Utilisation de la liaison série pour la transmission de données

Pierre DONNAT
Lycée Janson de Sailly, 75016 Paris

Dans l'univers des micro-ordinateurs et de la mesure numérique, deux modes de transmission des données sont utilisés :

• la **liaison parallèle** est exploitée pour des liaisons courtes (moins de 20 m) à fort débit d'information, par exemple pour les imprimantes mais aussi par de nombreux systèmes automatisés de mesure (conçus le plus souvent sous la norme IEEE 488.75 dite «General Purpose Interface Bus» GPIB ou HPIB) ;

• la **liaison série** est exploitée pour des liaisons longues (moins de 1 km) à débit d'information plus lent, par exemple pour une souris, un Modem, etc... La norme la plus courante est RS232 (Recommended Standart), mais d'autres existent qui permettent des débits plus grands.

Je me propose de vous présenter le principe d'une transmission par liaison série (RS232), en mettant en œuvre des mini-programmes permettant de démystifier la **communication entre un micro-ordinateur PC et un oscilloscope à mémoire numérique**.

1. OSCILLOSCOPE À MÉMOIRE NUMÉRIQUE

Ces oscilloscopes permettent la visualisation, sur leur écran, du signal numérisé ainsi que des paramètres utilisés lors de la saisie en mémoire (voie du signal, calibre de la base de temps, calibre de la tension, etc...), mais aussi la transmission à un micro-ordinateur de ces informations . Un programme qui effectue le transfert de ces informations vers le PC comporte donc une unité de pilotage de la liaison PC-oscilloscope.

Quelle information est numérisée ?

Une information visible ou qui sera visualisée : le signal, les calibres, peut-être une position de curseur. Mais aussi une information moins visible, qui sera rarement visualisée :

– le type de l'oscilloscope, sa version ;

– quelques mots d'un mini-langage que l'oscilloscope reconnaîtra si le PC les lui adresse, l'oscilloscope répondant dans son langage aux ordres reçus et compris.

Quel codage est employé ?

La plus petite information mémorisable dans un ordinateur est le bit. Un bit est positionné soit à 1, soit à 0, le représentant physique étant une tension appliquée, avec 2 valeurs ou 2 états possibles. Le groupement de 8 bits définit un octet. Un octet permet de représenter 256 valeurs ($256 = 2^8$), caractères ou nombres ; sa signification dépend de l'usage que l'on veut en faire.

Dans un souci de normalisation, les 128 premières «valeurs» d'octet sont nommées par le code ASCII ; on y trouve les chiffres, les lettres alphabétiques minuscules et majuscules, les signes de ponctuation, etc... On appellera, dans ce qui suit, «caractère» un octet représenté par son codage ASCII. Arrêtons-nous de suite sur cette représentation : quelle signification donner, par exemple, aux deux caractères successifs CE ?

– d'abord celui de notre langage, français, le pronom démonstratif ;

– mais pourquoi pas aussi celui du nombre 206 : avec les chiffres 0123456789ABCDEF pour une représentation hexadécimale : CE en Hexa = 206 Décimal.

La signification d'un caractère (octet) dépend donc d'un protocole à connaître avant toute interprétation, avant toute transmission d'informations .

Quelle est la stucture de l'information stockée ?

Le signal est échantillonné à des instants régulièrement répartis sur la durée d'un balayage écran ; le nombre d'échantillons est fréquemment de 2048 (ou 2 kilooctets : 2 ko). Les oscilloscopes Métrix OX750, Métrix OX7520, Hameg, numérisent chaque échantillon avec 8 bits, ce qui permet une représentation par un octet. Exemple : une tension visualisée par un spot, situé à l'instant «saisie de l'échantillon», en

haut de l'écran, sera représentée par l'octet 00000000 ou 00 Hexa ; pour un spot situé en bas de l'écran, par 11111111 ou FF Hexa. Le signal numérisé est donc un tableau (2048 valeurs), chaque valeur étant un octet : nous dirons alors que le signal est échantillonné par 2048 «mots» sur 8 bits. Les résolutions seront donc :

– celle du pas de quantification en temps : 10*calibre de la base de temps / (2048 – 1),

– celle du pas de quantification en tension : 8*calibre / (256 – 1), pour un écran 8 cm*10 cm.

Exemple de signal : rampe montante à partir du coin inférieur gauche de l'écran : FF FE FD FC FB FA F9 F8 ..... pour une représentation à l'aide des codes ASCII et une numérotation hexadécimale. Ce sera avec ce codage que l'oscilloscope transmettra le signal.

