SMAKY6

FACILITES SYSTEME

Février 1982

EPSITEC-system sa

CONTENU:

1. INTRODUCTION

Chaque SMAKY6 est doté d'un programme de base appelé SYSMON et utilisant les adresses 0-7777.

Ce programme contient:

- les programmes d'initalisation et de dialogue
- les routines système (appels système)
- les routines graphiques
- le programme moniteur.

Cette notice concerne la révision 1. La révision 2, prévue pour 1981, restera compatible sauf pour les appels marqués d'un □ .

Les appels les plus importants sont repérés par un ● .

2.2 FACILITES SYSTEME

Un certain nombre de routines sont mises à disposition de l'utilisateur avec un triple objectif:

- faciliter la programmation
- cacher l'aspect compliqué des périphériques
- permettre la modification ultérieure du système sans modification des programmes des utilisateurs.

Les appels système permettent d'ignorer les adresses réelles et de dire ce que l'on veut faire sans se préoccuper des détails d'implémentation.
Ces routines sont caractérisées par un numéro, et appelées par une instruction de restart, un CALL ·40 , suivi du numéro de la routine. L'adresse de la routine correspondante est recherchée dans une table, et la routine est exécutée. Cette technique ralentit l'exécution, mais raccourcit les programmes (appels en 2 bytes) et augmente la flexibilité et l'indépendance aux révisions successives du système.

Les numéros donnés aux routines résultent d'une évolution historique et n'ont aucune signification particulière.

Les appels système peuvent être classés en différentes catégories expliquées en détail plus loin.

- extension des instructions du Z80
- initialisation des indirections et interruptions
- initialisation de l'écran
- test de la dimension de la mémoire
- bruitages
- mise en place des caractères et textes sur l'écran
- lecture du clavier
- lecture et affichage de nombres introduits au clavier
- recherche dans les tables
- transferts avec les périphériques
- calcul du temps
- transferts spéciaux avec les périphériques

L'utilisateur peut définir 128. appels supplémentaires dont le numéro est compris entre 200 (octal) et 377. L'adresse de la table des adresses des appels doit être chargée préalablement dans la position TABEX (= 42532).

2.3 EXTENSION DU REPERTOIRE D'INSTRUCTIONS

Les routines suivantes, nécessaires pour les programmes système, ont été mises à disposition de l'utilisateur.

56 ?COMPHL équivalent à COMP HL,DE

Le seul registre modifié est le registre de flag:

Z = 1	càd	EQ	si	HL = DE	
Z = 0	càd	NE	si	HL ≠ DE	
C = 1	càd	LO	si	HL < DE	(nombres logiques positifs)
C = 0	càd	HS	si	HL ≥ DE	

Cette routine ne permet pas de comparer directement des nombres arithmétiques (en complément à 2) ou des nombres signés.
Le routine comporte 6 bytes (voir listing complet) et s'exécute en 20 µs.
Le temps d'exécution de l'appel est de 200 µs environ.

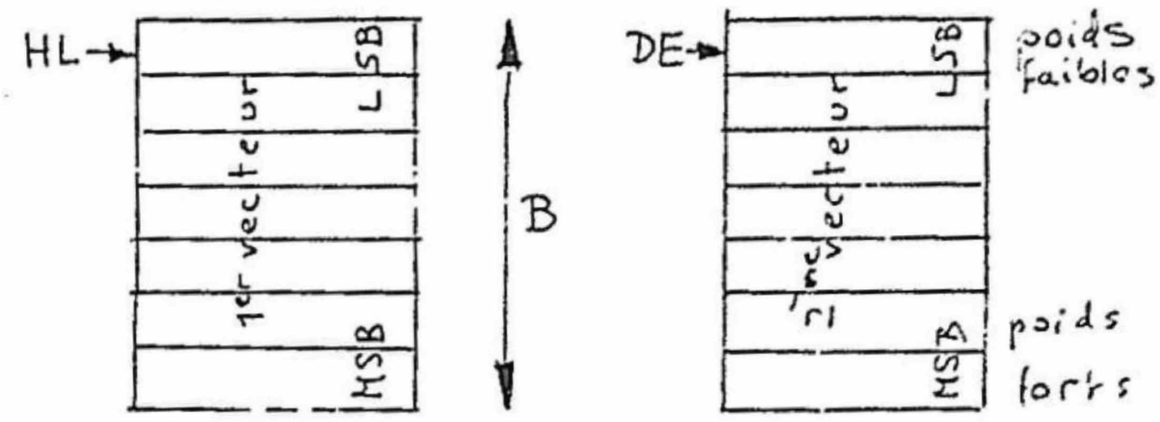
PROGRAMME DE TEST DE LA ROUTINE ET EXEMPLE D'APPLICATION

Le programme attend deux nombres séparés par un caractère quelconque et affiche ensuite le signe correspondant au résultat de la comparaison.

```
TEST2:  .W      ?RETURN          ;retour de ligne (voir p.2.9-2)
        .W      ?INOC            ;lecture nombre octal. (voir p.2.10-2)
        .W      ?SPACE
        PUSH    HL
        .W      ?INOC
        .W      ?SPACE
        EX      HL,DE            ;deuxième nombre dans DE
        POP     HL               ;premier nombre dans HL
        .W      ?COMPHLDE
T1:     JUMP.,LO T22             ;saut si <
        JUMP.,EQ T24             ;saut si =

        LOAD    A,# '>           ; >
T21:    .W      ?DICAR           ;affichage
        JUMP    TEST2
T22:    LOAD    A,# '<
        JUMP    T21
T24:    LOAD    A,# '=
        JUMP    T21
```

116 ?COMPTIME Compare 2 nombres en mémoire.
Equivalent à COMP (HL),(DE),B.

Les vecteurs sont des nombres positifs, organisés poids faibles en tête, et sont pointés respectivement par HL et DE. La longueur est de B bytes.
Des nombres binaires,
BCD, ou des heures
peuvent être comparés.



EQ: valeurs égales
NE: ≠
LO: valeur 1er vect < valeur 2e
HS: valeur 1er vect ⩾ valeur 2e
SS: valeur 1er vect > valeur 2e

En sortie HL et DE pointent la fin+1 des vecteurs. ATTENTION: dans les révisions 1-7 et précédentes, HL et DE sont rendus inversés si B est impair.

TEST ET EXEMPLE D'APPLICATION.

Les vecteurs représentent les JOURS-HEURES-MINUTES, codés en BCD (voir p.2.13.1)

```
              LONGTIME = 3
TEST5:    LOAD    HL,# TIM1
          LOAD    DE,# TIM2
          LOAD    B,# LONGTIME
          .W      ?COMPTIME
          JUMP    T1                ;voir exemple ci-dessus
TIM1:     .BLKB   LONGTIME          ;1er vecteur (placer les valeurs avec
TIM2:     .BLKB   LONGTIME          ;2e vecteur   le moniteur)
```

● 57 ?MUL multiplication 16 bits*16 bits

Le résultat à 32 bits, et l'opération est équivalente à

DE*BC + HL ⟹ HLDE

Si l'on veut simplement multiplier, il faut donc initialiser HL à zéro auparavant. CS si dépassement de capacité.

● 60 ?DIV division 32 bits : 16 bits.

L'opération est équivalente à

HLDE : BC ⟹ DE, reste dans HL

S'il y a un dépassement de capacité (quotient > 16 bits), le retour se fait CS.

● 124 ?BINBCD Convertit de binaire (16 bits) en BCD (6 digits).

Le résultat est donné dans HL, et le résultat BCD dans AHL.

La valeur maximale est

A	HL				
0	6	5	5	3	5	base 10

● 125 ?BCDBIN Convertit de BCD (4 digits) en binaire (16 bits).

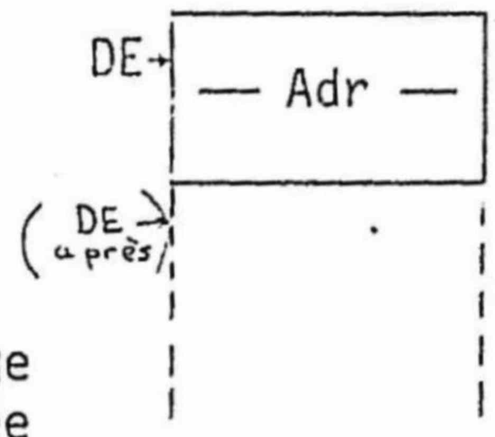
Le nombre est donné dans HL, le résultat se trouve dans HL et aucun autre registre n'est modifié.

Exemple: affichage octal de l'équivalent d'un nombre tapé en décimal.

```
          .W      ?INDEC, ?BCDBIN, ?SPACE, ?AFOHL
```

55 ?JUMPIDE équivalent à JUMP @(DE)

Cet appel permet de sauter à l'adresse pointée par DE



Le registre DE est le seul registre modifié, mais il pointe l'adresse suivante, ce qui peut permettre le saut à plusieurs programmes successivement.

La routine JUMPIDE est une routine simple, que l'on a avantage à incorporer directement dans son programme si elle est exécutée fréquemment:

```
JUMPIDE:  EX      (SP),HL       ;sauve HL et corrige le stack
          EX      HL,DE
          LOAD    E,(HL)
          INC     HL
          LOAD    D,(HL)
          INC     HL
          EX      HL,DE
          EX      (SP),HL
          RET
```

Le temps d'exécution de la routine est 40 µs.
Le temps d'exécution de l'appel ?JUMPIDE est 240 µs.
On perd donc 200 µs à chaque appel système.

EXEMPLES D'APPLICATION.

1) Saut à l'un des 4 programmes selon une touche chiffre pressée.

```
XAP1:     .W      ?GETCAR       ;lecture clavier  (voir p. 2.9-2)
          COMP    A,# 'Ø
          JUMP,LO ERROR         ;les touches autres que 0,1,2,3
          COMP    A,# '3+1      ;sont rejetées
          JUMP,HS ERROR
          SUB     A,# 'Ø
          RL      A             ;*2
          LOAD    DE,#TBLE
          ADD     A,E           ;la table doit être dans la même
          LOAD    E,A           ;page, car il n'y a pas de correction sur D
          .W      ?JUMPIDE
          JUMP    XAP1          (ou  JUMP ERROR:  on n'a pas pressé sur
                                                  les touches prévues)
TBLE:     .W      DOØ,DO1,DO2,DO3     ;adresses des programmes à exécuter

DOØ:      ...                   ;premier programme
          ...
```

TBLE: DOØ, DO1, DO2, DO3

2) Le programme suivant mesure le temps d'exécution des appels. L'appel est fait 100000 fois, pour permettre la mesure du temps avec une montre.

```
?JUMPIDE = 54*400+347
TEST1:    IOF
          LOAD    BC,# 100000.
TES2:     LOAD    DE,# ADIND  }  équivalent à  JUMP @ADIND
          .W      ?JUMPIDE    }
;..
ADIND:    .W      TES4
;...
TES4:     LOAD    $HP,A         ;crépitement dans le haut-parleur pendant l'exécution
          DEC     BC            }
          LOAD    A,B           } la durée de ces instructions, < 12 µs,
          OR      A,C           } doit être déduite de la durée d'un appel.
          JUMP,NE TES2          }
          TRAP
```

REMARQUE 1: Dans tous les exemples donnés, la pile (stack) n'est pas initialisée. Le moniteur place la pile 50 positions octales en dessous de l'écran. Le débordement éventuel dans l'écran met en évidence les piles très grandes et les erreurs.
SMILE, par contre, place la pile en fin de mémoire

63 ?CALLIDE équivalent à CALL @(DE)

La routine à l'adresse pointée par DE est exécutée.
Le registre DE est modifié et pointe l'adresse suivante.

EXEMPLE D'APPLICATION: quatre routines doivent être exécutées dans l'ordre 1-2-4-2-3-4
— (Cet exemple est artificiel, il serait plus court d'appeler directement les routines.)

```
TAROU:  .W      R1,R2,R3,R4 ;table d'adresse des routines
        ;---
TEST:   LOAD    DE,# TAROU
        .W      ?CALLIDE    ;lit R1
        .W      ?CALLIDE    ;R1 et R2 ne doivent pas modifier DE
        INC     DE
        INC     DE          ;saute par-dessus l'adresse de la routine R3
        .W      ?CALLIDE
        ;...
        LOAD    B,# 3
        LOAD    DE,# TAROU+2
TS2:    .W      ?CALLIDE        ;la routine appelée ne doit pas modifier B,
        DECJ,NE B,TS2           ;ni DE
        ;...
```

117 ?GETARG équivalent à LOAD DE,@(SP)+ (valeur de SP avant l'appel).

Cet appel permet de chercher des arguments qui suivent l'appel de la routine. L'adresse de retour est modifiée pour revenir après l'argument.

EXEMPLE: routine de multiplication par une constante donnée depuis l'appel

```
;calcul de HL
        CALL    MULT
        .W      FACTEUR1
;on a calculé  HL*FACTEUR1
MULT:   .W      ?GETARG
        MUL     HL,DE   ;ou routine équivalente
        RET
```

2.4 RECHERCHE DANS DES TABLES

● 61 ?JUMPCAR Saute selon le caractère.

Une table pointée par DE terminée par un caractère Ø contient successivement les codes de caractères ASCII et des adresses de routines exécutées lorsque le contenu de A correspond au code ASCII. Si le caractère n'est pas trouvé, le retour se fait, et DE pointe la première position d'une deuxième table suivante éventuelle.

EXEMPLE: voir p. 2.9-1.

62 ?SCAR Cherche un caractère dans une table .BW.

La table doit se terminer par un .B Ø.
Si le caractère est trouvé, on revient avec CC et DE pointant le mot associé.
Si le caractère n'est pas trouvé, le retour se fait avec CS et DE pointant le début d'une table suivante éventuelle.

Exemple: on veut modifier un facteur correctif de 16 bits selon la touche 0,1,2,3 pressée.

```
MOD:    .W      ?GETCAR              ;modifie HL et DE
        LOAD    DE,# TABCOR
        .W      ?SCAR
        JUMP,CS ERROR
        EX      HL,DE
        LOAD    E,(HL)    } équivalent à  LOAD DE,(DE)
        INC     HL
        LOAD    D,(HL)
        LOAD    FACTEURCOR,DE
        ...                              ;suite

TABCOR: .BW     '0, FACTØ
        .BW     '1, FACT1
        .BW     '2, FACT2
        .BW     '3, FACT3
        .B      Ø
```

34 ?BRANCH Cherche un nom en mémoire

HL pointe le nom donné.

En mémoire, le mot doit être défini par

.ASCII "<252>mot_clé<200>

Le retour se fait avec CS si non trouvé.

Si le nom a été trouvé, HL pointe 1 position plus loin que la fin du nom donné et DE la position mémoire qui suit le mot clé et le byte 200 .

Cette routine présente aussi un danger de saut à une adresse non voulue. La présence du 252 et du 200 diminue la probabilité d'erreur à 10^{-9} ou moins.

2.5 INITIALISATION DES INDIRECTIONS ET INTERRUPTIONS

12 ?IRST50 Initialisation de l'adresse 50

Sur les SMAKY floppy cette adresse est librement utilisable pour appeler une routine fréquemment utilisée. L'appel n'utilise qu'un seul byte.

L'appel ?IRST50 permet de définir l'adresse qui doit suivre le saut en 50. La durée de l'appel est de 27 µs, au lieu de 7µspour l'appel direct.

EXEMPLE: la routine TRUC est souvent utilisée. Son nom est changé en TTRUC et on écrit une fois dans le programme, avant que la routine soit appelée

```
TRUC =  50
LOAD    HL,# TTRUC      ;si HL ne doit pas être modifié,
.W      ?IRST50         ;on peut le sauver sur la pile
```

13 ?ITRAP Initialisation de l'adresse 60.

7 ?IRST10
10 ?IRST20
} Réservés pour le floppy disque

11 ?IRST70
17 ?ENI50
} Utilisable pour une gestion particulière des interruptions. Pour le spécialiste uniquement

14 ?IADNMI Initialise l'adresse de la routine exécutée en cas de NMI (touche BREAK)

Le système initialise cette adresse comme un Ø, le moniteur l'initialise comme un TRAP (arrêt de l'exécution) et le CLI l'initialise pour se réinitialiser.

Comme exemple d'application, remarquons que l'éditeur initialise le NMI de façon neutre, pour éviter que l'action de la touche "BREAK" pendant les déplacements de pointeur n'empêche de récupérer le fichier. Le NMI est activé toutefois avant chaque transfert avec un périphérique pour reprendre le contrôle.

ATTENTION: L'instruction RETN doit terminer les routines appelées par un NMI, afin de rétablir l'intégrité des interruptions.

Exemple: l'utilisateur veut que la touche NMI n'ait pas d'effet.
La routine appelée est alors réduite à un RETN (return).

```
Le programme commence par      LOAD  HL,# DORET
                               .W    ?IADNMI
                               ...
et contient ailleurs    DORET: RETN
```

2.5-1 Les registres ne sont pas sauvés automatiquement.

37 ?INTUS Initialise la routine exécutée en cas d'interruption simple non due à l'horloge.

Le système initialise une position mémoire à Ø et n'accepte pas d'autre interruption que le 50 Hz. Si un saut en 70 a lieu (interruption, erreur de programme, saut à une mémoire inexistante), le message "INT" apparaît. Si l'utilisateur veut qu'une routine particulière soit exécutée à la place, il doit définir l'adresse de sa routine en écrivant le début du programme.

Exemple:

```
        ...
        LOAD    HL,# ADI70
        .W      ?INTUS
        ION
        ...
ADI70:  TRAP
```

La routine d'interruption de l'utilisateur doit sauver les registres qu'elle utilise, sauf F,A,BC,DE,HL qui sont déjà sauvés.

4 ?IRTC Initialisation de l'adresse d'une routine exécutée toutes les 20 ms.

Le système initialise cette adresse à Ø, et aucune routine n'est exécutée.

EXEMPLE D'APPLICATION. L'utilisateur veut que la date, l'heure et les minutes soient affichées au coin supérieur de l'écran. Toutes les 20 ms, le compte des centièmes et des secondes est incrémenté. Toutes les minutes, le nombre des heures et minutes est augmenté et l'affichage est effectué. Toutes les 24 heures le jour est incrémenté; tous les 28, 30 ou 31 jours, le mois est incrémenté.

```
;au début du programme
        ...                     ;initialisation de l'heure et de la date (voir TEMPS:)
        LOAD    HL,# AFTEMPS
        .W      ?IRTC
        CALL    AFTI            ;pour afficher immédiatement
;routine de comptage du temps et affichage toutes les minutes
AFTEMPS:LOAD    HL,# TEMPS
        .W      ?INCSEC         ;voir p. 2.13-1
        RET,CC
        .W      ?INCHOUR
        JUMP,CC AFT1
        .W      ?INCDAY,?INCYEAR
AFT1:   .W      ?GETCURSOR
        PUSH    HL              ;sauvetage ancien
        LOAD    HL,#0*400+49.;affichage 1ère ligne (ligne Ø)
        .W      ?SETCURSOR  ;voir p. 2.8-1    49e caractère
        .RDX    2
        LOAD    A,# 00001000;format affichage
        .RDX    8.
        LOAD    HL,# TEMPS+6
        .W      ?AFTIME         ;voir p. 2.13-1
        POP     HL
        .W      ?SETCURSOR  ;voir p. 2.8-1
        RET

        .RDX    16.             ;initialisation: 27 novembre à 12h30
TEMPS:  .B      Ø               ;centièmes
        .B      Ø               ;secondes
        .B      30              ;minutes
        .B      12              ;heures
        .B      27              ;jour
        .B      11              ;mois
        .B      78              ;année
        .RDX    8.
```

TEMPS:	centièmes
	secondes
	minutes
	heures
	jour
	mois
TEMPS+6	année

49. 63.

78 11 27 12:30

Remarque: le 1er affichage ne sera effectué que lorsque la première minute sera écoulée.

2 ?ITIMER Initialise le temps du timer (dans HL).
Chaque unité correspond à 20. millisecondes.
Le compteur décompte et se bloque à zéro. Si l'on ne veut plus utiliser le timer, il faut l'initialiser avec un zéro.

3 ?ITIMAD Lorsque le timer arrive la première fois à zéro, la routine dont l'adresse a été donnée dans HL avec ?ITIMAD est exécutée. Cette routine doit être rapide (elle est exécutée pendant les 20 ms entre chaque interruption) et doit être terminée par un RET.

EXEMPLE: on veut attendre et lire des caractères du lecteur papier, avec arrêt du programme de lecture dés qu'il y a un silence de plus de 5 secondes (time out).

```
TEST7:  LOAD    HL,# INTER          ;adresse de la routine d'interruption utilisateu
        .W      ?ITIMAD
        LOAD    A,# 1               ;mise à 1 du flag
        LOAD    FLAG,A
LECT:   LOAD    HL,# 5*1000./20.    ;nombre de fois 20 ms dans 5 sec
        .W      ?ITIMER
        .W      ?IFRPR              ;attente et lecture (voir p. 2-12-1)
        JUMP,CS LEC1
        ...                         ;traitement du caractère
LEC1:   LOAD    A,FLAG              ;si FLAG est toujours à 1, il faut continuer
        OR      A,A                 ;à lire car il n'y a pas eu de time-out
        JUMP,NE LECT

ADFIN:                              ;suite du programme après lecture

INTER:  XOR     A,A                 ;mise à 0 de FLAG
        LOAD    FLAG,A
        RET

FLAG:   .BLKB   1                   ;en RAM
```

2.6 TEST DE LA DIMENSION MEMOIRE

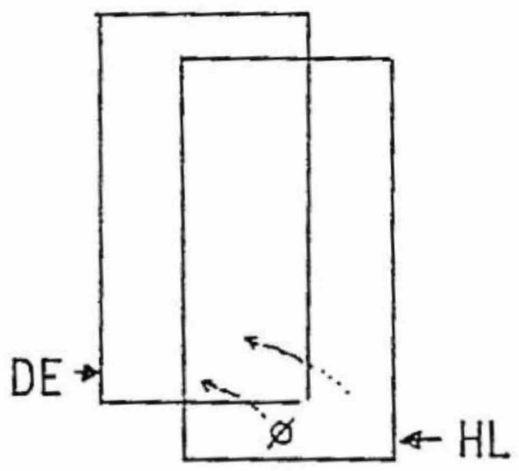
33 ?MEM Test de dimension

Au retour de l'appel, HL contient la dernière position mémoire utilisable.

EXEMPLE D'APPLICATION: Ecriture de Ø dans toute la mémoire à partir de 43 000 (! il n'est pas permis de modifier les positions 42400 à 42777).

Le programme doit se trouver en ROM ou dans l'écran alphanumérique.

```
TEST9:  .W      ?MEM
        LOAD    DE,# 43000
        OR      A,A
        SUBC    HL,DE           ;calcule la longueur
        EX      HL,DE           ;HL=pointeur,  DE=compteur
T92:    LOAD    (HL),# 0
        INC     HL
        DEC     DE
        LOAD    A,D
        OR      A,E
        JUMP,NE T92
        TRAP
```



Une variante plus rapide utilise le LDDR

```
TEST90: .W      ?MEM
        LOAD    (HL),# 0
        PUSH    HL
        LOAD    BC,# 43000+1    ;adresse début corrigée de 1
        OR      A,A
        SUBC    HL,BC
        LOAD    B,H
        LOAD    C,L             ;BC=longueur
        POP     HL
        LOAD    D,H
        LOAD    E,L
        DEC     DE              ;DE= 2e pointeur
        LDDR
        TRAP
```

2.7 BRUITAGES

- **32 ?BUZZ** — Donne un coup de buzzer.
  Pas de paramètre d'entrée, aucun registre modifié.

On peut également obtenir un coup de buzzer en insérant le code 7 dans un texte, ou en écrivant:

```
LOAD   A,# BEL        BEL = 7
.W     ?DICAR
```

- **76 ?BEEP** — Donne un coup de buzzer dont on peut définir la fréquence et la longueur dans A, de la manière suivante:

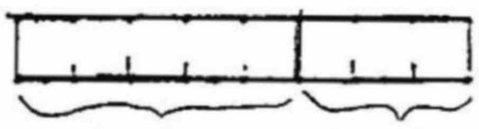
7 3	2 . 0
période	durée

Si A vaut 377, on obtient un silence d'une durée égale à environ 0,4 s.

Le son le plus court et le plus aigu est obtenu si le contenu de A est 11.

- **75 ?PLAY** — Joue un morceau dont le début est pointé par HL.
  Le morceau est terminé par un Ø. A la fin, HL pointe un éventuel morceau suivant.

Format des "notes":



Durée minimum: 1 (256 périodes)
Durée intermédiaires: 2 à 7
Durée maximum: 0 (4000 périodes)

Sons usuels:

	Court		Long (1 sec)
Aigu	103	...	107
	:		:
	:		:
Bas	301		304

EXEMPLE:

```
PLAY:   LOAD    HL,# MOR
        .W      ?PLAY
        RST     60

MOR:    .B      143,163,203,223     ;decrescendo
        .B      203,163,143         ;crescendo
        .B      0                   ;fin du morceau
```

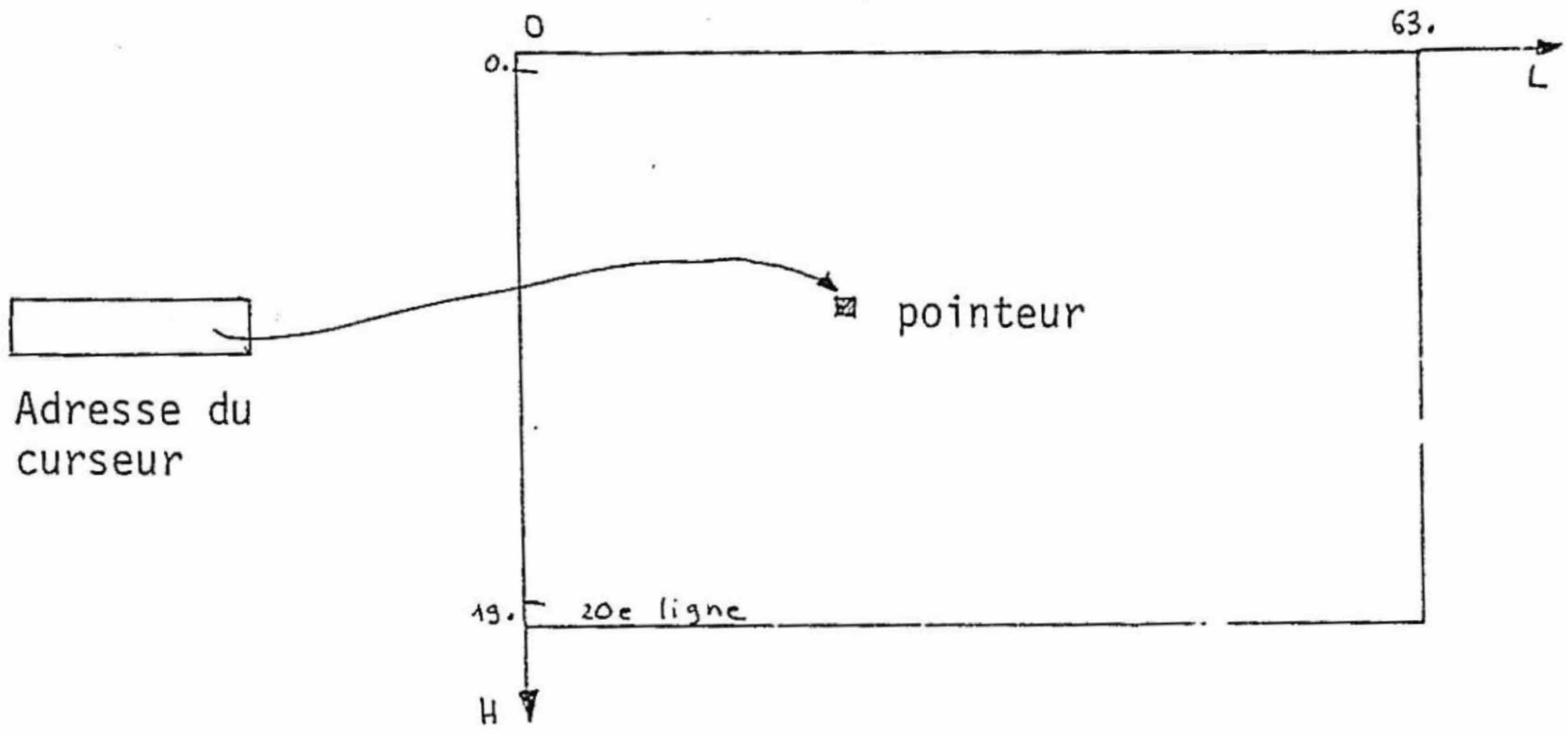
Codage de la gamme (noires):

DO = 274	MI = 235	SOL = 175	SI = 156
RE = 254	FA = 215	LA = 166	DO2= 146

Le manque de place a empêché le décodage d'un format plus riche.
A vous d'écrire ce programme !

2.8 AFFICHAGE SUR L'ECRAN ALPHANUMERIQUE

L'écran du SMAKY6 correspond aux positions mémoires 40 000 à 42 377. Ceci est susceptible de changer (plus grand écran, écran DATAC à une autre adresse), et l'utilisateur ne doit pas utiliser directement des adresses, mais des coordonnées dans l'écran.



Les lignes sont numérotées de 0 à 19. et les colonnes de 0 à 63..

Un curseur définit un point de l'écran, caractérisant l'emplacement où le prochain caractère peut être affiché. Le curseur est une adresse mémorisée quelque part, que l'utilisateur ne connaît pas directement.

- 40 ?SETCURS. Place le curseur aux coordonnées données dans HL.

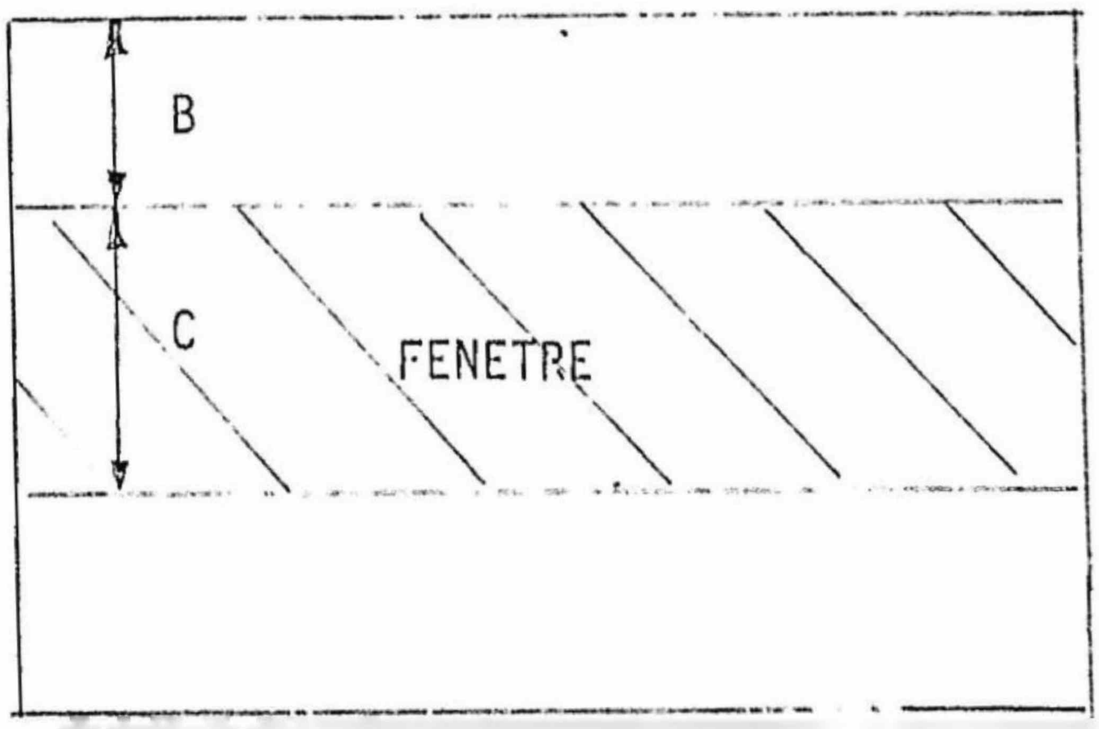
- 41 ?GETCURS. Permet de savoir où est le curseur (rend les coordonnées dans HL) et de connaître le caractère sous le pointeur (dans A).

Ces deux appels ne modifient aucun autre registre. EXEMPLES: p.2.10-1 et 2.10-2

54 ?IPOINTER Permet de choisir le caractère utilisé comme pointeur.

Le pointeur est placé par les routines ?GETCAR et ?GETLINE ?GETLBCE ?INOC ?INHEX ?INDEC ?INMON

Dans l'écran, on peut définir une fenêtre dans laquelle les caractères sont affichés. En arrivant au bas de la fenêtre, tout le contenu de la fenêtre monte d'une ligne, simulant une imprimante.



20 ?IDICAR — Initialise les bornes de la fenêtre.
Il faut préparer avant l'appel dans B le nombre de lignes vides en dessus, dans C le nombre de lignes de la fenêtre.

Au retour de l'appel, le curseur est en haut de la fenêtre; le pointeur n'apparaît pas.

EXEMPLE: Initialiser une fenêtre dans les dernières lignes, effacer cette fenêtre et placer le pointeur au début.

```
LOAD    BC,# 12.*400+8. ;initialise une fenêtre de 8 lignes (à partir de
.W      ?IDICAR                                                la 12e ligne)
LOAD    A,# CLEARW
.W      ?DICAR
```

- 126 ?IDIS — Initialise une fenêtre commençant en haut de l'écran et efface l'écran en mettant le pointeur en haut.
Le nombre de lignes est donné dans C.

Exemple:
```
LOAD    C,# NLI
.W      ?IDIS
```

est équivalent à

```
LOAD    BC,# 0*400+NLI
.W      ?IDICAR, ?CGRA, ?IALPHA
```

- 0 ?DICAR — Affiche le caractère mémorisé dans A.

Les caractères suivants ne déplacent pas le pointeur.

NUL = 0		ignoré
BEL = 7	CTRL G	donne un coup de buzzer
CLEARW = 2	CTRL B	efface l'écran alphanumérique
NORM = 4		mode normal d'écriture
INV = 5	CTRL E	inverse les caractères suivants sur l'écran

Les caractères suivants ont une action spéciale.

BS = 10	Recule le pointeur. Si, en reculant, le pointeur est remonté d'une ligne, et qu'il y a un espace à la fin de la ligne, le pointeur continue à remonter jusqu'au premier caractère non nul ou jusqu'au début de la fenêtre ou de l'écran.
TAB = 11	Avance jusqu'en colonne modulo 8.
CR = 15	Avance jusqu'à la ligne suivante, sans effacer les caractères de fin de ligne.
LF = 12	Efface la fin de la ligne sans déplacer le pointeur

Tous les autres caractères font avancer le pointeur. Dans tous les cas, si le pointeur atteint la fin de la fenêtre ou de l'écran; il y a décalage vers le haut et le pointeur est placé au début de la dernière ligne.
Avant l'affichage sur l'écran, la position mémoire OUTCAR=42520 est testée Si elle contient Ø (comme après une initialisation du système), ce qui a été dit ci-dessus est vrai. Si OUTCAR≠0, aucun affichage ou test n'a lieu et la routine dont l'adresse est dans OUTCAR est exécutée.
Il n'y a pas d'appel système pour modifier OUTCAR.

EXEMPLES D'APPLICATION: voir pages 2.3-3 et 2.10-2

- 127 ?DELAY — Attente de 1 ms à 65 secondes.

  La valeur de l'attente suit l'appel (unité de 1 ms). Précision: 10%.

  EXEMPLE: Métronome à la seconde.

```
LOOP:   .W      ?DELAY,1000.
        LOAD    A,# 102
        .W      ?BEEP
        JUMP    LOOP
```

- 133 ?HLDELAY — Idem avec la valeur dans HL.

- 42 ?SPACE — Affiche un espace

- 43 ?RETURN — Exécute un retour de chariot

```
PUSH    AF                      PUSH    AF
LOAD    A,# ␣          et       LOAD    A,# CR
.W      ?DICAR                  .W      ?DICAR
POP     AF                      POP     AF
```

- 122 ?CLEAR — Efface la fenêtre et met le pointeur en haut.

  132 ?EFECR — Efface la fin de l'écran (retour avec A=Ø)

- 121 ?TAB — Exécute un tabulateur

  74 ?EFLI — Efface la ligne après le pointeur, sans déplacer le pointeur. (exécute un LF)
  Retour avec A = 0.

