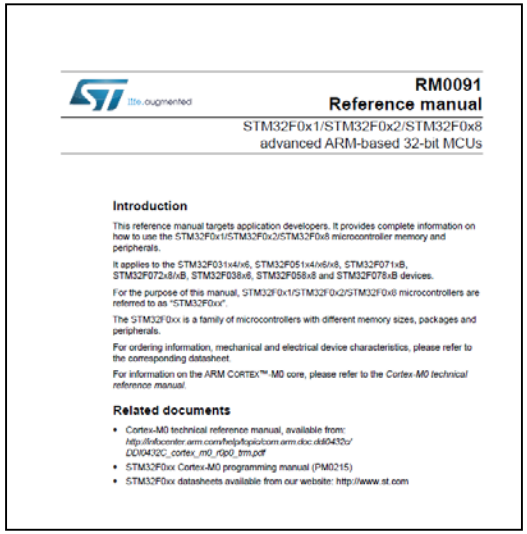
Embedded Communication

Serial Peripheral Interface (SPI)

Chapter 15 SPI
"Mastering STM32" by Carmine Noviello
https://leanpub.com/mastering-stm32

Chapter 27 Serial Peripheral Interface (SPI)
RM0091: STM32F0x1 Reference manual



life.augmented

RM0091
Reference manual

STM32F0x1/STM32F0x2/STM32F0x8
advanced ARM-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F0x1/STM32F0x2/STM32F0x8 microcontroller memory and peripherals.

It applies to the STM32F031x4/x6, STM32F051x4/x6/x8, STM32F071xB, STM32F072x8/xB, STM32F038x6, STM32F058x8 and STM32F078xB devices.

For the purpose of this manual, STM32F0x1/STM32F0x2/STM32F0x8 microcontrollers are referred to as "STM32F0xx".

The STM32F0xx is a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics, please refer to the corresponding datasheet.

For information on the ARM CORTEX™-M0 core, please refer to the *Cortex-M0 technical reference manual*.