Pour les autres informations numérisées, fixes ou variables, il est convenu de les représenter par les caractères ASCII des octets associés. Il faudra donc connaître les codes qu'a implantés le constructeur dans tel oscilloscope pour :

– commander à l'oscilloscope telle ou telle opération (déclenchement d'un balayage et saisie-échantillonnage d'un signal, ou transmission vers un PC du signal),

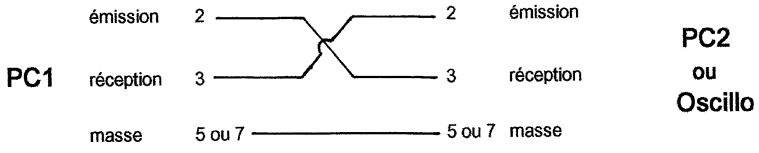
– interpréter les caractères reçus.

Ces codes sont spécifiques à un modèle d'oscilloscope. Un programme de communication avec l'oscilloscope doit donc inclure une unité de pilotage spécifique à chaque modèle ; ces unités sont appelées «pilotes» ou «drivers».

2. TRANSMISSION D'INFORMATION PAR LA LIAISON SÉRIE

Le câble.... essentiel

Dans une liaison série, l'émetteur et le récepteur sont reliés par 3 fils (2 «actifs» + 1 de masse).



Sur PC, le brochage sortie Série est souvent mâle ; si elle est de 9 broches (DB9) alors masse sur broche 5, si de 25 broches (DB25) alors masse sur broche 7.

Figure 1 : Câble à 3 fils pour la liaison Série.

L'émetteur fait varier la tension de sa sortie émission, le récepteur teste la tension sur son entrée réception. Chaque partenaire étant alternativement émetteur et récepteur, il y a nécessité de croiser les fils.

Les modes de transmission

Dans une transmission série, les octets sont envoyés bit par bit, en succession temporelle. La vitesse de transmission se chiffre en bits par seconde ou bauds (une valeur courante : 9600 bauds, soit au maximum 9600 / 8 = 1200 octets par seconde dans ce cas). Dans le protocole asynchrone, cas de l'interface RS232, chaque caractère est :

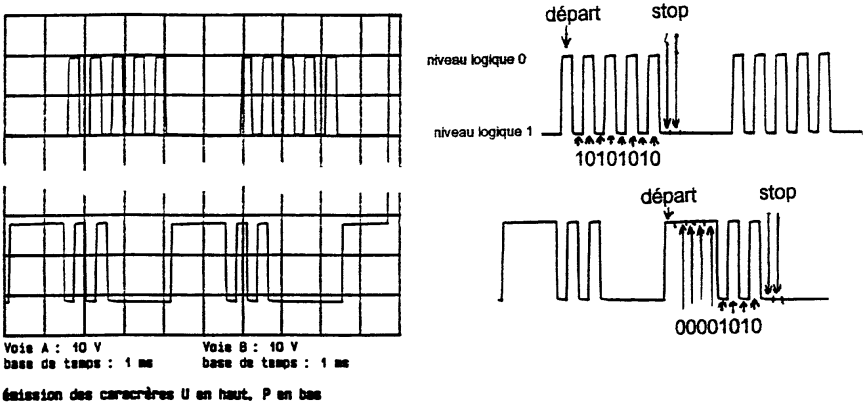
– devancé par un bit de départ qui indique au récepteur le début du caractère envoyé,

– suivi de bits d'arrêt (1, 2, voire 1,5 !!) qui indique la fin du caractère.

En outre, pour le contrôle de la conformité de la transmission, un ou deux bits, dits de parité, peuvent être insérés devant les bits d'arrêt. Ces bits qui encadrent les bits de données sont dits bits de contrôle.

La vitesse réelle de transmission des données en octets est donc nécessairement inférieure au nombre de bauds / 8 ; soit approximativement les 2/3 au plus.

Paramètres de l'émission : 3 600 bauds ;
8 bits de données (le code ASCII du caractère transmis) ;
pas de bit de parité ; 2 bits d'arrêt ;
soit 11 bits pour 8 bits de données ; (8 + 1 de départ et 2 d'arrêt).
Émission continue de codes ASCII identiques séparés de temps morts approximativement égaux, commandée par le logiciel, une simple modification de programme en annexe 1. Nous avons relevé avec un oscilloscope la tension entre les broches 2 et 5 sur une sortie en DB9, en ouvrant la broche de ce connecteur ; nous visualisons ainsi la transmission bit par bit des caractères ; en voici deux enregistrements à l'aide d'un Métrix OX750.



Observation : l'état d'attente d'émission est à – 10 Volts ;
le bit de départ fait passer à l'état + 10 Volts ;
le caractère U étant codé ASCII par 85 décimal soit 01010101 binaire et
le caractère P étant codé ASCII par 80 décimal soit 01010000 binaire, on vérifie qu'un bit positionné à 0 place la ligne à + 10 Volts, et à – 10 Volts pour un bit à 1 ; on parle alors de «logique négative» ;
les bits de données sont envoyés du bit 0 en tête au bit 7 pour terminer ;
les bits d'arrêt ramènent la tension sur la position attente.

