Septième partie: INTERFACE DISPLAY

INTRODUCTION

La télévision est un périphérique très puissant et bon marché. Toute information digitale peut être visualisée sur son écran, avec une résolution maximale de 300 fois 300 points. Dans la pratique, on se restreint à un nombre plus limité de points ou de combinaisons de points représentant des caractères, à cause de la nécessité de rafraîchir l'information 50 fois par seconde et de la grande capacité mémoire qui serait nécessaire pour mémoriser tant de bits.

SIGNAUX DU TELEVISEUR

Dans une télévision, deux générateurs de rampe dévient un spot dont l'intensité est modulée. La première rampe déplace le spot horizontalement et revient rapidement. La période est de 64 µs (15,625 kHz), le quart de cette durée étant utilisé par le retour de trace.

La deuxième rampe déplace le spot verticalement avec une période de 20 ms (50 Hz). Ainsi, 312 lignes horizontales sont définies sur l'écran, et si l'impulsion de retour vertical est donnée au milieu de la 313ème ligne, le prochain balayage est décalé d'une demi ligne vers le haut, ce qui définit le standard usuel de 625 lignes.

L'interface alphanumérique d'un système microordinateur ne nécessite pas une résolution de 625 lignes; 312 lignes suffisent, simplifiant l'interface. Il y a 3 signaux de contrôle. Le flanc positif de l'impulsion horizontale H déclenche le retour du spot. La durée minimale de H dépend du moniteur. Nous supposons ici que H est actif durant le retour du spot (rien à visualiser sur l'écran). La fréquence de H a une tolérance qui dépend du moniteur et peut s'élever à 10%.

Le flanc positif de l'impulsion de synchronisation verticale V déclenche le retour vertical du spot. Les impulsions H et V sont indépendantes l'une de l'autre. Comme dans le cas de H, le spot doit aussi être inhibé aussi longtemps que V est à l'état "1".

Le signal de modulation d'intensité Z est un signal binaire dans le cas d'un display digital. La figure 1 illustre l'exemple d'un balayage de 10 lignes.

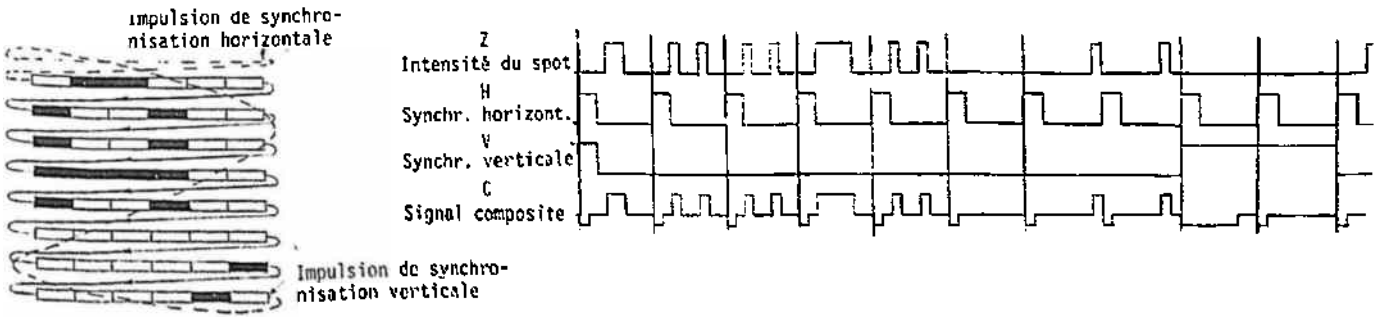


Fig. 1. Signaux de contrôle pour un display simplifié avec un balayage de 10 lignes, montrant 6x8 points.

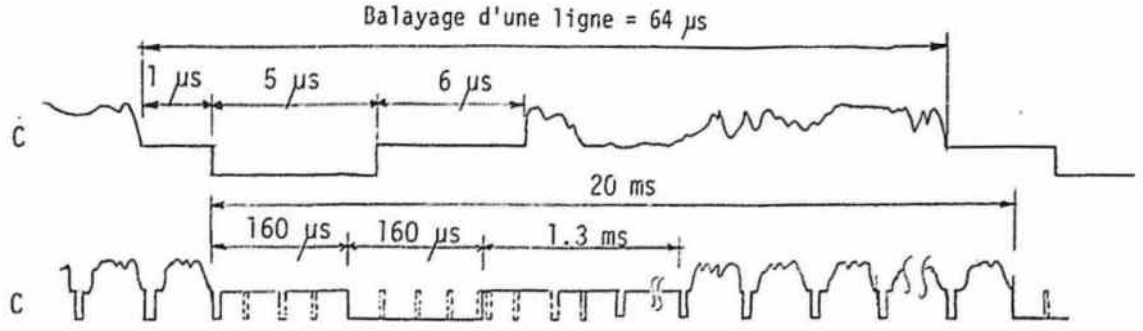


Fig. 2. Impulsions de synchronisation sur un signal vidéo composite. Les impulsions de pré- et post-égalisation n'ont pas à être générées dans le cas d'un display digital.

Les trois signaux H,V et Z sont directement accessibles sur les moniteurs vidéo fabriqués pour des applications digitales. La plupart des moniteurs TV nécessitent un signal composite C avec des contraintes plus strictes concernant les durées et les moments où les impulsions de synchronisation doivent intervenir (Fig. 2). Le signal à entrer sur l'antenne d'une télévision standard est équivalent à la modulation haute-fréquence du signal composite.

La résolution horizontale de l'écran (nombre maximum de points par ligne) est limitée par la largeur de bande de l'amplificateur et du tube TV. La fréquence maximale est d'environ 5 MHz pour une TV standard et 10 MHz pour un moniteur digital.

GENERATEUR DE CARACTERES

Comme nous l'avons vu sur la Fig. 1, un caractère alphanumérique peut être généré par une modulation adéquate du signal Z. Dans la pratique, on utilise des circuits intégrés générateurs de caractères. Ce sont des mémoires préprogrammées qui font correspondre à chaque code ASCII un ensemble de points représentant la lettre, générés sur la ligne Z.

Le paramètre important est le temps d'accès, c'est à dire le temps nécessaire pour générer la lettre à partir du dernier changement d'adresse.

Il faut trouver le meilleur compromis entre la densité de caractères, la lisibilité de chaque caractère et la simplicité des circuits de contrôle. Pour une bonne lisibilité, chaque caractère doit être entouré de suffisamment d'espace.

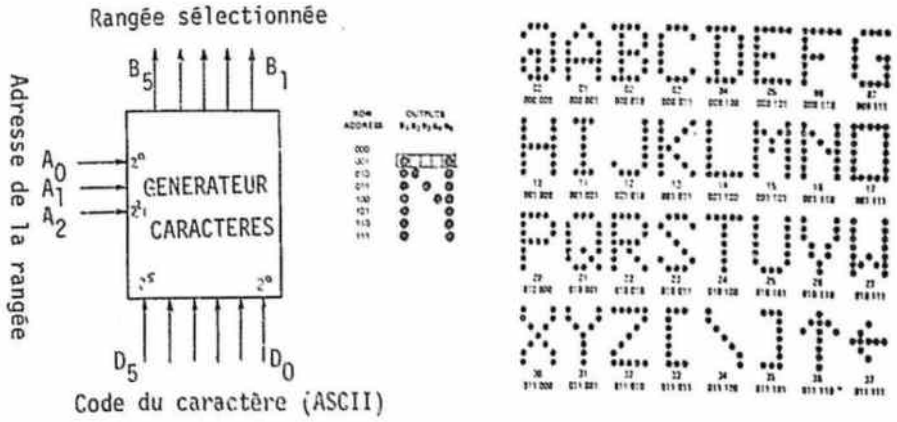


Fig. 3. Exemple de générateur de caractères 5x7.

Une chaine de division génère les signaux nécessaires. Le premier diviseur dépend du nombre de points par ligne de chaque caractère. Le deuxième diviseur compte le nombre de caractères par balayage, y compris les caractères fictifs qui ne sont pas visualisés pendant le retour de spot. La synchronisation horizontale et l'impulsion d'effacement H sont générés par ce compteur.

Le compteur suivant de la chaîne compte le nombre de balayages pour une ligne complète de caractères, y compris les lignes de séparation. Le dernier compteur détermine le nombre de lignes de texte sur l'écran, y compris les lignes non visualisées durant le retour du spot vertical, lorsque le signal V est actif.

Le nombre total de balayages devrait être 312 avec une télévision européenne. La plupart des moniteurs sont tolérants concernant cette valeur, et les moniteurs digitaux acceptent de 280 à 330 lignes, avec des fréquences de 45 à 60 Hz.

Pour afficher les caractères successifs, on pourrait utiliser un multiplexeur contrôlé par le premier compteur. Il est plus simple de charger un registre à décalage au moment où tous les bits correspondant à une ligne sont disponibles à la sortie du générateur de caractères, et de transmettre ensuite ces bits sur la ligne Z à la fréquence des points (Fig. 4). L'inhibition du signal Z durant les retours horizontaux et verticaux peut se faire au moyen d'une logique adéquate sur la sortie du registre à décalage ou sur la ligne de contrôle LOAD. Les sorties Q et R du compteur correspondent aux adresses du caractère sur l'écran. En reliant directement les signaux de sortie Q,R,S au registre à décalage, sans aucun générateur de caractères, on fait apparaître un motif régulier sur l'écran, très utile pour dépanner le système.

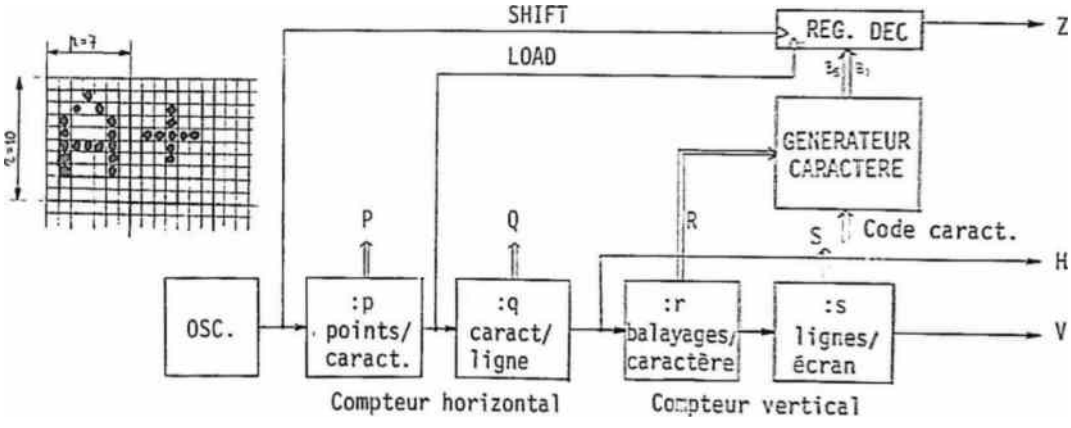


Fig. 4. Schéma-bloc de la chaine de division et du générateur de caractères.

GENERATION D'UNE PAGE DE TEXTE

Une page de texte est mémorisée dans une mémoire locale spéciale. Les caractères sont transmis au display en parallèle, selon le code ASCII (Fig. 5).
En général les aiguillages de contrôle de la mémoire de texte sélectionnent le display et les contenus de la mémoire sont transférés au générateur de caractères en synchronisation avec les compteurs. Si le processeur doit lire ou modifier une position dans la mémoire de texte, il contrôle les interrupteurs de son côté et fait le transfert. Si le transfert se produit lorsque le spot est sur une partie visible de l'écran, il se produit une petite perturbation (pour l'éviter, il ne faut permettre l'accès à la mémoire de texte uniquement durant le retour de spot).

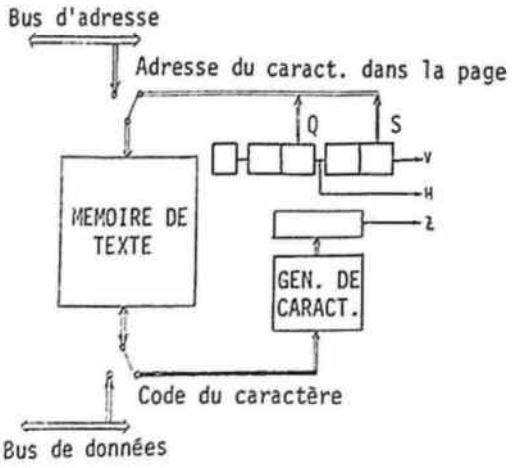


Fig. 5. Organisation de la mémoire du display.

INTERFACE GRAPHIQUE SIMPLE

Une mémoire de 256x8 points permet d'afficher 2048 points sur un écran, selon une grille de 64x32 points. Le générateur de caractères n'est pas nécessaire puisque chaque bit de la mémoire est directement affiché sans transformation. La figure 6 donne le schéma bloc de ce type d'affichage. Chaque points est formé de 8 balayages successifs, afin de lui donner une forme grossièrement carrée.

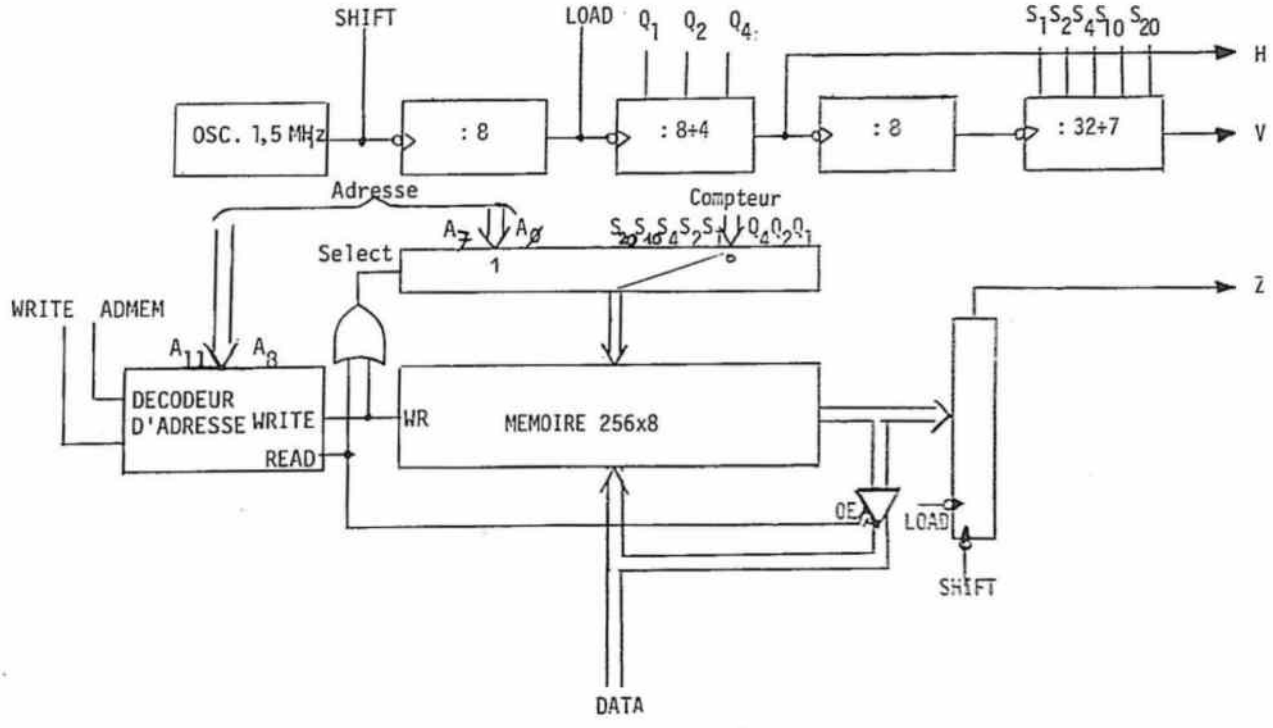


Fig. 6. Schéma bloc fonctionnel d'un affichage graphique simple.

Il y a avantage à utiliser des mémoires avec des entrées et sorties de données séparées. Une porte 3 états est nécessaire pour relire le contenu d'une position affichée. Lorsque le processeur sélectionne la mémoire, l'écran affiche le caractère sélectionné (lu ou écrit) quelle que soit la position du spot.

EXPLICATION DU SCHEMA DE L'INTERFACE GRAPHIQUE ET TEST

La disposition du schéma correspond très sensiblement à la disposition de la figure 6.

La chaîne de compteurs fournit les signaux de synchronisation pour la mémoire et la TV. Les sorties Z,H,$\overline{V}$ peuvent contrôler directement un moniteur digital Ball Brother, MDS, INELCO, etc.

La sortie composite permet de contrôler un moniteur Sanyo, Sony, etc, ou une télévision modifiée. Un modulateur HF peut être branché entre cette prise et la prise antenne d'une TV ordinaire.

Deux potentiomètres permettent le réglage du centrage de l'image (vers la prise DIN) et deux autres (vers le quartz) règlent le niveau de synchronisation et l'amplitude du signal composite.

Le décodeur d'adresse sélectionne l'adresse 1400. Une adresse quelconque peut être choisie grâce aux jumpers, et il est possible de placer plusieurs de ces plaques et plusieurs écrans sur un seul DAUPHIN, avec les adresses différentes pour chaque plaque naturellement (attention à la charge du bus).

Les aiguillages, la mémoire et le sérialiseur se trouvent au centre du schéma. Une impulsion retardée d'écriture est fabriquée par une porte ET, et le chargement du registre série est inhibé lorsque la mémoire est sélectionnée (avec un retard supplémentaire grâce à une porte OU). Ceci évite que lorsque la mémoire est sélectionnée par le processeur, un trait apparaisse sur l'écran. En fait le trait existe, mais sous forme d'absence d'intensité, moins visible si l'écran comporte une image peu dense.

Pour le dépannage, vérifier dans l'ordre

- l'oscillation à 3,072 MHz et 1,5 MHz
- les signaux NOH et NOV, puis H, $\overline{V}$
- l'écriture et la lecture correcte en mémoire à l'adresse 1400
- le signal Z et le signal composite.