  130 ?BACKDEL — Efface le caractère précédent (exécute un DELETE)

  137 ?BACKSP — Recule le pointeur

- 6 ?DITEX — Affiche un texte

  L'adresse de début du texte est donnée dans HL avant l'appel.
  Après l'appel, HL pointe l'éventuel texte suivant et aucun autre registre n'est modifié.

  Le texte donné en mémoire est terminé par un zéro.

```
EXEMPLE:   affichage en début d'exécution d'un programme
        .TITLE  TRUC.SR
        REV     = '2            ;Numéro de la révision  (0 à 9)
        ;...
        LOAD    HL,# TXREV
        .W      ?DITEX
        ;...
TXREV:  .ASCIZ "PROGRAMME TRUC, REVISION <REV> <CR>"
```

- 123 ?TEXT — Affiche un texte

Comme ?DITEX, mais l'adresse du texte suit l'appel.
Le registre HL n'est pas modifié.

On peut écrire plus simplement:

```
            ...
        .W  ?DITEX, TXREV
            ...
```

36 ?DITEB — Affiche un texte terminé par le caractère dans B.

Comme DITEXZ, mais le caractère de fin n'est pas nécessairement Ø.
Pour un texte, on utilise un .ASCII "texte<CARFIN>" avec LOAD B,# CARFIN

Le terminateur n'est pas affiché

136 ?TEXTIM — Affiche le texte défini par un .ASCIZ et qui suit immédiatement l'appel.

Par exemple:

```
...
.W      ?TEXTIM
.ASCIZ  "Programme truc, révision <REV><CR>"
...
```

2.9 LECTURE DU CLAVIER

Le clavier balayé du SMAKY6 est difficile à gérer directement.
Une routine du système s'en occupe, en plaçant à chaque interruption les caractères des touches pressées dans une queue (FIFO). La lecture du clavier est en fait une lecture du FIFO.

1 ?GETCAR Attend un caractère du clavier et le lit.

Le retour de cet appel ne se fait que lorsqu'une touche au moins est pressée (ou a été pressée depuis la dernière fois où l'appel a été fait). Le FIFO du clavier permet d'avoir au maximum 32 anciennes touches du clavier en attente d'être lues. Le caractère lu est dans A.

Attention: les touches fonction ne sont pas stockées dans le FIFO dans la révision 1.

EXEMPLE: Affichage sur l'écran des touches tapées:

```
ECHO:   .W      ?GETCAR
        .W      ?DICAR
        JUMP    ECHO
```

Comme pour ?DICAR, un débranchement est possible à une routine dont l'adresse est donnée dans INCAR = 42522

15 ?IFCAR Teste si le FIFO est vide et lit un caractère s'il n'est pas vide.

Il n'y a pas de paramètre d'entrée. Le retour est immédiat (200 µs), avec le CS si aucune touche n'est en attente.

Si le carry est à zéro (CC), le code de la touche est dans A et le FIFO est vidé d'un caractère.

EXEMPLE: on veut faire un jeu utilisant les 4 touches ← ↑ ↓ → pour déplacer un mobile.

```
        ;.,...
        .W      ?IFCAR
        JUMP,CC FOUND
        ;...
FOUND:  COMP    A,# 'R    ↑
        JUMP,EQ MOVEUP
        COMP    A,# 'C    ↓
        JUMP,EQ MOVEDOWN
        COMP    A,# 'D    ←
        JUMP,EQ MOVELEFT
        COMP    A,# 'F    →
        JUMP,EQ MOVERIGHT
        ;c'est une autre touche
        JUMP    FOUND
```

Voici une autre manière d'écrire la routine FOUND, plus élégante, mais plus lente (à cause de l'appel système).

```
FOUND:  LOAD    DE,# TADEC
        .W      JUMPCAR          ;voir p.2.11-1
        ;touches non prévues: ignorer ou signaler l'erreur
        ;...
TADEC:  .BW     'R,MOVEUP
        .BW     'C,MOVEDOWN
        .BW     'D,MOVELEFT
        .BW     'F,MOVERIGHT
        ;autres touches éventuellement
        .B'     Ø
```

16 ?GETFON Lit l'état des touches fonctions au moment de l'appel.

Cet appel teste directement le clavier. Le code des touches fonction pressées est dans A. A=0 et EQ si aucune touche n'est pressée.

EXEMPLE: on veut effacer tout l'écran graphique lorsque la touche de gauche (KILL) est pressée. Dans la boucle du programme, il faut placer les instructions suivantes.

```
        KILL  = 100
        ;...
        PUSH    AF
        .W      ?GETFON
        COMP    A,# KILL
        JUMP.,NE XX
        .W      ?CGRA
XX:     POP     AF
        ;...
```

5 ?GETLINE Attend une ligne terminée par un CR.

Il n'y a pas de paramètre d'entrée. Pendant l'appel, les caractères tapés sont affichés et mémorisés dans une zone mémoire.

Au premier CR tapé, le retour se fait avec HL pointant au début de la zone. Cette zone est terminée par un code = 0 qui remplace le CR. Si plus de 62 caractères sont tapés, le retour est forcé, et le carry est à 1 (CS).

EXEMPLE: le programme attend un nom, et compte le nombre de ses lettres.

```
TRI:    .W      ?GETLINE
        PUSH    HL              ;sauvetage adresse début nom tapé
        LOAD    B,# -1
T2:     INC     B               ;B compte le nombre de lettres
        LOAD    A,(HL)
        INC     HL
        OR      A,A             ;dernier caractère ?
        JUMP,NE T2
```

```
T4:     INC     A               ;conversion en BCD
        DA      A
        DECJ,NE B,T4            ;A .contient le nombre en BCD
        LOAD    HL,# TXDEB
        .W      ?DITEX
        POP     HL              ;adresse nom tapé
        .W      ?DITEX
        LOAD    HL,# TXMIL
        .W      ?DITEX
        .W      ?AFHEX          ;affichage BCD de A
        .W      ?DITEX          ;HL pointe déjà le début du texte
        JUMP    TRI
TXDEB:  .ASCIZ  "LE MOT "
TXMIL:  .ASCIZ  " CONTIENT"
TXFIN:  .ASCIZ  " LETTRES <CR>"
```

35 ?GELBCDE — Attend une ligne terminée par une touche située dans les intervalles 0 à B (non compris), C(compris) à D (non compris), E (compris) à 377 (B,C,D,E sont les registres du Z80).
Identique à ?GETLINE pour tout le reste.

EXEMPLE: voir appels INOC, INDEC, INHEX dans le listing.

2.10 LECTURE ET AFFICHAGE DE NOMBRES

- 70 ?AFBIN Affiche le contenu de A en binaire.
  Aucun registre n'est modifié.

EXEMPLE: Afficher de façon répétée le contenu du périphérique 0 (clavier) en binaire au centre de l'écran.

```
TEST:   .W      ?CALPHA
TES2:   LOAD    HL,# 10.*400+28.        ;centre
        .W      ?SETCURSOR
        LOAD    A,$CLA
        .W      ?AFBIN
        JUMP    TES2
```

- 71 ?AFHEX Affiche A en BCD ou hexa.

FXFMPLE: affichage de HL en hexa ou BCD.

```
        LOAD    A,H
        .W      ?AFHEX
        LOAD    A,L
        .W      ?AFHEX
        ...
```

- 72 ?AFOCA Affiche le carry et A en octal (9 bits)

- 73 ?AFOHL Affiche HL en octal (16 bits).

EXEMPLE: affichage de HLA (24 bits) en octal.

```
        SRC     H
        RRC     L
        .W      ?AFOHL
        .W      ?AFOCA
        ...
```

(ce programme affiche toujours un zéro inutile au début).

EXEMPLE2: affichage de HL avec suppression des zéros non significatifs.

```
        EX      HL,DE           ;sauvetage dans DE
        .W      ?GETCURSOR
        PUSH    HL              ;sauvetage du pointeur
        EX      HL,DE
        .W      ?AFOHL          ;affichage avec des zéros
        POP     HL
        .W      ?SETCURSOR      ;pointeur au début
        LOAD    B,# 5           ;seules les 5 premières positions sont
                                ;effacées jusqu'à ce qu'il y ait un
                                ;chiffre différent de zéro
LOOP:   .W      ?GETCURSOR      ;caractère sous le pointeur
        COMP    A,# '0
        JUMP,NE LOP2
        .W      ?SPACE
        DECJ,NE B,LOOP

LOP2:   INC     B
LOP4:   INC     HL              ;pointeur remis à la fin
        DECJ,NE B,LOP4
        .W      ?SETCURSOR
```

EXEMPLE3: calculatrice octale additionnant deux nombres avec signalement des dépassements de capacité.

```
EXN2:   LOAD    HL,# 5*400+0    ;6e ligne
        .W      ?SETCURSOR
        .W      ?INOC
        PUSH    HL
        LOAD    A,# '+
        .W      ?DICAR
        .W      ?INOC
        LOAD    A,# '=
        .W      ?DICAR
        POP     DE
        ADD     HL,DE
        JUMP,CS EXN24

        .W      ?AFO HL
        .W      ?RETURN
        JUMP    EXN2

EXN24:  LOAD    HL,≠TXERR
        .W      ?DITEX
        JUMP    EXN2

TXERR:  .ASCIZ  "␣dépassement de capacité <CR>"
```

131 ?AFXHL — Affiche HL en BCD ou en hexa.

HL = ␣1 0 1 0 0 1 0 0 0 1 1 1 1 1 1 1␣ il affichera A47F
A 4 7 F

101 ?AFMONA — Affiche A en octal ou hexa selon l'état d'un flag interne.

102 ?AFMOHL — Affiche HL en octal ou hexa selon le flag.

120 ?IMODE — Initialise le flag octal/hexa.

65 ?INDEC — Attend un nombre BCD

Le retour de l'appel se fait au premier caractère non BCD, avec dans A ce caractère, dans HL le nombre tapé (4 derniers digits seulement), dans C le nombre de digits tapés. On peut en cours d'introduction effacer les derniers chiffres introduits.

EXEMPLE: lecture de l'heure et transfert dans la position mémoire TIM2 (voir page 2.13-1).
12h24 doit être tapé 1224␣. 7h25 peut être tapé 725␣ ou 0725␣.
15h doit être tapé 1500␣. (␣ = espace)

```
GETH:   .W      ?INDEC
        LOAD    TIM2,HL
        ;suite...
```

- 66 ?INHEX — Attend un nombre hexa

- 67 ?INOC — Attend un nombre octal

100 ?INMON — Attend un nombre octal ou hexa selon le flag interne (moniteur)

120 ?IMODE — Initialise le mode écran (NORM, INV)
(porte le même numéro d'appel que l'initialisation du flag octal/hexa)

2.11 INITIALISATION D'ECRAN ET CONVERSION DE POINTEURS

30 ?CALPHA Effacement de l'écran alphanumérique.

Il n'y a pas de paramètre d'entrée. Au retour, le registre A vaut Ø, le carry est à 1 (CS), et le registre HL pointe la 1ère position de l'écran (=SALPHA). Cet effacement ne déplace pas le pointeur du texte.

31 ?CGRA Effacement de l'écran graphique.

Pas de paramètre d'entrée. Le registre A est mis à zéro, la carry à 0 (CC). Les autres registres ne sont pas modifiés.

21 ?IALPHA Mode alphanumérique uniquement.

Pas de paramètres et pas de registres modifiés.

22 ?IGRA Mode graphique uniquement.

23 ?IAGRA Mode alphanumérique et graphique superposés.

Selon la valeur du carry, le mode graphique a des petits points (CS) ou des gros points (CC).

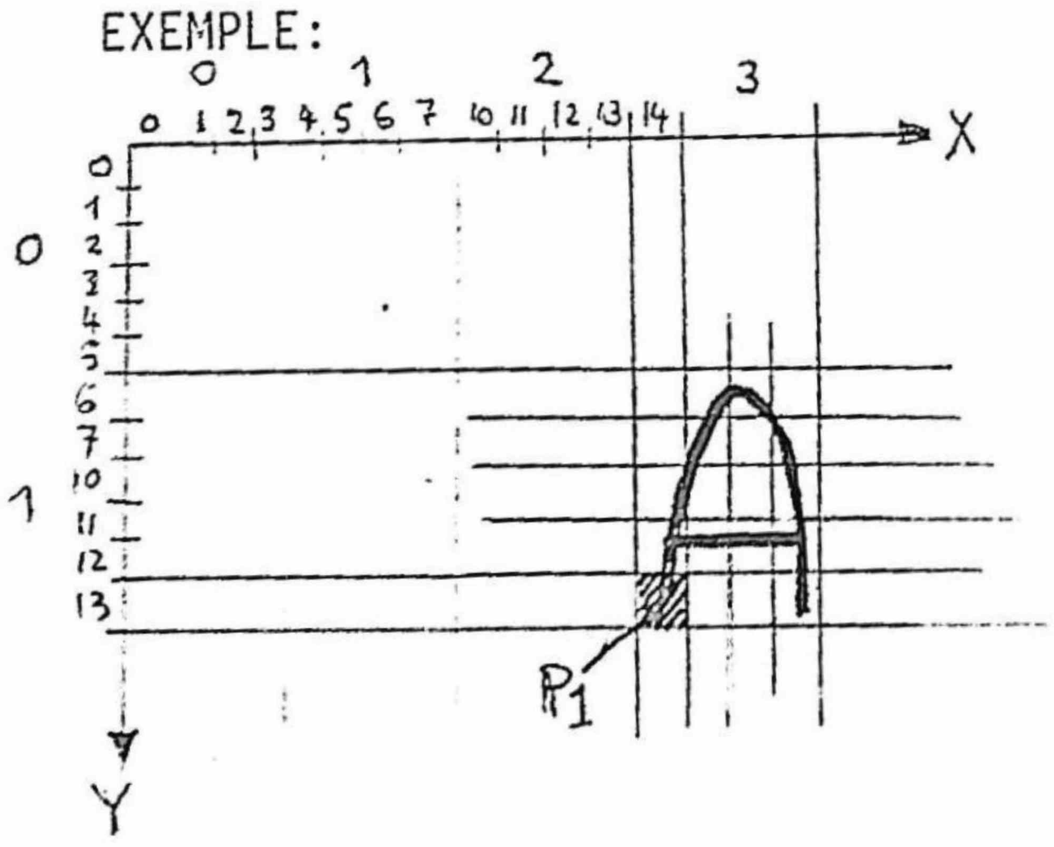
EXEMPLE: effacement des deux écrans et initialisation en mode superposé.

```
.W      ?CALPHA                     .W      ?CGRA
.W      ?CGRA                       .W      ?CALPHA
.W      ?IAGRA    ;gros points      .W      ?IAGRA     ;petits points
```

64 ?ALGRA Trouve la coordonnée dans l'écran graphique (donnée dans DE) correspondant à la coordonnée d'un caractère dans l'écran alphanumérique (donnée dans HL).

Le point graphique est situé en bas à gauche du caractère.

EXEMPLE:




```
LOAD    HL,# 1*400+3
.W      ?ALGRA
```

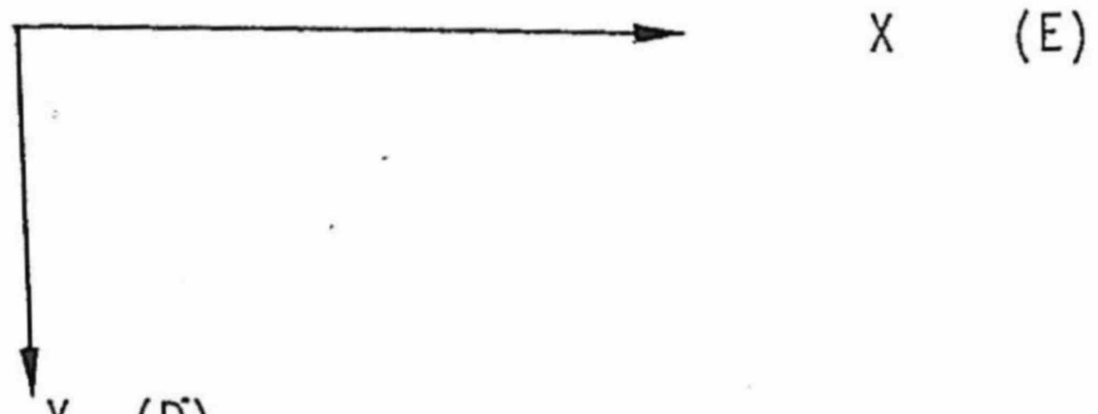
On trouve dans DE 5414, soit 14 en X et 13 en Y, qui sont les coordonnées graphiques du point P1.

77 ?GRAAL Trouve la coordonnée d'un caractère sur l'écran alphanumérique (donnée dans HL) correspondant à un point graphique (donné dans DE).

Il y a donc 6*4 points graphiques qui correspondent au même caractère alphanumérique.

ROUTINES GRAPHIQUES

Toutes les routines graphiques travaillent sur la base d'un système de coordonnées en (x;y)



x: 0 à 255. (377 octal) registre E
y: 0 à 119. (167 octal) registre D

Pour être exécutées plus rapidement, les routines graphiques sont atteintes par des CALL.

Pour charger une coordonnée dans la paire DE, il y a avantage à définir XX=1, YY=400 et à écrire

LOAD DE,# x*XX + y*YY
(x : coord. en X ; y : coord. en Y)

TRAITEMENT D'UN POINT

Quatre routines sont prévues pour traiter directement un seul point. Avant l'appel d'une de ces routines, il faut que E contienne la coordonnée en X et D celle en Y.

3000 SETP	met un point
3003 CLRP	effece un point
3006 INVP	inverse le point
3011 TESTP	regarde s'il y a un point.

Retour avec: Z=1 (EQ) s'il n'y en a pas
Z=Ø (NE) s'il y en a un

TRAITEMENT D'UN SEGMENT

Deux routines permettent de relier deux points par un segment de droite
Avant d'appeler ces routines il faut dire comment doit être le trait.

3014 CMOD — A doit contenir le numéro du trait

1: trait mince continu
2: trait gros continu
3: pointillés
4: tirets
5: trait point
6 plus

Par ailleurs:
Ø: efface le trait
1Ø: saute sans rien modifier.

3017 SEGA — On donne les coordonnées des deux points en (X,Y):

Premier point: x=E y=D
Deuxième pt: x=L y=H.

SEGA relie ces deux points par un segment selon le mode choisi.
Sortie: E=X D=Y du point d'arrivée

3022 SEGRE — On donne la coordonnée du premier point en (X,Y). Les coordonnées du deuxième sont données relativement au premier. Si ces coordonnées sont négatives, elles doivent avoir le bit 27 à 1.

Premier point: x=E y=D
Deuxième pt: Δx=L Δy=H.

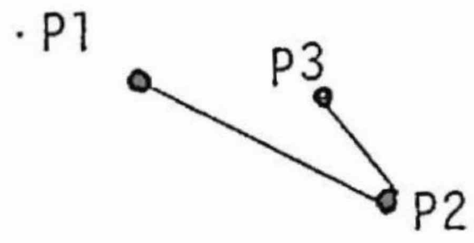
SEGRE relie ces deux points par une droite selon le mode choisi.

EXEMPLE 1:

Dessinons une diagonale en gros trait à travers tout l'écran.

```
        TGROS= 2
        ;initialisation en mode graphique
        .W      ?CGRA, ?IAGRA
        LOAD    A,# TGROS
        CALL    CMOD:
        LOAD    DE,# 0*XX+0*YY          ;P1
        LOAD    HL,# 255.*XX+119.*YY    ;P2
        CALL    SEGA
        ;suite du programme ou TRAP
```

Maintenant DE contient les coordonnées du point d'arrivée, ce qui permet de charger dans HL les coordonnées d'un point P3 et, en appelant SEGA, d'obtenir:



Idem pour la routine SEGRE. Il suffit de donner Δx et Δy pour P3 dans HL.

TRAITEMENT DES MOTIFS

Deux routines permettent de dessiner un motif dont on a donné les coordonnées en absolu ou en relatif dans une table. L'adresse de la table est dans IX et la table se termine par un .BYTE 377 (ENDC=377).

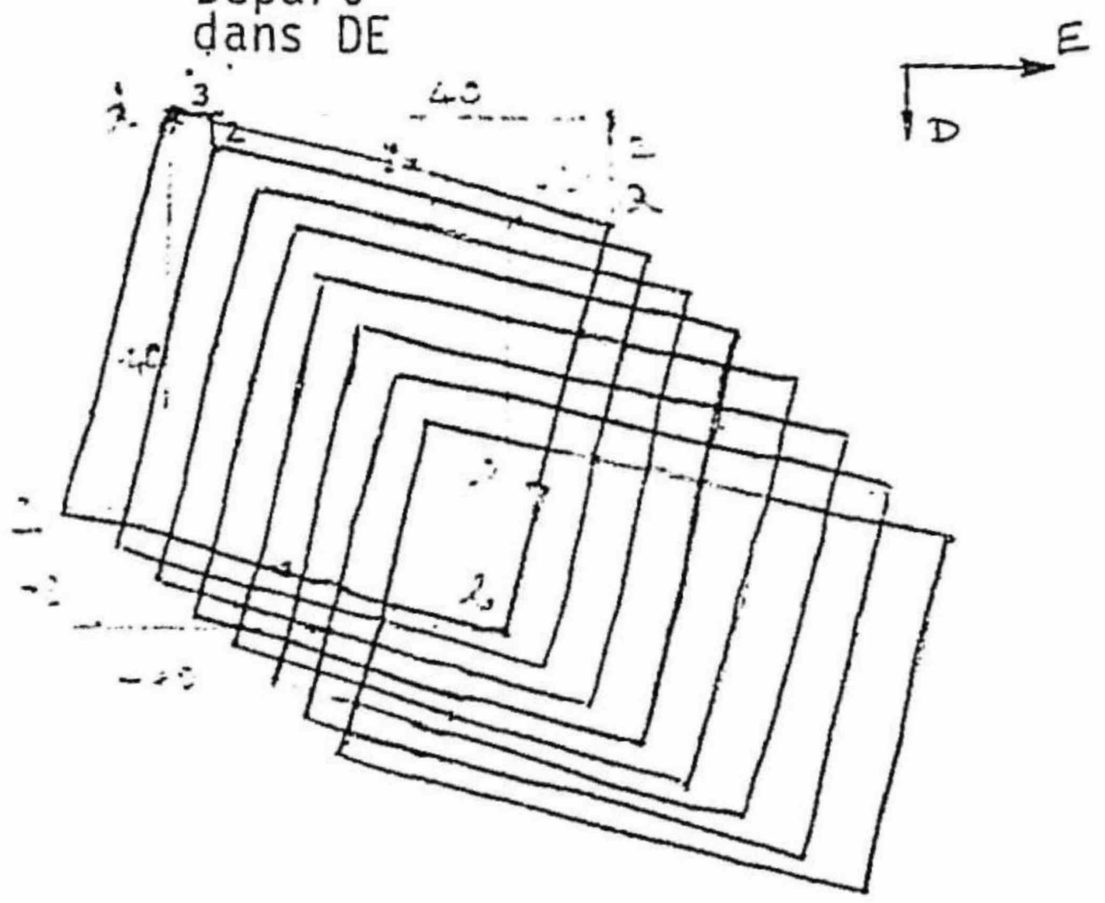
Les bytes sont donnés dans l'ordre y,x.

3025CONTA — Trace le contour avec les points donnés en absolu. Le motif ainsi défini sera donc toujours dessiné à la même place sur l'écran.

3030 CONTRE Trace le contour avec les points donnés en relatif. Il suffit de donner en absolu les coordonnées du premier point, tous les autres seront relatifs par rapport au précédent.
En modifiant les coordonnées du premier point, il est alors possible de déplacer le motif sur l'écran.
(IX contient l'adresse de la table, DE les coordonnées du premier point)

EXEMPLE 2:

On veut dessiner le motif suivant:




```
                .TITLE  TESTGRA.NO
                ;Dessin de carrés décalés
                .PROC Z80
 3106           CONTRE= 3030
    1           TMINCE = 1
  376           CMODE = 376
  377           ENDC =  377                 .REF SM6
14747           ?CGRA = 347+400*31
11747           ?IAGRA =347+400*23
14347           ?CALPHA=347+400*30

053000                          .LOC    53000

053000 303 016 126              JUMP    START

053003 376 001          TABLE:  .B      CMODE, TMINCE
053005 002 040                  .B      2, 40
053007 030 202                  .B      40, 202
053011 202 240                  .B      202, 240
053013 230 002                  .B      240, 2
053015 377                      .B      ENDC

053016 347 023 347 031  START:  .W      ?IAGRA,?CGRA,?CALPHA
053022 347 030
053024 021 024 060      DESSIN: LOAD    DE,#24+60*400
053027 006 010                  LOAD    B,#10
                        LOOP:
053031 335 041 003 126          LOAD    IX,#TABLE
053035 315 030 006              CALL    CONTRE
053040 034                      INC     E
053041 034                      INC     E
053042 034                      INC     E
053043 024                      INC     D
053044 024                      INC     D
053045 020 362                  DECJ,NE B,LOOP
053047 367
                53016           .END    START
```

Remarque: pour exprimer un nombre négatif,
mettre le bit de poids fort à 1,
c'est-à-dire: ajouter 200 à la valeur.

ECRAN GRAPHIQUE

(E) x

(D) y

0 10 20 30 40 50 60 70 100 110 120 130 140 150 160 170 200 210 220 230 240 250 260 270 300 310 320 330 340 350 360 370 400

0 10 20 30 40 50 60 70 100 110 120 130 140 150 160 170

2.12 TRANSFERTS AVEC LES PERIPHERIQUES

50 ?RPAR Attente et lecture de l'interface parallèle.

Le caractère lu est dans A.

EXEMPLE: Lecture d'un clavier supplémentaire et affichage sur l'écran, supposé initialisé.

```
ECHO:   .W      ?RPAR
        .W      ?DICAR
        JUMP    ECHO
```

51 ?IFRPAR Test et lecture éventuelle de l'interface parallèle.

On revient avec CS et A modifié si aucun caractère n'est en attente (FULL = 0). On a CC et le caractère dans A sinon.

EXEMPLE: on veut lire l'interface parallèle jusqu'à ce qu'une touche soit pressée sur le clavier.

```
TEST:   .W      ?IFKEY
        JUMP,CC EXIT
        .W      ?IFRPAR
        JUMP,CS TEST
        ;s'occuper du caractère lu
        JUMP    TEST
EXIT:   ...
```

52 ?WPAR Attend si nécessaire et écrit sur l'interface parallèle.

Le caractère à écrire est dans A; le transfert se fait dès que l'interface est prêt (READY).

EXEMPLE: transfert de ce qui entre de l'interface parallèle sur la sortie // avec comptage du nombre de bytes qui transitent dans HL.

```
TRANSIT:
        LOAD    HL,#Ø
TR21:   .W      ?RPAR
        .W      ?WPAR
        INC     HL
        JUMP    TR21
```

53 ?IFWPAR Test et écriture éventuelle sur l'interface parallèle.

On revient avec CS et A non modifié si l'interface n'est pas prêt (READY = 0). On a CC si le caractère a été écrit.

24	?RPR
44	?IFRPR
26	?WPP
46	?IFWPP

Lecture
Test et lecture év. } Entrée bande papier

Ecriture } Sortie perforateur papier
Test
et écriture éventuelle

USART 4
(côté touche
CR du clavier)

25	?RMOD
45	?IFRMOD
27	?WMOD
47	?IFWMOD

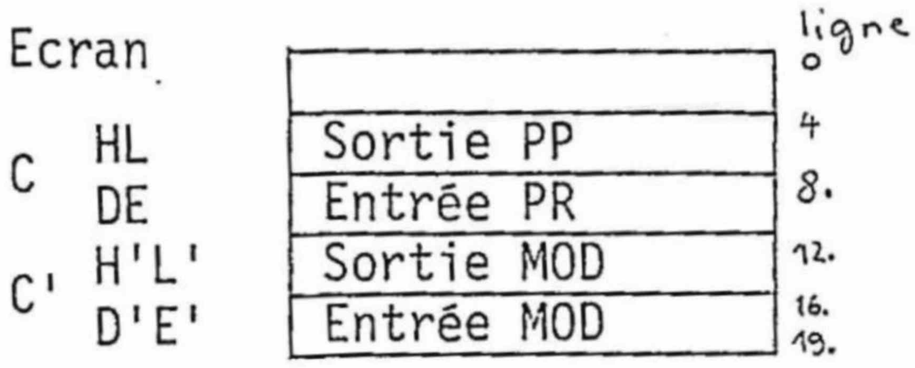
Lecture
Test et lecture év. } Entrée Modem

Ecriture } Sortie Modem
Test
et écriture éventuelle

USART 6
(côté touche
ESC du clavier)

Ces appels sont pendants des appels vus à la page précédente.

EXEMPLE: Programme de test avec par exemple un câble reliant les deux prises. Le programme cherche à écrire dans les deux sorties en affichant ce qui sort.
Tout ce qui rentre est également affiché.




```
        LOAD    HL,# SALPHA+400
        LOAD    DE,# SALPHA+1000
        LOAD    C,# 0
        EX      BL
        LOAD    HL,# SALPHA+1400
        LOAD    DE,# SALPHA+2000
        LOAD    C,# 0
        EX      BL

TØ:     LOAD    A,C
        .W      ?IFWPP
        JUMP,CS T2
        LOAD    (HL),C
        INC     C
        INC     L
T2:     .W      ?IFRPR
        JUMP.,CS T4
        LOAD    (DE),A
        INC     E
        EX      BL
T4:     LOAD    A,C
        .W      ?IFWMOD
        JUMP,CS T6
        LOAD    (HL),C
        INC     C
        INC     L
T6:     .W      ?IFRMOD
        JUMP.,CS T8
        LOAD    (DE),A
        INC     E
        EX      BL
T8:     JUMP    TØ
```

103 ?LOADBI Lecture de bande papier format PDP11

Si le registre C = Ø au départ, le chargement est absolu.
Si C=1 au départ, l'adresse dans HL définit l'adresse de chargement du premier bloc, et les blocs suivants sont chargés avec une position relative respectée par rapport au premier bloc. Le start automatique n'est pas effectué dans ce cas.

104 ?PUNBI Perforation format PDP11

On ne peut puncher qu'un bloc contigu, sans amorce de fin.
HL: adresse de début du bloc
DE: longueur du bloc (0 = bloc nul).
BC: adresse de restart
IY: offset
L'adresse de chaque bloc est corrigée en soustrayant IY.

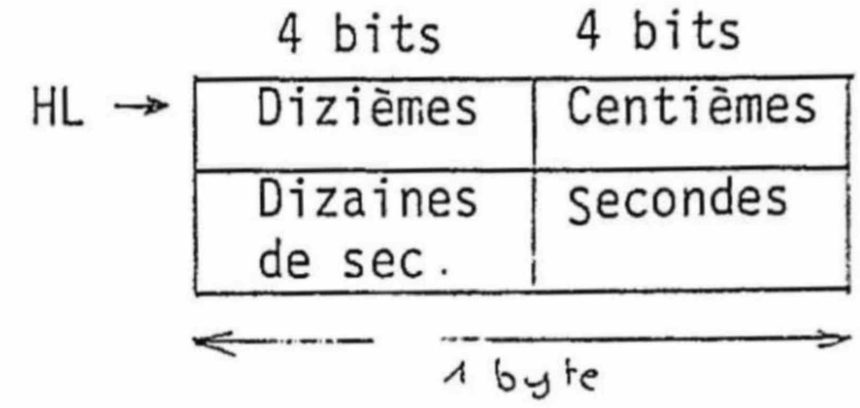
EXEMPLE: L'assembleur (EASY, SMILE) place le binaire en 53 000.
S'il y a par exemple un .LOC 40400 au début, IY =
= 53000 - 40400 = 12 400. La bande perforée par l'ordre T
a la bonne adresse.

105 ?AMORCE Perfore une amorce (code 0 si papier)

2.13 MESURE DU TEMPS

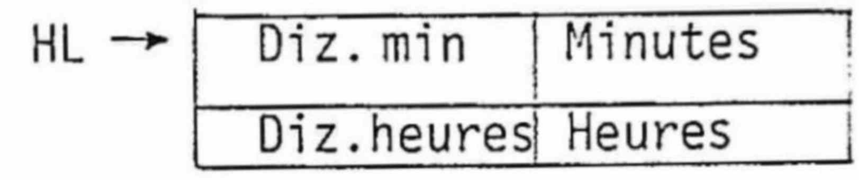
111 ?INCSEC — Ajoute 2 centièmes de seconde.

Le registre HL pointe le mot mémoire de 4 digits. Au retour, HL pointe le mot suivant et le carry est à 1 si on a atteint 60. = 0 secondes.



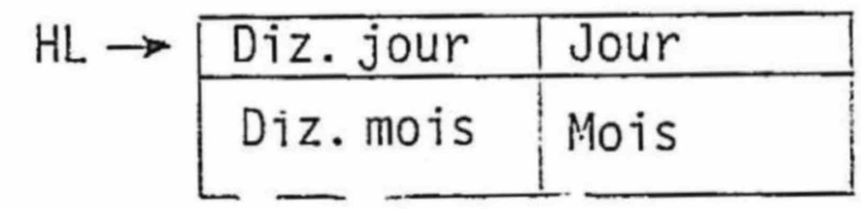
112 ?INCHOUR — Ajoute une minute si le carry est à 1.

Au retour HL pointe le mot suivant et le carry est à 1 si on a atteint 24h = 0 h.



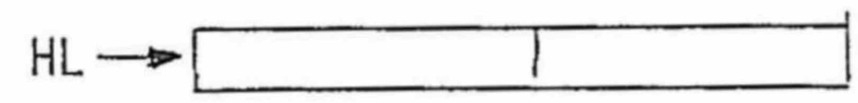
113 ?INCDAY — Ajoute un jour si le Carry est à 1.

Les jours sont comptés de 1 à 28, 30 ou 31 selon le mois. Pour une année bissextile, il faudra corriger le 1er mars en 29 février à la transition du 28.
Les mois sont comptés de 1 à 12..



114 ?INCYEAR — Ajoute le carry au mot de 8 bits pointé par HL (compteur par 100).

Au retour, le carry est à 1 s'il y a dépassement, et HL pointe le mot suivant.

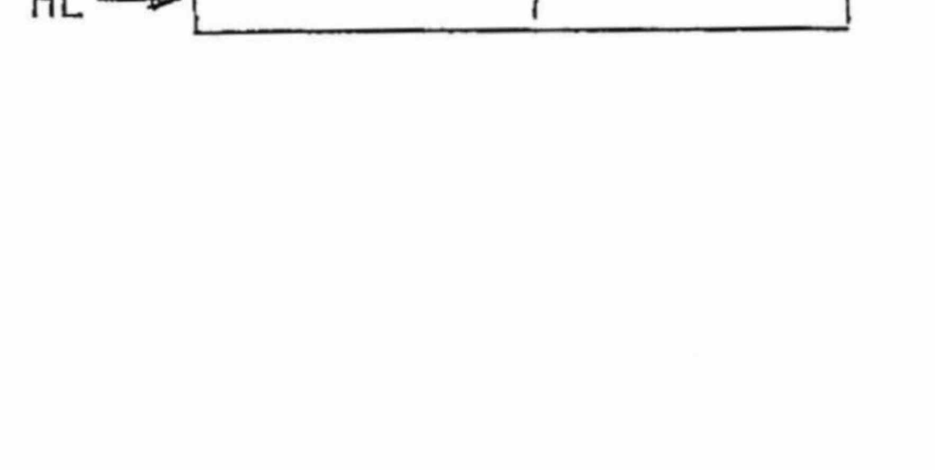


Cette routine sert à incrémenter les années ou les siècles. Elle est utilisable pour faire un compteur BCD de précision quelconque.

115 ?AFTIME — Affiche l'heure selon un format variable

HL pointe les poids forts de l'heure, A contient le format.
En commençant par le début, le nombre de bits à 0 donne le nombre de paires de chiffres affichés devant le ":". Le nombre de 1 donne le nombre de chiffres affichés derrière le ":". Il ne doit y avoir que des 0 derrière.