Durée de l'état d'un bit mesuré sur un chronogramme : 0.27 ms ;
Durée calculée avec la vitesse : 1/3600 en seconde = 0.28 ms.

Figure 2 : Chronogrammes d'émissions de codes ASCII par un PC.

Dans une transmission parallèle, les bits de chaque octet sont transmis simultanément. Cette méthode est plus rapide, mais est «coûteuse» en conducteurs : il faut 8 + n + 1 conducteurs si n est le nombre de bits de contrôle à envoyer simultanément à l'octet.

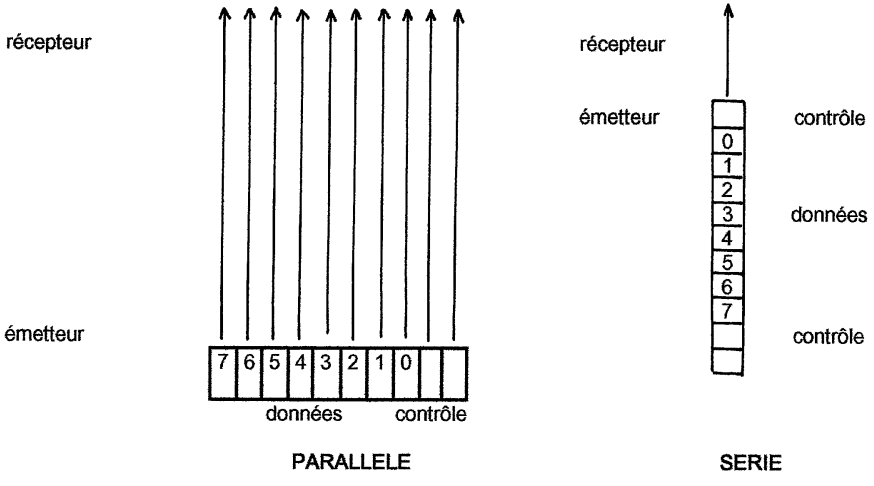


Figure 3 : Modes de transmission des données, en parallèle, en série.

La carte d'interface série, l'«UART»

Pour les transmissions série, un circuit spécial, ou carte série, implantée dans le PC, et sous une forme analogue dans l'oscilloscope numérique, réalise les fonctions suivantes :

– conversion de l'octet en une succession temporelle des bits de cet octet,

– opération inverse pour la réception,

– émission des bits spéciaux (départ, arrêt, parité),

– écriture et conservation des paramètres nécessaires à la transmission (vitesse, structures des données transmises),

– tenue du cahier d'état des commucations en cours, exemples :

- «un octet vient d'être reçu»,

- «une erreur de transmission est détectée»,

– écriture et maintien du mode de travail de ce circuit ; schématiquement, il existe deux modes :

- mode par interruption : quand un octet est reçu, le circuit le transmet immédiatement au cœur du PC, ce qui nécessite une «interruption» ;

- mode par scrutation : l'octet est placé dans le tampon de l'interface sans transmission au cœur du PC, ce qui fait que si un nouvel octet est reçu, le précédent, s'il n'a pas été lu, est écrasé.

Le mode par scrutation est donc plus risqué, il faut scruter fréquemment le tampon pour voir si un caractère est stocké. Nous choisissons cependant ce mode car il est très simple à mettre en œuvre.

La carte série est souvent nommée UART (Universal Asynchron Receiver Transmitter), circuit 8250 pour les PC.

Pour réaliser ces fonctions, l'UART possède des registres (mémoires) dont certains sont accessibles en écriture et ou en lecture :

– un registre de transmission de données,

– un registre pour initialiser les paramètres de transmission (vitesse, stucture, ...) des données,

– un registre de l'état de la transmission (status) :

- tampon d'émission plein ou vide ?
- tampon de réception plein ou vide ?
- erreur (de quel type ?) ou pas d'erreur ?

– un registre pour stocker le mode de fonctionnement (scrutation, interruption, ...),

– etc...

Les adresses de ces registres sont codées dans le PC sous 16 bits ; cf mini-programme. Ces adresses permettent de lire et (ou) d'écrire des données sur un emplacement particulier de l'UART ; cet emplacement est nommé «port» ; il ne fait pas partie de la mémoire du PC, il sert de mémoire intermédiaire de communication avec les périphériques :

– une interface série COM1 a des adresses de $03F8 à $03FF,

– une interface série COM2, de $02F8 à $02FF.

