F2833x Serial Peripheral Interface

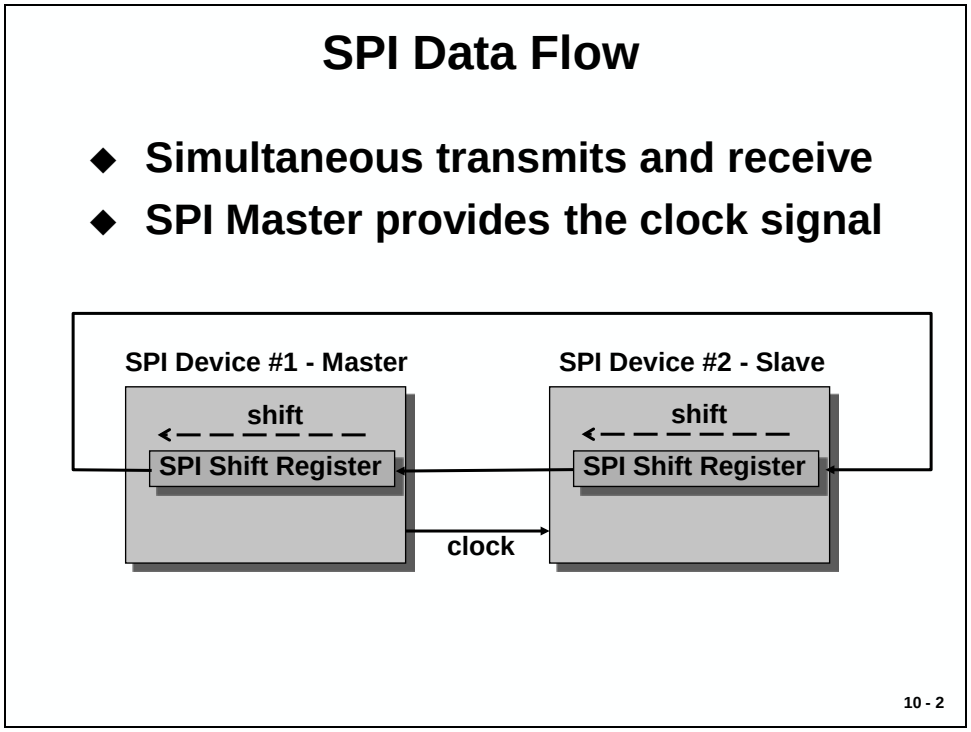
Introduction

The TMS320F2833x contains built-in features that allow several methods of communication and data exchange between the F2833x and other devices. In the previous chapter we discussed the asynchronous UART interface SCI. Chapter 10 introduces a first synchronous interface, the Serial Peripheral Interface (SPI). More synchronous interface techniques, such as McBSP and CAN, will be discussed in later chapters.

The SPI module is a synchronous serial I/O port that shifts a serial bit stream of variable length and data rate between the 'F2833x' and other peripheral devices. Here "synchronous" means that the data transmission is synchronized to a clock signal.

During data transfers, one SPI device must be configured as the transfer MASTER, and all other devices configured as SLAVES. The master drives the transfer clock signal for all SLAVES on the bus. SPI communication can be implemented in any of three different modes:

- MASTER sends data, SLAVES send dummy data
- MASTER sends data, one SLAVE sends data
- MASTER sends dummy data, one SLAVE sends data



Module Topics

F2833x Serial Peripheral Interface 10-1

Introduction 10-1

Module Topics 10-2

Serial Peripheral Interface (SPI) - Overview 10-3

SPI Data Transfer 10-4

SPI Register Set 10-5

SPI Confguration Control Register - SPICCR 10-6
SPI Operation Control Register - SPICTL 10-6
SPI Receive Emulation Buffer Register - SPIRXEMU 10-6
SPI Baud Rate Register - SPIBRR 10-7
SPI Status Register - SPISTS 10-7
SPI FIFO Transmit Register 10-8

