PLAQUE MEMOIRE (1¼K RAM et 2½ K ROM)
AVEC PROGRAMMATEUR D'EPROMS 2708

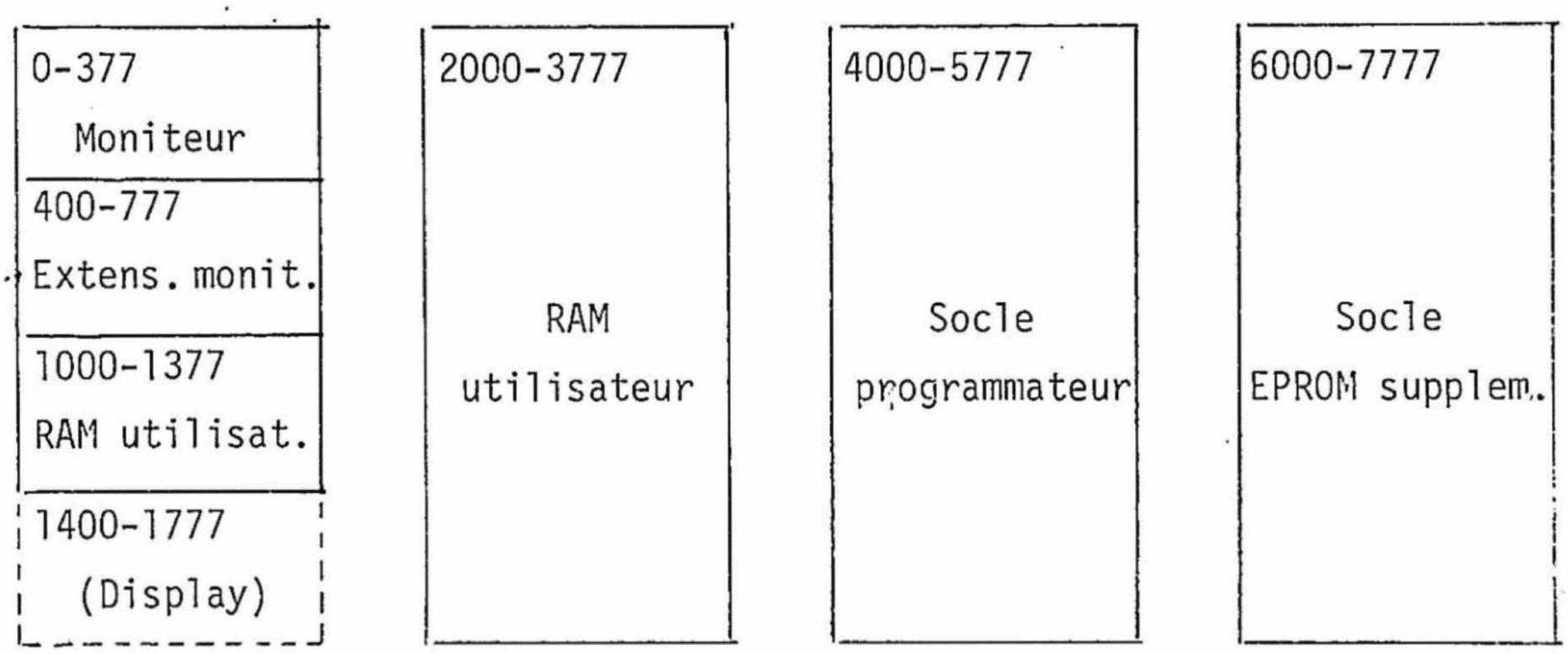
La plaque extension mémoire utilise tout l'espace d'adressage du DAUPHIN (4k), sauf une page (adresses 1400 à 1777) réservée pour le display ou le programmateur de 74S471/74S288.

La plaque comporte des socles pour
- 2 ROMs 256x8 bits (type 74S471)
- 2 EPROMs 1024x8 bits (type 2108).

L'un des socles de 2708 peut servir à la programmation des 2708.

De plus deux groupes de RAM sont prévus: une page (256x8) pour compatibilité avec la configuration minimum du DAUPHIN et 1024x8 mots supplémentaires qui reçoivent en particulier lors de la programmation d'une mémoire les données à recopier dans l'EPROM 2708.

SCHEMA BLOC



1. EXPLICATION DU SCHEMA

La sélection des mémoires se fait à l'aide d'un décodeur en deux étages. Le premier étage repère les 4 groupes de 1k et le second sélectionne les pages (de 256. positions).

Dans les circuits tirés en 1977, la compatibilité avec les 2111 (18 pins) nécessite une modification importante du circuit imprimé, à faire avant le montage des composants. Par contre, la plaque fonctionne sans aucune modification avec les mémoires de type 2112 (16 pins).

La sélection de la mémoire 2708 que l'on veut programmer passe par différents circuits. Si la partie programmateur n'est pas montée, un pont doit rétablir la liaison manquante.