Le langage Turbo Pascal utilise le même mot Port pour accéder aux ports de données en lecture ou en écriture.

3. MINIPROGRAMME EXPLOITANT LA LIAISON SÉRIE PAR SCRUTATION

Je propose comme réalisation simpliste et démystifiante, l'utilisation d'un PC comme émetteur et d'un second comme récepteur, en proposant de frapper un caractère au clavier de l'émetteur et de le faire écrire à l'écran du récepteur.

Initialisations

Les programmes lancés sur les deux PC doivent effectués les mêmes initialisations de leurs interfaces série :

– mêmes vitesses de transmission,

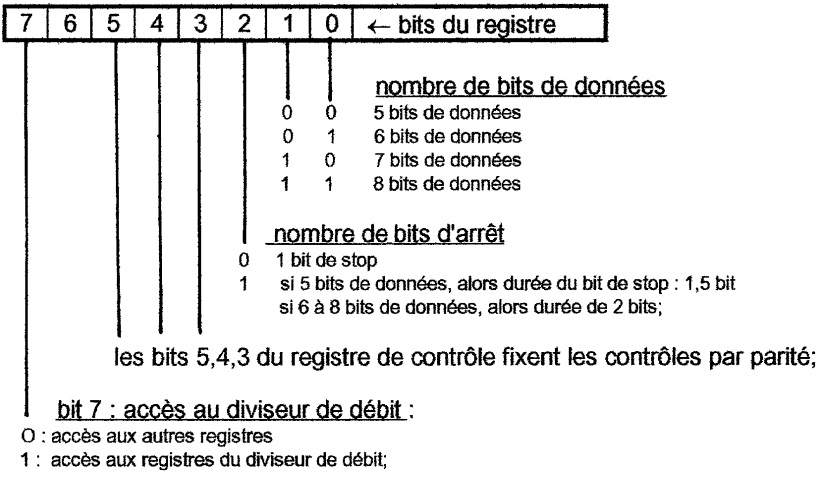
– structures de données reçues et transmises identiques.

Nous pouvons donc placer la même procédure d'initialisation dans les progammes émetteur et récepteur ; Cf Annexe 1 : programmes «reçois» et «émets».

Initialisation de la vitesse : la vitesse d'émission des bits est commandée par une horloge interne à l'UART. Sa fréquence est 1843200 Hertz. La vitesse d'émission est obtenue par une division de fréquence. Un coefficient diviseur est à écrire pour obtenir la vitesse, il sera stocké sous forme de 16 bits, dans les registres dit du «diviseur de débit» : vitesse = 1843200 / (coefficient * 16), d'où le tableau de correspondance :

Vitesse bauds	50	150	300	600	1200	1800	2400	3600	4800	9600	19200
Coefficient hexa	0900	0300	0180	00C0	0060	0040	0030	0020	0018	000C	0006
Coefficient décimal	2304	768	384	192	96	64	48	32	24	12	6

L'accès en écriture sur les registres du diviseur de débit est obtenu en positionnant à 1 le bit 7 du registre de contrôle de ligne. La figure 4 donne le contenu de ce registre. On écrit ensuite le coefficient sur le registre de transmission des données, puis on doit refermer l'accès au registre du diviseur de débit. Cf lignes 1 et 2 de la procédure Initialisations du miniprogramme dans l'annexe 1.



NB 1 : le numérotage des bits d'un octet est fait de 0 à 7, le bit 0 étant le bit de poids le plus faible, un bit 0 positionné à 0 donne un octet pair, positionné à 1 octet impair.
NB 2 : nous n'utilisons pas les bits 3, 4, 5 et 6 ; nous ne décrirons pas leurs rôles.

Figure 4 : Signification des bits du registre de contrôle de ligne.

Structure des données : la figure 4 donne l'interprétation des bits du registre de contrôle de ligne. Pour transmettre des données de 8 bits (cas d'octets «complets»), sans contrôle de parité, avec 2 bits de stop, les 6 bits de poids les plus faibles seront donc : 000111 ; on n'utilise pas ici le bit 6 qui sera positionné à 0 ; en positionnant à 0 le bit 7, l'accès aux registres du diviseur de débit est simultanément fermé. L'octet à écrire sur le registre de contrôle de ligne est donc : 00000111, soit 07 Hexa ou Décimal ; Cf ligne 3 de la procédure Initialisations du mini-programme dans l'annexe 1.

Deux autres registres doivent aussi être initialisés ; lignes 4 et 5 de la procédure initialisations que nous ne décrirons pas.