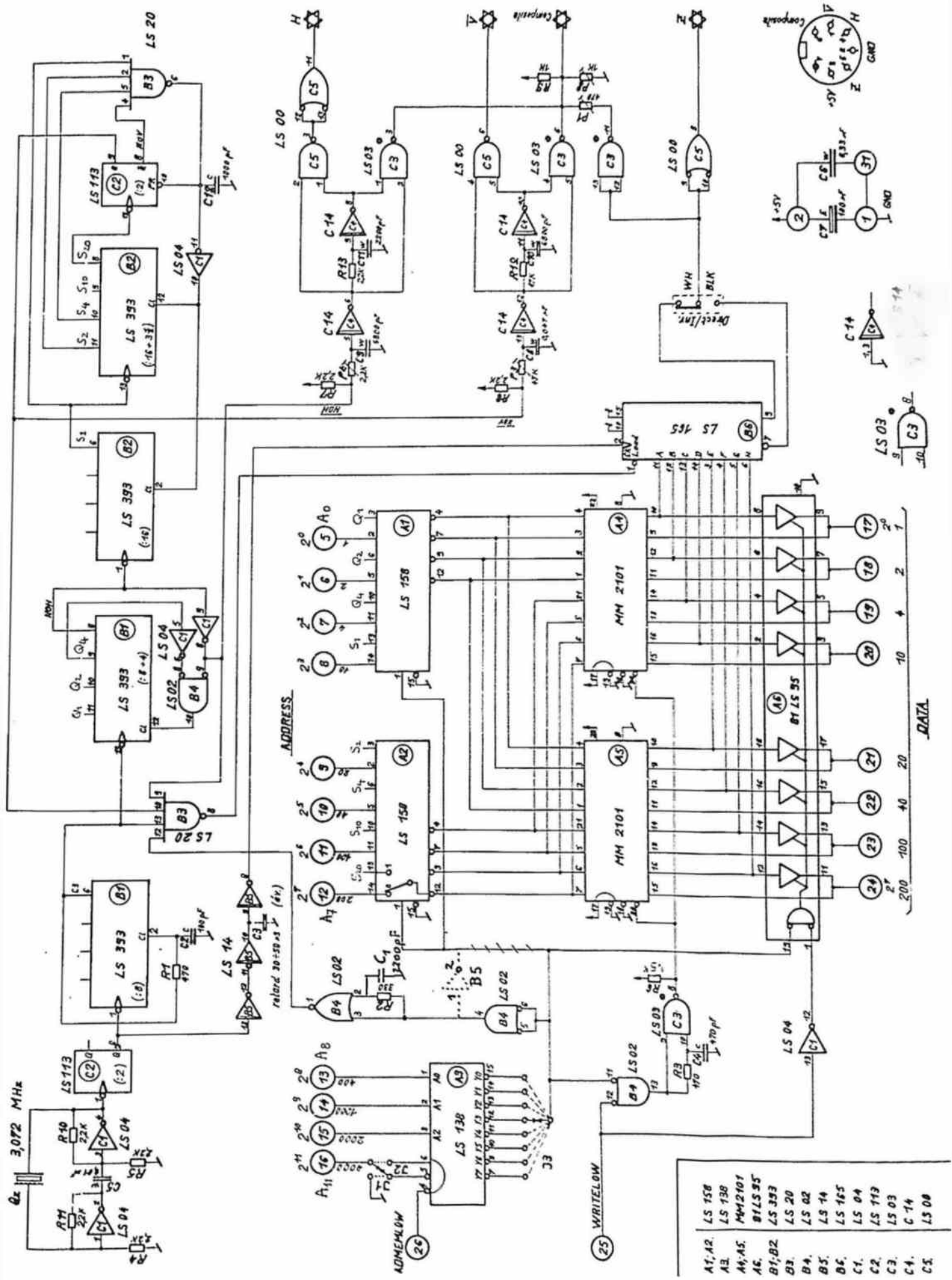
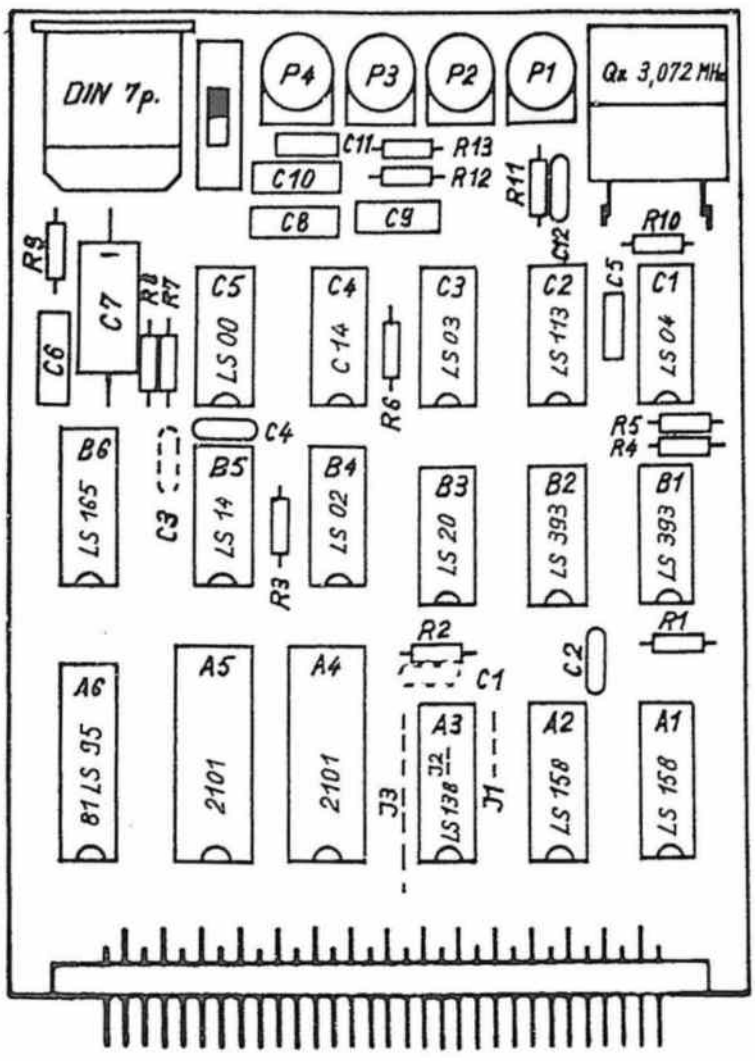


Schéma de l'interface display graphique



R1 – 470
R2 – 330
R3 – 470
R4 – 2,2K
R5 – 2,2K
R6 – 1,5K
R7 – 2,2K
R8 – 2,2K
R9 – 1K
R10 – 2,2K
R11 – 2,2K
R12 – 47K
R13 – 2,2K

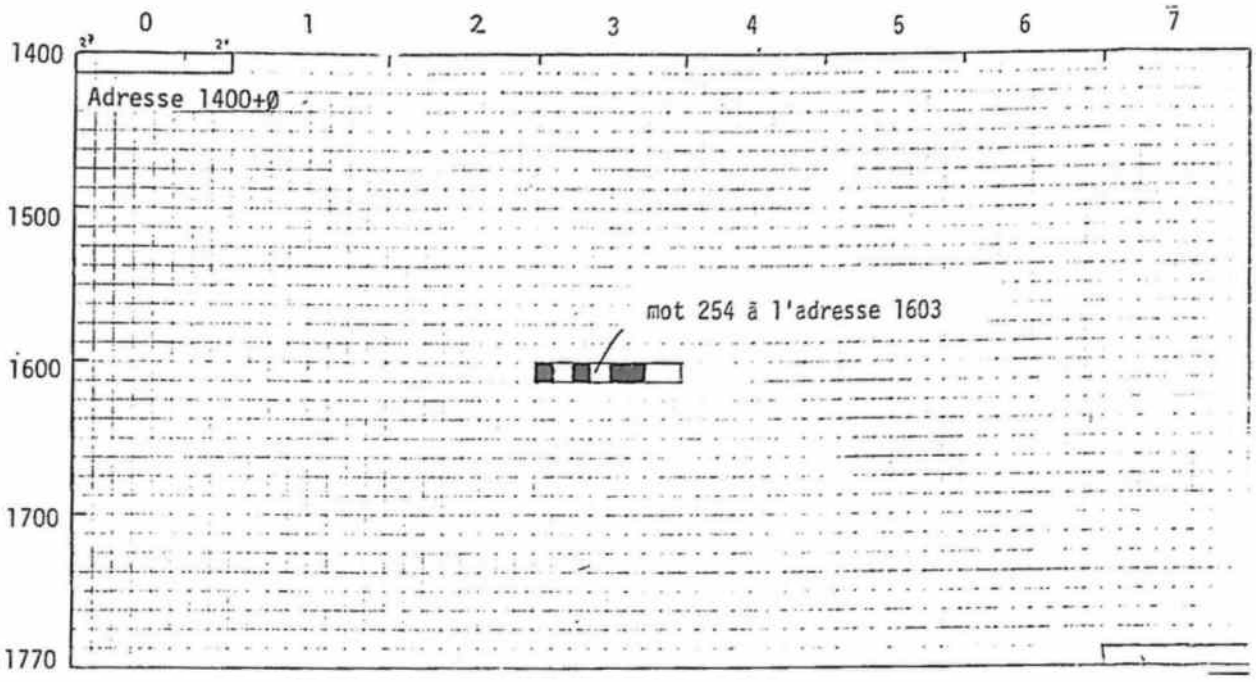
C1 – 2200 pF/C
C2 – 100 pF/C
C3 év.
C4 – 470 pF/C
C5 – 0,01 µF/W
C6 – 0,33 µF/W
C7 – 100 µF/E
C8 – 0,047 µF/W
C9 – 6800 pF/W
C10 – 6800 pF/W
C11 – 2200 pF/W
C12 – 1000 pF/C

P1 – 470
P2 – 1K
P3 – 47K
P4 – 2,2K

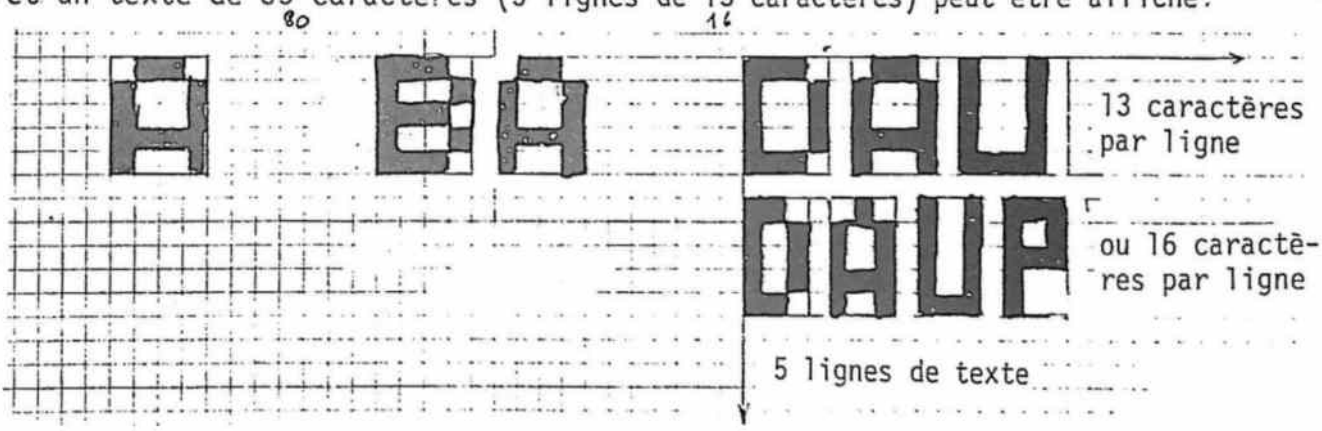
Implantation de la carte display graphique

UTILISATION DU DISPLAY GRAPHIQUE

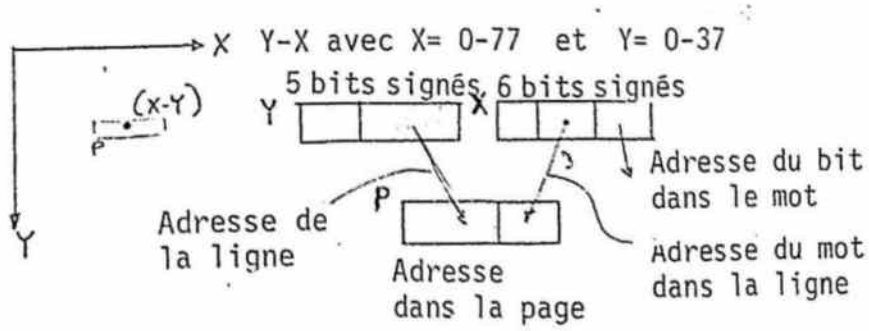
Le display se comporte comme une mémoire (adresse 1400) qui montre les groupes de 8 bits selon les segments horizontaux. Au total 64 points par ligne (8 mots) et 32 lignes de points.



Sur cet écran peuvent apparaître des figures simples, mobiles à une vitesse dépendant du temps de calcul des points suivants. Un générateur de caractères peut être mis dans la mémoire principale (utilise 160. mots pour 64. caractères 4x5) et un texte de 65 caractères (5 lignes de 13 caractères) peut être affiché.



En mode graphique, il y a avantage à faire correspondre à chaque point une adresse.



PROGRAMME

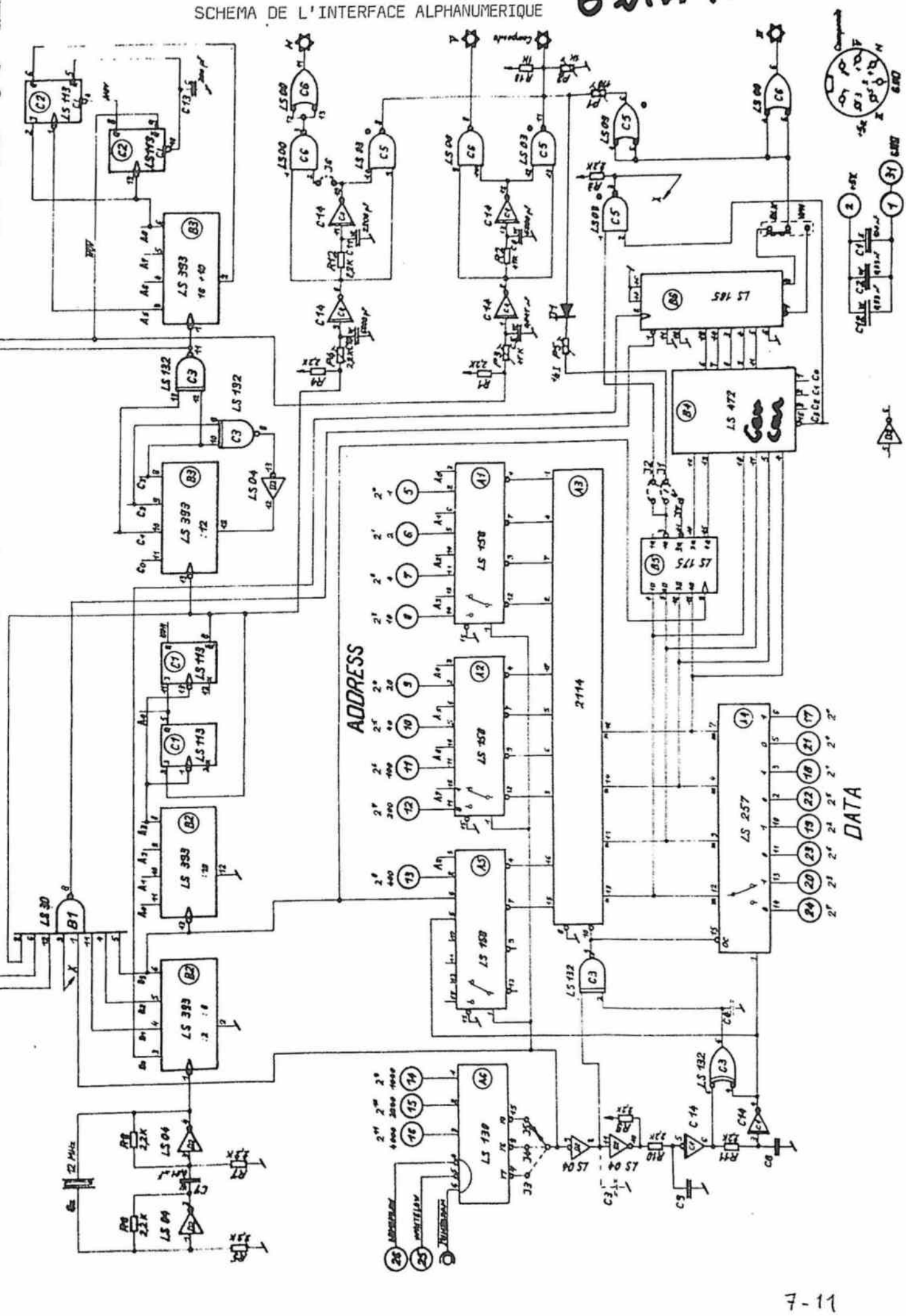
Le programme suivant permet de déplacer le spot (avec ou sans trace) selon les indications inscrites sur les touches.

