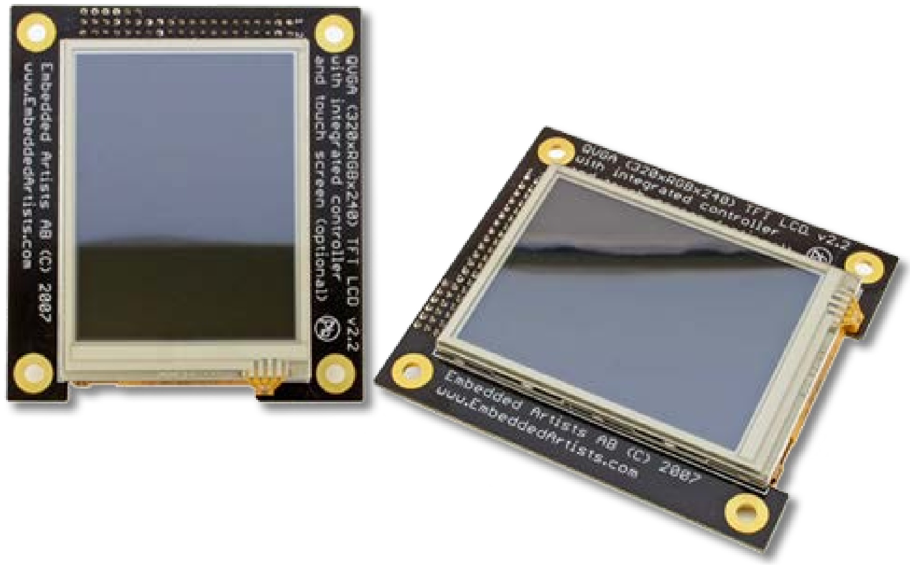
Programming Microcontroller

Serial Peripheral Interface (SPI) reduced Version

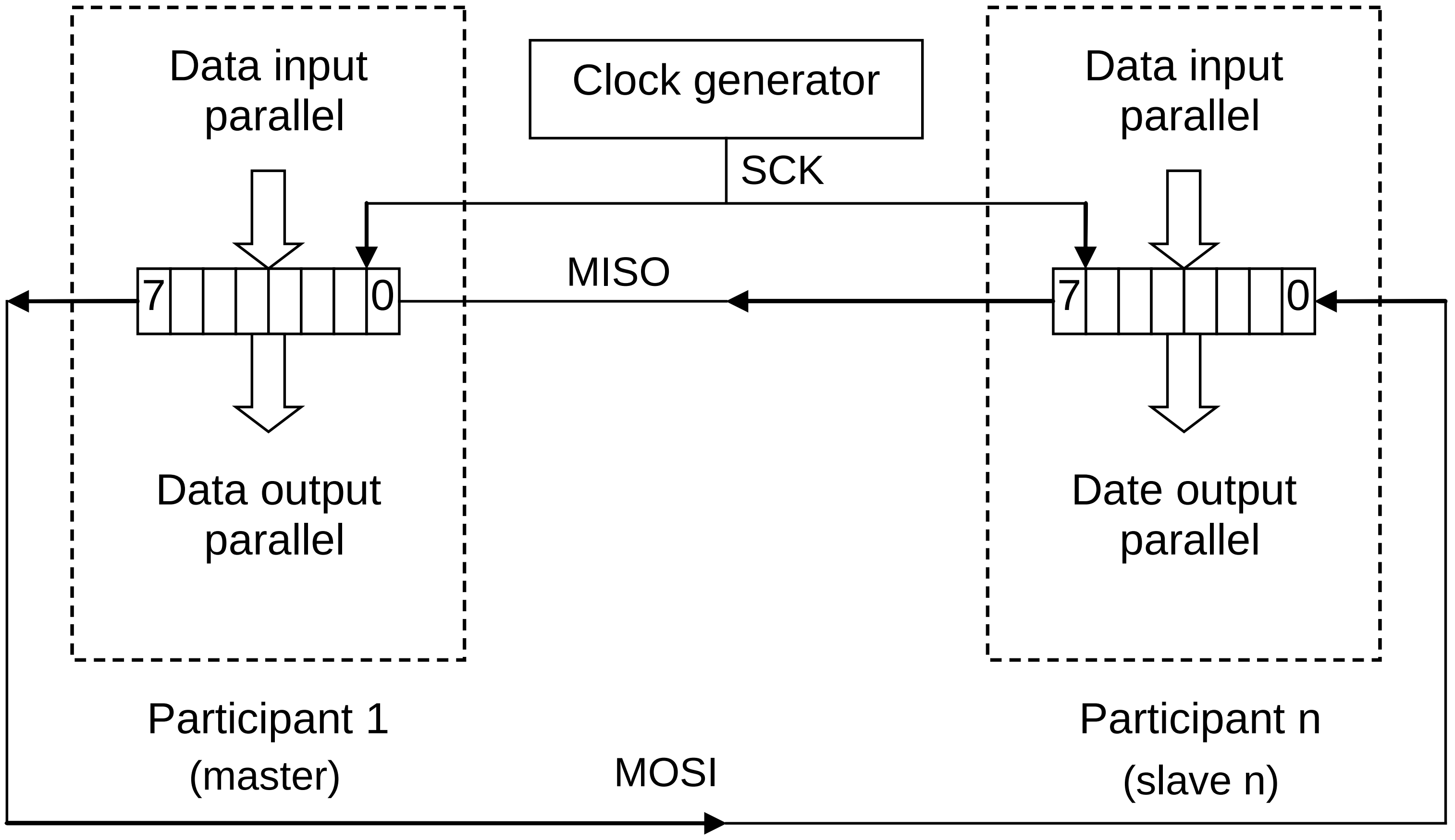
Aumn term 2016

Serial Peripheral Interface (SPI)

- SPI is serial high-speed bus system
  - Display, SD cards, AD- & DA converters
- SPI is always used in a master-slave mode
  - Master is responsible for the clock generation
- SPI may work in full duplex mode
  - MOSI (Master Out Slave In)
  - MISO (Master In Slave Out)
- Hardware CRC may be used


QVGA (320xRGBx240) TFT LCD v2.2
with integrated controller
and touch screen (optional)
Embedded Artists AB (C) 2007
www.EmbeddedArtists.com

SPI Hardware architecture: Synchronous shift register



SPI features (1/2)

Ref. RM0008 Reference manual, 25.2.1

- Full-duplex synchronous transfers on 3 lines
- Simple synchronous transfers on 2 lines
  - With or without a bidirectional data line
- 8- or 16-bit transfer format selection
- Master & Slave operation
- Multi master mode capability
- 8-bit master mode baud rate pre scaler (f_{PCLK}/2 max.)
- Slave mode frequency (f_{PCLK}/2 max.)
- Faster communication for both master and slave
- NSS management by hardware or software
- Programmable clock polarity

