

Service
Service
Service sa

DT
DEPARTEMENT FORMATION TECHNIQUE

VG 5000^μ

MICRO ORDINATEUR



PHILIPS



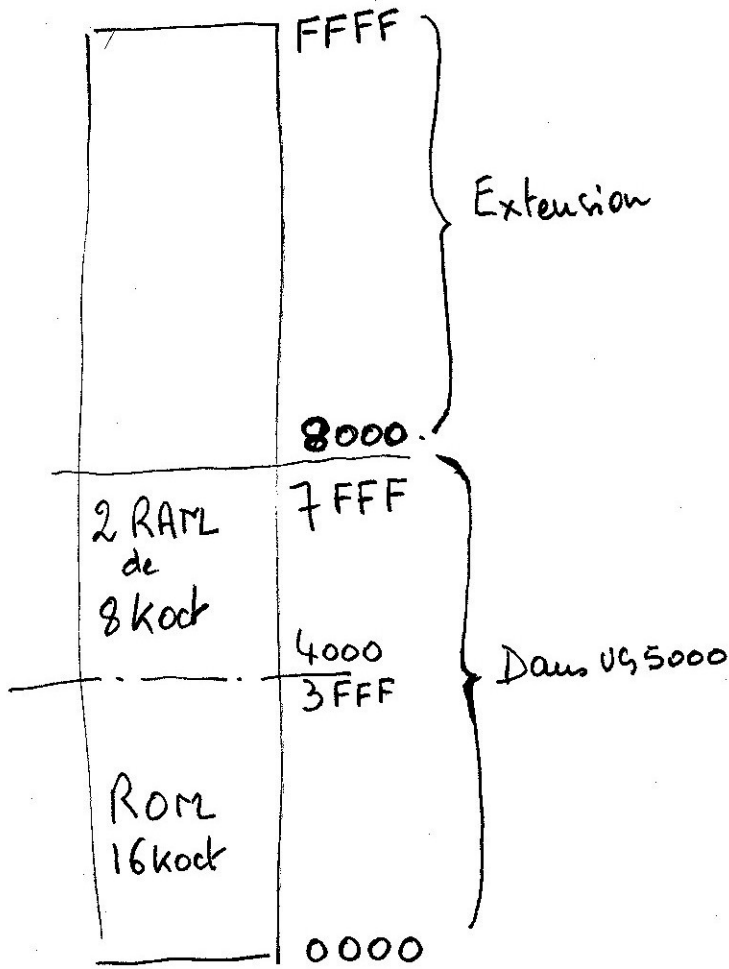
*
* -V G 5 0 0 0- *
* - S O M M A I R E - *
*

- I SCHEMA SYNOPTIQUE DU VG5000
- II LE Z80 DANS LE VG5000
- II-1 CYCLE DE RECHERCHE D'INSTRUCTION
- II-2 LES HORLOGES
- II-3 LE COMPTEUR DE RAFRAICHISSEMENT
- II-4 LES INITIALISATIONS
- III LE DECODEUR D'ADRESSES DES MEMOIRES
- IV LA MEMOIRE DE PROGRAMME
- V LES MEMOIRES VIVES
- V-1 RAPPELS SUR LES MEMOIRES PSEUDO-STATICS
- VI LE DECODEUR D'ADRESSES DES ENTREE-SORTIES
- VII LE CLAVIER
- VIII L'INTERFACE K7 ET SON
- VIII-1 LA SORTIE SON
- VIII-2 L'INTERFACE K7
- VIII-2-1 EN SORTIE
- VIII-2-2 EN ENTREE
- VIII-2-3 PROCEDURE DE COMMUNICATION
- IX L'INTERFACE VIDEO
- IX-1 COMMUNICATIONS ENTRE L'EF9345 ET LE TELEVISEUR
- IX-2 COMMUNICATIONS ENTRE LE Z80 ET L'EF 9345
- IX-3 COMMUNICATIONS ENTRE LA RAM ET L'EF 9345
- X LE CONNECTEUR D'EXTENSIONS

ADDITIFS:

- ADDITIF n1 QUELQUES MOTS SUR LE Z80
- ADDITIF n2 ORGANISATION MEMOIRE DE LA RAM
- ADDITIF n3 MANIPULATION SUR LA SORTIE SON
- ADDITIF n4 L'INTERFACE VIDEO EFS345
- ADDITIF n5 PROGRAMME BASIC PERMETTANT DE TRAVAILLER
LE JEU D'INSTRUCTIONS DU Z80 EN HEXADECIMAL
- ADDITIF n6 QUELQUES PROGRAMMES EN LANGAGE BASIC

64k oct = 65536 .



ROM

16k oct + 2k oct → 18k ROM
ROM Basic
initialisation
editeur
interpréter
Mémoire de caractères
matrice . ex : A

16k oct + 2
texte basic

vitesse 4MHz

I Synoptique

le VG5000 est construit autour d'un Z80A. Il constitue de façon typique la structure d'un P.ordinateur

- on distingue en figure 1 :
- a - 3 bus
 - bus de donnée bidirectionnel
 - bus adresse
 - lignes de contrôle
 - b - l'espace mémoire partagé en
 - mémoire de programme (ROM)
 - mémoire de lecture (RAM)
 - c - les systèmes d'Entrées/Sorties
 - le clavier
 - l'interface video
 - interface son et K7.
 - d - un système de décodage permet la validation ou l'inhibition de chacun de ces blocs; en écriture ou lecture pour les mémoires et en entrée ou sortie pour les périphériques

touche A = ligne NMI

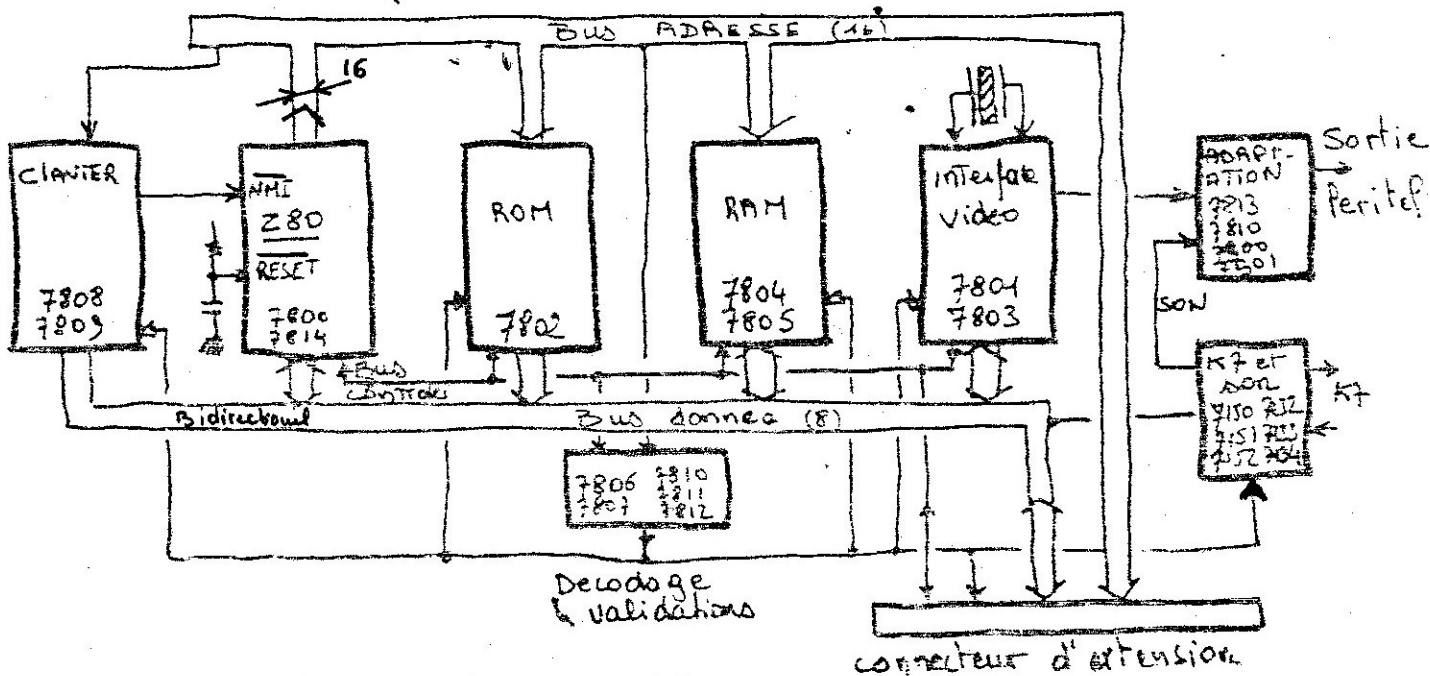


figure 1: synoptique du VG5000

II. le Z80 dans le V65000

Vous trouverez dans l'additif des informations complémentaires sur le Z80.

II-1 - cycle de recherche d'instruction.

Nous étudierons plus loin que la mémoire de programme peut être externe et à dire reliée au Z80 au travers du connecteur d'extensions. Le connecteur peut recevoir tous les types de mémoire quelque soit leurs temps d'accès. Afin d'être certain d'avoir le temps d'aller chercher une instruction en mémoire de programme; le cycle M1 est rallongé d'une phase d'horloge (250ns) grâce à l'étage 7814 (voir la figure 4)

Le cycle M1 dure normalement quatre périodes d'horloge. Sur le Z de T2 le Z80 regarde sa broche WAIT. Si celle-ci est au niveau haut, rien ne change. Mais si elle est au niveau bas; le Z80 insère une ou plusieurs périodes entre les temps T2 et T3. Pour sortir de cette phase d'attente; il faudra un niveau "1" sur wait pendant le Z du cycle d'attente.

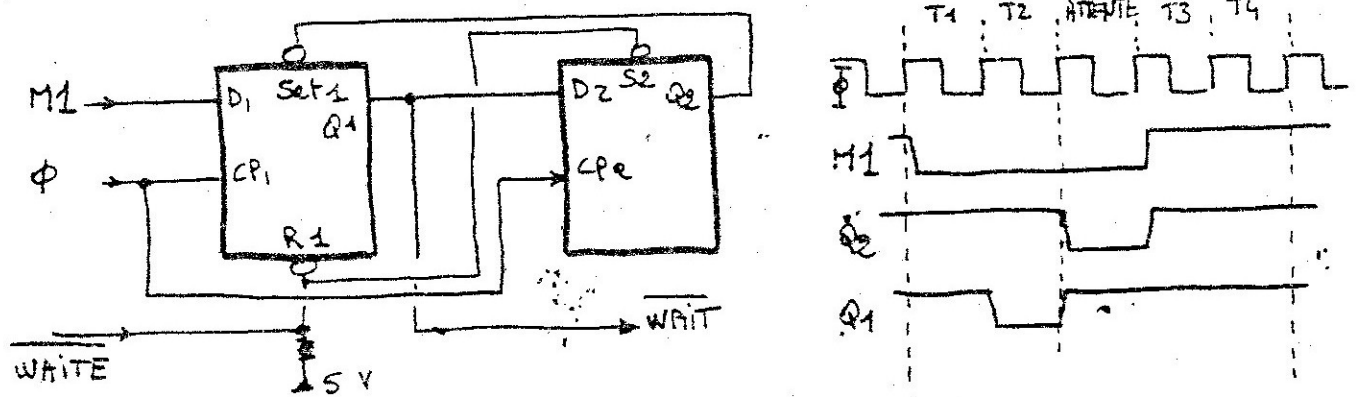


Figure 4: fabrication de WAIT

Remarque: Une ligne WAIT sur le connecteur arrière aide en entrée-sortie toutes extensions possibles -


II-2 - les horloges

Le V65000 utilise un quartz à 12 MHz qui alimente l'interface video EF9345. (12 MHz avec un rapport cyclique de 1). La broche HP du 9345 délivre du 4 MHz utilisé ici pour alimenter le Z80 A.

II.3 - le compteur de rafraichissement -

Dans le V65000 ; seule l'impulsion $\overline{\text{refresh}}$ est utilisée pour rafraichir les RAMS PSEUDO STATICS. le compteur de rafraichissement du Z80 n'est pas utilisé car les RAMS possèdent leur propre compteur.

II.4 - Les initialisations

- le reset s'effectue de façon Hardware dès la mise sous tension grâce à un réseau R.C.
- le ligne NMI effectue un "reset" Software à partir du clavier. (CTRL + )
- le Z80 exécute son programme de routine de scannage clavier et de modification d'affichage dès que la broche $\overline{\text{INT}}$ tombe à "0", c'est à dire synchronisé en trame, toutes les 20 ms. la ligne $\overline{\text{INT}}_{\text{EX}}$ sur le connecteur arriere ne demande qu'à descendre à "0" si un périphérique d'extension l'exige, pour toutes applications spéciales

Les lignes $\overline{\text{BUSREQ}}$ et $\overline{\text{BUSACK}}$ ne sont pas utilisés -

III Decodeur d'adresses des memoires - (voir la fig 5)

Le 7806 decode les 64Ko adressables par le Z80 en 8 zones de 8Ko chacune. Ces zones sont validées si $\overline{\text{MAEQ}} = 0$ et $\overline{\text{RFSH}} = 1$ c'est à dire pour des adresses destinées à des memoires et non au rafraichissement

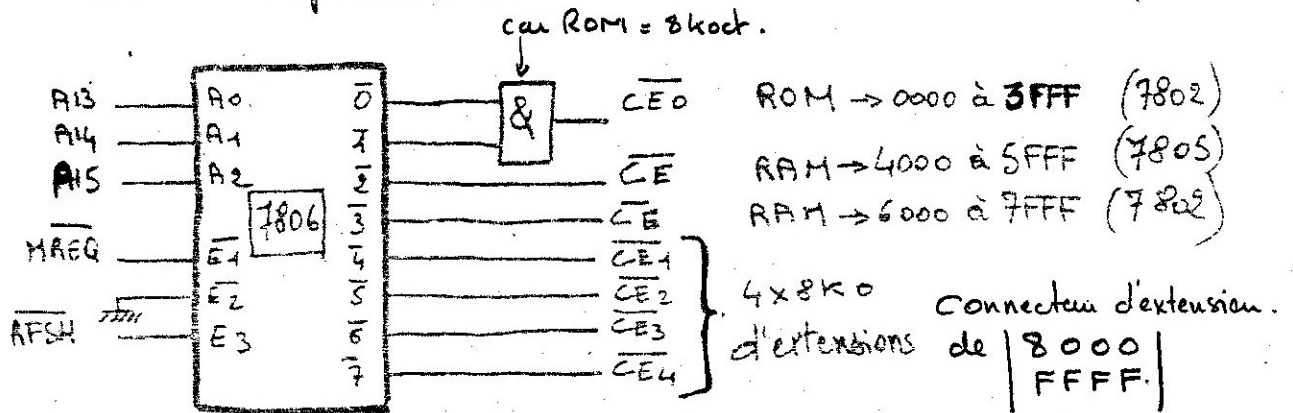


Figure 5: Découpage de la géographie mémoire

IV - la mémoire de programme.

les 16 Ko de mémoire de programme sont placés dans la géographique mémoire du Z80 de 0000 à 3FFF(H)

la ligne \bar{rd} échantillonne l'instant de lecture en positionnant \bar{OD} (OUTPUT DISABLE) à "0".

la puce 7811 valide la ROM pour les adresses allant de 0000 à 1FFF(H) et de 2000(H) à 3FFF(H).

la broche 27 (CS: chip select) est reliée au 5V via une résistance de 3,3K Ω . La ligne "CSROM1" est disponible sur le connecteur d'extension ainsi que la ligne \bar{CE}_0 . Ceci autorise la présence d'une mémoire de programme externe placée dans la même zone d'adresse que la ROM interne (voir la figure 6)

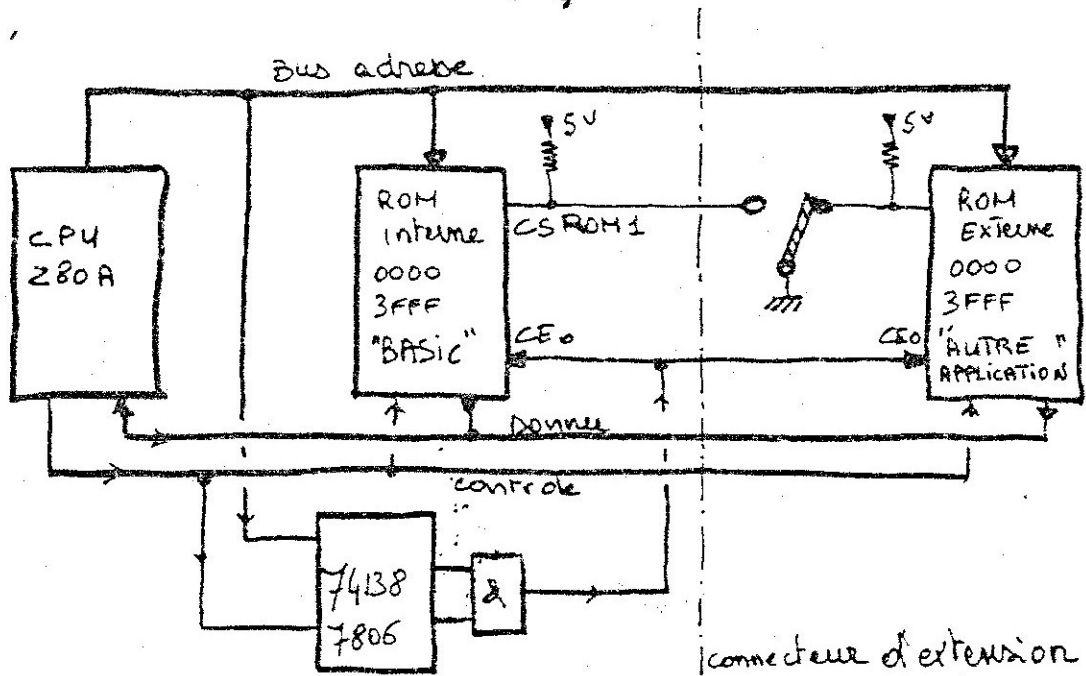


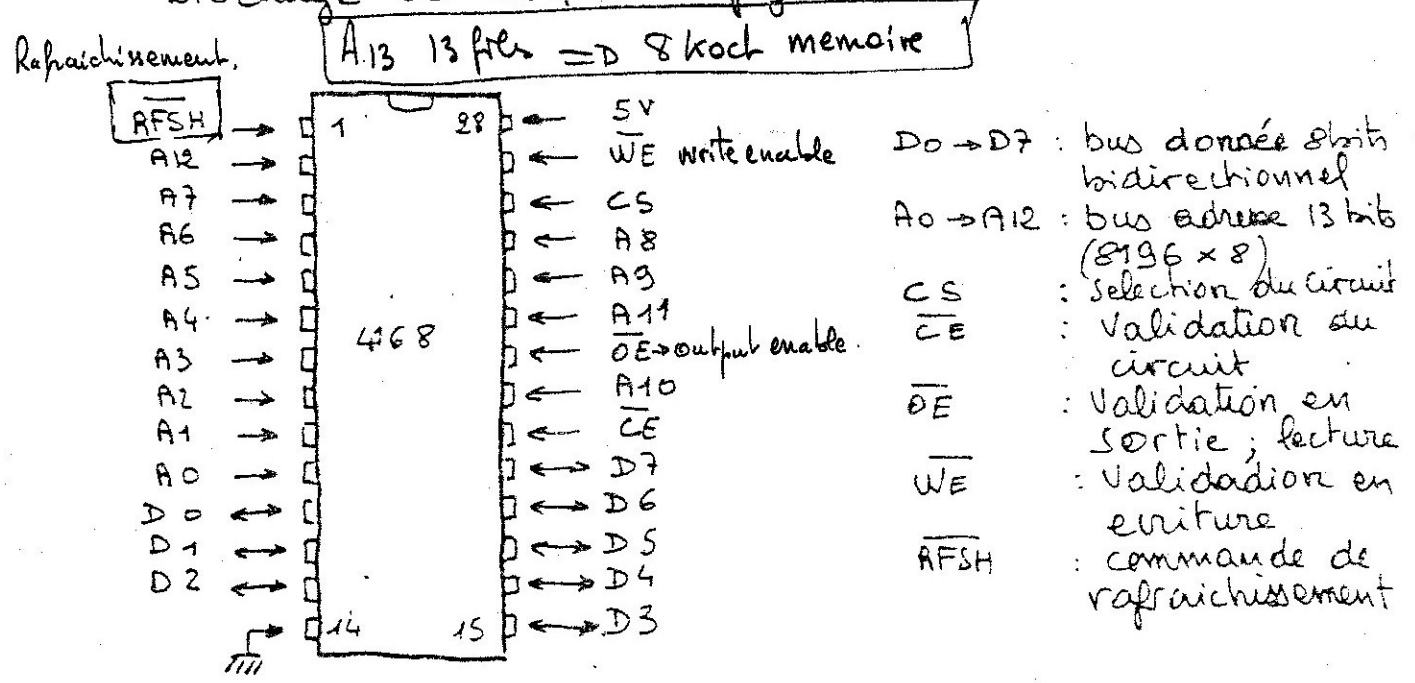
Figure 6: utilisation d'une ROM d'extension.