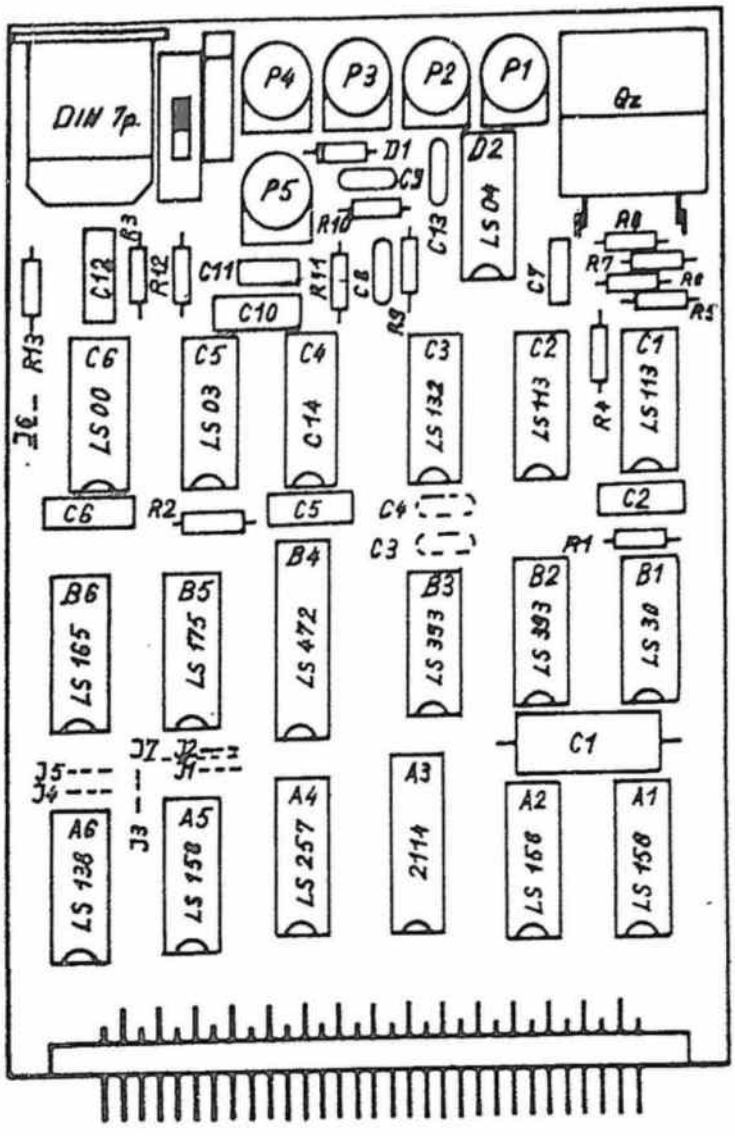
```
                                .TITLE  GRA80.SR
                                ;DIMITRIJEVIC 770518
                                ;JDN 770603/770614

                                .Z80

001000                          .LOC    1000
                     30         MFRAP = 30
                     20         MWRITE = 20
                     20         DEL = 20
                      7         CLA = 7
                      7         MDEPL = 7

                   1400         DISP = 1400
                   1360         STACK=  1360
                    370         MLIGNE = 370
                      7         MMOT = 7

001000 041 000 003      CLEAR:  LOAD    HL,#DISP
001003 066 000          CL2:    LOAD    (HL),#0
001005 054                      INC     L
001006 040 373                  JUMP,NE CL2
001010 001 020 040              LOAD    BC,#40*400+20
001013 061 360 002      DESSIN: LOAD    SP,#STACK
001016 315 127 002              CALL    XY
001021 333 007          DES2:   LOAD    A,SCLA
001023 267                      OR      A,A
001024 362 021 002              JUMP,SC DES2
001027 376 377                  COMP    A,#377
001031 050 345                  JUMP,EQ CLEAR
001033 137                      LOAD    E,A
001034 346 030                  AND     A,#MFRAP
001036 050 361                  JUMP,EQ DES2
001040 137                      LOAD    E,A
001041 346 020                  AND     A,#MWRITE
001043 302 052 002              JUMP,NE WRITE
001046 067                      SETC
001047 315 127 002              CALL    XY
001052 333 007          WRITE:  LOAD    A,SCLA
001054 346 007                  AND     A,#MDEPL
001056 137                      LOAD    E,A
001057 315 070 002              CALL    DEPLACE
001062 267                      OR      A,A
001063 315 127 002              CALL    XY
001066 030 331                  JUMP    DES2

001070 041 107 002      DEPLACE:LOAD    HL,#TABX
001073 031                      ADD     HL,DE
001074 176                      LOAD    A,(HL)
001075 200                      ADD     A,B
001076 107                      LOAD    B,A
001077 041 117 002              LOAD    HL,#TABY
001102 031                      ADD     HL,DE
001103 176                      LOAD    A,(HL)
001104 201                      ADD     A,C
001105 117                      LOAD    C,A
001106 311                      RET
```

```
001107 001 377 000 001  TABX:    .BYTE   1,377,0,1,377,377,0,1
001113 377 377 000 001

001117 000 001 001 001  TABY:    .BYTE   0,1,1,1,0,377,377,377
001123 000 377 377 377

001127 332 145 002      XY:      JUMP,CS EFFACE
001132 315 164 002               CALL    COORDON
001135 041 000 003               LOAD    HL,#DISP
001140 031                       ADD     HL,DE
001141 266                       OR      A,(HL)
001142 303 156 002               JUMP    CHARGE

001145 315 164 002      EFFACE:  CALL    COORDON
001150 057                       CPL     A
001151 041 000 003               LOAD    HL,#DISP
001154 031                       ADD     HL,DE
001155 246                       AND     A,(HL)
001156 041 000 003      CHARGE:  LOAD    HL,#DISP
001161 031                       ADD     HL,DE
001162 167                       LOAD    (HL),A
001163 311                       RET

001164 171              COORDON: LOAD    A,C
001165 007                       RL      A
001166 007                       RL      A
001167 007                       RL      A
001170 346 370                   AND     A,#MLIGNE
001172 137                       LOAD    E,A
001173 170                       LOAD    A,B
001174 017                       RR      A
001175 017                       RR      A
001176 017                       RR      A
001177 346 007                   AND     A,#MMOT
001201 203                       ADD     A,E
001202 137                       LOAD    E,A
001203 170                       LOAD    A,B
001204 346 007                   AND     A,#MMOT
001206 325                       PUSH    DE
001207 137                       LOAD    E,A
001210 041 217 002               LOAD    HL,#TABLE
001213 031                       ADD     HL,DE
001214 176                       LOAD    A,(HL)
001215 321                       POP     DE
001216 311                       RET

001217 200 100 040 020  TABLE:   .BYTE   200,100,40,20,10,4,2,1
001223 010 004 002 001

                1000     .END CLEAR
09/12/77  19.45.48               ASSEMBLAGE CORRECT.
```

SCHEMA DE L'INTERFACE ALPHANUMERIQUE

6 bit ASCII





R1 — 2,2K
R2 — 47K
R3 — 2,2K
R4 — 2,2K
R5 — 2,2K
R6 — 2,2K
R7 — 2,2K
R8 — 2,2K
R9 — 2,2K
R10 — 2,2K
R11 — 2,2K
R12 — 2,2K
R13 — 1K

C1 — 100 µF/E
C2 — 0,33 µF/W
C3 —
C4 —
C5 — 0,047 µF/W
C6 — 6800 PF/W
C7 — 0,01 µF/W
C8 —
C9 —
C10 — 6800 PF/W
C11 — 2200 PF/W
C12 — 0,33 µF/W
C13 200 pF/c

P1 — 470
P2 — 1K
P3 — 47K
P4 — 2,2K
P5 —

D1 —

IMPLANTATION DE L'INTERFACE ALPHANUMERIQUE

UTILISATION DU DISPLAY ALPHANUMERIQUE

La carte Display alphanumérique permet de visualiser sur un écran tous les caractères de la liste ASCII ci-après (p. 7-14).

Cette plaque possède sa propre mémoire de type 2114 (1024 x 4 bits), ce qui implique un multiplexage pour obtenir les 7 bits nécessaires au code ASCII.

Les adresses correspondantes se trouvent de Ø à 777. Il est ainsi possible d'écrire 512. caractères sur l'écran soit 16 lignes de 32 caractères chacune. Remarquons que ces adresses sont déjà utilisées par le programme Moniteur. Cette double présence de mémoire sur chaque adresse est possible du fait que le programme Moniteur est utilisé en lecture par le processeur, alors que la mémoire du display n'est employée qu'en écriture. Le signal WRITELOW permet de différencier ces deux blocs de mémoire. Notons encore que cet artifice ne permet pas de tester un caractère déjà présent sur l'écran.

D'autre part , il est possible de déplacer le display sur d'autres adresses à l'aide des jumpers J3 ou J4 soit dès l'adresse 6ØØØ respectivement 7ØØØ.

Les jumpers J1, 2 et 7 permettent de créer une demi-intensité sur l'écran ainsi que de souligner les caractères.

La demi-intensité est règlable par le potentiomètre P5.

Un DAUPHIN Z80, complété par un interface display alphanumérique et un clavier ASCII permet de travailler en BASIC (voir notice BASIC).

CODES ASCII

Parity	Octal	Character	Key (Teletype)		Parity	Octal	Character	Parity	Octal	Character	Parity	Octal	Character
0	000	NUL	CTRL↑ P	Null, tape feed	200	040	SP	200	100	@	0	140	`
200	001	SOH	CTRL A	Start of heading, (SOM, start of message)	0	041	!	0	101	A	200	141	a
200	002	STX	CTRL B	Start of text, (EOA, end of address)	0	042	"	0	102	B	200	142	b
0	003	ETX	CTRL C	End of text, (EOM, end of message)	200	043	#	200	103	C	0	143	c
200	004	EOT	CTRL D	End of transmission (end)	0	044	$	0	104	D	200	144	d
0	005	ENQ	CTRL E	Enquiry (ENQRY), (WRU, "who are you?")	200	045	%	200	105	E	0	145	e
0	006	ACK	CTRL F	Acknowledge, (RU, "Are you?")	200	046	&	200	106	F	0	146	f
200	007	BEL	CTRL G	Ring the bell	0	047	'	0	107	G	200	147	g
200	010	BS	CTRL H	Backspace, (FEO, format effector)	0	050	(	0	110	H	200	150	h
0	011	HT	CTRL I	Horizontal tab (TAB)	200	051	)	200	111	I	0	151	i
0	012	LF	CTRL J	Line feed or line space (LINE FEED)	200	052	*	200	112	J	0	152	j
200	013	VT	CTRL K	Vertical tab (VTAB)	0	053	+	0	113	K	200	153	k
0	014	FF	CTRL L	Form feed to top of next page (PAGE)	200	054	,	200	114	L	0	154	l
200	015	CR	CTRL M	Carriage return to beginning of line (CR)	0	055	-	0	115	M	200	155	m
200	016	SO	CTRL N	Shift out, changes ribbon color to red	0	056	.	0	116	N	200	156	n
0	017	SI	CTRL O	Shift in, changes ribbon color to black	200	057	/	200	117	O	0	157	o
200	020	DLE	CTRL P	Data link escape	0	060	0	0	120	P	200	160	p
0	021	DC1	CTRL Q	Device control 1 (X ON, reader on)	200	061	1	200	121	Q	0	161	q
0	022	DC2	CTRL R	Device control 2 (AUX ON, auxiliary on)	200	062	2	200	122	R	0	162	r
200	023	DC3	CTRL S	Device control 3 (X OFF, reader off)	0	063	3	0	123	S	200	163	s
0	024	DC4	CTRL T	Device control 4 (AUX OFF, auxiliary off)	200	064	4	200	124	T	0	164	t
200	025	NAK	CTRL U	Negative acknowledge, (ERR, error)	0	065	5	0	125	U	200	165	u
200	026	SYN	CTRL V	Synchronous idle (SYNC)	0	066	6	0	126	V	200	166	v
0	027	ETB	CTRL W	End of transmission block	200	067	7	200	127	W	0	167	w
0	030	CAN	CTRL X	Cancel (CANCL)	200	070	8	200	130	X	0	170	x
200	031	EM	CTRL Y	End of medium	0	071	9	0	131	Y	200	171	y
200	032	SUB	CTRL Z	Substitute	0	072	:	0	132	Z	200	172	z
0	033	ESC	CTRL↑ K	Escape, prefix	200	073	;	200	133	[	0	173	{
200	034	FS	CTRL↑ L	File separator	0	074	<	0	134	\	200	174	|
0	035	GS	CTRL↑ M	Group separator	200	075	=	200	135	]	0	175	}
0	036	RS	CTRL↑ N	Record separator	200	076	>	200	136	↑	0	176	~
200	037	US	CTRL↑ O	Unit separator	0	077	?	0	137	←	200	177	DEL

↑ = shift key

```
.TITLE LETR.CN

;       ECRIT L'ALPHABET SUR DAUPHIN

NBLETR  = 26.    ;NOMBRE DE LETTRES (EN DECIMAL)
ADECRAN = 0
FINECRAN= ADECRAN+512.
POS     = ADECRAN+100   ;ADRESSE DU DEBUT DE LA 4EME LIGNE
CODE    = 'A    ;CODE ASCII DE LA LETTRE A
SPACE   = 40

PROG    = 51000 ;DEBUT DU PROGRAMME

.LOC    PROG
INIT:   LOAD    HL,#ADECRAN
IN2:    LOAD    (HL),#SPACE
        INC     HL
        LOAD    A,H
        COMP    A,#FINECRAN/400
        JUMP,NE IN2
PREP:   LOAD    B,#NBLETR
        LOAD    HL,#POS
        LOAD    A,#CODE
TRACE:  LOAD    (HL),A
        INC     HL
        INC     A
        DECJ,NE B,TRACE
        JUMP    0               ;TRAP
```

Huitième partie: INTERFACE PARALLELE

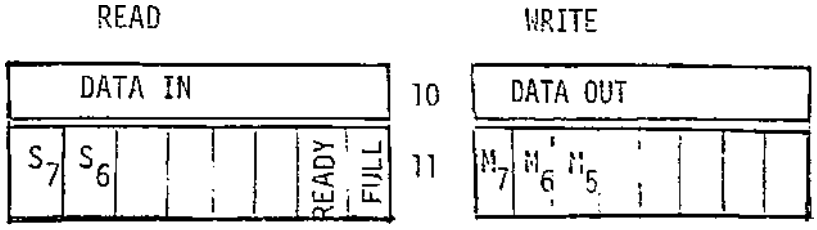
INTRODUCTION

L'interface parallèle est un interface très complet comportant les parties suivantes:

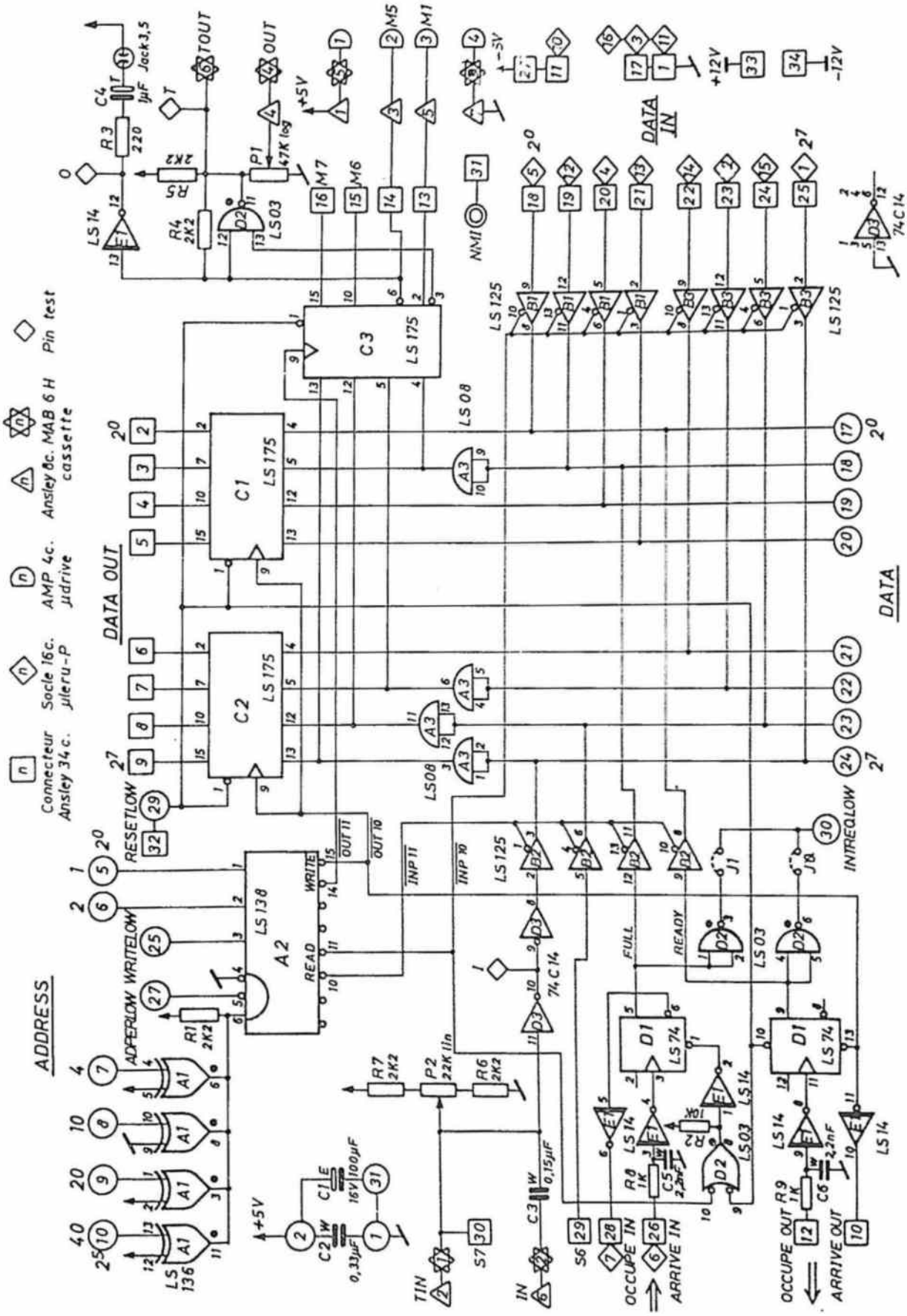
- 8 lignes d'entrée parallèles (TTL LS)
- 8 lignes de sortie parallèles
- 2 lignes ARRIVE/OCCUPE liées à un indicateur FULL et servant usuellement à synchroniser les transferts sur l'interface parallèle d'entrée.
- 2 lignes ARRIVE/OCCUPE liées à un indicateur READY et servant usuellement à synchroniser les transferts sur l'interface parallèle de sortie.
- 4 lignes de sortie supplémentaires, utilisées pour contrôler un interface cassettes SIMCA ou l'unité MICRODRIVE du MICROLERU.
- 2 lignes supplémentaires d'entrée, dont l'une est utilisée par l'interface cassettes SIMCA.

L'interface parallèle est compatible avec l'interface série DAUPHIN utilisant le circuit 8251. Tous les programmes de lecture/perforation de bandes (interface SIMSER) et de lecture/écriture cassettes (SIMCA) tournent sur l'interface parallèle à condition de remplacer le MICROLERU série par un MICROLERU parallèle. et le perforateur SIMSER par un perforateur SIMPA.

L'adresse de l'interface est 10/11, comme pour l'interface série.
Cette adresse peut être facilement changée contre une adresse quelconque.
Il faut naturellement la changer si l'interface parallèle est rajouté en plus de l'interface série.



SCHEMA DE L'INTERFACE PARALLELE



EXPLICATION DU SCHEMA

Le schéma est très simple puisqu'il ne comporte que des regsitres et portes à trois états pour tranférer l'information avec le processeur. La charge sur le bus wst une charge TTL LS, sauf pour la ligne RESETLOW (5 charges).

Le RESET met à zéro tous les registres, et initialise FULL = Ø et READY = 1.

FULL passe à 1 suite à un flanc descendant de ARRIVE 1, et est remis à zéro toutes les fois toutes les fois que le mot de 8 bits sur l'interface parallèle est lu.

