SPI (SERIAL PERIPHERAL INTERFACE)

1. Introduction

En fonction du type de famille et du package utilisé, les microcontrôleurs STM32 peuvent fournir jusqu'à six périphériques SPI indépendants. Un module SPI est une interface série synchrone, utile pour communiquer avec d'autres périphériques ou microcontrôleurs. Ces périphériques peuvent être des EEPROM série, des registres à décalage, des pilotes d'affichage, des convertisseurs A / N, etc. Le module SPI prend en charge à la fois le mode maître et le mode esclave. En mode maître, l'STM32 fournit l'horloge pour toutes les transactions, tandis qu'en mode esclave le périphérique externe lui fournit l'horloge.

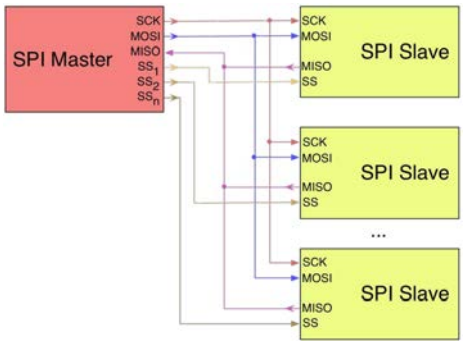
Selon le protocole SPI, l'interconnexion du microcontrôleur avec un autre périphérique nécessite généralement, trois broches :

- Master Out Slave In date (MOSI)
- Master In Slave Out data (MISO)
- Serial Clock (SCK)

En outre, une quatrième broche peut être utilisée pour la sélection de l'esclave :

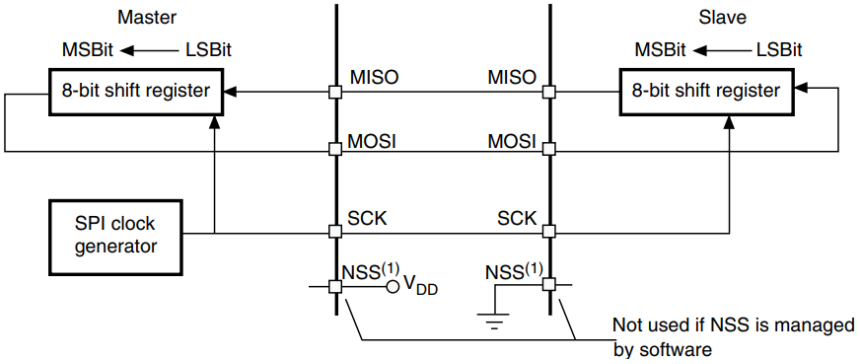
- Slave Select (NSS)

En mode maître, le microcontrôleur peut être connecté à plus qu'un esclave ; si c'est le cas, d'autres broches d'E/S doivent être utilisées pour la sélection des esclaves. La Figure suivante montre un exemple de connexion (full duplex) du microcontrôleur avec deux esclaves.



1.1. Principales caractéristiques du module SPI

- Transferts synchrones en duplex intégral sur trois lignes
- Transferts synchrones simplex sur deux lignes avec ou sans ligne de données bidirectionnelle.
- Sélection du format de trame de transfert 8 ou 16 bits.
- Fonctionnement maître ou esclave
- Vitesse maximale en mode maître ou esclave $f_{PCLK}/2$
- Gestion de NSS par matériel ou logiciel pour maître et esclave
- Polarité et phase d'horloge programmables
- Ordre des données programmable avec décalage MSB ou LSB en premier.



1.2. Polarite et phase de l'horloge

Outre le réglage de la fréquence d'horloge du bus, le maître et les esclaves doivent également s'accorder sur la polarité et la phase d'horloge par rapport aux données échangées sur les lignes MOSI et MISO. La spécification SPI de Motorola nomme ces deux paramètres respectivement CPOL et CPHA. Les combinaisons de polarité et de phase sont souvent appelées modes de bus SPI qui sont généralement numérotés selon le tableau suivant.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

- A CPOL = 0, l'état initial de l'horloge est zéro, c'est-à-dire que l'état actif est 1 et l'état inactif est 0.

  - Pour CPHA = 0, les données sont capturées sur le front montant du SCK (transition LOW → HIGH) et les données en sortie changent sur un front descendant (transition d'horloge HIGH → LOW).

