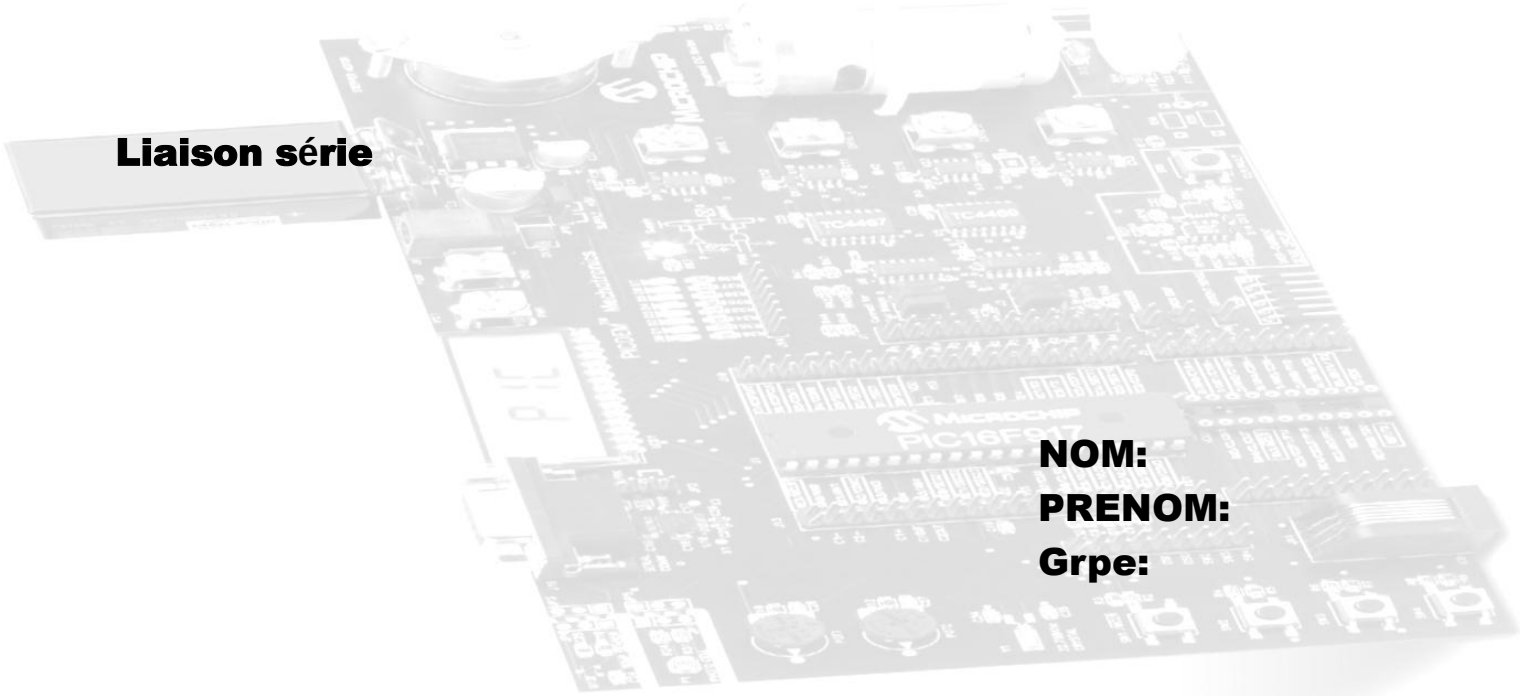
Cours de PIC



Liaison série

NOM:
PRENOM:
Grpe:

Principe de fonctionnement

Définition

- A la différence des liaisons parallèles la transmission série consiste à transmettre des informations binaires bit par bit sur un fil électrique.
- La transmission se fait :
  - ✓ Soit en **synchronisme** avec une **horloge de référence** commune au 2 systèmes et transmise sur une ligne supplémentaire :
    - Exemple : liaison SSP du PIC (Synchronous Serial Port).
  - ✓ Soit de façon **indépendante sans horloge de référence** : dans ce cas la vitesse de transmission doit être identique sur une même ligne qui relie les circuits d'émission et de réception. Par contre elle n'est pas forcément la même sur les 2 lignes :
    - Exemple liaison USART du PIC (Asynchronous Synchronous Receiver Transmitter)

