SPI_DMA_1
for KIT_AURIX_TC275_LK
SPI data communication via DMA

AURIX™ TC2xx Microcontroller Training
V1.0.0

Scope of work

QSPI is used to conduct SPI Master Slave communication using the DMA module.

This example implements an SPI full duplex communication.
QSPI1 is configured as an SPI master and QSPI3 is configured as an SPI slave. Both master and slave exchange eight bytes of data.
Four DMA channels are used to enable data transfer between RAM and QSPI FIFOs without CPU intervention:

- DMA channel 1 is configured as SPI master Tx
- DMA channel 2 is configured as SPI master Rx
- DMA channel 3 is configured as SPI slave Tx
- DMA channel 4 is configured as SPI slave Rx

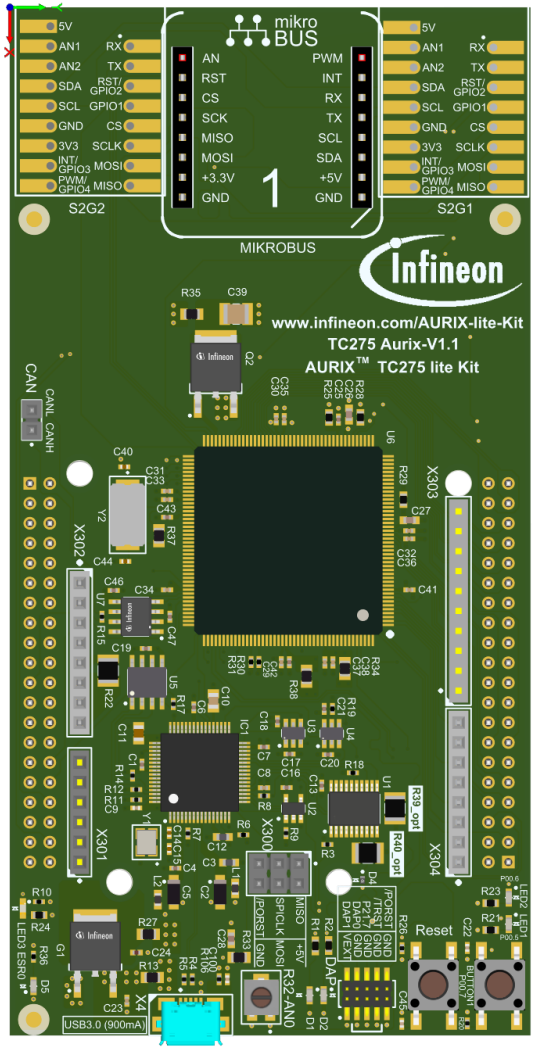
An LED is used to signal the successful data communication.

Introduction

- The **Q**ueued **S**ynchronous **P**eripheral **I**nterface (QSPI) enables any synchronous serial communication with external devices based on the standardized SPI-bus signals: clock, data-in, data-out and slave select
- The QSPI works in full duplex mode either as Master or Slave with up to 50 Mbit/s
- The DMA module channels can be configured to transfer data from/to QSPI FIFOs to/from internal RAM Memory without any CPU intervention
- This example is based on the Infineon Low Level Drivers to demonstrate SPI Master Slave Communication with minimum CPU intervention

Hardware setup

This code example has been developed for the board KIT_AURIX_TC275_LITE.

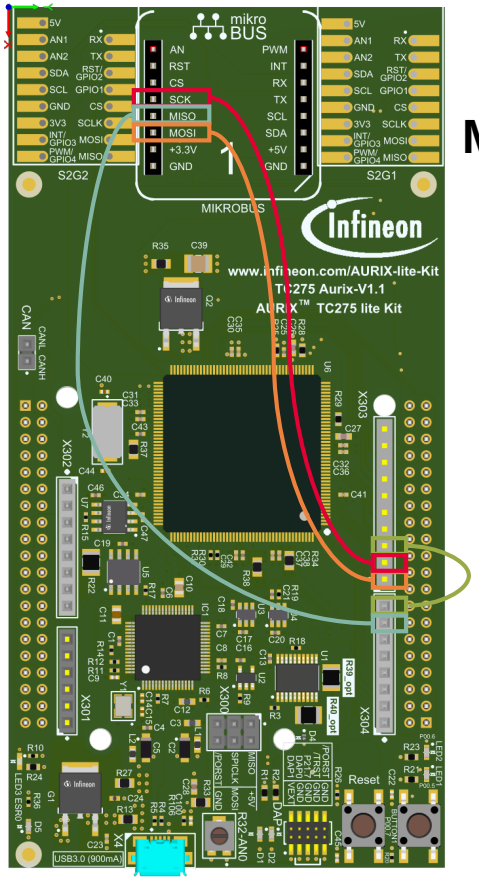


Hardware Setup

› Connect following pins as described and illustrated using wires

Slave:

