45 HAL SPI Generic Driver

45.1 SPI Firmware driver registers structures

45.1.1 SPI_HandleTypeDef

SPI_HandleTypeDef is defined in the stm32f4xx_hal_spi.h

Data Fields

- ***SPI_TypeDef * Instance***
- ***SPI_InitTypeDef Init***
- ***uint8_t * pTxBuffPtr***
- ***uint16_t TxXferSize***
- ***uint16_t TxXferCount***
- ***uint8_t * pRxBuffPtr***
- ***uint16_t RxXferSize***
- ***uint16_t RxXferCount***
- ***DMA_HandleTypeDef * hdmatx***
- ***DMA_HandleTypeDef * hdmarx***
- ***void(* RxISR***
- ***void(* TxISR***
- ***HAL_LockTypeDef Lock***
- ***__IO HAL_SPI_StateTypeDef State***
- ***__IO HAL_SPI_ErrorTypeDef ErrorCode***

Field Documentation

- ***SPI_TypeDef* SPI_HandleTypeDef::Instance***
- ***SPI_InitTypeDef SPI_HandleTypeDef::Init***
- ***uint8_t* SPI_HandleTypeDef::pTxBuffPtr***
- ***uint16_t SPI_HandleTypeDef::TxXferSize***
- ***uint16_t SPI_HandleTypeDef::TxXferCount***
- ***uint8_t* SPI_HandleTypeDef::pRxBuffPtr***
- ***uint16_t SPI_HandleTypeDef::RxXferSize***
- ***uint16_t SPI_HandleTypeDef::RxXferCount***
- ***DMA_HandleTypeDef* SPI_HandleTypeDef::hdmatx***
- ***DMA_HandleTypeDef* SPI_HandleTypeDef::hdmarx***
- ***void(* SPI_HandleTypeDef::RxISR)(struct __SPI_HandleTypeDef *hspi)***
- ***void(* SPI_HandleTypeDef::TxISR)(struct __SPI_HandleTypeDef *hspi)***
- ***HAL_LockTypeDef SPI_HandleTypeDef::Lock***
- ***__IO HAL_SPI_StateTypeDef SPI_HandleTypeDef::State***
- ***__IO HAL_SPI_ErrorTypeDef SPI_HandleTypeDef::ErrorCode***

45.1.2 SPI_InitTypeDef

SPI_InitTypeDef is defined in the stm32f4xx_hal_spi.h

Data Fields

- ***uint32_t Mode***
- ***uint32_t Direction***
- ***uint32_t DataSize***
- ***uint32_t CLKPolarity***
- ***uint32_t CLKPhase***
- ***uint32_t NSS***
- ***uint32_t BaudRatePrescaler***
- ***uint32_t FirstBit***
- ***uint32_t TIMode***
- ***uint32_t CRCCalculation***
- ***uint32_t CRCPolynomial***

