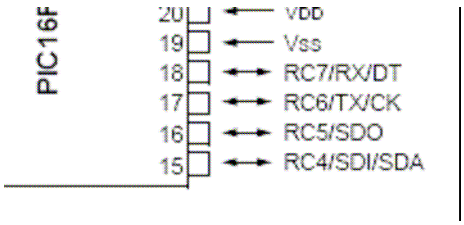
Mise en œuvre d'une liaison série

Le PIC16F877A intègre trois ports de communications :
- un port synchrone série « SSP » avec les protocoles « SPI » et « I2C » ;
- un port de communication parallèle « PSP » 8 bits ;
- un port asynchrone série universel « USART ».

C'est à ce dernier que nous allons nous intéresser.

Principe de fonctionnement

Le module « USART » (Universal Synchronous Asynchronous Receiver Transmetter) utilise les entrées sorties « **RC6/TX** » et « **RC7/RX** » pour communiquer ;



Le registre « **TXSTA** » permet de configurer la transmission :

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC**: Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
Note: SREN/CREN overrides TXEN in Sync mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be Parity bit

tandis que « **RCSTA** » configure la réception :

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

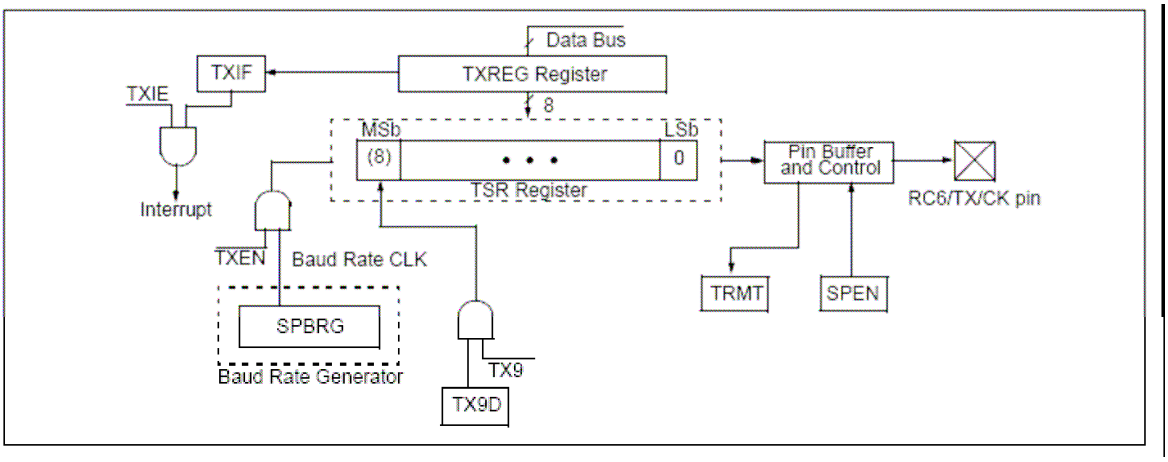
bit 7	**SPEN:** Serial Port Enable bit 1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins) 0 = Serial port disabled
bit 6	**RX9**: 9-bit Receive Enable bit 1 = Selects 9-bit reception 0 = Selects 8-bit reception
bit 5	**SREN**: Single Receive Enable bit Asynchronous mode: Don't care. Synchronous mode – Master: 1 = Enables single receive 0 = Disables single receive This bit is cleared after reception is complete. Synchronous mode – Slave: Don't care.
bit 4	**CREN**: Continuous Receive Enable bit Asynchronous mode: 1 = Enables continuous receive 0 = Disables continuous receive Synchronous mode: 1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN) 0 = Disables continuous receive
bit 3	**ADDEN:** Address Detect Enable bit Asynchronous mode 9-bit (RX9 = 1): 1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set 0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
bit 2	**FERR**: Framing Error bit 1 = Framing error (can be updated by reading RCREG register and receive next valid byte) 0 = No framing error
bit 1	**OERR**: Overrun Error bit 1 = Overrun error (can be cleared by clearing bit CREN) 0 = No overrun error
bit 0	**RX9D:** 9th bit of Received Data (can be parity bit but must be calculated by user firmware)

L'USART peut fonctionner en liaison asynchrone duplex-intégral ou liaison synchrone semi-duplex maître ou esclave (l'horloge est fournie ou reçue) suivant l'état des bits « **SYNC** » (synchrone ou asynchrone) et « **CSRC** » (maître ou esclave – ne concerne que le synchrone) de « **TXSTA** ».

