Bern University
of Applied Sciences

Exercise - 23

UART with DMA

Andreas Habegger | Adrian Steiner
BTS4230 | Version 1.0.0 of 17.03.2025

The blocking HAL UART send function is inefficient and time consuming for long messages. The CPU is busy copying data into the UART data register and waiting for the message to be transmitted. To reduce this effort, two DMAs are provided in our MCU, which can be used for this simple task instead. In this exercise we will redesign the Ex-22: UART Interrupt Transmit using DMA for data transfer.

Objectives

- DMA configuration
- UART transmit with DMA
- Interrupt priority handling

Outcomes

- Advantage of DMA transfers
- Limitations of DMA
- Using DMA for transmit messages with UART
- Interrupt prioritisation

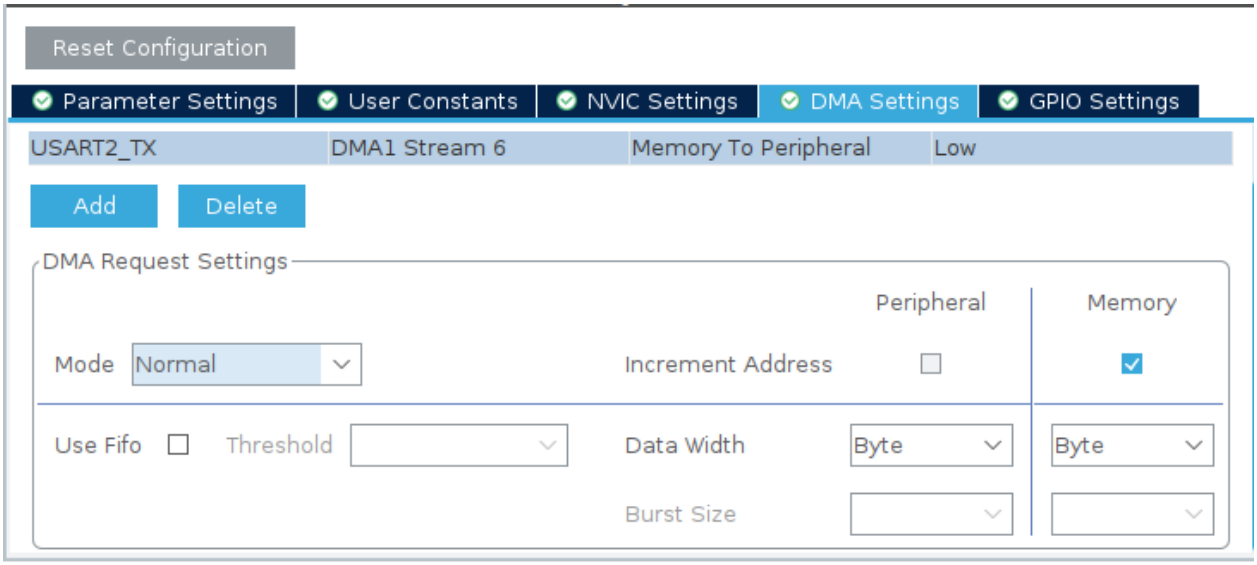
Description

With **Direct Memory Access** (DMA), memory can be copied from peripheral to memory and vice versa without any CPU effort during the data transfer. As the STM32F446re MCU uses fly-through DMA, the memory-to-memory transfer can also be performed with DMA. In this exercise, the memory to peripheral mode is used to copy the data from memory to the UART peripheral data register. The aim is to extend the Ex-22 by solving it with DMA and checking its behaviour.

Tasks

DMA and interrupt must be enabled for the USART2 peripheral to solve this exercise. The STM32CubeMX automatically uses the correct DMA and stream. See the data sheet of the MCU for detailed information about DMA and stream mapping. The mode and the data width are configurable. The mode can be set to normal mode or circular mode. For more information about these two modes see section Pointer incrementation and circular mode in RM0390 page 212. To send a message via UART, the **normal** mode is required and the data size is one byte.

