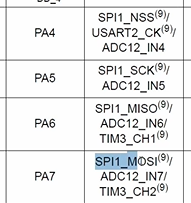
# SPI

## Clock:

SPI1 in APB2 in bit 12 , SPI2 in APB1 in bit 14, SPI3 in APB1 in bit 15.

## Pin:



## Clock activation:

* Enable AFIO in APB2 in bit 0.
* Enable Pin A in APB2 in bit 2
* Enable SPI1 in APB2 in bit 12

## Set up pin:

PA4 --> SS --> output GP-push-pull – 0011 (While NSS is controlled by software SSM=1, SSOE=0.)

PA4-->SS--> Output AF-pushpull-1011(while NSS is controlled by Hardware. SSM=0, SSOE=1) Used here.

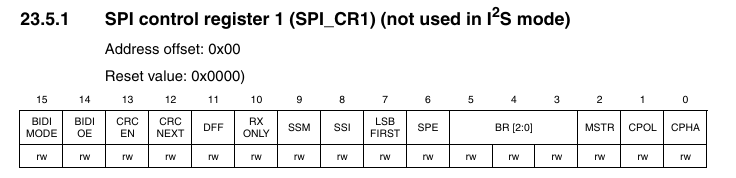
PA5 --> SCLK --> output AF-push-pull - 1011

PA6 --> MISO --> Input Floting-input - 0100

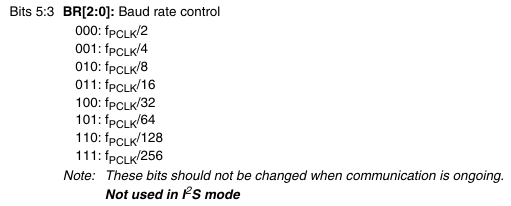
PA7 --> MOSI --> output AF-Push-pull 1011

## Setup SPI peripherals

### Control Register1 SPI\_CR1

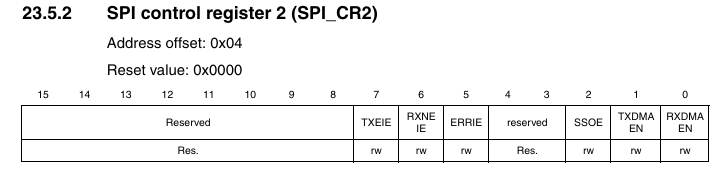


* Bit 2: MSTR: Set 1 for master device , 0 for slave device
* Bit [5:3] : BR[2:0]: Select the baud rate. Slower one is taken, 111



* Bit 6: SPE: enable SPI by setting this bit to 1.
* Bit 9 and 8: for software management of NSS. Is I want to change the valu of ss in software, not in physical pin, then SSM must be 1 and value of NSS will be in SSI ( if SSM is set, SSOE needs to be zero, as pysical NSS PA4 needs to be off)

### Control Register 2 SPI\_CR2

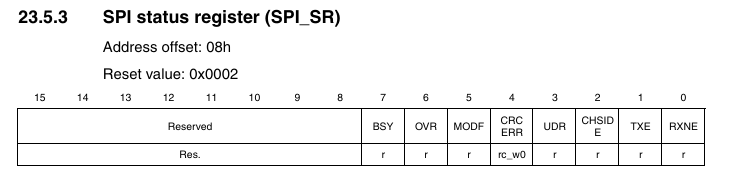


* Bit 2: SSOE: Set the SSOE to Enable Hardware NSS management. While set, NSS will be pulled low if SPI is enable. And SPI is always ready to send data. As soon as data is loaded in DR, transmission starts imidiately.

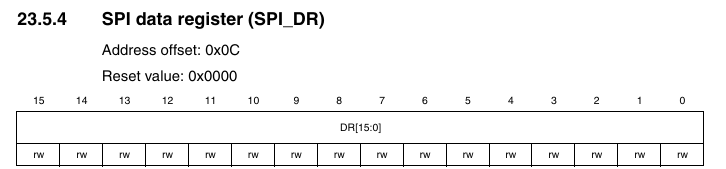
### NSS config:

|  |  |  |  |
| --- | --- | --- | --- |
| SSM | SSI | SSOE | Task |
| 0 | x | 0 | * NSS needs to be set manually, externally. |
| 0 | x | 1 | * NSS will be set and clear by SPI peripheral automatic. * Data will be sent as soon as load in SPI->DR. when * SPI is enable in CR1, hardware pulls the NSS (PA4) down. And wait to have message in DR. when massage is loaded in DR, it imidiately starts transmission starting SCLK. * So PA4 can be used as NSS for Slave * Multiple Master is not possible because Master with hardware NSS config pulls NSS down for all the time, Other Master willnot have scope to send Data. |
| 1 | 1 | 0 | * NSS will be set by software by coding. SSI holds the value of NSS |
| 1 | 0 | 0 | * NSS will be set by software. SSI holds the value of NSS |