READY passe à zéro toutes les fois qu'un mot est transféré sur le registre parallèle, et READY repasse à 1 au flanc descendant de OCCUPEØ.

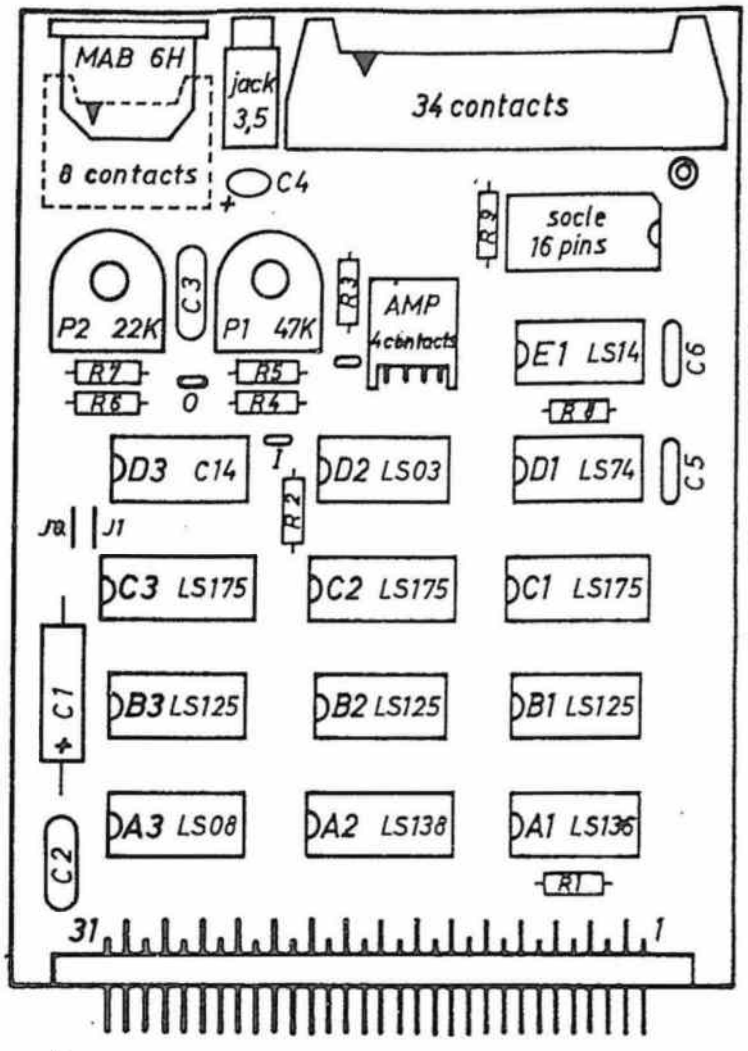
L'interface cassette est identique à ce que l'on trouve sur l'interface série. Vois la documentation correspondante.

T E S T

Pour le test, vérifier que les 4 impulsions de sélection des 4 périphériques constituant l'interface parallèle arrivent sur les circuits correspondants lorsqu'on fait avec le panneau de test ou le moniteur des INP 10 (lecture périphérique 10), INP 11, OUT 10, OUT 11.

Réaliser un câble de test reliant l'entrée parallèle sur la sortie parallèle et liant ARRIVEØ à ARRIVE 1 et OCCUPE Ø à OCCUPE 1, M6 à S6 et M7 à S7.

Vérifier que l'information sortie en 10 ou 11 se relit à la même adresse.



R1 = 2K2
R2 = 10K
R3 = 220
R4 = 2K2
R5 = 2K2
R6 = 2K2
R7 = 2K2
R8-R9 = 1K

C1 = 100 µF
C2 = 0,33µF
C3 = 0,15µF
C4 = 1µF
C5-C6 = 2,2 nF

Implantation de l'interface parallèle

Neuvième partie: INTERFACE CASSETTES

Sauver des programmes sur un simple magnétophone à cassettes est non seulement efficace, mais indispensable pour pouvoir écrire des programmes toujours plus longs. Un mode d'enregistrement très redondant est nécessaire à cause de la faible qualité "digitale" des magnétophones et bandes audio.

Le système utilisé dans le DAUPHIN est appelé SIMCA;il est simple et a fait ses preuves. Le signal enregistré est représenté sur la figure 1: trois impulsions à 2 kHz sont enregistrées pour un "un", six pour un "zéro", et un espace équivalent à 4 impulsions sépare les groupes d'impulsions. Le temps d'enregistrement d'un "un" et d'un "zéro" n'est donc pas le même et l'on peut compter sur une vitesse moyenne de l'ordre de 300 bits par seconde.

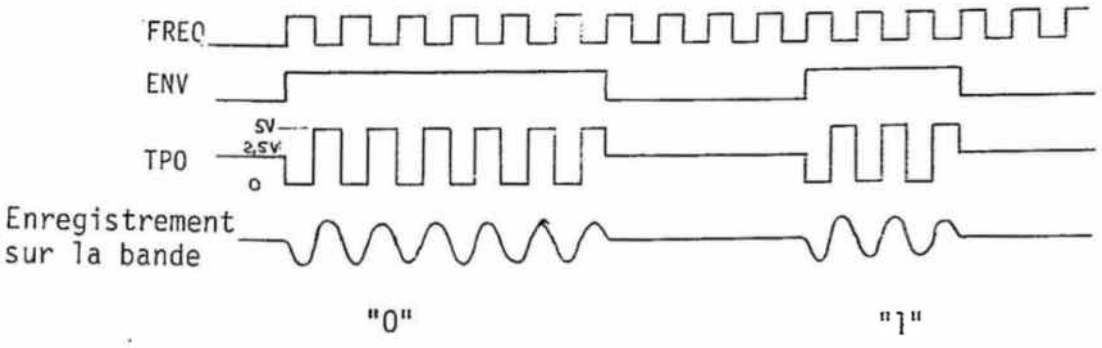


Fig. 1. Forme d'onde pour l'enregistrement d'un "0" et d'un "un" avec le système SIMCA.

A la lecture, le processeur compte les impulsions et considère que le groupe d'impulsions est terminé lorsqu'il n'y a pas eu d'impulsion pendant 1 ms. Ce principe admet de grandes variations de vitesse (± 40%) et tolère, dans une certaine mesure, des défauts d'enregistrement (impulsions manquantes, parasites) par simple vérification du nombre d'impulsions dans chaque groupe: une impulsion isolée est ignorée, 2 à 4 impulsions sont considérées comme un "un", 5 à 8 impulsions comme un "zéro".

INTERFACE DE SORTIE

Deux lignes de sortie sont nécessaires pour générer le signal TPO en créneau symétrique de la figure 1. La ligne FREQ génère un signal de fréquence 2 kHz et la ligne ENV définit l'enveloppe des groupes d'impulsions. Ces deux lignes sont prises sur un interface parallèle (PIA, 8255), série (8251) ou sur un interface spécialement construit avec un décodeur d'adresses et deux bascules. La figure 2 donne un schéma possible et les signaux correspondants. On remarque l'utilisation astucieuse d'une porte NAND en collecteur ouvert pour générer un signal à trois niveaux, qui est ensuite atténué avant d'être envoyé sur l'entrée auxiliaire (ou éventuellement l'entrée micro) du magnétophone. Un haut-parleur branché sur la ligne ENV est très utile pour vérifier que l'enregistrement se déroule correctement.

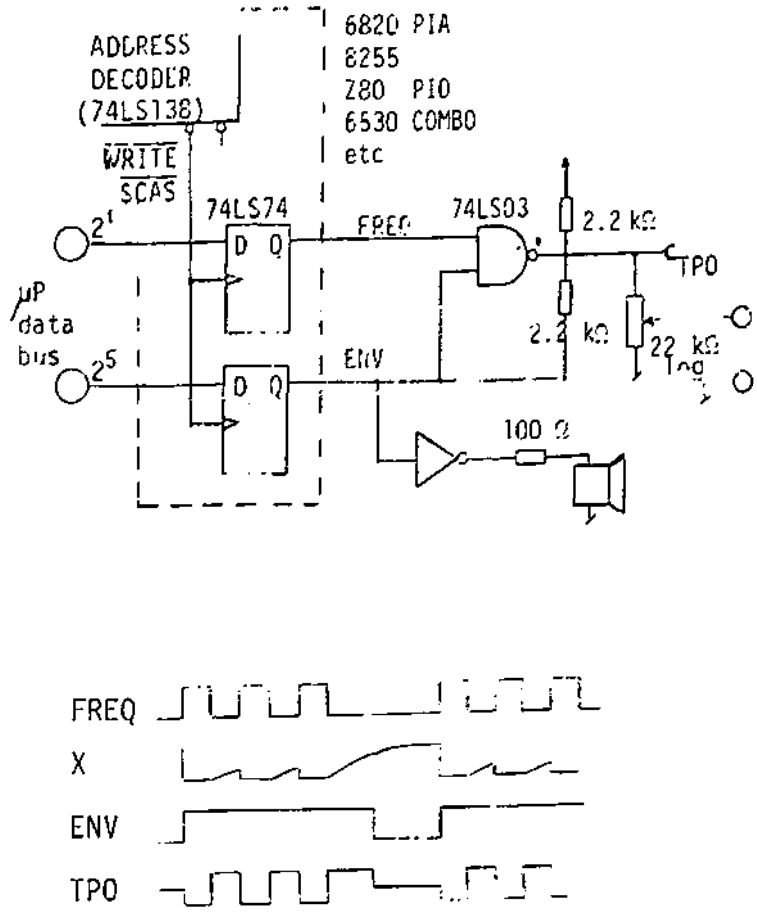


Fig. 2. Schéma pour la génération des signaux de la Fig. 1 à partir de deux lignes de sortie d'un interface.

PROGRAMME D'ECRITURE

Le programme qui génère le train d'impulsions est constitué d'une routine PULSE (write bit) qui écrit le groupe d'impulsions pour un bit ("0" ou "1"). Cette routine est appelée par la routine WRBYTE qui génère les impulsions correspondant à un mot de 8 bits et le programme principal envoie successivement sur la cassette les bytes d'une zone mémoire, selon un certain format. Tout enregistrement doit commencer par des mots de synchronisation, suivis de l'adresse et de la longueur de la zone mémoire. L'enregistrement se termine par une somme de contrôle ("checksum") et plusieurs enregistrements peuvent se suivre, le dernier contenant une adresse optionnelle d'exécution du programme.

Sur le DAUPHIN, le format a dû être simplifié pour raccourcir le programme. Un seul groupe de 256. bytes est sauvé d'un coup et les paramètres liés à ce transfert (adresse du bloc, adresse du programme à exécuter après lecture) doivent être préparés en mémoire avec le moniteur préalablement à l'exécution du programme d'écriture.

Pour une compréhension plus complète, on pourra se référer au listing donné en annexe.

INTERFACE D'ENTREE

Le signal amplifié de la bande est disponible sur la sortie "écouteur" de l'amplificateur. En général elle a une amplitude de plus de 3V et est directement utilisable avec le schéma de la figure 3.

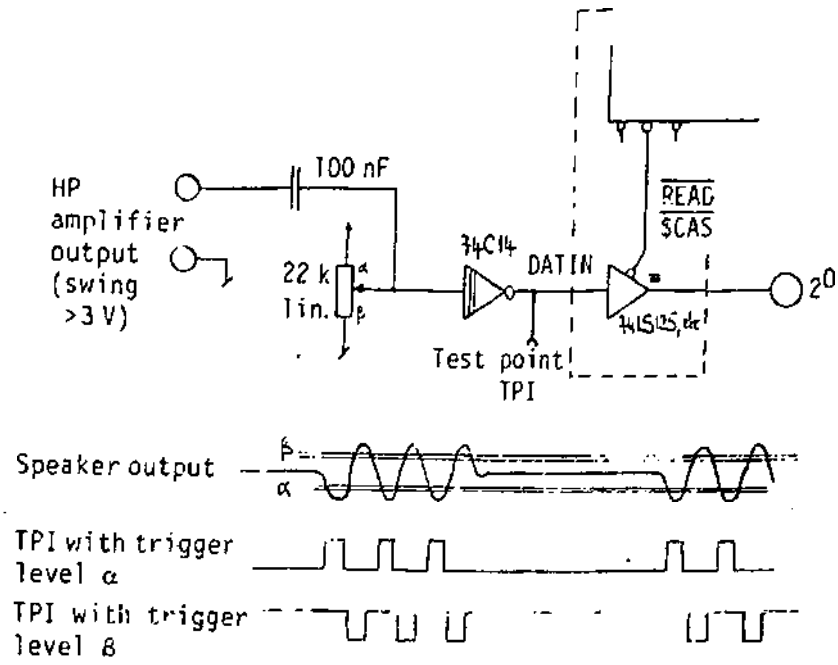


Fig. 3. Schéma pour la lecture des signaux amplifiés et leur transfert dans le microprocesseur.

Le potentiomètre permet de déplacer le niveau de basculement du Schmitt-trigger par rapport au signal de sortie. Selon le magnétophone, les impulsions positives ou négatives peuvent s'avérer plus favorables. Le 74C14 a deux seuils de basculement séparés par 1,5 à 2 V, et nécessite donc une amplitude importante du signal de sortie. Les circuits 4093, 14584, et éventuellement 74LS14 peuvent parfois mieux convenir (potentiomètre de 2,2 kOhm avec le 74LS14).

Le signal rectangulaire de sortie du Schmitt-trigger est envoyé directement à un interface ou sur une ligne d'entrée du microprocesseur (charge de 0,2 TTL maximum).

PROGRAMME DE LECTURE

La routine GETBIT de plus bas niveau attend une première transition de la sortie du Schmitt-trigger. Lorsqu'elle arrive, elle est comptée, et une boucle d'attente de 1 ms est initialisée. Chaque nouvelle transition est comptée et réinitialise la boucle d'attente (implémentation software d'un monostable retriggerable). S'il n'y a pas de transition pendant 1 ms, la routine vérifie le nombre d'impulsions reçues, signale une erreur (clignotement de l'affichage) si le nombre d'impulsions est faux, et place le carry à "1" ou "0" selon que le nombre d'impulsions correspond à un "1" ou à un "0". Le programme génère sur la sortie ENV qui controle le haut-parleur un signal qui permet un controle auditif simple.

Au début de la lecture d'un enregistrement, le programme commence par lire chaque bit et vérifie si les 8 derniers bits lus sont identiques au mot de synchronisation 00010110. Si oui, la routine GETBYTE est appelée et prend chaque fois 8 bits. Tous les caractères de synchronisation suivants sont ignorés, jusqu'à la lecture d'un caractère nul qui indique le début de l'enregistrement.

Le message proprement dit commence alors, et à partir de l'avant-dernier caractère de synchronisation, une resynchronisation en cas d'erreur n'est plus possible. La somme de contrôle en fin de bloc permet alors de détecter la plupart des erreurs. L'organigramme de ce programme est représenté à la figure 4. Après avoir vérifié que la somme de contrôle est correcte, le programme saute à l'adresse spécifiée lors de l'enregistrement.

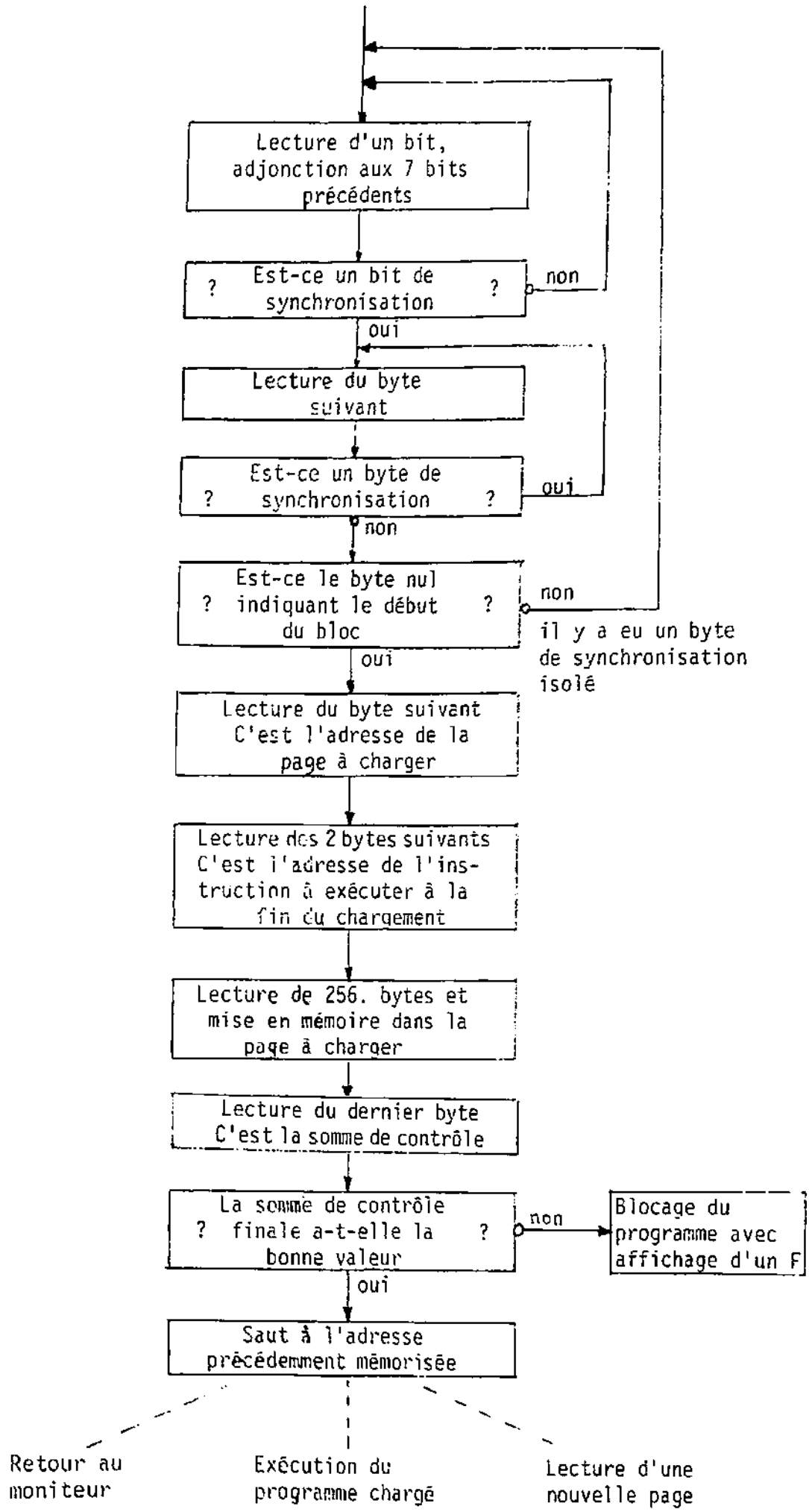
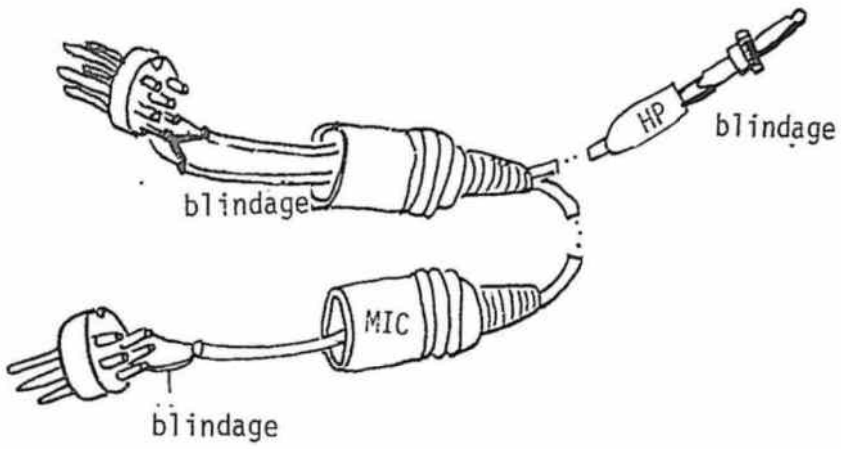


Fig. 4. Organigramme général de la lecture de l'information.