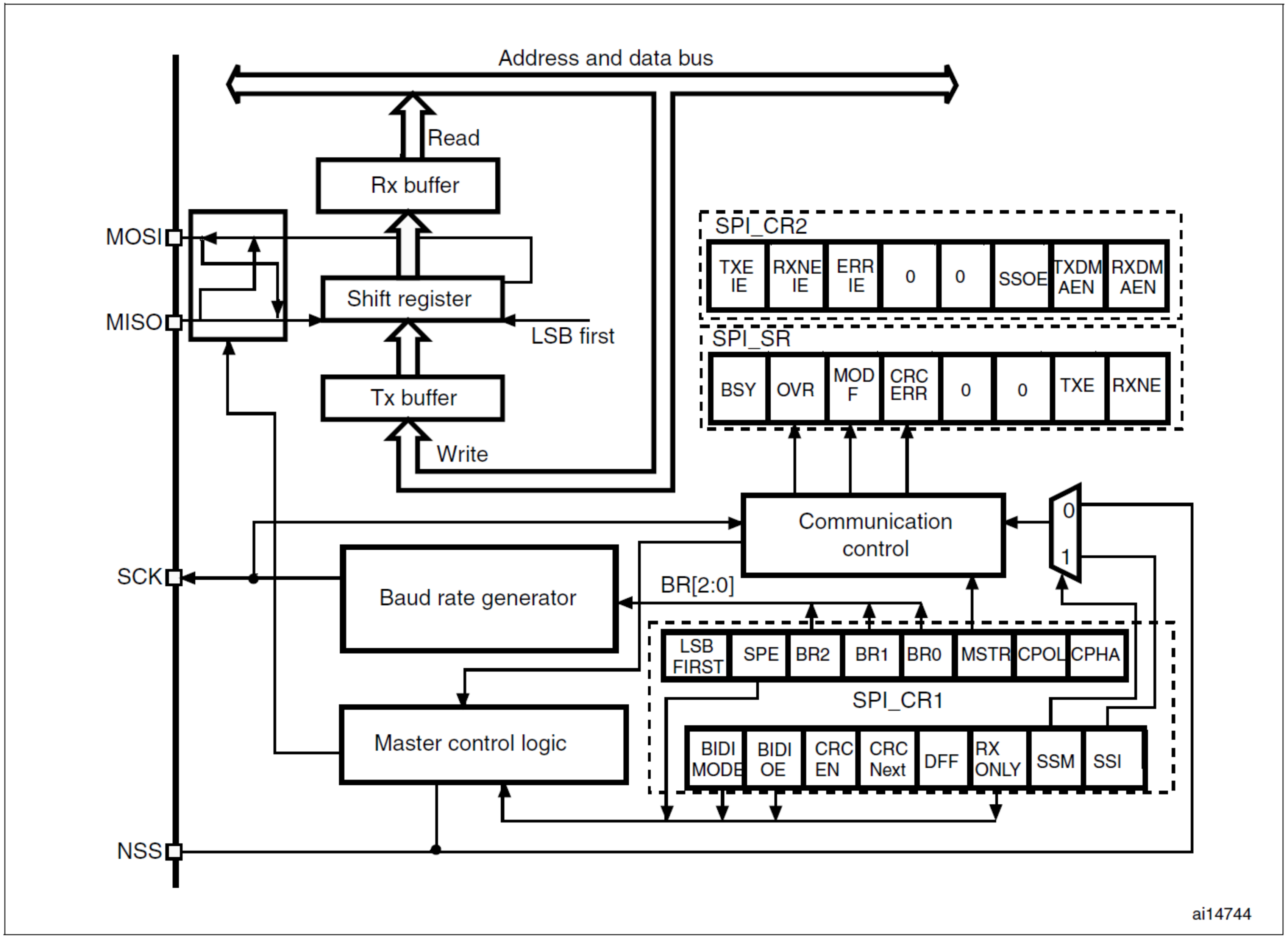
SPI features (2/2)

Ref. RM0008 Reference manual, page 25.2.1

- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- Hardware CRC feature for reliable communication
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun and CRC error flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability:
  - Tx and Rx requests