Field Documentation

- ***uint32_t SPI_InitTypeDef::Mode***
  - Specifies the SPI operating mode. This parameter can be a value of ***SPI_mode***
- ***uint32_t SPI_InitTypeDef::Direction***
  - Specifies the SPI Directional mode state. This parameter can be a value of ***SPI_Direction_mode***
- ***uint32_t SPI_InitTypeDef::DataSize***
  - Specifies the SPI data size. This parameter can be a value of ***SPI_data_size***
- ***uint32_t SPI_InitTypeDef::CLKPolarity***
  - Specifies the serial clock steady state. This parameter can be a value of ***SPI_Clock_Polarity***
- ***uint32_t SPI_InitTypeDef::CLKPhase***
  - Specifies the clock active edge for the bit capture. This parameter can be a value of ***SPI_Clock_Phase***
- ***uint32_t SPI_InitTypeDef::NSS***
  - Specifies whether the NSS signal is managed by hardware (NSS pin) or by software using the SSI bit. This parameter can be a value of ***SPI_Slave_Select_management***
- ***uint32_t SPI_InitTypeDef::BaudRatePrescaler***
  - Specifies the Baud Rate prescaler value which will be used to configure the transmit and receive SCK clock. This parameter can be a value of ***SPI_BaudRate_Prescaler***
- ***uint32_t SPI_InitTypeDef::FirstBit***
  - Specifies whether data transfers start from MSB or LSB bit. This parameter can be a value of ***SPI_MSB_LSB_transmission***
- ***uint32_t SPI_InitTypeDef::TIMode***
  - Specifies if the TI mode is enabled or not. This parameter can be a value of ***SPI_TI_mode***
- ***uint32_t SPI_InitTypeDef::CRCCalculation***
  - Specifies if the CRC calculation is enabled or not. This parameter can be a value of ***SPI_CRC_Calculation***
- ***uint32_t SPI_InitTypeDef::CRCPolynomial***
  - Specifies the polynomial used for the CRC calculation. This parameter must be a number between Min_Data = 0 and Max_Data = 65535

45.1.3 SPI_TypeDef

SPI_TypeDef is defined in the stm32f439xx.h

Data Fields

- ***__IO uint32_t CR1***
- ***__IO uint32_t CR2***
- ***__IO uint32_t SR***
- ***__IO uint32_t DR***
- ***__IO uint32_t CRCPR***
- ***__IO uint32_t RXCRCR***
- ***__IO uint32_t TXCRCR***
- ***__IO uint32_t I2SCFGR***
- ***__IO uint32_t I2SPR***

Field Documentation

- ***__IO uint32_t SPI_TypeDef::CR1***
  - SPI control register 1 (not used in I2S mode), Address offset: 0x00
- ***__IO uint32_t SPI_TypeDef::CR2***
  - SPI control register 2, Address offset: 0x04
- ***__IO uint32_t SPI_TypeDef::SR***
  - SPI status register, Address offset: 0x08
- ***__IO uint32_t SPI_TypeDef::DR***
  - SPI data register, Address offset: 0x0C
- ***__IO uint32_t SPI_TypeDef::CRCPR***
  - SPI CRC polynomial register (not used in I2S mode), Address offset: 0x10
- ***__IO uint32_t SPI_TypeDef::RXCRCR***
  - SPI RX CRC register (not used in I2S mode), Address offset: 0x14
- ***__IO uint32_t SPI_TypeDef::TXCRCR***
  - SPI TX CRC register (not used in I2S mode), Address offset: 0x18
- ***__IO uint32_t SPI_TypeDef::I2SCFGR***
  - SPI_I2S configuration register, Address offset: 0x1C
- ***__IO uint32_t SPI_TypeDef::I2SPR***
  - SPI_I2S prescaler register, Address offset: 0x20

45.2 SPI Firmware driver API description

The following section lists the various functions of the SPI library.

45.2.1 How to use this driver

The SPI HAL driver can be used as follows:

1. Declare a SPI_HandleTypeDef handle structure, for example: SPI_HandleTypeDef hspi;
2. Initialize the SPI low level resources by implementing the HAL_SPI_MspInit ()API:
   a. Enable the SPIx interface clock

b. SPI pins configuration
   - Enable the clock for the SPI GPIOs
   - Configure these SPI pins as alternate function push-pull

c. NVIC configuration if you need to use interrupt process
   - Configure the SPIx interrupt priority
   - Enable the NVIC SPI IRQ handle

d. DMA Configuration if you need to use DMA process
   - Declare a DMA_HandleTypeDef handle structure for the transmit or receive stream
   - Enable the DMAx interface clock using
   - Configure the DMA handle parameters
   - Configure the DMA Tx or Rx Stream
   - Associate the initilalized hdma_tx handle to the hspi DMA Tx or Rx handle
   - Configure the priority and enable the NVIC for the transfer complete interrupt on the DMA Tx or Rx Stream

3. Program the Mode, Direction , Data size, Baudrate Prescaler, NSS management, Clock polarity and phase, FirstBit and CRC configuration in the hspi Init structure.
4. Initialize the SPI registers by calling the HAL_SPI_Init() API:
   - This API configures also the low level Hardware GPIO, CLOCK, CORTEX...etc) by calling the customed HAL_SPI_MspInit() API.