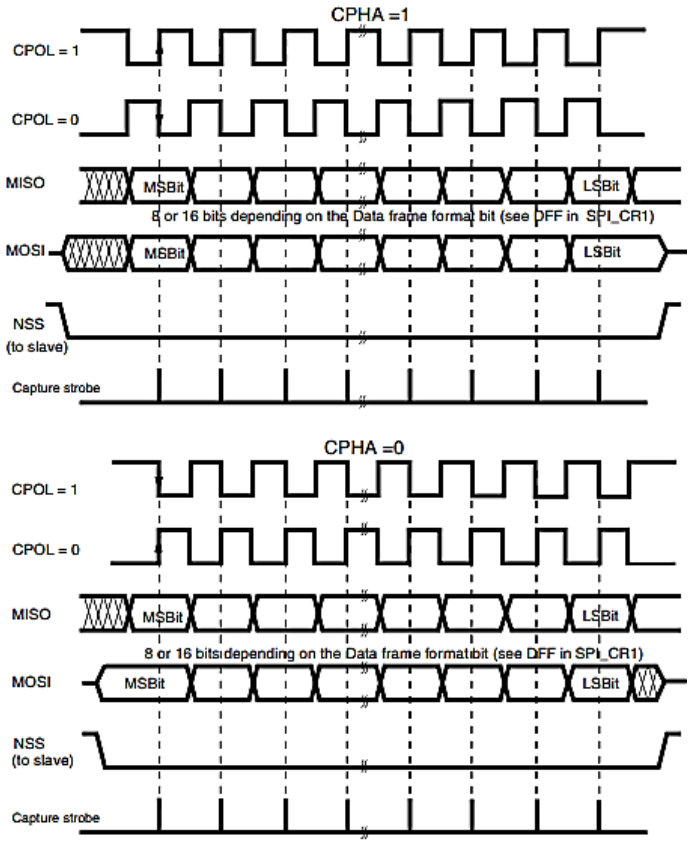
  - Pour CPHA = 1, les données sont capturées sur le front descendant du SCK et les données en sortie changent sur un front montant.

- A CPOL = 1, l'état initial de l'horloge est un (inversion de CPOL = 0), c'est-à-dire que l'état actif est 0 et l'état inactif est 1.

  - Pour CPHA = 0, les données sont capturées sur le front descendant du SCK et les données en sortie changent sur un front montant.

  - Pour CPHA = 1, les données sont capturées sur le front montant du SCK et les données en sortie changent sur un front descendant

La figure suivante montre le transfert SPI avec les quatre combinaisons de bits CPHA et CPOL. Le diagramme peut être interprété comme un chronogramme maître ou esclave où la broche SCK, la broche MISO et la broche MOSI sont directement connectées entre les équipements maître et esclave.



2. Gestion de l'SPI avec la librairie HAL

2.1. Initialisation du module SPI

Le librairie CubeHAL définie trois structures pour la gestion du module SPI. La structure *SPI_TypeDef* contient la définition des registres, la structure *SPI_InitTypeDef* définie les paramètres du module SPI et la structure *SPI_HandleTypeDef* contient les données nécessaires pour l'exploitation du module SPI :

```
typedef struct {
uint32_t Mode;         /* Spécifie le mode l'SPI */
                              SPI_MODE_SLAVE
                              SPI_MODE_MASTER
uint32_t Direction;    /* Spécifie le SPI mode Uni/bidirectionnel */
                              SPI_DIRECTION_2LINES
                              SPI_DIRECTION_2LINES_RXONLY
                              SPI_DIRECTION_1LINE
uint32_t DataSize;     /* Spécifie la taille de la donnée */
                              SPI_DATASIZE_8BIT
                              SPI_DATASIZE_16BIT
uint32_t CLKPolarity; /* Spécifie la polarité de l'horloge */
                              SPI_POLARITY_LOW
                              SPI_POLARITY_HIGH
uint32_t CLKPhase;     /* Spécifie le front actif pour la capture */
                              SPI_PHASE_1EDGE
                              SPI_PHASE_2EDGE
uint32_t NSS;          /* Spécifie le gestion du signal NSS matériel (NSS pin) ou logiciel */
                              SPI_NSS_SOFT
                              SPI_NSS_HARD_INPUT
                              SPI_NSS_HARD_OUTPUT
uint32_t BaudRatePrescaler;  /* Spécifie la valeur du prédiviseur d'horloge */
                              SPI_BAUDRATEPRESCALER_2
                              SPI_BAUDRATEPRESCALER_4
                              SPI_BAUDRATEPRESCALER_8
                              SPI_BAUDRATEPRESCALER_16
                              SPI_BAUDRATEPRESCALER_32
```

```
                            SPI_BAUDRATEPRESCALER_64
                            SPI_BAUDRATEPRESCALER_128
                            SPI_BAUDRATEPRESCALER_256
uint32_t FirstBit;    /* Spécifie si le transfert de données commence par l'MSB ou l'LSB */
                            SPI_FIRSTBIT_MSB
                            SPI_FIRSTBIT_LSB
uint32_t TIMode;      /* Spécifie si le mode TI est valide ou non */
                            SPI_TIMODE_DISABLE
uint32_t CRCCalculation; /* validation du calcul CRC */
uint32_t CRCPolynomial; /* Spécifie le polynôme  utilisé pour la calcul de CRC */
} SPI_InitTypeDef;
```