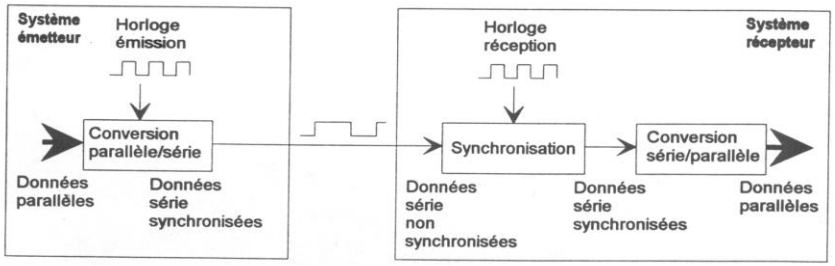
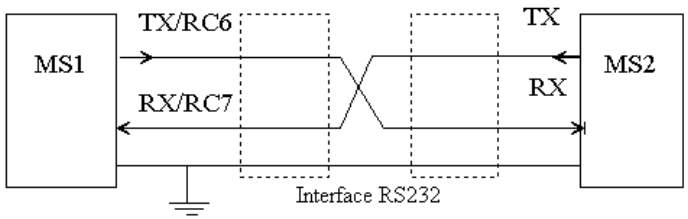


Figure 3.1. *Synchronisation en réception*

La transmission asynchrone

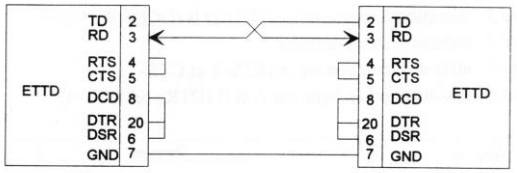
- Les données sont transmises octet par octet, la synchronisation entre les 2 micro-systèmes indépendants MS1 et MS2 se fait grâce au bit de start et au(x) bit(s) de stop qui sont rajoutés aux bits de données
- Au repos la ligne est au 5V, le bit de start 0V indique qu'une information (octet) va suivre, le ou les bits de stop à 5V indiquent que le transfert de l'octet est terminé
- Câblage minimum (hors norme – voir page suivante)



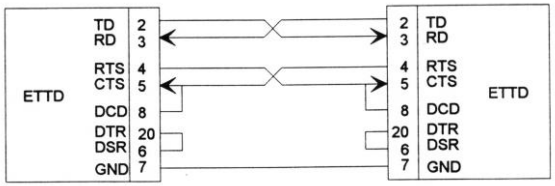
- La ligne d'émission série : TX sur RC6.
- La ligne de réception série : RX sur RC7
- La ligne de masse.

Principe de fonctionnement

- **norme RS 232 ou V24**
  - Définit les aspects fonctionnels, mécaniques et électriques d'une liaison série RS232
    - ✓ Cablâges normés

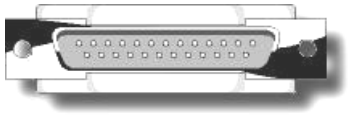


Cablâge Null-modem



Avec flux matériel (signaux de contrôles RTS,CTS)

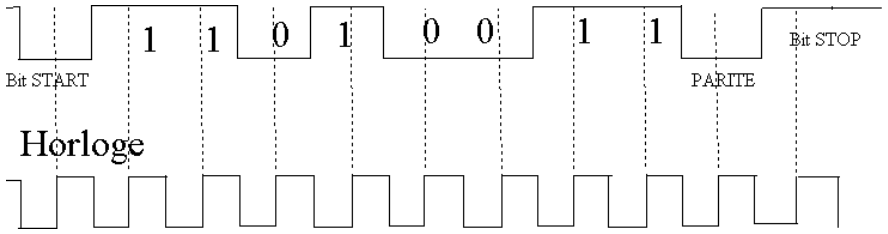
    - ✓ connecteurs



DB25

DB9

    - ✓ Constitution d'une trame

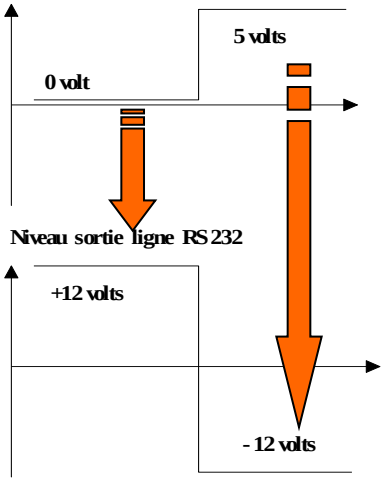


✓1 bit de start.
✓7, 8 ou 9 bits d'information.
✓1 ou pas de bit de parité (paire ou impaire)

✓ ou 2 bits de stop.

