AN13686

Transmitting and Receiving Data using USART in LPC5500 EVK

Rev. 0 — 24 August 2022 Application note

Document information

Information	Content
Keywords	USART, LPC5500 EVK, DMA, Interrupt
Abstract	This application note helps you to create a DMA_TX + INT_RX example based on SDK that can be used to transmit and receive data using USART on LPC5500 EVK board.

1 Introduction

USART is one of the most popular peripherals used by customers in numerous MCU applications. The convenient way to use USART is to transmit the data with direct memory access, `DMA_TX`, and receive the data with an interrupt, `INT_RX`.

Many applications must send a relatively large amount of data and receive only a small amount. The data can be sent using the DMA, which reduces CPU loading. Whereas small data such as control commands can be received using UART. In such a case, `DMA_TX` + `INT_RX` is the most commonly used model for USART programming, but SDK does not provide such an example. SDK provides rich USART-related driver example code, as shown in Figure 1.

This application note helps you to create a `DMA_TX` + `INT_RX` example based on SDK.

SDK USART example folder location: `\SDK_2_11_1_LPCXpresso55S69\boards\lpcxpresso55s69\driver_examples\usart`

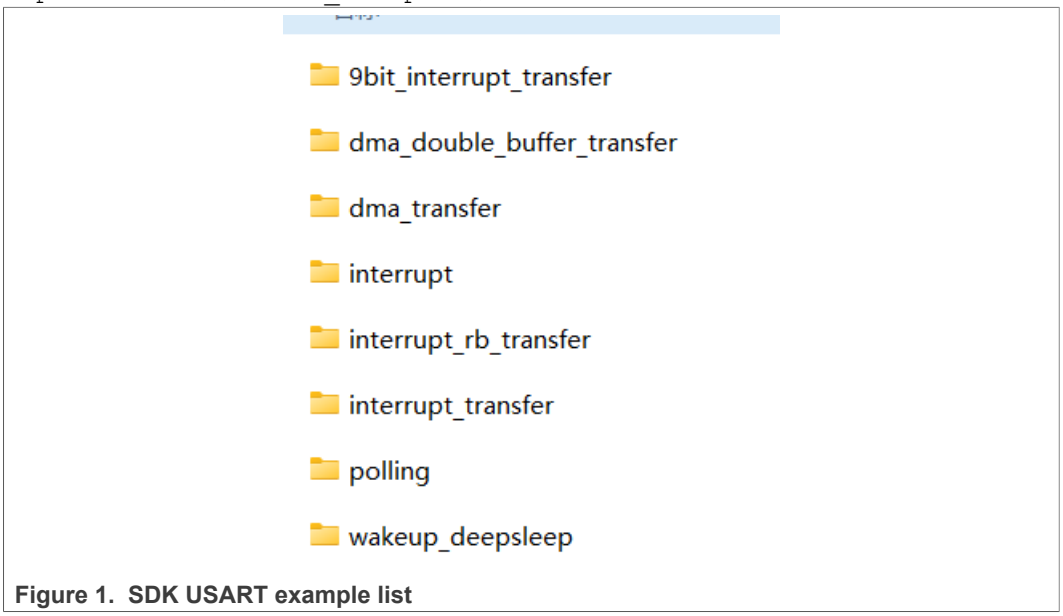

Figure 1. SDK USART example list

2 Implementation

Prerequisites

- LPC5500 EVK board
- SDK is downloaded to your local drive

To create a `DMA_TX` + `INT_RX` example based on SDK, follow the steps below:

1. Go to the folder: `\SDK_2_11_1_LPCXpresso55S69\boards\lpcxpresso55s69\driver_examples\usart`.
2. Copy the `dma_transfer` example to the same folder and rename it `dma_tx_int_rx`, as shown in Figure 2.
   Note: *Ensure that you are familiar with the `dma_transfer` example and have already run this demo on your board. The `dma_transfer` has already demoed the USART `DMA_TX` and `DMA_RX` features.*

