AN13833

LPC86x UART receive IDLE interrupt

Rev. 0 — 10 May 2023 Application note

Document Information

Information	Content
Keywords	LPC86x, USART, UART
Abstract	LPC86x is an Arm Cortex-M0+ based, low-cost, 32-bit MCU family and it supports three USARTs, one I^2C, one I3C, and two SPI ports. LPC86x supports up to 64 KB of flash memory and 8 KB of SRAM.

1 Introduction

LPC86x is an Arm Cortex-M0+ based, low-cost, 32-bit MCU family and it supports three USARTs, one I^2C, one I3C, and two SPI ports. LPC86x supports up to 64 KB of flash memory and 8 KB of SRAM.

LPC86x feature an updated UART to support the receive idle timeout interrupt based on LPC84x. This means that only LPC86x support the UART receive idle timeout flag and interrupt; the LPC80x, LPC81x, LPC82x, LPC83x, and LPC84x do not.

LPC86x's UART receive idle timeout flag and interrupt are just a flag and the interrupt is not used to stop the UART DMA transfer. You must stop the DMA transfer by software if you detect the UART receive idle timeout.

When LPC86x's UART receiver is idle for a certain period of time (set in **CTL.RXIDLETOCFG**), the register bit **STAT.RXIDLETO** flag is set and an interrupt is asserted (if enabled). Writing 1 to **STAT.RXIDLETO** clears the flag. When it is cleared, it cannot be set again until the receiver receives a new character (non-idle).

2 Registers

Configuring the following registers can enable the UART receive idle timeout interrupt and get the idle timeout status.

2.1 USART STAT register

The USART STAT register primarily provides a set of USART status flags (not including the FIFO status) for the software to read. Flags other than the read-only flags may be cleared by writing ones to the corresponding bits of the STAT. The interrupt status flags that are read-only and cannot be cleared by software can be masked using the INTENCLR register; see Table 1.

The error flags for the received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by the software action in STAT.

Note: *The receive idle timeout flag uses bit:17 in the STAT register.*

STAT: Offset = 0x008

Table 1. Receive idle timeout status bit in USART status register (STAT)

Bit	Name	Type	Description
17	RXIDLETO	R/W	RX IDLE Timeout flag. It is set when the receiver is idle for a certain period of time, as specified by CTRL.RXIDLETOCFG. Writing 1 clears this bit and lowers the corresponding interrupt. When cleared, it cannot be set again until the receiver receives a new character (non-idle).

2.2 USART CTL register

The CTL register controls the aspects of USART operation that are more likely to change during operation.

The USART **CTL.RXIDLETOCFG** (bit20:18) register used to set the receive idle timeout period. See Table 2.

CTRL: Offset = 0x004

Table 2. Receive idle timeout configure bit in USART control register (CTL)

Bit	Name	Type	Description
20:18	RXIDLETOCFG	R/W	RX IDLE timeout configuration.

Table 2. Receive idle timeout configure bit in USART control register (CTL)

Bit	Name	Type	Description
			Configures the number of idle characters that must be received before the RXIDLETO flag is set. An idle character is a character whose entire frame are all 1s, including start bit, data bits, and stop bits (no activity on the receiver line). 000b - 1 idle character 001b - 2 idle characters 010b - 4 idle characters 011b - 8 idle characters 100b - 16 idle characters 101b - 32 idle characters 110b - 64 idle characters 111b - 128 idle characters

2.3 USART INTENSET register

The INTENSET (interrupt enable read and set) register is used to enable various USART interrupt sources. The enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register makes those bits set. The INTENCLR register is used to clear bits in this register. See Table 3.

INTENSET: Offset = 0x00C

Table 3. Receive idle timeout INTSET bit in USART interrupt enable read and set register (INTENSET)

Bit	Name	Type	Description
17	RXIDLETOEN	R/W	RX IDLE timeout interrupt enable. 0: The RX IDLE timeout interrupt is disabled. 1: When STAT.RXIDLETO is set, an interrupt is asserted.

2.4 USART INTENCLR register

The INTENCLR register is used to clear the bits in the INTENSET register. See Table 4.