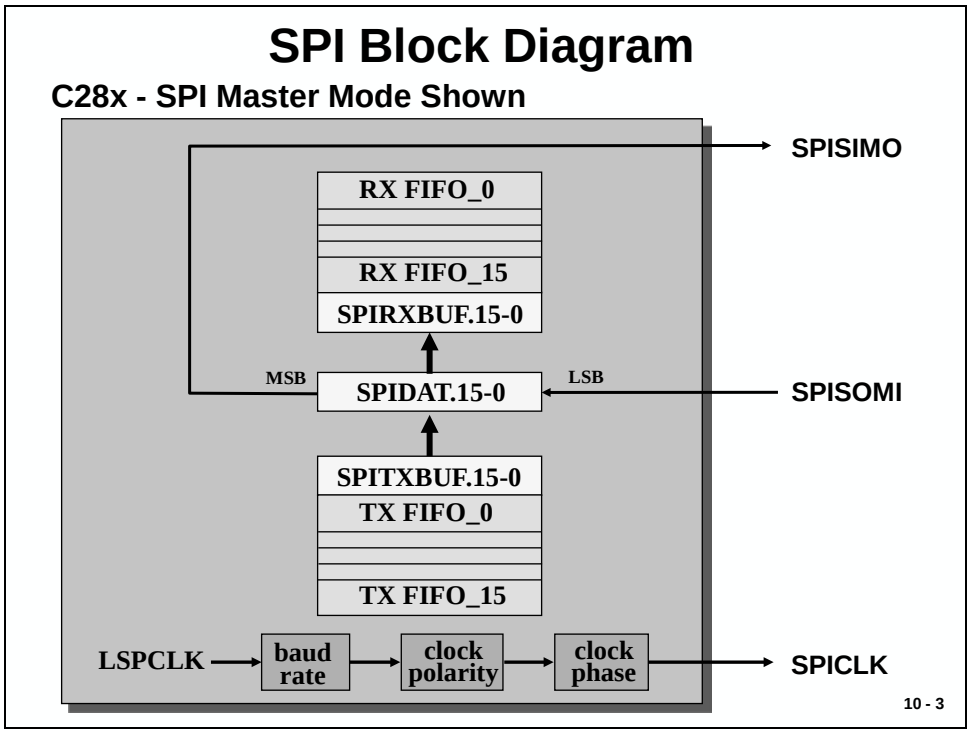
SPI Summary 10-9

SPI Lab Exercises 10-9

Serial Peripheral Interface (SPI) - Overview

In its simplest form, the SPI can be thought of as a programmable shift register. Data bits are shifted in and out of the SPI through the SPIDAT register. Two more registers configure the programming interface. To transmit a data frame, we have to write the 16-bit message into the SPITXBUF buffer. A received frame will be read by the SPI directly into the SPIRXBUF buffer. For our lab exercises, this means we write directly to SPITXBUF and we read from SPIRXBUF.

There are two operating modes for the SPI: “basic mode” and “enhanced FIFO-buffered mode”. In “basic mode”, a receive operation is double-buffered, that is the CPU need not read the current received data from SPIRXBUF before a new receive operation can be started. However, the CPU must read SPIRXBUF before the new operation is complete or a receiver-overrun error will occur. Double-buffered transmit is not supported in this mode; the current transmission must be complete before the next data character is written to SPITXDAT or the current transmission will be corrupted. The Master can initiate a data transfer at any time because it controls the SPICLK signal.