SPI Block diagram

Ref. RM0008 Reference manual, 25.3.1

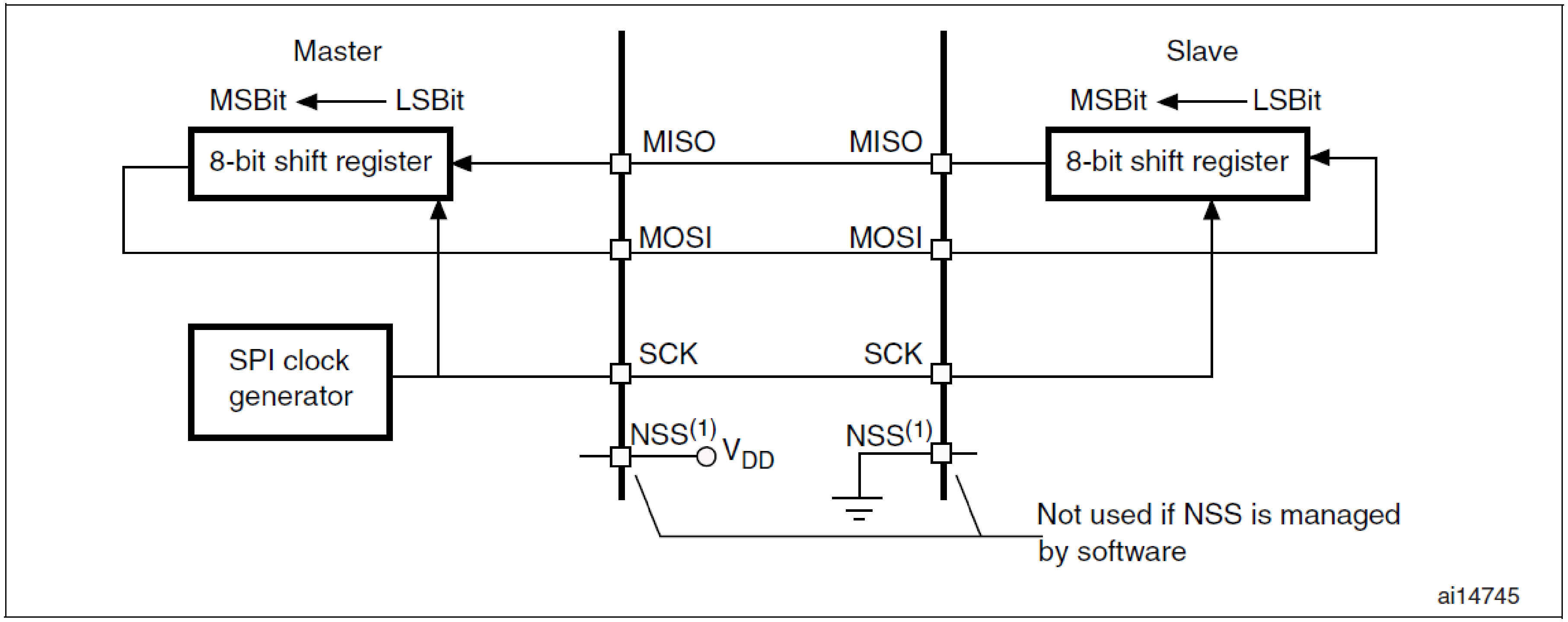


SPI pin description

Ref. RM0008 Reference manual, 25.3.1

⌘ Single master / single slave application

- Communication is always initiated by the master



Slave select (NSS) pin management

