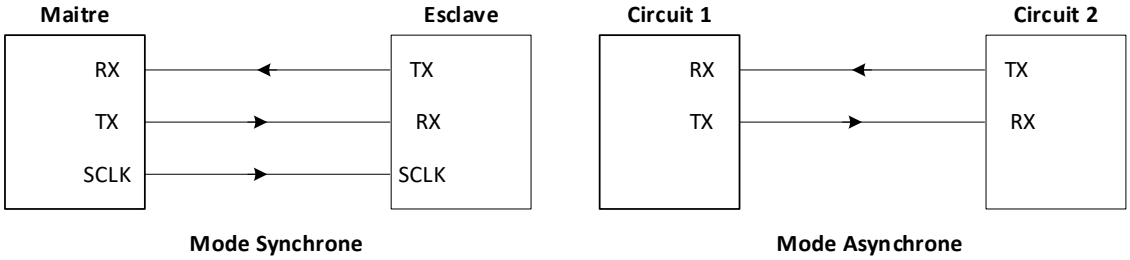
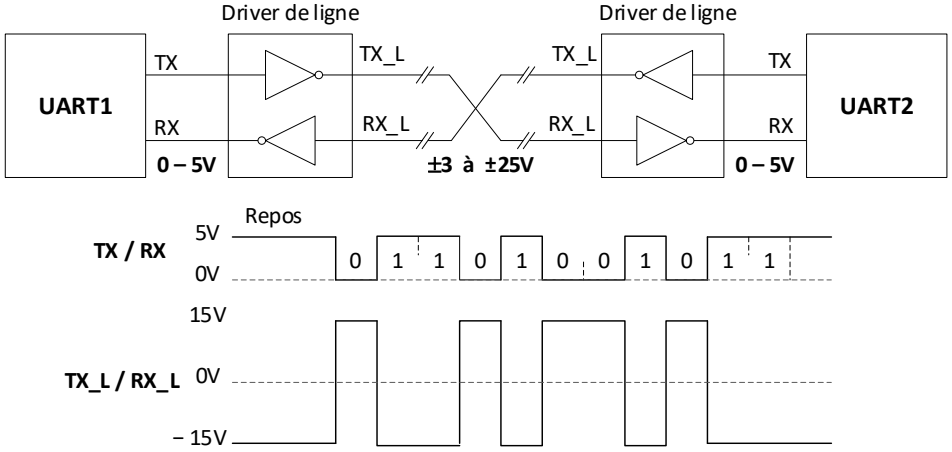
USART (UNIVERSAL SYNCHRONOUS RECEIVER - TRANSMITTER)

Ce type d'interface permet au microcontrôleur de communiquer avec d'autres systèmes à base de microprocesseur. Les données envoyées ou reçues se présentent sous la forme d'une succession temporelle (sur un seul bit) de valeurs binaires images d'un mot. Il y a deux types de liaison série : synchrone et asynchrone. Nous traitons dans ce chapitre la liaison série asynchrone.



La liaison série asynchrone ne possède pas un signal d'horloge de synchronisation. Les unités en liaison possèdent chacune une horloge interne cadencée à la même fréquence. Lorsqu'une unité veut émettre un mot binaire, elle génère un front descendant sur sa ligne émettrice. A la fin de l'émission de ce mot, la ligne repasse au niveau haut. La donnée à transmettre peut contenir un bit supplémentaire appelé "bit de parité" servant à la correction d'erreurs.

Paramètres entrant en jeu pour la norme RS232 :
- Longueur des mots : 7 bits (ex : caractère ASCII) ou 8 bits
- La vitesse de transmission : elle est définie en bits par seconde ou bauds. Elle peut prendre des valeurs allant de 110 à 115 200 bps.
- Parité : le mot transmis peut être suivi ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission.
- Bit de start : la ligne au repos est à l'état logique 1 pour indiquer qu'un mot va être transmis la ligne passe à l'état bas avant de commencer le transfert. Ce bit permet de synchroniser l'horloge du récepteur.
- Bit de stop : après la transmission, la ligne est positionnée au repos pendant 1, 2 ou 1,5 périodes d'horloge selon le nombre de bits de stop.
- Niveau de tension : Un "0" logique est matérialisé par une tension comprise entre 3 et 25V, un "1" par une tension comprise entre -25 et -3 V. Des circuits spécialisés comme le MAX 232 réalise la conversion à partir de niveau TTL.