Émission d'un caractère

L'interface série ne doit émettre le caractère que si le tampon d'émission est vide pour ne pas écraser un caractère qui y aurait été placé antérieurement. D'où la boucle d'attente (repeat....until) de la procédure emetsCaractere(c:char), Cf annexe 1, qui attend que le tampon soit libre en sondant le bit 5 du registre d'état de ligne dont la figure 5 donne l'interprétation des bits utiles dans notre utilisation.

Quand le tampon est libre le caractère y est rangé, l'UART effectue automatiquement ensuite l'émission des bits correspondants, à son rythme, relibérant le tampon dès que cette émission est terminée.

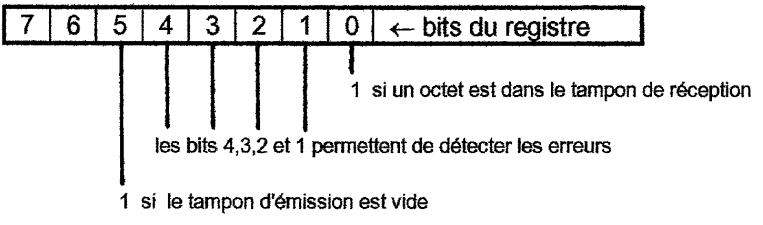


Figure 5 : Signification des bits du registre d'état de ligne (status).

Lecture d'un caractère

Une lecture de caractère ne peut se faire que si le tampon de réception en contient effectivement un. Nous avons choisi une fonction pour résultat de l'opération lecture, Cf function f-lisCaractere:char dans l'annexe 1. Elle comporte donc une boucle d'attente (repeat...until) s'arrêtant dès que le tampon de réception est plein, en sondant le bit 0 du registre d'état de ligne, Cf figure 5. Dès que ce bit est positionné à 1, la lecture du registre de transmission de données est effectuée : il contient un caractère reçu.

Programme principal du récepteur

Par lui même il est très simple : 5 lignes utiles en TurboPascal : initialisations, et une boucle lecture puis écriture à l'écran du caractère reçu jusqu'à ce que le caractère reçu soit 'Q'. Programme principal de l'émetteur : par lui-même aussi très simple : initialisations puis une boucle comprenant lecture du caractère au clavier, son écriture à l'écran et son émission, jusqu'à l'émission du caractère 'Q' décrété comme terminateur de l'expérience.

Le programme devra être plus étoffé si on souhaite que chaque PC soit émetteur et récepteur ; il faudra apprendre à gérer certains conflits : émission et réception simultanée pour un PC par exemple. Le mode de travail par «interruption» s'avère alors plus efficace.

4. MINI-PROGRAMME D'ACQUISITION : VERS UN «DRIVER» SPÉCIFIQUE DU MÉTRIX OX750

Nous conserverons le mode de transmission par scrutation : l'oscilloscope n'est pas capable d'initiative, il ne fait qu'exécuter les ordres qui seront envoyés par le PC ; le programme sera conçu pour permettre l'émission d'un ordre vers l'oscilloscope puis la lecture de la réponse à cet ordre. Il n'y aura pas de conflit de simultanéité émission-réception si on attend la fin d'une réponse avant l'émission de l'ordre suivant. Mais qu'est-ce-que la fin d'une commande ? Qu'est-ce-que la fin d'une émission ? Nous voyons donc la nécessité d'un protocole dans ce type de transmission.

Émission du PC vers l'oscilloscope

La commande est une suite de caractères dont l'oscilloscope reconnait la signification. La suite doit être terminée par un caractère que l'oscilloscope identifiera comme la fin de l'ordre de commande. Je nomme ce caractère d'usage réservé dans l'émission vers l'oscilloscope, terminateur en commande. Nous avions fait usage dans le mini-programme de transmission entre les deux PC du caractère 'Q' comme terminateur commun à la fin de l'expérience de transmission.

Réception par le PC

L'oscilloscope émet, suite à une commande, un flot de caractères dont il doit identifier la fin, sinon il attendrait.... attendrait. Un ou des caractères seront donc d'usage réservé dans ces phases de réception par le PC et émission par l'oscilloscope, comme terminateur d'envoi d'une réponse. Je les nomme pour la suite terminateurs en réponse.

Nous avons donc à consulter les notices des constructeurs de matériels, pour connaître les caractères réservés à ces usages, en l'absence de normalisation faisant totalement autorité. Le matériel utilisé par la suite pour illustrer ces transmissions est le Métrix OX750, dont le nombre de commandes est relativement restreint. Voir ci-dessous pour leur détail.

Mini-programme d'acquisition

Nous nous proposerons simplement de recevoir de l'oscilloscope son identificateur : «je suis l'oscilloscope untel», sans tester la conformité du matériel utilisé par rapport à celui que le programme a prévu ; puis d'écrire à l'écran le codage du signal numérisé.