La transmission

Un timer (autre que les trois timers utilisables du microcontrôleur) joue le rôle de générateur de bauds afin de configurer la vitesse de l'horloge de transmission ; cette vitesse est configurée par le registre « **SPBGR** » et le bit « **BGRH** » de « **TXSTA** ».

En mode asynchrone, les données sont transmises et émises en NRZ, bit start, mot de 8 ou 9 bits (bits « **RX9** » et « **TX9** » de « **RCSTA** » et « **TXSTA** »), LSB en tête, un bit stop. La parité n'est pas supportée matériellement (mais peut être implantée par logiciel). Les formats et vitesses sont identiques en émission et réception.



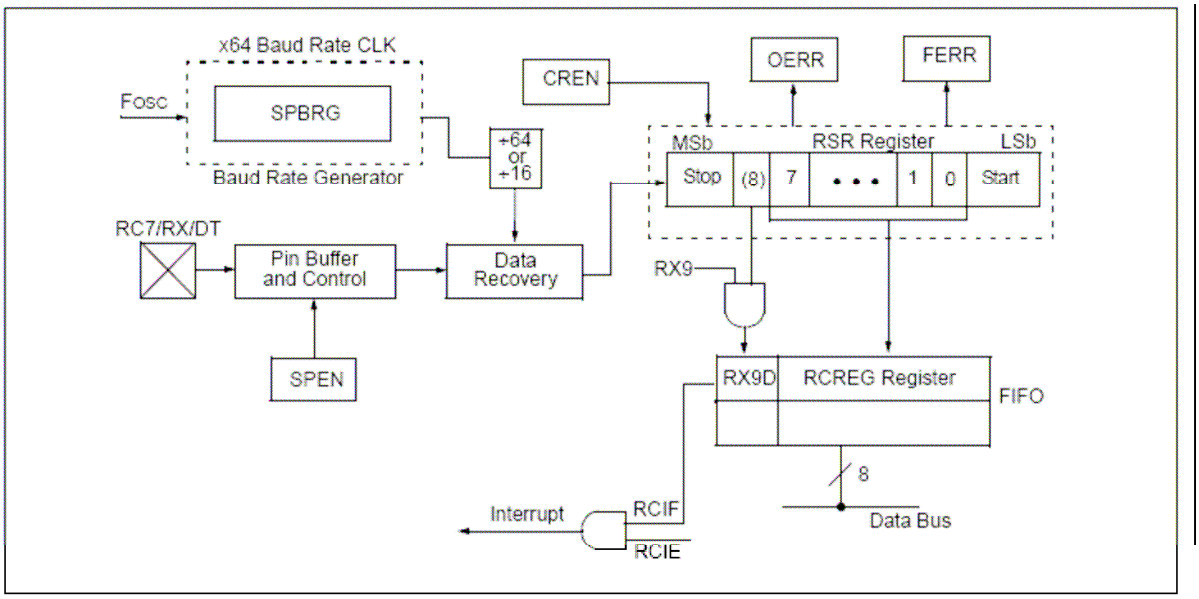
La transmission asynchrone se déroule comme suit :

- le mot à envoyer est placé dans un registre 8 bits « **TXREG** » ;
- ce mot sera ensuite chargé dans un registre à décalage « **TSR** » (non accessible) dès que le bit stop du mot précédent sera envoyé ; si le format d'envoi est de 9 bits, le complément est réalisé par « **TX9D** »de « **TXSTA** » (à placer avant le chargement de « **TXREG** ») ;
- lorsque le transfert de « **TXREG** » vers « **TSR** » a eu lieu, le drapeau « **TXIF** » du registre « **PIR1** » passe à 1, générant une interruption si « **TXIE** » du registre « **PIE**1 » est à 1. Le drapeau repassera seul à 0 lorsque « **TXREG** » sera de nouveau plein.
- A la fin de l'envoi du mot, lorsque « **TSR** » est vide, le bit « **TMRT** » du registre « **TXSTA** » passe à 1 ; ce bit ne génère pas d'interruption, il est accessible uniquement par scrutation.