INTCLR: Offset = 0x010

Table 4. Receive idle timeout INTCLR bit in USART interrupt enable clear register (INTENCLR)

Bit	Name	Type	Description
17	RXIDLETOCLR	WO	Writing 1 to clear INTENSET.RXIDLETOEN. Writing 0 has no effect. 0: The RX IDLE time out interrupt is disabled. 1: When STAT.RXIDLETO is set, an interrupt is asserted.

2.5 USART INTSTAT register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 5 for a detailed description of the interrupt flags.

INTCLR: Offset = 0x010

Table 5. Receive idle timeout status bit in USART interrupt status register (INTSTAT)

Bit	Name	Type	Description
17	RXIDLETOINT	RO	RX IDLE timeout interrupt flag

3 Software

3.1 Software configuration process

Step1 : Configure a PIN as USART TXD and RXD with Switch Matrix API.

Step2 : Select the USART clock source using CLOCK_Select().

Step3 : Initialize the USART configuration parameter using USART_GetDefaultConfig().

Step4 : Initialize the USART using USART_Init().

Step5 : **Enable the receive idle timeout timing using USARTx->CTL.**

Step6 : **Set the USART interrupt using USART_EnableInterrupts().**

Step7 : Enable the USART interrupt in NVIC using EnableIRQ().

When the USART receives an idle timeout, bit 17 in USARTx->STAT will be set as 1 and trigger USARTx_IRQHandler().

3.2 Source code

The test code for the UART receive idle timeout is as follows:

```
void USART0_IRQHandler(void)
{
    GPIO->NOT[1] = 1UL<<6;  // toggle GPIO
    uint8_t data;
    uint32_t status;
    status = USART0->STAT; // Get USART status
    /* If new data arrived. */
    if ((kUSART_RxReady) & status)
    {
        data = (uint8_t)USART0->RXDAT & 0xFFU; // Received 1 byte
    }
    if((status&0x20000) != 0) // Receive idle time out
    {
        USART0->STAT = (1UL << 17); // Clear USART timeout int Status flag
    }
}
/*!
 * @brief Main function
*/
int main(void)
{
    usart_config_t config;
    /* Define the init structure for the output pin*/
    gpio_pin_config_t output_config = {
        kGPIO_DigitalOutput,
        1,
    };
    /* Board pin, clock, debug console init */
    CLOCK_EnableClock(kCLOCK_Iocon);     /* Enables clock for IOCON.: enable */
```

```
    CLOCK_EnableClock(kCLOCK_Swm);        /* Enables clock for switch matrix.:
 enable */
    CLOCK_EnableClock(kCLOCK_Gpio0);      /* Enables the clock for the GPIO0
 module */
    CLOCK_EnableClock(kCLOCK_Gpio1);      /* Enables the clock for the GPIO1
 module */
    /* USART0 */
    IOCON->PIO[IOCON_INDEX_PIO1_16] =  (IOCON_MODE_PULLUP | IOCON_HYS_EN);
    IOCON->PIO[IOCON_INDEX_PIO1_17] =  (IOCON_MODE_PULLUP | IOCON_HYS_EN);
    SWM_SetMovablePinSelect(SWM0, kSWM_USART0_TXD, kSWM_PortPin_P1_17); /*
 USART0_TXD connect to P1_17 */
    SWM_SetMovablePinSelect(SWM0, kSWM_USART0_RXD, kSWM_PortPin_P1_16);
  /* USART0_RXD connect to P1_16 */
    /* Select the main clock as source clock of USART0. */
    CLOCK_Select(kUART0_Clk_From_MainClk);
    USART_GetDefaultConfig(&config);
    config.enableRx     = true;
    config.enableTx     = true;
    config.baudRate_Bps = 115200;
    /* Initialize the USART with configuration. */
    USART_Init(USART0, &config, SystemCoreClock);
    USART0->CTL &= (0xFFC3FFFF);  // Clean USART CTL.RXIDLETOCFG
    USART0->CTL |= (0UL << 18);   // 0 idle characters
    USART_EnableInterrupts(USART0, (uint32_t)((kUSART_RxReadyInterruptEnable)|
(1UL<<17)));
    USART_ClearStatusFlags(USART0, (1UL << 17));
    EnableIRQ(USART0_IRQn);
    /* P1_6 */
    IOCON->PIO[IOCON_INDEX_PIO1_6]  =  (IOCON_MODE_PULLUP | IOCON_HYS_EN);
  /* P1_6 output */
    GPIO_PinInit(GPIO, 1, 6, &output_config);
    while(1)
    }
}
```