RAM pseudo statique \equiv à dynamique constituée de Combs de quelques PF,

IV - les mémoires vives -

- le VG 500 utilise trois RAMs pseudo statiques identiques de 8 Ko : de type 4168.
Deux de ces RAMs ; reperées 7804 et 7805 constituent la mémoire vive du Z80. Elles sont respectivement placées de 4000(H) à 5FFF(H) et de 6000(H) à 7FFF(H) grâce au deuxièm d'adresse 7806.
- la troisième RAM est utilisée en mémoire d'écran et mémoire de caractères d'extensions pour l'interface video

brochage de la 4168 figure 7



II. 1 rappels sur les mémoires pseudo-statics

les mémoires pseudo-statics sont des mémoires dynamiques vues par l'unité centrale comme des mémoires statiques. En effet, le système de rafraichissement est interne à la RAM.

la RAM 4168 est organisée en une matrice de 128 rangées de 64 octets chacune (8192 octets). Elle comporte son propre compteur de rafraichissement sur 7 bits. Celui-ci s'effectue rangée après rangée ; en 2 ms pour l'ensemble de la mémoire c'est à dire à raison de 16 μ s par rangée.

la RAM 4168 possède 3 modes de rafraichissement

- a. Rafraichissement par adresse externe (AFSH = 5V)
A chaque cycle de lecture ou d'écriture ;

l'ensemble de la rangée, correspondant à la case mémoire manipulée, est alors rafraîchi.

- Ce mode n'est pas directement utilisable dans le cas d'une mémoire de service car l'adressage s'effectue de façon aléatoire. Le rafraîchissement est, alors lui aussi aléatoire.
- Dans le cas d'une utilisation en mémoire d'écran, les rangées sont rafraîchies de façon méthodique; du début jusqu'à la fin. Une scrutation méthodique de la RAM nous amène alors un rafraîchissement méthodique et complet.

Remarque: c'est dans le mode de fonctionnement que travaille la mémoire de l'interface vidéo.

- b. Rafraîchissement par adresse interne sur une impulsion de RFSH. (~~7804~~) ~~7805~~

Un système externe génère régulièrement des impulsions sur la ligne RFSH. A chaque impulsion, une RAM pointée par le compteur interne de la RAM est rafraîchie. Ce compteur s'incrémente. Il conviendra de générer 128 impulsions en 2ms afin de rafraîchir l'ensemble de la mémoire.

Remarque: Dans le V6 5000, ce mode est utilisé pour les RAM reperés 7804 et 7805. L'impulsion RFSH est générée par le Z80 à un moment où il travaille sur lui-même (decodage de l'instruction). Ce temps de rafraîchissement lui est donc totalement transparent.

- c. mode d'auto rafraîchissement.

Si la ligne RFSH est maintenue à "0", un cycle interne est effectué automatiquement toutes les 15 μ s jusqu'à ce que la ligne RFSH repasse au niveau haut.

Dans ce mode toutes les entrées (sauf RFSH) sont inhibées. La consommation de la RAM tombe alors à 98 mW.

Ce mode est utilisé en cas d'une défaillance catastrophique de l'alimentation. Une batterie est nécessaire dans ce mode pour garder la RAM alimentée dans une position "stand by".

Ce mode n'est pas utilisé dans le V6 5000.

- organisation de la RAM:

voir additif n° 2

VI - Decodage d'adresses des entrées sorties

le decodage repere "7407" fabrique les signaux de validation des differents peripheriques du VG5000 (voir la figure 8)

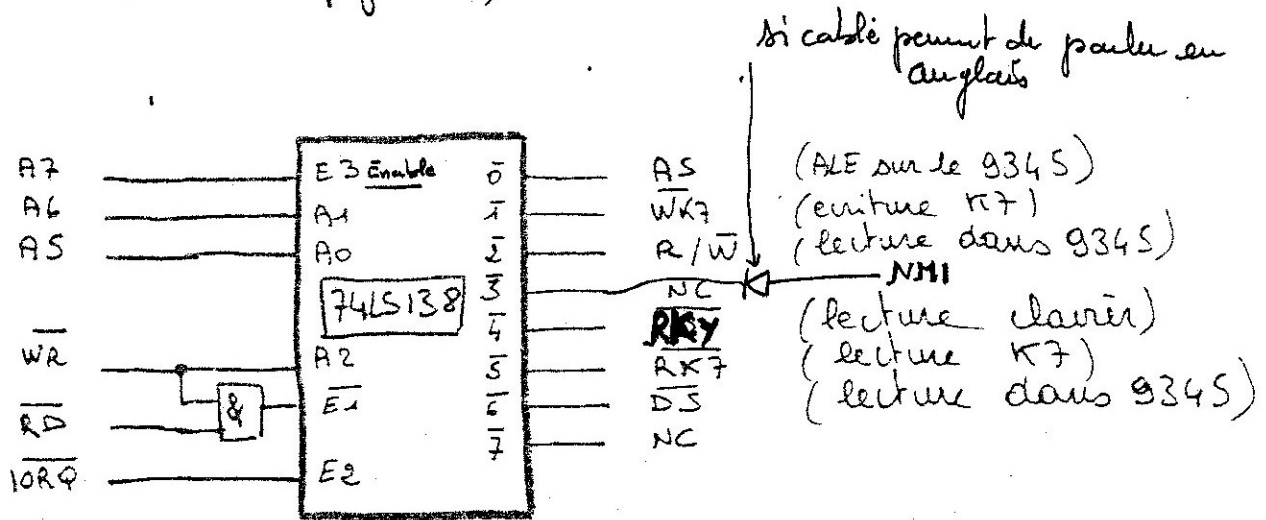


figure 8: Decodage d'adresses des Entrées - sorties.

le bit A7 du bus adresse est connecté à une entrée de validation active au niveau Haut. Ainsi ce decodage pointe des circuits placés entre les adresses 80(H) et FF(H). ces peripheriques sont internes au VG 5000. Les adresses comprises entre 00(H) et 7F(H) sont destinées à des peripheriques d'extension. (imprimante, poignées de jeu, ect...)

les bits d'adresse A5 et A6 partagent l'espace 80(H) à FF(H) en 4 parties selon l'état de la broche WR; ces quatre parties peuvent manipuler un peripherique soit en entrée; soit en sortie. (voir la figure 9)

ce decodage n'est valide que si IORQ est à "0" et si RA ou WR est à "0".

00

7F

80

9F

A0

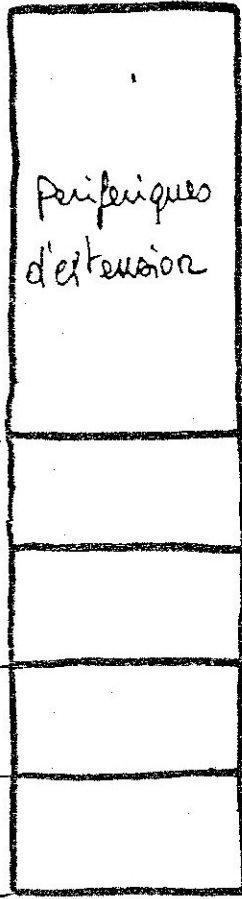
BF

CO

DF

EO

FF



écriture

lecture

écriture				lecture			
A7	A6	A5		A7	A6	A5	
0	0	0	80 → 9F (ALE)	1	0	0	80 → 9F (RKY)
0	0	1	A0 → BF (WK7)	1	0	1	A0 → BF (RW7)
0	1	0	CO → DF (WA)	1	1	0	CO → DF (RA)
0	1	1	EO → FF (NC)	1	1	1	EO → FF (NC)

(ALE du 9345 en écriture) ou (lecture clavier en "lecture")
 (sortie K7 en écriture) ou (entrée K7 en lecture)
 (sortie de donnée vers 9345 en écriture) ou (entrée de donnée du 9345 en lecture)
 NON utilisé -

figure 9 : repartition des 256 octets d'entrée - sortie

VII - le clavier -

le clavier comprend 62 touches montées ^{dans une} matrice 8x8 et un ^{supplémentaire} touche d'initialisation software (NMI). Dès que la ligne NMI est à 0, le programme d'interruption non masquable est exécuté. Celui-ci réalise la lecture de la touche "CTRL" (contrôle). Si celle-ci est à "0", le V65000 est initialisé (écran, pointeur) et la touche "CTRL" n'est pas active; rien ne se passe. Cette précaution est prise afin de ne pas réinitialiser ^(accidentelle) le V65000 par une seule action sur la touche .

le circuit 74LS156 repère 7808 constitue un double décodeur 2 vers 4 câblé en décodeur 3 vers 8

En faisant évoluer les 4 bits de poids faible du bus adresse de 0 à 7; nous validerons successivement les 8 rangées de la matrice du clavier.

- le circuit 74LS244 repère 7809 est un buffer 3 états
- le circuit met les colonnes du clavier en liaison avec le bus de donnée du Z80 dès que la ligne RKY tombe à "0". (RKY = Read KEY = lecture clavier)

mesures sur le clavier -

- ces mesures intéressent : le bus adresse, le bus de donnée, la ligne NMI et la ligne RKY

- afin de mesurer un signal exploitable ; il faut être certain que le Z80 interroge son clavier - pour cela ; se placer en position "OK!"

la scrutation s'effectue toutes les 20ms dans le programme d'interruption - voir la fig 10

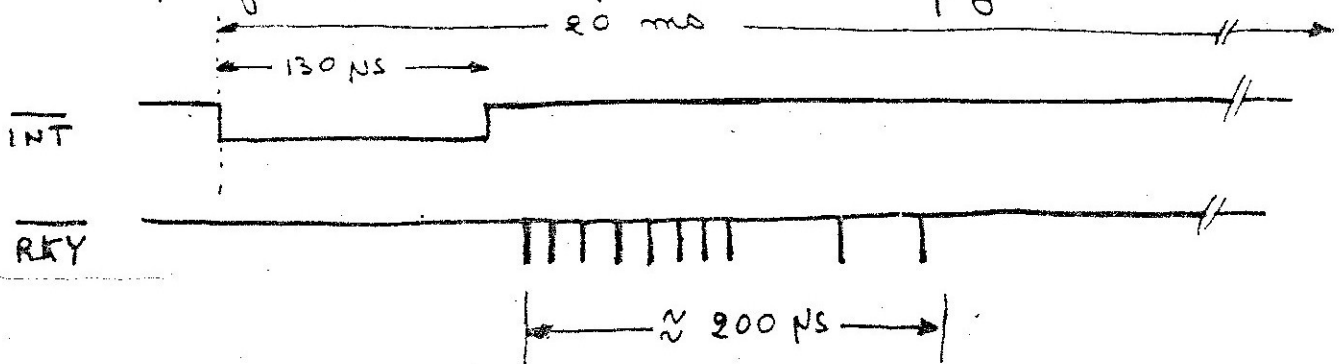


figure 10: scrutation clavier -

VIII L'interface K7 et son (voir la figure 13)

VIII-1 Sortie Son -

le son passe par la platine interface K7 -

le bit 3 du bus de donnée est LATCHÉ dans une bascule repérée 7150 - le coup d'horloge est le signal wrk7 fabriqué par le décodeur d'adresses d'entrée-sorties - Ce "0" ou ce "1" stable, sortant de Q2 du 7150 est appliqué en "sound 0" à l'entrée de la platine principale - Le Transistor 7901 est un adaptateur pour la prise péritelvision - Il faut remarquer que le signal "Sound 0" est également disponible sur le connecteur arriére -

voir dans l'additif n°3 manipulation en langage Z80 sur la sortie SON

VIII-2 L'interface K7:

VIII-2-1 En sortie l'interface K7 travaille suivant le même principe que l'interface SON - Mais on utilise les bits 0 et 1 du bus de donnée pour la sortie K7 et la Télécommande de Remote (Telecommande).

la Télécommande "Remote" sert à alimenter ou pas le moteur cabestan - Les transistors 7133 et 7134 sont montés "Têtes bêche" afin d'éliminer tous problèmes de polarités entre le G5000 et le magnétophone utilisé.

VIII-2-2 En entrée, le signal est mis en forme et amplifié dans l'étage 7151. le buffer 3 états 7425 125 repéré 7152⁰ laissera passer le signal vers le bit 7 du bus de donnée si la ligne RK7 (read K7) est à "0".

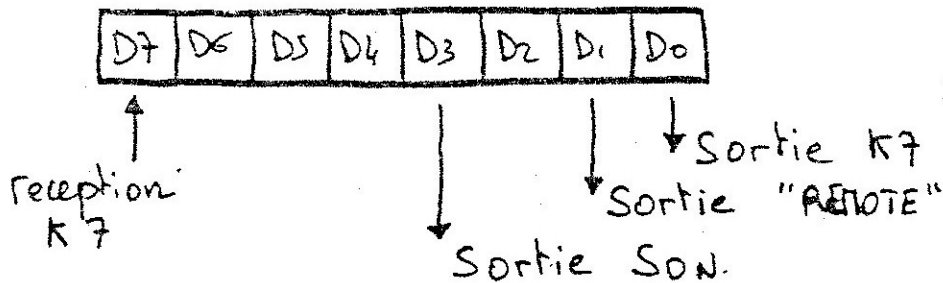


figure 11: organisation du bus de donnée dans les fonctions SON et K7

et de reception serie

VIII-2-3 Procédure de transmission de la donnée -

la transmission et la réception sont réalisées de façon série à un rythme de 1200 ou 2400 Bauds selon l'instruction Base demandée

les niveaux binaires "0" et "1" sont codés en fréquences (voir la figure 12)

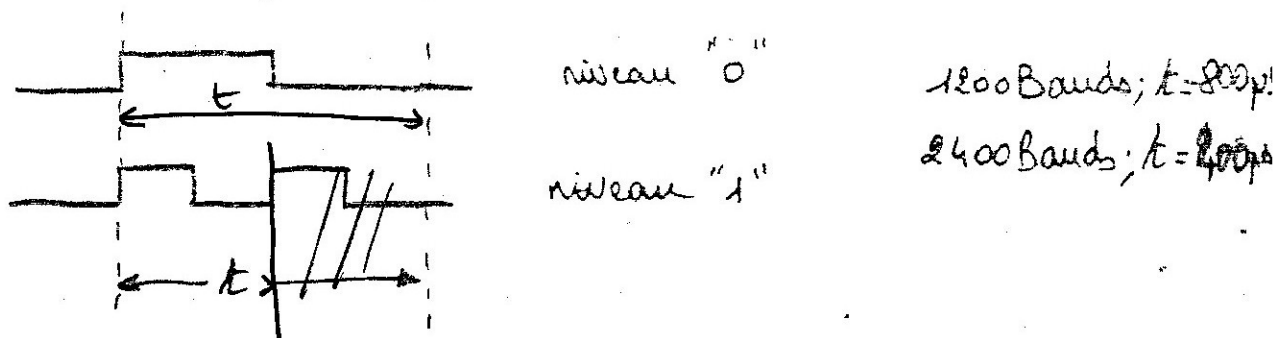


figure 12: codage des niveaux binaires par l'interface K7

VERS PLATINE PRINCIPALE

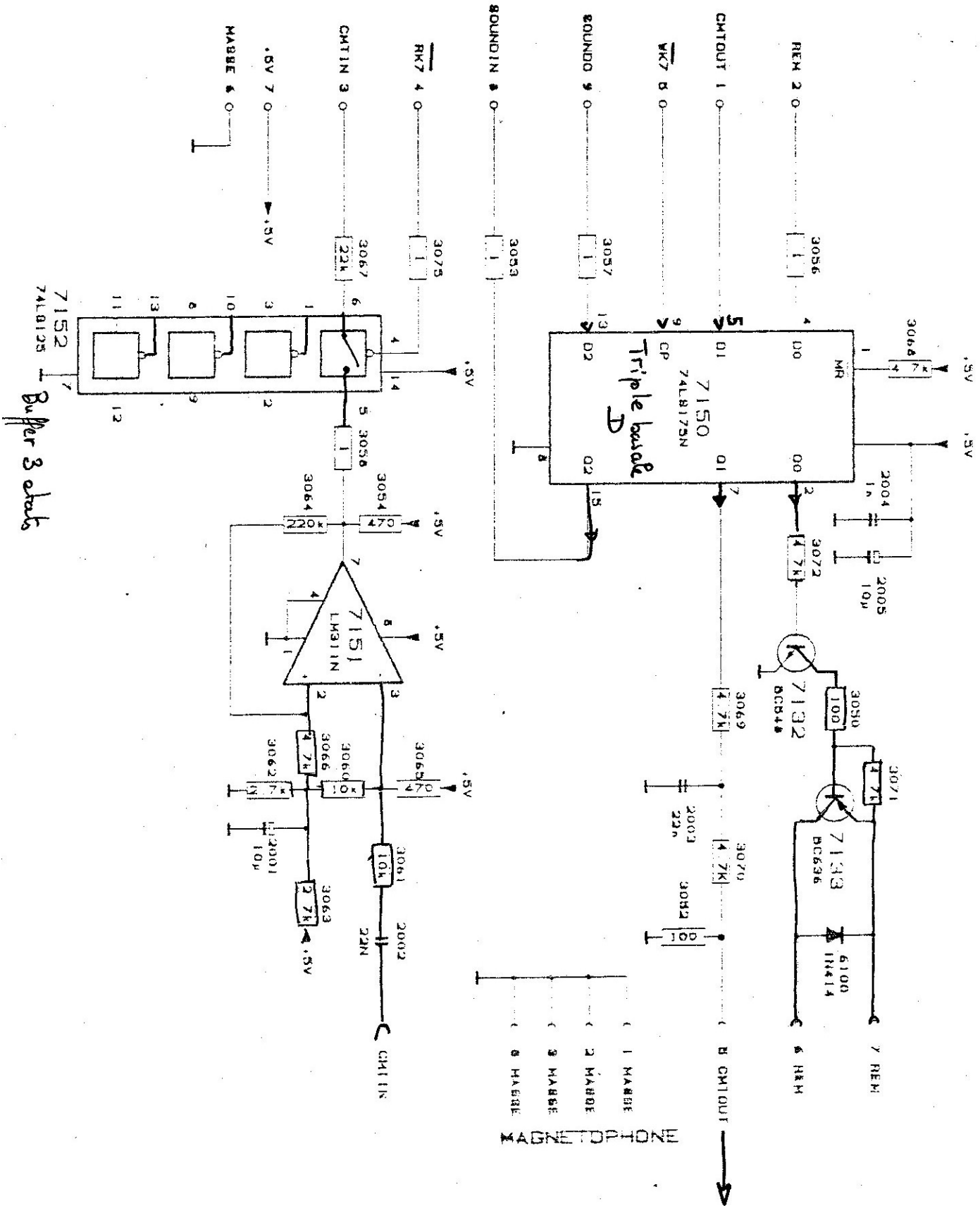


Figure 13: interface SON de Video

L'interface Video est principalement constituée des deux circuits repérés 7801 et 7803 -

7801 est un circuit de visualisation ^{conforme} aux normes de Videotex Européen - (EF 9345)

7803 est une mémoire RAM pseudo statique (4168) du même type que les RAMs utilisés par le Z80.