45.2.2 Initialization and de-initialization functions

This subsection provides a set of functions allowing to initialize and de-initialiaze the SPIx peripheral:

- User must implement HAL_SPI_MspInit() function in which he configures all related peripherals resources (CLOCK, GPIO, DMA, IT and NVIC).
- Call the function HAL_SPI_Init() to configure the selected device with the selected configuration:
  - Mode
  - Direction
  - Data Size
  - Clock Polarity and Phase
  - NSS Management
  - BaudRate Prescaler
  - FirstBit
  - TIMode
  - CRC Calculation
  - CRC Polynomial if CRC enabled
- Call the function HAL_SPI_DeInit() to restore the default configuration of the selected SPIx periperal.
- ***HAL_SPI_Init()***
- ***HAL_SPI_DeInit()***
- ***HAL_SPI_MspInit()***
- ***HAL_SPI_MspDeInit()***

45.2.3 IO operation functions

The SPI supports master and slave mode :

1. There are two modes of transfer:

- Blocking mode: The communication is performed in polling mode. The HAL status of all data processing is returned by the same function after finishing transfer.
- No-Blocking mode: The communication is performed using Interrupts or DMA, These APIs return the HAL status. The end of the data processing will be indicated through the dedicated SPI IRQ when using Interrupt mode or the DMA IRQ when using DMA mode. The HAL_SPI_TxCpltCallback(), HAL_SPI_RxCpltCallback() and HAL_SPI_TxRxCpltCallback() user callbacks will be executed respectivelly at the end of the transmit or Receive process The HAL_SPI_ErrorCallback()user callback will be executed when a communication error is detected

2. Blocking mode APIs are :
   - HAL_SPI_Transmit()in 1Line (simplex) and 2Lines (full duplex) mode
   - HAL_SPI_Receive() in 1Line (simplex) and 2Lines (full duplex) mode
   - HAL_SPI_TransmitReceive() in full duplex mode
3. Non Blocking mode API's with Interrupt are :
   - HAL_SPI_Transmit_IT()in 1Line (simplex) and 2Lines (full duplex) mode
   - HAL_SPI_Receive_IT() in 1Line (simplex) and 2Lines (full duplex) mode
   - HAL_SPI_TransmitReceive_IT()in full duplex mode
   - HAL_SPI_IRQHandler()
4. Non Blocking mode functions with DMA are :
   - HAL_SPI_Transmit_DMA()in 1Line (simplex) and 2Lines (full duplex) mode
   - HAL_SPI_Receive_DMA() in 1Line (simplex) and 2Lines (full duplex) mode
   - HAL_SPI_TransmitReceie_DMA() in full duplex mode
5. A set of Transfer Complete Callbacks are provided in non Blocking mode:
   - HAL_SPI_TxCpltCallback()
   - HAL_SPI_RxCpltCallback()
   - HAL_SPI_ErrorCallback()
   - HAL_SPI_TxRxCpltCallback()

- ***HAL_SPI_Transmit()***
- ***HAL_SPI_Receive()***
- ***HAL_SPI_TransmitReceive()***
- ***HAL_SPI_Transmit_IT()***
- ***HAL_SPI_Receive_IT()***
- ***HAL_SPI_TransmitReceive_IT()***
- ***HAL_SPI_Transmit_DMA()***
- ***HAL_SPI_Receive_DMA()***
- ***HAL_SPI_TransmitReceive_DMA()***
- ***HAL_SPI_IRQHandler()***
- ***HAL_SPI_TxCpltCallback()***
- ***HAL_SPI_RxCpltCallback()***
- ***HAL_SPI_TxRxCpltCallback()***
- ***HAL_SPI_ErrorCallback()***

45.2.4 Peripheral State and Errors functions

This subsection provides a set of functions allowing to control the SPI.