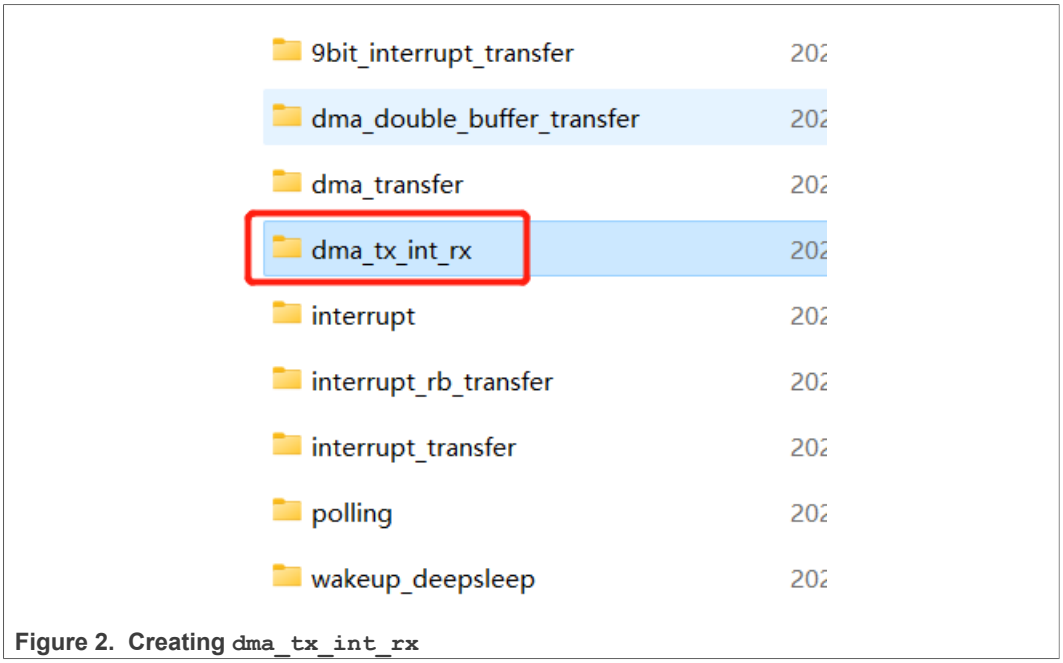

Figure 2. Creating `dma_tx_int_rx`

3. Our aim is to keep `DMA_TX` and change the `DMA_RX` to `INT_RX`. The first thing is to modify `USART_TransferCreateHandleDMA` function. Do not create `g_uartRxDmaHandle`. Pass a NULL handle to the function. Thus, the API does not initialize USART RX DMA-related stuff.

```
USART_TransferCreateHandleDMA(DEMO_USART, &g_uartDmaHandle,
 USART_UserCallback, NULL, &g_uartTxDmaHandle, NULL);
```

4. Open USART RX interrupt enable bit.

```
/* Enable RX interrupt. */
    USART_EnableInterrupts(DEMO_USART, kUSART_RxLevelInterruptEnable |
 kUSART_RxErrorInterruptEnable);
    EnableIRQ(DEMO_USART_IRQn);
```

5. To handle the interrupt, define USART hardware interrupt handler code at the application level (main.c).

```
#define DEMO_USART_IRQHandler FLEXCOMM0_IRQHandler
#define DEMO_USART_IRQn       FLEXCOMM0_IRQn
void DEMO_USART_IRQHandler(void)
{
    uint8_t data;
    /* If new data arrived. */
    if ((kUSART_RxFifoNotEmptyFlag | kUSART_RxError | kUSART_RxFifoFullFlag)
 & USART_GetStatusFlags(DEMO_USART))
    {
        data = USART_ReadByte(DEMO_USART);
        PRINTF("usart rx interrupt:%c\r\n", data);

        if(data == 's')
        {
            /* Send g_tipString out. */
            xfer.data     = g_tipString;
            xfer.dataSize = sizeof(g_tipString) - 1;
            g_uartDmaHandle.txState = kUSART_TxIdle;
            USART_TransferSendDMA(DEMO_USART, &g_uartDmaHandle, &xfer);
        }

    }
    if ((0U != (DEMO_USART->INTENSET & USART_INTENSET_TXIDLEEN_MASK)) &&
 (0U != (DEMO_USART->INTSTAT & USART_INTSTAT_TXIDLE_MASK)))
    {
        USART_TransferDMAHandleIRQ(DEMO_USART, &g_uartDmaHandle);
```

```
    }
}
```

Note: *To follow* `usart_dma driver(fsl_usart_dma.c)` *algorithm of SDK, application level code must call* `USART_TransferDMAHandleIRQ` *in USART interrupt handler. The SDK driver uses TXIDLE interrupt to notify application level code for* `DMA_TX` *complete event.*

3 Result log

Once the code is modified, you can compile, download, and run the code.