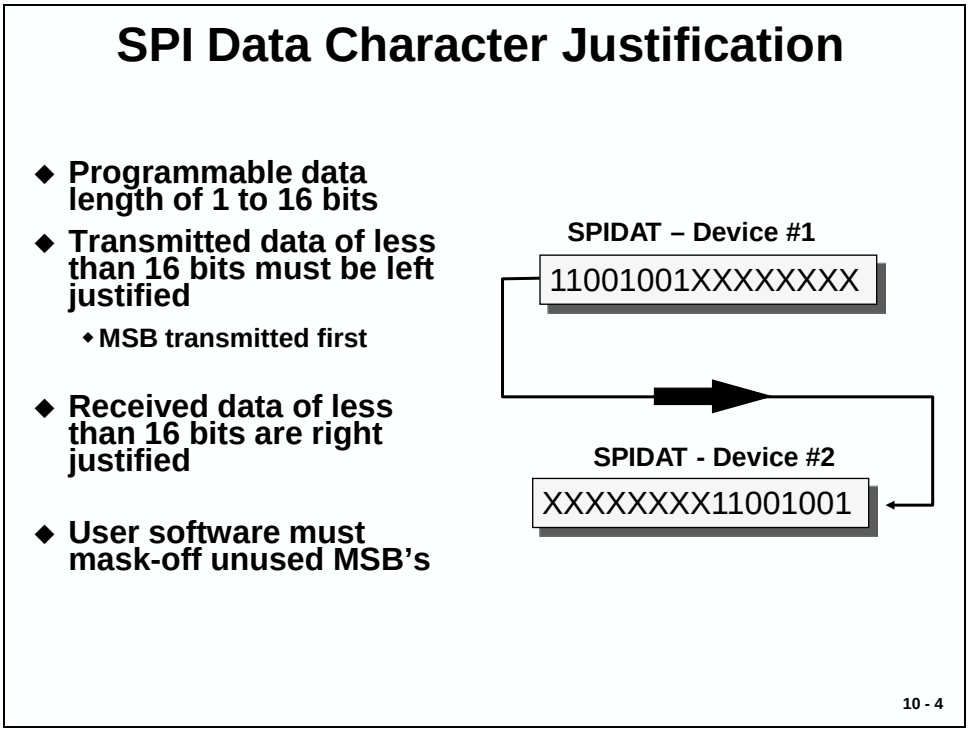


In “enhanced FIFO - buffered mode” we can build up to 16 levels of transmit and receive FIFO memory. Again, our program interfaces to the SPI unit are the registers SPITXBUF and SPIRXBUF. This expands the SPI’s buffer capacity for receive and transmit by up to 16 times. In this mode we are also able to specify an interrupt level that depends on the filled state of the two FIFOs.

SPI Data Transfer

As you can see from the previous slide, the SPI master is responsible for generating the data rate of the communication. Beeing derived from the internal low speed clock prescaler (LSPCLK), we can specify an individual baud rate for the SPI. Because not all SPI devices are interfaced in the same way, we can adjust the shape of the clock signal by two more bits, “clock polarity” and “clock phase”. Strictly speaking, the SPI is not a standard; slave devices such as EEPROMs, DACs, ADCs, Real-time clocks and temperature sensors do have different requirements for the interface timing. For this reason, TI includes options to adjust the SPI timing.

Data transmission always starts with the MSB (most significant bit) out of SPIDAT first and received data will be shifted into the device, also with MSB first. Both transmitter and receiver perform a left shift with every SPI clock period. For frames of less than 16 bits, data to be transmitted must be left justified before transmission starts. Received frames of less than 16 bits must be masked by user software to suppress unused bits.



SPI Register Set

The next slide summarizes all SPI control registers. Some of the devices in the C2000 family feature more than one SPI channel, numbered A, B, C and so on. Therefore the register names of the first SPI are expanded with an 'A'.