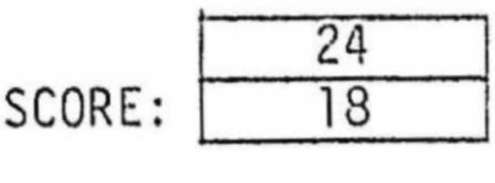
EXEMPLE: affichage de la zone encadrée




```
LOAD   HL , # TIM3
.RDX   2
LOAD   A,# 00110000 (affichage JH:MS)
.RDX   8.
.W     ?AFTIME
```

AUTRE EXEMPLE: affichage du score dans un jeu

```
LOAD   HL,# SCORE
.RDX   2
LOAD   A,# 01000000
.RDX   8
.W     AFTIME        ;affiche 18:24
```

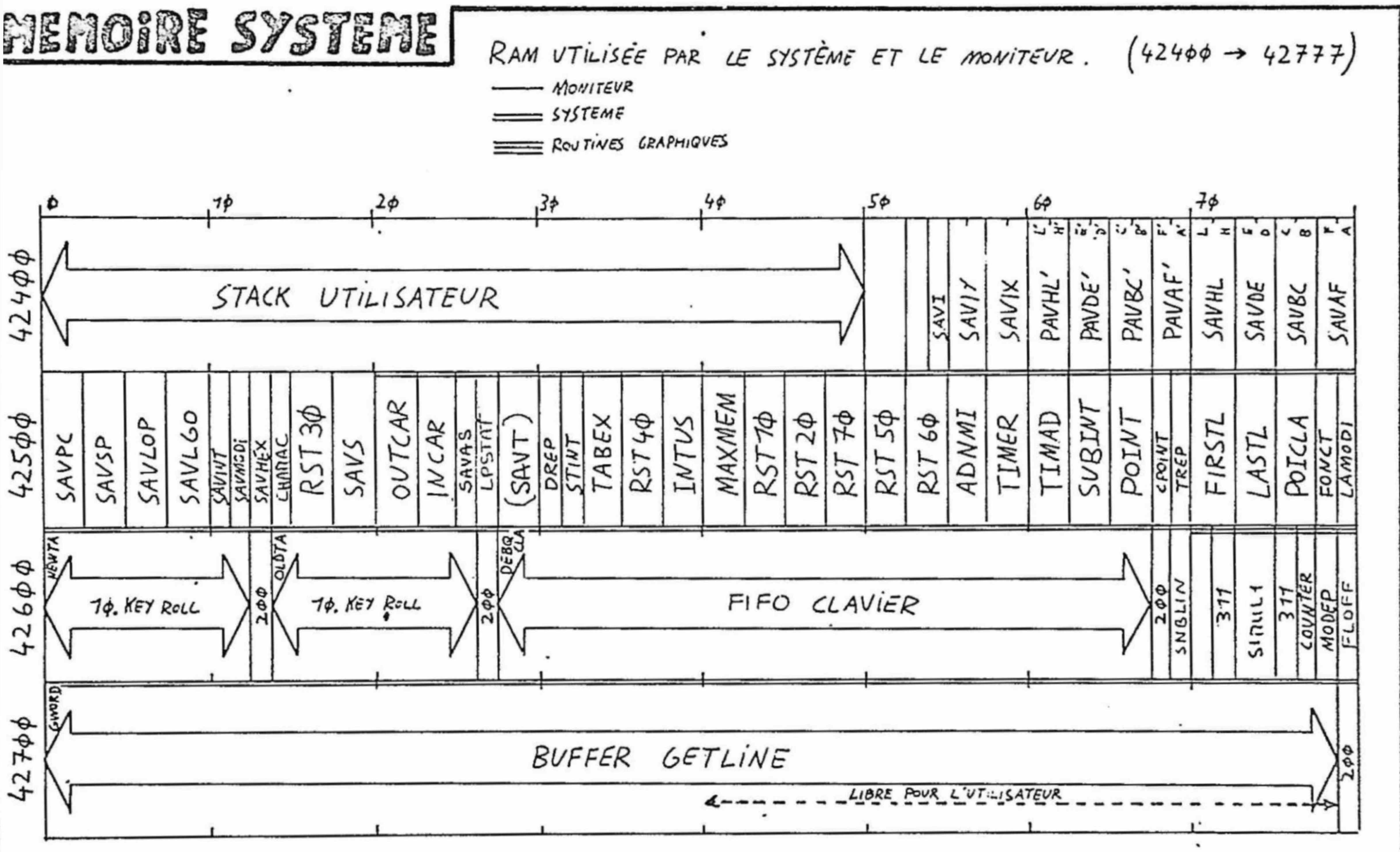


2.14 APPELS POUR EXECUTION EN PAS A PAS

106 ?SAVESTAT — Sauve l'état de tous les registres dans la zone réservée du moniteur.

La valeur du pointeur de pile mémorisé est de deux supérieure à la valeur avant l'appel.

Cette routine est prévue pour mémoriser la trace des instructions dans un programme de simulation ou de test. Il faut naturellement chaque fois transférer la zone des registres (25. positions) dans une autre zone mémoire.



107 ?PRSTAT — Affichage de la zone registre sur l'écran, selon la disposition ci-dessous:

```
F': 42476   A': 42477   B': 42475   C': 42474   D': 42473   E': 42472   DE':42473   HL':42471   SP':42502   PC':42500
                                                                        DE':42472   HL':42470   SP':42503   PC':42501
11001010 123 000 321 000 101 000101 123456  000000 111111 222222
SZ-H-VNC  A   B   C   D   E    DE     HL    IX  SP IY  (SP) I PC
10000001 310 110 207 124 312 124312 001473 100102 012321 123 ION
F': 42466   A': 42467   B': 42465   C': 42464   D': 42463   E': 42462   D'E':42463   H'L':42461   IX: 42457   IY: 42455   I: 42453
                                                                        D'E':42462   H'L':42460   IX: 42456   IY: 42454
```

110 ?EXECUTE — Continue l'exécution selon l'état sauvé par SAVSTAT et affichable par PRSTAT.

2.15 POINTS D'ENTREE DANS LE SYSTEME ET DANS LE MONITEUR

Les programmes utilisateur qui doivent revenir au système et moniteur peuvent utiliser les points d'entrées suivants, que l'on accède par un saut (un appel est aussi accepté, car la pile est réinitialisée).

SYS = 0 Retour au système comme après un RESET

134 ?REMON Retour au moniteur sans initialisation

135 ?TRAPPE Retour au moniteur avec initialisation et affichage de l'état

- REMON = 4003 Retour au moniteur sans initialisation
- TRAPPE = 4006 Retour au moniteur avec init. et affichage de l'état

La touche S saute à l'adresse mémorisée en SAVS = 42516 (retour à SMILE).

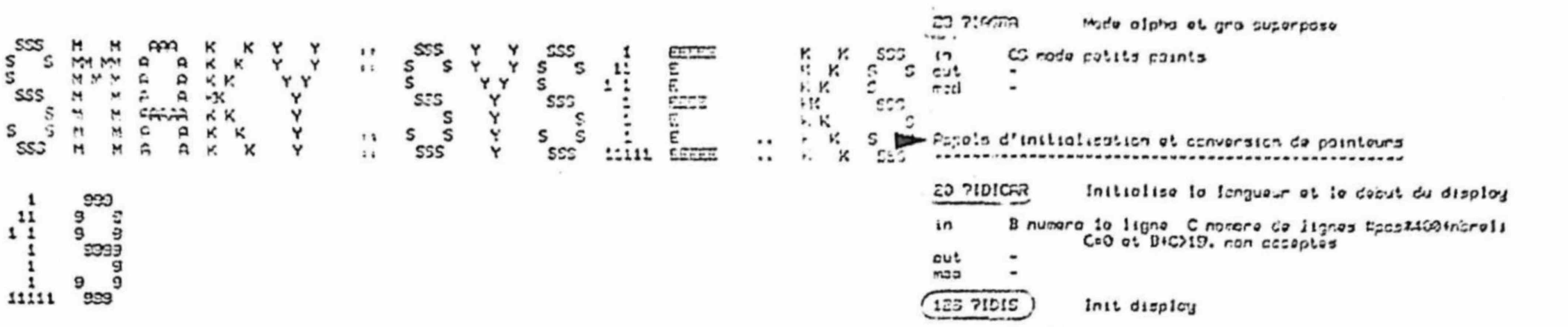
Aucun contrôle n'est fait pour ce saut, qui est initialisé comme un retour au système par SYS et comme une trappe par MON.

Les noms réservés par des programmes de base sont:

TEST	programme de test
PROM, SPROM	programmateur de ?708, 2716, 2732
TTY	simulateur de TTY
SMILE,SSMILE	éditeur-assembleur avec éditeur immédiat
BASIC, SBASIC	interpréteur BASIC
FORTH, SFORTH	interpréteur FORTH

EPRO, SEPRO
ETEX, SETEX
PASCAL
XMON

```
        .TITLE  SYS.RF
        .PROC   Z80

        R E V I S I O N   1 - E

000001          FLOPPY  =       1
000000          COBUS   =       0

000000          DATAC   =       0
000001          MINI    =       1
000001          CLOCK   =       1

000400 SYS      =       400             ; System call number
000347 TEM      =       347             ; System restart

000327 FLO      =       327             ; Floppy restart
000357 COB      =       357             ; Cobus restart

015747 ?MEM     =       33*SYS+TEM      ; Give memory size
042540 MAXMEM   =       42540           ; Last available ram location

005347 ?IRST50  =       12*SYS+TEM      ; Free for user
005747 ?IRST60  =       13*SYS+TEM      ; Free for user
006347 ?IRCNMI  =       14*SYS+TEM      ; Non masquable interrupts
007747 ?EN:50   =       17*SYS+TEM      ; Enable 50 Hz int70
017747 ?INTUS   =       37*SYS+TEM      ; Adr int70 rout under system
022347 ?IRTC    =       44*SYS+TEM      ; Adr routine exec each 20 ms
001347 ?ITIMER  =       2*SYS+TEM       ; Timer initial value
001747 ?ITIMAD  =       3*SYS+TEM       ; Adr exec when timer reach 0

014347 ?CALPHA  =       30*SYS+TEM      ; Clear alphanumeric screen
014747 ?CGRA    =       31*SYS+TEM      ; Clear graphic screen
010747 ?IALPHA  =       21*SYS+TEM'     ; Alpha only
011347 ?IGRA    =       22*SYS+TEM      ; Graphic only
011747 ?IAGRA   =       23*SYS+TEM      ; Alpha and graphic together
000001 XX       =       1               ; Mult factor for x coord
000400 YY       =       400             ; Mult factor for y coord
003000 SETP     =       3000            ; Set point
003003 CLRP     =       3003            ; Clear point
003006 INVP     =       3006            ; Invert point
003011 TESTP    =       3011            ; Test if point set
003014 CMOD     =       3014            ; Segment mode
003017 SEGA     =       3017            ; Segment (absolute mode)
003022 SEGRE    =       3022            ; Segment (relative mode)
003025 CONTA    =       3025            ; Draw consec segments (abs)
003030 CONTRE   =       3030            ; Draw consec segments (rel)
000376 CMODE    =       376             ; Define segment mode
000377 ENDC     =       377             ; Code for end of table
000000 EFFAC    =       0               ; Efface le trait correspondant
000001 TMINCE   =       1               ; Trait mince continu
000002 TGROS    =       2               ; Gros trait continu
000003 PSERRE   =       3               ; ..............
000004 PLARGE   =       4               ; - - - - - - - -
000005 TPOINT   =       5               ; -.-.-.-.-.
000006 CROIX    =       6               ; + + + + + + + +
000010 SAUT     =       10              ; Saute sans rien modifier

032347 ?ALGRA   =       64*SYS+TEM      ; Translate coor. alpha in gra
037747 ?GRAAL   =       77*SYS+TEM      ; Translate coor. gra in alpha

010347 ?IDICAR  =       20*SYS+TEM      ; Init text window
053347 ?IDIS    =       126*SYS+TEM     ; Init display like a TTY
020347 ?SETCURSO=       40*SYS+TEM      ; Set cursor at given coord.
020747 ?GETCURSO=       41*SYS+TEM      ; Gives coordinates and char.
026347 ?IPOINTER=       54*SYS+TEM      ; Init pointer character
000347 ?DICAR   =       0*SYS+TEM       ; Display character
042520 OUTCAR   =       42520           ; Address print routine (0 no p
050347 ?IMODE   =       120*SYS+TEM     ; Inverted display if A=1
000001 INVERT   =       1               ; Inverted
000110 HEXA     =       110             ; Hexa normal

021347 ?SPACE   =       42*SYS+TEM      ; Put a space
050747 ?TAB     =       121*SYS+TEM     ; Put a tab
051347 ?CLEAR   =       122*SYS+TEM     ; Clear window
054347 ?DEL     =       130*SYS+TEM     ; Delete last displayed char.
054347 ?BACKDEL=        ?DEL
057747 ?BACKSP  =       137*SYS+TEM     ; Skip pointer back one position
021747 ?RETURN  =       43*SYS+TEM      ; Do a carriage return
055347 ?EFECR   =       132*SYS+TEM     ; Clear to end of screen
036347 ?EFLI    =       74*SYS+TEM      ; Clear end of line, dont move^

000015 CR       =       15              ; Carriage return
000014 SCR      =       14              ; Shift cr
000012 CCR      =       12              ; Control CR
000012 LF       =       CCR             ; Line feed
000014 FF       =       SCR             ; Form feed

000011 TAB      =       11              ; Horizontal tab
000013 STAB     =       13              ; Shift tab
000003 CTAB     =       3               ; Control tab

000004 END      =       4               ; end key
000005 SEND     =       5               ; shift end

000007 CEND     =       7               ; control end
000007 BELL     =       CEND            ; Buzze

000010 BS       =       10              ; Back space
000177 SBS      =       177             ; shift bs
000001 CBS      =       1               ; control bs
000177 DEL      =       SBS             ; Delete Key
000001 CADR     =       CBS             ; Cadratin

000040 SPACE    =       40              ; Space bar
000002 EFIX     =       2               ; Fixed space

000033 ESC      =       33              ; escape (select) (alt mode)
000006 ESCG     =       6               ; Smaky escape
000033 CESC     =       ESC             ; control esc
000006 SESC     =       ESCG            ; shift escape

000002 CLEARW   =       'B-100          ; Effacement fenetre
000005 INV      =       'E-100          ; Display in inverted video
000004 NORM     =       'D-100          ;   "     "  normal    "
000037 DEFINE   =       37              ; Define key
000036 MACRO    =       36              ; Macro key
000016 FCNT     =       16              ; Function Key

000100 CURSOR   =       100 -
000040 COPY     =       40
000020 KILL     =       20
000010 PROGRA   =       10              ; all thesex are function Keys
000004 CHAN     =       4
000002 SEARCH   =       2
000001 CHANGE   =       1

003347 ?DITEX   =       6*SYS+TEM       ; Display text terminated by 0
021747 ?TEXT    =       123*SYS+TEM     ; Come w address fol the call
053347 ?TEXTIM  =       133*SYS+TEM     ; Print text following call
017347 ?DITEB   =       36*SYS+TEM      ; Display text term by (B)

006747 ?IFCAR   =       15*SYS+TEM      ; Test if key FIFO empty
007347 ?GETFCN  =       16*SYS+TEM      ; Read function Key
000747 ?GETCAR  =       1*SYS+TEM       ; Get char from FIFO, wait for
042522 INCAR    =       42522           ; Addr for input rout (0 no in)
002747 ?GETLINE=        5*SYS+TEM       ; Get a line terminated by a CR
016747 ?GETLBCDE=       35*SYS+TEM      ; Get a line term by -B C-D E-
042700 BUFLINE=         42700           ; Addr of GETLINE buffer

026747 ?JUMPIDE=        55*SYS+TEM      ; JUMP  @(DE)
031747 ?CALLIDE=        63*SYS+TEM      ; CALL  @(DE)
047747 ?GETARG  =       117*SYS+TEM     ; LOAD  DE,@(SP)+

027347 ?COMPHLDE=       56*SYS+TEM      ; COMP  HL,DE
027747 ?MUL     =       57*SYS+TEM      ; MUL 16 bits HLDE=HL*DEXDC
030347 ?DIV     =       60*SYS+TEM      ; DIV  16 bits DE=HLDE/BC

052347 ?BINBCD  =       124*SYS+TEM     ; Convert HL into BCD in AHL
052747 ?BCDBIN  =       125*SYS+TEM     ; Convert HL BCD into HL binary

030747 ?JUMPCAR=        61*SYS+TEM      ; Search char in .SW table jump
031347 ?SCAR    =       62*SYS+TEM      ; Search char in table
016347 ?ENANCH  =       34*SYS+TEM      ; Search string in RAM
055747 ?HLDELAY=        133*SYS+TEM     ; Delay prop. to HL
053747 ?DELAY   =       127*SYS+TEM     ; Delay prop to fol. word

015347 ?BUZZ    =       32*SYS+TEM      ; Buzzer
037347 ?BEEP    =       76*SYS+TEM      ; Buzzer with length and freq
036747 ?PLAY    =       75*SYS+TEM      ; Play a piece of music
000274 DO       =       274
000254 RE       =       254
000235 MI       =       235
000215 FA       =       215             ; Theses are the notes to be
000175 SOL      =       175             ; played with ?PLAY system call
000165 LA       =       165
000156 SI       =       156
000146 DO1      =       146
000137 RE1      =       137
000127 MI1      =       127
000377 SILENCE=         377
000377 SIL      =       SILENCE

012347 ?RPR     =       24*SYS+TEM      ; Get byte Paper Reader (USA 4)
022347 ?IFRPR   =       44*SYS+TEM      ; Test if full, read if yes
013347 ?UPP     =       26*SYS+TEM      ; Write byte to Paper Punch
023347 ?IFUPP   =       46*SYS+TEM      ; Test if ready

012747 ?RMOD    =       25*SYS+TEM      ; Read byte from Modem
022747 ?IFRMOD  =       45*SYS+TEM      ; Test if full, read if yes
013747 ?UMOD    =       27*SYS+TEM      ; Write byte to MODEM (USART 6)
023747 ?IFUMOD  =       47*SYS+TEM      ; Test if ready

024347 ?RPAR    =       50*SYS+TEM      ; Read byte from par interface
024747 ?IFRPAR  =       51*SYS+TEM      ; Test if full, read if yes
025347 ?UPAR    =       52*SYS+TEM      ; Write byte to par interface
025747 ?IFUPAR  =       53*SYS+TEM      ; Test if ready

034347 ?AFBIN   =       70*SYS+TEM      ; Display (A) in binary
034747 ?AFHEX   =       71*SYS+TEM      ; Display (A) in hex
054747 ?AFXHL   =       131*SYS+TEM     ; Display HL in BCD or HEXA
035347 ?AFOCA   =       72*SYS+TEM      ; Display (A) in octal
035747 ?AFOHL   =       73*SYS+TEM      ; Display (HL) in octal

032747 ?INDEC   =       65*SYS+TEM      ; Get a decimal number from kbd
033347 ?INHEX   =       66*SYS+TEM      ; Get an hexa number from Kbd
033747 ?INOC    =       67*SYS+TEM      ; Get an octal number from kbd

040347 ?INMON   =       100*SYS+TEM     ; Get in octal or hexa
040747 ?AFMONA  =       101*SYS+TEM     ; Display in oct or hexa
041347 ?AFMOHL  =       102*SYS+TEM     ; Display in oct or hexa

041747 ?LOADBI  =       103*SYS+TEM     ; Load binary PDP11 format
042347 ?PUNBI   =       104*SYS+TEM     ; Punch binary (without end tr)
042747 ?AMORCE  =       105*SYS+TEM     ; Punch trailer

060347 ?RDCLK   =       140*SYS+TEM     ; Read clock module
060747 ?WRCLK   =       141*SYS+TEM     ; Write clock module
061347 ?SJOUR   =       142*SYS+TEM     ; Set day of week
061747 ?SDAY    =       143*SYS+TEM     ; Set date, month, year
062347 ?STOD    =       144*SYS+TEM     ; Set hour, mn, sec
062747 ?GJOUR   =       145*SYS+TEM     ; Get day of week
063347 ?GDAY    =       146*SYS+TEM     ; Get date, month, year
063747 ?GTOD    =       147*SYS+TEM     ; Get hour, mn, sec

043347 ?SAVESTAT=       106*SYS+TEM     ; Save complete status
043747 ?PRSTAT  =       107*SYS+TEM     ; Print status
044347 ?EXECUTE=        110*SYS+TEM     ; Execute from saved status

056347 ?MON     =       134*SYS+TEM     ; Monitor re-entry without init
056747 ?TRAPPE  =       135*SYS+TEM     ; Monitor entry with status dis

042506 SAVLGO   =       42506           ; Address for key G
042516 SAVS     =       42516           ; Address for key S
042532 TABEX    =       42532           ; Adr of sys call ex table

042525 LPSTAT   =       42525           ; Printer status register
042567 SNBLIN   =       42567           ; Printer paper line format
042677 FLOFF    =       42677           ; Floppy timeout register

040000 SALPHA   =       40000           ; Alphanumeric screen
000024 LINES    =       20.             ; Number of text lines
000100 NCAR     =       64.             ; Number of char. per line
000024 NLI      =       LINES
042400 ENDALPHA =       SALPHA+LINES*NCAR

043000 SGRA     =       43000           ; Graphic screen
000074 LIGRA    =       20.*3           ; 60 lines or 120 dot lines
052400 ENDGRA   =       SGRA+LIGRA*NCAR

        ; I/O addresses

000002 PAR      =       2               ; Parallel I/O connector
                                        ; Data in and out
000003 SPAR     =       PAR+1
000000  BADYP   =       0
000001  BFULP   =       1

000003 HP       =       3               ; Loudspeaker peripheral

        ; Floppy systems calls

000727 ?DELETE  =       1*SYS+FLO       ; Remove entry in directory
001327 ?RENAME  =       2*SYS+FLO       ; Change directory entry of a f
002327 ?CLOSE   =       4*SYS+FLO       ; Close a file after read or wr
002727 ?RESET   =       5*SYS+FLO       ; Close all currently opened f
003727 ?RILINE  =       7*SYS+FLO       ; Read on
004727 ?WFLINE  =       11*SYS+FLO      ; Write one line to file
005327 ?LGO     =       12*SYS+FLO      ; Load and execute a binary f
007727 ?RUN     =       17*SYS+FLO      ; Load all
011727 ?CREATE  =       23*SYS+FLO      ; Create a file
012327 ?OPEN    =       24*SYS+FLO      ; Open a file for reading
012727 ?RDBYTE  =       25*SYS+FLO      ; Read bytes from file
013327 ?WRBYTE  =       26*SYS+FLO      ; Write bytes to file
014327 ?ERROR   =       30*SYS+FLO      ; Print error message

        ; Errors

000000 EREOF    =       0               ; End of file
000011 ERFRE    =       11              ; File already exists
000012 ERFILE   =       12              ; File does not exist
000013 ERNAM    =       13              ; Illegal file name
000005 ERLLI    =       5               ; line too long

000001          .END
```



```
.TITLE APPELS.RF        ;JDN.FAUCON
CARACTERISTIQUES DES APPELS
&:FAUCON.
```

Appels d'extension du repertoire d'instruction

57 ?MUL HL+DE*BC => HLDE

```
in      DE multiplicande, BC multiplicateur, HL valeur add
out     HLDE produit 32 bits, A=0
mod     F, A, HL, DE
```

60 ?DIV HLDE : BC => DE, reste HL

```
in      HLDE dividende, BC diviseur<>0
out     DE contient, HL reste, CS si dep avec A=nbre cycles restant
mod     F, A, HL, DE
```

117 ?GETARG LOAD DE,@(SP)+ Lecture d'un argument suivant l'appel

```
in      SP pointe l'adresse de retour
out     DE parametre, SP pointe la nouv adresse de retour
mod     DE
```

Appels de recherche et débranchement

62 ?SCAR Cherche un caractere ASCII

```
in      A caractere
        DE adresse table  (.BW  car_ascii,adresse_appel)
                .B 0 car fin table
out     CC si trouvé: retour avec DE=adresse du mot suivant le byte
        CS si non trouvé: retour avec DE=adresse suiv la table
mod     F, DE
```

61 ?JUMPCAR Jump selon un caractère ASCII

```
in      A caractere
        DE adresse table  (.BW  car_ascii,adresse_saut)
                .B 0 car fin table
out     -trouvé: saut avec pile corrigée
        -non trouvé: retour avec DE=adresse suiv la table
mod     DE
```

34 ?BRANCH Cherche un nom en memoire

```
in      HL pointeur mot donne
out     CC si trouve
         DE adresse debut en RAM, HL pointe la fin du mot donne+2 bytes
        CS si non trouve
         HL pointe le debut du mot donne
mod     F,A,BC,DE,HL

Exemple de debut en RAM
        .ASCII  "<252>mot_clef<200>"
        prem instruction executee
```

Appels d'initialisation de positions

33 ?MEM Determine la fin de la memoire

```
in      -
out     HL adresse de fin
mod     HL
```

2 ?ITIMER
3 ?ITIMAD
4 ?ITTC
37 ?INTUS Init adresses indirectes
7 ?IRST10
10 ?IRST20
12 ?IPSTE0
13 ?IRST60
11 ?IRST70
14 ?IADREI

```
in      HL adresses
out     -
mod     -
```

17 ?INIE0 Initialisation interrupt 50 hz

```
in      A =0 plus d'interruptions 50 Hz
out     -
mod     -
```

30 ?CALPHA Efface l'ecran alpha

```
in      -
out     HL=CALPHA, A=0, CC
mod     F, A, HL
```

31 ?CGRA Efface l'ecran graphique

```
in      -
out     A=0, CC
mod     F, A
```

21 ?IALPHA Mode alpha seul

```
in      -
out     -
mod     -
```

22 ?IGRA Mode graphique seul

20 ?IMIX Mode alpha et gra superpose

```
in      C0 mode petits points
out     -
mod     -
```

Appels d'initialisation et conversion de pointeurs

20 ?IDICRR Initialise la longueur et le debut du display

```
in      B numero 1e ligne  C nombre de lignes tpos?400(octets)
                C=0 et D+C>19, non acceptes
out     -
mod     -
```

125 ?IDIS Init display

```
in      C number of lines
out     -(curseur)
mod     B
```

54 ?IPOINTER A caractere pointeur

```
in      -
out     -
mod     -
```

40 ?CETOURSOR Repositionne le pointeur

```
in      HL coord
out     pointeur
mod     pointeur
```

Appel de bruitage

32 ?DUMMY

```
in      -
out     -
mod     -
```

76 ?SOUND Note approximative

```
in      A duree (A:0-2) et période (A,3-7)
        Si A=377 --> silence
out     -
mod     -
```

75 ?PLAY Joue un morceau (notes approximatives)

```
in      HL pointeur au debut du morceau (terminé par un 0)
out     HL pointe morceau suivant éventuel
mod     F, A, HL
```

Appels d'affichage de caracteres sur l'ecran

132 ?EFEC Efface l'ecran depuis le pointeur

```
in      -
out     A=0
mod     F A
```

0 ?DICAR Affiche un car dans la fenetre
ou appelle une routine d'impr. COUTCAR

```
in      pointeur, A caractere, OUTCAR
out     pointeur
mod     pointeur
```

Appels d'affichage de chaines de caracteres

6 ?DITEX Affiche une chaine .ASCIZ (DITEXC accepte)

```
in      pointeur, HL adresse debut, 0 terminateur, OUTCAR
out     pointeur, HL pointe pos suivante (deb texte suiv)
mod     pointeur, HL
```

123 ?TEXT Idem avec adresse suivant l'appel

```
in      (SP)
out     -
mod     -
```

136 ?TEXTIM

```
in      PC pointe le texte
out     PC pointe la suite du programme
mod     -
```

35 ?DITEB Affiche une chaine avec un terminateur quelconque

```
in      pointeur, HL adresse debut, B terminateur, OUTCAR
out     pointeur, HL pointe pos suiv
mod     pointeur, HL
```

74 ?EFLI Efface la ligne depuis le pointeur sans le deplacer

```
in      pointeur
out     A=0
mod     A
```

Appels de lecture du clavier (par son FIFO)

1 ?GETCAR Lit le FIFO du clavier ou appelle
une routine de lecture GINCAR

```
in      INCAR
out     A caractere
mod     F,A
```

15 ?IFCAR Lit l'etat du clavier (FIFO)

```
in      -
out     CS aucune touche en attente, A non mod
        CC touche, A code ASCII
mod     F,A
```

16 ?GETFON Lit les touches fonctions

```
in      -
out     A touches fonctions, 00 si aucune touche fonction pressee
mod     F,A
```

5 ?GETLINE Attend une ligne terminee par un CR

```
in      pointeur, INCAR
out     pointeur, HL adr debut de chaine, 0 a la fin de la chaine
                CS si retour car buffer plein
mod     pointeur, HL, F,A
```

36 ?GELBCDE Attend une ligne terminee par diff car
voir exemple routines INCC, INDEC

```
in      pointeur, B, C, D, E, INCAR
out     pointeur, HL adr debut buffer, A der car, 0 term
                CS si retour car buffer plein
                revi B touche fonction der car
mod     F, A, HL, revi B
```

Appels pour periphériques

```
21 ?CI      Attend et lit
23 ?CIO     Transferts lecture USART
60 ?PAR     Interface parallele

in      -
out     A caractere
mod     F, A

44 ?IFRER
45 ?IFRMOD  Test et lit si le caractere est disponible
51 ?IFRPAR

in      -
out     CS pas de caractere (FULL=0)
        CC car lu dans A (clear le FULL)
mod     F, A

26 ?LPP
27 ?LMOD    Transferts ecriture USART
52 ?LPAR    ou interf //

in      A car
out     -
mod     F

46 ?LFUL
47 ?LFUMOD  Teste et envoie si pret
53 ?LFUPAR
in      -
out     CC pas pret
        CC pret (READY=1)
mod     F
```

Appels d'extension du repertoire d'instruction (suite)

```
55 ?JUMPIDE     JUMP @(IX)

in      DE
out     DE incr de 2, saut effectué, pile correcte
mod     DE

63 ?CALLIDE     CALL @(IX)

in      DE
out     DE incr de 2, appel effectué, pile correcte
mod     DE

41 ?GETCURSOR   Donne les coordonnees du pointeur
                et le caract sous le pointeur

in      pointeur
out     A caract        HL coord
mod     HL,A

;.B     XMOD,xxxx peut intervenir n'importe où
;.B     XMOD,xxxx n'est pas nécessaire dans
        INC     IX              ;.B     XMOD,xxxx
```

* S O F T G R A P H I Q U E *

```
        ----------)
        I         X
        I
        I
        I
        V Y
 0 <= X <= 377
 0 <= Y <= 177
XX      = 1
YY      = 400   ;Facteurs multiplicatifs pour init DE, HL
```

3025 CONTA

```
Trace le contour avec les points donnes en absolu.
Format de la table:
TABLE:.BYTE XMODE,TMINCE        ;Mode trait mince
       .BYTE   m        n       ;m -> Y , n -> X
       .BYTE   m        n
       .BYTE   ...
       .BYTE   ENDC             ;Fin de la table

ENDC    =377
XMODE   =376
EFFAC   =0      ;Efface le trait correspondant
TMINCE  =1      ;Trait mince continu
TGROS   =2      ;Gros trait continu
PSEPPE  =3      ; . . . . . .
PLARGE  =4      ; - - - - - - -
TPOINT  =5      ; . - . - . - .
CROIX   =6      ; + + + + + + +
SAUTE   =10     ;Saute sans rien modifier

;.BYTE XMODE,xxx .La definition du mode se fera alors
in      IX: table
out     IX: fin table
mod     IX
```

3030 CONTR

```
Trace le contour avec les points donnes en relatif.
Les valeurs m et n peuvent etres positives ou
negatives. Par exemple, la valeur -3 sera representee
par 203 (et non 375 !!)
Les valeurs m et n doivent etres comprises entre
-177 et +177. (impossibilite de traverser completement
l'ecran horizontalement en une seule fois).

in      IX: table       DEBUT -> ED (X,Y)
out     IX: fin table   ED pointe le point d'arrivee
mod     IX
```

3017 SEGA

```
Relie deux points par une droite selon le mode predefini
Absolu.
in      DEBUT   -> ED (X,Y)
        FIN     -> HL (X,Y)
out     ED -> FIN
mod     F,A,BC,DE,HL
```

3020 SEGRE

```
Relie deux points par une droite selon le mode predefini.
Relatif.
in      L -> DEX (7 BITS + SIGNE)   H -> DEY (7 BITS + SIGNE)
out     ED -> FIN DU TRAIT
mod     tout
```

3014 CMOD

```
Change le mode de trait
in      A -> MODE
out     MODE
mod     A
```

```
3000 SETP   ---
3003 CLRP   ---
3006 INVP   ---

Set, clear, inverse un point sur l'ecran
dont on a donne la coordonnee dans (E,D).

in      E -> COORDONNEE EN X
        D -> COORDONNEE EN Y
out     -
mod     -
```

3011 TESTP

```
Teste un point
dont on a donne la coordonnee dans (E,D).

in      E -> COORDONNEE EN X
        D -> COORDONNEE EN Y
out     EQ si un point est present, NE autrement
mod     - (AF pour TESTP)
```

Appels de lecture et conversion de pointeur

```
116 ?COMPTIME   Compare (HL) et (DE) longueur B octets
                poids faibles en tete

in      HL, DE pointeurs, B nb d'octets a comparer
out     idem, pointent la fin
        EQ si =, CS si <, CC si >=, SS si >
mod     HL, DE, F, A

64 ?LGRA        Translate coor. alpha in gra

in      HL coor. alpha
out     DE coor. gra
mod     DE

77 ?GRAL        Translate coor. gra in alpha

in      DE coor. gra
out     HL coor. alpha
mod     HL
```

Appels pour lecture et affichages de nombres

```
73 ?AFCHL       Affiche HL en octal

in      pointeur, HL contenu
out     pointeur
mod     pointeur

72 ?AFOCA       Affiche le carry et A (9 bits) en octal

in      pointeur, A nombre
out     pointeur
mod     pointeur

70 ?AFBIN       Affiche un nombre binaire

in      pointeur, A nombre
out     pointeur
mod     pointeur

71 ?AFHEX       Affiche un byte hexa ou BCD

in      pointeur, A nombre
out     pointeur
mod     pointeur

131 ?AFXHL      Affiche HL en hex ou BCD

in      HL nombre
out     -
mod     -

65 ?INDEC       Lecture nbre BCD avec delete
67 ?INOC        Lecture nombre octal
66 ?INHEX       Lecture nombre hexa

in      pointeur, INCAR
out     HL nombre, C nbre digits, EQ si C=0, A der car
mod     F, A, B, C, HL
```

Appels de controle d'execution

```
110 ?EXECUTE    ;Execute selon l'etat sauve en memoire
in      (positions memoire de 42453 a 42503)
out     tous les reg
mod     id

135 ?TRAFFE

in      -
out     -
mod

106 ?SAVESTAT   Sauvetage des registres dans la zone reservee du moniteur

in      tous les registres
out     - (adresses 42452 a 42503)
mod     A, F

107 ?RSTAT      Impression de l'etat du processeur

in      - (positions memoire de 42452 a 42503)
out     -
mod     A, F, B, DE, HL
```

Appels de transfert papier/cassette

```
103 ?LOADBU     Charge un binaire PDP11

in      C=0 chargeur absolu
        C=1-377  HL=adresse debut chargement
out     CALGO  initialise si auto start
mod     F A B C D E H L

102 ?PUNCH      ;PUNCH A LOADER OF MAX 250 BYTES

in      -
out     -
mod     F,A,B,

104 ?PUNBI      Perfore une amorce et le bloc format PDP11

in      HL adresse de debut, DE longueur, BC adr de restart
        IY correction d'adresse
out     -
mod     F A B C D E HL IY
```

Appels pour affichage et lecture d'un nombre sous moniteur

120 ?IMODE — Set INV flag, hex and cass flag

```
in      A=0 Octal and paper
        A=10 Hexa and paper
        A=20 Octal and cassette
        A=30 Hexa and cassette
        A=1 Inverted characters
out     -
mod     -
```

102 ?AFMON — Affichage HL en octal ou hexa

```
in      HL, pointeur, SAVHEX, OUTCAR
out     pointeur
mod     A, pointeur
```

101 ?AFMONA — Affichage A en octal ou hexa

```
in      pointeur, A, SAVHEX, OUTCAR
out     pointeur
mod     F, pointeur
```

100 ?INMON — Lecture pour moniteur selon SAVHEX

```
in      SAVHEX, INCAR
out     HL nombre, C nbre digits, EQ si C=0, A der car
mod     F, A, D, C, HL
```

Appels pour mesure du temps

111 ?INCSEC — Ajoute 2 centièmes de seconde

```
in      HL pointeur, (HL) centièmes et secondes
out     HL incr de 2, (HL), CS si report des secondes
mod     HL, (HL), A, F
```