MikroBus™ Left Conn.	
AN26	1
RST - P10.6	2
CS (OUT) - P10.0	3
SCK - P10.2	4
MISO - P10.1	5
MOSI - P10.3	6
VEXT - +3.3V	7
GND	8



Master (+CS):

X303

10	P13.1 / P40.1 - SCL0
9	P13.2 / P40.0 - SDA0
8	VAREF2
7	GND
6	P10.2 - SPICLK
5	P10.1 - MISO
4	P10.3 - MOSI
3	P10.5
2	P02.7
1	P02.6

X304

8	P02.4
7	P02.5
6	P02.3
5	P10.4
4	P02.1
3	P02.0
2	P15.2 - UART_TX
1	P15.3 - UART_RX

QSPI1 (Master)	WIRE	QSPI3 (Slave)
P10.2 : SCLKO	←→	P02.7 : SCLKA
P10.5 : SLSO_9	←→	P02.4 : SLSI_A
P10.1 : MRST_A	←→	P02.5 : MRST
P10.3 : MTSR	←→	P02.6 : MTSR_A

Implementation

Configuring the SPI communication

The configuration of the SPI communication is done through the function ***initQSPI()*** in two different steps:

- QSPI Slave initialization
- QSPI Master initialization

QSPI Slave initialization

- The initialization of the QSPI slave module is done by defining an instance of the ***IfxQspi_SpiSlave_Config*** structure
- The structure is filled with default values by the function ***IfxQspi_SpiSlave_initModuleConfig()***
- Afterwards, the following parameters are modified to enable the DMA usage, set its channels, interrupt priorities and IO port pins:
  - DMA configuration: ***dma.useDma***, ***dma.txDmaChannelId***, ***dma.rxDmaChannelId***
  - Interrupts configuration: ***base.txPriority***, ***base.rxPriority***, ***base.erPriority,*** ***base.isrProvider***
  - Pins configuration: ***pins***

Implementation

QSPI Slave initialization (Cont.)

- The function ***IfxQspi_SpiSlave_initModule()*** is used to initialize the QSPI slave module
- Finally, the buffers used by the QSPI slave are initialized

The functions needed to initialize the QSPI Slave can be found in the iLLD header ***IfxQspi_SpiSlave.h***.

QSPI Master initialization

- The initialization of the QSPI master module is done by defining an instance of the ***IfxQspi_SpiMaster_Config*** structure
- The structure is filled with default values by the function ***IfxQspi_SpiMaster_initModuleConfig()***
- Afterwards, the following parameters are modified to enable the DMA usage, set its channels, interrupt priorities and IO port pins:
  - DMA configuration: ***dma.useDma***, ***dma.txDmaChannelId***, ***dma.rxDmaChannelId***
  - Interrupts configuration: ***base.txPriority***, ***base.rxPriority***, ***base.erPriority,*** ***base.isrProvider***
  - Pins configuration: ***pins***

Implementation

QSPI Master initialization (Cont.)

- The function ***IfxQspi_SpiMaster_initModule()*** is used to initialize the QSPI master module
- A QSPI module controls 16 communication channels, which are individually programmable. In this example, the function ***initQSPI1MasterChannel()*** initializes the channel 9 using an instance of the structure ***IfxQspi_SpiMaster_ChannelConfig***. Afterwards, the slave select channel number is set through the parameter ***sls.output*** and the baud rate is modified via the parameter ***base.baudrate***
- The function ***IfxQspi_SpiMaster_initChannel()*** is used to initialize the QSPI master channel
- Finally, the buffers used by the QSPI master are initialized

The functions needed to initialize the QSPI Master can be found in the iLLD header ***IfxQspi_SpiMaster.h***.