MATERIEL NECESSAIRE

- 1 DAUPHIN avec interface série ou parallèle
- un câble rallonge (dessin ci-dessous)
- 1 ROM 74S471 (MOZ81CU) ou l'extension moniteur sur EPROM 2708.
- 1 magnétophone à cassette avec une entrée (MICRO,...) et une sortie haut-parleur. Dans le cas d'un "tape-deck" ne possédant pas d'ampli, la sortie "casque" peut convenir si l'amplitude de la tension atteint 3V.
  Le contrôle d'enregistrement (ALC) n'est pas une aide, et s'il peut être debranché, il faut le faire.

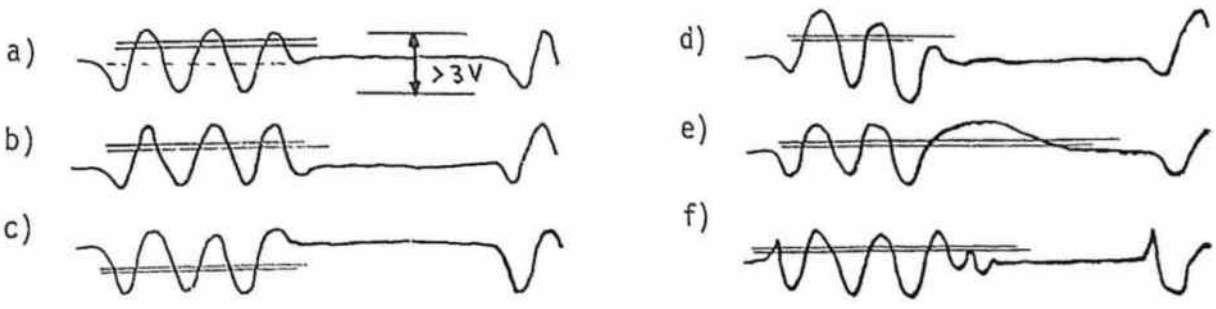


REGLAGES

Un oscilloscope n'est pas nécessaire pour le réglage et le dépannage initial, mais il est toutefois recommandé. Il est facile de régler convenablement le système pour un magnétophone donné. Le réglage permettant un transfert de cassettes d'un magnétophone à un autre est plus délicat et doit commencer par un réglage de la tête.

Le premier réglage concerne l'amplitude du signal enregistré. Il faut environ 100 mV pour une entrée auxiliaire, et 10 mV pour une entrée micro. Si le magnétophone a un contrôle automatique de gain, cette valeur est peu critique. Il est préférable qu'il n'en ait pas, et le réglage peut se faire d'après l'aiguille d'enregistrement, qui doit indiquer une légère saturation. Le programme de test permet de générer une suite de 0 et de 1.

La lecture montre, selon l'amplitude de signal d'entrée et la structure de l'amplificateur, des signaux qui peuvent avoir différentes formes.



La forme idéale est évidemment celle de la figure a).
b) et c) conviennent bien, mais dans les cas d), e) et f), il faut chercher à améliorer les réglages, et s'il n'est pas possible de les améliorer, changer de magnétophone.

MANIPULATION POUR REGLAGE

Préparer le DAUPHIN et le magnétophone avec les fils de raccordement.
Connecter le haut-parleur du DAUPHIN au jack de la plaque interface.

Mettre les potentiomètres de l'interface cassettes en position médiane.
Mettre en route le magnétophone (PLAY + RECORD) et la routine d'écriture du motif de test du DAUPHIN (G+5). On enregistre ainsi une suite de 1 et de 0.
Régler le niveau du magnétophone (s'il possède un VU METER, l'aiguille doit se trouver juste au début de la saturation). Le haut-parleur de la carte USART 8251 fait écho de l'enregistrement. Pour l'arrêter, faire un RESET.

Revenir au début de la bande. Mettre le haut-parleur du DAUPHIN sur la plaque clavier. Débrancher le câble MICRO du magnétophone.
Presser sur PLAY et sur les touches G+4 du DAUPHIN.
Le réglage s'effectue maintenant avec le potentiomètre 22k linéaire.

En le tournant on devrait avoir successivement les états suivants:

- pas de son
- son intermittent, F clignotant sur l'affichage
- son normal, pas de F (1)
- son plus ou moins normal, F sur l'affichage (2)
- son normal, pas de F (3)
- son intermittent, F sur l'affichage
- pas de son

L'état (2) correspond à un seuil proche de la tension de repos de la cassette. Il n'est pas toujours bien marqué. Choisir entre l'état (1) et l'état (3) celui dont le réglage est le moins critique et se placer au milieu de la fourchette.

Si l'on n'arrive pas à avoir un son normal sans affichage de F, recommencer l'enregistrement du motif de test en mettant le potentiomètre réglant le niveau de sortie de l'enregistreur dans une autre position.

Si l'on n'arrive toujours pas à avoir un son normal sans affichage de F, recommencer l'enregistrement du motif de test en mettant le potentiomètre 47K de l'interface cassettes dans une autre position.

Faire un essai de longue durée (10 minutes). Aucun F ne doit apparaître.

COMMENT SAUVER UN PROGRAMME

L'ordre d'enregistrement doit contenir les indications suivantes:

- l'adresse de début du programme à sauver
- sa longueur (400 au maximum pour une "tranche")
- une adresse à laquelle le processeur saute lorsque la lecture est finie.

Taper:- L'adresse de début si le programme doit s'exécuter immédiatement

- Ø s'il faut retourner au moniteur après lecture
- L'adresse du programme de lecture (c'est-à-dire 7400) si le programme continue à la page suivante

Brancher le haut-parleur sur la carte interface cassettes,
presser sur RECORD+PLAY (magnétophone), puis donner l'ordre suivant au DAUPHIN:

déb (WRITE G+1) 400 (WRITE G+1) cont (WRITE G+1)

L'affichage s'éteint pendant une fraction de monute. Quand l'enregistrement est terminé, le bruit cesse et l'affichage indique de nouveau 0000.

COMMENT RELIRE UNE CASSSETTE

Pour relire un enregistrement, mettre le haut-parleur sur le clavier, ôter la rallonge "micro", s'assurer que la cassette est au début de l'enregistrement.

Presser sur la touche PLAY du magnétophone et sur les touches G+0 (READ) du DAUPHIN. L'affichage s'éteint. Lorsque le chargement s'effectue, on entend un bruit dans le haut-parleur. Sitôt que le bruit cesse, le chargement est terminé. Selon l'adresse "cont" que l'on avait donnée lors de l'enregistrement, le programme s'exécute tout-de-suite, le système redonne le contrôle au moniteur ou le système lit la suite de la cassette.

Dixième partie: INTERFACE SÉRIE

Le circuit 8251 est un interface programmable entre un bus de microprocesseur et une ligne série asynchrone ou synchrone.

La carte interface série du DAUPHIN comporte un circuit 8251 répondant aux adresses 10/11 avec les inverseurs nécessaires pour la liaison SIMSER. Des circuits additionnels sur les lignes de contrôle du 8251 permettent de relier un magnétophone à cassette avec enregistrement de type SIMCA.

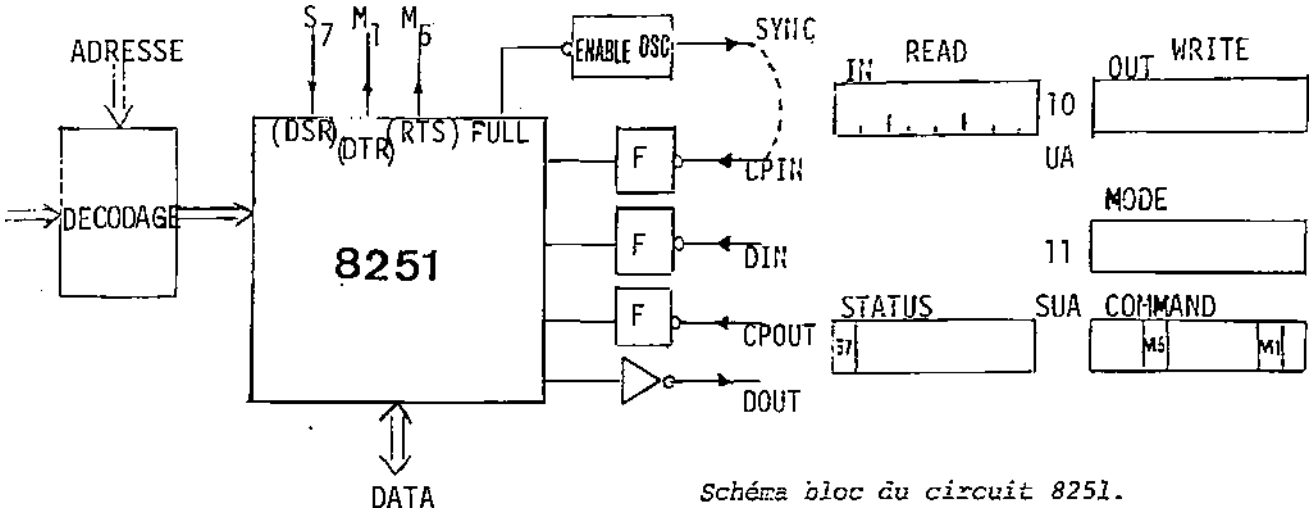


Schéma bloc du circuit 8251.

L'initialisation du circuit 8251 en mode asynchrone, après un RESET hardware consiste à envoyer successivement un mot de "MODE", puis un mot de "COMMAND" sur le périphérique d'adresse 11 (status).

Le mode définit:

a) Le facteur de division de l'horloge

```
DIV16 = 2    ;mode usuel asynchrone
```

b) La longueur des caractères transmis

```
HUITBITS = 14  ;transmission 8 bits
```

c) La présence ou non d'un bit de parité et sa parité

```
PAREV = 60      ;parité paire (even)
NOPAR = 0
```

d) Le nombre de bits d'arrêt

```
ONESTOP = 100  ;un bit d'arrêt
TWOSTOP = 300  ;deux bits d'arrêt
```

Le mode SIMSER normal (compatible MICROLERU) est

```
MODES = DIV16 + HUITBITS + NOPAR + TWOSTOP = 316
```

Pour une transmission entre 2 systèmes, il est recommandé d'utiliser la parité:

```
MODEP = DIV16 + HUIBITS + PAREV + ONESTOP  = 176
```

Le mot de commande définit:

a) Si la transmission et la réception sont autorisées

```
ENABLE = 5      ;autorise les deux sens
```

b) La remise à zéro des sémaphores d'erreur

```
ERRESET = 20
```

c) La remise à zéro du circuit 8251 pour retourner dans le mode MODE

```
RESET = 100
```

d) Les lignes de contrôle directement accessibles

```
M1 = 2          ;contrôlent la cassette
M5 = 40
```

En règle générale, la commande d'initialisation est

```
COMMAND = ENABLE
```

Pour remettre à zéro une erreur reconnue, il faut sortir temporairement la commande ENABLE + ERRESET.

Les indicateurs d'état *status/* permettent de savoir:

a) Si le registre d'entrée est plein

```
FULL = 2
```

Cette bascule est remise à zéro par le RESET et lorsque le registre d'entrée est lu par le processeur.

b) si le registre de sortie est vide (la transmission d'un caractère s'est terminée), c'est-à-dire que le circuit 8251 est prêt à transmettre un caractère

```
READY = 1
```

Cette bascule est mise à 1 par le RESET et mise à zéro par l'écriture dans le registre de sortie du circuit 8251.

c) Les flags d'erreur

```
ERPA = 10       ;erreur de parité
EROV = 20       ;erreur d'"overflow"  (3 caractères reçus avant la lecture du
                ;premier)
ERFRA = 40      ;erreur de format (pas de stop bit)
```

Ces indicateurs sont remis à zéro par le RESET et par le bit ERRESET du mot de commande.

d) L'état de la ligne d'entrée directe

```
S7 = 200        ;utilisé par la cassette.
```

Les routines de dialogue en mode SIMSER sont les suivantes:

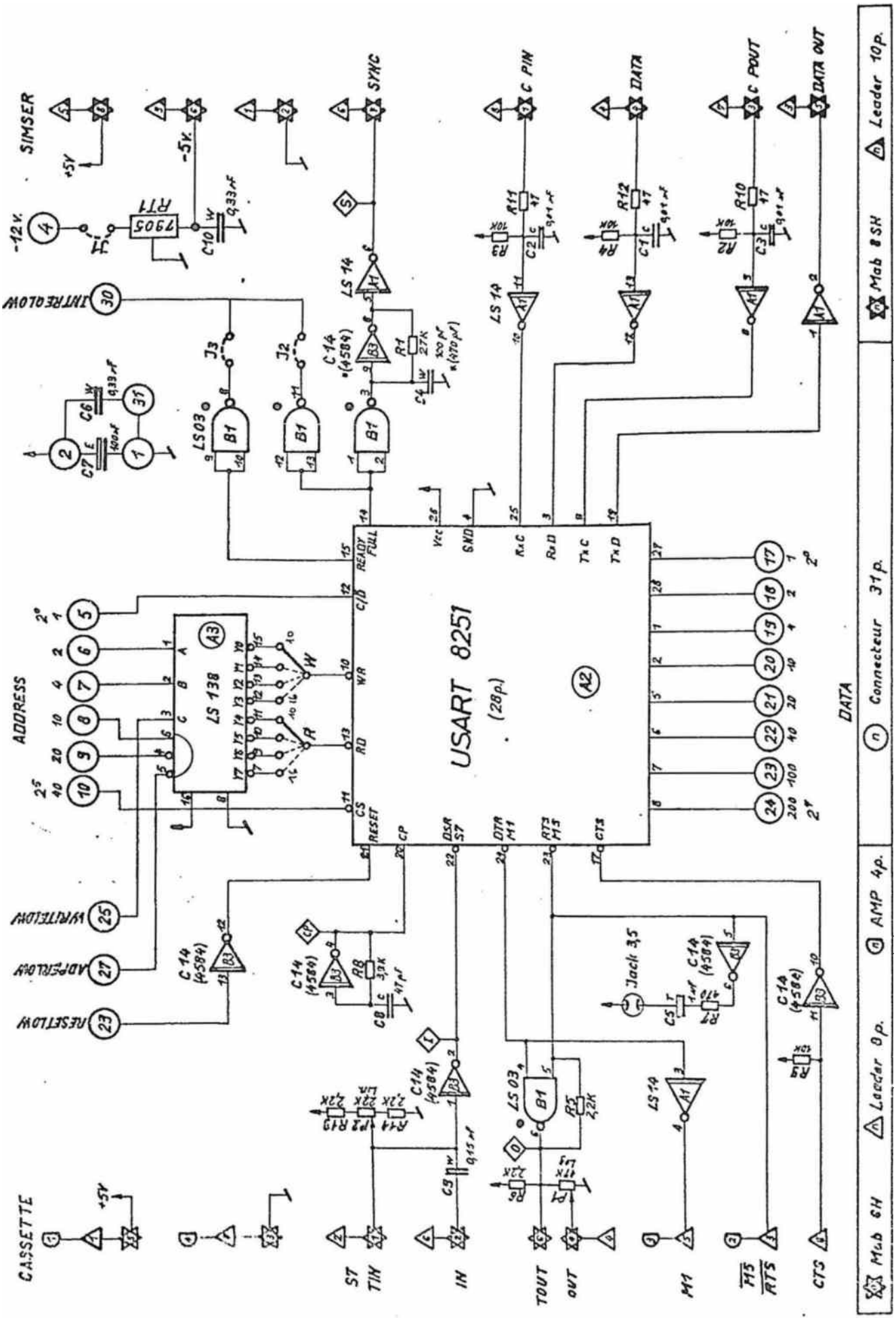
```
;UA=      10
;SUA=     11

;MODE=    316
;COMMAND=5

;FULL=    2
;READY=   1


INIT:     LOAD    A,# MODE      ;initialisation après un RESET
          LOAD    $SUA,A
          LOAD    A,# COMMAND
          LOAD    $SUA,A
          RET
GETCAR:   LOAD    A,$SUA        ;attend un caractère et revient avec
          AND     A,# FULL      ;le mot lu dans A et B
          JUMP,EQ GETCAR
          LOAD    A,$UA
          LOAD    B,A
          RET

WRCAR:    LOAD    A,$SUA        ;transmet le caractère dans B
          AND     A,# READY     ;dès que le caractère précédent
          JUMP,EQ WRCAR         ;a été transmis
          LOAD    A,B
          LOAD    $UA,A
          RET
```