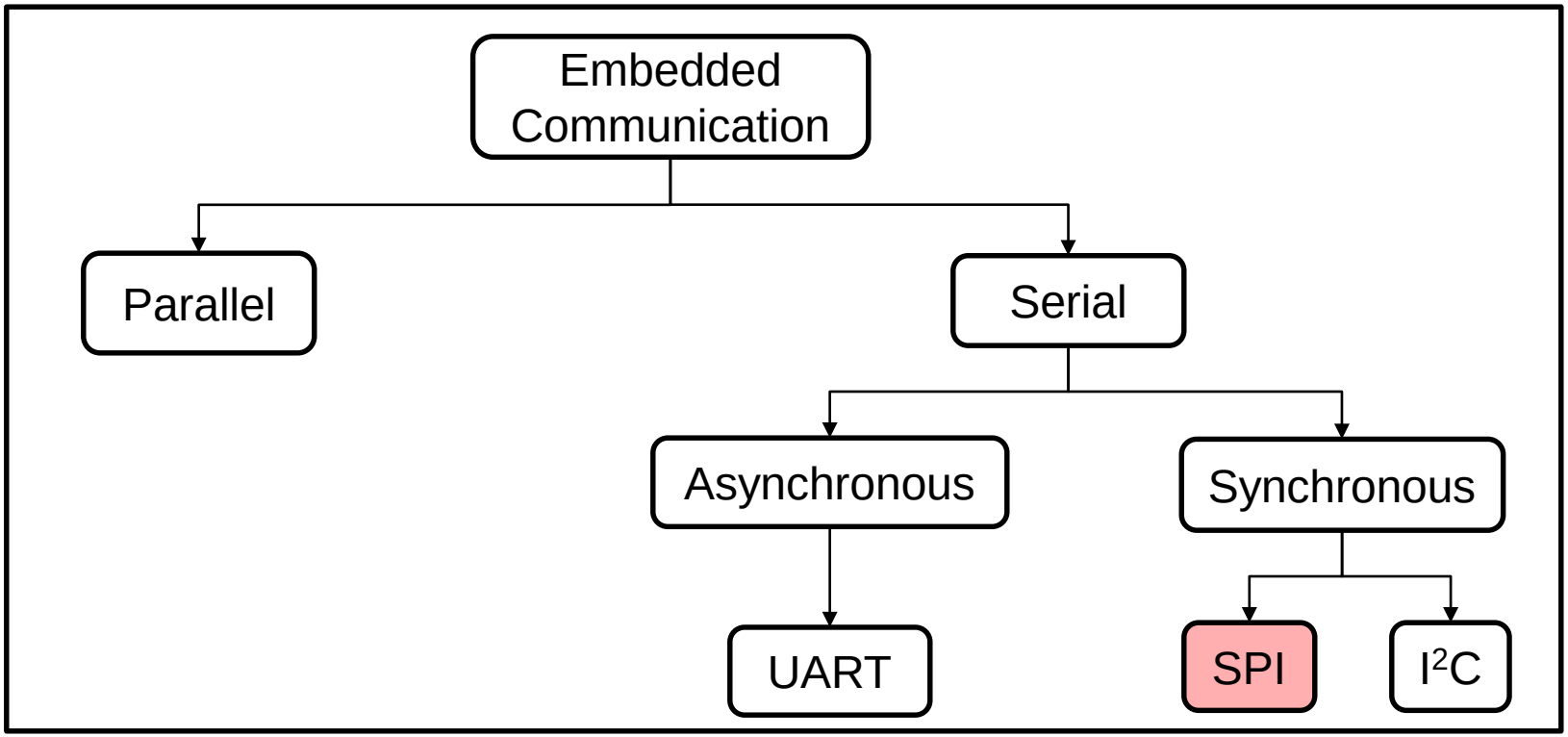
Related documents

- Cortex-M0 technical reference manual, available from: *http://infocenter.arm.com/help/topic/com.arm.doc.ddi0432c/DDI0432C_cortex_m0_r0p0_trm.pdf*
- STM32F0xx Cortex-M0 programming manual (PM0215)
- STM32F0xx datasheets available from our website: http://www.st.com

Embedded Systems II – EEE3096
R. Verrinder
Y. Abdul Gaffar

SPI and the big picture
Context

- SPI is a serial synchronous type of communication
  - Two devices share a common clock signal to synchronise the exchange of data



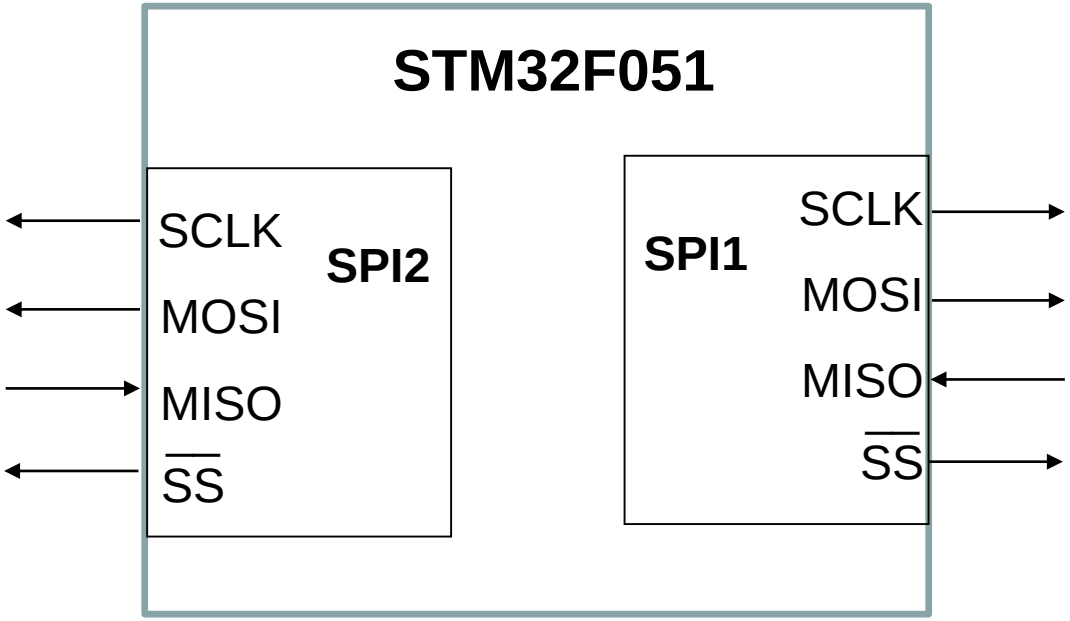
Overview of embedded communication

Serial Peripheral Interface

Implementation on the STM32F0

Serial Peripheral Interface
Implementation on the STM32F0

- STM32F051 has two SPI hardware modules
  - SPI 1
  - SPI 2



Serial Peripheral Interface
Implementation on the STM32F0

- STM32F051 has two SPI hardware modules
  - SPI 1
  - SPI 2

- SPI configurability (list below is not complete. See Chapter 27 for more details)
  - Master of slave operation
  - Full-duplex synchronous data transfer
  - Length of data: 4-bits to 16 bits
  - Order of data: LSB first or MSB first
  - Clock frequency (SCLK): 8 different settings
  - SPI modes: choose from mode 0, 1, 2 or 3

Serial Peripheral Interface
Implementation on the STM32F0

- Pins on the STM32F0 that can be used and their AF
  - PA4 (AF0) – SPI1_NSS (slave select)
  - PA5 (AF0) – SPI1_SCLK (clock)
  - PA6 (AF0) – SPI1_MISO (master input slave output)
  - PA7 (AF0) – SPI1_MOSI (master output slave input)