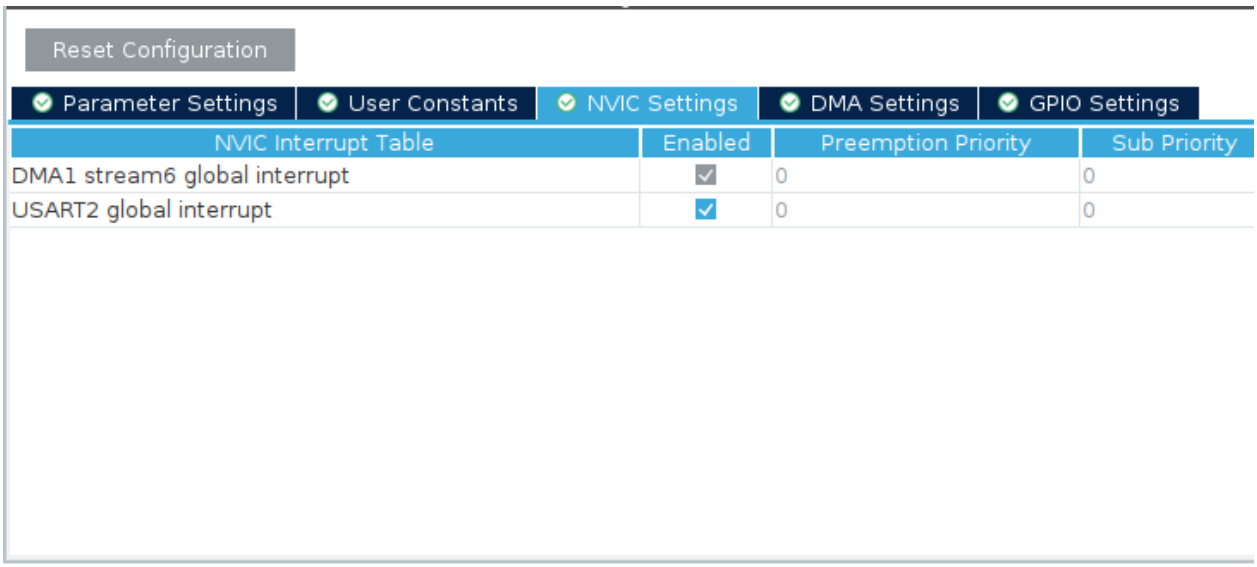
DMA settings for UART2



Enabling DMA for USART2_TX

An interrupt is used to acknowledge that the transmission has been completed. This requires the global interrupt to be enabled for the USART2.

Enable Interrupt for UART2



Enabling USART2 global interrupt

Implementation

To ensure that the configuration works, start the exercise by sending a message using UART and DMA.

Send a message with DMA in the superloop

1. Create a new project with default configuration.
2. Enable DMA and interrupt for UART2 as described in the tasks section.
3. Send a message to the computer in the superloop using the `HAL_UART_Transmit_DMA()` function.
4. Test your implementation and configuration to make sure you are getting data continuously.

Once the DMA transfer to the UART has been successfully commissioned, a message should also be sent from a basic timer interrupt.

Additional interrupt message

5. Initialise a simple timer with an interrupt with a frequency of $10\,\mathrm{Hz}$.
6. Send a message to the computer using DMA and UART.

7. Test your implementation and check your configuration by answering the following questions:

Question

a. Are you receiving a message from both tasks?

b. What is the priority of the interrupts (timer and UART)?

c. What is the return value of the send function?

The first try with DMA from the interrupt and from the superloop may not be successful. This is because the function cannot register a new address to send until it has completed the last transmission. HAL deletes the transmission in the interrupt callback `UART_DMATransmitCplt()`. To be sure that the message in the interrupt can be sent, wait until the message from the superloop has finished.

Note

As you are blocking the system in the interrupt, this is not a good solution.

Synchronisation of interrupt message

8. Set the interrupt priority of the timer interrupt lower than the UART interrupt.
9. Wait in the superloop until the message can be sent (check the return value).
10. Test your implementation and check your configuration by answering the following questions:

Question

a. Are you receiving the messages from both tasks?

b. Does this implementation work forever? (Let it run for at least a minute)

c. What's wrong with this implementation?

Hint

Take a look into the function `HAL_UART_Transmit_DMA()`. When is the UART instance set to busy? Is it possible to interrupt between the setting of the busy status and the start of the DMA process by the timer? What effect does this situation have?

d. How would you improve it?