Principe de fonctionnement

- ✓ Signaux électriques

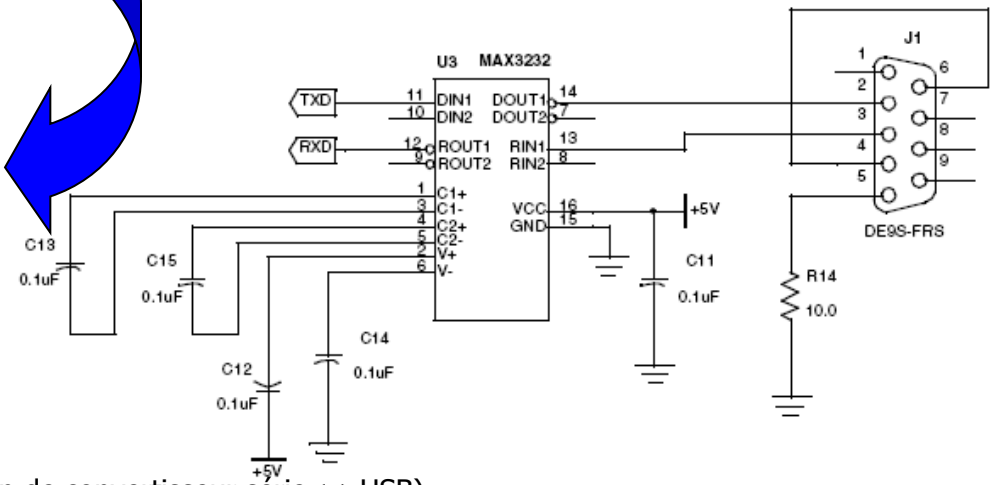


En sortie du PIC

Définition de la norme V24

Utilisation d'un coupleur pour réaliser l'interface entre PIC et RS232

Un composant: le *MAX232 et ses variantes*



- **Côté PC**
  - Tend à disparaître au profit de l'USB (apparition de convertisseur série<->USB)
  - Connectique DB9 en majorité
  - UART intégré dans la carte mère
    - ✓ Ancêtre: 8250 PC XT -> nouvelles générations: 16750 chez Texas Instrument
  - Adresses de configuration des ports sériels (accessible via le BIOS)
    - ✓ COM1: 3F8h COM2: 2F8h
  - Un utilitaire intégré à Windows
    - ✓ HYPER TERMINAL :à utiliser en TP pour faire communiquer la carte PIC et le PC

Programmation

Programmer la liaison du PIC

- Les fonctions disponibles

```
OpenUSART( USART_TX_INT_OFF &
USART_RX_INT_ON &
USART_ASYNCH_MODE &
USART_EIGHT_BIT &
USART_CONT_RX &
USART_BRGH_HIGH,
25 );
```

+ d'infos dans la doc MPLAB-C18-Libraries.pdf

Function	Description
BusyUSART	Is the USART transmitting?
CloseUSART	Disable the USART.
DataRdyUSART	Is data available in the USART read buffer?
getcUSART	Read a byte from the USART.
getsUSART	Read a string from the USART.
OpenUSART	Configure the USART.
putcUSART	Write a byte to the USART.
putsUSART	Write a string from data memory to the USART.
putrsUSART	Write a string from program memory to the USART.
ReadUSART	Read a byte from the USART.
WriteUSART	Write a byte to the USART.
baudUSART	Set the baud rate configuration bits for enhanced USART.