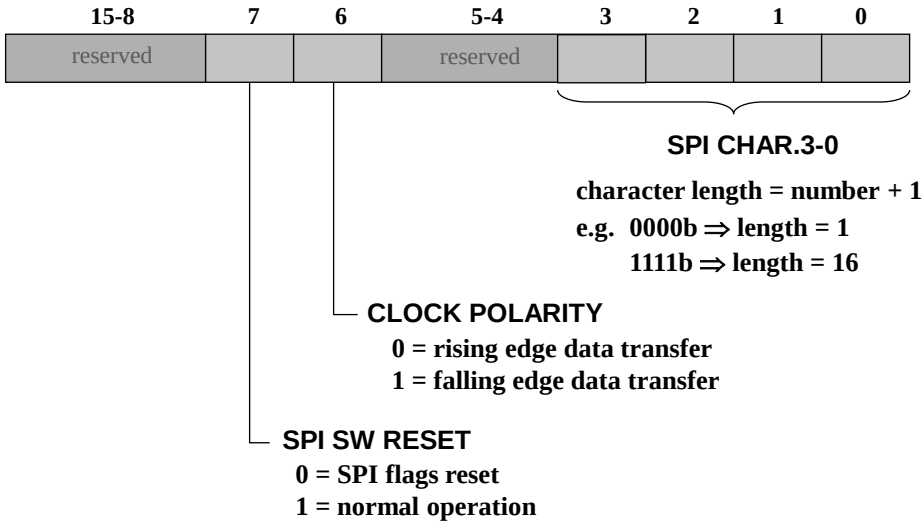
SPI-A Registers

Address	Register	Name
0x007040	SPICCR	SPI-A configuration control register
0x007041	SPICTL	SPI-A operation control register
0x007042	SPISTS	SPI-A status register
0x007044	SPIBRR	SPI-A baud rate register
0x007046	SPIRXEMU	SPI-A receive emulation buffer register
0x007047	SPIRXBUF	SPI-A serial receive buffer register
0x007048	SPITXBUF	SPI-A serial transmit buffer register
0x007049	SPIDAT	SPI-A serial data register
0x00704A	SPIFFTX	SPI-A FIFO transmit register
0x00704B	SPIFFRX	SPI-A FIFO receive register
0x00704C	SPIFFCT	SPI-A FIFO control register
0x00704F	SPIPRI	SPI-A priority control register