ADMEMLOW 2^11 40α 20α 2^10 WRITELOW 2^9 100α 400 200 100 40 20 ADDRESS 10 4 2 1 2^0

26 16 15 25 14 13 12 11 10 9 8 7 6 5

A6 LS139 CS B A 0 1 2 3

(6) 2708 6000-7777 +12V -5V

(4) 2708 4000-5777 +12V -5V

3400 3000 2400 2000 1000

A5 A4 A3 A2 A1 2112

B5 3400-3777 B4 3000-3377 B3 2400-2777 B2 2000-2377 B1 1000-1377

B6 LS139 CS B A C 1 2 3

1 S471 400-777

Ø S471 0-377

LS00 C4

DATA 27 24 23 22 21 20 19 18 17 20α 10α 40 20 10 4 2 1 2^0

+5V +12V GND -5V

100µF 16V C2 3,3µF C3 0,33µF C4 7905

STOPLOW 28

+26V 47µF 40V C1 4,7k R7 47k R9 2,7k R6 100 R2 47 R1 4,7k R8 BC327 BC337 2,2µ

D5 06 LS14 LS16 LS32 C4 C5 C6 C14

D1 D2 D3 47k R10 10k R12 4,7k R11 2,2k R4 1k R3 1,1k R5

150pF C5 2,2n C6 470pF C6

Programmateur de 2708

EXTENSION MEMOIRE AVEC PROGRAMMATEUR

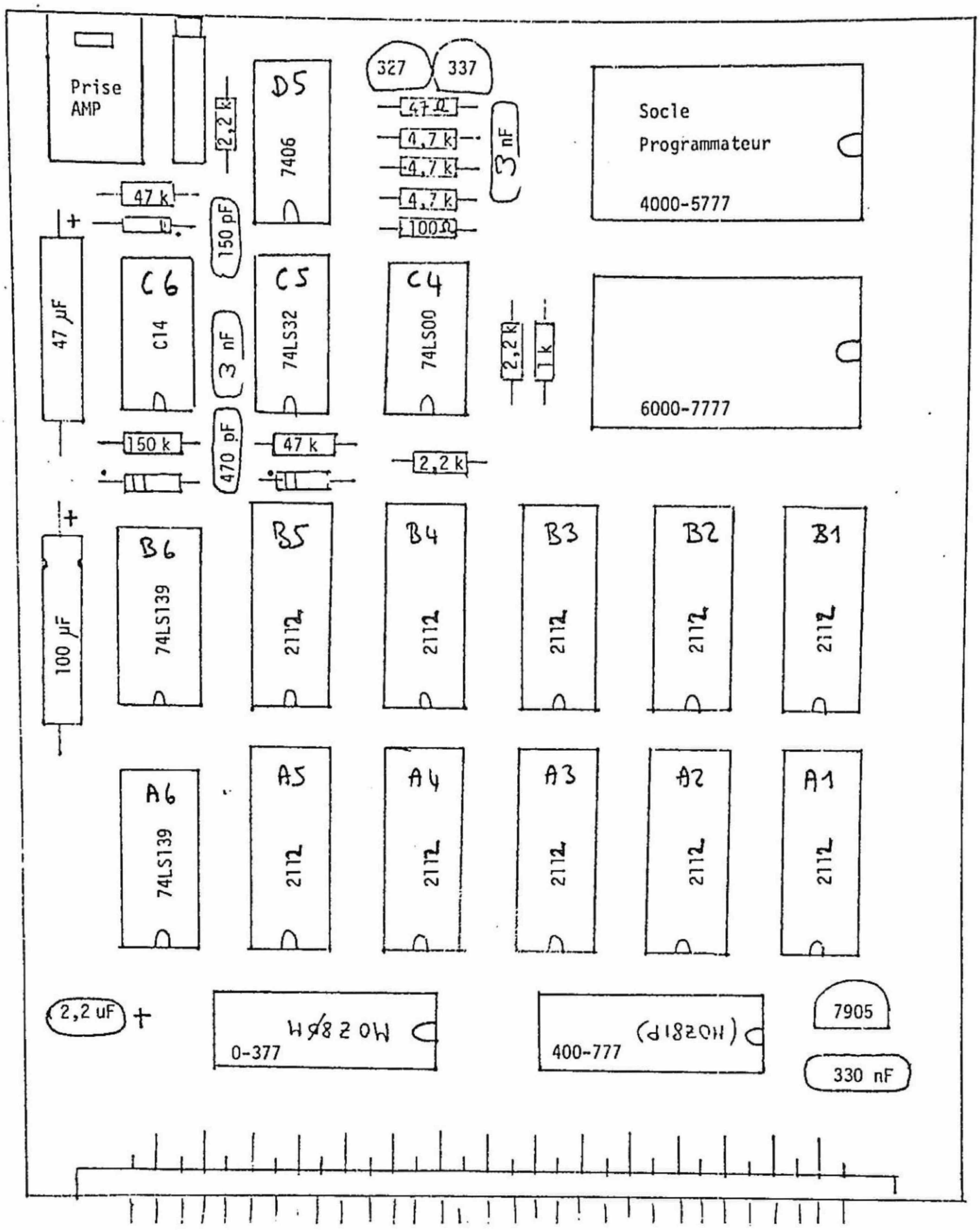
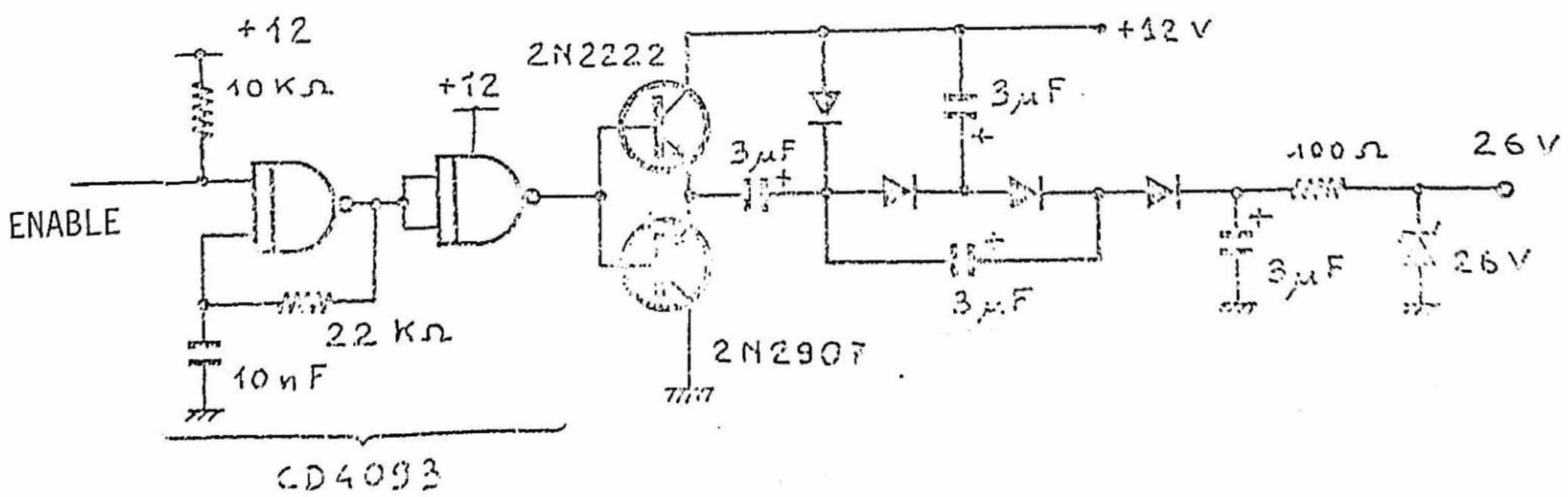


Schéma d'une alimentation 26V



2. TEST DE LA MEMOIRE

Insérer le moniteur MO26ØM. Vérifier qu'il tourne.

Sinon, écrire quelques mots différents avec le panneau de test aux adresses 0, 1, 2 puis aux adresses 1000, 1376, 1377 et les relire.

Avec le moniteur, vérifier que quelques positions en 1000 et 2000 se chargent correctement.

Charger le programme de test suivant en RAM: ce programme tourne tant qu'il ne trouve pas d'erreur en 2000- 3777. Modifier les paramètres pour tester de 1000 à 1377.

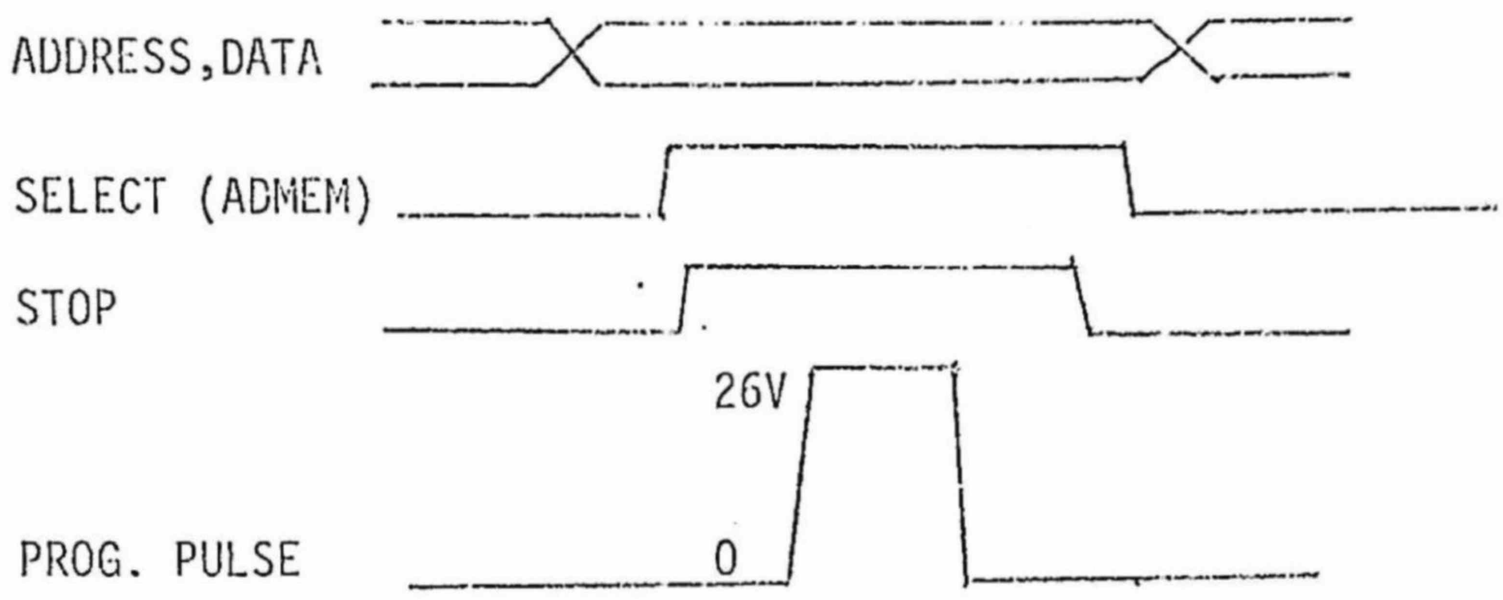
```
                DEBMEM=  2000                                    SIGNETICS
                LONG=    2000

TMEM:     1000     4    LOAD     A,# DEBMEM/400
             1     4
             2   310    LOAD     POINT,A
             3    54
             4     4    LOAD     A,# DEB&377
             5     0
             6   310    LOAD     POINT+1,A
             7    51
          1010     6    LOAD     C,# LONG/400
             1     4
             2     7    LOAD     D,# Ø
             3     0
CHECK:       4     4    LOAD     A,# Ø
             5     Ø
CHECK2:      6   317    LOAD      (D)+@POINT,A   ;écriture
             7   342
          1020    60
             1   357    COMP     A,(D)+@ POINT   ;relecture et comparaison
             2   342
             3    60
             4   230    JUMP,NE  ERROR
             5    16
             6   330    INC      A
             7     Ø                    -12
          1030   333    INCJ,NE  D,CHECK2
             1   164
             2    10    LOAD     A,POINT
             3    24
             4   330    INC      A
             5     Ø
             6   310    LOAD     POINT,A
             7    20
          1040   372    DECJ,NE  C,CHECK   370 + 2
             1   152                        -C
             2   233    TRAP
             3     0
ERROR:       4     4    LOAD     A,# LETF
             5   161
             6   324    LOAD     $DIGØ,A
             7     Ø
          1050    33    JUMP     ERROR
             1   172


POINT:    1060          .WORD    Ø              ;pointeur
             1
```

Ce programme est très imparfait car la comparaison est immédiate.