Le circuit EF 9345 reçoit une horloge d'origine quartz à 12MHz sur sa broche 12. un diviseur interne par 3 nous délivre du 4MHz sur la broche 11. Ce signal active notre bon vieux Z80, et n'attend qu'un utilisateur sur le connecteur d'extensions.

il nous faut considérer 3 types d'échanges.

- 2- entre le Z80 et le 9345
- 2- entre le 9345 et la mémoire RAM
- 1- entre le 9345 et le Téléviseur.

IX. 1. échanges entre le 9345 et le Téléviseur -

- le 9345 délivre des signaux R, V, B ainsi que les signaux de synchro TT et TL

Les lignes TT et TL ne génèrent aucun signal à la mise sous tension. En effet le 9345 comporte un registre interne de contrôle de base de Temps et ce registre doit être initialisé dès le début du programme.

Ce registre permet -

- l'adaptation 525 / 625 ligne
- le mode entrelacé ou non
- de valider ou pas les tops lignes et trames
- de générer sur TL directement une synchronisation composite

Dans le VG 5000 les signaux TT et TL délivrent les tops lignes et trames.

Les adaptateurs repérés 7810 et 7812 fabriquent le signal de synchro composite. Ce signal est mixé avec les signaux R, V, B grâce à un réseau de résistances dans la base du transistor 7900. Les adaptateurs 7813 délivrent R, V, B directement en Petite-télévision.

IV.2 échanges entre le Z80 et le 9345.

- le Z80 et le EF9345 communiquent ensemble via un bus 8bits multiplexé dans lequel une adresse précède une donnée. - le bus est en donnée; bidirectionnel -
- le Z80 doit donc; grâce au décodeur 7807; laisser croire au circuit de visualisation que son bus est multiplexé. Les lignes DS; R/W et AS issues du décodeur 7807 gèrent le bus multiplexé.
- au repos ;

DS = 1	DS = Data STROBE
R/W = 1	R/W = lecture écriture
AS = 0	AS = Address STROBE
- lorsque le Z80 adresse un registre interne au 9345; la ligne AS est le siège d'une impulsion positive (voir la fig 14)
- lorsque le Z80 génère une donnée vers le registre précédemment adressé; la ligne R/W est le siège d'une impulsion vers '0'.
- lorsque le Z80 consulte une registre de donnée du circuit video; la ligne DS est le siège d'une impulsion vers '0'

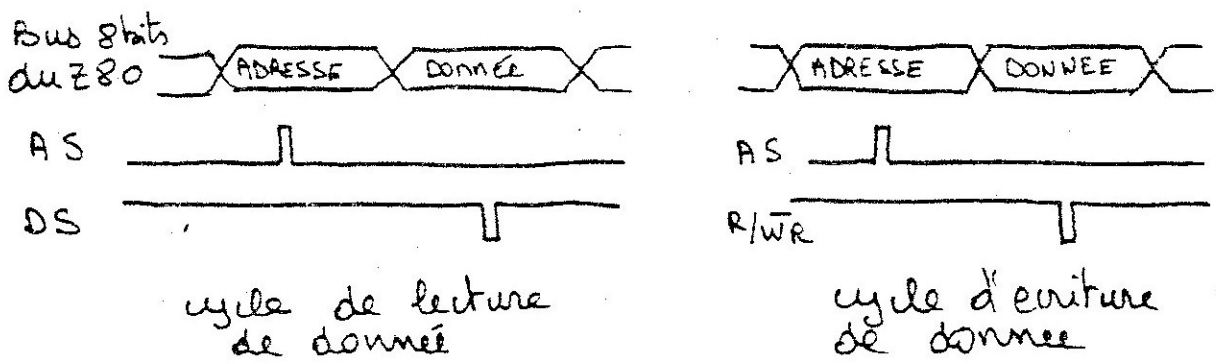


figure 14; échanges entre le Z80 et l'EF9345.

IX.3 - échanges entre le 9345 et sa RAM

- le bus adresse comporte 13 fils. L'EF9345 est donc en mesure d'avoir un dialogue sérieux avec une RAM pouvant aller jusqu'à 16Kx8.
- Dans le VG5000 le bit A13 n'est pas utilisé.

Notre 9345 est un circuit 40 broches - or une communication avec une RAM est gourmande en nombre de broches. afin de le réduire; la technique du multiplexage est là aussi utilisée.

les 8 bits de poids faibles (AD0 à AD7) peuvent être le siège d'une adresse sur le front descendant de \overline{ASM} (broche 4: Address select Memory) et d'une donnée en entrée ou sortie sur le front montant de \overline{ASM} . (figure 14)

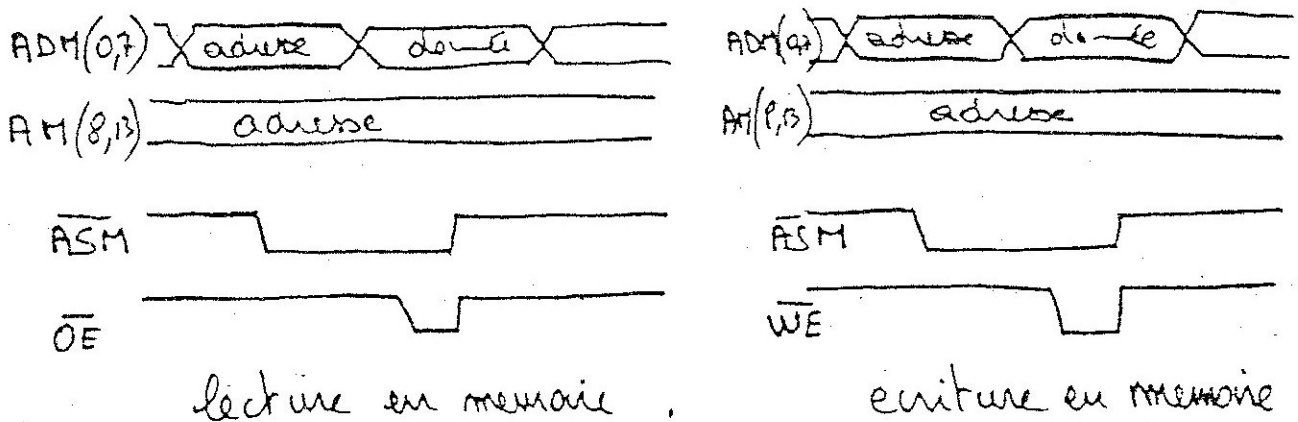


figure 14 échanges entre le 9345 et sa RAM

Que doit-on mesurer en fonctionnement normal sur ces différentes lignes de contrôle.

AS = 0 stable
 R/W = 1 stable
 DS = 1 stable

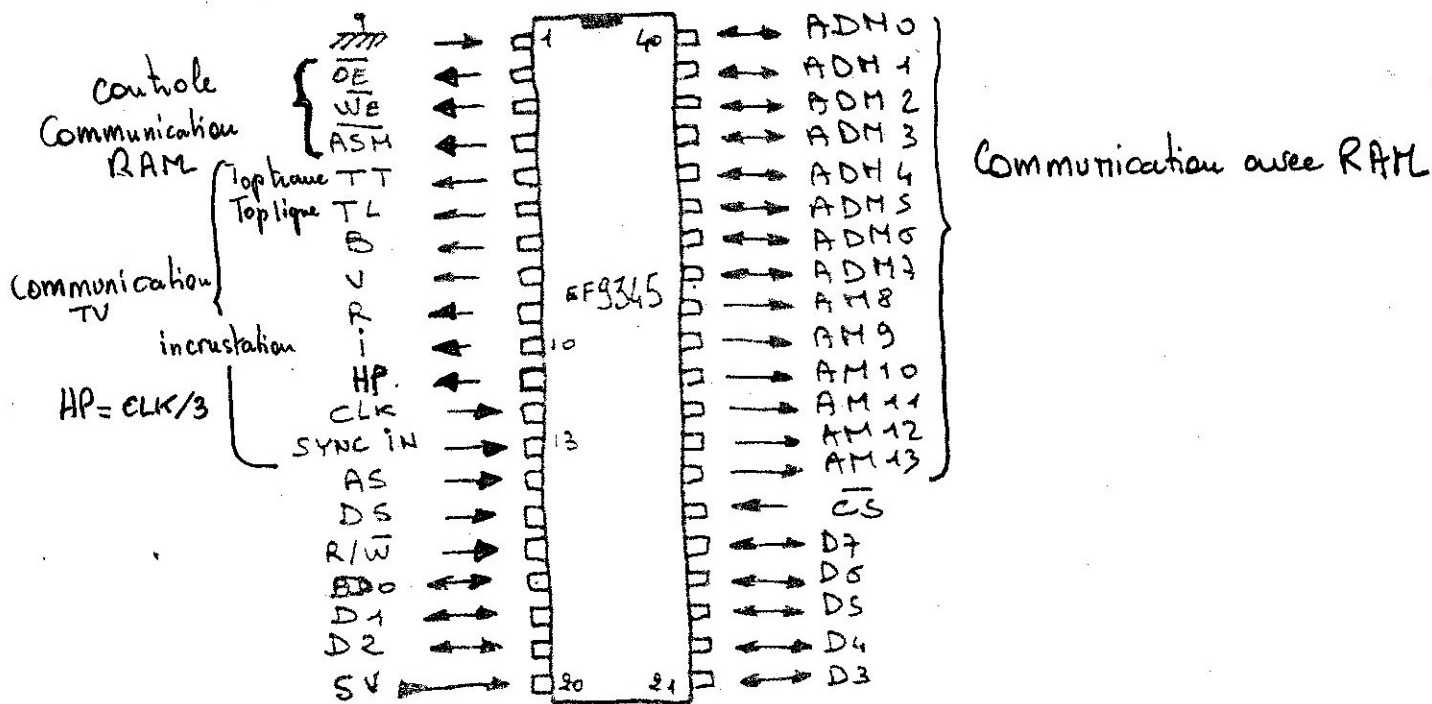
ces niveaux ne changent pas tant que l'affichage n'est pas sollicité.

\overline{ASM} } signaux de scintillation de la memoire d'ecran
 \overline{OE}

\overline{WE} = 1 . son niveau reste stable tant que l'affichage n'est pas modifié ce qui correspond alors à une écriture en memoire d'ecran.

Vous Trouvez dans l'additif n°4; l'explication plus

figure 15 : brochage du 9345 -



- D0 à D7 bus multiplexé en liaison avec le Microprocesseur
- AS = 1 indique une adresse sur le bus D0 à D7
- DS = 0 indique une donnée sortante du 9345
- R/W = 0 indique une donnée entrante dans le 9345
- TT sortie de synchro trame du TV
- TL sortie de synchro ligne du TV
- R, V, B Sorties couleurs
- I sortie d'incrustation du 9345 avec une autre image vidéo (I est contrôlée par programme)
- CLK entrée d'horloge : 12 MHz
- HP sortie 4 MHz
- SYNC IN entrée de synchro (VCO)
- CS validation du circuit
- AM0 à AM7 bus multiplexé en liaison avec la RAM
- AM8 à AM9 bus adresse parallèle -
- Adresse select memory. → ASM $\begin{cases} \overline{E} & \text{indique en (AM0 à AM7) une adresse} \\ \overline{S} & \text{indique en (AM0 à AM7) une donnée} \end{cases}$
- \overline{OE} lecture dans la RAM → output enable
- \overline{WE} écriture en RAM → write enable

X - le connecteur d'extensions

- le VG 5000 peut recevoir différents accessoires
- au extensions au travers d'un connecteur
50 contacts prévus sur l'arrière de l'appareil -
(voir la figure 16)
- le schéma des prises arrières est représenté en
figure 17.

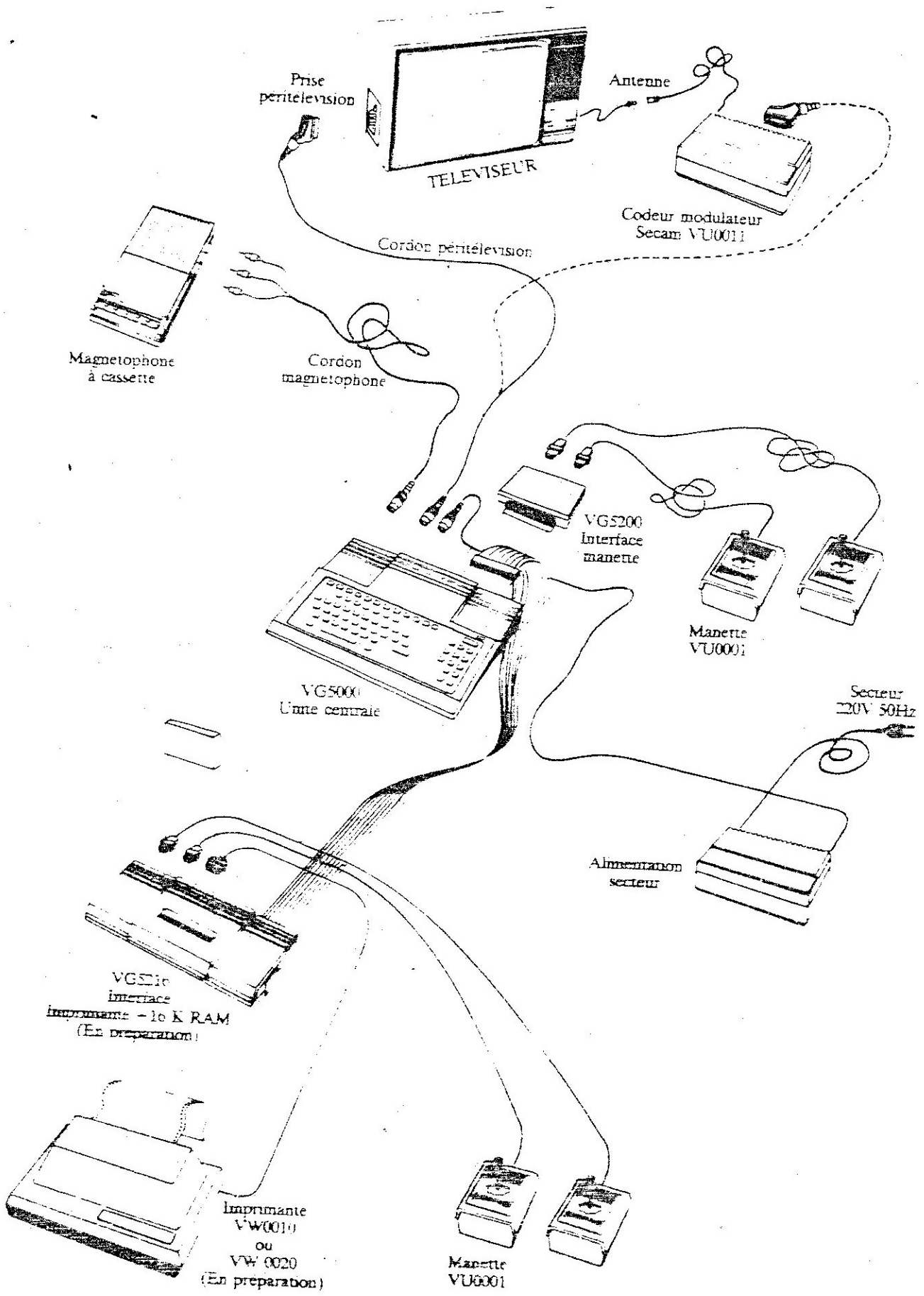
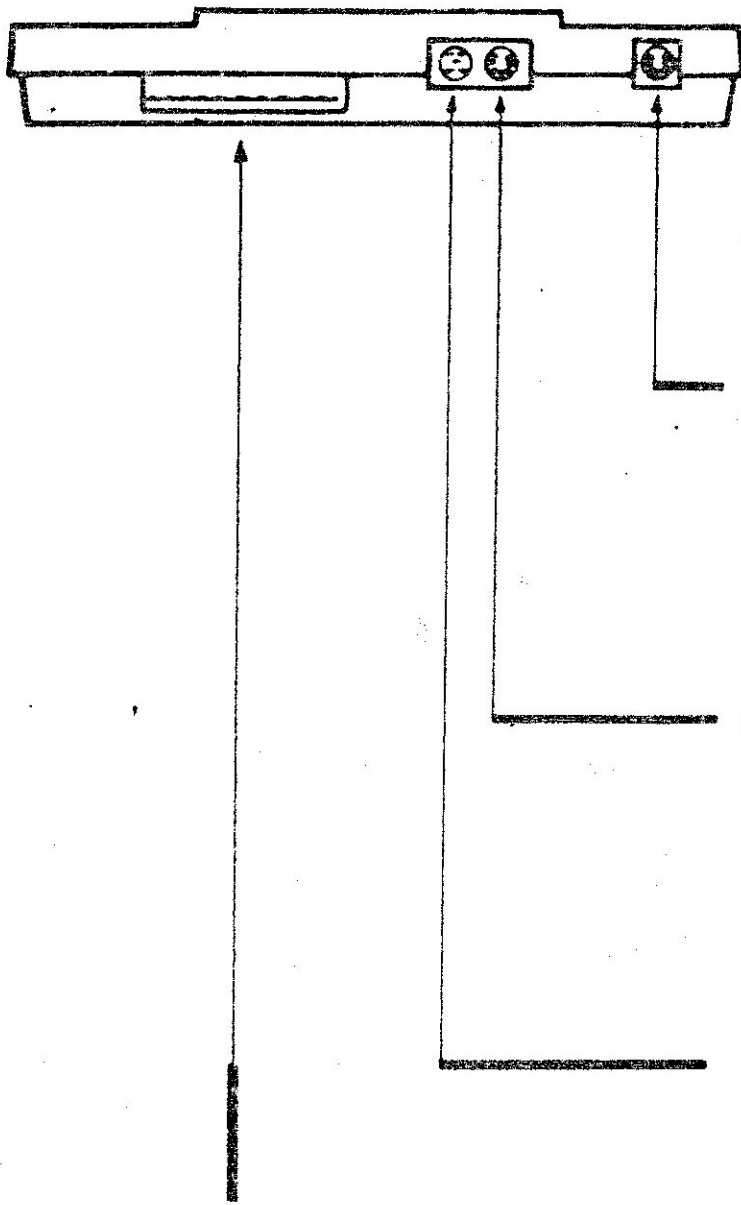