4 Timeout timing

The receive idle timeout timing is configured in the CTRL.RXIDLETOCFG register (bit[20:18]). This bit configures the number of idle characters that must be received before the RXIDLETO flag is set. A character whose entire frame are all 1s, including the start bit, data bit, and stop bit.

4.1 What is a character

Figure 1 is a standard UART transfer data packet. It includes the start bit, data frame, parity bit, and stop bits.

Figure 1. UART transfer character packet

4.2 The effect after setting RXIDLETOCFG

In this part, we use the UART set to a baud rate of 115200 bps, 8-bit, no parity, and 1 stop bit (115200, 8n1) as an example. This setting should be a common setting in the UART application.

For 115200,8n1, the character timing is 86.81 µS = 1/ (115200/(1start+8data+0parity+1stop))

If RXIDLETOCFG is set to 000b, 1 idle character means 86.81 µS.

If RXIDLETOCFG is set to 001b, 2 idle characters mean 173.62 µS.

If RXIDLETOCFG is set to 010b, 4 idle characters mean 347.24 µS.

If RXIDLETOCFG is set to 011b, 8 idle characters mean 694.48 µS.

If RXIDLETOCFG is set to 100b, 16 idle characters mean 1,388.96 µS.

If RXIDLETOCFG is set to 101b, 32 idle characters mean 2,777.92 µS.

If RXIDLETOCFG is set to 110b, 64 idle characters mean 5,555.84 µS.

If RXIDLETOCFG is set to 111b, 128 idle characters mean 11,111.68 µS.

In the example code, we set a GPIO toggle in the UART interrupt handler. This GPIO toggles the status when the UART receives valid data and an idle timeout happens and assigns the UART interrupt handler into the SRAM to measure the timing with higher accuracy. The timeout timing with different **RXIDLETOCFG** settings is shown in the following figures.