  SPI1

  - PA15 (AF0) – SPI1_NSS (slave select)
  - PB3 (AF0) – SPI1_SCLK (clock)
  - PB4 (AF0) – SPI1_MISO (master input slave output)
  - PB5 (AF0) – SPI1_MOSI (master output slave input)

  SPI1

  - PB12 (AF0) – SPI2_NSS (slave select)
  - PB13 (AF0) – SPI2_SCLK (clock)
  - PB14 (AF0) – SPI2_MISO (master input slave output)
  - PB15 (AF0) – SPI2_MOSI (master output slave input)

  SPI2

GPIO

Output mode: bit set/rest register

Configuring a GPIO pin
Output mode: bit set/reset register

- The lower 16-bits of the GPIO port bit set/reset register (GPIOx_BSRR), bit0 to bit15, is used to set the individual logic state of a pin

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)
0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Extracted from the STM32F051 reference manual

Configuring a GPIO pin
Output mode: bit set/reset register

- The lower 16-bits of the GPIO port bit set/reset register (GPIOx_BSRR), bit0 to bit15, is used to set the individual logic state of a pin
  - Example: writing a ‘1’ to bit 0 of the GPIOB_BSRR will set PB0 to logic HIGH

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)
0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Extracted from the STM32F051 reference manual

Configuring a GPIO pin
Output mode: bit set/reset register

- The lower 16-bits of the GPIO port bit set/reset register (GPIOx_BSRR), bit0 to bit15, is used to set the individual logic state of a pin
  - Example: writing a ‘1’ to bit 0 of the GPIOB_BSRR will set PB0 to logic HIGH

- The higher 16-bits of the GPIO port bit set/reset register (GPIOx_BSRR), bit16 to bit 31, is used to clear the individual logic state of a pin

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Extracted from the STM32F051 reference manual

Configuring a GPIO pin

Output mode: bit set/reset register

- The lower 16-bits of the GPIO port bit set/reset register (GPIOx_BSRR), bit0 to bit15, is used to set the individual logic state of a pin
  - Example: writing a ‘1’ to bit 0 of the GPIOB_BSRR will set PB0 to logic HIGH

- The higher 16-bits of the GPIO port bit set/reset register (GPIOx_BSRR), bit16 to bit 31, is used to clear the individual logic state of a pin
  - Example: writing a ‘1’ to bit 16 of the GPIOB_BSRR will set PB0 to logic LOW

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

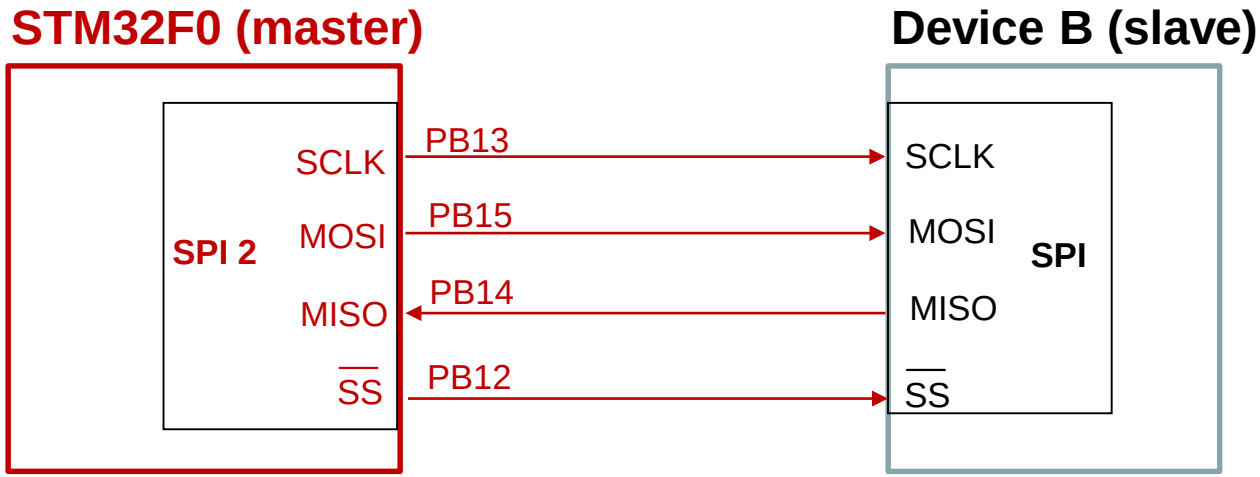
Extracted from the STM32F051 reference manual

Serial Peripheral Interface

Steps to program the STM32F0

Serial Peripheral Interface
Steps to program the STM32F051

- **Objective**
  - Setup STM32F0 as the master
  - Use SPI 2
  - Clock frequency: SCLK = 3MHz
  - Data bits: 8
  - Full duplex
- **Pin mapping**
  - SS : PB12
  - SCLK : PB13
  - MISO : PB14
  - MOSI : PB15
- **Assumptions**
  - f_{CLK} = 48MHz