10 - 5

SPI-A Configuration Control Register

SPICCR @ 0x007040

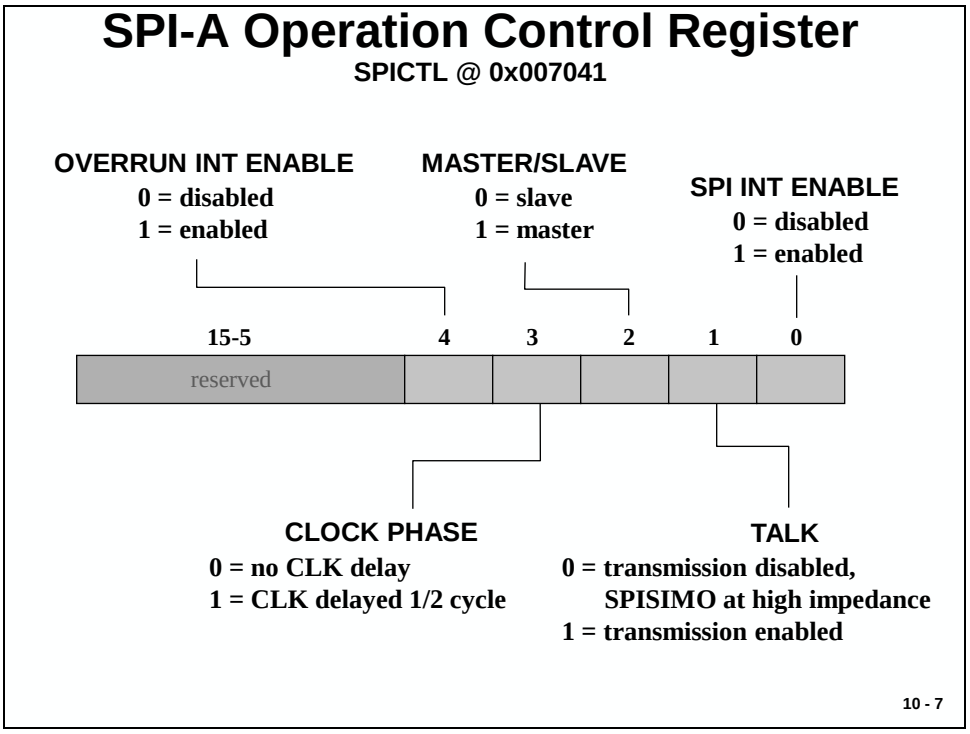


10 - 6

SPI Confguration Control Register - SPICCR

It is good practice to RESET the SPI unit at the beginning of the initialization procedure. This is done by clearing bit 7 (SPI SW RESET) to 0 using a first instruction, followed by setting it back to 1 using a second instruction. Bit 6 selects the active clock edge to declare the data as valid. This setup depends on the particular SPI - device. Bits 3...0 define the character length of the SPI-frame.

SPI Operation Control Register - SPICTL

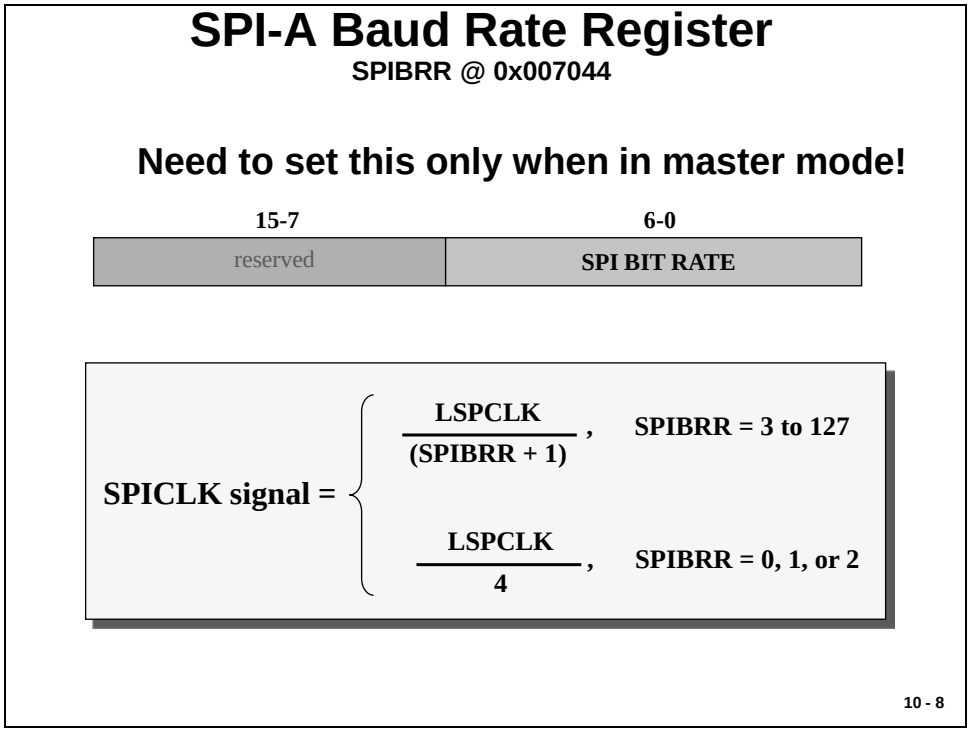


Bit 4 and bit 0 enable or disable the SPI- interrupts; Bit 4 enables the receiver's overflow interrupt. Bit 2 defines the operating mode for the F2833x to be master or slave of the SPI-chain. With the help of bit 3 we can implement another half clock cycle delay between the active clock edge and the point of time, when data are valid. Again, this bit depends on the particular SPI-device. Bit 1 controls whether the F2833x listens only (bit 1 = 0) or if it is initialized as receiver and transmitter (bit 1 = 1).