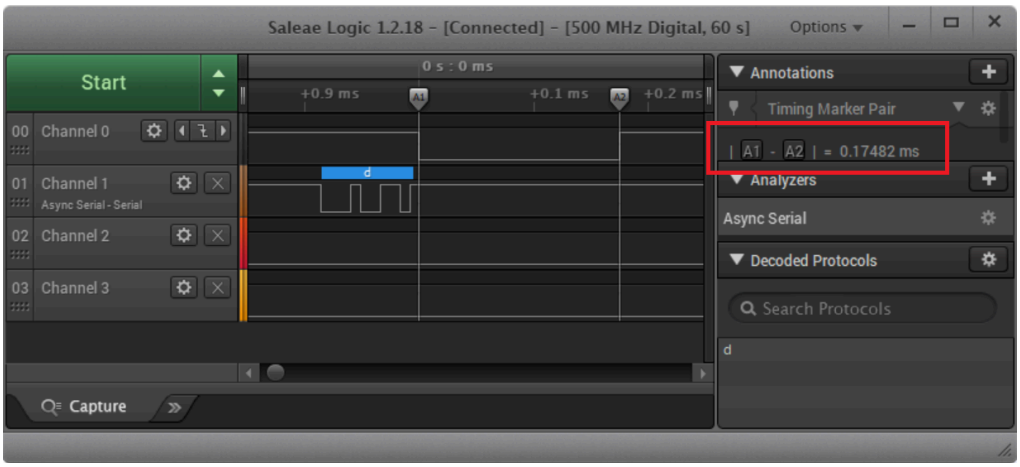

Figure 2. Receive idle timeout timing when RXIDLETOCFG is set to 000b

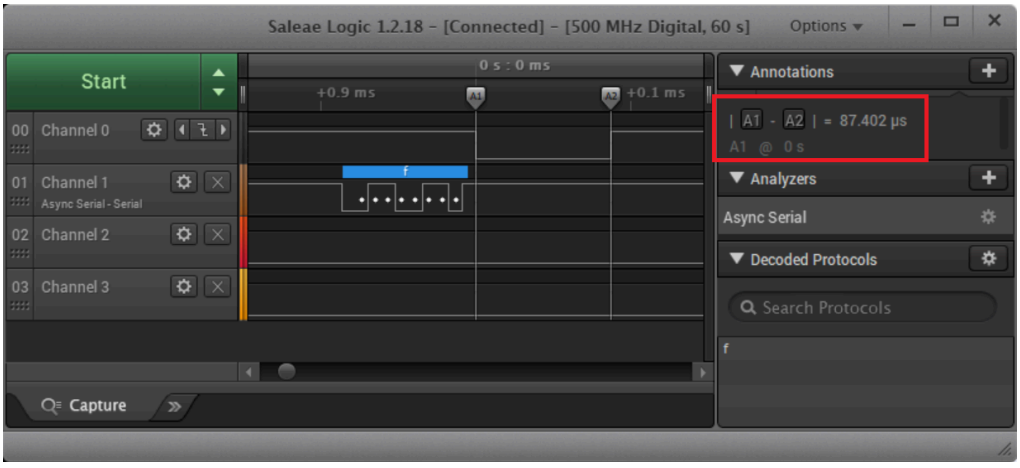

Figure 3. Receive idle timeout timing when RXIDLETOCFG is set to 001b

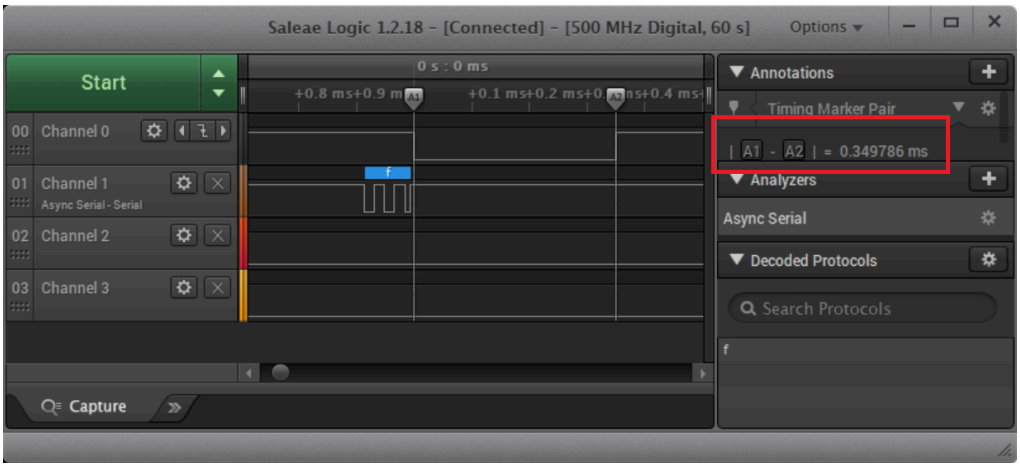

Figure 4. Receive idle timeout timing when RXIDLETOCFG is set to 010b

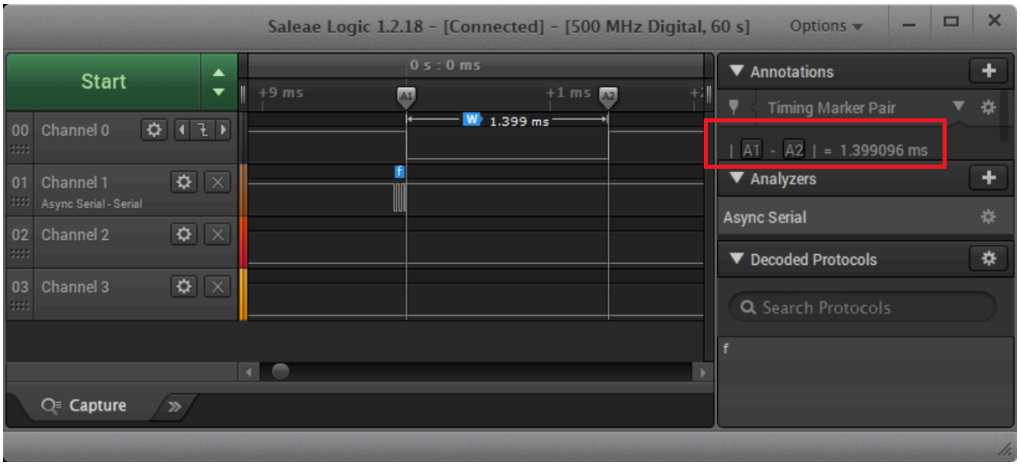

Figure 5. Receive idle timeout timing when RXIDLETOCFG is set to 011b

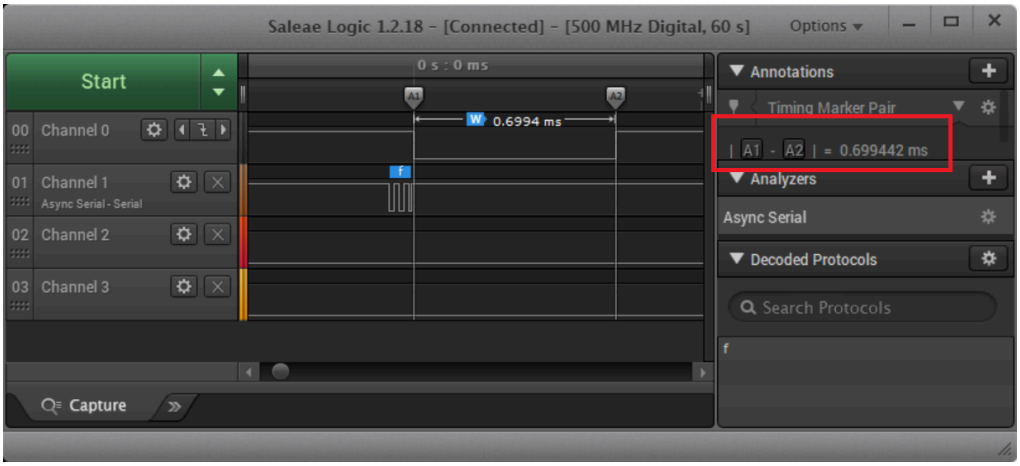

Figure 6. Receive idle timeout timing when RXIDLETOCFG is set to 100b

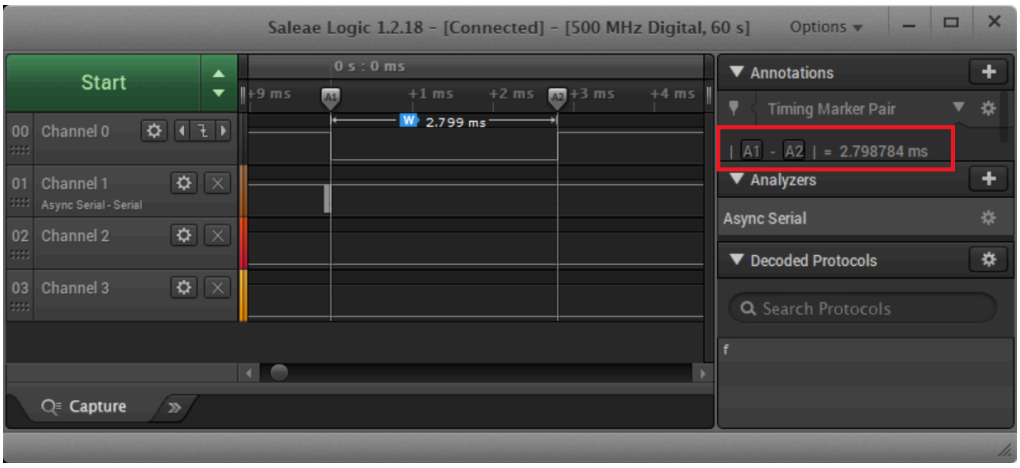