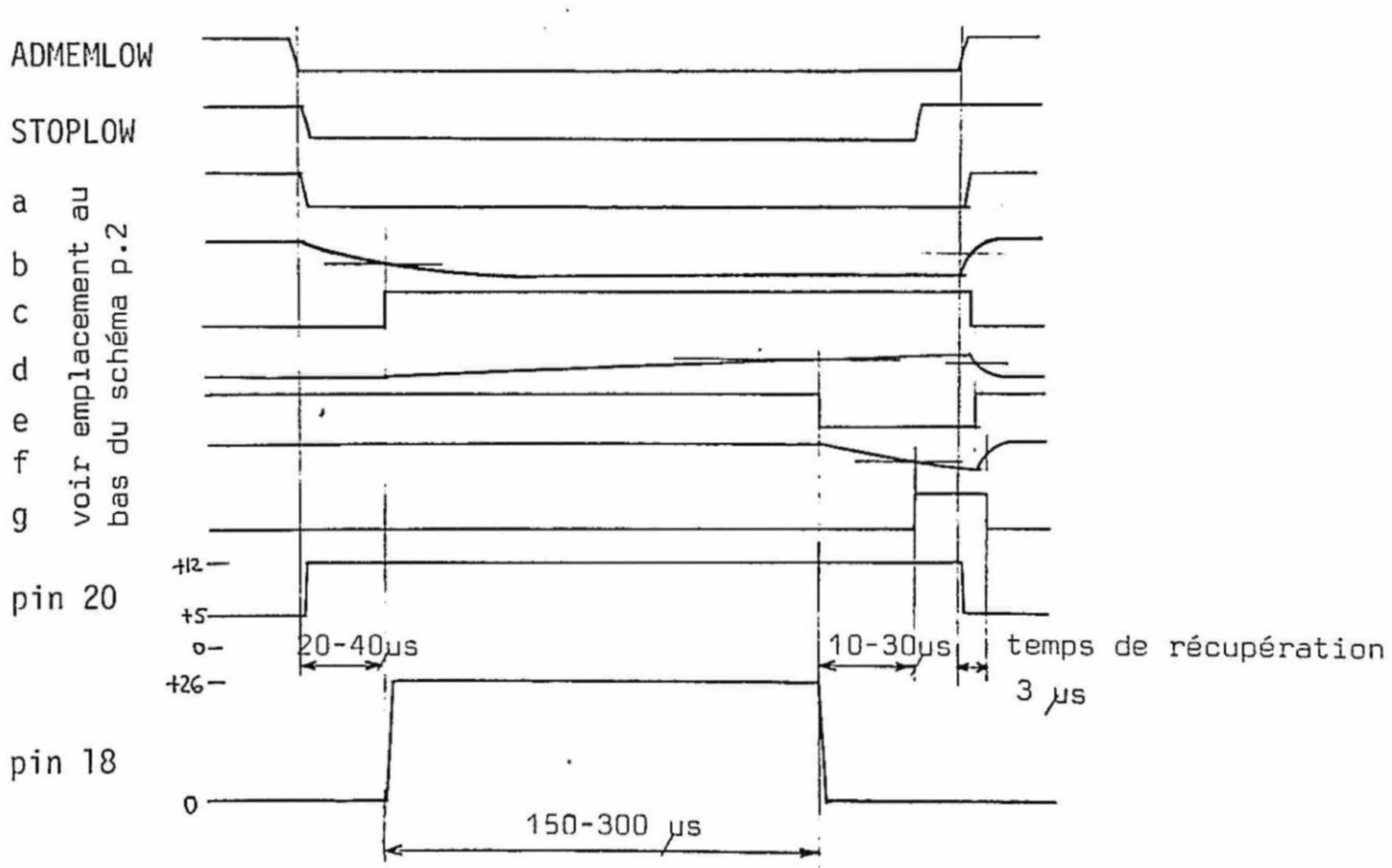
Pour l'améliorer, relire une deuxième fois les positions mémoire avant le TRAP (ce qui revient à récrire les instructtions 1000 à 1041 à l'exclusion de LOAD (D)+@POINT,A)

3. PRINCIPE DE LA PROGRAMMATION

La programmation des 2708 se fait par des impulsions 26V lorsque les adresses et les données sont stables. La durée des impulsions est de 0,1 à 1 ms, et le processeur est bloqué sur la ligne STOP (NOTREADY) pendant ce temps.



Au lieu de monostables, ce sont des retards et une logique adéquate qui garantissent les temps spécifiés par le fabricant.



Pour vérifier ces timings, synchroniser l'oscilloscope avec le signal SELECT et observer successivement les autres signaux après avoir chargé et demandé l'exécution du programme suivant:

```
TEST:  1000  314   LOAD    2000,A
          1    4
          2    0
          3   33   JUMP    1000
          4  373
```

Ne pas mettre d'EPROM sur le socle, évidemment, et si le 26V n'est pas disponible, relier le +26V au +12V avec un fil provisoire.

4. PROGRAMME

Le programme commence par lire l'EPROM et il s'arrête sur la première position non vide (≠ 377) en montrant le contenu sur le premier digit de l'affichage (binaire non converti, le contenu est incrémenté avant affichage). Si la mémoire est vide, le programme s'arrête aussi à la fin de la lecture des 1024 positions, et n'affiche rien. Il faut peser sur une touche chiffre quelconque pour programmer. Le programme affiche un P entre chaque cycle et essaie 30 fois au maximum. Dès que le programme, qui compare entre chaque cycle le contenu de la ROM et le modèle, a constaté que l'EPROM est programmée, il confirme la programmation avec 40 cycles supplémentaires. Le fabricant demande 256 cycles pour couvrir le cas pire des EPROMs difficiles à programmer.

Si l'EPROM a été longue à programmer (plus de 20 cycles P), il est suggéré de faire un deuxième ordre de programmation. L'affichage montrera que l'EPROM n'est plus vide, mais comme son contenu est compatible, la programmation pourra s'effectuer.

5. MANIPULATIONS

a) S'il faut programmer une EPROM à partir d'un programme introduit par le clavier.

- Charger le programme que l'on désire mettre dans l'EPROM (à partir de 2000).
- Introduire une EPROM sur le socle supérieur
- Connecter le 24V
- Charger le programme PR026 en 1000 (listing donné ci-après)
- Faire RESET, puis 1104 GO.
  Si l'affichage reste éteint, l'EPROM est vide.
  Sinon elle n'est pas effacée.
- Presser n'importe quelle touche "chiffre". La programmation s'effectue.
  Le nombre de cycles de programmation est visualisé par des "P" successifs qui apparaissent sur l'affichage. Ces "P" sont suivis par des "c", cycles de confirmation.

b) S'il faut copier une EPROM

- Charger le programme PR026 en 1000
- Mettre le commutateur de la plaque processeur sur HOLD
- Placer l'EPROM "modèle" sur le socle supérieur
- Connecter le 24V
- Mettre le commutateur de la plaque processeur sur RUN
  Faire RESET, puis 1000 GO : ceci a pour effe-t de recopier le contenu de l'EPROM (4000 à 5777) dans les positions de mémoire vive 2000-3777.
- Remettre le commutateur de la plaque processeur sur HOLD
- Remplace l'EPROM par une EPROM vide
- Commuter sur RUN (plaque processeur)
- RESET, puis 1104 GO
- Si l'EPROM est bien vide, l'affichage reste éteint
- Presser n'importe quelle touche "chiffre" et la programmation se fait.

Ces manipulations sont peu commodes. C'est la raison pour laquelle nous préparons une ROM , qui contiendra les programmes "PR026" et lecture de bandes papier.