SPI Receive Emulation Buffer Register - SPIRXEMU

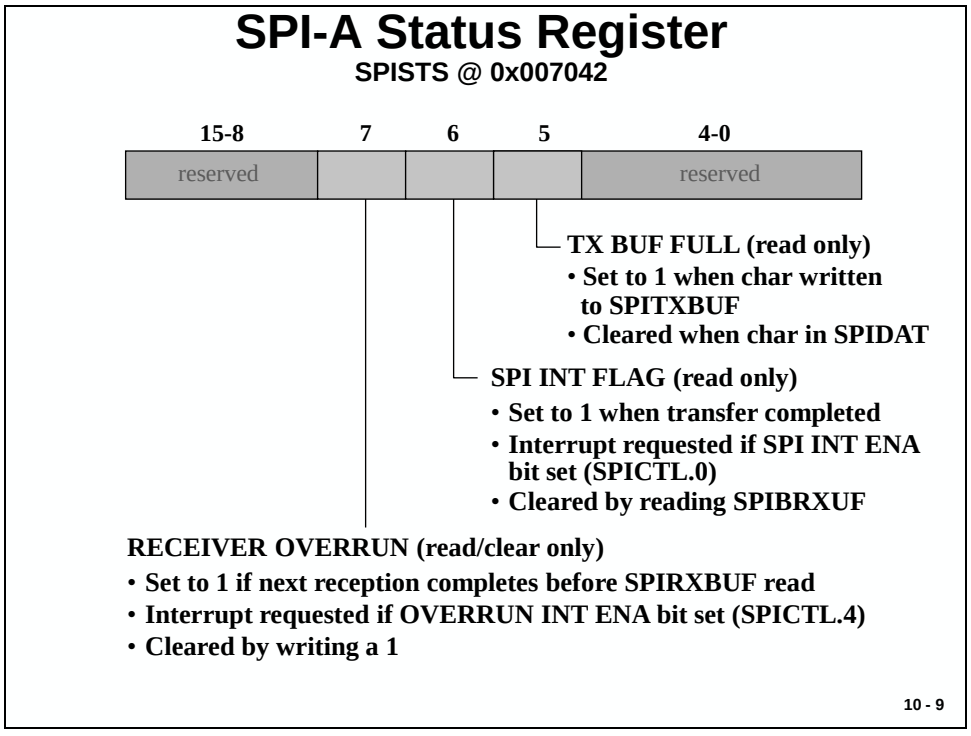
By reading register SPIRXBUF the corresponding interrupt flag SPI INT FLAG (SPISTS.6) is cleared automatically to allow a next character to be received. However, when we read SPIRXBUF just for test purposes, e.g. in a watch window, we would not want to have this bit cleared automatically. Therefore SPIRXEMU contains the same received data as SPIRXBUF, but reading SPIRXEMU does not clear the SPI INT FLAG bit. So every emulator access to SPIRXBUF actually reads data from SPIRXEMU without clearing the SPI INT FLAG.

