# SPI_CPU_1
# for KIT_AURIX_TC334_LK
## SPI communication via QSPI

AURIX™ TC3xx Microcontroller Training
V1.0.0

# Scope of work

**A QSPI module configured as SPI master sends five bytes to another QSPI module which is configured as SPI slave.**

QSPI2 is configured in master mode and used to send five bytes to QSPI4 configured in slave mode. The received data is read by the CPU and compared against the transmitted data. Port pin 00.5, to which LED1 is connected, indicates the successful transfer.

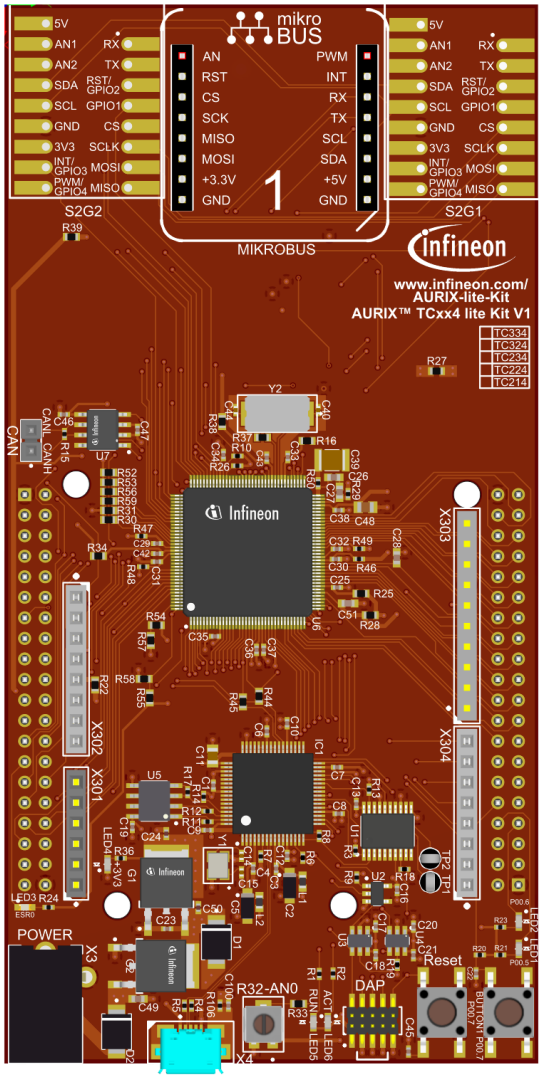
# Introduction

- The **Q**ueued **S**ynchronous **P**eripheral **I**nterface (QSPI) enables 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

# Hardware setup

This code example has been developed for the board KIT_A2G_TC334_LITE.



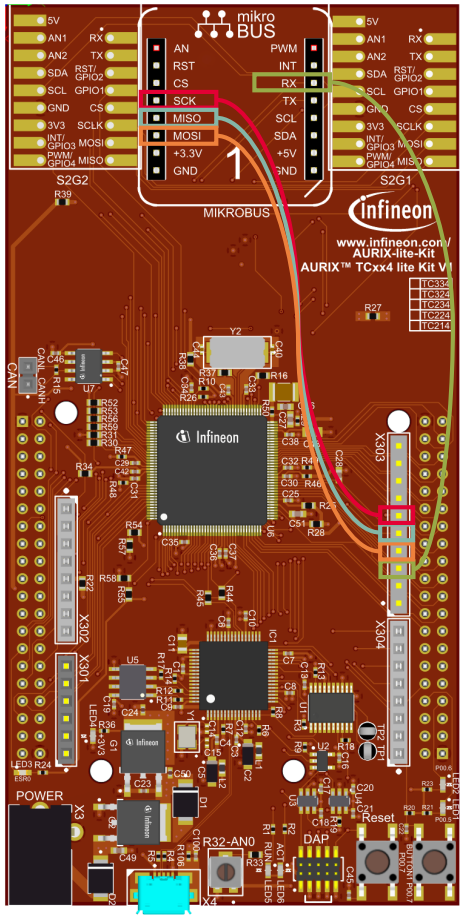
# Hardware Setup

› Connect following pins as described and illustrated using wires

**Slave:**

**MikroBus™ Left Conn.**

| | |
|---|---|
| AN34 | 1 |
| RST - P10.6 | 2 |
| CS (OUT) - P20.6 | 3 |
| SCK - P15.8 | 4 |
| MISO - P15.7 | 5 |
| MOSI - P15.6 | 6 |
| +3.3V | 7 |
| GND | 8 |




**MikroBus™ Right Conn.**

| | |
|---|---|
| 16 | P2.8 - PWM |
| 15 | P15.4 - INT |
| 14 | P15.1 - RX |
| 13 | P15.0 - TX |
| 12 | P13.1 |
| 11 | P13.2 |
| 10 | +5V |
| 9 | GND |

| QSPI1 (Master) | WIRE | QSPI2 (Slave) |
|---|---|---|
| P10.2 : SCLKO | ←→ | P15.8 : SCLKI |
| P10.5 : SLSO_9 | ←→ | P15.1 : SLSI_B |
| P10.1 : MRST_A | ←→ | P15.7 : MRST |
| P10.3 : MTSR | ←→ | P15.6 : MTSR_B |