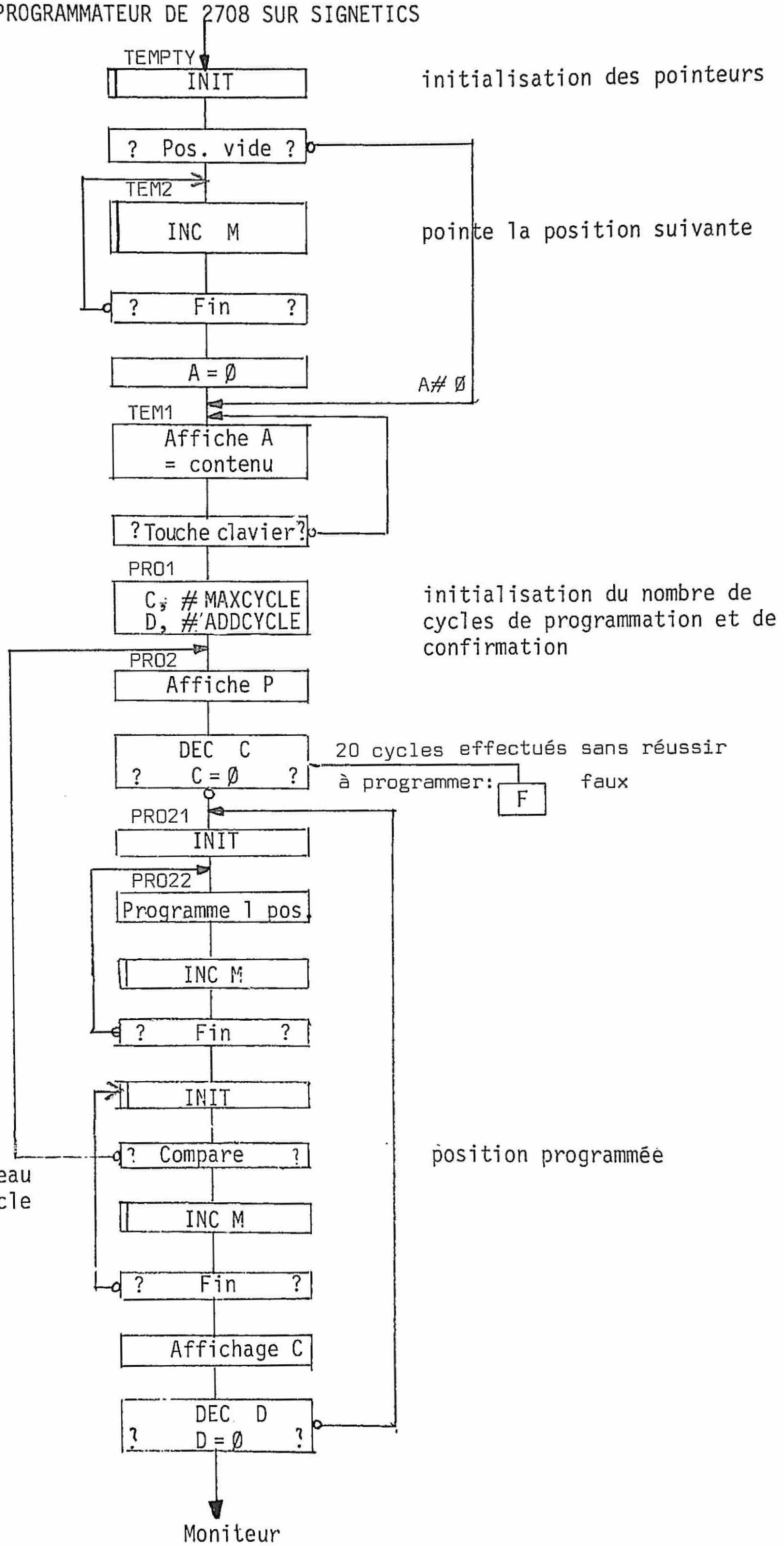
```
                        .TITLE PRO26.SR
                        ;2708 PROM PROGRAMMER
                        ;772512;JLP 7707;JDN770806

                        ;Switch routine not yet typed
                        ;                FC-4    PROGRAMPROM
                        ;                FC-5    GET     6000 -> 2000
                        ;                FC-6    MOVE    sourceMOVEdestMOVElengthMOVI

000000                  DIG0=   0
000001                  DIG1=   1
000007                  CLA=    7
000020                  GKEY=   20

000034                  KPROG=  34
000035                  KGET=   35
000036                  KMOVE=  36
000030                  KLOAD=  30

000000                  ROM0=   0
001000                  RAM=    1000
002000                  RAMEX=  2000
004000                  ROMPRO= 4000
006000                  ROM2=   6000
001350                  ARAM=   RAM+350 ;RAM address
001352                  AROM=   ARAM+2  ;ROM address

000067                  INOC=   67

000163                  LETP=   163
000124                  LETN=   124
000161                  LETF=   161
000130                  LETC=   130


000020                  MAXCYCLE= 20    ;max number of ccycles
000020                  ADDCYCLE= 20    ;number of additional cycles

001000                  .LOC    RAM

                        ;+++    Programm for reading ROM        ;can be used for mo
ing a page
001000      073     014 MOVE:   CALL.   INI     ;4 pages
001002      014 202 352 MOV2:   LOAD    A,@AROM
001005      314 202 350         LOAD    @ARAM,A
001010      073     030         CALL.   INCM
001012      131     166         JUMP,BNE        MOV2
001014      233     000         JUMP    ROM0    ;return to monitor


                        ;---    Routines

                        ;---    INIT    ;initialise

001016      040         INI:    CLR     A
001017      314 002 351         LOAD    ARAM+1,A
001022      314 002 353         LOAD    AROM+1,A
001025      004     004         LOAD    A,#RAMEX/400
001027      314 002 350         LOAD    ARAM,A
001032      004     010         LOAD    A,#ROMPRO/400
001034      314 002 352         LOAD    AROM,A
001037      005     004         LOAD    B,#4    ;4 pages in 2708
001041      027                 RET

                        ;---    INCM    increment ARAM and AROM

001042      014 002 351 INCM:   LOAD    A,ARAM+1
001045      204     001         ADD     A,#1
001047      314 002 351         LOAD    ARAM+1,A
001052      314 002 353         LOAD    AROM+1,A
001055      130     022         JUMP.,ANE RETOUR        ;end of page ?
001057      014 002 350         LOAD    A,ARAM
001062      204     001         ADD     A,#1
001064      314 002 350         LOAD    ARAM,A
001067      014 002 352         LOAD    A,AROM
001072      204     001         ADD     A,#1
001074      314 002 352         LOAD    AROM,A
001077  371 000                 DEC     B
001101      027         RETOUR: RET

001102      033     112 INIT:   JUMP    INI
```