Serial Peripheral Interface
Steps for intialisation: STM32F051

- Step 1: Enable clock on GPIO port B
  - Set IOPBEN (bit 18) in RCC_AHBENR

Bit 18 **IOPBEN:** I/O port B clock enable
Set and cleared by software.
0: I/O port B clock disabled
1: I/O port B clock enabled

7.4.6 AHB peripheral clock enable register (RCC_AHBENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN	Res.	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.
							rw		rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC EN	Res.	FLITF EN	Res.	SRAM EN	Res.	DMA EN
									rw		rw		rw		rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- Step 1: Enable clock on GPIO port B
  - Set IOPBEN (bit 18) in RCC_AHBENR

```
RCC -> AHBENR |= RCC_AHBENR_GPIOBEN;      // enable clock PORT B
```

Bit 18 **IOPBEN:** I/O port B clock enable
Set and cleared by software.
0: I/O port B clock disabled
1: I/O port B clock enabled

7.4.6 AHB peripheral clock enable register (RCC_AHBENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN	Res.	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.
							rw		rw	rw	rw	rw	rw	rw	

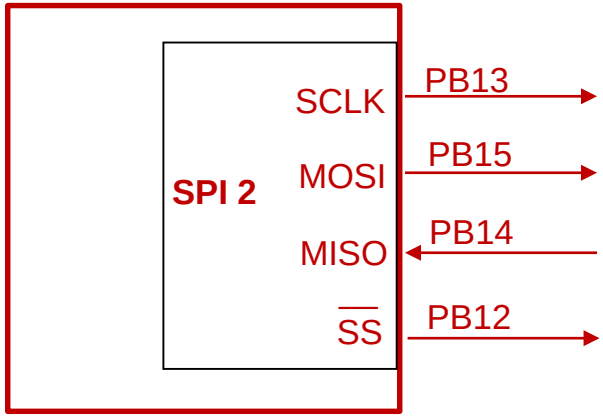
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC EN	Res.	FLITF EN	Res.	SRAM EN	Res.	DMA EN
									rw		rw		rw		rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- Step 1: Enable clock on GPIO port B
- Step 2: Configure port pins
  - To configure **PB13, PB14, PB15 to AF**, set MODER13[1:0], MODER14[1:0] and MODER15[1:0] to ‘10’ in GPIOB_MODER
  - To configure **PB12 to output**, set MODER12[1:0] to ’01’

STM32F0 (master)



9.4.1 GPIO port mode register (GPIOx_MODER) (x = A..F)

Address offset: 0x00

Reset values:

- 0x2800 0000 for port A
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
~~01: General purpose output mode~~
10: Alternate function mode
11: Analog mode

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- Step 1: Enable clock on GPIO port B
- Step 2: Configure port pins
  - To configure **PB13, PB14, PB15 to AF**, set MODER13[1:0], MODER14[1:0] and MODER15[1:0] to '10' in GPIOB_MODER
  - To configure **PB12 to output**, set MODER12[1:0] to '01'

```
GPIOB -> MODER |= GPIO_MODER_MODER13_1;     // set PB13 to AF mode
GPIOB -> MODER |= GPIO_MODER_MODER14_1;     // set PB14 to AF mode
GPIOB -> MODER |= GPIO_MODER_MODER15_1;    // set PB14 to AF mode

GPIOB -> MODER |= GPIO_MODER_MODER12_0;   // set PB12 to output mode
```

Serial Peripheral Interface
Steps for intialisation: STM32F051

- Step 1: Enable clock on GPIO port B
- Step 2: Configure port pins
- Step 3: Map PB13, PB14 and PB15 to SPI2_SCLK, SPI2_MISO and SPI2_MOSI respectively
  - Set AFR13 to ‘0000’ in GPIOB_AFRH
  - Set ARF14 to ‘0000’ in GPIOB_AFRH
  - Set ARF15 to ‘0000’ in GPIOB_AFRH

AFRy selection:
0000: AF0
0001: AF1
0010: AF2
0011: AF3

9.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A..E)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Table 15. Alternate functions selected through GPIOB_AFR registers for port B