- NSS pin management can be realized either by hardware or by software
  - **SSM** (Slave Select Management) bit of the **SPI_CR1** register

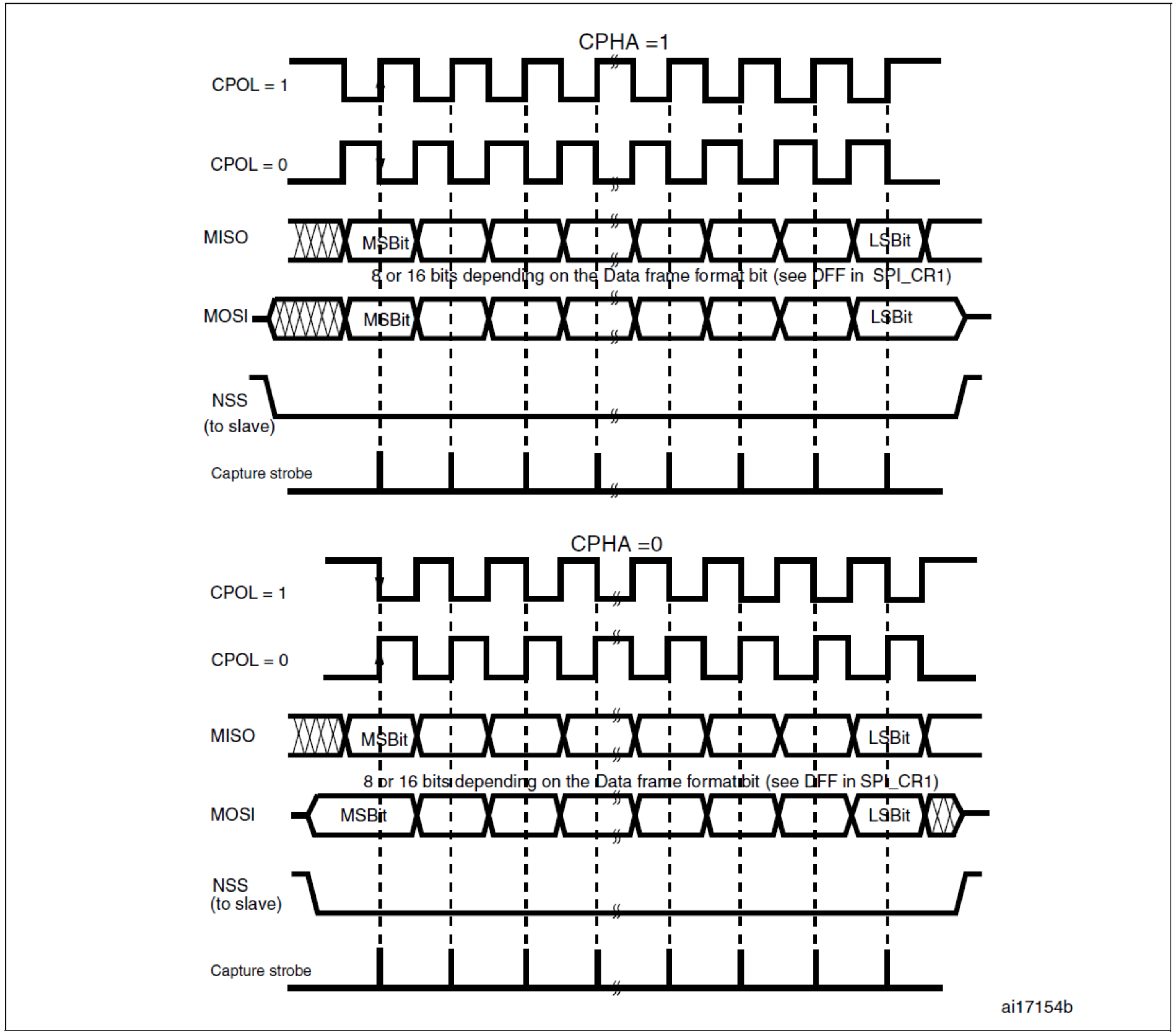
- Software NSS pin management (**SSM** = 1)
  - **SSI** (Internal Slave Select) bit in the **SPI_CR1** register