- L'initialisation
  - ✓ OpenUSART
  - ✓ Nécessite le calcul de la vitesse
    - Vitesse : deux possibilités :
      - High Speed Vitesse en bit/s = Fosc / (16 spbrg + 1)
      - Low Speed Vitesse en bit/s = Fosc / (64 spbrg + 1)

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

- Emettre un caractère

```
c = 'A' ;
WriteUSART(c);
```

- Envoyer une chaîne de caractères

```
stdout = _H_USART ;
printf("Coucou !\r");
```

Programmation

- Réception
  - ✓ Privilégier une utilisation avec interruption
    - Principe

```
if(DataRdyUSART() )
  {
   c = ReadUSART() ;
}
```

Compléments sur les chaînes

- En C une chaîne de caractères est une suite de char dont la fin est marquée par le caractère de valeur 0x00 (noté'\0')
- Déclaration en zone variable
  - ✓ char Ch[4]=« IUT!" ;
- Déclaration en zone programme
  - ✓ const rom Ch[4]=" IUT!";
- Fonction de manipulation de chaîne
  - ✓ Spécifique au PIC => NON ANSI
  - ✓ **Ex: strcpypgm2ram** : de mémoire programme à data

```
if(r){// variable détection chaine reçu complète
   if(strcmppgm2ram(msg,"ON")==0 ) {

                  GreenLed = 1 ; }
   else {
      if(strcmppgm2ram(msg,"OFF")==0 ) {

                  GreenLed = 0 ; }
   }
```

```
// High priority interrupt routine
#pragma code
#pragma interrupt InterruptHandlerHigh
void
InterruptHandlerHigh ()
{
  static char i ; // doit être statique pour conserver sa valeur entre les IT
  char c ;
 // Partie réception d'un caractère
 if(PIR1bits.RCIF) // si un car arrivé
  {
   c=ReadUSART(); // le lire
   if(c!=0x0D)
     {
      msg[i++]=c ; // stockage
     }
   else
     {
      msg[i]='\0'; // fin de chaîne si CR
      i=0;
      r=1;
     }
   }
  }
 // Placer ici les autres parties
 // if(Autre bit F)
  // {
  //   Raz du bit F ;
  //   Traitement ;
  // }
}
```

Le bit RCIF est remis tout seul à 0 lors d'une lecture du registre de réception RCREG.

Programmation

- **Alternative de programmation**
  - Utilisation des bits associés à la liaison série
  - Voir autres périphériques
  - Signification des bits dans la doc du PIC

Exemple en émission

9600,8,N,1

```
#define S2 PORTAbits.RA4

char f=0;

void init(void);

void main()
{
                    init();
                    for(;;)
                    {
                     if(S2==0){
                         if(f==0){
                             f=1;

                    while(TXSTAbits.TRMT==0);

                             TXREG='A';

                                  }
                                }
                    else
                       {f=0;}
                   }
}

void init(void)
{                   TXSTAbits.SYNC=0;
                    TXSTAbits.BRGH=1;
                    BAUDCONbits.BRG16=0;
                    SPBRG=25;
                    TXSTAbits.TXEN=1;
                    RCSTAbits.SPEN=1;
                    TXSTAbits.TX9=0;
                    TRISC.TRISC6=0;// TX en sortie
                    TRISC.TRISC7=1;// RX en entrée
```

Exemple en réception

9600,8,N,1

```
// Directives de compilation

#include <p18cxxx.h>
#include <stdio.h>
#include "xlcd100.h"

// configuration
#pragma config OSC = HS   //Oscillateur externe
#pragma config WDT = OFF  //Watch Dog inhibé.
#pragma config LVP = OFF  //Low Voltage ICSP dévalidé.
#pragma config PBADEN = OFF //Si PBADEN=ON RB4:RB0 analogiques au reset !


Char c;

void init(void);

void main()
{
                    init();
                    OpenXLCD(OPEN_PICDEM_LCD);
                    stdout = _H_USER;
                    for(;;)
                    {
                                        if(PIR1bits.RCIF==1)
                                        {
                                                            gotoXLCD(LCD_LINE_ONE);
                                                            c=RCREG;
                                                            printf("%c",c);
                                        }

                    }
}

void init(void)
{
                    TXSTAbits.SYNC=0;
                    TXSTAbits.BRGH=1;
                    BAUDCONbits.BRG16=0;
                    SPBRG=25;
                    RCSTAbits.SPEN=1;
                    RCSTAbits.CREN=1;
                    RCSTAbits.RX9=0;
}
```