Pin name	AF0	AF1	AF2	AF3
PB0	EVENTOUT	TIM3_CH3	TIM1_CH2N	TSC_G3_IO2
PB1	TIM14_CH1	TIM3_CH4	TIM1_CH3N	TSC_G3_IO3
PB2				TSC_G3_IO4
PB3	SPI1_SCK, I2S1_CK	EVENTOUT	TIM2_CH2	TSC_G5_IO1
PB4	SPI1_MISO, I2S1_MCK	TIM3_CH1	EVENTOUT	TSC_G5_IO2
PB5	SPI1_MOSI, I2S1_SD	TIM3_CH2	TIM16_BKIN	I2C1_SMBA
PB6	USART1_TX	I2C1_SCL	TIM16_CH1N	TSC_G5_IO3
PB7	USART1_RX	I2C1_SDA	TIM17_CH1N	TSC_G5_IO4
PB8	CEC	I2C1_SCL	TIM16_CH1	TSC_SYNC
PB9	IR_OUT	I2C1_SDA	TIM17_CH1	EVENTOUT
PB10	CEC	I2C2_SCL	TIM2_CH3	TSC_SYNC
PB11	EVENTOUT	I2C2_SDA	TIM2_CH4	TSC_G6_IO1
PB12	SPI2_NSS	EVENTOUT	TIM1_BKIN	TSC_G6_IO2
PB13	SPI2_SCK		TIM1_CH1N	TSC_G6_IO3
PB14	SPI2_MISO	TIM15_CH1	TIM1_CH2N	TSC_G6_IO4
PB15	SPI2_MOSI	TIM15_CH2	TIM1_CH3N	TIM15_CH1N

Extracted from the STM32F051 datasheet

Serial Peripheral Interface
Steps for intialisation: STM32F051

- Step 1: Enable clock on GPIO port B
- Step 2: Configure port pins
- Step 3: Map PB13, PB14 and PB15 to SPI2_SCLK, SPI2_MISO and SPI2_MOSI respectively
  - Set AFR13 to ‘0000’ in GPIOB_AFRH
  - Set ARF14 to ‘0000’ in GPIOB_AFRH
  - Set ARF15 to ‘0000’ in GPIOB_AFRH

```
# define GPIO_AFRH_PIN13_AFR = 0b00000000111100000000000000000000;
# define GPIO_AFRH_PIN14_AFR = 0b00001111000000000000000000000000;
# define GPIO_AFRH_PIN15_AFR = 0b11110000000000000000000000000000;

GPIOA -> AFR[1] &= ~GPIO_AFRH_PIN13_AFR;      // clear bits AFR13[3:0]
GPIOA -> AFR[1] &= ~GPIO_AFRH_PIN14_AFR;      // clear bits AFR14[3:0]
GPIOA -> AFR[1] &= ~GPIO_AFRH_PIN15_AFR;      // clear bits AFR15[3:0]
```

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
  - Set PB12 high

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
  - Set PB12 high

```
GPIOB -> BSRR |= GPIO_BSRR_BS_12;   // disable the slave device by setting PB12 high
```

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
  - Set SPI2EN (bit 14) in APB1ENR

Bit 14 **SPI2EN:** SPI2 clock enable
Set and cleared by software.
0: SPI2 clock disabled
1: SPI2 clock enabled

7.4.8 APB peripheral clock enable register 1 (RCC_APB1ENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEC EN	DAC EN	PWR EN	CRS EN	Res.	CAN EN	Res.	USB EN	I2C2 EN	I2C1 EN	Res.	USART 4EN	USART 3EN	USART 2EN	Res.
	rw	rw	rw	rw		rw		rw	rw	rw		rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2 EN	Res.	Res.	WWDG EN	Res.	Res.	TIM14 EN	Res.	Res.	TIM7 EN	TIM6 EN	Res.	Res.	TIM3 EN	TIM2 EN
	rw			rw			rw			rw	rw			rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
  - Set SPI2EN (bit 14) in APB1ENR

```
RCC -> APB1ENR |= RCC_APB1ENR_SPI2EN;        // enable clock for SPI 2
```

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
  - Set BIDOE (bit 14) in SPI2_CR1

Bit 14 **BIDIOE:** Output enable in bidirectional mode

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

27.7.1 SPI control register 1 (SPIx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
  - Set BIDOE (bit 14) in SPI2_CR1

```
SPI2 -> CR1 |= SPI_CR1_BIDIOE;        // enable output to transmit only
```

Bit 14 **BIDIOE:** Output enable in bidirectional mode

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

27.7.1 SPI control register 1 (SPIx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
  - Assuming f_{CLK} = 48MHz, then set f_{SCLK} = 3MHz by writing '011' to BR[2:0]. [Note: f_{CLK} /16 = 3MHz]

Bits 5:3 **BR[2:0]:** Baud rate control
000: $f_{PCLK}/2$
001: $f_{PCLK}/4$
010: $f_{PCLK}/8$
011: $f_{PCLK}/16$
100: $f_{PCLK}/32$

27.7.1 SPI control register 1 (SPIx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
  - Assuming f_{CLK} = 48MHz, then set f_{SCLK} = 3MHz by writing '011' to BR[2:0]. [Note: f_{CLK} /16 = 3MHz]

Bits 5:3 **BR[2:0]**: Baud rate control
000: $f_{PCLK}/2$
001: $f_{PCLK}/4$
010: $f_{PCLK}/8$
011: $f_{PCLK}/16$
100: $f_{PCLK}/32$

```
SPI2 -> CR1 |= (SPI_CR1_BR_0| SPI_CR1_BR_1);   // write '011' to BR[2:0]
```