- Hardware NSS pin management (**SSM** = 0)
  - NSS output enable (**SSOE** = 1) (SS Output Enable)
    - Only in master mode
    - NSS is drive low when the master starts the communication

  - NSS output disable (**SSOE** = 0)
    - Master mode: NSS is used for multi master capability
    - Slave mode: NSS selects the slave

Clock phase & polarity

Ref. RM0008 Reference manual, figure 239



SPI register map (Offset)

Ref. RM0008 Reference manual, 25.5.10

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPI_CR1**	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPI_CR2**	Reserved																								TXEIE	RXNEIE	ERRIE	Reserved		SSOE	TXDMAEN	RXDMAEN
	Reset value																									0	0	0			0	0	0
0x08	**SPI_SR**	Reserved																								BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																									0	0	0	0	0	0	1	0
0x0C	**SPI_DR**	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPI_CRCPR**	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPI_RXCRCR**	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPI_TXCRCR**	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**SPI_I2SCFGR**	Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x20	**SPI_I2SPR**	Reserved																						MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

SPI register base addresses (1/2)

Ref. RM0008 Reference manual, page 50

Boundary address	Peripheral	Bus	Register map
0x4000 7800 - 0x4000 FFFF	Reserved		
0x4000 7400 - 0x4000 77FF	DAC		*Section 12.5.14 on page 264*
0x4000 7000 - 0x4000 73FF	Power control PWR		*Section 5.4.3 on page 78*
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)		*Section 6.4.5 on page 83*
0x4000 6800 - 0x4000 6BFF	Reserved		
0x4000 6400 - 0x4000 67FF	bxCAN1		*Section 24.9.5 on page 669*
0x4000 6800 - 0x4000 6BFF	bxCAN2		*Section 24.9.5 on page 669*
0x4000 6000[1] - 0x4000 63FF	Shared USB/CAN SRAM 512 bytes		
0x4000 5C00 - 0x4000 5FFF	USB device FS registers		*Section 23.5.4 on page 626*
0x4000 5800 - 0x4000 5BFF	I2C2		*Section 26.6.10 on page 756*
0x4000 5400 - 0x4000 57FF	I2C1		*Section 26.6.10 on page 756*
0x4000 5000 - 0x4000 53FF	UART5		*Section 27.6.8 on page 799*
0x4000 4C00 - 0x4000 4FFF	UART4		*Section 27.6.8 on page 799*
0x4000 4800 - 0x4000 4BFF	USART3		*Section 27.6.8 on page 799*
0x4000 4400 - 0x4000 47FF	USART2		*Section 27.6.8 on page 799*
0x4000 4000 - 0x4000 43FF	Reserved	APB1	
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S		*Section 25.5 on page 714*

SPI register base addresses (2/2)