Les microcontrôleurs STM32 fournissent un certain nombre d'USART, qui peuvent être configurés pour fonctionner en mode synchrone ou asynchrone. Le microcontrôleur STM32F407 dispose de quatre USART (USART1, USART2, USART3 et USART6) et deux UART (UART4 et UART5). La plupart des USART sont également capables d'implémenter automatiquement le contrôle de flux matériel, à la fois pour la norme RS232 et RS485.

Le CubeHAL sépare l'API pour la gestion des interfaces UART et USART. Toutes les fonctions de type C et descripteurs utilisés pour la gestion des USART commencent par le préfixe HAL_USART et sont contenus dans les fichiers stm32xxx_hal_usart. {C, h}, tandis que ceux liés à la gestion des UART commencent par le préfixe HAL_UART et sont contenus dans les fichiers stm32xxx_hal_uart. {c, h}. Nous étudions ici les fonctionnalités du module HAL_UART.

1. Initialisation de l'UART

Comme tous les périphériques STM32, même les USART sont mappés dans la région périphérique en mémoire, qui commence à 0x4000 0000. Le CubeHAL fait abstraction de l'emplacement effectif de chaque USART pour un MCU STM32 donné grâce au descripteur *USART_TypeDef*. La structure *UART_InitTypeDef* définie les paramètres en mode UART :

```
typedef struct {
uint32_t    BaudRate;        /* vitesse de trnamission en bit par seconde */
                              2400, 9600, 19200, 38400, 57600, 115200, ...
uint32_t    WordLength;      /* longueur du mot à envoyer/recevoir */
                                    UART_WORDLENGTH_8B
                                    UART_WORDLENGTH_9B
uint32_t    StopBits;        /* bits de STOP */
                                    UART_STOPBITS_1
                                    UART_STOPBITS_2
uint32_t    Parity;          /* bit de parité */
                                    UART_PARITY_NONE
                                    UART_PARITY_EVEN
                                    UART_PARITY_ODD
uint32_t    Mode;            /* Emission, Réception, ou Emission/Réception */
                                    UART_MODE_RX
                                    UART_MODE_TX
                                    UART_MODE_TX_RX
uint32_t    HwFlowCtl;       /* Contrôle de flux */
                                    UART_HWCONTROL_NONE
                                    UART_HWCONTROL_RTS
                                    UART_HWCONTROL_CTS
                                    UART_HWCONTROL_RTS_CTS
uint32_t    OverSampling;    /* Echantillonnage de l'entrée en réception */
                                    UART_OVERSAMPLING_16
                                    UART_OVERSAMPLING_8
} UART_InitTypeDef
```

De façon similaire à l'ADC, la structure *UART_HandleTypeDef*, fournit les paramètres nécessaires à la gestion du module UART.

```
typedef struct {
USART_TypeDef     *Instance;        /* UART registers base address */
UART_InitTypeDef   Init;            /* UART communication parameters */
uint8_t           *pTxBuffPtr;      /* Pointer to UART Tx transfer Buffer */
```

```
uint16_t    TxXferSize;      /* UART Tx Transfer size */
uint16_t    TxXferCount;     /* UART Tx Transfer Counter */
uint8_t     *pRxBuffPtr;     /* Pointer to UART Rx transfer Buffer */
uint16_t    RxXferSize;      /* UART Rx Transfer size */
uint16_t    RxXferCount;     /* UART Rx Transfer Counter */
DMA_HandleTypeDef *hdmatx;   /* UART Tx DMA Handle parameters */
DMA_HandleTypeDef *hdmarx;   /* UART Rx DMA Handle parameters */
HAL_LockTypeDef Lock;        /* Locking object */
__IO HAL_UART_StateTypeDef gState;         /* UART communication global state Tx */
__IO HAL_UART_StateTypeDef RxState;        /* UART communication state Rx */
__IO HAL_UART_ErrorTypeDef ErrorCode;      /* UART Error code */
} UART_HandleTypeDef;
```

La fonction *HAL_StatusTypeDef HAL_UART_Init(UART_HandleTypeDef *huart)* permet d'affecter les paramètres de la structure ***UART_HandleTypeDef*** aux registres du périphérique USART.

Transmission d'un message :

HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)

Réception d'un massage :

HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)