SOLUTION EXERCICE 7-1 : UART & HANDSHAKE

OBJECTIFS

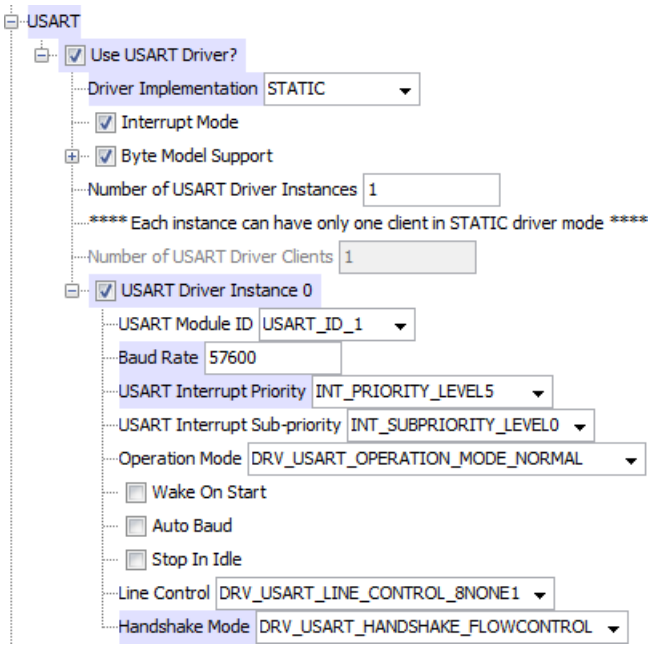
Cet exercice a pour objectif de permettre aux étudiants d'utiliser le handshaking "automatique" de l'USART et d'observer concrètement le comportement de ce mécanisme en situation de surcharge.

a) REALISATION PRATIQUE

Copiez votre projet labo Tp2_PWM&RS232 sous C:\microchip\harmony\v<n>\apps\MINF\Exercices et renommez le Ex7_1.

MODIFICATION DE LA CONFIGURATION

Au niveau de la configuration de l'USART, il faut établir la configuration comme ci-dessous (changement du "Handshake mode") :



Avec cette configuration, les signaux de contrôle de flux sont maintenant gérés automatiquement par hardware.

Info du datasheet du PIC, chapitre USART :
Le signal RTS est drivé à 0 (prêt à recevoir) quand l'usart a la place pour 2 caractères dans son FIFO.

MODIFICATIONS DU CODE

- Régler l'interruption de réception lorsque le buffer à moitié plein (= 4 bytes sur les 8 du buffer). Il est nécessaire de le rajouter à la main dans le code d'initialisation de l'usart si pas disponible via Harmony.
- Enlever le contrôle de flux géré par software : tous les tests de la valeur de l'entrée CTS et les assignations de la sortie RTS.
    - Interruption RX
    - GetMessage
    - Interruption TX
    - SendMessage
    - InitFifoComm
- Pour s'assurer que le contrôle de flux entre en action, votre application doit envoyer une trame à chaque cycle (pas un cycle sur 5 comme demandé au TP).
- Attention également à gérer les activations, désactivations et clear des interruptions TX et RX.

MODIFICATION SIGNALISATION PAR LED

Il est demandé d'introduire les modifications suivantes :

- Toggle LED_3 dans interruption RX
- Toggle LED_4 dans interruption TX
- LED_0 à 1 au début traitement application puis à 0 en fin traitement application (éliminer les autres traitements sur LED_0).