### Status Register SPI\_SR



### Data Register SPI\_DR



## Working process

* Setup clock.
* Configure pin
* SPI Config ( set master, set baud rate, set SSOE for hardware NSS config, enable spi)
* Put the data in SPI1->DR( as soon as data is loaded in DR, transmission will start automatically in hardware NSS config. )
* Wait untill message is sent (while sending, BSY in SPI1->SR is high)
* Send second message

# Code:

|  |
| --- |
| #include <stm32f10x.h>  /\*  Actual setup  SPI - 1  -------  PA4 --> SS  PA5 --> SCLK  PA6 --> MISO  PA7 --> MOSI  \*/  uint32\_t tik = 0;  void En\_clock(void);  void gpio\_setup(void);  void delay\_ms(void);  void delay(uint32\_t count);  void systick\_config(void);  void SPI\_setup(void);  int main(void)  {  // Activate SPI1 Peripheral / AFIO Function  En\_clock();  // config systic timer  systick\_config();  // Set up pin  gpio\_setup();  // Setup SPI peripherals  SPI\_setup();  while (1)  {  // Loading data in DR  SPI1->DR = 0x50;  // waiting for completing transmission  while ((SPI1->SR & SPI\_SR\_BSY))  {  // Truning PA3 on if data is sending and spi is busy.  GPIOA->ODR |= GPIO\_ODR\_ODR3;  };  delay(500);  GPIOA->ODR &= ~GPIO\_ODR\_ODR3;  delay(500);  }  return 0;  }  void En\_clock(void)  {  RCC->APB2ENR |= RCC\_APB2ENR\_AFIOEN | RCC\_APB2ENR\_SPI1EN | RCC\_APB2ENR\_IOPAEN;  }  void gpio\_setup(void)  {  // PA0 as push button  GPIOA->CRL &= ~(GPIO\_CRL\_CNF0 | GPIO\_CRL\_MODE0);  GPIOA->CRL |= GPIO\_CRL\_CNF0\_0;  // PA1 as analog in put cnf=00, mode =00;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF1 | GPIO\_CRL\_MODE1);  GPIOA->CRL |= 0UL;  // PA2 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF2 | GPIO\_CRL\_MODE2);  GPIOA->CRL |= GPIO\_CRL\_MODE2;  // PA3 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF3 | GPIO\_CRL\_MODE3);  GPIOA->CRL |= GPIO\_CRL\_MODE3;  // PA4 as AFoutput cnf =10, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF4 | GPIO\_CRL\_MODE4);  GPIOA->CRL |= GPIO\_CRL\_CNF4\_1 | GPIO\_CRL\_MODE4;  // PA5 as AF output cnf =10, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF5 | GPIO\_CRL\_MODE5);  GPIOA->CRL |= GPIO\_CRL\_CNF5\_1 | GPIO\_CRL\_MODE5;  // PA6 as input cnf =01, mode = 00;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF6 | GPIO\_CRL\_MODE6);  GPIOA->CRL |= GPIO\_CRL\_CNF6\_0;  // PA7 as AF output cnf =10, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF7 | GPIO\_CRL\_MODE7);  GPIOA->CRL |= GPIO\_CRL\_CNF7\_1 | GPIO\_CRL\_MODE7;  }  void systick\_config(void)  {  SysTick->LOAD = 72000 - 1;  SysTick->VAL = 0;  SysTick->CTRL = SysTick\_CTRL\_CLKSOURCE | SysTick\_CTRL\_ENABLE;  }  void delay\_ms(void)  {  while (!(SysTick->CTRL & SysTick\_CTRL\_COUNTFLAG))  ;  }  void delay(uint32\_t count)  {  while (count--)  {  delay\_ms();  }  }  void SPI\_setup(void)  {  // Config as master device  SPI1->CR1 |= SPI\_CR1\_MSTR;  // Setting baud rate at lowest one 111  SPI1->CR1 |= SPI\_CR1\_BR;  // Enabling SS output in CR2. Hardware will pulls down NSS (PA4).  // And Transmition willbe strted autometic as soon as there is a load of data in SPI->DR  SPI1->CR2 |= SPI\_CR2\_SSOE;  // Enabling SPI  SPI1->CR1 |= SPI\_CR1\_SPE;  } |

# Slave configuration:

In CR1:

* Config clock
* Clear MSTR bit, set bit SPE.
* Enable SSOE. ( PA4 will be connected to masters PA4. Both are configured in Hardware NSS mode)
* Pin config:

|  |  |  |
| --- | --- | --- |
| **SPI Signal** | **Pin** | **config** |
| NSS | PA4 | Input float 0100 |
| SCK | PA5 | Input float 0100 |
| MISO | PA6 | Output (AF) push-pull 1011 |
| MOSI | PA7 | Input float 0100 |

* Check for RXNE flag. If set, dta is in SPI\_DR.