112 ?INCHOUR — Ajoute une minute si CS

```
in      CS, HL pointeur, (HL) minutes et heures
out     HL incr de 2, (HL), CS si report des heures
mod     HL, (HL), A, F
```

113 ?INCDAY — Ajoute un jour si CS
Pour les ann biss, tester si 29 fev et forcer en 29

```
in      CS, HL pointeur, (HL) jour et mois
out     HL incr de 2, (HL), CS si report des mois
mod     HL, (HL), A, F
```

114 ?INCYEAR — Ajoute une année si CS (ADDC (HL))

```
in      HL pointeur, (HL) année (2 digits), CS ajoute 1
out     HL incr de 1 si CS, (HL), CS si report
mod     HL, (HL), A, F
```

115 ?AFTIME — Affiche l'heure

```
in      pointeur, HL pointeur aux poids forts
        A format ex 00010000 pour 03 12 18 54 (12 mars a 18h54)
        3 octets avant . 1 octet apres
out     pointeur
mod     pointeur, HL
```

124 ?BINBCD — Convertit 16 bits de binaire en BCD

```
in      HL nombre binaire
out     AHL nombre BCD equivlent
mod     F A HL
```

125 ?BCDBIN — Convertit 4 digits de BCD en binaire

```
in      HL nombre BCD
out     HL nombre binaire
```

127 ?DELAY / **133 ?HLDELAY** — Attente de .. millisecondes

```
in      HL durée
out     -
mod     -
```

42 ?SPACE / **43 ?RETURN** / **121 ?TAB** — Execute un CR

```
in      pointeur, OUTCAR
out     pointeur
mod     pointeur
```

122 ?CLEAR / **130 ?DELETE**

```
in      -
out     -
mod     -
```

137 ?BACKSPACE

```
in      -
out     A car sous le pointeur deplace
        EQ si debut de ligne
```

56 ?COMPHLDE — Compare HL et DE

```
in      HL, DE
out     EQ si HL=DE, LO si HL<DE
mod     F
```

140 ?RDCLK — Lecture de toutes les informations du chip horloge

```
in      DE ^to a 7 bytes buffer
out     -
mod     -
```

141 ?WRCLK — Initialisation du chip horloge

```
in      DE ^to a 7 bytes buffer to be written
out     -
mod     buffer
```

142 ?SJOUR — Initialisation du jour de la semaine

```
in      A =1 lundi  ...  =7 dimanche
out     -
mod     F, DE, HL
```

143 ?SDAY — Initialisation de la date seulement

```
in      A=dd, D=mm, C=yy
out     -
mod     F,DE,HL
```

144 ?STOD — Initialisation de l'heure seulement

```
in      A=hh, D=mm, C=ss
out     -
mod     F,DE,HL
```

145 ?GJOUR — Lecture du jour de la semaine seulement

```
in      -
out     A=dd, jour de la semaine (1=lundi, 7=dimanche,etc.)
mod     A,F,HL
```

146 ?GDAY — Lecture de la date seulement

```
in      -
out     A=dd, D=mm, C=yy
mod     A,F,BC,DE,HL
```

147 ?GTOD — Lecture de l'heure seulement

```
in      -
out     A=hh, D=mm, C=ss
mod     A,F,BC,DE,HL
```