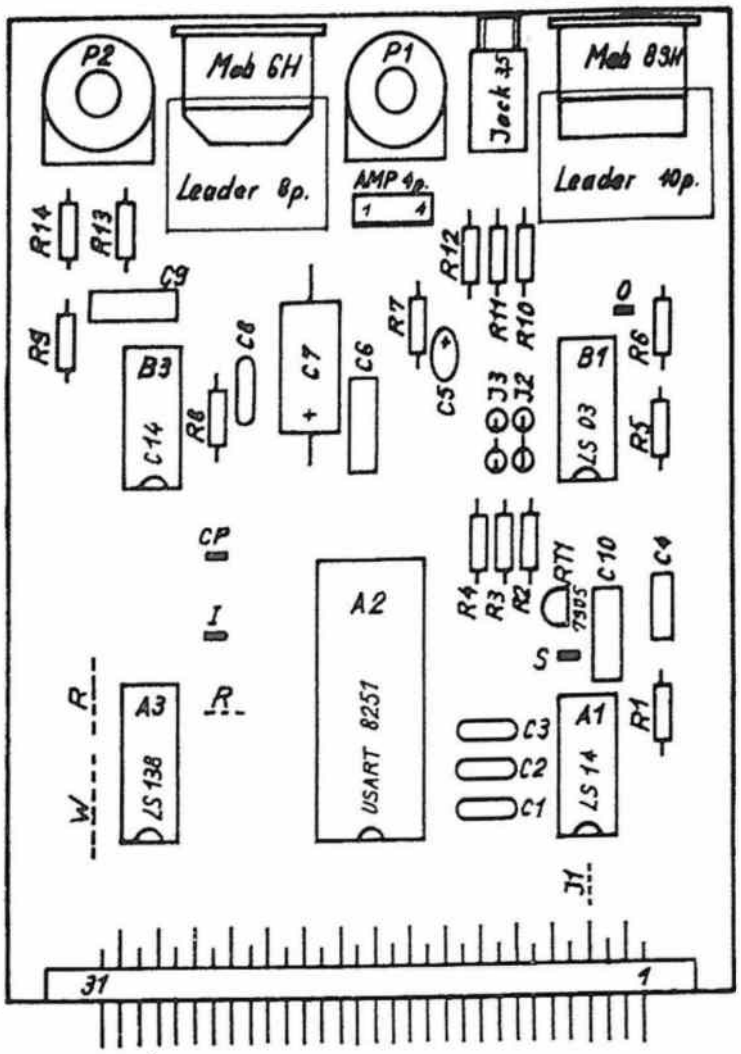
Un programme simple qui renvoie tous les caractères reçus s'écrit:

```
ECHO:     LOAD    SP,# STACK    ;initialisation
          CALL    INIT
ECH2:     CALL    GETCAR
          CALL    WRCAR
          JUMP    ECH2
```

Si l'on veut indiquer les erreurs de parité et d'overrun dans le Carry, avec code d'erreur dans A, la routine GETCAR devient:

```
GETCAR:   LOAD    A,$SUA
          TEST    A:BFULL       ;BFULL=1 car FULL=2=2^1
          JUMP,EQ GETCAR
          TEST    A:BERPA       ;BERPA = 3  (2^3 = 10)
          JUMP,NE ERRORPA
          TEST    A:BEROV       ;BEROV = 4  (2^4 = 20)
          JUMP,NE ERROROV
          LOAD    A,$UA
          LOAD    B,A
          OR      A,A           ;sortie avec CC
          RET

ERRORPA:  LOAD    A,# CERPA     ;code d'erreur de parité
          SETC
          RET

ERROROV:  LOAD    A,# CEROV     ;code d'erreur d'overrun
          SETC
          RET
```





R1 — 27 K
R2 — 10 K
R3 — 10K
R4 — 10K
R5 — 2,2 K
R6 — 2,2K
R7 — 470
R8 — 3,9K
R9 — 10K
R10 — 47
R11 — 47
R12 — 47
R13 — 2,2K
R14 — 2,2K

C1 — 0,01 µF/c
C2 — 0,01 µF/c
C3 — 0,01 µF/c
C4 — 100 pF/w
C5 — 1 µF/T
C6 — 0,33 µF/w
C7 — 100 µF/E
C8 — 47 pF/c
C9 — 0,15 µF/w
C10 — 0,33 µF/w

P1 — 47K Log
P2 — 22K Lin

Mob 6H ou Leader 8p.
Mob 8SH ou Leader 10p.

INDICATIONS POUR LE TEST ET LE DEPANNAGE

Pour le test de la transmission, relier sur le prise DIN 5 pôles les pins 7, 1, 3 et les pins 4, 5. Ceci a pour effet de lier la sortie de l'USART sur son entrée. Le moniteur peut vérifier que le dialogue se fait correctement.

```
RESET
316  OUT  11  OUT  }
  5  OUT  11  OUT  } initialisation

123  OUT  10  OUT

 10  INP  ------> 123

321  OUT  10  OUT

 10  INP  ------> 321

 11  OUT  10  OUT  }
 22  OUT  10  OUT  }
 10  INP  ------> 11  } double buffering
 10  INP  ------> 22  }

  1  OUT  10  OUT
  2  OUT  10  OUT
  3  OUT  10  OUT

 11  INP  ------> 25 ou 125    erreur d'overrun
```

On peut aussi vérifier que le bit de lecture de la cassette charge en fonction du réglage du potentiomètre de réglage d'entrée.

```
 11  INP  ------> 5 ou 105     selon la position
```

En cas de mauvais fonctionnement, laisser ou mettre le jumper ci-dessus et vérifier au crayon lumineux ou mieux à l'oscilloscope les impulsions sur la pin 20 (environ 1,5 MHz) et sur les pins 25 et 9 (environ 160 kHz). Vérifier que lors d'une sortie d'information (125 OUT 10 après initialisation), le crayon lumineux indique un transfert d'information sur les pins 3 et 19.

LECTURE DE BANDES AVEC MICROLERU

Brancher un MICROLERU-S sur l'interface série, ou un MICROLERU-P sur l'interface parallèle.

L'ordre LOAD (F+G+Ø) doit indiquer à quelle adresse le programme de la bande perforée doit être mis en mémoire (en général en 1000 ou 2000).

a (LOAD) charge une bande papier selon le format PDP11 à travers l'interface série ou parallèle. L'adresse de l'interface est 10 (data) et 11 (status et mode).

Comme contrôle de bon fonctionnement, la lampe du clavier clignote à chaque lecture du trou de synchronisation de la bande papier (si le haut-parleur est branché, un crépitement se fait entendre).

Les caractères significatifs lus sur la bande et transférés en mémoire sont affichés sur le premier affichage 7 segments du clavier. En cas d'erreur, un F apparaît sur le premier digit (erreur de format), ou un caractère quelconque apparaît sur le deuxième digit (erreur de check-sum). Il faut presser RESET et recommencer la manipulation.

N.B. Avec l'interface parallèle, l'ordre 10 INPUT permet de vérifier le caractère lu depuis la bande papier.
Avec l'interface série, qui est programmable, il faut commencer par l'initialiser, ce qui nécessite après un RESET la séquence:

```
316 OUTPUT 11 OUTPUT   initialisation
  5 OUTPUT 11 OUTPUT
 10 INPUT              lit le dernier caractère reçu par
 10 INPUT              l'interface
```

PUNCH

Pour puncher un programme chargé dans le DAUPHIN au moyen d'un puncher, il faut donner l'ordre suivant:

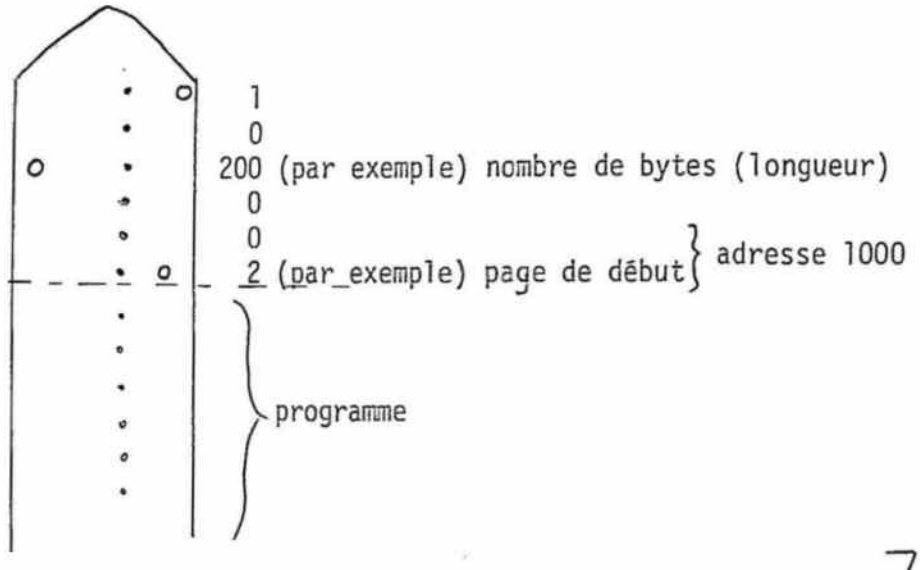
ADDRESS (PUNCH) LONG (PUNCH) START (PUNCH)

les trois nombres étant successivement l'adresse du début du programme la longueur du programme, et l'adresse de restart (0 si le contrôle est donné au moniteur).

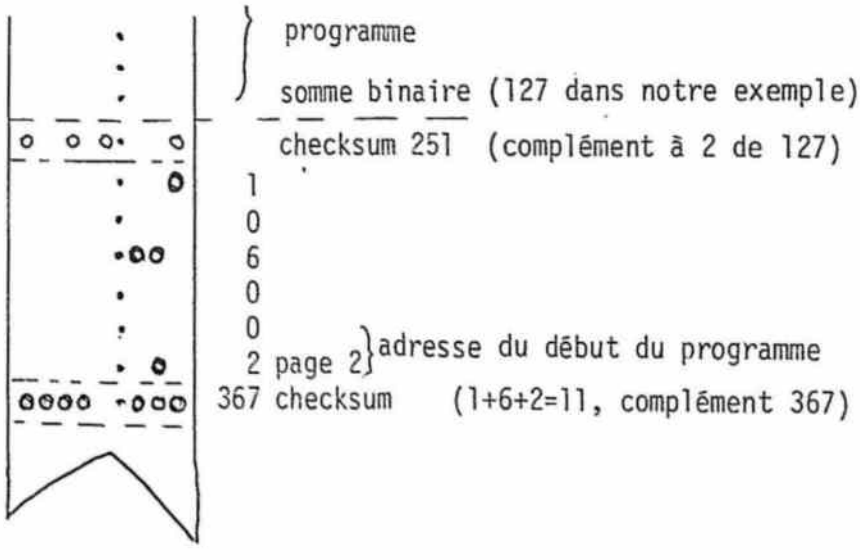
FORMAT PUNCH SIMPLIFIE

Il est possible de perforer des bandes papier au moyen de perforateur mécaniques très simples, en tenant compte des instructions qui suivent.

Avant de perforer le programme, il faut donner les codes suivants:



Au chargement, il y a une erreur de checksum. Noter le caractère affiché. et coder en octal sa valeur (127). Reprendre la bande pour puncher la checksum et le bloc de fin.



Le programme "PUNCH" fait des blocs plus petits (50 octal).
La longueur maximum d'un bloc est de 377.

EXTENSION MONITEUR MOZ86E

Cette EPROM 2708 utilisée avec les deux PROM MOZ8ØM et MOZ81R complète le moniteur du système DAUPHIN INDUSTRIE.

Les fonctions suivantes sont contenues dans cette extension moniteur:

- adressage RELATIF-ABSOLU
- programmateur de 74S470/471; 74S472/473; 74S188/288
- lecteur/perforateur de bandes papier, format PDP11
- lecture d'une bande Telex
- enregistrement sur cassette et lecture
- routines décimales

LISTE DES ORDRES ET TOUCHES CORRESPONDANTES

	a GO 4	DELETE 5	a INPUT 6	d OUTPUT 7
F	a OPEN Ø	d NEXT 1	d PREVIOUS 2	d CLOSE 3

G	TPLAY 4	TWRITE 5	 6	 7
	READ Ø	a WRITE 1	a RELATIVE 2	LABS 3

F+G	PROGRAM 4	GET 5	MOVE 6	CHECK 7
	a LOAD Ø	a PUNCH 1	LOTEL 2	(PUTEL) 3

Pour les ordres du moniteur de base, voir "Comprendre les microprocesseurs" Vol. I, page 4-6.

ORDRES DU MONITEUR MOZ86E

a (INPUT) lit le périphérique d'adresse a et affiche son contenu (précédé du signe égal) jusqu'à ce que l'on presse une nouvelle touche.

EXEMPLE: 7 INPUT lit le clavier à l'instant de l'exécution de l'ordre et affiche toujours 156. car l'utilisateur n'a pas eu le temps de relâcher la touche INPUT (code 10+6, plus les deux bits 100 et 40 qui sont laissés en "l'air" dans l'interface clavier).

d (OUTPUT) a (OUTPUT) écrit dans le périphérique d'adresse a la valeur d. Après avoir tapé la valeur à transférer et la première fois sur OUTPUT, on lit A000 sur l'affichage, qui attend l'adresse du périphérique.

Lorsqu'on a tapé l'adresse du périphérique et le deuxième OUTPUT (ou n'importe quelle combinaison de touches non numériques, l'ordre de sortie s'effectue et l'affichage montre 0000.

EXEMPLE: Ø OUTPUT 6 OUTPUT change l'état de la lampe.
6 INPUT aussi, d'ailleurs, car le périphérique 6 ne tient pas compte de la ligne WRITE (lecture/écriture).

a (REL) Calcule la différence entre l'adresse qui suit l'adresse pointée après un OPEN, NEXT ou PREVIOUS et l'adresse tapée.

EXEMPLE: 1020 OPEN montre le contenu de la position mémoire 1020. Après avoir tapé 1035 REL, on voit sur l'affichage 0014 (qui est la différence entre 1035 et 1020 + 1).
Si l'on veut introduire directement cette valeur dans la position 1020, il suffit de taper ensuite NEXT, PREVIOUS ou CLOSE.
Si la différence est supérieure à 177 en valeur absolue (ce qui correspond aux nombres 0 - 377 en complément à 2), le programme signale l'erreur en affichant $\overline{r}$ sur le display de gauche.
Les différences négatives apparaissent sous forme de complément, comme on doit les calculer pour l'adressage relatif.

ℓ (ABS) Convertit une adresse relative en adresse absolue, en additionnant la valeur ℓ à l'adresse qui suit l'adresse pointée.

Après un ordre OPEN, NEXT, PREVIOUS, fait le même calcul sur le contenu de l'adresse pointée.

PROGRAMMATION DES CIRCUITS 2708 (ou S471/472 288/188 sur une carte spéciale)

PROGRAM pour les 2708
Ø PROGRAM pour les 471/2
40 PROGRAM pour les 288

programme la ROM placée sur le socle de la carte programmateur correspondante selon les valeurs mémorisées en RAM (2000)

GET pour les 2708
Ø GET pour les 471/288

recopie en RAM (à partir de 2000) le contenu de la ROM située sur le socle de programmateur

dest MOVE long MOVE déb MOVE

permet de déplacer des parties de programmes d'une partie de la RAM dans une autre partie. Plus précisément, cet ordre recopie un programme de longueur "long", commençant en "déb" dans les positions mémoire "dest" et suivantes.

déb CHECK long CHECK Ø CHECK
(F+G+7)

calcule la "checksum" (somme des nombres binaires) des contenus des positions mémoire à partir de "déb". "long" indique le nombre de positions à additionner.

Utile par exemple, pour vérifier qu'une mémoire est programmée correctement.

LECTURE DE BANDES (par l'interface série ou parallèle) ET PERFORATION

a LOAD charge une bande papier selon le format PDP11 à travers l'interface à partir de l'adresse a.
(Pour plus de détails, voir "Comprendre les microprocesseurs" Volume II, Extensions et interfaces).

déb PUNCH long PUNCH start PUNCH perfore une bande papier (format PDP11). Les trois nombres qu'il faut indiquer sont successivement: l'adresse du début du programme, la longueur, et l'adresse de restart (0 si le contrôle est donnée au moniteur, l'adresse correspondant à GO si on veut un start automatique du programme).

LOTEL lit une bande papier dans le format Telex.

LECTURE DE CASSETTES ET ENREGISTREMENT (UTILISE UN INTERFACE SERIE OU //)

(TWRITE) enregistre une suite de "0" et de "1" (motif de test)

(TPLAY) relit le motif de test en comptant le nombre d'erreurs.

déb(WRITE) 400(WRITE)cont(WRITE) sauve une page mémoire sur cassettes

déb: adresse du début de la zone sauvée
400: longueur de la zone sauvée (peut en fait être quelconque)
cont: adresse indiquant au système ce qu'il doit faire sitôt la lecture achevée.

Taper - l'adresse de début si le programme doit s'exécuter
- Ø s'il faut retourner au moniteur après chargement
- l'adresse du programme de lecture (c'est à dire 7400 avec MOZ86E) si le programme continue à la page suivante

EXEMPLE: le programme à sauver occupe les positions 2250 à 2600. Son adresse de GO est en 2300, et l'on désire que le programme s'exécute sitôt chargé.

```
2000 WRITE 400 WRITE 7400 WRITE  sauve la première page
(2250 WRITE 400 WRITE 7400 WRITE  est aussi correct)
2400 WRITE 400 WRITE 2300 WRITE  sauve la deuxième page
```

(READ) lit une cassette

CALCULATRICE DECIMALE POUR L'ADDITION ET LA SOUSTRACTION

Une calculatrice octale pour l'addition et la soustraction utilisant la routine d'affichage du moniteur s'écrit en quelques instructions.

Il est beaucoup plus long d'effectuer et d'afficher les résultats des opérations dans le système décimal.

Le moniteur MOZ86E contient quelques routines qui permettent de faire une calculatrice décimale pour l'addition et la soustraction avec visualisation sur les affichages à 7 segments, ainsi que des compteurs/décompteurs décimaux.

Pour utiliser le programme calculatrice, faire 7034 GO.
Taper le premier nombre d'une suite d'additions et de soustractions, puis G+4 (+) s'il faut ajouter ou G+5 (-) s'il faut soustraire, et ainsi de suite. Le premier nombre doit être positif, il faut donc commencer par zéro, si l'on désire soustraire le premier nombre.

Ce programme comprend plusieurs routines qui peuvent être réutilisées dans d'autres programmes avec des opérations décimales. Ces routines sont:

AFB (7110) affiche les deux digits dans B aux adresses (C) et (C+1). Initialiser C à la valeur 0, 1 ou 2 pour avoir quelquechose de visible.
Cette routine modifie les registres A,B,C,DE

AFHL (7125) affiche les 4 digits contenus dans HL (nombres BCD)
Modifie A et DE.