MODIFICATIONS INTERRUPTION RX

Voici la solution pour la parie RX de l'interruption de l'USART :

```
void __ISR(_UART_1_VECTOR, ipl5AUTO) _IntHandlerDrvUsartInstance0(void)
{
    uint8_t ErrFiFoFull = 0;
    uint8_t freeSize, TXsize;
    int8_t c;
    int8_t i_cts = 0;
    BOOL  TxBuffFull;

    USART_ERROR  UsartStatus;

    // Toggle LED3 pour indiquer activité
    //BSP_LEDToggle(BSP_LED_3);

    // Is this an RX interrupt ?
    if ( PLIB_INT_SourceFlagGet(INT_ID_0, INT_SOURCE_USART_1_RECEIVE) &&
                 PLIB_INT_SourceIsEnabled(INT_ID_0, INT_SOURCE_USART_1_RECEIVE) ) {

        BSP_LEDToggle(BSP_LED_3); // debug : LED3 pour indiquer Rx

        // Oui Test si erreur parité ou overrun
        UsartStatus = PLIB_USART_ErrorsGet(USART_ID_1);

        if ( (UsartStatus & (USART_ERROR_PARITY |
                             USART_ERROR_FRAMING | USART_ERROR_RECEIVER_OVERRUN)) ==
                                  0) {

            // transfert dans le fifo de tous les chars reçus
            while (PLIB_USART_ReceiverDataIsAvailable(USART_ID_1) &&
              GetWriteSpace(&descrFifoRX) >= 1)
            {
                c = PLIB_USART_ReceiverByteReceive(USART_ID_1);
                PutCharInFifo ( &descrFifoRX, c);
            }

             // buffer is empty, clear interrupt flag
             PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_USART_1_RECEIVE);
        } else {
             // Suppression des erreurs
             // La lecture des erreurs les efface sauf pour overrun
             if ( (UsartStatus & USART_ERROR_RECEIVER_OVERRUN) ==
                    USART_ERROR_RECEIVER_OVERRUN) {
                   PLIB_USART_ReceiverOverrunErrorClear(USART_ID_1);
             }
        }

        // Si tampon non vide (on n'a pas pu le vider) -> disable INT RX
        if (PLIB_USART_ReceiverDataIsAvailable(USART_ID_1)
                                                == true) {
            PLIB_INT_SourceDisable(INT_ID_0,
                                   INT_SOURCE_USART_1_RECEIVE);
        }
    } // end if RX
```

Pour forcer la gestion automatique de RTS, il faut arrêter de vider le tampon de réception lorsque le FIFO software est presque plein. Ceci a pour conséquence que l'on peut quitter la réponse à l'interruption avec un tampon non vide, ce qui va provoquer une répétition de l'interruption. C'est pour cela que si le tampon contient encore des caractères, il est nécessaire de "disable" l'interruption de réception.

MODIFICATIONS GETMESSAGE

La section gestion du contrôle de flux n'est pas supprimée. On y gère l'autorisation de l'interruption de réception lorsqu'il y a une certaine place dans le FIFO.

```
    // Gestion interruption réception
    if((GetWriteSpace(&descrFifoRX) >= MESS_SIZE) &&
       PLIB_USART_ReceiverDataIsAvailable(USART_ID_1)) {
        // autorise émission par l'autre
        //RS232_RTS = 0;
        // Autorise interruption de réception
        PLIB_INT_SourceEnable(INT_ID_0, INT_SOURCE_USART_1_RECEIVE);

    }
    return CommStatus;
} // GetMessage
```

La valeur de MESS_SIZE est discutable.

MODIFICATIONS INTERRUPTION TX

Dans le traitement de l'interruption de transmission, il faut supprimer la condition testant RS232_CTS. On ne vérifie plus que 2 conditions.

```
// Is this an TX interrupt ?
if ( PLIB_INT_SourceFlagGet(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT) &&
             PLIB_INT_SourceIsEnabled(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT) ) {

    BSP_LEDToggle(BSP_LED_4); // debug : LED4 pour indiquer Tx

     TXsize = GetReadSize (&descrFifoTX);
     // i_cts = input(RS232_CTS);
     // On vérifie 3 conditions :
     //    Si CTS = 0 (autorisation d'émettre)
     //    Si il y a un caratères à émettre
     //    Si le txreg est bien disponible

     //i_cts = RS232_CTS;

     // TxPossible = UARTTransmitterIsReady(UART1);
     TxBuffFull = PLIB_USART_TransmitterBufferIsFull(USART_ID_1);
     if ( /*(i_cts == 0) && */( TXsize > 0 ) &&  TxBuffFull==false )  {

         do {
             GetCharFromFifo(&descrFifoTX, &c);
             PLIB_USART_TransmitterByteSend(USART_ID_1, c);
             i_cts = RS232_CTS;
             TXsize = GetReadSize (&descrFifoTX);
             TxBuffFull = PLIB_USART_TransmitterBufferIsFull(USART_ID_1);

         } while ( /*(i_cts == 0) &&*/ ( TXsize > 0 ) &&
                                          TxBuffFull==false  );

        // Clear the TX interrupt Flag
        // (Seulement après TX)
        PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT);

        if (TXsize == 0) {
            // disable TX interrupt
            PLIB_INT_SourceDisable(INT_ID_0,
                                   INT_SOURCE_USART_1_TRANSMIT);
        }


    } else {
       // disable TX interrupt
       PLIB_INT_SourceDisable(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT);
    }
    //BSP_LEDOff(BSP_LED_4); // debug : LED4 pour indiquer Tx
} // end if TX
```

L'interruption de transmission n'est pas modifiée dans son principe, la gestion de CTS a été supprimée.