SPI Baud Rate Register - SPIBRR

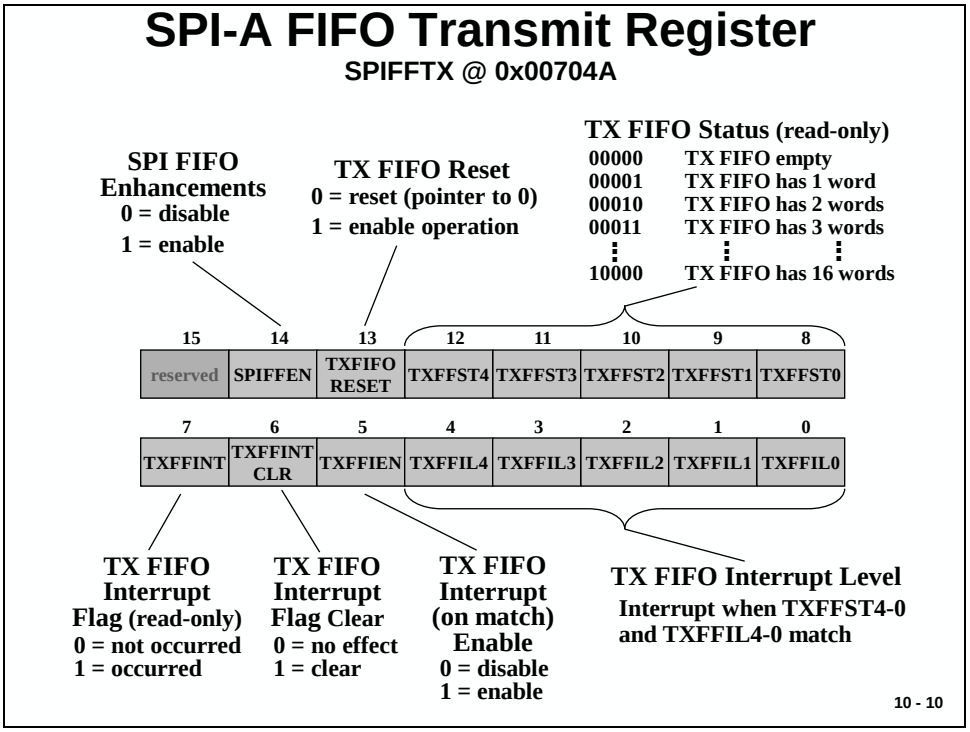


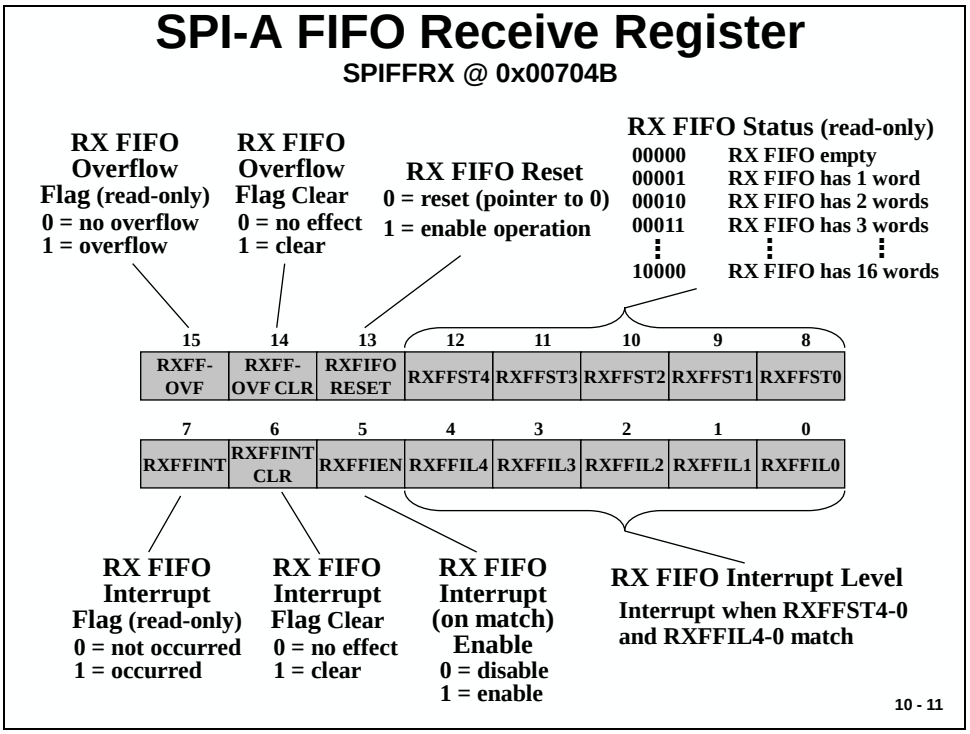
Clock base for the SPI baud rate selection is the Low speed Clock Prescaler (LSPCLK).