INDCØ (7145) attend un nombre BCD du clavier (pour 8 et 9 taper respectivement 10+Ø et 1Ø+1) et l'affiche jusqu'à ce qu'une touche FONCTION soit pressée. Le retour de la routine s'effectue alors avec dans HL, le nombre tapé, dans B et A, le résultat et dans C, le nombre + 200.
Modifie: A,B,C,DE,HL

DINCX (7215) incrémente le contenu de la position mémoire dont l'adresse est contenue dans HL (en décimal).
Modifie A.

DADDHLDE (7233) additionne les contenus de HL et DE (en décimal).
Modifie: A,HL.

DADDHH (7244) double le contenu de HL (en décimal)
Modifie: A,HL.

DSUBHL (7255) ôte au contenu de HL le contenu de DE.
Modifie A,HL.

COMPHL (7266) compare les contenus de HL et de DE.
Modifie F,A.

DINCHL (7274) incrémente le contenu de HL (en décimal).
Modifie A,HL.

LISTING DU MONITEUR MOZ86E

```
                .TITLE MOZ86E.SR        ;JOH:DZILOG

                ;770127...;JOH 770430

                ;EXTENSION FOR MONITOR FOR Z80 ON  D A U P H I N

                .Z80

000000          .LOC    6000

                ;Peripheral and parameter definitions
      0   DIG0=   0
      1   DIG1=   1
      2   DIG2=   2
      3   DIG3=   3
      4   SEL4=   4
      5   SEL5=   5
      6   HP=     6
      7   CLA=    7

    200   FULL=   200
     37   MCLA=   37
      7   MDIG=   7
     10   FKEY=   10
     20   GKEY=   20

     77   ZERO=   77
      6   UN=     6
    133   DEUX=   133
    117   TROIS=  117
    146   QUATRE= 146
    155   CINQ=   155
    175   SIX=    175
      7   SEPT=   7
    110   EGAL=   110
    100   MOINS=  100
    167   LETA=   167
    174   LETB=   174
    130   LETC=   130
    136   LETD=   136
    171   LETE=   171
    161   LETF=   161
     70   LETL=   70
    134   LETO=   134
    163   LETP=   163
    200   DOT=    200
    323   QUES=   323

      0   ROM0=   0
    400   ROM1=   400
   1000   RAM=    1000
   2000   RAMEX=  2000
   6000   ROM6=   6000

   1361   SIMU=   RAM+361
   1364   SAUL=   RAM+364
   1376   SAUOPEN=RAM+376
   1374   ADGO=   RAM+374

     45   INOC00= 45
     50   INOC0=  50
     55   INOC=   55
     12   FONC=   12

                ;               F-6     INPUT
                ;               F-7     OUTPUT

                ;Orders:        G-0     READ    cassette
                ;               G-1     WRITE   according to AAA,...
                ;               G-2     REL     relative address
                ;               G-3     ABS     absolute address

                ;               G-4     TPLA    test play on cassette
                ;               G-5     TWRITE  write pattern
                ;               G-6     RLOAD

                ;               FG-1    PUNCH   addrPUNCHlengthPUNCHstartaddress
                ;               FG-2    LOTEL

                ;               FG-4    P471    Program 471
                ;               FG-5    GET     Move 471 content to RAM
                ;               FG-7    CHECKTOT addrCHECKlengthCHECKanyCHECK

                ;===    DEFINITION:

   1400   PROM=   1400
   2000   LROM=   2000
    400   L471=   400

     26   SYN=    26
     40   SPARI=  40
      4   ERRO=   4
     17   PARO=   17
     11   ZERO=   11
    100   SYNL=   100
     10   BYL=    8.
     20   ENV=    20
     25   CDO=    ENV+5
     33   CDZ=    ENV+11.
     10   CDF=    ENV-8.
     36   PWDUR=  36
      1   FREQ=   1

     10   PR=     10      ;DATA INPUT
     11   SPR=    PR+1    ;STATUS INPUT
     10   PP=     PR
     11   SPP=    SPR
    310    MODE=  310     ;(8+4,no parity,x16)
      5     ENABLE=       5       ;in and out
     40     RTS=  40
      2     DTR=  2
      5    COMMAND=ENABLE
    100    REMOTE= 100

      2   FULP=   2       ; FULP IS BIT 2?
      1   READY=  1
    200   DSR=    200
```

```
000000          .LOC ROM6

                ;+++    DEBUT DU PROGRAMME


                DEBUT:
000000 076 005          LOAD    A,#COMTRMD
000002 223 011          LOAD    $SPR,A
000004 076 100          LOAD    A,#REMODE
000006 323 011          LOAD    $SPR,A
000010 076 316          LOAD    A,#MODE
000012 323 011          LOAD    $SPR,A
000014 076 005          LOAD    A,#COMTRMD
000016 323 011          LOAD    $SPR,A


000020 170              LOAD    A,B
000021 346 037          AND     A,#MCLA
000023 326 016          SUB     A,#16
000025 007              RL      A
000026 021 043 014      LOAD    DE,#TOPDPE
000031 203              ADD     A,E
000032 137              LOAD    E,A
000033 032              LOAD    A,(DE)
000034 107              LOAD    B,A
000035 023              INC     DE
000036 032              LOAD    A,(DE)
000037 127              LOAD    D,A
000040 130              LOAD    E,B
000041 325              PUSH    DE
000042 311              RET

000043 247 015 206 015 TOPDPE: .WORD   FIN,FOUT
000047 000 017 075 017         .WORD   READ,WRITE,REL,ABS
000053 167 014 160 014
000057 223 017 353 017         .WORD   TPLA,TWRY,0,0
000063 000 000 000 000
000067 000 000 265 014         .WORD   0,PUNCH,LOTEL,0
000073 230 015 000 000
000077 035 015 146 015         .WORD   P471,GET,0,CHECKTOT
000103 000 000 102 015
```

RELATIF - ABSOLU

```
                ;RELATIVE and ABSOLUTE address calculation

                        117000  IMPOSSIBLE      =       117000

000107 333 133 376 002  REL:    LOAD    DE,SAUOPEN
000113 067              SETC
000114 333 122          SUBC    HL,DE
000116 332 136 014              JMP,LO REL1
000121 174              LOAD    A,H
000122 267              OR      A,A
000123 302 147 014              JMP,NE LERROR
000126 175              LOAD    A,L
000127 267              OR      A,A
000130 372 147 014              JMP,M1 LERROR
000133 303 152 014              JMP     MON1
000136 044              REL1:   INC     H
000137 302 147 014              JMP,NE LERROR
000142 175              LOAD    A,L
000143 267              OR      A,A
000144 372 152 014              JMP,M1 MON1
000147 041 000 236      LERROR: LOAD    HL,#IMPOSSIBLE
000152 267              MON1:   CALL    INOC0
000153 016 200          LOAD    C,#200
000155 303 012 000              JMP     FONC

000160 052 376 002      ABS:    LOAD    HL,SAUOPEN
000163 176              LOAD    A,(HL)
000164 267              OR      A,A
000165 026 000          LOAD    D,00
000167 362 174 014              JMP,PL ABS1
000172 026 377          LOAD    D,#377
000174 137              ABS1:   LOAD    E,A
000175 067              SETC
000176 355 132          ADDC    HL,DE
000200 030 250          JMP     MON1
```

PERFORATION

```
                        ;EEE.   PUNCH A TAPE IN PDP11 FORMAT ON A SYM1
                        ;
                        ;---    CALL    PERA    ;PUNCH BYTE IN A
                        ;---    CALL    PERDE   ;PUNCH WORD IN DE
                        ;                       ;COMPUTE CHECKSUM IN C
                        ;                       ;DO NOT CHANGE THE FLAGS
                        ;
006202 173              PERDE:  LOAD    A,E
006203 315 207 014              CALL    PERA
006206 172                      LOAD    A,D
                        ;
006207 365              PERA:   PUSH    AF
006210 201                      ADD     A,C
006211 117                      LOAD    C,A
006212 333 011          OUTPP:  LOAD    A,$SPP
006214 346 001                  AND     A,#READY
006216 050 372                  JUMP,EQ OUTPP
006220 361                      POP     AF
006221 323 010                  LOAD    $PP,A
006223 323 006                  LOAD    $HP,A
006225 311                      RET
                        ;
                        ;---    CALL    ANORCE  ;PUNCH A LEADER OF MAX 256 BYTES
                    200 LONGAM= 200
                      0 CARAM=  0               ;LEADER CHARACTER
                        ;
006226 076 000          ANORCE: LOAD    A,#CARAM
006230 006 200                  LOAD    B,#LONGAM
006232 315 207 014      AN02:   CALL    PERA
006235 005                      DEC     B
006236 040 372                  JUMP,NE AN02
006240 311                      RET
                        ;
                        ;---    CALL    CHECKSUM;COMPUTE AND PUNCH LAST CHECKSUM
                        ;
006241 171              CHECKSUM:LOAD   A,C     ;2'S COMPLEMENT OF C
006242 057                      CPL     A
006243 074                      INC     A
006244 315 207 014              CALL    PERA
006247 311                      RET
                        ;
                        ;---    CALL    GETARG  Get transfer arguments
                        ;
006250 345              GETARG: PUSH    HL
006251 041 000 030              LOAD    HL,#14003       ;L
006254 357                      CALL    INOCO
006255 345                      PUSH    HL
006256 041 000 026              LOAD    HL,#13000       ;A
006261 357                      CALL    INOCO
006262 301                      POP     BC
006263 321                      POP     DE
006264 311                      RET
                        ;
                     50 LONBLO= 50              ;STANDARD BLOCK LENGTH
                        ;
                        ;+++    Punch program
                        ;
                        PUNCH:
006265 315 250 014              CALL    GETARG
006270 042 374 002              LOAD    RDGO,HL
006273 305                      PUSH    BC
006274 341                      POP     HL
006275 325                      PUSH    DE
006276 315 226 014              CALL    ANORCE
                        ;
006301 016 000          BLOCK:  LOAD    C,#0    ;INIT CHECKSUM
006303 021 001 000              LOAD    DE,#1
006306 315 202 014              CALL    PERDE   ;PUNCH TRAILER
006311 021 050 000              LOAD    DE,#LONBLO
006314 175              RSUB:   LOAD    A,L
006315 223                      SUB     A,E
006316 157                      LOAD    L,A
006317 174                      LOAD    A,H
006320 232                      SUBC    A,D
006321 147                      LOAD    H,A
006322 322 332 014              JUMP,CC BL02
006325 031                      ADD     HL,DE   ;NO GET OLD LENGTH AGAIN
006326 021 000 000              LOAD    DE,#0   ;NO REMAINING LENGTH
006331 353                      EX      DE,HL   ;DE IS NOW THE BLOCK LENGTH
                        ;
006332 345              BL02:   PUSH    HL      ;SAVE REMAINING LENGTH
006333 103                      LOAD    B,E     ;SAVE DATA BLOCK LENGTH
006334 353                      EX      DE,HL   ;CORRECT BLOCK LENGTH IN DE
006335 021 006 000              LOAD    DE,#6
006340 031                      ADD     HL,DE
006341 353                      EX      DE,HL
006342 315 202 014              CALL    PERDE
006345 005                      DEC     B
006346 004                      INC     B
006347 050 025                  JUMP,EQ LASTP
006351 341                      POP     HL
006352 321                      POP     DE      ;GET BLOCK ADDRESS
006353 315 202 014              CALL    PERDE
                        ;
006356 032              BL04:   LOAD    A,(DE)
006357 323 000                  LOAD    $DIGO,A
006361 023                      INC     DE
006362 315 207 014              CALL    PERA
006365 005                      DEC     B       ;? LAST BLOCK ?
006366 040 366                  JUMP,NE BL04
006370 325                      PUSH    DE
006371 315 241 014              CALL    CHECKSUM
006374 030 303                  JUMP    BLOCK
                        ;
006376 052 374 002      LASTP:  LOAD    HL,RDGO
006401 353                      EX      DE,HL
006402 315 202 014              CALL    PERDE
006405 315 241 014              CALL    CHECKSUM
006410 315 226 014              CALL    ANORCE
006413 307              LAZ:    RST     0
```

PROGRAMMATION

```
                        ;Programmer
                        ;
                        ;---    Routines
                        ;
006414 072 354 002      INI4:   LOAD    A,SAUL
006417 107                      LOAD    B,A
006420 021 000 003              LOAD    DE,#PROM
006423 041 000 004              LOAD    HL,#RAMEX
006426 311                      RET
                        ;
006427 076 101          ERROR:  LOAD    A,#LETF
006431 323 000                  LOAD    $DIGO,A
006433 030 372                  JUMP    ERROR
                        ;
                        ;+++    Program
                        ;
006435 175              P471:   LOAD    A,L
006436 062 354 002              LOAD    SAUL,A
006441 315 014 015              CALL    INI4
006444 032              P472:   LOAD    A,(DE)
006445 267                      OR      A,A
006446 040 004                  JUMP,NE P474
006450 023                      INC     DE
006451 043                      INC     HL
006452 020 370                  DECJ,NE B,P472
006454 323 000          P474:   LOAD    $DIGO,A ;If any wrong char, display it (+1)
006456 107                      LOAD    B,A
006457 333 007                  LOAD    A,$CLA
006461 346 200                  AND     A,#FULL
006463 170                      LOAD    A,B
006464 050 366                  JUMP,EQ P474
                        ;
006466 315 014 015      CE2:    CALL    INI4
006471 032              CE2:    LOAD    A,(DE)  ;If not compatible, display address
006472 117                      LOAD    C,A
006473 176                      LOAD    A,(HL)
006474 257                      CPL     A
006475 241                      AND     A,C
006476 040 042                  JUMP,NE RETRON
006500 032                      LOAD    A,(DE)  ;Try to program if not equal
006501 276                      CMP     A,(HL)
006502 040 020                  JUMP,NE PROG3
006504 023              CE4:    INC     DE
006505 043                      INC     HL
006506 020 361                  DECJ,NE B,CE2
                        ;
006510 315 014 015      CE3:    CALL    INI4
006513 032              CE3:    LOAD    A,(DE)
006514 276                      CMP     A,(HL)
006515 040 210                  JUMP,NE ERROR
006517 023                      INC     DE
006520 043                      INC     HL
006521 020 370                  DECJ,NE B,CE3
006523 307                      RST     0
                        ;
006524 016 001          PROG3:  LOAD    C,#1    ;Program 1 bit at a time
006526 176              PROG1T: LOAD    A,(HL)
006527 241                      AND     A,C
006530 050 002                  JUMP,EZ PROG2
006532 057                      CPL     A
006533 022                      LOAD    (DE),A
006534 313 001          PROG2:  RL      C
                        ;
006536 050 366                  JUMP,CC PROG1T
006540 030 342                  JUMP    CE4
                        ;
006542 357              RETRON: CALL    INOCO
006543 303 012 000              JUMP    FONC
                        ;
                        ;+++    Move ROM to RAM
                        ;
006546 041 000 003      RET:    LOAD    HL,#ROM
006551 021 000 004              LOAD    DE,#RAMEX
006554 001 000 001              LOAD    BC,#100
006557 355 260                  LDIR
006561 307                      RST     0
                        ;
                        ;+++    Check total of ROM : Type length
                        ;
006562 315 250 014      CHECKTOT CALL   GETARG
006565 041 000 000              LOAD    HL,#0
006570 032              CHE2:   LOAD    A,(DE)
006571 206                      ADD     A,L
006572 157                      LOAD    L,A
006573 050 001                  JUMP,CC CHE3
006575 044                      INC     H
006576 023              CHE3:   INC     DE
006577 013                      DEC     BC
006600 170                      LOAD    A,B
006601 201                      OR      A,C
006602 040 364                  JUMP,NE CHE2
006604 030 334                  JUMP    RETRON
                        ;
                        ;Already in MC2SOM. Included for compatibility with MC2501 with interrupt
                        ;
006606 042 376 002      FOUT    LOAD    SAVOPEN,HL
006611 041 000 026      FOO:    LOAD    HL,#13000       ;LETR
006614 357                      CALL    INOCO
006615 026 333                  LOAD    D,#333
006617 046 311                  LOAD    H,#311
006621 345                      PUSH    HL
006622 325                      PUSH    DE
006623 072 376 002              LOAD    A,SAVOPEN
006626 315 361 002              CALL    SIMU
006631 041 000 000              LOAD    HL,#0
006634 016 000          FIN2:   LOAD    C,#0
006636 321                      POP     DE
006637 321                      POP     DE
006640 021 012 000      FOP2:   LOAD    DE,#FONC
006643 325                      PUSH    DE
006644 303 055 000              JUMP    INOC
                        ;
006647 026 333          FIN:    LOAD    D,#333
006651 046 311                  LOAD    H,#311
006653 345                      PUSH    HL
006654 325                      PUSH    DE
006655 315 361 002              CALL    SIMU
006660 157                      LOAD    L,A
006661 046 020                  LOAD    H,#20   ;Signe +
006663 030 347                  JUMP    FIN2
```