Ref. RM0008 Reference manual, page 50

Boundary address	Peripheral	Bus	Register map
0x4001 5800 - 0x4001 7FFF	Reserved		
0x4001 5400 - 0x4001 57FF	TIM11 timer		*Section 16.6.11 on page 449*
0x4001 5000 - 0x4001 53FF	TIM10 timer		*Section 16.6.11 on page 449*
0x4001 4C00 - 0x4001 4FFF	TIM9 timer		*Section 16.5.13 on page 440*
0x4001 4000 - 0x4001 4BFF	Reserved		
0x4001 3C00 - 0x4001 3FFF	ADC3		*Section 11.12.15 on page 243*
0x4001 3800 - 0x4001 3BFF	USART1		*Section 27.6.8 on page 799*
0x4001 3400 - 0x4001 37FF	TIM8 timer		*Section 14.4.21 on page 348*
0x4001 3000 - 0x4001 33FF	SPI1		*Section 25.5 on page 714*
0x4001 2C00 - 0x4001 2FFF	TIM1 timer		*Section 14.4.21 on page 348*
0x4001 2800 - 0x4001 2BFF	ADC2	APB2	*Section 11.12.15 on page 243*
0x4001 2400 - 0x4001 27FF	ADC1		*Section 11.12.15 on page 243*
0x4001 2000 - 0x4001 23FF	GPIO Port G		*Section 9.5 on page 188*
0x4001 1C00 - 0x4001 1FFF	GPIO Port F		*Section 9.5 on page 188*
0x4001 1800 - 0x4001 1BFF	GPIO Port E		*Section 9.5 on page 188*
0x4001 1400 - 0x4001 17FF	GPIO Port D		*Section 9.5 on page 188*
0x4001 1000 - 0x4001 13FF	GPIO Port C		*Section 9.5 on page 188*

SPI Configuration in slave mode

Ref. RM0008 Reference manual, page 678

- Select the clock polarity and phase
  - **CPOL** and **CPHA** pins **SPI_CR1** Register
- Select the data frame format
  - **DFF** (Data Frame Format) bit of **SPI_CR1** Register
- Define the frame format
  - **LSBFIRST** bit of **SPI_CR1** Register
- Set the SPI in slave mode
  - Clear the **MSTR** (Master Selection) and set **SPE** (SPI Enable) bits in the **SPI_CR1** Register

- NSS Pin
  - Hardware mode
    - Connect the NSS pin to low level signal
  - Software mode
    - Set **SSM** bit and clear **SSI** bit in the **SPI_CR1** Register

SPI Configuration in master mode

Ref. RM0008 Reference manual, page 680

- Define the transmission baud rate
  - **BR[2:0]** bits of **SPI_CR1** Register
- Select the clock polarity and phase
  - **CPOL** and **CPHA** pins **SPI_CR1** Register
- Select the data frame format
  - **DFF** bit of **SPI_CR1** Register
- Define the frame format
  - **LSBFIRST** bit of **SPI_CR1** Register
- Set the SPI in master mode
  - Set the **MSTR** and **SPE** bits in the **SPI_CR1** Register

- NSS Pin
  - Hardware mode: connect the NSS pin to high level signal
  - Software mode: set the **SSM** and **SSI** bits in the **SPI_CR1** Register