La figure 6 en donne un organigramme, l'annexe 2 un listing rédigé en Turbo Pascal .

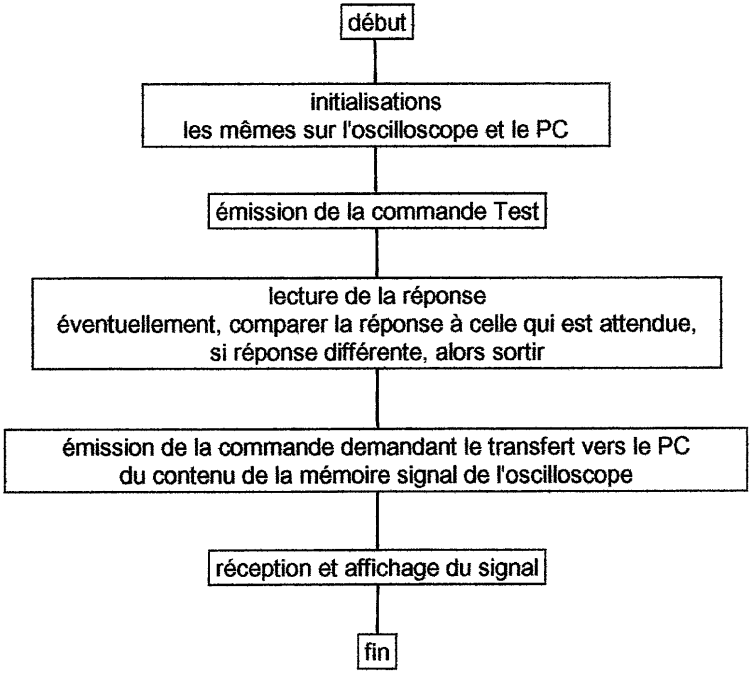


Figure 6 : Organigramme du mini-programme d'acquisition.

Initialisations

La mise sous tension de l'oscilloscope installe des paramètres de transmission prédéfinis. Nous en avons connaissance par la notice constructeur ; nous initialisons donc seulement l'interface série du PC avec ces paramètres. La procédure d'initialisations du mini-progamme de transmission de PC à PC sera reprise.

Émission d'une commande

La procédure consistera en l'émission de tous les caractères de la chaîne définissant la commande, suivie de l'émission du terminateur en commande.

Le Métrix OX750 utilise le caractère code ASCII = 13 décimal comme terminateur en commande.

Le Métrix OX750 utilise le protocole dit «XON / XOFF» : l'envoi par le PC du caractère de code ASCII = 17, dit «XON», «réveille» l'oscilloscope et le rend prêt à recevoir une commande, celui-ci répond alors par le même caractère ; nous inclurons donc dans la procédure de commande, l'envoi de ce caractère et la lecture en retour de ce même caractère. Cf procédure EmetsCommandeMetrix750, annexe 2. L'envoi du caractère de code ASCII 19, dit «XOFF», place l'oscilloscope en veille.

Réception d'une réponse

La réponse étant une suite de caractères, il suffira de lire les caractères reçus jusqu'à lecture d'un des terminateurs en réponse, les caractères reçus constituant la chaîne réponse.

Le Métrix OX750 renvoie des réponses structurées en suites de blocs de 48 caractères suivis de 2 terminateurs en réponse, la fin de chaque bloc comportant un premier terminateur de code ASCII 13, et un deuxième terminateur, soit de code ASCII 4 décimal si la réponse se termine par ce bloc, soit de code ASCII 10 si d'autres blocs seront envoyés derrière celui-ci. La procédure lecture d'une réponse testera le caractère fin de bloc pour s'assurer de la fin effective de la transmission de la réponse.

La procédure lisBlocMetrix750 de l'annexe 2 renvoie la chaîne reçue ; de plus, elle renvoie la valeur de la booléenne "BlocTerminal" indiquant si le bloc est un bloc terminal de réponse.

Réception du signal numérisé

Rappelons que l'oscilloscope transmettra 2048 mots de 8 bits soit 2048 octets ; chaque octet étant codé en Hexadécimal (Cf § 1., structure de l'information numérisée) nécessite 2 caractères pour ce codage, à prendre dans la liste 0,1 2,...,9,A...E,F. Les 2048 octets sont transmis par blocs de 16, chaque couple de 2 caractères représentatif d'un octet étant suivi d'un caractère «Espace» code ASCII 32 décimal ; on retrouve ainsi la structure des blocs précédents de taille 48 caractères (16*(2+1) = 48) ; chaque bloc se terminant par les deux terminateurs en réponse.