2.16 LISTING DU SYSTEME ET DU MONITEUR

```
SMAKY:SYS1E.LS
19
80/09/21 15:37
```

```
/09/21 15 26:28  TABLE OF CONTENTS                              01-01
```



```
/09/21 15:26:28  SYSMON                                         01-01

                        .TITLE  SYSMON
                        .PROC   Z80

                ;*** SYSTEM SMAKY ET MONITEUR ***
                ;DANIEL ROUX 27.02.78,REVU JUIN 78
                ;781026;781023;781022;780823;780817;780812;JDN 780810
                ;DR781222,790120;790413;JDN790503 791025/29
                ;AC800310

000105          VAR     =       'E      ;No de variante affiche

                ;Conditionnal assembly system parameter

000000          S262    =       0       ;Generateur de caractere 74S262
000001          CLAFAU  =       1       ; scanned keyboard

000000          DATAS   =       0       , datas like smaky
000000          DATAC   =       0       , datac like datac

000000          TESTRAM =       0       ;For test in RAM ?? A verifierx

000001          AUTOST  =       1       ;Automatic BRANCH in BOOT

000000          BEGUIN  =       0       ;LOAD and PUNCH on //

000000          COBUS   =       0       ;First line of alpha screen

000000          DAT     =DATAS+DATAC

        ;*************************************************************
        ;*                                                           *
        ;*      A T T E N T I O N   ! ! !                            *
        ;*                                                           *
        ;*      IL EST INTERDIT DE FAIRE DES MODIFICATIONS           *
        ;*      DANS CE PROGRAMME A PARTIR D'ICI.                    *
        ;*      UNE EVENTUELLE MODIFICATION DOIT ETRE FAITE          *
        ;*      DANS LE FICHIER SYS1A.SR                             *
        ;*                        --------                           *
        ;*                                                           *
        ;*                                              DR           *
        ;*                                                           *
        ;*************************************************************

000137 CARPOINT= '_     .Car displayed as pointer

        ;
        ;       R E V I S I O N  1
        ;       ******************

        ;       C L A V I E R   B A L A Y E

        ;APRES UN RESET, LE SYSTEME VIDE TOUS LES BUFFERS DU CLAVIER
        ;INITIALISE LE TIMER PUIS REGARDE COMBIEN IL Y A DE RAM.
        ;IL FAUT ALORS DIRE QUEL EST LE PROGRAMME QUE L'ON VEUT
        ;DEMARRER EN DONNANT SON NOM PUIS UN "CR".

        ;VOIR NOTICE DES APPELS SYSTEMES DANS SYS.RF, APPELS.RF

        ;..TITLE APPELS.RF        ;JDN FRUCON
        ;.CARACTERISTIQUES DES APPELS
        ;Generation sous LED par 1000DN:.01FAZZo3b.
        ;puis renomer ZZ et tuer la 1ere page de APPELS.RF
```

```
/09/21 15 26:28  SYS1.RF                                        02-01

        .TITLE SYS1.RF

000400 SYS      = 400
000347 TEM      = 347

003747 ?IRST10  =  7*SYS+TEM    .Floppy interrupt address
004347 ?IRST20  = 10*SYS+TEM    .Floppy system calls
005347 ?IRST50  = 12*SYS+TEM    .Free for user
006747 ?ITRAP   = 13*SYS+TEM    ,Trap or special user's interrupts
004747 ?IRST70  = 11*SYS+TEM    ;System interrupts
006347 ?IADNMI  = 14*SYS+TEM    ;Non masquable interrupts

007747 ?ENI50   = 17*SYS+TEM    ,Enable 50 Hz int70
017747 ?INTUS   = 37*SYS+TEM    ;Adr int70 routine under system
002347 ?IRTC    =  4*SYS+TEM    ;Adr routine exec each 20 ms
001347 ?ITIMER  =  2*SYS+TEM    ,Timer initial value
001747 ?ITIMAD  =  3*SYS+TEM    ;Adr routine exec when timer reach 0

014347 ?CLRALPHA = 30*SYS+TEM   ,Clear alphanumeric screen
014747 ?CLRGRA  = 31*SYS+TEM    ;Clear graphic screen
```

```
010747 ?IFLAPHA = 21*SYS+TEM    ,Alpha only
011347 ?IGRA    = 22*SYS+TEM    ,Graphic only
011747 ?IAGRA   = 23*SYS+TEM    ,Alpha and graphic together

032347 ?ALGRA   = 64*SYS+TEM    ,Translate coor. alpha in gra
037747 ?GRAAL   = 77*SYS+TEM    ,Translate coor. gra in alpha

015747 ?MEM     = 33*SYS+TEM    ,Give memory size

010347 ?IDICAR  = 20*SYS+TEM    ,Init text window
053347 ?IDIS    =126*SYS+TEM
020347 ?SETCURSO=40*SYS+TEM     ,Set cursor at given coordinates
020747 ?GETCURSO=41*SYS+TEM     ,Gives coordinates and char. of cursor
026347 ?IPOINTER=54*SYS+TEM     ,Init pointer character

000347 ?DICAR   =  0*SYS+TEM    ,Display character (ctrl-U clear)
021347 ?SPACE   = 42*SYS+TEM    ,Put a space
021747 ?RETURN  = 43*SYS+TEM    ,Put a CR
050747 ?TAB     = 121*SYS+TEM   ,Put a TAB
054347 ?DEL     = 130*SYS+TEM
057747 ?BACKSPACE=127*SYS+TEM   ,Back one position
051347 ?CLEAR   = 122*SYS+TEM   ,Clear text window
037347 ?EFLI    = 74*SYS+TEM    ,Clear end of line without moving poi
051747 ?EFECR   = 123*SYS+TEM   ,Clear page from pointer

003347 ?DITEX   =  6*SYS+TEM    ,Display text terminated by 0
051747 ?TEXT    =123*SYS+TEM    ,Same with address following the call
057347 ?TEXTIM  = 113*SYS+TEM   ,Same with text following the call
017347 ?DITXB   = 36*SYS+TEM    ,Display text terminated by (B)

006747 ?IFCIR   = 15*SYS+TEM    ,Test if key FIFO empty
017347 ?GETFON  = 36*SYS+TEM    ,Read function key
000747 ?GETCHAR =  1*SYS+TEM    ,Get char from FIFO, wait for it

002747 ?GETLINE =  5*SYS+TEM    ,Get a line terminated by a CR
046747 ?GETLBCDE=35*SYS+TEM     ,Get a line term by -B C-D E-

026747 ?JUMPIDE= 55*SYS+TEM     ,JUMP @(DE)
031747 ?CALLIDE= 63*SYS+TEM     ,CALL @(DE)
047747 ?GETARG  =117*SYS+TEM    ;LOAD  HL @(SP)+

027347 ?COMPHLDE=56*SYS+TEM     ,COMP HL,DE
027747 ?MUL     = 57*SYS+TEM    ,MUL  HL*DE*BC => HLDE
030347 ?DIV     = 60*SYS+TEM    ,DIV  HLDE BC =>DE reste HL
052347 ?BINBCD  = 124*SYS+TEM   ,binaire --> BCD
052747 ?BCDBIN  = 125*SYS+TEM   ,BCD --> binaire

010747 ?JUMPCAR= 21*SYS+TEM     ;Search char in a .BJ table and jump
031347 ?SCAR    = 62*SYS+TEM    ,Search char in table
016347 ?BRANCH  = 34*SYS+TEM    ,Search string in RAM and LOADROM

053747 ?DELAY   =127*SYS+TEM    ;Delay in ms
055747 ?HLDELAY=133*SYS+TEM     ,Delay in HL

015347 ?BUZZ    = 32*SYS+TEM    ,Buzzer
037347 ?BEEP    = 76*SYS+TEM    ;Buzzer with length and frequency con
036747 ?PLAY    = 75*SYS+TEM    ;Play a piece of music

012347 ?RPR     = 24*SYS+TEM    ,Get byte from Paper Reader (USART 4)
022347 ?IFRPR   = 44*SYS+TEM    ,Test if full, read if yes
013347 ?WPP     = 26*SYS+TEM    ,Write byte to Paper PunchX
023347 ?IFWPP   = 46*SYS+TEM    ,Test if ready

012747 ?RMOD    = 25*SYS+TEM    ,Read byte from Modem
022747 ?IFRMOD  = 45*SYS+TEM    ,Test if full, read if yes
013747 ?WMOD    = 27*SYS+TEM    ,Write byte to MODEM (USART 6)
023747 ?IFWMOD  = 47*SYS+TEM    ,Test if ready

024347 ?RPAR    = 50*SYS+TEM    ,Read byte from parallel interface
024747 ?IFRPAR  = 51*SYS+TEM    ,Test if full, read if yes
025347 ?WPAR    = 52*SYS+TEM    ,Write byte to parallel interface
025747 ?IFWPAR  = 53*SYS+TEM    ;Test if ready

034347 ?AFBIN   = 70*SYS+TEM    ,Display (A) in binary
034747 ?AFHEX   = 71*SYS+TEM    ,Display (A) in hex
035347 ?AFOCA   = 72*SYS+TEM    ,Display (A) in octal
035747 ?AFOHL   = 73*SYS+TEM    ,Display (HL) in octal
054747 ?AFXHL   =131*SYS+TEM    ;Display hl in hex

032747 ?INDEC   = 65*SYS+TEM    ,Get a decimal number from keyboard
033347 ?INHEX   = 66*SYS+TEM    ,Get an hexa number from keyboard
033747 ?INOC    = 67*SYS+TEM    ,Get an octal number from keyboard

040347 ?INMON   = 100*SYS+TEM   ,Get in octal or hexa
040747 ?AFMONA  = 101*SYS+TEM   ;Display in oct or hexa .
041347 ?AFMOHL  = 102*SYS+TEM   ;Display in oct or hexa

050347 ?IMODE   = 120*SYS+TEM   ,Set octal/hex, paper/cassette flag

041747 ?LOADBI  = 103*SYS+TEM   ,Load binary PDP11 format
042347 ?PUNBI   = 104*SYS+TEM   ,Punch binary (without end trailer)
042747 ?AMORCE  = 105*SYS+TEM   ,Punch trailer

043347 ?SAVESTAT=106*SYS+TEM    ,Save complete status
043747 ?PRSTAT  = 107*SYS+TEM   ,Print status
044347 ?EXECUTE= 110*SYS+TEM    ,Execute from saved status

044747 ?INCSEC  = 111*SYS+TEM   ,Inc 1/100 and seconds
045347 ?INCHOUR= 112*SYS+TEM    ,Inc minutes and hours
045747 ?INCDAY  = 113*SYS+TEM   ,Inc day, month and day of week
046347 ?INCYEAR= 114*SYS+TEM    ,DINC (HL)      Inc year

046747 ?AFTIME  = 115*SYS+TEM   ,Display time

047347 ?COMPTIME= 116*SYS+TEM   ,Compare time

060347 ?RDCLK   = 140*SYS+TEM   ,Read clock module
060747 ?WRCLK   = 141*SYS+TEM   ,Write clock module
061347 ?SJOUR   = 142*SYS+TEM   ,Set day of week
061747 ?SDAY    = 143*SYS+TEM   ,Set date, month, year
062347 ?STIME   = 144*SYS+TEM   ,Set hour, mn, sec
062747 ?GJOUR   = 145*SYS+TEM   ,Get day of week
063347 ?GDAY    = 146*SYS+TEM   ,Get date, month, year
063747 ?GTIME   = 147*SYS+TEM   ,Get hour, mn, sec

056347 ?MON     = 134*SYS+TEM   ,Call monitor
056747 ?TRAPPE  = 135*SYS+TEM   ,Call trappe

        ,***    Memory definition

000000 ROM0     = 0
004000 ROM2     = ROM0+4000
010000 ROM3     = ROM0+10000

040000 RAM      = 40000
041000 [illegible] = RAM+1000
[illegible]
[illegible]
[illegible]
[illegible]
000100 [illegible] = 100
000004 [illegible] = 4                  ,No of lines for PROF

040000 SALPHA   = RAM           ,Alphanumeric sreen
000024 LINES    = 20.           ,Number of text lines
000100 NCAR     = 64.           ,Number of char. per line (fixe)
042400 ENDALPHA = SALPHA+LINES*NCAR

043000 SGRA     = RAM+3000      ,Graphic screen
000024 LIGRA    = 20.*[illegible]  ,20 lines or 120 dot lines
052400 ENDGRA   = SGRA+LIGRA*NCAR

[illegible]                     PROV
[illegible]
[illegible]
[illegible]
```

```
042077 FLCFF    = 42E77          .Floppy watchdog

        ;***    Peripheral definition

000000 CLA      = 0       ;Keyboard input, reset FULL and FULCLA
000177 MCLA     = 177     ;Data in bits 0-6
000200 FULC     = 200     ;Full bit on same [illegible]
000000 DELAY1   = 700./20.       ;First delay for repeat
000003 DELAY2   = 60./20.        ;Second delay for repeat

000001 SCLA     = CLA+1
000004 FULCLA   = 4
000020 S4       = 20      ;Free input bits
000040 S5       = 40

000015 CR       = 15
000040 SPH      = 40
000012 LF       = 12
000011 TAB      = 11
000177 DEL      = 177
000010 BS       = 10
000007 BELL     = 7
000007 BEL      = 7       ;Buzze
000000 NUL      = 0       ;Ignore par DICAR
000002 CLEARW   ='B-100 ;Effacement fenetre
000004 NORM     = 'D-100         ;Mode normal
000005 INV      = 'E-100         ;Mode inverse

000001 I50      = 1       ;50 Hz interrupt register
000010 FL50     = 10      ;50 Hz flag cleared by output RZ50 instruction
000001 RZ50     = 1       ;Reset 50 hz interrupt
000001 ENINT50  = 1       ;Interrupt enable

000000 MIDI     = 0       ;Display mode
000000 ALPHA    = 0
000010 NOALPHA  = 10
000004 GRA      = 4       ;Graphic (longer DMA)
000002 SMALL    = 2       ;Large dots on graphic screen
000001 SECOND   = 1       ;Optional second alpha screen (SMAKY 4 only)
000020 M4       = 20      ;4 bits available on keyboard or SIMSER connector
000040 M5       = 40      ; reserved for interrupt in next print
000100 M6       = 100
000200 M7       = 200

000002 PAR      = 2       ;Parralel I/O connector
        ;Data in and out
000003 SPAR     = PAR+1
        ;input
000001 RDYP     = 1
000002 FULP     = 2       ;Doit etre id a USART
000000 BRDYP    = 0
000001 BFULP    = 1
000100 S6       = 100     ;Free input bits
000200 S7       = 200

000003 HP       = 3

000004 USA1     = 4
000004 FR       = USA1
000004 FP       = FR
        ;serial data in and out
000005 SUSA1    = USA1+1
000005 SFR      = SUSA1
000005 SFP      = SFR
        ;output
000316 MODE     = 316     ;(8+2,no parity,x16)

000040 MS1      = 40
000002 M11      = 2
000005 COMMAND  = 5       ;in and out
000100 REMODE   = 100     ;Return in MODE
        ;input
000001 RDY1     = 1
000002 FUL1     = 2
000000 BRDY1    = 0
000001 BFUL1    = 1
000200 S71      = 200

000006 USA2     = USA1+2
000007 SUSA2    = USA2+1
000006 CAS      = USA2
000007 SCAS     = CAS+1
        ;output
000040 ME2      = 40
000002 M12      = 2
        ;input
000001 RDY2     = 1
000002 FUL2     = 2
000000 BRDY2    = 0
000001 BFUL2    = 1
000200 S72      = 200

000000                  .LOC    ROM0
000000  000             NOP     ;Pour PROM loader

                ;***    RAM POUR MONITEUR   ***

042453                  .LOC    ODSAVE+53

042453          000001 SAVI:    .BLKB   1
042454          000002 SAVIY:   .BLKW   1
042456          000002 SAVIX:   .BLKW   1
042460          000002 PAVHL.   .BLKW   1       ;REGISTRES SECONDAIRES
042462          000002 PAVDE    .BLKW   1
042464          000002 PAVBC:   .BLKW   1
042466          000002 PAVAF:   .BLKW   1
042470          000002 SAVHL:   .BLKW   1       ;REGISTRES PRIMAIRES
042472          000002 SAVDE:   .BLKW   1
042474          000002 SAVBC:   .BLKW   1
042476          000002 SAVAF:   .BLKW   1
042500          000002 SAVPC:   .BLKW   1
042502          000002 SAVSP:   .BLKW   1

042504          000002 SAVLGP:  .BLKW   1
042506          000002 SAVLGO:  .BLKW   1

042510          000001 SAVINT:  .BLKB   1
042511          000001 SAVMODI: .BLKB   1
042512          000001 SAVHEX   .BLKB   1
042513          000001 CHAMAC:  .BLKB   1       ;RESERVE POUR FLOPPY
042514          000002 RST30:   .BLKW   1
042516          000002 SAVS.    .BLKW   1

                ;***    RAM POUR SYSTEME   ***

042520          000002 OUTCAR.  .BLKW   1       ;Indirextion pour DICAR
042522          000002 INCAR    .BLKW   1       ;Indirection pour GETCAR
042524          000001 SAVAS.   .BLKB   1       ;Sauvetage temp routine systeme
042525          000001          .BLKB   1       ;Libre
042526          000002 SAVT     .BLKW   1       ;Touche TAB (retour au mon)
042530          000001 DREP     .BLKB   1       ;Delay for repeat
042531          000001 STINT:   .BLKB   1       ;Dernier etat de l'interruption
042532          000002 TABEX    .BLKW   1       ;Adr table appels > 200

042534          000002 RST40.   .BLKW   1
042536          000002 INTUS    .BLKW   1
042540          000002 MAXMEM:  .BLKW   1       ;Adr limite memoire
042542          000002 RST10    .BLKW   1
042544          000002 RST20    .BLKW   1
042546          000002 RST70    .BLKW   1
042550          000002 RST50.   .BLKW   1
042552          000002 RST60    .BLKW   1
042554          000002 ADNMI    .BLKW   1

042556          000002 TIMER    .BLKW   1       ;Valeur decompteur de temps
042560          000002 TIMAD    .BLKW   1       ;Adresse routine exec si TIMAD=0 (la prem
```

```
042562          000002 SUBINT.  .BLKW   1       Adr routine exec toutes les 20 ms (si #/
[illegible]                                     [illegible] ecran
[illegible]     000001 [illegible]TER .BLKB 1   [illegible] comme pointeur+text.
[illegible]                                     [illegible]
[illegible]                                     [illegible] fenetre
[illegible]                                     [illegible] fenetre (+1)
[illegible]
[illegible]
042577          000001 LA[illegible]    .BLKB   1       Dernier point de l'ecran et inter

                000200 ENDO     = 200

042600                  .LOC    ODSAVZ+200
042600          000012 NEWTA    .BLKB   10.
042612  200                     .BYTE   ENDO
042613          000012 OLDTA.   .BLKB   10.
042625  200                     .BYTE   ENDO
042626          000040 DEBSCLA: .BLKB   64./2
042666  200                     .B      ENDO

                        ;***    RAM POUR SOFT GRAPHIQUE ***

042670                  .LOC    SYSTSAV+150

042670          000001 SIMUL0:  .BLKB   1
042671  311                     .BYTE   311     .Code RET
042672          000002 SIMUL1   .BLKB   2
042674  311                     .BYTE   311     .Code RET
042675          000001 COUNTER: .BLKB   1
042676          000001 MODEP:   .BLKB   1

                042672 SAVDA=   SIMUL1
                042673 SAVADIO=SIMUL1+1

                        ;***    RAM POUR BUFFER DE GETLINE ***

042700                  .LOC    SYSTSAV+160

042700          000077 GWORD:   .BLKB   63.
042777  200                     .BYTE   200


                000100 LGSTACK =100             ,Place pour le stack en fin de RAM

000000                  .LOC    ROM0

000000  363                     IOF
000001  061 060 105             LOAD    SP,#SPUTIL+10   , pour eviter un debordement
000004  303 065 001             JUMP    RESET

000010                  .LOC    ROM0+10

000010  345             R10:    PUSH    HL
000011  052 142 105             LOAD    HL,RST10
000014  343                     EX      (SP),HL
000015  311                     RET

000020                  .LOC    ROM0+20

000020  345             R20:    PUSH    HL
000021  052 144 105             LOAD    HL,RST20
000024  343                     EX      (SP),HL
000025  311                     RET

000030                  .LOC    ROM0+30

000030  345             R30:    PUSH    HL
000031  052 114 105             LOAD    HL,RST30
000034  343                     EX      (SP),HL
000035  311                     RET

000040                  .LOC    ROM0+40

000040  345             R40:    PUSH    HL
000041  052 134 105             LOAD    HL,RST40
000044  343                     EX      (SP),HL
000045  311                     RET

000050                  .LOC    ROM0+50

000050  345             R50:    PUSH    HL
000051  052 150 105             LOAD    HL,RST50
000054  343                     EX      (SP),HL
000055  311                     RET

000060                  .LOC    ROM0+60

000060  345             R60:    PUSH    HL
000061  052 152 105             LOAD    HL,RST60
000064  343                     EX      (SP),HL
000065  311                     RET

000070                  .LOC    ROM0+70

000070  345             R70:    PUSH    HL
000071  052 146 105             LOAD    HL,RST70
000074  343                     EX      (SP),HL
000075  311                     RET

000076  365             INTER:  PUSH    AF
000077  305                     PUSH    BC
000100  325                     PUSH    DE
000101  345                     PUSH    HL
000102  333 001                 LOAD    A,$I50
000104  346 010                 AND     A,#FL50
000106  050 071                 JUMP,,EQ INTES  ;erreur ou inter utilis
000110  323 001                 LOAD    $RZ50,A
000112  315 116 001             CALL    ECLAV
000115  315 233 001             CALL    RCLOVER
000120  052 156 105             LOAD    HL,TIMER
000123  174                     LOAD    A,H
000124  265                     OR      A,L
000125  050 010                 JUMP,,EQ INTE2
000127  053                     DEC     HL
000130  042 156 105             LOAD    TIMER,HL
000133  174                     LOAD    A,H
000134  265                     OR      A,L
000135  050 015                 JUMP,,EQ INTE1
000137  052 162 105     INTE2:  LOAD    HL,SUBINT
000142  174                     LOAD    A,H
000143  265                     OR      A,L
000144  030 016                 JUMP,   RELAI


000146                  .LOC    ROM0+146

000146  345             NMI     PUSH    HL
000147  052 154 105             LOAD    HL,ADNMI
000152  343                     EX      (SP),HL
000153  311                     RET

000154  052 160 105     INTE1:  LOAD    HL,TIMAD
000157  021 137 000             LOAD    DE,#INTE2
000162  325                     PUSH    DE
000163  351                     JUMP    (HL)            .JUMP TO USER END TIMER PROGRAM

000164  050 005         RELAI.  JUMP,,EQ INTE3          .IF USER SUBROUTINE ?
000166  021 173 000     INTE21: LOAD    DE,#INTE3
000171  325                     PUSH    DE
000172  351                     JUMP    (HL)            .YES, GO TO AND RETURN TO NEXT IN

000173  341             INTE3   POP     HL
000174  321                     POP     DE
000175  301                     POP     BC
```

```
000176  361                         POP     AF
000177  373                         ION
000200  311                         RET

                            ;Optional user interupt
000201  052 136 105         INTES:  LOAD    HL,INTUS
000204  174                         LOAD    A,H
000205  265                         OR      A,L
000206  040 356                     JUMP,,NE INTE21

000210  347 136             IERROR: .W      ?TEXTIM
000212  111 116 124 000             .ASCIZ  "INT"
000216  341                         POP     HL
000217  321                         POP     DE
000220  301                         POP     BC
000221  361                         POP     AF
000222  303 137 012                 JUMP    TRA


                            ;---    Initialisation du systeme

                            ;in     -
                            ;out    HL  dernier byte de RAM
                            ;       B:  nombre de Kbytes de RAM
                            ;mod    A,F,BC,DE,HL
                            ;Initialise les adresses indirectes

000225  041 112 105         INISYS: LOAD    HL,#SAVMEM
000230  006 260                     LOAD    B,#ODSAVE+400-SYSTSAV
000232  257                         XOR     A,A
000233  167                 RS2:    LOAD    (HL),A
000234  043                         INC     HL
000235  020 374                     DECJ,NE B,RS2

                            ;Point d'entree pour moniteur
                            ;in     -
                            ;out    HL  dernier byte de RAM
                            ;       B:  nombre de Kbytes de RAM
                            ;mod    A,F,BC,DE,HL

                            ;INITIALISE LES TABLES POUR LE CLAVIER BALAYE.
                            ;(NEUTA, OLDTA ET DEBOCLA)

000237  076 200             INISY2: LOAD    A,#200
000241  062 212 105                 LOAD    NEUTA+10.,A
000244  062 225 105                 LOAD    OLDTA+10.,A
000247  062 223 105                 LOAD    DEBOCLA+84./2,A
000252  062 377 105                 LOAD    GWORD+23.,A
000255  041 221 105                 LOAD    HL,#DEBOCLA
000260  042 174 105                 LOAD    POICLA,HL
000263  020 376             ATTU:   DECJ,NE B,.        ;Attente touche relachee
000265  333 020                     LOAD    A,SCLA
000267  346 200                     AND     A,#FULC
000271  040 370                     JUMP,NE ATTU

                            ;INITIALISE LES DEUX USARTS

000273  016 005                     LOAD    C,#SUSA1
000275  315 152 005                 CALL    INUSA
000300  016 007                     LOAD    C,#SUSA2
000302  315 152 005                 CALL    INUSA

                            ;INITIALISE LES ROUTINES GRAPHIQUES.
000305  076 311                     LOAD    A,#311          ;CODE RETURN
000307  062 271 105                 LOAD    SIMUL0+1,A
000312  062 274 105                 LOAD    SIMUL1+2,A

                            ;in     -
                            ;out    HL  dernier byte de RAM
                            ;       B   nombre de Kbytes de RAM
                            ;mod    A,F,BC,DE,HL

000315  041 076 020         INITRA: LOAD    HL,#INTER
000320  042 146 105                 LOAD    RST70,HL
000323  041 065 001                 LOAD    HL,#SYSTEM
000326  042 134 105                 LOAD    RST40,HL

                            ;PERMET L'INTERRUPT 50HZ

000331  076 104                     LOAD    A,#104
000333  347 076                     .W      ?BEEP

000335  016 024                     LOAD    C,#LINES
000337  347 126                     .W      ?IDIS

000341  076 001                     LOAD    A,#ENINT50
000343  323 000                     LOAD    SMODI,A
000345  062 177 105                 LOAD    LAMODI,A

                            ;Teste la dim de la memoire

000350  021 000 100         TSTMEM: LOAD    DE,#40000
000353  257                         XOR     A,A
000354  157                         LOAD    L,A
000355  147                         LOAD    H,A
000356  006 004                     LOAD    B,#4
000360  031                 TSTME1: ADD     HL,DE
000361  116                         LOAD    C,(HL)
000362  167                         LOAD    (HL),A
000363  276                         COMP    A,(HL)
000364  161                         LOAD    (HL),C
000365  040 004                     JUMP,,NE TSTME2
000367  306 020                     ADD     A,#16.
000371  020 365                     DECJ,NE B,TSTME1
000373  345                 TSTME2: PUSH    HL
000374  046 000                     LOAD    H,#0
000376  157                         LOAD    L,A
000377  347 124                     .W      ?BINBCD
000401  105                         LOAD    B,L
000402  341                         POP     HL
000403  053                         DEC     HL
000404  311                         RET


                            ;++++++++++     ENTREE APRES RESET
                            RESET:
                            ;IOF et LOAD    SP,#SPUTIL avant

000405  315 225 000                 CALL    INISYS
000410  170                         LOAD    A,B
000411  042 140 105                 LOAD    MAXMEM,HL
000414  371                         LOAD    SP,HL           ;Place stack at end of ram
000415  041 013 016                 LOAD    HL,#TERESET
000420  347 006 347 071             .W      ?DITEX,?AFHEX,?DITEX

000425  041 371 007                 LOAD    HL,#TBOOT
000431  347 031                     .W      ?SEARCH
000433  050 006                     JUMP,,CC RES3

                            ;Point d'entree RES4

000435  347 005             RES4:   .W      ?GETLINE
000437  347 034                     .W      ?BRANCH
000441  070 002                     JUMP,,CS RESG   ;not found in main memory
000443  325                 RES3:   PUSH    DE
000444  311                         RET

000445  347 136             RESG:   .W      ?TEXTIM
000447  040 077 015 000 TERROR:     .ASCIZ  " ?<CR>"
000453  030 360                     JUMP.   RES4

                            ; ENTREE DANS LE SYSTEME
                            ; ********************
                            ;
```

```
                            ;ON EST INTERRUPT ON APRES UN APPEL SYSTEM.

000455  363                 SYSTEM: IOF
000456  062 124 105                 LOAD    SAVAG,A
000461  [illegible]                 [illegible]
000462  [illegible]                 LOAD    A,(HL)
000463  043                         INC     HL
000464  343                         EX      (SP),HL
000465  [illegible]                 PUSH    HL
000466  [illegible]                 PUSH    AF
000467  313 047                     SLA     A               ;*2
000471  052 132 105                 LOAD    HL,TABEX        ;!!Dans page unique
000474  070 003                     JUMP,,CS SYSTM4         ;Saut si appel >200
000476  041 030 017                 LOAD    HL,#TAROUT
000501  205                 SYSTM4: ADD     A,L
000502  157                         LOAD    L,A
000503  176                         LOAD    A,(HL)
000504  043                         INC     HL
000505  146                         LOAD    H,(HL)
000506  157                         LOAD    L,A             ;HL = ADR OF ROUTINE
000507  361                         POP     AF              ;RESTORE USER'S F
000510  072 124 105                 LOAD    A,SAVAG
000513  343                         EX      (SP),HL         ;ROUT ADR ON STACK
000514  373                         ION
000515  311                         RET                     ;GO TO SELECTED ROUTINE

                            ;THE EXIT OF SYSTEM IS MADE DIRECTLY BY THE ROUTINES
                            ;THE ENTRY PROGRAM DO NOT DESTROY ANY REGISTER OR STACK.
                            ;THE CALLED ROUTINE CAN RETURN TO THE USER'S PROGRAM BY
                            ;USING THE "RET" INSTRUCTION.


                            .PAGE1


                            ;--- BCLAV ---

                            ;CETTE ROUTINE BALAYE LE CLAVIER ET MET DANS UNE
                            ;TABLE ("NEUTA") LE CODE DE TOUTES LES TOUCHES PRESSEES.
                            ;L'ORDRE DANS LEQUEL LES TOUCHES SONT STOKEES EST QUELCONQUE
                            ;ET PEUT CHANGER D'UNE FOIS A L'AUTRE.
                            ;LE CODE DE LA TOUCHE FONCTION PRESSEE EST MIS DANS "FONCT".
                            ;LE TEMPS D'EXECUTION DE CETTE ROUTINE EST TRES COURT SI AUC
                            ;TOUCHE N'EST PRESSEE. PAR CONTRE, IL PEUT DEVENIR IMPORTANT
                            ;SI PLUSIEURS TOUCHES SONT APPUYEES SIMULTANEMENT.

                            ;! detruit B

000516  041 277 105         BCLAV:  LOAD    HL,#FLOFF
000521  176                         LOAD    A,(HL)
000522  267                         OR      A,A
000523  050 006                     JUMP,,EQ BCLAV3
000525  075                         DEC     A
000526  040 002                     JUMP,,NE BCLAV9
000530  323 031                     LOAD    S31,A
000532  167                 BCLAV9: LOAD    (HL),A
000533  041 200 105         BCLAV8: LOAD    HL,#NEUTA       ;VIDE NEUTA
000536  066 000                     LOAD    (HL),#0
000540  333 000                     LOAD    A,SCLA
000542  313 177                     TEST    A,7             ;TEST LE FLAG FOUND
000544  040 010                     JUMP,,NE BCLAV2         ;TOUCHE FONCTION ?
000546  062 176 105                 LOAD    FONCT,A         ;Ul:le???
000551  257                         XOR     A,A             ;Init delay for repeat
000552  062 130 105                 LOAD    DREP,A
000555  311                         RET

000556  346 177             BCLAV2: AND     A,#MCLA         ;STOCKE LA TOUCHE DANS NEUTA
000560  167                 BCLAV3: LOAD    (HL),A
000561  043                         INC     HL
000562  066 000                     LOAD    (HL),#0
000564  043                         INC     HL              ;EST-CE QU'IL Y A TROP DE
000565  176                         LOAD    A,(HL)          ;TOUCHES ENFONCEES SIMULTANE
000566  376 200                     COMP    A,#ENDO
000570  310                         RET,EQ
000571  053                         DEC     HL
000572  006 000             BCLAV4: LOAD    B,#0            ;ATTENTE DE 200 US
000574  020 376                     DECJ,NE B,.
000576  333 001                     LOAD    A,SSCLA         ;UNE SEULE TOUCHE ?
000600  346 004                     AND     A,#FULCLA
000602  310                         RET,EQ
000603  333 000             BCLAV7: LOAD    A,SCLA
000605  346 177                     AND     A,#MCLA
000607  107                         LOAD    B,A
000610  345                         PUSH    HL
000611  041 177 105                 LOAD    HL,#NEUTA-1
000614  043                 BCLAV5: INC     HL              ;EST-CE QUE CETTE TOUCHE A
000615  176                         LOAD    A,(HL)          ;DEJA ETE LUE ?
000616  267                         OR      A,A
000617  050 005                     JUMP,,EQ BCLAV6
000621  270                         COMP    A,B
000622  040 370                     JUMP,,NE BCLAV5
000624  341                         POP     HL
000625  311                         RET

000626  063                 BCLAV6: INC     SP              ;SIMULE UN "POP" QUELCONQUE
000627  063                         INC     SP
000630  170                         LOAD    A,B
000631  030 225                     JUMP.   BCLAV3


                            ;ROUTINE SIMULANT LE n KEY ROLL-OVER

                            ;SI UNE NOUVELLE TOUCHE EST PRESSEE, INSERE SON
                            ;CODE DANS OLDTA ET MET LE CODE DE LA TOUCHE DANS
                            ;LA QUEUE D'ATTENTE DU CLAVIER.
                            ;SI UNE TOUCHE N'EST PLUS PRESSEE, SUPPRIME SON
                            ;CODE DANS OLDTA.
                            ;LE CODE DE TOUTES LES TOUCHES PRESSEES EST
                            ;DANS NEUTA.
                            ; "DE"  POINTE DANS NEUTA
                            ; "HL"  POINTE DANS OLDTA
                            ;
                            ; EXEMPLE
                            ; *******
                            ;
                            ; A V A N T
                            ;
                            ;NEUTA: 'A              OLDTA:  'A
                            ;       'C                      'B
                            ;       'E                      'C
                            ;       'G                      'G
                            ;                               'H
                            ;
                            ;
                            ; A P R E S
                            ;
                            ;NEUTA: 'A              OLDTA:  'A
                            ;       'C                      'C
                            ;       'E                      'E
                            ;       'G                      'G
                            ;
                            ;LE CODE "E" A ETE PLACE DANS LA QUEUE DU CLAVIER.
                            ;


000633  021 200 105         ROLOVER:LOAD    DE,#NEUTA
000636  041 212 105         ROLOV0: LOAD    HL,#OLDTA-1
000641  043                 ROLOV1: INC     HL
000642  043                         INC     HL
000643  176                         LOAD    A,(HL)
000644  376 200                     COMP    A,#ENDO         ;TROP DE TOUCHES ?
000646  050 007                     JUMP,,EQ REPEAT
000650  053                         DEC     HL
000651  032                 ROLOV2: LOAD    A,(DE)          ;IF END OF NEUTA?
000652  267                         OR      A,A
```

```
000662  050 024                JUMP..EQ ROLOV6
000664  276                    COMP    A.(HL)            ;IF (NEWTA)=(OLDTA)?
000665  040 005                JUMP..NE ROLOV4
000667  313 376       ROLOV3   SET     (HL).7
000671  023                    INC     DE
000672  030 351                JUMP.   ROLOV6
000674  176           ROLOV4   LOAD    A.(HL)            ,IF END OF OLDTA?
000675  267                    OR      A.A
000676  040 350                JUMP..NE ROLOV4
000670  032           ROLOV5:  LOAD    A.(DE)            .LOAD (OLDTA).(NEWTA)
000672  167                    LOAD    (HL).A
000673  315 363 001            CALL    PUOCLA
000676  076 043                LOAD    A.#DELAY1         ;Init delay for repeat
000700  062 130 105            LOAD    DREP.A
000703  043                    INC     HL
000704  066 000                LOAD    (HL).#0
000706  053                    DEC     HL
000707  030 347                JUMP.   ROLOV3

                      ;SUPPRIME DANS OLDTA TOUS LES CODES DONT LE BIT
                      , 7 EST A ZERO. PUIS CLEAR TOUS LES BITS 7 DE.
                      .OLDTA.

000711  041 212 105   ROLOV6   LOAD    HL.#OLDTA-1
000714  124                    LOAD    D.H               ;LOAD DE.HL
000715  135                    LOAD    E.L
000716  023           ROLOV7   INC     DE
000717  043           ROLOV8-  INC     HL
000720  176                    LOAD    A.(HL)            ;IF END OF OLDTA?
000721  267                    OR      A.A
000722  022                    LOAD    (DE).A
000723  050 012                JUMP..EQ REPEAT
000725  313 176                TEST    (HL).7
000727  313 276                CLR     (HL).7
000731  176                    LOAD    A.(HL)
000732  022                    LOAD    (DE).A
000733  040 361                JUMP..NE ROLOV7
000735  030 360                JUMP.   ROLOV8

000737  052 174 105   REPEAT:  LOAD    HL.POICLA
000742  053                    DEC     HL
000743  313 176                TEST    (HL).7
000745  310                    RET.EQ                    ;RET if newta full
000746  041 130 105            LOAD    HL.#DREP
000751  176                    LOAD    A.(HL)
000752  267                    OR      A.A
000753  310                    RET.EQ
000754  065                    DEC     (HL)
000755  300                    RET.NE
000756  066 003                LOAD    (HL).#DELAY2
000760  072 167 105            LOAD    A.TREP

000763  345           PUOCLA:  PUSH    HL
000764  062 167 105            LOAD    TREP.A
000767  107                    LOAD    B.A
000770  052 174 105            LOAD    HL.POICLA
000773  176                    LOAD    A.(HL)
000774  376 200                COMP    A.#ENDQ
000776  050 005                JUMP..EQ PUOCL2
001000  160                    LOAD    (HL).B
001001  043                    INC     HL
001002  042 174 105            LOAD    POICLA.HL
001005  341           PUOCL2:  POP     HL
001006  311                    RET

                      ;;Appels d'extension du repertoire d'instruction
                      ;;-----------------------------------------------
                      ;;

                      ;??????????
                      ;;57 ?MUL          HL+DExBC => HLDE
                      ;??????????
                      ;;
                      ;;in    DE multiplicande, BC multiplicateur, HL valeur add
                      ;;out   HLDE produit 32 bits, A=0
                      ;;mod   F, A, HL, DE
                      ;;
001007  076 020       MUL.     LOAD    A.#16.
001011  267           MUL2.    OR      A.A       ;CLRCarry
001012  313 103                TEST    E:0
001014  050 001                JUMP..EQ MUL4
001016  011                    ADD     HL.BC
001017  313 034       MUL4:    RRC     H
001021  313 035                RRC     L
001023  313 032                RRC     D
001025  313 033                RRC     E
001027  075                    DEC     A
001030  040 357                JUMP.NE MUL2
001032  311                    RET

                      ,??????????
                      ,;60 ?DIV          HLDE : BC => DE. reste HL
                      ;??????????
                      ;;
                      ;;in    HLDE dividende. BC diviseur<>0
                      ;;out   DE quotient. HL reste. CS si dep avec A=nbre cycles resta
                      ;;mod   F. A. HL. DE
                      ;;
001033  076 020       DIV:     LOAD    A.#16.
001035  067           DIV2:    SETC
001036  313 023                RLC     E
001040  313 022                RLC     D
001042  313 025                RLC     L
001044  313 024                RLC     H
001046  330                    RET.CS
001047  355 102                SUBC    HL.BC
001051  050 003                JUMP..HS DIV4
001053  011                    ADD     HL.BC
001054  313 203                CLR     E:0       ;Bit 0 si retablissement
001056  075           DIV4:    DEC     A
001057  040 354                JUMP.NE DIV2
001061  267                    OR      A.A
001062  311                    RET

                      ;;
                      ,??????????????
                      ,;117 ?GETARG   LOAD    DE.8(SP)+ Lecture d'un argument suivant l
                      ,??????????????

                      ;;in    SP pointe l'adresse de retour
                      ,;out   DE parametre. SP pointe la nouv adresse de retour
                      ,;mod   DE
                      ;;
001063  363           GETARG:  IOF
001064  063                    INC     SP
001065  063                    INC     SP
001066  343                    EX      (SP).HL
001067  136                    LOAD    E.(HL)
001070  043                    INC     HL
001071  126                    LOAD    D.(HL)
001072  043                    INC     HL
001073  343                    EX      (SP).HL
001074  073                    DEC     SP
001075  073                    DEC     SP
001076  373                    ION
001077  311                    RET

                      ;.
                      ;.Appels de recherche et débranchement
                      ;;-----------------------------------
                      ;;
                      ;;
                      ;??????????
                      ,.62 ?SCAR        Chercho un caractere ASCII
                      ;??????????
                      ;;
                      ;;in    A caractere
                      ;;      DE adresse table  (.DW  car_ascii,adresse_appel)
                      ;;              .B 0 car fin table
                      ;;out   CC si trouvé, ret  avec DE=adresse du mot suivant
                      ;;      CS si non trouvé  retour avec DE=adresse suiv la table
                      ;;mod   F, DE
                      ;;
001100  305           SCAR     PUSH    BC
001101  353                    EX      HL.DE
001102  106           CAL2     LOAD    B.(HL)
001103  043                    INC     HL
001104  004                    INC     B
001105  005                    DEC     B
001106  050 007                JUMP..EQ CAL4
001110  270                    COMP    A.B
001111  050 005                JUMP..EQ CAL6
001113  043                    INC     HL
001114  043                    INC     HL
001115  030 363                JUMP.   CAL2

001117  067           CAL4.    SETC
001120  353           CAL6:    EX      HL.DE
001121  301                    POP     BC
001122  311                    RET

                      ;;
                      ,??????????
                      ,.61 ?JUMPCAR   Jump selon un caractère ASCII
                      ;??????????
                      ;;
                      ;;in    A caractere
                      ;;      DE adresse table  (.DW  car_ascii,adresse_saut)
                      ;;              .B 0 car fin table
                      ,;out   -trouvé: saut avec pile corrigée
                      ;,      -non trouvé: retour avec DE=adresse suiv la table
                      ,,mod   DE
                      ;;
001123  365           JUMPCAR:PUSH    AF
001124  347 062                .U      ?SCAR
001125  070 004                JUMP..CS JUM2
001130  361                    POP     AF
001131  303 324 005            JUMP    JUMPIDE

001134  361           JUM2     POP     AF
001135  311                    RET

                      ;;
                      ,?????????
                      ,.34 ?BRANCH    Cherche un nom en memoire
                      ,?????????
                      ;;
                      ;;in    HL pointeur mot donne
                      ,.out   CC si trouve
                      ;;       DE adresse debut en RAM, HL pointe la fin du mot do
                      ;,      CS si non trouve
                      ,;       HL pointe le debut du mot donne
                      ;,mod   F,A,BC,DE,HL
                      ;;
                      ;,Exemple de debut en RAM
                      ;;      .ASCII  "<252>mot_clef<200>"
                      ;;      prem instruction executee
                      ;;
                      ;;
            000252 BWORD   =252
            000200 EWORD   =200

001136  345           BRANCH:  PUSH    HL               .pointeur au mot_clef
                      ;Recherche dans la mem princ
001137  041 000 000            LOAD    HL.#ROM0
001142  001 000 000            LOAD    BC.#0   ;64K

001145  321           BRA2.    POP     DE       ;recupere le pointeur au mot_clef
001146  325                    PUSH    DE
001147  076 252       BRA4     LOAD    A.#BWORD
001151  355 261                CPIR
001153  342 177 002            JUMP.VC BRA10   ;not found
001156  032           BRA6.    LOAD    A.(DE)
001157  023                    INC     DE
001160  276                    COMP    A.(HL)
001161  043                    INC     HL
                      ;S!! ;Pas de test sur BC. mots cle courts
001162  050 372                JUMP..EQ BRA6
001164  053                    DEC     HL
001165  176                    LOAD    A.(HL)
001166  376 200                COMP    A.#EWORD
001170  040 353                JUMP..NE BRA2   ;not good
                      ;Carry clear car comparaison EQ
001172  043                    INC     HL      .retour avec DE=derniere adresse
001173  353           BRA8     EX      HL.DE   ; et.HL pointant sur le mot donne
001174  063                    INC     SP
001175  063                    INC     SP      .corrige la pile
001176  311                    RET

001177  067           BRA10:   SETC
001200  030 371                JUMP.   BRA8

                      ;;
                      ;;Appels d'initialisation de positions
                      ;;-----------------------------------
                      ;;

                      ;??????????
                      ;.33 ?MEM                        Determine la fin de la memoi
                      ;??????????
                      ;;
                      ;;in    -
                      ;;out   HL adresse de fin
                      ;;mod   HL
                      ;;
001202  052 140 105   MEM.     LOAD    HL.MAXMEM
001205  311                    RET

                      ,?????????????
                      ,;2  ?ITIMER
                      ,;3  ?ITIMAD
                      ;;4  ?IRTC
                      ,;37 ?INTUS       Init adresses indirectes
                      ,;7  ?IRST10
                      ,;10 ?IRST20
                      ,;12 ?IRST60
                      ,;13 ?IRST60
                      ,;11 ?IRST70
                      ,;14 ?IADNMI
                      ,?????????????
                      ;;
                      ;;in    HL adresses
                      ;;out   -
                      ;;mod   -
                      ;;
001206  042 153 105   ITIMER.  LOAD    TIMER.HL
001211  311                    RET

001212  042 160 105   ITIMAD   LOAD    TIMAD.HL
001215  311                    RET

001216  042 162 105   IRTC.    LOAD    OUDINT.HL
001221  311                    RET

001222  042 136 105   IUSINT.  LOAD    INTUS.HL
001225  311                    RET

001226  042 142 105   IRST10:  LOAD    RST10.HL
001231  311                    RET

001232  042 144 105   IRST20:  LOAD    RST20.HL
001235  311                    RET

001236  042 140 105   IRST70:  LOAD    RST70.HL
001241  311                    RET
```

```
001242  042 150 105           IRST50: LOAD    RST50,HL
001245  311                           RET

001246  042 152 105   IRSTE0  LOAD  · RSTE0,HL
001251  311                   RET

001252  042 154 105   IRINMI. LOAD    RINMI,HL
001255  311                   RET

                      ;????????
                      ;.17 ?ENI50      Initialisation interrupt 50 hz
                      ;????????
                      ;;
                      ;;in     A =0 plus d'interruptions 50 Hz
                      ;;cut    -
                      ;;mod    -
                      ;;
001256  365           ENI50.  PUSH    AF
001257  267                   OR      A,A
001260  072 177 105           LOAD    A,LAMODI
001263  313 207               CLR     A:0             ;DESABLE INT50
001265  050 002               JUMP.,EQ ENI51
001267  313 307               SET     A:0             ;ENABLE INT50
001271  062 177 105   ENI51.  LOAD    LAMODI,A
001274  323 000               LOAD    $MODI,A
001276  361                   POP     AF
001277  311                   RET

                      ;??????????
                      ;.30 ?CALPHA     Efface l'ecran alpha
                      ;??????????
                      ;;
                      ;;in     -
                      ;;cut    HL=SALPHA, A=0, CS
                      ;;mod    F, A, HL
                      ;;
001300  041 377 104   CALPHA  LOAD    HL,#ENDALPHA-1
001303  066 040       CALP2   LOAD    (HL),#SPA
001305  053                   DEC     HL
001306  174                   LOAD    A,H
001307  376 100               COMP    A,#SALPHA/400
001311  060 370               JUMP.,HS CALP2
001313  043                   INC     HL
001314  076 000               LOAD    A,#0
001316  311                   RET

                      ;????????????
                      ;.31 ?CGRA       Efface l'ecran graphique
                      ;????????????
                      ;;
                      ;;in     -
                      ;;out    A=0, CC
                      ;;mod    F, A
                      ;;
001317  345           CGRA:   PUSH    HL
001320  041 000 106           LOAD    HL,#SGRA
001323  066 000       CGR2.   LOAD    (HL),#0
001325  043                   INC     HL
001326  174                   LOAD    A,H
001327  376 126               COMP    A,#ENDGRA/400
001331  040 370               JUMP.,NE CGR2
001333  341                   POP     HL
001334  257                   XOR     A,A
001335  311                   RET

                      ;????????
                      ;.21 ?IALPHA     Mode alpha seul
                      ;????????
                      ;;
                      ;;in     -
                      ;;cut    -
                      ;;mod    -
                      ;;
001336  365           IALPHA: PUSH    AF
001337  072 177 105           LOAD    A,LAMODI
001342  346 363               AND     A,#377-NOALPHA-GRA
001344  030 323               JUMP.   ENI51

                      ;????????
                      ;;22 ?IGRA       Mode graphique seul
                      ;;23 ?IAGRA      Mode alpha et gra superpose
                      ;????????
                      ;;
                      ;;in     CS mode petits points
                      ;;cut    -
                      ;;mod    -
                      ;;
001346  365           IGRA:   PUSH    AF
001347  072 177 105           LOAD    A,LAMODI
001352  313 337               SET     A:3
001354  030 006               JUMP.   IGR1

001356  365           IAGRA:  PUSH    AF
001357  072 177 105           LOAD    A,LAMODI
001362  313 237               CLR     A:3
001364  313 327       IGR1:   SET     A:2
001366  313 317               SET     A:1
001370  070 277               JUMP.,CS ENI51
001372  313 217               CLR     A:1
001374  030 273               JUMP.   ENI51

                      ;;
                      ;;Appels d'initialisation et conversion de pointeurs
                      ;;--------------------------------------------------
                      ;;

                      ;????????
                      ;.20 ?IDICAR     Initialise la longueur et le debut du display