- HAL_SPI_GetState() API can be helpful to check in run-time the state of the SPI peripheral
- HAL_SPI_GetError() check in run-time Errors occurring during communication
- ***HAL_SPI_GetState()***
- ***HAL_SPI_GetError()***

45.2.5 Initialization and de-initialization functions

45.2.5.1 HAL_SPI_Init

Function Name	**HAL_StatusTypeDef HAL_SPI_Init (*SPI_HandleTypeDef** * hspi)**
Function Description	Initializes the SPI according to the specified parameters in the SPI_InitTypeDef and create the associated handle.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• **HAL status**
Notes	• None.

45.2.5.2 HAL_SPI_DeInit

Function Name	**HAL_StatusTypeDef HAL_SPI_DeInit (*SPI_HandleTypeDef** * hspi)**
Function Description	DeInitializes the SPI peripheral.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• **HAL status**
Notes	• None.

45.2.5.3 HAL_SPI_MspInit

Function Name	**void HAL_SPI_MspInit (*SPI_HandleTypeDef** * hspi)**
Function Description	SPI MSP Init.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• None.
Notes	• None.

45.2.5.4 HAL_SPI_MspDeInit

Function Name	**void HAL_SPI_MspDeInit (*SPI_HandleTypeDef* * hspi)**
Function Description	SPI MSP DeInit.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• None.
Notes	• None.

45.2.6 IO operation functions

45.2.6.1 HAL_SPI_Transmit

Function Name	**HAL_StatusTypeDef HAL_SPI_Transmit (*SPI_HandleTypeDef* * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)**
Function Description	Transmit an amount of data in blocking mode.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pData :** pointer to data buffer • **Size :** amount of data to be sent • **Timeout :** Timeout duration
Return values	• **HAL status**
Notes	• None.

45.2.6.2 HAL_SPI_Receive

Function Name	**HAL_StatusTypeDef HAL_SPI_Receive (*SPI_HandleTypeDef* * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)**

Function Description	Receive an amount of data in blocking mode.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pData :** pointer to data buffer • **Size :** amount of data to be sent • **Timeout :** Timeout duration
Return values	• **HAL status**
Notes	• None.

45.2.6.3 HAL_SPI_TransmitReceive

Function Name	**HAL_StatusTypeDef HAL_SPI_TransmitReceive (*SPI_HandleTypeDef ** hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)**
Function Description	Transmit and Receive an amount of data in blocking mode.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pTxData :** pointer to transmission data buffer • **pRxData :** pointer to reception data buffer to be • **Size :** amount of data to be sent • **Timeout :** Timeout duration
Return values	• **HAL status**
Notes	• None.

45.2.6.4 HAL_SPI_Transmit_IT

Function Name	**HAL_StatusTypeDef HAL_SPI_Transmit_IT (*SPI_HandleTypeDef ** hspi, uint8_t * pData, uint16_t Size)**
Function Description	Transmit an amount of data in no-blocking mode with Interrupt.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pData :** pointer to data buffer • **Size :** amount of data to be sent
Return values	• **HAL status**

Notes	• None.

45.2.6.5 HAL_SPI_Receive_IT

Function Name	**HAL_StatusTypeDef HAL_SPI_Receive_IT (*SPI_HandleTypeDef** * hspi, uint8_t * pData, uint16_t Size)**
Function Description	Receive an amount of data in no-blocking mode with Interrupt.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pData :** pointer to data buffer • **Size :** amount of data to be sent
Return values	• **HAL status**
Notes	• None.

