... the world's most energy friendly microcontrollers

USART - Synchronous mode (SPI)

AN0008 - Application Note

This application note describes how to use the EFM32 USART in synchronous (SPI) mode.

This application note includes:

- This PDF document
- Source files (zip)
  - Example C-code
  - Multiple IDE projects

1 USART

1.1 Introduction

The EFM32 Universal Synchronous Asynchronous serial Receiver and Transmitter (USART) is a very flexible serial communication module. It operates in either synchronous or asynchronous mode.

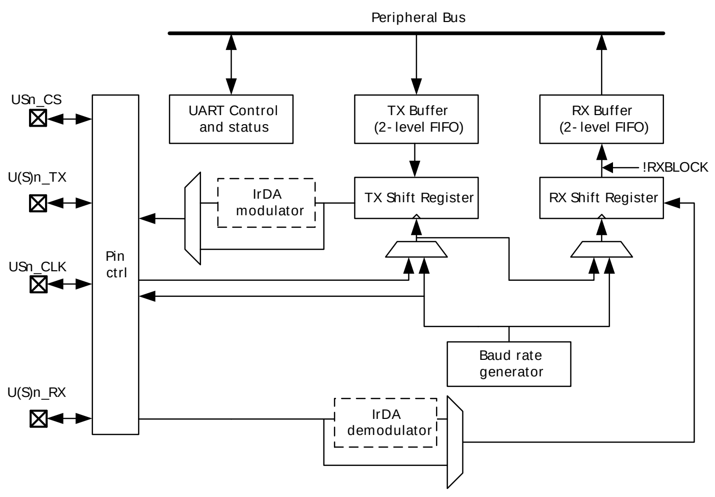
In synchronous mode, a separate clock signal is transmitted with the data. This clock signal is generated by the bus master, and both the master and slave sample and transmit data according to this clock. Both master and slave modes are supported by the USART. The synchronous communication mode is compatible with the Serial Peripheral Interface Bus (SPI) standard.

In asynchronous mode, no separate clock signal is transmitted with the data on the bus. The USART receiver thus has to determine when to sample the data on the bus. To make this possible, additional synchronization bits are added to the data when operating in asynchronous mode, resulting in a slight overhead.

1.2 Operation Overview

An overview of the USART is shown in Figure 1.1 (p. 2) .

Figure 1.1. USART Overview



Transmission is enabled by writing to the TXEN bit in the USARTn_CMD register. Any data written to the TX buffer will be transmitted. The USART performs this by first moving the data to the shift register, from which the data is shifted to the TX pin. When data has been moved from the TX buffer, the TXBL bit in the USARTn_STATUS register is set, indicating that a new transmit byte may be written. When all available transmit data (both in the shift register and in the TX buffer) have been transmitted, the TXC bit in the USARTn_STATUS register is set. Both the TXC and TXBL are also available as interrupt flags.

When the RXEN bit in the USARTn_CMD register has been written, data received on the RX pin will be accepted. When new data is available from in the RX buffer, the RXDATAV bit in the USARTn_STATUS register is set. This bit will be cleared when the RX buffer is read. The RXDATAV is also available as an interrupt.

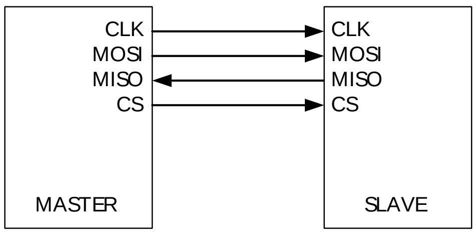
For a detailed description of the USART please refer to the specific product family reference manual.