Figure 7. Receive idle timeout timing when RXIDLETOCFG is set to 101b

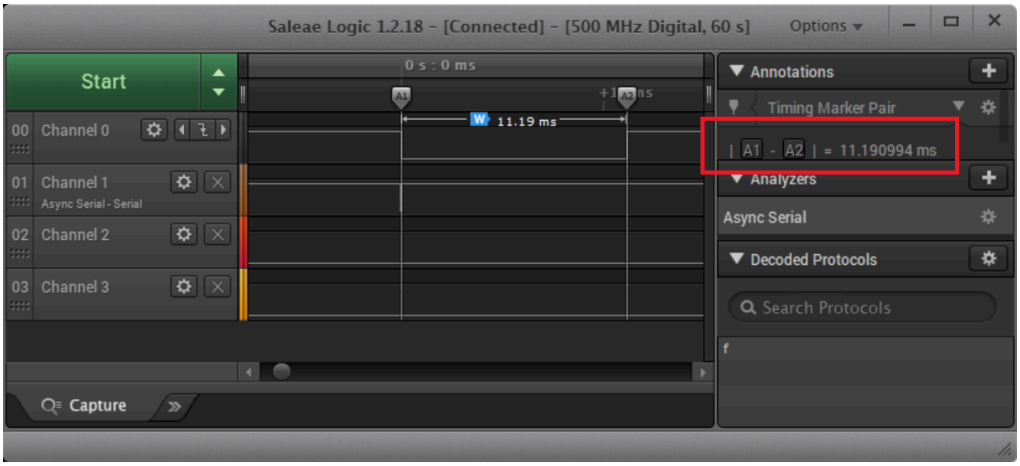

Figure 8. Receive idle timeout timing when RXIDLETOCFG is set to 110b

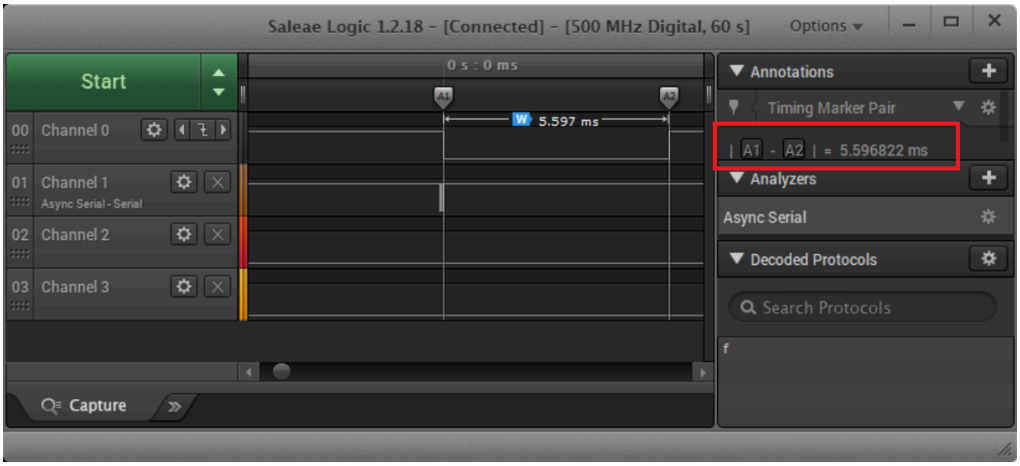

Figure 9. Receive idle timeout timing when RXIDLETOCFG is set to 111b

5 Conclusion

This application note describes the receive idle timeout feature of the LPC86x USART. With this timeout function supported, you can stop the DMA by software or receive an unknown-length data from the UART.

6 References

[1] *LPC86x User Manual* (document UM11607)

7 Revision history

Table 6 summarizes the revisions to this document.

Table 6: Revision history

Revision	Date	Substantive changes
0	10 May 2023	Initial revision.

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 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

9 Legal information

9.1 Definitions

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

9.2 Disclaimers

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.3 Trademarks

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

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

Contents

Date of release: 10 May 2023
Document identifier: AN13833