TD2

EXERCICE 1

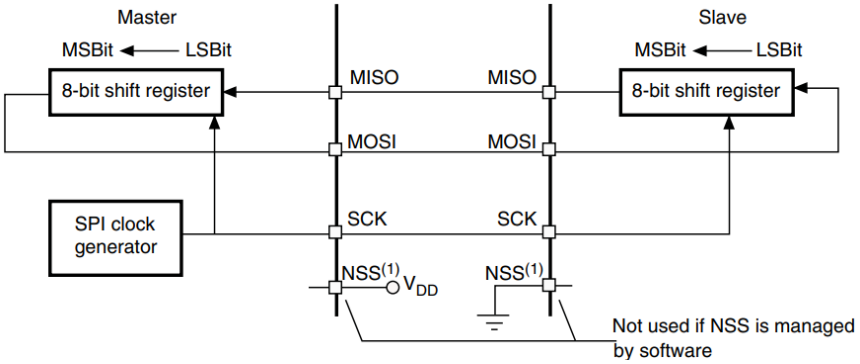
Le MCP3201est un convertisseur analogique numérique à sorte série compatible SPI. Son datasheet des donné par le lien : https://ww1.microchip.com/downloads/en/DeviceDoc/21290F.pdf

1. Donner la résolution de ce convertisseur,
2. Pour une tension de référence de 4,096V :
    a. Donner le pas de quantification,
    b. Donner la valeur numérique correspondant à une tension d'entrée de 2V.
3. Pour une tension d'alimentation de 5V :
    a. Donner la fréquence maximale de l'horloge SCK.
    b. Donner le temps de conversion total en fonction de SCK
4. Un exemple d'utilisation de ce circuit est donné à la figure 6-3 de la page 22.
    a. Quel est le rôle du montage autour du circuit MCP601 ?
    b. Quel est le rôle du circuit MCP1541.
5. Expliquer brièvement les chronogrammes des figures 6-1 et 6-2 à la page 21.
6. Tirer d'après ces diagrammes le contenu des l'octet haut et l'octet bas.
7. Ecrire la fonction d'initialisation du module SPI pour les paramètres suivants :
    - Horloge système 100MHz
    - SPI mode maître, full duplex
    - Fréquence FSCK ≈ 1MHz
    - CPOL = 1
    - CPHA = 1
    - Donnée sur 8 bits
    - Sélection de l'esclave logicielle à travers la broche PB8
    - Le bit MSB en premier
8. Ecrire le pseudocode de la conversion d'un échantillon, le résultat de conversion est chargé dans la variable ADC_VAL

EXERCICE 2

On se propose de faire l'étude du circuit RTC DS1307. Le datasheet est téléchargeable à travers le lien suivant : https://datasheets.maximintegrated.com/en/ds/DS1307.pdf

1. Donner les caractéristiques de ce circuit,
2. A quoi sert la zone de RAM non volatile de 56 octets
3. Quelle est la fréquence maximale de SCK ?
4. Ecrire la procédure d'initialisation du module SPI
5. Ecrire le procédure de réglage de la date et l'heure selon les paramètres suivants :
    - Heure : 15h 30nm 53s
    - Date : 11/12/2020
6. Ecrire la procédure de lecture de la date et l'heure chaque seconde.



1.1. 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).

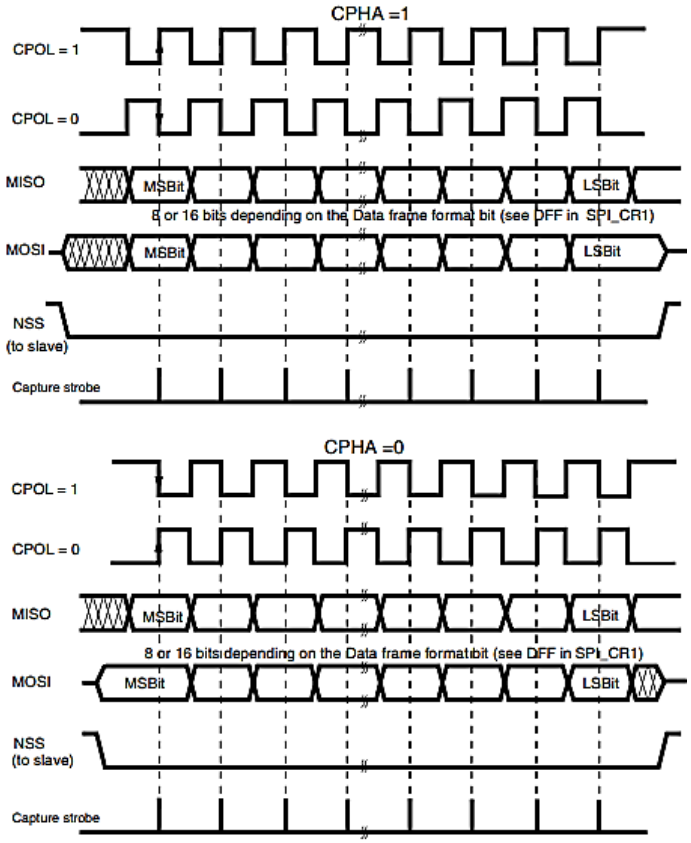
  - Pour CPHA = 1, les données sont capturées sur le front descendant du SCK et les données en sortie changent sur un front montant.

- 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) ;