                      ;????????
                      ;;
                      ;;in     B numero 1e ligne  C nombre de lignes #pos*400+nbreli
                      ;;                 C=0 et B+C>19, non acceptes
                      ;;out    -
                      ;;mod    -
                      ;;
001376  365           IDICAR: PUSH    AF
001377  345                   PUSH    HL
001400  305                   PUSH    BC
001401  325                   PUSH    DE
001402  171                   LOAD    A,C
001403  267                   OR      A,A
001404  050 037               JUMP.,EQ IDIC4
001406  200                   ADD     A,B
001407  376 025               COMP    A,#LINES+1
001411  060 032               JUMP.,HS IDIC4
001413  004                   INC     B
001414  041 000 100           LOAD    HL,#SALPHA
001417  021 100 000           LOAD    DE,#NCAR
001422  031           IDIC2:  ADD     HL,DE
001423  020 375               DECJ.NE B,IDIC2
001425  267                   OR      A,A
001426  355 122               SUBC    HL,DE
001430  042 170 105           LOAD    FIRSTL,HL
001433  042 104 105           LOAD    POINT,HL
001436  031           IDIC3   ADD     HL,DE
001437  015                   DEC     C
001440  040 374               JUMP.,NE IDIC3
001442  042 172 105           LOAD    LAGTL,HL
001445  321           IDIC4.  POP     DE
001446  301                   POP     BC
001447  341           IDIC5:  POP     HL
001450  361                   POP     AF
001451  311                   RET

                      ;??????????
```

```
                      ;;120 ?IDIS      Init display
                      ;???????????
                      ;;
                      ;;in     C number of lines
                      ;;out    (curseur)
                      ;;mod    B
001452  365           IDIS    PUSH    AF
001453  076 137               LOAD    A,#CPOINT
001455  062 103 105           LOAD    CPOINTEUR,A
001460  006 000               LOAD    B,#0
001462  347 020 347 122       .W      ?IDICAR,?CLEAR,?IALPHA
001470  361                   POP     AF
001471  311                   RET

                      ;??????????
                      ;.54 ?IPOINTER
                      ;??????????
                      ;;
                      ;;in     A caractere pointeur
                      ;;out    -
                      ;;mod    -
001472  062 103 105   IPOINTER LOAD   CPOINTEUR,A
                      ;-               PUSH    HL
                      ;-               LOAD    HL,POINT
                      ;-               LOAD    (HL),A
                      ;-               POP     HL
001475  311                   RET

                      ;?????????
                      ;.40 ?SETCURSOR Repositionne le pointeur
                      ;?????????
                      ;;
                      ;;in     HL coord
                      ;;out    pointeur
                      ;;mod    pointeur
001476  365           SETCURSOR PUSH  AF
001477  345                   PUSH    HL
001500  313 045               SLC     L
001502  313 045               SLC     L
001504  174                   LOAD    A,H
001505  376 024       TRAN2   COMP    A,#LINES
001507  070 004               JUMP.,LO TRAN4
001511  326 024               SUB     A,#LINES
001513  030 370               JUMP.   TRAN2
001515  313 077       TRAN4:  SRC     A
001517  313 035               RRC     L
001521  313 077               SRC     A
001523  313 035               RRC     L
001525  306 100               ADD     A,#SALPHA/400
001527  147                   LOAD    H,A
001530  042 104 105           LOAD    POINT,HL
001533  030 312               JUMP.   IDIC5

                      ;;
                      ;;Appel de bruitage
                      ;;-----------------
                      ;;
                      ;??????????
                      ;;32 ?BUZZ
                      ;??????????
                      ;;
                      ;---     Buzze a frequence fixe
                      ;;in     -
                      ;;out    -
                      ;;mod    -
                000203 FRQ     = 203

001535  365           BUZZ:   PUSH    AF
001536  076 203               LOAD    A,#FRQ
001540  347 076               .W      ?BEEP
001542  361                   POP     AF
001543  311                   RET

                      ;?????????????
                      ;;76 ?BEEP       Note approximative
                      ;?????????????
                      ;;
                      ;;in     A duree (A:0-2) et période (A:3-7)
                      ;;       Si A=377 --> silence
                      ;;out    -
                      ;;mod    -
                      ;;
001544  363           BEEP:   IOF
001545  305                   PUSH    BC
001546  365           BE2     PUSH    AF      ;A = periode
001547  346 007               AND     A,#7
001551  107                   LOAD    B,A     ;B = duree

001552  016 040       BE3:    LOAD    C,#40
001554  361           BE4:    POP     AF
001555  365                   PUSH    AF
001556  074                   INC     A       ;A=377 ?
001557  050 002               JUMP.,EQ BE6
001561  323 003               LOAD    $HP,A
001563  075           BE6:    DEC     A
001564  040 375               JUMP.,NE BE6
001566  015                   DEC     C
001567  040 363               JUMP.,NE BE4
001571  020 357               DECJ.NE B,BE3
001573  361                   POP     AF
001574  301                   POP     BC
001575  373                   ION
001576  311                   RET

                      ;???????????
                      ;.75 ?PLAY       Joue un morceau (notes approximatives)
                      ;???????????
                      ;;
                      ;;in     HL pointeur au debut du morceau (terminé par un 0)
                      ;;out    HL pointe morceau suivant éventuel
                      ;;mod    F, A, HL
                      ;;
                      PLAY1.
001577  347 076               .W      ?BEEP
001601  361                   POP     AF
001602  365           PLAY:   PUSH    AF
001603  176                   LOAD    A,(HL)
001604  043                   INC     HL
001605  267                   OR      A,A
001606  040 367               JUMP.,NE PLAY1
001610  361                   POP     AF
001611  311                   RET

                      ;;
                      ;;Appels d'affichage de caracteres sur l'ecran
                      ;;--------------------------------------------
                      ;;
                      ;???????????
                      ;.172 ?EFEC      Efface l'ecran depuis le pointeur
                      ;???????????
                      ;;
                      ;;in     -
                      ;;out    A=0
                      ;;mod    F A
                      ;;
001612  325           EFEC:   PUSH    DE
001613  345                   PUSH    HL
001614  355 133 104 105       LOAD    DE,POINT
001620  315 232 003           CALL    CLRWI
001623  341                   POP     HL
001624  321                   POP     DE
001625  311                   RET

                      ;----    Efface la fenetre
```

```
                        ;-in    -
                        ;-out   HL=FIRST, A=0
                        ;-mod   F, A, HL, ! DE !
                        ;--
001618  365 135 170 105 CLR:    LOAD    DE,FIRSTL
001622  056 172 105     CLR1:   LOAD    HL,LASTL
001625  257             CLR2:   XOR     A,A
001626  315 273 003             CALL    DECPHL
001631  066 040                 LOAD    (HL),#SPA
001633  310                     RET,EQ
001634  030 370                 JUMP.   CLR2

                        ;---    Incremente HL dans la fenetre
                        ;---    Teste si HL=LASTL ou HL=ENDALPHA
                        ;-in    HL
                        ;-out   HL incremente si permis, HL=LASTL si frontiere
                        ;       EQ si frontiere atteinte
                        ;-mod   HL, ! DE !
                        ;--
001646  043             INCPHL: INC     HL
001647  365 133 172 105 TESTDOWN LOAD   DE,LASTL
001653  315 115 016             CALL    COMPHLDE
001656  310                     RET,EQ
001657  320                     RET,LO
001660  021 000 105             LOAD    DE,#ENDALPHA
001663  315 115 016             CALL    COMPHLDE
001666  330                     RET,LO
001667  052 170 105             LOAD    HL,FIRSTL
001672  311                     RET

                        ;---    Decremente si permis
                        ;---    Teste si HL=DE ou HL=SALPHA
                        ;-in    HL, DE
                        ;-out   HL decr si ok, HL=DE ou HL=SALPHA si frontiere
                        ;--     EQ si frontiere atteinte
                        ;-mod   HL
                        ;--
001673  053             DECPHL: DEC     HL
001674  315 115 016     TESTUP: CALL    COMPHLDE
001677  310                     RET,EQ
001700  325                     PUSH    DE
001701  021 000 100             LOAD    DE,#SALPHA
001704  315 115 016             CALL    COMPHLDE
001707  321                     POP     DE
001710  311                     RET

                        ;---    Effece le precedent car si permis, en remontant
                        ;       tout les espaces d'un coup.
                        ;-in    HL, DE
                        ;-out   HL decr ou =DE ou SALPHA
                        ;-mod   HL,F, A, ! DE !
                        ;--
001711  076 040         GOBACK: LOAD    A,#SPA
001713  315 174 003             CALL    TESTUP
001716  310                     RET,EQ
001717  315 273 003             CALL    DECPHL
001722  310                     RET,EQ
001723  276                     COMP    A,(HL)
001724  300                     RET,NE
001725  315 273 003     GOBAC2: CALL    DECPHL
001730  310                     RET,EQ
001731  276                     COMP    A,(HL)
001732  050 371                 JUMP.,EQ GOBAC2
001734  043                     INC     HL
001735  311                     RET

                        ;---    Caractere BELL
                        ;-in    -
                        ;-out   A=0
                        ;-mod   F, A
                        ;--
001736  315 135 003     DOBEL:  CALL    BUZZ
001741  257             BEL2:   XOR     A,A
001742  311                     RET

                        ;---    Effece le precedent car si permis, en remontant
                        ;       tout les espaces d'un coup.
                        ;-in    HL
                        ;-out   HL decr ou =FIRSTL ou SALPHA, A=0
                        ;-mod   HL, F, A, ! DE !
                        ;--
001743  365 133 170 105 DODEL:  LOAD    DE,FIRSTL
001747  315 311 003             CALL    GOBACK
001752  030 365                 JUMP.   BEL2

                        ;---    Avance le pointeur jusqu'au tab donne
                        ;-in    HL, B module du tab (10, 20, 40 ,100 seulement)
                        ;-out   HL deplace ou =LASTL ou ENDALPHA, A=0
                        ;-mod   HL, F, A, ! DE !
                        ;--
001754  025             DOTAB:  DEC     B
001755  315 246 003     DOT2:   CALL    INCPHL
001760  050 034                 JUMP.,EQ DOT3   ;bas ecran
001762  175                     LOAD    A,L
001763  240                     AND     A,B
001764  040 367                 JUMP.,NE DOT2
001766  004             DOT3:   INC     B
001767  030 350                 JUMP.   BEL2

                        ;---    Decalage de l'ecran vers le haut
                        ;-      Ravi testera le FIFO clavier
                        ;-in    -
                        ;-out   HL=debut_derniere_ligne
                        ;-mod   HL
                        ;--
001771  325             ROLL:   PUSH    AF
001772  325                     PUSH    BC
001773  325                     PUSH    DE
001774  052 172 105             LOAD    HL,LASTL
001777  345                     PUSH    HL
002000  021 100 000             LOAD    DE,#NCAR
002003  267                     OR      A,A
002004  355 122                 SUBC    HL,DE
002006  355 133 170 105         LOAD    DE,FIRSTL
002012  267                     OR      A,A
002013  355 122                 SUBC    HL,DE           ;HL <- LASTL-FIRSTL
002015  050 011                 JUMP.,EQ ROL3
002017  104                     LOAD    B,H             ;LOAD   BC,HL
002020  115                     LOAD    C,L
002021  041 100 000             LOAD    HL,#NCAR
002024  031                     ADD     HL,DE           ;HL <- FIRSTL+NCAR
002025  355 260                 LDIR
002027  353                     EX      HL,DE
002030  046 100         ROL3:   LOAD    B,#NCAR
002032  341                     POP     HL
002033  053             ROL4:   DEC     HL
002034  066 040                 LOAD    (HL),#SPA       ;Effece der ligne
002036  020 373                 DEC.,NE B,ROL4
002040  301             ROL0:   POP     BC
002041  301                     POP     BC
002042  361                     POP     AF
002043  311                     RET

                        ;??????????
                        ;.0 ?DICAR      Affiche un car dans la fenetre
                        ;               ou appelle une routine d'impr. COUTCAR
                        ;??????????
                        ;;
                        ;;in    pointeur, A caractere, OUTCAR
                        ;;out   pointeur
                        ;;mod   pointeur
                        ;;
002044  365             DICAR   PUSH    AF
002045  305                     PUSH    BC
002046  325                     PUSH    DE
002047  345                     PUSH    HL

002050  072 120 105             LOAD    HL,OUTCAR       ;Indirection pour imprimante
002053  107                     LOAD    B,A
002054  175                     LOAD    A,L

2.16-6
```

```
002055  004                     OR      A,H
002056  050                     LOAD    A,B
002057  030 005                 JUMP.,EQ DIC1
002061  043 002 004             LOAD    IX,#DIC3
002064  043                     PUSH    IX              ;For return
002065  351                     JUMP    (HL)

002066  052 164 105     DIC1:   LOAD    HL,POINT        ;Clear pointer (later stop blin?
002071  066 040                 LOAD    (HL),#SPA
002073  176 007                 CMP     A,#BELL
002075  314 235 003             CALL,EQ DOBEL
002100  376 002                 CMP     A,#CLEARW
002102  314 226 003             CALL,EQ CLRW
                        ;-              CMP     A,#CLEARG
                        ;-              CALL,EQ CGRA
002105  345                     PUSH    HL
002106  041 112 105             LOAD    HL,#CRMHEX
002111  376 006                 CMP     A,#INV
002113  314 272 013             CALL,EQ MODINV
002116  376 004                 CMP     A,#UNDM
002120  314 226 013             CALL,EQ MODUNR
002123  341                     POP     HL
002124  376 010                 CMP     A,#BS
002126  314 343 003             CALL,EQ DOBEL
002131  006 010                 LOAD    B,#10
002133  376 011                 CMP     A,#TAB
002135  314 354 003             CALL,EQ DOTAB
002140  006 100                 LOAD    B,#NCAR
002142  376 015                 CMP     A,#CR
002144  314 331 003             CALL,EQ DOTAB
002147  376 012                 CMP     A,#LF
002151  314 253 004             CALL,EQ EFLI

002154  267                     OR      A,A
002155  050 012                 JUMP.,EQ DIC2
002157  315 224 014             CALL    TESTHEX
002162  352 167 004             JUMP,PE DIC13
002165  346 200                 AND     A,#200  ;Inversion
002167  167           DIC13:    LOAD    (HL),A
002170  043                     INC     HL

002171  315 247 003     DIC2:   CALL    TESTDOWN
002174  314 371 003             CALL,EQ ROLL
002177  042 164 105             LOAD    POINT,HL
002202  341             DIC3    POP     IX
002203  030 233                 JUMP.   ROL6


                        ;;
                        ;;Appels d'affichage de chaines de caracteres
                        ;;-----------------------------------------
                        ;;
                        ;;????????
                        ;;.S ?DITEX     Affiche une chaine .ASCIZ (DITEXZ accepte)
                        ;;????????
                        ;;
                        ;;in    pointeur, HL adresse debut, 0 terminateur, OUTCAR
                        ;;out   pointeur / HL pointe pos suivante (deb texte suiv)
                        ;;mod   pointeur, HL
                        ;;
002205  365             DITEX:  PUSH    AF
002206  176             DITZ1:  LOAD    A,(HL)
002207  043                     INC     HL
002210  267                     OR      A,A
002211  050 227         DITZ2:  JUMP.,EQ ROL8
002213  347 020                 .U      ?DICAR
002215  030 367                 JUMP.   DITZ1

                        ;??????????
                        ;;123 ?TEXT     Idem avec adresse suivant l'appel
                        ;??????????
                        ;;
                        ;;in    (SP)
                        ;;out   -
                        ;;mod   -
                        ;;
002217  042 272 105     TEXT    LOAD    SIMUL1,HL
002223  353                     EX      HL,DE
002224  347 117                 .U      ?GETWRD
002226  353                     EX      HL,DE
002227  347 026                 .U      ?DITEX
002231  052 272 105             LOAD    HL,SIMUL1
002234  311                     RET


                        ;??????????
                        ;.133 ?TEXTIM
                        ;??????????
                        ;;
                        ;;in    PC pointe le texte
                        ;;out   PC pointe la suite du programme
                        ;;mod   -
                        ;;
                        TEXTIM:
002235  343                     EX      (SP),HL
002236  347 026                 .U      ?DITEX
002240  343                     EX      (SP),HL
002241  311                     RET


                        ;????????
                        ;.S8 ?DITEB     Affiche une chaine avec un terminateur qualco
                        ;????????
                        ;;
                        ;;in    pointeur, HL adresse debut, B terminateur, OUTCAR
                        ;;out   pointeur, HL pointe pos suiv
                        ;;mod   pointeur, HL
                        ;;
002242  365             DITXB:  PUSH    AF
002243  176             DITE1:  LOAD    A,(HL)
002244  043                     INC     HL
002245  270                     COMP    A,B
002246  050 342                 JUMP.,EQ DITZ2
002250  347 020                 .U      ?DICAR
002251  030 367                 JUMP.   DITE1


                        ;??????????
                        ;.Z4 ?EFLI      Efface la ligne depuis le pointeur sans le deplo
                        ;??????????
                        ;;
                        ;;in    pointeur
                        ;;out   A=0
                        ;;mod   A
                        ;;
002253  345             EFLI    PUSH    HL
002254  052 164 105             LOAD    HL,POINT
002257  175             EFL2    LOAD    A,L
002260  346 077                 AND     A,#77
002262  376 077                 CMP     A,#77
002264  050 003                 JUMP.,EQ EFL4
002266  043                     INC     HL
002267  066 040                 LOAD    (HL),#SPA
002271  030 371                 JUMP.   EFL2

002273  341             EFL4    POP     HL
002274  257                     XOR     A,A
002275  311                     RET

.PAGE 2

                        ;;
                        ;;Appels de lecture du clavier (par son FIFO)
                        ;;-----------------------------------------
                        ;;
                        ;;????????
                        ;;.1 ?GETCAR    Lit le FIFO du clavier ou appelle
                        ;;              une routine de lecture @INCAR
                        ;;????????
```

```
                         ;;
                         ;;in    INCAR
                         ;;out   A caractere
                         ;;mod   F,A
                         ;N. modifie pas l'etat de l'interruption
                         ;;
002276  305              GETCAR: PUSH   BC
002277  325                      PUSH   DE
002300  345                      PUSH   HL

002301  052 122 105              LOAD   HL,INCAR         ;Indirection pour GETCAR
002304  175                      LOAD   A,L
002305  264                      OR     A,H
002306  050 005                  JUMP,.EQ GETCA1
002310  021 332 004              LOAD   DE,#GETCA4
002313  325                      PUSH   DE
002314  351                      JUMP   (HL)

002315  052 164 105      GETCA1: LOAD   HL,POINT
002320  072 166 105              LOAD   A,CPOINT         ,MET LE POINTEUR
002323  167                      LOAD   (HL),A
002324  315 366 004      GETCA2. CALL   GETCA7
002327  050 373                  JUMP,.EQ GETCA2
002331  345                      PUSH   HL
002332  052 164 105              LOAD   HL,POINT         ;EFFACE LE POINTEUR
002335  066 040                  LOAD   (HL),#SPA
002337  341                      POP    HL
002340  365                      PUSH   AF
002341  361              GETCA3  POP    AF
002342  104                      LOAD   B,H
002343  115                      LOAD   C,L
002344  052 174 105              LOAD   HL,POICLA
002347  053                      DEC    HL
002350  042 174 105              LOAD   POICLA,HL
002353  142                      LOAD   H,D
002354  153                      LOAD   L,E
002355  176                      LOAD   A,(HL)
002356  043                      INC    HL
002357  355 260                  LDIR
002361  267                      OR     A,A
002362  341              GETCA4. POP    HL
002363  321              GETCA5, POP    DE
002364  301                      POP    BC
002365  311                      RET

002366  021 226 105      GETCA7: LOAD   DE,#DEBOCLA
002371  052 174 105              LOAD   HL,POICLA
002374  176                      LOAD   A,(HL)
002375  376 220                  COMP   A,#ENDQ
002377  040 004                  JUMP,.NE GETCA6
002401  053                      DEC    HL
002402  042 174 105              LOAD   POICLA,HL
002405  267              GETCA6: OR     A,A
002406  355 122                  SUBC   HL,DE
002410  311                      RET

                         ,????????
                         ;;15 ?IFCAR     Lit l'etat du clavier (FIFO)
                         ;????????
                         ;;
                         ;;in    -
                         ;;out   CS aucune touche en attente, A non mod
                         ;;      CC touche, A code ASCII
                         ;;mod   F,A
                         ;;
002411  305              IFCAR:  PUSH   BC
002412  325                      PUSH   DE
002413  345                      PUSH   HL
002414  365                      PUSH   AF
002415  315 366 004              CALL   GETCA7
002420  040 317                  JUMP,.NE GETCA3
002422  361                      POP    AF
002423  067                      SETC
002424  030 334                  JUMP.  GETCA4

                         ;????????
                         ;;16 ?GETFON           Lit les touches fonctions
                         ,????????
                         ;;
                         ;;in    -
                         ;;out   A touches fonctions, EO si aucune touche fonction pressee
                         ;;mod   F,A
                         ;;
002426  072 176 105      GETFON: LOAD   A,FONCT
002431  267                      OR     A,A
002432  311                      RET

                         ,??????????
                         ;;5 ?GETLINE    Attend une ligne terminee par un CR
                         ,??????????
                         ;;
                         ;;in    pointeur, INCAR
                         ;;out   pointeur, HL Adr debut de chaine, 0 a la fin de la chaine
                         ;;              CS si retour car buffer plein
                         ;;mod   pointeur, HL, F,A
                         ;;
002433  305              GETLINE:PUSH   BC
002434  325                      PUSH   DE
002435  001 015 000              LOAD   BC,#0*400+CR
002440  021 377 016              LOAD   DE,#(CR+1)*400+400-1
002443  315 050 005              CALL   GETLI
002446  030 313                  JUMP   GETCA5

                         ;????????????
                         ;;35 ?GETLBCDE  Attend une ligne terminee par diff car
                         ;;              voir exemple routines INOC, INDEC
                         ;????????????
                         ;;
                         ;;in    pointeur, B, C, D, E, INCAR
                         ;;out   pointeur, HL adr debut buffer, A der car, 0 term
                         ;;              CS si retour car buffer plein
                         ;;              rev1 B touche fonction der car
                         ;;mod   F, A, HL, rev1 B
                         ;;
                         GETLBCDE.
002450  041 300 105      GETLI:  LOAD   HL,#GWORD
002453  347 001          GETL2:  .W     ?GETCAR
002455  347 000                  .W     ?DICAR
002457  376 010                  COMP   A,#BS
002461  050 043                  JUMP,.EQ GETL8   ,Effacement der car
002463  376 177                  COMP   A,#DEL
002465  050 051                  JUMP,.EQ GETL9

002467  270                      COMP   A,B
002470  070 031                  JUMP,.LO GETL6
002472  271                      COMP   A,C
002473  070 000                  JUMP,.LO GETL4
002475  272                      COMP   A,D
002476  070 023                  JUMP,.LO GETL6
002500  273                      COMP   A,E
002501  060 020                  JUMP,.HS GETL6
                         GETL4:
002502  167                      LOAD   (HL),A
002504  043                      INC    HL
002505  043                      INC    HL       ;fin buffer?
002506  176                      LOAD   A,(HL)
002507  053                      DEC    HL
002510  376 220                  COMP   A,#ENDQ
002512  040 337                  JUMP,.NE GETL2
002514  067              GETL42, SETC                    ;Erreur
002515  066 000          GETL5:  LOAD   (HL),#0
002517  041 300 105              LOAD   HL,#GWORD
002522  311                      RET

002523  267              GETL6   OR     A,A      ,Retour correct
002524  030 367                  JUMP.  GETL5

002526  325              GETL8   PUSH   DE
```

```
002527  021 300 105              LOAD   DE,#GWORD
002532  315 311 003              CALL   ?DFALK
002535  321                      POP    DE
002536  030 313                  JUMP.  GETL2

002540  337 043          GETL9.  ...    ?????
002542  030 304                  JUMP   GETLI

                         ;;Appels pour peripheriques
                         ;;------------------------
                         ;;
002544  000 000 000 100  TINIT:  .BYTE  0,0,0,RESMODE,MODE,COMM!CMS1!M11
             000008 NBPARA  =.-TINIT

                         ,---    Init USART C contient l'adr de mode  (5 OU 7)

002552  006 008          INUSA:  LOAD   D,#NBPARA
002554  041 144 005              LOAD   HL,#TINIT
002557  355 263                  OUTIR
002561  311                      RET

                         ,????????
                         ;;24 ?RPR        Attend et lit
                         ;;25 ?RMOD       Transferts lecture USART
                         ;;50 ?RPAR       Interface parallele
                         ,????????
                         ;;
                         ;;in    -
                         ;;out   A caractere
                         ;;mod   F, A
                         ;;
002562  305              RPR.    PUSH   BC
002563  016 005                  LOAD   C,#SPR
002565  030 010                  JUMP.  RDA

002567  305              RMOD.   PUSH   BC
002570  016 007                  LOAD   C,#SUSA2
002572  030 003                  JUMP.  RDA

002574  305              RPAR:   PUSH   BC
002575  016 003                  LOAD   C,#SPAR

                         RDA:
002577  355 170          RDA2:   LOAD   A,S(C)
002601  346 002                  AND    A,#FULP
002603  050 372                  JUMP,.EQ RDA2
002605  015              RDA4    DEC    C
002606  355 170                  LOAD   A,S(C)
002610  030 053                  JUMP.  WDA5

                         ,?????????
                         ;;44 ?IFRPR
                         ;;45 ?IFRMOD     Test et lit si le caractere est disponible
                         ;;51 ?IFRPAR
                         ,?????????
                         ;;
                         ;;in    -
                         ;;out   CS pas de caractere (FULL=0)
                         ;;       CC car lu dans A (clear le FULL)
                         ;;mod   F, A
                         ;;
002612  305              IFRPR:  PUSH   BC
002613  016 005                  LOAD   C,#SPR
002615  030 010                  JUMP.  IFDA

002617  305              IFRMOD: PUSH   BC
002620  016 007                  LOAD   C,#SUSA2
002622  030 003                  JUMP.  IFDA

002624  305              IFRPAR. PUSH   BC
002625  016 003                  LOAD   C,#SPAR

002627  355 170          IFDA:   LOAD   A,S(C)
002631  346 002                  AND    A,#FULP
002633  040 350                  JUMP,.NE RDA4
002635  067              IFDA2:  SETC
002636  301                      POP    BC
002637  311                      RET

                         ,????????
                         ;;26 ?WPR
                         ;;27 ?WMOD       Transferts ecriture USART
                         ;;52 ?WPAR       ou interf //
                         ,????????
                         ;;
                         ;;in    A car
                         ;;out   -
                         ;;mod   F
                         ;;
002640  305              WPR.    PUSH   BC
002641  016 005                  LOAD   C,#SPR
002643  030 010                  JUMP.  WDA

002645  305              WMOD:   PUSH   BC
002646  016 007                  LOAD   C,#SUSA2
002650  030 003                  JUMP.  WDA

002652  305              WPAR.   PUSH   BC
002653  016 003                  LOAD   C,#SPAR

002655  365              WDA     PUSH   AF
002656  355 170          WDA2:   LOAD   A,S(C)
002660  346 001                  AND    A,#RDYP
002662  050 372                  JUMP,.EQ WDA2
002664  361              WDA4:   POP    AF
002665  015                      DEC    C
002666  355 171                  LOAD   S(C),A
002670  301              WDA5:   POP    BC
002671  267                      OR     A,A
002672  323 003                  LOAD   $HP,A
002674  311                      RET

                         ,??????????
                         ;;46 ?IFWPR
                         ;;47 ?IFWMOD     Teste et envoie si pret
                         ;;53 ?IFWPAR
                         ,??????????
                         ;;in    -
                         ;;out   CS pas pret
                         ;;       CC pret (READY=1)
                         ;;mod   F
                         ;;
002675  305              IFWPR   PUSH   BC
002676  016 005                  LOAD   C,#SPR
002700  030 010                  JUMP.  IFWDA

002702  305              IFWMOD  PUSH   BC
002703  016 007                  LOAD   C,#SUSA2
002705  030 003                  JUMP.  IFWDA

002707  305              IFWPAR: PUSH   BC
002710  016 003                  LOAD   C,#SPAR

002712  365              IFWDA   PUSH   AF
002713  355 170                  LOAD   A,S(C)
002716  346 001                  AND    A,#RDYP
002717  040 343                  JUMP,.NE WDA4
002721  361                      POP    AF
002722  030 311                  JUMP   IFDA2

                         ;Appels d'extension du repertoire d'instruction (suite)
                         ;;------------------------------------------------------
```

```
                            ;;
                            ;??????????
                            ;;55 ?JUMPIDE   JUMP  @(DE)
                            ;??????????
                            ;;
                            ;;in    DE
                            ;;out   DE incr de 2, saut effectué, pile correcte
                            ;;mod   DE
                            ;;
002724  063                 JUMPIDE:INC     SP
002725  063                         INC     SP

                            ;??????????
                            ;;63 ?CALLIDE   CALL  @(DE)
                            ;??????????
                            ;;
                            ;;in    DE
                            ;;out   DE incr de 2, appel effectué, pile correcte
                            ;;mod   DE
                            ;;
002726  345                 CALLIDE:PUSH    HL
002727  353                         EX      HL,DE
002730  136                         LOAD    E,(HL)
002731  043                         INC     HL
002732  126                         LOAD    D,(HL)
002733  043                         INC     HL
002734  353                         EX      HL,DE
002735  343                         EX      (SP),HL
002736  311                         RET


                            ;??????????
                            ;;41 ?GETCURSOR Donne les coordonnees du pointeur
                            ;;              et le caract sous le pointeur
                            ;??????????
                            ;;
                            ;;in    pointeur
                            ;;out   A caract        HL coord
                            ;;mod   HL,A
                            ;;
                            GETCURSOR:
002737  052 164 105                 LOAD    HL,POINT
002742  176                         LOAD    A,(HL)
002743  365                         PUSH    AF
002744  174                         LOAD    A,H
002745  326 103                     SUB     A,#SALPHA/400
002747  313 045                     SLC     L
002751  027                         RLC     A
002752  313 045                     SLC     L
002754  027                         RLC     A
002755  147                         LOAD    H,A
002756  313 075                     SRC     L
002760  313 075                     SRC     L
002762  361                         POP     AF
002763  311                         RET


                            ;====================================
                            ;ROUTINE FOR CLOCK MODULE

002764  325                 RDCLK:  PUSH    DE
002765  021 000 105                 LOAD    DE,#CLKBUF
002770  347 140                     .W      ?RDCLK
002772  365                         PUSH    AF
002773  031                         ADD     HL,DE
002774  361                         POP     AF
002775  321                         POP     DE
002776  311                         RET
                            ;====================================


003000                              .LOC    ROM2-1000

                            ;;
                            ;;
                            ;;
                            ;;*    S O F T   G R A P H I Q U E    *
                            ;;-----------------------------------------;
                            ;
                            ; PRINCIPES GENERAUX
                            ; ------------------
                            ;
                            ;VOUS DONNEZ DANS UNE TABLE UNE SUITE DE POINTS
                            ;(COORDONNEE EN X,Y) ET LA ROUTINE "CONT" RELIE TOUS
                            ;CES POINTS ENTRE EUX.
                            ;VOUS DEVEZ ENCORE DIRE COMMENT SERA LE TRAIT QUI VA
                            ;RELIER TOUS LES POINTS (TRAIT PLEIN, POINTILLIE ... ).
                            ;IL EXISTE DEUX MODES PRINCIPAUX.
                            ;
                            ; 1)    LE MODE ABSOLU (ROUTINE CONTA).
                            ;       ON DONNE LA COORDONNEE DE CHAQUE POINT PAR
                            ;       RAPPORT A L'ORIGINE. LE MOTIF AINSI DEFINIT .
                            ;       NE PEUT PAS ETRE DEPLACER SUR L'ECRAN.
                            ;
                            ; 2)    LE MODE RELATIF (ROUTINE CONTR).
                            ;       ON DONNE LA COORDONNE DE CHAQUE POINT PAR
                            ;       RAPPORT AU PRECEDANT. IL FAUT ALORS DONNER
                            ;       LA COORDONNEE (PAR RAPPORT A L'ORIGINE) DU
                            ;       PREMIER POINT. EN MODIFIANT CETTE COORDONNEE,
                            ;       IL EST POSSIBLE DE METTRE LE MOTIF N'IMPORTE
                            ;       OU SUR L'ECRAN.
                            ;
                            ;LES COORDONNEES SONT LES SUIVANTES
                            ;;
                            ;;      ------------>
                            ;;      :           X
                            ;;      I
                            ;;      I
                            ;;      I
                            ;;      V Y
                            ;;
                            ;; 0 <= X <= 377
                            ;; 0 <= Y <= 177
                            ;;XX    = 1
                            ;;YY    = 400   ;Facteurs multiplicatifs pour init DE, HL
                            ;;


                            ;INDIRECTION POUR ROUTINES GRAPHIQUES

003000  303 276 007         JUMP    SETP
003003  303 314 007         JUMP    CLRP
003006  303 330 007         JUMP    INVP
003011  303 343 007         JUMP    TESTP
003014  303 233 006         JUMP    CMOD
003017  303 142 006         JUMP    SEGA
003022  303 ??? 006         JUMP    SEGRA
003025  303 033 006         JUMP    CONTA
003030  303 120 006         JUMP    CONTR

                000377 ENDC     =377
                000376 XMODE    =376

                            ;ROUTINES GRAPHIQUES POUR FAUCON.


                            ;;
                            ;;3025 CONTA ---
                            ;;
                            ;;Trace le contour avec les points donnes en absolu.
                            ;;Format de la table:
                            ;;TABLE .BYTE XMODE.TMINCE        Mode trait mince
                            ;;      .BYTE   m       n         m => Y   n => X
                            ;;      .BYTE   m       n
                            ;;      .BYTE   ..
                            ;;      .BYTE   ENDC              Fin de la table
                            ;;
                            ;;ENDC  =377
                            ;;XMODE =376

                            ;;EFFAC         =0      ;Efface le trait correspondant
                            ;;TMINCE        =1      ;Trait mince continu
                            ;;TGROS         =2      ;Gros trait continu
                            ;;PSERRE        =3      ; ...............
                            ;;PLARGE        =4      ; - - - - - - - -
                            ;;TPOINT        =5      ; _ . _ . _ . _ .
                            ;;CROIX         =6      ; + + + + + + + +
                            ;;SAUTE         =10     ;Saute sans rien modifier
                            ;;
                            ;Le .BYTE XMODE.xxx peut intervenir n'importe quand.
                            ;tous les points suivants seront alors relies
                            ;par le mode correspondant au dernier .BYTE XMODE.xxx.
                            ;Une table peut eventuellement ne pas avoir de
                            ;.BYTE XMODE.xxx .La definition du mode se fera alors
                            ;a l'aide de la routine "CMOD" avant l'appel de "CONTA".
                            ;;in    IX  table
                            ;;out   IX  fin table
                            ;;mod   IX
                            ;;
003033  345                 CONTA   PUSH    HL
003034  325                         PUSH    DE
003035  305                         PUSH    BC
003036  365                         PUSH    AF
003037  315 073 006                 CALL    CONTAS
003042  353                         EX      DE,HL
003043  315 073 006         CONTA2: CALL    CONTAS
003046  332 307 007                 JUMP.CS RETOUR
003051  315 142 006                 CALL    SEGA
003054  315 226 006                 CALL    CMOD1
003057  030 362                     JUMP    CONTA2

003061  067                 CONTA3: SETC
003062  311                         RET

003063  335 176 000         CONTA4  LOAD    A,(IX)
003066  335 043                     INC     IX
003070  315 233 006                 CALL    CMOD
003073  335 176 000         CONTAS  LOAD    A,(IX)
003076  335 043                     INC     IX
003100  376 377                     COMP    A,#ENDC
003102  050 355                     JUMP.EQ CONTA3
003104  376 376                     COMP    A,#XMODE
003106  050 353                     JUMP.EQ CONTA4
003110  147                         LOAD    H,A
003111  335 156 000                 LOAD    L,(IX)
003114  335 043                     INC     IX
003116  267                         OR      A,A             ;CARRY <- 0
003117  311                         RET

                            ;;
                            ;;3030 CONTR ---
                            ;;
                            ;;Trace le contour avec les points donnes en relatif.
                            ;Format table:
                            ;voir explications de "CONTA"
                            ;
                            ;;Les valeurs m et n peuvent etres positives ou
                            ;;negatives. Par exemple, la valeur -3 sera representee
                            ;;par 203 (et non 375 !!)
                            ;;Les valeurs m et n doivent etres comprises entre
                            ;;-177 et +177. (impossibilite de traverser completement
                            ;;l'ecran horizontalement en une seule fois).
                            ;;
                            ;;in    IX  table       DEBUT -> ED (X;Y)
                            ;;out   IX  fin table   ED pointe le point d'arrivee
                            ;;mod   IX
                            ;;
                            ;       NPX     -> H (7 BITS + SIGNE)
                            ;       NPY     -> L (       "       )
                            ;
003120  345                 CONTR:  PUSH    HL
003121  325                         PUSH    DE
003122  305                         PUSH    BC
003123  365                         PUSH    AF
003124  315 073 006         CONTR2: CALL    CONTAS
003127  332 307 007                 JUMP.CS RETOUR
003132  315 323 006                 CALL    SEGRE
003135  315 226 006                 CALL    CMOD1
003140  030 362                     JUMP    CONTR2

                            ;;
                            ;;3017 SEGA ---
                            ;;
                            ;;Relie deux points par une droite selon le mode predefini
                            ;;Absolu.

                            ;;in    DEBUT   -> ED (X;Y)
                            ;;      FIN     -> LH (X;Y)
                            ;;out   ED -> FIN
                            ;;mod   F,A,BC,DE,HL
                            ;;

003142  325                 SEGA:   PUSH    DE
003143  173                         LOAD    A,E
003144  275                         COMP    A,L
003145  040 011                     JUMP.NE SEGA1
003147  172                         LOAD    A,D
003150  274                         COMP    A,H
003151  040 005                     JUMP.NE SEGA1
003153  315 364 006                 CALL    POINTE          ;DEPART=ARRIVEE
003156  321                         POP     DE
003157  311                         RET

003160  016 024             SEGA1:  LOAD    C,#24           ;CODE INC D
003162  006 034                     LOAD    B,#34           ;CODE INC E
003164  174                         LOAD    A,H
003165  222                         SUB     A,D
003166  060 003                     JUMP.CC SEGA2           ;NBRE POS.
003170  355 104                     NEG     A
003172  014                         INC     C               ;CODE DEC D
003173  147                 SEGA2:  LOAD    H,A             ;NPX -> H
003174  175                         LOAD    A,L
003175  223                         SUB     A,E
003176  060 003                     JUMP.CC SEGA3           ;NBRE POS.
003200  355 104                     NEG     A
003202  004                         INC     B               ;CODE DEC E
003203  157                 SEGA3:  LOAD    L,A             ;NPY -> L
003204  124                         LOAD    D,H             ;D NPT SUPPOSE
003205  135                         LOAD    E,L             ;E NPD SUPPOSE
003206  171                         LOAD    A,C             ;CODE PAS SIMPLE
003207  062 270 105                 LOAD    SIMULO,A
003212  172                         LOAD    A,D
003213  223                         SUB     A,E
003214  060 003                     JUMP.CC SEGA4           ;SUPPOSITION JUSTE
003216  125                         LOAD    D,L             ;CORRECTION
003217  134                         LOAD    E,H
003220  170                         LOAD    A,B
003221  062 270 105                 LOAD    SIMULO,A
003224  355 103 272 105 SEGA4:      LOAD    SIMUL1,BC
003230  102                         LOAD    B,D             ;NPT -> B
003231  143                         LOAD    H,E             ;NPD -> H
003232  170                         LOAD    A,B
003233  224                         SUB     A,H             ;CALCUL NPS
003234  070 002                     JUMP.CS SEGA5
003236  355 104                     NEG     A
003240  157                 SEGA5   LOAD    L,A             ;COMPL. NPS -> L
003241  321                         POP     DE

003242  315 364 006                 CALL    POINTE
003245  170                         LOAD    A,B
003246  313 077                     SRC     A               ;NPT/2
003250  205                         ADD     A,L
```

```
[illegible]  070 021                     JUMP.,CS PDBLE
[illegible]  365                         PUSH    AF
[illegible]  267                         OR      A,A
[illegible]  275                         COMP    A,L
[illegible]  050 021                     JUMP.,EQ ENUL
[illegible]  361                         POP     AF

003261  315 270 105       PS.M:      CALL    SIMULO
003264  315 364 006                  CALL    POINTE
[illegible]  204                     ADD     A,H
[illegible]  005                     DEC     B
003271  310                          RET,EQ
003272  060 365                      JUMP.CC PSIM
003274  315 272 105       PDBLE:     CALL    SIMUL1
003277  315 364 006                  CALL    POINTE
003302  205                          ADD     A,L
003303  005                          DEC     B
003304  310                          RET,EQ
003305  060 352                      JUMP.CC PSIM
003307  030 363                      JUMP.   PDBLE

003311  361               ENUL:      POP     AF
003312  315 272 105       CHLYDBL:   CALL    SIMUL1
003316  315 364 006                  CALL    POINTE
003320  020 370                      DECJ.NE B.CHLYDBL
003322  311                          RET

                          ;;
                          ;.2022 SEGRE ---
                          ;;
                          ;.Relie deux points par une droite selon le mode predefini.
                          ;.Relatif.

                          ;;in     L -> NPX (7 BITS + SIGNE)  H -> NPY (7 BITS + SIGNE)
                          ;;out    ED -> FIN DU TRAIT
                          ;;mod    tout
                          ;;
003323  325               SEGRE.     PUSH    DE
003324  016 024                      LOAD    C,#24         ;CODE INC D
003326  006 034                      LOAD    B,#34         ;CODE INC E
003330  174                          LOAD    A,H           ;NPX -> A
003331  267                          OR      A,A
003332  362 340 006                  JUMP.PL SEGR1
003335  346 177                      AND     A,#177
003337  014                          INC     C             ;CODE DEC D
003340  147               SEGR1:     LOAD    H,A           ;NPX -> H
003341  175                          LOAD    A,L           ;NPY -> A
003342  267                          OR      A,A
003343  362 203 006                  JUMP.PL SEGA3
003346  346 177                      AND     A,#177
003350  004                          INC     B             ;CODE DEC E
003351  030 230                      JUMP    SEGA3

                          ;;
                          ;.3014 CMOD ---
                          ;;
                          ;.Change le mode de trait
                          ;
                          ;;in     A -> MODE
                          ;;out    NEANT
                          ;;mod    A
                          ;;
003353  062 276 105       CMOD:      LOAD    MODEP,A
003356  076 002           CMOD1:     LOAD    A,#2          ;RESET COUNTER
003360  062 275 105                  LOAD    COUNTER,A
003363  311                          RET

                          ;--- POINTE ---

                          ;CETTE ROUTINE DECIDE SI UN POINT VA ETRE AFFICHE
                          ;OU PAS SUR L'ECRAN SUIVANT LE MODE DE TRAIT CHOISI.
                          ;DIFFERENTS MODES POSSIBLES,

              000000 EFFAC    =0      ;EFFACE LE TRAIT CORRESPONDANT
              000001 TMINCE   =1      ;TRAIT MINCE CONTINU
              000002 TGROS    =2      ;GROS TRAIT CONTINU