SPI Status Register - SPISTS



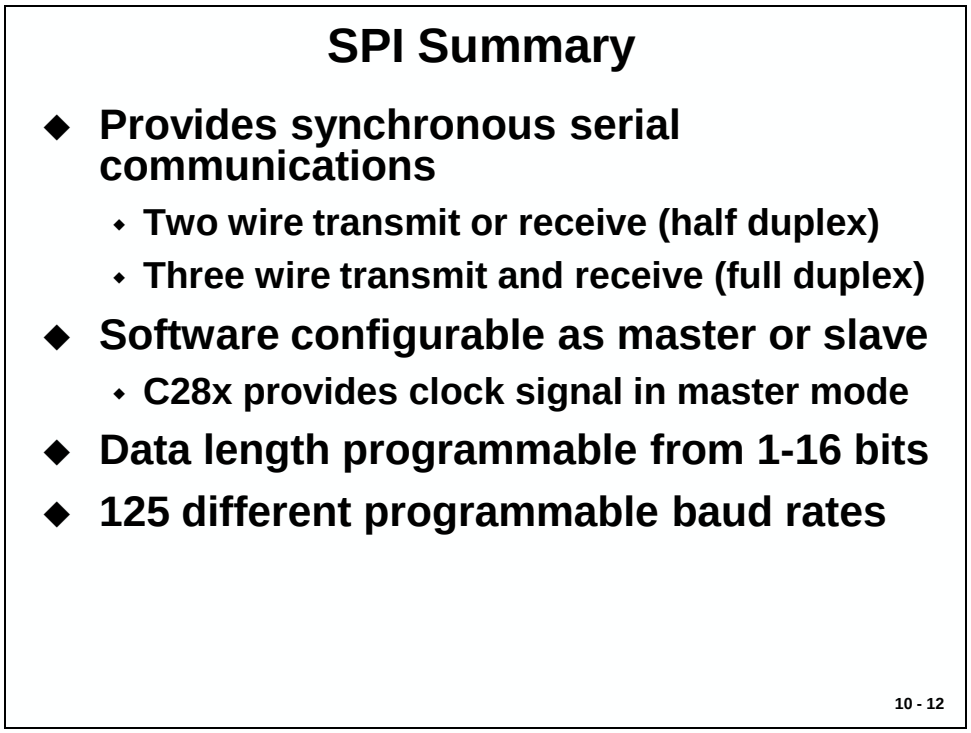
SPI FIFO Transmit Register





The FIFO operation of the SPI is controlled by bit 14 as master switch. The SPI-Transmit FIFO interrupt service call depends on the match between TX FIFO Status and TX FIFO Interrupt Level. The TX FIFO Reset can be used to reset the FIFO state machine (bit13= 0) and to re-enable it (bit 13=1).

SPI Summary



SPI Lab Exercises

At the Peripheral Explorer board the SPI-A channel is hard wired to the control lines of an audio codec device TLV320AIC23 (U7), which will be discussed at the end of the McBSP-chapter. In this context we will also discuss the setup of channel SPI-A.

Because of the hard wired SPI-A lines it is not possible to connect other SPI devices, such as EEPROMs, FRAMs, DACs or real-time clocks to the Peripheral Explorer Board. Unfortunately, we cannot perform an exercise with SPI-EEPROMs or DACs. However, if you use the previous Version 2.0 of this teaching CD-ROM based on the F2812, you will be able to perform an exercise with a SPI-EEPROM and a SPI-DAC.

Note: The Peripheral Explorer Board has an onboard SPI-EEPROM AT25256 (U6). However, this device has been hard wired to McBSP channel B. The Interface McBSP is able to operate in a “SPI-Emulation” operating mode, which will be also used at the end of the McBSP chapter to access the SPI-EEPROM.

blank page.