GFI
MILL.

PROGRAMMATION DE LA CARTE DTL V 23 EN ASSOCIATION AVEC L'INTERFACE APPLE II

Cette partie de la notice contient toutes les informations dont vous avez besoin pour programmer la carte ***DTL V 23*** *depuis votre APPLE II.*

*

La carte ***DTL V 23*** *peut fonctionner selon deux modes bien distincts :*

- *le mode "asymétrique", qui permet l'accès à un serveur de type TELETEL,*
- *le mode "symétrique", qui permet la communication avec un autre micro-ordinateur.*

Fonctionnement en mode asymétrique

Les données sont émises par le serveur à 1 200 Bauds (bits par seconde) et par le terminal (APPLE + DTL 2 000) à 75 Bauds, d'après l'avis V 23 du CCITT (Comité Consultatif International Télégraphique et Téléphonique). Les données transmises sont en fait des caractères codés selon l'alphabet ASCII sur 7 bits.

Le modem ***DTL 2 000*** *occupe 6 octets de la mémoire de votre APPLE qui sont (en hexadécimal) : $CØS8, $CØS9, $CØSA, $CØSB, $CØSD, où S représente le numéro du slot utilisé plus 8.*

Les octets $CØSA et $CØSB ne sont pas utilisés par la carte DTL V 23.

Comme tout système informatique, le ***DTL 2 000*** *doit être initialisé au début de chaque programme le concernant. Cette initialisation peut s'écrire ainsi (les adresses doivent être converties en décimal, suivant la valeur de S) :*

```
1Ø POKE $CØS9, Ø
2Ø POKE $CØS8, 244
3Ø POKE $CØSC, 3
4Ø POKE $CØS9, 4
5Ø POKE $CØS8, 212
6Ø POKE $CØSC, 73.
```

Dans ce qui suit, le bit Ø désigne, pour un octet donné, le bit de poids le plus faible, et le bit 7 le bit de poids le plus fort.

* *

PROGRAMMATION

Une fois l'initialisation effectuée, les octets dont vous avez besoin pour gérer une communication de type asymétrique avec le serveur sont : $CØS8, $CØSC, et $CØSD.

*

$CØS8 :

Cet octet permet la connexion ou la déconnexion du modem sur la ligne. Pour connecter le modem, faire POKE $CØS8,208 (fermeture de la ligne) ; pour le déconnecter, faire POKE $CØS8,212 (ouverture de la ligne).

Dans le cas qui nous concerne, à savoir la communication asymétrique utilisant la carte DTL V 23, il ne faut pas poker d'autres valeurs dans cet octet.

Puisque la composition d'un numéro téléphonique est une suite de coupures de ligne, l'utilisation de cet octet permet une numérotation automatique. Ainsi, la composition du chiffre 7, par exemple, nécessite l'envoi sur la ligne de 7 impulsions composées chacune d'une ouverture de ligne d'une durée de 66 ms, suivie d'une fermeture d'une durée de 33 ms. La composition du chiffre 7 peut donc se traduire par le programme Basic :

```
11Ø  FOR I = 1 TO 7
12Ø  POKE $CØS8,212
13Ø  FOR J = 1 TO 44 : NEXT J
14Ø  POKE $CØS8,208
15Ø  FOR J = 1 TO 22 : NEXT J
16Ø  NEXT
```

(Note : La composition du chiffre Ø correspond à l'envoi de 10 impulsions).

Entre deux trains d'impulsions, la ligne doit être fermée, et cela pendant 1 s environ. Après la composition du numéro, la ligne doit être maintenue fermée jusqu'à la fin de la communication. Enfin, dans un programme de composition automatique de numéros, n'oubliez pas de prévoir un ou des délais permettant d'attendre l'apparition de la ou des tonalités nécessaires à la composition.

*

$CØSC :

Cet octet permet le contrôle de la transmission.

Contrairement à un octet de mémoire "classique", $CØSC n'est physiquement pas le même, suivant qu'on y accède en lecture ou en écriture.

***En écriture**, $CØSC permet de commander l'émission de porteuse du **DTL 2 000**. Pour commencer l'émission, faire POKE $CØSC,9 ; pour l'arrêter, faire POKE $CØSC, 73.*

Dans le cas d'une communication asymétrique utilisant la carte DTL V 23, il ne faut pas poker d'autres valeurs dans cet octet.

***En lecture**, $CØSC contient des informations essentielles sur le déroulement de la transmission (PRINT PEEK [$CØSC]) :*

- *le bit Ø indique qu'un caractère vient d'être reçu s'il est à 1 (remis à Ø par lecture du caractère : cf. $CØSD) ;*
- *le bit 1 indique que le caractère que l'on vient d'émettre est effectivement parti s'il est à 1 ;*
- *le bit 2 indique que la porteuse du modem du serveur avec lequel on communique est présente s'il est à Ø ;*
- *le bit 3 indique que le modem est prêt à émettre s'il est à Ø.*

Nous allons voir dans la suite comment utiliser ces indications.