              000003 PSERRE   =3      ; ..............
              000004 PLARGE   =4      ; - - - - - - - - -
              000005 TPOINT   =5      ; _ . _ . _ . _ .
              000006 CROIX    =6      ; + + + + + + + +
              000010 SAUTE    =10     ;SAUTE SANS RIEN MODIFIER

                          ;INPUT: ED -> COORDONNEE EN (X;Y) DU POINT
                          ;OUTPUT NEANT
                          ;REGISTRES MODIFIES: AUCUN

003364  345               POINTE:    PUSH    HL
003365  325                          PUSH    DE
003366  305                          PUSH    BC
003367  365                          PUSH    AF
003370  072 275 105                  LOAD    A,COUNTER     ;COUNTER -> B
003373  107                          LOAD    B,A
003374  072 276 105                  LOAD    A,MODEP       ;MODE -> A
003377  267                          OR      A,A
003400  050 032                      JUMP.,EQ CLR
003402  075                          DEC     A
003403  312 302 007                  JUMP.EQ RMINCE
003406  075                          DEC     A
003407  050 063                      JUMP.,EQ RGROS
003411  075                          DEC     A
003412  050 076                      JUMP.,EQ RPFIN
003414  075                          DEC     A
003415  050 105                      JUMP.,EQ RPLARG
003417  075                          DEC     A
003420  050 113                      JUMP.,EQ RTPTP
003422  075                          DEC     A
003423  050 144                      JUMP.,EQ RCROIX
003425  170               POINT2:    LOAD    A,B           ;B -> COUNTER
003426  062 275 105                  LOAD    COUNTER,A
003431  303 307 007                  JUMP    RETOUR

                          ;EFFACE UNE ZONE LARGE DE 3 POINTS.

003434  024               CLR:       INC     E
003435  315 314 007                  CALL    CLRP
003440  025                          DEC     D
003441  315 314 007                  CALL    CLRP
003444  035                          DEC     E
003445  315 314 007                  CALL    CLRP
003450  035                          DEC     E
003451  315 314 007                  CALL    CLRP
003454  024                          INC     D
003455  315 314 007                  CALL    CLRP
003460  024                          INC     D
003461  315 314 007                  CALL    CLRP
003464  034                          INC     E
003465  315 314 007                  CALL    CLRP
003470  034                          INC     E
003471  303 320 007                  JUMP    CLRP1    ;Attention aux POPs

                          ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

003474  315 276 007       RGROS:     CALL    SETP
003477  024                          INC     D
003500  315 276 007                  CALL    SETP
003503  034                          INC     E
003504  315 276 007                  CALL    SETP
003507  025                          DEC     D
003510  030 170                      JUMP.   RMINCE   ;Attention aux POPs

                          . . . . . . . . . . . . . . . . . . .

003512  004               RPFIN:     INC     B
003513  170                          LOAD    A,B
003514  037                          RRC     A
003515  060 306                      JUMP.,CC POINT2
003517  315 276 007                  CALL    SETP
003522  030 301                      JUMP    POINT2

                          ; - - - - - - - - - - - - - - - - - -
003524  020 004           RPLARG:    DECJ.NE B.RPLAR2
003526  006 005                      LOAD    B,#5
003530  030 273                      JUMP    POINT2

003532  170               RPLAR2:    LOAD    A,B
003533  030 021                      JUMP.   RTPT3

                          ; _ . _ . _ . _ . _ . _ . _ . _ . _
003535  020 004           RTPTP:     DECJ.NE B.RTPT2
003537  006 010                      LOAD    B,#10
003541  030 262                      JUMP    POINT2

003543  170               RTPT2:     LOAD    A,B
003544  075                          DEC     A
003545  050 256                      JUMP.EQ POINT2
003547  075                          DEC     A
003550  050 253                      JUMP.EQ POINT2
003552  075                          DEC     A
003553  075                          DEC     A
003554  050 247                      JUMP.EQ POINT2
003556  075               RTPT3      DEC     A
003557  050 244                      JUMP.EQ POINT2
003561  075                          DEC     A
003562  050 241                      JUMP.EQ POINT2
003564  315 276 007                  CALL    SETP
003567  030 234                      JUMP    POINT2

                          ; + + + + + + + + + + + + + + + + + +

003571  020 030           RCROIX:    DECJ.NE B.RCROI1
003573  006 004                      LOAD    B,#4
003575  315 276 007                  CALL    SETP
003600  035                          DEC     E
003601  315 276 007                  CALL    SETP
003604  034                          INC     E
003605  025                          DEC     D
003606  315 276 007                  CALL    SETP
003611  024                          INC     D
003612  024                          INC     D
003613  315 276 007                  CALL    SETP
003616  025                          DEC     D
003617  034                          INC     E
003620  315 276 007                  CALL    SETP
003623  030 200           RCROI1:    JUMP    POINT2


                          ;--- PUTCO ---

                          ;CETTE ROUTINE PLACE DANS HL L'ADRESSE CORRESPONDANT AU
                          ;POINT DONT LES COORDONNEES EN (X;Y) ONT ETE DONNEES
                          ;DANS (E D).
                          ;MET DANS A UNE VALEUR DONT UN SEUL BIT EST A UN ET
                          ;QUI PERMET DE TROUVER QUEL EST LE BIT CONCERNE DANS
                          ;LE BYTE POINTE PAR HL.
                          ;
                          ;
                          ;L'ECRAN GRAPHIQUE EST CONSIDERE COMME AYANT 4K
                          ;(256.*128.).
                          ;
                          ;INPUT: D -> COORDONNEE EN Y
                          ;       E ->     "       " X
                          ;
                          ;OUTPUT A -> MASQUE
                          ;       HL-> ADRESSE DANS SGRA
                          ;
                          ;REGISTRES MODIFIES: A.B.C.D.E.H.L
                          ;
                          ;E:     X7.X6.X5.X4.X3.X2.X1.X0
                          ;D:     0 .Y6.Y5.Y4.Y3.Y2.Y1.Y0
                          ;
                          ;HL:    0 ,0 ,0 ,0 ,Y6,Y5,Y4,Y3,Y2,Y1,X7,X6,X5,X4,X3,X2
                          ;       \______________________/
                          ;                  V
                          ;              +SGRA/400

003625  172               PUTCO:     LOAD    A,D
003626  376 170                      COMP    A,#LIGRA*2    ;Protection a partir de 524
003630  070 002                      JUMP.,LO PUTC1
003632  026 167                      LOAD    D,#LIGRA*2-1
003634  257               PUTC1.     XOR     A,A           ;LOAD   A,#0
003635  107                          LOAD    B,A
003636  313 072                      SRC     D             ;CARRY <- Y0
003640  027                          RLC     A
003641  313 072                      SRC     D             ;CARRY <- Y1
003643  313 033                      RRC     E             ;CARRY <- X0
003645  027                          RLC     A
003646  313 072                      SRC     D             ;CARRY <- Y2
003650  313 033                      RRC     E             ;CARRY <- X1
003652  027                          RLC     A
003653  117                          LOAD    C,A
003654  041 266 007                  LOAD    HL,#MTABLE
003657  011                          ADD     HL,BC
003660  176                          LOAD    A,(HL)        ;A <- MASQUE
003661  041 000 106                  LOAD    HL,#SGRA
003664  031                          ADD     HL,DE         ;HL <- ADRESSE DANS SGRA
003665  311                          RET

003666  200 040 100 020 MTABLE: .BYTE 200.40.100.20.10.2.4.1

                          ;;
                          ;;3000 SETP  ---
                          ;;3003 CLRP  ---
                          ;;3006 INVP ---
                          ;;
                          ;;Set, clear, inverse un point sur l'ecran
                          ;;dont on a donne la coordonnee dans (E,D).
                          ;;
                          ;;in     E -> COORDONNEE EN X
                          ;;       D -> COORDONNEE EN Y
                          ;;out    -
                          ;;mod    -
                          ;;
003676  345               SETP       PUSH    HL
003677  325                          PUSH    DE
003700  305                          PUSH    BC
003701  365                          PUSH    AF
003702  315 225 007       RMINCE     CALL    PUTCO     ;POP pour RMINCE !
003705  263                          OR      A,(HL)
003706  167                          LOAD    (HL),A
003707  361               RETOUR:    POP     AF
003710  301               RETOU1:    POP     BC
003711  321                          POP     DE
003712  341                          POP     HL
003713  311                          RET

003714  345               CLRP       PUSH    HL
003715  325                          PUSH    DE
003716  305                          PUSH    BC
003717  365                          PUSH    AF
003720  315 225 007       CLRP1:     CALL    PUTCO     ;POP pour CLRP1 !
003723  057                          CPL     A
003724  246                          AND     A,(HL)
003725  167                          LOAD    (HL),A
003726  030 337                      JUMP    RETOUR

003730  345               INVP       PUSH    HL
003731  325                          PUSH    DE
003732  305                          PUSH    BC
003733  365                          PUSH    AF
003734  315 225 007                  CALL    PUTCO
003737  256                          XOR     A,(HL)
003740  167                          LOAD    (HL),A
003741  030 344                      JUMP    RETOUR
```

```
                            ;;2011 TESTP ---
                            ;;
                            ;;Teste un point
                            ;;dont on a donne la coordonnee dans (E,D).
                            ;;
                            ;;in     E -> COORDONNEE EN X
                            ;;       D -> COORDONNEE EN Y
                            ;;out    EQ si un point est present, NE autrement
                            ;;mod    - (AF pour TESTP)
                            ;;
003743  345                 TESTP:  PUSH    HL
003744  325                         PUSH    DE
003745  305                         PUSH    BC
003746  315 225 007                 CALL    PUTCO
003751  246                         AND     A,(HL)
003752  030 334                     JUMP    RETOU1


003754  115 117 116 040 TMON:       .ASCIZ  "MON 1-<VAR>"
003764  110 105 130 101 THEX.       .ASCIZ  /HEXA/
003771  102 117 117 124 TBOOT:      .ASCII  /BOOT/

                            .PAGE 4

004000                      .LOC    ROM2

004000  303 101 011         JUMP"   DEB
004003  303 115 011         JUMP"   DEB1
004006  303 137 012         JUMP"   TRA
004011  303 035 001         JUMP"   FEE4
004014  303 101 011         JUMP"   DEB0

004017  347 032             .W      7BUZZ   ;Cassette suprimée !!!
004021  311                 RET
004022  347 032             .W      7BUZZ
004024  311                 RET
004025  347 032             .W      7BUZZ
004027  311                 RET

004030  303 322 013         JUMP"   HEADER
004033  303 225 000         JUMP"   INISYS
004036  303 145 011         JUMP"   DEB3
004041  303 323 013         JUMP"   FRBYTE

                            ;;
                            ;;
                            ;;
                            ;;
                            ;;Appels de lecture et conversion de pointeur
                            ;;------------------------------------------
                            ;;
                            ;??????????
                            ;;116 7COMPTIME Compare (HL) et (DE) longueur B octets
                            ;;              poids faibles en tete
                            ;??????????
                            ;;
                            ;;in     HL, DE pointeurs, B nb d'octets a comparer
                            ;;out    idem, pointent la fin
                            ;;        EQ si = , CS si <, CC si >=, SS si >
                            ;;mod    HL, DE, F, A
                            ;;
004044  305                 COMPTIME:PUSH   BC
004045  016 000                     LOAD    C,#0
004047  353                         EX      HL,DE
004050  032                 COM2:   LOAD    A,(DE)
004051  023                         INC     DE
004052  276                         COMP    A,(HL)
004053  043                         INC     HL
004054  050 010                     JUMP,.EQ COM6
004056  070 004                     JUMP,.CS COM4
004060  016 200                     LOAD    C,#200
004062  030 002                     JUMP.   COM6
004064  016 100             COM4:   LOAD    C,#100
004066  020 360             COM6:   DECJ.NE B,COM2
004070  313 041                     SLC     C
004072  353                         EX      HL,DE
004073  301                         POP     BC
004074  311                         RET


                            ;??????????
                            ;;64 7ALGRA      Translate coor. alpha in gra
                            ;??????????
                            ;;
                            ;;in     HL coor. alpha
                            ;;out    DE coor. gra
                            ;;mod    DE
                            ;;
                            ; D <-- (H*6)+5          ;Coin en bas
                            ; E <-- L*4              ;Coin a gauche

004075  365                 ALGRA:  PUSH    AF
004076  345                         PUSH    HL
004077  044                         INC     H       ;H+1
004100  174                         LOAD    A,H
004101  051                         ADD     HL,HL   ;(H+1)*2 et L*2
004102  204                         ADD     A,H     ;(H+1)*3
004103  147                         LOAD    H,A
004104  051                         ADD     HL,HL   ;(H+1)*6 et L*4
004105  045                         DEC     H       ;((H+1)*6)-1 = (H*6)+5
004106  353                         EX      HL,DE
004107  341                         POP     HL
004110  361                         POP     AF
004111  311                         RET

                            ;??????????
                            ;;77 7GRAAL      Translate coor. gra in alpha
                            ;??????????
                            ;;
                            ;;in     DE coor. gra
                            ;;out    HL coor. alpha
                            ;;mod    HL
                            ;;
                            ; H <-- D/6
                            ; L <-- E/4

004112  365                 GRAAL   PUSH    AF
004113  305                         PUSH    BC
004114  325                         PUSH    DE
004115  001 006 000                 LOAD    BC,#6
004120  150                         LOAD    L,B     ;LOAD HL,#0
004121  140                         LOAD    H,B
004122  347 060                     .W      7DIV    ;D/6
004124  142                         LOAD    H,D
004125  321                         POP     DE
004126  153                         LOAD    L,E
004127  313 075                     SRC     L       ;E/2
004131  313 075                     SRC     L       ;E/4
004133  301                         POP     BC
004134  361                         POP     AF
004135  311                         RET


                            ;;
                            ;;
                            ;;Appels pour lecture et affichages de nombres
                            ;;------------------------------------------
                            ;;
                            ;?????????
                            ;;73 7AFOHL      Affiche HL en octal
```

```
                            ;?????????
                            ;;
                            ;;in     pointeur, HL contenu
                            ;;out    pointeur
                            ;;mod    pointeur
                            ;;
004136  365                 AFOHL:  PUSH    AF
004137  174                         LOAD    A,H
004140  267                         OR      A,A     ;CLRC
004141  037                         RRC     A
004142  365                         PUSH    AF
004143  267                         OR      A,A     ;CLRC

004144  347 072                     .W      7AFOCA
004146  361                         POP     AF
004147  175                         LOAD    A,L

004150  347 072                     .W      7AFOCA
004152  361                         POP     AF
004153  311                         RET


                            ;?????????
                            ;;72 7AFOCA      Affiche le carry et A (9 bits) en octal
                            ;?????????
                            ;;
                            ;;in     pointeur, A nombre
                            ;;out    pointeur
                            ;;mod    pointeur
                            ;;
004154  267                 AFOCA3: OR      A,A
004155  365                 AFOCA   PUSH    AF
004156  305                         PUSH    BC
004157  006 003                     LOAD    B,#3
004161  027                 AFOC2:  RLC     A
004162  027                         RLC     A
004163  027                         RLC     A
004164  365                         PUSH    AF
004165  343 007                     AND     A,#7    ;Masque
004167  306 060                     ADD     A,#'0

004171  347 000                     .W      7DICAR
004173  361                         POP     AF
004174  020 363                     DECJ.NE B,AFOC2
004176  301                 AFOC4:  POP     BC
004177  361                         POP     AF
004200  311                         RET


                            ;??????????
                            ;;70 7AFBIN      Affiche un nombre binaire
                            ;??????????
                            ;;
                            ;;in     pointeur, A nombre
                            ;;out    pointeur
                            ;;mod    pointeur
                            ;;
004201  365                 AFBIN:  PUSH    AF
004202  305                         PUSH    BC
004203  117                         LOAD    C,A
004204  006 010                     LOAD    B,#8.
004206  257                 AFB2    XOR     A,A
004207  213 021                     RLC     C
004211  316 060                     ADDC    A,#'0

004213  347 000                     .W      7DICAR
004215  020 367                     DECJ.NE B,AFB2
004217  030 355                     JUMP.   AFOC4

                            ;??????????
                            ;;71 7AFHEX      Affiche un byte hexa ou BCD
                            ;??????????
                            ;;
                            ;;in     pointeur, A nombre
                            ;;out    pointeur
                            ;;mod    pointeur
                            ;;

004221  365                 AFHEX:  PUSH    AF
004222  305                 AFH2    PUSH    BC
004223  107                         LOAD    B,A
004224  007                         RL      A
004225  007                         RL      A
004226  007                         RL      A
004227  007                         RL      A
004230  315 241 010                 CALL    AFNIB
004233  170                         LOAD    A,B
004234  315 241 010                 CALL    AFNIB
004237  030 335                     JUMP.   AFOC4

                            ;---     Affichage 4 low bits de A

004241  346 017             AFNIB:  AND     A,#17
004243  306 060                     ADD     A,#'0
004245  376 072                     COMP    A,#'9+1
004247  070 002                     JUMP,.LO AFN2
004251  306 007                     ADD     A,#'A-('9+1)
004253  347 000             AFN2    .W      7DICAR
004255  311                         RET

                            ;???????????
                            ;;101 7AFXHL     Affiche HL en hex ou BCD
                            ;???????????
                            ;;
                            ;;in     HL nombre
                            ;;out    -
                            ;;mod    -
                            ;;
004256  365                 AFXHL:  PUSH    AF
004257  174                         LOAD    A,H
004260  347 071                     .W      7AFHEX
004262  175                         LOAD    A,L
004263  030 335                     JUMP    AFH2


                            ;???????????
                            ;;65 7INDEC      Lecture nbre BCD avec delete
                            ;;67 7INOC       Lecture nombre octal
                            ;;66 7INHEX      Lecture nombre hexa
                            ;???????????
                            ;;
                            ;;in     pointeur, INCAR
                            ;;out    HL nombre, C nbre digits, EQ si C=0, A der car
                            ;;mod    F, A, B, C, HL
                            ;;
004265  001 072 060         INDEC   LOAD    DE,#'0*400+'9+1
004270  325                 INDE2.  PUSH    DE
004271  021 377 377                 LOAD    DE,#-1
004274  030 014                     JUMP.   INNUM

004276  001 070 060         INOC:   LOAD    DE,#'0*400+'7+1
004301  030 365                     JUMP.   INDE2

004303  001 072 060         INHEX:  LOAD    DE,#'0*400+'9+1
004306  325                         [illegible]
004307  021 107 101                 LOAD    DE,#'A*400+'F+1

                            ;---     Lecture selon  B       C       D       E
                            ;                Decimal '0      '9+1    -1      -1
                            ;                Octal   '0      '7+1    -1      -1
                            ;                Hexa    '0      '9+1    'A      'F+1

004312  315 050 005         INNUM   CALL    GETLBCDE
004315  365                         PUSH    AF      ;sauve der car
004316  335 345                     PUSH    IX
004320  335 041 300 103             LOAD    IX,#GWORD       ;Buffer de ligne
004324  101                         LOAD    B,C
004325  016 000                     LOAD    C,#0
004327  041 000 000                 LOAD    HL,#0
```

```
004352  335 176 000     INN2:   LOAD    A,(IX)
004355  335 043                 INC     IX
004357  267                     OR      A,A
004360  050 031                 JUMP.,EQ INN8                   ;Fin du buffer
004362  272                     COMP    A,D
004363  060 002                 JUMP.,HS INN3
004365  220                     SUB     A,D
004366  030 002                 JUMP.   INN4

004350  326 067         INN3:   SUB     A,#'A-10.
004352  014             INN4:   INC     C
004353  051                     ADD     HL,HL
004354  051                     ADD     HL,HL
004355  051                     ADD     HL,HL
004356  313 113                 TEST    E 1                     ;Bit 2
004360  050 001                 JUMP.,EQ INN6
004362  051                     ADD     HL,HL
004363  205             INN6:   ADD     A,L
004364  157                     LOAD    L,A
004365  060 343                 JUMP.,CC INN2
004367  044                     INC     H
004370  030 340                 JUMP.   INN2

004372  335 341         INN8:   POP     IX
004374  361                     POP     AF
004375  321                     POP     DE
004376  311                     RET


                        ;.TITLE MONITEUR.SR      ;JDN:FAUCON

                        ;*** MONITEUR POUR FAUCON ET DATAC ***

004377  040 162 011     OTABLE: .BW     SPA,DOSE        ;SEPARATOR
004402  177 127 011             .BW     DEL,DEB2        ;CLEAR LINE AND ORDER DELETE
004405  134 020 012             .BW     '\,DOPE         ;OPEN LOC. \
004410  015 100 012             .BW     CR,DOCP         ;CLOSE LOC. CR
004413  136 050 012             .BW     '>,DONEXT       ;CLOSE AND OPEN NEXT LOC. >
004416  133 061 012             .BW     '<,DOPREV       ;CLOSE AND OPEN PREVIOUS LOC. <
004421  107 275 011             .BW     'G,DOG          ;GO. G
004424  113 320 011             .BW     'K,DOC          ;CONTINUE AFTER A BREAKPOINT. C
                        ;-      .BW     'N,DOE                  ;ENABLE INTERUPT. X
004427  111 105 013             .BW     'I,DOI          ;INPUT. I
004432  114 131 012             .BW     'L,DOLOAD       ;PIP LOADER. L
004435  122 156 012             .BW     'R,DOPRSTAT     ;PRINT PROCESSOR STATUS. R
004440  117 146 013             .BW     'O,DDOO         ;OUTPUT. O
004443  127 245 013             .BW     'W,MOWIN        ;MOVE IN WINDOW
004446  110 374 011             .BW     'H,DOCAL        ;CALL ROUTINE AND RETURN TO MONIT
004451  121 272 013             .BW     'Q,ENAEX        ;ENABLE OCTAL
004454  126 254 011             .BW     'V,CMODA
004457  123 231 011             .BW     'S,STEP
004462  132 274 011             .BW     'Z,DOZ
004465  011 237 011             .BW     TAB,GOTAB
004470  010 127 011             .BW     BS,DEB2
004473  000                     .BYTE   0

                        ;2 or 3 Arguments only orders
                        ;               M       MOVE
                        ;               P       PUNCH
                        ;               =       INIT

                        ; ENTREE DANS LE MONITEUR.
                        ; INITIALISE LES ADRESSE DE RESTART 50 ET NMI.

004474  252 115 117 116         .ASCII  "<252>MON<200>"

                        ;?????????
                        ;.134 ?MON       la deplacer
                        ;?????????
                        MON:

                        DEB:
                        DEB0:
                        ;-              LOAD    SP,#SPUTIL
                        ;       LOAD    HL,#TRA         ;Init pour moniteur
                        ;       LOAD    SAVLGO,HL
                        ;       LOAD    SAVPC,HL
004501  041 000 200             LOAD    HL,#BINAIRE
004504  042 104 105             LOAD    SAVLOP,HL

004507  072 177 105             LOAD    A,LAMODI                ;Sauvetage etat et passag
004512  062 111 105             LOAD    SAVMODI,A

                        DEB1:
                        ;-              LOAD    HL,MAXMEM
                        ;-              LOAD    SP,HL
                        ;-              CALL    INISY2
004515  016 004                 LOAD    C,#SELPROF
004517  347 126                 .W      ?IDIS
004521  347 032                 .W      ?BUZZ
004523  347 123 354 007         .W      ?TEXT,TXMON

004527  041 111 105     DEB2:   LOAD    HL,#SAVMODI     ;Pour permettre l'int. 50 Hz
004532  313 306                 SET     (HL) 0          ;lorsqu'on quitte le moniteur.
004534  347 043                 .W      ?RETURN
004536  041 137 012             LOAD    HL,#TRA
004541  347 014                 .W      ?IADNMI
004543  347 013                 .W      ?ITRAP

                        DEB3:
                        ;-              LOAD    SP,MAXMEM
004545  347 100                 .W      ?INMON
004547  021 377 010             LOAD    DE,#OTABLE
004552  347 061                 .W      ?JUMPCAR

                        ;-      ;Traitement des touches ctrl 0 --> ctrl 9
                        ;-      ;Recherche dans la memoire un motif de la forme
                        ;-      ;       .B      252
                        ;-      ;       .B      code touche (compris entre 20 et 32)
                        ;-      ;       .B      'F
                        ;-      ;       .B      200
                        ;-      ;et saute a l'adresse suivante.
                        ;-
                        ;-
                        ;-      AIGC.   COMP    A,#'0-40
                        ;-              JUMP.,LO AIG3
                        ;-              COMP    A,#'9+1-40
                        ;-              JUMP.,HS AIG3           ;not one
                        ;-              LOAD    L,A
                        ;-              LOAD    H,#'F
                        ;-              LOAD    FBRANCH,HL
                        ;-              LOAD    HL,#FBRANCH
                        ;-              CALL    BRANCH
                        ;-              JUMP.,CS AIG3
                        ;-              EX      HL,DE
                        ;-              JUMP    (HL)

004554  347 123 047 001 AIG3    .W      ?TEXT,TERROR
004560  030 303                 JUMP.   DEB3


                        ;SEPARATOR (SPACE OR COMMA)
                        ;FIRST PARAMETRE IN "HL"

004562  345             DOSE.   PUSH    HL
004563  347 100                 .W      ?INMON
004565  321                     POP     DE

                        ;TEST SECOND PARAMETER AND ORDER
                        ;FIRST PARAMETRE IN "DE"
                        ;SECOND PARAMETRE IN "HL"

004566  376 117                 COMP    A,#'O           ;OUTPUT TO PERIPH.
004570  312 170 013             JUMP.EQ DOO
004573  376 123                 COMP    A,#'S
004575  312 111 012             JUMP.EQ SEARCH
004600  267             DOSE1:  OR      A,A             ;CLRC
```

```
004601  355 122                 SUBC    HL,DE
004603  043                     INC     HL
004604  353                     EX      DE,HL
004605  070 315                 JUMP.,LO AIG3
004607  376 054                 COMP    A,#'A
004611  040 341                 JUMP.,NE AIG3

                        ;GET AND TEST THIRD PARAM. AND ORDER
                        ;FIRST PARAMETRE IN "HL"
                        ;SECOND-FIRST+1   IN "DE"
                        ;THIRD PARAMETRE IN "BC"

004613  331             DOSE2.  EX      EL
004614  347 100                 .W      ?INMON
004616  345                     PUSH    HL
004617  331                     EX      EL
004620  301                     POP     BC
004621  376 075                 COMP    A,#'=           ;INITIALISE WITH GIVEN NUMBER
004623  312 242 011             JUMP,EQ DOSEG
004626  376 120                 COMP    A,#'P           ;PAPER PUNCH
004630  312 220 014             JUMP,EQ DOPUNCH
004633  376 115                 COMP    A,#'M
004635  312 233 013             JUMP,EQ MOVE
004640  030 312                 JUMP    AIG3

                        ;++++++++++++++

                        ;INITIALISE MEMORY

004642  174             DOSEG.  LOAD    A,H
004643  161                     LOAD    (HL),C
004644  043             DOSEGS: INC     HL
004645  033                     DEC     DE
004646  172                     LOAD    A,D
004647  263                     OR      A,E
004650  040 370                 JUMP.,NE DOSEG
004652  030 253                 JUMP.   DEB2

004654  115             CMODA   LOAD    C,L
004655  347 126                 .W      ?IDIS
004657  030 246                 JUMP.   DEB2

                        ;-      ;ENABLE INTERUPT
                        ;-
                        ;-      DOE.    LOAD    A,L
                        ;-              LOAD    SAVINT,A
                        ;-              JUMP    DEB2

                        ;STEP (if loaded)

004661  345             STEP:   PUSH    HL
004662  052 116 105             LOAD    HL,SAVS
004665  343                     EX      (SP),HL
004666  311                     RET

004667  345             GOTAB:  PUSH    HL
004670  052 126 105             LOAD    HL,SAVT
004673  343                     EX      (SP),HL
                        DOZ:
004674  311                     RET

                        ;GO TO USER PROGRAM

004675  050 003         DOG:    JUMP.,EQ DOG2
004677  042 106 105             LOAD    SAVLGO,HL
004702  052 106 105     DOG2:   LOAD    HL,SAVLGO
004705  042 100 105             LOAD    SAVPC,HL
004710  041 050 105             LOAD    HL,#SPUTIL      ;the user's stack will be SPU

004713  042 102 105             LOAD    SAVSP,HL
004716  030 005                 JUMP.   DOC2

                        ;CONTINUE AFTER A TRAP

004720  050 003         DOC:    JUMP.,EQ DOC2
004722  042 100 105             LOAD    SAVPC,HL        ;YES, PLACE IT

                        ;;
                        ;;
                        ;;Appels de controle d'execution
                        ;;------------------------------
                        ;;

                        ;????????????
                        ;; 110 ?EXECUTE ;Execute selon l'etat sauve en memoire
                        ;????????????
                        ;;in    (positions memoire de 42453 a 52503)
                        ;;out   tous les reg
                        ;;mod   id
                        ;;
                        EXECUTE:
                        DOC2
004725  072 111 105             LOAD    A,SAVMODI
004730  323 000                 LOAD    $MODI,A
004732  062 177 105             LOAD    LAMODI,A
004735  363                     IOF
004736  061 054 105             LOAD    SP,#SPOOT-24
004741  375 341                 POP     IY
004743  335 341                 POP     IX
004745  010                     EX      AF
004746  331                     EX      EL
004747  341                     POP     HL
004750  321                     POP     DE
004751  301                     POP     BC
004752  361                     POP     AF
004753  010                     EX      AF
004754  331                     EX      EL
004755  341                     POP     HL
004756  321                     POP     DE
004757  301                     POP     BC
004760  361                     POP     AF
004761  355 173 102 105         LOAD    SP,SAVSP
004765  345                     PUSH    HL
004766  052 100 105             LOAD    HL,SAVPC
004771  343                     EX      (SP),HL
                        ;-              PUSH    AF
                        ;-              IOF
                        ;-              LOAD    A,SAVINT
                        ;-              TEST    A 0
                        ;-              JUMP.,EQ DOC4
                        ;-              ION
                        ;-      DOC4:   POP     AF
004772  373                     ION
004773  311                     RET

                        ;***    CALL    ROUTINE

                        ;The print of PC then a TRAP arrive will be false

004774  050 003         DOCAL.  JUMP.,EQ DOCAL2
004776  042 100 105             LOAD    SAVPC,HL
005001  061 050 105     DOCAL2: LOAD    SP,#SPUTIL
005004  041 137 012             LOAD    HL,#TRA
005007  345                     PUSH    HL
005010  041 046 105             LOAD    HL,#SPUTIL-2
005013  042 102 105             LOAD    SAVSP,HL
005016  030 306                 JUMP.   DOC2

                        ;OPEN LOCATION AND PRINT CONTENT

005020  040 003         DOPE.   JUMP.,NE DOPE3
005022  052 104 105             LOAD    HL,SAVLOP
                        DOPE3
005025  347 043                 .W      ?RETURN
                        DOPE33:
```

```
[illegible]  347 102                       .W      ?AFMOHL
[illegible]  076 072                       LOAD    A.#'
[illegible]  347 [illegible]               .W      ?DICAR
[illegible]  042 104 105   DOFE4.          LOAD    SAVLOP.HL
[illegible]  [illegible]                   LOAD    A.(HL)
[illegible]  347 101                       .W      ?AFMONA
[illegible]  347 042                       .W      ?SPACE
005043  303 145 011        DOFE5.          JUMP    DEB3

                           ;LOAD NUMBER AND OPEN NEXT LOCATION

                           DONEXT.
[illegible]  347 043                       .W      ?RETURN
[illegible]  353                           EX      HL.DE
[illegible]  052 104 105                   LOAD    HL.SAVLOP
[illegible]  050 001                       JUMP..EQ DONEX4 ;Test C
[illegible]  163                           LOAD    (HL).E
[illegible]  043           DONEX4. INC     HL
005062  030 343            DONEXS JUMP.    DOFE33

                           .LOAD NUMBER AND OPEN PREVIOUS

[illegible]  347 043       DOPREV. .W      ?RETURN
[illegible]  353                           EX      DE.HL
[illegible]  052 104 105                   LOAD    HL.SAVLOP
[illegible]  050 001                       JUMP..EQ DOPRV4
[illegible]  163                           LOAD    (HL).E
[illegible]  053           DOPRV4 DEC      HL
[illegible]  030 [illegible]               JUMP.   DONEXS

                           ;LOAD NUMBER AND CLOSE LOCATION

[illegible]  353           DOCR:           EX      DE.HL
[illegible]  052 104 105                   LOAD    HL.SAVLOP
[illegible]  050 001                       JUMP..EQ DOCR2
[illegible]  163                           LOAD    (HL).E
[illegible]  030 334       DOCR2:          JUMP    DOFE5

                           ;SEARCH IN MEMORY  (FROM REV 1-6)

005111  353                SEARCH. EX      HL.DE
[illegible]  173                           LOAD    A.E
[illegible]  001 000 000   SEAR2.          LOAD    BC.#0
[illegible]  355 261                       CPIR
[illegible]  342 127 011                   JUMP.VC DEB2
005123  053                                DEC     HL
[illegible]  042 104 105                   LOAD    SAVLOP.HL
[illegible]  030 274                       JUMP.   DOFE3

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

005131  315 [illegible] 014   DOLOAD: CALL  LOADBI
005134  322 115 011                   JUMP.CC DEB1

                           ,Retour apres un trap ou NMI

005137  315 202 012        TRA:    CALL    SAVESTAT
005142  341                        POP     HL              ;Corrige le stack
005143  042 100 105                LOAD    SAVPC.HL
005146  355 163 102 105 -          LOAD    SAVSP.SP
005152  052 140 105        TRA1:   LOAD    HL.MAXMEM       ;Place le stack a la fin de la RA
005155  371                        LOAD    SP.HL
                           ;-              CALL    INITRA
005155  347 107            DOPRSTAT .W     ?PRSTAT
005160  030 325                    JUMP    DOCR2

                           ;,
                           ;??????????
                           .;135 ?TRAPPE
                           ;??????????
                           .;
                           ;;in    -
                           ..cut   -
                           ;;mod
                           ;;
005162  315 202 012        TRAPPE: CALL    SAVESTAT
005165  321                        POP     DE
005166  341                        POP     HL
005167  042 100 105                LOAD    SAVPC.HL
005172  355 163 102 105            LOAD    SAVSP.SP
005176  325                        PUSH    DE
                           ;-              CALL    INITRA  ;?
005177  347 107                    .W      ?PRSTAT
005201  311                        RET

                           ;;
                           ;?????????
                           ;;106 ?SAVESTAT Sauvetage des registres dans la zone reservee du
                           ;?????????
                           ;;
                           ;;in    tous les registres
                           ;;out   - (adresses 42452 a 42503)
                           ;;mod   A, F
                           ;;
                           SAVESTAT.
005202  365                        PUSH    AF

[illegible]  072 177 105           LOAD    A.LAMODI        ;SAUVE L'ETAT DE L'ECRAN DANS SAV
005206  062 111 105                LOAD    SAVMODI.A

                           ;-              LOAD    A.I
                           ;-              LOAD    A.#1
                           ;-              JUMP.PE TRA2
                           ;-              XOR     A.A
                           ;-      TRA2    LOAD    SAVINT.A
005211  361                        POP     AF
005212  363                        IOF             ;A CAUSE D'UN EVENTUEL INTERRUPT
005213  343                        EX      (SP).HL
005214  042 100 105                LOAD    SAVPC.HL
005217  355 163 102 105            LOAD    SAVSP.SP
005223  243                        EX      (SP).HL
[illegible]  061 100 105           LOAD    SP.#SPODT
[illegible]  365                   PUSH    AF
[illegible]  305                   PUSH    BC
005231  325                        PUSH    DE
[illegible]  345                   PUSH    HL
[illegible]  010                   EX      AF
[illegible]  331                   EX      BL
[illegible]  365                   PUSH    AF
[illegible]  305                   PUSH    BC
[illegible]  325                   PUSH    DE
[illegible]  345                   PUSH    HL
[illegible]  010                   EX      AF
[illegible]  331                   EX      BL
[illegible]  335 345               PUSH    IX
[illegible]  375 345               PUSH    IY
[illegible]  355 173 102 105       LOAD    SP.SAVSP
[illegible]  373                   ION
[illegible]  311                   RET

                           ;?????????
                           ;;107 ?PRSTAT   Impression de l'etat du processeur
                           ;?????????
                           ;;
                           ;;in    - (positions memoire de 42452 a 42503)
                           ;;out   -
                           ;;mod   A, F, B, DE, HL
                           ;;
005255  347 043            PRSTAT. .W      ?RETURN
005257  021 076 106                LOAD    DE.#SPODT-2     ;(DE) --> SAVF
005262  315 035 013                CALL    AFFTOL          ;PRINT: F A B C D E DE HL
005265  052 102 105                LOAD    HL.SAVSP
005270  347 042                    .W      ?SPACE
005272  347 102                    .W      ?AFMOHL         ;PRINT: SP
[illegible]  176                   LOAD    A.(HL)
[illegible]  043                   INC     HL
[illegible]  146                   LOAD    H.(HL)
```

```
[illegible]  157                           LOAD    L.A
[illegible]  347 042                       .W      ?SPACE
[illegible]  347 102                       .W      ?AFMOHL   ;PRINT: (SP)
[illegible]  347 042                       .W      ?SPACE
[illegible]  052 100 105                   LOAD    HL.SAVPC
005321  347 102                            .W      ?AFMOHL   ;PRINT: PC
[illegible]  347 103                       .W      ?TCRLF
[illegible]  123 122 0[illegible] 110 TYPIS  .ASCII  "SZHPNC A B C D E DE"
[illegible]  040 040 040 040               .ASCII  "  HL  IX SP IY (SP) I PC "
                           ;-      TXION   .ASCIZ  " ION"
                           ;-      TXIOF   .ASCIZ  " IOF"
005416  033                        DEC     DE              ;(DE) --> SAVF'
005417  315 035 013                CALL    AFFTOL          ;PRINT: A' ... HL'
005422  315 065 013                CALL    AFFT2           ;PRINT: IX IY
005425  072 [illegible] 105        LOAD    A.SAVI          ;PRINT: I
005430  347 101                    .W      ?AFMONA
                           ;-              LOAD    A.SAVINT
                           ;-              TEST    A.0
                           ;-              LOAD    HL.#TXIOF
                           ;-              JUMP..EQ PRS2
                           ;-              LOAD    HL.#TXION
                           ;-      PRS2.
                           ;-              .W      ?DITEX2
005432  347 043                    .W      ?RETURN
005434  311                        RET

005435  [illegible]        AFFTOL. LOAD    A.(DE)          ;(DE) --> SAVF
[illegible]  347 070               .W      ?AFDIN
005440  347 042                    .W      ?SPACE
005442  023                        INC     DE              ;(DE) --> SAVA
005443  032                        LOAD    A.(DE)
005444  347 101                    .W      ?AFMONA
005446  347 042                    .W      ?SPACE
005450  033                        DEC     DE
005451  033                        DEC     DE              ;(DE) --> SAVB
005452  006 004                    LOAD    B.#4
[illegible]  032           AFFT2:  LOAD    A.(DE)
[illegible]  033                   DEC     DE
[illegible]  347 101               .W      ?AFMONA
005460  347 042                    .W      ?SPACE
[illegible]  020 370               DECJ.NE B.AFFT2
005464  023                        INC     DE
[illegible]  023                   INC     DE              ;(DE) --> SAVD
[illegible]  006 002       AFT02   LOAD    B.#2
[illegible]  032           AFF4    LOAD    A.(DE)
005471  033                        DEC     DE              ;(DE) --> SAVE OR SAVL
005472  147                        LOAD    H.A
005473  032                        LOAD    A.(DE)
[illegible]  033                   DEC     DE
[illegible]  157                   LOAD    L.A
[illegible]  347 102               .W      ?AFMOHL
[illegible]  347 042               .W      ?SPACE
[illegible]  020 334               DECJ.NE B.AFF4
[illegible]  311                   RET

                           ;INPUT FROM PERIPHERAL

005505  040 005            DOI:    JUMP..NE DOI2
005507  072 273 105                LOAD    A.SAVADIO       ;GET LAST DEVICE CODE
005512  157                        LOAD    L.A
005513  267                        OR      A.A
005514  175                DOI2:   LOAD    A.L
005515  347 043                    .W      ?RETURN
005517  347 101                    .W      ?AFMONA         ;PRINT IT
005521  062 273 105                LOAD    SAVADIO.A       ;REMEMBER IT
[illegible]  076 044               LOAD    A.#'$
[illegible]  347 000               .W      ?DICAR
005530  076 072                    LOAD    A.#':
005532  347 000                    .W      ?DICAR
005534  115                        LOAD    C.L
005535  355 170                    LOAD    A.S(C)
005537  347 101                    .W      ?AFMONA
005541  347 042                    .W      ?SPACE          ;PRINT DATA INPUT
005543  303 145 011                JUMP    DEB3

                           ;OUTPUT TO PERIPHERALS

005546  135                DOO0:   LOAD    E.L             ;PLACE EVENTUAL DATA IN E
005547  040 005                    JUMP..NE DOO2
005551  072 272 105                LOAD    A.SAVDA         ;0 PARAM, PLACE OLD DATA
005554  074                        INC     A
005555  137                        LOAD    E.A
005556  173                DOO2:   LOAD    A.E
005557  347 042                    .W      ?SPACE
005561  347 101                    .W      ?AFMONA
005563  347 042                    .W      ?SPACE
005565  072 273 105                LOAD    A.SAVADIO       ;GET LAST DEVICE CODE
005570  157                        LOAD    L.A             ;PLACE IT
005571  267                        OR      A.A
005572  347 101                    .W      ?AFMONA         ;PRINT IT
005574  030 016                    JUMP.   DO2

                           DOO.
005576  347 043                    .W      ?RETURN
005600  173                        LOAD    A.E
005601  347 101                    .W      ?AFMONA
005603  347 042                    .W      ?SPACE
005605  175                        LOAD    A.L
005606  347 101                    .W      ?AFMONA
005610  175                        LOAD    A.L             ;GET DEVICE CODE
005611  062 273 105                LOAD    SAVADIO.A       ;REMEMBER IT
005614  076 044            DO2:    LOAD    A.#'$
005616  347 000                    .W      ?DICAR
005620  173                        LOAD    A.E             ;PLACE DATA TO OUTPUT
005621  062 272 105                LOAD    SAVDA.A         ;REMEMBER IT
005624  115                        LOAD    C.L
005625  355 171                    LOAD    S(C).A
005627  303 127 011        DO4:    JUMP    DEB2

                           ;+++    MOVE

005632  325                MOVE:   PUSH    DE
005633  305                        PUSH    BC
005634  321                        POP     DE
005635  301                        POP     BC
005636  355 260            MOV2.   LDIR
005640  042 104 105                LOAD    SAVLOP.HL
005643  030 362                    JUMP.   DO4

                           ;+++    MOVE 1K IN WINDOW

005645  040 003            MOWIN:  JUMP..NE MOW2
005647  052 104 105                LOAD    HL.SAVLOP
                           MOW2.
[illegible]  021 000 101           LOAD    DE.[illegible]
[illegible]  001 000 004           LOAD    BC.[illegible]
[illegible]  030 [illegible]       JUMP.   MOV2

                           ;+++    Switch to inverted mode

[illegible]  313 [illegible]  MODINV  [illegible]  (HL) 0
[illegible]  [illegible]              [illegible]  A.A
005663  311                           RET

                           ;+++    switch to normal mode

005666  313 206            MODNOR. CLR     (HL) 0
005670  257                        XOR     A.A
005671  311                        RET

                           ;+++    Switch to hexa

[illegible]  041 112 105   [illegible]  LOAD  HL.[illegible]
[illegible]  076 110                    LOAD  A.#110
[illegible]  [illegible]                [illegible]  A.(HL)
005700  [illegible]                     LOAD  (HL).A
```

```
005701  317 137                   TEST    A.6
        ... ...                   JUMP,.NE EXHE2
        ... ... 015               LOAD    HL.#TOCT
        ... ...          EXHE1    ...     ...
        ... ...                   ...     ...
005714  030 311                   JUMP.   D34

005715  041 331 027      EXHE2:   LOAD    HL.#THEX
005721  030 ...                   JUMP.   EXHE1

                         .duns INCDAY    TOCT:    .ASCIZ  /OCTAL/