figure 16 - configuration VG5000



Prise magnétophone

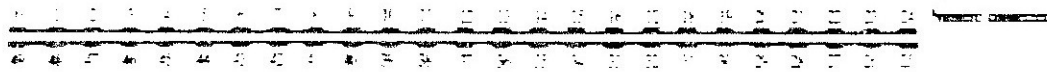
- 1 masse
- 2 masse
- 3 masse
- 4 rouge (micro)
- 5 blanc (casque ou ligne)
- 6 noir (télécommande)
- 7 masse
- 8 masse

Prise péritélévision

- 1 Commande rapide
- 2 Masse
- 3 Bleu
- 4 Sync.
- 5 Rouge
- 6 Commande lente
- 7 BF
- 8 Vert

Prise alimentation

- 1 NC
- 2 -12 V
- 3 -5 V
- 4 -12 V
- 5 Masse



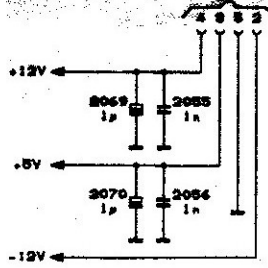
Connecteur bord de carte

0 <u>RFSH</u>	7 <u>Sound Ex</u>	14 <u>D₂</u>	21 <u>A₄</u>	28 <u>A₅</u>	35 <u>D₃</u>	42 <u>INTEN</u>	49 <u>WAITE</u>
1 <u>NC</u>	8 <u>MI</u>	15 <u>D₀</u>	22 <u>A₂</u>	29 <u>A₇</u>	36 <u>D₁</u>	43 <u>CE₁</u>	
2 <u>+5V</u>	9 <u>WR</u>	16 <u>A₁₄</u>	23 <u>A₁</u>	30 <u>A₆</u>	37 <u>D₄</u>	44 <u>CE₂</u>	
3 <u>+5V</u>	10 <u>IORKO</u>	17 <u>A₁₂</u>	24 <u>-12V</u>	31 <u>A₁₁</u>	38 <u>MREO</u>	45 <u>CSROM</u>	
4 <u>CE₃</u>	11 <u>Reset</u>	18 <u>A₁₀</u>	25 <u>+12V</u>	32 <u>A₁₀</u>	39 <u>RD</u>	46 <u>Masse</u>	
5 <u>CE₂</u>	12 <u>D₆</u>	19 <u>A₈</u>	26 <u>A₁</u>	33 <u>A₁₃</u>	40 <u>RFSH</u>	47 <u>Masse</u>	
6 <u>CE₁</u>	13 <u>D₄</u>	20 <u>A₆</u>	27 <u>A₃</u>	34 <u>D₁</u>	41 <u>Horl.</u>	48 <u>NC</u>	

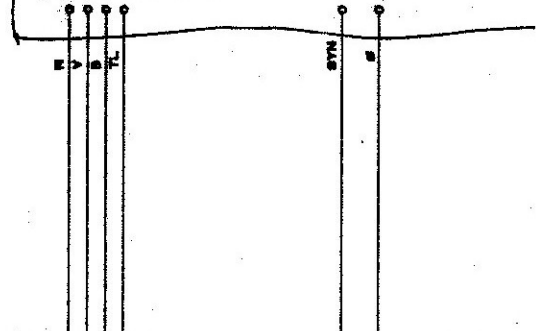
figure 17 détail des prises arrières du VG 5000

REFRENC	TYPE	.5V	L	I
7810	N74LS04N	14	7	2062 1n
7811	74LS04N	14	7	2064 1n
7812	SN74LS00N	14	7	2068 1n
7815	7407N	14	7	2078 1p

ALIMENTATION



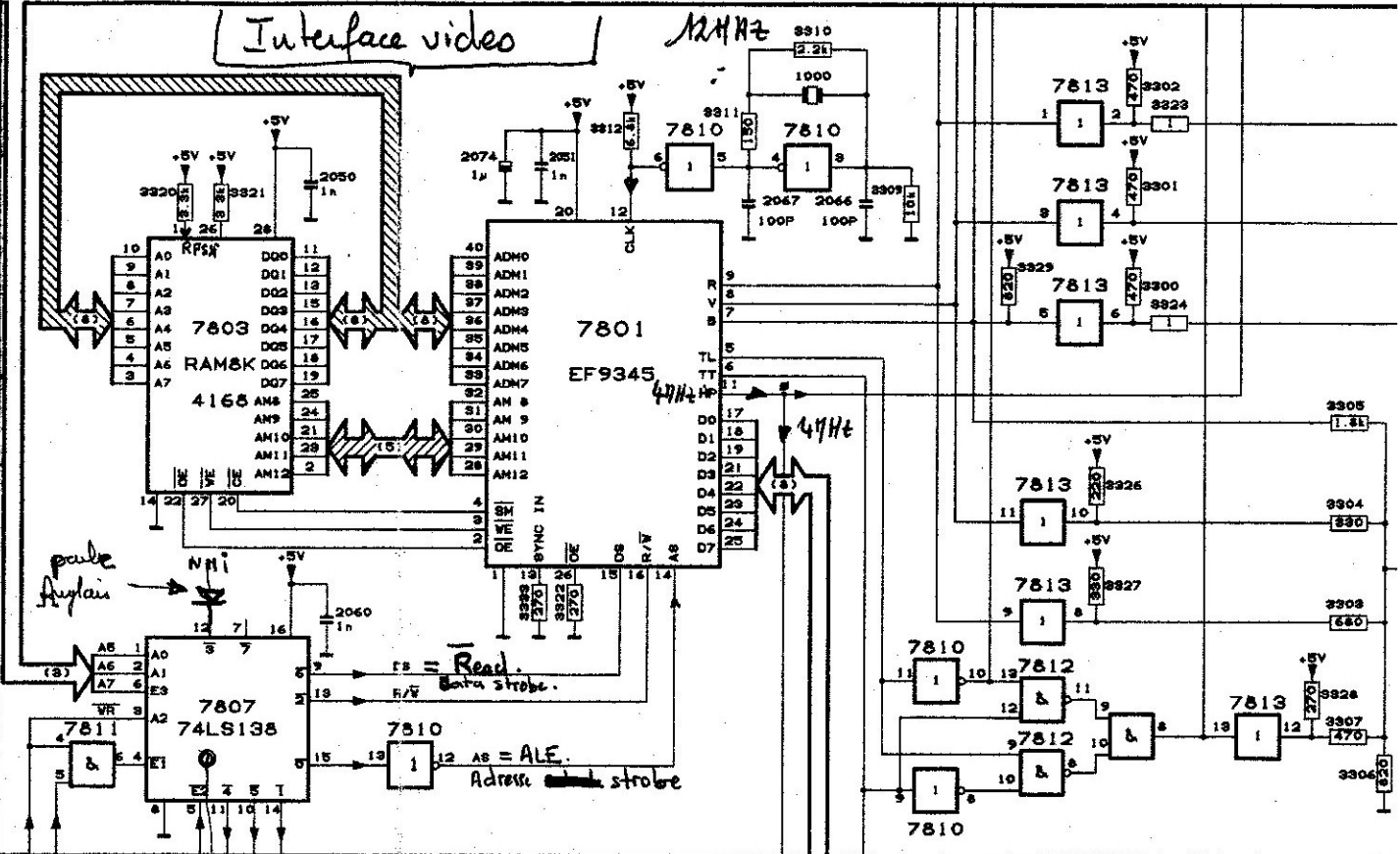
Connecteur Extension



BUS D'ADRESSES

Interface video

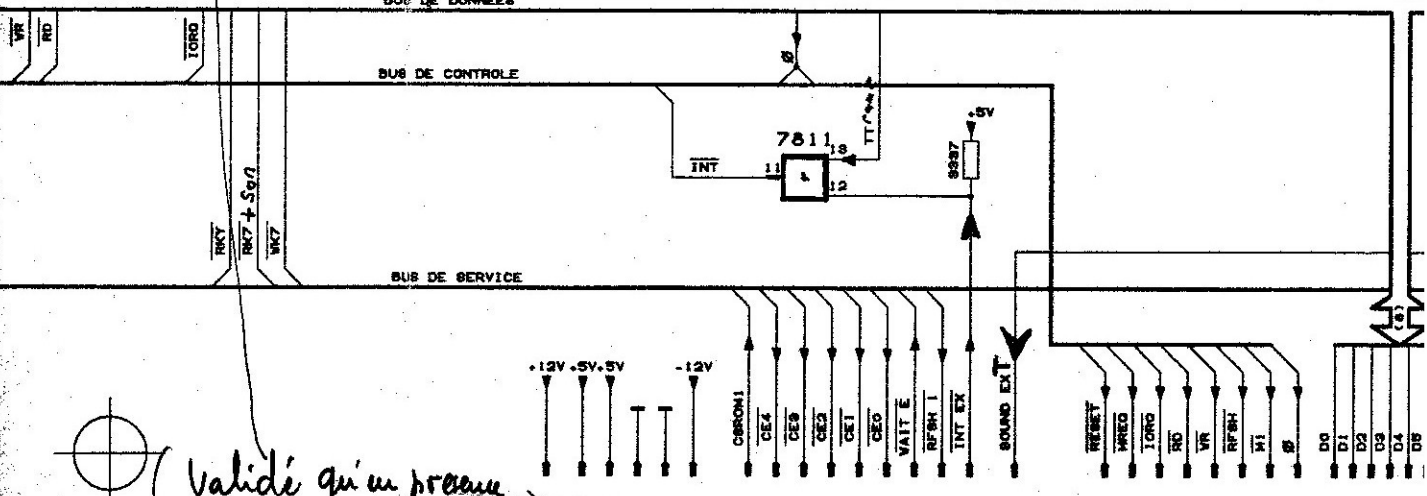
12MHz



BUS DE DONNEES

BUS DE CONTROLE

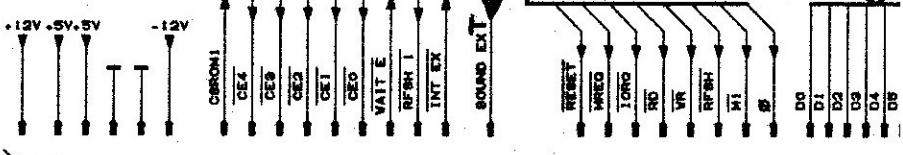
BUS DE SERVICE



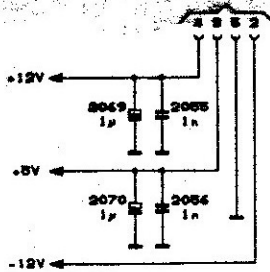
Validé qui en présence de IORG.
 permet de rendre compatible le 7800 et le EF9345

Alim. file controle

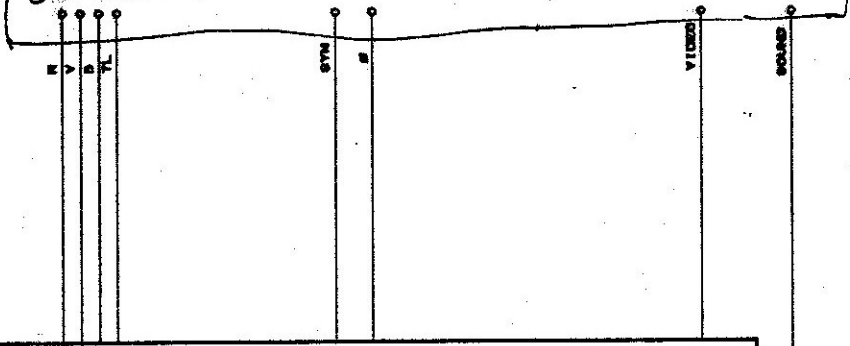
CONNECTEUR D'EXTENSION



ALIMENTATION



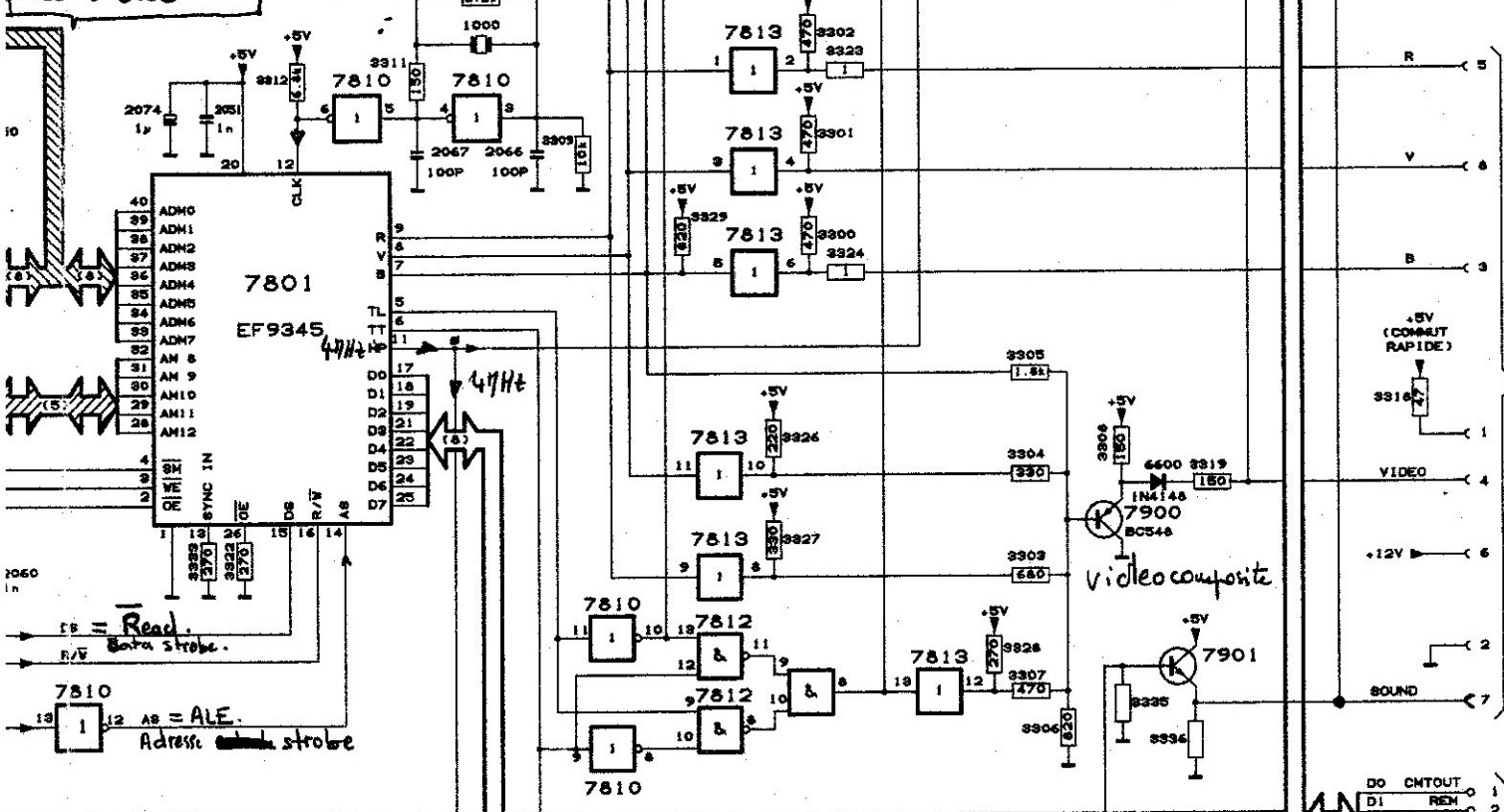
Connecteur Extension Modulateur PAL.



BUS D'ADRESSES

face video

12MHz



PRISE PERITELEVISION

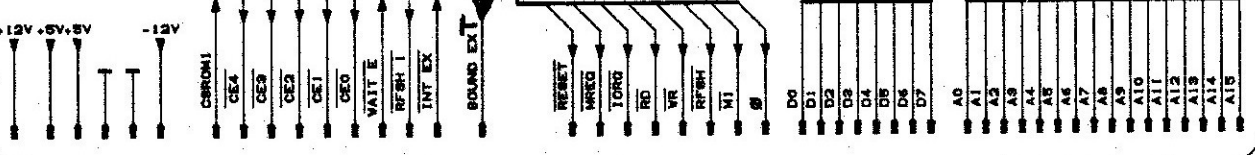
VERS INTERFACE CASSETTE ET SON.