27.7.1 SPI control register 1 (SPIx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
  - Set MSTR (bit 2) to configure SPI2 in STM32F0 to be the master

Bit 2 **MSTR:** Master selection
0: Slave configuration
1: Master configuration

27.7.1 SPI control register 1 (SPIx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
  - Set MSTR (bit 2) to configure SPI2 in STM32F0 to be the master

```
SPI2 -> CR1 |= SPI_CR1_MSTR;   // set SPI2 of STM32F0 to be the master
```

Bit 2 **MSTR:** Master selection
0: Slave configuration
1: Master configuration

27.7.1 SPI control register 1 (SPIx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
- Step 9: Set data length to 8-bits
  - Write ‘0111’ to DS[3:0] in SPI2_CR2
  - Set FRXTH (bit 12) in SPI2_CR2

Bit 11:8 **DS [3:0]**: Data size

0000: Not used
0001: Not used
0010: Not used
0011: 4-bit
0100: 5-bit
0101: 6-bit
0110: 7-bit
0111: 8-bit

Bit 12 **FRXTH**: FIFO reception threshold
This bit is used to set the threshold of the RXFIFO that triggers an RXNE event
0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)
1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)

27.7.2 SPI control register 2 (SPIx_CR2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LDMA_TX	LDMA_RX	FRXTH	DS [3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
- Step 9: Set data length to 8-bits
  - Write ‘0111’ to DS[3:0] in SPI2_CR2
  - Set FRXTH (bit 12) in SPI2_CR2

```
SPI2 -> CR2 |= (SPI_CR2_DS_0| SPI_CR2_DS_1| SPI_CR2_DS_2);   // set data size to 8-bits
SPI2 -> CR2 |= SPI_CR2_FRXTH;   // RXNE event is trigger when 8-bits is received
```

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
- Step 9: Set data length to 8-bits
- Step 10: Enable slave output
  - Clear PB12

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
- Step 9: Set data length to 8-bits
- Step 10: Enable slave output
  - Clear PB12

```
GPIOB -> BSRR |= GPIO_BSRR_BR_12;   // enable the slave device by setting PB12 low
```

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
- Step 9: Set data length to 8-bits
- Step 10: Enable slave output
- Step 11: Enable SPI2
  - Set SPE (bit x) in SPI2_CR1

Bit 6 **SPE:** SPI enable
0: Peripheral disabled
1: Peripheral enabled

Serial Peripheral Interface
Steps for intialisation: STM32F051

- …
- Step 4: Disable the slave device
- Step 5: Enable clock to SPI2
- Step 6: Set output (MOSI) to be transmit only
- Step 7: Set clock frequency
- Step 8: Configure SPI2 in STM32F0 to be the master
- Step 9: Set data length to 8-bits
- Step 10: Enable slave output
- Step 11: Enable SPI2
  - Set SPE (bit x) in SPI2_CR1

Bit 6 **SPE:** SPI enable
0: Peripheral disabled
1: Peripheral enabled

```
SPI2 -> CR1 |= SPI_CR1_SPE;   // Enable SPI2
```

Serial Peripheral Interface
C code for intialisation: STM32F051

```
void init_SPI2(void)
{

 # define GPIO_AFRH_PIN13_AFR = 0b00000000111100000000000000000000;
 # define GPIO_AFRH_PIN14_AFR = 0b00001111000000000000000000000000;
 # define GPIO_AFRH_PIN15_AFR = 0b11110000000000000000000000000000;

RCC -> AHBENR |= RCC_AHBENR_GPIOBEN;       // enable clock PORT B

GPIOB -> MODER |= GPIO_MODER_MODER13_1;     // set PB13 to AF mode
GPIOB -> MODER |= GPIO_MODER_MODER14_1;     // set PB14 to AF mode
GPIOB -> MODER |= GPIO_MODER_MODER15_1;    // set PB14 to AF mode
GPIOB -> MODER |= GPIO_MODER_MODER12_0;   // set PB12 to output mode

GPIOA -> AFR[1] &= ~GPIO_AFRH_PIN13_AFR;       // clear bits AFR13[3:0]
GPIOA -> AFR[1] &= ~GPIO_AFRH_PIN14_AFR;      // clear bits AFR14[3:0]
GPIOA -> AFR[1] &= ~GPIO_AFRH_PIN15_AFR;     // clear bits AFR15[3:0]
```