La structure *SPI_HandleTypeDef* est définie comme suit :

```
typedef struct __SPI_HandleTypeDef
{ SPI_TypeDef                *Instance;      /*!< SPI registers base address              */
  SPI_InitTypeDef            Init;          /*!< SPI communication parameters            */
  uint8_t                    *pTxBuffPtr;   /*!< Pointer to SPI Tx transfer Buffer       */
  uint16_t                   TxXferSize;    /*!< SPI Tx Transfer size                    */
  __IO uint16_t              TxXferCount;   /*!< SPI Tx Transfer Counter                 */
  uint8_t                    *pRxBuffPtr;   /*!< Pointer to SPI Rx transfer Buffer       */
  uint16_t                   RxXferSize;    /*!< SPI Rx Transfer size                    */
  __IO uint16_t              RxXferCount;   /*!< SPI Rx Transfer Counter                 */
  void (*RxISR)(struct __SPI_HandleTypeDef *hspi);   /*!< function pointer on Rx ISR      */
  void (*TxISR)(struct __SPI_HandleTypeDef *hspi);   /*!< function pointer on Tx ISR      */
  DMA_HandleTypeDef          *hdmatx;       /*!< SPI Tx DMA Handle parameters            */
  DMA_HandleTypeDef          *hdmarx;       /*!< SPI Rx DMA Handle parameters            */
  HAL_LockTypeDef            Lock;          /*!< Locking object                          */
  __IO HAL_SPI_StateTypeDef  State;         /*!< SPI communication state                 */
  __IO uint32_t              ErrorCode;     /*!< SPI Error code } UART_HandleTypeDef;
} SPI_HandleTypeDef;
```

La fonction *HAL_StatusTypeDef HAL_SPI_Init(UART_HandleTypeDef *hspi)* permet d'affecter les paramètres de la structure ***SPI_HandleTypeDef*** aux registres du périphérique SPI.

2.2. Echange de messages avec le module SPI

Une fois le périphérique SPI configuré, nous pouvons commencer à échanger des données avec des équipements esclaves. Le CubeHAL offre trois façons de communiquer sur un bus SPI : polling, interruption et le mode DMA. Nous présentons ici le mode polling (scrutation).

TRANSMISSION D'UN MESSAGE

HAL_StatusTypeDef HAL_SPI_Transmit(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size, uint32_t Timeout) ;

La structure de la fonction est presque identique aux autres routines de communication utilisées pour la manipulation de l'UART. Cette fonction peut être utilisée si le périphérique SPI est configuré pour fonctionner à la fois en modes SPI_DIRECTION_1LINE ou SPI_DIRECTION_ 2LINES.

RECEPTION D'UN MASSAGE

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

Cette fonction peut être utilisée dans les trois modes de direction.

TRANSMISSION ET RECEPTION

Si le périphérique esclave prend en charge le mode duplex intégral, nous pouvons utiliser la fonction :

HAL_StatusTypeDef HAL_SPI_TransmitReceive(SPI_HandleTypeDef *hspi, uint8_t *pTxData, uint8_t *pRxData, uint16_t Size, uint32_t Timeout) ;