LECTURE BANDE TELEX

```
                        ;PAPER LOADER IN TELTEL FORMAT
                        ;Derived from MicroScope 8 listing

                  3F    MASK=  3F       ;5 low bytes are significant
                  17    NIBL=  17       ;4 bits for a digit
                  16    BEGIN= 16       ;Block beginning character

                        ;***   Routines

                        ;***   GETCAR          ;Read a character

006665 333 011          GETCAR: LOAD    A,$SR
006667 346 002                  AND     A,=SVLP
006671 050 372                  JMP,EQ GETCAR
006673 333 010                  LOAD    A,$SR
006675 323 000                  LOAD    $P,A    ;Echo on loop-loudspeaker
006677 311                      RET

                        ;***   CALL    GETDIGIT
                        ;       Read character and mask

                        GETDIGIT:
006700 315 265 015              CALL    GETCAR
006703 017                      RR      A
006704 346 017                  AND     A,=NIBL
006706 311                      RET

                        ;***   GETBYTE         ;Get a byte and add checksum in C
                                                ;Flags depends on checksum

                        GETBYTE:
006707 315 300 015              CALL    GETDIGIT
006712 007                      RL      A       ;Mask result or check if carry clear if
006713 007                      RL      A       ;RLC instruction must be used
006714 007                      RL      A
006715 007                      RL      A
006716 127                      LOAD    D,A     ;Save shifted digit in A
006717 315 300 015              CALL    GETDIGIT
006722 202                      OR      A,D     ;ADD A,B is also good for moving high half of D int
006723 127                      LOAD    D,A     ;Save A during checksum calculation
006724 201                      ADD     A,C
006725 117                      LOAD    C,A
006726 172                      LOAD    A,D
006727 311                      RET

                        ;++++  Loader program

                        LOTEL:
006730 315 265 015              CALL    GETCAR  ;Wait for begin character
006733 346 037                  AND     A,=MASK
006735 376 016                  COMP    A,=BEGIN
006737 040 367                  JUMP,NE LOTEL

006741 016 000                  LOAD    C,=0    ;Init checksum
006743 315 307 015              CALL    GETBYTE ;Get address
006746 323 000                  LOAD    $DIG0,A
006750 147                      LOAD    H,A
006751 315 307 015              CALL    GETBYTE
006754 157                      LOAD    L,A
006755 315 307 015              CALL    GETBYTE ;Get byte count
006760 107                      LOAD    B,A
006761 267                      OR      A,A     ;Last block?
006762 312 007 016              JMP,ZO LABLOCK

006765 315 307 015      DATA:   CALL    GETBYTE
006770 323 001                  LOAD    $DIG1,A
006772 167                      LOAD    (HL),A
006773 043                      INC     HL
006774 020 367                  DECJ,NE B,DATA

006776 315 307 015              CALL    GETBYTE ;Checksum good?
007001 323 002                  LOAD    $DIG2,A
007003 312 301 014              JMP,EQ SLOCK
007006 307              TERROR: CALL    0       ;Return to DRUPHIN monitor

007007 315 307 015      LABLOCK: CALL   GETBYTE ;Checksum good?
007012 323 003                  LOAD    $DIG3,A
007014 040 370                  JMP,NE TERROR
007016 351                      JUMP    (HL)    ;Jump to loaded program
```

ROUTINES DECIMALES

```
                        ;***    I/O definitions

                177     HUIT=   177
                157     NEUF=   157
                200     POINT=  200

                106     PLUS=   106
                 24     FOIS=   24
                  0     BLANC=  0


                        ;Add or then subtract

007017 346 001          CALCUL: AND     A,#1    ;Odd key?
007021 050 005                  JUMP.,ED CAL2
007023 353                      EX      DE,HL
007024 315 255 016              CALL    DSUBHL
007027 311                      RET
007030 315 233 016      CAL2:   CALL    DADDHL
007033 311                      RET

                        ;CALCULATOR ON 7-SEGMENT DISPLAY

007034 315 142 016      T2:     CALL    INDC00
007037 345              TES3:   PUSH    HL
007040 315 145 016              CALL    INDC0
007043 321                      POP     DE
007044 315 017 016              CALL    CALCUL
007047 030 366                  JUMP    TES3


                        ;---    CALL    CHIFF   Mask, shift and convert digit

                        ;in     B: number
                        ;out    B: shifted number
                        ;       A: converted segments
                        ;modif: A,B,DE

007051 170              CHIF:   LOAD    A,B
007052 017                      RR      A
007053 017                      RR      A
007054 017                      RR      A
007055 017                      RR      A
007056 107                      LOAD    B,A
007057 346 017                  AND     A,#17
007061 021 070 016      CHI2:   LOAD    DE,#CTABLE
007064 203                      ADD     A,E
007065 137                      LOAD    E,A
007066 032                      LOAD    A,(DE)
007067 311                      RET

                        ;next table must be fully in a same page

007070 077 006 133 117  CTABLE: .BYTE ZERO, UN, DEUX, TROIS, QUATRE, CINQ, SIX, SEPT,HUIT, NEUF
007074 146 155 175 007
007100 177 157
007102 110 200 106 100          .BYTE EGAL, POINT, PLUS, MOINS, FOIS, BLANC
007106 024 000

                        ;---    CALL    AFB     ;display 2 digits in B at (C) and (C+1)

                        ;in     B: 2 digit number
                        ;       C: address pointer of MSD
                        ;out    C: incremented by 2
                        ;modif  A,B,C,DE
                        ;calls  CHIFF+

007110 315 051 016      AFB:    CALL    CHIF
007113 355 171                  LOAD    $(C),A
007115 014                      INC     C
007116 315 051 016              CALL    CHIF
007121 355 171                  LOAD    $(C),A
007123 014                      INC     C
007124 311                      RET

                        ;---    CALL    AFHL    display 4 digits in HL

                        ;in HL  :4 digit
                        ;       C: address pointer of MSD
                        ;modif. A,DE
                        ;calls  AFB++

007125 305              AFHL:   PUSH    BC
007126 016 000                  LOAD    C,#0
007130 104                      LOAD    B,H
007131 315 110 016              CALL    AFB
007134 105                      LOAD    B,L
007135 315 110 016              CALL    AFB
007140 301                      POP     BC
007141 311                      RET

                        ;---    CALL INDC, INDC0, INDC00        display HL and build
                        ;                       new number in HL until order

                        ;in     HL:number to display while waiting (=0 with INDC00)
                        ;       C: counter (=0 with INDC0) if =0, first key clear HL
                        ;out    HL: new number
                        ;       C: number of digits introduced
                        ;modif  A,B,C,DE,HL
                        ;calls  AFHL+++

007142 041 000 000      INDC00: LOAD    HL,#0
007145 016 000          INDC0:  LOAD    C,#0
007147 315 125 016      INDC:   CALL    AFHL
007152 315 100 016              CALL    TKEY
007155 300                      RET,CS
007156 030 367                  JUMP    INDC
```

```
                        ;---    CALL    TKEY    Test keyboard and get key

                        ;in     HL: BCD number
                        ;       C: first key flag
                        ;out    HL: with optional new digit added
                        ;       CS or CC if order or not
                        ;       SS or SC if number key or no key

007160 333 007          TKEY:   LOAD    A,$CLA
007162 267                      OR      A,A
007163 360                      RET,SC          ;CC
007164 346 037                  AND     A,#CLA
007166 107                      LOAD    B,A
007167 306 360                  ADD     A,#360
007171 330                      RET,CS
007172 171                      LOAD    A,C
007173 267                      OR      A,A
007174 302 204 016              JUMP,NE INC2
007177 041 000 000              LOAD    HL,#0
007202 016 200                  LOAD    C,#200
007204 051              INC2:   ADD     HL,HL   ;add new digit
007205 051                      ADD     HL,HL
007206 051                      ADD     HL,HL
007207 051                      ADD     HL,HL
007210 170                      LOAD    A,B
007211 265                      OR      A,L     ;CC
007212 157                      LOAD    L,A
007213 014                      INC     C       ;SS if less than 128. Keystroke
007214 311                      RET


                        ;---    CALL    DINCX   decimal INC (HL) 16 bits= 4 di

                        ;in     HL: pointer to incremented locations (low-high)
                        ;modif  A

007215 176              DINCX:  LOAD    A,(HL)
007216 306 001                  ADD     A,#1
007220 047                      DA      A
007221 167                      LOAD    (HL),A
007222 300                      RET,NE
007223 043                      INC     HL
007224 176                      LOAD    A,(HL)
007225 316 000                  ADDC    A,#0
007227 047                      DA      A
007230 167                      LOAD    (HL),A
007231 053                      DEC     HL
007232 311                      RET             ;CS if overflow

                        ;---CALL        DADDHLDE        ;decimal ADD HL,DE

                        ;in     HL,DE
                        ;out    HL
                        ;       Carry Set if overflow
                        ;modif  A,HL

007233 175              DADDHL: LOAD    A,L
007234 203                      ADD     A,E
007235 047                      DA      A
007236 157                      LOAD    L,A
007237 174              DADDH:  LOAD    A,H
007240 212                      ADDC    A,D
007241 047                      DA      A
007242 147                      LOAD    H,A
007243 311                      RET

                        ;---    CALL    DADDHH  decimal ADD HL,HL (DBSL HL)
                        ;in,out HL
                        ;modif  A,HL

007244 175              DADDHH: LOAD    A,L
007245 205                      ADD     A,L
007246 047                      DA      A
007247 157                      LOAD    L,A
007250 174                      LOAD    A,H
007251 214                      ADD     A,H
007252 047                      DA      A
007253 147                      LOAD    H,A
007254 311                      RET

                        ;---    CALL    DSUBHL decimal SUB HL,DE
                        ;see above

007255 175              DSUBHL: LOAD    A,L
007256 223                      SUB     A,E
007257 047                      DA      A
007260 157                      LOAD    L,A

007261 174              DSUBH:  LOAD    A,H
007262 232                      SUBC    A,D
007263 047                      DA      A
007264 147                      LOAD    H,A
007265 311                      RET

                        ;---    CALL    CMPHL   Compare HL and DE

                        ;in     DE,HL
                        ;out    EQ if equal, MS if HL >= DE, LO if HL < DE
                        ;modif  F,A

007266 174              CMPHL:  LOAD    A,H
007267 272                      CMP     A,D
007270 320                      RET,NE
007271 175                      LOAD    A,L
007272 273                      CMP     A,E
007273 311                      RET

                        ;---    CALL    DINCHL  ;decimal INC HL

                        ;in     HL
                        ;out    HL
                        ;       CS if overflow (HL=0)
                        ;modif  A,HL

007274 175              DINCHL: LOAD    A,L
007275 306 001                  ADD     A,#1
007277 047                      DA      A
007300 157                      LOAD    L,A
007301 320                      RET,CC
007302 174              DINCH:  LOAD    A,H
007303 316 000                  ADDC    A,#0
007305 047                      DA      A
007306 147                      LOAD    H,A
007307 311                      RET

                        ;---    CALL    DDECHL  ;decimal DEC HL
                        ;see above

007310 175              DDECHL: LOAD    A,L
007311 326 001                  SUB     A,#1
007313 047                      DA      A
007314 157                      LOAD    L,A
007315 330                      RET,CC
007316 174              DDECH:  LOAD    A,H
007317 336 000                  SUBC    A,#0
007321 047                      DA      A
007322 147                      LOAD    H,A
007323 311                      RET
```

CASSETTES

```
                        .LOC    7400

                        ;CASSETTE
                        ;Ce programme sauve sur cassette ou restitue
                        ;des blocs de 256. bytes.
                        ;Variables a initialiser avec le moniteur
                        ;avant enregistrement.
                        ;PARA            Adresse du debut du programme
                        ;                a sauver
                        ;       PARA+1. page
                        ;       PARA:   adresse dans la page
                        ;               toujours nulle
                        ;PARA+2          Adresse du programme executé
                        ;                a la fin de la lecture
                        ;                de la bande
                        ;       PARA+3: page
                        ;       PARA+2: adresse dans la page
                        ;               (quelconque)

                        ; 1350  Always 0
                        ; 1351  Page to be saved
                        ; 1352  Start address
                        ;       0 if no auto start

                        ;       25 if second cassette file to load
                        ; 1353  Page start address
                        ;       0 if no auto start
                        ;       2 if second cassette file to load

          1350  PARA=   1350

                        ;+++    PROGRAMME DE LECTURE

007400 257              READ:   XOR     A,A
007401 041 350 002              LOAD    HL,#PARA
007404 167                      LOAD    (HL),A
007405 315 105 017      READ1:  CALL    GETBIT
007410 173                      LOAD    A,E
007411 037                      RRC     A
007412 137                      LOAD    E,A
007413 376 026                  COMP    A,#SYN
007415 040 366                  JUMP,NE READ1
007417 315 241 017      READ2:  CALL    CGTBYTE
007422 376 026                  COMP    A,#SYN
007424 050 371                  JUMP,EQ READ2
007426 267                      OR      A,A
007427 040 354                  JUMP,NE READ1
007431 137                      LOAD    E,A
007432 006 003                  LOAD    B,#3
007434 043                      INC     HL
007435 315 241 017      READ3.  CALL    CGTBYTE
007440 167                      LOAD    (HL),A
007441 043                      INC     HL
007442 020 371                  DECJ,NE B,READ3
007444 052 350 002              LOAD    HL,PARA
007447 315 241 017      READ4:  CALL    CGTBYTE
007452 167                      LOAD    (HL),A
007453 043                      INC     HL
007454 020 371                  DECJ,NE B,READ4
007456 052 352 002              LOAD    HL,PARA+2
007461 315 241 017              CALL    CGTBYTE
007464 034                      INC     E
007465 040 001                  JUMP.,NE CERROR
007467 351                      JUMP    (HL)
007470 315 261 017      CERROR: CALL    FLASH
007473 030 373                  JUMP    CERROR

                        ;+++    PROGRAMME D'ECRITURE

007475 315 250 014      WRITE:  CALL    GETARG
007500 042 352 002              LOAD    PARA+2,HL
007503 142                      LOAD    H,D
007504 056 000                  LOAD    L,#0
007506 042 350 002              LOAD    PARA,HL

007511 006 100                  LOAD    B,#SYNL
007513 016 026          WRIT1:  LOAD    C,#SYN
007515 315 266 017              CALL    WRBYTE
007520 020 371                  DECJ,NE B,WRIT1
007522 016 000                  LOAD    C,#0
007524 315 266 017              CALL    WRBYTE
007527 036 000                  LOAD    E,#0
007531 006 003                  LOAD    B,#3
007533 041 351 002              LOAD    HL,#PARA+1
007536 116              WRIT2:  LOAD    C,(HL)
007537 315 266 017              CALL    WRBYTE
007542 043                      INC     HL
007543 020 371                  DECJ,NE B,WRIT2
007545 052 350 002              LOAD    HL,PARA
007550 116              WRIT3:  LOAD    C,(HL)
007551 315 266 017              CALL    WRBYTE
007554 043                      INC     HL
007555 020 371                  DECJ,NE B,WRIT3
007557 173                      LOAD    A,E
007560 057                      CPL     A
007561 117                      LOAD    C,A
007562 315 266 017              CALL    WRBYTE
007565 307                      RST     RST0

                        ;---ROUTINES

                        ;---GETBIT
                        ;Lecture d un bit
                        ;CARRY CLEAR si 0 ,CARRY SET si 1
                        ;Registres utilises:
                        ;       A:
                        ;       B:      delai
                        ;       C:      nombre de transition
                        ;       D:
                        ;registres modifies:  A,D

007566 305              GETBIT: PUSH    BC
                        ;       LOAD    A,#COMMAND+DTR
                        ;       LOAD    $SPR,A
007567 016 001                  LOAD    C,#1
007571 333 011                  LOAD    A,$SPR

007573 127                      LOAD    D,A
007574 333 011          GETB1:  LOAD    A,$SPR
007576 272                      COMP    A,D
007577 050 373                  JUMP,EQ GETB1
                        ;       LOAD    A,#RTS+COMMAND+DTR
                        ;       LOAD    $SPR,A
007601 323 006                  LOAD    $AP,A
007603 006 010          GETB2:  LOAD    B,#SPDB1
007605 333 011                  LOAD    A,$SPR
007607 127                      LOAD    D,A
007610 014                      INC     C
007611 333 011          GETB4:  LOAD    A,$SPR
007613 272                      COMP    A,D
007614 040 365                  JUMP,NE GETB2
007616 020 371                  DECJ,NE B,GETB4
                        ;       LOAD    A,#COMMAND
                        ;       LOAD    $SPR,A
007620 323 006                  LOAD    $AP,A
007622 171                      LOAD    A,C
007623 376 004                  COMP    A,#ZERO
007625 334 251 017              CALL,LO FLASH
007630 376 017                  COMP    A,#MARK
007632 324 251 017              CALL,HS FLASH
007635 326 011                  SUB     A,#ZERO
007637 301                      POP     BC
007640 311                      RET

                        ;---CGTBYTE

                        ;Lecture d un byte
                        ;Registres utilises:
                        ;       A:      byte lu
                        ;       B       formation du byte
                        ;       D       utilise par GETBIT
                        ;Registres modifies:  A,D,E

007641 305              CGTBYTE PUSH    BC
007642 006 200                  LOAD    B,#200
007644 315 166 017      GETB1:  CALL    GETBIT
007647 313 030                  RRC     B
007651 050 371                  JUMP,CC GETB1
007653 170                      LOAD    A,B
007654 203                      ADD     A,E
007655 137                      LOAD    E,A
007656 170                      LOAD    A,B
007657 301                      POP     BC
007660 311                      RET

                        ;---FLASH

007661 076 161          FLASH:  LOAD    A,#LETT
007663 323 006                  LOAD    $DISP,A
007665 311                      RET


                        ;---WRBYTE

                        ;Envoi d un byte
                        ;Registres utilises:
                        ;       A:
                        ;       B       longueur du byte
                        ;       C       byte a envoyer
                        ;       E:      checksum
                        ;Registres modifies:  A,C,E

007666 305              WRBYTE: PUSH    BC
007667 006 010                  LOAD    B,#8*L
007671 171                      LOAD    A,C
007672 203                      ADD     A,E
007673 137                      LOAD    E,A
007674 313 031          WRBYT:  RRC     C

                        ;---PULSE
                        ;Envoie 6 impulsions pour un 0 (CARRY CLEAR)
                        ;Envoie 3 impulsions pour un 1 (CARRY SET)
                        ;Registres utilises:
                        ;       A:
                        ;       B:      delai
                        ;       C:      nombre de transition
                        ;Registres modifies:  A

007676 305              PULSE:  PUSH    BC
007677 016 006                  LOAD    C,#006
007701 070 002                  JUMP.,CS PULZ
007703 016 003                  LOAD    C,#002
007705 171              PULZ:   LOAD    A,C
007706 007                      RLC     A
007707 057                      CPL     A
007710 346 042                  AND     A,#TAPEDTR
007712 323 011                  LOAD    $SPR,A
007714 006 036                  LOAD    B,#PULSE
007716 020 376          PUL4:   DECJ,NE B,PUL4
007720 015                      DEC     C
007721 171                      LOAD    A,C
007722 376 010                  COMP    A,#COMP
007724 040 007                  JUMP,NE PULZ
007726 301                      POP     BC

007727 020 343                  DECJ,NE B,WRBYT
007731 301                      POP     BC
007732 311                      RET

007733 315 241 017      TPLA:   CALL    CGTBYTE
007736 376 126                  CMP     A,#126
007740 050 371                  JUMP,EQ TPLA
007742 376 052                  COMP    A,#052
007744 050 366                  JUMP,EQ TPLA
007746 315 251 017              CALL    FLASH
007751 030 360                  JUMP    TPLA

007753 016 126          TARY:   LOAD    C,#126
007755 315 266 017              CALL    WRBYTE
007760 030 371                  JUMP    TARY

                        .END
```