45.2.6.6 HAL_SPI_TransmitReceive_IT

Function Name	**HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT (*SPI_HandleTypeDef** * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)**
Function Description	Transmit and Receive an amount of data in no-blocking mode with Interrupt.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pTxData :** pointer to transmission data buffer • **pRxData :** pointer to reception data buffer to be • **Size :** amount of data to be sent
Return values	• **HAL status**
Notes	• None.

45.2.6.7 HAL_SPI_Transmit_DMA

Function Name	**HAL_StatusTypeDef HAL_SPI_Transmit_DMA (*SPI_HandleTypeDef** * **hspi, uint8_t * pData, uint16_t Size)**
Function Description	Transmit an amount of data in no-blocking mode with DMA.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pData :** pointer to data buffer • **Size :** amount of data to be sent
Return values	• **HAL status**
Notes	• None.

45.2.6.8 HAL_SPI_Receive_DMA

Function Name	**HAL_StatusTypeDef HAL_SPI_Receive_DMA (*SPI_HandleTypeDef** * **hspi, uint8_t * pData, uint16_t Size)**
Function Description	Receive an amount of data in no-blocking mode with DMA.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pData :** pointer to data buffer
Parameters	• **Size :** amount of data to be sent
Return values	• **HAL status**
Notes	• When the CRC feature is enabled the pData Length must be Size + 1.

45.2.6.9 HAL_SPI_TransmitReceive_DMA

Function Name	**HAL_StatusTypeDef HAL_SPI_TransmitReceive_DMA (*SPI_HandleTypeDef** * **hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)**
Function Description	Transmit and Receive an amount of data in no-blocking mode with DMA.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module. • **pTxData :** pointer to transmission data buffer

- **pRxData :** pointer to reception data buffer

Parameters
- **Size :** amount of data to be sent

Return values
- **HAL status**

Notes
- When the CRC feature is enabled the pRxData Length must be Size + 1

45.2.6.10 HAL_SPI_IRQHandler

Function Name **void HAL_SPI_IRQHandler (*SPI_HandleTypeDef** * hspi)**

Function Description This function handles SPI interrupt request.

Parameters
- **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.

Return values
- **HAL status**

Notes
- None.

45.2.6.11 HAL_SPI_TxCpltCallback

Function Name **void HAL_SPI_TxCpltCallback (*SPI_HandleTypeDef** * hspi)**

Function Description Tx Transfer completed callbacks.

Parameters
- **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.

Return values
- None.

Notes
- None.

45.2.6.12 HAL_SPI_RxCpltCallback

Function Name **void HAL_SPI_RxCpltCallback (*SPI_HandleTypeDef** * hspi)**

Function Description	Rx Transfer completed callbacks.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• None.
Notes	• None.

45.2.6.13 HAL_SPI_TxRxCpltCallback

Function Name	**void HAL_SPI_TxRxCpltCallback (*SPI_HandleTypeDef* * hspi)**
Function Description	Tx and Rx Transfer completed callbacks.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• None.
Notes	• None.

45.2.6.14 HAL_SPI_ErrorCallback

Function Name	**void HAL_SPI_ErrorCallback (*SPI_HandleTypeDef* * hspi)**
Function Description	SPI error callbacks.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• None.
Notes	• None.

45.2.7 Peripheral State and Errors functions

45.2.7.1 HAL_SPI_GetState

Function Name	**HAL_SPI_StateTypeDef HAL_SPI_GetState (*SPI_HandleTypeDef** * **hspi)**
Function Description	Return the SPI state.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• **HAL state**
Notes	• None.

45.2.7.2 HAL_SPI_GetError

Function Name	**HAL_SPI_ErrorTypeDef HAL_SPI_GetError (*SPI_HandleTypeDef** * **hspi)**
Function Description	Return the SPI error code.
Parameters	• **hspi :** pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
Return values	• **SPI Error Code**
Notes	• None.