*

$CØSD :

C'est par cet octet que transitent toutes les données transmises dans les deux sens. Comme $CØSC, cette adresse correspond en fait à deux octets, suivant qu'on y accède en lecture ou en écriture.

***En lecture**, $CØSD contient le dernier caractère reçu, qu'il faut lire lorsque le bit Ø de $CØSC est passé à 1.*

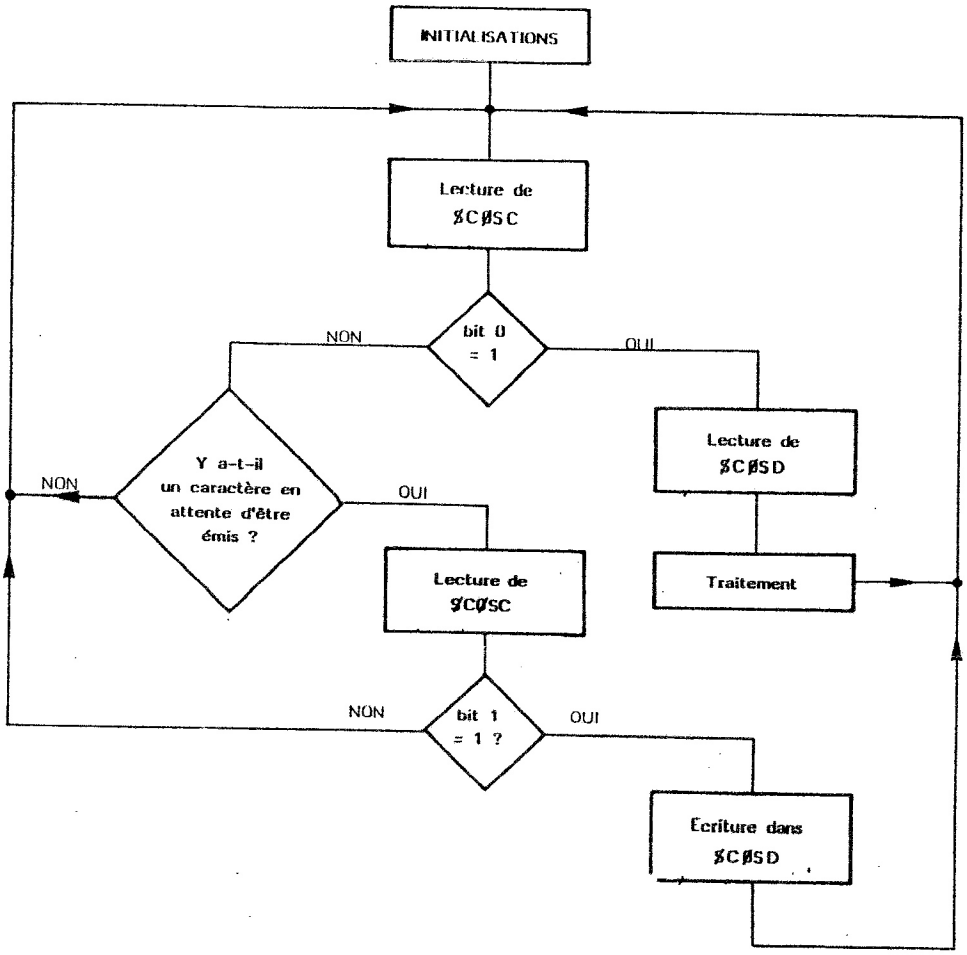
***En écriture**, $CØSD reçoit le caractère que l'on veut émettre. On ne peut écrire dans $CØSD qu'après avoir vérifié que le bit 1 de $CØSC est passé à 1.*

N'oubliez pas qu'en émission comme en réception, les caractères sont codés sur 7 bits uniquement, conformément à l'alphabet ASCII (le bit 7 est toujours à 1).

*

Algorithme de communication :

Voici un exemple d'algorithme utilisant les indications précédentes.



Les initialisations comprennent :

- *l'initialisation générale évoquée plus haut,*
- *l'établissement de la communication (numérotation et connexion manuelle ou automatique),*
- *la vérification de la détection de porteuse (soit visuelle par le voyant* ***Détection****, soit automatique par un test sur le bit 2 de $C0SC (NB : dans ce dernier cas, la lecture de $C0SC doit être suivie d'une lecture "blanche" de $C0SD, qui est nécessaire à la réinitialisation du bit 2 de $C0SC),*
- *l'ordre d'émission (écriture de 9 dans $C0SC).*

Le traitement du caractère lu dans $C0SC consiste soit en un rangement dans une mémoire (transfert de fichiers), soit en un décodage suivi d'un affichage sur l'écran.

Les caractères en attente d'être émis peuvent provenir soit de la mémoire, soit du clavier.

Dans la pratique, étant données les vitesses de transmission, un tel logiciel de communication doit être écrit en Assembleur. Dans le cas du raccordement à un serveur de type TELETEL, les caractères que vous recevrez se répartiront en caractères ASCII, composant les caractères, et en caractères de commande (mise en page, etc...) codés suivant le protocole VIDEOTEX. La description de ces caractères de commande est relativement longue et déborderait de l'objet de cette notice.