La réception

La réception se déroule comme suit :

- les données sont reçues dans un registre à décalage « **RSR** » ;
- puis transférées dans le registre « **RCREG** » qui positionne à 1 le drapeau « **RC1F** » du registre « **PIR1** », ce qui génère une interruption si le bit d'autorisation « **RC1E** » du registre « **PIE1** » est mis à 1.



Les erreurs de réception :

- Le registre « **RCREG** » est une pile FIFO à 2 niveaux. Lorsque la pile est pleine et que le bit stop du troisième mot dans « **RSR** » vient d'être détecté, le bit « **OERR** » du registre « **RCSTA** » passe à 1, signalant une erreur de type dépassement. Le dernier mot dans « **RSR** » est alors perdu.
- Le bit « **OERR** » doit être remis à 0 en plaçant le bit « **CREN** » de « **RCSTA** » à 0 puis à 1.
- Le bit « **FERR** » de « **RCSTA** » passe à 1 lorsqu'une erreur de format est détectée (bit stop à 0).

Configuration de broches

Le bit « **SPEN** » de « **RCSTA** », ainsi que les bits 6 et 7 de « **TRISC** » doivent être positionnés à 1 (entrée) pour pouvoir utiliser RC6 et RC7.

Programmation

On se propose d'envoyer un message depuis notre carte PICDEM 2 PLUS vers l'ordinateur PC. Pour cela nous allons utiliser, d'une part le connecteur RS232 de la carte, un port série du PC et l'hyperterminal de Windows XP (programme intégré au système d'exploitation, permettant de recevoir et de transmettre des données depuis ou vers un port série).

La liaison se fera à 9600 Bd avec des mots de 8 bits, 1 bit stop, pas de bit de parité.

Configuration du PIC

Pour obtenir la vitesse de transmission souhaitée, il existe plusieurs configuration possible du générateur de bauds par « **SPBGR** » et le bit « **BGRH** » de « **TXSTA** ».
Les tableaux 10.3 et 10.4 du « datasheet » permettent de configurer le timer pour obtenir le taux d'erreur le plus faible.
Déterminer les octets à placer dans « **TXSTA** », « **RCSTA** » et « **SPBGR** ».

Ouvrir un nouveau projet TP5_usart pour PIC16F877A, y inclure le fichier « Prog1.asm » du répertoire « Ressources \ TP5_usart », sauvegarder le fichier dans votre répertoire sous un nom différent, supprimer la version d'origine du projet.

Analyser le programme (reproduit en annexe), noter en particulier :
- l'utilisation de la directive d'assemblage « **CBLOCK** » qui définit les adresses d'une zone de variables en RAM (même si dans notre cas, il n'y a qu'une seule variable) ;
- la configuration de l'USART, vérifier que les octets correspondent à ceux de votre prédétermination ;
- la manière dont est adressée la table de caractères (revoir éventuellement « Structure d'un programme en assembleur)
- la manière de repérer la fin de la table.

Compiler et simuler le programme.

Configuration du PC

Pour exécuter l'hyperterminal : « démarrer accessoires communications hyperterminal »
- sous XP pro, s'il n'est pas installé : « Démarrer » « Tous les programmes » « Accessoires » « Communications » « Hyperterminal ».

Gestion par interruption

Proposer une modification du programme, avec :
- interruption du timer pour indiquer l'envoi du nouveau caractère ;
- interruption de l'USART signalant que « **TXREG** » est vide.

Comme nous avons deux sources d'interruptions possibles, il sera nécessaire en scrutant les bits drapeaux, de savoir qui a généré cette interruption pour réaliser le traitement adapté.
D'autre part, ces interruptions pouvant arriver n'importe où dans le programme, le contexte devra être sauvegardé, en particulier les registres « **W** » et « **STATUS** ».