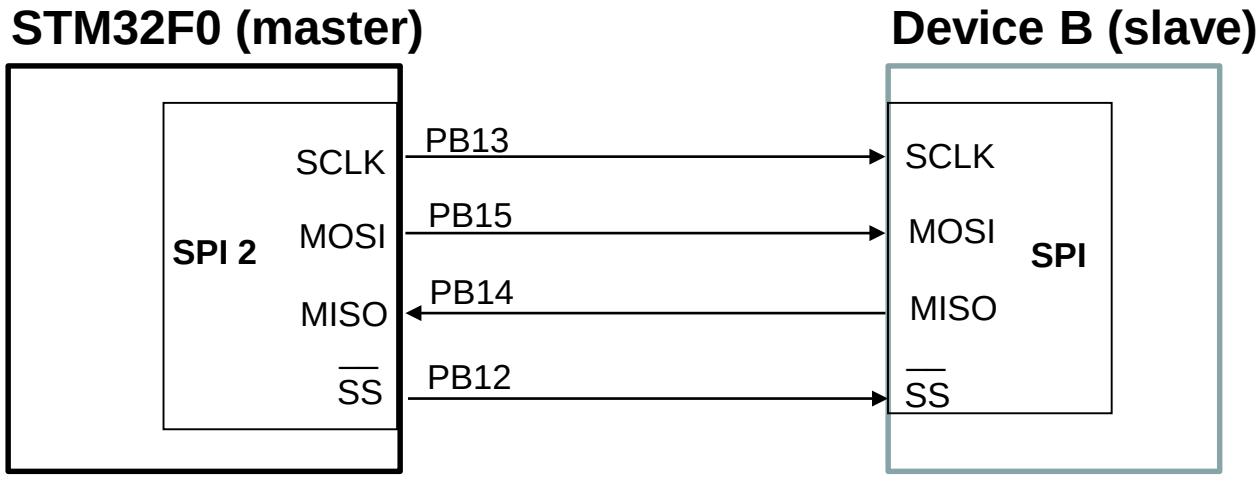
Serial Peripheral Interface
C code for intialisation: STM32F051

```
GPIOB -> BSRR |= GPIO_BSRR_BS_12;            // disable the slave device by setting PB12 high
RCC -> APB1ENR |= RCC_APB1ENR_SPI2EN;        // enable clock for SPI 2

SPI2 -> CR1 |= SPI_CR1_BIDIOE;               // enable output to transmit only
SPI2 -> CR1 |= (SPI_CR1_BR_0| SPI_CR1_BR_1);   // write '011' to BR[2:0]
SPI2 -> CR1 |= SPI_CR1_MSTR;                 // set SPI2 of STM32F0 to be the master

SPI2 -> CR2 |= (SPI_CR2_DS_0| SPI_CR2_DS_1| SPI_CR2_DS_2);   // set data size to 8-bits
SPI2 -> CR2 |= SPI_CR2_FRXTH;                // RXNE event is trigger when 8-bits is received
SPI2 -> CR2 |= SPI_CR2_SSOE;                 // Enable slave

SPI2 -> CR1 |= SPI_CR1_SPE;                  // Enable SPI2

}
```

Serial Peripheral Interface

Steps to program the master (STM32F0) to **transmit data to a slave**, in polling mode

Master (STM32F0) transmits data to slave (device B)



Serial Peripheral Interface
Steps and C code for transmitting data

- Step 1: Enable the slave device by clearing $\overline{SS}$

```
#define DataToTx 0b00000010          // 8-bit data to transmit from master to slave

GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
```

Serial Peripheral Interface
Steps and C code for transmitting data

- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write transmit data in SPI2_DR

```
#define DataToTx 0b00000010          // 8-bit data to transmit from master to slave

GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DataToTx;               // write transmit data to data register of SPI2
```

Serial Peripheral Interface
Steps and C code for transmitting data

- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write transmit data in SPI2_DR
- Step 3: Wait until receive buffer ≥ 8 bits

```
#define DataToTx 0b00000010          // 8-bit data to transmit from master to slave

GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DataToTx;               // write transmit data to data register of SPI2
while(SPI2 -> SR & SPI_SR_RXNE) ==0);  // wait until 8-bits are put into the rx buffer
```

Serial Peripheral Interface
Steps and C code for transmitting data

- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write transmit data in SPI2_DR
- Step 3: Wait until receive buffer ≥ 8 bits
- Step 4: Read ‘dummy’ data in the receive buffer

```
#define DataToTx 0b00000010          // 8-bit data to transmit from master to slave

GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DataToTx;               // write transmit data to data register of SPI2
while(SPI2 -> SR & SPI_SR_RXNE) ==0);  // wait until 8-bits are put into the rx buffer
DummyData = SPI2 -> DR;              // read ‘dummy’ data shifted into the master
```