45.3 SPI Firmware driver defines

45.3.1 SPI

SPI

SPI_BaudRate_Prescaler

- #define: ***SPI_BAUDRATEPRESCALER_2 ((uint32_t)0x00000000)***
- #define: ***SPI_BAUDRATEPRESCALER_4 ((uint32_t)0x00000008)***
- #define: ***SPI_BAUDRATEPRESCALER_8 ((uint32_t)0x00000010)***

- #define: ***SPI_BAUDRATEPRESCALER_16 ((uint32_t)0x00000018)***

- #define: ***SPI_BAUDRATEPRESCALER_32 ((uint32_t)0x00000020)***

- #define: ***SPI_BAUDRATEPRESCALER_64 ((uint32_t)0x00000028)***

- #define: ***SPI_BAUDRATEPRESCALER_128 ((uint32_t)0x00000030)***

- #define: ***SPI_BAUDRATEPRESCALER_256 ((uint32_t)0x00000038)***

SPI_Clock_Phase

- #define: ***SPI_PHASE_1EDGE ((uint32_t)0x00000000)***

- #define: ***SPI_PHASE_2EDGE SPI_CR1_CPHA***

SPI_Clock_Polarity

- #define: ***SPI_POLARITY_LOW ((uint32_t)0x00000000)***

- #define: ***SPI_POLARITY_HIGH SPI_CR1_CPOL***

SPI_CRC_Calculation

- #define: ***SPI_CRCCALCULATION_DISABLED ((uint32_t)0x00000000)***

- #define: ***SPI_CRCCALCULATION_ENABLED SPI_CR1_CRCEN***

SPI_data_size

- #define: ***SPI_DATASIZE_8BIT ((uint32_t)0x00000000)***

- #define: ***SPI_DATASIZE_16BIT SPI_CR1_DFF***

SPI_Direction_mode

- #define: ***SPI_DIRECTION_2LINES ((uint32_t)0x00000000)***

- #define: ***SPI_DIRECTION_2LINES_RXONLY SPI_CR1_RXONLY***

- #define: ***SPI_DIRECTION_1LINE SPI_CR1_BIDIMODE***

SPI_Flag_definition

- #define: ***SPI_FLAG_RXNE SPI_SR_RXNE***

- #define: ***SPI_FLAG_TXE SPI_SR_TXE***

- #define: ***SPI_FLAG_CRCERR SPI_SR_CRCERR***

- #define: ***SPI_FLAG_MODF SPI_SR_MODF***

- #define: ***SPI_FLAG_OVR SPI_SR_OVR***

- #define: ***SPI_FLAG_BSY SPI_SR_BSY***

- #define: ***SPI_FLAG_FRE SPI_SR_FRE***

SPI_Interrupt_configuration_definition

- #define: ***SPI_IT_TXE SPI_CR2_TXEIE***

- #define: ***SPI_IT_RXNE SPI_CR2_RXNEIE***

- #define: ***SPI_IT_ERR SPI_CR2_ERRIE***

SPI_mode

- #define: ***SPI_MODE_SLAVE ((uint32_t)0x00000000)***

- #define: ***SPI_MODE_MASTER (SPI_CR1_MSTR | SPI_CR1_SSI)***

SPI_MSB_LSB_transmission

- #define: ***SPI_FIRSTBIT_MSB ((uint32_t)0x00000000)***

- #define: ***SPI_FIRSTBIT_LSB SPI_CR1_LSBFIRST***

SPI_Slave_Select_management

- #define: ***SPI_NSS_SOFT SPI_CR1_SSM***

- #define: ***SPI_NSS_HARD_INPUT ((uint32_t)0x00000000)***

- #define: ***SPI_NSS_HARD_OUTPUT ((uint32_t)0x00040000)***

SPI_TI_mode

- #define: ***SPI_TIMODE_DISABLED ((uint32_t)0x00000000)***

- #define: ***SPI_TIMODE_ENABLED SPI_CR2_FRF***