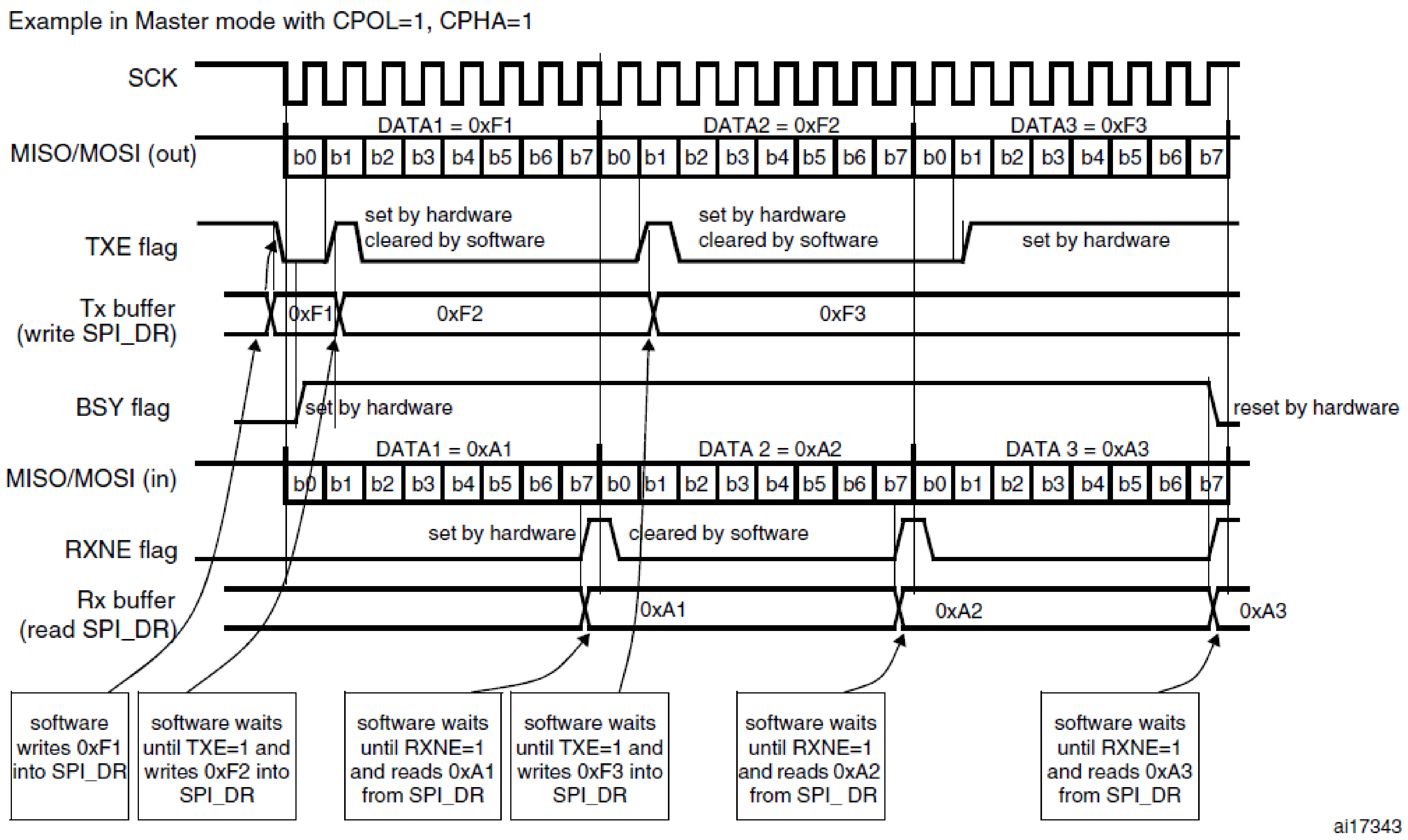
SPI Transmission & reception sequences

Ref. RM0008 Reference manual, page 680

- Transmit sequence
  - Write a byte in to the transmission buffer
    - **SPI_DR** register
  - **TXE** flag of the **CPI_SR** is set
    - An interrupt is generated if the **TXEIE** bit of the **SPI_CR2** is set

- Receive sequence
  - Received byte is transferred in to the reception buffer
    - **SPI_DR** register
  - **RXNE** flag of the **CPI_SR** is set
    - An interrupt is generated if the **TXEIE** bit of the **SPI_CR2** is set

Flag behaviour on transmit and receive



RM0008, fig 242

Code example: SPI configuration in master mode

⌘ Keil library function

```
SPI_InitStructure.SPI_Direction = \
                  SPI_Direction_2Lines_FullDuplex;
SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_High;      // CPOL = 1
SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge;     // CPHA = 1
SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
SPI_InitStructure.SPI_BaudRatePrescaler = \
                  SPI_BaudRatePrescaler_8;        // 4.5MHz
SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
SPI_InitStructure.SPI_CRCPolynomial = 7;

SPI_Init(SPI1, &SPI_InitStructure);
```

Code example: SPI transmission sequence

```
/*!< Loop while DR register in not empty */
while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_TXE) == RESET);

SPI_I2S_SendData(SPI1, (uint16_t) c);

/* Wait until byte has been written */
while(SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_BSY) != RESET);
```

!! Used like this, the uC has to wait until everything was written to the SPI bus.
This is called "blocking access" and eats up 100% of CPU !!

SPI alternate function remapping

Ref. RM0008 Reference manual, page 176

⌘ SPI remapping table

- *Ref. AF remap and debug I/O configuration register (AFIO_MAPR)*

Alternate function	SPI1_REMAP = 0	SPI1_REMAP = 1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

⌘ SD connection of the STM32F107xx board

- SPI_REMAP = 0
  - SCK $\Rightarrow$ PA5
  - MISO $\Rightarrow$ PA6
  - MOSI $\Rightarrow$ PA7