La lecture du signal comprendra donc une boucle (repeat until) comportant :

– lecture d'un bloc,

– écriture à l'écran du bloc transmis,

– un test par une variable booléenne («CestFini») renvoyée à chaque lecture de bloc indiquant si le bloc lu est terminal ou non,

– un compteur permettant de suivre le nombre d'échantillons signal reçus.

La transmission du signal de 2048 «mots» de 8 bits aura donc nécessité la transmission de 128 blocs, soient 128*(48+2) = 6400 caractères effectifs. Voilà qui couvre plusieurs pages écran ; d'où le test «if (i mod 16) = 0» qui permet de marquer une pause tous les 16*16= 256 échantillons codés transmis.

Conclusions provisoires

J'ai présenté le principe de la transmission de l'information stockée dans la mémoire d'un oscilloscope en utilisant un nombre de procédures relativement restreint : 6. Elles ont été volontairement mises en œuvre dans deux programmes distincts en tentant de dégager ce qui est dû :

– d'une part à la nature de la transmission par liaison série,

– d'autre part à la nature de l'oscilloscope.

Les quatre premières procédures sont employables dans tout «driver» d'acquisition utilisant le mode de transmission série par scrutation. Il suffira de déclarer comme variables d'une part le coefficient définissant la vitesse de transmission, d'autre part, la structure des données. Les deux dernières seront à adapter au matériel utilisé.

Reste ensuite la réalisation du logiciel d'exploitation du signal ; cela est une autre histoire... Qu'on me permette de remercier le technicien du laboratoire du lycée, M. DROGUET, qui a bien voulu réaliser matériellement les multiples essais, avec une compétence et une curiosité bien agréable et sympathique.

BIBLIOGRAPHIE

Lang TRAN TIEN - Systèmes de mesures informatisées ; Masson, Paris.

David C WILLEN and... - L'assembleur du 8088 et de l'IBM PC ; Mémoire Vive, Québec.

Les numéros 29 et 30 de la revue Pascalissime ; Librairie «L'Institut Pascal» 28, rue Lamartine 75009 Paris ; revue fortement recommandée pour les exemples variés de programmation Pascal.

Annexe 1

Mini-programme de communication entre deux PC

```
program recois;          { Cf programme principal, emets; }

const
    k_tamponReceptionPlein = $01;
{ 01 Hexa = 00000001 binaire, le bit 0 du registre d'état est  positionné à 1 si un caractère a été reçu    }

    k_tamponEmissionVide = $20;
{                20 Hexa = 00100000 binaire, le bit 5 du registre d'état est positionné                    }
{                        à 1 si le tampon d'émission est vide                                              }
{                              utilisation du port série COM 1                                             }
    k_AdresseRegistreControleLigne =          $03FB;     {      Line Control                             }
    k_AdresseRegistreTransmissionDonnees =    $03F8;     { Tampon émission réception                     }
    k_AdresseRegistreControleModem =          $03FC;     {    Modem Control                              }
    k_AdresseRegistreInterruptions =          $03F9;     {    Interrupt enable                           }
    k_AdresseRegistreEtatLigne =              $03FD;     {       Status                                  }

procedure initialisations;
 begin
{ 1 }       port[k_AdresseRegistreControleLigne]:=$80;
        {                              80 Hexa = 10000000 binaire                                          }
        {en positionnant à 1 le bit 7, l'accès aux registres du diviseur de débit est ouvert               }

{ 2 }       portW[k_AdresseRegistreTransmissionDonnees]:=$000C;
        {  initialise la vitesse à 9600 bauds                                                              }
        { l'instruction PortW... permet d'écrire simultanément sur les 2 ports d'adresses                  }
        { successives 3F8 et 3F9 un mot de 16 bits ( 2 octets ), elle est équivalente à                    }
        {          Port[$3F8]:=$0C ; Port[$3F9]:=$00;                                                      }
        { sur 3F8 l'octet de poids faible, sur 3F9 l'octet de poids fort                                   }

{ 3 }       port[k_AdresseRegistreControleLigne]:=$07;
        {    longueur 8 bits, pas de bit de parité, 2 bits d'arrêt                                         }
        {    voir par ailleurs la structure de ce registre contrôle de ligne                               }

{ 4 }       port[k_AdresseRegistreControleModem]:=$03;
        {  pas de MODEM, ordinateur prêt, bits 0 et 1 de ce registre de contrôle positonnés à 1 }

{ 5 }       port[k_AdresseRegistreInterruptions]:=$00;
        {    émission et réception par scrutation                                                          }
 end;

function f_LisCaractere:char;
 begin
  repeat
  until port[k_AdresseRegistreEtatLigne] and k_tamponReceptionPlein = k_tamponReceptionPlein;
        {    attend que le bit 0 soit positionné à 1 avant de lire sur le port                             }
  f_LisCaractere:=chr(Port[k_AdresseRegistreTransmissionDonnees]);
 end;

procedure EmetsCaractere(c:char);
 begin
  repeat
  until port[k_AdresseRegistreEtatLigne] and k_tamponEmissionVide = k_tamponEmissionVide;
        {          attend que le bit 5 soit positionné à 1 avant d'affecter au port                        }
  Port[k_AdresseRegistreTransmissionDonnees]:=ord(c);
 end;
```

```
procedure EmetsChaine(s:string);
 var i:byte;
 begin
   for i:=1 to length(s) do EmetsCaractere(s[i]);
 end;


var c:char;
{ corps du programme principal recois  }
begin
 initialisations;
 writeln('récepteur');
 writeln('Arrêt ... la réception de "Q"');
 repeat
   c:=f_lisCaractere;
   write(c);
 until c='Q';
end.
```

```
{ corps programme principal emets          }
{ begin                                    }
{   initialisations;                       }
{   writeln('Emetteur');                   }
{   writeln('Arrêt par l''émission de "Q"'); }
{   repeat                                 }
{     c:=readkey;write(c);                 }
{     emetsCaractere(c);                   }
{   until c='Q';                           }
{ end.                                     }
```