Annexe 1 : programme de communication

```
;*******************************************************************************:
; Ce programme envoi un message à 9600 Bd via l'USART

;*******************************************************************************

        LIST P=16F877A        F=INHX32      ; directive qui définit le processeur utilisé
                                            ; et le format des données envoyées
        #include <P16F877A.INC>             ; fichier de définition des constantes

;*******************************************************************************
;                   BITS DE CONFIGURATION
;*******************************************************************************
        __CONFIG  _HS_OSC & _WDT_OFF & _CP_OFF & _CPD_OFF & _LVP_OFF

;*******************************************************************************
;                   DECLARATION de VARIABLES                    *
;*******************************************************************************
        CBLOCK      h'20'           ; début de la zone en 20h en RAM
        INDEX       :1              ;INDEX occupe un octet à l'adresse 20h
        ENDC                        ; Fin de la zone

                                    ; on aurait pu écrire aussi INDEX EQU Ox2O
;*******************************************************************************
;                   DEMARRAGE SUR RESET                                     *
;*******************************************************************************
        org     0x0                 ; Adresse de départ après reset
        goto    debut

        org     0x10                ; adresse de début du programme
debut

;*******************************************************************************
;                   PROGRAMME PRINCIPAL                         *
;*******************************************************************************
        call init
caract0
        BANKSEL     INDEX
        clrf    INDEX           ; initialisation de l'index qui pointe le premier caractère
boucle
        movf    INDEX,W         ; W pointe le caractère à afficher
        call    MESSAGE         ; recherche du caractère dans la table
        andlw   0xFF            ; test de fin de message (le dernier caractère est 0)
        btfsc   STATUS,Z
        goto    caract0         ; si c'est le dernier caractère on recommence au premier
        call    emission        ; sous programme d'émission
        call    tempo           ; attente
        incf    INDEX ,F        ; incrémentation du caractère à afficher
        goto    boucle          ; rebouclage
;*******************************************************************************
;                   SOUS PROGRAMMES                             *
;*******************************************************************************
;*******************************************************************************
;                   INITIALISATION                              *
```

```
;*******************************************************************************
init
                        ; initialisation du PORTC en entrée
        BANKSEl     PORTC       ; passage en banque 0
        clrf        PORTC       ; RAZ des bascules D du port B
        BANKSEL     TRISC       ; passage en banque 1
        movlw  0xFF
        movwf  TRISC            ; PORTC en entrée pour utiliser l'usart

                        ; initialisation de l'USART
        BANKSEL SPBRG           ; passage en banque 1
        movlw  d'25'            ; avec BRGH=1, il faut 25 dans SPBRG pour 9,6 kBd
        movwf  SPBRG
        movlw  h'24'            ;
        movwf  TXSTA            ; 8 bits, autorisé, asynchrone, haute vitesse
        BANKSEL     RCSTA       ; passage en banque 0
        movlw  h'80'
        movwf  RCSTA            ; port série

                        ; initialisation du Timer 1 :division par 8 de l'horloge interne
        BANKSEL T1CON       ; passage en banque 0
        movlw  b'00110001'  ; T1CGPS1=1 T1CGPS0=1 T1OSCEN=0 TMR1CS=0 TMR1ON=1
        movwf  T1CON        ; predivision TMR1 timer 1:8, hor ext off, hor int
                            ; timer on
        return
;*******************************************************************************
;                   TEMPORISATION 100ms                                       *
;*******************************************************************************
                        ; fréquence du quartz : 4 MHz, temps de cycle 1 µs , prédivision : 8
                        ; 2^E16 - 100ms/8µs= 53036d = CF2Ch
tempo
        movlw  0x2C
        movwf  TMR1L            ; initialisation de TMR1L
        movlw  0xCF
        movwf  TMR1H            ; initialisation de TMR1H
boucle_tmr
        btfss   PIR1,TMR1IF         ; Test bit TMR1F de PIR1
        goto    boucle_tmr          ; boucle d'attente si timer non nul
        bcf             PIR1,TMR1IF ; RAZ du drapeau
        return
;*******************************************************************************
;                   EMISSION                                                  *
;*******************************************************************************
emission
        movwf  TXREG            ; le caractère dans W est prêt à être émis
boucle_emission
        btfss   PIR1,TXIF           ; attente de l'envoi du dernier bit
        goto    boucle_emission
        return
;*******************************************************************************
;                   TABLE                                                     *
;*******************************************************************    *******
MESSAGE
        ADDWF       PCL,F
        DT          "Salam Aleikum"     ; directive de définition d'une table de caractère
        RETLW       h'0D'               ; Retour chariot
        RETLW       h'00'               ; indicateur de fin de message
;*******************************************************************************
END
```