Implementation

Interrupt Service Routines (ISR):

- The following ISRs are implemented to ensure a proper SPI communication in DMA mode:
  - SPI Master error interrupt ***QSPI1ErrorISR()*** ISR calls the function:
    - ***IfxQspi_SpiMaster_isrError()***
  - SPI Slave error interrupt ***QSPI3ErrorISR()*** ISR calls the function:
    - ***IfxQspi_SpiSlave_isrError()***
  - SPI Master transmit interrupt ***DMAChn1ISR()*** ISR calls the function:
    - ***IfxQspi_SpiMaster_isrDmaTransmit()***
  - SPI Master receive interrupt ***DMAChn2ISR()*** ISR calls the function:
    - ***IfxQspi_SpiMaster_isrDmaReceive()***
  - SPI Slave transmit interrupt ***DMAChn3ISR()*** ISR calls the function:
    - ***IfxQspi_SpiSlave_isrDmaTransmit()***
  - SPI Slave receive interrupt ***DMAChn4ISR()*** ISR calls the function:
    - ***IfxQspi_SpiSlave_isrDmaReceive()***

- The functions listed above can be found in the iLLD headers ***IfxQspi_SpiMaster.h*** and ***IfxQspi_SpiSlave.h***

Implementation

SPI Master Slave Communication:

The SPI Master Slave communication is established through the following steps:

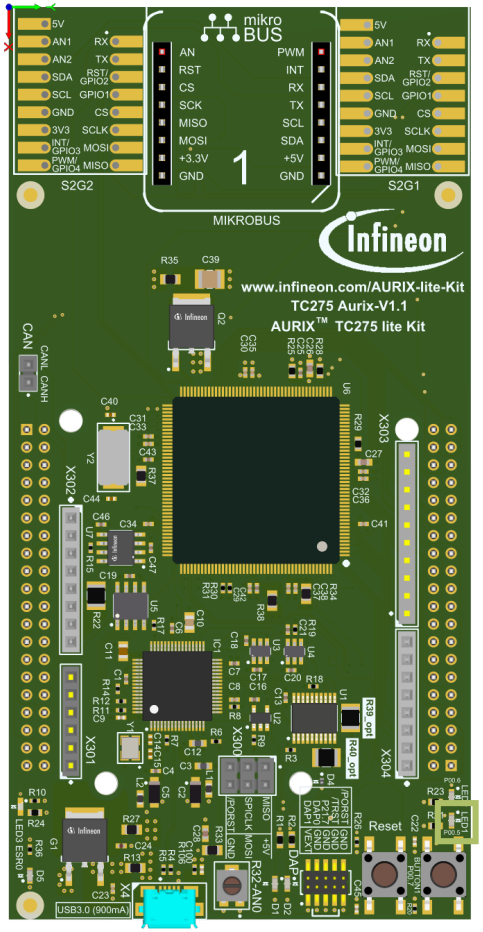
- Enable SPI Slave for data communication using the function:
  IfxQspi_SpiSlave_exchange()
- Enable and Start SPI Master data communication using the function:
  IfxQspi_SpiMaster_exchange()
- Poll for SPI slave data reception using the function:
  IfxQspi_SpiSlave_getStatus()
- The received and transmitted data are compared byte by byte and the number of errors are counted

Run and Test

After code compilation and flashing the device, perform the following steps:

- Run the project and check if the LED1 is on.
  - Data transmitted without errors

- Additionally, using the debugger, the behavior can be checked:
  - Add ***g_qspiDma*** to Watch window
  - Check if ***g_qspiDma.qspiBuffer.spiSlaveRxBuffer*** and ***g_qspiDma.qspiBuffer.spiMasterRxBuffer*** are the same as ***g_qspiDma.qspiBuffer.spiMasterTxBuffer*** and, respectively, ***g_qspiDma.qspiBuffer.spiSlaveTxBuffer***



References

- AURIX™ Development Studio is available online:
- https://www.infineon.com/aurixdevelopmentstudio
- Use the *„Import...“* function to get access to more code examples.

- More code examples can be found on the GIT repository:
- https://github.com/Infineon/AURIX_code_examples

- For additional trainings, visit our webpage:
- https://www.infineon.com/aurix-expert-training

- For questions and support, use the AURIX™ Forum:
- https://www.infineonforums.com/forums/13-Aurix-Forum

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-06
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
SPI_DMA_1_KIT_TC275_LK

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie") .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.