2 Synchronous (SPI) Mode

2.1 Theory

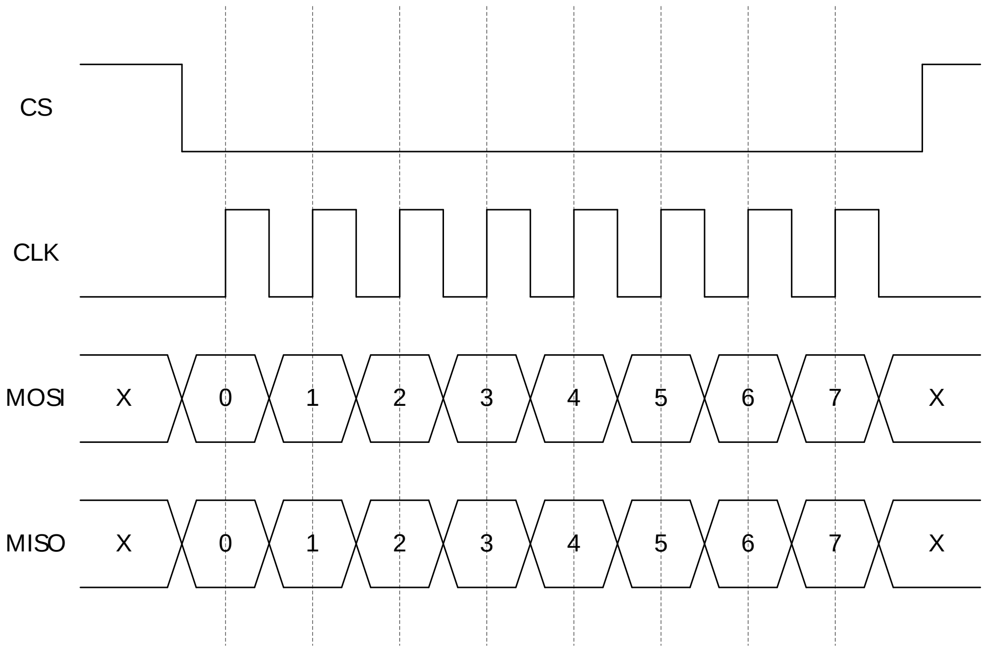
Synchronous communication is normally set up using 4 lines: clock (CLK), data in and out (MISO and MOSI), and chip select (CS, also known as slave select, SS). See Figure 2.1 (p. 3). The data lines consist of MOSI (Master Out Slave In) and MISO (Master In Slave Out). These lines are driven by the master and the slave, respectively.

Figure 2.1. Typical SPI Setup



A typical synchronous transaction is shown in Figure 2.2 (p. 3). The master initiates a transfer by asserting the slave select line. Then the clock is driven by the master, and both master and slave shift data onto their output lines while sampling the other (i.e., the slave drives the MISO line while listening to the MOSI line). This enables data transmission in both directions simultaneously. When the master has received/sent the desired data, it terminates the transaction by deasserting the CS line.

Figure 2.2. Typical SPI Transaction



In this example, the USART is configured to set up the data on negative clock edges, and sample data on positive clock edges.

The polarity of the signals as well as the sampling instant may be changed. Please see the product family Reference Manual for further details.

2.2 SPI Example

The attached SPI example illustrates the use of the USART in synchronous mode. USART1 is configured as slave, whereas USART2 is master. The following transactions take place:

- Data transmission from master to slave.
- Data transmission from slave to master.
- Data transmission from master to slave and from slave to master simultaneously.

In order to make the attached SPI example work as intended, the IO lines of USART1 and 2 must be connected as specified in Table 2.1 (p. 4). If the EFM32WG Development Kit (DVK) is used, these connections are easy to implement on the Prototyping Board. USART1 uses IO location 1 for its pins, whereas USART2 uses IO location 0.

Table 2.1. SPI Connection Table

Function	USART1 #1	Protoboard Pin	USART2 #0	Protoboard Pin
MOSI	PD0	P5.3	PC2	P4.5
MISO	PD1	P5.4	PC3	P4.6
CLK	PD2	P5.5	PC4	P4.7
CS	PD3	P5.6	PC5	P4.8

3 Revision History

3.1 Revision 1.13

2013-09-03

New cover layout

3.2 Revision 1.12

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

3.3 Revision 1.11

2012-11-12

Adapted software projects to new kit-driver and bsp structure.

3.4 Revision 1.10

2012-07-16

Removed chapter on asynchronous mode. Asynchronous mode USART/UART communciation is now covered in AN0045.

Changed name to reflect that this application note now is focused on SPI.

Some minor corrections.

3.5 Revision 1.06

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

Fixed a pin configuration issue with the uart example.

3.6 Revision 1.05

2012-03-14

Removed AF-Pin definitions from code example.

Fixed makefile-error for CodeSourcery projects.

3.7 Revision 1.04

2011-08-19

Corrected GPIO configuration for SPI slave.

3.8 Revision 1.03

2011-03-24

Changed the location description for USART1.

3.9 Revision 1.02

February 1st, 2011.

Changed GPIO configuration to reduce code size.

3.10 Revision 1.01

November 16th, 2010.

Changed example folder structure, removed build and src folders.

Added chip-init function.

Small code changes in spi.c

3.11 Revision 1.00

September 20th, 2010.

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

A.2 Trademark Information

B Contact Information

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

List of Tables

silabs.com

ZERO
ARM Cortex-M0+

TINY
ARM Cortex-M3

GECKO
ARM Cortex-M3

LEOPARD
ARM Cortex-M3

GIANT
ARM Cortex-M3

WONDER
ARM Cortex-M4