop; for this case -16
 bytes offset
DMA4->TCD[0].SLAST_SDA = DMA4_SLAST_SDA_SLAST_SDA(-16);
// Set destination address - address of Ch0BufferOCRAM2
DMA4->TCD[0].DADDR = DMA_TCD_DADDR_DADDR( &Ch0BufferOCRAM2 );
// Destination offset - after each read of source add this offset to write
 address; for this case +8 bytes
DMA4->TCD[0].DOFF = DMA_TCD_DOFF_DOFF(8);
// Set current iteration number to 1
DMA4->TCD[0].CITER_ELINKNO = 1;
// Set source destination address offset after end of the major loop; for
 this case -32bytes offset
DMA4->TCD[0].DLAST_SGA = -32;
// Enable channel linking after major loop completes and set the number of
 the channel to be linked
DMA4->TCD[0].CSR = DMA4_CSR_MAJORLINKCH(0x1) | DMA4_CSR_MAJORELINK_MASK;
// Set default iteration count to 1
DMA4->TCD[0].BITER_ELINKNO = DMA_TCD_BITER_ELINKNO_BITER(1);
```

5. Create TCD0 and TCD1 for CH1 as given below:

```
// Set source address - address of Ch1BufferOCRAM1
Ch1TCDArr[0].SADDR = &Ch1BufferOCRAM1;
// Source offset - after each read of source add this offset to read address
Ch1TCDArr[0].SOFF = 0x4;
// set read transaction size to 4 bytes, set store transaction to 4 bytes
Ch1TCDArr[0].ATTR = DMA_TCD_ATTR_DMOD(0x0) | DMA_TCD_ATTR_SMOD(0x0) |
 DMA_TCD_ATTR_SSIZE(0x2) | DMA_TCD_ATTR_DSIZE(0x2);
```

```
// Set number of bytes to be transferred each minor loop, for this case 16
 bytes
Ch1TCDArr[0].NBYTES = DMA_TCD_NBYTES_MLOFFYES_NBYTES(16); // 4 bytes per
 transaction
// Set source address offset after end of the major loop; for this case -16
 bytes offset
Ch1TCDArr[0].SLAST = DMA4_SLAST_SDA_SLAST_SDA(-16);
// Set destination address - address of Ch1BufferOCRAM2
Ch1TCDArr[0].DADDR = DMA_TCD_DADDR_DADDR( &Ch1Buffer1OCRAM2 );
// Destination offset - after each read of source add this offset to write
 address; for this case +8 bytes
Ch1TCDArr[0].DOFF = DMA_TCD_DOFF_DOFF(8);
// Set current iteration number to 1
Ch1TCDArr[0].CITER = 1;
// Set address of new TCD which will be loaded at the end of the Major loop
Ch1TCDArr[0].DLAST_SGA = &Ch1TCDArr[1];
// Enable channel linking after major loop completes and set the number of
 the channel to be linked, enable S/G
Ch1TCDArr[0].CSR = DMA4_CSR_ESG_MASK | DMA4_CSR_MAJORLINKCH(0x2) |
 DMA4_CSR_MAJORELINK_MASK;
// Set default iteration count to 1
Ch1TCDArr[0].BITER = 1;
Ch1TCDArr[1].SADDR = &Ch1BufferOCRAM1;
Ch1TCDArr[1].SOFF = 0x4;
Ch1TCDArr[1].ATTR = DMA_TCD_ATTR_DMOD(0x0) | DMA_TCD_ATTR_SMOD(0x0) |
 DMA_TCD_ATTR_SSIZE(0x2) | DMA_TCD_ATTR_DSIZE(0x2);
Ch1TCDArr[1].NBYTES = DMA_TCD_NBYTES_MLOFFYES_NBYTES(16) |
 DMA_TCD_NBYTES_MLOFFYES_DMLOE_MASK | DMA_TCD_NBYTES_MLOFFYES_MLOFF(32); // 4
 bytes per transaction
Ch1TCDArr[1].SLAST = DMA4_SLAST_SDA_SLAST_SDA(-16);
Ch1TCDArr[1].DADDR = DMA_TCD_DADDR_DADDR( &Ch1Buffer2OCRAM2 );
Ch1TCDArr[1].DOFF = DMA_TCD_DOFF_DOFF(8);
Ch1TCDArr[1].CITER = 1;
Ch1TCDArr[1].DLAST_SGA = &Ch1TCDArr[0];
Ch1TCDArr[1].CSR = DMA4_CSR_ESG_MASK | DMA4_CSR_MAJORLINKCH(0x2) |
 DMA4_CSR_MAJORELINK_MASK;
Ch1TCDArr[1].BITER = 1;
```

6. Create TCD1 and TCD2 for CH2 as given below:

```
// Set source address - address of Ch2BufferOCRAM1
Ch2TCDArr[0].SADDR = &Ch2BufferOCRAM1;
// Source offset - after each read of source add this offset to read address
Ch2TCDArr[0].SOFF = 0x4;
// set read transaction size to 4 bytes, set store transaction to 4 bytes
Ch2TCDArr[0].ATTR = DMA_TCD_ATTR_DMOD(0x0) | DMA_TCD_ATTR_SMOD(0x0) |
 DMA_TCD_ATTR_SSIZE(0x2) | DMA_TCD_ATTR_DSIZE(0x2);
// Set number of bytes to be transferred each minor loop, for this case 16
 bytes
Ch2TCDArr[0].NBYTES = DMA_TCD_NBYTES_MLOFFYES_NBYTES(16); // 4 bytes per
 transaction
// Set source address offset after end of the major loop; for this case -16
 bytes offset
Ch2TCDArr[0].SLAST = DMA4_SLAST_SDA_SLAST_SDA(-16);
// Set destination address - address of Ch2BufferOCRAM2
Ch2TCDArr[0].DADDR = DMA_TCD_DADDR_DADDR( &Ch2Buffer1OCRAM2 );
// Destination offset - after each read of source add this offset to write
 address; for this case +8 bytes
Ch2TCDArr[0].DOFF = DMA_TCD_DOFF_DOFF(8);
// Set current iteration number to 1
Ch2TCDArr[0].CITER = 1;
```

```
// Set address of new TCD which will be loaded at the end of the Major loop
Ch2TCDArr[0].DLAST_SGA = &Ch2TCDArr[1];
// enable S/G and enable interrupt for this channel
Ch2TCDArr[0].CSR = DMA4_CSR_ESG_MASK | DMA4_CSR_INTMAJOR_MASK;
Ch2TCDArr[0].BITER = 1;
Ch2TCDArr[1].SADDR = &Ch2BufferOCRAM1;
Ch2TCDArr[1].SOFF = 0x4;
Ch2TCDArr[1].ATTR = DMA_TCD_ATTR_DMOD(0x0) | DMA_TCD_ATTR_SMOD(0x0) |
 DMA_TCD_ATTR_SSIZE(0x2) | DMA_TCD_ATTR_DSIZE(0x2);
Ch2TCDArr[1].NBYTES = DMA_TCD_NBYTES_MLOFFYES_NBYTES(16) |
 DMA_TCD_NBYTES_MLOFFYES_DMLOE_MASK | DMA_TCD_NBYTES_MLOFFYES_MLOFF(32); // 4
 bytes per transaction
Ch2TCDArr[1].SLAST = DMA4_SLAST_SDA_SLAST_SDA(-16);
Ch2TCDArr[1].DADDR = DMA_TCD_DADDR_DADDR( &Ch2Buffer2OCRAM2 );
Ch2TCDArr[1].DOFF = DMA_TCD_DOFF_DOFF(8);
Ch2TCDArr[1].CITER = 1;
Ch2TCDArr[1].DLAST_SGA = &Ch2TCDArr[0];
Ch2TCDArr[1].CSR = DMA4_CSR_ESG_MASK | DMA4_CSR_INTMAJOR_MASK;
Ch2TCDArr[1].BITER = 1;
```

7. In PIT, the periodic handler is CH0. It is periodically triggered by the software initiation as given below:

```
void DEMO_LPIT_IRQHandler(void)
{
/* Clear interrupt flag.*/
DMA4->TCD[0].CH_CSR |= DMA4_CH_CSR_DONE_MASK;
DMA4->TCD[0].CSR |= DMA4_CSR_START_MASK;
SDK_ISR_EXIT_BARRIER;
}
```

6 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 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.

7 Revision history

Table 1 summarizes the revisions done to this document.

Table 1. Revision history

Document ID	Release date	Description
AN14726 v.1.0	25 July 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.

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.

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.

Suitability for use in industrial applications (functional safety) — This NXP product has been qualified for use in industrial applications. It has been developed in accordance with IEC 61508, and has been SIL-classified accordingly. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

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.

Tables

Figures

Contents