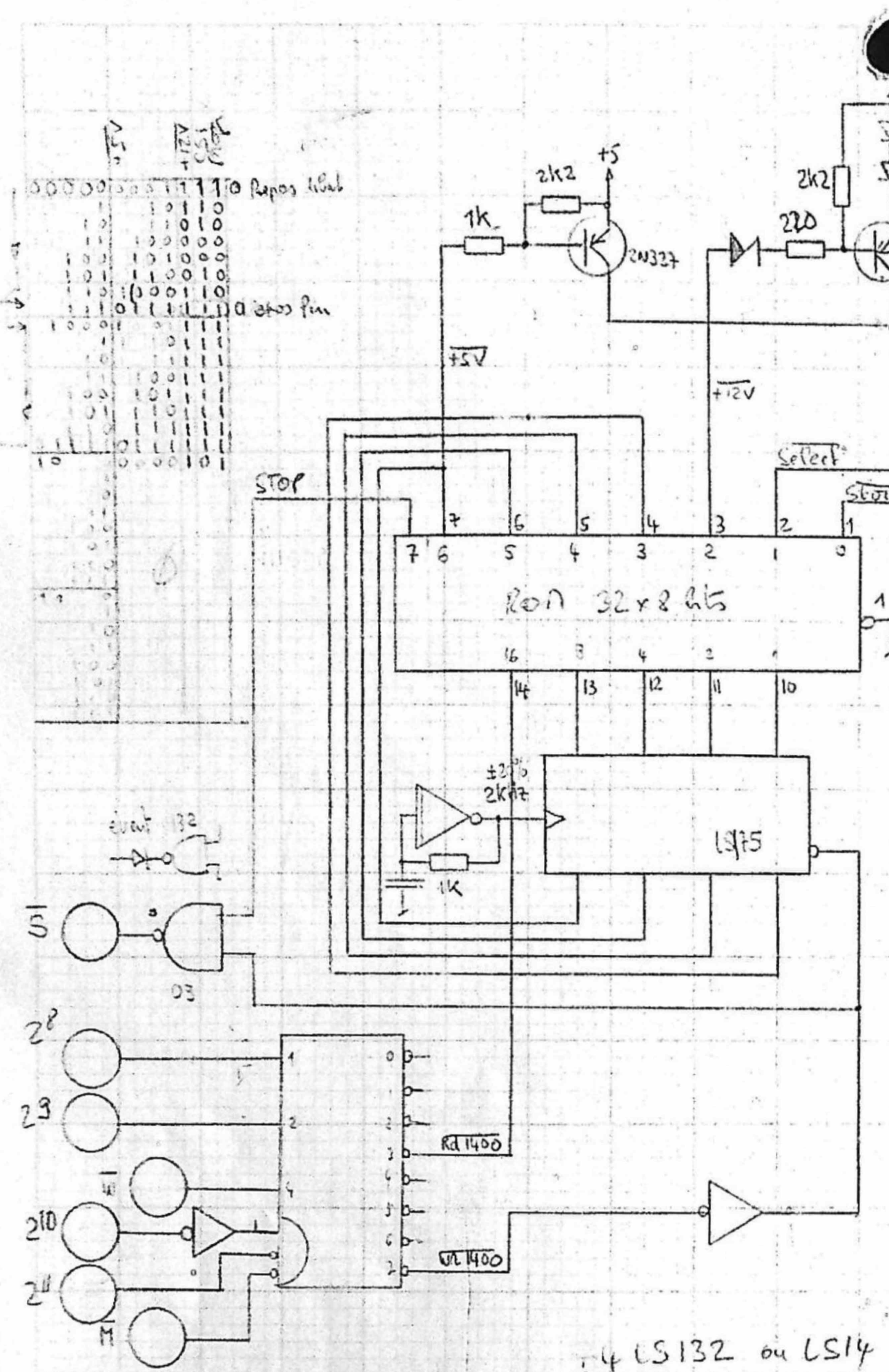
```
                                ;       Programm for wrting an EPROM

001104                          PROM:
                                ;test if ROM empty
001104      073       174       TEMPTY: CALL    INIT
001106      014   202 352       TEM2:   LOAD    A,@AROM
001111      330       006               INCJ.,NE A,TEM1
001113      073       125               CALL    INCM
001115      131       167               JUMP,BNE TEM2
001117      040                         CLR     A
001120      223                         LOAD    L,A     ;CLR LOGICOMP

                                ;Wait key;Display first not empty char plus one if not empt
001121      324       000       TEM1:   LOAD    SDIGO,A
001123      125       007               LOAD    B,SCLA
001125      232       172               JUMP.,GE TEM1
001127      006       020       PR01:   LOAD    C,#MAXCYCLE     ;entry point if not empty
001131      007       020               LOAD    D,#ADDCYCLE
001133      004       163       PR02:   LOAD    A,#LETP
001135      324       000               LOAD    SDIGO,A
001137      372       005               DECJ.,NE C,PR021
001141      004       161       PR03:   LOAD    A,#LETF ;Unprogrammable
001143      324       000               LOAD    SDIGO,A
001145      033       172               JUMP    PR03

001147      073       131       PR021:  CALL    INIT    ;Program
001151      014   202 350       PR022:  LOAD    A,@ARAM
001154      314   202 352               LOAD    @AROM,A
001157      077   002 042               CALL    INCM
001162      131       165               JUMP,BNE PR022
001164      073       114       PR04:   CALL    INIT    ;check
001166      014   202 350       PR06:   LOAD    A,@ARAM
001171      354   202 352               COMP    A,@AROM
001174      230       135               JUMP,NE PR02    ;Try to program again
001176      077   002 042               CALL    INCM
001201      131       163               JUMP,BNE PR06
001203      004       130               LOAD    A,#LETC
001205      324       001               LOAD    SDIG1,A
001207      373       136               DECJ,NE D,PR021
001211      233       000               JUMP    0       ;retur. t/ monitor

                                ;+++    MOVE BLOCK      ;start in ARAM, dest in AROM, lengtl
)n BC

001213      315   002 350       GET:    LOAD    ARAM,B
001216      316   002 351               LOAD    ARAM+1,C
001221      005       022               LOAD    B,#22
001223      006       000               LOAD    C,#0
001225      273       067               CALL    INOC
001227      315   002 352               LOAD    AROM,B
001232      316   002 353               LOAD    AROM+1,C
001235      005       024               LOAD    B,#24
001237      006       000               LOAD    C,#0
001241      273       067               CALL    INOC
001243      014   202 350       GE2:    LOAD    A,@ARAM
001246      314   202 352               LOAD    @AROM,A
001251      077   002 042               CALL    INCM
001254      372       165               DECJ,NE C,GE2
001256      371       163               DECJ,NE B,GE2
001260      233       000               JUMP    0       ;Return to monitor

            000001'             .END
```

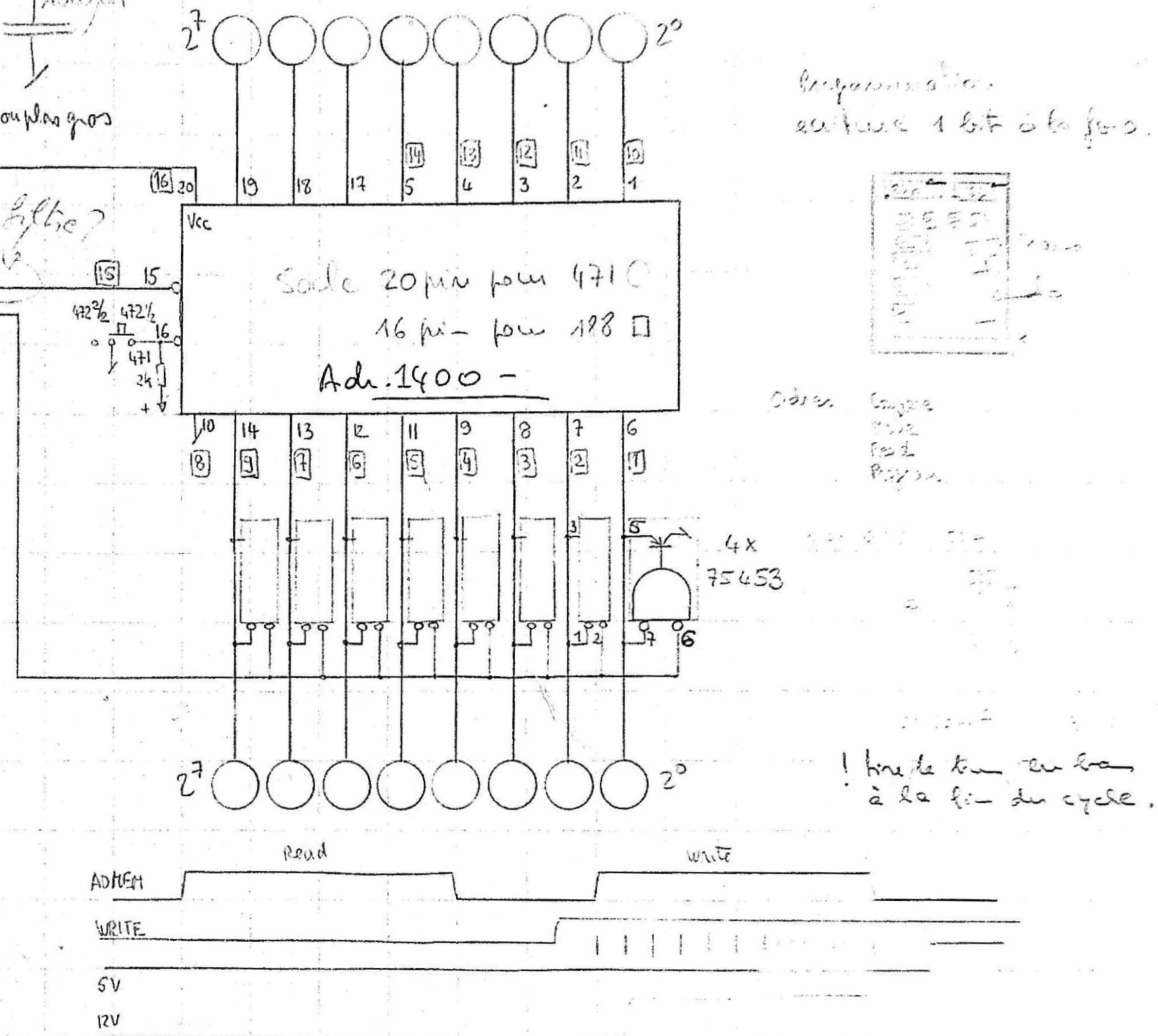
SYMBOL TABLE

```
ADDCYC=  000020          ARAM   =  001350    AROM   =  001352    CLA    =  000007
DIG0   =  000000          DIG1   =  000001    GET       001213    GE2       001243
GKEY   =  000020          INCM      001042    INI       001016    INIT      001102
INOC   =  000067          KGET   =  000035    KLOAD  =  000030    MOVE   =  000036
KPROG  =  000034          LETC   =  000130    LETF   =  000161    LETH   =  000124
LETP   =  000163          MAXCYC=   000020    MOVE      001300    MOV2      001002
PROM      001104          PR01      001127    PR02      001133    PR021     001147
PR022     001151          PR03      001141    PR04      001164    PR06      001166
RAM    =  001000          RAMEX  =  002000    RETOUR    001101    ROMPRO=   004000
ROM0   =  000000          ROM2   =  006000    TEMPTY    001104    TEM1      001121
TEM2      001106
. ABS.    001262      000
ERRORS DETECTED: 0
FREE CORE: 16889. WORDS

PR026,PR026=PR026
```

ORGANIGRAMME DU PROGRAMMATEUR DE 2708 SUR SIGNETICS







Programmateur 471 / 1472 /

Table de write - 188 prof. 47 l.