BUS DE DONNEES

BUS DE CONTROLE

BUS DE SERVICE

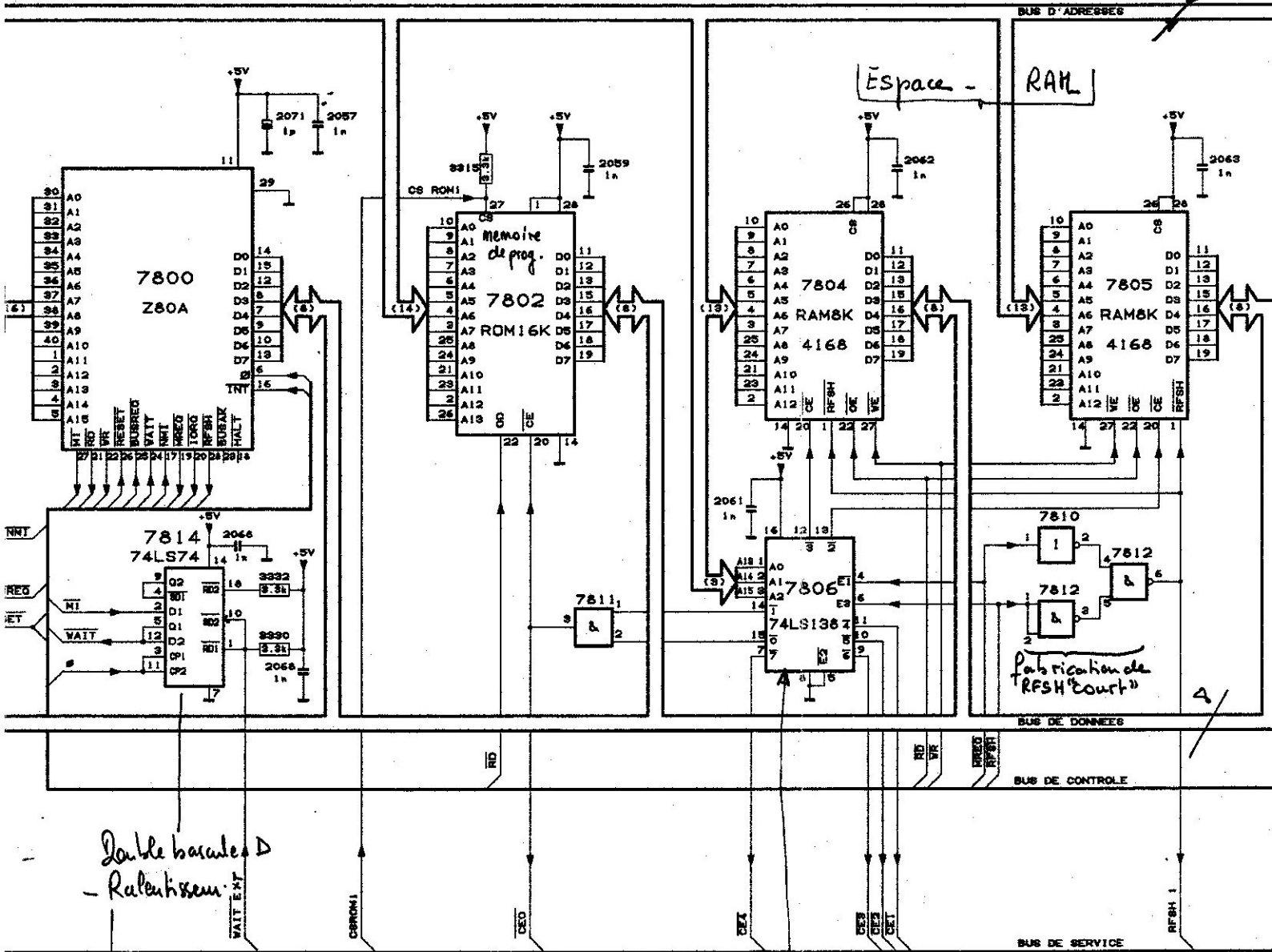
à un presse



Alim. fils controle CONNECTEUR D'EXTENSION Donnée Adresse

rendre de le 7800 9345

SCHEMA DE PRINCIPE DE L'UNITE CENTRALE.



16

Espace - RAM

fabriquant de RFSH court

4

Double barade D
- Ralentisseur

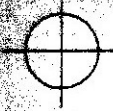
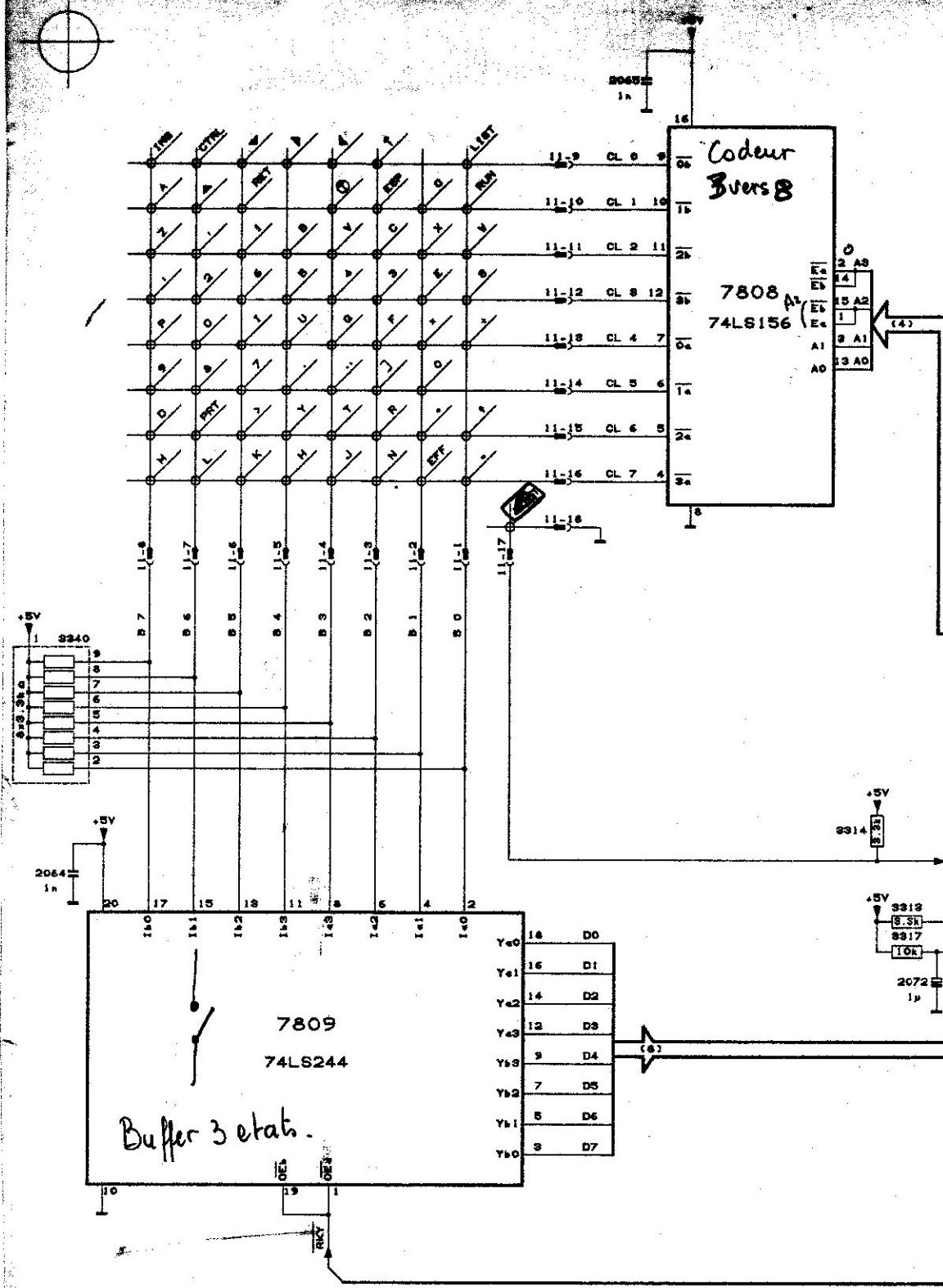
$\mu P \rightarrow$ ROM ralentit le Z80 pour adapter la ROM.
Cycle M1 4 phase Horloge - $1 \mu S$

gare au ralentisseur. $1,250 \mu S$

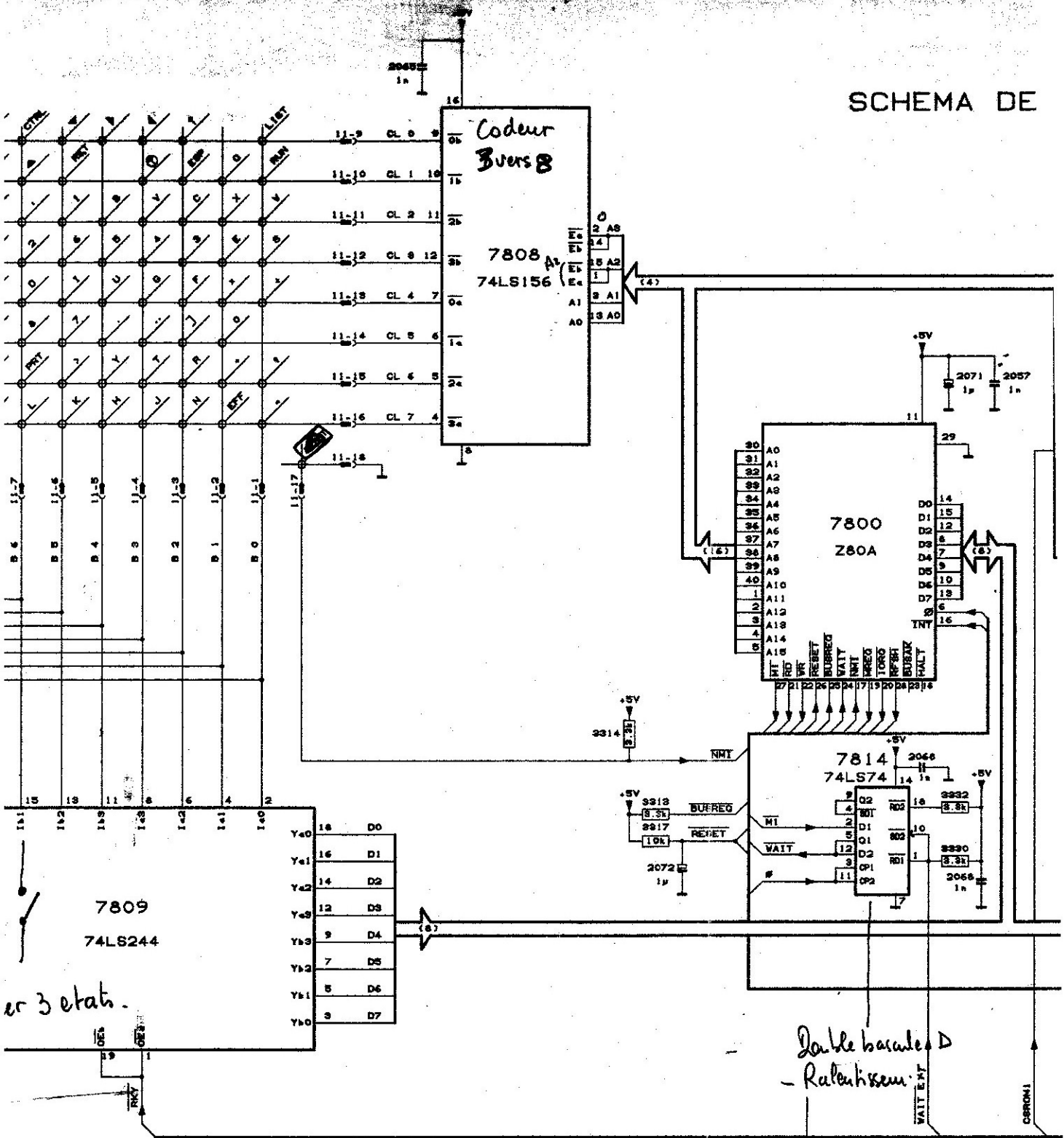
~~XXXXXXXX~~

(Codeur 3 vers 8.
Validé qu'en presence de Q.





SCHEMA DE



er 3 etat.

Double bascule D
- Ralentisseur

µP → ROM ralentisseur Z80
adapté au ROM.
Cycle M1 4 phase Hold

gare au ralentisseur. 1,250 µS

~~XXXXXXXX~~

I/OREQ → input output Request

MREQ → memory request

NMI → interruption non masquable

HALT → Wait ou interrupt

RFSH → rafraichissement de memoire dynamique

M1 → c'est une sortie lorsque M1 passe à 0 780 va chercher l'octet dans la memoire programme.

WAIT → est une entree permit lorsque la ligne est à zero d'arrêter le µP. Cas de dialogue entre un imprimante et µP.

BUSACK → } permet de travailler en DMA . Direct memory access

BUS. Req (pas utilisé sur VG5000)

↑ mettre Bus Req à la remise en Hz le Bus de données + Bus d'adresse.



Quelques mots sur le Z80.

le Z80 constitue une unité centrale 8 bits, disponible en plusieurs versions selon les vitesses d'horloge.
(voir le Tableau 1)

le Z80 a un jeu de 158 instructions dont 78 sont compatibles 8080A - certains codes instructions nécessitent deux octets.

L'alimentation est unique (5V ± 5%) pour une consommation allant de 30 à 200mA.

plus rapide. →

TYPE	horloge
Z80	2,5 MHz
Z80A	4 MHz
Z80B	6 MHz
Z80H	8 MHz
Z80L	1 MHz

Tableau 1:
vitesse de travail
des différents types
de Z80

Description des broches du Z80 (voir la fig 2)

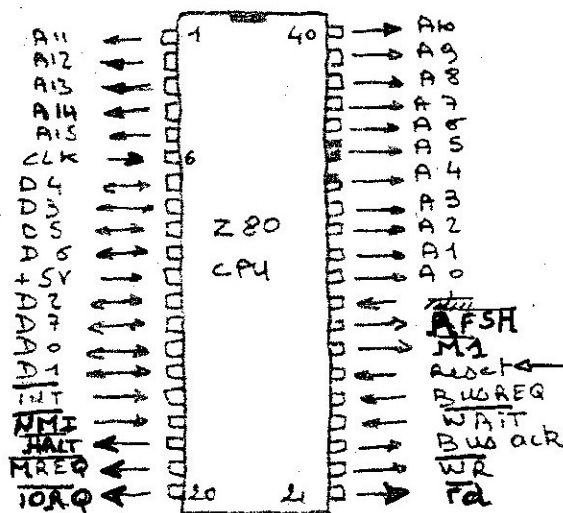


figure 2: brochage
du Z80

interruption prioritaire

- bus de donnée de 8 bits ; bidirectionnel, 3 états (D0 à D7). c'est par ces lignes que les échanges s'effectuent entre le Z80, les mémoires et les périphériques du système
- bus adresse de 16 bits (sortie). Ce bus comporte trois fonctions.
 - sur 16 bits ; il pointe une adresse sur 65536 en mémoire (ROM ou RAM)
 - sur 8 bits (LSB) il pointe un système d'entrée-sortie sur 256
 - sur 7 bits (LSB) il pointe une rangée d'adresses à rafraîchir dans une RAM dynamique

la bus de contrôle

- Trois lignes du bus de contrôle permettent de définir la fonction présente du bus adresse
 - $\overline{\text{MAEQ}}$ (Memory Request: Requête en Mémoire)
Cette broche à "0" nous indique que le bus adresse pointe une case mémoire sur 65536
 - $\overline{\text{IORQ}}$ (INPUT-OUTPUT Request: Requête en Entrée-Sortie)
Cette broche à "0" nous indique que le bus adresse pointe un périphérique sur 256
 - $\overline{\text{RFSH}}$ (Refresh: rafraichissement). Cette broche à "0" nous indique que le bus adresse pointe une rangée sur 128 de cases mémoire à rafraichir en RAM dynamique.
- Les lignes $\overline{\text{rd}}$ et $\overline{\text{wr}}$ (read et write) nous indiquent le sens des échanges sur le bus de donnée -
le sens est vu par rapport au micro processeur -
 $\overline{\text{rd}} = 0 \rightarrow$ entrée
 $\overline{\text{wr}} = 0 \rightarrow$ sortie
- Les lignes $\overline{\text{INT}}$ et $\overline{\text{NMI}}$ (interrupt et NON Masquable interrupt) constituent les deux entrées d'interruption -
 - $\overline{\text{NMI}}$ à "0" branche le programme à l'adresse 0066H du Z80
 - $\overline{\text{INT}}$ à "0" Dans le V65000, branche le programme à l'adresse 0038 (H) (MODE 1 d'interruption du Z80). ATTENTION cette fonction peut être masquée par le programme -
- la ligne $\overline{\text{reset}}$ à "0" initialise le programme à l'adresse 0000 du Z80.
- la sortie M1 (Premier cycle) échantillonne le cycle de recherche d'instruction en mémoire de programme (voir la fig 3)
- l'entrée wait: (attente) A chaque cycle M1 et à chaque cycle d'exécution d'une instruction d'entrée-sortie, l'unité centrale écoute son entrée "WAIT". Rien ne change si "WAIT" est au niveau haut. Par contre si WAIT est au niveau bas, le Z80 se place en position d'attente jusqu'à ce que "WAIT" repasse au niveau haut -
cette entrée permet de synchroniser le Z80 par rapport à un périphérique, plus lent que lui, qui le demande,

- la Sortie "HALT" Tombe à "0" lorsque le Z80 exécute l'instruction software HALT (76 (H)) cette fonction porte un autre nom; "Attente sur interruption". En effet; NMI ou Reset permet de redemander le programme.
- L'entrée Busreq (Bus request: requête de bus) permet à un périphérique qui le demande de travailler en DMA (Direct Memory access: accès direct en mémoire). Le Z80 indique qu'il a bien reçu l'ordre Busreq par une réponse sur sa ligne BUSACK (BUS ACKNOWLEDGE: accusé de réception de Bus request). Le Z80 place alors ses bus de données, d'adresse, et certains fils du bus de contrôle en Haute impédance. Ainsi le périphérique peut maintenant réaliser directement sans passer par le Z80 tous les échanges voulus. Pour "redonner la main" au Z80 il faudrait replacer la ligne Busreq au niveau haut.
- n'oublions pas l'alimentation de 5V; la masse et l'entrée d'horloge.