MODIFICATIONS SENDMESSAGE

Dans la section gestion du contrôle de flux, il n'y a plus que la condition FreeSize > 0 à utiliser pour autoriser l'interruption d'émission.

```
    // Gestion du controle de flux
    // si on a un caractère à envoyer et que CTS = 0
    FreeSize = GetReadSize(&descrFifoTX);
    if (/*(RS232_CTS == 0) &&*/ (FreeSize > 0))
    {
        // Autorise int émission
        PLIB_INT_SourceEnable(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT);
    }
} // SendMessage
```

MODIFICATIONS INITFIFOCOMM

Il ne faut plus établir RTS à 1.

MODIFICATION SIGNALISATION PAR LED

Il est demandé d'introduire les modifications suivantes :

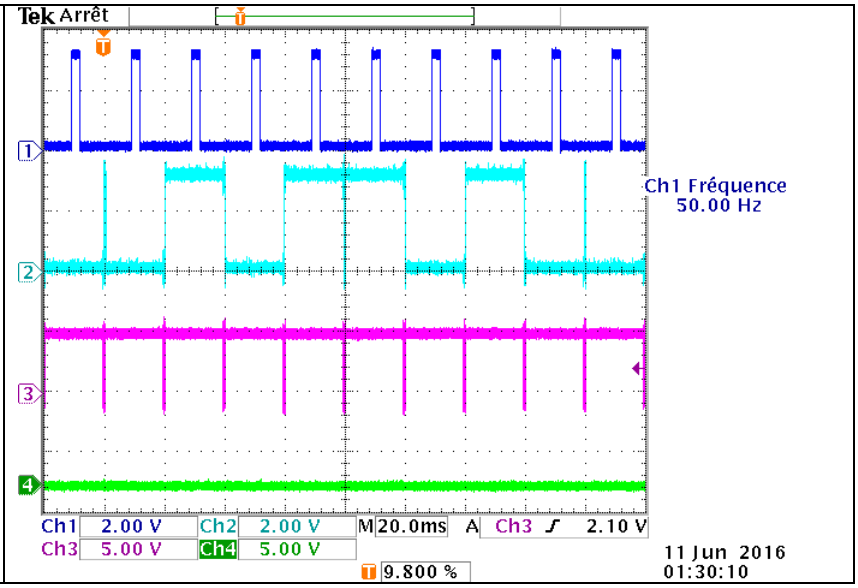
- Toggle LED_3 dans interruption RX
- Toggle LED_4 dans interruption TX
- LED_0 à 1 au début traitement application puis à 0 en fin traitement application (éliminer les autres traitements sur LED_0).

TEST DE FONCTIONNEMENT ET OBSERVATIONS

b) TEST INITIAL

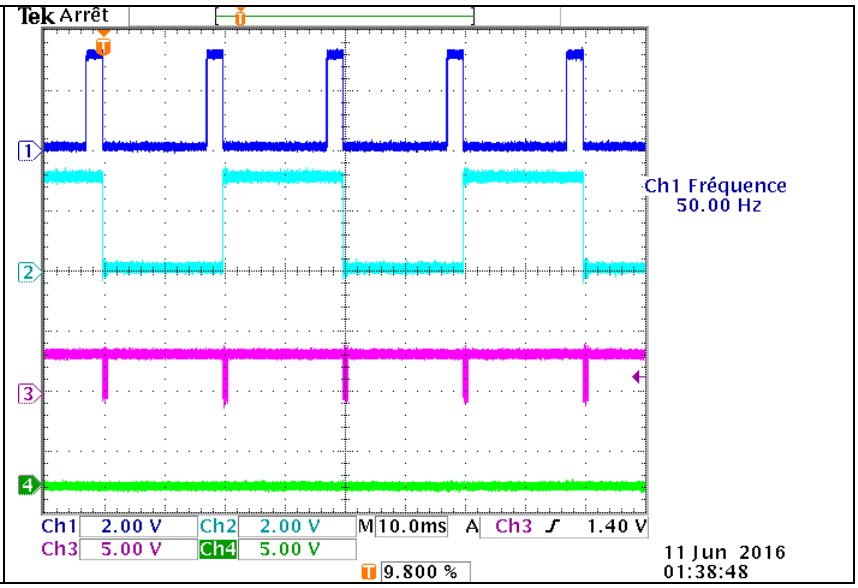
Avant de créer une surcharge mettant en œuvre le handshaking, il faut tester le bon fonctionnement des modifications. Utilisation de 2 kits chargés chacun avec une application qui émet un message par cycle.