                         .cn 3743 THEX:   .ASCIZ  /HEXA/


                         .PAGE

                         .LOADER PAPER ABSOLUTE/RELATIVE

                         .---     PRBYTE  ;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 AND HL

005720  347 024          PRBYTE.  .U      ?PPR
005722  107                       LOAD    B.A             ;SAVE IN D
005723  203                       ADD     A.E             ,ADD TO CHECKSUM
005724  137                       LOAD    E.A
005730  170                       LOAD    A.B             ;PUT DATA IN A AGAIN
005731  311                       RET

                         ,---     HEADER  ---
                         .WAIT FOR HEADER. RETURN WITH ADDRESS IN HL. BLOCK LENGTH IN D,
                         .BYTE COUNT IN E

005732  315 323 013      HEADER:  CALL    PRBYTE
005735  170                       LOAD    A.B
005736  376 000                   COMP    A.#0
005740  050 370                   JUMP,.EQ HEADER
005742  075                       DEC     A               ,NO, ?IS IT A 1 ?
005743  040 365                   JUMP,.NE HEADER         ;MAYBE A RECOVERABLE ERROR

005745  315 323 013      HEA4.    CALL    PRBYTE
005750  267                       OR      A.A             ;?IS IT A 0 ?
005751  040 357                   JUMP,.NE HEADER         ,MAYBE A RECOVERABLE ERROR
005753  036 001                   LOAD    E.#1            ;INIT. CHECKSUM
005755  315 323 013               CALL    PRBYTE          ,GET BYTE COUNT
005760  326 005                   SUB     A.#5
005762  127                       LOAD    D.A             ;LOAD   COUNTER
005763  315 323 013               CALL    PRBYTE
005766  267                       OR      A.A             .?IS IT WELL A 0 ?
005767  040 102                   JUMP,.NE ERROR          ;NO, TRAP
005771  315 323 013               CALL    PRBYTE          ;OK,PUT ADDRESS IN HL
005774  157                       LOAD    L.A
005775  315 323 013               CALL    PRBYTE
006000  147                       LOAD    H.A
006001  025                       DEC     D               .?IS IT THE LAST BLOCK ?
006002  311                       RET

                         ;---     DATA    ---
                         ;GET A BYTE AND SAVE IT AT (HL+)

006003  315 323 013      DATA:    CALL    PRBYTE
006006  167                       LOAD    (HL).A

006007  315 017 014               CALL    ECHO

006012  043                       INC     HL
006013  025                       DEC     D
006014  040 365                   JUMP,.NE DATA
006016  311                       RET

006017  323 002          ECHO:    LOAD    $HP,A           ;Give a toc on the HP
006021  345                       PUSH    HL
006022  174                       LOAD    A.H
006023  052 164 105               LOAD    HL.POINT
006026  167                       LOAD    (HL).A          ;Echo on screen
006027  341                       POP     HL
006030  311                       RET


                         ;;Appels de transfert papier/cassette
                         .;-----------------------------------

006031  252 114 117 101 .ASCII   "<252>LOAD<200>"
006037  257                       XOR     A.A
006040  062 112 105               LOAD    SAVHEX.A                ;For paper
006043  117                       LOAD    C.A
006044  041 000 000               LOAD    HL.#0
006047  345                       PUSH    HL      ;For return
                         ;;
                         ;?????????
                         .;103 ?LOADBI   Charge un binaire PDP11
                         .?????????
                         ;;
                         .;in     C=0 chargeur absolu
                         ;;        C=1-377  HL=adresse debut chargement
                         ;;out    SAVLGO  initialise si auto start
                         .;mod    F A B C D E H L
                         ;;
006050  345              LOADBI:  PUSH    HL              ;Save address
006051  171                       LOAD    A.C
006052  267                       OR      A.A     ;Relative?
006053  040 036                   JUMP,.NE RLOAD
                         ;Absolute loader

006055  341                       POP     HL      ;Correct stack
006056  315 332 013      ALOAD:   CALL    HEADER
006061  050 012                   JUMP,.EQ LAST
006063  315 003 014               CALL    DATA
006066  315 323 013               CALL    PRBYTE
006071  050 263                   JUMP,.EQ ALOAD
006073  067              ERROR:   SETC
006074  311                       RET

006075  315 323 013      LAST:    CALL    PRBYTE
006100  040 371                   JUMP,.NE ERROR
006102  175                       LOAD    A.L
006103  075                       DEC     A
006104  264                       OR      A.H
006105  310                       RET.EQ          ;Carry Clear
                         ;?stack?
006106  042 106 105               LOAD    SAVLGO.HL
006111  343                       EX      (SP).HL
006112  311                       RET

                         ;Relative, address on stack
                         RLOAD                            ,Beginning address on stack
006113  315 332 013               CALL  . HEADER  ,Get absolute address in HL
006116  301                       POP     BC
006117  257                       OR      A.A     ,Clear carry
006120  355 102                   SUBC    HL.BC
006122  345                       PUSH    HL      ;Difference on stack
006123  140                       LOAD    H.B
006124  151                       LOAD    L.C

006125  315 003 014      RLO2:    CALL    DATA
006130  315 323 013               CALL    PRBYTE          .?CHECKSUM GOOD?
006133  040 017                   JUMP,.NE RLO3           ,YES, READ NEXT BLOCK
006135  315 332 013               CALL    HEADER
006140  050 007                   JUMP,.EQ RLO4
006142  301                       POP     BC
006143  305                       PUSH    BC
006144  257                       OR      A.A
006145  355 102                   SUBC    HL.BC
006147  030 334                   JUMP.   RLO2
```

```
006151  315 323 013      RLO4     CALL    PRBYTE  ,Checksum on start address
006154  341              RLO3     POP     HL
006155  040 314                   JUMP,.NE ERROR
006157  267                       OR      A.A     ;Carry <-- 0
006160  311                       RET


                         ,***     PUNCH A TAPE IN PDP11 FORMAT ON A SMAKY8

                         ;---     CALL    PERDE   ;PUNCH WORD IN DE

006161  173              PERDE,   LOAD    A.E
006162  315 166 014               CALL    PERBY
006165  172                       LOAD    A.D

                         ;---     CALL    PERBY   ---
                         .SWITCH TO PAPER OR CASSETTE
                         .COMPUTE CHECKSUM IN C

006166  365              PERBY:   PUSH    AF
006167  201                       ADD     A.C     ,COMPUTE CHECKSUM
006170  117                       LOAD    C.A
006171  361                       POP     AF

                         ;---     CALL    PERA     PUNCH BYTE IN A
                         ,                         ,DO NOT CHANGE THE FLAGS


006172  365              PERA:    PUSH    AF
006173  347 026                   .U      ?UPP
006175  361                       POP     AF
006176  311                       RET

                         ;????????????
                         ;;105 ?AMORCE    ,PUNCH A LEADER OF MAX 250 BYTES
                         ,????????????
                         ;;
                         ,,in     -
                         ,,out    -
                         ,,mod    F.A.B.

                 000100 LONGAM=  100
                 000000 CARAM=   0                        ;LEADER CHARACTER
                 000026 SSYN=    26

006177  006 100          AMORCE:  LOAD    B.#LONGAM
006201  076 000          AMO2     LOAD    A.#CARAM
006203  315 166 014               CALL    PERBY
006206  020 371                   DECJ,NE B.AMO2
006210  311                       RET

                         ,---     CALL    CHECKSUM,COMPUTE AND PUNCH LAST CHECKSUM

006211  171              CHECKSUM LOAD    A.C     ,2'S COMPLEMENT OF C
006212  057                       CPL     A
006213  074                       INC     A
006214  315 166 014               CALL    PERBY
006217  311                       RET


                 000050 LONBLO=  50               ,STANDARD BLOCK LENGTH

                         ;+++     Punch program

006220  375 041 000 000 DOPUNCH;LOAD     IY.#0           ;Offset null
006224  315 235 014               CALL    PUNBI
006227  315 177 014               CALL    AMORCE
006232  303 115 011               JUMP    DEB1

                         ;?????????
                         ;;104 ?PUNBI      Perfore une amorce et le bloc format PDP11
                         ;?????????        ! pas d'amorce de fin
                         ;;
                         ;;in     HL adresse de debut, DE longueur, BC adr de restart
                         ;;                IY correction d'adresse
                         ;;out    -
                         ;;mod    F A B C D E HL IY
                         ;;
006235  345              PUNBI:   PUSH    HL
006236  325                       PUSH    DE      ;Diff
006237  305                       PUSH    BC      ,Restart addr
006240  341                       POP     HL
006241  042 106 105               LOAD    SAVLGO.HL
006244  341                       POP     HL      ,Diff
                         ;Begin address still on stack
006245  315 177 014               CALL    AMORCE

006250  016 000          BLOCK:   LOAD    C.#0    ;INIT CHECKSUM
006252  021 001 000               LOAD    DE.#1
006255  315 161 014               CALL    PERDE   ;PUNCH TRAILER
006260  021 050 000               LOAD    DE.#LONBLO
006263  267                       OR      A.A
006264  355 122                   SUBC    HL.DE
006266  060 005                   JUMP,.CC BLO2
006270  031                       ADD     HL.DE   ,NO GET OLD LENGTH AGAIN
006271  021 000 000               LOAD    DE.#0   ;NO REMAINING LENGTH
006274  353                       EX      DE.HL   .DE IS NOW THE BLOCK LENGTH

006275  345              BLO2.    PUSH    HL      ,SAVE REMAINING LENGTH
006276  103                       LOAD    B.E     ,SAVE DATA BLOCK LENGTH
006277  353                       EX      DE.HL   ;CORRECT BLOCK LENGTH IN DE
006300  021 006 000               LOAD    DE.#6
006303  031                       ADD     HL.DE
006304  353                       EX      DE.HL
006305  315 161 014               CALL    PERDE
006310  005                       DEC     B
006311  004                       INC     B
006312  341                       POP     HL
006313  321                       POP     DE      ,GET BLOCK ADDRESS
006314  050 035                   JUMP,.EQ LASTP

006316  345                       PUSH    HL
006317  325                       PUSH    DE
006320  375 345                   PUSH    IY
006322  341                       POP     HL
006323  353                       EX      HL.DE
006324  267                       OR      A.A
006325  355 122                   SUBC    HL.DE
006327  353                       EX      HL.DE
006330  315 161 014               CALL    PERDE
006333  321                       POP     DE
006334  341                       POP     HL

006335  032              BLO4     LOAD    A.(DE)

006336  023                       INC     DE
006337  315 166 014               CALL    PERBY
006342  005                       DEC     B       ,? LAST BLOCK ?
006343  040 370                   JUMP,.NE BLO4
006345  325                       PUSH    DE
006346  315 211 014               CALL    CHECKSUM
006351  030 275                   JUMP.   BLOCK

006353  052 106 105      LASTP,   LOAD    HL.SAVLGO
006356  353                       EX      DE.HL
006357  315 161 014               CALL    PERDE
006362  030 225                   JUMP.   CHECKSUM


                         ;;
                         ,,
                         ,,Appels pour affichage et lecture d'un nombre sous moniteur
                         ;;**************************************************************
                         ;;
                         ,---     Test flag hexa et cassette
                         ,;out    SC octal, SS hexa, CS cassette, CC papier, PO inverse
                         ,;mod    F
                         ,-
```

```
006364  305              TESTHEX PUSH   BC
006365  107                      LOAD   B,A
006366  072 112 105              LOAD   A,SAVHEX
006371  313 007                  BIT    A
006373  170                      LOAD   A,B
006374  301                      POP    BC
006375  311                      RET

                         ;????????????
                         ;;120 ?IMODE      Set INV flag. hex and cass flag
                         ;????????????
                         ;;
                         ;;in    A=0 Octal and paper
                         ;;      A=110 Hexa and paper
                         ;;      A=220 Octal and cassette
                         ;;      A=300 Hexa and cassette
                         ;;      A=1 Inverted characters
                         ;;out   -
                         ;;mod   -
                         ;;
006376  062 112 105      IMODE:  LOAD   SAVHEX,A
006401  311                      RET

                         ;??????????
                         ;;102 ?AFMOHL   Affichage HL en octal ou hexa
                         ;??????????
                         ;;in    HL, pointeur, SAVHEX, OUTCAR
                         ;;out   pointeur
                         ;;mod   A, pointeur
                         ;;
006402  315 364 014      AFMOHL: CALL   TESTHEX
006405  362 136 010              JUMP,SC AFOHL
006410  347 042                  .W     ?SPACE
006412  347 131                  .W     ?AFXHL
006414  347 042                  .W     ?SPACE
006416  311                      RET

                         ;?????????
                         ;;101 ?AFMONA   Affichage A en octal ou hexa
                         ;?????????
                         ;;
                         ;;in    pointeur, A, SAVHEX, OUTCAR
                         ;;out   pointeur
                         ;;mod   F, pointeur
                         ;;
006417  315 364 014      AFMONA: CALL   TESTHEX
006422  362 154 010              JUMP,SC AFOCAB
006425  347 042                  .W     ?SPACE
006427  303 221 010              JUMP   AFHEX

                         ;????????
                         ;;100 ?INMON    Lecture pour moniteur selon SAVHEX
                         ;????????
                         ;;
                         ;;in    SAVHEX, INCAR
                         ;;out   HL nombre, C nbre digits, EQ si C=0, A der car
                         ;;mod   F, A, B, C, HL
                         ;;
006432  315 364 014      INMON:  CALL   TESTHEX
006435  362 047 015              JUMP,SC INM02
006440  347 066                  .W     ?INHEX
006442  107              INM01:  LOAD   B,A
006443  171                      LOAD   A,C
006444  257                      OR     A,A     ;Test C flag (number of key depressed)
006445  170                      LOAD   A,B
006446  311                      RET

                         INM02:
006447  347 067                  .W     ?INOC
006451  030 367                  JUMP,  INM01


                         ;;
                         ;;Appels pour mesure du temps
                         ;;--------------------------
                         ;;
                         ;??????????
                         ;;111 ?INCSEC   Ajoute 2 centièmes de seconde
                         ;??????????
                         ;;
                         ;;in    HL pointeur, (HL) centièmes et secondes
                         ;;out   HL incr de 2, (HL), CS si report des secondes
                         ;;mod   HL, (HL), A, F
                         ;;

006453  176              INCSEC: LOAD   A,(HL)
006454  306 002                  ADD    A,#2
006456  047                      DA     A
006457  167                      LOAD   (HL),A
006460  043                      INC    HL
006461  176              INC60:  LOAD   A,(HL)
006462  316 000                  ADDC   A,#0
006464  047                      DA     A
006465  376 140                  COMP   A,#6*16.
006467  070 001          INC61:  JUMP,,LO INC62
006471  257                      XOR    A,A
006472  077              INC62.  CFLC
006473  167                      LOAD   (HL),A
006474  043                      INC    HL
006475  311                      RET

                         ;??????????
                         ;;112 ?INCHOUR  Ajoute une minute si CS
                         ;??????????
                         ;;
                         ;;in    CS, HL pointeur, (HL) minutes et heures
                         ;;out   HL incr de 2, (HL), CS si report des heures
                         ;;mod   HL, (HL), A, F
                         ;;

006476  315 261 015      INCHOUR.CALL   INC60
006501  176                      LOAD   A,(HL)
006502  316 000                  ADDC   A,#0
006504  047                      DA     A
006505  376 044                  COMP   A,#2*16.+4
006507  030 356                  JUMP,  INC61

                         ;??????????
                         ;;113 ?INCDAY   Ajoute un jour si CS
                         ;;               Pour les ann biss, tester si 28 fev et forcer en
                         ;??????????
                         ;;
                         ;;in    CS, HL pointeur, (HL) jour et mois
                         ;;out   HL incr de 2, (HL), CS si report des mois
                         ;;mod   HL, (HL), A, F
                         ;;

006511  176              INCDAY: LOAD   A,(HL)
006512  316 000                  ADDC   A,#0
006514  047                      DA     A
                                 ;Comparaison avec la table des jours/mois:
006515  305                      PUSH   BC
006516  345                      PUSH   HL
006517  043                      INC    HL
006520  116                      LOAD   C,(HL)
006521  006 000                  LOAD   B,#0
006523  041 150 015              LOAD   HL,#JRMOIS
006526  011                      ADD    HL,BC
006527  276                      COMP   A,(HL)
006530  341                      POP    HL
006531  301                      POP    BC
006532  315 143 015              CALL   INCD2
                                 ;Increment des mois:
006535  176                      LOAD   A,(HL)
006536  316 000                  ADDC   A,#0
006540  047                      DA     A
006541  376 023                  COMP   A,#1*16.+3
006543  070 335          INCD2   JUMP,,LO INC62
006545  076 001                  LOAD   A,#1
006547  030 321                  JUMP,  INC62
```

```
                                 ;Table du nombre de jours/mois
        006550                   .RADIX  8.,+1
        000000                   .WORD   10.
006551  062 061 062 061          .BYTE   22,20,22,31,22,31,22,22,31
                         ;Recuperation des 2 bytes libres
006562  117 103 124 101 TOCT     .ASCII  "OCTA"
006570  062 061 062              .BYTE   22,31,12
        000010                   .RDX    8.

                         ;??????????
                         ;;114 ?INCYEAR  Ajoute une année si CS ( ADDC (HL) )
                         ;??????????
                         ;;
                         ;;in    HL pointeur, (HL) année (2 digits), CS ajoute 1
                         ;;out   HL incr de 1 si CS, (HL), CS  si report
                         ;;mod   HL, (HL), A, F
                         ;;
006573  176              INCYEAR:LOAD   A,(HL)
006574  316 000                  ADDC   A,#0
006576  047                      DA     A
006577  167                      LOAD   (HL),A
006600  043                      INC    HL
006601  311                      RET


                         ;???????????
                         ;;113 ?AFTIME   Affiche l'heure
                         ;???????????
                         ;;
                         ;;in    pointeur, HL pointeur aux poids forts
                         ;;      A format ex.00010000 pour 03 12 18 54 (12 mars a 18
                         ;;      3 octets avant   1 octet apres
                         ;;out   pointeur
                         ;;mod   pointeur, HL
                         ;;
006602  365              AFTIME  PUSH   AF
006603  305                      PUSH   BC
006604  107                      LOAD   B,A
006605  313 040          AFTI2:  SLC    B
006607  070 021                  JUMP,,CS AFTI5
006611  076 072                  LOAD   A,#':
006613  347 000                  .W     ?OUTCAR
006615  176              AFTI4:  LOAD   A,(HL)
006616  053                      DEC    HL
006617  347 071                  .W     ?AFHEX
006621  347 042                  .W     ?SPACE
006623  313 040                  SLC    B
006625  070 356                  JUMP,,CS AFTI4
006627  301                      POP    BC
006630  361                      POP    AF
006631  311                      RET

006632  176              AFTI5:  LOAD   A,(HL)
006633  053                      DEC    HL
006634  347 042                  .W     ?SPACE
006636  347 071                  .W     ?AFHEX
006640  030 343                  JUMP,  AFTI2

                         ;;??????????
                         ;;124 ?BINBCD   Converti 16 bits de binaire en BCD
                         ;??????????
                         ;;
                         ;;in    HL nombre binaire
                         ;;out   AHL nombre BCD equivlent
                         ;;mod   F A HL
                         ;;
006642  305              BINBCD: PUSH   BC
006643  325                      PUSH   DE
006644  353                      EX     HL,DE
006645  041 272 105              LOAD   HL,#SIMUL1
006650  006 004                  LOAD   B,#4
006652  345              BINB2:  PUSH   HL
006653  305                      PUSH   BC
006654  001 012 000              LOAD   BC,#10.  ;Diviseur
006657  041 000 000              LOAD   HL,#0    ;Poids forts
006662  347 060                  .W     ?DIV
006664  175                      LOAD   A,L
006665  301                      POP    BC
006666  341                      POP    HL
006667  355 147                  RRD    A,(HL)
006671  313 100                  TEST   B:0      ;?pair
006673  050 001                  JUMP,,EQ BINB4
006675  043                      INC    HL
006676  020 352          BINB4:  DECJ,NE B,BINB2
006700  052 272 105              LOAD   HL,SIMUL1
006703  173                      LOAD   A,E
006704  321                      POP    DE
006705  301                      POP    BC
006706  311                      RET

                         ;??????????
                         ;;125 ?BCDBIN   Converti 4 digits de BCD en binaire
                         ;??????????
                         ;;
                         ;;in    HL nombre BCD
                         ;;out   HL nombre binaire
                         ;;mod   HL
                         ;;
006707  365              BCDBIN: PUSH   AF
006710  305                      PUSH   BC
006711  325                      PUSH   DE
006712  042 272 105              LOAD   SIMUL1,HL
006715  041 273 105              LOAD   HL,#SIMUL1+1
006720  021 000 000              LOAD   DE,#0
006723  006 004                  LOAD   B,#4
006725  257              BCDB2:  XOR    A,A
006726  355 157                  RLD    A,(HL)
006730  345                      PUSH   HL
006731  305                      PUSH   BC
006732  001 012 000              LOAD   BC,#10.  ;Multiplicateur
006735  140                      LOAD   H,B      ;H=0
006736  157                      LOAD   L,A
006737  347 057                  .W     ?MUL
006741  301                      POP    BC
006742  341                      POP    HL
006743  313 100                  TEST   B:0      ;?pair
006745  050 001                  JUMP,,EQ BCDB4
006747  053                      DEC    HL
006750  020 353          BCDB4:  DECJ,NE B,BCDB2
006752  353                      EX     HL,DE
006753  321                      POP    DE
006754  301              BCDB3:  POP    BC
006755  361                      POP    AF
006756  311                      RET


                         ;?????????
                         ;;127 ?DELAY    Attente de .. millisecondes
                         ;;133 ?HLDELAY
                         ;?????????
                         ;;
                         ;;in    HL durée
                         ;;out   -
                         ;;mod   -

                         HLDELAY:
006757  042 272 105              LOAD   SIMUL1,HL
006762  030 007                  JUMP,  DELA0

006764  042 272 105      DELAY:  LOAD   SIMUL1,HL
006767  353                      EX     HL,DE
006770  347 117                  .W     ?GETARG
006772  353                      EX     HL,DE
                         DELA0:
006773  365                      PUSH   AF
006774  305                      PUSH   BC
006775  006 057          DELA1   LOAD   B,#F0000./52.
006777  020 376                  DECJ,NE B,.
007001  053                      DEC    HL
```

```
007002  175                      LOAD    A.L
007003  264                      CP      A.H
007004  040 357                  JUMP.NE DEL42
007006  052 272 105              LOAD    HL.SIMUL1          ;JUMP TEXT2 econom 1 byte
007011  030 341                  JUMP    BCDIG


007013  123 131 123 040 TERESET .ASCIZ   "SYS 1-<VAR> X "
007023  040 113 015 000         .ASCIZ   " K<CR>"

                        ;???????
                        ;;42 ?SPACE
                        ;;43 ?RETURN              Execute un CR
                        ;;121 ?TAB
                        ;????????
                        ;;
                        ;;in     pointeur, OUTCAR
                        ;;out    pointeur
                        ;;mod    pointeur
                        ;;
007032  365             SPACE:   PUSH    AF
007033  076 040                  LOAD    A.#SPA
                        SPA2:
007035  347 000                  .W      ?DICAR
007037  361             SPA4:    POP     AF
007040  311                      RET

007041  365             RETURN:  PUSH    AF
007042  076 015                  LOAD    A.#CR
007044  030 367                  JUMP.   SPA2

007046  365             TABULAT: PUSH    AF
007047  076 011                  LOAD    A.#TAB
007051  030 362                  JUMP    SPA2

                        ;???????
                        ;;122 ?CLEAR
                        ;;130 ?DELETE
                        ;???????
                        ;;
                        ;;in     -
                        ;;out    -
                        ;;mod    -
                        ;;
007053  365             DELETE:  PUSH    AF
007054  076 010                  LOAD    A.#BS
007056  030 355                  JUMP    SPA2

007060  365             CLEAR:   PUSH    AF
007061  076 022                  LOAD    A.#CLEARW
007063  030 350                  JUMP    SPA2

                        ;??????????
                        ;;137 ?BACKSPACE
                        ;??????????
                        ;;
                        ;;in     -
                        ;;out    A car sous le pointeur deplace
                        ;;       EQ si debut de ligne
                        ;;
007065  345             BACKS:   PUSH    HL
007066  325                      PUSH    DE
007067  052 164 105              LOAD    HL.POINT
007072  053                      DEC     HL
007073  021 000 100              LOAD    DE.#SALFHA
007076  347 056                  .W      ?COMPHLDE
007100  060 001                  JUMP..HS BAC2
007102  043                      INC     HL
007103  175             BAC2:    LOAD    A.L
007104  346 077                  AND     A.#77
007106  176                      LOAD    A.(HL)
007107  042 164 105              LOAD    POINT.HL
007112  321                      POP     DE
007113  341                      POP     HL
007114  311                      RET


                        ;??????????
                        ;;56 ?COMPHLDE  Compare HL et DE
                        ;??????????
                        ;;
                        ;;in     HL, DE
                        ;;out    EQ si HL=DE, LO si HL<DE
                        ;;mod    F
                        ;;
007115  345             COMPHLDE:PUSH    HL
007116  267                      OR      A.A
007117  355 122                  SUBC    HL.DE
007121  341                      POP     HL
007122  311                      RET


                        ;==========================================================

                                 ;E 4050-8 micro clock drivers

                                 ; revision       1-1
                                 ; Ronald Forster november 1979

                        ; These are the minimal drivers for the clock module E4050
                        ; produced by Marin electronics.
                        ; They are designed to work with the interface of the 32K ram
                        ; extension board in the revision 1-1.

                        ; These minimal drivers provide the possibility to read all
                        ; the informations contained in chip, that means hour, minutes
                        ; date, month year, day of the week, and seconds, and dump
                        ; them in that order into a seven bytes buffer provided by the
                        ; user. The possibility of writing information to the chip is
                        ; accomplished by writing into the chip the seven bytes buffer
                        ; provided by the user in the same format as before.

                        ; The informations are coded in packed decimal (BCD).


                                 ; peripherals addresses
                                 ; ---------------------

                000010 CLOCK     =       10              ; I/O port
                000000  BIO      =       0               ; I/O bit in port
                000001  BGATE    =       1               ; I/O driver gate
                000002  BCS      =       2               ; chip select bit
                000003  BCK      =       3               ; clock pulse bit


                                 ; clock modes
                                 ; -----------

                000010 READ      =       10              ; read data from chip
                000000 WRITE     =       0               ; write data to chip

                000007 CONT      =       7               ; continuous I/O

                000000 OHOUR     =       0               ; offset for hours
                000001 OMIN      =       OHOUR+1         ;    "   for minutes
                000002 ODATE     =       OMIN+1          ;    "   for date
                000003 OMONTH    =       ODATE+1         ;    "   for month
                000004 OYEAR     =       OMONTH+1        ;    "   for year
                000005 ODAY      =       OYEAR+1         ;    "   for day of the week
                000006 OSEC      =       ODAY+1          ;    "   for seconds

                000005 SOJOUR    =       5
                000002 SODAY     =       2
                000001 SOYEAR    =       1
                000000 SOHOUR    =       0
                000005 SOSEC     =       5
```

```
                042400 CLKBUF    =       42400

                        ;;
                        ;;
                        ;
                        ;------\
                        ; RDCLK >
                        ;======/
                        ;;140 ?RDCLK    Lecture de toutes les informations du chip ho

                        ;;
                        ;;in     DE ^to a 7 bytes buffer
                        ;;out    -
                        ;;mod    -
                        ;;
                        RDCLK:
007123  325                      PUSH    DE
007124  305                      PUSH    BC                 ; save all what we use
007125  365                      PUSH    AF
007126  353                      EX      HL.DE
007127  257                      XOR     A.A
007130  323 010                  LOAD    $CLOCK.A

007132  006 004                  LOAD    B.#4               ; and send a continuous read
007134  016 017                  LOAD    C.#CONT+READ       ; command
007136  313 327                  SET     A:BCS
007140  313 317                  SET     A:BGATE            ; chip select and gate are set
                        RDC1:                               ; to true
007142  037                      RRC     A
007143  313 031                  RRC     C                  ; then data is clocked one bi
007145  027                      RLC     A                  ; at a time into port
007146  323 010                  LOAD    $CLOCK.A
007150  343                      EX      (SP).HL
007151  343                      EX      (SP).HL
007152  313 337                  SET     A:BCK
007154  323 010                  LOAD    $CLOCK.A           ; the the cp is set to one
007156  343                      EX      (SP).HL
007157  343                      EX      (SP).HL
007160  313 237                  CLR     A:BCK              ; and the zeroed
007162  020 356                  DECJ.NE B.RDC1             ; we loop for the four bits
007164  313 217                  CLR     A:BGATE            ; and finally clear gate
007166  016 007                  LOAD    C.#7               ; we now will read 7 bytes
                        RDC2:
007170  006 010                  LOAD    B.#8.              ; composed of 8 bits
                        RDC3:
007172  323 010                  LOAD    $CLOCK.A           ; at negative going edge of
007174  343                      EX      (SP).HL
007175  343                      EX      (SP).HL
007176  365                      PUSH    AF                 ; 14us later
007177  333 010                  LOAD    A.$CLOCK
007201  037                      RRC     A                  ; we read the data port and
007202  313 036                  RRC     (HL)               ; move one bit to the user's
007204  361                      POP     AF                 ; buffer
007205  313 337                  SET     A:BCK              ; now we rise the cp and wait
007207  323 010                  LOAD    $CLOCK.A           ; again a little time
007211  343                      EX      (SP).HL
007212  343                      EX      (SP).HL
007213  313 237                  CLR     A:BCK
007215  020 353                  DECJ.NE B.RDC3             ; have we done all bits ?
007217  043                      INC     HL
007220  015                      DEC     C                  ; have we done all bytes ?
007221  040 345                  JUMP.NE RDC2
                        EXIT:
007223  257                      XOR     A.A
007224  323 010                  LOAD    $CLOCK.A           ; chip select
007226  353                      EX      HL.DE
007227  361                      POP     AF                 ; and restore all registers
007230  301                      POP     BC
007231  321                      POP     DE
007232  311                      RET


                        ;;
                        ;------\
                        ; WRCLK >
                        ;======/
                        ;;141 ?WRCLK    Initialisation du chip horloge
                        ;;
                        ;;in     DE ^to a 7 bytes buffer to be written
                        ;;out    -
                        ;;mod    buffer
                        ;;
                        WRCLK:
007233  325                      PUSH    DE
007234  305                      PUSH    BC                 ; we save what we use
007235  365                      PUSH    AF
007236  353                      EX      HL.DE
007237  257                      XOR     A.A

007240  323 010                  LOAD    $CLOCK.A           ; be sure to reset the chip

007242  006 004                  LOAD    B.#4               ; we now send a 4 bits command
007244  016 007                  LOAD    C.#CONT+WRITE      ; continuous writing
007246  313 327                  SET     A:BCS
007250  313 317                  SET     A:BGATE            ; we set the chip select and
                        WRC1:                               ; open the gate
007252  037                      RRC     A
007253  313 031                  RRC     C                  ; the data is then clocked one
007255  027                      RLC     A                  ; bit at a time into perich.
007256  323 010                  LOAD    $CLOCK.A
007260  343                      EX      (SP).HL
007261  343                      EX      (SP).HL
007262  313 337                  SET     A:BCK              ; have to wait a little time
007264  323 010                  LOAD    $CLOCK.A           ; the positive going edge of
007266  343                      EX      (SP).HL
007267  343                      EX      (SP).HL
007270  313 237                  CLR     A:BCK              ; and here we lower cp
007272  020 356                  DECJ.NE B.WRC1             ; we loop the four bits

007274  016 007                  LOAD    C.#7               ; now we'll write 7 bytes
                        WRC2:
007276  006 010                  LOAD    B.#8.              ; composed of 8 bits
                        WRC3:
007300  037                      RRC     A
007301  313 026                  RLC     (HL)               ; we get one bit at a time
007303  027                      RLC     A                  ; from the user's buffer
007304  323 010                  LOAD    $CLOCK.A           ; and store it into perich.
007306  343                      EX      (SP).HL
007307  343                      EX      (SP).HL
007310  313 337                  SET     A:BCK              ; the positive going edge of
007312  323 010                  LOAD    $CLOCK.A           ; cp, latches the data
007314  343                      EX      (SP).HL
007315  343                      EX      (SP).HL
007316  313 237                  CLR     A:BCK              ; to wait 14us
007320  020 356                  DECJ.NE B.WRC3             ; have we done all the bits ?
007322  043                      INC     HL
007323  015                      DEC     C                  ; have we done all the bytes ?
007324  040 350                  JUMP.NE WRC2
007326  030 273                  JUMP    EXIT               ; we can exit
                        ;;
                        ;;142 ?SJOUR    Initialisation du jour de la semaine
                        ;;
                        ;;in     A =1 lundi  ...  =7 dimanche
                        ;;out    -
                        ;;mod    F. DE. HL
                        ;;
007330  041 005 000     SJOUR    LOAD    HL.#SOJOUR
007333  267                      OR      A.A
007334  030 017                  JUMP.   SETCL1
                        ;;
                        ;;143 ?SDAY     Initialisation de la date seulement
                        ;;in     A=dd,B=mm, C=yy
                        ;;out    -
                        ;;mod    F.DE.HL
                        ;;
007336  041 002 000     SDAY:    LOAD    HL.#SODAY
007341  021 001 000              LOAD    DE.#SOYEAR
007344  030 006                  JUMP.   SETCLK
                        ;;
                        ;;144   ?STOD   Initialisation de l'heure seulement
                        ;;in     A=hh, B=mm, C=ss
                        ;;out    -
```

```
                                ;;mod   F.DE.HL

007346  041 000 000             STIME:  LOAD    HL,#STHOUR
007351  021 005 000                     LOAD    DE,#SOSEC

007354  067                     SETCLK: SETC
007355  315 364 005             SETCL1: CALL    DRDCLK
007360  167                             LOAD    (HL),A
007361  050 004                         JUMP,.CC SETCL2
007363  043                             INC     HL
007364  160                             LOAD    (HL),B
007365  031                             ADD     HL,DE
007366  161                             LOAD    (HL),C
007367  021 000 103             SETCL2: LOAD    DE,#CLKBUF
007372  347 141                         .W      ?WRCLK
007374  311                             RET

                                ;;
                                ;;145    ?GJOUR  Lecture du jour de la semaine seulement
                                ;;in     -
                                ;;out    A=wd, jour de la semaine (1=lundi, 7=dimanche,etc.)
                                ;;mod    A,F,HL
                                ;;
007375  041 005 000             GJOUR:  LOAD    HL,#SOJOUR
007400  267                             OR      A,A
007401  030 017                         JUMP,   GETCL1

                                ;;
                                ;;146    ?GDAY   Lecture de la date seulement
                                ;;in     -
                                ;;out    A=dd, B=mm, C=yy
                                ;;mod    A,F,BC,DE,HL
                                ;;
007403  041 002 000             GDAY:   LOAD    HL,#SODAY
007406  021 001 000                     LOAD    DE,#SOYEAR
007411  030 006                         JUMP,   GETCLK

                                ;;
                                ;;?147   ?GTOD   Lecture de l'heure seulement
                                ;;in     -
                                ;;out    A=hh, B=mm, C=ss
                                ;;mod    A,F,BC,DE,HL
                                ;;
007413  041 003 000             GTIME:  LOAD    HL,#SOHOUR
007416  021 003 000                     LOAD    DE,#SOSEC

007421  067                     GETCLK: SETC
007422  315 364 005             GETCL1: CALL    DRDCLK
007425  176                             LOAD    A,(HL)
007426  330                             RET,CC
007427  043                             INC     HL
007430  106                             LOAD    B,(HL)
007431  031                             ADD     HL,DE
007432  116                             LOAD    C,(HL)
007433  311                             RET


                                ;-------------


                                ;TOUTE LA TABLE DOIT ETRE DANS LA MEME PAGE.
                                ;ATTENTION, ORDRE !!!

007460                          .LOC    7460

007460  044 004                 TABOUT: .W      DICAR           ;0
007462  276 004                         .W      GETCAR          ;1
007464  206 002                         .W      ITIMER          ;2
007466  212 002                         .W      ITIMAD          ;3
007470  216 002                         .W      IRTC            ;4
007472  033 005                         .W      GETLINE         ;5
007474  205 004                         .W      DITEX           ;6
007476  225 002                         .W      IRST10          ;7
007500  232 002                         .W      IRST20          ;10
007502  236 002                         .W      IRST70          ;11
007504  242 002                         .W      IRST60          ;12
007506  246 002                         .W      IRST60          ;13
007510  252 002                         .W      IADNMI          ;14
007512  011 005                         .W      IFCAR           ;15
007514  026 005                         .W      GETFON          ;16
007516  256 002                         .W      ENIS0           ;17
007520  376 002                         .W      IDICAR          ;20
007522  335 002                         .W      IALPHA          ;21
007524  346 002                         .W      IGRA            ;22
007526  356 002                         .W      IAGRA           ;23
007530  162 005                         .W      RPR             ;24
007532  167 005                         .W      RMOD            ;25
007534  240 005                         .W      WPP             ;26
007536  245 005                         .W      WMOD            ;27
007540  300 002                         .W      CALPHA          ;30
007542  317 002                         .W      CGRA            ;31
007544  135 003                         .W      BUZZ            ;32
007546  202 002                         .W      MEM             ;33
007550  136 002                         .W      BRANCH          ;34
007552  050 003                         .W      GETLBCDE        ;35
007554  211 004                         .W      DITXB           ;36
007556  222 002                         .W      IUSINT          ;37
007560  076 003                         .W      SETCURSOR       ;40
007562  337 005                         .W      GETCURSOR       ;41
007564  032 016                         .W      SPACE           ;42
007566  041 016                         .W      RETURN          ;43
007570  212 003                         .W      IFRPR           ;44
007572  217 005                         .W      IFRMOD          ;45
007574  275 005                         .W      IFWPP           ;46
007576  302 005                         .W      IFWMOD          ;47
007600  174 005                         .W      RPAR            ;50
007602  224 005                         .W      IFRPAR          ;51
007604  252 005                         .W      WPAR            ;52
007606  307 005                         .W      IFWPAR          ;53
007610  072 003                         .W      IPOINTEUR       ;54
007612  324 005                         .W      JUMPIDE         ;55
007614  115 010                         .W      COMPHLDE        ;56
007616  007 002                         .W      MUL             ;57
007620  033 002                         .W      DIV             ;60
007622  123 002                         .W      JUMPCAR         ;61
007624  160 002                         .W      SCAR            ;62
007626  326 005                         .W      CALLIDE         ;63
007630  075 010                         .W      ALGRA           ;64
007632  255 010                         .W      INDEC           ;65
007634  303 010                         .W      INHEX           ;66
007636  276 010                         .W      INOC            ;67
007640  201 010                         .W      AFBIN           ;70
007642  221 010                         .W      AFHEX           ;71
007644  155 010                         .W      AFOCA           ;72
007646  136 010                         .W      AFOHL           ;73
007650  253 004                         .W      EFLI            ;74
007652  262 003                         .W      PLAY            ;75
007654  144 003                         .W      BEEP            ;76
007656  112 010                         .W      GRAAL           ;77
007660  000 015                         .W      INMON           ;100
007662  017 015                         .W      AFMONA          ;101
007664  012 015                         .W      AFMOHL          ;102

007666  050 014                         .W      LOADBI          ;103
007670  235 014                         .W      RUNBI           ;104
007672  177 014                         .W      AMOPCE          ;105

007674  210 012                         .W      SAVESTAT        ;106
007676  253 012                         .W      RESTAT          ;107
007700  325 011                         .W      EXECUTE         ;110

007702  053 015                         .W      INCSEC          ;111
007704  070 015                         .W      INCHOUR         ;112
007706  111 015                         .W      INCDAY          ;113
007710  170 015                         .W      INCMOIS         ;114
007712  200 015                         .W      AFTIME          ;115
007714  044 010                         .W      COMPTIME        ;116
007716  000 002                         .W      GETARG          ;117
007720  376 014                         .W      [illegible]     ;120
007722  043 016                         .W      [illegible]     ;121
007724  053 016                         .W      [illegible]     ;122
007726  217 004                         .W      [illegible]     ;123

007730  242 015                         .W      BYTEOD          ;124
007732  107 015                         .W      LODBIN          ;125

007734  032 003                         .W      IDIS
007736  364 015                         .W      DELAY
007740  033 016                         .W      DELETE          ;130
007742  256 010                         .W      AFXHL           ;131

007744  212 003                         .W      EFEC            ;132
007746  357 015                         .W      HLDELAY         ;133
007750  101 011                         .W      MON             ;134
007752  162 012                         .W      TRAPPE          ;135
007754  234 004                         .W      TEXTIM          ;136
007756  065 016                         .W      BACKS           ;137

007760  123 016                         .W      RDCLK           ;140
007762  233 016                         .W      WRCLK           ;141
007764  330 016                         .W      SJOUR           ;142
007766  336 016                         .W      SDAY            ;143
007770  345 016                         .W      STIME           ;144
007772  375 016                         .W      GJOUR           ;145
007774  003 017                         .W      GDAY            ;146
007776  013 017                         .W      GTIME           ;147

                000001          .END

002060 references
Source file 002726 usefull lines long
Binary file 043000 bytes long
Assembly time: 0213 seconds      0767 lines/min
```