Se Souvenir:

Le bus de contrôle nous indique le travail que le Z80 exécute

- \overline{IORQ} à "0" → le Z80 s'adresse à une entrée-sortie sur 256
- \overline{MREQ} à "0" → le Z80 s'adresse à sa mémoire
- \overline{RFSH} à "0" → le Z80 rafraîchit sa RAM dynamique
- $\overline{M1}$ à "0" → le Z80 va chercher l'instruction en ROM
- \overline{RD} à "0" → la donnée entre -
- \overline{WR} à "0" → la donnée sort -
- \overline{HALT} à "0" → le Z80 est en attente d'interruption -

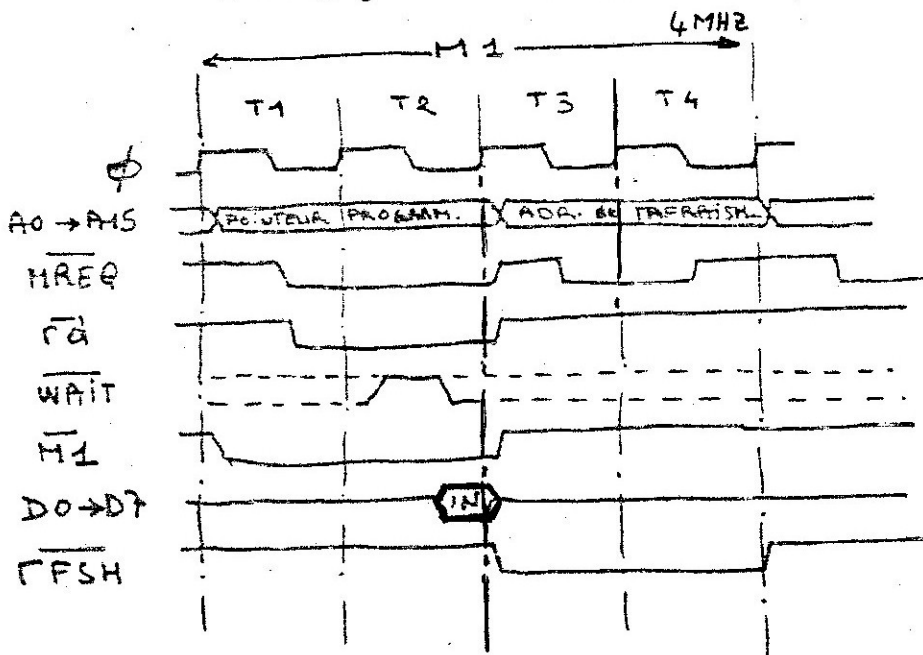


Figure 3:
 oscillogrammes
 du cycle M1
 : recherche du
 code instruction.

organisation de la RAM du V65000 (figure 8)

la mémoire RAM est découpée en différentes parties

- une mémoire écran vue par le Z80
 - une zone de pointeurs
 - une zone correspondant au texte du programme rédigé en langage basic
 - une zone dans laquelle sont sauvegardées toutes les variables utilisées dans le programme basic
 - la pike du Z80
 - une zone définie par l'instruction "clear" du programme basic dans laquelle sont rangés les chaînes de caractères.
- la mémoire écran, vue par le Z80 occupe 2000 octets géographiques entre les adresses 4000(H) et 47CF(H). Cet ensemble est constitué de 1000 blocs de 2 octets chacun. Un octet définissant le caractère à afficher ; 1 octet définissant ses attributs. Le caractère est codé ASCII ; l'attribut est structuré selon les caractéristiques de l'interface vidéo ; le circuit EF 8345 -

les adresses 4000(H) et 4001(H) gèrent la cellule située en haut à gauche de l'écran
les adresses 47CF(H) et 47CF(H) gèrent la cellule située en bas à droite de l'écran -

le programme d'application suivant permet de charger une paire d'octets dans la RAM, dans le cas présent ; aux adresses 4002 et 4003 les variables "C" et "A" y sont respectivement chargées comme "caractère" et son "attribut"

41 = code ASCII du "A"

00 = (entre autre) couleur Noire

exemple : pour afficher un "B" rouge ; charger
C = 42 et A = 01

NOTES : _____


```

10 REM:-----INITIALISATION-----
20 PRINT CHR$(31):PRINT :PRINT
30 TX4,0,0:PRINT :PRINT :PRINT
40 PRINT "ECRITURE DANS LA MEMOIRE ECRAN"
50 PRINT "VUE PAR LE -Z80-"
60 REM:-----
70 :
80 :
90 :
100 C="41": REM C=CODE CARACTERE
110 A="00": REM A=CODE ATTRIBUT
120 POKE &"4002",C:REM PLACE LE CARACT.
130 : REM EN MEMOIRE ECRAN
140 POKE &"4003",A:REM PLACE L'ATTRIBUT
150 : REM EN MEMOIRE ECRAN
160 PRINT :END

```

Remarque : le VG 5000 utilise 2 mémoires d'écran -

- une mémoire écran gérée par l'éditeur donc vue par le Z 80
- une mémoire d'écran affichée grâce au circuit EF 9345 - la seconde est renversée à jour par la première -

- la zone de pointeurs est comprise entre les adresses 47D0(H) et 49FB(H) . c'est ici que sont pointées par paires d'octets les adresses clé du système .

exemple : les cases 4895 et 4896 pointent la pile du Z80
les cases 488E et 488F pointent le début du Texte du programme basic (voir la fig 18)

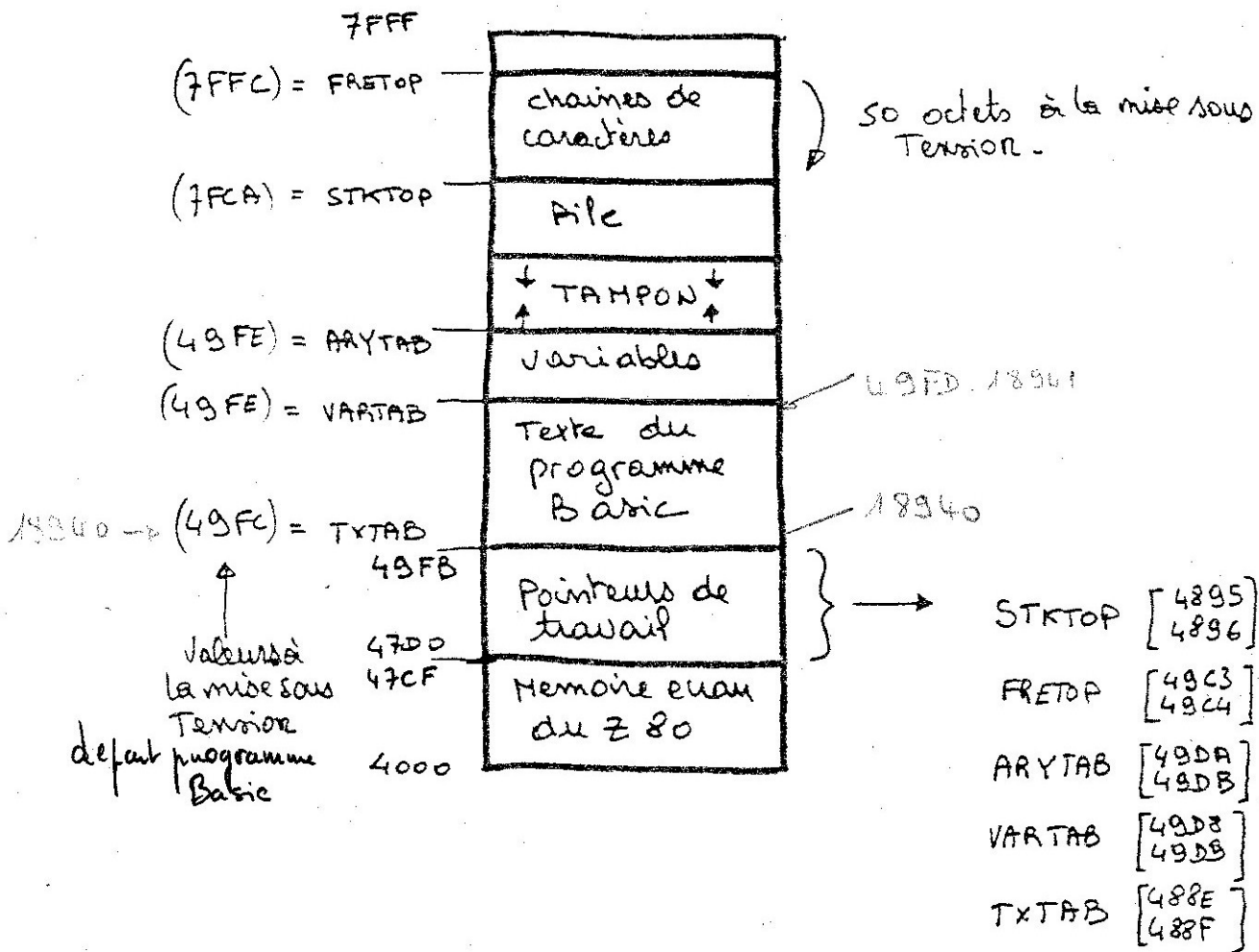
- programme d'application sur la mémoire d'écran -

```

10 REM "*****"
20 REM "* MANIPULATION SUR LA *"
25 REM "* MEMOIRE D'ECRAN : *"
30 REM "* ++++++"
35 REM "* + FABRICATION D'UNE + *"
40 REM "* + MIRE DE BARRES + *"
45 REM "* ++++++"
50 REM "* MOI/DFT SEPT/85 *"
55 REM "*****"
56 :
57 :
58 :
80 PRINT "UN INSTANT SVP. "
100 FOR L=16384 TO 18304 STEP 80
110 FOR M=0 TO 79 STEP 10:READ C
120 FOR N=0 TO 9 STEP 2
130 POKE (L+M+N),127:POKE (L+M+N+1),C
140 NEXT N:NEXT M:RESTORE
150 NEXT L:PRINT :END
200 DATA 7,3,6,2,5,1,4,0

```


Figure 18: Organisation de la RAM du V65000.



Remarque: 50 octets sont attribués d'office à la chaîne de caractères dès la mise sous Tension $(7FFC_{(H)} - 7FCA_{(H)} = 32_{(H)})$

L'instruction Basic CLEAR I, J modifie les pointeurs FRETOP et STKTOP.

exemple (1) CLEAR 250 → FRETOP = 7FFC
STKTOP = 7F0E } 250 octets

(2) CLEAR 252, 7000 → FRETOP = 7000
STKTOP = 6904 } 252 octets

L'instruction Basic FRE (0) donne le nombre d'octets disponibles entre ARYTAB et le bas de la pile.

le programme rédigé en langage Basic est chargé à partir de TXTAB (pointeur Fixe) jusqu'à VARTAB (Fin du programme ; Debut des Variables). Les Variables (6 octets par variable) sont stockées entre VTAB et ARYTAB - les chaînes de caractère sont stockées à partir de FRETOP jusqu'à STKTOP. C'est donc l'instruction CLEAR, qui définit : la butée haute de la RAM occupé par l'interpréteur Basic ; et la place réservée aux chaînes de caractères.

exemple (1). CLEAR &"FA" réserve FA(H) octets en chaîne de caractères (FA(H) = 250(10))
 FRETOP = 7FFC (H)
 STKTOP = 7F02 (H)

(2) CLEAR &"FA, &"7000" réserve FA(H) octets en chaîne de caractères. la butée haute FRETOP est placée à 7000 (H)
 FA(H) = 250(10) ; 7000(H) = 28672(10)
 FRETOP = 7000 (H)
 STKTOP = 6902 (H)

```

10 PRINT CHR$(31):TX1,0,0
20 PRINT :PRINT :PRINT
30 PRINT " ";
40 PRINT "-----"
50 PRINT " ";
60 PRINT "MANIPULATION SUR LES POINTEURS"
70 PRINT " ";
80 PRINT " 'FRETOP'=HAUT DE RAM "
90 PRINT " ";
100 PRINT " 'STKTOP'=HAUT DE PILE"
110 PRINT " ";
120 PRINT "(FRETOP-STKTOP=espace chaine)"
125 PRINT " ";
130 PRINT "-----"
140 PRINT
150 PRINT
160 TX4,0,0
170 REM--INITIALISATION DES POINTEURS--
180 INPUT "ENTREZ L'ESPACE CHAINE ":";A
190 PRINT
200 INPUT "ENTREZ 'FRETOP' (en decimal)";B
210 CLEAR A,B
220 PRINT :PRINT :PRINT :TX0,0,0
1000 PRINT " "":PRINT "FRETOP=";
1010 C=PEEK("&"49C4"):M=C:GOSUB 10000
1020 C=PEEK("&"49C3"):N=C:GOSUB 10000
1030 GOSUB 15000
1050 PRINT " "":PRINT "STKTOP=";
1060 C=PEEK("&"4896"):M=C:GOSUB 10000
1070 C=PEEK("&"4895"):N=C:GOSUB 10000
1080 GOSUB 15000
1090 END
1101 PRINT
1102 PRINT
1103 PRINT
10000 REM-----
10010 REM CALCUL DECIMAL ==> HEXA
10020 REM-----
10021 :
10030 X=INT(C/16):X=X+48
10040 IF X>57 THEN X=X+7
10050 PRINT CHR$(X);
10060 X=((C/16)-INT(C/16))*16
10070 X=X+48:IF X>57 THEN X=X*7
10080 PRINT CHR$(X);
10090 RETURN
14998 :
14999 :
15000 PRINT"(H)";:TX1,0,0
15010 PRINT" (";M*255+N;")"
15020 PRINT:PRINT:TX0,0,0
15030 RETURN

```

Manipulation sur la sortie son.

- le premier programme, rédigé en langage Basic, manipule directement le Z80 en ligne 50 grâce à l'instruction "CALL". Le programme Z80 a été préalablement chargé grâce à une pile de DATA et l'instruction "poke" en ligne 40.
- le second programme n'est rien d'autre que la pile de data du premier programme; mais rédigée maintenant en langage Z80 de base assembleur -

NOTES personnelles: _____

```

4 REM -----
5 REM   MANIPULATION SUR LA
6 REM   SORTIE SON
7 REM -----
8 :
9 :
10 CLEAR (&"FF"), (&"6FFF")
20 LET A=&"7000" : REM A=DEBUT
30 FOR I=0 TO 33:READ D:REM CHARGE LE
40 POKE A+I,D:NEXT I :REM PROGRAMME
50 CALL &"7000" :REM EXECUTE
55 REM
60 REM LISTING DU PROGRAMME EN DATA
65 REM-----
68 DATA&"F3", &"3E", 0, &"D3", &"AF", &"3E"
70 DATA1, &"CD", &"16", &"70", &"3E", 8
72 DATA&"D3", &"AF", &"3E", 1, &"CD", &"16"
74 DATA&"70", &"C3", 1, &"70", &"57", &"1E"
76 DATA&"F8", &"1D", &"C2", &"19", &"70"
78 DATA&"15", &"C2", &"17", &"70", &"C9"
80 REM-----

```

```

-----
PROGRAMME D'APPLICATION SUR
LA SORTIE SON DU VG5000
DFT/MCI SEPT/84
-----

```

```

7000          F3          DI          :INHIBE INTERRUPT
7001 DEBU 3E 00          LD  A, 00      : MET LA LIGNE
7003          D3 AF       OUT A, (AF)   : SON A ZERO
7005          3E 01       LD  A, 01      : TEMPORISATION
7007          CD 16 70    CALL TEMP     : DE 1 MILLISEC
700A          3E 08       LD  A, 08      : MET LA LIGNE
700C          D3 AF       OUT A, (AF)   : SON A '1'
700E          3E 01       LD  A, 01      : TEMPORISATION
7010          CD 16 70    CALL TEMP     : DE 1 MILLISEC
7013          C3 01 70    JP   BEBU     : RETOUR DEBUT

7015 TEMP 57          LD  D, A          : D=PRODUIT
7017   B1 1E F8       LD  E, F8        : E=RESOLUTION
7019   B2 1D          DEC  E           : DE UNE
701A          C2 19 70  JNZ  B1         : MILLISECONDE
701D          15        DEC  D           :
701E          C2 17 70  JNZ  B2         :
7021          C9        RET              : FIN DE TEMPO

```

L'interface video -

- Ce generateur d'image video est un circuit programmable commande par un microprocesseur. JP est utilise avec une memoire RAM pouvant aller de $2k \times 8$ jusqu'a $16k \times 8$.
- Caracteristiques generales.
 - Affichage par cellules
 - Compatibilite 50 Hz / 60 Hz (25 ou 21 rangees)
 - Deux formats d'ecran (40 ou 80 colonnes)
 - memoire de caracteres (alphanumerique et graphique) interne aux normes Teletex
 - memoire de caracteres d'extension possible
 - attributs en mode 40 colonnes.
 - Couleur caractere, couleur du fond, double hauteur (et/ou) largeur, clignotement, inversion video, souligne, amulation, uration, accentuation.
 - attributs en mode 80 colonnes.
 - souligne, clignotement, inversion, couleur.
 - Systeme de synchronisation souple : entrelace ou non composite ou non VCO interne
 - Scholling, affichage ou non du curseur.

Schema bloc du 9345

en figure 19. on distingue

- 8 registres adresables directement par le microprocesseur (R_0 a R_7)
- 5 registres de controle adresables indirectement (AOR, DOR, PAT, NAT, TGS)
- un registre adresse ach f sur le front descendant de AS (RA)
- Un bloc acces avec le microprocesseur
- Un bloc acces avec la memoire externe (memoire d'ecran et eventuellement memoire de caracteres d'extension)
- un bloc qui recoit $CLK = 12 MHz$ et eventuellement une synchronisation externe et qui genere les signaux de service
- un bloc de sortie video. Dans ce dernier bloc, un buffer compote les codes caracteres et codes attributs correspondant d'une rangee complete - un code caractere pointe un caractere en ROM. Ce caractere est serialise tranche apres tranche en video; apres avoir recu ces attributs (couleur, taille, inversion etc...)