					ASV next adr.				+12V	Select	Stroke	
					8	4	2	1				
		0	217	1	0	0	0	1	1	1	1	Repos début
		1	226	1	0		1	0	1	1	0	
	1	0	232	1	0		1	1	0	1	0	
	1	1	240	1	0	1	0	0	0	1	0	
1		0	250	1	0	1	0	1	0	1	0	
1		1	262	1	0	1	1	0	0	1	0	
1	1	0	306	1	1	0	0	0	1	1	0	
1	1	1	32	0	0	1	1	1	1	1	1	repos Pin
		0	311	1	1	0	0	1	1	1	1	
		1	319	1	1	0	1	0	1	1	1	
	1	0	327	1	1	0	1	1	1	1	1	
	1	1	335	1	1	1	0	0	1	1	1	
1		0	343	1	1	1	0	1	1	1	1	
1		1	351	1	1	1	1	0	1	1	1	
1	1	0	359	1	1	1	1	1	1	1	1	
1	1	1	367	1	0	1	1	1	1	1	1	
		0	375	1	0	0	0	0	1	0	1	
		1	383	1	0	0	0	1	1	0	1	
	1	0	391	1	0	0	1	0	1	0	1	
	1	1	399	1	0	0	1	1	1	0	1	
1		0	407	1	0	1	0	0	1	0	1	
1		1	415	1	0	1	0	1	1	0	1	
1	1	0	423	1	0	1	1	0	1	0	1	
1	1	1	431	1	0	1	1	1	1	0	1	
		0	439	1	1	0	0	0	1	0	1	
		1	447	1	1	0	0	1	1	0	1	
	1	0	455	1	1	0	1	0	1	0	1	
	1	1	463	1	1	0	1	1	1	0	1	
1		0	471	1	1	1	0	0	1	0	1	
1		1	479	1	1	1	0	1	1	0	1	
1	1	0	487	1	1	1	1	0	1	0	1	
1	1	1	495	1	1	1	1	1	1	0	1	
		0	503	1	0	0	0	0	1	0	1	
		1	511	1	0	0	0	1	1	0	1	
	1	0	519	1	0	0	1	0	1	0	1	
	1	1	527	1	0	0	1	1	1	0	1	
1		0	535	1	0	1	0	0	1	0	1	
1		1	543	1	0	1	0	1	1	0	1	
1	1	0	551	1	0	1	1	0	1	0	1	
1	1	1	559	1	0	1	1	1	1	0	1	
		0	567	1	1	0	0	0	1	0	1	
		1	575	1	1	0	0	1	1	0	1	
	1	0	583	1	1	0	1	0	1	0	1	
	1	1	591	1	1	0	1	1	1	0	1	
1		0	599	1	1	1	0	0	1	0	1	
1		1	607	1	1	1	0	1	1	0	1	
1	1	0	615	1	1	1	1	0	1	0	1	
1	1	1	623	1	1	1	1	1	1	0	1	
		0	631	1	0	0	0	0	1	0	1	
		1	639	1	0	0	0	1	1	0	1	
	1	0	647	1	0	0	1	0	1	0	1	
	1	1	655	1	0	0	1	1	1	0	1	
1		0	663	1	0	1	0	0	1	0	1	
1		1	671	1	0	1	0	1	1	0	1	
1	1	0	679	1	0	1	1	0	1	0	1	
1	1	1	687	1	0	1	1	1	1	0	1	
		0	695	1	1	0	0	0	1	0	1	
		1	703	1	1	0	0	1	1	0	1	
	1	0	711	1	1	0	1	0	1	0	1	
	1	1	719	1	1	0	1	1	1	0	1	
1		0	727	1	1	1	0	0	1	0	1	
1		1	735	1	1	1	0	1	1	0	1	
1	1	0	743	1	1	1	1	0	1	0	1	
1	1	1	751	1	1	1	1	1	1	0	1	
		0	759	1	0	0	0	0	1	0	1	
		1	767	1	0	0	0	1	1	0	1	
	1	0	775	1	0	0	1	0	1	0	1	
	1	1	783	1	0	0	1	1	1	0	1	
1		0	791	1	0	1	0	0	1	0	1	
1		1	799	1	0	1	0	1	1	0	1	
1	1	0	807	1	0	1	1	0	1	0	1	
1	1	1	815	1	0	1	1	1	1	0	1	
		0	823	1	1	0	0	0	1	0	1	
		1	831	1	1	0	0	1	1	0	1	
	1	0	839	1	1	0	1	0	1	0	1	
	1	1	847	1	1	0	1	1	1	0	1	
1		0	855	1	1	1	0	0	1	0	1	
1		1	863	1	1	1	0	1	1	0	1	
1	1	0	871	1	1	1	1	0	1	0	1	
1	1	1	879	1	1	1	1	1	1	0	1	
		0	887	1	0	0	0	0	1	0	1	
		1	895	1	0	0	0	1	1	0	1	
	1	0	903	1	0	0	1	0	1	0	1	
	1	1	911	1	0	0	1	1	1	0	1	
1		0	919	1	0	1	0	0	1	0	1	
1		1	927	1	0	1	0	1	1	0	1	
1	1	0	935	1	0	1	1	0	1	0	1	
1	1	1	943	1	0	1	1	1	1	0	1	
		0	951	1	1	0	0	0	1	0	1	
		1	959	1	1	0	0	1	1	0	1	
	1	0	967	1	1	0	1	0	1	0	1	
	1	1	975	1	1	0	1	1	1	0	1	
1		0	983	1	1	1	0	0	1	0	1	
1		1	991	1	1	1	0	1	1	0	1	
1	1	0	999	1	1	1	1	0	1	0	1	
1	1	1	1007	1	1	1	1	1	1	0	1	
		0	1015	1	0	0	0	0	1	0	1	
		1	1023	1	0	0	0	1	1	0	1	
	1	0	31	0	1	1	1	1	1	1	1	repos Pin
	1	1	39	0	1	1	1	1	1	1	1	
1		0	47	0	1	1	1	1	1	1	1	
1		1	55	0	1	1	1	1	1	1	1	
1	1	0	63	0	1	1	1	1	1	1	1	
1	1	1	71	0	1	1	1	1	1	1	1	
		0	79	0	1	1	1	1	1	1	1	
		1	87	0	1	1	1	1	1	1	1	
	1	0	95	0	1	1	1	1	1	1	1	
	1	1	103	0	1	1	1	1	1	1	1	
1		0	111	0	1	1	1	1	1	1	1	
1		1	119	0	1	1	1	1	1	1	1	
1	1	0	127	0	1	1	1	1	1	1	1	
1	1	1	135	0	1	1	1	1	1	1	1	
		0	143	0	1	1	1	1	1	1	1	
		1	151	0	1	1	1	1	1	1	1	
	1	0	159	0	1	1	1	1	1	1	1	
	1	1	167	0	1	1	1	1	1	1	1	
1		0	175	0	1	1	1	1	1	1	1	
1		1	183	0	1	1	1	1	1	1	1	
1	1	0	191	0	1	1	1	1	1	1	1	
1	1	1	199	0	1	1	1	1	1	1	1	
		0	207	0	1	1	1	1	1	1	1	
		1	215	0	1	1	1	1	1	1	1	
	1	0	223	0	1	1	1	1	1	1	1	
	1	1	231	0	1	1	1	1	1	1	1	
1		0	239	0	1	1	1	1	1	1	1	
1		1	247	0	1	1	1	1	1	1	1	
1	1	0	255	0	1	1	1	1	1	1	1	
1	1	1	263	0	1	1	1	1	1	1	1	
		0	271	0	1	1	1	1	1	1	1	
		1	279	0	1	1	1	1	1	1	1	
	1	0	287	0	1	1	1	1	1	1	1	
	1	1	295	0	1	1	1	1	1	1	1	
1		0	303	0	1	1	1	1	1	1	1	
1		1	311	0	1	1	1	1	1	1	1	
1	1	0	319	0	1	1	1	1	1	1	1	
1	1	1	327	0	1	1	1	1	1	1	1	
		0	335	0	1	1	1	1	1	1	1	
		1	343	0	1	1	1	1	1	1	1	
	1	0	351	0	1	1	1	1	1	1	1	
	1	1	359	0	1	1	1	1	1	1	1	
1		0	367	0	1	1	1	1	1	1	1	
1		1	375	0	1	1	1	1	1	1	1	
1	1	0	383	0	1	1	1	1	1	1	1	
1	1	1	391	0	1	1	1	1	1	1	1	
		0	399	0	1	1	1	1	1	1	1	
		1	407	0	1	1	1	1	1	1	1	
	1	0	415	0	1	1	1	1	1	1	1	
	1	1	423	0	1	1	1	1	1	1	1	
1		0	431	0	1	1	1	1	1	1	1	
1		1	439	0	1	1	1	1	1	1	1	
1	1	0	447	0	1	1	1	1	1	1	1	
1	1	1	455	0	1	1	1	1	1	1	1	
		0	463	0	1	1	1	1	1	1	1	
		1	471	0	1	1	1	1	1	1	1	
	1	0	479	0	1	1	1	1	1	1	1	
	1	1	487	0	1	1	1	1	1	1	1	
1		0	495	0	1	1	1	1	1	1	1	
1		1	503	0	1	1	1	1	1	1	1	
1	1	0	511	0	1	1	1	1	1	1	1	
1	1	1	519	0	1	1	1	1	1	1	1	