Enter any character on the PC terminal. The MCU echoes the character that you have entered. However, when you enter character 's', the MCU calls `USART_TransferSendDMA` and sends a string using DMA.

```
USART: TX DMA, RN INTERRUPT
press 's for DMA TX transit
This string is send from UART_DNA
USART_UserCallback, status:Ox1646
usart rx interrupt:a
usart rx interrupt:b
usart rx interrupt:c
usart rx interrupt:d
usart rx interrupt:s
This string is send from UART_DMA
USART_UserCallback, status:Ox1646
```

4 Source code

The updated `usart_dma_transfer.c` file is as follows:

```
/*
 * Copyright (c) 2016, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */
#include "pin_mux.h"
#include "board.h"
#include "fsl_usart.h"
#include "fsl_usart_dma.h"
#include "fsl_dma.h"
#include "fsl_debug_console.h"
#include <stdbool.h>
#include "fsl_power.h"
#define DEMO_USART                USART0
#define DEMO_USART_CLK_SRC        kCLOCK_Flexcomm0
#define DEMO_USART_CLK_FREQ       CLOCK_GetFlexCommClkFreq(0U)
#define USART_RX_DMA_CHANNEL      4
#define USART_TX_DMA_CHANNEL      5
#define EXAMPLE_UART_DMA_BASEADDR DMA0
#define DEMO_USART_IRQHandler FLEXCOMM0_IRQHandler
#define DEMO_USART_IRQn       FLEXCOMM0_IRQn
#define ECHO_BUFFER_LENGTH 8
usart_transfer_t xfer;
usart_dma_handle_t g_uartDmaHandle;
dma_handle_t g_uartTxDmaHandle;
uint8_t g_tipString[]                  = "This string is send from UART_DMA\r
\n";
#define kUSART_TxIdle 0
void USART_UserCallback(USART_Type *base, usart_dma_handle_t *handle, status_t
 status, void *userData)
{
    userData = userData;
```

```
    if (kStatus_USART_TxIdle == status)
    {
        PRINTF("USART_UserCallback, status:0x%X\r\n", status);
    }
}
void DEMO_USART_IRQHandler(void)
{
    uint8_t data;
    /* If new data arrived. */
    if ((kUSART_RxFifoNotEmptyFlag | kUSART_RxError | kUSART_RxFifoFullFlag) &
 USART_GetStatusFlags(DEMO_USART))
    {
        data = USART_ReadByte(DEMO_USART);
        PRINTF("usart rx interrupt:%c\r\n", data);

        if(data == 's')
        {
            /* Send g_tipString out. */
            xfer.data     = g_tipString;
            xfer.dataSize = sizeof(g_tipString) - 1;
            g_uartDmaHandle.txState = kUSART_TxIdle;
            USART_TransferSendDMA(DEMO_USART, &g_uartDmaHandle, &xfer);
        }

    }
    if ((0U != (DEMO_USART->INTENSET & USART_INTENSET_TXIDLEEN_MASK)) && (0U !=
 (DEMO_USART->INTSTAT & USART_INTSTAT_TXIDLE_MASK)))
    {
        USART_TransferDMAHandleIRQ(DEMO_USART, &g_uartDmaHandle);
    }
}
int main(void)
{
    usart_config_t config;
    /* set BOD VBAT level to 1.65V */
    POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv,
 false);
    /* attach 12 MHz clock to FLEXCOMM0 (debug console) */
    CLOCK_AttachClk(kFRO12M_to_FLEXCOMM0);
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();
    PRINTF("USART: TX DMA, RX INTERRUPT\r\n");
    PRINTF("press 's' for DMA TX tranmsit\r\n");

    USART_GetDefaultConfig(&config);
    config.baudRate_Bps = BOARD_DEBUG_UART_BAUDRATE;
    config.enableTx     = true;
    config.enableRx     = true;
    USART_Init(DEMO_USART, &config, DEMO_USART_CLK_FREQ);
    /* Configure DMA. */
    DMA_Init(EXAMPLE_UART_DMA_BASEADDR);
    DMA_EnableChannel(EXAMPLE_UART_DMA_BASEADDR, USART_TX_DMA_CHANNEL);
    DMA_EnableChannel(EXAMPLE_UART_DMA_BASEADDR, USART_RX_DMA_CHANNEL);
    DMA_CreateHandle(&g_uartTxDmaHandle, EXAMPLE_UART_DMA_BASEADDR,
 USART_TX_DMA_CHANNEL);
    USART_TransferCreateHandleDMA(DEMO_USART, &g_uartDmaHandle,
 USART_UserCallback, NULL, &g_uartTxDmaHandle, NULL);

    /* Send g_tipString out. */
    xfer.data     = g_tipString;
    xfer.dataSize = sizeof(g_tipString) - 1;
    USART_TransferSendDMA(DEMO_USART, &g_uartDmaHandle, &xfer);

    /* Enable RX interrupt. */
    USART_EnableInterrupts(DEMO_USART, kUSART_RxLevelInterruptEnable |
 kUSART_RxErrorInterruptEnable);
    EnableIRQ(DEMO_USART_IRQn);
    while(1);
}
```

5 Revision history

The following table summarizes the changes done to this document since the initial release.

Revision history

Rev.	Date	Description
0	24 August 2022	Initial release

6 Legal information

6.1 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.

6.2 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.

6.3 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

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

Date of release: 24 August 2022
Document identifier: AN13686