OBSERVATION EN RECEPTION

canal 1: LED_0 (marque traitement dans l'application) canal 2: LED_3 (inversion à chaque int RX) canal 3 : broche 52 U1RX canal 4: RS232_RTS (broche 48)	

Remarque : De par la situation (taille message = 5 et interruption de réception réglée sur half full, donc 4 caractères), on obtient une interruption ou 2 pour chaque message.

OBSERVATION EN EMISSION

canal 1: LED_0 (marque traitement dans l'application) canal 2: LED_4 (inversion dans réponse à l'interruption TX) canal 3 : broche 53 U1TX canal 4: RS232_CTS (broche 47)	

Remarque : En émission et réception, on observe bien qu'il n'y a pas de surcharge (signaux RTS et CTS stables à '0').

c) TEST AVEC SURCHARGE

Pour tester le comportement du handshaking, il est nécessaire d'augmenter le débit d'émission. Il faut modifier l'application pour envoyer 1 message supplémentaire tous les 3 cycles de l'application.

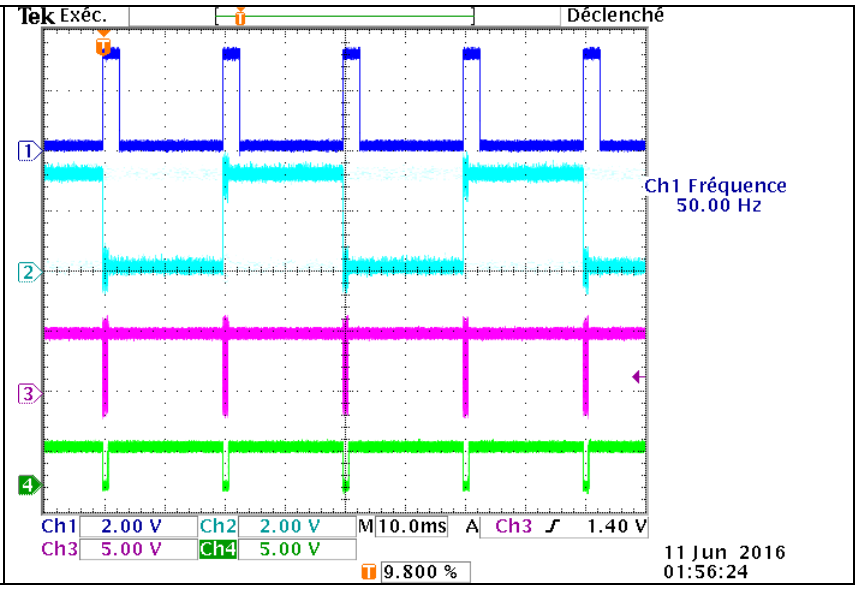
OBSERVATION EN RECEPTION (AVEC SURCHARGE)

canal 1: LED_0 (marque traitement dans l'application)

canal 2: LED_3 (inversion à chaque int RX)

canal 3 : broche 52 U1RX

canal 4: RS232_RTS (broche 48)



On constate que RTS est presque constamment au niveau haut et que le débit des messages est limité. Dès que la carte détecte une place dans son buffer, elle baisse sa sortie RTS et les données arrivent immédiatement du correspondant.