PROGRAMMATEUR DE 74S471 et 74S288

LISTING

```
.TITLE M02815.SR        JDN;DZILOG

;771229;771217;771210;770724;JDN 770130

; MONITOR FOR Z80 CH  D A U P H I N
;ROM 1 Paper loader and EPROM programmer for 471 and 288


;Orders:        FG-0    LOAD:   relativeaddressLOAD
;               FG-4    PROGRAM PROM 471 (0 displayed when order)
;                       Type 40 FG-4 for 74S288
;               FG-5    GET     4000 -> 2000
;               FG-6    MOVE    sourceMOVElengthMOVEdestinationMOVE

        .Z80

        ;===   Peripheral and parameter definitions

   0    DIG0=   0
   1    DIG1=   1
   2    DIG2=   2
   3    DIG3=   3

   6    KP=     6
   7    CLR=    7
 200    FULL=   200
  37    NCLA=   37
   7    NDIG=   7
  10    FKEY=   10
  20    GKEY=   20


 167    LETA=   167
 130    LETC=   130
 161    LETF=   161
  70    LETL=   70
 163    LETP=   163     ;programmer only
 100    MOINS=  100

   0    ROM0=   0
 400    ROM1=   400
1000    RAM=    1000
2000    RAMEX=  2000
1400    EPROM=  1400
1400    LROM=   1400

   0    L471=   0
  40    L288=   32.
6000    ROM5=   6000

1374    STACK=  RAM+374
1304    SAVE=   RAM+304

  12    FONC=   12
  45    INOC0=  45
  50    INOC=   53
  55    INOC=   55


000400                  .LOC    RAM

                        FONC2:
                        ;       LOAD    A,#00111000     ;Init 8251 before jumping to program
                        ;       LOAD    $SPR,A
                        ;       LOAD    A,SREMODE
                        ;       LOAD    $SPR,A
000400 076 316                  LOAD    A,#MODE
000402 323 011                  LOAD    $SPR,A
000404 076 005                  LOAD    A,#COMMAND
000406 323 011                  LOAD    $SPR,A

000410 170                      LOAD    A,B
000411 346 037                  AND     A,#NCLA
000413 376 036                  COMP    A,#FKEY+GKEY+6
000415 050 030                  JUMP.,EQ MOVE
000417 376 030                  COMP    A,#FKEY+GKEY
000421 050 131                  JUMP.,EQ LOAD
000423 376 034                  COMP    A,#FKEY+GKEY+4
000425 312 206 001              JUMP,EQ FROM471
000430 376 035                  COMP    A,#FKEY+GKEY+5
000432 302 000 016              JUMP,NE ROM5

000435 315 253 001      GET:    CALL    INIT
000440 032              MOV2:   LOAD    A,(DE)
000441 167                      LOAD    (HL),A
000442 023                      INC     DE
000443 043                      INC     HL
000444 020 372                  DECJ,NE B,MOV2
000446 307                      RST     0

000447 345              MOVE:   PUSH    HL      ;Destination
000450 041 000 030              LOAD    HL,#14000       ;Lettre L
000453 357                      CALL    INOC0
000454 345                      PUSH    HL      ;Length
000455 041 000 026              LOAD    HL,#12000       ;Lettre A
000460 357                      CALL    INOC0   ;Destination
000461 301                      POP     BC
000462 101                      LOAD    B,C     ;Get length in B
000463 321                      POP     DE
000464 030 352                  JUMP    MOV2


                        ;RELATIVE LOADER FOR PDP11 FORMAT
                        ;FOR DAUPHIN Z80 WITH PROM PROGRAMMER


  10    PR=     10      ;DATA INPUT
  11    SPR=    PR+1    ;STATUS INPUT
 316    MODE=   316     ;(8+2,no parity,x16)
   5    ENABLE= 5       ;In and out
  40    RTS=    40
   2    DTR=    2
 100    RETOUR= 100
   5    CONFIG=ENABLE

   2    FULL=   2       ; FULL IS BIT 2?
   1    EMPTY=  1
 200    PROG=   200
```

```
        ;---    CALL    RBYTE   ;READ NEXT BYTE FROM PAPER READER
                                ;ADD VALUE TO CHECKSUM IN E
                                ;DO NOT CHANGE BYTE COUNT IN D
                                ;RETURN WITH CHARACTER IN A AND B
                                ;LOAD FLAGS ACCORDING TO CHECKSUM
                                ;DO NOT CHANGE D, C AND HL

000466 333 011          RBYTE:  LOAD    A,SSPR
000470 346 002                  AND     A,#FULL         ;?DATA HERE?
000472 050 372                  JUMP,EQ RBYTE
000474 333 010                  LOAD    A,SPR           ;YES,GET DATA
000476 323 006                  LOAD    SHP,A
000500 107                      LOAD    B,A             ;SAVE IN B
000501 203                      ADD     A,E             ;ADD TO CHECKSUM
000502 137                      LOAD    E,A
000503 170                      LOAD    A,B             ;PUT DATA IN A AGAIN
000504 311                      RET

000505 315 066 001      HEADER: CALL    RBYTE
000510 267                      OR      A,A             ;?IS IT A 0 ?
000511 050 372                  JUMP,EQ HEADER
000513 075                      DEC     A               ;NO, ?IS IT A 1 ?
000514 040 367                  JUMP,NE HEADER          ;MAYBE A RECOVERABLE ERROR
000516 315 066 001              CALL    RBYTE
000521 267                      OR      A,A             ;?IS IT A 0 ?
000522 040 361                  JUMP,NE HEADER          ;MAYBE A RECOVERABLE ERROR
000524 036 001                  LOAD    E,#1            ;INIT. CHECKSUM
000526 315 066 001              CALL    RBYTE           ;GET BYTE COUNT
000531 326 005                  SUB     A,#5
000533 127                      LOAD    D,A             ;LOAD   COUNTER
000534 315 066 001              CALL    RBYTE
000537 267                      OR      A,A             ;?IS IT STILL A 0 ?
000540 040 102                  JUMP,.NE ERROR
000542 315 066 001              CALL    RBYTE           ;OK,PUT ADDRESS IN HL
000545 157                      LOAD    L,A
000546 315 066 001              CALL    RBYTE
000551 147                      LOAD    H,A
000552 025                      DEC     D               ;?IS IT THE LAST BLOCK ?
000553 311                      RET

                        ;***    LOAD80  ;PAPER LOADER IN PDP11 FORMAT
                                        ;HL POINT LOADED ADDRESSES

                        LOAD:
000554 345                      PUSH    HL
000555 315 105 001              CALL    HEADER  ;Get absolute address in HL
000560 301                      POP     BC
000561 267                      OR      A,A     ;Clear carry
000562 355 102                  SUBC    HL,BC
000564 345                      PUSH    HL      ;Difference on stack
000565 140                      LOAD    H,B
000566 151                      LOAD    L,C
                                ;Get address for first block
                        DATABLOCK:
000567 315 066 001              CALL    RBYTE           ;NO,READ AND STORE NEXT DATA
000572 333 007                  LOAD    A,SCLA  ;Test if data may not be loaded
000574 346 010                  AND     A,#FKEY
000576 312 212 001              JUMP,EQ DATA1
000601 174                      LOAD    A,H
000602 376 004                  COMP    A,#RAMEX/400
000604 070 006                  JUMP,.LO DATA2
000606 376 007                  COMP    A,#(RAMEX+LROM)/400
000610 060 002                  JUMP,.HS DATA2
000612 170              DATA1:  LOAD    A,B
000613 167                      LOAD    (HL),A
000614 323 000          DATA2:  LOAD    $DIG0,A
000616 043                      INC     HL
000617 025                      DEC     D               ;?LAST DATA IN BLOCK ?
000620 040 345                  JUMP,NE DATABLOCK
000622 315 066 001              CALL    RBYTE           ;?CHECKSUM GOOD?
000625 040 015                  JUMP,.NE ERROR          ;YES, READ NEXT BLOCK
000627 315 105 001              CALL    HEADER
000632 312 252 001              JUMP,EQ LOG
000635 301                      POP     BC
000636 305                      PUSH    BC
000637 267                      OR      A,A
000640 355 102                  SUBC    HL,BC
000642 030 323                  JUMP    DATABLOCK
000644 076 161          ERROR:  LOAD    A,#LETF
000646 323 003                  LOAD    $DIG3,A
000650 030 372                  JUMP    ERROR

                        ;LAST BLOCK
000652 307              LOG:    RST     0


                        ;Programmer

                        ;---    Routines

000653 072 354 002      INIT:   LOAD    A,SAL.
000656 107                      LOAD    B,A
000657 021 000 003              LOAD    DE,#EPROM
000662 041 000 004              LOAD    HL,#RAMEX
000665 311                      RET


                        ;===    Program

000666 175              PROM1471: LOAD  A,L
000667 062 354 002              LOAD    SAL.,A
000672 315 253 001              CALL    INIT
000675 032              EMP2:   LOAD    A,(DE)
000676 267                      OR      A,A
000677 040 004                  JUMP,.NE EMP4
000701 023                      INC     DE
000702 043                      INC     HL
000703 020 370                  DECJ,NE B,EMP2
000705 323 000          EMP4:   LOAD    $DIG0,A ;If any wrong char, display it (+1)
000707 107                      LOAD    B,A
000710 333 007                  LOAD    A,SCLA
000712 346 200                  AND     A,#FULL
000714 170                      LOAD    A,B
000715 050 366                  JUMP,EQ EMP4

000717 315 253 001      CHECK:  CALL    INIT
000722 032              CHE2:   LOAD    A,(DE)  ;If not compatible, display address
000723 117                      LOAD    C,A
000724 176                      LOAD    A,(HL)
000725 057                      CPL     A
000726 241                      AND     A,C
000727 040 042                  JUMP,.NE RETMON
000731 032                      LOAD    A,(DE)  ;Try to program if not equal
000732 276                      COMP    A,(HL)
000733 040 020                  JUMP,.NE PROG3
000735 023              CHE4:   INC     DE
000736 043                      INC     HL
000737 020 361                  DECJ,NE B,CHE2

000741 315 253 001              CALL    INIT
000744 032              CHE6:   LOAD    A,(DE)
000745 276                      COMP    A,(HL)
000746 040 274                  JUMP,NE ERROR
000750 023                      INC     DE
000751 043                      INC     HL
000752 020 370                  DECJ,NE B,CHE6
000754 307                      RST     0

000755 016 001          PROG3:  LOAD    C,#1    ;Program 1 bit at a time
000757 176              PROBIT: LOAD    A,(HL)
000760 241                      AND     A,C
000761 050 002                  JUMP,.EQ PROG2
000763 057                      CPL     A
000764 022                      LOAD    (DE),A
000765 313 001          PROG2:  RL      C
000767 060 366                  JUMP,CC PROBIT
000771 030 342                  JUMP    CHE4

000773 337              RETMON: CALL    INCO
000774 303 012 000              JUMP    FANC

                        .END
20/12/77  21.03.33              ASSEMBLY CORRECT.
```