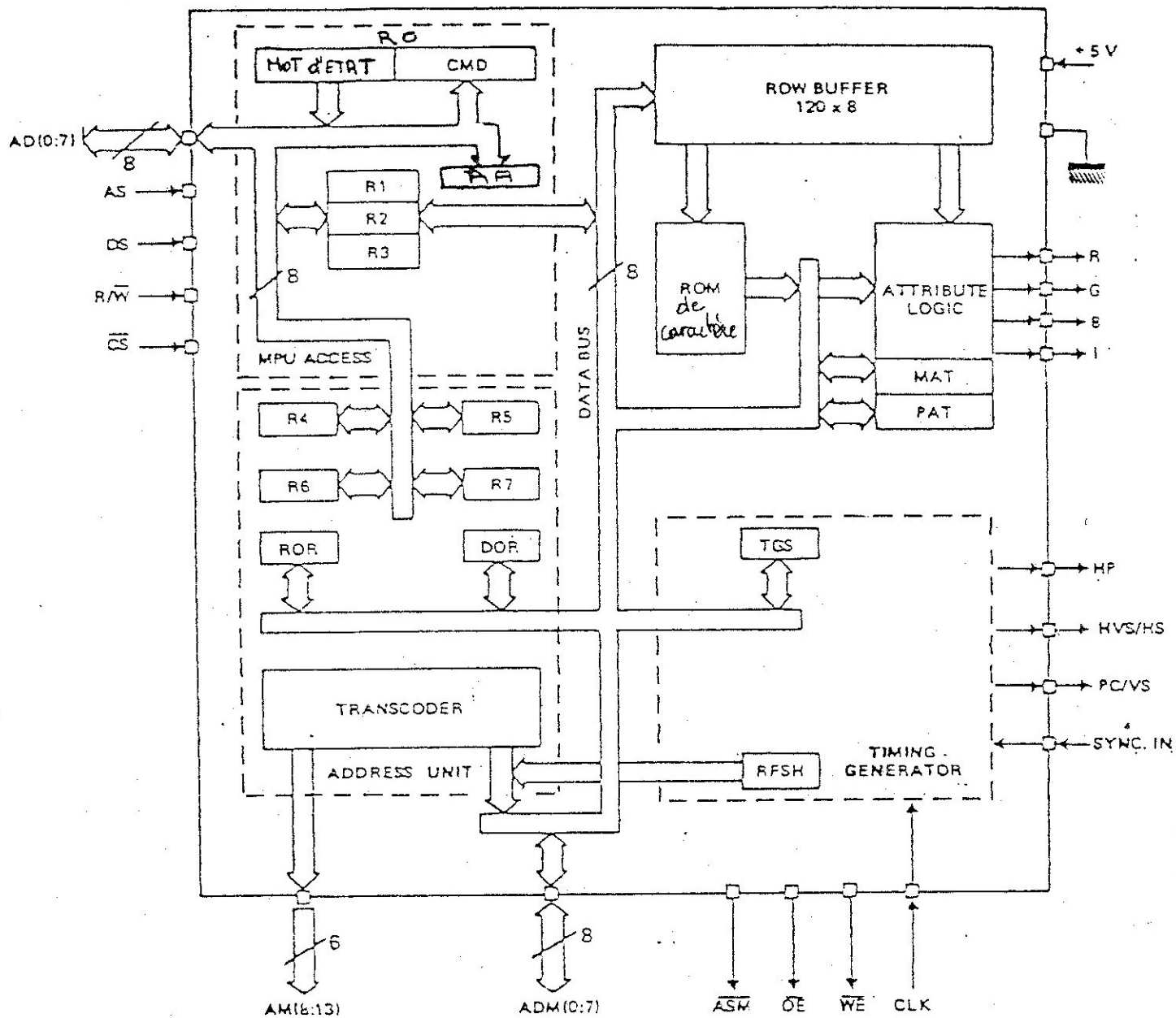


Figure 19. schema bloc de l'EF9345 -

- Programmation de L'ET 9345 -

le registre d'adresse

le 9345 utilise un registre d'adresse dans lequel ira se loger un octet venant du microprocesseur sur le front descendant de AS

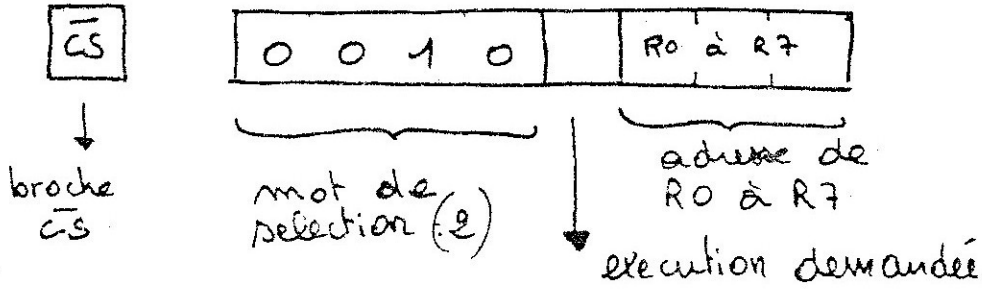


figure 20 constitution du registre d'adresse.

- les registres à accès direct -

- registre R0

D'une manière générale; avant d'envoyer un octet vers le 9345 il faut vérifier si ce circuit n'est pas occupé à gérer son erran.

Le bit 7 ^{baptisé busy} du mot d'état - nous indique si le 9345 est disposé ou pas à recevoir un octet. Une "attente sur busy à 0" est donc nécessaire pour toute communication.

le 9345 comporte un jeu de 17 instructions; (voir la figure 24). le registre R0 est utilisé comme registre d'instruction.

Remarque: le 9345 utilise un double registre R0.
 - R0 en lecture seule = mot d'état
 - R0 en écriture seule = registre d'instruction.

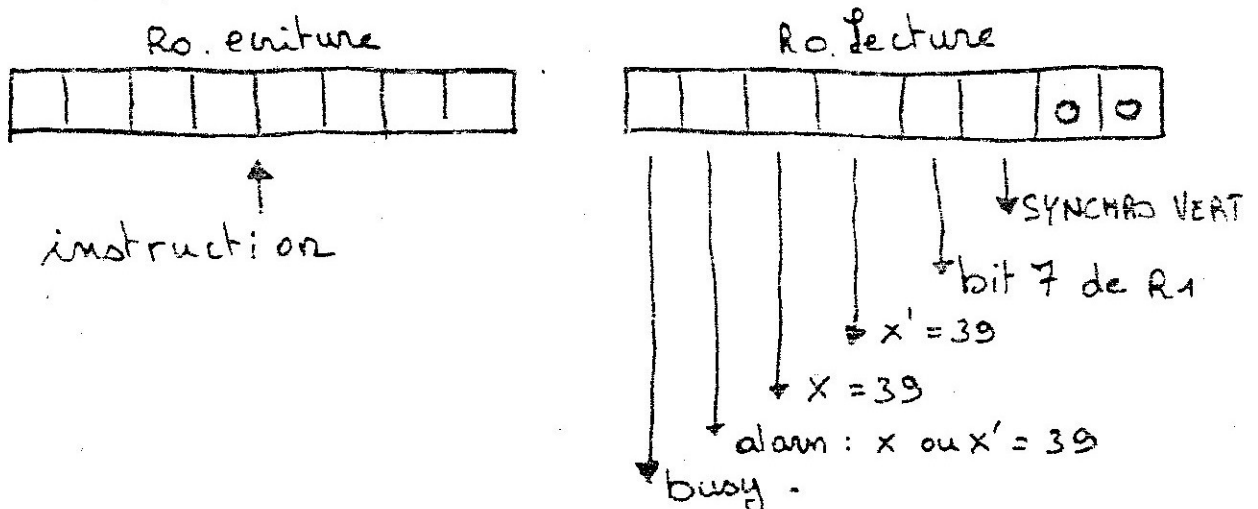


figure 21 Constitution de R0

- les registres R1 à R3 voir la fig. 23.

Ils sont utilisés comme registres de données.

exemple (1) voir le jeu d'instruction en figure 24.

En mode 40 caractères de 16 bits par rangée, dans la colonne "argument" on lit qu'il faut placer le caractère et son attribut (A et B) dans les registres R1 et R2

(2) en mode 40 caractères de 24 bits par rangée le caractère et ses attributs doivent être placés en R1, R2, R3

- les registres R4 et R5 voir la figure 22.

Ils constituent une adresse en RAM externe pour un caractère d'extension.

- les registres R6 et R7

memoire d'ecran

- Ils constituent une adresse en memoire d'ecran externe dans laquelle on veut lire ou écrire

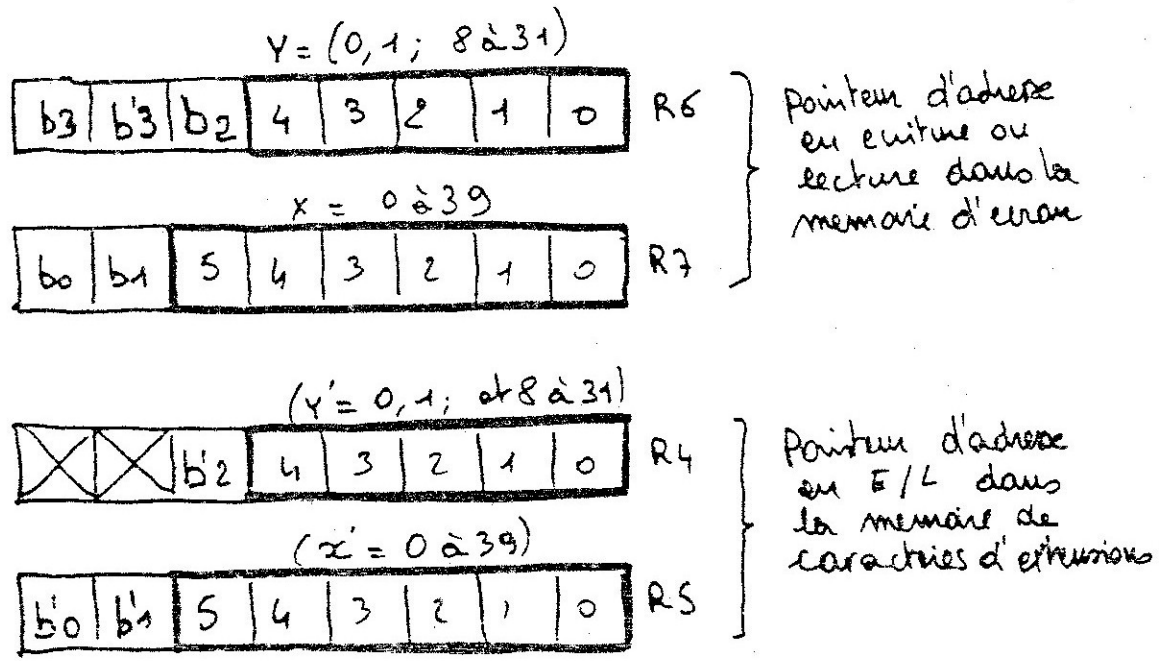


figure 22. constitution des pointeurs d'adresse en RAM externe

x et y pointent la position du curseur
 b0 à b3 pointent un bloc de 1000 cellules

Remarque: observons que $1k = 1024$ n'est pas exactement divisible par 40.
un block compte donc 25 rangées complètes et 24 bits restent incorpés -

- afin de "cadrer" exactement la mémoire d'écran avec la mémoire RAM; il est utilisé 2 lignes de service
- une ligne de service complète pour $Y=0$ tous les blocs pairs (40 octets)
 - une ligne de service partielle pour $Y=1$ et $X=32$ à 39 tous les blocs impairs (8 octets)

ainsi tous les 2 blocs ou remplis en mémoire $2000 + 40 + 8 = 2048$ octets.

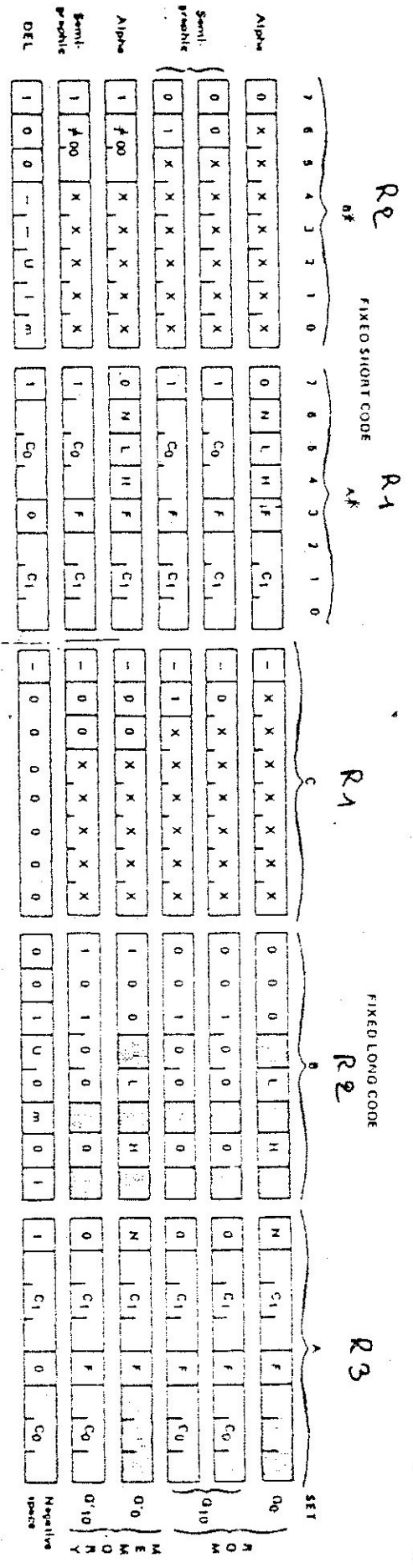


Figure 23. utilisation des registres R1, R2, R3 de données

- 1/ Translation process
 - The translation process operates through 3 elementary operations:
 - Field-to-field: a character code or an attribute value like C₀, flashing! It directly loaded from short to long code.
 - Field-to-content: the decoding of a short code forces the value of the equivalent long code attribute. For example, semi-graphic short characters forces normal size (H = 0, L = 0) attributes.
 - Latched attributes: at the beginning of each row, these attributes are reset (no underline, not concealed, no insert, black background). Then, they keep their current value until modified by either a field to field or field to content operation.
 - 2/ EFG340/A1 compatibility
 - It is binary code compatible with few exceptions:
 - flashing attribute is negated.
 - A7 is negated in delimiters.
- It is also display compatible with 2 exceptions concerning the underlining:
- an alphanumeric belonging to G:0 may be underlined.
 - any alphanumeric following a semi-graphic cannot be underlined.

COMMAND TABLE

TYPE	MEMO	CODE	PARAMETER	STATUS	ARGUMENTS							EXECUTION TIME (1)		
					R1	R2	R3	R4	R5	R6	R7	WRITE	READ	
INDIRECT	IND	7 0 5 4	R/W, R	0 0 0 0 0	D	-	-	-	-	-	-	MP	2	3.5
40 CHARACTERS - 24 BITS	KRF	0 0 0 0	R/W 0 0 1	X X 0 0 0	C	B	A	-	-	-	-	MP	4	7.5
40 CHARACTERS - 16 BITS	KRG	0 0 0 0	R/W 0 1 1	X X 0 0 0	A	R	W	-	-	-	-	MP	5.5	7.5
80 CHARACTERS - 8 BITS	KRC	0 1 0 0	R/W 0 0 1	X X 0 0 0	C	-	-	-	-	-	-	MP	9	9.5
80 CHARACTERS - 12 BITS	KRL	0 1 0 1	R/W 0 0 1	X X 0 0 0	C	-	A	-	-	-	-	MP	12.5	11.5
40 CHARACTERS VARIABLE	KRV	0 0 1 0	R/W 0 0 1	X X X X X	C	B	A	-	XF	MP	(2) 3 + 3 * J	MP	(2) 3 + 3 * J	3.5 + 8 * J
EXPANSION	EXP	0 1 1 0	0 0 0 0 0	X 0 X 0 0	C	B	A	-	PW	XF	MP	(3) < 247	-	-
COMPRESSION	CMP	0 1 1 1	0 0 0 0 0	X 0 X 0 0	C	B	A	-	PW	XF	MP	(3) < 402	-	-
EXPANDED CHARACTERS	KRE	0 0 0 1	R/W 0 0 1	X X 0 0 0	C	B	A	-	PW	-	MP	4	4	7.5
BYTE	OCT	0 0 1 1	R/W P 0 1	X X X X 0	D	-	-	-	-	-	-	MP	4	4.5
MOVE BUFFER	MVB	1 1 0 1	S F B B B	0 0 0 0 0	W	-	-	-	-	-	-	MP	(2) 2 + 4 * n	-
MOVE DOUBLE BUFFER	MVD	1 1 1 0	S F B B B	0 0 0 0 0	W	-	-	-	-	-	-	MP	(2) 2 + 8 * n	-
MOVE TRIPLE BUFFER	MVT	1 1 1 1	S F B B B	0 0 0 0 0	W	-	-	-	-	-	-	MP	(2) 2 + 12 * n	-
VERTICAL SYNC MASK SET	VSM	1 0 0 1	1 0 0 1	0 0 0 0 0	-	-	-	-	-	-	-	-	1	-
VERTICAL SYNC MASK RESET	VNM	1 0 0 1	0 1 0 1	-	-	-	-	-	-	-	-	-	1	-
INCREMENT Y	INY	1 0 1 1	0 0 0 0 0	0 0 0 0 0	-	-	-	-	-	-	-	Y	TBD	-
NO OPERATION	NOP	1 0 0 1	0 0 0 0 1	-	-	-	-	-	-	-	-	-	1	-

P : Pointer select
 1 : auxiliary pointer
 0 : main pointer.
S, T : Source, destination
 01 : source - MP ; destination - AP
 10 : source - AP ; destination - MP
X, n : Stop condition
 01 : stop at end of buffer
 10 : no stop
r : Indirect register number.

- : Not affected
W : Used as working register
PW (ZW, YW) : Working buffer
X : Set or Reset
XF : X File
I : Pointer Incrementation
D : Data
MP : Main pointer
AP : Auxiliary pointer.

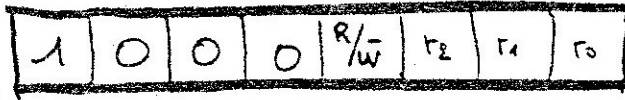
(1) Unit : 12 clock periods ($\approx 1 \mu s$) without possible suspension.
 (2) n : total number of words ≤ 40 ; J = 1 for long codes, J = 0 for short codes.
 (3) Worst case (20 long codes + 20 short codes).

Figure: 2X
 Jeu d'instruction de L'EE 8345.

- les registres d'accès indirect -

- Le code instruction "IND". (figure 25)

En exécutant l'instruction "IND" dans le registre R0; le registre ~~peut~~ doit être manipulé juste après ROR, PAT, MAT, DOR ou TGS. L'adresse de ce registre est incluse dans le code instruction.



0	0	1	TGS
0	1	0	MAT
0	1	1	PAT
1	0	0	DOR
1	0	1	-
1	1	0	-
1	1	1	ROR

figure 25. code instruction "IND".

la figure 26. nous indique le détail des fonctions des registres à accès indirect.