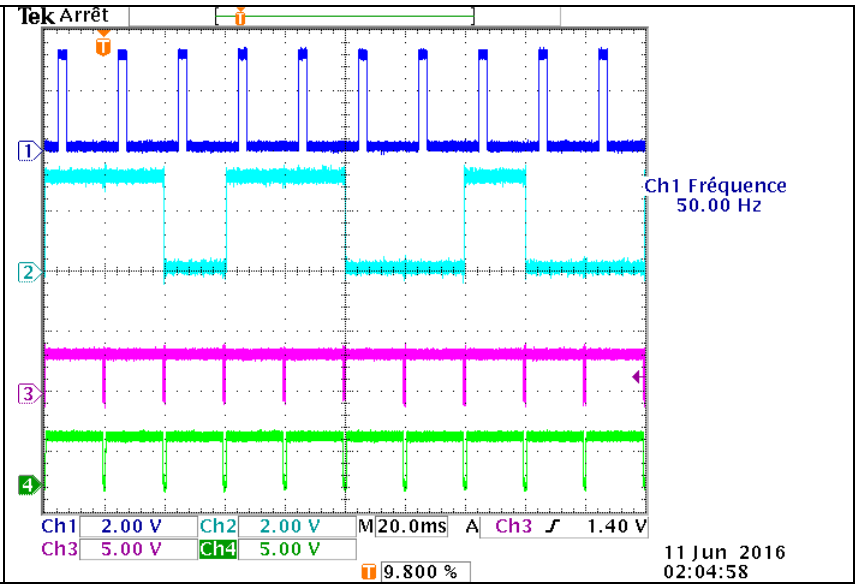
OBSERVATION EN EMISSION (AVEC SURCHARGE)

canal 1: LED_0 (marque traitement dans l'application)

canal 2: LED_4 (inversion dans réponse à l'interruption TX)

canal 3 : broche 53 U1TX

canal 4: RS232_CTS (broche 47)



Il y a transmission dès que possible (toutes les 20 ms), c'est à dire dès que le correspondant vide son buffer de réception et libère de la place. Par contre, il n'y a pas forcément d'interruption TX au même rythme (le FIFO TX étant déjà plein, l'interruption n'est pas réactivée - voir code).

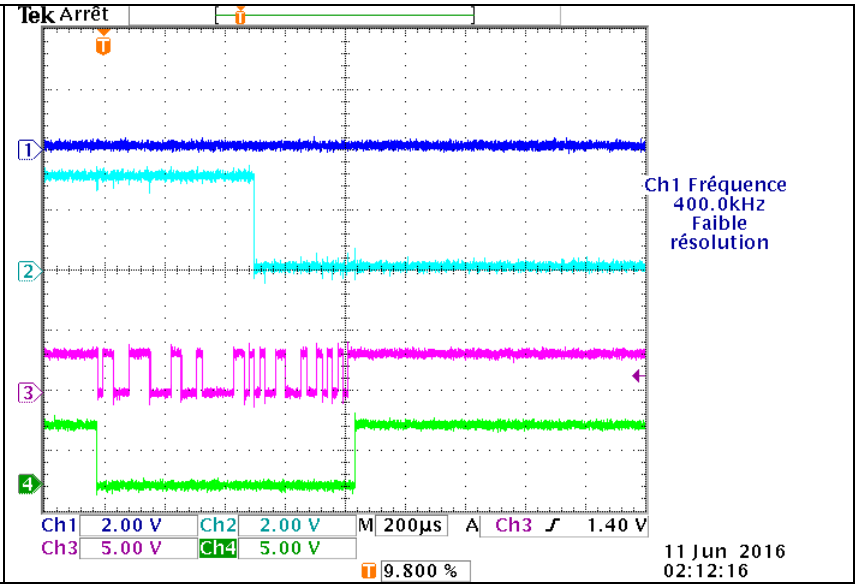
Observation détails

canal 1: LED_0 (marque traitement dans l'application)

canal 2: LED_4 (inversion dans réponse à l'interruption TX)

canal 3 : broche 53 U1TX

canal 4: RS232_CTS (broche 47)



On constate que l'émission démarre dès que CTS est bas. Après quelques caractères, le tampon est vide, ce qui provoque une interruption. Dès que CTS est haut, l'émission cesse immédiatement.

CONCLUSION

La gestion automatique de RTS et CTS par l'UART fonctionne. Elle peut s'avérer nécessaire dès que l'émission est trop rapide par rapport à la capacité de traitement du récepteur, afin de ne pas perdre de données.

Cependant avec l'utilisation d'un FIFO software et d'interruptions de l'usart, cela rend la gestion plus délicate.