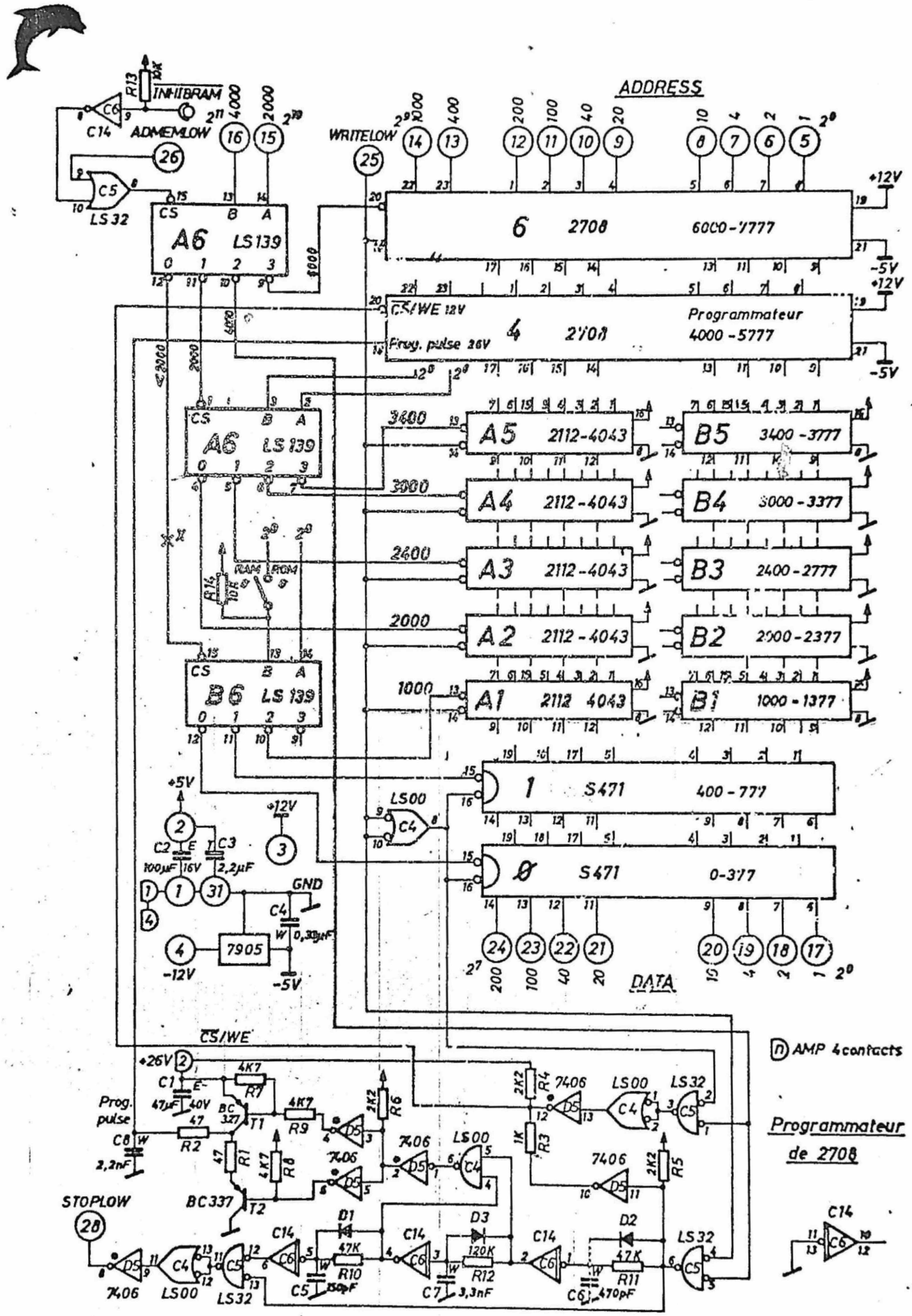


Schéma de la plaque
mémoire et programmateur