**Master:**

**X303**

| | |
|---|---|
| 10 | P13.1 |
| 9 | P13.2 |
| 8 | VAREF |
| 7 | GND |
| 6 | P10.2 - SPICLK |
| 5 | P10.1 - MISO |
| 4 | P10.3 - MOSI |
| 3 | P10.5 |
| 2 | P02.7 |
| 1 | P02.6 |

# Implementation

## Configuring the SPI communication

The configuration of the SPI communication is done once in the setup phase 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 pins, ISR service provider and the priorities are set
- The function ***IfxQspi_SpiSlave_initModule()*** is used to initialize the QSPI slave module
- Additionally, the buffers used by the QSPI slave are initialized

The above functions can be found in the iLLD header ***IfxQspi_SpiSlave.h***.

# Implementation

## 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 interface operation mode, the pins, ISR service provider and the priorities are set
- 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 ***initQSPI2MasterChannel()*** 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
- Additionally, the buffers used by the QSPI master are initialized

The above functions can be found in the iLLD header ***IfxQspi_SpiMaster.h***.

# Implementation

## QSPI Master - Slave communication

- The function ***transferData()*** triggers the data transfer between the SPI-Master and the SPI-Slave
- The functions ***IfxQspi_SpiSlave_getStatus()*** and ***IfxQspi_SpiMaster_getStatus()*** are used to check the status of the master and the slave in order to delay the transfer until both are free
- The function ***IfxQspi_SpiSlave_exchange()*** instructs the slave to receive a data stream of predefined length
- The function ***IfxQspi_SpiMaster_exchange()*** is called in order to instruct the master to send the data
- Finally, the function ***verifyData()*** checks if the data received by the Slave matches the data sent by the Master
- If no errors have occured during the communication, the LED1, connected to port pin 00.5, is turned on to signal that the transmission was successful

# Implementation

**Configure and control the LEDs**

The LED is turned on and off by **controlling the port pin** to which it is connected using methods from the iLLD headers ***IfxPort.h***.

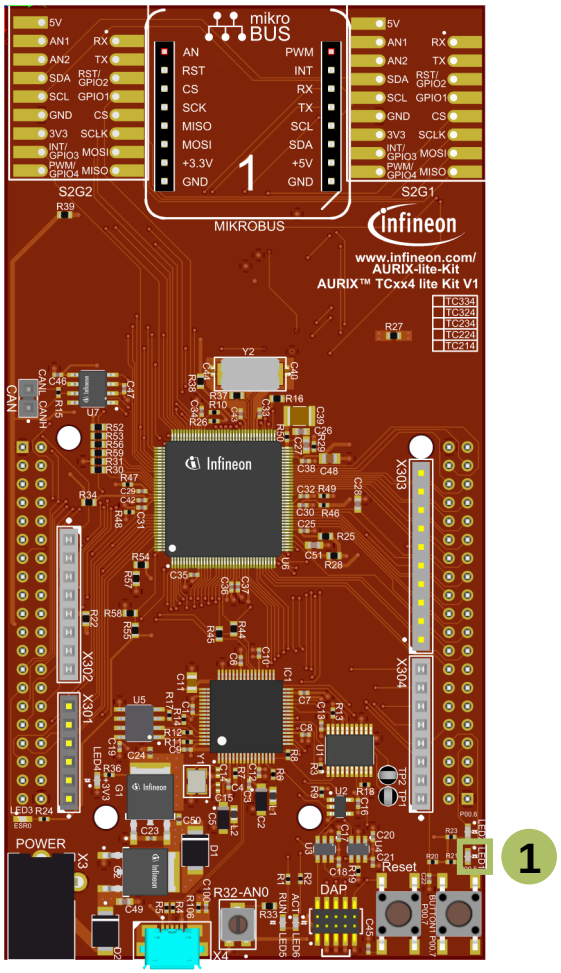
The LED port pin is **configured to output push-pull mode** using the function ***IfxPort_setPinModeOutput()***.

During program execution, the LED is **switched on and off** using the functions ***IfxPort_setPinLow()*** and ***IfxPort_setPinHigh()***.

# Run and Test

After code compilation and programming the device, start a debug session and perform the following steps:

- Set a breakpoint to ***transferData()*** in the ***Cpu0_main.c*** and check the ***spiMasterTxBuffer*** and ***spiSlaveRxBuffer*** inside ***spiBuffers*** structure
- Run the code example and check if the LED1 (1) is on (Data transmitted without errors)
- The ***spiMasterTxBuffer*** and ***spiSlaveRxBuffer*** should now show the same transmitted and received data
- Remove a cable (e.g. SCLKx), perform a Reset and re-run the application to see that the data transmission is interrupted and the LED1 (1) is off (Data transmission blocked)




**Note:** when checking the buffers' data, the debug session must be paused.

# 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 2022-03**
**Published by**
**Infineon Technologies AG**
**81726 Munich, Germany**



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

**Document reference**
**SPI_CPU_1_KIT_TC334_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.