Annexe 2

Mini-programme d'acquisition du signal numérisé sur la voie A de l'oscilloscope Métrix OX750

```
program miniAcq;
uses crt,dos;
{ réinsérer les constantes,                                                    }
{ les procédures initialisations,emetsCaractere,emetsChaine, et fonction f_lisCaractere  }
{ définies dans l'annexe 1                                                     }

const
{       les 6 constantes sont imposées par l'utilisation de l'oscilloscope Métrix OX 750  }
        k_TerminateurCommande =#13;
        k_PremierTerminateurReception =#13;
        k_SecondTerminateurReception1=#10;      { termine un bloc qui ne sera pas terminal }
        k_SecondTerminateurReception2=#4;       { termine un bloc qui sera terminal        }
        k_Eveille=#17;
        k_MetsEnVeille=#19;

procedure EmetsCommandeMETRIX750(commande:string);
 var l_caractere : char;
 begin
  emetsCaractere(k_eveille);
  l_caractere:=f_lisCaractere;      {   lis le caractère de retour signalant le réveil   }
  emetsChaine(commande+k_TerminateurCommande);
 end;

procedure lisBlocMETRIX750(var blocReponse : string;var BlocTerminal : boolean);
 var i:byte;l_caractere:char;
 begin
  i := 0;
  repeat
   l_caractere := f_lisCaractere;
   inc(i);          {          procédure équivalente à i:=i+1, en code optimisé          }
   blocReponse[i] := l_caractere; {      la chaine est construite caractère par caractère  }
  until l_caractere=k_PremierTerminateurReception;
  blocReponse[0] := chr(i-1);       { la longueur de la chaine est à ranger dans blocReponse[0] }
  l_caractere := f_lisCaractere;
        {     selon le dernier caractère lu, le bloc est un bloc terminal  de réponse ou non  }
  if l_caractere=k_SecondTerminateurReception1 then BlocTerminal := false;
  if l_caractere=k_SecondTerminateurReception2 then BlocTerminal := true;
end;

var     i,j:integer; c:char;
        reponse : string;
        CestFini : boolean;
begin           { programme principal d'acquisition sur Métrix OX750                   }
 clrscr;
 initialisations;
 emetsCommandeMetrix750('T');
 lisBlocMetrix750(reponse,CestFini);
 emetscaractere(k_MetsEnVeille);delay(2000);
                { un temps mort  s'avère nécessaire entre deux commandes                }
 writeln('Identification de l''oscilloscope : '+reponse);writeln;
 emetsCommandeMetrix750('A');
 lisBlocMetrix750(reponse,CestFini);writeln(reponse);
{ le premier bloc réponse ne contient pas le signal numérisé , mais une identication de la voie }
{       du signal lu par la commande, voir notice Métrix OX750                         }
 emetscaractere(k_MetsEnVeille);
 i:=0;
 writeln('contenu de la Voie A');
```

```
 emetscaractere(k_Eveille);
 repeat
    inc(i);j:=i*16;
    lisBlocMetrix750(reponse,CestFini);
    emetscaractere(k_MetsEnVeille);
    writeln(reponse);
    if (i mod 16)=0 then  begin  { on passe sur une nouvelle page écran  après 16*16 valeurs  }
         write(j,' échantillons transmis; frapper une touche pour continuer');
           while keypressed do c:=readkey;repeat until keypressed;writeln;
         end;
    if not CestFini then emetsCaractere(k_Eveille);
 until CestFini;
 while keypressed do c:=readkey;writeln('Fin');repeat until keypressed;
end.
```