Serial Peripheral Interface
Steps and C code for transmitting data

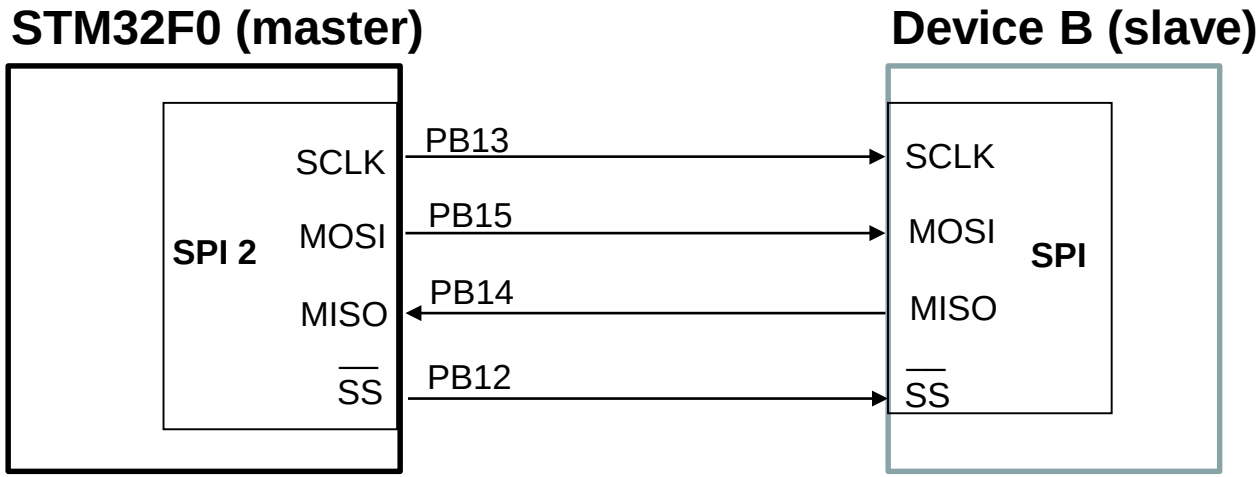
- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write transmit data in SPI2_DR
- Step 3: Wait until receive buffer ≥ 8 bits
- Step 4: Read ‘dummy’ data in the receive buffer
- Step 5: Disable the slave device by setting $\overline{SS}$

```
#define DataToTx 0b00000010          // 8-bit data to transmit from master to slave

GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DataToTx;               // write transmit data to data register of SPI2
while(SPI2 -> SR & SPI_SR_RXNE) ==0);  // wait until 8-bits are put into the rx buffer
DummyData = SPI2 -> DR;              // read ‘dummy’ data shifted into the master
GPIOB -> BSRR |= GPIO_BSRR_BS_12;  // disable the slave: set PB12
```

Serial Peripheral Interface

Steps to program the master (STM32F0) to read data from a slave, in polling mode

Master (STM32F0) receives data from a slave (device B)



Serial Peripheral Interface
Steps and C code for receiving data

- Step 1: Enable the slave device by clearing $\overline{SS}$

```
#define DummyDataToTx 0b00000001        // 8-bit dummy data to transmit
GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
```

Serial Peripheral Interface
Steps and C code for receiving data

- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write ‘dummy’ data in SPI2_DR

```
#define DummyDataToTx 0b00000001        // 8-bit dummy data to transmit
GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DummyDataToTx;       // write transmit data to data register of SPI2
```

Serial Peripheral Interface
Steps and C code for receiving data

- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write ‘dummy’ data in SPI2_DR
- Step 3: Wait until receive buffer ≥ 8 bits

```
#define DummyDataToTx 0b00000001        // 8-bit dummy data to transmit
GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DummyDataToTx;        // write transmit data to data register of SPI2
while(SPI2 -> SR & SPI_SR_RXNE) ==0);  // wait until 8-bits are put into the rx buffer
```

Serial Peripheral Interface
Steps and C code for receiving data

- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write ‘dummy’ data in SPI2_DR
- Step 3: Wait until receive buffer ≥ 8 bits
- Step 4: Read receive data in the receive buffer

```
#define DummyDataToTx 0b00000001        // 8-bit dummy data to transmit
GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DummyDataToTx;        // write transmit data to data register of SPI2
while(SPI2 -> SR & SPI_SR_RXNE) ==0);  // wait until 8-bits are put into the rx buffer
ReceivedData = SPI2 -> DR;                // read received data shifted into the master
```

Serial Peripheral Interface
Steps and C code for receiving data

- Step 1: Enable the slave device by clearing $\overline{SS}$
- Step 2: Write ‘dummy’ data in SPI2_DR
- Step 3: Wait until receive buffer $\geq$ 8 bits
- Step 4: Read receive data in the receive buffer
- Step 5: Disable the slave device by setting $\overline{SS}$

```
#define DummyDataToTx 0b00000001        // 8-bit dummy data to transmit
GPIOB -> BSRR |= GPIO_BSRR_BR_12;  // enable the slave: clear PB12
SPI2 -> DR = DummyDataToTx;        // write transmit data to data register of SPI2
while(SPI2 -> SR & SPI_SR_RXNE) ==0);  // wait until 8-bits are put into the rx buffer
ReceivedData = SPI2 -> DR;                 // read received data shifted into the master
GPIOB -> BSRR |= GPIO_BSRR_BS_12;  // disable the slave: set PB12
```