Format d'ecran	PAT7	TGS7	TGS6
40 caract Long	0	0	0
40 caract var	0	0	1
40 caract courts	1	0	0
80 caract Longs	0	1	1
80 caract courts	0	1	0

mode d'insertion	PAT5	PAT4
incrustation	0	0
Encart	0	1
Forage	1	0
inhibition	1	1

mode d'affichage du curseur	MAT5	MAT4
Fixe inverse	0	0
clignotant inverse	0	1
fixe souligné	1	0
clignotant souligné	1	1

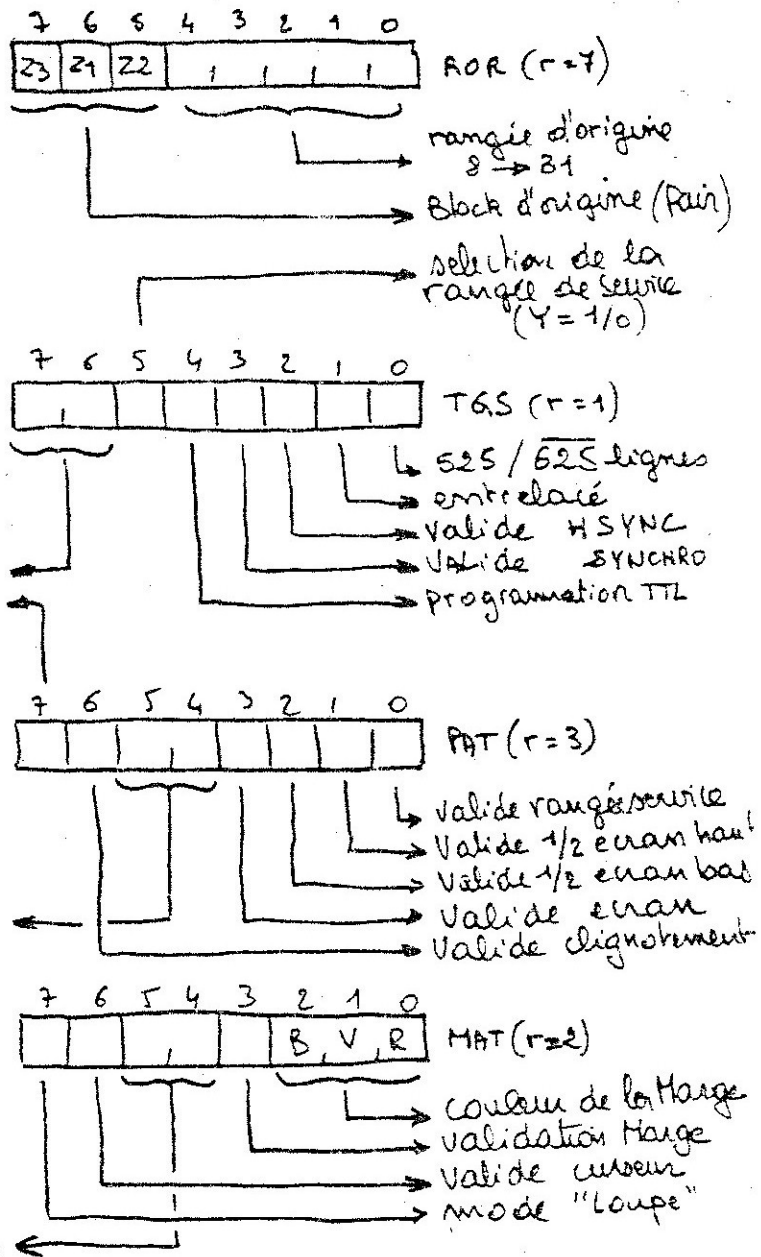


figure : 26 detail des registres d'accès indirect.

- Ce registre a pour fonction de réserver de la place en RAM afin d'y loger les caractères d'extension par paquets de 1Kx8.
- Un caractère d'extension est constitué d'une matrice de 10 octets. ainsi on dispose de 100 caractères dans chaque paquet de 1Kx8.
- On distingue 3 types de caractères d'extensions.
 - 100 caractères ayant les mêmes attributs que les caractères alpha numériques.
 - 2x100 caractères ayant les mêmes attributs que les caractères graphiques (jointifs et séparés)
 - 8x100 caractères quadrichromiques.

les registres utilisés pour cette fonction sont :

- DOR : qui place les paquets de 100, 200, ou 800 caractères en RAM.
- Les bits B3, B4, B5 du registre d'attributs (R2) qui séparent les jeux de 200 et 800 caractères en jeux de 8x100 et 2x100
- le registre caractère (R1) qui donne le numéro du caractère dans un jeu de 100 (ce numéro sera 0 à 3 et 32 à 127)

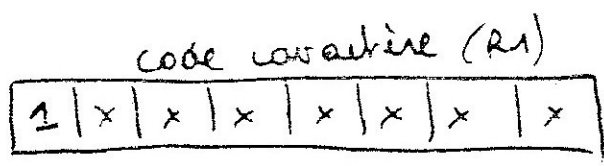
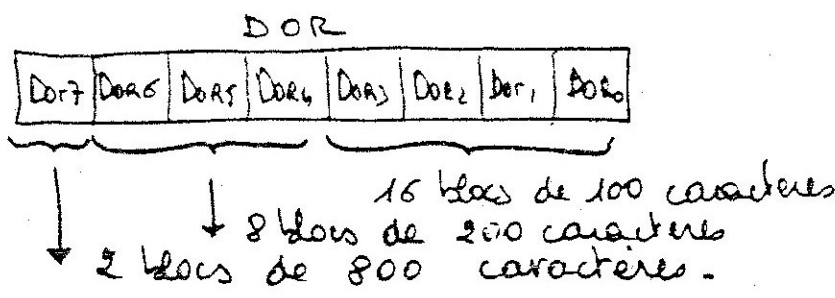


figure 27. le registre DOR et le registre code caractère



Mode "alpha"

R1	R2	R3	DOR	DOR	DOR	DOR
7	6	5	3	2	1	0

1 0 0 X X X X

place le bloc de 100 sur 16kx8

mode "graphic"

R1	R2	R3	DOR	DOR	DOR (R2)
7	6	5	6	5	4

1 0 1 X X X X

place le bloc de 200 sur 16kx8

separe le bloc de 200 en 2 bloc de 100

"mode quadri"

R2	R2	DOR	R2	R2	R2
7	6	7	5	3	4

1 1 X X X X

place le bloc de 800 sur 16kx8

separe le bloc de 800 en 8 blocs de 100

fig. 28

le registre DOR, associe au registre attribut architecturalement la RAM en blocs de 100 caracteres.

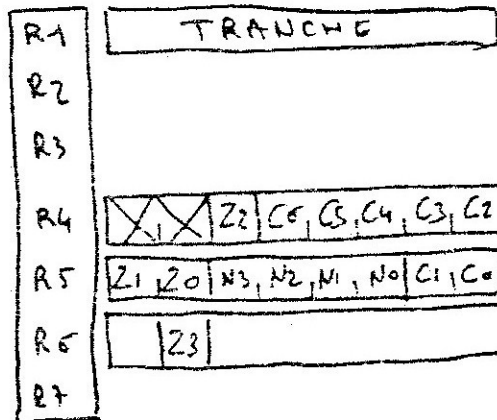
chargement d'un caractere d'extension en RAM:

ce chargement s'effectue tranche après tranche.

les registres utilises sont :

- R1; qui recoit la donnée (1 tranche)
- R4, R5, R6 qui pointent en RAM cette tranche

voir la fig. 29



Z0 → Z3 pointent le bloc de 1k sur 16kx8

C0 → C5 pointent le numero du caractere sur R7

N0 → N3 pointent la tranche sur 10

fig. 29

écriture d'un caractere d'extension en memoire RAM.


```

10 REM =====
11 REM   PROGRAMMATION DU -Z80-
12 REM   -----
13 REM   ..CPY J.MINICILLI..
14 REM   ..SERVICE SA M.L.V..
15 REM   ..DEPART FORM TECH..
16 REM =====
17 :
18 :
19 :
20 REM   -----
21 REM   PRESENTATION
22 REM   -----
23 :
24 :
25 :
40 CLEAR 500,&"6FFF"
50 DIMH$(50):DIMA$(50):DIMB$(50)
60 DATA 0,1,2,3,4,5,6,7,8,9,A
70 DATA B,C,D,E,F,a,b,c,d,e,f
80 FOR I=0 TO 21:READH$(I):NEXT I
110 DATA INIT 3,3:STORE
120 TX4,1,0
130 CURSORX 7
140 PRINT"*****"
145 CURSORX 7
150 PRINT"*****"
155 CURSORX 7
160 PRINT"* PROGRAMMATION DU -Z80- *"
165 CURSORX 7
170 PRINT"* PROGRAMMATION DU -Z80- *"
175 CURSORX 7
180 PRINT"*****"
185 CURSORX 7
190 PRINT"*****"
195 CURSORX 7
200 PRINT:TX6,0,0
210 PRINT" -CPY-J.MINICILLI-SERVICE SA-DFT-8/84-
220 PRINT:PRINT:TX1,0,0:CURSORX 4
230 PRINT"RAM DISPONIBLE:de 7000(H) 77FF(H)"
240 CURSDRY 13 :CURSORX 4:TX4,0,0
250 PRINT' FONCTION          TOUCHE "
260 PRINT:PRINT" PROGRAMMATION.....-P-"
270 PRINT" LECTURE PROGRAMME.....-L-"
280 PRINT" EXECUTION PROGRAMME.....-G-"
300 CURSORY 21:TX 1,0,0:CURSORX 5
310 PRINT"CHOISISSEZ PUIS TAPPEZ -RET-";
320 DISPLAY :SCREEN
330 INPUT A$
340 IF A$="P"ORA$="p"GOTO 1000
350 IF A$="L"ORA$="l"GOTO 2000
360 IF A$="G"ORA$="g"GOTO 3000
380 CURSORY 21:CURSORX 34:TX 5,0,1
390 PRINT"ERREUR"
400 GOSUB 20000
420 CURSORY 21:CURSORX 34:TX1,0,0
430 PRINT"
440 GOTO 300
998 :
999 :

```

```

1000 REM =====
1010 REM   PROGRAMMATION
1020 REM =====
1025 :
1030 :
1035 :
1040 INIT 3,3
1050 GOSUB 5000:TX4,0,0
1060 AP=A
1070 C=INT(AP/256):GOSUB 15000
1080 C=((AP/256)-INT(AP/256))*256
1090 GOSUB 15000
1100 CURSORX 9:C=PEEK(AP):GOSUB 15000
1110 CURSORX 7:TX1,0,0
1120 INPUT A$
1125 A$=LEFT$(A$,2)
1130 TX4,0,0:B$=LEFT$(A$,1)
1140 IF B$="L"OR B$="l"GOTO 2030
1150 IF B$="P"OR B$="p"GOTO 1050
1160 IF B$="G"OR B$="g"GOTO 3050
1165 X=0:Z=0
1170 FOR J=0 TO 21
1175 IF H$(J)=LEFT$(A$,1) THEN Z=1
1180 IFH$(J)=MID$(A$,2,1)THENX=1
1182 NEXT J
1185 IF X=0 OR Z=0 THEN 1300
1187 GOSUB 10000
1190 PRINTCHR$(136);:CURSORX 7:PRINT " ";:CURSORX 11:PRINT"
1195 POKE AP,A:AP=AP+1
1200 IF AP<(&"7800") GOTO 1070
1205 TX5,0,0
1210 PRINT "FIN DE MEMOIRE 7800(H)"
1215 TX4,0,0
1220 GOSUB 20000
1230 AP=AP-1
1240 TX1,0,0:PRINT
1245 PRINT"ENTREZ P,L ou G ";
1250 GOTO 1120
1300 GOSUB 8000
1310 AP=AP-1:GOTO 1187
1997 :
1998 :
1999 :

```

```

2000 REM -----
2010 REM   LISTING DU PROGRAMME
2020 REM -----
2021 :
2022 :
2023 :
2025 PRINT CHR$(159):INIT 3,3
2030 GOSUB 6000:TX4,0,0
2035 IF(A+40)>32768 GOTO 7000
2040 FOR I=0 TO 19
2050 C=INT(A+I)/256:GOSUB 15000
2060 C=((A+I)/256-INT((A+I)/256))*256
2070 GOSUB 15000:PRINT "  ";
2080 C=PEEK(A+I):GOSUB 15000
2090 CURSORX 20
2100 C=INT(A+I+20)/256:GOSUB 15000
2110 C=((A+I+20)/256-INT((A+I+20)/256))*256:GOSUB 15000:PRINT "
2115 C=PEEK(A+I+20):GOSUB 15000
2120 PRINT :NEXT I
2125 TX1,0,0
2127 PRINT"TAPEZ 'S et RET' POUR SUITE"
2128 TX4,0,0
2130 INPUT A$
2140 B$=LEFT$(A$,1)
2150 IFB$="P"ORB$="p"GOTO 1050
2160 IFB$="G"ORB$="g"GOTO 3050
2170 IFB$="L"ORB$="l"GOTO 2030
2175 IFB$="S"ORB$="s"GOTO 2195
2180 GOSUB 8000
2190 GOTO 1240
2195 A=A+40:GOTO 2035
2997 :
2998 :
2999 :
3000 REM -----
3010 REM   EXECUTION PROGRAMME
3020 REM -----
3030 :
3040 :
3045 INIT 3,3
3050 GOSUB 5000
3060 CALL(A)
3070 GOTO 1240
3997 :
3998 :
3999 :

```

```

5000 REM -----
5010 REM  APPEL ADRESSE ORIGINE
5020 REM -----
5030 :
5040 :
5050 TX1,0,0:PRINT
5060 INPUT "ADRESSE ORIGINE:";A$
5062 B$=LEFT$(A$,1)
5064 IFB$="P"ORB$="p"GOTO 1050
5066 IFB$="G"ORB$="g"GOTO 3050
5068 IFB$="L"ORB$="l"GOTO 2030
5070 GOSUB 10000
5080 IF A)((&"77FF")GOTO 5105
5090 IF A)((&"7000")GOTO 5140
5100 RETURN
5105 TX5,0,0
5110 PRINT "DONNEZ UNE ADRESSE INF. a 7800":GOSUB 20000
5120 GOTO 5050
5140 TX5,0,0
5150 PRINT "DONNEZ UNE ADRESSE SUP. a 6FFF"
5160 GOSUB 20000
5170 GOTO 5050
5997 :
5998 :
5999 :
6000 REM -----
6010 REM  APPEL ADRESSE ORIGINE
6015 REM      DE 0000 a FFFF
6020 REM -----
6030 :
6040 :
6050 TX1,0,0:PRINT
6060 INPUT "ADRESSE ORIGINE:";A$
6062 B$=LEFT$(A$,1)
6064 IFB$="P"ORB$="p"GOTO 1050
6066 IFB$="G"ORB$="g"GOTO 3050
6068 IFB$="L"ORB$="l"GOTO 2030
6070 GOSUB 10000
6090 RETURN
6997 :
6998 :
6999 :
7000 TX1,0,0:PRINT
7010 PRINT "40 DERNIERS OCTETS:"
7020 PRINT " LISTABLES:7FDS a 7FFF"
7030 PRINT
7040 TX4,0,0:GOTO 1220
7997 :
7998 :
7999 :

```

```

8000 REM -----
8010 REM  MESSAGE D'ERREUR
8020 REM -----
8030 :
8040 :
8050 :
8060 TX5,0,1:PRINT"ERRARE HUMANUM ES.."
8070 GOSUB 20000:PRINT CHR$(137);
8080 TX4,0,0:PRINT" "
8090 RETURN
8997 :
8998 :
8999 :
10000 REM -----
10010 REM  CALCUL HEXA==>DECIMAL
10020 REM -----
10021 :
10022 :
10023 A=0
10030 FOR I=1 TO LEN(A$)
10040 X=ASC(MID$(A$,I,1))
10050 IF X>47ANDX<58THENM=X-48:A=M+16*A
10060 IF X>64ANDX<71THENM=X-55:A=M+16*A
10070 IF X>96ANDX<103THENM=X-87:A=M+16*A
10080 NEXT I
10090 RETURN
10997 :
10998 :
10999 :
15000 REM -----
15010 REM  CALCUL DECIMAL==>HEXA
15020 REM -----
15030 X=INT(C/16):X=X+48
15040 IF X>57 THEN X=X+7
15050 PRINT CHR$(X);
15060 X=((C/16)-INT(C/16))*16
15070 X=X+48:IF X>57 THENX=X+7
15080 PRINT CHR$(X);
15090 RETURN
15997 :
15998 :
15999 :
20000 REM -----
20010 REM  MESSAGE SONORE
20020 REM -----
20030 :
20040 FOR I=1 TO 50:S=RND(1)*255
20050 SOUND S,2:NEXT I
20060 RETURN
21000 REM-----

```

Quelques programmes en langage Basic

```
10 REM =====
20 REM KALEIDOSCOPE CPY/DFT
30 REM
40 REM J. MINICILLI SSA/MLV
50 REM =====
60 REM
70 REM
80 REM
90 INIT 0,0:PAGE
100 FOR I=0 TO 50 : S=RND(1)*250
110 SOUND S,3 : NEXT I
120 FOR W=3 TO 50
130 FOR I=1 TO 9
140 FOR J=0 TO 9
150 K=I+J:S=RND(1)*250
160 C=(J+1)*3/(I+3)+I*W/7
170 GR C,0,0:STORE
180 CURSORX I+10:CURSOR Y K
190 PRINT CHR$(63);
200 CURSORX K+10:CURSOR Y I
210 PRINTCHR$(63);
220 CURSORX (20-I)+10:CURSOR Y 20-K
230 PRINTCHR$(63);
240 CURSORX (20-K)+10:CURSOR Y 20-I
250 PRINTCHR$(63);
260 CURSORX K+10:CURSOR Y 20-I
270 PRINTCHR$(63);
- 280 CURSORX (20-I)+10:CURSOR Y K -
290 PRINTCHR$(63);
300 CURSORX I+10:CURSOR Y 20-K
310 PRINTCHR$(63);
320 CURSORX (20-K)+10:CURSOR Y I
330 PRINTCHR$(63);
340 SOUND S,2
350 SCREEN:DISPLAY :NEXT J
360 NEXT I
370 NEXT W
380 GOTO 120
```

```
10 REM &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
20 REM &
30 REM & LE BRUIT DE &
40 REM & L'ELECTRON COGITEUR &
50 REM &
60 REM &&&&&&&&&&&&&&&&&&&&&&&&&&&
70 :
80 :
100 S=RND(1)*255:SOUND S,3
110 GOTO 100
```

```
5 REM ++++++
6 REM +
7 REM + SINUSOIDE +
8 REM +
9 REM ++++++
10 FOR I=1 TO 360 STEP 15
20 N=I*.01745
30 X=15*SIN(N)+20
40 CURSORX X :PRINT"+"
50 NEXT I
60 GOTO 10
```

```
10 REM *****
20 REM * "J'AI DU BON TABAC" *
30 REM * PROGRAMME MUSICAL *
40 REM *****
41 :
42 :
43 :
50 CLEAR 2000
60 A$="T15 02 CDECD32DEF32F32E32E32"
70 B$="T15 02 CDECD32DEF32G32C64"
80 C$="T15 02 G32GFE32DEF32G32D64"
90 D$="T15 02 CDECD32DEF32G32 03 C99"
100 PLAY A$+B$+C$+D$:END
101 :
102 :
```