Fonctionnement en mode symétrique

Contrairement à ce qui se passe dans le mode asymétrique, les données sont émises à tour de rôle par les deux interlocuteurs (half-duplex). En pratique, il s'agit de transmettre soit un programme, soit un fichier de données, d'un ordinateur vers un autre.

La vitesse est de 1 200 Bauds, indépendamment du sens de transmission.

L'utilisation des octets $CØS8, $CØS9, $CØSC, $CØSD est la même qu'en mode asymétrique, aux différences suivantes près :

* *

INITIALISATIONS

```
1Ø POKE $CØS9, Ø
2Ø POKE $CØS8, 244
3Ø POKE $CØSC, 4
4Ø POKE $CØS9, 3
5Ø POKE $CØS8, 196
6Ø POKE $CØSC, 85.
```

*

Connexion :

Pour connecter, faire : POKE $CØS8,192
pour déconnecter, faire : POKE $CØS8,196.

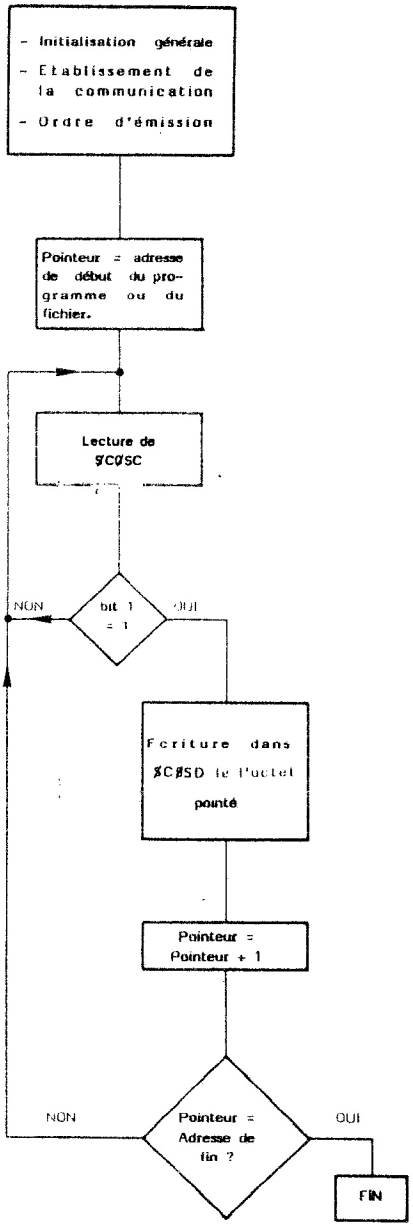
Emission :

Début d'émission, faire : POKE $CØSC, 21,
fin d'émission, faire : POKE $CØSC, 85.

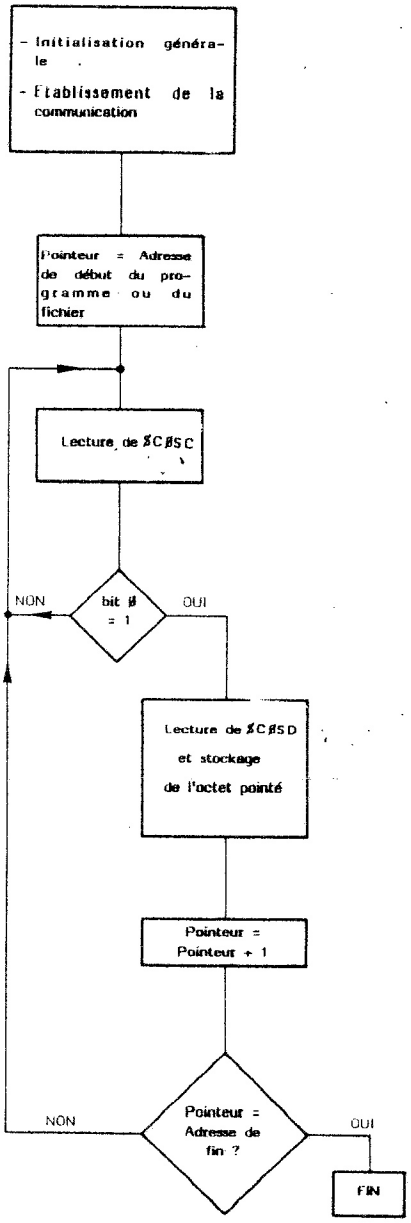
Les données sont transmises sur 8 bits.

Voici deux exemples d'algorithmes, l'un en émission, l'autre en réception, que vous pouvez utiliser pour transmettre vos programmes ou vos fichiers.

Emission :



Réception :



Là encore, il est nécessaire, compte tenu de la vitesse, de travailler en Assembleur.

Il est également possible, pour les deux correspondants d'assumer à tour de rôle les fonctions d'émetteur et de récepteur, et donc d'échanger des messages, par exemple, en temps réel.

* *